```
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright (c) 2016 - 2017 , NXP
 * 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 copyright holder 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" AN
 * 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 FO
 * 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 "fsl_clock.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/* Macro definition remap workaround. */
#if (defined(MCG_C2_EREFS_MASK) && !(defined(MCG_C2_EREFS0_MASK)))
#define MCG_C2_EREFS0_MASK MCG_C2_EREFS_MASK
#endif
#if (defined(MCG_C2_HGO_MASK) && !(defined(MCG_C2_HGO0_MASK)))
#define MCG_C2_HGO0_MASK MCG_C2_HGO_MASK
#endif
#if (defined(MCG_C2_RANGE_MASK) && !(defined(MCG_C2_RANGE0_MASK)))
#define MCG_C2_RANGE0_MASK MCG_C2_RANGE_MASK
#endif
#if (defined(MCG_C6_CME_MASK) && !(defined(MCG_C6_CME0_MASK)))
#define MCG_C6_CME0_MASK MCG_C6_CME_MASK
#endif
```

```
/* PLL fixed multiplier when there is not PRDIV and VDIV. */
#define PLL_FIXED_MULT (375U)
/* Max frequency of the reference clock used for internal clock trim. */
#define TRIM_REF_CLK_MIN (8000000U)
/* Min frequency of the reference clock used for internal clock trim. */
#define TRIM_REF_CLK_MAX (16000000U)
/* Max trim value of fast internal reference clock. */
#define TRIM_FIRC_MAX (5000000U)
/* Min trim value of fast internal reference clock. */
#define TRIM_FIRC_MIN (3000000U)
/* Max trim value of fast internal reference clock. */
#define TRIM_SIRC_MAX (39063U)
/* Min trim value of fast internal reference clock. */
#define TRIM_SIRC_MIN (31250U)

#define MCG_S_IRCST_VAL ((MCG->S & MCG_S_IRCST_MASK) >> MCG_S_IRCST_SHIFT)
#define MCG_S_CLKST_VAL ((MCG->S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT)
#define MCG_S_IREFST_VAL ((MCG->S & MCG_S_IREFST_MASK) >> MCG_S_IREFST_SHIFT)
#define MCG_S_PLLST_VAL ((MCG->S & MCG_S_PLLST_MASK) >> MCG_S_PLLST_SHIFT)
#define MCG_C1_FRDIV_VAL ((MCG->C1 & MCG_C1_FRDIV_MASK) >> MCG_C1_FRDIV_SHIFT)
#define MCG_C2_LP_VAL ((MCG->C2 & MCG_C2_LP_MASK) >> MCG_C2_LP_SHIFT)
#define MCG_C2_RANGE_VAL ((MCG->C2 & MCG_C2_RANGE_MASK) >> MCG_C2_RANGE_SHIFT)
#define MCG_SC_FCRDIV_VAL ((MCG->SC & MCG_SC_FCRDIV_MASK) >> MCG_SC_FCRDIV_SHIF
#define MCG_S2_PLLCST_VAL ((MCG->S2 & MCG_S2_PLLCST_MASK) >> MCG_S2_PLLCST_SHIFT
#define MCG_C7_OSCSEL_VAL ((MCG->C7 & MCG_C7_OSCSEL_MASK) >> MCG_C7_OSCSEL_SHI
#define MCG_C4_DMX32_VAL ((MCG->C4 & MCG_C4_DMX32_MASK) >> MCG_C4_DMX32_SHIFT)
#define MCG_C4_DRST_DRS_VAL ((MCG->C4 & MCG_C4_DRST_DRS_MASK) >> MCG_C4_DRST_D
#define MCG_C7_PLL32KREFSEL_VAL ((MCG->C7 & MCG_C7_PLL32KREFSEL_MASK) >> MCG_C7_
#define MCG_C5_PLLREFSEL0_VAL ((MCG->C5 & MCG_C5_PLLREFSEL0_MASK) >> MCG_C5_PLLR
#define MCG_C11_PLLREFSEL1_VAL ((MCG->C11 & MCG_C11_PLLREFSEL1_MASK) >> MCG_C11_
#define MCG_C11_PRDIV1_VAL ((MCG->C11 & MCG_C11_PRDIV1_MASK) >> MCG_C11_PRDIV1_SH
#define MCG_C12_VDIV1_VAL ((MCG->C12 & MCG_C12_VDIV1_MASK) >> MCG_C12_VDIV1_SHIFT)
#define MCG_C5_PRDIV0_VAL ((MCG->C5 & MCG_C5_PRDIV0_MASK) >> MCG_C5_PRDIV0_SHIFT)
#define MCG_C6_VDIV0_VAL ((MCG->C6 & MCG_C6_VDIV0_MASK) >> MCG_C6_VDIV0_SHIFT)

#define OSC_MODE_MASK (MCG_C2_EREFS0_MASK | MCG_C2_HGO0_MASK | MCG_C2_RANGE0_

#define SIM_CLKDIV1_OUTDIV1_VAL ((SIM->CLKDIV1 & SIM_CLKDIV1_OUTDIV1_MASK) >> SIM_CL
#define SIM_CLKDIV1_OUTDIV4_VAL ((SIM->CLKDIV1 & SIM_CLKDIV1_OUTDIV4_MASK) >> SIM_CL
#define SIM_SOPT1_OSC32KSEL_VAL ((SIM->SOPT1 & SIM_SOPT1_OSC32KSEL_MASK) >> SIM_SO
#define SIM_SOPT2_PLLFLLSEL_VAL ((SIM->SOPT2 & SIM_SOPT2_PLLFLLSEL_MASK) >> SIM_SOP

/* MCG_S_CLKST definition. */
enum _mcg_clkout_stat
{
    kMCG_ClkOutStatFll, /* FLL.            */
    kMCG_ClkOutStatInt, /* Internal clock. */
    kMCG_ClkOutStatExt, /* External clock. */
    kMCG_ClkOutStatPll  /* PLL.            */
};

/* MCG_S_PLLST definition. */
enum _mcg_pllst
```

```
{
    kMCG_PllstFll, /* FLL is used. */
    kMCG_PllstPll  /* PLL is used. */
};

/*******************************************************************************
 * Variables
 ******************************************************************************/

/* Slow internal reference clock frequency. */
static uint32_t s_slowIrcFreq = 32768U;
/* Fast internal reference clock frequency. */
static uint32_t s_fastIrcFreq = 4000000U;

/* External XTAL0 (OSC0) clock frequency. */
uint32_t g_xtal0Freq;
/* External XTAL32K clock frequency. */
uint32_t g_xtal32Freq;

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

/*!
 * @brief Get the MCG external reference clock frequency.
 *
 * Get the current MCG external reference clock frequency in Hz. It is
 * the frequency select by MCG_C7[OSCSEL]. This is an internal function.
 *
 * @return MCG external reference clock frequency in Hz.
 */
static uint32_t CLOCK_GetMcgExtClkFreq(void);

/*!
 * @brief Get the MCG FLL external reference clock frequency.
 *
 * Get the current MCG FLL external reference clock frequency in Hz. It is
 * the frequency after by MCG_C1[FRDIV]. This is an internal function.
 *
 * @return MCG FLL external reference clock frequency in Hz.
 */
static uint32_t CLOCK_GetFllExtRefClkFreq(void);

/*!
 * @brief Get the MCG FLL reference clock frequency.
 *
 * Get the current MCG FLL reference clock frequency in Hz. It is
 * the frequency select by MCG_C1[IREFS]. This is an internal function.
 *
 * @return MCG FLL reference clock frequency in Hz.
 */
static uint32_t CLOCK_GetFllRefClkFreq(void);
```

```
/*!
 * @brief Get the frequency of clock selected by MCG_C2[IRCS].
 *
 * This clock's two output:
 *  1. MCGOUTCLK when MCG_S[CLKST]=0.
 *  2. MCGIRCLK when MCG_C1[IRCLKEN]=1.
 *
 * @return The frequency in Hz.
 */
static uint32_t CLOCK_GetInternalRefClkSelectFreq(void);

/*!
 * @brief Get the MCG PLL/PLL0 reference clock frequency.
 *
 * Get the current MCG PLL/PLL0 reference clock frequency in Hz.
 * This is an internal function.
 *
 * @return MCG PLL/PLL0 reference clock frequency in Hz.
 */
static uint32_t CLOCK_GetPll0RefFreq(void);

/*!
 * @brief Calculate the RANGE value base on crystal frequency.
 *
 * To setup external crystal oscillator, must set the register bits RANGE
 * base on the crystal frequency. This function returns the RANGE base on the
 * input frequency. This is an internal function.
 *
 * @param freq Crystal frequency in Hz.
 * @return The RANGE value.
 */
static uint8_t CLOCK_GetOscRangeFromFreq(uint32_t freq);

/*******************************************************************************
 * Code
 ******************************************************************************/

#ifndef MCG_USER_CONFIG_FLL_STABLE_DELAY_EN
/*!
 * @brief Delay function to wait FLL stable.
 *
 * Delay function to wait FLL stable in FEI mode or FEE mode, should wait at least
 * 1ms. Every time changes FLL setting, should wait this time for FLL stable.
 */
void CLOCK_FllStableDelay(void)
{
    /*
       Should wait at least 1ms. Because in these modes, the core clock is 100MHz
       at most, so this function could obtain the 1ms delay.
     */
    volatile uint32_t i = 30000U;
    while (i--)
    {
        __NOP();
    }
}
#else  /* With MCG_USER_CONFIG_FLL_STABLE_DELAY_EN defined. */
/* Once user defines the MCG_USER_CONFIG_FLL_STABLE_DELAY_EN to use their own delay functio
 * create his own CLOCK_FllStableDelay() function in application code. Since the clock functions in this
 * file would call the CLOCK_FllStableDelay() regardness how it is defined.
 */
extern void CLOCK_FllStableDelay(void);
#endif /* MCG_USER_CONFIG_FLL_STABLE_DELAY_EN */

static uint32_t CLOCK_GetMcgExtClkFreq(void)
{
    /* Please call CLOCK_SetXtal0Freq base on board setting before using OSC0 clock. */
    assert(g_xtal0Freq);
    return g_xtal0Freq;
}

static uint32_t CLOCK_GetFllExtRefClkFreq(void)
{
    /* FllExtRef = McgExtRef / FllExtRefDiv */
    uint8_t frdiv;
    uint8_t range;

    uint32_t freq = CLOCK_GetMcgExtClkFreq();

    if (!freq)
    {
        return freq;
    }

    frdiv = MCG_C1_FRDIV_VAL;
    freq >>= frdiv;

    range = MCG_C2_RANGE_VAL;

    /*
       When should use divider 32, 64, 128, 256, 512, 1024, 1280, 1536.
       1. MCG_C7[OSCSEL] selects IRC48M.
       2. MCG_C7[OSCSEL] selects OSC0 and MCG_C2[RANGE] is not 0.
    */
    if (((0U != range)))
    {
        switch (frdiv)
        {
            case 0:
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
                freq >>= 5u;
                break;
            case 6:
                /* 64*20=1280 */
                freq /= 20u;
                break;
            case 7:
                /* 128*12=1536 */
                freq /= 12u;
                break;
            default:
                freq = 0u;
                break;
        }
    }

    return freq;
}

static uint32_t CLOCK_GetInternalRefClkSelectFreq(void)
{
    if (kMCG_IrcSlow == MCG_S_IRCST_VAL)
    {
        /* Slow internal reference clock selected*/
        return s_slowIrcFreq;
    }
    else
    {
        /* Fast internal reference clock selected*/
        return s_fastIrcFreq >> MCG_SC_FCRDIV_VAL;
    }
}

static uint32_t CLOCK_GetFllRefClkFreq(void)
{
    /* If use external reference clock. */
    if (kMCG_FllSrcExternal == MCG_S_IREFST_VAL)
    {
        return CLOCK_GetFllExtRefClkFreq();
    }
    /* If use internal reference clock. */
    else
    {
        return s_slowIrcFreq;
    }
}

static uint32_t CLOCK_GetPll0RefFreq(void)
{
    /* MCG external reference clock. */
    return CLOCK_GetMcgExtClkFreq();
}

static uint8_t CLOCK_GetOscRangeFromFreq(uint32_t freq)
{
    uint8_t range;

    if (freq <= 39063U)
    {
        range = 0U;
    }
    else if (freq <= 8000000U)
    {
        range = 1U;
    }
    else
    {
        range = 2U;
    }

    return range;
}

uint32_t CLOCK_GetOsc0ErClkFreq(void)
{
    if (OSC0->CR & OSC_CR_ERCLKEN_MASK)
    {
        /* Please call CLOCK_SetXtal0Freq base on board setting before using OSC0 clock. */
        assert(g_xtal0Freq);
        return g_xtal0Freq;
    }
    else
    {
        return 0U;
    }
}

uint32_t CLOCK_GetEr32kClkFreq(void)
{
    uint32_t freq;

    switch (SIM_SOPT1_OSC32KSEL_VAL)
    {
        case 0U: /* OSC 32k clock  */
            freq = (CLOCK_GetOsc0ErClkFreq() == 32768U) ? 32768U : 0U;
            break;
        case 2U: /* RTC 32k clock  */
            /* Please call CLOCK_SetXtal32Freq base on board setting before using XTAL32K/RTC_CLKIN cl
            assert(g_xtal32Freq);
            freq = g_xtal32Freq;
            break;
        case 3U: /* LPO clock      */
            freq = LPO_CLK_FREQ;
            break;
        default:
            freq = 0U;
            break;
    }
    return freq;
}

uint32_t CLOCK_GetPllFllSelClkFreq(void)
{
    uint32_t freq;

    switch (SIM_SOPT2_PLLFLLSEL_VAL)
    {
        case 0U: /* FLL. */
            freq = CLOCK_GetFllFreq();
            break;
        case 1U: /* PLL. */
            freq = CLOCK_GetPll0Freq();
            freq >>= 1U;
            break;
        default:
            freq = 0U;
            break;
    }

    return freq;
}

uint32_t CLOCK_GetPlatClkFreq(void)
{
    return CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
}

uint32_t CLOCK_GetFlashClkFreq(void)
{
    uint32_t freq;

    freq = CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
    freq /= (SIM_CLKDIV1_OUTDIV4_VAL + 1);

    return freq;
}

uint32_t CLOCK_GetBusClkFreq(void)
{
    uint32_t freq;

    freq = CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
    freq /= (SIM_CLKDIV1_OUTDIV4_VAL + 1);

    return freq;
}

uint32_t CLOCK_GetCoreSysClkFreq(void)
{
    return CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
}
```

```
uint32_t CLOCK_GetFreq(clock_name_t clockName)
{
   uint32_t freq;

   switch (clockName)
   {
      case kCLOCK_CoreSysClk:
      case kCLOCK_PlatClk:
         freq = CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
         break;
      case kCLOCK_BusClk:
      case kCLOCK_FlashClk:
         freq = CLOCK_GetOutClkFreq() / (SIM_CLKDIV1_OUTDIV1_VAL + 1);
         freq /= (SIM_CLKDIV1_OUTDIV4_VAL + 1);
         break;
      case kCLOCK_PllFllSelClk:
         freq = CLOCK_GetPllFllSelClkFreq();
         break;
      case kCLOCK_Er32kClk:
         freq = CLOCK_GetEr32kClkFreq();
         break;
      case kCLOCK_McgFixedFreqClk:
         freq = CLOCK_GetFixedFreqClkFreq();
         break;
      case kCLOCK_McgInternalRefClk:
         freq = CLOCK_GetInternalRefClkFreq();
         break;
      case kCLOCK_McgFllClk:
         freq = CLOCK_GetFllFreq();
         break;
      case kCLOCK_McgPll0Clk:
         freq = CLOCK_GetPll0Freq();
         break;
      case kCLOCK_LpoClk:
         freq = LPO_CLK_FREQ;
         break;
      case kCLOCK_Osc0ErClk:
         freq = CLOCK_GetOsc0ErClkFreq();
         break;
      default:
         freq = 0U;
         break;
   }

   return freq;
}

void CLOCK_SetSimConfig(sim_clock_config_t const *config)
{
   SIM->CLKDIV1 = config->clkdiv1;
   CLOCK_SetPllFllSelClock(config->pllFllSel);
   CLOCK_SetEr32kClock(config->er32kSrc);
```

```
}

bool CLOCK_EnableUsbfs0Clock(clock_usb_src_t src, uint32_t freq)
{
    bool ret = true;

    CLOCK_DisableClock(kCLOCK_Usbfs0);

    if (kCLOCK_UsbSrcExt == src)
    {
        SIM->SOPT2 &= ~SIM_SOPT2_USBSRC_MASK;
    }
    else
    {
        if (48000000U != freq)
        {
            ret = false;
        }

        SIM->SOPT2 = ((SIM->SOPT2 & ~(SIM_SOPT2_PLLFLLSEL_MASK | SIM_SOPT2_USBSRC_MAS
    }

    CLOCK_EnableClock(kCLOCK_Usbfs0);

    return ret;
}

uint32_t CLOCK_GetOutClkFreq(void)
{
    uint32_t mcgoutclk;
    uint32_t clkst = MCG_S_CLKST_VAL;

    switch (clkst)
    {
        case kMCG_ClkOutStatPll:
            mcgoutclk = CLOCK_GetPll0Freq();
            break;
        case kMCG_ClkOutStatFll:
            mcgoutclk = CLOCK_GetFllFreq();
            break;
        case kMCG_ClkOutStatInt:
            mcgoutclk = CLOCK_GetInternalRefClkSelectFreq();
            break;
        case kMCG_ClkOutStatExt:
            mcgoutclk = CLOCK_GetMcgExtClkFreq();
            break;
        default:
            mcgoutclk = 0U;
            break;
    }
    return mcgoutclk;
}
```

```
uint32_t CLOCK_GetFllFreq(void)
{
   static const uint16_t fllFactorTable[4][2] = {{640, 732}, {1280, 1464}, {1920, 2197}, {2560, 2929}};

   uint8_t drs, dmx32;
   uint32_t freq;

   /* If FLL is not enabled currently, then return 0U. */
   if ((MCG->C2 & MCG_C2_LP_MASK) || (MCG->S & MCG_S_PLLST_MASK))
   {
      return 0U;
   }

   /* Get FLL reference clock frequency. */
   freq = CLOCK_GetFllRefClkFreq();
   if (!freq)
   {
      return freq;
   }

   drs = MCG_C4_DRST_DRS_VAL;
   dmx32 = MCG_C4_DMX32_VAL;

   return freq * fllFactorTable[drs][dmx32];
}

uint32_t CLOCK_GetInternalRefClkFreq(void)
{
   /* If MCGIRCLK is gated. */
   if (!(MCG->C1 & MCG_C1_IRCLKEN_MASK))
   {
      return 0U;
   }

   return CLOCK_GetInternalRefClkSelectFreq();
}

uint32_t CLOCK_GetFixedFreqClkFreq(void)
{
   uint32_t freq = CLOCK_GetFllRefClkFreq();

   /* MCGFFCLK must be no more than MCGOUTCLK/8. */
   if ((freq) && (freq <= (CLOCK_GetOutClkFreq() / 8U)))
   {
      return freq;
   }
   else
   {
      return 0U;
   }
}

uint32_t CLOCK_GetPll0Freq(void)
```

```
{
    uint32_t mcgpll0clk;

    /* If PLL0 is not enabled, return 0. */
    if (!(MCG->S & MCG_S_LOCK0_MASK))
    {
        return 0U;
    }

    mcgpll0clk = CLOCK_GetPll0RefFreq();

    /*
     * Please call CLOCK_SetXtal0Freq base on board setting before using OSC0 clock.
     * Please call CLOCK_SetXtal1Freq base on board setting before using OSC1 clock.
     */
    assert(mcgpll0clk);

    mcgpll0clk /= (FSL_FEATURE_MCG_PLL_PRDIV_BASE + MCG_C5_PRDIV0_VAL);
    mcgpll0clk *= (FSL_FEATURE_MCG_PLL_VDIV_BASE + MCG_C6_VDIV0_VAL);

    return mcgpll0clk;
}

status_t CLOCK_SetExternalRefClkConfig(mcg_oscsel_t oscsel)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
#endif /* MCG_CONFIG_CHECK_PARAM */

    return kStatus_Success;
}

status_t CLOCK_SetInternalRefClkConfig(uint8_t enableMode, mcg_irc_mode_t ircs, uint8_t fcrdiv)
{
    uint32_t mcgOutClkState = MCG_S_CLKST_VAL;
    mcg_irc_mode_t curIrcs = (mcg_irc_mode_t)MCG_S_IRCST_VAL;
    uint8_t curFcrdiv = MCG_SC_FCRDIV_VAL;

#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    /* If MCGIRCLK is used as system clock source. */
    if (kMCG_ClkOutStatInt == mcgOutClkState)
    {
        /* If need to change MCGIRCLK source or driver, return error. */
        if (((kMCG_IrcFast == curIrcs) && (fcrdiv != curFcrdiv)) || (ircs != curIrcs))
        {
            return kStatus_MCG_SourceUsed;
        }
    }
#endif

    /* If need to update the FCRDIV. */
    if (fcrdiv != curFcrdiv)
    {
        /* If fast IRC is in use currently, change to slow IRC. */
        if ((kMCG_IrcFast == curIrcs) && ((mcgOutClkState == kMCG_ClkOutStatInt) || (MCG->C1 & MCG_C
        {
            MCG->C2 = ((MCG->C2 & ~MCG_C2_IRCS_MASK) | (MCG_C2_IRCS(kMCG_IrcSlow)));
            while (MCG_S_IRCST_VAL != kMCG_IrcSlow)
            {
            }
        }
        /* Update FCRDIV. */
        MCG->SC = (MCG->SC & ~(MCG_SC_FCRDIV_MASK | MCG_SC_ATMF_MASK | MCG_SC_LOCS
    }

    /* Set internal reference clock selection. */
    MCG->C2 = (MCG->C2 & ~MCG_C2_IRCS_MASK) | (MCG_C2_IRCS(ircs));
    MCG->C1 = (MCG->C1 & ~(MCG_C1_IRCLKEN_MASK | MCG_C1_IREFSTEN_MASK)) | (uint8_t)enal

    /* If MCGIRCLK is used, need to wait for MCG_S_IRCST. */
    if ((mcgOutClkState == kMCG_ClkOutStatInt) || (enableMode & kMCG_IrclkEnable))
    {
        while (MCG_S_IRCST_VAL != ircs)
        {
        }
    }

    return kStatus_Success;
}

uint32_t CLOCK_CalcPllDiv(uint32_t refFreq, uint32_t desireFreq, uint8_t *prdiv, uint8_t *vdiv)
{
    uint8_t ret_prdiv;           /* PRDIV to return. */
    uint8_t ret_vdiv;            /* VDIV to return.  */
    uint8_t prdiv_min;           /* Min PRDIV value to make reference clock in allowed range. */
    uint8_t prdiv_max;           /* Max PRDIV value to make reference clock in allowed range. */
    uint8_t prdiv_cur;           /* PRDIV value for iteration.    */
    uint8_t vdiv_cur;            /* VDIV value for iteration.     */
    uint32_t ret_freq = 0U;      /* PLL output fequency to return. */
    uint32_t diff = 0xFFFFFFFFU; /* Difference between desireFreq and return frequency. */
    uint32_t ref_div;            /* Reference frequency after PRDIV. */

    /*
       Steps:
       1. Get allowed prdiv with such rules:
          1). refFreq / prdiv >= FSL_FEATURE_MCG_PLL_REF_MIN.
          2). refFreq / prdiv <= FSL_FEATURE_MCG_PLL_REF_MAX.
       2. For each allowed prdiv, there are two candidate vdiv values:
          1). (desireFreq / (refFreq / prdiv)).
          2). (desireFreq / (refFreq / prdiv)) + 1.
          If could get the precise desired frequency, return current prdiv and
          vdiv directly. Otherwise choose the one which is closer to desired
          frequency.
     */

    /* Reference frequency is out of range. */
    if ((refFreq < FSL_FEATURE_MCG_PLL_REF_MIN) ||
        (refFreq > (FSL_FEATURE_MCG_PLL_REF_MAX * (FSL_FEATURE_MCG_PLL_PRDIV_MAX + FS
    {
        return 0U;
    }

    /* refFreq/PRDIV must in a range. First get the allowed PRDIV range. */
    prdiv_max = refFreq / FSL_FEATURE_MCG_PLL_REF_MIN;
    prdiv_min = (refFreq + FSL_FEATURE_MCG_PLL_REF_MAX - 1U) / FSL_FEATURE_MCG_PLL_REF

    /* PRDIV traversal. */
    for (prdiv_cur = prdiv_max; prdiv_cur >= prdiv_min; prdiv_cur--)
    {
        /* Reference frequency after PRDIV. */
        ref_div = refFreq / prdiv_cur;

        vdiv_cur = desireFreq / ref_div;

        if ((vdiv_cur < FSL_FEATURE_MCG_PLL_VDIV_BASE - 1U) || (vdiv_cur > FSL_FEATURE_MCG_PI
        {
            /* No VDIV is available with this PRDIV. */
            continue;
        }

        ret_freq = vdiv_cur * ref_div;

        if (vdiv_cur >= FSL_FEATURE_MCG_PLL_VDIV_BASE)
        {
            if (ret_freq == desireFreq) /* If desire frequency is got. */
            {
                *prdiv = prdiv_cur - FSL_FEATURE_MCG_PLL_PRDIV_BASE;
                *vdiv = vdiv_cur - FSL_FEATURE_MCG_PLL_VDIV_BASE;
                return ret_freq;
            }
            /* New PRDIV/VDIV is closer. */
            if (diff > desireFreq - ret_freq)
            {
                diff = desireFreq - ret_freq;
                ret_prdiv = prdiv_cur;
                ret_vdiv = vdiv_cur;
            }
        }
        vdiv_cur++;
        if (vdiv_cur <= (FSL_FEATURE_MCG_PLL_VDIV_BASE + 31U))
        {
            ret_freq += ref_div;
            /* New PRDIV/VDIV is closer. */
            if (diff > ret_freq - desireFreq)
            {
                diff = ret_freq - desireFreq;
                ret_prdiv = prdiv_cur;
                ret_vdiv = vdiv_cur;
            }
        }
    }

    if (0xFFFFFFFFU != diff)
    {
        /* PRDIV/VDIV found. */
        *prdiv = ret_prdiv - FSL_FEATURE_MCG_PLL_PRDIV_BASE;
        *vdiv = ret_vdiv - FSL_FEATURE_MCG_PLL_VDIV_BASE;
        ret_freq = (refFreq / ret_prdiv) * ret_vdiv;
        return ret_freq;
    }
    else
    {
        /* No proper PRDIV/VDIV found. */
        return 0U;
    }
}

void CLOCK_EnablePll0(mcg_pll_config_t const *config)
{
    assert(config);

    uint8_t mcg_c5 = 0U;

    mcg_c5 |= MCG_C5_PRDIV0(config->prdiv);
    MCG->C5 = mcg_c5; /* Disable the PLL first. */

    MCG->C6 = (MCG->C6 & ~MCG_C6_VDIV0_MASK) | MCG_C6_VDIV0(config->vdiv);

    /* Set enable mode. */
    MCG->C5 |= ((uint32_t)kMCG_PllEnableIndependent | (uint32_t)config->enableMode);

    /* Wait for PLL lock. */
    while (!(MCG->S & MCG_S_LOCK0_MASK))
    {
    }
}

void CLOCK_SetOsc0MonitorMode(mcg_monitor_mode_t mode)
{
    /* Clear the previous flag, MCG_SC[LOCS0]. */
    MCG->SC &= ~MCG_SC_ATMF_MASK;

    if (kMCG_MonitorNone == mode)
    {
        MCG->C6 &= ~MCG_C6_CME0_MASK;
    }
    else
    {
        if (kMCG_MonitorInt == mode)
        {
            MCG->C2 &= ~MCG_C2_LOCRE0_MASK;
        }
        else
        {
            MCG->C2 |= MCG_C2_LOCRE0_MASK;
        }
        MCG->C6 |= MCG_C6_CME0_MASK;
    }
}

void CLOCK_SetPll0MonitorMode(mcg_monitor_mode_t mode)
{
    uint8_t mcg_c8;

    /* Clear previous flag. */
    MCG->S = MCG_S_LOLS0_MASK;

    if (kMCG_MonitorNone == mode)
    {
        MCG->C6 &= ~MCG_C6_LOLIE0_MASK;
    }
    else
    {
        mcg_c8 = MCG->C8;

        if (kMCG_MonitorInt == mode)
        {
            mcg_c8 &= ~MCG_C8_LOLRE_MASK;
        }
        else
        {
            mcg_c8 |= MCG_C8_LOLRE_MASK;
        }
        MCG->C8 = mcg_c8;
        MCG->C6 |= MCG_C6_LOLIE0_MASK;
    }
}

uint32_t CLOCK_GetStatusFlags(void)
{
    uint32_t ret = 0U;
    uint8_t mcg_s = MCG->S;

    if (MCG->SC & MCG_SC_LOCS0_MASK)
    {
        ret |= kMCG_Osc0LostFlag;
    }
    if (mcg_s & MCG_S_OSCINIT0_MASK)
    {
        ret |= kMCG_Osc0InitFlag;
    }
    if (mcg_s & MCG_S_LOLS0_MASK)
    {
        ret |= kMCG_Pll0LostFlag;
    }
    if (mcg_s & MCG_S_LOCK0_MASK)
    {
        ret |= kMCG_Pll0LockFlag;
    }
    return ret;
}

void CLOCK_ClearStatusFlags(uint32_t mask)
{
    if (mask & kMCG_Osc0LostFlag)
    {
        MCG->SC &= ~MCG_SC_ATMF_MASK;
    }
    if (mask & kMCG_Pll0LostFlag)
    {
        MCG->S = MCG_S_LOLS0_MASK;
    }
}

void CLOCK_InitOsc0(osc_config_t const *config)
{
    uint8_t range = CLOCK_GetOscRangeFromFreq(config->freq);

    OSC_SetCapLoad(OSC0, config->capLoad);
    OSC_SetExtRefClkConfig(OSC0, &config->oscerConfig);

    MCG->C2 = ((MCG->C2 & ~OSC_MODE_MASK) | MCG_C2_RANGE(range) | (uint8_t)config->workMo

    if ((kOSC_ModeExt != config->workMode) && (OSC0->CR & OSC_CR_ERCLKEN_MASK))
    {
        /* Wait for stable. */
        while (!(MCG->S & MCG_S_OSCINIT0_MASK))
        {
        }
    }
}

void CLOCK_DeinitOsc0(void)
{
    OSC0->CR = 0U;
    MCG->C2 &= ~OSC_MODE_MASK;
}

status_t CLOCK_TrimInternalRefClk(uint32_t extFreq, uint32_t desireFreq, uint32_t *actualFreq, mcg_atm
{
    uint32_t multi; /* extFreq / desireFreq */
    uint32_t actv;  /* Auto trim value. */
    uint8_t mcg_sc;

    static const uint32_t trimRange[2][2] = {
        /*     Min           Max      */
        {TRIM_SIRC_MIN, TRIM_SIRC_MAX}, /* Slow IRC. */
        {TRIM_FIRC_MIN, TRIM_FIRC_MAX}  /* Fast IRC. */
    };
```

```
if ((extFreq > TRIM_REF_CLK_MAX) || (extFreq < TRIM_REF_CLK_MIN))
{
    return kStatus_MCG_AtmBusClockInvalid;
}

/* Check desired frequency range. */
if ((desireFreq < trimRange[atms][0]) || (desireFreq > trimRange[atms][1]))
{
    return kStatus_MCG_AtmDesiredFreqInvalid;
}

/*
   Make sure internal reference clock is not used to generate bus clock.
   Here only need to check (MCG_S_IREFST == 1).
 */
if (MCG_S_IREFST(kMCG_FllSrcInternal) == (MCG->S & MCG_S_IREFST_MASK))
{
    return kStatus_MCG_AtmIrcUsed;
}

multi = extFreq / desireFreq;
actv = multi * 21U;

if (kMCG_AtmSel4m == atms)
{
    actv *= 128U;
}

/* Now begin to start trim. */
MCG->ATCVL = (uint8_t)actv;
MCG->ATCVH = (uint8_t)(actv >> 8U);

mcg_sc = MCG->SC;
mcg_sc &= ~(MCG_SC_ATMS_MASK | MCG_SC_LOCS0_MASK);
mcg_sc |= (MCG_SC_ATMF_MASK | MCG_SC_ATMS(atms));
MCG->SC = (mcg_sc | MCG_SC_ATME_MASK);

/* Wait for finished. */
while (MCG->SC & MCG_SC_ATME_MASK)
{
}

/* Error occurs? */
if (MCG->SC & MCG_SC_ATMF_MASK)
{
    /* Clear the failed flag. */
    MCG->SC = mcg_sc;
    return kStatus_MCG_AtmHardwareFail;
}

*actualFreq = extFreq / multi;
```

```
    if (kMCG_AtmSel4m == atms)
    {
        s_fastIrcFreq = *actualFreq;
    }
    else
    {
        s_slowIrcFreq = *actualFreq;
    }

    return kStatus_Success;
}

mcg_mode_t CLOCK_GetMode(void)
{
    mcg_mode_t mode = kMCG_ModeError;
    uint32_t clkst = MCG_S_CLKST_VAL;
    uint32_t irefst = MCG_S_IREFST_VAL;
    uint32_t lp = MCG_C2_LP_VAL;
    uint32_t pllst = MCG_S_PLLST_VAL;

    /*------------------------------------------------------------------
                           Mode and Registers

     Mode   |   CLKST    |  IREFST   |  PLLST   |     LP

     FEI    | 00(FLL)   |  1(INT)   |  0(FLL) |     X

     FEE    | 00(FLL)   |  0(EXT)   |  0(FLL) |     X

     FBE    | 10(EXT)   |  0(EXT)   |  0(FLL) |  0(NORMAL)

     FBI    | 01(INT)   |  1(INT)   |  0(FLL) |  0(NORMAL)

     BLPI   | 01(INT)   |  1(INT)   |  0(FLL) |  1(LOW POWER)

     BLPE   | 10(EXT)   |  0(EXT)   |    X    |  1(LOW POWER)

     PEE    | 11(PLL)   |  0(EXT)   |  1(PLL) |     X

     PBE    | 10(EXT)   |  0(EXT)   |  1(PLL) |  O(NORMAL)

     PBI    | 01(INT)   |  1(INT)   |  1(PLL) |  0(NORMAL)
```

```
 PEI   | 11(PLL)  |  1(INT)  |  1(PLL) |     X
```

```
--------------------------------------------------------------------*/

switch (clkst)
{
   case kMCG_ClkOutStatFll:
      if (kMCG_FllSrcExternal == irefst)
      {
         mode = kMCG_ModeFEE;
      }
      else
      {
         mode = kMCG_ModeFEI;
      }
      break;
   case kMCG_ClkOutStatInt:
      if (lp)
      {
         mode = kMCG_ModeBLPI;
      }
      else
      {
         {
            mode = kMCG_ModeFBI;
         }
      }
      break;
   case kMCG_ClkOutStatExt:
      if (lp)
      {
         mode = kMCG_ModeBLPE;
      }
      else
      {
         if (kMCG_PllstPll == pllst)
         {
            mode = kMCG_ModePBE;
         }
         else
         {
            mode = kMCG_ModeFBE;
         }
      }
      break;
   case kMCG_ClkOutStatPll:
   {
      mode = kMCG_ModePEE;
   }
   break;
   default:
```

```
            break;
    }

    return mode;
}

status_t CLOCK_SetFeiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void))
{
    uint8_t mcg_c4;
    bool change_drs = false;

#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    mcg_mode_t mode = CLOCK_GetMode();
    if (!((kMCG_ModeFEI == mode) || (kMCG_ModeFBI == mode) || (kMCG_ModeFBE == mode) || (kMCG_
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif
    mcg_c4 = MCG->C4;

    /*
       Errata: ERR007993
       Workaround: Invert MCG_C4[DMX32] or change MCG_C4[DRST_DRS] before
       reference clock source changes, then reset to previous value after
       reference clock changes.
     */
    if (kMCG_FllSrcExternal == MCG_S_IREFST_VAL)
    {
        change_drs = true;
        /* Change the LSB of DRST_DRS. */
        MCG->C4 ^= (1U << MCG_C4_DRST_DRS_SHIFT);
    }

    /* Set CLKS and IREFS. */
    MCG->C1 =
        ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_IREFS_MASK))) | (MCG_C1_CLKS(kMCG_Clk
                                                        | MCG_C1_IREFS(kMCG_FllSrcInternal)); /* IREFS = 1 */

    /* Wait and check status. */
    while (kMCG_FllSrcInternal != MCG_S_IREFST_VAL)
    {
    }

    /* Errata: ERR007993 */
    if (change_drs)
    {
        MCG->C4 = mcg_c4;
    }

    /* In FEI mode, the MCG_C4[DMX32] is set to 0U. */
    MCG->C4 = (mcg_c4 & ~(MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) | (MCG_C4_DM

    /* Check MCG_S[CLKST] */
    while (kMCG_ClkOutStatFll != MCG_S_CLKST_VAL)
    {
    }

    /* Wait for FLL stable time. */
    if (fllStableDelay)
    {
        fllStableDelay();
    }

    return kStatus_Success;
}

status_t CLOCK_SetFeeMode(uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(vo
{
    uint8_t mcg_c4;
    bool change_drs = false;

#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    mcg_mode_t mode = CLOCK_GetMode();
    if (!((kMCG_ModeFEE == mode) || (kMCG_ModeFBI == mode) || (kMCG_ModeFBE == mode) || (kMCG
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif
    mcg_c4 = MCG->C4;

    /*
       Errata: ERR007993
       Workaround: Invert MCG_C4[DMX32] or change MCG_C4[DRST_DRS] before
       reference clock source changes, then reset to previous value after
       reference clock changes.
     */
    if (kMCG_FllSrcInternal == MCG_S_IREFST_VAL)
    {
        change_drs = true;
        /* Change the LSB of DRST_DRS. */
        MCG->C4 ^= (1U << MCG_C4_DRST_DRS_SHIFT);
    }

    /* Set CLKS and IREFS. */
    MCG->C1 = ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_FRDIV_MASK | MCG_C1_IREFS_MA
               (MCG_C1_CLKS(kMCG_ClkOutSrcOut)         /* CLKS = 0 */
                | MCG_C1_FRDIV(frdiv)                  /* FRDIV */
                | MCG_C1_IREFS(kMCG_FllSrcExternal))); /* IREFS = 0 */

    /* If use external crystal as clock source, wait for it stable. */
    {
        if (MCG->C2 & MCG_C2_EREFS_MASK)
        {
            while (!(MCG->S & MCG_S_OSCINIT0_MASK))
            {
            }
        }
    }

    /* Wait and check status. */
    while (kMCG_FllSrcExternal != MCG_S_IREFST_VAL)
    {
    }

    /* Errata: ERR007993 */
    if (change_drs)
    {
        MCG->C4 = mcg_c4;
    }

    /* Set DRS and DMX32. */
    mcg_c4 = ((mcg_c4 & ~(MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) | (MCG_C4_DMX3
    MCG->C4 = mcg_c4;

    /* Wait for DRST_DRS update. */
    while (MCG->C4 != mcg_c4)
    {
    }

    /* Check MCG_S[CLKST] */
    while (kMCG_ClkOutStatFll != MCG_S_CLKST_VAL)
    {
    }

    /* Wait for FLL stable time. */
    if (fllStableDelay)
    {
        fllStableDelay();
    }

    return kStatus_Success;
}

status_t CLOCK_SetFbiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void))
{
    uint8_t mcg_c4;
    bool change_drs = false;

#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    mcg_mode_t mode = CLOCK_GetMode();

    if (!((kMCG_ModeFEE == mode) || (kMCG_ModeFBI == mode) || (kMCG_ModeFBE == mode) || (kMCG
          (kMCG_ModeBLPI == mode)))

    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif
```

```
    mcg_c4 = MCG->C4;

    MCG->C2 &= ~MCG_C2_LP_MASK; /* Disable lowpower. */

    /*
       Errata: ERR007993
       Workaround: Invert MCG_C4[DMX32] or change MCG_C4[DRST_DRS] before
       reference clock source changes, then reset to previous value after
       reference clock changes.
     */
    if (kMCG_FllSrcExternal == MCG_S_IREFST_VAL)
    {
        change_drs = true;
        /* Change the LSB of DRST_DRS. */
        MCG->C4 ^= (1U << MCG_C4_DRST_DRS_SHIFT);
    }

    /* Set CLKS and IREFS. */
    MCG->C1 =
        ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_IREFS_MASK)) | (MCG_C1_CLKS(kMCG_ClkO
                                                       | MCG_C1_IREFS(kMCG_FllSrcInternal))); /* IREFS = 1 */

    /* Wait and check status. */
    while (kMCG_FllSrcInternal != MCG_S_IREFST_VAL)
    {
    }

    /* Errata: ERR007993 */
    if (change_drs)
    {
        MCG->C4 = mcg_c4;
    }

    while (kMCG_ClkOutStatInt != MCG_S_CLKST_VAL)
    {
    }

    MCG->C4 = (mcg_c4 & ~(MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) | (MCG_C4_DMX

    /* Wait for FLL stable time. */
    if (fllStableDelay)
    {
        fllStableDelay();
    }

    return kStatus_Success;
}

status_t CLOCK_SetFbeMode(uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(vo
{
    uint8_t mcg_c4;
    bool change_drs = false;
```

```
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    mcg_mode_t mode = CLOCK_GetMode();
    if (!((kMCG_ModeFEE == mode) || (kMCG_ModeFBI == mode) || (kMCG_ModeFBE == mode) || (kMCG
          (kMCG_ModePBE == mode) || (kMCG_ModeBLPE == mode)))
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif

    /* Change to FLL mode. */
    MCG->C6 &= ~MCG_C6_PLLS_MASK;
    while (MCG->S & MCG_S_PLLST_MASK)
    {
    }

    /* Set LP bit to enable the FLL */
    MCG->C2 &= ~MCG_C2_LP_MASK;

    mcg_c4 = MCG->C4;

    /*
       Errata: ERR007993
       Workaround: Invert MCG_C4[DMX32] or change MCG_C4[DRST_DRS] before
       reference clock source changes, then reset to previous value after
       reference clock changes.
     */
    if (kMCG_FllSrcInternal == MCG_S_IREFST_VAL)
    {
        change_drs = true;
        /* Change the LSB of DRST_DRS. */
        MCG->C4 ^= (1U << MCG_C4_DRST_DRS_SHIFT);
    }

    /* Set CLKS and IREFS. */
    MCG->C1 = ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_FRDIV_MASK | MCG_C1_IREFS_MA
               (MCG_C1_CLKS(kMCG_ClkOutSrcExternal)    /* CLKS = 2 */
                | MCG_C1_FRDIV(frdiv)                  /* FRDIV = frdiv */
                | MCG_C1_IREFS(kMCG_FllSrcExternal))); /* IREFS = 0 */

    /* If use external crystal as clock source, wait for it stable. */
    {
        if (MCG->C2 & MCG_C2_EREFS_MASK)
        {
            while (!(MCG->S & MCG_S_OSCINIT0_MASK))
            {
            }
        }
    }

    /* Wait for Reference clock Status bit to clear */
    while (kMCG_FllSrcExternal != MCG_S_IREFST_VAL)
    {
    }
    /* Errata: ERR007993 */
    if (change_drs)
    {
        MCG->C4 = mcg_c4;
    }

    /* Set DRST_DRS and DMX32. */
    mcg_c4 = ((mcg_c4 & ~(MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) | (MCG_C4_DMX3

    /* Wait for clock status bits to show clock source is ext ref clk */
    while (kMCG_ClkOutStatExt != MCG_S_CLKST_VAL)
    {
    }

    /* Wait for fll stable time. */
    if (fllStableDelay)
    {
        fllStableDelay();
    }

    return kStatus_Success;
}

status_t CLOCK_SetBlpiMode(void)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    if (MCG_S_CLKST_VAL != kMCG_ClkOutStatInt)
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif /* MCG_CONFIG_CHECK_PARAM */

    /* Set LP. */
    MCG->C2 |= MCG_C2_LP_MASK;

    return kStatus_Success;
}

status_t CLOCK_SetBlpeMode(void)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    if (MCG_S_CLKST_VAL != kMCG_ClkOutStatExt)
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif

    /* Set LP bit to enter BLPE mode. */
    MCG->C2 |= MCG_C2_LP_MASK;

    return kStatus_Success;
}
```

```
status_t CLOCK_SetPbeMode(mcg_pll_clk_select_t pllcs, mcg_pll_config_t const *config)
{
    assert(config);

    /*
       This function is designed to change MCG to PBE mode from PEE/BLPE/FBE,
       but with this workflow, the source mode could be all modes except PEI/PBI.
     */
    MCG->C2 &= ~MCG_C2_LP_MASK; /* Disable lowpower. */

    /* Change to use external clock first. */
    MCG->C1 = ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_IREFS_MASK)) | MCG_C1_CLKS(kM

    /* Wait for CLKST clock status bits to show clock source is ext ref clk */
    while ((MCG->S & (MCG_S_IREFST_MASK | MCG_S_CLKST_MASK)) !=
           (MCG_S_IREFST(kMCG_FllSrcExternal) | MCG_S_CLKST(kMCG_ClkOutStatExt)))
    {
    }

    /* Disable PLL first, then configure PLL. */
    MCG->C6 &= ~MCG_C6_PLLS_MASK;
    while (MCG->S & MCG_S_PLLST_MASK)
    {
    }

    /* Configure the PLL. */
    {
        CLOCK_EnablePll0(config);
    }

    /* Change to PLL mode. */
    MCG->C6 |= MCG_C6_PLLS_MASK;

    /* Wait for PLL mode changed. */
    while (!(MCG->S & MCG_S_PLLST_MASK))
    {
    }

    return kStatus_Success;
}

status_t CLOCK_SetPeeMode(void)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    mcg_mode_t mode = CLOCK_GetMode();
    if (kMCG_ModePBE != mode)
    {
        return kStatus_MCG_ModeUnreachable;
    }
#endif

    /* Change to use PLL/FLL output clock first. */
    MCG->C1 = (MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcOut);

    /* Wait for clock status bits to update */
    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatPll)
    {
    }

    return kStatus_Success;
}

status_t CLOCK_ExternalModeToFbeModeQuick(void)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    if (MCG->S & MCG_S_IREFST_MASK)
    {
        return kStatus_MCG_ModeInvalid;
    }
#endif /* MCG_CONFIG_CHECK_PARAM */

    /* Disable low power */
    MCG->C2 &= ~MCG_C2_LP_MASK;

    MCG->C1 = ((MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcExternal));
    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatExt)
    {
    }

    /* Disable PLL. */
    MCG->C6 &= ~MCG_C6_PLLS_MASK;
    while (MCG->S & MCG_S_PLLST_MASK)
    {
    }

    return kStatus_Success;
}

status_t CLOCK_InternalModeToFbiModeQuick(void)
{
#if (defined(MCG_CONFIG_CHECK_PARAM) && MCG_CONFIG_CHECK_PARAM)
    if (!(MCG->S & MCG_S_IREFST_MASK))
    {
        return kStatus_MCG_ModeInvalid;
    }
#endif

    /* Disable low power */
    MCG->C2 &= ~MCG_C2_LP_MASK;

    MCG->C1 = ((MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcInternal));
    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatInt)
    {
    }
    return kStatus_Success;
}

status_t CLOCK_BootToFeiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void))
{
    return CLOCK_SetFeiMode(dmx32, drs, fllStableDelay);
}

status_t CLOCK_BootToFeeMode(
    mcg_oscsel_t oscsel, uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void))
{
    CLOCK_SetExternalRefClkConfig(oscsel);

    return CLOCK_SetFeeMode(frdiv, dmx32, drs, fllStableDelay);
}

status_t CLOCK_BootToBlpiMode(uint8_t fcrdiv, mcg_irc_mode_t ircs, uint8_t ircEnableMode)
{
    /* If reset mode is FEI mode, set MCGIRCLK and always success. */
    CLOCK_SetInternalRefClkConfig(ircEnableMode, ircs, fcrdiv);

    /* If reset mode is not BLPI, first enter FBI mode. */
    MCG->C1 = (MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcInternal);
    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatInt)
    {
    }

    /* Enter BLPI mode. */
    MCG->C2 |= MCG_C2_LP_MASK;

    return kStatus_Success;
}

status_t CLOCK_BootToBlpeMode(mcg_oscsel_t oscsel)
{
    CLOCK_SetExternalRefClkConfig(oscsel);

    /* Set to FBE mode. */
    MCG->C1 =
        ((MCG->C1 & ~(MCG_C1_CLKS_MASK | MCG_C1_IREFS_MASK)) | (MCG_C1_CLKS(kMCG_ClkO
                                                    | MCG_C1_IREFS(kMCG_FllSrcExternal))); /* IREFS = 0 */

    /* If use external crystal as clock source, wait for it stable. */
    {
        if (MCG->C2 & MCG_C2_EREFS_MASK)
        {
            while (!(MCG->S & MCG_S_OSCINIT0_MASK))
            {
            }
        }
    }

    /* Wait for MCG_S[CLKST] and MCG_S[IREFST]. */
    while ((MCG->S & (MCG_S_IREFST_MASK | MCG_S_CLKST_MASK)) !=
           (MCG_S_IREFST(kMCG_FllSrcExternal) | MCG_S_CLKST(kMCG_ClkOutStatExt)))
    {
    }

    /* In FBE now, start to enter BLPE. */
    MCG->C2 |= MCG_C2_LP_MASK;

    return kStatus_Success;
}

status_t CLOCK_BootToPeeMode(mcg_oscsel_t oscsel, mcg_pll_clk_select_t pllcs, mcg_pll_config_t cons
{
    assert(config);

    CLOCK_SetExternalRefClkConfig(oscsel);

    CLOCK_SetPbeMode(pllcs, config);

    /* Change to use PLL output clock. */
    MCG->C1 = (MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcOut);
    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatPll)
    {
    }

    return kStatus_Success;
}

/*
   The transaction matrix. It defines the path for mode switch, the row is for
   current mode and the column is target mode.
   For example, switch from FEI to PEE:
   1. Current mode FEI, next mode is mcgModeMatrix[FEI][PEE] = FBE, so swith to FBE.
   2. Current mode FBE, next mode is mcgModeMatrix[FBE][PEE] = PBE, so swith to PBE.
   3. Current mode PBE, next mode is mcgModeMatrix[PBE][PEE] = PEE, so swith to PEE.
   Thus the MCG mode has changed from FEI to PEE.
 */
static const mcg_mode_t mcgModeMatrix[8][8] = {
    {kMCG_ModeFEI, kMCG_ModeFBI, kMCG_ModeFBI, kMCG_ModeFEE, kMCG_ModeFBE, kMCG_Mo
     kMCG_ModeFBE}, /* FEI */
    {kMCG_ModeFEI, kMCG_ModeFBI, kMCG_ModeBLPI, kMCG_ModeFEE, kMCG_ModeFBE, kMCG_M
     kMCG_ModeFBE}, /* FBI */
    {kMCG_ModeFBI, kMCG_ModeFBI, kMCG_ModeBLPI, kMCG_ModeFBI, kMCG_ModeFBI, kMCG_Mo
     kMCG_ModeFBI}, /* BLPI */
    {kMCG_ModeFEI, kMCG_ModeFBI, kMCG_ModeFBI, kMCG_ModeFEE, kMCG_ModeFBE, kMCG_Mo
     kMCG_ModeFBE}, /* FEE */
    {kMCG_ModeFEI, kMCG_ModeFBI, kMCG_ModeFBI, kMCG_ModeFEE, kMCG_ModeFBE, kMCG_Mo
     kMCG_ModePBE}, /* FBE */
    {kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_M
     kMCG_ModePBE}, /* BLPE */
    {kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_ModeFBE, kMCG_M
     kMCG_ModePEE}, /* PBE */
    {kMCG_ModePBE, kMCG_ModePBE, kMCG_ModePBE, kMCG_ModePBE, kMCG_ModePBE, kMCG_
```

```
     kMCG_ModePBE} /* PEE */
    /*    FEI          FBI          BLPI          FEE          FBE          BLPE          PBE          PEE */
};

status_t CLOCK_SetMcgConfig(const mcg_config_t *config)
{
    mcg_mode_t next_mode;
    status_t status = kStatus_Success;

    mcg_pll_clk_select_t pllcs = kMCG_PllClkSelPll0;

    /* Re-configure MCGIRCLK, if MCGIRCLK is used as system clock source, then change to FEI/PEI first.
    if (MCG_S_CLKST_VAL == kMCG_ClkOutStatInt)
    {
        MCG->C2 &= ~MCG_C2_LP_MASK; /* Disable lowpower. */

        {
            CLOCK_SetFeiMode(config->dmx32, config->drs, CLOCK_FllStableDelay);
        }
    }

    /* Configure MCGIRCLK. */
    CLOCK_SetInternalRefClkConfig(config->irclkEnableMode, config->ircs, config->fcrdiv);

    next_mode = CLOCK_GetMode();

    do
    {
        next_mode = mcgModeMatrix[next_mode][config->mcgMode];

        switch (next_mode)
        {
            case kMCG_ModeFEI:
                status = CLOCK_SetFeiMode(config->dmx32, config->drs, CLOCK_FllStableDelay);
                break;
            case kMCG_ModeFEE:
                status = CLOCK_SetFeeMode(config->frdiv, config->dmx32, config->drs, CLOCK_FllStableDela
                break;
            case kMCG_ModeFBI:
                status = CLOCK_SetFbiMode(config->dmx32, config->drs, (void (*)(void))0);
                break;
            case kMCG_ModeFBE:
                status = CLOCK_SetFbeMode(config->frdiv, config->dmx32, config->drs, (void (*)(void))0);
                break;
            case kMCG_ModeBLPI:
                status = CLOCK_SetBlpiMode();
                break;
            case kMCG_ModeBLPE:
                status = CLOCK_SetBlpeMode();
                break;
            case kMCG_ModePBE:
                /* If target mode is not PBE or PEE, then only need to set CLKS = EXT here. */
                if ((kMCG_ModePEE == config->mcgMode) || (kMCG_ModePBE == config->mcgMode))
                {
                    {
                        status = CLOCK_SetPbeMode(pllcs, &config->pll0Config);
                    }
                }
                else
                {
                    MCG->C1 = ((MCG->C1 & ~MCG_C1_CLKS_MASK) | MCG_C1_CLKS(kMCG_ClkOutSrcExt
                    while (MCG_S_CLKST_VAL != kMCG_ClkOutStatExt)
                    {
                    }
                }
                break;
            case kMCG_ModePEE:
                status = CLOCK_SetPeeMode();
                break;
            default:
                break;
        }
        if (kStatus_Success != status)
        {
            return status;
        }
    } while (next_mode != config->mcgMode);

    if (config->pll0Config.enableMode & kMCG_PllEnableIndependent)
    {
        CLOCK_EnablePll0(&config->pll0Config);
    }
    else
    {
        MCG->C5 &= ~(uint32_t)kMCG_PllEnableIndependent;
    }
    return kStatus_Success;
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
#include "fsl_smc.h"
#include "fsl_flash.h"

#if (defined(FSL_FEATURE_SMC_HAS_PARAM) && FSL_FEATURE_SMC_HAS_PARAM)
void SMC_GetParam(SMC_Type *base, smc_param_t *param)
{
    uint32_t reg = base->PARAM;
    param->hsrunEnable = (bool)(reg & SMC_PARAM_EHSRUN_MASK);
    param->llsEnable = (bool)(reg & SMC_PARAM_ELLS_MASK);
    param->lls2Enable = (bool)(reg & SMC_PARAM_ELLS2_MASK);
    param->vlls0Enable = (bool)(reg & SMC_PARAM_EVLLS0_MASK);
}
#endif /* FSL_FEATURE_SMC_HAS_PARAM */

void SMC_PreEnterStopModes(void)
{
    flash_prefetch_speculation_status_t speculationStatus =
    {
        kFLASH_prefetchSpeculationOptionDisable, /* Disable instruction speculation.*/
        kFLASH_prefetchSpeculationOptionDisable, /* Disable data speculation.*/
    };

    __disable_irq();
    __ISB();

    /*
     * Before enter stop modes, the flash cache prefetch should be disabled.
     * Otherwise the prefetch might be interrupted by stop, then the data and
     * and instruction from flash are wrong.
     */
    FLASH_PflashSetPrefetchSpeculation(&speculationStatus);
}

void SMC_PostExitStopModes(void)
{
    flash_prefetch_speculation_status_t speculationStatus =
    {
        kFLASH_prefetchSpeculationOptionEnable, /* Enable instruction speculation.*/
        kFLASH_prefetchSpeculationOptionEnable, /* Enable data speculation.*/
    };
    FLASH_PflashSetPrefetchSpeculation(&speculationStatus);

    __enable_irq();
    __ISB();
}

status_t SMC_SetPowerModeRun(SMC_Type *base)
{
    uint8_t reg;

    reg = base->PMCTRL;
    /* configure Normal RUN mode */
    reg &= ~SMC_PMCTRL_RUNM_MASK;
    reg |= (kSMC_RunNormal << SMC_PMCTRL_RUNM_SHIFT);
    base->PMCTRL = reg;

    return kStatus_Success;
}

#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
status_t SMC_SetPowerModeHsrun(SMC_Type *base)
{
    uint8_t reg;

    reg = base->PMCTRL;
    /* configure High Speed RUN mode */
    reg &= ~SMC_PMCTRL_RUNM_MASK;
    reg |= (kSMC_Hsrun << SMC_PMCTRL_RUNM_SHIFT);
    base->PMCTRL = reg;

    return kStatus_Success;
}
#endif /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */

status_t SMC_SetPowerModeWait(SMC_Type *base)
{
    /* configure Normal Wait mode */
    SCB->SCR &= ~SCB_SCR_SLEEPDEEP_Msk;
    __DSB();
    __WFI();
    __ISB();

    return kStatus_Success;
}

status_t SMC_SetPowerModeStop(SMC_Type *base, smc_partial_stop_option_t option)
{
    uint8_t reg;

#if (defined(FSL_FEATURE_SMC_HAS_PSTOPO) && FSL_FEATURE_SMC_HAS_PSTOPO)
    /* configure the Partial Stop mode in Noraml Stop mode */
    reg = base->STOPCTRL;
    reg &= ~SMC_STOPCTRL_PSTOPO_MASK;
    reg |= ((uint32_t)option << SMC_STOPCTRL_PSTOPO_SHIFT);
    base->STOPCTRL = reg;
#endif

    /* configure Normal Stop mode */
    reg = base->PMCTRL;
    reg &= ~SMC_PMCTRL_STOPM_MASK;
    reg |= (kSMC_StopNormal << SMC_PMCTRL_STOPM_SHIFT);
    base->PMCTRL = reg;

    /* Set the SLEEPDEEP bit to enable deep sleep mode (stop mode) */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* read back to make sure the configuration valid before enter stop mode */
    (void)base->PMCTRL;
    __DSB();
    __WFI();
    __ISB();

    /* check whether the power mode enter Stop mode succeed */
    if (base->PMCTRL & SMC_PMCTRL_STOPA_MASK)
    {
        return kStatus_SMC_StopAbort;
    }
    else
    {
        return kStatus_Success;
    }
}

status_t SMC_SetPowerModeVlpr(SMC_Type *base
#if (defined(FSL_FEATURE_SMC_HAS_LPWUI) && FSL_FEATURE_SMC_HAS_LPWUI)
                              ,
                              bool wakeupMode
#endif
                              )
{
    uint8_t reg;

    reg = base->PMCTRL;
#if (defined(FSL_FEATURE_SMC_HAS_LPWUI) && FSL_FEATURE_SMC_HAS_LPWUI)
    /* configure whether the system remains in VLP mode on an interrupt */
    if (wakeupMode)
    {
        /* exits to RUN mode on an interrupt */
        reg |= SMC_PMCTRL_LPWUI_MASK;
    }
    else
    {
        /* remains in VLP mode on an interrupt */
        reg &= ~SMC_PMCTRL_LPWUI_MASK;
    }
```

```
#endif /* FSL_FEATURE_SMC_HAS_LPWUI */

    /* configure VLPR mode */
    reg &= ~SMC_PMCTRL_RUNM_MASK;
    reg |= (kSMC_RunVlpr << SMC_PMCTRL_RUNM_SHIFT);
    base->PMCTRL = reg;

    return kStatus_Success;
}

status_t SMC_SetPowerModeVlpw(SMC_Type *base)
{
    /* configure VLPW mode */
    /* Set the SLEEPDEEP bit to enable deep sleep mode */
    SCB->SCR &= ~SCB_SCR_SLEEPDEEP_Msk;
    __DSB();
    __WFI();
    __ISB();

    return kStatus_Success;
}

status_t SMC_SetPowerModeVlps(SMC_Type *base)
{
    uint8_t reg;

    /* configure VLPS mode */
    reg = base->PMCTRL;
    reg &= ~SMC_PMCTRL_STOPM_MASK;
    reg |= (kSMC_StopVlps << SMC_PMCTRL_STOPM_SHIFT);
    base->PMCTRL = reg;

    /* Set the SLEEPDEEP bit to enable deep sleep mode */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* read back to make sure the configuration valid before enter stop mode */
    (void)base->PMCTRL;
    __DSB();
    __WFI();
    __ISB();

    /* check whether the power mode enter VLPS mode succeed */
    if (base->PMCTRL & SMC_PMCTRL_STOPA_MASK)
    {
        return kStatus_SMC_StopAbort;
    }
    else
    {
        return kStatus_Success;
    }
}

#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
status_t SMC_SetPowerModeLls(SMC_Type *base
#if ((defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMC
    (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO))
                             ,
                             const smc_power_mode_lls_config_t *config
#endif
                             )
{
    uint8_t reg;

    /* configure to LLS mode */
    reg = base->PMCTRL;
    reg &= ~SMC_PMCTRL_STOPM_MASK;
    reg |= (kSMC_StopLls << SMC_PMCTRL_STOPM_SHIFT);
    base->PMCTRL = reg;

/* configure LLS sub-mode*/
#if (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMO
    reg = base->STOPCTRL;
    reg &= ~SMC_STOPCTRL_LLSM_MASK;
    reg |= ((uint32_t)config->subMode << SMC_STOPCTRL_LLSM_SHIFT);
    base->STOPCTRL = reg;
#endif /* FSL_FEATURE_SMC_HAS_LLS_SUBMODE */

#if (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO)
    if (config->enableLpoClock)
    {
        base->STOPCTRL &= ~SMC_STOPCTRL_LPOPO_MASK;
    }
    else
    {
        base->STOPCTRL |= SMC_STOPCTRL_LPOPO_MASK;
    }
#endif /* FSL_FEATURE_SMC_HAS_LPOPO */

    /* Set the SLEEPDEEP bit to enable deep sleep mode */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* read back to make sure the configuration valid before enter stop mode */
    (void)base->PMCTRL;
    __DSB();
    __WFI();
    __ISB();

    /* check whether the power mode enter LLS mode succeed */
    if (base->PMCTRL & SMC_PMCTRL_STOPA_MASK)
    {
        return kStatus_SMC_StopAbort;
    }
    else
    {
        return kStatus_Success;
    }
```

```
}
#endif /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */

#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
status_t SMC_SetPowerModeVlls(SMC_Type *base, const smc_power_mode_vlls_config_t *config)
{
    uint8_t reg;

#if (defined(FSL_FEATURE_SMC_HAS_PORPO) && FSL_FEATURE_SMC_HAS_PORPO)
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
    (defined(FSL_FEATURE_SMC_USE_STOPCTRL_VLLSM) && FSL_FEATURE_SMC_USE_STOPCTR
    (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMOD
    if (config->subMode == kSMC_StopSub0)
#endif
    {
        /* configure whether the Por Detect work in Vlls0 mode */
        if (config->enablePorDetectInVlls0)
        {
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
            base->VLLSCTRL &= ~SMC_VLLSCTRL_PORPO_MASK;
#else
            base->STOPCTRL &= ~SMC_STOPCTRL_PORPO_MASK;
#endif
        }
        else
        {
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
            base->VLLSCTRL |= SMC_VLLSCTRL_PORPO_MASK;
#else
            base->STOPCTRL |= SMC_STOPCTRL_PORPO_MASK;
#endif
        }
    }
#endif /* FSL_FEATURE_SMC_HAS_PORPO */

#if (defined(FSL_FEATURE_SMC_HAS_RAM2_POWER_OPTION) && FSL_FEATURE_SMC_HAS_RAM
    else if (config->subMode == kSMC_StopSub2)
    {
        /* configure whether the Por Detect work in Vlls0 mode */
        if (config->enableRam2InVlls2)
        {
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
            base->VLLSCTRL |= SMC_VLLSCTRL_RAM2PO_MASK;
#else
            base->STOPCTRL |= SMC_STOPCTRL_RAM2PO_MASK;
#endif
        }
        else
        {
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
            base->VLLSCTRL &= ~SMC_VLLSCTRL_RAM2PO_MASK;
#else
            base->STOPCTRL &= ~SMC_STOPCTRL_RAM2PO_MASK;
```

```
#endif
        }
    }
    else
    {
    }
#endif /* FSL_FEATURE_SMC_HAS_RAM2_POWER_OPTION */

    /* configure to VLLS mode */
    reg = base->PMCTRL;
    reg &= ~SMC_PMCTRL_STOPM_MASK;
    reg |= (kSMC_StopVlls << SMC_PMCTRL_STOPM_SHIFT);
    base->PMCTRL = reg;

/* configure the VLLS sub-mode */
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
    reg = base->VLLSCTRL;
    reg &= ~SMC_VLLSCTRL_VLLSM_MASK;
    reg |= ((uint32_t)config->subMode << SMC_VLLSCTRL_VLLSM_SHIFT);
    base->VLLSCTRL = reg;
#else
#if (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMO
    reg = base->STOPCTRL;
    reg &= ~SMC_STOPCTRL_LLSM_MASK;
    reg |= ((uint32_t)config->subMode << SMC_STOPCTRL_LLSM_SHIFT);
    base->STOPCTRL = reg;
#else
    reg = base->STOPCTRL;
    reg &= ~SMC_STOPCTRL_VLLSM_MASK;
    reg |= ((uint32_t)config->subMode << SMC_STOPCTRL_VLLSM_SHIFT);
    base->STOPCTRL = reg;
#endif /* FSL_FEATURE_SMC_HAS_LLS_SUBMODE */
#endif

#if (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO)
    if (config->enableLpoClock)
    {
        base->STOPCTRL &= ~SMC_STOPCTRL_LPOPO_MASK;
    }
    else
    {
        base->STOPCTRL |= SMC_STOPCTRL_LPOPO_MASK;
    }
#endif /* FSL_FEATURE_SMC_HAS_LPOPO */

    /* Set the SLEEPDEEP bit to enable deep sleep mode */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* read back to make sure the configuration valid before enter stop mode */
    (void)base->PMCTRL;
    __DSB();
    __WFI();
    __ISB();
    /* check whether the power mode enter LLS mode succeed */
    if (base->PMCTRL & SMC_PMCTRL_STOPA_MASK)
    {
        return kStatus_SMC_StopAbort;
    }
    else
    {
        return kStatus_Success;
    }
}
#endif /* FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
#include "fsl_uart.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/* UART transfer state. */
enum _uart_tansfer_states
{
    kUART_TxIdle,         /* TX idle. */
```

```
    kUART_TxBusy,         /* TX busy. */
    kUART_RxIdle,         /* RX idle. */
    kUART_RxBusy,         /* RX busy. */
    kUART_RxFramingError, /* Rx framing error */
    kUART_RxParityError   /* Rx parity error */
};

/* Typedef for interrupt handler. */
typedef void (*uart_isr_t)(UART_Type *base, uart_handle_t *handle);

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

/*!
 * @brief Get the UART instance from peripheral base address.
 *
 * @param base UART peripheral base address.
 * @return UART instance.
 */
uint32_t UART_GetInstance(UART_Type *base);

/*!
 * @brief Get the length of received data in RX ring buffer.
 *
 * @param handle UART handle pointer.
 * @return Length of received data in RX ring buffer.
 */
static size_t UART_TransferGetRxRingBufferLength(uart_handle_t *handle);

/*!
 * @brief Check whether the RX ring buffer is full.
 *
 * @param handle UART handle pointer.
 * @retval true  RX ring buffer is full.
 * @retval false RX ring buffer is not full.
 */
static bool UART_TransferIsRxRingBufferFull(uart_handle_t *handle);

/*!
 * @brief Read RX register using non-blocking method.
 *
 * This function reads data from the TX register directly, upper layer must make
 * sure the RX register is full or TX FIFO has data before calling this function.
 *
 * @param base UART peripheral base address.
 * @param data Start addresss of the buffer to store the received data.
 * @param length Size of the buffer.
 */
static void UART_ReadNonBlocking(UART_Type *base, uint8_t *data, size_t length);

/*!
 * @brief Write to TX register using non-blocking method.
 *
 * This function writes data to the TX register directly, upper layer must make
 * sure the TX register is empty or TX FIFO has empty room before calling this function.
 *
 * @note This function does not check whether all the data has been sent out to bus,
 * so before disable TX, check kUART_TransmissionCompleteFlag to ensure the TX is
 * finished.
 *
 * @param base UART peripheral base address.
 * @param data Start addresss of the data to write.
 * @param length Size of the buffer to be sent.
 */
static void UART_WriteNonBlocking(UART_Type *base, const uint8_t *data, size_t length);

/*******************************************************************************
 * Variables
 ******************************************************************************/
/* Array of UART handle. */
#if (defined(UART5))
#define UART_HANDLE_ARRAY_SIZE 6
#else /* UART5 */
#if (defined(UART4))
#define UART_HANDLE_ARRAY_SIZE 5
#else /* UART4 */
#if (defined(UART3))
#define UART_HANDLE_ARRAY_SIZE 4
#else /* UART3 */
#if (defined(UART2))
#define UART_HANDLE_ARRAY_SIZE 3
#else /* UART2 */
#if (defined(UART1))
#define UART_HANDLE_ARRAY_SIZE 2
#else /* UART1 */
#if (defined(UART0))
#define UART_HANDLE_ARRAY_SIZE 1
#else /* UART0 */
#error No UART instance.
#endif /* UART 0 */
#endif /* UART 1 */
#endif /* UART 2 */
#endif /* UART 3 */
#endif /* UART 4 */
#endif /* UART 5 */
static uart_handle_t *s_uartHandle[UART_HANDLE_ARRAY_SIZE];
/* Array of UART peripheral base address. */
static UART_Type *const s_uartBases[] = UART_BASE_PTRS;

/* Array of UART IRQ number. */
static const IRQn_Type s_uartIRQ[] = UART_RX_TX_IRQS;
#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
/* Array of UART clock name. */
static const clock_ip_name_t s_uartClock[] = UART_CLOCKS;
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */
```

```
/* UART ISR for transactional APIs. */
static uart_isr_t s_uartIsr;

/*******************************************************************************
 * Code
 ******************************************************************************/

uint32_t UART_GetInstance(UART_Type *base)
{
    uint32_t instance;
    uint32_t uartArrayCount = (sizeof(s_uartBases) / sizeof(s_uartBases[0]));

    /* Find the instance index from base address mappings. */
    for (instance = 0; instance < uartArrayCount; instance++)
    {
        if (s_uartBases[instance] == base)
        {
            break;
        }
    }

    assert(instance < uartArrayCount);

    return instance;
}

static size_t UART_TransferGetRxRingBufferLength(uart_handle_t *handle)
{
    assert(handle);

    size_t size;

    if (handle->rxRingBufferTail > handle->rxRingBufferHead)
    {
        size = (size_t)(handle->rxRingBufferHead + handle->rxRingBufferSize - handle->rxRingBufferTail);
    }
    else
    {
        size = (size_t)(handle->rxRingBufferHead - handle->rxRingBufferTail);
    }

    return size;
}

static bool UART_TransferIsRxRingBufferFull(uart_handle_t *handle)
{
    assert(handle);

    bool full;

    if (UART_TransferGetRxRingBufferLength(handle) == (handle->rxRingBufferSize - 1U))
    {
        full = true;
    }
    else
    {
        full = false;
    }

    return full;
}

status_t UART_Init(UART_Type *base, const uart_config_t *config, uint32_t srcClock_Hz)
{
    assert(config);
    assert(config->baudRate_Bps);
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    assert(FSL_FEATURE_UART_FIFO_SIZEn(base) >= config->txFifoWatermark);
    assert(FSL_FEATURE_UART_FIFO_SIZEn(base) >= config->rxFifoWatermark);
#endif

    uint16_t sbr = 0;
    uint8_t temp = 0;
    uint32_t baudDiff = 0;

    /* Calculate the baud rate modulo divisor, sbr*/
    sbr = srcClock_Hz / (config->baudRate_Bps * 16);
    /* set sbrTemp to 1 if the sourceClockInHz can not satisfy the desired baud rate */
    if (sbr == 0)
    {
        sbr = 1;
    }
#if defined(FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT) && FSL_FEATURE_
    /* Determine if a fractional divider is needed to fine tune closer to the
     * desired baud, each value of brfa is in 1/32 increments,
     * hence the multiply-by-32. */
    uint32_t tempBaud = 0;

    uint16_t brfa = (2 * srcClock_Hz / (config->baudRate_Bps)) - 32 * sbr;

    /* Calculate the baud rate based on the temporary SBR values and BRFA */
    tempBaud = (srcClock_Hz * 2 / ((sbr * 32 + brfa)));
    baudDiff =
        (tempBaud > config->baudRate_Bps) ? (tempBaud - config->baudRate_Bps) : (config->baudRate_Bp

#else
    /* Calculate the baud rate based on the temporary SBR values */
    baudDiff = (srcClock_Hz / (sbr * 16)) - config->baudRate_Bps;

    /* Select the better value between sbr and (sbr + 1) */
    if (baudDiff > (config->baudRate_Bps - (srcClock_Hz / (16 * (sbr + 1)))))
    {
        baudDiff = config->baudRate_Bps - (srcClock_Hz / (16 * (sbr + 1)));
        sbr++;
    }
```

```
#endif

    /* next, check to see if actual baud rate is within 3% of desired baud rate
     * based on the calculate SBR value */
    if (baudDiff > ((config->baudRate_Bps / 100) * 3))
    {
        /* Unacceptable baud rate difference of more than 3%*/
        return kStatus_UART_BaudrateNotSupport;
    }

#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
    /* Enable uart clock */
    CLOCK_EnableClock(s_uartClock[UART_GetInstance(base)]);
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */

    /* Disable UART TX RX before setting. */
    base->C2 &= ~(UART_C2_TE_MASK | UART_C2_RE_MASK);

    /* Write the sbr value to the BDH and BDL registers*/
    base->BDH = (base->BDH & ~UART_BDH_SBR_MASK) | (uint8_t)(sbr >> 8);
    base->BDL = (uint8_t)sbr;

#if defined(FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT) && FSL_FEATURE_
    /* Write the brfa value to the register*/
    base->C4 = (base->C4 & ~UART_C4_BRFA_MASK) | (brfa & UART_C4_BRFA_MASK);
#endif

    /* Set bit count and parity mode. */
    temp = base->C1 & ~(UART_C1_PE_MASK | UART_C1_PT_MASK | UART_C1_M_MASK);

    if (kUART_ParityDisabled != config->parityMode)
    {
        temp |= (UART_C1_M_MASK | (uint8_t)config->parityMode);
    }

    base->C1 = temp;

#if defined(FSL_FEATURE_UART_HAS_STOP_BIT_CONFIG_SUPPORT) && FSL_FEATURE_UART_H
    /* Set stop bit per char */
    base->BDH = (base->BDH & ~UART_BDH_SBNS_MASK) | UART_BDH_SBNS((uint8_t)config->stopBi
#endif

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    /* Set tx/rx FIFO watermark */
    base->TWFIFO = config->txFifoWatermark;
    base->RWFIFO = config->rxFifoWatermark;

    /* Enable tx/rx FIFO */
    base->PFIFO |= (UART_PFIFO_TXFE_MASK | UART_PFIFO_RXFE_MASK);

    /* Flush FIFO */
    base->CFIFO |= (UART_CFIFO_TXFLUSH_MASK | UART_CFIFO_RXFLUSH_MASK);
#endif
```

```
    /* Enable TX/RX base on configure structure. */
    temp = base->C2;

    if (config->enableTx)
    {
        temp |= UART_C2_TE_MASK;
    }

    if (config->enableRx)
    {
        temp |= UART_C2_RE_MASK;
    }

    base->C2 = temp;

    return kStatus_Success;
}

void UART_Deinit(UART_Type *base)
{
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    /* Wait tx FIFO send out*/
    while (0 != base->TCFIFO)
    {
    }
#endif
    /* Wait last char shoft out */
    while (0 == (base->S1 & UART_S1_TC_MASK))
    {
    }

    /* Disable the module. */
    base->C2 = 0;

#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
    /* Disable uart clock */
    CLOCK_DisableClock(s_uartClock[UART_GetInstance(base)]);
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */
}

void UART_GetDefaultConfig(uart_config_t *config)
{
    assert(config);

    config->baudRate_Bps = 115200U;
    config->parityMode = kUART_ParityDisabled;
#if defined(FSL_FEATURE_UART_HAS_STOP_BIT_CONFIG_SUPPORT) && FSL_FEATURE_UART_H
    config->stopBitCount = kUART_OneStopBit;
#endif
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    config->txFifoWatermark = 0;
    config->rxFifoWatermark = 1;
```

```
#endif
    config->enableTx = false;
    config->enableRx = false;
}

status_t UART_SetBaudRate(UART_Type *base, uint32_t baudRate_Bps, uint32_t srcClock_Hz)
{
    assert(baudRate_Bps);

    uint16_t sbr = 0;
    uint32_t baudDiff = 0;
    uint8_t oldCtrl;

    /* Calculate the baud rate modulo divisor, sbr*/
    sbr = srcClock_Hz / (baudRate_Bps * 16);
    /* set sbrTemp to 1 if the sourceClockInHz can not satisfy the desired baud rate */
    if (sbr == 0)
    {
        sbr = 1;
    }
#if defined(FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT) && FSL_FEATURE_
    /* Determine if a fractional divider is needed to fine tune closer to the
     * desired baud, each value of brfa is in 1/32 increments,
     * hence the multiply-by-32. */
    uint32_t tempBaud = 0;

    uint16_t brfa = (2 * srcClock_Hz / (baudRate_Bps)) - 32 * sbr;

    /* Calculate the baud rate based on the temporary SBR values and BRFA */
    tempBaud = (srcClock_Hz * 2 / ((sbr * 32 + brfa)));
    baudDiff = (tempBaud > baudRate_Bps) ? (tempBaud - baudRate_Bps) : (baudRate_Bps - tempBaud);
#else
    /* Calculate the baud rate based on the temporary SBR values */
    baudDiff = (srcClock_Hz / (sbr * 16)) - baudRate_Bps;

    /* Select the better value between sbr and (sbr + 1) */
    if (baudDiff > (baudRate_Bps - (srcClock_Hz / (16 * (sbr + 1)))))
    {
        baudDiff = baudRate_Bps - (srcClock_Hz / (16 * (sbr + 1)));
        sbr++;
    }
#endif

    /* next, check to see if actual baud rate is within 3% of desired baud rate
     * based on the calculate SBR value */
    if (baudDiff < ((baudRate_Bps / 100) * 3))
    {
        /* Store C2 before disable Tx and Rx */
        oldCtrl = base->C2;

        /* Disable UART TX RX before setting. */
        base->C2 &= ~(UART_C2_TE_MASK | UART_C2_RE_MASK);
        /* Write the sbr value to the BDH and BDL registers*/
        base->BDH = (base->BDH & ~UART_BDH_SBR_MASK) | (uint8_t)(sbr >> 8);
        base->BDL = (uint8_t)sbr;

#if defined(FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT) && FSL_FEATURE_
        /* Write the brfa value to the register*/
        base->C4 = (base->C4 & ~UART_C4_BRFA_MASK) | (brfa & UART_C4_BRFA_MASK);
#endif
        /* Restore C2. */
        base->C2 = oldCtrl;

        return kStatus_Success;
    }
    else
    {
        /* Unacceptable baud rate difference of more than 3%*/
        return kStatus_UART_BaudrateNotSupport;
    }
}

void UART_EnableInterrupts(UART_Type *base, uint32_t mask)
{
    mask &= kUART_AllInterruptsEnable;

    /* The interrupt mask is combined by control bits from several register: ((CFIFO<<24) | (C3<<16) | (C2<<
     */
    base->BDH |= mask;
    base->C2 |= (mask >> 8);
    base->C3 |= (mask >> 16);

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    base->CFIFO |= (mask >> 24);
#endif
}

void UART_DisableInterrupts(UART_Type *base, uint32_t mask)
{
    mask &= kUART_AllInterruptsEnable;

    /* The interrupt mask is combined by control bits from several register: ((CFIFO<<24) | (C3<<16) | (C2<<
     */
    base->BDH &= ~mask;
    base->C2 &= ~(mask >> 8);
    base->C3 &= ~(mask >> 16);

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    base->CFIFO &= ~(mask >> 24);
#endif
}

uint32_t UART_GetEnabledInterrupts(UART_Type *base)
{
    uint32_t temp;
    temp = base->BDH | ((uint32_t)(base->C2) << 8) | ((uint32_t)(base->C3) << 16);

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    temp |= ((uint32_t)(base->CFIFO) << 24);
#endif

    return temp & kUART_AllInterruptsEnable;
}

uint32_t UART_GetStatusFlags(UART_Type *base)
{
    uint32_t status_flag;

    status_flag = base->S1 | ((uint32_t)(base->S2) << 8);

#if defined(FSL_FEATURE_UART_HAS_EXTENDED_DATA_REGISTER_FLAGS) && FSL_FEATURE_U
    status_flag |= ((uint32_t)(base->ED) << 16);
#endif

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    status_flag |= ((uint32_t)(base->SFIFO) << 24);
#endif

    return status_flag;
}

status_t UART_ClearStatusFlags(UART_Type *base, uint32_t mask)
{
    uint8_t reg = base->S2;
    status_t status;

#if defined(FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_UART_HAS_LIN_B
    reg &= ~(UART_S2_RXEDGIF_MASK | UART_S2_LBKDIF_MASK);
#else
    reg &= ~UART_S2_RXEDGIF_MASK;
#endif

    base->S2 = reg | (uint8_t)(mask >> 8);

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    base->SFIFO = (uint8_t)(mask >> 24);
#endif

    if (mask & (kUART_IdleLineFlag | kUART_NoiseErrorFlag | kUART_FramingErrorFlag | kUART_ParityE
    {
        /* Read base->D to clear the flags. */
        (void)base->S1;
        (void)base->D;
    }

    if (mask & kUART_RxOverrunFlag)
    {
        /* Read base->D to clear the flags and Flush all data in FIFO. */
        (void)base->S1;
        (void)base->D;
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        /* Flush FIFO date, otherwise FIFO pointer will be in unknown state. */
        base->CFIFO |= UART_CFIFO_RXFLUSH_MASK;
#endif
    }

    /* If some flags still pending. */
    if (mask & UART_GetStatusFlags(base))
    {
        /* Some flags can only clear or set by the hardware itself, these flags are: kUART_TxDataRegEmptyF
        kUART_TransmissionCompleteFlag, kUART_RxDataRegFullFlag, kUART_RxActiveFlag, kUART_No
        kUART_ParityErrorInRxDataRegFlag, kUART_TxFifoEmptyFlag, kUART_RxFifoEmptyFlag. */
        status = kStatus_UART_FlagCannotClearManually;
    }
    else
    {
        status = kStatus_Success;
    }

    return status;
}

void UART_WriteBlocking(UART_Type *base, const uint8_t *data, size_t length)
{
    /* This API can only ensure that the data is written into the data buffer but can't
    ensure all data in the data buffer are sent into the transmit shift buffer. */
    while (length--)
    {
        while (!(base->S1 & UART_S1_TDRE_MASK))
        {
        }
        base->D = *(data++);
    }
}

static void UART_WriteNonBlocking(UART_Type *base, const uint8_t *data, size_t length)
{
    assert(data);

    size_t i;

    /* The Non Blocking write data API assume user have ensured there is enough space in
    peripheral to write. */
    for (i = 0; i < length; i++)
    {
        base->D = data[i];
    }
}

status_t UART_ReadBlocking(UART_Type *base, uint8_t *data, size_t length)
```

```
{
    assert(data);

    uint32_t statusFlag;

    while (length--)
    {
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        while (!base->RCFIFO)
#else
        while (!(base->S1 & UART_S1_RDRF_MASK))
#endif
        {
            statusFlag = UART_GetStatusFlags(base);

            if (statusFlag & kUART_RxOverrunFlag)
            {
                return kStatus_UART_RxHardwareOverrun;
            }

            if (statusFlag & kUART_NoiseErrorFlag)
            {
                return kStatus_UART_NoiseError;
            }

            if (statusFlag & kUART_FramingErrorFlag)
            {
                return kStatus_UART_FramingError;
            }

            if (statusFlag & kUART_ParityErrorFlag)
            {
                return kStatus_UART_ParityError;
            }
        }
        *(data++) = base->D;
    }

    return kStatus_Success;
}

static void UART_ReadNonBlocking(UART_Type *base, uint8_t *data, size_t length)
{
    assert(data);

    size_t i;

    /* The Non Blocking read data API assume user have ensured there is enough space in
    peripheral to write. */
    for (i = 0; i < length; i++)
    {
        data[i] = base->D;
    }
```

```
}

void UART_TransferCreateHandle(UART_Type *base,
                               uart_handle_t *handle,
                               uart_transfer_callback_t callback,
                               void *userData)
{
    assert(handle);

    uint32_t instance;

    /* Zero the handle. */
    memset(handle, 0, sizeof(*handle));

    /* Set the TX/RX state. */
    handle->rxState = kUART_RxIdle;
    handle->txState = kUART_TxIdle;

    /* Set the callback and user data. */
    handle->callback = callback;
    handle->userData = userData;

#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    /* Note:
       Take care of the RX FIFO, RX interrupt request only assert when received bytes
       equal or more than RX water mark, there is potential issue if RX water
       mark larger than 1.
       For example, if RX FIFO water mark is 2, upper layer needs 5 bytes and
       5 bytes are received. the last byte will be saved in FIFO but not trigger
       RX interrupt because the water mark is 2.
     */
    base->RWFIFO = 1U;
#endif

    /* Get instance from peripheral base address. */
    instance = UART_GetInstance(base);

    /* Save the handle in global variables to support the double weak mechanism. */
    s_uartHandle[instance] = handle;

    s_uartIsr = UART_TransferHandleIRQ;
    /* Enable interrupt in NVIC. */
    EnableIRQ(s_uartIRQ[instance]);
}

void UART_TransferStartRingBuffer(UART_Type *base, uart_handle_t *handle, uint8_t *ringBuffer, size_t
{
    assert(handle);
    assert(ringBuffer);

    /* Setup the ringbuffer address */
    handle->rxRingBuffer = ringBuffer;
    handle->rxRingBufferSize = ringBufferSize;
    handle->rxRingBufferHead = 0U;
    handle->rxRingBufferTail = 0U;

    /* Enable the interrupt to accept the data when user need the ring buffer. */
    UART_EnableInterrupts(
        base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterruptEnable | kUART_Framing
    /* Enable parity error interrupt when parity mode is enable*/
    if (UART_C1_PE_MASK & base->C1)
    {
        UART_EnableInterrupts(base, kUART_ParityErrorInterruptEnable);
    }
}

void UART_TransferStopRingBuffer(UART_Type *base, uart_handle_t *handle)
{
    assert(handle);

    if (handle->rxState == kUART_RxIdle)
    {
        UART_DisableInterrupts(base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterruptE
                                     kUART_FramingErrorInterruptEnable);
        /* Disable parity error interrupt when parity mode is enable*/
        if (UART_C1_PE_MASK & base->C1)
        {
            UART_DisableInterrupts(base, kUART_ParityErrorInterruptEnable);
        }
    }

    handle->rxRingBuffer = NULL;
    handle->rxRingBufferSize = 0U;
    handle->rxRingBufferHead = 0U;
    handle->rxRingBufferTail = 0U;
}

status_t UART_TransferSendNonBlocking(UART_Type *base, uart_handle_t *handle, uart_transfer_t *xfer
{
    assert(handle);
    assert(xfer);
    assert(xfer->dataSize);
    assert(xfer->data);

    status_t status;

    /* Return error if current TX busy. */
    if (kUART_TxBusy == handle->txState)
    {
        status = kStatus_UART_TxBusy;
    }
    else
    {
        handle->txData = xfer->data;
        handle->txDataSize = xfer->dataSize;
        handle->txDataSizeAll = xfer->dataSize;
        handle->txState = kUART_TxBusy;

        /* Enable transmiter interrupt. */
        UART_EnableInterrupts(base, kUART_TxDataRegEmptyInterruptEnable);

        status = kStatus_Success;
    }

    return status;
}

void UART_TransferAbortSend(UART_Type *base, uart_handle_t *handle)
{
    assert(handle);

    UART_DisableInterrupts(base, kUART_TxDataRegEmptyInterruptEnable | kUART_TransmissionCompl

    handle->txDataSize = 0;
    handle->txState = kUART_TxIdle;
}

status_t UART_TransferGetSendCount(UART_Type *base, uart_handle_t *handle, uint32_t *count)
{
    assert(handle);
    assert(count);

    if (kUART_TxIdle == handle->txState)
    {
        return kStatus_NoTransferInProgress;
    }

    *count = handle->txDataSizeAll - handle->txDataSize;

    return kStatus_Success;
}

status_t UART_TransferReceiveNonBlocking(UART_Type *base,
                                         uart_handle_t *handle,
                                         uart_transfer_t *xfer,
                                         size_t *receivedBytes)
{
    assert(handle);
    assert(xfer);
    assert(xfer->data);
    assert(xfer->dataSize);

    uint32_t i;
    status_t status;
    /* How many bytes to copy from ring buffer to user memory. */
    size_t bytesToCopy = 0U;
    /* How many bytes to receive. */
    size_t bytesToReceive;
    /* How many bytes currently have received. */
```

```
size_t bytesCurrentReceived;

/* How to get data:
   1. If RX ring buffer is not enabled, then save xfer->data and xfer->dataSize
      to uart handle, enable interrupt to store received data to xfer->data. When
      all data received, trigger callback.
   2. If RX ring buffer is enabled and not empty, get data from ring buffer first.
      If there are enough data in ring buffer, copy them to xfer->data and return.
      If there are not enough data in ring buffer, copy all of them to xfer->data,
      save the xfer->data remained empty space to uart handle, receive data
      to this empty space and trigger callback when finished. */

if (kUART_RxBusy == handle->rxState)
{
    status = kStatus_UART_RxBusy;
}
else
{
    bytesToReceive = xfer->dataSize;
    bytesCurrentReceived = 0U;

    /* If RX ring buffer is used. */
    if (handle->rxRingBuffer)
    {
        /* Disable UART RX IRQ, protect ring buffer. */
        UART_DisableInterrupts(base, kUART_RxDataRegFullInterruptEnable);

        /* How many bytes in RX ring buffer currently. */
        bytesToCopy = UART_TransferGetRxRingBufferLength(handle);

        if (bytesToCopy)
        {
            bytesToCopy = MIN(bytesToReceive, bytesToCopy);

            bytesToReceive -= bytesToCopy;

            /* Copy data from ring buffer to user memory. */
            for (i = 0U; i < bytesToCopy; i++)
            {
                xfer->data[bytesCurrentReceived++] = handle->rxRingBuffer[handle->rxRingBufferTail];

                /* Wrap to 0. Not use modulo (%) because it might be large and slow. */
                if (handle->rxRingBufferTail + 1U == handle->rxRingBufferSize)
                {
                    handle->rxRingBufferTail = 0U;
                }
                else
                {
                    handle->rxRingBufferTail++;
                }
            }
        }
```

```
            /* If ring buffer does not have enough data, still need to read more data. */
            if (bytesToReceive)
            {
                /* No data in ring buffer, save the request to UART handle. */
                handle->rxData = xfer->data + bytesCurrentReceived;
                handle->rxDataSize = bytesToReceive;
                handle->rxDataSizeAll = bytesToReceive;
                handle->rxState = kUART_RxBusy;
            }

            /* Enable UART RX IRQ if previously enabled. */
            UART_EnableInterrupts(base, kUART_RxDataRegFullInterruptEnable);

            /* Call user callback since all data are received. */
            if (0 == bytesToReceive)
            {
                if (handle->callback)
                {
                    handle->callback(base, handle, kStatus_UART_RxIdle, handle->userData);
                }
            }
        }
        /* Ring buffer not used. */
        else
        {
            handle->rxData = xfer->data + bytesCurrentReceived;
            handle->rxDataSize = bytesToReceive;
            handle->rxDataSizeAll = bytesToReceive;
            handle->rxState = kUART_RxBusy;

            /* Enable RX/Rx overrun/framing error interrupt. */
            UART_EnableInterrupts(base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterrup
                                            kUART_FramingErrorInterruptEnable);
            /* Enable parity error interrupt when parity mode is enable*/
            if (UART_C1_PE_MASK & base->C1)
            {
                UART_EnableInterrupts(base, kUART_ParityErrorInterruptEnable);
            }
        }

        /* Return the how many bytes have read. */
        if (receivedBytes)
        {
            *receivedBytes = bytesCurrentReceived;
        }

        status = kStatus_Success;
    }

    return status;
}

void UART_TransferAbortReceive(UART_Type *base, uart_handle_t *handle)
```

```
{
    assert(handle);

    /* Only abort the receive to handle->rxData, the RX ring buffer is still working. */
    if (!handle->rxRingBuffer)
    {
        /* Disable RX interrupt. */
        UART_DisableInterrupts(base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterruptE
                                     kUART_FramingErrorInterruptEnable);
        /* Disable parity error interrupt when parity mode is enable*/
        if (UART_C1_PE_MASK & base->C1)
        {
            UART_DisableInterrupts(base, kUART_ParityErrorInterruptEnable);
        }
    }

    handle->rxDataSize = 0U;
    handle->rxState = kUART_RxIdle;
}

status_t UART_TransferGetReceiveCount(UART_Type *base, uart_handle_t *handle, uint32_t *count)
{
    assert(handle);
    assert(count);

    if (kUART_RxIdle == handle->rxState)
    {
        return kStatus_NoTransferInProgress;
    }

    if (!count)
    {
        return kStatus_InvalidArgument;
    }

    *count = handle->rxDataSizeAll - handle->rxDataSize;

    return kStatus_Success;
}

void UART_TransferHandleIRQ(UART_Type *base, uart_handle_t *handle)
{
    assert(handle);

    uint8_t count;
    uint8_t tempCount;

    /* If RX framing error */
    if (UART_S1_FE_MASK & base->S1)
    {
        /* Read base->D to clear framing error flag, otherwise the RX does not work. */
        while (base->S1 & UART_S1_RDRF_MASK)
        {
            (void)base->D;
        }
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        /* Flush FIFO date, otherwise FIFO pointer will be in unknown state. */
        base->CFIFO |= UART_CFIFO_RXFLUSH_MASK;
#endif

        handle->rxState = kUART_RxFramingError;
        handle->rxDataSize = 0U;
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_UART_FramingError, handle->userData);
        }
    }

    /* If RX parity error */
    if (UART_S1_PF_MASK & base->S1)
    {
        /* Read base->D to clear parity error flag, otherwise the RX does not work. */
        while (base->S1 & UART_S1_RDRF_MASK)
        {
            (void)base->D;
        }
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        /* Flush FIFO date, otherwise FIFO pointer will be in unknown state. */
        base->CFIFO |= UART_CFIFO_RXFLUSH_MASK;
#endif

        handle->rxState = kUART_RxParityError;
        handle->rxDataSize = 0U;
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_UART_ParityError, handle->userData);
        }
    }

    /* If RX overrun. */
    if (UART_S1_OR_MASK & base->S1)
    {
        /* Read base->D to clear overrun flag, otherwise the RX does not work. */
        while (base->S1 & UART_S1_RDRF_MASK)
        {
            (void)base->D;
        }
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        /* Flush FIFO date, otherwise FIFO pointer will be in unknown state. */
        base->CFIFO |= UART_CFIFO_RXFLUSH_MASK;
#endif
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_UART_RxHardwareOverrun, handle->userData);
        }
    }

    /* Receive data register full */
    if ((UART_S1_RDRF_MASK & base->S1) && (UART_C2_RIE_MASK & base->C2))
    {
/* Get the size that can be stored into buffer for this interrupt. */
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        count = base->RCFIFO;
#else
        count = 1;
#endif

        /* If handle->rxDataSize is not 0, first save data to handle->rxData. */
        while ((count) && (handle->rxDataSize))
        {
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
            tempCount = MIN(handle->rxDataSize, count);
#else
            tempCount = 1;
#endif

            /* Using non block API to read the data from the registers. */
            UART_ReadNonBlocking(base, handle->rxData, tempCount);
            handle->rxData += tempCount;
            handle->rxDataSize -= tempCount;
            count -= tempCount;

            /* If all the data required for upper layer is ready, trigger callback. */
            if (!handle->rxDataSize)
            {
                handle->rxState = kUART_RxIdle;

                if (handle->callback)
                {
                    handle->callback(base, handle, kStatus_UART_RxIdle, handle->userData);
                }
            }
        }

        /* If use RX ring buffer, receive data to ring buffer. */
        if (handle->rxRingBuffer)
        {
            while (count--)
            {
                /* If RX ring buffer is full, trigger callback to notify over run. */
                if (UART_TransferIsRxRingBufferFull(handle))
                {
                    if (handle->callback)
                    {
                        handle->callback(base, handle, kStatus_UART_RxRingBufferOverrun, handle->userData);
                    }
                }

                /* If ring buffer is still full after callback function, the oldest data is overrided. */
                if (UART_TransferIsRxRingBufferFull(handle))
                {
                    /* Increase handle->rxRingBufferTail to make room for new data. */
                    if (handle->rxRingBufferTail + 1U == handle->rxRingBufferSize)
                    {
                        handle->rxRingBufferTail = 0U;
                    }
                    else
                    {
                        handle->rxRingBufferTail++;
                    }
                }

                /* Read data. */
                handle->rxRingBuffer[handle->rxRingBufferHead] = base->D;

                /* Increase handle->rxRingBufferHead. */
                if (handle->rxRingBufferHead + 1U == handle->rxRingBufferSize)
                {
                    handle->rxRingBufferHead = 0U;
                }
                else
                {
                    handle->rxRingBufferHead++;
                }
            }
        }

        else if (!handle->rxDataSize)
        {
            /* Disable RX interrupt/overrun interrupt/fram error interrupt */
            UART_DisableInterrupts(base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterru
                                             kUART_FramingErrorInterruptEnable);

            /* Disable parity error interrupt when parity mode is enable*/
            if (UART_C1_PE_MASK & base->C1)
            {
                UART_DisableInterrupts(base, kUART_ParityErrorInterruptEnable);
            }
        }
        else
        {
        }
    }

    /* If framing error or parity error happened, stop the RX interrupt when ues no ring buffer */
    if (((handle->rxState == kUART_RxFramingError) || (handle->rxState == kUART_RxParityError)) &&
        (!handle->rxRingBuffer))
    {
        UART_DisableInterrupts(base, kUART_RxDataRegFullInterruptEnable | kUART_RxOverrunInterruptE
                                    kUART_FramingErrorInterruptEnable);

        /* Disable parity error interrupt when parity mode is enable*/
        if (UART_C1_PE_MASK & base->C1)
        {
            UART_DisableInterrupts(base, kUART_ParityErrorInterruptEnable);
        }
    }

    /* Send data register empty and the interrupt is enabled. */
    if ((base->S1 & UART_S1_TDRE_MASK) && (base->C2 & UART_C2_TIE_MASK))
    {
/* Get the bytes that available at this moment. */
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        count = FSL_FEATURE_UART_FIFO_SIZEn(base) - base->TCFIFO;
#else
        count = 1;
#endif

        while ((count) && (handle->txDataSize))
        {
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
            tempCount = MIN(handle->txDataSize, count);
#else
            tempCount = 1;
#endif

            /* Using non block API to write the data to the registers. */
            UART_WriteNonBlocking(base, handle->txData, tempCount);
            handle->txData += tempCount;
            handle->txDataSize -= tempCount;
            count -= tempCount;

            /* If all the data are written to data register, TX finished. */
            if (!handle->txDataSize)
            {
                handle->txState = kUART_TxIdle;

                /* Disable TX register empty interrupt. */
                base->C2 = (base->C2 & ~UART_C2_TIE_MASK);

                /* Trigger callback. */
                if (handle->callback)
                {
                    handle->callback(base, handle, kStatus_UART_TxIdle, handle->userData);
                }
            }
        }
    }
}

void UART_TransferHandleErrorIRQ(UART_Type *base, uart_handle_t *handle)
{
    /* To be implemented by User. */
}

#if defined(UART0)
#if ((!(defined(FSL_FEATURE_SOC_LPSCI_COUNT))) || \
     ((defined(FSL_FEATURE_SOC_LPSCI_COUNT)) && (FSL_FEATURE_SOC_LPSCI_COUNT == 0)))
void UART0_DriverIRQHandler(void)
{
    s_uartIsr(UART0, s_uartHandle[0]);
}

void UART0_RX_TX_DriverIRQHandler(void)
{
    UART0_DriverIRQHandler();
}
#endif
#endif

#if defined(UART1)
void UART1_DriverIRQHandler(void)
{
    s_uartIsr(UART1, s_uartHandle[1]);
}

void UART1_RX_TX_DriverIRQHandler(void)
{
    UART1_DriverIRQHandler();
}
#endif

#if defined(UART2)
void UART2_DriverIRQHandler(void)
{
    s_uartIsr(UART2, s_uartHandle[2]);
}

void UART2_RX_TX_DriverIRQHandler(void)
{
    UART2_DriverIRQHandler();
}
#endif

#if defined(UART3)
void UART3_DriverIRQHandler(void)
{
    s_uartIsr(UART3, s_uartHandle[3]);
}

void UART3_RX_TX_DriverIRQHandler(void)
{
    UART3_DriverIRQHandler();
}
#endif
```

```
#if defined(UART4)
void UART4_DriverIRQHandler(void)
{
    s_uartIsr(UART4, s_uartHandle[4]);
}

void UART4_RX_TX_DriverIRQHandler(void)
{
    UART4_DriverIRQHandler();
}
#endif

#if defined(UART5)
void UART5_DriverIRQHandler(void)
{
    s_uartIsr(UART5, s_uartHandle[5]);
}

void UART5_RX_TX_DriverIRQHandler(void)
{
    UART5_DriverIRQHandler();
}
#endif
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 */

#include "fsl_common.h"
#include "fsl_debug_console.h"

#ifndef NDEBUG
#if (defined(__CC_ARM)) || (defined(__ICCARM__))
void __aeabi_assert(const char *failedExpr, const char *file, int line)
{
    PRINTF("ASSERT ERROR \" %s \": file \"%s\" Line \"%d\" \n", failedExpr, file, line);
    for (;;)
    {
        __BKPT(0);
    }
}
#elif(defined(__REDLIB__))

#if SDK_DEBUGCONSOLE
void __assertion_failed(char *_Expr)
{
    PRINTF("%s\n", _Expr);
    for (;;)
    {
        __asm("bkpt #0");
    }
}
#endif

#elif(defined(__GNUC__))
void __assert_func(const char *file, int line, const char *func, const char *failedExpr)
{
    PRINTF("ASSERT ERROR \" %s \": file \"%s\" Line \"%d\" function name \"%s\" \n", failedExpr, file, line,
    for (;;)
    {
        __BKPT(0);
    }
}
#endif /* (defined(__CC_ARM)) ||  (defined (__ICCARM__)) */
#endif /* NDEBUG */

#ifndef __GIC_PRIO_BITS
uint32_t InstallIRQHandler(IRQn_Type irq, uint32_t irqHandler)
{
/* Addresses for VECTOR_TABLE and VECTOR_RAM come from the linker file */
#if defined(__CC_ARM)
    extern uint32_t Image$$VECTOR_ROM$$Base[];
    extern uint32_t Image$$VECTOR_RAM$$Base[];
    extern uint32_t Image$$RW_m_data$$Base[];

#define __VECTOR_TABLE Image$$VECTOR_ROM$$Base
#define __VECTOR_RAM Image$$VECTOR_RAM$$Base
#define __RAM_VECTOR_TABLE_SIZE (((uint32_t)Image$$RW_m_data$$Base - (uint32_t)Image$$VEC
#elif defined(__ICCARM__)
```

```
    extern uint32_t __RAM_VECTOR_TABLE_SIZE[];
    extern uint32_t __VECTOR_TABLE[];
    extern uint32_t __VECTOR_RAM[];
#elif defined(__GNUC__)
    extern uint32_t __VECTOR_TABLE[];
    extern uint32_t __VECTOR_RAM[];
    extern uint32_t __RAM_VECTOR_TABLE_SIZE_BYTES[];
    uint32_t __RAM_VECTOR_TABLE_SIZE = (uint32_t)(__RAM_VECTOR_TABLE_SIZE_BYTES);
#endif /* defined(__CC_ARM) */
    uint32_t n;
    uint32_t ret;
    uint32_t irqMaskValue;

    irqMaskValue = DisableGlobalIRQ();
    if (SCB->VTOR != (uint32_t)__VECTOR_RAM)
    {
        /* Copy the vector table from ROM to RAM */
        for (n = 0; n < ((uint32_t)__RAM_VECTOR_TABLE_SIZE) / sizeof(uint32_t); n++)
        {
            __VECTOR_RAM[n] = __VECTOR_TABLE[n];
        }
        /* Point the VTOR to the position of vector table */
        SCB->VTOR = (uint32_t)__VECTOR_RAM;
    }

    ret = __VECTOR_RAM[irq + 16];
    /* make sure the __VECTOR_RAM is noncachable */
    __VECTOR_RAM[irq + 16] = irqHandler;

    EnableGlobalIRQ(irqMaskValue);

    return ret;
}
#endif

#ifndef CPU_QN908X
#if (defined(FSL_FEATURE_SOC_SYSCON_COUNT) && (FSL_FEATURE_SOC_SYSCON_COUNT > 0)

void EnableDeepSleepIRQ(IRQn_Type interrupt)
{
    uint32_t index = 0;
    uint32_t intNumber = (uint32_t)interrupt;
    while (intNumber >= 32u)
    {
        index++;
        intNumber -= 32u;
    }

    SYSCON->STARTERSET[index] = 1u << intNumber;
    EnableIRQ(interrupt); /* also enable interrupt at NVIC */
}

void DisableDeepSleepIRQ(IRQn_Type interrupt)
```

```
{
    uint32_t index = 0;
    uint32_t intNumber = (uint32_t)interrupt;
    while (intNumber >= 32u)
    {
        index++;
        intNumber -= 32u;
    }

    DisableIRQ(interrupt); /* also disable interrupt at NVIC */
    SYSCON->STARTERCLR[index] = 1u << intNumber;
}
#endif /* FSL_FEATURE_SOC_SYSCON_COUNT */
#else
void EnableDeepSleepIRQ(IRQn_Type interrupt)
{
    uint32_t index = 0;
    uint32_t intNumber = (uint32_t)interrupt;
    while (intNumber >= 32u)
    {
        index++;
        intNumber -= 32u;
    }

    /*   SYSCON->STARTERSET[index] = 1u << intNumber; */
    EnableIRQ(interrupt); /* also enable interrupt at NVIC */
}

void DisableDeepSleepIRQ(IRQn_Type interrupt)
{
    uint32_t index = 0;
    uint32_t intNumber = (uint32_t)interrupt;
    while (intNumber >= 32u)
    {
        index++;
        intNumber -= 32u;
    }

    DisableIRQ(interrupt); /* also disable interrupt at NVIC */
                          /*   SYSCON->STARTERCLR[index] = 1u << intNumber; */
}
#endif /*CPU_QN908X */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 "fsl_lpsci.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/* LPSCI transfer state. */
enum _lpsci_tansfer_state
{
    kLPSCI_TxIdle,         /*!< TX idle. */
    kLPSCI_TxBusy,         /*!< TX busy. */
    kLPSCI_RxIdle,         /*!< RX idle. */
    kLPSCI_RxBusy,         /*!< RX busy. */
    kLPSCI_RxFramingError, /* Rx framing error */
    kLPSCI_RxParityError   /* Rx parity error */
};

/* Typedef for interrupt handler. */
typedef void (*lpsci_isr_t)(UART0_Type *base, lpsci_handle_t *handle);

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

/*!
 * @brief Get the LPSCI instance from peripheral base address.
 *
 * @param base LPSCI peripheral base address.
 * @return LPSCI instance.
 */
uint32_t LPSCI_GetInstance(UART0_Type *base);

/*!
```

```
 * @brief Get the length of received data in RX ring buffer.
 *
 * @userData handle LPSCI handle pointer.
 * @return Length of received data in RX ring buffer.
 */
static size_t LPSCI_TransferGetRxRingBufferLength(lpsci_handle_t *handle);

/*!
 * @brief Check whether the RX ring buffer is full.
 *
 * @parram handle LPSCI handle pointer.
 * @retval true  RX ring buffer is full.
 * @retval false RX ring buffer is not full.
 */
static bool LPSCI_TransferIsRxRingBufferFull(lpsci_handle_t *handle);

/*!
 * @brief Write to TX register using non-blocking method.
 *
 * This function writes data to the TX register directly, upper layer must make
 * sure the TX register is empty before calling this function.
 *
 * @note This function does not check whether all the data has been sent out to bus,
 * so before disable TX, check kLPSCI_TransmissionCompleteFlag to ensure the TX is
 * finished.
 *
 * @param base LPSCI peripheral base address.
 * @param data Start addresss of the data to write.
 * @param length Size of the buffer to be sent.
 */
static void LPSCI_WriteNonBlocking(UART0_Type *base, const uint8_t *data, size_t length);

/*!
 * @brief Read RX data register using blocking method.
 *
 * This function polls the RX register, waits for the RX register full
 * then read data from TX register.
 *
 * @param base LPSCI peripheral base address.
 * @param data Start addresss of the buffer to store the received data.
 * @param length Size of the buffer.
 */
static void LPSCI_ReadNonBlocking(UART0_Type *base, uint8_t *data, size_t length);

/*******************************************************************************
 * Variables
 ******************************************************************************/
/* Array of LPSCI handle. */
static lpsci_handle_t *s_lpsciHandle[FSL_FEATURE_SOC_LPSCI_COUNT];

/* Array of LPSCI peripheral base address. */
static UART0_Type *const s_lpsciBases[] = UART0_BASE_PTRS;
```

```
/* Array of LPSCI IRQ number. */
static const IRQn_Type s_lpsciIRQ[] = UART0_RX_TX_IRQS;
#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
/* Array of LPSCI clock name. */
static const clock_ip_name_t s_lpsciClock[] = UART0_CLOCKS;
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */

/* LPSCI ISR for transactional APIs. */
static lpsci_isr_t s_lpsciIsr;

/*******************************************************************************
 * Code
 ******************************************************************************/

uint32_t LPSCI_GetInstance(UART0_Type *base)
{
    uint32_t instance;

    /* Find the instance index from base address mappings. */
    for (instance = 0; instance < ARRAY_SIZE(s_lpsciBases); instance++)
    {
        if (s_lpsciBases[instance] == base)
        {
            break;
        }
    }

    assert(instance < ARRAY_SIZE(s_lpsciBases));

    return instance;
}

static size_t LPSCI_TransferGetRxRingBufferLength(lpsci_handle_t *handle)
{
    assert(handle);

    size_t size;

    if (handle->rxRingBufferTail > handle->rxRingBufferHead)
    {
        size = (size_t)(handle->rxRingBufferHead + handle->rxRingBufferSize - handle->rxRingBufferTail);
    }
    else
    {
        size = (size_t)(handle->rxRingBufferHead - handle->rxRingBufferTail);
    }

    return size;
}

static bool LPSCI_TransferIsRxRingBufferFull(lpsci_handle_t *handle)
{
    assert(handle);
    bool full;

    if (LPSCI_TransferGetRxRingBufferLength(handle) == (handle->rxRingBufferSize - 1U))
    {
        full = true;
    }
    else
    {
        full = false;
    }

    return full;
}

static void LPSCI_ReadNonBlocking(UART0_Type *base, uint8_t *data, size_t length)
{
    assert(data);

    /* The Non Blocking read data API assume user have ensured there is enough space in
    peripheral to write. */
    size_t i;

    for (i = 0; i < length; i++)
    {
        data[i] = base->D;
    }
}

static void LPSCI_WriteNonBlocking(UART0_Type *base, const uint8_t *data, size_t length)
{
    assert(data);

    /* The Non Blocking write data API assume user have ensured there is enough space in
    peripheral to write. */
    size_t i;

    for (i = 0; i < length; i++)
    {
        base->D = data[i];
    }
}

status_t LPSCI_Init(UART0_Type *base, const lpsci_config_t *config, uint32_t srcClock_Hz)
{
    assert(config);
    assert(config->baudRate_Bps);

    uint8_t temp;
    uint16_t sbr = 0;
    uint16_t sbrTemp;
    uint32_t osr = 0;
    uint32_t osrTemp;
    uint32_t tempDiff, calculatedBaud, baudDiff;

    /* This LPSCI instantiation uses a slightly different baud rate calculation
     * The idea is to use the best OSR (over-sampling rate) possible
     * Note, OSR is typically hard-set to 16 in other LPSCI instantiations
     * loop to find the best OSR value possible, one that generates minimum baudDiff
     * iterate through the rest of the supported values of OSR */

    baudDiff = config->baudRate_Bps;
    for (osrTemp = 4; osrTemp <= 32; osrTemp++)
    {
        /* calculate the temporary sbr value   */
        sbrTemp = (srcClock_Hz / (config->baudRate_Bps * osrTemp));
        /* set sbrTemp to 1 if the sourceClockInHz can not satisfy the desired baud rate */
        if (sbrTemp == 0)
        {
            sbrTemp = 1;
        }
        /* Calculate the baud rate based on the temporary OSR and SBR values */
        calculatedBaud = (srcClock_Hz / (osrTemp * sbrTemp));

        tempDiff = calculatedBaud - config->baudRate_Bps;

        /* Select the better value between srb and (sbr + 1) */
        if (tempDiff > (config->baudRate_Bps - (srcClock_Hz / (osrTemp * (sbrTemp + 1)))))
        {
            tempDiff = config->baudRate_Bps - (srcClock_Hz / (osrTemp * (sbrTemp + 1)));
            sbrTemp++;
        }

        if (tempDiff <= baudDiff)
        {
            baudDiff = tempDiff;
            osr = osrTemp; /* update and store the best OSR value calculated*/
            sbr = sbrTemp; /* update store the best SBR value calculated*/
        }
    }

    /* next, check to see if actual baud rate is within 3% of desired baud rate
     * based on the best calculate OSR value */
    if (baudDiff > ((config->baudRate_Bps / 100) * 3))
    {
        /* Unacceptable baud rate difference of more than 3%*/
        return kStatus_LPSCI_BaudrateNotSupport;
    }

#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
    /* Enable LPSCI clock */
    CLOCK_EnableClock(s_lpsciClock[LPSCI_GetInstance(base)]);
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */

    /* Disable TX RX before setting. */
    base->C2 &= ~(UART0_C2_TE_MASK | UART0_C2_RE_MASK);
    /* Acceptable baud rate */
    /* Check if OSR is between 4x and 7x oversampling*/
    /* If so, then "BOTHEDGE" sampling must be turned on*/
    if ((osr > 3) && (osr < 8))
    {
        base->C5 |= UART0_C5_BOTHEDGE_MASK;
    }

    /* program the osr value (bit value is one less than actual value)*/
    base->C4 = ((base->C4 & ~UART0_C4_OSR_MASK) | (osr - 1));

    /* program the sbr (divider) value obtained above*/
    base->BDH = ((base->C4 & ~UART0_BDH_SBR_MASK) | (uint8_t)(sbr >> 8));
    base->BDL = (uint8_t)sbr;

    /* set parity mode */
    temp = base->C1 & ~(UART0_C1_PE_MASK | UART0_C1_PT_MASK | UART0_C1_M_MASK);

    if (kLPSCI_ParityDisabled != config->parityMode)
    {
        temp |= (uint8_t)config->parityMode | UART0_C1_M_MASK;
    }

    base->C1 = temp;

#if defined(FSL_FEATURE_LPSCI_HAS_STOP_BIT_CONFIG_SUPPORT) && FSL_FEATURE_LPSCI_H
    /* set stop bit per char */
    base->BDH &= ~UART0_BDH_SBNS_MASK;
    base->BDH |= UART0_BDH_SBNS((uint8_t)config->stopBitCount);
#endif

    /* Enable TX/RX base on configure structure. */
    temp = base->C2;

    if (config->enableTx)
    {
        temp |= UART0_C2_TE_MASK;
    }

    if (config->enableRx)
    {
        temp |= UART0_C2_RE_MASK;
    }

    base->C2 = temp;

    return kStatus_Success;
}

void LPSCI_Deinit(UART0_Type *base)
{
    /* Wait last char out */
    while (0 == (base->S1 & UART0_S1_TC_MASK))
    {
    }

#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLO
    /* Disable LPSCI clock */
    CLOCK_DisableClock(s_lpsciClock[LPSCI_GetInstance(base)]);
#endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */
}

void LPSCI_GetDefaultConfig(lpsci_config_t *config)
{
    assert(config);

    config->baudRate_Bps = 115200U;
    config->parityMode = kLPSCI_ParityDisabled;
    config->stopBitCount = kLPSCI_OneStopBit;
    config->enableTx = false;
    config->enableRx = false;
}

status_t LPSCI_SetBaudRate(UART0_Type *base, uint32_t baudRate_Bps, uint32_t srcClock_Hz)
{
    assert(baudRate_Bps);

    uint16_t sbrTemp;
    uint32_t osr = 0, sbr = 0;
    uint8_t osrTemp;
    uint32_t tempDiff, calculatedBaud, baudDiff;
    uint8_t oldCtrl;

    /* This LPSCI instantiation uses a slightly different baud rate calculation
     * The idea is to use the best OSR (over-sampling rate) possible
     * Note, OSR is typically hard-set to 16 in other LPSCI instantiations
     * First calculate the baud rate using the minimum OSR possible (4). */
    baudDiff = baudRate_Bps;
    for (osrTemp = 4; osrTemp <= 32; osrTemp++)
    {
        /* calculate the temporary sbr value   */
        sbrTemp = (srcClock_Hz / (baudRate_Bps * osrTemp));
        /* set sbrTemp to 1 if the sourceClockInHz can not satisfy the desired baud rate */
        if (sbrTemp == 0)
        {
            sbrTemp = 1;
        }
        /* Calculate the baud rate based on the temporary OSR and SBR values */
        calculatedBaud = (srcClock_Hz / (osrTemp * sbrTemp));

        tempDiff = calculatedBaud - baudRate_Bps;

        /* Select the better value between srb and (sbr + 1) */
        if (tempDiff > (baudRate_Bps - (srcClock_Hz / (osrTemp * (sbrTemp + 1)))))
        {
            tempDiff = baudRate_Bps - (srcClock_Hz / (osrTemp * (sbrTemp + 1)));
            sbrTemp++;
        }

        if (tempDiff <= baudDiff)
        {
            baudDiff = tempDiff;
            osr = osrTemp; /* update and store the best OSR value calculated*/
            sbr = sbrTemp; /* update store the best SBR value calculated*/
        }
    }

    /* next, check to see if actual baud rate is within 3% of desired baud rate
     * based on the best calculate OSR value */
    if (baudDiff < ((baudRate_Bps / 100) * 3))
    {
        /* Store C2 before disable Tx and Rx */
        oldCtrl = base->C2;

        /* Disable LPSCI TX RX before setting. */
        base->C2 &= ~(UART0_C2_TE_MASK | UART0_C2_RE_MASK);

        /* Acceptable baud rate */
        /* Check if OSR is between 4x and 7x oversampling*/
        /* If so, then "BOTHEDGE" sampling must be turned on*/
        if ((osr > 3) && (osr < 8))
        {
            base->C5 |= UART0_C5_BOTHEDGE_MASK;
        }

        /* program the osr value (bit value is one less than actual value)*/
        base->C4 = ((base->C4 & ~UART0_C4_OSR_MASK) | (osr - 1));

        /* program the sbr (divider) value obtained above*/
        base->BDH = ((base->C4 & ~UART0_BDH_SBR_MASK) | (uint8_t)(sbr >> 8));
        base->BDL = (uint8_t)sbr;

        /* Restore C2. */
        base->C2 = oldCtrl;

        return kStatus_Success;
    }
    else
    {
        /* Unacceptable baud rate difference of more than 3%*/
        return kStatus_LPSCI_BaudrateNotSupport;
    }
}

void LPSCI_EnableInterrupts(UART0_Type *base, uint32_t mask)
{
    mask &= kLPSCI_AllInterruptsEnable;
    /* The interrupt mask is combined by control bits from several register: ((C3<<16) | (C2<<8) |(BDH))
     */
    base->BDH |= mask;
    base->C2 |= (mask >> 8);
    base->C3 |= (mask >> 16);
}

void LPSCI_DisableInterrupts(UART0_Type *base, uint32_t mask)
{
    mask &= kLPSCI_AllInterruptsEnable;

    /* The interrupt mask is combined by control bits from several register: ((C3<<16) | (C2<<8) |(BDH))
     */
    base->BDH &= ~mask;
    base->C2 &= ~(mask >> 8);
    base->C3 &= ~(mask >> 16);
}

uint32_t LPSCI_GetEnabledInterrupts(UART0_Type *base)
{
    uint32_t temp;
    temp = base->BDH | ((uint32_t)(base->C2) << 8) | ((uint32_t)(base->C3) << 16);

    return temp & kLPSCI_AllInterruptsEnable;
}

uint32_t LPSCI_GetStatusFlags(UART0_Type *base)
{
    uint32_t status_flag;
    status_flag = base->S1 | ((uint32_t)(base->S2) << 8);

#if defined(FSL_FEATURE_LPSCI_HAS_EXTENDED_DATA_REGISTER_FLAGS) && FSL_FEATURE_L
    status_flag |= ((uint32_t)(base->ED) << 16);
#endif

    return status_flag;
}

status_t LPSCI_ClearStatusFlags(UART0_Type *base, uint32_t mask)
{
    volatile uint8_t dummy = 0;
    status_t status;
    dummy++; /* For unused variable warning */

#if defined(FSL_FEATURE_LPSCI_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_LPSCI_HAS_LIN_E
    if (mask & kLPSCI_LinBreakFlag)
    {
        base->S2 = UART0_S2_LBKDIF_MASK;
        mask &= ~(uint32_t)kLPSCI_LinBreakFlag;
    }
#endif

    if (mask & kLPSCI_RxActiveEdgeFlag)
    {
       base->S2 = UART0_S2_RXEDGIF_MASK;
       mask &= ~(uint32_t)kLPSCI_RxActiveEdgeFlag;
    }

    if ((mask & (kLPSCI_IdleLineFlag | kLPSCI_RxOverrunFlag | kLPSCI_NoiseErrorFlag | kLPSCI_Framing
               kLPSCI_ParityErrorFlag)))
    {
       base->S1 = (mask & (kLPSCI_IdleLineFlag | kLPSCI_RxOverrunFlag | kLPSCI_NoiseErrorFlag |
                       kLPSCI_FramingErrorFlag | kLPSCI_ParityErrorFlag));
       mask &= ~(uint32_t)(kLPSCI_IdleLineFlag | kLPSCI_RxOverrunFlag | kLPSCI_NoiseErrorFlag |
                       kLPSCI_FramingErrorFlag | kLPSCI_ParityErrorFlag);
    }

    if (mask)
    {
       /* Some flags can only clear or set by the hardware itself, these flags are: kLPSCI_TxDataRegEmptyF
       kLPSCI_TransmissionCompleteFlag, kLPSCI_RxDataRegFullFlag, kLPSCI_RxActiveFlag,
       kLPSCI_NoiseErrorInRxDataRegFlag,
       kLPSCI_ParityErrorInRxDataRegFlag*/

       status = kStatus_LPSCI_FlagCannotClearManually;
    }
    else
    {
       status = kStatus_Success;
    }

    return status;
}

void LPSCI_WriteBlocking(UART0_Type *base, const uint8_t *data, size_t length)
{
    assert(data);

    /* This API can only ensure that the data is written into the data buffer but can't
    ensure all data in the data buffer are sent into the transmit shift buffer. */
    while (length--)
    {
       while (!(base->S1 & UART0_S1_TDRE_MASK))
       {
       }
       base->D = *(data++);
    }
}

status_t LPSCI_ReadBlocking(UART0_Type *base, uint8_t *data, size_t length)
{
    assert(data);

    uint32_t statusFlag;

    while (length--)
    {
        while (!(base->S1 & UART0_S1_RDRF_MASK))
        {
            statusFlag = LPSCI_GetStatusFlags(base);

            if (statusFlag & kLPSCI_RxOverrunFlag)
            {
                LPSCI_ClearStatusFlags(base, kLPSCI_RxOverrunFlag);
                return kStatus_LPSCI_RxHardwareOverrun;
            }

            if (statusFlag & kLPSCI_NoiseErrorFlag)
            {
                LPSCI_ClearStatusFlags(base, kLPSCI_NoiseErrorFlag);
                return kStatus_LPSCI_NoiseError;
            }

            if (statusFlag & kLPSCI_FramingErrorFlag)
            {
                LPSCI_ClearStatusFlags(base, kLPSCI_FramingErrorFlag);
                return kStatus_LPSCI_FramingError;
            }

            if (statusFlag & kLPSCI_ParityErrorFlag)
            {
                LPSCI_ClearStatusFlags(base, kLPSCI_ParityErrorFlag);
                return kStatus_LPSCI_ParityError;
            }
        }
        *(data++) = base->D;
    }

    return kStatus_Success;
}

void LPSCI_TransferCreateHandle(UART0_Type *base,
                                lpsci_handle_t *handle,
                                lpsci_transfer_callback_t callback,
                                void *userData)
{
    assert(handle);

    uint32_t instance;

    /* Zero the handle. */
    memset(handle, 0, sizeof(lpsci_handle_t));

    /* Set the TX/RX state. */
    handle->rxState = kLPSCI_RxIdle;
    handle->txState = kLPSCI_TxIdle;

    /* Set the callback and user data. */
    handle->callback = callback;
    handle->userData = userData;

    /* Get instance from peripheral base address. */
    instance = LPSCI_GetInstance(base);

    /* Save the handle in global variables to support the double weak mechanism. */
    s_lpsciHandle[instance] = handle;

    s_lpsciIsr = LPSCI_TransferHandleIRQ;

    /* Enable interrupt in NVIC. */
    EnableIRQ(s_lpsciIRQ[instance]);
}

void LPSCI_TransferStartRingBuffer(UART0_Type *base, lpsci_handle_t *handle, uint8_t *ringBuffer, size_
{
    assert(handle);
    assert(ringBuffer);

    /* Setup the ringbuffer address */
    handle->rxRingBuffer = ringBuffer;
    handle->rxRingBufferSize = ringBufferSize;
    handle->rxRingBufferHead = 0U;
    handle->rxRingBufferTail = 0U;

    /* Enable the interrupt to accept the data when user need the ring buffer. */
    LPSCI_EnableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterruptEn
                                 kLPSCI_FramingErrorInterruptEnable);
    /* Enable parity error interrupt when parity mode is enable*/
    if (UART0_C1_PE_MASK & base->C1)
    {
        LPSCI_EnableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
    }
}

void LPSCI_TransferStopRingBuffer(UART0_Type *base, lpsci_handle_t *handle)
{
    assert(handle);

    if (handle->rxState == kLPSCI_RxIdle)
    {
        LPSCI_DisableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterrupt
                                      kLPSCI_FramingErrorInterruptEnable);

        /* Disable parity error interrupt when parity mode is enable*/
        if (UART0_C1_PE_MASK & base->C1)
        {
            LPSCI_DisableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
        }
    }

    handle->rxRingBuffer = NULL;
    handle->rxRingBufferSize = 0U;
    handle->rxRingBufferHead = 0U;
    handle->rxRingBufferTail = 0U;
}

status_t LPSCI_TransferSendNonBlocking(UART0_Type *base, lpsci_handle_t *handle, lpsci_transfer_t *x
{
    assert(handle);
    assert(xfer);
    assert(xfer->dataSize);
    assert(xfer->data);

    status_t status;

    /* Return error if current TX busy. */
    if (kLPSCI_TxBusy == handle->txState)
    {
        status = kStatus_LPSCI_TxBusy;
    }
    else
    {
        handle->txData = xfer->data;
        handle->txDataSize = xfer->dataSize;
        handle->txDataSizeAll = xfer->dataSize;
        handle->txState = kLPSCI_TxBusy;

        /* Enable transmiter interrupt. */
        LPSCI_EnableInterrupts(base, kLPSCI_TxDataRegEmptyInterruptEnable);

        status = kStatus_Success;
    }

    return status;
}

void LPSCI_TransferAbortSend(UART0_Type *base, lpsci_handle_t *handle)
{
    assert(handle);

    LPSCI_DisableInterrupts(base, kLPSCI_TxDataRegEmptyInterruptEnable | kLPSCI_TransmissionComp

    handle->txDataSize = 0;
    handle->txState = kLPSCI_TxIdle;
}

status_t LPSCI_TransferGetSendCount(UART0_Type *base, lpsci_handle_t *handle, uint32_t *count)
{
    assert(handle);
    assert(count);

    if (kLPSCI_TxIdle == handle->txState)
    {
        return kStatus_NoTransferInProgress;
    }
    *count = handle->txDataSizeAll - handle->txDataSize;

    return kStatus_Success;
}

status_t LPSCI_TransferReceiveNonBlocking(UART0_Type *base,
                                          lpsci_handle_t *handle,
                                          lpsci_transfer_t *xfer,
                                          size_t *receivedBytes)
{
    assert(handle);
    assert(xfer);
    assert(xfer->dataSize);
    assert(xfer->data);

    uint32_t i;
    status_t status;
    /* How many bytes to copy from ring buffer to user memory. */
    size_t bytesToCopy = 0U;
    /* How many bytes to receive. */
    size_t bytesToReceive;
    /* How many bytes currently have received. */
    size_t bytesCurrentReceived;

    /* How to get data:
       1. If RX ring buffer is not enabled, then save xfer->data and xfer->dataSize
          to lpsci handle, enable interrupt to store received data to xfer->data. When
          all data received, trigger callback.
       2. If RX ring buffer is enabled and not empty, get data from ring buffer first.
          If there are enough data in ring buffer, copy them to xfer->data and return.
          If there are not enough data in ring buffer, copy all of them to xfer->data,
          save the xfer->data remained empty space to lpsci handle, receive data
          to this empty space and trigger callback when finished. */

    if (kLPSCI_RxBusy == handle->rxState)
    {
        status = kStatus_LPSCI_RxBusy;
    }
    else
    {
        bytesToReceive = xfer->dataSize;
        bytesCurrentReceived = 0U;

        /* If RX ring buffer is used. */
        if (handle->rxRingBuffer)
        {
            /* Disable LPSCI RX IRQ, protect ring buffer. */
            LPSCI_DisableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable);

            /* How many bytes in RX ring buffer currently. */
            bytesToCopy = LPSCI_TransferGetRxRingBufferLength(handle);
        if (bytesToCopy)
        {
            bytesToCopy = MIN(bytesToReceive, bytesToCopy);

            bytesToReceive -= bytesToCopy;

            /* Copy data from ring buffer to user memory. */
            for (i = 0U; i < bytesToCopy; i++)
            {
                xfer->data[bytesCurrentReceived++] = handle->rxRingBuffer[handle->rxRingBufferTail];

                /* Wrap to 0. Not use modulo (%) because it might be large and slow. */
                if (handle->rxRingBufferTail + 1U == handle->rxRingBufferSize)
                {
                    handle->rxRingBufferTail = 0U;
                }
                else
                {
                    handle->rxRingBufferTail++;
                }
            }
        }

        /* If ring buffer does not have enough data, still need to read more data. */
        if (bytesToReceive)
        {
            /* No data in ring buffer, save the request to lpsci handle. */
            handle->rxData = xfer->data + bytesCurrentReceived;
            handle->rxDataSize = bytesToReceive;
            handle->rxDataSizeAll = bytesToReceive;
            handle->rxState = kLPSCI_RxBusy;
        }

        /* Enable LPSCI RX IRQ if previously enabled. */
        LPSCI_EnableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable);

        /* Call user callback since all data are received. */
        if (0 == bytesToReceive)
        {
            if (handle->callback)
            {
                handle->callback(base, handle, kStatus_LPSCI_RxIdle, handle->userData);
            }
        }
    }
    /* Ring buffer not used. */
    else
    {
        handle->rxData = xfer->data + bytesCurrentReceived;
        handle->rxDataSize = bytesToReceive;
        handle->rxDataSizeAll = bytesToReceive;
        handle->rxState = kLPSCI_RxBusy;
```

```
            /* Enable RX interrupt. */
            LPSCI_EnableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterru
                                   kLPSCI_FramingErrorInterruptEnable);
            /* Enable parity error interrupt when parity mode is enable*/
            if (UART0_C1_PE_MASK & base->C1)
            {
                LPSCI_EnableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
            }
        }

        /* Return the how many bytes have read. */
        if (receivedBytes)
        {
            *receivedBytes = bytesCurrentReceived;
        }

        status = kStatus_Success;
    }

    return status;
}

void LPSCI_TransferAbortReceive(UART0_Type *base, lpsci_handle_t *handle)
{
    assert(handle);

    /* Only abort the receive to handle->rxData, the RX ring buffer is still working. */
    if (!handle->rxRingBuffer)
    {
        /* Disable RX interrupt. */
        LPSCI_DisableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterrupt
                                kLPSCI_FramingErrorInterruptEnable);
        /* Disable parity error interrupt when parity mode is enable*/
        if (UART0_C1_PE_MASK & base->C1)
        {
            LPSCI_DisableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
        }
    }

    handle->rxDataSize = 0U;
    handle->rxState = kLPSCI_RxIdle;
}

status_t LPSCI_TransferGetReceiveCount(UART0_Type *base, lpsci_handle_t *handle, uint32_t *count)
{
    assert(handle);
    assert(count);

    if (kLPSCI_RxIdle == handle->rxState)
    {
        return kStatus_NoTransferInProgress;
    }
    *count = handle->rxDataSizeAll - handle->rxDataSize;

    return kStatus_Success;
}

void LPSCI_TransferHandleIRQ(UART0_Type *base, lpsci_handle_t *handle)
{
    assert(handle);

    uint8_t count;
    uint8_t tempCount;

    /* If RX parity error */
    if (UART0_S1_PF_MASK & base->S1)
    {
        handle->rxState = kLPSCI_RxParityError;

        LPSCI_ClearStatusFlags(base, kLPSCI_ParityErrorFlag);
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_LPSCI_ParityError, handle->userData);
        }
    }

    /* If RX framing error */
    if (UART0_S1_FE_MASK & base->S1)
    {
        handle->rxState = kLPSCI_RxFramingError;

        LPSCI_ClearStatusFlags(base, kLPSCI_FramingErrorFlag);
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_LPSCI_FramingError, handle->userData);
        }
    }
    /* If RX overrun. */
    if (UART0_S1_OR_MASK & base->S1)
    {
        while (UART0_S1_RDRF_MASK & base->S1)
        {
            (void)base->D;
        }

        LPSCI_ClearStatusFlags(base, kLPSCI_RxOverrunFlag);
        /* Trigger callback. */
        if (handle->callback)
        {
            handle->callback(base, handle, kStatus_LPSCI_RxHardwareOverrun, handle->userData);
        }
    }
    /* Receive data register full */
    if ((UART0_S1_RDRF_MASK & base->S1) && (UART0_C2_RIE_MASK & base->C2))
    {
/* Get the size that can be stored into buffer for this interrupt. */
#if defined(FSL_FEATURE_LPSCI_HAS_FIFO) && FSL_FEATURE_LPSCI_HAS_FIFO
        count = base->RCFIFO;
#else
        count = 1;
#endif

        /* If handle->rxDataSize is not 0, first save data to handle->rxData. */
        while ((count) && (handle->rxDataSize))
        {
#if defined(FSL_FEATURE_LPSCI_HAS_FIFO) && FSL_FEATURE_LPSCI_HAS_FIFO
            tempCount = MIN(handle->rxDataSize, count);
#else
            tempCount = 1;
#endif

            /* Using non block API to read the data from the registers. */
            LPSCI_ReadNonBlocking(base, handle->rxData, tempCount);
            handle->rxData += tempCount;
            handle->rxDataSize -= tempCount;
            count -= tempCount;

            /* If all the data required for upper layer is ready, trigger callback. */
            if (!handle->rxDataSize)
            {
                handle->rxState = kLPSCI_RxIdle;

                if (handle->callback)
                {
                    handle->callback(base, handle, kStatus_LPSCI_RxIdle, handle->userData);
                }
            }
        }

        /* If use RX ring buffer, receive data to ring buffer. */
        if (handle->rxRingBuffer)
        {
            while (count--)
            {
                /* If RX ring buffer is full, trigger callback to notify over run. */
                if (LPSCI_TransferIsRxRingBufferFull(handle))
                {
                    if (handle->callback)
                    {
                        handle->callback(base, handle, kStatus_LPSCI_RxRingBufferOverrun, handle->userData);
                    }
                }

                /* If ring buffer is still full after callback function, the oldest data is overrided. */
                if (LPSCI_TransferIsRxRingBufferFull(handle))
                {
                    /* Increase handle->rxRingBufferTail to make room for new data. */
                    if (handle->rxRingBufferTail + 1U == handle->rxRingBufferSize)
                    {
                        handle->rxRingBufferTail = 0U;
                    }
                    else
                    {
                        handle->rxRingBufferTail++;
                    }
                }

                /* Read data. */
                handle->rxRingBuffer[handle->rxRingBufferHead] = base->D;

                /* Increase handle->rxRingBufferHead. */
                if (handle->rxRingBufferHead + 1U == handle->rxRingBufferSize)
                {
                    handle->rxRingBufferHead = 0U;
                }
                else
                {
                    handle->rxRingBufferHead++;
                }
            }
        }
        /* If no receive requst pending, stop RX interrupt. */
        else if (!handle->rxDataSize)
        {
            LPSCI_DisableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterru
                                    kLPSCI_FramingErrorInterruptEnable);

            /* Disable parity error interrupt when parity mode is enable*/
            if (UART0_C1_PE_MASK & base->C1)
            {
                LPSCI_DisableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
            }
        }
        else
        {
        }
    }
    /* If framing error or parity error happened, stop the RX interrupt when ues no ring buffer */
    if (((handle->rxState == kLPSCI_RxFramingError) || (handle->rxState == kLPSCI_RxParityError)) &&
        (!handle->rxRingBuffer))
    {
        LPSCI_DisableInterrupts(base, kLPSCI_RxDataRegFullInterruptEnable | kLPSCI_RxOverrunInterrupt
                                kLPSCI_FramingErrorInterruptEnable);

        /* Disable parity error interrupt when parity mode is enable*/
        if (UART0_C1_PE_MASK & base->C1)
        {
            LPSCI_DisableInterrupts(base, kLPSCI_ParityErrorInterruptEnable);
        }
    }

    /* Send data register empty and the interrupt is enabled. */
    if ((base->S1 & UART0_S1_TDRE_MASK) && (base->C2 & UART0_C2_TIE_MASK))
    {
/* Get the bytes that available at this moment. */
#if defined(FSL_FEATURE_LPSCI_HAS_FIFO) && FSL_FEATURE_LPSCI_HAS_FIFO
        count = FSL_FEATURE_LPSCI_FIFO_SIZEn(base) - base->TCFIFO;
#else
        count = 1;
#endif

        while ((count) && (handle->txDataSize))
        {
#if defined(FSL_FEATURE_LPSCI_HAS_FIFO) && FSL_FEATURE_LPSCI_HAS_FIFO
            tempCount = MIN(handle->txDataSize, count);
#else
            tempCount = 1;
#endif

            /* Using non block API to write the data to the registers. */
            LPSCI_WriteNonBlocking(base, handle->txData, tempCount);
            handle->txData += tempCount;
            handle->txDataSize -= tempCount;
            count -= tempCount;

            /* If all the data are written to data register, enable TX complete interrupt. */
            if (!handle->txDataSize)
            {
                handle->txState = kLPSCI_TxIdle;

                /* Disable TX register empty interrupt. */
                base->C2 &= ~UART0_C2_TIE_MASK;

                /* Trigger callback. */
                if (handle->callback)
                {
                    handle->callback(base, handle, kStatus_LPSCI_TxIdle, handle->userData);
                }
            }
        }
    }
}

void LPSCI_TransferHandleErrorIRQ(UART0_Type *base, lpsci_handle_t *handle)
{
    /* To be implemented by User. */
}

#if defined(UART0)
void UART0_DriverIRQHandler(void)
{
    s_lpsciIsr(UART0, s_lpsciHandle[0]);
}

#endif
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 "fsl_gpio.h"

/*******************************************************************************
 * Variables
 ******************************************************************************/
static PORT_Type *const s_portBases[] = PORT_BASE_PTRS;
static GPIO_Type *const s_gpioBases[] = GPIO_BASE_PTRS;

/*******************************************************************************
* Prototypes
******************************************************************************/

/*!
* @brief Gets the GPIO instance according to the GPIO base
*
* @param base    GPIO peripheral base pointer(PTA, PTB, PTC, etc.)
* @retval GPIO instance
*/
```

```
static uint32_t GPIO_GetInstance(GPIO_Type *base);

/*******************************************************************************
 * Code
 ******************************************************************************/

static uint32_t GPIO_GetInstance(GPIO_Type *base)
{
    uint32_t instance;

    /* Find the instance index from base address mappings. */
    for (instance = 0; instance < ARRAY_SIZE(s_gpioBases); instance++)
    {
        if (s_gpioBases[instance] == base)
        {
            break;
        }
    }

    assert(instance < ARRAY_SIZE(s_gpioBases));

    return instance;
}

void GPIO_PinInit(GPIO_Type *base, uint32_t pin, const gpio_pin_config_t *config)
{
    assert(config);

    if (config->pinDirection == kGPIO_DigitalInput)
    {
        base->PDDR &= ~(1U << pin);
    }
    else
    {
        GPIO_WritePinOutput(base, pin, config->outputLogic);
        base->PDDR |= (1U << pin);
    }
}

uint32_t GPIO_GetPinsInterruptFlags(GPIO_Type *base)
{
    uint8_t instance;
    PORT_Type *portBase;
    instance = GPIO_GetInstance(base);
    portBase = s_portBases[instance];
    return portBase->ISFR;
}

void GPIO_ClearPinsInterruptFlags(GPIO_Type *base, uint32_t mask)
{
    uint8_t instance;
    PORT_Type *portBase;
    instance = GPIO_GetInstance(base);
    portBase = s_portBases[instance];
    portBase->ISFR = mask;
}

#if defined(FSL_FEATURE_GPIO_HAS_ATTRIBUTE_CHECKER) && FSL_FEATURE_GPIO_HAS_ATTR
void GPIO_CheckAttributeBytes(GPIO_Type *base, gpio_checker_attribute_t attribute)
{
    base->GACR = ((uint32_t)attribute << GPIO_GACR_ACB0_SHIFT) | ((uint32_t)attribute << GPIO_GAC
                 ((uint32_t)attribute << GPIO_GACR_ACB2_SHIFT) | ((uint32_t)attribute << GPIO_GACR_ACB
}
#endif

#if defined(FSL_FEATURE_SOC_FGPIO_COUNT) && FSL_FEATURE_SOC_FGPIO_COUNT

/*******************************************************************************
 * Variables
 ******************************************************************************/
static FGPIO_Type *const s_fgpioBases[] = FGPIO_BASE_PTRS;

/*******************************************************************************
* Prototypes
******************************************************************************/
/*!
* @brief Gets the FGPIO instance according to the GPIO base
*
* @param base    FGPIO peripheral base pointer(PTA, PTB, PTC, etc.)
* @retval FGPIO instance
*/
static uint32_t FGPIO_GetInstance(FGPIO_Type *base);

/*******************************************************************************
 * Code
 ******************************************************************************/

static uint32_t FGPIO_GetInstance(FGPIO_Type *base)
{
    uint32_t instance;

    /* Find the instance index from base address mappings. */
    for (instance = 0; instance < ARRAY_SIZE(s_fgpioBases); instance++)
    {
        if (s_fgpioBases[instance] == base)
        {
            break;
        }
    }

    assert(instance < ARRAY_SIZE(s_fgpioBases));

    return instance;
}

void FGPIO_PinInit(FGPIO_Type *base, uint32_t pin, const gpio_pin_config_t *config)
```

```
{
    assert(config);

    if (config->pinDirection == kGPIO_DigitalInput)
    {
        base->PDDR &= ~(1U << pin);
    }
    else
    {
        FGPIO_WritePinOutput(base, pin, config->outputLogic);
        base->PDDR |= (1U << pin);
    }
}

uint32_t FGPIO_GetPinsInterruptFlags(FGPIO_Type *base)
{
    uint8_t instance;
    instance = FGPIO_GetInstance(base);
    PORT_Type *portBase;
    portBase = s_portBases[instance];
    return portBase->ISFR;
}

void FGPIO_ClearPinsInterruptFlags(FGPIO_Type *base, uint32_t mask)
{
    uint8_t instance;
    instance = FGPIO_GetInstance(base);
    PORT_Type *portBase;
    portBase = s_portBases[instance];
    portBase->ISFR = mask;
}

#if defined(FSL_FEATURE_FGPIO_HAS_ATTRIBUTE_CHECKER) && FSL_FEATURE_FGPIO_HAS_AT
void FGPIO_CheckAttributeBytes(FGPIO_Type *base, gpio_checker_attribute_t attribute)
{
    base->GACR = (attribute << FGPIO_GACR_ACB0_SHIFT) | (attribute << FGPIO_GACR_ACB1_SHIFT
                 (attribute << FGPIO_GACR_ACB2_SHIFT) | (attribute << FGPIO_GACR_ACB3_SHIFT);
}
#endif

#endif /* FSL_FEATURE_SOC_FGPIO_COUNT */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015-2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 "fsl_flash.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*!
 * @name Misc utility defines
 * @{
 */
/*! @brief Alignment utility. */
#ifndef ALIGN_DOWN
#define ALIGN_DOWN(x, a) ((x) & (uint32_t)(-((int32_t)(a))))
#endif
#ifndef ALIGN_UP
#define ALIGN_UP(x, a) (-((int32_t)((uint32_t)(-((int32_t)(x))) & (uint32_t)(-((int32_t)(a))))))
#endif

/*! @brief Join bytes to word utility. */
#define B1P4(b) (((uint32_t)(b)&0xFFU) << 24)
#define B1P3(b) (((uint32_t)(b)&0xFFU) << 16)
#define B1P2(b) (((uint32_t)(b)&0xFFU) << 8)
#define B1P1(b) ((uint32_t)(b)&0xFFU)
#define B2P3(b) (((uint32_t)(b)&0xFFFFU) << 16)
#define B2P2(b) (((uint32_t)(b)&0xFFFFU) << 8)
#define B2P1(b) ((uint32_t)(b)&0xFFFFU)
#define B3P2(b) (((uint32_t)(b)&0xFFFFFFU) << 8)
#define B3P1(b) ((uint32_t)(b)&0xFFFFFFU)
#define BYTES_JOIN_TO_WORD_1_3(x, y) (B1P4(x) | B3P1(y))
#define BYTES_JOIN_TO_WORD_2_2(x, y) (B2P3(x) | B2P1(y))
#define BYTES_JOIN_TO_WORD_3_1(x, y) (B3P2(x) | B1P1(y))
#define BYTES_JOIN_TO_WORD_1_1_2(x, y, z) (B1P4(x) | B1P3(y) | B2P1(z))
#define BYTES_JOIN_TO_WORD_1_2_1(x, y, z) (B1P4(x) | B2P2(y) | B1P1(z))
#define BYTES_JOIN_TO_WORD_2_1_1(x, y, z) (B2P3(x) | B1P2(y) | B1P1(z))
```

```
#define BYTES_JOIN_TO_WORD_1_1_1_1(x, y, z, w) (B1P4(x) | B1P3(y) | B1P2(z) | B1P1(w))
/*@}*/

/*!
 * @name Secondary flash configuration
 * @{
 */
/*! @brief Indicates whether the secondary flash has its own protection register in flash module. */
#if defined(FSL_FEATURE_FLASH_HAS_MULTIPLE_FLASH) && defined(FTFE_FPROTS_PROTS_MAS
#define FLASH_SSD_SECONDARY_FLASH_HAS_ITS_OWN_PROTECTION_REGISTER (1)
#else
#define FLASH_SSD_SECONDARY_FLASH_HAS_ITS_OWN_PROTECTION_REGISTER (0)
#endif

/*! @brief Indicates whether the secondary flash has its own Execute-Only access register in flash module.
#if defined(FSL_FEATURE_FLASH_HAS_MULTIPLE_FLASH) && defined(FTFE_FACSSS_SGSIZE_S_M
#define FLASH_SSD_SECONDARY_FLASH_HAS_ITS_OWN_ACCESS_REGISTER (1)
#else
#define FLASH_SSD_SECONDARY_FLASH_HAS_ITS_OWN_ACCESS_REGISTER (0)
#endif
/*@}*/

/*!
 * @name Flash cache ands speculation control defines
 * @{
 */
#if defined(MCM_PLACR_CFCC_MASK) || defined(MCM_CPCR2_CCBC_MASK)
#define FLASH_CACHE_IS_CONTROLLED_BY_MCM (1)
#else
#define FLASH_CACHE_IS_CONTROLLED_BY_MCM (0)
#endif
#if defined(FMC_PFB0CR_CINV_WAY_MASK) || defined(FMC_PFB01CR_CINV_WAY_MASK)
#define FLASH_CACHE_IS_CONTROLLED_BY_FMC (1)
#else
#define FLASH_CACHE_IS_CONTROLLED_BY_FMC (0)
#endif
#if defined(MCM_PLACR_DFCS_MASK)
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MCM (1)
#else
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MCM (0)
#endif
#if defined(MSCM_OCMDR_OCM1_MASK) || defined(MSCM_OCMDR_OCMC1_MASK)
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM (1)
#else
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM (0)
#endif
#if defined(FMC_PFB0CR_S_INV_MASK) || defined(FMC_PFB0CR_S_B_INV_MASK) || defined(FMC_PF
    defined(FMC_PFB01CR_S_B_INV_MASK)
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC (1)
#else
#define FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC (0)
#endif
/*@}*/
```

```
/*! @brief Data flash IFR map Field*/
#if defined(FSL_FEATURE_FLASH_IS_FTFE) && FSL_FEATURE_FLASH_IS_FTFE
#define DFLASH_IFR_READRESOURCE_START_ADDRESS 0x8003F8U
#else /* FSL_FEATURE_FLASH_IS_FTFL == 1 or FSL_FEATURE_FLASH_IS_FTFA = =1 */
#define DFLASH_IFR_READRESOURCE_START_ADDRESS 0x8000F8U
#endif

/*!
 * @name Reserved FlexNVM size (For a variety of purposes) defines
 * @{
 */
#define FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED 0xFFFFFFFFU
#define FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_RESERVED 0xFFFFU
/*@}*/

/*!
 * @name Flash Program Once Field defines
 * @{
 */
#if defined(FSL_FEATURE_FLASH_IS_FTFA) && FSL_FEATURE_FLASH_IS_FTFA
/* FTFA parts(eg. K80, KL80, L5K) support both 4-bytes and 8-bytes unit size */
#define FLASH_PROGRAM_ONCE_MIN_ID_8BYTES \
    0x10U /* Minimum Index indcating one of Progam Once Fields which is accessed in 8-byte records */
#define FLASH_PROGRAM_ONCE_MAX_ID_8BYTES \
    0x13U /* Maximum Index indcating one of Progam Once Fields which is accessed in 8-byte records */
#define FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT 1
#define FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT 1
#elif defined(FSL_FEATURE_FLASH_IS_FTFE) && FSL_FEATURE_FLASH_IS_FTFE
/* FTFE parts(eg. K65, KE18) only support 8-bytes unit size */
#define FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT 0
#define FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT 1
#elif defined(FSL_FEATURE_FLASH_IS_FTFL) && FSL_FEATURE_FLASH_IS_FTFL
/* FTFL parts(eg. K20) only support 4-bytes unit size */
#define FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT 1
#define FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT 0
#endif
/*@}*/

/*!
 * @name Flash security status defines
 * @{
 */
#define FLASH_SECURITY_STATE_KEYEN 0x80U
#define FLASH_SECURITY_STATE_UNSECURED 0x02U
#define FLASH_NOT_SECURE 0x01U
#define FLASH_SECURE_BACKDOOR_ENABLED 0x02U
#define FLASH_SECURE_BACKDOOR_DISABLED 0x04U
/*@}*/

/*!
 * @name Flash controller command numbers
 * @{
```

```
 */
#define FTFx_VERIFY_BLOCK 0x00U                    /*!< RD1BLK*/
#define FTFx_VERIFY_SECTION 0x01U                  /*!< RD1SEC*/
#define FTFx_PROGRAM_CHECK 0x02U                   /*!< PGMCHK*/
#define FTFx_READ_RESOURCE 0x03U                   /*!< RDRSRC*/
#define FTFx_PROGRAM_LONGWORD 0x06U                /*!< PGM4*/
#define FTFx_PROGRAM_PHRASE 0x07U                  /*!< PGM8*/
#define FTFx_ERASE_BLOCK 0x08U                     /*!< ERSBLK*/
#define FTFx_ERASE_SECTOR 0x09U                    /*!< ERSSCR*/
#define FTFx_PROGRAM_SECTION 0x0BU                 /*!< PGMSEC*/
#define FTFx_GENERATE_CRC 0x0CU                    /*!< CRCGEN*/
#define FTFx_VERIFY_ALL_BLOCK 0x40U                /*!< RD1ALL*/
#define FTFx_READ_ONCE 0x41U                       /*!< RDONCE or RDINDEX*/
#define FTFx_PROGRAM_ONCE 0x43U                    /*!< PGMONCE or PGMINDEX*/
#define FTFx_ERASE_ALL_BLOCK 0x44U                 /*!< ERSALL*/
#define FTFx_SECURITY_BY_PASS 0x45U                /*!< VFYKEY*/
#define FTFx_SWAP_CONTROL 0x46U                    /*!< SWAP*/
#define FTFx_ERASE_ALL_BLOCK_UNSECURE 0x49U        /*!< ERSALLU*/
#define FTFx_VERIFY_ALL_EXECUTE_ONLY_SEGMENT 0x4AU /*!< RD1XA*/
#define FTFx_ERASE_ALL_EXECUTE_ONLY_SEGMENT 0x4BU  /*!< ERSXA*/
#define FTFx_PROGRAM_PARTITION 0x80U               /*!< PGMPART)*/
#define FTFx_SET_FLEXRAM_FUNCTION 0x81U            /*!< SETRAM*/
                                           /*@}*/

/*!
 * @name Common flash register info defines
 * @{
 */
#if defined(FTFA)
#define FTFx FTFA
#define FTFx_BASE FTFA_BASE
#define FTFx_FSTAT_CCIF_MASK FTFA_FSTAT_CCIF_MASK
#define FTFx_FSTAT_RDCOLERR_MASK FTFA_FSTAT_RDCOLERR_MASK
#define FTFx_FSTAT_ACCERR_MASK FTFA_FSTAT_ACCERR_MASK
#define FTFx_FSTAT_FPVIOL_MASK FTFA_FSTAT_FPVIOL_MASK
#define FTFx_FSTAT_MGSTAT0_MASK FTFA_FSTAT_MGSTAT0_MASK
#define FTFx_FSEC_SEC_MASK FTFA_FSEC_SEC_MASK
#define FTFx_FSEC_KEYEN_MASK FTFA_FSEC_KEYEN_MASK
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_RAM) && FSL_FEATURE_FLASH_HAS_FLEX_RAM
#define FTFx_FCNFG_RAMRDY_MASK FTFA_FCNFG_RAMRDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_RAM */
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_NVM) && FSL_FEATURE_FLASH_HAS_FLEX_NVM
#define FTFx_FCNFG_EEERDY_MASK FTFA_FCNFG_EEERDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_NVM */
#elif defined(FTFE)
#define FTFx FTFE
#define FTFx_BASE FTFE_BASE
#define FTFx_FSTAT_CCIF_MASK FTFE_FSTAT_CCIF_MASK
#define FTFx_FSTAT_RDCOLERR_MASK FTFE_FSTAT_RDCOLERR_MASK
#define FTFx_FSTAT_ACCERR_MASK FTFE_FSTAT_ACCERR_MASK
#define FTFx_FSTAT_FPVIOL_MASK FTFE_FSTAT_FPVIOL_MASK
#define FTFx_FSTAT_MGSTAT0_MASK FTFE_FSTAT_MGSTAT0_MASK
#define FTFx_FSEC_SEC_MASK FTFE_FSEC_SEC_MASK
```

```
#define FTFx_FSEC_KEYEN_MASK FTFE_FSEC_KEYEN_MASK
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_RAM) && FSL_FEATURE_FLASH_HAS_FLEX_RAM
#define FTFx_FCNFG_RAMRDY_MASK FTFE_FCNFG_RAMRDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_RAM */
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_NVM) && FSL_FEATURE_FLASH_HAS_FLEX_NVM
#define FTFx_FCNFG_EEERDY_MASK FTFE_FCNFG_EEERDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_NVM */
#elif defined(FTFL)
#define FTFx FTFL
#define FTFx_BASE FTFL_BASE
#define FTFx_FSTAT_CCIF_MASK FTFL_FSTAT_CCIF_MASK
#define FTFx_FSTAT_RDCOLERR_MASK FTFL_FSTAT_RDCOLERR_MASK
#define FTFx_FSTAT_ACCERR_MASK FTFL_FSTAT_ACCERR_MASK
#define FTFx_FSTAT_FPVIOL_MASK FTFL_FSTAT_FPVIOL_MASK
#define FTFx_FSTAT_MGSTAT0_MASK FTFL_FSTAT_MGSTAT0_MASK
#define FTFx_FSEC_SEC_MASK FTFL_FSEC_SEC_MASK
#define FTFx_FSEC_KEYEN_MASK FTFL_FSEC_KEYEN_MASK
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_RAM) && FSL_FEATURE_FLASH_HAS_FLEX_RAM
#define FTFx_FCNFG_RAMRDY_MASK FTFL_FCNFG_RAMRDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_RAM */
#if defined(FSL_FEATURE_FLASH_HAS_FLEX_NVM) && FSL_FEATURE_FLASH_HAS_FLEX_NVM
#define FTFx_FCNFG_EEERDY_MASK FTFL_FCNFG_EEERDY_MASK
#endif /* FSL_FEATURE_FLASH_HAS_FLEX_NVM */
#else
#error "Unknown flash controller"
#endif
/*@}*/

/*!
 * @name Common flash register access info defines
 * @{
 */
#define FTFx_FCCOB3_REG (FTFx->FCCOB3)
#define FTFx_FCCOB5_REG (FTFx->FCCOB5)
#define FTFx_FCCOB6_REG (FTFx->FCCOB6)
#define FTFx_FCCOB7_REG (FTFx->FCCOB7)

#if defined(FTFA_FPROTH0_PROT_MASK) || defined(FTFE_FPROTH0_PROT_MASK) || defined(FTFL_
#define FTFx_FPROT_HIGH_REG (FTFx->FPROTH3)
#define FTFx_FPROTH3_REG (FTFx->FPROTH3)
#define FTFx_FPROTH2_REG (FTFx->FPROTH2)
#define FTFx_FPROTH1_REG (FTFx->FPROTH1)
#define FTFx_FPROTH0_REG (FTFx->FPROTH0)
#endif

#if defined(FTFA_FPROTL0_PROT_MASK) || defined(FTFE_FPROTL0_PROT_MASK) || defined(FTFL_F
#define FTFx_FPROT_LOW_REG (FTFx->FPROTL3)
#define FTFx_FPROTL3_REG (FTFx->FPROTL3)
#define FTFx_FPROTL2_REG (FTFx->FPROTL2)
#define FTFx_FPROTL1_REG (FTFx->FPROTL1)
#define FTFx_FPROTL0_REG (FTFx->FPROTL0)
#elif defined(FTFA_FPROT0_PROT_MASK) || defined(FTFE_FPROT0_PROT_MASK) || defined(FTFL_FF
#define FTFx_FPROT_LOW_REG (FTFx->FPROT3)
#define FTFx_FPROTL3_REG (FTFx->FPROT3)
#define FTFx_FPROTL2_REG (FTFx->FPROT2)
#define FTFx_FPROTL1_REG (FTFx->FPROT1)
#define FTFx_FPROTL0_REG (FTFx->FPROT0)
#endif

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
#define FTFx_FPROTSH_REG (FTFx->FPROTSH)
#define FTFx_FPROTSL_REG (FTFx->FPROTSL)
#endif

#define FTFx_XACCH3_REG (FTFx->XACCH3)
#define FTFx_XACCL3_REG (FTFx->XACCL3)

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
#define FTFx_XACCSH_REG (FTFx->XACCSH)
#define FTFx_XACCSL_REG (FTFx->XACCSL)
#endif
/*@}*/

/*!
 * @brief Enumeration for access segment property.
 */
enum _flash_access_segment_property
{
    kFLASH_AccessSegmentBase = 256UL,
};

/*!
 * @brief Enumeration for flash config area.
 */
enum _flash_config_area_range
{
    kFLASH_ConfigAreaStart = 0x400U,
    kFLASH_ConfigAreaEnd = 0x40FU
};

/*!
 * @name Flash register access type defines
 * @{
 */
#define FTFx_REG8_ACCESS_TYPE volatile uint8_t *
#define FTFx_REG32_ACCESS_TYPE volatile uint32_t *
/*@}*/

/*!
 * @brief MCM cache register access info defines.
 */
#if defined(MCM_PLACR_CFCC_MASK)
#define MCM_CACHE_CLEAR_MASK MCM_PLACR_CFCC_MASK
#define MCM_CACHE_CLEAR_SHIFT MCM_PLACR_CFCC_SHIFT
#if defined(MCM)
#define MCM0_CACHE_REG MCM->PLACR
```

```
#elif defined(MCM0)
#define MCM0_CACHE_REG MCM0->PLACR
#endif
#if defined(MCM1)
#define MCM1_CACHE_REG MCM1->PLACR
#endif
#elif defined(MCM_CPCR2_CCBC_MASK)
#define MCM_CACHE_CLEAR_MASK MCM_CPCR2_CCBC_MASK
#define MCM_CACHE_CLEAR_SHIFT MCM_CPCR2_CCBC_SHIFT
#if defined(MCM)
#define MCM0_CACHE_REG MCM->CPCR2
#elif defined(MCM0)
#define MCM0_CACHE_REG MCM0->CPCR2
#endif
#if defined(MCM1)
#define MCM1_CACHE_REG MCM1->CPCR2
#endif
#endif

/*!
 * @brief MSCM cache register access info defines.
 */
#if defined(MSCM_OCMDR_OCM1_MASK)
#define MSCM_SPECULATION_DISABLE_MASK MSCM_OCMDR_OCM1_MASK
#define MSCM_SPECULATION_DISABLE_SHIFT MSCM_OCMDR_OCM1_SHIFT
#define MSCM_SPECULATION_DISABLE(x) MSCM_OCMDR_OCM1(x)
#elif defined(MSCM_OCMDR_OCMC1_MASK)
#define MSCM_SPECULATION_DISABLE_MASK MSCM_OCMDR_OCMC1_MASK
#define MSCM_SPECULATION_DISABLE_SHIFT MSCM_OCMDR_OCMC1_SHIFT
#define MSCM_SPECULATION_DISABLE(x) MSCM_OCMDR_OCMC1(x)
#endif

/*!
 * @brief MSCM prefetch speculation defines.
 */
#define MSCM_OCMDR_OCMC1_DFDS_MASK (0x10U)
#define MSCM_OCMDR_OCMC1_DFCS_MASK (0x20U)

#define MSCM_OCMDR_OCMC1_DFDS_SHIFT (4U)
#define MSCM_OCMDR_OCMC1_DFCS_SHIFT (5U)

/*!
 * @brief Flash size encoding rule.
 */
#define FLASH_MEMORY_SIZE_ENCODING_RULE_K1_2 (0x00U)
#define FLASH_MEMORY_SIZE_ENCODING_RULE_K3 (0x01U)

#if defined(K32W042S1M2_M0P_SERIES) || defined(K32W042S1M2_M4_SERIES)
#define FLASH_MEMORY_SIZE_ENCODING_RULE (FLASH_MEMORY_SIZE_ENCODING_RULE_K3)
#else
#define FLASH_MEMORY_SIZE_ENCODING_RULE (FLASH_MEMORY_SIZE_ENCODING_RULE_K1_
#endif
```

```
/*******************************************************************************
 * Prototypes
 ******************************************************************************/

#if FLASH_DRIVER_IS_FLASH_RESIDENT
/*! @brief Copy flash_run_command() to RAM*/
static void copy_flash_run_command(uint32_t *flashRunCommand);
/*! @brief Copy flash_cache_clear_command() to RAM*/
static void copy_flash_common_bit_operation(uint32_t *flashCommonBitOperation);
/*! @brief Check whether flash execute-in-ram functions are ready*/
static status_t flash_check_execute_in_ram_function_info(flash_config_t *config);
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

/*! @brief Internal function Flash command sequence. Called by driver APIs only*/
static status_t flash_command_sequence(flash_config_t *config);

/*! @brief Perform the cache clear to the flash*/
void flash_cache_clear(flash_config_t *config);

/*! @brief Process the cache to the flash*/
static void flash_cache_clear_process(flash_config_t *config, flash_cache_clear_process_t process);

/*! @brief Validates the range and alignment of the given address range.*/
static status_t flash_check_range(flash_config_t *config,
                                  uint32_t startAddress,
                                  uint32_t lengthInBytes,
                                  uint32_t alignmentBaseline);
/*! @brief Gets the right address, sector and block size of current flash type which is indicated by address.*
static status_t flash_get_matched_operation_info(flash_config_t *config,
                                                 uint32_t address,
                                                 flash_operation_config_t *info);
/*! @brief Validates the given user key for flash erase APIs.*/
static status_t flash_check_user_key(uint32_t key);

#if FLASH_SSD_IS_FLEXNVM_ENABLED
/*! @brief Updates FlexNVM memory partition status according to data flash 0 IFR.*/
static status_t flash_update_flexnvm_memory_partition_status(flash_config_t *config);
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
/*! @brief Validates the range of the given resource address.*/
static status_t flash_check_resource_range(uint32_t start,
                                           uint32_t lengthInBytes,
                                           uint32_t alignmentBaseline,
                                           flash_read_resource_option_t option);
#endif /* FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD) && FSL_FEATURE_FLASH_HAS_S
/*! @brief Validates the gived swap control option.*/
static status_t flash_check_swap_control_option(flash_swap_control_option_t option);
#endif /* FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP) && FSL_FEATURE_FLASH_HAS_P
```

```
/*! @brief Validates the gived address to see if it is equal to swap indicator address in pflash swap IFR.*/
static status_t flash_validate_swap_indicator_address(flash_config_t *config, uint32_t address);
#endif /* FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP */

#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
/*! @brief Validates the gived flexram function option.*/
static inline status_t flasn_check_flexram_function_option_range(flash_flexram_function_option_t option);
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

/*! @brief Gets the flash protection information (region size, region count).*/
static status_t flash_get_protection_info(flash_config_t *config, flash_protection_config_t *info);

#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL) && FSL_FEATURE_FLASH_HAS_ACC
/*! @brief Gets the flash Execute-Only access information (Segment size, Segment count).*/
static status_t flash_get_access_info(flash_config_t *config, flash_access_config_t *info);
#endif /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */

#if FLASH_CACHE_IS_CONTROLLED_BY_MCM
/*! @brief Performs the cache clear to the flash by MCM.*/
void mcm_flash_cache_clear(flash_config_t *config);
#endif /* FLASH_CACHE_IS_CONTROLLED_BY_MCM */

#if FLASH_CACHE_IS_CONTROLLED_BY_FMC
/*! @brief Performs the cache clear to the flash by FMC.*/
void fmc_flash_cache_clear(void);
#endif /* FLASH_CACHE_IS_CONTROLLED_BY_FMC */

#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
/*! @brief Sets the prefetch speculation buffer to the flash by MSCM.*/
void mscm_flash_prefetch_speculation_enable(bool enable);
#endif /* FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM */

#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC
/*! @brief Performs the prefetch speculation buffer clear to the flash by FMC.*/
void fmc_flash_prefetch_speculation_clear(void);
#endif /* FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC */

/*******************************************************************************
 * Variables
 ******************************************************************************/

/*! @brief Access to FTFx->FCCOB */
volatile uint32_t *const kFCCOBx = (volatile uint32_t *)&FTFx_FCCOB3_REG;
/*! @brief Access to FTFx->FPROT */
volatile uint32_t *const kFPROTL = (volatile uint32_t *)&FTFx_FPROT_LOW_REG;
#if defined(FTFx_FPROT_HIGH_REG)
volatile uint32_t *const kFPROTH = (volatile uint32_t *)&FTFx_FPROT_HIGH_REG;
#endif

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
volatile uint8_t *const kFPROTSL = (volatile uint8_t *)&FTFx_FPROTSL_REG;
volatile uint8_t *const kFPROTSH = (volatile uint8_t *)&FTFx_FPROTSH_REG;
#endif
```

```
#if FLASH_DRIVER_IS_FLASH_RESIDENT
/*! @brief A function pointer used to point to relocated flash_run_command() */
static void (*callFlashRunCommand)(FTFx_REG8_ACCESS_TYPE ftfx_fstat);
/*! @brief A function pointer used to point to relocated flash_common_bit_operation() */
static void (*callFlashCommonBitOperation)(FTFx_REG32_ACCESS_TYPE base,
                                           uint32_t bitMask,
                                           uint32_t bitShift,
                                           uint32_t bitValue);

/*!
 * @brief Position independent code of flash_run_command()
 *
 * Note1: The prototype of C function is shown as below:
 * @code
 *   void flash_run_command(FTFx_REG8_ACCESS_TYPE ftfx_fstat)
 *   {
 *       // clear CCIF bit
 *       *ftfx_fstat = FTFx_FSTAT_CCIF_MASK;
 *
 *       // Check CCIF bit of the flash status register, wait till it is set.
 *       // IP team indicates that this loop will always complete.
 *       while (!((*ftfx_fstat) & FTFx_FSTAT_CCIF_MASK))
 *       {
 *       }
 *   }
 * @endcode
 * Note2: The binary code is generated by IAR 7.70.1
 */
const static uint16_t s_flashRunCommandFunctionCode[] = {
    0x2180, /* MOVS  R1, #128 ; 0x80 */
    0x7001, /* STRB  R1, [R0] */
    /* @4: */
    0x7802, /* LDRB  R2, [R0] */
    0x420a, /* TST   R2, R1 */
    0xd0fc, /* BEQ.N @4 */
    0x4770  /* BX    LR */
};

/*!
 * @brief Position independent code of flash_common_bit_operation()
 *
 * Note1: The prototype of C function is shown as below:
 * @code
 *   void flash_common_bit_operation(FTFx_REG32_ACCESS_TYPE base, uint32_t bitMask, uint32_t bitS
 * bitValue)
 *   {
 *       if (bitMask)
 *       {
 *           uint32_t value = (((uint32_t)(((uint32_t)(bitValue)) << bitShift)) & bitMask);
 *           *base = (*base & (~bitMask)) | value;
 *       }
 *
 *      __ISB();
 *      __DSB();
 *   }
 * @endcode
 * Note2: The binary code is generated by IAR 7.70.1
 */
const static uint16_t s_flashCommonBitOperationFunctionCode[] = {
    0xb510, /* PUSH  {R4, LR} */
    0x2900, /* CMP   R1, #0 */
    0xd005, /* BEQ.N @12 */
    0x6804, /* LDR   R4, [R0] */
    0x438c, /* BICS  R4, R4, R1 */
    0x4093, /* LSLS  R3, R3, R2 */
    0x4019, /* ANDS  R1, R1, R3 */
    0x4321, /* ORRS  R1, R1, R4 */
    0x6001, /* STR   R1, [R0] */
    /*  @12: */
    0xf3bf, 0x8f6f, /* ISB */
    0xf3bf, 0x8f4f, /* DSB */
    0xbd10          /* POP   {R4, PC} */
};
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

#if (FLASH_DRIVER_IS_FLASH_RESIDENT && !FLASH_DRIVER_IS_EXPORTED)
/*! @brief A static buffer used to hold flash_run_command() */
static uint32_t s_flashRunCommand[kFLASH_ExecuteInRamFunctionMaxSizeInWords];
/*! @brief A static buffer used to hold flash_common_bit_operation() */
static uint32_t s_flashCommonBitOperation[kFLASH_ExecuteInRamFunctionMaxSizeInWords];
/*! @brief Flash execute-in-ram function information */
static flash_execute_in_ram_function_config_t s_flashExecuteInRamFunctionInfo;
#endif

/*!
 * @brief Table of pflash sizes.
 *
 *  The index into this table is the value of the SIM_FCFG1.PFSIZE bitfield.
 *
 *  The values in this table have been right shifted 10 bits so that they will all fit within
 *  an 16-bit integer. To get the actual flash density, you must left shift the looked up value
 *  by 10 bits.
 *
 *  Elements of this table have a value of 0 in cases where the PFSIZE bitfield value is
 *  reserved.
 *
 *  Code to use the table:
 *  @code
 *      uint8_t pfsize = (SIM->FCFG1 & SIM_FCFG1_PFSIZE_MASK) >> SIM_FCFG1_PFSIZE_SHIFT;
 *      flashDensity = ((uint32_t)kPFlashDensities[pfsize]) << 10;
 *  @endcode
 */
#if (FLASH_MEMORY_SIZE_ENCODING_RULE == FLASH_MEMORY_SIZE_ENCODING_RULE_K1_2)
const uint16_t kPFlashDensities[] = {
    8,    /* 0x0 - 8192, 8KB */
    16,   /* 0x1 - 16384, 16KB */
    24,   /* 0x2 - 24576, 24KB */
    32,   /* 0x3 - 32768, 32KB */
    48,   /* 0x4 - 49152, 48KB */
    64,   /* 0x5 - 65536, 64KB */
    96,   /* 0x6 - 98304, 96KB */
    128,  /* 0x7 - 131072, 128KB */
    192,  /* 0x8 - 196608, 192KB */
    256,  /* 0x9 - 262144, 256KB */
    384,  /* 0xa - 393216, 384KB */
    512,  /* 0xb - 524288, 512KB */
    768,  /* 0xc - 786432, 768KB */
    1024, /* 0xd - 1048576, 1MB */
    1536, /* 0xe - 1572864, 1.5MB */
    /* 2048,  0xf - 2097152, 2MB */
};
#elif(FLASH_MEMORY_SIZE_ENCODING_RULE == FLASH_MEMORY_SIZE_ENCODING_RULE_K3)
const uint16_t kPFlashDensities[] = {
    0,    /* 0x0 - undefined */
    0,    /* 0x1 - undefined */
    0,    /* 0x2 - undefined */
    0,    /* 0x3 - undefined */
    0,    /* 0x4 - undefined */
    0,    /* 0x5 - undefined */
    0,    /* 0x6 - undefined */
    0,    /* 0x7 - undefined */
    0,    /* 0x8 - undefined */
    0,    /* 0x9 - undefined */
    256,  /* 0xa - 262144, 256KB */
    0,    /* 0xb - undefined */
    1024, /* 0xc - 1048576, 1MB */
    0,    /* 0xd - undefined */
    0,    /* 0xe - undefined */
    0,    /* 0xf - undefined */
};
#endif

/*******************************************************************************
 * Code
 ******************************************************************************/

status_t FLASH_Init(flash_config_t *config)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED
    if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
    {
/* calculate the flash density from SIM_FCFG1.PFSIZE */
#if defined(SIM_FCFG1_CORE1_PFSIZE_MASK)
```

```
        uint32_t flashDensity;
        uint8_t pfsize = (SIM->FCFG1 & SIM_FCFG1_CORE1_PFSIZE_MASK) >> SIM_FCFG1_CORE1_PF
        if (pfsize == 0xf)
        {
            flashDensity = FSL_FEATURE_FLASH_PFLASH_1_BLOCK_COUNT * FSL_FEATURE_FLASH_F
        }
        else
        {
            flashDensity = ((uint32_t)kPFlashDensities[pfsize]) << 10;
        }
        config->PFlashTotalSize = flashDensity;
#else
        /* Unused code to solve MISRA-C issue*/
        config->PFlashBlockBase = kPFlashDensities[0];
        config->PFlashTotalSize = FSL_FEATURE_FLASH_PFLASH_1_BLOCK_COUNT * FSL_FEATURE_
#endif
        config->PFlashBlockBase = FSL_FEATURE_FLASH_PFLASH_1_START_ADDRESS;
        config->PFlashBlockCount = FSL_FEATURE_FLASH_PFLASH_1_BLOCK_COUNT;
        config->PFlashSectorSize = FSL_FEATURE_FLASH_PFLASH_1_BLOCK_SECTOR_SIZE;
    }
    else
#endif /* FLASH_SSD_IS_SECONDARY_FLASH_ENABLED */
    {
        uint32_t flashDensity;

/* calculate the flash density from SIM_FCFG1.PFSIZE */
#if defined(SIM_FCFG1_CORE0_PFSIZE_MASK)
        uint8_t pfsize = (SIM->FCFG1 & SIM_FCFG1_CORE0_PFSIZE_MASK) >> SIM_FCFG1_CORE0_PF
#elif defined(SIM_FCFG1_PFSIZE_MASK)
        uint8_t pfsize = (SIM->FCFG1 & SIM_FCFG1_PFSIZE_MASK) >> SIM_FCFG1_PFSIZE_SHIFT;
#else
#error "Unknown flash size"
#endif
        /* PFSIZE=0xf means that on customer parts the IFR was not correctly programmed.
         * We just use the pre-defined flash size in feature file here to support pre-production parts */
        if (pfsize == 0xf)
        {
            flashDensity = FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT * FSL_FEATURE_FLASH_PFL
        }
        else
        {
            flashDensity = ((uint32_t)kPFlashDensities[pfsize]) << 10;
        }

        /* fill out a few of the structure members */
        config->PFlashBlockBase = FSL_FEATURE_FLASH_PFLASH_START_ADDRESS;
        config->PFlashTotalSize = flashDensity;
        config->PFlashBlockCount = FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT;
        config->PFlashSectorSize = FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE;
    }

    {
#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL) && FSL_FEATURE_FLASH_HAS_ACCE
```

```
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
        if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
        {
            config->PFlashAccessSegmentSize = kFLASH_AccessSegmentBase << FTFx->FACSSS;
            config->PFlashAccessSegmentCount = FTFx->FACSNS;
        }
        else
#endif
        {
            config->PFlashAccessSegmentSize = kFLASH_AccessSegmentBase << FTFx->FACSS;
            config->PFlashAccessSegmentCount = FTFx->FACSN;
        }
#else
        config->PFlashAccessSegmentSize = 0;
        config->PFlashAccessSegmentCount = 0;
#endif /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */
    }

    config->PFlashCallback = NULL;

/* copy required flash commands to RAM */
#if (FLASH_DRIVER_IS_FLASH_RESIDENT && !FLASH_DRIVER_IS_EXPORTED)
    if (kStatus_FLASH_Success != flash_check_execute_in_ram_function_info(config))
    {
        s_flashExecuteInRamFunctionInfo.activeFunctionCount = 0;
        s_flashExecuteInRamFunctionInfo.flashRunCommand = s_flashRunCommand;
        s_flashExecuteInRamFunctionInfo.flashCommonBitOperation = s_flashCommonBitOperation;
        config->flashExecuteInRamFunctionInfo = &s_flashExecuteInRamFunctionInfo.activeFunctionCount;
        FLASH_PrepareExecuteInRamFunctions(config);
    }
#endif

    config->FlexRAMBlockBase = FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS;
    config->FlexRAMTotalSize = FSL_FEATURE_FLASH_FLEX_RAM_SIZE;

#if FLASH_SSD_IS_FLEXNVM_ENABLED
    {
        status_t returnCode;
        config->DFlashBlockBase = FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS;
        returnCode = flash_update_flexnvm_memory_partition_status(config);
        if (returnCode != kStatus_FLASH_Success)
        {
            return returnCode;
        }
    }
#endif

    return kStatus_FLASH_Success;
}

status_t FLASH_SetCallback(flash_config_t *config, flash_callback_t callback)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    config->PFlashCallback = callback;

    return kStatus_FLASH_Success;
}

#if FLASH_DRIVER_IS_FLASH_RESIDENT
status_t FLASH_PrepareExecuteInRamFunctions(flash_config_t *config)
{
    flash_execute_in_ram_function_config_t *flashExecuteInRamFunctionInfo;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    flashExecuteInRamFunctionInfo = (flash_execute_in_ram_function_config_t *)config->flashExecuteInRa

    copy_flash_run_command(flashExecuteInRamFunctionInfo->flashRunCommand);
    copy_flash_common_bit_operation(flashExecuteInRamFunctionInfo->flashCommonBitOperation);
    flashExecuteInRamFunctionInfo->activeFunctionCount = kFLASH_ExecuteInRamFunctionTotalNum;

    return kStatus_FLASH_Success;
}
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

status_t FLASH_EraseAll(flash_config_t *config, uint32_t key)
{
    status_t returnCode;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* preparing passing parameter to erase all flash blocks */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_ERASE_ALL_BLOCK, 0xFFFFFFU);

    /* Validate the user key */
    returnCode = flash_check_user_key(key);
    if (returnCode)
    {
        return returnCode;
    }

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);
    flash_cache_clear(config);

#if FLASH_SSD_IS_FLEXNVM_ENABLED
    /* Data flash IFR will be erased by erase all command, so we need to
     *  update FlexNVM memory partition status synchronously */
    if (returnCode == kStatus_FLASH_Success)
    {
        returnCode = flash_update_flexnvm_memory_partition_status(config);
    }
#endif

    return returnCode;
}

status_t FLASH_Erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key)
{
    uint32_t sectorSize;
    flash_operation_config_t flashOperationInfo;
    uint32_t endAddress;      /* storing end address */
    uint32_t numberOfSectors; /* number of sectors calculated by endAddress */
    status_t returnCode;

    flash_get_matched_operation_info(config, start, &flashOperationInfo);

    /* Check the supplied address range. */
    returnCode = flash_check_range(config, start, lengthInBytes, flashOperationInfo.sectorCmdAddressAlig
    if (returnCode)
    {
        return returnCode;
    }

    /* Validate the user key */
    returnCode = flash_check_user_key(key);
    if (returnCode)
    {
        return returnCode;
    }

    start = flashOperationInfo.convertedAddress;
    sectorSize = flashOperationInfo.activeSectorSize;

    /* calculating Flash end address */
    endAddress = start + lengthInBytes - 1;

    /* re-calculate the endAddress and align it to the start of the next sector
     * which will be used in the comparison below */
    if (endAddress % sectorSize)
    {
        numberOfSectors = endAddress / sectorSize + 1;
        endAddress = numberOfSectors * sectorSize - 1;
    }

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);
    /* the start address will increment to the next sector address
     * until it reaches the endAdddress */
    while (start <= endAddress)
    {
        /* preparing passing parameter to erase a flash block */
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_ERASE_SECTOR, start);

        /* calling flash command sequence function to execute the command */
        returnCode = flash_command_sequence(config);

        /* calling flash callback function if it is available */
        if (config->PFlashCallback)
        {
            config->PFlashCallback();
        }

        /* checking the success of command execution */
        if (kStatus_FLASH_Success != returnCode)
        {
            break;
        }
        else
        {
            /* Increment to the next sector */
            start += sectorSize;
        }
    }

    flash_cache_clear(config);

    return (returnCode);
}

#if defined(FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD) && FSL_FEATUR
status_t FLASH_EraseAllUnsecure(flash_config_t *config, uint32_t key)
{
    status_t returnCode;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Prepare passing parameter to erase all flash blocks (unsecure). */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_ERASE_ALL_BLOCK_UNSECURE, 0xFFFFFFU

    /* Validate the user key */
    returnCode = flash_check_user_key(key);
    if (returnCode)
    {
        return returnCode;
    }
    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    flash_cache_clear(config);

#if FLASH_SSD_IS_FLEXNVM_ENABLED
    /* Data flash IFR will be erased by erase all unsecure command, so we need to
     *  update FlexNVM memory partition status synchronously */
    if (returnCode == kStatus_FLASH_Success)
    {
        returnCode = flash_update_flexnvm_memory_partition_status(config);
    }
#endif

    return returnCode;
}
#endif /* FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD */

status_t FLASH_EraseAllExecuteOnlySegments(flash_config_t *config, uint32_t key)
{
    status_t returnCode;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* preparing passing parameter to erase all execute-only segments
     * 1st element for the FCCOB register */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_ERASE_ALL_EXECUTE_ONLY_SEGMENT, 0xF

    /* Validate the user key */
    returnCode = flash_check_user_key(key);
    if (returnCode)
    {
        return returnCode;
    }

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    flash_cache_clear(config);

    return returnCode;
}

status_t FLASH_Program(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t lengthInBytes)
{
```

```
status_t returnCode;
flash_operation_config_t flashOperationInfo;

if (src == NULL)
{
    return kStatus_FLASH_InvalidArgument;
}

flash_get_matched_operation_info(config, start, &flashOperationInfo);

/* Check the supplied address range. */
returnCode = flash_check_range(config, start, lengthInBytes, flashOperationInfo.blockWriteUnitSize);
if (returnCode)
{
    return returnCode;
}

start = flashOperationInfo.convertedAddress;

flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

while (lengthInBytes > 0)
{
    /* preparing passing parameter to program the flash block */
    kFCCOBx[1] = *src++;
    if (4 == flashOperationInfo.blockWriteUnitSize)
    {
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_PROGRAM_LONGWORD, start);
    }
    else if (8 == flashOperationInfo.blockWriteUnitSize)
    {
        kFCCOBx[2] = *src++;
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_PROGRAM_PHRASE, start);
    }
    else
    {
    }

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    /* calling flash callback function if it is available */
    if (config->PFlashCallback)
    {
        config->PFlashCallback();
    }

    /* checking for the success of command execution */
    if (kStatus_FLASH_Success != returnCode)
    {
        break;
    }
    else
```

```
        {
            /* update start address for next iteration */
            start += flashOperationInfo.blockWriteUnitSize;

            /* update lengthInBytes for next iteration */
            lengthInBytes -= flashOperationInfo.blockWriteUnitSize;
        }
    }

    flash_cache_clear(config);

    return (returnCode);
}

status_t FLASH_ProgramOnce(flash_config_t *config, uint32_t index, uint32_t *src, uint32_t lengthInBytes
{
    status_t returnCode;

    if ((config == NULL) || (src == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* pass paramters to FTFx */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_1_2(FTFx_PROGRAM_ONCE, index, 0xFFFFU);

    kFCCOBx[1] = *src;

/* Note: Have to seperate the first index from the rest if it equals 0
 * to avoid a pointless comparison of unsigned int to 0 compiler warning */
#if FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT
#if FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT
    if (((index == FLASH_PROGRAM_ONCE_MIN_ID_8BYTES) ||
         /* Range check */
         ((index >= FLASH_PROGRAM_ONCE_MIN_ID_8BYTES + 1) && (index <= FLASH_PROGRAM_ON
        (lengthInBytes == 8))
#endif /* FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT */
    {
        kFCCOBx[2] = *(src + 1);
    }
#endif /* FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT */

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    flash_cache_clear(config);

    return returnCode;
}

#if defined(FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD) && FSL_FEATURE_FLASH_HA
```

```
status_t FLASH_ProgramSection(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t lengthInByte
{
    status_t returnCode;
    uint32_t sectorSize;
    flash_operation_config_t flashOperationInfo;
#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
    bool needSwitchFlexRamMode = false;
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

    if (src == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    flash_get_matched_operation_info(config, start, &flashOperationInfo);

    /* Check the supplied address range. */
    returnCode = flash_check_range(config, start, lengthInBytes, flashOperationInfo.sectionCmdAddressAli
    if (returnCode)
    {
        return returnCode;
    }

    start = flashOperationInfo.convertedAddress;
    sectorSize = flashOperationInfo.activeSectorSize;

#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
    /* Switch function of FlexRAM if needed */
    if (!(FTFx->FCNFG & FTFx_FCNFG_RAMRDY_MASK))
    {
        needSwitchFlexRamMode = true;

        returnCode = FLASH_SetFlexramFunction(config, kFLASH_FlexramFunctionOptionAvailableAsRam)
        if (returnCode != kStatus_FLASH_Success)
        {
            return kStatus_FLASH_SetFlexramAsRamError;
        }
    }
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    while (lengthInBytes > 0)
    {
        /* Make sure the write operation doesn't span two sectors */
        uint32_t endAddressOfCurrentSector = ALIGN_UP(start, sectorSize);
        uint32_t lengthTobeProgrammedOfCurrentSector;
        uint32_t currentOffset = 0;

        if (endAddressOfCurrentSector == start)
        {
            endAddressOfCurrentSector += sectorSize;
        }
```

```
if (lengthInBytes + start > endAddressOfCurrentSector)
{
    lengthTobeProgrammedOfCurrentSector = endAddressOfCurrentSector - start;
}
else
{
    lengthTobeProgrammedOfCurrentSector = lengthInBytes;
}

/* Program Current Sector */
while (lengthTobeProgrammedOfCurrentSector > 0)
{
    /* Make sure the program size doesn't exceeds Acceleration RAM size */
    uint32_t programSizeOfCurrentPass;
    uint32_t numberOfPhases;

    if (lengthTobeProgrammedOfCurrentSector > kFLASH_AccelerationRamSize)
    {
        programSizeOfCurrentPass = kFLASH_AccelerationRamSize;
    }
    else
    {
        programSizeOfCurrentPass = lengthTobeProgrammedOfCurrentSector;
    }

    /* Copy data to FlexRAM */
    memcpy((void *)FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS, src + currentOffset / 4,
    /* Set start address of the data to be programmed */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_PROGRAM_SECTION, start + currentOffset
    /* Set program size in terms of FEATURE_FLASH_SECTION_CMD_ADDRESS_ALIGMENT */
    numberOfPhases = programSizeOfCurrentPass / flashOperationInfo.sectionCmdAddressAligment;

    kFCCOBx[1] = BYTES_JOIN_TO_WORD_2_2(numberOfPhases, 0xFFFFU);

    /* Peform command sequence */
    returnCode = flash_command_sequence(config);

    /* calling flash callback function if it is available */
    if (config->PFlashCallback)
    {
        config->PFlashCallback();
    }

    if (returnCode != kStatus_FLASH_Success)
    {
        flash_cache_clear(config);
        return returnCode;
    }

    lengthTobeProgrammedOfCurrentSector -= programSizeOfCurrentPass;
    currentOffset += programSizeOfCurrentPass;
}
```

```
        src += currentOffset / 4;
        start += currentOffset;
        lengthInBytes -= currentOffset;
    }

    flash_cache_clear(config);

#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
    /* Restore function of FlexRAM if needed. */
    if (needSwitchFlexRamMode)
    {
        returnCode = FLASH_SetFlexramFunction(config, kFLASH_FlexramFunctionOptionAvailableForEepr
        if (returnCode != kStatus_FLASH_Success)
        {
            return kStatus_FLASH_RecoverFlexramAsEepromError;
        }
    }
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

    return returnCode;
}
#endif /* FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD */

#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromWrite(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t lengthInBytes)
{
    status_t returnCode;
    bool needSwitchFlexRamMode = false;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Validates the range of the given address */
    if ((start < config->FlexRAMBlockBase) ||
        ((start + lengthInBytes) > (config->FlexRAMBlockBase + config->EEpromTotalSize)))
    {
        return kStatus_FLASH_AddressError;
    }

    returnCode = kStatus_FLASH_Success;

    /* Switch function of FlexRAM if needed */
    if (!(FTFx->FCNFG & FTFx_FCNFG_EEERDY_MASK))
    {
        needSwitchFlexRamMode = true;

        returnCode = FLASH_SetFlexramFunction(config, kFLASH_FlexramFunctionOptionAvailableForEepr
        if (returnCode != kStatus_FLASH_Success)
        {
            return kStatus_FLASH_SetFlexramAsEepromError;
        }
    }

    /* Write data to FlexRAM when it is used as EEPROM emulator */
    while (lengthInBytes > 0)
    {
        if ((!(start & 0x3U)) && (lengthInBytes >= 4))
        {
            *(uint32_t *)start = *(uint32_t *)src;
            start += 4;
            src += 4;
            lengthInBytes -= 4;
        }
        else if ((!(start & 0x1U)) && (lengthInBytes >= 2))
        {
            *(uint16_t *)start = *(uint16_t *)src;
            start += 2;
            src += 2;
            lengthInBytes -= 2;
        }
        else
        {
            *(uint8_t *)start = *src;
            start += 1;
            src += 1;
            lengthInBytes -= 1;
        }
        /* Wait till EEERDY bit is set */
        while (!(FTFx->FCNFG & FTFx_FCNFG_EEERDY_MASK))
        {
        }

        /* Check for protection violation error */
        if (FTFx->FSTAT & FTFx_FSTAT_FPVIOL_MASK)
        {
            return kStatus_FLASH_ProtectionViolation;
        }
    }

    /* Switch function of FlexRAM if needed */
    if (needSwitchFlexRamMode)
    {
        returnCode = FLASH_SetFlexramFunction(config, kFLASH_FlexramFunctionOptionAvailableAsRam)
        if (returnCode != kStatus_FLASH_Success)
        {
            return kStatus_FLASH_RecoverFlexramAsRamError;
        }
    }

    return returnCode;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */
```

```
#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
status_t FLASH_ReadResource(
    flash_config_t *config, uint32_t start, uint32_t *dst, uint32_t lengthInBytes, flash_read_resource_option_
{
    status_t returnCode;
    flash_operation_config_t flashOperationInfo;

    if ((config == NULL) || (dst == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    flash_get_matched_operation_info(config, start, &flashOperationInfo);

    /* Check the supplied address range. */
    returnCode =
        flash_check_resource_range(start, lengthInBytes, flashOperationInfo.resourceCmdAddressAligment,
    if (returnCode != kStatus_FLASH_Success)
    {
        return returnCode;
    }

    while (lengthInBytes > 0)
    {
        /* preparing passing parameter */
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_READ_RESOURCE, start);
        if (flashOperationInfo.resourceCmdAddressAligment == 4)
        {
            kFCCOBx[2] = BYTES_JOIN_TO_WORD_1_3(option, 0xFFFFFFU);
        }
        else if (flashOperationInfo.resourceCmdAddressAligment == 8)
        {
            kFCCOBx[1] = BYTES_JOIN_TO_WORD_1_3(option, 0xFFFFFFU);
        }
        else
        {
        }

        /* calling flash command sequence function to execute the command */
        returnCode = flash_command_sequence(config);

        if (kStatus_FLASH_Success != returnCode)
        {
            break;
        }

        /* fetch data */
        *dst++ = kFCCOBx[1];
        if (flashOperationInfo.resourceCmdAddressAligment == 8)
        {
            *dst++ = kFCCOBx[2];
        }
        /* update start address for next iteration */
        start += flashOperationInfo.resourceCmdAddressAligment;
        /* update lengthInBytes for next iteration */
        lengthInBytes -= flashOperationInfo.resourceCmdAddressAligment;
    }

    return (returnCode);
}
#endif /* FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD */

status_t FLASH_ReadOnce(flash_config_t *config, uint32_t index, uint32_t *dst, uint32_t lengthInBytes)
{
    status_t returnCode;

    if ((config == NULL) || (dst == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* pass paramters to FTFx */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_1_2(FTFx_READ_ONCE, index, 0xFFFFU);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    if (kStatus_FLASH_Success == returnCode)
    {
        *dst = kFCCOBx[1];
/* Note: Have to seperate the first index from the rest if it equals 0
 *       to avoid a pointless comparison of unsigned int to 0 compiler warning */
#if FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT
#if FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT
        if (((index == FLASH_PROGRAM_ONCE_MIN_ID_8BYTES) ||
             /* Range check */
             ((index >= FLASH_PROGRAM_ONCE_MIN_ID_8BYTES + 1) && (index <= FLASH_PROGRAM_
            (lengthInBytes == 8))
#endif /* FLASH_PROGRAM_ONCE_IS_4BYTES_UNIT_SUPPORT */
        {
            *(dst + 1) = kFCCOBx[2];
        }
#endif /* FLASH_PROGRAM_ONCE_IS_8BYTES_UNIT_SUPPORT */
    }

    return returnCode;
}

status_t FLASH_GetSecurityState(flash_config_t *config, flash_security_state_t *state)
{
    /* store data read from flash register */
    uint8_t registerValue;

    if ((config == NULL) || (state == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Get flash security register value */
    registerValue = FTFx->FSEC;

    /* check the status of the flash security bits in the security register */
    if (FLASH_SECURITY_STATE_UNSECURED == (registerValue & FTFx_FSEC_SEC_MASK))
    {
        /* Flash in unsecured state */
        *state = kFLASH_SecurityStateNotSecure;
    }
    else
    {
        /* Flash in secured state
         * check for backdoor key security enable bit */
        if (FLASH_SECURITY_STATE_KEYEN == (registerValue & FTFx_FSEC_KEYEN_MASK))
        {
            /* Backdoor key security enabled */
            *state = kFLASH_SecurityStateBackdoorEnabled;
        }
        else
        {
            /* Backdoor key security disabled */
            *state = kFLASH_SecurityStateBackdoorDisabled;
        }
    }

    return (kStatus_FLASH_Success);
}

status_t FLASH_SecurityBypass(flash_config_t *config, const uint8_t *backdoorKey)
{
    uint8_t registerValue; /* registerValue */
    status_t returnCode;   /* return code variable */

    if ((config == NULL) || (backdoorKey == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* set the default return code as kStatus_Success */
    returnCode = kStatus_FLASH_Success;

    /* Get flash security register value */
    registerValue = FTFx->FSEC;

    /* Check to see if flash is in secure state (any state other than 0x2)
     * If not, then skip this since flash is not secure */
    if (0x02 != (registerValue & 0x03))
    {
        /* preparing passing parameter to erase a flash block */
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_SECURITY_BY_PASS, 0xFFFFFFU);
        kFCCOBx[1] = BYTES_JOIN_TO_WORD_1_1_1_1(backdoorKey[0], backdoorKey[1], backdoorKey[2
        kFCCOBx[2] = BYTES_JOIN_TO_WORD_1_1_1_1(backdoorKey[4], backdoorKey[5], backdoorKey[6

        /* calling flash command sequence function to execute the command */
        returnCode = flash_command_sequence(config);
    }

    return (returnCode);
}

status_t FLASH_VerifyEraseAll(flash_config_t *config, flash_margin_value_t margin)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* preparing passing parameter to verify all block command */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_1_2(FTFx_VERIFY_ALL_BLOCK, margin, 0xFFFFU);

    /* calling flash command sequence function to execute the command */
    return flash_command_sequence(config);
}

status_t FLASH_VerifyErase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, flash_margin_va
{
    /* Check arguments. */
    uint32_t blockSize;
    flash_operation_config_t flashOperationInfo;
    uint32_t nextBlockStartAddress;
    uint32_t remainingBytes;
    status_t returnCode;

    flash_get_matched_operation_info(config, start, &flashOperationInfo);

    returnCode = flash_check_range(config, start, lengthInBytes, flashOperationInfo.sectionCmdAddressAli
    if (returnCode)
    {
        return returnCode;
    }

    flash_get_matched_operation_info(config, start, &flashOperationInfo);
    start = flashOperationInfo.convertedAddress;
    blockSize = flashOperationInfo.activeBlockSize;

    nextBlockStartAddress = ALIGN_UP(start, blockSize);
    if (nextBlockStartAddress == start)
    {
        nextBlockStartAddress += blockSize;
    }

    remainingBytes = lengthInBytes;

    while (remainingBytes)
    {
        uint32_t numberOfPhrases;
        uint32_t verifyLength = nextBlockStartAddress - start;
        if (verifyLength > remainingBytes)
        {
            verifyLength = remainingBytes;
        }

        numberOfPhrases = verifyLength / flashOperationInfo.sectionCmdAddressAligment;

        /* Fill in verify section command parameters. */
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_VERIFY_SECTION, start);
        kFCCOBx[1] = BYTES_JOIN_TO_WORD_2_1_1(numberOfPhrases, margin, 0xFFU);

        /* calling flash command sequence function to execute the command */
        returnCode = flash_command_sequence(config);
        if (returnCode)
        {
            return returnCode;
        }

        remainingBytes -= verifyLength;
        start += verifyLength;
        nextBlockStartAddress += blockSize;
    }

    return kStatus_FLASH_Success;
}

status_t FLASH_VerifyProgram(flash_config_t *config,
                             uint32_t start,
                             uint32_t lengthInBytes,
                             const uint32_t *expectedData,
                             flash_margin_value_t margin,
                             uint32_t *failedAddress,
                             uint32_t *failedData)
{
    status_t returnCode;
    flash_operation_config_t flashOperationInfo;

    if (expectedData == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    flash_get_matched_operation_info(config, start, &flashOperationInfo);

    returnCode = flash_check_range(config, start, lengthInBytes, flashOperationInfo.checkCmdAddressAlign
    if (returnCode)
    {
        return returnCode;
    }
    start = flashOperationInfo.convertedAddress;

    while (lengthInBytes)
    {
        /* preparing passing parameter to program check the flash block */
        kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_PROGRAM_CHECK, start);
        kFCCOBx[1] = BYTES_JOIN_TO_WORD_1_3(margin, 0xFFFFFFU);
        kFCCOBx[2] = *expectedData;

        /* calling flash command sequence function to execute the command */
        returnCode = flash_command_sequence(config);

        /* checking for the success of command execution */
        if (kStatus_FLASH_Success != returnCode)
        {
            if (failedAddress)
            {
                *failedAddress = start;
            }
            if (failedData)
            {
                *failedData = 0;
            }
            break;
        }

        lengthInBytes -= flashOperationInfo.checkCmdAddressAligment;
        expectedData += flashOperationInfo.checkCmdAddressAligment / sizeof(*expectedData);
        start += flashOperationInfo.checkCmdAddressAligment;
    }

    return (returnCode);
}

status_t FLASH_VerifyEraseAllExecuteOnlySegments(flash_config_t *config, flash_margin_value_t margin
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* preparing passing parameter to verify erase all execute-only segments command */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_1_2(FTFx_VERIFY_ALL_EXECUTE_ONLY_SEGMENT, r

    /* calling flash command sequence function to execute the command */
    return flash_command_sequence(config);
}

status_t FLASH_IsProtected(flash_config_t *config,
                           uint32_t start,
                           uint32_t lengthInBytes,
                           flash_protection_state_t *protection_state)
{
```

```
uint32_t endAddress;           /* end address for protection check */
uint32_t regionCheckedCounter; /* increments each time the flash address was checked for
                                * protection status */
uint32_t regionCounter;        /* incrementing variable used to increment through the flash
                                * protection regions */
uint32_t protectStatusCounter; /* increments each time a flash region was detected as protected */

uint8_t flashRegionProtectStatus[FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT
                                                         * status for each
                                                         * protection region */
uint32_t flashRegionAddress[FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT +
                      1];               /* array of the start addresses for each flash
                       * protection region. Note this is REGION_COUNT+1
                       * due to requiring the next start address after
                       * the end of flash for loop-check purposes below */
flash_protection_config_t flashProtectionInfo; /* flash protection information */
status_t returnCode;

if (protection_state == NULL)
{
   return kStatus_FLASH_InvalidArgument;
}

/* Check the supplied address range. */
returnCode = flash_check_range(config, start, lengthInBytes, FSL_FEATURE_FLASH_PFLASH_BLOC
if (returnCode)
{
   return returnCode;
}

/* Get necessary flash protection information. */
returnCode = flash_get_protection_info(config, &flashProtectionInfo);
if (returnCode)
{
   return returnCode;
}

/* calculating Flash end address */
endAddress = start + lengthInBytes;

/* populate the flashRegionAddress array with the start address of each flash region */
regionCounter = 0; /* make sure regionCounter is initialized to 0 first */

/* populate up to 33rd element of array, this is the next address after end of flash array */
while (regionCounter <= flashProtectionInfo.regionCount)
{
   flashRegionAddress[regionCounter] =
      flashProtectionInfo.regionBase + flashProtectionInfo.regionSize * regionCounter;
   regionCounter++;
}

/* populate flashRegionProtectStatus array with status information
 * Protection status for each region is stored in the FPROT[3:0] registers
```

```
     * Each bit represents one region of flash
     * 4 registers * 8-bits-per-register = 32-bits (32-regions)
     * The convention is:
     * FPROT3[bit 0] is the first protection region (start of flash memory)
     * FPROT0[bit 7] is the last protection region (end of flash memory)
     * regionCounter is used to determine which FPROT[3:0] register to check for protection status
     * Note: FPROT=1 means NOT protected, FPROT=0 means protected */
    regionCounter = 0; /* make sure regionCounter is initialized to 0 first */
    while (regionCounter < flashProtectionInfo.regionCount)
    {
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
        if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
        {
            if (regionCounter < 8)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTSL_REG >> regionCounter) & (0x01u
            }
            else if ((regionCounter >= 8) && (regionCounter < 16))
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTSH_REG >> (regionCounter - 8)) & (0
            }
            else
            {
                break;
            }
        }
        else
#endif
        {
            /* Note: So far protection region count may be 16/20/24/32/64 */
            if (regionCounter < 8)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTL3_REG >> regionCounter) & (0x01u)
            }
            else if ((regionCounter >= 8) && (regionCounter < 16))
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTL2_REG >> (regionCounter - 8)) & (0x
            }
#if defined(FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT) && (FSL_FEATURE_FL
#if (FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT == 20)
            else if ((regionCounter >= 16) && (regionCounter < 20))
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTL1_REG >> (regionCounter - 16)) & (0
            }
#else
            else if ((regionCounter >= 16) && (regionCounter < 24))
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTL1_REG >> (regionCounter - 16)) & (0
            }
#endif /* (FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT == 20) */
#endif
#if defined(FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT) && (FSL_FEATURE_FL
            else if ((regionCounter >= 24) && (regionCounter < 32))
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTL0_REG >> (regionCounter - 24)) & (0
            }
#endif
#if defined(FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT) && \
    (FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT == 64)
            else if (regionCounter < 40)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTH3_REG >> (regionCounter - 32)) & (
            }
            else if (regionCounter < 48)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTH2_REG >> (regionCounter - 40)) & (
            }
            else if (regionCounter < 56)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTH1_REG >> (regionCounter - 48)) & (
            }
            else if (regionCounter < 64)
            {
                flashRegionProtectStatus[regionCounter] = (FTFx_FPROTH0_REG >> (regionCounter - 56)) & (
            }
#endif
            else
            {
                break;
            }
        }

        regionCounter++;
    }

    /* loop through the flash regions and check
     * desired flash address range for protection status
     * loop stops when it is detected that start has exceeded the endAddress */
    regionCounter = 0; /* make sure regionCounter is initialized to 0 first */
    regionCheckedCounter = 0;
    protectStatusCounter = 0; /* make sure protectStatusCounter is initialized to 0 first */
    while (start < endAddress)
    {
        /* check to see if the address falls within this protection region
         * Note that if the entire flash is to be checked, the last protection
         * region checked would consist of the last protection start address and
         * the start address following the end of flash */
        if ((start >= flashRegionAddress[regionCounter]) && (start < flashRegionAddress[regionCounter + 1]))
        {
            /* increment regionCheckedCounter to indicate this region was checked */
            regionCheckedCounter++;

            /* check the protection status of this region
             * Note: FPROT=1 means NOT protected, FPROT=0 means protected */
            if (!flashRegionProtectStatus[regionCounter])
            {
```

```
            /* increment protectStatusCounter to indicate this region is protected */
            protectStatusCounter++;
        }
        start += flashProtectionInfo.regionSize; /* increment to an address within the next region */
    }
    regionCounter++; /* increment regionCounter to check for the next flash protection region */
}

/* if protectStatusCounter == 0, then no region of the desired flash region is protected */
if (protectStatusCounter == 0)
{
    *protection_state = kFLASH_ProtectionStateUnprotected;
}
/* if protectStatusCounter == regionCheckedCounter, then each region checked was protected */
else if (protectStatusCounter == regionCheckedCounter)
{
    *protection_state = kFLASH_ProtectionStateProtected;
}
/* if protectStatusCounter != regionCheckedCounter, then protection status is mixed
 * In other words, some regions are protected while others are unprotected */
else
{
    *protection_state = kFLASH_ProtectionStateMixed;
}

return (returnCode);
}

status_t FLASH_IsExecuteOnly(flash_config_t *config,
                             uint32_t start,
                             uint32_t lengthInBytes,
                             flash_execute_only_access_state_t *access_state)
{
#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL) && FSL_FEATURE_FLASH_HAS_ACC
    flash_access_config_t flashAccessInfo; /* flash Execute-Only information */
#endif                                     /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */
    status_t returnCode;

    if (access_state == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Check the supplied address range. */
    returnCode = flash_check_range(config, start, lengthInBytes, FSL_FEATURE_FLASH_PFLASH_BLOC
    if (returnCode)
    {
        return returnCode;
    }

#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL) && FSL_FEATURE_FLASH_HAS_ACC
    /* Get necessary flash Execute-Only information. */
    returnCode = flash_get_access_info(config, &flashAccessInfo);
    if (returnCode)
    {
        return returnCode;
    }

    {
        uint32_t executeOnlySegmentCounter = 0;

        /* calculating end address */
        uint32_t endAddress = start + lengthInBytes;

        /* Aligning start address and end address */
        uint32_t alignedStartAddress = ALIGN_DOWN(start, flashAccessInfo.SegmentSize);
        uint32_t alignedEndAddress = ALIGN_UP(endAddress, flashAccessInfo.SegmentSize);

        uint32_t segmentIndex = 0;
        uint32_t maxSupportedExecuteOnlySegmentCount =
            (alignedEndAddress - alignedStartAddress) / flashAccessInfo.SegmentSize;

        while (start < endAddress)
        {
            uint32_t xacc;

            segmentIndex = (start - flashAccessInfo.SegmentBase) / flashAccessInfo.SegmentSize;

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
            if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
            {
                /* For secondary flash, The two XACCS registers allow up to 16 restricted segments of equal me
                 */
                if (segmentIndex < 8)
                {
                    xacc = *(const volatile uint8_t *)&FTFx_XACCSL_REG;
                }
                else if (segmentIndex < flashAccessInfo.SegmentCount)
                {
                    xacc = *(const volatile uint8_t *)&FTFx_XACCSH_REG;
                    segmentIndex -= 8;
                }
                else
                {
                    break;
                }
            }
            else
#endif
            {
                /* For primary flash, The eight XACC registers allow up to 64 restricted segments of equal memo
                 */
                if (segmentIndex < 32)
                {
                    xacc = *(const volatile uint32_t *)&FTFx_XACCL3_REG;
                }
                else if (segmentIndex < flashAccessInfo.SegmentCount)
                {
                    xacc = *(const volatile uint32_t *)&FTFx_XACCH3_REG;
                    segmentIndex -= 32;
                }
                else
                {
                    break;
                }
            }

            /* Determine if this address range is in a execute-only protection flash segment. */
            if ((~xacc) & (1u << segmentIndex))
            {
                executeOnlySegmentCounter++;
            }

            start += flashAccessInfo.SegmentSize;
        }

        if (executeOnlySegmentCounter < 1u)
        {
            *access_state = kFLASH_AccessStateUnLimited;
        }
        else if (executeOnlySegmentCounter < maxSupportedExecuteOnlySegmentCount)
        {
            *access_state = kFLASH_AccessStateMixed;
        }
        else
        {
            *access_state = kFLASH_AccessStateExecuteOnly;
        }
    }
#else
    *access_state = kFLASH_AccessStateUnLimited;
#endif /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */

    return (returnCode);
}

status_t FLASH_GetProperty(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t *value)
{
    if ((config == NULL) || (value == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    switch (whichProperty)
    {
        case kFLASH_PropertyPflashSectorSize:
            *value = config->PFlashSectorSize;
            break;
        case kFLASH_PropertyPflashTotalSize:
            *value = config->PFlashTotalSize;
            break;

        case kFLASH_PropertyPflashBlockSize:
            *value = config->PFlashTotalSize / FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT;
            break;

        case kFLASH_PropertyPflashBlockCount:
            *value = (uint32_t)config->PFlashBlockCount;
            break;

        case kFLASH_PropertyPflashBlockBaseAddr:
            *value = config->PFlashBlockBase;
            break;

        case kFLASH_PropertyPflashFacSupport:
#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL)
            *value = FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL;
#else
            *value = 0;
#endif /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */
            break;

        case kFLASH_PropertyPflashAccessSegmentSize:
            *value = config->PFlashAccessSegmentSize;
            break;

        case kFLASH_PropertyPflashAccessSegmentCount:
            *value = config->PFlashAccessSegmentCount;
            break;

        case kFLASH_PropertyFlexRamBlockBaseAddr:
            *value = config->FlexRAMBlockBase;
            break;

        case kFLASH_PropertyFlexRamTotalSize:
            *value = config->FlexRAMTotalSize;
            break;

#if FLASH_SSD_IS_FLEXNVM_ENABLED
        case kFLASH_PropertyDflashSectorSize:
            *value = FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE;
            break;
        case kFLASH_PropertyDflashTotalSize:
            *value = config->DFlashTotalSize;
            break;
        case kFLASH_PropertyDflashBlockSize:
            *value = FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE;
            break;
        case kFLASH_PropertyDflashBlockCount:
            *value = FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT;
            break;
        case kFLASH_PropertyDflashBlockBaseAddr:
            *value = config->DFlashBlockBase;
            break;
        case kFLASH_PropertyEepromTotalSize:
            *value = config->EEpromTotalSize;
            break;
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

        default: /* catch inputs that are not recognized */
            return kStatus_FLASH_UnknownProperty;
    }

    return kStatus_FLASH_Success;
}

status_t FLASH_SetProperty(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t value)
{
    status_t status = kStatus_FLASH_Success;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    switch (whichProperty)
    {
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED
        case kFLASH_PropertyFlashMemoryIndex:
            if ((value != (uint32_t)kFLASH_MemoryIndexPrimaryFlash) &&
                (value != (uint32_t)kFLASH_MemoryIndexSecondaryFlash))
            {
                return kStatus_FLASH_InvalidPropertyValue;
            }
            config->FlashMemoryIndex = (uint8_t)value;
            break;
#endif /* FLASH_SSD_IS_SECONDARY_FLASH_ENABLED */

        case kFLASH_PropertyFlashCacheControllerIndex:
            if ((value != (uint32_t)kFLASH_CacheControllerIndexForCore0) &&
                (value != (uint32_t)kFLASH_CacheControllerIndexForCore1))
            {
                return kStatus_FLASH_InvalidPropertyValue;
            }
            config->FlashCacheControllerIndex = (uint8_t)value;
            break;

        case kFLASH_PropertyPflashSectorSize:
        case kFLASH_PropertyPflashTotalSize:
        case kFLASH_PropertyPflashBlockSize:
        case kFLASH_PropertyPflashBlockCount:
        case kFLASH_PropertyPflashBlockBaseAddr:
        case kFLASH_PropertyPflashFacSupport:
        case kFLASH_PropertyPflashAccessSegmentSize:
```

```
        case kFLASH_PropertyPflashAccessSegmentCount:
        case kFLASH_PropertyFlexRamBlockBaseAddr:
        case kFLASH_PropertyFlexRamTotalSize:
#if FLASH_SSD_IS_FLEXNVM_ENABLED
        case kFLASH_PropertyDflashSectorSize:
        case kFLASH_PropertyDflashTotalSize:
        case kFLASH_PropertyDflashBlockSize:
        case kFLASH_PropertyDflashBlockCount:
        case kFLASH_PropertyDflashBlockBaseAddr:
        case kFLASH_PropertyEepromTotalSize:
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */
            status = kStatus_FLASH_ReadOnlyProperty;
            break;
        default: /* catch inputs that are not recognized */
            status = kStatus_FLASH_UnknownProperty;
            break;
    }

    return status;
}

#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
status_t FLASH_SetFlexramFunction(flash_config_t *config, flash_flexram_function_option_t option)
{
    status_t status;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    status = flasn_check_flexram_function_option_range(option);
    if (status != kStatus_FLASH_Success)
    {
        return status;
    }

    /* preparing passing parameter to verify all block command */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_1_2(FTFx_SET_FLEXRAM_FUNCTION, option, 0xFFFFU

    /* calling flash command sequence function to execute the command */
    return flash_command_sequence(config);
}
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD) && FSL_FEATURE_FLASH_HAS_S
status_t FLASH_SwapControl(flash_config_t *config,
                           uint32_t address,
                           flash_swap_control_option_t option,
                           flash_swap_state_config_t *returnInfo)
{
    status_t returnCode;
    if ((config == NULL) || (returnInfo == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    if (address & (FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT - 1)
    {
        return kStatus_FLASH_AlignmentError;
    }

    /* Make sure address provided is in the lower half of Program flash but not in the Flash Configuration Fie
    if ((address >= (config->PFlashTotalSize / 2)) ||
        ((address >= kFLASH_ConfigAreaStart) && (address <= kFLASH_ConfigAreaEnd)))
    {
        return kStatus_FLASH_SwapIndicatorAddressError;
    }

    /* Check the option. */
    returnCode = flash_check_swap_control_option(option);
    if (returnCode)
    {
        return returnCode;
    }

    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_3(FTFx_SWAP_CONTROL, address);
    kFCCOBx[1] = BYTES_JOIN_TO_WORD_1_3(option, 0xFFFFFFU);

    returnCode = flash_command_sequence(config);

    returnInfo->flashSwapState = (flash_swap_state_t)FTFx_FCCOB5_REG;
    returnInfo->currentSwapBlockStatus = (flash_swap_block_status_t)FTFx_FCCOB6_REG;
    returnInfo->nextSwapBlockStatus = (flash_swap_block_status_t)FTFx_FCCOB7_REG;

    return returnCode;
}
#endif /* FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP) && FSL_FEATURE_FLASH_HAS_P
status_t FLASH_Swap(flash_config_t *config, uint32_t address, flash_swap_function_option_t option)
{
    flash_swap_state_config_t returnInfo;
    status_t returnCode;

    memset(&returnInfo, 0xFFU, sizeof(returnInfo));

    do
    {
        returnCode = FLASH_SwapControl(config, address, kFLASH_SwapControlOptionReportStatus, &retu
        if (returnCode != kStatus_FLASH_Success)
        {
            return returnCode;
        }
```

```
if (kFLASH_SwapFunctionOptionDisable == option)
{
    if (returnInfo.flashSwapState == kFLASH_SwapStateDisabled)
    {
        return kStatus_FLASH_Success;
    }
    else if (returnInfo.flashSwapState == kFLASH_SwapStateUninitialized)
    {
        /* The swap system changed to the DISABLED state with Program flash block 0
         * located at relative flash address 0x0_0000 */
        returnCode = FLASH_SwapControl(config, address, kFLASH_SwapControlOptionDisableSysten
    }
    else
    {
        /* Swap disable should be requested only when swap system is in the uninitialized state */
        return kStatus_FLASH_SwapSystemNotInUninitialized;
    }
}
else
{
    /* When first swap: the initial swap state is Uninitialized, flash swap inidicator address is unset,
     *    the swap procedure should be Uninitialized -> Update-Erased -> Complete.
     * After the first swap has been completed, the flash swap inidicator address cannot be modified
     *    unless EraseAllBlocks command is issued, the swap procedure is changed to Update -> Updat
     *    Complete. */
    switch (returnInfo.flashSwapState)
    {
        case kFLASH_SwapStateUninitialized:
            /* If current swap mode is Uninitialized, Initialize Swap to Initialized/READY state. */
            returnCode =
                FLASH_SwapControl(config, address, kFLASH_SwapControlOptionIntializeSystem, &returr
            break;
        case kFLASH_SwapStateReady:
            /* Validate whether the address provided to the swap system is matched to
             * swap indicator address in the IFR */
            returnCode = flash_validate_swap_indicator_address(config, address);
            if (returnCode == kStatus_FLASH_Success)
            {
                /* If current swap mode is Initialized/Ready, Initialize Swap to UPDATE state. */
                returnCode =
                    FLASH_SwapControl(config, address, kFLASH_SwapControlOptionSetInUpdateState, &
            }
            break;
        case kFLASH_SwapStateUpdate:
            /* If current swap mode is Update, Erase indicator sector in non active block
             * to proceed swap system to update-erased state */
            returnCode = FLASH_Erase(config, address + (config->PFlashTotalSize >> 1),
                                     FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT,
            break;
        case kFLASH_SwapStateUpdateErased:
            /* If current swap mode is Update or Update-Erased, progress Swap to COMPLETE State */
            returnCode =
                FLASH_SwapControl(config, address, kFLASH_SwapControlOptionSetInCompleteState, &
                break;
            case kFLASH_SwapStateComplete:
                break;
            case kFLASH_SwapStateDisabled:
                /* When swap system is in disabled state, We need to clear swap system back to uninitialized
                 * by issuing EraseAllBlocks command */
                returnCode = kStatus_FLASH_SwapSystemNotInUninitialized;
                break;
            default:
                returnCode = kStatus_FLASH_InvalidArgument;
                break;
        }
    }
    if (returnCode != kStatus_FLASH_Success)
    {
        break;
    }
} while (!((kFLASH_SwapStateComplete == returnInfo.flashSwapState) && (kFLASH_SwapFunctionOpt

return returnCode;
}
#endif /* FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP */

#if defined(FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD) && FSL_FEATURE_FLASH_H
status_t FLASH_ProgramPartition(flash_config_t *config,
                                flash_partition_flexram_load_option_t option,
                                uint32_t eepromDataSizeCode,
                                uint32_t flexnvmPartitionCode)
{
    status_t returnCode;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* eepromDataSizeCode[7:6], flexnvmPartitionCode[7:4] should be all 1'b0
     *  or it will cause access error. */
    /* eepromDataSizeCode &= 0x3FU;  */
    /* flexnvmPartitionCode &= 0x0FU; */

    /* preparing passing parameter to program the flash block */
    kFCCOBx[0] = BYTES_JOIN_TO_WORD_1_2_1(FTFx_PROGRAM_PARTITION, 0xFFFFU, option);
    kFCCOBx[1] = BYTES_JOIN_TO_WORD_1_1_2(eepromDataSizeCode, flexnvmPartitionCode, 0xFFFF

    flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);

    /* calling flash command sequence function to execute the command */
    returnCode = flash_command_sequence(config);

    flash_cache_clear(config);

#if FLASH_SSD_IS_FLEXNVM_ENABLED
```

```
    /* Data flash IFR will be updated by program partition command during reset sequence,
     * so we just set reserved values for partitioned FlexNVM size here */
    config->EEpromTotalSize = FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_RESERVED;
    config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif

    return (returnCode);
}
#endif /* FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD */

status_t FLASH_PflashSetProtection(flash_config_t *config, pflash_protection_status_t *protectStatus)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
    if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
    {
        *kFPROTSL = protectStatus->valueLow32b.prots16b.protsl;
        if (protectStatus->valueLow32b.prots16b.protsl != *kFPROTSL)
        {
            return kStatus_FLASH_CommandFailure;
        }

        *kFPROTSH = protectStatus->valueLow32b.prots16b.protsh;
        if (protectStatus->valueLow32b.prots16b.protsh != *kFPROTSH)
        {
            return kStatus_FLASH_CommandFailure;
        }
    }
    else
#endif
    {
        *kFPROTL = protectStatus->valueLow32b.protl32b;
        if (protectStatus->valueLow32b.protl32b != *kFPROTL)
        {
            return kStatus_FLASH_CommandFailure;
        }

#if defined(FTFx_FPROT_HIGH_REG)
        *kFPROTH = protectStatus->valueHigh32b.proth32b;
        if (protectStatus->valueHigh32b.proth32b != *kFPROTH)
        {
            return kStatus_FLASH_CommandFailure;
        }
#endif
    }

    return kStatus_FLASH_Success;
}
```

```
status_t FLASH_PflashGetProtection(flash_config_t *config, pflash_protection_status_t *protectStatus)
{
    if ((config == NULL) || (protectStatus == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
    if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
    {
        protectStatus->valueLow32b.prots16b.protsl = *kFPROTSL;
        protectStatus->valueLow32b.prots16b.protsh = *kFPROTSH;
    }
    else
#endif
    {
        protectStatus->valueLow32b.protl32b = *kFPROTL;
#if defined(FTFx_FPROT_HIGH_REG)
        protectStatus->valueHigh32b.proth32b = *kFPROTH;
#endif
    }

    return kStatus_FLASH_Success;
}

#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_DflashSetProtection(flash_config_t *config, uint8_t protectStatus)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    if ((config->DFlashTotalSize == 0) || (config->DFlashTotalSize == FLEX_NVM_DFLASH_SIZE_FOR_DE
    {
        return kStatus_FLASH_CommandNotSupported;
    }

    FTFx->FDPROT = protectStatus;

    if (FTFx->FDPROT != protectStatus)
    {
        return kStatus_FLASH_CommandFailure;
    }

    return kStatus_FLASH_Success;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_DflashGetProtection(flash_config_t *config, uint8_t *protectStatus)
{
    if ((config == NULL) || (protectStatus == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    if ((config->DFlashTotalSize == 0) || (config->DFlashTotalSize == FLEX_NVM_DFLASH_SIZE_FOR_DE
    {
        return kStatus_FLASH_CommandNotSupported;
    }

    *protectStatus = FTFx->FDPROT;

    return kStatus_FLASH_Success;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromSetProtection(flash_config_t *config, uint8_t protectStatus)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    if ((config->EEpromTotalSize == 0) || (config->EEpromTotalSize == FLEX_NVM_EEPROM_SIZE_FOR_
    {
        return kStatus_FLASH_CommandNotSupported;
    }

    FTFx->FEPROT = protectStatus;

    if (FTFx->FEPROT != protectStatus)
    {
        return kStatus_FLASH_CommandFailure;
    }

    return kStatus_FLASH_Success;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromGetProtection(flash_config_t *config, uint8_t *protectStatus)
{
    if ((config == NULL) || (protectStatus == NULL))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    if ((config->EEpromTotalSize == 0) || (config->EEpromTotalSize == FLEX_NVM_EEPROM_SIZE_FOR_
    {
        return kStatus_FLASH_CommandNotSupported;
    }

    *protectStatus = FTFx->FEPROT;
    return kStatus_FLASH_Success;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

status_t FLASH_PflashSetPrefetchSpeculation(flash_prefetch_speculation_status_t *speculationStatus)
{
#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MCM
    {
        FTFx_REG32_ACCESS_TYPE regBase;
#if defined(MCM)
        regBase = (FTFx_REG32_ACCESS_TYPE)&MCM->PLACR;
#elif defined(MCM0)
        regBase = (FTFx_REG32_ACCESS_TYPE)&MCM0->PLACR;
#endif
        if (speculationStatus->instructionOption == kFLASH_prefetchSpeculationOptionDisable)
        {
            if (speculationStatus->dataOption == kFLASH_prefetchSpeculationOptionEnable)
            {
                return kStatus_FLASH_InvalidSpeculationOption;
            }
            else
            {
                *regBase |= MCM_PLACR_DFCS_MASK;
            }
        }
        else
        {
            *regBase &= ~MCM_PLACR_DFCS_MASK;
            if (speculationStatus->dataOption == kFLASH_prefetchSpeculationOptionEnable)
            {
                *regBase |= MCM_PLACR_EFDS_MASK;
            }
            else
            {
                *regBase &= ~MCM_PLACR_EFDS_MASK;
            }
        }
    }
#elif FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC
    {
        FTFx_REG32_ACCESS_TYPE regBase;
        uint32_t b0dpeMask, b0ipeMask;
#if defined(FMC_PFB01CR_B0DPE_MASK)
        regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB01CR;
        b0dpeMask = FMC_PFB01CR_B0DPE_MASK;
        b0ipeMask = FMC_PFB01CR_B0IPE_MASK;
#elif defined(FMC_PFB0CR_B0DPE_MASK)
        regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB0CR;
        b0dpeMask = FMC_PFB0CR_B0DPE_MASK;
        b0ipeMask = FMC_PFB0CR_B0IPE_MASK;
#endif
        if (speculationStatus->instructionOption == kFLASH_prefetchSpeculationOptionEnable)
        {
            *regBase |= b0ipeMask;
        }
        else
        {
            *regBase &= ~b0ipeMask;
        }
        if (speculationStatus->dataOption == kFLASH_prefetchSpeculationOptionEnable)
        {
            *regBase |= b0dpeMask;
        }
        else
        {
            *regBase &= ~b0dpeMask;
        }

/* Invalidate Prefetch Speculation Buffer */
#if defined(FMC_PFB01CR_S_INV_MASK)
        FMC->PFB01CR |= FMC_PFB01CR_S_INV_MASK;
#elif defined(FMC_PFB01CR_S_B_INV_MASK)
        FMC->PFB01CR |= FMC_PFB01CR_S_B_INV_MASK;
#elif defined(FMC_PFB0CR_S_INV_MASK)
        FMC->PFB0CR |= FMC_PFB0CR_S_INV_MASK;
#elif defined(FMC_PFB0CR_S_B_INV_MASK)
        FMC->PFB0CR |= FMC_PFB0CR_S_B_INV_MASK;
#endif
    }
#elif FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
    {
        FTFx_REG32_ACCESS_TYPE regBase;
        uint32_t flashSpeculationMask, dataPrefetchMask;
        regBase = (FTFx_REG32_ACCESS_TYPE)&MSCM->OCMDR[0];
        flashSpeculationMask = MSCM_OCMDR_OCMC1_DFCS_MASK;
        dataPrefetchMask = MSCM_OCMDR_OCMC1_DFDS_MASK;

        if (speculationStatus->instructionOption == kFLASH_prefetchSpeculationOptionDisable)
        {
            if (speculationStatus->dataOption == kFLASH_prefetchSpeculationOptionEnable)
            {
                return kStatus_FLASH_InvalidSpeculationOption;
            }
            else
            {
                *regBase |= flashSpeculationMask;
            }
        }
        else
        {
            *regBase &= ~flashSpeculationMask;
            if (speculationStatus->dataOption == kFLASH_prefetchSpeculationOptionEnable)
            {
                *regBase &= ~dataPrefetchMask;
            }
            else
            {
                *regBase |= dataPrefetchMask;
            }
        }
    }
#endif /* FSL_FEATURE_FTFx_MCM_FLASH_CACHE_CONTROLS */

    return kStatus_FLASH_Success;
}

status_t FLASH_PflashGetPrefetchSpeculation(flash_prefetch_speculation_status_t *speculationStatus)
{
    memset(speculationStatus, 0, sizeof(flash_prefetch_speculation_status_t));

    /* Assuming that all speculation options are enabled. */
    speculationStatus->instructionOption = kFLASH_prefetchSpeculationOptionEnable;
    speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionEnable;

#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MCM
    {
        uint32_t value;
#if defined(MCM)
        value = MCM->PLACR;
#elif defined(MCM0)
        value = MCM0->PLACR;
#endif
        if (value & MCM_PLACR_DFCS_MASK)
        {
            /* Speculation buffer is off. */
            speculationStatus->instructionOption = kFLASH_prefetchSpeculationOptionDisable;
            speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionDisable;
        }
        else
        {
            /* Speculation buffer is on for instruction. */
            if (!(value & MCM_PLACR_EFDS_MASK))
            {
                /* Speculation buffer is off for data. */
                speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionDisable;
            }
        }
    }
#elif FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC
    {
        uint32_t value;
        uint32_t b0dpeMask, b0ipeMask;
#if defined(FMC_PFB01CR_B0DPE_MASK)
        value = FMC->PFB01CR;
        b0dpeMask = FMC_PFB01CR_B0DPE_MASK;
        b0ipeMask = FMC_PFB01CR_B0IPE_MASK;
#elif defined(FMC_PFB0CR_B0DPE_MASK)
        value = FMC->PFB0CR;
        b0dpeMask = FMC_PFB0CR_B0DPE_MASK;
        b0ipeMask = FMC_PFB0CR_B0IPE_MASK;
#endif
        if (!(value & b0dpeMask))
        {
            /* Do not prefetch in response to data references. */
            speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionDisable;
        }
        if (!(value & b0ipeMask))
        {
            /* Do not prefetch in response to instruction fetches. */
            speculationStatus->instructionOption = kFLASH_prefetchSpeculationOptionDisable;
        }
    }
#elif FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
    {
        uint32_t value;
        uint32_t flashSpeculationMask, dataPrefetchMask;
        value = MSCM->OCMDR[0];
        flashSpeculationMask = MSCM_OCMDR_OCMC1_DFCS_MASK;
        dataPrefetchMask = MSCM_OCMDR_OCMC1_DFDS_MASK;

        if (value & flashSpeculationMask)
        {
            /* Speculation buffer is off. */
            speculationStatus->instructionOption = kFLASH_prefetchSpeculationOptionDisable;
            speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionDisable;
        }
        else
        {
            /* Speculation buffer is on for instruction. */
            if (value & dataPrefetchMask)
            {
                /* Speculation buffer is off for data. */
                speculationStatus->dataOption = kFLASH_prefetchSpeculationOptionDisable;
            }
        }
    }
#endif

    return kStatus_FLASH_Success;
}

#if FLASH_DRIVER_IS_FLASH_RESIDENT
/*!
 * @brief Copy PIC of flash_run_command() to RAM
 */
static void copy_flash_run_command(uint32_t *flashRunCommand)
{
    assert(sizeof(s_flashRunCommandFunctionCode) <= (kFLASH_ExecuteInRamFunctionMaxSizeInWord

    /* Since the value of ARM function pointer is always odd, but the real start address
     * of function memory should be even, that's why +1 operation exist. */
    memcpy((void *)flashRunCommand, (void *)s_flashRunCommandFunctionCode, sizeof(s_flashRunCom
    callFlashRunCommand = (void (*)(FTFx_REG8_ACCESS_TYPE ftfx_fstat))((uint32_t)flashRunComman
}
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

/*!
 * @brief Flash Command Sequence
 *
 * This function is used to perform the command write sequence to the flash.
 *
 * @param driver Pointer to storage for the driver runtime state.
 * @return An error code or kStatus_FLASH_Success
 */
static status_t flash_command_sequence(flash_config_t *config)
{
    uint8_t registerValue;

#if FLASH_DRIVER_IS_FLASH_RESIDENT
    /* clear RDCOLERR & ACCERR & FPVIOL flag in flash status register */
    FTFx->FSTAT = FTFx_FSTAT_RDCOLERR_MASK | FTFx_FSTAT_ACCERR_MASK | FTFx_FSTAT_F

    status_t returnCode = flash_check_execute_in_ram_function_info(config);
    if (kStatus_FLASH_Success != returnCode)
    {
        return returnCode;
    }

    /* We pass the ftfx_fstat address as a parameter to flash_run_comamnd() instead of using
     * pre-processed MICRO sentences or operating global variable in flash_run_comamnd()
     * to make sure that flash_run_command() will be compiled into position-independent code (PIC). */
    callFlashRunCommand((FTFx_REG8_ACCESS_TYPE)(&FTFx->FSTAT));
#else
    /* clear RDCOLERR & ACCERR & FPVIOL flag in flash status register */
    FTFx->FSTAT = FTFx_FSTAT_RDCOLERR_MASK | FTFx_FSTAT_ACCERR_MASK | FTFx_FSTAT_F

    /* clear CCIF bit */
    FTFx->FSTAT = FTFx_FSTAT_CCIF_MASK;

    /* Check CCIF bit of the flash status register, wait till it is set.
     * IP team indicates that this loop will always complete. */
    while (!(FTFx->FSTAT & FTFx_FSTAT_CCIF_MASK))
    {
    }
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

    /* Check error bits */
    /* Get flash status register value */
    registerValue = FTFx->FSTAT;

    /* checking access error */
    if (registerValue & FTFx_FSTAT_ACCERR_MASK)
    {
        return kStatus_FLASH_AccessError;
    }
    /* checking protection error */
    else if (registerValue & FTFx_FSTAT_FPVIOL_MASK)
    {
        return kStatus_FLASH_ProtectionViolation;
    }
    /* checking MGSTAT0 non-correctable error */
    else if (registerValue & FTFx_FSTAT_MGSTAT0_MASK)
    {
        return kStatus_FLASH_CommandFailure;
    }
    else
    {
        return kStatus_FLASH_Success;
    }
}

#if FLASH_DRIVER_IS_FLASH_RESIDENT
/*!
 * @brief Copy PIC of flash_common_bit_operation() to RAM
 *
 */
static void copy_flash_common_bit_operation(uint32_t *flashCommonBitOperation)
{
    assert(sizeof(s_flashCommonBitOperationFunctionCode) <= (kFLASH_ExecuteInRamFunctionMaxSize

    /* Since the value of ARM function pointer is always odd, but the real start address
     * of function memory should be even, that's why +1 operation exist. */
    memcpy((void *)flashCommonBitOperation, (void *)s_flashCommonBitOperationFunctionCode,
           sizeof(s_flashCommonBitOperationFunctionCode));
    callFlashCommonBitOperation = (void (*)(FTFx_REG32_ACCESS_TYPE base, uint32_t bitMask, uint32
                                           uint32_t bitValue))((uint32_t)flashCommonBitOperation + 1);
    /* Workround for some devices which doesn't need this function */
    callFlashCommonBitOperation((FTFx_REG32_ACCESS_TYPE)0, 0, 0, 0);
}
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

#if FLASH_CACHE_IS_CONTROLLED_BY_MCM
/*! @brief Performs the cache clear to the flash by MCM.*/
void mcm_flash_cache_clear(flash_config_t *config)
{
    FTFx_REG32_ACCESS_TYPE regBase = (FTFx_REG32_ACCESS_TYPE)&MCM0_CACHE_REG;

#if defined(MCM0) && defined(MCM1)
    if (config->FlashCacheControllerIndex == (uint8_t)kFLASH_CacheControllerIndexForCore1)
    {
        regBase = (FTFx_REG32_ACCESS_TYPE)&MCM1_CACHE_REG;
    }
#endif

#if FLASH_DRIVER_IS_FLASH_RESIDENT
    callFlashCommonBitOperation(regBase, MCM_CACHE_CLEAR_MASK, MCM_CACHE_CLEAR_SHIFT
#else  /* !FLASH_DRIVER_IS_FLASH_RESIDENT */
```

```
    *regBase |= MCM_CACHE_CLEAR_MASK;

    /* Memory barriers for good measure.
     * All Cache, Branch predictor and TLB maintenance operations before this instruction complete */
    __ISB();
    __DSB();
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */
}
#endif /* FLASH_CACHE_IS_CONTROLLED_BY_MCM */

#if FLASH_CACHE_IS_CONTROLLED_BY_FMC
/*! @brief Performs the cache clear to the flash by FMC.*/
void fmc_flash_cache_clear(void)
{
#if FLASH_DRIVER_IS_FLASH_RESIDENT
    FTFx_REG32_ACCESS_TYPE regBase = (FTFx_REG32_ACCESS_TYPE)0;
#if defined(FMC_PFB01CR_CINV_WAY_MASK)
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB01CR;
    callFlashCommonBitOperation(regBase, FMC_PFB01CR_CINV_WAY_MASK, FMC_PFB01CR_CINV_
#else
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB0CR;
    callFlashCommonBitOperation(regBase, FMC_PFB0CR_CINV_WAY_MASK, FMC_PFB0CR_CINV_W
#endif
#else /* !FLASH_DRIVER_IS_FLASH_RESIDENT */
#if defined(FMC_PFB01CR_CINV_WAY_MASK)
    FMC->PFB01CR = (FMC->PFB01CR & ~FMC_PFB01CR_CINV_WAY_MASK) | FMC_PFB01CR_CINV
#else
    FMC->PFB0CR = (FMC->PFB0CR & ~FMC_PFB0CR_CINV_WAY_MASK) | FMC_PFB0CR_CINV_WA
#endif
    /* Memory barriers for good measure.
     * All Cache, Branch predictor and TLB maintenance operations before this instruction complete */
    __ISB();
    __DSB();
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */
}
#endif /* FLASH_CACHE_IS_CONTROLLED_BY_FMC */

#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
/*! @brief Performs the prefetch speculation buffer clear to the flash by MSCM.*/
void mscm_flash_prefetch_speculation_enable(bool enable)
{
    uint8_t setValue;
    if (enable)
    {
        setValue = 0x0U;
    }
    else
    {
        setValue = 0x3U;
    }

/* The OCMDR[0] is always used to prefetch main Pflash*/
/* For device with FlexNVM support, the OCMDR[1] is used to prefetch Dflash.
```

```
 * For device with secondary flash support, the OCMDR[1] is used to prefetch secondary Pflash. */
#if FLASH_DRIVER_IS_FLASH_RESIDENT
    callFlashCommonBitOperation((FTFx_REG32_ACCESS_TYPE)&MSCM->OCMDR[0], MSCM_SPECUL
                                MSCM_SPECULATION_DISABLE_SHIFT, setValue);
#if FLASH_SSD_IS_FLEXNVM_ENABLED || BL_HAS_SECONDARY_INTERNAL_FLASH
    callFlashCommonBitOperation((FTFx_REG32_ACCESS_TYPE)&MSCM->OCMDR[1], MSCM_SPECUL
                                MSCM_SPECULATION_DISABLE_SHIFT, setValue);
#endif
#else /* !FLASH_DRIVER_IS_FLASH_RESIDENT */
    MSCM->OCMDR[0] |= MSCM_SPECULATION_DISABLE(setValue);

    /* Memory barriers for good measure.
     * All Cache, Branch predictor and TLB maintenance operations before this instruction complete */
    __ISB();
    __DSB();
#if FLASH_SSD_IS_FLEXNVM_ENABLED || BL_HAS_SECONDARY_INTERNAL_FLASH
    MSCM->OCMDR[1] |= MSCM_SPECULATION_DISABLE(setValue);

    /* Each cahce clear instaruction should be followed by below code*/
    __ISB();
    __DSB();
#endif

#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */
}
#endif /* FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM */

#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC
/*! @brief Performs the prefetch speculation buffer clear to the flash by FMC.*/
void fmc_flash_prefetch_speculation_clear(void)
{
#if FLASH_DRIVER_IS_FLASH_RESIDENT
    FTFx_REG32_ACCESS_TYPE regBase = (FTFx_REG32_ACCESS_TYPE)0;
#if defined(FMC_PFB01CR_S_INV_MASK)
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB01CR;
    callFlashCommonBitOperation(regBase, FMC_PFB01CR_S_INV_MASK, FMC_PFB01CR_S_INV_SHI
#elif defined(FMC_PFB01CR_S_B_INV_MASK)
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB01CR;
    callFlashCommonBitOperation(regBase, FMC_PFB01CR_S_B_INV_MASK, FMC_PFB01CR_S_B_INV
#elif defined(FMC_PFB0CR_S_INV_MASK)
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB0CR;
    callFlashCommonBitOperation(regBase, FMC_PFB0CR_S_INV_MASK, FMC_PFB0CR_S_INV_SHIFT
#elif defined(FMC_PFB0CR_S_B_INV_MASK)
    regBase = (FTFx_REG32_ACCESS_TYPE)&FMC->PFB0CR;
    callFlashCommonBitOperation(regBase, FMC_PFB0CR_S_B_INV_MASK, FMC_PFB0CR_S_B_INV_S
#endif
#else /* !FLASH_DRIVER_IS_FLASH_RESIDENT */
#if defined(FMC_PFB01CR_S_INV_MASK)
    FMC->PFB01CR |= FMC_PFB01CR_S_INV_MASK;
#elif defined(FMC_PFB01CR_S_B_INV_MASK)
    FMC->PFB01CR |= FMC_PFB01CR_S_B_INV_MASK;
#elif defined(FMC_PFB0CR_S_INV_MASK)
    FMC->PFB0CR |= FMC_PFB0CR_S_INV_MASK;
```

```
#elif defined(FMC_PFB0CR_S_B_INV_MASK)
    FMC->PFB0CR |= FMC_PFB0CR_S_B_INV_MASK;
#endif
    /* Memory barriers for good measure.
     * All Cache, Branch predictor and TLB maintenance operations before this instruction complete */
    __ISB();
    __DSB();
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */
}
#endif /* FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC */

/*!
 * @brief Flash Cache Clear
 *
 * This function is used to perform the cache and prefetch speculation clear to the flash.
 */
void flash_cache_clear(flash_config_t *config)
{
    flash_cache_clear_process(config, kFLASH_CacheClearProcessPost);
}

/*!
 * @brief Flash Cache Clear Process
 *
 * This function is used to perform the cache and prefetch speculation clear process to the flash.
 */
static void flash_cache_clear_process(flash_config_t *config, flash_cache_clear_process_t process)
{
#if FLASH_DRIVER_IS_FLASH_RESIDENT
    status_t returnCode = flash_check_execute_in_ram_function_info(config);
    if (kStatus_FLASH_Success != returnCode)
    {
        return;
    }
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

    /* We pass the ftfx register address as a parameter to flash_common_bit_operation() instead of using
     * pre-processed MACROs or a global variable in flash_common_bit_operation()
     * to make sure that flash_common_bit_operation() will be compiled into position-independent code (PIC
    if (process == kFLASH_CacheClearProcessPost)
    {
#if FLASH_CACHE_IS_CONTROLLED_BY_MCM
        mcm_flash_cache_clear(config);
#endif
#if FLASH_CACHE_IS_CONTROLLED_BY_FMC
        fmc_flash_cache_clear();
#endif
#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
        mscm_flash_prefetch_speculation_enable(true);
#endif
#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_FMC
        fmc_flash_prefetch_speculation_clear();
#endif
```

```
    }
    if (process == kFLASH_CacheClearProcessPre)
    {
#if FLASH_PREFETCH_SPECULATION_IS_CONTROLLED_BY_MSCM
        mscm_flash_prefetch_speculation_enable(false);
#endif
    }
}

#if FLASH_DRIVER_IS_FLASH_RESIDENT
/*! @brief Check whether flash execute-in-ram functions are ready  */
static status_t flash_check_execute_in_ram_function_info(flash_config_t *config)
{
    flash_execute_in_ram_function_config_t *flashExecuteInRamFunctionInfo;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    flashExecuteInRamFunctionInfo = (flash_execute_in_ram_function_config_t *)config->flashExecuteInRa

    if ((config->flashExecuteInRamFunctionInfo) &&
        (kFLASH_ExecuteInRamFunctionTotalNum == flashExecuteInRamFunctionInfo->activeFunctionCour
    {
        return kStatus_FLASH_Success;
    }

    return kStatus_FLASH_ExecuteInRamFunctionNotReady;
}
#endif /* FLASH_DRIVER_IS_FLASH_RESIDENT */

/*! @brief Validates the range and alignment of the given address range.*/
static status_t flash_check_range(flash_config_t *config,
                                  uint32_t startAddress,
                                  uint32_t lengthInBytes,
                                  uint32_t alignmentBaseline)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Verify the start and length are alignmentBaseline aligned. */
    if ((startAddress & (alignmentBaseline - 1)) || (lengthInBytes & (alignmentBaseline - 1)))
    {
        return kStatus_FLASH_AlignmentError;
    }

    /* check for valid range of the target addresses */
    if (
#if FLASH_SSD_IS_FLEXNVM_ENABLED
        ((startAddress >= config->DFlashBlockBase) &&
        ((startAddress + lengthInBytes) <= (config->DFlashBlockBase + config->DFlashTotalSize))) ||
#endif
        ((startAddress >= config->PFlashBlockBase) &&
        ((startAddress + lengthInBytes) <= (config->PFlashBlockBase + config->PFlashTotalSize))))
    {
        return kStatus_FLASH_Success;
    }

    return kStatus_FLASH_AddressError;
}

/*! @brief Gets the right address, sector and block size of current flash type which is indicated by address.*
static status_t flash_get_matched_operation_info(flash_config_t *config,
                                                 uint32_t address,
                                                 flash_operation_config_t *info)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Clean up info Structure*/
    memset(info, 0, sizeof(flash_operation_config_t));

#if FLASH_SSD_IS_FLEXNVM_ENABLED
    if ((address >= config->DFlashBlockBase) && (address <= (config->DFlashBlockBase + config->DFlash
    {
        /* When required by the command, address bit 23 selects between program flash memory
         * (=0) and data flash memory (=1).*/
        info->convertedAddress = address - config->DFlashBlockBase + 0x800000U;
        info->activeSectorSize = FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE;
        info->activeBlockSize = config->DFlashTotalSize / FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CO

        info->blockWriteUnitSize = FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE;
        info->sectorCmdAddressAligment = FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRES
        info->sectionCmdAddressAligment = FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRE
        info->resourceCmdAddressAligment = FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_AD
        info->checkCmdAddressAligment = FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS
    }
    else
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */
    {
        info->convertedAddress = address - config->PFlashBlockBase;
        info->activeSectorSize = config->PFlashSectorSize;
        info->activeBlockSize = config->PFlashTotalSize / config->PFlashBlockCount;
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED
        if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
        {
#if FLASH_SSD_SECONDARY_FLASH_HAS_ITS_OWN_PROTECTION_REGISTER || FLASH_SSD_SE
            /* When required by the command, address bit 23 selects between main flash memory
             * (=0) and secondary flash memory (=1).*/
            info->convertedAddress += 0x800000U;
#endif
```

```
            info->blockWriteUnitSize = FSL_FEATURE_FLASH_PFLASH_1_BLOCK_WRITE_UNIT_SIZE;
        }
        else
#endif /* FLASH_SSD_IS_SECONDARY_FLASH_ENABLED */
        {
            info->blockWriteUnitSize = FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE;
        }

        info->sectorCmdAddressAligment = FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_
        info->sectionCmdAddressAligment = FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS
        info->resourceCmdAddressAligment = FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDR
        info->checkCmdAddressAligment = FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_A
    }

    return kStatus_FLASH_Success;
}

/*! @brief Validates the given user key for flash erase APIs.*/
static status_t flash_check_user_key(uint32_t key)
{
    /* Validate the user key */
    if (key != kFLASH_ApiEraseKey)
    {
        return kStatus_FLASH_EraseKeyError;
    }

    return kStatus_FLASH_Success;
}

#if FLASH_SSD_IS_FLEXNVM_ENABLED
/*! @brief Updates FlexNVM memory partition status according to data flash 0 IFR.*/
static status_t flash_update_flexnvm_memory_partition_status(flash_config_t *config)
{
    struct
    {
        uint32_t reserved0;
        uint8_t FlexNVMPartitionCode;
        uint8_t EEPROMDataSetSize;
        uint16_t reserved1;
    } dataIFRReadOut;
    status_t returnCode;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
    /* Get FlexNVM memory partition info from data flash IFR */
    returnCode = FLASH_ReadResource(config, DFLASH_IFR_READRESOURCE_START_ADDRESS, (u
                                    sizeof(dataIFRReadOut), kFLASH_ResourceOptionFlashIfr);
    if (returnCode != kStatus_FLASH_Success)
    {
        return kStatus_FLASH_PartitionStatusUpdateFailure;
    }
#else
#error "Cannot get FlexNVM memory partition info"
#endif

    /* Fill out partitioned EEPROM size */
    dataIFRReadOut.EEPROMDataSetSize &= 0x0FU;
    switch (dataIFRReadOut.EEPROMDataSetSize)
    {
        case 0x00U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x01U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x02U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x03U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x04U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x05U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x06U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x07U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x08U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x09U:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0AU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0BU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0CU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0DU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0EU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        case 0x0FU:
            config->EEpromTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZ
            break;
        default:
            config->EEpromTotalSize = FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_RESERVED;
            break;
    }

    /* Fill out partitioned DFlash size */
    dataIFRReadOut.FlexNVMPartitionCode &= 0x0FU;
    switch (dataIFRReadOut.FlexNVMPartitionCode)
    {
        case 0x00U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000 */
            break;
        case 0x01U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001 */
            break;
        case 0x02U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010 */
            break;
        case 0x03U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011 */
            break;
        case 0x04U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100 */
            break;
        case 0x05U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
```

```
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101 */
            break;
        case 0x06U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110 */
            break;
        case 0x07U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111 */
            break;
        case 0x08U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000 */
            break;
        case 0x09U:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001 */
            break;
        case 0x0AU:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010 */
            break;
        case 0x0BU:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011 */
            break;
        case 0x0CU:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100 */
            break;
        case 0x0DU:
```

```
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101 */
            break;
        case 0x0EU:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110 */
            break;
        case 0x0FU:
#if (FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111 != 0xFFFFFFFF)
            config->DFlashTotalSize = FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_
#else
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
#endif /* FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111 */
            break;
        default:
            config->DFlashTotalSize = FLEX_NVM_DFLASH_SIZE_FOR_DEPART_RESERVED;
            break;
    }

    return kStatus_FLASH_Success;
}
#endif /* FLASH_SSD_IS_FLEXNVM_ENABLED */

#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
/*! @brief Validates the range of the given resource address.*/
static status_t flash_check_resource_range(uint32_t start,
                                           uint32_t lengthInBytes,
                                           uint32_t alignmentBaseline,
                                           flash_read_resource_option_t option)
{
    status_t status;
    uint32_t maxReadbleAddress;

    if ((start & (alignmentBaseline - 1)) || (lengthInBytes & (alignmentBaseline - 1)))
    {
        return kStatus_FLASH_AlignmentError;
    }

    status = kStatus_FLASH_Success;

    maxReadbleAddress = start + lengthInBytes - 1;
    if (option == kFLASH_ResourceOptionVersionId)
    {
        if ((start != kFLASH_ResourceRangeVersionIdStart) ||
            ((start + lengthInBytes - 1) != kFLASH_ResourceRangeVersionIdEnd))
        {
            status = kStatus_FLASH_InvalidArgument;
        }
    }
    else if (option == kFLASH_ResourceOptionFlashIfr)
    {
        if (maxReadbleAddress < kFLASH_ResourceRangePflashIfrSizeInBytes)
        {
        }
#if defined(FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP) && FSL_FEATURE_FLASH_HAS_P
        else if ((start >= kFLASH_ResourceRangePflashSwapIfrStart) &&
                 (maxReadbleAddress <= kFLASH_ResourceRangePflashSwapIfrEnd))
        {
        }
#endif /* FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP */
        else if ((start >= kFLASH_ResourceRangeDflashIfrStart) &&
                 (maxReadbleAddress <= kFLASH_ResourceRangeDflashIfrEnd))
        {
        }
        else
        {
            status = kStatus_FLASH_InvalidArgument;
        }
    }
    else
    {
        status = kStatus_FLASH_InvalidArgument;
    }

    return status;
}
#endif /* FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD) && FSL_FEATURE_FLASH_HAS_S
/*! @brief Validates the gived swap control option.*/
static status_t flash_check_swap_control_option(flash_swap_control_option_t option)
{
    if ((option == kFLASH_SwapControlOptionIntializeSystem) || (option == kFLASH_SwapControlOptionSe
        (option == kFLASH_SwapControlOptionSetInCompleteState) || (option == kFLASH_SwapControlOpti
        (option == kFLASH_SwapControlOptionDisableSystem))
    {
        return kStatus_FLASH_Success;
    }

    return kStatus_FLASH_InvalidArgument;
}
#endif /* FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD */

#if defined(FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP) && FSL_FEATURE_FLASH_HAS_P
/*! @brief Validates the gived address to see if it is equal to swap indicator address in pflash swap IFR.*/
static status_t flash_validate_swap_indicator_address(flash_config_t *config, uint32_t address)
{
    flash_swap_ifr_field_data_t flashSwapIfrFieldData;
    uint32_t swapIndicatorAddress;
    status_t returnCode;
#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
    returnCode =
        FLASH_ReadResource(config, kFLASH_ResourceRangePflashSwapIfrStart, flashSwapIfrFieldData.fl
                           sizeof(flashSwapIfrFieldData.flashSwapIfrData), kFLASH_ResourceOptionFlashIfr);

    if (returnCode != kStatus_FLASH_Success)
    {
        return returnCode;
    }
#else
    {
        /* From RM, the actual info are stored in FCCOB6,7 */
        uint32_t returnValue[2];
        returnCode = FLASH_ReadOnce(config, kFLASH_RecordIndexSwapAddr, returnValue, 4);
        if (returnCode != kStatus_FLASH_Success)
        {
            return returnCode;
        }
        flashSwapIfrFieldData.flashSwapIfrField.swapIndicatorAddress = (uint16_t)returnValue[0];
        returnCode = FLASH_ReadOnce(config, kFLASH_RecordIndexSwapEnable, returnValue, 4);
        if (returnCode != kStatus_FLASH_Success)
        {
            return returnCode;
        }
        flashSwapIfrFieldData.flashSwapIfrField.swapEnableWord = (uint16_t)returnValue[0];
        returnCode = FLASH_ReadOnce(config, kFLASH_RecordIndexSwapDisable, returnValue, 4);
        if (returnCode != kStatus_FLASH_Success)
        {
            return returnCode;
        }
        flashSwapIfrFieldData.flashSwapIfrField.swapDisableWord = (uint16_t)returnValue[0];
    }
#endif

    /* The high bits value of Swap Indicator Address is stored in Program Flash Swap IFR Field,
     * the low severval bit value of Swap Indicator Address is always 1'b0 */
    swapIndicatorAddress = (uint32_t)flashSwapIfrFieldData.flashSwapIfrField.swapIndicatorAddress *
                           FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT;
    if (address != swapIndicatorAddress)
    {
        return kStatus_FLASH_SwapIndicatorAddressError;
    }

    return returnCode;
}
#endif /* FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP */

#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
/*! @brief Validates the gived flexram function option.*/
static inline status_t flasn_check_flexram_function_option_range(flash_flexram_function_option_t option)
{
    if ((option != kFLASH_FlexramFunctionOptionAvailableAsRam) &&
        (option != kFLASH_FlexramFunctionOptionAvailableForEeprom))
    {
        return kStatus_FLASH_InvalidArgument;
    }

    return kStatus_FLASH_Success;
}
#endif /* FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD */

/*! @brief Gets the flash protection information (region size, region count).*/
static status_t flash_get_protection_info(flash_config_t *config, flash_protection_config_t *info)
{
    uint32_t pflashTotalSize;

    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Clean up info Structure*/
    memset(info, 0, sizeof(flash_protection_config_t));

/* Note: KW40 has a secondary flash, but it doesn't have independent protection register*/
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && (!FLASH_SSD_SECONDARY_FLASH_HAS_I
    pflashTotalSize = FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT * FSL_FEATURE_FLASH_PFLA
                      FSL_FEATURE_FLASH_PFLASH_1_BLOCK_COUNT * FSL_FEATURE_FLASH_PFLASH_
    info->regionBase = FSL_FEATURE_FLASH_PFLASH_START_ADDRESS;
#else
    pflashTotalSize = config->PFlashTotalSize;
    info->regionBase = config->PFlashBlockBase;
#endif

#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && FLASH_SSD_SECONDARY_FLASH_HAS_IT
    if (config->FlashMemoryIndex == (uint8_t)kFLASH_MemoryIndexSecondaryFlash)
    {
        info->regionCount = FSL_FEATURE_FLASH_PFLASH_1_PROTECTION_REGION_COUNT;
    }
    else
#endif
    {
        info->regionCount = FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT;
    }

    /* Calculate the size of the flash protection region
     * If the flash density is > 32KB, then protection region is 1/32 of total flash density
     * Else if flash density is < 32KB, then flash protection region is set to 1KB */
    if (pflashTotalSize > info->regionCount * 1024)
    {
        info->regionSize = (pflashTotalSize) / info->regionCount;
    }
    else
    {
        info->regionSize = 1024;
    }

    return kStatus_FLASH_Success;
}

#if defined(FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL) && FSL_FEATURE_FLASH_HAS_ACC
/*! @brief Gets the flash Execute-Only access information (Segment size, Segment count).*/
static status_t flash_get_access_info(flash_config_t *config, flash_access_config_t *info)
{
    if (config == NULL)
    {
        return kStatus_FLASH_InvalidArgument;
    }

    /* Clean up info Structure*/
    memset(info, 0, sizeof(flash_access_config_t));

/* Note: KW40 has a secondary flash, but it doesn't have independent access register*/
#if FLASH_SSD_IS_SECONDARY_FLASH_ENABLED && (!FLASH_SSD_SECONDARY_FLASH_HAS_I
    info->SegmentBase = FSL_FEATURE_FLASH_PFLASH_START_ADDRESS;
#else
    info->SegmentBase = config->PFlashBlockBase;
#endif
    info->SegmentSize = config->PFlashAccessSegmentSize;
    info->SegmentCount = config->PFlashAccessSegmentCount;

    return kStatus_FLASH_Success;
}
#endif /* FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//*****************************************************************************
// MKL25Z4 startup code for use with MCUXpresso IDE
//
// Version : 050117
//*****************************************************************************
//
// Copyright(C) NXP Semiconductors, 2017
// 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 copyright holder 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" AN
// 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 FO
// 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 (DEBUG)
#pragma GCC push_options
#pragma GCC optimize ("Og")
#endif // (DEBUG)

#if defined (__cplusplus)
#ifdef __REDLIB__
#error Redlib does not support C++
#else
//*****************************************************************************
//
// The entry point for the C++ library startup
//
//*****************************************************************************
extern "C" {
   extern void __libc_init_array(void);
}
#endif
#endif

#define WEAK __attribute__ ((weak))
#define WEAK_AV __attribute__ ((weak, section(".after_vectors")))
#define ALIAS(f) __attribute__ ((weak, alias (#f)))

//*****************************************************************************
#if defined (__cplusplus)
extern "C" {
#endif

//*****************************************************************************
// Flash Configuration block : 16-byte flash configuration field that stores
// default protection settings (loaded on reset) and security information that
// allows the MCU to restrict access to the Flash Memory module.
// Placed at address 0x400 by the linker script.
//*****************************************************************************

__attribute__ ((used,section(".FlashConfig"))) const struct {
   unsigned int word1;
   unsigned int word2;
   unsigned int word3;
   unsigned int word4;
} Flash_Config = {0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFE};

//*****************************************************************************
// Declaration of external SystemInit function
//*****************************************************************************
#if defined (__USE_CMSIS)
extern void SystemInit(void);
#endif // (__USE_CMSIS)

//*****************************************************************************
// Forward declaration of the core exception handlers.
// When the application defines a handler (with the same name), this will
// automatically take precedence over these weak definitions
//*****************************************************************************
     void ResetISR(void);
WEAK void NMI_Handler(void);
WEAK void HardFault_Handler(void);
WEAK void SVC_Handler(void);
WEAK void PendSV_Handler(void);
WEAK void SysTick_Handler(void);
WEAK void IntDefaultHandler(void);

//*****************************************************************************
// Forward declaration of the application IRQ handlers. When the application
// defines a handler (with the same name), this will automatically take
// precedence over weak definitions below
//*****************************************************************************
WEAK void DMA0_IRQHandler(void);
WEAK void DMA1_IRQHandler(void);
WEAK void DMA2_IRQHandler(void);
WEAK void DMA3_IRQHandler(void);
WEAK void Reserved20_IRQHandler(void);
WEAK void FTFA_IRQHandler(void);
WEAK void LVD_LVW_IRQHandler(void);
WEAK void LLWU_IRQHandler(void);
WEAK void I2C0_IRQHandler(void);
WEAK void I2C1_IRQHandler(void);
WEAK void SPI0_IRQHandler(void);
WEAK void SPI1_IRQHandler(void);
WEAK void UART0_IRQHandler(void);
WEAK void UART1_IRQHandler(void);
WEAK void UART2_IRQHandler(void);
WEAK void ADC0_IRQHandler(void);
WEAK void CMP0_IRQHandler(void);
WEAK void TPM0_IRQHandler(void);
WEAK void TPM1_IRQHandler(void);
WEAK void TPM2_IRQHandler(void);
WEAK void RTC_IRQHandler(void);
WEAK void RTC_Seconds_IRQHandler(void);
WEAK void PIT_IRQHandler(void);
WEAK void Reserved39_IRQHandler(void);
WEAK void USB0_IRQHandler(void);
WEAK void DAC0_IRQHandler(void);
WEAK void TSI0_IRQHandler(void);
WEAK void MCG_IRQHandler(void);
WEAK void LPTMR0_IRQHandler(void);
WEAK void Reserved45_IRQHandler(void);
WEAK void PORTA_IRQHandler(void);
WEAK void PORTD_IRQHandler(void);

//*****************************************************************************
// Forward declaration of the driver IRQ handlers. These are aliased
// to the IntDefaultHandler, which is a 'forever' loop. When the driver
// defines a handler (with the same name), this will automatically take
// precedence over these weak definitions
//*****************************************************************************
void DMA0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void DMA1_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void DMA2_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void DMA3_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void Reserved20_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void FTFA_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void LVD_LVW_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void LLWU_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void I2C0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void I2C1_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void SPI0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void SPI1_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void UART0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void UART1_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void UART2_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void ADC0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void CMP0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void TPM0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void TPM1_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void TPM2_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void RTC_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void RTC_Seconds_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void PIT_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void Reserved39_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void USB0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void DAC0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void TSI0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void MCG_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void LPTMR0_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void Reserved45_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void PORTA_DriverIRQHandler(void) ALIAS(IntDefaultHandler);
void PORTD_DriverIRQHandler(void) ALIAS(IntDefaultHandler);

//*****************************************************************************
// The entry point for the application.
// __main() is the entry point for Redlib based applications
// main() is the entry point for Newlib based applications
//*****************************************************************************
#if defined (__REDLIB__)
extern void __main(void);
```

```
#endif
extern int main(void);

//*****************************************************************************
// External declaration for the pointer to the stack top from the Linker Script
//*****************************************************************************
extern void _vStackTop(void);

//*****************************************************************************
#if defined (__cplusplus)
} // extern "C"
#endif

//*****************************************************************************
// The vector table.
// This relies on the linker script to place at correct location in memory.
//*****************************************************************************
extern void (* const g_pfnVectors[])(void);
extern void * __Vectors __attribute__ ((alias ("g_pfnVectors")));

__attribute__ ((used, section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    // Core Level - CM0P
    &_vStackTop,                    // The initial stack pointer
    ResetISR,                       // The reset handler
    NMI_Handler,                    // The NMI handler
    HardFault_Handler,              // The hard fault handler
    0,                              // Reserved
    0,                              // Reserved
    0,                              // Reserved
    0,                              // Reserved
    0,                              // Reserved
    0,                              // Reserved
    0,                              // Reserved
    SVC_Handler,                    // SVCall handler
    0,                              // Reserved
    0,                              // Reserved
    PendSV_Handler,                 // The PendSV handler
    SysTick_Handler,                // The SysTick handler

    // Chip Level - MKL25Z4
    DMA0_IRQHandler,        // 16: DMA channel 0 transfer complete
    DMA1_IRQHandler,        // 17: DMA channel 1 transfer complete
    DMA2_IRQHandler,        // 18: DMA channel 2 transfer complete
    DMA3_IRQHandler,        // 19: DMA channel 3 transfer complete
    Reserved20_IRQHandler,  // 20: Reserved interrupt
    FTFA_IRQHandler,        // 21: Command complete and read collision
    LVD_LVW_IRQHandler,     // 22: Low-voltage detect, low-voltage warning
    LLWU_IRQHandler,        // 23: Low leakage wakeup Unit
    I2C0_IRQHandler,        // 24: I2C0 interrupt
    I2C1_IRQHandler,        // 25: I2C1 interrupt
    SPI0_IRQHandler,        // 26: SPI0 single interrupt vector for all sources
    SPI1_IRQHandler,        // 27: SPI1 single interrupt vector for all sources
    UART0_IRQHandler,        // 28: UART0 status and error
    UART1_IRQHandler,        // 29: UART1 status and error
    UART2_IRQHandler,        // 30: UART2 status and error
    ADC0_IRQHandler,         // 31: ADC0 interrupt
    CMP0_IRQHandler,         // 32: CMP0 interrupt
    TPM0_IRQHandler,         // 33: TPM0 single interrupt vector for all sources
    TPM1_IRQHandler,         // 34: TPM1 single interrupt vector for all sources
    TPM2_IRQHandler,         // 35: TPM2 single interrupt vector for all sources
    RTC_IRQHandler,          // 36: RTC alarm
    RTC_Seconds_IRQHandler,  // 37: RTC seconds
    PIT_IRQHandler,          // 38: PIT interrupt
    Reserved39_IRQHandler,   // 39: Reserved interrupt
    USB0_IRQHandler,         // 40: USB0 interrupt
    DAC0_IRQHandler,         // 41: DAC0 interrupt
    TSI0_IRQHandler,         // 42: TSI0 interrupt
    MCG_IRQHandler,          // 43: MCG interrupt
    LPTMR0_IRQHandler,       // 44: LPTMR0 interrupt
    Reserved45_IRQHandler,   // 45: Reserved interrupt
    PORTA_IRQHandler,        // 46: PORTA Pin detect
    PORTD_IRQHandler,        // 47: PORTD Pin detect
}; /* End of g_pfnVectors */

//*****************************************************************************
// Functions to carry out the initialization of RW and BSS data sections. These
// are written as separate functions rather than being inlined within the
// ResetISR() function in order to cope with MCUs with multiple banks of
// memory.
//*****************************************************************************
__attribute__ ((section(".after_vectors.init_data")))
void data_init(unsigned int romstart, unsigned int start, unsigned int len) {
 unsigned int *pulDest = (unsigned int*) start;
 unsigned int *pulSrc = (unsigned int*) romstart;
 unsigned int loop;
 for (loop = 0; loop < len; loop = loop + 4)
  *pulDest++ = *pulSrc++;
}

__attribute__ ((section(".after_vectors.init_bss")))
void bss_init(unsigned int start, unsigned int len) {
 unsigned int *pulDest = (unsigned int*) start;
 unsigned int loop;
 for (loop = 0; loop < len; loop = loop + 4)
  *pulDest++ = 0;
}

//*****************************************************************************
// The following symbols are constructs generated by the linker, indicating
// the location of various points in the "Global Section Table". This table is
// created by the linker via the Code Red managed linker script mechanism. It
// contains the load address, execution address and length of each RW data
// section and the execution and length of each BSS (zero initialized) section.
//*****************************************************************************
extern unsigned int __data_section_table;
extern unsigned int __data_section_table_end;
extern unsigned int __bss_section_table;
extern unsigned int __bss_section_table_end;

//*****************************************************************************
// Reset entry point for your code.
// Sets up a simple runtime environment and initializes the C/C++
// library.
//*****************************************************************************
__attribute__ ((section(".after_vectors.reset")))
void ResetISR(void) {

    // Disable interrupts
    __asm volatile ("cpsid i");

#if defined (__USE_CMSIS)
// If __USE_CMSIS defined, then call CMSIS SystemInit code
    SystemInit();
#else
    // Disable Watchdog
    // SIM->COPC register: COPT=0,COPCLKS=0,COPW=0
    *((volatile unsigned int *)0x40048100) = 0x00u;
#endif // (__USE_CMSIS)

    //
    // Copy the data sections from flash to SRAM.
    //
 unsigned int LoadAddr, ExeAddr, SectionLen;
 unsigned int *SectionTableAddr;

 // Load base address of Global Section Table
 SectionTableAddr = &__data_section_table;

    // Copy the data sections from flash to SRAM.
 while (SectionTableAddr < &__data_section_table_end) {
  LoadAddr = *SectionTableAddr++;
  ExeAddr = *SectionTableAddr++;
  SectionLen = *SectionTableAddr++;
  data_init(LoadAddr, ExeAddr, SectionLen);
 }

 // At this point, SectionTableAddr = &__bss_section_table;
 // Zero fill the bss segment
 while (SectionTableAddr < &__bss_section_table_end) {
  ExeAddr = *SectionTableAddr++;
  SectionLen = *SectionTableAddr++;
  bss_init(ExeAddr, SectionLen);
 }

#if !defined (__USE_CMSIS)
// Assume that if __USE_CMSIS defined, then CMSIS SystemInit code
// will setup the VTOR register
```

```
    // Check to see if we are running the code from a non-zero
    // address (eg RAM, external flash), in which case we need
    // to modify the VTOR register to tell the CPU that the
    // vector table is located at a non-0x0 address.
    unsigned int * pSCB_VTOR = (unsigned int *) 0xE000ED08;
    if ((unsigned int *)g_pfnVectors!=(unsigned int *) 0x00000000) {
        *pSCB_VTOR = (unsigned int)g_pfnVectors;
    }
#endif // (__USE_CMSIS)

#if defined (__cplusplus)
    //
    // Call C++ library initialisation
    //
    __libc_init_array();
#endif

    // Reenable interrupts
    __asm volatile ("cpsie i");

#if defined (__REDLIB__)
 // Call the Redlib library, which in turn calls main()
 __main();
#else
 main();
#endif

 //
 // main() shouldn't return, but if it does, we'll just enter an infinite loop
 //
 while (1) {
  ;
 }
}

//*****************************************************************************
// Default core exception handlers. Override the ones here by defining your own
// handler routines in your application code.
//*****************************************************************************
WEAK_AV void NMI_Handler(void)
{ while(1) {}
}

WEAK_AV void HardFault_Handler(void)
{ while(1) {}
}

WEAK_AV void SVC_Handler(void)
{ while(1) {}
}

WEAK_AV void PendSV_Handler(void)
{ while(1) {}
```

```
}

WEAK_AV void SysTick_Handler(void)
{ while(1) {}
}

//*****************************************************************************
// Processor ends up here if an unexpected interrupt occurs or a specific
// handler is not present in the application code.
//*****************************************************************************
WEAK_AV void IntDefaultHandler(void)
{ while(1) {}
}

//*****************************************************************************
// Default application exception handlers. Override the ones here by defining
// your own handler routines in your application code. These routines call
// driver exception handlers or IntDefaultHandler() if no driver exception
// handler is included.
//*****************************************************************************
WEAK_AV void DMA0_IRQHandler(void)
{   DMA0_DriverIRQHandler();
}

WEAK_AV void DMA1_IRQHandler(void)
{   DMA1_DriverIRQHandler();
}

WEAK_AV void DMA2_IRQHandler(void)
{   DMA2_DriverIRQHandler();
}

WEAK_AV void DMA3_IRQHandler(void)
{   DMA3_DriverIRQHandler();
}

WEAK_AV void Reserved20_IRQHandler(void)
{   Reserved20_DriverIRQHandler();
}

WEAK_AV void FTFA_IRQHandler(void)
{   FTFA_DriverIRQHandler();
}

WEAK_AV void LVD_LVW_IRQHandler(void)
{   LVD_LVW_DriverIRQHandler();
}

WEAK_AV void LLWU_IRQHandler(void)
{   LLWU_DriverIRQHandler();
}

WEAK_AV void I2C0_IRQHandler(void)
```

```
{   I2C0_DriverIRQHandler();
}

WEAK_AV void I2C1_IRQHandler(void)
{   I2C1_DriverIRQHandler();
}

WEAK_AV void SPI0_IRQHandler(void)
{   SPI0_DriverIRQHandler();
}

WEAK_AV void SPI1_IRQHandler(void)
{   SPI1_DriverIRQHandler();
}

WEAK_AV void UART0_IRQHandler(void)
{   UART0_DriverIRQHandler();
}

WEAK_AV void UART1_IRQHandler(void)
{   UART1_DriverIRQHandler();
}

WEAK_AV void UART2_IRQHandler(void)
{   UART2_DriverIRQHandler();
}

WEAK_AV void ADC0_IRQHandler(void)
{   ADC0_DriverIRQHandler();
}

WEAK_AV void CMP0_IRQHandler(void)
{   CMP0_DriverIRQHandler();
}

WEAK_AV void TPM0_IRQHandler(void)
{   TPM0_DriverIRQHandler();
}

WEAK_AV void TPM1_IRQHandler(void)
{   TPM1_DriverIRQHandler();
}

WEAK_AV void TPM2_IRQHandler(void)
{   TPM2_DriverIRQHandler();
}

WEAK_AV void RTC_IRQHandler(void)
{   RTC_DriverIRQHandler();
}

WEAK_AV void RTC_Seconds_IRQHandler(void)
{   RTC_Seconds_DriverIRQHandler();
}

WEAK_AV void PIT_IRQHandler(void)
{   PIT_DriverIRQHandler();
}

WEAK_AV void Reserved39_IRQHandler(void)
{   Reserved39_DriverIRQHandler();
}

WEAK_AV void USB0_IRQHandler(void)
{   USB0_DriverIRQHandler();
}

WEAK_AV void DAC0_IRQHandler(void)
{   DAC0_DriverIRQHandler();
}

WEAK_AV void TSI0_IRQHandler(void)
{   TSI0_DriverIRQHandler();
}

WEAK_AV void MCG_IRQHandler(void)
{   MCG_DriverIRQHandler();
}

WEAK_AV void LPTMR0_IRQHandler(void)
{   LPTMR0_DriverIRQHandler();
}

WEAK_AV void Reserved45_IRQHandler(void)
{   Reserved45_DriverIRQHandler();
}

WEAK_AV void PORTA_IRQHandler(void)
{   PORTA_DriverIRQHandler();
}

WEAK_AV void PORTD_IRQHandler(void)
{   PORTD_DriverIRQHandler();
}

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

#if defined (DEBUG)
#pragma GCC pop_options
#endif // (DEBUG)
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright 2017-2019 NXP
 * 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 NXP 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" AN
 * 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 FC
 * 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.
 */

/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!GlobalInfo
product: Peripherals v1.0
* BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS **********/

/**
 * @file    peripherals.c
 * @brief   Peripherals initialization file.
 */

/* This is a template for board specific configuration created by MCUXpresso IDE Project Wizard.*/

#include "peripherals.h"

/**
 * @brief Set up and initialize all required blocks and functions related to the peripherals hardware.
 */
void BOARD_InitBootPeripherals(void) {
 /* The user initialization should be placed here */
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*****************************************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits made to this file
 * will be overwritten if the respective MCUXpresso Config Tools is used to update this file.
 ****************************************************************************************************************/
/*
 * How to setup clock using clock driver functions:
 *
```

```
 * 1. CLOCK_SetSimSafeDivs, to make sure core clock, bus clock, flexbus clock
 *    and flash clock are in allowed range during clock mode switch.
 *
 * 2. Call CLOCK_Osc0Init to setup OSC clock, if it is used in target mode.
 *
 * 3. Set MCG configuration, MCG includes three parts: FLL clock, PLL clock and
 *    internal reference clock(MCGIRCLK). Follow the steps to setup:
 *
 *    1). Call CLOCK_BootToXxxMode to set MCG to target mode.
 *
 *    2). If target mode is FBI/BLPI/PBI mode, the MCGIRCLK has been configured
 *        correctly. For other modes, need to call CLOCK_SetInternalRefClkConfig
 *        explicitly to setup MCGIRCLK.
 *
 *    3). Don't need to configure FLL explicitly, because if target mode is FLL
 *        mode, then FLL has been configured by the function CLOCK_BootToXxxMode,
 *        if the target mode is not FLL mode, the FLL is disabled.
 *
 *    4). If target mode is PEE/PBE/PEI/PBI mode, then the related PLL has been
 *        setup by CLOCK_BootToXxxMode. In FBE/FBI/FEE/FBE mode, the PLL could
 *        be enabled independently, call CLOCK_EnablePll0 explicitly in this case.
 *
 * 4. Call CLOCK_SetSimConfig to set the clock configuration in SIM.
 */

/* clang-format off */
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!GlobalInfo
product: Clocks v6.0
processor: MKL25Z128xxx4
package_id: MKL25Z128VLK4
mcu_data: ksdk2_0
processor_version: 6.0.0
board: FRDM-KL25Z
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS **********/
/* clang-format on */

#include "fsl_smc.h"
#include "clock_config.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define MCG_PLL_DISABLE                                 0U  /*!< MCGPLLCLK disabled */
#define OSC_CAP0P                                     0U  /*!< Oscillator 0pF capacitor load */
#define OSC_ER_CLK_DISABLE                               0U  /*!< Disable external reference clock */
#define SIM_OSC32KSEL_LPO_CLK                             3U  /*!< OSC32KSEL select: LPO clock */
#define SIM_PLLFLLSEL_MCGFLLCLK_CLK                         0U  /*!< PLLFLL select: MCGFLLCLK clock */
#define SIM_PLLFLLSEL_MCGPLLCLK_CLK                         1U  /*!< PLLFLL select: MCGPLLCLK clock */

/*******************************************************************************
 * Variables
 ******************************************************************************/
```

```
/* System clock frequency. */
extern uint32_t SystemCoreClock;

/*******************************************************************************
 * Code
 ******************************************************************************/
/*FUNCTION**********************************************************************
 *
 * Function Name : CLOCK_CONFIG_SetFllExtRefDiv
 * Description   : Configure FLL external reference divider (FRDIV).
 * Param frdiv   : The value to set FRDIV.
 *
 *END**************************************************************************/
static void CLOCK_CONFIG_SetFllExtRefDiv(uint8_t frdiv)
{
    MCG->C1 = ((MCG->C1 & ~MCG_C1_FRDIV_MASK) | MCG_C1_FRDIV(frdiv));
}

/*******************************************************************************
 ************************ BOARD_InitBootClocks function ************************
 ******************************************************************************/
void BOARD_InitBootClocks(void)
{
    BOARD_BootClockRUN();
}

/*******************************************************************************
 ********************** Configuration BOARD_BootClockRUN ***********************
 ******************************************************************************/
/* clang-format off */
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!Configuration
name: BOARD_BootClockRUN
called_from_default_init: true
outputs:
- {id: Bus_clock.outFreq, value: 24 MHz}
- {id: Core_clock.outFreq, value: 48 MHz, locked: true, accuracy: '0.001'}
- {id: ERCLK32K.outFreq, value: 1 kHz}
- {id: Flash_clock.outFreq, value: 24 MHz}
- {id: LPO_clock.outFreq, value: 1 kHz}
- {id: MCGIRCLK.outFreq, value: 32.768 kHz}
- {id: OSCERCLK.outFreq, value: 8 MHz}
- {id: PLLFLLCLK.outFreq, value: 48 MHz}
- {id: System_clock.outFreq, value: 48 MHz}
settings:
- {id: MCGMode, value: PEE}
- {id: MCG.FCRDIV.scale, value: '1', locked: true}
- {id: MCG.FRDIV.scale, value: '32'}
- {id: MCG.IREFS.sel, value: MCG.FRDIV}
- {id: MCG.PLLS.sel, value: MCG.PLL}
- {id: MCG.PRDIV.scale, value: '2', locked: true}
- {id: MCG.VDIV.scale, value: '24', locked: true}
- {id: MCG_C1_IRCLKEN_CFG, value: Enabled}
- {id: MCG_C2_OSC_MODE_CFG, value: ModeOscLowPower}
- {id: MCG_C2_RANGE0_CFG, value: High}
- {id: MCG_C2_RANGE0_FRDIV_CFG, value: High}
- {id: OSC0_CR_ERCLKEN_CFG, value: Enabled}
- {id: OSC_CR_ERCLKEN_CFG, value: Enabled}
- {id: SIM.CLKOUTSEL.sel, value: SIM.OUTDIV4}
- {id: SIM.OSC32KSEL.sel, value: PMC.LPOCLK}
- {id: SIM.OUTDIV1.scale, value: '2'}
- {id: SIM.PLLFLLSEL.sel, value: SIM.MCGPLLCLK_DIV2}
- {id: SIM.TPMSRCSEL.sel, value: SIM.PLLFLLSEL}
- {id: SIM.UART0SRCSEL.sel, value: SIM.PLLFLLSEL}
- {id: SIM.USBSRCSEL.sel, value: SIM.PLLFLLSEL}
sources:
- {id: OSC.OSC.outFreq, value: 8 MHz, enabled: true}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS **********/
/* clang-format on */

/*******************************************************************************
 * Variables for BOARD_BootClockRUN configuration
 ******************************************************************************/
const mcg_config_t mcgConfig_BOARD_BootClockRUN =
    {
        .mcgMode = kMCG_ModePEE,                  /* PEE - PLL Engaged External */
        .irclkEnableMode = kMCG_IrclkEnable,      /* MCGIRCLK enabled, MCGIRCLK disabled in STOP mo
        .ircs = kMCG_IrcSlow,                     /* Slow internal reference clock selected */
        .fcrdiv = 0x0U,                           /* Fast IRC divider: divided by 1 */
        .frdiv = 0x0U,                            /* FLL reference clock divider: divided by 32 */
        .drs = kMCG_DrsLow,                       /* Low frequency range */
        .dmx32 = kMCG_Dmx32Default,               /* DCO has a default range of 25% */
        .pll0Config =
            {
                .enableMode = MCG_PLL_DISABLE,    /* MCGPLLCLK disabled */
                .prdiv = 0x1U,                    /* PLL Reference divider: divided by 2 */
                .vdiv = 0x0U,                     /* VCO divider: multiplied by 24 */
            },
    };
const sim_clock_config_t simConfig_BOARD_BootClockRUN =
    {
        .pllFllSel = SIM_PLLFLLSEL_MCGPLLCLK_CLK, /* PLLFLL select: MCGPLLCLK clock */
        .er32kSrc = SIM_OSC32KSEL_LPO_CLK,        /* OSC32KSEL select: LPO clock */
        .clkdiv1 = 0x10010000U,                   /* SIM_CLKDIV1 - OUTDIV1: /2, OUTDIV4: /2 */
    };
const osc_config_t oscConfig_BOARD_BootClockRUN =
    {
        .freq = 8000000U,                         /* Oscillator frequency: 8000000Hz */
        .capLoad = (OSC_CAP0P),                   /* Oscillator capacity load: 0pF */
        .workMode = kOSC_ModeOscLowPower,         /* Oscillator low power */
        .oscerConfig =
            {
                .enableMode = kOSC_ErClkEnable,   /* Enable external reference clock, disable external referen
            }
    };
```

```
/*******************************************************************************
 * Code for BOARD_BootClockRUN configuration
 ******************************************************************************/
void BOARD_BootClockRUN(void)
{
    /* Set the system clock dividers in SIM to safe value. */
    CLOCK_SetSimSafeDivs();
    /* Initializes OSC0 according to board configuration. */
    CLOCK_InitOsc0(&oscConfig_BOARD_BootClockRUN);
    CLOCK_SetXtal0Freq(oscConfig_BOARD_BootClockRUN.freq);
    /* Configure FLL external reference divider (FRDIV). */
    CLOCK_CONFIG_SetFllExtRefDiv(mcgConfig_BOARD_BootClockRUN.frdiv);
    /* Set MCG to PEE mode. */
    CLOCK_BootToPeeMode(kMCG_OscselOsc,
                        kMCG_PllClkSelPll0,
                        &mcgConfig_BOARD_BootClockRUN.pll0Config);
    /* Configure the Internal Reference clock (MCGIRCLK). */
    CLOCK_SetInternalRefClkConfig(mcgConfig_BOARD_BootClockRUN.irclkEnableMode,
                                  mcgConfig_BOARD_BootClockRUN.ircs,
                                  mcgConfig_BOARD_BootClockRUN.fcrdiv);
    /* Set the clock configuration in SIM module. */
    CLOCK_SetSimConfig(&simConfig_BOARD_BootClockRUN);
    /* Set SystemCoreClock variable. */
    SystemCoreClock = BOARD_BOOTCLOCKRUN_CORE_CLOCK;
}

/*******************************************************************************
 ********************* Configuration BOARD_BootClockVLPR ***********************
 ******************************************************************************/
/* clang-format off */
/* TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!Configuration
name: BOARD_BootClockVLPR
outputs:
- {id: Bus_clock.outFreq, value: 800 kHz}
- {id: Core_clock.outFreq, value: 4 MHz}
- {id: ERCLK32K.outFreq, value: 1 kHz}
- {id: Flash_clock.outFreq, value: 800 kHz}
- {id: LPO_clock.outFreq, value: 1 kHz}
- {id: MCGIRCLK.outFreq, value: 4 MHz}
- {id: System_clock.outFreq, value: 4 MHz}
settings:
- {id: MCGMode, value: BLPI}
- {id: powerMode, value: VLPR}
- {id: MCG.CLKS.sel, value: MCG.IRCS}
- {id: MCG.FCRDIV.scale, value: '1', locked: true}
- {id: MCG.FRDIV.scale, value: '32'}
- {id: MCG.IRCS.sel, value: MCG.FCRDIV}
- {id: MCG_C1_IRCLKEN_CFG, value: Enabled}
- {id: MCG_C2_OSC_MODE_CFG, value: ModeOscLowPower}
- {id: MCG_C2_RANGE0_CFG, value: High}
- {id: MCG_C2_RANGE0_FRDIV_CFG, value: High}
- {id: SIM.OSC32KSEL.sel, value: PMC.LPOCLK}
- {id: SIM.OUTDIV4.scale, value: '5'}
sources:
- {id: OSC.OSC.outFreq, value: 8 MHz}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS **********/
/* clang-format on */

/*******************************************************************************
 * Variables for BOARD_BootClockVLPR configuration
 ******************************************************************************/
const mcg_config_t mcgConfig_BOARD_BootClockVLPR =
    {
        .mcgMode = kMCG_ModeBLPI,                 /* BLPI - Bypassed Low Power Internal */
        .irclkEnableMode = kMCG_IrclkEnable,      /* MCGIRCLK enabled, MCGIRCLK disabled in STOP mo
        .ircs = kMCG_IrcFast,                     /* Fast internal reference clock selected */
        .fcrdiv = 0x0U,                           /* Fast IRC divider: divided by 1 */
        .frdiv = 0x0U,                            /* FLL reference clock divider: divided by 32 */
        .drs = kMCG_DrsLow,                       /* Low frequency range */
        .dmx32 = kMCG_Dmx32Default,               /* DCO has a default range of 25% */
        .pll0Config =
            {
                .enableMode = MCG_PLL_DISABLE,    /* MCGPLLCLK disabled */
                .prdiv = 0x0U,                    /* PLL Reference divider: divided by 1 */
                .vdiv = 0x0U,                     /* VCO divider: multiplied by 24 */
            },
    };
const sim_clock_config_t simConfig_BOARD_BootClockVLPR =
    {
        .pllFllSel = SIM_PLLFLLSEL_MCGFLLCLK_CLK, /* PLLFLL select: MCGFLLCLK clock */
        .er32kSrc = SIM_OSC32KSEL_LPO_CLK,        /* OSC32KSEL select: LPO clock */
        .clkdiv1 = 0x40000U,                      /* SIM_CLKDIV1 - OUTDIV1: /1, OUTDIV4: /5 */
    };
const osc_config_t oscConfig_BOARD_BootClockVLPR =
    {
        .freq = 0U,                               /* Oscillator frequency: 0Hz */
        .capLoad = (OSC_CAP0P),                   /* Oscillator capacity load: 0pF */
        .workMode = kOSC_ModeOscLowPower,         /* Oscillator low power */
        .oscerConfig =
            {
                .enableMode = OSC_ER_CLK_DISABLE, /* Disable external reference clock */
            }
    };

/*******************************************************************************
 * Code for BOARD_BootClockVLPR configuration
 ******************************************************************************/
void BOARD_BootClockVLPR(void)
{
    /* Set the system clock dividers in SIM to safe value. */
    CLOCK_SetSimSafeDivs();
    /* Set MCG to BLPI mode. */
    CLOCK_BootToBlpiMode(mcgConfig_BOARD_BootClockVLPR.fcrdiv,
                         mcgConfig_BOARD_BootClockVLPR.ircs,
                         mcgConfig_BOARD_BootClockVLPR.irclkEnableMode);
    /* Set the clock configuration in SIM module. */
    CLOCK_SetSimConfig(&simConfig_BOARD_BootClockVLPR);
    /* Set VLPR power mode. */
    SMC_SetPowerModeProtection(SMC, kSMC_AllowPowerModeAll);
#if (defined(FSL_FEATURE_SMC_HAS_LPWUI) && FSL_FEATURE_SMC_HAS_LPWUI)
    SMC_SetPowerModeVlpr(SMC, false);
#else
    SMC_SetPowerModeVlpr(SMC);
#endif
    while (SMC_GetPowerModeState(SMC) != kSMC_PowerStateVlpr)
    {
    }
    /* Set SystemCoreClock variable. */
    SystemCoreClock = BOARD_BOOTCLOCKVLPR_CORE_CLOCK;
}

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/***********************************************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits made to this file
 * will be overwritten if the respective MCUXpresso Config Tools is used to update this file.
 **********************************************************************************************************************/

/* clang-format off */
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS *************************************
!!GlobalInfo
product: Pins v6.0
processor: MKL25Z128xxx4
package_id: MKL25Z128VLK4
mcu_data: ksdk2_0
processor_version: 6.0.0
board: FRDM-KL25Z
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */
/* clang-format on */

#include "fsl_common.h"
#include "fsl_port.h"
#include "pin_mux.h"
#include "fsl_gpio.h"

/* FUNCTION ************************************************************************************************************
 *
 * Function Name : BOARD_InitBootPins
 * Description   : Calls initialization functions.
 *
 * END ****************************************************************************************************************/
void BOARD_InitBootPins(void)
{
    BOARD_InitPins();
}

/* clang-format off */
```

```
/*
 * TEXT BELOW IS USED AS SETTING FOR TOOLS ***********************************
BOARD_InitPins:
- options: {callFromInitBoot: 'true', coreID: core0, enableClock: 'true'}
- pin_list:
  - {pin_num: '28', peripheral: UART0, signal: TX, pin_signal: TSI0_CH3/PTA2/UART0_TX/TPM2_CH1}
  - {pin_num: '27', peripheral: UART0, signal: RX, pin_signal: TSI0_CH2/PTA1/UART0_RX/TPM2_CH0}
  - {pin_num: '74', peripheral: GPIOD, signal: 'GPIO, 1', pin_signal: ADC0_SE5b/PTD1/SPI0_SCK/TPM0_C
  - {pin_num: '53', peripheral: GPIOB, signal: 'GPIO, 18', pin_signal: TSI0_CH11/PTB18/TPM2_CH0}
  - {pin_num: '54', peripheral: GPIOB, signal: 'GPIO, 19', pin_signal: TSI0_CH12/PTB19/TPM2_CH1}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR TOOLS ***********
 */
/* clang-format on */

/* FUNCTION ************************************************************************************************************
 *
 * Function Name : BOARD_InitPins
 * Description   : Configures pin routing and optionally pin electrical features.
 *
 * END *****************************************************************************************************************
void BOARD_InitPins(void)
{
    /* Port A Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortA);
    /* Port B Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortB);
    /* Port D Clock Gate Control: Clock enabled */
    CLOCK_EnableClock(kCLOCK_PortD);

    /* DEBUG GPIO pin on Port D Pin 7 */
    gpio_pin_config_t gpiod_pin80_config = {
            .pinDirection = kGPIO_DigitalOutput,
            .outputLogic = 1U
        };
        /* Initialize GPIO functionality on pin PTD7 (pin 80)  */
        GPIO_PinInit(GPIOD, 7U, &gpiod_pin80_config);

    /* PORTA1 (pin 27) is configured as UART0_RX */
    PORT_SetPinMux(BOARD_INITPINS_DEBUG_UART_RX_PORT, BOARD_INITPINS_DEBUG_UART_

    /* PORTA2 (pin 28) is configured as UART0_TX */
    PORT_SetPinMux(BOARD_INITPINS_DEBUG_UART_TX_PORT, BOARD_INITPINS_DEBUG_UART_

    /* PORTB18 (pin 53) is configured as PTB18 */
    PORT_SetPinMux(BOARD_INITPINS_LED_RED_PORT, BOARD_INITPINS_LED_RED_PIN, kPORT_

    /* PORTB19 (pin 54) is configured as PTB19 */
    PORT_SetPinMux(BOARD_INITPINS_LED_GREEN_PORT, BOARD_INITPINS_LED_GREEN_PIN, kP

    /* PORTD1 (pin 74) is configured as PTD1 */
    PORT_SetPinMux(BOARD_INITPINS_LED_BLUE_PORT, BOARD_INITPINS_LED_BLUE_PIN, kPOR

    /* PORTD7 (pin 80) is configured as PTD7 */
 PORT_SetPinMux(PORTD, 7U, kPORT_MuxAsGpio);

    SIM->SOPT5 = ((SIM->SOPT5 &
                  /* Mask bits to zero which are setting */
                  (~(SIM_SOPT5_UART0TXSRC_MASK | SIM_SOPT5_UART0RXSRC_MASK)))

                 /* UART0 transmit data source select: UART0_TX pin. */
                 | SIM_SOPT5_UART0TXSRC(SOPT5_UART0TXSRC_UART_TX)

                 /* UART0 receive data source select: UART0_RX pin. */
                 | SIM_SOPT5_UART0RXSRC(SOPT5_UART0RXSRC_UART_RX));
}
/*******************************************************************************************************************
 * EOF
 ******************************************************************************************************************/
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
#include <stdint.h>
#include "board.h"
#include "fsl_debug_console.h"
#include "fsl_common.h"

/*******************************************************************************
 * Variables
```

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

/*****************************************************************************
 * Code
 ****************************************************************************/
/* Initialize debug console. */
void BOARD_InitDebugConsole(void)
{
    uint32_t uartClkSrcFreq;
    /* SIM_SOPT2[27:26]:
     *  00: Clock Disabled
     *  01: IRC48M
     *  10: OSCERCLK
     *  11: MCGIRCCLK
     */
    CLOCK_SetLpsci0Clock(1);

    uartClkSrcFreq = BOARD_DEBUG_UART_CLK_FREQ;
    DbgConsole_Init(BOARD_DEBUG_UART_BASEADDR, BOARD_DEBUG_UART_BAUDRATE, BOARD
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
** ###################################################################
**     Processors:          MKL25Z128VFM4
**                          MKL25Z128VFT4
**                          MKL25Z128VLH4
**                          MKL25Z128VLK4
**                          MKL25Z32VFM4
**                          MKL25Z32VFT4
**                          MKL25Z32VLH4
**                          MKL25Z32VLK4
**                          MKL25Z64VFM4
**                          MKL25Z64VFT4
**                          MKL25Z64VLH4
**                          MKL25Z64VLK4
**
**     Compilers:           Keil ARM C/C++ Compiler
**                          Freescale C/C++ for Embedded ARM
**                          GNU C Compiler
**                          IAR ANSI C/C++ Compiler for ARM
**                          MCUXpresso Compiler
**
**     Reference manual:    KL25P80M48SF0RM, Rev.3, Sep 2012
**     Version:             rev. 2.5, 2015-02-19
**     Build:               b170112
**
**     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) 2016 Freescale Semiconductor, Inc.
**     Copyright 2016 - 2017 NXP
```

```
**     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 the copyright holder 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"
**     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
**     ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGE
**     (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
**     LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
**     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.nxp.com
**     mail:                 support@nxp.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 "fsl_device_registers.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) */

}

/* ----------------------------------------------------------------------------
   -- 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_SHI
      MCGOUTClock = (uint32_t) (CPU_INT_FAST_CLK_HZ / Divider); /* Fast internal reference clock selec
    } /* (!((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) >>
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * This is a modified version of the file printf.c, which was distributed
 * by Motorola as part of the M5407C3BOOT.zip package used to initialize
 * the M5407C3 evaluation board.
 *
 * Copyright:
 *      1999-2000 MOTOROLA, INC. All Rights Reserved.
 *  You are hereby granted a copyright license to use, modify, and
 *  distribute the SOFTWARE so long as this entire notice is
 *  retained without alteration in any modified and/or redistributed
 *  versions, and that such modified versions are clearly identified
 *  as such. No licenses are granted by implication, estoppel or
 *  otherwise under any patents or trademarks of Motorola, Inc. This
 *  software is provided on an "AS IS" basis and without warranty.
 *
 *  To the maximum extent permitted by applicable law, MOTOROLA
 *  DISCLAIMS ALL WARRANTIES WHETHER EXPRESS OR IMPLIED, INCLUDING
 *  IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
 *  PURPOSE AND ANY WARRANTY AGAINST INFRINGEMENT WITH REGARD TO THE
 *  SOFTWARE (INCLUDING ANY MODIFIED VERSIONS THEREOF) AND ANY
 *  ACCOMPANYING WRITTEN MATERIALS.
 *
 *  To the maximum extent permitted by applicable law, IN NO EVENT
 *  SHALL MOTOROLA BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING
 *  WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS
 *  INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY
 *  LOSS) ARISING OF THE USE OR INABILITY TO USE THE SOFTWARE.
 *
```

```
#include <stdarg.h>
#include <stdlib.h>
#if defined(__CC_ARM)
#include <stdio.h>
#endif
#include <math.h>
#include "fsl_debug_console.h"

#if (defined(FSL_FEATURE_SOC_UART_COUNT) && (FSL_FEATURE_SOC_UART_COUNT > 0)) || \
    (defined(FSL_FEATURE_SOC_IUART_COUNT) && (FSL_FEATURE_SOC_IUART_COUNT > 0))
#include "fsl_uart.h"
#endif /* FSL_FEATURE_SOC_UART_COUNT || FSL_FEATURE_SOC_IUART_COUNT */

#if defined(FSL_FEATURE_SOC_LPSCI_COUNT) && (FSL_FEATURE_SOC_LPSCI_COUNT > 0)
#include "fsl_lpsci.h"
#endif /* FSL_FEATURE_SOC_LPSCI_COUNT */

#if defined(FSL_FEATURE_SOC_LPUART_COUNT) && (FSL_FEATURE_SOC_LPUART_COUNT > 0)
#include "fsl_lpuart.h"
#endif /* FSL_FEATURE_SOC_LPUART_COUNT */
```

```
#if defined(FSL_FEATURE_SOC_USB_COUNT) && (FSL_FEATURE_SOC_USB_COUNT > 0) && define
#include "usb_device_config.h"
#include "usb.h"
#include "usb_device_cdc_acm.h"
#include "usb_device_ch9.h"
#include "virtual_com.h"
#endif

#if defined(FSL_FEATURE_SOC_FLEXCOMM_COUNT) && (FSL_FEATURE_SOC_FLEXCOMM_COUN
#include "fsl_usart.h"
#endif /* FSL_FEATURE_SOC_FLEXCOMM_COUNT */

/*! @brief Keil: suppress ellipsis warning in va_arg usage below. */
#if defined(__CC_ARM)
#pragma diag_suppress 1256
#endif /* __CC_ARM */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @brief This definition is maximum line that debugconsole can scanf each time.*/
#define IO_MAXLINE 20U

/*! @brief The overflow value.*/
#ifndef HUGE_VAL
#define HUGE_VAL (99.e99)
#endif /* HUGE_VAL */

#if SCANF_FLOAT_ENABLE
static double fnum = 0.0;
#endif /* SCANF_FLOAT_ENABLE */

/*! @brief Operation functions definitions for debug console. */
typedef struct DebugConsoleOperationFunctions
{
    union
    {
        void (*PutChar)(void *base, const uint8_t *buffer, size_t length);
#if (defined(FSL_FEATURE_SOC_UART_COUNT) && (FSL_FEATURE_SOC_UART_COUNT > 0)) || \
    (defined(FSL_FEATURE_SOC_IUART_COUNT) && (FSL_FEATURE_SOC_IUART_COUNT > 0))
        void (*UART_PutChar)(UART_Type *base, const uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_UART_COUNT || FSL_FEATURE_SOC_IUART_COUNT */
#if defined(FSL_FEATURE_SOC_LPSCI_COUNT) && (FSL_FEATURE_SOC_LPSCI_COUNT > 0)
        void (*LPSCI_PutChar)(UART0_Type *base, const uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_LPSCI_COUNT */
#if defined(FSL_FEATURE_SOC_LPUART_COUNT) && (FSL_FEATURE_SOC_LPUART_COUNT > 0)
        void (*LPUART_PutChar)(LPUART_Type *base, const uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_LPUART_COUNT */
#if defined(FSL_FEATURE_SOC_USB_COUNT) && (FSL_FEATURE_SOC_USB_COUNT > 0) && define
        void (*USB_PutChar)(usb_device_handle base, const uint8_t *buf, size_t count);
#endif /* FSL_FEATURE_SOC_USB_COUNT && BOARD_USE_VIRTUALCOM*/
#if defined(FSL_FEATURE_SOC_FLEXCOMM_COUNT) && (FSL_FEATURE_SOC_FLEXCOMM_COUN
        void (*USART_PutChar)(USART_Type *base, const uint8_t *data, size_t length);
#endif /* FSL_FEATURE_SOC_FLEXCOMM_COUNT */
    } tx_union;
    union
    {
        status_t (*GetChar)(void *base, const uint8_t *buffer, size_t length);
#if (defined(FSL_FEATURE_SOC_UART_COUNT) && (FSL_FEATURE_SOC_UART_COUNT > 0)) || \
    (defined(FSL_FEATURE_SOC_IUART_COUNT) && (FSL_FEATURE_SOC_IUART_COUNT > 0))
        status_t (*UART_GetChar)(UART_Type *base, uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_UART_COUNT || FSL_FEATURE_SOC_IUART_COUNT*/
#if defined(FSL_FEATURE_SOC_LPSCI_COUNT) && (FSL_FEATURE_SOC_LPSCI_COUNT > 0)
        status_t (*LPSCI_GetChar)(UART0_Type *base, uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_LPSCI_COUNT */
#if defined(FSL_FEATURE_SOC_LPUART_COUNT) && (FSL_FEATURE_SOC_LPUART_COUNT > 0)
        status_t (*LPUART_GetChar)(LPUART_Type *base, uint8_t *buffer, size_t length);
#endif /* FSL_FEATURE_SOC_LPUART_COUNT */
#if defined(FSL_FEATURE_SOC_USB_COUNT) && (FSL_FEATURE_SOC_USB_COUNT > 0) && define
        status_t (*USB_GetChar)(usb_device_handle base, uint8_t *buf, size_t count);
#endif /* FSL_FEATURE_SOC_USB_COUNT && BOARD_USE_VIRTUALCOM*/
#if defined(FSL_FEATURE_SOC_FLEXCOMM_COUNT) && (FSL_FEATURE_SOC_FLEXCOMM_COUN
        status_t (*USART_GetChar)(USART_Type *base, uint8_t *data, size_t length);
#endif
    } rx_union;
} debug_console_ops_t;

/*! @brief State structure storing debug console. */
typedef struct DebugConsoleState
{
    uint8_t type;           /*!< Indicator telling whether the debug console is initialized. */
    void *base;             /*!< Base of the IP register. */
    debug_console_ops_t ops; /*!< Operation function pointers for debug UART operations. */
} debug_console_state_t;

/*! @brief Type of KSDK printf function pointer. */
typedef int (*PUTCHAR_FUNC)(int a);

#if PRINTF_ADVANCED_ENABLE
/*! @brief Specification modifier flags for printf. */
enum _debugconsole_printf_flag
{
    kPRINTF_Minus = 0x01U,              /*!< Minus FLag. */
    kPRINTF_Plus = 0x02U,               /*!< Plus Flag. */
    kPRINTF_Space = 0x04U,              /*!< Space Flag. */
    kPRINTF_Zero = 0x08U,               /*!< Zero Flag. */
    kPRINTF_Pound = 0x10U,              /*!< Pound Flag. */
    kPRINTF_LengthChar = 0x20U,         /*!< Length: Char Flag. */
    kPRINTF_LengthShortInt = 0x40U,     /*!< Length: Short Int Flag. */
    kPRINTF_LengthLongInt = 0x80U,      /*!< Length: Long Int Flag. */
    kPRINTF_LengthLongLongInt = 0x100U, /*!< Length: Long Long Int Flag. */
};
#endif /* PRINTF_ADVANCED_ENABLE */

/*! @brief Specification modifier flags for scanf. */
```

```
enum _debugconsole_scanf_flag
{
   kSCANF_Suppress = 0x2U,      /*!< Suppress Flag. */
   kSCANF_DestMask = 0x7cU,     /*!< Destination Mask. */
   kSCANF_DestChar = 0x4U,      /*!< Destination Char Flag. */
   kSCANF_DestString = 0x8U,    /*!< Destination String FLag. */
   kSCANF_DestSet = 0x10U,      /*!< Destination Set Flag. */
   kSCANF_DestInt = 0x20U,      /*!< Destination Int Flag. */
   kSCANF_DestFloat = 0x30U,    /*!< Destination Float Flag. */
   kSCANF_LengthMask = 0x1f00U, /*!< Length Mask Flag. */
#if SCANF_ADVANCED_ENABLE
   kSCANF_LengthChar = 0x100U,        /*!< Length Char Flag. */
   kSCANF_LengthShortInt = 0x200U,    /*!< Length ShortInt Flag. */
   kSCANF_LengthLongInt = 0x400U,     /*!< Length LongInt Flag. */
   kSCANF_LengthLongLongInt = 0x800U, /*!< Length LongLongInt Flag. */
#endif                                /* SCANF_ADVANCED_ENABLE */
#if PRINTF_FLOAT_ENABLE
   kSCANF_LengthLongLongDouble = 0x1000U, /*!< Length LongLongDuoble Flag. */
#endif                                    /*PRINTF_FLOAT_ENABLE */
   kSCANF_TypeSinged = 0x2000U,           /*!< TypeSinged Flag. */
};

/*******************************************************************************
 * Variables
 ******************************************************************************/
/*! @brief Debug UART state information. */
static debug_console_state_t s_debugConsole = {.type = DEBUG_CONSOLE_DEVICE_TYPE_NONE, .ba

/*******************************************************************************
 * Prototypes
 ******************************************************************************/
#if SDK_DEBUGCONSOLE
static int DbgConsole_PrintfFormattedData(PUTCHAR_FUNC func_ptr, const char *fmt, va_list ap);
static int DbgConsole_ScanfFormattedData(const char *line_ptr, char *format, va_list args_ptr);
double modf(double input_dbl, double *intpart_ptr);
#endif /* SDK_DEBUGCONSOLE */

/*******************************************************************************
 * Code
 ******************************************************************************/

/*************Code for DbgConsole Init, Deinit, Printf, Scanf *******************************/

/* See fsl_debug_console.h for documentation of this function. */
status_t DbgConsole_Init(uint32_t baseAddr, uint32_t baudRate, uint8_t device, uint32_t clkSrcFreq)
{
   if (s_debugConsole.type != DEBUG_CONSOLE_DEVICE_TYPE_NONE)
   {
      return kStatus_Fail;
   }

   /* Set debug console to initialized to avoid duplicated initialized operation. */
   s_debugConsole.type = device;
```

```
    /* Switch between different device. */
    switch (device)
    {
#if (defined(FSL_FEATURE_SOC_UART_COUNT) && (FSL_FEATURE_SOC_UART_COUNT > 0)) || \
    (defined(FSL_FEATURE_SOC_IUART_COUNT) && (FSL_FEATURE_SOC_IUART_COUNT > 0))
        case DEBUG_CONSOLE_DEVICE_TYPE_UART:
        case DEBUG_CONSOLE_DEVICE_TYPE_IUART:
        {
            uart_config_t uart_config;
            s_debugConsole.base = (UART_Type *)baseAddr;
            UART_GetDefaultConfig(&uart_config);
            uart_config.baudRate_Bps = baudRate;
            /* Enable clock and initial UART module follow user configure structure. */
            UART_Init(s_debugConsole.base, &uart_config, clkSrcFreq);
            UART_EnableTx(s_debugConsole.base, true);
            UART_EnableRx(s_debugConsole.base, true);
            /* Set the function pointer for send and receive for this kind of device. */
            s_debugConsole.ops.tx_union.UART_PutChar = UART_WriteBlocking;
            s_debugConsole.ops.rx_union.UART_GetChar = UART_ReadBlocking;
        }
        break;
#endif /* FSL_FEATURE_SOC_UART_COUNT */
#if defined(FSL_FEATURE_SOC_LPSCI_COUNT) && (FSL_FEATURE_SOC_LPSCI_COUNT > 0)
        case DEBUG_CONSOLE_DEVICE_TYPE_LPSCI:
        {
            lpsci_config_t lpsci_config;
            s_debugConsole.base = (UART0_Type *)baseAddr;
            LPSCI_GetDefaultConfig(&lpsci_config);
            lpsci_config.baudRate_Bps = baudRate;
            /* Enable clock and initial UART module follow user configure structure. */
            LPSCI_Init(s_debugConsole.base, &lpsci_config, clkSrcFreq);
            LPSCI_EnableTx(s_debugConsole.base, true);
            LPSCI_EnableRx(s_debugConsole.base, true);
            /* Set the function pointer for send and receive for this kind of device. */
            s_debugConsole.ops.tx_union.LPSCI_PutChar = LPSCI_WriteBlocking;
            s_debugConsole.ops.rx_union.LPSCI_GetChar = LPSCI_ReadBlocking;
        }
        break;
#endif /* FSL_FEATURE_SOC_LPSCI_COUNT */
#if defined(FSL_FEATURE_SOC_LPUART_COUNT) && (FSL_FEATURE_SOC_LPUART_COUNT > 0)
        case DEBUG_CONSOLE_DEVICE_TYPE_LPUART:
        {
            lpuart_config_t lpuart_config;
            s_debugConsole.base = (LPUART_Type *)baseAddr;
            LPUART_GetDefaultConfig(&lpuart_config);
            lpuart_config.baudRate_Bps = baudRate;
            /* Enable clock and initial UART module follow user configure structure. */
            LPUART_Init(s_debugConsole.base, &lpuart_config, clkSrcFreq);
            LPUART_EnableTx(s_debugConsole.base, true);
            LPUART_EnableRx(s_debugConsole.base, true);
            /* Set the function pointer for send and receive for this kind of device. */
            s_debugConsole.ops.tx_union.LPUART_PutChar = LPUART_WriteBlocking;
            s_debugConsole.ops.rx_union.LPUART_GetChar = LPUART_ReadBlocking;
        }
        break;
#endif /* FSL_FEATURE_SOC_LPUART_COUNT */
#if defined(FSL_FEATURE_SOC_USB_COUNT) && (FSL_FEATURE_SOC_USB_COUNT > 0) && define
        case DEBUG_CONSOLE_DEVICE_TYPE_USBCDC:
        {
            s_debugConsole.base = USB_VcomInit();
            s_debugConsole.ops.tx_union.USB_PutChar = USB_VcomWriteBlocking;
            s_debugConsole.ops.rx_union.USB_GetChar = USB_VcomReadBlocking;
        }
        break;
#endif /* FSL_FEATURE_SOC_USB_COUNT && BOARD_USE_VIRTUALCOM*/
#if defined(FSL_FEATURE_SOC_FLEXCOMM_COUNT) && (FSL_FEATURE_SOC_FLEXCOMM_COUN
        case DEBUG_CONSOLE_DEVICE_TYPE_FLEXCOMM:
        {
            usart_config_t usart_config;
            s_debugConsole.base = (USART_Type *)baseAddr;
            USART_GetDefaultConfig(&usart_config);
            usart_config.baudRate_Bps = baudRate;
            /* Enable clock and initial UART module follow user configure structure. */
            USART_Init(s_debugConsole.base, &usart_config, clkSrcFreq);
            /* Set the function pointer for send and receive for this kind of device. */
            s_debugConsole.ops.tx_union.USART_PutChar = USART_WriteBlocking;
            s_debugConsole.ops.rx_union.USART_GetChar = USART_ReadBlocking;
        }
        break;
#endif  /* FSL_FEATURE_SOC_FLEXCOMM_COUNT*/
        /* If new device is required as the low level device for debug console,
         * Add the case branch and add the preprocessor macro to judge whether
         * this kind of device exist in this SOC. */
        default:
            /* Device identified is invalid, return invalid device error code. */
            return kStatus_InvalidArgument;
    }

    return kStatus_Success;
}

/* See fsl_debug_console.h for documentation of this function. */
status_t DbgConsole_Deinit(void)
{
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return kStatus_Success;
    }

    switch (s_debugConsole.type)
    {
#if (defined(FSL_FEATURE_SOC_UART_COUNT) && (FSL_FEATURE_SOC_UART_COUNT > 0)) || \
    (defined(FSL_FEATURE_SOC_IUART_COUNT) && (FSL_FEATURE_SOC_IUART_COUNT > 0))
        case DEBUG_CONSOLE_DEVICE_TYPE_UART:
        case DEBUG_CONSOLE_DEVICE_TYPE_IUART:
            /* Disable UART module. */
            UART_Deinit(s_debugConsole.base);
            break;
#endif /* FSL_FEATURE_SOC_UART_COUNT */
#if defined(FSL_FEATURE_SOC_LPSCI_COUNT) && (FSL_FEATURE_SOC_LPSCI_COUNT > 0)
        case DEBUG_CONSOLE_DEVICE_TYPE_LPSCI:
            /* Disable LPSCI module. */
            LPSCI_Deinit(s_debugConsole.base);
            break;
#endif /* FSL_FEATURE_SOC_LPSCI_COUNT */
#if defined(FSL_FEATURE_SOC_LPUART_COUNT) && (FSL_FEATURE_SOC_LPUART_COUNT > 0)
        case DEBUG_CONSOLE_DEVICE_TYPE_LPUART:
            /* Disable LPUART module. */
            LPUART_Deinit(s_debugConsole.base);
            break;
#endif /* FSL_FEATURE_SOC_LPUART_COUNT */
#if defined(FSL_FEATURE_SOC_USB_COUNT) && (FSL_FEATURE_SOC_USB_COUNT > 0) && define
        case DEBUG_CONSOLE_DEVICE_TYPE_USBCDC:
            /* Disable USBCDC module. */
            USB_VcomDeinit(s_debugConsole.base);
            break;
#endif /* FSL_FEATURE_SOC_USB_COUNT && BOARD_USE_VIRTUALCOM*/
#if defined(FSL_FEATURE_SOC_FLEXCOMM_COUNT) && (FSL_FEATURE_SOC_FLEXCOMM_COUN
        case DEBUG_CONSOLE_DEVICE_TYPE_FLEXCOMM:
        {
            USART_Deinit((USART_Type *)s_debugConsole.base);
        }
        break;
#endif /* FSL_FEATURE_SOC_FLEXCOMM_COUNT*/
        default:
            s_debugConsole.type = DEBUG_CONSOLE_DEVICE_TYPE_NONE;
            break;
    }

    /* Device identified is invalid, return invalid device error code. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return kStatus_InvalidArgument;
    }

    s_debugConsole.type = DEBUG_CONSOLE_DEVICE_TYPE_NONE;
    return kStatus_Success;
}

#if SDK_DEBUGCONSOLE
/* See fsl_debug_console.h for documentation of this function. */
int DbgConsole_Printf(const char *fmt_s, ...)
{
    va_list ap;
    int result;

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }
    va_start(ap, fmt_s);
    result = DbgConsole_PrintfFormattedData(DbgConsole_Putchar, fmt_s, ap);
    va_end(ap);

    return result;
}

/* See fsl_debug_console.h for documentation of this function. */
int DbgConsole_Putchar(int ch)
{
    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }
    s_debugConsole.ops.tx_union.PutChar(s_debugConsole.base, (uint8_t *)(&ch), 1);

    return 1;
}

/* See fsl_debug_console.h for documentation of this function. */
int DbgConsole_Scanf(char *fmt_ptr, ...)
{
    /* Plus one to store end of string char */
    char temp_buf[IO_MAXLINE + 1];
    va_list ap;
    int32_t i;
    char result;

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }
    va_start(ap, fmt_ptr);
    temp_buf[0] = '\0';

    for (i = 0; i < IO_MAXLINE; i++)
    {
        temp_buf[i] = result = DbgConsole_Getchar();

        if ((result == '\r') || (result == '\n'))
        {
            /* End of Line. */
            if (i == 0)
            {
                temp_buf[i] = '\0';
                i = -1;
            }
            else
            {
                break;
            }
        }
    }

    if ((i == IO_MAXLINE))
    {
        temp_buf[i] = '\0';
    }
    else
    {
        temp_buf[i + 1] = '\0';
    }
    result = DbgConsole_ScanfFormattedData(temp_buf, fmt_ptr, ap);
    va_end(ap);

    return result;
}

/* See fsl_debug_console.h for documentation of this function. */
int DbgConsole_Getchar(void)
{
    char ch;
    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }
    while (kStatus_Success != s_debugConsole.ops.rx_union.GetChar(s_debugConsole.base, (uint8_t *)(&c
    {
        return -1;
    }

    return ch;
}

/*************Code for process formatted data*******************************/
/*!
 * @brief Scanline function which ignores white spaces.
 *
 * @param[in]   s The address of the string pointer to update.
 * @return      String without white spaces.
 */
static uint32_t DbgConsole_ScanIgnoreWhiteSpace(const char **s)
{
    uint8_t count = 0;
    uint8_t c;

    c = **s;
    while ((c == ' ') || (c == '\t') || (c == '\n') || (c == '\r') || (c == '\v') || (c == '\f'))
    {
        count++;
        (*s)++;
        c = **s;
    }
    return count;
}

/*!
 * @brief This function puts padding character.
 *
 * @param[in] c         Padding character.
 * @param[in] curlen    Length of current formatted string .
 * @param[in] width     Width of expected formatted string.
 * @param[in] count     Number of characters.
 * @param[in] func_ptr  Function to put character out.
 */
static void DbgConsole_PrintfPaddingCharacter(
    char c, int32_t curlen, int32_t width, int32_t *count, PUTCHAR_FUNC func_ptr)
{
    int32_t i;

    for (i = curlen; i < width; i++)
    {
        func_ptr(c);
        (*count)++;
    }
}

/*!
 * @brief Converts a radix number to a string and return its length.
 *
 * @param[in] numstr    Converted string of the number.
 * @param[in] nump      Pointer to the number.
 * @param[in] neg       Polarity of the number.
 * @param[in] radix     The radix to be converted to.
 * @param[in] use_caps  Used to identify %x/X output format.

 * @return Length of the converted string.
 */
static int32_t DbgConsole_ConvertRadixNumToString(char *numstr, void *nump, int32_t neg, int32_t radix
{
#if PRINTF_ADVANCED_ENABLE
    int64_t a;
    int64_t b;
    int64_t c;

    uint64_t ua;
    uint64_t ub;
    uint64_t uc;
#else
    int32_t a;
    int32_t b;
    int32_t c;
    uint32_t ua;
    uint32_t ub;
    uint32_t uc;
#endif /* PRINTF_ADVANCED_ENABLE */

    int32_t nlen;
    char *nstrp;

    nlen = 0;
    nstrp = numstr;
    *nstrp++ = '\0';

    if (neg)
    {
#if PRINTF_ADVANCED_ENABLE
        a = *(int64_t *)nump;
#else
        a = *(int32_t *)nump;
#endif /* PRINTF_ADVANCED_ENABLE */
        if (a == 0)
        {
            *nstrp = '0';
            ++nlen;
            return nlen;
        }
        while (a != 0)
        {
#if PRINTF_ADVANCED_ENABLE
            b = (int64_t)a / (int64_t)radix;
            c = (int64_t)a - ((int64_t)b * (int64_t)radix);
            if (c < 0)
            {
                uc = (uint64_t)c;
                c = (int64_t)(~uc) + 1 + '0';
            }
#else
            b = a / radix;
            c = a - (b * radix);
            if (c < 0)
            {
                uc = (uint32_t)c;
                c = (uint32_t)(~uc) + 1 + '0';
            }
#endif /* PRINTF_ADVANCED_ENABLE */
            else
            {
                c = c + '0';
            }
            a = b;
            *nstrp++ = (char)c;
            ++nlen;
        }
    }
    else
    {
#if PRINTF_ADVANCED_ENABLE
        ua = *(uint64_t *)nump;
#else
        ua = *(uint32_t *)nump;
#endif /* PRINTF_ADVANCED_ENABLE */
        if (ua == 0)
        {
            *nstrp = '0';
            ++nlen;
            return nlen;
        }
        while (ua != 0)
        {
#if PRINTF_ADVANCED_ENABLE
            ub = (uint64_t)ua / (uint64_t)radix;
            uc = (uint64_t)ua - ((uint64_t)ub * (uint64_t)radix);
#else
            ub = ua / (uint32_t)radix;
            uc = ua - (ub * (uint32_t)radix);
#endif /* PRINTF_ADVANCED_ENABLE */

            if (uc < 10)
            {
                uc = uc + '0';
            }
            else
            {
                uc = uc - 10 + (use_caps ? 'A' : 'a');
            }
            ua = ub;
            *nstrp++ = (char)uc;
            ++nlen;
        }
    }
    return nlen;
}

#if PRINTF_FLOAT_ENABLE
/*!
 * @brief Converts a floating radix number to a string and return its length.
 *
 * @param[in] numstr            Converted string of the number.
 * @param[in] nump              Pointer to the number.
 * @param[in] radix             The radix to be converted to.
 * @param[in] precision_width   Specify the precision width.

 * @return Length of the converted string.
 */
static int32_t DbgConsole_ConvertFloatRadixNumToString(char *numstr,
                                                       void *nump,
                                                       int32_t radix,
                                        uint32_t precision_width)
{
    int32_t a;
    int32_t b;
    int32_t c;
    int32_t i;
    uint32_t uc;
    double fa;
    double dc;
    double fb;
    double r;
    double fractpart;
    double intpart;

    int32_t nlen;
    char *nstrp;
    nlen = 0;
    nstrp = numstr;
    *nstrp++ = '\0';
    r = *(double *)nump;
    if (!r)
    {
        *nstrp = '0';
        ++nlen;
        return nlen;
    }
    fractpart = modf((double)r, (double *)&intpart);
    /* Process fractional part. */
    for (i = 0; i < precision_width; i++)
    {
        fractpart *= radix;
    }
    if (r >= 0)
    {
        fa = fractpart + (double)0.5;
        if (fa >= pow(10, precision_width))
        {
            intpart++;
        }
    }
    else
    {
        fa = fractpart - (double)0.5;
        if (fa <= -pow(10, precision_width))
        {
            intpart--;
        }
    }
    for (i = 0; i < precision_width; i++)
    {
        fb = fa / (int32_t)radix;
        dc = (fa - (int64_t)fb * (int32_t)radix);
        c = (int32_t)dc;
        if (c < 0)
        {
            uc = (uint32_t)c;
            c = (int32_t)(~uc) + 1 + '0';
        }
        else
        {
            c = c + '0';
        }
        fa = fb;
        *nstrp++ = (char)c;
        ++nlen;
    }
    *nstrp++ = (char)'.';
    ++nlen;
    a = (int32_t)intpart;
    if (a == 0)
    {
        *nstrp++ = '0';
        ++nlen;
    }
    else
    {
        while (a != 0)
        {
            b = (int32_t)a / (int32_t)radix;
            c = (int32_t)a - ((int32_t)b * (int32_t)radix);
            if (c < 0)
            {
                uc = (uint32_t)c;
                c = (int32_t)(~uc) + 1 + '0';
            }
            else
            {
                c = c + '0';
            }
            a = b;
            *nstrp++ = (char)c;
            ++nlen;
        }
    }
    return nlen;
}
#endif /* PRINTF_FLOAT_ENABLE */

/*!
 * @brief This function outputs its parameters according to a formatted string.
 *
 * @note I/O is performed by calling given function pointer using following
 * (*func_ptr)(c);
 *
 * @param[in] func_ptr  Function to put character out.
 * @param[in] fmt_ptr   Format string for printf.
 * @param[in] args_ptr  Arguments to printf.
 *
 * @return Number of characters
 */
static int DbgConsole_PrintfFormattedData(PUTCHAR_FUNC func_ptr, const char *fmt, va_list ap)
{
    /* va_list ap; */
    char *p;
    int32_t c;

    char vstr[33];
    char *vstrp = NULL;
    int32_t vlen = 0;

    int32_t done;
    int32_t count = 0;

    uint32_t field_width;
    uint32_t precision_width;
    char *sval;
    int32_t cval;
    bool use_caps;
    uint8_t radix = 0;

#if PRINTF_ADVANCED_ENABLE
    uint32_t flags_used;
    int32_t schar, dschar;
    int64_t ival;
    uint64_t uval = 0;
    bool valid_precision_width;
#else
    int32_t ival;
    uint32_t uval = 0;
#endif /* PRINTF_ADVANCED_ENABLE */

#if PRINTF_FLOAT_ENABLE
    double fval;
#endif /* PRINTF_FLOAT_ENABLE */

    /* Start parsing apart the format string and display appropriate formats and data. */
    for (p = (char *)fmt; (c = *p) != 0; p++)
    {
        /*
         * All formats begin with a '%' marker.  Special chars like
         * '\n' or '\t' are normally converted to the appropriate
         * character by the __compiler__.  Thus, no need for this
         * routine to account for the '\' character.
         */
        if (c != '%')
        {
            func_ptr(c);
            count++;
            /* By using 'continue', the next iteration of the loop is used, skipping the code that follows. */
            continue;
        }

        use_caps = true;

#if PRINTF_ADVANCED_ENABLE
        /* First check for specification modifier flags. */
        flags_used = 0;
        done = false;
        while (!done)
        {
            switch (*++p)
            {
                case '-':
                    flags_used |= kPRINTF_Minus;
                    break;
                case '+':
                    flags_used |= kPRINTF_Plus;
                    break;
                case ' ':
                    flags_used |= kPRINTF_Space;
                    break;
                case '0':
                    flags_used |= kPRINTF_Zero;
                    break;
                case '#':
                    flags_used |= kPRINTF_Pound;
                    break;
                default:
                    /* We've gone one char too far. */
                    --p;
                    done = true;
                    break;
            }
        }
#endif /* PRINTF_ADVANCED_ENABLE */

        /* Next check for minimum field width. */
        field_width = 0;
        done = false;
        while (!done)
        {
            c = *++p;
            if ((c >= '0') && (c <= '9'))
            {
                field_width = (field_width * 10) + (c - '0');
            }
#if PRINTF_ADVANCED_ENABLE
            else if (c == '*')
            {
                field_width = (uint32_t)va_arg(ap, uint32_t);
            }
#endif /* PRINTF_ADVANCED_ENABLE */
```

```
            else
            {
                /* We've gone one char too far. */
                --p;
                done = true;
            }
        }
        /* Next check for the width and precision field separator. */
        precision_width = 6;
#if PRINTF_ADVANCED_ENABLE
        valid_precision_width = false;
#endif /* PRINTF_ADVANCED_ENABLE */
        if (*++p == '.')
        {
            /* Must get precision field width, if present. */
            precision_width = 0;
            done = false;
            while (!done)
            {
                c = *++p;
                if ((c >= '0') && (c <= '9'))
                {
                    precision_width = (precision_width * 10) + (c - '0');
#if PRINTF_ADVANCED_ENABLE
                    valid_precision_width = true;
#endif /* PRINTF_ADVANCED_ENABLE */
                }
#if PRINTF_ADVANCED_ENABLE
                else if (c == '*')
                {
                    precision_width = (uint32_t)va_arg(ap, uint32_t);
                    valid_precision_width = true;
                }
#endif /* PRINTF_ADVANCED_ENABLE */
                else
                {
                    /* We've gone one char too far. */
                    --p;
                    done = true;
                }
            }
        }
        else
        {
            /* We've gone one char too far. */
            --p;
        }
#if PRINTF_ADVANCED_ENABLE
        /*
         * Check for the length modifier.
         */
        switch (/* c = */ *++p)
        {
            case 'h':
                if (*++p != 'h')
                {
                    flags_used |= kPRINTF_LengthShortInt;
                    --p;
                }
                else
                {
                    flags_used |= kPRINTF_LengthChar;
                }
                break;
            case 'l':
                if (*++p != 'l')
                {
                    flags_used |= kPRINTF_LengthLongInt;
                    --p;
                }
                else
                {
                    flags_used |= kPRINTF_LengthLongLongInt;
                }
                break;
            default:
                /* we've gone one char too far */
                --p;
                break;
        }
#endif /* PRINTF_ADVANCED_ENABLE */
        /* Now we're ready to examine the format. */
        c = *++p;
        {
            if ((c == 'd') || (c == 'i') || (c == 'f') || (c == 'F') || (c == 'x') || (c == 'X') || (c == 'o') ||
                (c == 'b') || (c == 'p') || (c == 'u'))
            {
                if ((c == 'd') || (c == 'i'))
                {
#if PRINTF_ADVANCED_ENABLE
                    if (flags_used & kPRINTF_LengthLongLongInt)
                    {
                        ival = (int64_t)va_arg(ap, int64_t);
                    }
                    else
#endif /* PRINTF_ADVANCED_ENABLE */
                    {
                        ival = (int32_t)va_arg(ap, int32_t);
                    }
                    vlen = DbgConsole_ConvertRadixNumToString(vstr, &ival, true, 10, use_caps);
                    vstrp = &vstr[vlen];
#if PRINTF_ADVANCED_ENABLE
                    if (ival < 0)
                    {
                        schar = '-';
                        ++vlen;
```

```
}
else
{
    if (flags_used & kPRINTF_Plus)
    {
        schar = '+';
        ++vlen;
    }
    else
    {
        if (flags_used & kPRINTF_Space)
        {
            schar = ' ';
            ++vlen;
        }
        else
        {
            schar = 0;
        }
    }
}
dschar = false;
/* Do the ZERO pad. */
if (flags_used & kPRINTF_Zero)
{
    if (schar)
    {
        func_ptr(schar);
        count++;
    }
    dschar = true;

    DbgConsole_PrintfPaddingCharacter('0', vlen, field_width, &count, func_ptr);
    vlen = field_width;
}
else
{
    if (!(flags_used & kPRINTF_Minus))
    {
        DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
        if (schar)
        {
            func_ptr(schar);
            count++;
        }
        dschar = true;
    }
}
/* The string was built in reverse order, now display in correct order. */
if ((!dschar) && schar)
{
    func_ptr(schar);
    count++;
```

```
                }
#endif /* PRINTF_ADVANCED_ENABLE */
            }

#if PRINTF_FLOAT_ENABLE
            if ((c == 'f') || (c == 'F'))
            {
                fval = (double)va_arg(ap, double);
                vlen = DbgConsole_ConvertFloatRadixNumToString(vstr, &fval, 10, precision_width);
                vstrp = &vstr[vlen];

#if PRINTF_ADVANCED_ENABLE
                if (fval < 0)
                {
                    schar = '-';
                    ++vlen;
                }
                else
                {
                    if (flags_used & kPRINTF_Plus)
                    {
                        schar = '+';
                        ++vlen;
                    }
                    else
                    {
                        if (flags_used & kPRINTF_Space)
                        {
                            schar = ' ';
                            ++vlen;
                        }
                        else
                        {
                            schar = 0;
                        }
                    }
                }
                dschar = false;
                if (flags_used & kPRINTF_Zero)
                {
                    if (schar)
                    {
                        func_ptr(schar);
                        count++;
                    }
                    dschar = true;
                    DbgConsole_PrintfPaddingCharacter('0', vlen, field_width, &count, func_ptr);
                    vlen = field_width;
                }
                else
                {
                    if (!(flags_used & kPRINTF_Minus))
                    {
                        DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
                        if (schar)
                        {
                            func_ptr(schar);
                            count++;
                        }
                        dschar = true;
                    }
                }
                if ((!dschar) && schar)
                {
                    func_ptr(schar);
                    count++;
                }
#endif /* PRINTF_ADVANCED_ENABLE */
                }
#endif /* PRINTF_FLOAT_ENABLE */
                if ((c == 'X') || (c == 'x'))
                {
                if (c == 'x')
                {
                    use_caps = false;
                }
#if PRINTF_ADVANCED_ENABLE
                if (flags_used & kPRINTF_LengthLongLongInt)
                {
                    uval = (uint64_t)va_arg(ap, uint64_t);
                }
                else
#endif /* PRINTF_ADVANCED_ENABLE */
                {
                    uval = (uint32_t)va_arg(ap, uint32_t);
                }
                vlen = DbgConsole_ConvertRadixNumToString(vstr, &uval, false, 16, use_caps);
                vstrp = &vstr[vlen];

#if PRINTF_ADVANCED_ENABLE
                dschar = false;
                if (flags_used & kPRINTF_Zero)
                {
                    if (flags_used & kPRINTF_Pound)
                    {
                        func_ptr('0');
                        func_ptr((use_caps ? 'X' : 'x'));
                        count += 2;
                        /*vlen += 2;*/
                        dschar = true;
                    }
                    DbgConsole_PrintfPaddingCharacter('0', vlen, field_width, &count, func_ptr);
                    vlen = field_width;
                }
                else
                {
```

```
                    if (!(flags_used & kPRINTF_Minus))
                    {
                        if (flags_used & kPRINTF_Pound)
                        {
                            vlen += 2;
                        }
                        DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
                        if (flags_used & kPRINTF_Pound)
                        {
                            func_ptr('0');
                            func_ptr(use_caps ? 'X' : 'x');
                            count += 2;

                            dschar = true;
                        }
                    }
                }

                if ((flags_used & kPRINTF_Pound) && (!dschar))
                {
                    func_ptr('0');
                    func_ptr(use_caps ? 'X' : 'x');
                    count += 2;
                    vlen += 2;
                }
#endif /* PRINTF_ADVANCED_ENABLE */
            }
            if ((c == 'o') || (c == 'b') || (c == 'p') || (c == 'u'))
            {
#if PRINTF_ADVANCED_ENABLE
                if (flags_used & kPRINTF_LengthLongLongInt)
                {
                    uval = (uint64_t)va_arg(ap, uint64_t);
                }
                else
#endif /* PRINTF_ADVANCED_ENABLE */
                {
                    uval = (uint32_t)va_arg(ap, uint32_t);
                }
                switch (c)
                {
                    case 'o':
                        radix = 8;
                        break;
                    case 'b':
                        radix = 2;
                        break;
                    case 'p':
                        radix = 16;
                        break;
                    case 'u':
                        radix = 10;
                        break;
                    default:
                        break;
                }
                vlen = DbgConsole_ConvertRadixNumToString(vstr, &uval, false, radix, use_caps);
                vstrp = &vstr[vlen];
#if PRINTF_ADVANCED_ENABLE
                if (flags_used & kPRINTF_Zero)
                {
                    DbgConsole_PrintfPaddingCharacter('0', vlen, field_width, &count, func_ptr);
                    vlen = field_width;
                }
                else
                {
                    if (!(flags_used & kPRINTF_Minus))
                    {
                        DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
                    }
                }
#endif /* PRINTF_ADVANCED_ENABLE */
            }
#if !PRINTF_ADVANCED_ENABLE
            DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
#endif /* !PRINTF_ADVANCED_ENABLE */
            if (vstrp != NULL)
            {
                while (*vstrp)
                {
                    func_ptr(*vstrp--);
                    count++;
                }
            }
#if PRINTF_ADVANCED_ENABLE
            if (flags_used & kPRINTF_Minus)
            {
                DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
            }
#endif /* PRINTF_ADVANCED_ENABLE */
        }
        else if (c == 'c')
        {
            cval = (char)va_arg(ap, uint32_t);
            func_ptr(cval);
            count++;
        }
        else if (c == 's')
        {
            sval = (char *)va_arg(ap, char *);
            if (sval)
            {
#if PRINTF_ADVANCED_ENABLE
                if (valid_precision_width)
                {
                    vlen = precision_width;
                }
                else
                {
                    vlen = strlen(sval);
                }
#else
                vlen = strlen(sval);
#endif /* PRINTF_ADVANCED_ENABLE */
#if PRINTF_ADVANCED_ENABLE
                if (!(flags_used & kPRINTF_Minus))
#endif /* PRINTF_ADVANCED_ENABLE */
                {
                    DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
                }

#if PRINTF_ADVANCED_ENABLE
                if (valid_precision_width)
                {
                    while ((*sval) && (vlen > 0))
                    {
                        func_ptr(*sval++);
                        count++;
                        vlen--;
                    }
                    /* In case that vlen sval is shorter than vlen */
                    vlen = precision_width - vlen;
                }
                else
                {
#endif /* PRINTF_ADVANCED_ENABLE */
                    while (*sval)
                    {
                        func_ptr(*sval++);
                        count++;
                    }
#if PRINTF_ADVANCED_ENABLE
                }
#endif /* PRINTF_ADVANCED_ENABLE */

#if PRINTF_ADVANCED_ENABLE
                if (flags_used & kPRINTF_Minus)
                {
                    DbgConsole_PrintfPaddingCharacter(' ', vlen, field_width, &count, func_ptr);
                }
#endif /* PRINTF_ADVANCED_ENABLE */
            }
        }
        else
        {
            func_ptr(c);
            count++;
        }
    }
    }
    return count;
}

/*!
 * @brief Converts an input line of ASCII characters based upon a provided
 * string format.
 *
 * @param[in] line_ptr The input line of ASCII data.
 * @param[in] format   Format first points to the format string.
 * @param[in] args_ptr The list of parameters.
 *
 * @return Number of input items converted and assigned.
 * @retval IO_EOF When line_ptr is empty string "".
 */
static int DbgConsole_ScanfFormattedData(const char *line_ptr, char *format, va_list args_ptr)
{
    uint8_t base;
    int8_t neg;
    /* Identifier for the format string. */
    char *c = format;
    char temp;
    char *buf;
    /* Flag telling the conversion specification. */
    uint32_t flag = 0;
    /* Filed width for the matching input streams. */
    uint32_t field_width;
    /* How many arguments are assigned except the suppress. */
    uint32_t nassigned = 0;
    /* How many characters are read from the input streams. */
    uint32_t n_decode = 0;

    int32_t val;

    const char *s;
    /* Identifier for the input string. */
    const char *p = line_ptr;

    /* Return EOF error before any conversion. */
    if (*p == '\0')
    {
        return -1;
    }

    /* Decode directives. */
    while ((*c) && (*p))
    {
        /* Ignore all white-spaces in the format strings. */
        if (DbgConsole_ScanIgnoreWhiteSpace((const char **)&c))
        {
            n_decode += DbgConsole_ScanIgnoreWhiteSpace(&p);
        }
        else if ((*c != '%') || ((*c == '%') && (*(c + 1) == '%')))
        {
            /* Ordinary characters. */
            c++;
            if (*p == *c)
            {
                n_decode++;
                p++;
                c++;
            }
            else
            {
                /* Match failure. Misalignment with C99, the unmatched characters need to be pushed back to st
                 * However, it is deserted now. */
                break;
            }
        }
        else
        {
            /* convernsion specification */
            c++;
            /* Reset. */
            flag = 0;
            field_width = 0;
            base = 0;

            /* Loop to get full conversion specification. */
            while ((*c) && (!(flag & kSCANF_DestMask)))
            {
                switch (*c)
                {
#if SCANF_ADVANCED_ENABLE
                    case '*':
                        if (flag & kSCANF_Suppress)
                        {
                            /* Match failure. */
                            return nassigned;
                        }
                        flag |= kSCANF_Suppress;
                        c++;
                        break;
                    case 'h':
                        if (flag & kSCANF_LengthMask)
                        {
                            /* Match failure. */
                            return nassigned;
                        }

                        if (c[1] == 'h')
                        {
                            flag |= kSCANF_LengthChar;
                            c++;
                        }
                        else
                        {
                            flag |= kSCANF_LengthShortInt;
                        }
                        c++;
                        break;
                    case 'l':
                        if (flag & kSCANF_LengthMask)
                        {
                            /* Match failure. */
                            return nassigned;
                        }

                        if (c[1] == 'l')
                        {
                            flag |= kSCANF_LengthLongLongInt;
                            c++;
                        }
                        else
                        {
                            flag |= kSCANF_LengthLongInt;
                        }
                        c++;
                        break;
#endif /* SCANF_ADVANCED_ENABLE */
#if SCANF_FLOAT_ENABLE
                    case 'L':
                        if (flag & kSCANF_LengthMask)
                        {
                            /* Match failure. */
                            return nassigned;
                        }
                        flag |= kSCANF_LengthLongLongDouble;
                        c++;
                        break;
#endif /* SCANF_FLOAT_ENABLE */
                    case '0':
                    case '1':
                    case '2':
                    case '3':
                    case '4':
                    case '5':
                    case '6':
                    case '7':
                    case '8':
                    case '9':
                        if (field_width)
                        {
                            /* Match failure. */
                            return nassigned;
                        }
                        do
                        {
                            field_width = field_width * 10 + *c - '0';
                            c++;
                        } while ((*c >= '0') && (*c <= '9'));
                        break;
                    case 'd':
                        base = 10;
                        flag |= kSCANF_TypeSinged;
                        flag |= kSCANF_DestInt;
                        c++;
                        break;
                    case 'u':
                        base = 10;
                        flag |= kSCANF_DestInt;
                        c++;
                        break;
                    case 'o':
                        base = 8;
                        flag |= kSCANF_DestInt;
                        c++;
                        break;
                    case 'x':
                    case 'X':
                        base = 16;
                        flag |= kSCANF_DestInt;
                        c++;
                        break;
                    case 'i':
                        base = 0;
                        flag |= kSCANF_DestInt;
                        c++;
                        break;
#if SCANF_FLOAT_ENABLE
                    case 'a':
                    case 'A':
                    case 'e':
                    case 'E':
                    case 'f':
                    case 'F':
                    case 'g':
                    case 'G':
                        flag |= kSCANF_DestFloat;
                        c++;
                        break;
#endif /* SCANF_FLOAT_ENABLE */
                    case 'c':
                        flag |= kSCANF_DestChar;
                        if (!field_width)
                        {
                            field_width = 1;
                        }
                        c++;
                        break;
                    case 's':
                        flag |= kSCANF_DestString;
                    c++;
                    break;
                default:
                    return nassigned;
            }
        }

        if (!(flag & kSCANF_DestMask))
        {
            /* Format strings are exhausted. */
            return nassigned;
        }

        if (!field_width)
        {
            /* Large than length of a line. */
            field_width = 99;
        }

        /* Matching strings in input streams and assign to argument. */
        switch (flag & kSCANF_DestMask)
        {
            case kSCANF_DestChar:
                s = (const char *)p;
                buf = va_arg(args_ptr, char *);
                while ((field_width--) && (*p))
                {
                    if (!(flag & kSCANF_Suppress))
                    {
                        *buf++ = *p++;
                    }
                    else
                    {
                        p++;
                    }
                    n_decode++;
                }

                if ((!(flag & kSCANF_Suppress)) && (s != p))
                {
                    nassigned++;
                }
                break;
            case kSCANF_DestString:
                n_decode += DbgConsole_ScanIgnoreWhiteSpace(&p);
                s = p;
                buf = va_arg(args_ptr, char *);
                while ((field_width--) && (*p != '\0') && (*p != ' ') && (*p != '\t') && (*p != '\n') &&
                       (*p != '\r') && (*p != '\v') && (*p != '\f'))
                {
                    if (flag & kSCANF_Suppress)
                    {
                        p++;
                    }
                    else
                    {
                        *buf++ = *p++;
                    }
                    n_decode++;
                }

                if ((!(flag & kSCANF_Suppress)) && (s != p))
                {
                    /* Add NULL to end of string. */
                    *buf = '\0';
                    nassigned++;
                }
                break;
            case kSCANF_DestInt:
                n_decode += DbgConsole_ScanIgnoreWhiteSpace(&p);
                s = p;
                val = 0;
                if ((base == 0) || (base == 16))
                {
                    if ((s[0] == '0') && ((s[1] == 'x') || (s[1] == 'X')))
                    {
                        base = 16;
                        if (field_width >= 1)
                        {
                            p += 2;
                            n_decode += 2;
                            field_width -= 2;
                        }
                    }
                }

                if (base == 0)
                {
                    if (s[0] == '0')
                    {
                        base = 8;
                    }
                    else
                    {
                        base = 10;
                    }
                }

                neg = 1;
                switch (*p)
                {
                    case '-':
                        neg = -1;
                        n_decode++;
                        p++;
                        field_width--;
```

```
                        break;
                    case '+':
                        neg = 1;
                        n_decode++;
                        p++;
                        field_width--;
                        break;
                    default:
                        break;
                }

                while ((*p) && (field_width--))
                {
                    if ((*p <= '9') && (*p >= '0'))
                    {
                        temp = *p - '0';
                    }
                    else if ((*p <= 'f') && (*p >= 'a'))
                    {
                        temp = *p - 'a' + 10;
                    }
                    else if ((*p <= 'F') && (*p >= 'A'))
                    {
                        temp = *p - 'A' + 10;
                    }
                    else
                    {
                        temp = base;
                    }

                    if (temp >= base)
                    {
                        break;
                    }
                    else
                    {
                        val = base * val + temp;
                    }
                    p++;
                    n_decode++;
                }
                val *= neg;
                if (!(flag & kSCANF_Suppress))
                {
#if SCANF_ADVANCED_ENABLE
                    switch (flag & kSCANF_LengthMask)
                    {
                        case kSCANF_LengthChar:
                            if (flag & kSCANF_TypeSinged)
                            {
                                *va_arg(args_ptr, signed char *) = (signed char)val;
                            }
                            else
                    {
                        *va_arg(args_ptr, unsigned char *) = (unsigned char)val;
                    }
                    break;
                case kSCANF_LengthShortInt:
                    if (flag & kSCANF_TypeSinged)
                    {
                        *va_arg(args_ptr, signed short *) = (signed short)val;
                    }
                    else
                    {
                        *va_arg(args_ptr, unsigned short *) = (unsigned short)val;
                    }
                    break;
                case kSCANF_LengthLongInt:
                    if (flag & kSCANF_TypeSinged)
                    {
                        *va_arg(args_ptr, signed long int *) = (signed long int)val;
                    }
                    else
                    {
                        *va_arg(args_ptr, unsigned long int *) = (unsigned long int)val;
                    }
                    break;
                case kSCANF_LengthLongLongInt:
                    if (flag & kSCANF_TypeSinged)
                    {
                        *va_arg(args_ptr, signed long long int *) = (signed long long int)val;
                    }
                    else
                    {
                        *va_arg(args_ptr, unsigned long long int *) = (unsigned long long int)val;
                    }
                    break;
                default:
                    /* The default type is the type int. */
                    if (flag & kSCANF_TypeSinged)
                    {
                        *va_arg(args_ptr, signed int *) = (signed int)val;
                    }
                    else
                    {
                        *va_arg(args_ptr, unsigned int *) = (unsigned int)val;
                    }
                    break;
            }
#else
            /* The default type is the type int. */
            if (flag & kSCANF_TypeSinged)
            {
                *va_arg(args_ptr, signed int *) = (signed int)val;
            }
            else
```

```
                    {
                        *va_arg(args_ptr, unsigned int *) = (unsigned int)val;
                    }
#endif /* SCANF_ADVANCED_ENABLE */
                    nassigned++;
                }
                break;
#if SCANF_FLOAT_ENABLE
            case kSCANF_DestFloat:
                n_decode += DbgConsole_ScanIgnoreWhiteSpace(&p);
                fnum = strtod(p, (char **)&s);

                if ((fnum >= HUGE_VAL) || (fnum <= -HUGE_VAL))
                {
                    break;
                }

                n_decode += (int)(s) - (int)(p);
                p = s;
                if (!(flag & kSCANF_Suppress))
                {
                    if (flag & kSCANF_LengthLongLongDouble)
                    {
                        *va_arg(args_ptr, double *) = fnum;
                    }
                    else
                    {
                        *va_arg(args_ptr, float *) = (float)fnum;
                    }
                    nassigned++;
                }
                break;
#endif /* SCANF_FLOAT_ENABLE */
            default:
                return nassigned;
        }
    }
  }
  return nassigned;
}
#endif /* SDK_DEBUGCONSOLE */
/*************Code to support toolchain's printf, scanf *******************************/
/* These function __write and __read is used to support IAR toolchain to printf and scanf*/
#if (defined(__ICCARM__))
#pragma weak __write
size_t __write(int handle, const unsigned char *buffer, size_t size)
{
  if (buffer == 0)
  {
    /*
     * This means that we should flush internal buffers.  Since we don't we just return.
     * (Remember, "handle" == -1 means that all handles should be flushed.)
     */
```

```
        return 0;
    }

    /* This function only writes to "standard out" and "standard err" for all other file handles it returns failure.
    if ((handle != 1) && (handle != 2))
    {
        return ((size_t)-1);
    }

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return ((size_t)-1);
    }

    /* Send data. */
    s_debugConsole.ops.tx_union.PutChar(s_debugConsole.base, buffer, 1);
    return size;
}

#pragma weak __read
size_t __read(int handle, unsigned char *buffer, size_t size)
{
    /* This function only reads from "standard in", for all other file  handles it returns failure. */
    if (handle != 0)
    {
        return ((size_t)-1);
    }

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return ((size_t)-1);
    }

    /* Receive data. */
    s_debugConsole.ops.rx_union.GetChar(s_debugConsole.base, buffer, size);

    return size;
}

/* support LPC Xpresso with RedLib */
#elif(defined(__REDLIB__))

#if (!SDK_DEBUGCONSOLE) && (defined(SDK_DEBUGCONSOLE_UART))
int __attribute__((weak)) __sys_write(int handle, char *buffer, int size)
{
    if (buffer == 0)
    {
        /* return -1 if error. */
        return -1;
    }
    /* This function only writes to "standard out" and "standard err" for all other file handles it returns failure.
    if ((handle != 1) && (handle != 2))
    {
        return -1;
    }

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Send data. */
    s_debugConsole.ops.tx_union.PutChar(s_debugConsole.base, (uint8_t *)buffer, size);
    return 0;
}

int __attribute__((weak)) __sys_readc(void)
{
    char tmp;
    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Receive data. */
    s_debugConsole.ops.rx_union.GetChar(s_debugConsole.base, (uint8_t *)&tmp, sizeof(tmp));
    return tmp;
}
#endif

/* These function __write and __read is used to support ARM_GCC, KDS, Atollic toolchains to printf and sc
#elif(defined(__GNUC__))

#if ((defined(__GNUC__) && (!defined(__MCUXPRESSO))) || \
     (defined(__MCUXPRESSO) && (!SDK_DEBUGCONSOLE) && (defined(SDK_DEBUGCONSOLE_UAF

int __attribute__((weak)) _write(int handle, char *buffer, int size)
{
    if (buffer == 0)
    {
        /* return -1 if error. */
        return -1;
    }

    /* This function only writes to "standard out" and "standard err" for all other file handles it returns failure.
    if ((handle != 1) && (handle != 2))
    {
        return -1;
    }

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Send data. */
    s_debugConsole.ops.tx_union.PutChar(s_debugConsole.base, (uint8_t *)buffer, size);
    return size;
}

int __attribute__((weak)) _read(int handle, char *buffer, int size)
{
    /* This function only reads from "standard in", for all other file handles it returns failure. */
    if (handle != 0)
    {
        return -1;
    }

    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Receive data. */
    s_debugConsole.ops.rx_union.GetChar(s_debugConsole.base, (uint8_t *)buffer, size);
    return size;
}
#endif

/* These function fputc and fgetc is used to support KEIL toolchain to printf and scanf*/
#elif defined(__CC_ARM)
struct __FILE
{
    int handle;
    /*
     * Whatever you require here. If the only file you are using is standard output using printf() for debugging
     * no file handling is required.
     */
};

/* FILE is typedef in stdio.h. */
#pragma weak __stdout
#pragma weak __stdin
FILE __stdout;
FILE __stdin;

#pragma weak fputc
int fputc(int ch, FILE *f)
{
    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Send data. */
    s_debugConsole.ops.tx_union.PutChar(s_debugConsole.base, (uint8_t *)(&ch), 1);
    return 1;
}

#pragma weak fgetc
int fgetc(FILE *f)
{
    char ch;
    /* Do nothing if the debug UART is not initialized. */
    if (s_debugConsole.type == DEBUG_CONSOLE_DEVICE_TYPE_NONE)
    {
        return -1;
    }

    /* Receive data. */
    s_debugConsole.ops.rx_union.GetChar(s_debugConsole.base, (uint8_t *)(&ch), 1);
    return ch;
}
#endif /* __ICCARM__ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// ****************************************************************************
// semihost_hardfault.c
//              - Provides hard fault handler to allow semihosting code not
//                to hang application when debugger not connected.
//
// ****************************************************************************
// Copyright 2017-2019 NXP
// All rights reserved.
//
// Software that is described herein is for illustrative purposes only
// which provides customers with programming information regarding the
// NXP Cortex-M based MCUs. This software is supplied "AS IS" without any
// warranties of any kind, and NXP Semiconductors and its licensor disclaim any
// and all warranties, express or implied, including all implied warranties of
// merchantability, fitness for a particular purpose and non-infringement of
// intellectual property rights.  NXP Semiconductors assumes no responsibility
// or liability for the use of the software, conveys no license or rights under
// any patent, copyright, mask work right, or any other intellectual property
// rights in or to any products. NXP Semiconductors reserves the right to make
// changes in the software without notification. NXP Semiconductors also makes
// no representation or warranty that such application will be suitable for the
// specified use without further testing or modification.
//
// Permission to use, copy, modify, and distribute this software and its
// documentation is hereby granted, under NXP Semiconductors' and its
// licensor's relevant copyrights in the software, without fee, provided that it
// is used in conjunction with NXP Semiconductors microcontrollers.  This
// copyright, permission, and disclaimer notice must appear in all copies of
// this code.
```

```
// *****************************************************************************
//
//                    ===== DESCRIPTION =====
//
// One of the issues with applications that make use of semihosting operations
// (such as printf calls) is that the code will not execute correctly when the
// debugger is not connected. Generally this will show up with the application
// appearing to just hang. This may include the application running from reset
// or powering up the board (with the application already in FLASH), and also
// as the application failing to continue to execute after a debug session is
// terminated.
//
// The problem here is that the "bottom layer" of the semihosted variants of
// the C library, semihosting is implemented by a "BKPT 0xAB" instruction.
// When the debug tools are not connected, this instruction triggers a hard
// fault - and the default hard fault handler within an application will
// typically just contains an infinite loop - causing the application to
// appear to have hang when no debugger is connected.
//
// The below code provides an example hard fault handler which instead looks
// to see what the instruction that caused the hard fault was - and if it
// was a "BKPT 0xAB", then it instead returns back to the user application.
//
// In most cases this will allow applications containing semihosting
// operations to execute (to some degree) when the debugger is not connected.
//
// == NOTE ==
//
// Correct execution of the application containing semihosted operations
// which are vectored onto this hard fault handler cannot be guaranteed. This
// is because the handler may not return data or return codes that the higher
// level C library code or application code expects. This hard fault handler
// is meant as a development aid, and it is not recommended to leave
// semihosted code in a production build of your application!
//
// *****************************************************************************

// Allow handler to be removed by setting a define (via command line)
#if !defined (__SEMIHOST_HARDFAULT_DISABLE)

__attribute__((naked))
void HardFault_Handler(void){
    __asm(  ".syntax unified\n"
        // Check which stack is in use
            "MOVS   R0, #4  \n"
            "MOV    R1, LR  \n"
            "TST    R0, R1  \n"
            "BEQ    _MSP    \n"
            "MRS    R0, PSP \n"
            "B  _process      \n"
            "_MSP:  \n"
            "MRS    R0, MSP \n"
        // Load the instruction that triggered hard fault
        "_process:     \n"
            "LDR    R1,[R0,#24] \n"
            "LDRH    R2,[r1] \n"
        // Semihosting instruction is "BKPT 0xAB" (0xBEAB)
            "LDR    R3,=0xBEAB \n"
            "CMP     R2,R3 \n"
            "BEQ    _semihost_return \n"
        // Wasn't semihosting instruction so enter infinite loop
            "B . \n"
        // Was semihosting instruction, so adjust location to
        // return to by 1 instruction (2 bytes), then exit function
        "_semihost_return: \n"
            "ADDS    R1,#2 \n"
            "STR    R1,[R0,#24] \n"
     // Set a return value from semihosting operation.
     // 32 is slightly arbitrary, but appears to allow most
     // C Library IO functions sitting on top of semihosting to
     // continue to operate to some degree
          "MOVS   R1,#32 \n"
          "STR R1,[ R0,#0 ] \n" // R0 is at location 0 on stack
     // Return from hard fault handler to application
            "BX LR \n"
        ".syntax divided\n") ;
}

#endif

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file main.c
 * @brief Project 2
 *
 * @details This file calls the included prototypes and coordinates
 *          state for the LED program. Depending on the platform, this
 *          may just print to the screen or it will actually blink
 *          a physical LED.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */

#include <stdio.h>
#include "delay.h"
#include "setup_teardown.h"
#include "led_types.h"
#include "handle_led.h"
#include <stdint.h>
#include <stdbool.h>
```

```
/**
 * NUM_CYCLES
 *
 * @brief The number of times to loop through the timing cycles.
 */
#define NUM_CYCLES 10

/**
 * STEPS_PER_CYCLE
 *
 * The number of steps in a single cycle.
 */
#define STEPS_PER_CYCLE 20

/**
 * NUM_COLOR_STEPS
 *
 * @brief The number of times to flash a color before cycling to the next.
 */
#define NUM_COLOR_STEPS 3



int main()
{
 static const uint64_t TIMINGS[STEPS_PER_CYCLE] =
 {
        3000,
        1000,
        2000,
        600,
        1000,
        400,
        1000,
        200,
        500,
        100,
        500,
        100,
        500,
        100,
        1000,
        200,
        1000,
        400,
        2000,
        600
 };

    initialize();

    bool ledValue = 1;
    enum COLOR ledColor = RED;
    int counter = 0;

    for(int j = 0; j < NUM_CYCLES; j++)
    {
        for(int i = 0; i < STEPS_PER_CYCLE; i++)
        {
         set_led((uint8_t)ledValue, ledColor);
            delay(TIMINGS[i]);

            // update LED state
            ledValue = !ledValue;
            if(++counter % NUM_COLOR_STEPS == 0)
                ledColor = ((ledColor + ledValue) % NUM_COLORS);
        }
    }

    terminate();

    return 0;
}

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file setup_teardown.c
 * @brief Project 2
 *
 * @details On PC, there is no setup and teardown to be done, but
 *          we print program start and end in debug builds.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */
#include "setup_teardown.h"
#include <stdio.h>

/**
 * initialize
 *
 * @brief Print "program start" in debug builds. Shows that the program successfully started.
 *
 */
void initialize(void)
{
#ifdef DEBUG
    printf("\nprogram start\n");
#endif
}
```

```
/**
 * terminate
 *
 * @brief Print "program end" in debug builds. Shows that the program successfully completed.
 *
 */
void terminate(void)
{
#ifdef DEBUG
    printf("\nprogram end\n");
#endif
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file handle_led.c
 * @brief Project 2
 *
 * @details Contains the prototype for handling LEDs on various platforms.
 *          On PC, this implementation just prints what the LED state would be
 *          if an LED was present.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */
#include "handle_led.h"
#include <stdio.h>

/**
 * set_led
 *
 * @brief Sets the LED state.
 * @details This function will simply print the
 *          state of what the LED would be.
 * @param inValue The on/off state of the LED to set.
 * @param inColor The color of the LED to set.
 */
void set_led(uint8_t inValue, enum COLOR inColor)
{
    printf("\nLED %s %s", COLOR_STRINGS[inColor], inValue ? "ON" : "OFF");
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file delay.c
 * @brief Project 2
 *
 * @details This file contains prototypes for calculating a spin-wait
 *          on PC, used for delaying LED state changes.
```

```
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *        PC Linker: GNU ld 2.32
 *        PC Debugger: GNU gdb 8.2.91.20190405-git
 *        ARM Compiler: GNU gcc version 8.2.1 20181213
 *        ARM Linker: GNU ld 2.31.51.20181213
 *        ARM Debugger: GNU gdb 8.2.50.20181213-git
 *
 * - Calculating the timestamp formatting using time.h is from:
 *    http://www.cplusplus.com/reference/ctime/strftime/
 */

#include "delay.h"
#include <time.h>
#include <stdio.h>

/**
 * delay
 *
 * @brief Blocks execution for the specified time.
 * @param inDelayMs Then time in milliseconds to block.
 */
void delay(uint64_t inDelayMs)
{
#ifdef DEBUG
 static int64_t sTotalTime = 0;
 static int64_t sLastTotalTime = 0;
 sTotalTime += inDelayMs;

 // figure out time stamp
 // http://www.cplusplus.com/reference/ctime/strftime/
 time_t rawtime;
   struct tm * timeinfo;
 char buffer [80];
 time (&rawtime);
 timeinfo = localtime (&rawtime);
 strftime (buffer,80,"%T",timeinfo);
 printf(" %s %lld\n", buffer, sTotalTime - sLastTotalTime);
 sLastTotalTime = sTotalTime;
#endif

 const uint64_t clocksPerMillisecond = CLOCKS_PER_SEC / 1000UL;
   uint64_t waitUntil = clock() + inDelayMs*clocksPerMillisecond;
   while( clock() < waitUntil );
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file setup_teardown.c
 * @brief Project 2
 *
 * @details Initializes the LEDs, board, and debug console.
 *
```

```
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *        PC Linker: GNU ld 2.32
 *        PC Debugger: GNU gdb 8.2.91.20190405-git
 *        ARM Compiler: GNU gcc version 8.2.1 20181213
 *        ARM Linker: GNU ld 2.31.51.20181213
 *        ARM Debugger: GNU gdb 8.2.50.20181213-git
 */

#include "setup_teardown.h"
#include "fsl_debug_console.h"
#include "MKL25Z4.h"
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"

/**
 * initialize
 *
 * @details Initializes the LEDs, board, and debug console.
 *         Print "program start" in debug builds.
 *         Shows that the program successfully started.
 *
 */
void initialize()
{
 /* board setup */
 BOARD_InitBootPins();
 BOARD_InitBootClocks();
 BOARD_InitBootPeripherals();

 /* led setup */
 LED_RED_INIT(1);
 LED_BLUE_INIT(1);
 LED_GREEN_INIT(1);



#ifdef DEBUG
 /* serial debug console setup: use PRINTF("debug msg"); */
 BOARD_InitDebugConsole();

 PRINTF("program start");
#endif

}

/**
 * terminate
 *
 * @details Print "program end" in debug builds.
 *         Shows that the program successfully completed.
 *
 */
void terminate()
{
#ifdef DEBUG
 PRINTF("program end");
#endif
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file handle_led.c
 * @brief Project 2
 *
 * @details Contains the prototype for handling LEDs on the FB.
 *          In this implementation, we print debug info in debug builds
 *          and flash the LED either way.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */

#include <stdint.h>
#include "handle_led.h"
#include "board.h"
#include "fsl_debug_console.h"
#include "fsl_gpio.h"
#include "MKL25Z4.h"

/**
 * set_led
 *
 * @brief Sets the LED state.
 * @details This function controls a physical LED and prints
 *          debug info over UART on debug builds.
 * @param inValue The on/off state of the LED to set.
 * @param inColor The color of the LED to set.
 */
void set_led(uint8_t inValue, enum COLOR inColor)
{
#ifdef DEBUG
 GPIO_TogglePinsOutput(GPIOD, 1U << 7U);
 PRINTF("\nLED %s %s", COLOR_STRINGS[inColor], inValue ? "ON" : "OFF");
#endif

 switch(inColor)
 {
  case RED:
  {
```

```
   LED_BLUE_OFF();
   LED_GREEN_OFF();
   if(inValue)
   {
    LED_RED_ON();
   }
   else
   {
    LED_RED_OFF();
   }

   break;
  }
  case GREEN:
  {
   LED_BLUE_OFF();
   LED_RED_OFF();

   if(inValue)
   {
    LED_GREEN_ON();
   }
   else
   {
    LED_GREEN_OFF();
   }
   break;
  }
  case BLUE:
  {
   LED_GREEN_OFF();
   LED_RED_OFF();

   if(inValue)
   {
    LED_BLUE_ON();
   }
   else
   {
    LED_BLUE_OFF();
   }
   break;
  }
  default:
    break;
 }
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file delay.c
 * @brief Project 2
 *
 * @details This file contains prototypes for calculating a spin-wait
```

```
 *         on the FB, used for delaying LED state changes.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *        PC Linker: GNU ld 2.32
 *        PC Debugger: GNU gdb 8.2.91.20190405-git
 *        ARM Compiler: GNU gcc version 8.2.1 20181213
 *        ARM Linker: GNU ld 2.31.51.20181213
 *        ARM Debugger: GNU gdb 8.2.50.20181213-git
 */
#include "delay.h"
#include "fsl_debug_console.h"

/* GLOBALS */
const uint64_t CLOCKS_PER_MILLISECOND = 2600UL;
const uint64_t PRINTF_OFFSET = 900UL;

/**
 * delay
 *
 * @brief Blocks execution for the specified time.
 * @param inDelayMs Then time in milliseconds to block.
 */
void delay(uint64_t inDelayMs)
{
 volatile uint64_t number = inDelayMs * CLOCKS_PER_MILLISECOND;
#ifdef DEBUG
 PRINTF(" %llu", inDelayMs);
 number -= PRINTF_OFFSET;
#endif

 while(number--)
 {
  __asm volatile ("nop");
 }
}
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 _FSL_PORT_H_
#define _FSL_PORT_H_

#include "fsl_common.h"

/*!
 * @addtogroup port
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @name Driver version */
/*@{*/
/*! Version 2.0.2. */
#define FSL_PORT_DRIVER_VERSION (MAKE_VERSION(2, 0, 2))
/*@}*/

#if defined(FSL_FEATURE_PORT_HAS_PULL_ENABLE) && FSL_FEATURE_PORT_HAS_PULL_ENAB
/*! @brief Internal resistor pull feature selection */
enum _port_pull
{
    kPORT_PullDisable = 0U, /*!< Internal pull-up/down resistor is disabled. */
    kPORT_PullDown = 2U,    /*!< Internal pull-down resistor is enabled. */
    kPORT_PullUp = 3U,      /*!< Internal pull-up resistor is enabled. */
};
#endif /* FSL_FEATURE_PORT_HAS_PULL_ENABLE */

#if defined(FSL_FEATURE_PORT_HAS_SLEW_RATE) && FSL_FEATURE_PORT_HAS_SLEW_RATE
/*! @brief Slew rate selection */
enum _port_slew_rate
{
    kPORT_FastSlewRate = 0U, /*!< Fast slew rate is configured. */
    kPORT_SlowSlewRate = 1U, /*!< Slow slew rate is configured. */
};
#endif /* FSL_FEATURE_PORT_HAS_SLEW_RATE */

#if defined(FSL_FEATURE_PORT_HAS_OPEN_DRAIN) && FSL_FEATURE_PORT_HAS_OPEN_DRAIN
```

```
/*! @brief Open Drain feature enable/disable */
enum _port_open_drain_enable
{
    kPORT_OpenDrainDisable = 0U, /*!< Open drain output is disabled. */
    kPORT_OpenDrainEnable = 1U,  /*!< Open drain output is enabled. */
};
#endif /* FSL_FEATURE_PORT_HAS_OPEN_DRAIN */

#if defined(FSL_FEATURE_PORT_HAS_PASSIVE_FILTER) && FSL_FEATURE_PORT_HAS_PASSIVE_
/*! @brief Passive filter feature enable/disable */
enum _port_passive_filter_enable
{
    kPORT_PassiveFilterDisable = 0U, /*!< Passive input filter is disabled. */
    kPORT_PassiveFilterEnable = 1U,  /*!< Passive input filter is enabled. */
};
#endif

#if defined(FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH) && FSL_FEATURE_PORT_HAS_DRIVE_
/*! @brief Configures the drive strength. */
enum _port_drive_strength
{
    kPORT_LowDriveStrength = 0U,  /*!< Low-drive strength is configured. */
    kPORT_HighDriveStrength = 1U, /*!< High-drive strength is configured. */
};
#endif /* FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH */

#if defined(FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK) && FSL_FEATURE_PORT_HAS_PIN_
/*! @brief Unlock/lock the pin control register field[15:0] */
enum _port_lock_register
{
    kPORT_UnlockRegister = 0U, /*!< Pin Control Register fields [15:0] are not locked. */
    kPORT_LockRegister = 1U,   /*!< Pin Control Register fields [15:0] are locked. */
};
#endif /* FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK */

#if defined(FSL_FEATURE_PORT_PCR_MUX_WIDTH) && FSL_FEATURE_PORT_PCR_MUX_WIDTH
/*! @brief Pin mux selection */
typedef enum _port_mux
{
    kPORT_PinDisabledOrAnalog = 0U, /*!< Corresponding pin is disabled, but is used as an analog pin. */
    kPORT_MuxAsGpio = 1U,           /*!< Corresponding pin is configured as GPIO. */
    kPORT_MuxAlt2 = 2U,             /*!< Chip-specific */
    kPORT_MuxAlt3 = 3U,             /*!< Chip-specific */
    kPORT_MuxAlt4 = 4U,             /*!< Chip-specific */
    kPORT_MuxAlt5 = 5U,             /*!< Chip-specific */
    kPORT_MuxAlt6 = 6U,             /*!< Chip-specific */
    kPORT_MuxAlt7 = 7U,             /*!< Chip-specific */
    kPORT_MuxAlt8 = 8U,             /*!< Chip-specific */
    kPORT_MuxAlt9 = 9U,             /*!< Chip-specific */
    kPORT_MuxAlt10 = 10U,           /*!< Chip-specific */
    kPORT_MuxAlt11 = 11U,           /*!< Chip-specific */
    kPORT_MuxAlt12 = 12U,           /*!< Chip-specific */
    kPORT_MuxAlt13 = 13U,           /*!< Chip-specific */
    kPORT_MuxAlt14 = 14U,           /*!< Chip-specific */
    kPORT_MuxAlt15 = 15U,           /*!< Chip-specific */
} port_mux_t;
#endif /* FSL_FEATURE_PORT_PCR_MUX_WIDTH */

/*! @brief Configures the interrupt generation condition. */
typedef enum _port_interrupt
{
    kPORT_InterruptOrDMADisabled = 0x0U, /*!< Interrupt/DMA request is disabled. */
#if defined(FSL_FEATURE_PORT_HAS_DMA_REQUEST) && FSL_FEATURE_PORT_HAS_DMA_REQU
    kPORT_DMARisingEdge = 0x1U,  /*!< DMA request on rising edge. */
    kPORT_DMAFallingEdge = 0x2U, /*!< DMA request on falling edge. */
    kPORT_DMAEitherEdge = 0x3U,  /*!< DMA request on either edge. */
#endif
#if defined(FSL_FEATURE_PORT_HAS_IRQC_FLAG) && FSL_FEATURE_PORT_HAS_IRQC_FLAG
    kPORT_FlagRisingEdge = 0x05U,  /*!< Flag sets on rising edge. */
    kPORT_FlagFallingEdge = 0x06U, /*!< Flag sets on falling edge. */
    kPORT_FlagEitherEdge = 0x07U,  /*!< Flag sets on either edge. */
#endif
    kPORT_InterruptLogicZero = 0x8U,   /*!< Interrupt when logic zero. */
    kPORT_InterruptRisingEdge = 0x9U,  /*!< Interrupt on rising edge. */
    kPORT_InterruptFallingEdge = 0xAU, /*!< Interrupt on falling edge. */
    kPORT_InterruptEitherEdge = 0xBU,  /*!< Interrupt on either edge. */
    kPORT_InterruptLogicOne = 0xCU,    /*!< Interrupt when logic one. */
#if defined(FSL_FEATURE_PORT_HAS_IRQC_TRIGGER) && FSL_FEATURE_PORT_HAS_IRQC_TRIG
    kPORT_ActiveHighTriggerOutputEnable = 0xDU, /*!< Enable active high-trigger output. */
    kPORT_ActiveLowTriggerOutputEnable = 0xEU,  /*!< Enable active low-trigger output. */
#endif
} port_interrupt_t;

#if defined(FSL_FEATURE_PORT_HAS_DIGITAL_FILTER) && FSL_FEATURE_PORT_HAS_DIGITAL_F
/*! @brief Digital filter clock source selection */
typedef enum _port_digital_filter_clock_source
{
    kPORT_BusClock = 0U, /*!< Digital filters are clocked by the bus clock. */
    kPORT_LpoClock = 1U, /*!< Digital filters are clocked by the 1 kHz LPO clock. */
} port_digital_filter_clock_source_t;

/*! @brief PORT digital filter feature configuration definition */
typedef struct _port_digital_filter_config
{
    uint32_t digitalFilterWidth;                    /*!< Set digital filter width */
    port_digital_filter_clock_source_t clockSource; /*!< Set digital filter clockSource */
} port_digital_filter_config_t;
#endif /* FSL_FEATURE_PORT_HAS_DIGITAL_FILTER */

#if defined(FSL_FEATURE_PORT_PCR_MUX_WIDTH) && FSL_FEATURE_PORT_PCR_MUX_WIDTH
/*! @brief PORT pin configuration structure */
typedef struct _port_pin_config
{
#if defined(FSL_FEATURE_PORT_HAS_PULL_ENABLE) && FSL_FEATURE_PORT_HAS_PULL_ENAB
    uint16_t pullSelect : 2; /*!< No-pull/pull-down/pull-up select */
#else
```

```
    uint16_t : 2;
#endif /* FSL_FEATURE_PORT_HAS_PULL_ENABLE */

#if defined(FSL_FEATURE_PORT_HAS_SLEW_RATE) && FSL_FEATURE_PORT_HAS_SLEW_RATE
    uint16_t slewRate : 1; /*!< Fast/slow slew rate Configure */
#else
    uint16_t : 1;
#endif /* FSL_FEATURE_PORT_HAS_SLEW_RATE */

    uint16_t : 1;

#if defined(FSL_FEATURE_PORT_HAS_PASSIVE_FILTER) && FSL_FEATURE_PORT_HAS_PASSIVE_
    uint16_t passiveFilterEnable : 1; /*!< Passive filter enable/disable */
#else
    uint16_t : 1;
#endif /* FSL_FEATURE_PORT_HAS_PASSIVE_FILTER */

#if defined(FSL_FEATURE_PORT_HAS_OPEN_DRAIN) && FSL_FEATURE_PORT_HAS_OPEN_DRAIN
    uint16_t openDrainEnable : 1; /*!< Open drain enable/disable */
#else
    uint16_t : 1;
#endif /* FSL_FEATURE_PORT_HAS_OPEN_DRAIN */

#if defined(FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH) && FSL_FEATURE_PORT_HAS_DRIVE_
    uint16_t driveStrength : 1; /*!< Fast/slow drive strength configure */
#else
    uint16_t : 1;
#endif

    uint16_t : 1;

#if defined(FSL_FEATURE_PORT_PCR_MUX_WIDTH) && FSL_FEATURE_PORT_PCR_MUX_WIDTH
    uint16_t mux : 3; /*!< Pin mux Configure */
#else
    uint16_t : 3;
#endif

    uint16_t : 4;

#if defined(FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK) && FSL_FEATURE_PORT_HAS_PIN_
    uint16_t lockRegister : 1; /*!< Lock/unlock the PCR field[15:0] */
#else
    uint16_t : 1;
#endif /* FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK */
} port_pin_config_t;
#endif /* FSL_FEATURE_PORT_PCR_MUX_WIDTH */

/*******************************************************************************
* API
******************************************************************************/

#if defined(__cplusplus)
extern "C" {
```

```
#endif

#if defined(FSL_FEATURE_PORT_PCR_MUX_WIDTH) && FSL_FEATURE_PORT_PCR_MUX_WIDTH
/*! @name Configuration */
/*@{*/

/*!
 * @brief Sets the port PCR register.
 *
 * This is an example to define an input pin or output pin PCR configuration.
 * @code
 * // Define a digital input pin PCR configuration
 * port_pin_config_t config = {
 *      kPORT_PullUp,
 *      kPORT_FastSlewRate,
 *      kPORT_PassiveFilterDisable,
 *      kPORT_OpenDrainDisable,
 *      kPORT_LowDriveStrength,
 *      kPORT_MuxAsGpio,
 *      kPORT_UnLockRegister,
 * };
 * @endcode
 *
 * @param base   PORT peripheral base pointer.
 * @param pin    PORT pin number.
 * @param config PORT PCR register configuration structure.
 */
static inline void PORT_SetPinConfig(PORT_Type *base, uint32_t pin, const port_pin_config_t *config)
{
    assert(config);
    uint32_t addr = (uint32_t)&base->PCR[pin];
    *(volatile uint16_t *)(addr) = *((const uint16_t *)config);
}

/*!
 * @brief Sets the port PCR register for multiple pins.
 *
 * This is an example to define input pins or output pins PCR configuration.
 * @code
 * // Define a digital input pin PCR configuration
 * port_pin_config_t config = {
 *      kPORT_PullUp ,
 *      kPORT_PullEnable,
 *      kPORT_FastSlewRate,
 *      kPORT_PassiveFilterDisable,
 *      kPORT_OpenDrainDisable,
 *      kPORT_LowDriveStrength,
 *      kPORT_MuxAsGpio,
 *      kPORT_UnlockRegister,
 * };
 * @endcode
 *
 * @param base   PORT peripheral base pointer.
 * @param mask   PORT pin number macro.
 * @param config PORT PCR register configuration structure.
 */
static inline void PORT_SetMultiplePinsConfig(PORT_Type *base, uint32_t mask, const port_pin_config_t
{
    assert(config);

    uint16_t pcrl = *((const uint16_t *)config);

    if (mask & 0xffffU)
    {
        base->GPCLR = ((mask & 0xffffU) << 16) | pcrl;
    }
    if (mask >> 16)
    {
        base->GPCHR = (mask & 0xffff0000U) | pcrl;
    }
}

/*!
 * @brief Configures the pin muxing.
 *
 * @param base  PORT peripheral base pointer.
 * @param pin   PORT pin number.
 * @param mux   pin muxing slot selection.
 *        - #kPORT_PinDisabledOrAnalog: Pin disabled or work in analog function.
 *        - #kPORT_MuxAsGpio          : Set as GPIO.
 *        - #kPORT_MuxAlt2            : chip-specific.
 *        - #kPORT_MuxAlt3            : chip-specific.
 *        - #kPORT_MuxAlt4            : chip-specific.
 *        - #kPORT_MuxAlt5            : chip-specific.
 *        - #kPORT_MuxAlt6            : chip-specific.
 *        - #kPORT_MuxAlt7            : chip-specific.
 * @Note : This function is NOT recommended to use together with the PORT_SetPinsConfig, because
 *         the PORT_SetPinsConfig need to configure the pin mux anyway (Otherwise the pin mux is
 *         reset to zero : kPORT_PinDisabledOrAnalog).
 *        This function is recommended to use to reset the pin mux
 *
 */
static inline void PORT_SetPinMux(PORT_Type *base, uint32_t pin, port_mux_t mux)
{
    base->PCR[pin] = (base->PCR[pin] & ~PORT_PCR_MUX_MASK) | PORT_PCR_MUX(mux);
}
#endif /* FSL_FEATURE_PORT_PCR_MUX_WIDTH */

#if defined(FSL_FEATURE_PORT_HAS_DIGITAL_FILTER) && FSL_FEATURE_PORT_HAS_DIGITAL_F

/*!
 * @brief Enables the digital filter in one port, each bit of the 32-bit register represents one pin.
 *
 * @param base  PORT peripheral base pointer.
 * @param mask  PORT pin number macro.
 */
```

```
static inline void PORT_EnablePinsDigitalFilter(PORT_Type *base, uint32_t mask, bool enable)
{
    if (enable == true)
    {
        base->DFER |= mask;
    }
    else
    {
        base->DFER &= ~mask;
    }
}

/*!
 * @brief Sets the digital filter in one port, each bit of the 32-bit register represents one pin.
 *
 * @param base   PORT peripheral base pointer.
 * @param config PORT digital filter configuration structure.
 */
static inline void PORT_SetDigitalFilterConfig(PORT_Type *base, const port_digital_filter_config_t *config)
{
    assert(config);

    base->DFCR = PORT_DFCR_CS(config->clockSource);
    base->DFWR = PORT_DFWR_FILT(config->digitalFilterWidth);
}

#endif /* FSL_FEATURE_PORT_HAS_DIGITAL_FILTER */

/*@}*/

/*! @name Interrupt */
/*@{*/

/*!
 * @brief Configures the port pin interrupt/DMA request.
 *
 * @param base    PORT peripheral base pointer.
 * @param pin     PORT pin number.
 * @param config  PORT pin interrupt configuration.
 *        - #kPORT_InterruptOrDMADisabled: Interrupt/DMA request disabled.
 *        - #kPORT_DMARisingEdge : DMA request on rising edge(if the DMA requests exit).
 *        - #kPORT_DMAFallingEdge: DMA request on falling edge(if the DMA requests exit).
 *        - #kPORT_DMAEitherEdge : DMA request on either edge(if the DMA requests exit).
 *        - #kPORT_FlagRisingEdge : Flag sets on rising edge(if the Flag states exit).
 *        - #kPORT_FlagFallingEdge : Flag sets on falling edge(if the Flag states exit).
 *        - #kPORT_FlagEitherEdge : Flag sets on either edge(if the Flag states exit).
 *        - #kPORT_InterruptLogicZero  : Interrupt when logic zero.
 *        - #kPORT_InterruptRisingEdge : Interrupt on rising edge.
 *        - #kPORT_InterruptFallingEdge: Interrupt on falling edge.
 *        - #kPORT_InterruptEitherEdge : Interrupt on either edge.
 *        - #kPORT_InterruptLogicOne   : Interrupt when logic one.
 *        - #kPORT_ActiveHighTriggerOutputEnable : Enable active high-trigger output (if the trigger states ex
 *        - #kPORT_ActiveLowTriggerOutputEnable  : Enable active low-trigger output (if the trigger states ex
 */
static inline void PORT_SetPinInterruptConfig(PORT_Type *base, uint32_t pin, port_interrupt_t config)
{
    base->PCR[pin] = (base->PCR[pin] & ~PORT_PCR_IRQC_MASK) | PORT_PCR_IRQC(config);
}

/*!
 * @brief Reads the whole port status flag.
 *
 * If a pin is configured to generate the DMA request,  the corresponding flag
 * is cleared automatically at the completion of the requested DMA transfer.
 * Otherwise, the flag remains set until a logic one is written to that flag.
 * If configured for a level sensitive interrupt that remains asserted, the flag
 * is set again immediately.
 *
 * @param base PORT peripheral base pointer.
 * @return Current port interrupt status flags, for example, 0x00010001 means the
 *         pin 0 and 16 have the interrupt.
 */
static inline uint32_t PORT_GetPinsInterruptFlags(PORT_Type *base)
{
    return base->ISFR;
}

/*!
 * @brief Clears the multiple pin interrupt status flag.
 *
 * @param base PORT peripheral base pointer.
 * @param mask PORT pin number macro.
 */
static inline void PORT_ClearPinsInterruptFlags(PORT_Type *base, uint32_t mask)
{
    base->ISFR = mask;
}

/*@}*/

#if defined(__cplusplus)
}
#endif

/*! @}*/

#endif /* _FSL_PORT_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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
```

```
#ifndef _FSL_SMC_H_
#define _FSL_SMC_H_

#include "fsl_common.h"

/*! @addtogroup smc */
/*! @{ */


/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @name Driver version */
/*@{*/
/*! @brief SMC driver version 2.0.3. */
#define FSL_SMC_DRIVER_VERSION (MAKE_VERSION(2, 0, 3))
/*@}*/

/*!
 * @brief Power Modes Protection
 */
typedef enum _smc_power_mode_protection
{
#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
    kSMC_AllowPowerModeVlls = SMC_PMPROT_AVLLS_MASK, /*!< Allow Very-low-leakage Stop Mode.
#endif
#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
    kSMC_AllowPowerModeLls = SMC_PMPROT_ALLS_MASK, /*!< Allow Low-leakage Stop Mode.      */
#endif                                          /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */
    kSMC_AllowPowerModeVlp = SMC_PMPROT_AVLP_MASK, /*!< Allow Very-Low-power Mode.        */
```

```
#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
    kSMC_AllowPowerModeHsrun = SMC_PMPROT_AHSRUN_MASK, /*!< Allow High-speed Run mode.
#endif                                          /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */
    kSMC_AllowPowerModeAll = (0U
#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
                              |
                              SMC_PMPROT_AVLLS_MASK
#endif
#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
                              |
                              SMC_PMPROT_ALLS_MASK
#endif /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */
                              |
                              SMC_PMPROT_AVLP_MASK
#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
                              |
                              kSMC_AllowPowerModeHsrun
#endif                            /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */
                              ) /*!< Allow all power mode.              */
} smc_power_mode_protection_t;

/*!
 * @brief Power Modes in PMSTAT
 */
typedef enum _smc_power_state
{
    kSMC_PowerStateRun = 0x01U << 0U,  /*!< 0000_0001 - Current power mode is RUN   */
    kSMC_PowerStateStop = 0x01U << 1U, /*!< 0000_0010 - Current power mode is STOP  */
    kSMC_PowerStateVlpr = 0x01U << 2U, /*!< 0000_0100 - Current power mode is VLPR  */
    kSMC_PowerStateVlpw = 0x01U << 3U, /*!< 0000_1000 - Current power mode is VLPW  */
    kSMC_PowerStateVlps = 0x01U << 4U, /*!< 0001_0000 - Current power mode is VLPS  */
#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
    kSMC_PowerStateLls = 0x01U << 5U, /*!< 0010_0000 - Current power mode is LLS   */
#endif                                /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */
#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
    kSMC_PowerStateVlls = 0x01U << 6U, /*!< 0100_0000 - Current power mode is VLLS  */
#endif
#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
    kSMC_PowerStateHsrun = 0x01U << 7U /*!< 1000_0000 - Current power mode is HSRUN */
#endif                                 /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */
} smc_power_state_t;

/*!
 * @brief Run mode definition
 */
typedef enum _smc_run_mode
{
    kSMC_RunNormal = 0U, /*!< Normal RUN mode.             */
    kSMC_RunVlpr = 2U,   /*!< Very-low-power RUN mode.     */
#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
    kSMC_Hsrun = 3U /*!< High-speed Run mode (HSRUN). */
#endif              /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */
} smc_run_mode_t;
```

```
/*!
 * @brief Stop mode definition
 */
typedef enum _smc_stop_mode
{
    kSMC_StopNormal = 0U, /*!< Normal STOP mode.          */
    kSMC_StopVlps = 2U,   /*!< Very-low-power STOP mode.   */
#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
    kSMC_StopLls = 3U, /*!< Low-leakage Stop mode.      */
#endif                /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */
#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
    kSMC_StopVlls = 4U /*!< Very-low-leakage Stop mode. */
#endif
} smc_stop_mode_t;

#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
    (defined(FSL_FEATURE_SMC_USE_STOPCTRL_VLLSM) && FSL_FEATURE_SMC_USE_STOPCTR
    (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMOD
/*!
 * @brief VLLS/LLS stop sub mode definition
 */
typedef enum _smc_stop_submode
{
    kSMC_StopSub0 = 0U, /*!< Stop submode 0, for VLLS0/LLS0. */
    kSMC_StopSub1 = 1U, /*!< Stop submode 1, for VLLS1/LLS1. */
    kSMC_StopSub2 = 2U, /*!< Stop submode 2, for VLLS2/LLS2. */
    kSMC_StopSub3 = 3U  /*!< Stop submode 3, for VLLS3/LLS3. */
} smc_stop_submode_t;
#endif

/*!
 * @brief Partial STOP option
 */
typedef enum _smc_partial_stop_mode
{
    kSMC_PartialStop = 0U,  /*!< STOP - Normal Stop mode*/
    kSMC_PartialStop1 = 1U, /*!< Partial Stop with both system and bus clocks disabled*/
    kSMC_PartialStop2 = 2U, /*!< Partial Stop with system clock disabled and bus clock enabled*/
} smc_partial_stop_option_t;

/*!
 * @brief SMC configuration status.
 */
enum _smc_status
{
    kStatus_SMC_StopAbort = MAKE_STATUS(kStatusGroup_POWER, 0) /*!< Entering Stop mode is abor
};

#if (defined(FSL_FEATURE_SMC_HAS_VERID) && FSL_FEATURE_SMC_HAS_VERID)
/*!
 * @brief IP version ID definition.
 */
```

```
typedef struct _smc_version_id
{
    uint16_t feature; /*!< Feature Specification Number. */
    uint8_t minor;    /*!< Minor version number.         */
    uint8_t major;    /*!< Major version number.         */
} smc_version_id_t;
#endif /* FSL_FEATURE_SMC_HAS_VERID */

#if (defined(FSL_FEATURE_SMC_HAS_PARAM) && FSL_FEATURE_SMC_HAS_PARAM)
/*!
 * @brief IP parameter definition.
 */
typedef struct _smc_param
{
    bool hsrunEnable; /*!< HSRUN mode enable. */
    bool llsEnable;   /*!< LLS mode enable.   */
    bool lls2Enable;  /*!< LLS2 mode enable.  */
    bool vlls0Enable; /*!< VLLS0 mode enable. */
} smc_param_t;
#endif /* FSL_FEATURE_SMC_HAS_PARAM */

#if (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMOD
    (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO)
/*!
 * @brief SMC Low-Leakage Stop power mode configuration.
 */
typedef struct _smc_power_mode_lls_config
{
#if (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMOD
    smc_stop_submode_t subMode; /*!< Low-leakage Stop sub-mode */
#endif
#if (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO)
    bool enableLpoClock; /*!< Enable LPO clock in LLS mode */
#endif
} smc_power_mode_lls_config_t;
#endif /* (FSL_FEATURE_SMC_HAS_LLS_SUBMODE || FSL_FEATURE_SMC_HAS_LPOPO) */

#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
/*!
 * @brief SMC Very Low-Leakage Stop power mode configuration.
 */
typedef struct _smc_power_mode_vlls_config
{
#if (defined(FSL_FEATURE_SMC_USE_VLLSCTRL_REG) && FSL_FEATURE_SMC_USE_VLLSCTRL_
    (defined(FSL_FEATURE_SMC_USE_STOPCTRL_VLLSM) && FSL_FEATURE_SMC_USE_STOPCTR
    (defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMOD
    smc_stop_submode_t subMode; /*!< Very Low-leakage Stop sub-mode */
#endif
#if (defined(FSL_FEATURE_SMC_HAS_PORPO) && FSL_FEATURE_SMC_HAS_PORPO)
    bool enablePorDetectInVlls0; /*!< Enable Power on reset detect in VLLS mode */
#endif
#if (defined(FSL_FEATURE_SMC_HAS_RAM2_POWER_OPTION) && FSL_FEATURE_SMC_HAS_RAM
    bool enableRam2InVlls2; /*!< Enable RAM2 power in VLLS2 */
```

```
#endif
#if (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO)
    bool enableLpoClock; /*!< Enable LPO clock in VLLS mode */
#endif
} smc_power_mode_vlls_config_t;
#endif

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! @name System mode controller APIs*/
/*@{*/

#if (defined(FSL_FEATURE_SMC_HAS_VERID) && FSL_FEATURE_SMC_HAS_VERID)
/*!
 * @brief Gets the SMC version ID.
 *
 * This function gets the SMC version ID, including major version number,
 * minor version number, and feature specification number.
 *
 * @param base SMC peripheral base address.
 * @param versionId     Pointer to the version ID structure.
 */
static inline void SMC_GetVersionId(SMC_Type *base, smc_version_id_t *versionId)
{
    *((uint32_t *)versionId) = base->VERID;
}
#endif /* FSL_FEATURE_SMC_HAS_VERID */

#if (defined(FSL_FEATURE_SMC_HAS_PARAM) && FSL_FEATURE_SMC_HAS_PARAM)
/*!
 * @brief Gets the SMC parameter.
 *
 * This function gets the SMC parameter including the enabled power mdoes.
 *
 * @param base SMC peripheral base address.
 * @param param         Pointer to the SMC param structure.
 */
void SMC_GetParam(SMC_Type *base, smc_param_t *param);
#endif

/*!
 * @brief Configures all power mode protection settings.
 *
 * This function  configures the power mode protection settings for
 * supported power modes in the specified chip family. The available power modes
 * are defined in the smc_power_mode_protection_t. This should be done at an early
 * system level initialization stage. See the reference manual for details.
 * This register can only write once after the power reset.
 *
 * The allowed modes are passed as bit map. For example, to allow LLS and VLLS,
 * use SMC_SetPowerModeProtection(kSMC_AllowPowerModeVlls | kSMC_AllowPowerModeVlps).
 * To allow all modes, use SMC_SetPowerModeProtection(kSMC_AllowPowerModeAll).
 *
 * @param base SMC peripheral base address.
 * @param allowedModes Bitmap of the allowed power modes.
 */
static inline void SMC_SetPowerModeProtection(SMC_Type *base, uint8_t allowedModes)
{
    base->PMPROT = allowedModes;
}

/*!
 * @brief Gets the current power mode status.
 *
 * This function  returns the current power mode status. After the application
 * switches the power mode, it should always check the status to check whether it
 * runs into the specified mode or not. The application  should  check
 * this mode before switching to a different mode. The system  requires that
 * only certain modes can switch to other specific modes. See the
 * reference manual for details and the smc_power_state_t for information about
 * the power status.
 *
 * @param base SMC peripheral base address.
 * @return Current power mode status.
 */
static inline smc_power_state_t SMC_GetPowerModeState(SMC_Type *base)
{
    return (smc_power_state_t)base->PMSTAT;
}

/*!
 * @brief Prepares to enter stop modes.
 *
 * This function should be called before entering STOP/VLPS/LLS/VLLS modes.
 */
void SMC_PreEnterStopModes(void);

/*!
 * @brief Recovers after wake up from stop modes.
 *
 * This function should be called after wake up from STOP/VLPS/LLS/VLLS modes.
 * It is used with @ref SMC_PreEnterStopModes.
 */
void SMC_PostExitStopModes(void);

/*!
 * @brief Prepares to enter wait modes.
 *
 * This function should be called before entering WAIT/VLPW modes.
 */
```

```
static inline void SMC_PreEnterWaitModes(void)
{
    __disable_irq();
    __ISB();
}

/*!
 * @brief Recovers after wake up from stop modes.
 *
 * This function should be called after wake up from WAIT/VLPW modes.
 * It is used with @ref SMC_PreEnterWaitModes.
 */
static inline void SMC_PostExitWaitModes(void)
{
    __enable_irq();
    __ISB();
}

/*!
 * @brief Configures the system to RUN power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeRun(SMC_Type *base);

#if (defined(FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE) && FSL_FEATURE_SMC_HAS_H
/*!
 * @brief Configures the system to HSRUN power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeHsrun(SMC_Type *base);
#endif /* FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE */

/*!
 * @brief Configures the system to WAIT power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeWait(SMC_Type *base);

/*!
 * @brief Configures the system to Stop power mode.
 *
 * @param base SMC peripheral base address.
 * @param  option Partial Stop mode option.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeStop(SMC_Type *base, smc_partial_stop_option_t option);
```

```
#if (defined(FSL_FEATURE_SMC_HAS_LPWUI) && FSL_FEATURE_SMC_HAS_LPWUI)
/*!
 * @brief Configures the system to VLPR power mode.
 *
 * @param base SMC peripheral base address.
 * @param  wakeupMode Enter Normal Run mode if true, else stay in VLPR mode.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeVlpr(SMC_Type *base, bool wakeupMode);
#else
/*!
 * @brief Configures the system to VLPR power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeVlpr(SMC_Type *base);
#endif /* FSL_FEATURE_SMC_HAS_LPWUI */

/*!
 * @brief Configures the system to VLPW power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeVlpw(SMC_Type *base);

/*!
 * @brief Configures the system to VLPS power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeVlps(SMC_Type *base);

#if (defined(FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SMC_HAS
#if ((defined(FSL_FEATURE_SMC_HAS_LLS_SUBMODE) && FSL_FEATURE_SMC_HAS_LLS_SUBMO
    (defined(FSL_FEATURE_SMC_HAS_LPOPO) && FSL_FEATURE_SMC_HAS_LPOPO))
/*!
 * @brief Configures the system to LLS power mode.
 *
 * @param base SMC peripheral base address.
 * @param  config The LLS power mode configuration structure
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeLls(SMC_Type *base, const smc_power_mode_lls_config_t *config);
#else
/*!
 * @brief Configures the system to LLS power mode.
 *
 * @param base SMC peripheral base address.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeLls(SMC_Type *base);
#endif
#endif /* FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE */

#if (defined(FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE) && FSL_FEATURE_SM
/*!
 * @brief Configures the system to VLLS power mode.
 *
 * @param base SMC peripheral base address.
 * @param  config The VLLS power mode configuration structure.
 * @return SMC configuration error code.
 */
status_t SMC_SetPowerModeVlls(SMC_Type *base, const smc_power_mode_vlls_config_t *config);
#endif /* FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE */

/*@}*/

#if defined(__cplusplus)
}
#endif /* __cplusplus */

/*! @}*/

#endif /* _FSL_SMC_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright (c) 2016 - 2017 , NXP
 * 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 copyright holder 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" AN
 * 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 FO
 * 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 _FSL_CLOCK_H_
#define _FSL_CLOCK_H_

#include "fsl_common.h"

/*! @addtogroup clock */
/*! @{ */

/*! @file */

/*******************************************************************************
 * Configurations
 ******************************************************************************/

/*! @brief Configures whether to check a parameter in a function.
 *
 * Some MCG settings must be changed with conditions, for example:
 *  1. MCGIRCLK settings, such as the source, divider, and the trim value should not change when
 *     MCGIRCLK is used as a system clock source.
 *  2. MCG_C7[OSCSEL] should not be changed  when the external reference clock is used
 *     as a system clock source. For example, in FBE/BLPE/PBE modes.
 *  3. The users should only switch between the supported clock modes.
 *
 * MCG functions check the parameter and MCG status before setting, if not allowed
 * to change, the functions return error. The parameter checking increases code size,
 * if code size is a critical requirement, change #MCG_CONFIG_CHECK_PARAM to 0 to
 * disable parameter checking.
 */
#ifndef MCG_CONFIG_CHECK_PARAM
#define MCG_CONFIG_CHECK_PARAM 0U
#endif

/*! @brief Configure whether driver controls clock
 *
 * When set to 0, peripheral drivers will enable clock in initialize function
 * and disable clock in de-initialize function. When set to 1, peripheral
 * driver will not control the clock, application could contol the clock out of
 * the driver.
 *
 * @note All drivers share this feature switcher. If it is set to 1, application
 * should handle clock enable and disable for all drivers.
 */
#if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL))
#define FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL 0
#endif

/*******************************************************************************
 * Definitions
 ******************************************************************************/
```

```
/*! @name Driver version */
/*@{*/
/*! @brief CLOCK driver version 2.2.1. */
#define FSL_CLOCK_DRIVER_VERSION (MAKE_VERSION(2, 2, 1))
/*@}*/

/*! @brief External XTAL0 (OSC0) clock frequency.
 *
 * The XTAL0/EXTAL0 (OSC0) clock frequency in Hz. When the clock is set up, use the
 * function CLOCK_SetXtal0Freq to set the value in the clock driver. For example,
 * if XTAL0 is 8 MHz:
 * @code
 * CLOCK_InitOsc0(...); // Set up the OSC0
 * CLOCK_SetXtal0Freq(80000000); // Set the XTAL0 value to the clock driver.
 * @endcode
 *
 * This is important for the multicore platforms where only one core needs to set up the
 * OSC0 using the CLOCK_InitOsc0. All other cores need to call the CLOCK_SetXtal0Freq
 * to get a valid clock frequency.
 */
extern uint32_t g_xtal0Freq;

/*! @brief External XTAL32/EXTAL32/RTC_CLKIN clock frequency.
 *
 * The XTAL32/EXTAL32/RTC_CLKIN clock frequency in Hz. When the clock is set up, use the
 * function CLOCK_SetXtal32Freq to set the value in the clock driver.
 *
 * This is important for the multicore platforms where only one core needs to set up
 * the clock. All other cores need to call the CLOCK_SetXtal32Freq
 * to get a valid clock frequency.
 */
extern uint32_t g_xtal32Freq;

#if (defined(OSC) && !(defined(OSC0)))
#define OSC0 OSC
#endif

/*! @brief Clock ip name array for DMAMUX. */
#define DMAMUX_CLOCKS  \
    {                  \
        kCLOCK_Dmamux0 \
    }

/*! @brief Clock ip name array for RTC. */
#define RTC_CLOCKS  \
    {               \
        kCLOCK_Rtc0 \
    }

/*! @brief Clock ip name array for SPI. */
#define SPI_CLOCKS               \
    {                            \
        kCLOCK_Spi0, kCLOCK_Spi1 \
    }

/*! @brief Clock ip name array for PIT. */
#define PIT_CLOCKS  \
    {               \
        kCLOCK_Pit0 \
    }

/*! @brief Clock ip name array for PORT. */
#define PORT_CLOCKS                                                          \
    {                                                                        \
        kCLOCK_PortA, kCLOCK_PortB, kCLOCK_PortC, kCLOCK_PortD, kCLOCK_PortE \
    }

/*! @brief Clock ip name array for TSI. */
#define TSI_CLOCKS  \
    {               \
        kCLOCK_Tsi0 \
    }

/*! @brief Clock ip name array for DAC. */
#define DAC_CLOCKS  \
    {               \
        kCLOCK_Dac0 \
    }

/*! @brief Clock ip name array for LPTMR. */
#define LPTMR_CLOCKS  \
    {                 \
        kCLOCK_Lptmr0 \
    }

/*! @brief Clock ip name array for ADC16. */
#define ADC16_CLOCKS \
    {                \
        kCLOCK_Adc0  \
    }

/*! @brief Clock ip name array for DMA. */
#define DMA_CLOCKS  \
    {               \
        kCLOCK_Dma0 \
    }

/*! @brief Clock ip name array for LPSCI/UART0. */
#define UART0_CLOCKS \
    {                \
        kCLOCK_Uart0 \
    }

/*! @brief Clock ip name array for UART. */
#define UART_CLOCKS                              \
    {                                            \
        kCLOCK_IpInvalid, kCLOCK_Uart1, kCLOCK_Uart2 \
    }

/*! @brief Clock ip name array for TPM. */
#define TPM_CLOCKS                              \
    {                                          \
        kCLOCK_Tpm0, kCLOCK_Tpm1, kCLOCK_Tpm2 \
    }

/*! @brief Clock ip name array for I2C. */
#define I2C_CLOCKS                \
    {                            \
        kCLOCK_I2c0, kCLOCK_I2c1 \
    }

/*! @brief Clock ip name array for FTF. */
#define FTF_CLOCKS  \
    {              \
        kCLOCK_Ftf0 \
    }

/*! @brief Clock ip name array for CMP. */
#define CMP_CLOCKS  \
    {              \
        kCLOCK_Cmp0 \
    }

/*!
 * @brief LPO clock frequency.
 */
#define LPO_CLK_FREQ 1000U

/*! @brief Peripherals clock source definition. */
#define SYS_CLK kCLOCK_CoreSysClk
#define BUS_CLK kCLOCK_BusClk

#define I2C0_CLK_SRC BUS_CLK
#define I2C1_CLK_SRC BUS_CLK
#define SPI0_CLK_SRC BUS_CLK
#define SPI1_CLK_SRC SYS_CLK
#define UART1_CLK_SRC BUS_CLK
#define UART2_CLK_SRC BUS_CLK

/*! @brief Clock name used to get clock frequency. */
typedef enum _clock_name
{

    /* ----------------------------- System layer clock -------------------------------*/
    kCLOCK_CoreSysClk,   /*!< Core/system clock                                         */
    kCLOCK_PlatClk,      /*!< Platform clock                                            */
    kCLOCK_BusClk,       /*!< Bus clock                                                 */
    kCLOCK_FlexBusClk,   /*!< FlexBus clock                                             */
    kCLOCK_FlashClk,     /*!< Flash clock                                               */
    kCLOCK_PllFllSelClk, /*!< The clock after SIM[PLLFLLSEL].                         */

    /* ---------------------------------- OSC clock -----------------------------------*/
    kCLOCK_Er32kClk,  /*!< External reference 32K clock (ERCLK32K)                 */
    kCLOCK_Osc0ErClk, /*!< OSC0 external reference clock (OSC0ERCLK)                */

    /* ----------------------------- MCG and MCG-Lite clock ---------------------------*/
    kCLOCK_McgFixedFreqClk,   /*!< MCG fixed frequency clock (MCGFFCLK)                 */
    kCLOCK_McgInternalRefClk, /*!< MCG internal reference clock (MCGIRCLK)              */
    kCLOCK_McgFllClk,         /*!< MCGFLLCLK                                            */
    kCLOCK_McgPll0Clk,        /*!< MCGPLL0CLK                                           */
    kCLOCK_McgExtPllClk,      /*!< EXT_PLLCLK                                           */

    /* -------------------------------- Other clock ---------------------------------*/
    kCLOCK_LpoClk, /*!< LPO clock                                                */

} clock_name_t;

/*! @brief USB clock source definition. */
typedef enum _clock_usb_src
{
    kCLOCK_UsbSrcPll0 = SIM_SOPT2_USBSRC(1U) | SIM_SOPT2_PLLFLLSEL(1U), /*!< Use PLL0.
    kCLOCK_UsbSrcExt = SIM_SOPT2_USBSRC(0U)                             /*!< Use USB_CLKIN. */
} clock_usb_src_t;

/*------------------------------------------------------------------------------

 clock_gate_t definition:

 31                              16                              0
 -----------------------------------------------------------------
 | SIM_SCGC register offset       |   control bit offset in SCGC |
 -----------------------------------------------------------------

 For example, the SDHC clock gate is controlled by SIM_SCGC3[17], the
 SIM_SCGC3 offset in SIM is 0x1030, then kCLOCK_GateSdhc0 is defined as

              kCLOCK_GateSdhc0 = (0x1030 << 16) | 17;

------------------------------------------------------------------------------*/

#define CLK_GATE_REG_OFFSET_SHIFT 16U
#define CLK_GATE_REG_OFFSET_MASK 0xFFFF0000U
#define CLK_GATE_BIT_SHIFT_SHIFT 0U
#define CLK_GATE_BIT_SHIFT_MASK 0x0000FFFFU

#define CLK_GATE_DEFINE(reg_offset, bit_shift)                                  \
    ((((reg_offset) << CLK_GATE_REG_OFFSET_SHIFT) & CLK_GATE_REG_OFFSET_MASK) | \
     (((bit_shift) << CLK_GATE_BIT_SHIFT_SHIFT) & CLK_GATE_BIT_SHIFT_MASK))

#define CLK_GATE_ABSTRACT_REG_OFFSET(x) (((x)&CLK_GATE_REG_OFFSET_MASK) >> CLK_G
#define CLK_GATE_ABSTRACT_BITS_SHIFT(x) (((x)&CLK_GATE_BIT_SHIFT_MASK) >> CLK_GATE_
/*! @brief Clock gate name used for CLOCK_EnableClock/CLOCK_DisableClock. */
typedef enum _clock_ip_name
{
    kCLOCK_IpInvalid = 0U,
    kCLOCK_I2c0 = CLK_GATE_DEFINE(0x1034U, 6U),
    kCLOCK_I2c1 = CLK_GATE_DEFINE(0x1034U, 7U),
    kCLOCK_Uart0 = CLK_GATE_DEFINE(0x1034U, 10U),
    kCLOCK_Uart1 = CLK_GATE_DEFINE(0x1034U, 11U),
    kCLOCK_Uart2 = CLK_GATE_DEFINE(0x1034U, 12U),
    kCLOCK_Usbfs0 = CLK_GATE_DEFINE(0x1034U, 18U),
    kCLOCK_Cmp0 = CLK_GATE_DEFINE(0x1034U, 19U),
    kCLOCK_Spi0 = CLK_GATE_DEFINE(0x1034U, 22U),
    kCLOCK_Spi1 = CLK_GATE_DEFINE(0x1034U, 23U),

    kCLOCK_Lptmr0 = CLK_GATE_DEFINE(0x1038U, 0U),
    kCLOCK_Tsi0 = CLK_GATE_DEFINE(0x1038U, 5U),
    kCLOCK_PortA = CLK_GATE_DEFINE(0x1038U, 9U),
    kCLOCK_PortB = CLK_GATE_DEFINE(0x1038U, 10U),
    kCLOCK_PortC = CLK_GATE_DEFINE(0x1038U, 11U),
    kCLOCK_PortD = CLK_GATE_DEFINE(0x1038U, 12U),
    kCLOCK_PortE = CLK_GATE_DEFINE(0x1038U, 13U),

    kCLOCK_Ftf0 = CLK_GATE_DEFINE(0x103CU, 0U),
    kCLOCK_Dmamux0 = CLK_GATE_DEFINE(0x103CU, 1U),
    kCLOCK_Pit0 = CLK_GATE_DEFINE(0x103CU, 23U),
    kCLOCK_Tpm0 = CLK_GATE_DEFINE(0x103CU, 24U),
    kCLOCK_Tpm1 = CLK_GATE_DEFINE(0x103CU, 25U),
    kCLOCK_Tpm2 = CLK_GATE_DEFINE(0x103CU, 26U),
    kCLOCK_Adc0 = CLK_GATE_DEFINE(0x103CU, 27U),
    kCLOCK_Rtc0 = CLK_GATE_DEFINE(0x103CU, 29U),
    kCLOCK_Dac0 = CLK_GATE_DEFINE(0x103CU, 31U),

    kCLOCK_Dma0 = CLK_GATE_DEFINE(0x1040U, 8U),
} clock_ip_name_t;

/*!@brief SIM configuration structure for clock setting. */
typedef struct _sim_clock_config
{
    uint8_t pllFllSel;
    uint8_t er32kSrc; /*!< ERCLK32K source selection.   */
    uint32_t clkdiv1; /*!< SIM_CLKDIV1.                 */
} sim_clock_config_t;

/*! @brief OSC work mode. */
typedef enum _osc_mode
{
    kOSC_ModeExt = 0U, /*!< Use an external clock.   */
#if (defined(MCG_C2_EREFS_MASK) && !(defined(MCG_C2_EREFS0_MASK)))
    kOSC_ModeOscLowPower = MCG_C2_EREFS_MASK, /*!< Oscillator low power. */
#else
    kOSC_ModeOscLowPower = MCG_C2_EREFS0_MASK, /*!< Oscillator low power. */
#endif
    kOSC_ModeOscHighGain = 0U
```

```
#if (defined(MCG_C2_EREFS_MASK) && !(defined(MCG_C2_EREFS0_MASK)))
                      |
                      MCG_C2_EREFS_MASK
#else
                      |
                      MCG_C2_EREFS0_MASK
#endif
#if (defined(MCG_C2_HGO_MASK) && !(defined(MCG_C2_HGO0_MASK)))
                      |
                      MCG_C2_HGO_MASK, /*!< Oscillator high gain. */
#else
                      |
                      MCG_C2_HGO0_MASK, /*!< Oscillator high gain. */
#endif
} osc_mode_t;

/*! @brief Oscillator capacitor load setting.*/
enum _osc_cap_load
{
   kOSC_Cap2P = OSC_CR_SC2P_MASK,  /*!< 2  pF capacitor load */
   kOSC_Cap4P = OSC_CR_SC4P_MASK,  /*!< 4  pF capacitor load */
   kOSC_Cap8P = OSC_CR_SC8P_MASK,  /*!< 8  pF capacitor load */
   kOSC_Cap16P = OSC_CR_SC16P_MASK /*!< 16 pF capacitor load */
};

/*! @brief OSCERCLK enable mode. */
enum _oscer_enable_mode
{
   kOSC_ErClkEnable = OSC_CR_ERCLKEN_MASK,       /*!< Enable.              */
   kOSC_ErClkEnableInStop = OSC_CR_EREFSTEN_MASK /*!< Enable in stop mode. */
};

/*! @brief OSC configuration for OSCERCLK. */
typedef struct _oscer_config
{
   uint8_t enableMode; /*!< OSCERCLK enable mode. OR'ed value of @ref _oscer_enable_mode. */

} oscer_config_t;

/*!
 * @brief OSC Initialization Configuration Structure
 *
 * Defines the configuration data structure to initialize the OSC.
 * When porting to a new board, set the following members
 * according to the board setting:
 * 1. freq: The external frequency.
 * 2. workMode: The OSC module mode.
 */
typedef struct _osc_config
{
   uint32_t freq;              /*!< External clock frequency.    */
   uint8_t capLoad;            /*!< Capacitor load setting.      */
   osc_mode_t workMode;        /*!< OSC work mode setting.       */
```

```
    oscer_config_t oscerConfig; /*!< Configuration for OSCERCLK.  */
} osc_config_t;

/*! @brief MCG FLL reference clock source select. */
typedef enum _mcg_fll_src
{
    kMCG_FllSrcExternal, /*!< External reference clock is selected          */
    kMCG_FllSrcInternal  /*!< The slow internal reference clock is selected */
} mcg_fll_src_t;

/*! @brief MCG internal reference clock select */
typedef enum _mcg_irc_mode
{
    kMCG_IrcSlow, /*!< Slow internal reference clock selected */
    kMCG_IrcFast  /*!< Fast internal reference clock selected */
} mcg_irc_mode_t;

/*! @brief MCG DCO Maximum Frequency with 32.768 kHz Reference */
typedef enum _mcg_dmx32
{
    kMCG_Dmx32Default, /*!< DCO has a default range of 25% */
    kMCG_Dmx32Fine     /*!< DCO is fine-tuned for maximum frequency with 32.768 kHz reference */
} mcg_dmx32_t;

/*! @brief MCG DCO range select */
typedef enum _mcg_drs
{
    kMCG_DrsLow,     /*!< Low frequency range       */
    kMCG_DrsMid,     /*!< Mid frequency range       */
    kMCG_DrsMidHigh, /*!< Mid-High frequency range  */
    kMCG_DrsHigh     /*!< High frequency range      */
} mcg_drs_t;

/*! @brief MCG PLL reference clock select */
typedef enum _mcg_pll_ref_src
{
    kMCG_PllRefOsc0, /*!< Selects OSC0 as PLL reference clock                 */
    kMCG_PllRefOsc1  /*!< Selects OSC1 as PLL reference clock                 */
} mcg_pll_ref_src_t;

/*! @brief MCGOUT clock source. */
typedef enum _mcg_clkout_src
{
    kMCG_ClkOutSrcOut,      /*!< Output of the FLL is selected (reset default)  */
    kMCG_ClkOutSrcInternal, /*!< Internal reference clock is selected           */
    kMCG_ClkOutSrcExternal, /*!< External reference clock is selected           */
} mcg_clkout_src_t;

/*! @brief MCG Automatic Trim Machine Select */
typedef enum _mcg_atm_select
{
    kMCG_AtmSel32k, /*!< 32 kHz Internal Reference Clock selected  */
    kMCG_AtmSel4m   /*!< 4 MHz Internal Reference Clock selected   */
```

```
} mcg_atm_select_t;

/*! @brief MCG OSC Clock Select */
typedef enum _mcg_oscsel
{
    kMCG_OscselOsc, /*!< Selects System Oscillator (OSCCLK) */
    kMCG_OscselRtc, /*!< Selects 32 kHz RTC Oscillator      */
} mcg_oscsel_t;

/*! @brief MCG PLLCS select */
typedef enum _mcg_pll_clk_select
{
    kMCG_PllClkSelPll0, /*!< PLL0 output clock is selected  */
    kMCG_PllClkSelPll1  /* PLL1 output clock is selected    */
} mcg_pll_clk_select_t;

/*! @brief MCG clock monitor mode. */
typedef enum _mcg_monitor_mode
{
    kMCG_MonitorNone, /*!< Clock monitor is disabled.         */
    kMCG_MonitorInt,  /*!< Trigger interrupt when clock lost. */
    kMCG_MonitorReset /*!< System reset when clock lost.      */
} mcg_monitor_mode_t;

/*! @brief MCG status. */
enum _mcg_status
{
    kStatus_MCG_ModeUnreachable = MAKE_STATUS(kStatusGroup_MCG, 0),       /*!< Can't switch to ta
    kStatus_MCG_ModeInvalid = MAKE_STATUS(kStatusGroup_MCG, 1),           /*!< Current mode invalid
                                                                          function. */
    kStatus_MCG_AtmBusClockInvalid = MAKE_STATUS(kStatusGroup_MCG, 2),    /*!< Invalid bus clock f
    kStatus_MCG_AtmDesiredFreqInvalid = MAKE_STATUS(kStatusGroup_MCG, 3), /*!< Invalid desired fr
    kStatus_MCG_AtmIrcUsed = MAKE_STATUS(kStatusGroup_MCG, 4),            /*!< IRC is used when us
    kStatus_MCG_AtmHardwareFail = MAKE_STATUS(kStatusGroup_MCG, 5),       /*!< Hardware fail occu
    kStatus_MCG_SourceUsed = MAKE_STATUS(kStatusGroup_MCG, 6)             /*!< Can't change the clo
                                                                          it is in use. */
};

/*! @brief MCG status flags. */
enum _mcg_status_flags_t
{
    kMCG_Osc0LostFlag = (1U << 0U), /*!< OSC0 lost.         */
    kMCG_Osc0InitFlag = (1U << 1U), /*!< OSC0 crystal initialized. */
    kMCG_Pll0LostFlag = (1U << 5U), /*!< PLL0 lost.         */
    kMCG_Pll0LockFlag = (1U << 6U), /*!< PLL0 locked.       */
};

/*! @brief MCG internal reference clock (MCGIRCLK) enable mode definition. */
enum _mcg_irclk_enable_mode
{
    kMCG_IrclkEnable = MCG_C1_IRCLKEN_MASK,       /*!< MCGIRCLK enable.              */
    kMCG_IrclkEnableInStop = MCG_C1_IREFSTEN_MASK /*!< MCGIRCLK enable in stop mode. */
};
```

```
/*! @brief MCG PLL clock enable mode definition. */
enum _mcg_pll_enable_mode
{
   kMCG_PllEnableIndependent = MCG_C5_PLLCLKEN0_MASK, /*!< MCGPLLCLK enable independent
                                                    MCG clock mode. Generally, the PLL
                                                    is disabled in FLL modes
                                                    (FEI/FBI/FEE/FBE). Setting the PLL clock
                                                    enable independent, enables the
                                                    PLL in the FLL modes.          */
   kMCG_PllEnableInStop = MCG_C5_PLLSTEN0_MASK      /*!< MCGPLLCLK enable in STOP mode. *
};

/*! @brief MCG mode definitions */
typedef enum _mcg_mode
{
   kMCG_ModeFEI = 0U, /*!< FEI   - FLL Engaged Internal         */
   kMCG_ModeFBI,      /*!< FBI   - FLL Bypassed Internal        */
   kMCG_ModeBLPI,     /*!< BLPI  - Bypassed Low Power Internal  */
   kMCG_ModeFEE,      /*!< FEE   - FLL Engaged External         */
   kMCG_ModeFBE,      /*!< FBE   - FLL Bypassed External        */
   kMCG_ModeBLPE,     /*!< BLPE  - Bypassed Low Power External  */
   kMCG_ModePBE,      /*!< PBE   - PLL Bypassed External        */
   kMCG_ModePEE,      /*!< PEE   - PLL Engaged External         */
   kMCG_ModeError     /*!< Unknown mode                         */
} mcg_mode_t;

/*! @brief MCG PLL configuration. */
typedef struct _mcg_pll_config
{
   uint8_t enableMode; /*!< Enable mode. OR'ed value of @ref _mcg_pll_enable_mode. */
   uint8_t prdiv;      /*!< Reference divider PRDIV.    */
   uint8_t vdiv;       /*!< VCO divider VDIV.           */
} mcg_pll_config_t;

/*! @brief MCG mode change configuration structure
 *
 * When porting to a new board, set the following members
 * according to the board setting:
 * 1. frdiv: If the FLL uses the external reference clock, set this
 *    value to ensure that the external reference clock divided by frdiv is
 *    in the 31.25 kHz to 39.0625 kHz range.
 * 2. The PLL reference clock divider PRDIV: PLL reference clock frequency after
 *    PRDIV should be in the FSL_FEATURE_MCG_PLL_REF_MIN to
 *    FSL_FEATURE_MCG_PLL_REF_MAX range.
 */
typedef struct _mcg_config
{
   mcg_mode_t mcgMode; /*!< MCG mode.                   */

   /* ---------------------- MCGIRCCLK settings ------------------------ */
   uint8_t irclkEnableMode; /*!< MCGIRCLK enable mode.       */
   mcg_irc_mode_t ircs;     /*!< Source, MCG_C2[IRCS].       */
```

```
    uint8_t fcrdiv;          /*!< Divider, MCG_SC[FCRDIV].    */

    /* ------------------------ MCG FLL settings ------------------------- */
    uint8_t frdiv;      /*!< Divider MCG_C1[FRDIV].      */
    mcg_drs_t drs;      /*!< DCO range MCG_C4[DRST_DRS]. */
    mcg_dmx32_t dmx32; /*!< MCG_C4[DMX32].              */

    /* ------------------------ MCG PLL settings ------------------------- */
    mcg_pll_config_t pll0Config; /*!< MCGPLL0CLK configuration.   */

} mcg_config_t;

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*!
 * @brief Enable the clock for specific IP.
 *
 * @param name  Which clock to enable, see \ref clock_ip_name_t.
 */
static inline void CLOCK_EnableClock(clock_ip_name_t name)
{
    uint32_t regAddr = SIM_BASE + CLK_GATE_ABSTRACT_REG_OFFSET((uint32_t)name);
    (*(volatile uint32_t *)regAddr) |= (1U << CLK_GATE_ABSTRACT_BITS_SHIFT((uint32_t)name));
}

/*!
 * @brief Disable the clock for specific IP.
 *
 * @param name  Which clock to disable, see \ref clock_ip_name_t.
 */
static inline void CLOCK_DisableClock(clock_ip_name_t name)
{
    uint32_t regAddr = SIM_BASE + CLK_GATE_ABSTRACT_REG_OFFSET((uint32_t)name);
    (*(volatile uint32_t *)regAddr) &= ~(1U << CLK_GATE_ABSTRACT_BITS_SHIFT((uint32_t)name));
}

/*! @brief Set ERCLK32K source. */
static inline void CLOCK_SetEr32kClock(uint32_t src)
{
    SIM->SOPT1 = ((SIM->SOPT1 & ~SIM_SOPT1_OSC32KSEL_MASK) | SIM_SOPT1_OSC32KSEL(src
}

/*! @brief Set PLLFLLSEL clock source. */
static inline void CLOCK_SetPllFllSelClock(uint32_t src)
{
    SIM->SOPT2 = ((SIM->SOPT2 & ~SIM_SOPT2_PLLFLLSEL_MASK) | SIM_SOPT2_PLLFLLSEL(src));
}
```

```
/*! @brief Set TPM clock source. */
static inline void CLOCK_SetTpmClock(uint32_t src)
{
    SIM->SOPT2 = ((SIM->SOPT2 & ~SIM_SOPT2_TPMSRC_MASK) | SIM_SOPT2_TPMSRC(src));
}

/*! @brief Set LPSCI0 (UART0) clock source. */
static inline void CLOCK_SetLpsci0Clock(uint32_t src)
{
    SIM->SOPT2 = ((SIM->SOPT2 & ~SIM_SOPT2_UART0SRC_MASK) | SIM_SOPT2_UART0SRC(src));
}

/*! @brief Enable USB FS clock.
 *
 * @param src  USB FS clock source.
 * @param freq The frequency specified by src.
 * @retval true The clock is set successfully.
 * @retval false The clock source is invalid to get proper USB FS clock.
 */
bool CLOCK_EnableUsbfs0Clock(clock_usb_src_t src, uint32_t freq);

/*! @brief Disable USB FS clock.
 *
 * Disable USB FS clock.
 */
static inline void CLOCK_DisableUsbfs0Clock(void)
{
    CLOCK_DisableClock(kCLOCK_Usbfs0);
}

/*! @brief Set CLKOUT source. */
static inline void CLOCK_SetClkOutClock(uint32_t src)
{
    SIM->SOPT2 = ((SIM->SOPT2 & ~SIM_SOPT2_CLKOUTSEL_MASK) | SIM_SOPT2_CLKOUTSEL(src
}

/*! @brief Set RTC_CLKOUT source. */
static inline void CLOCK_SetRtcClkOutClock(uint32_t src)
{
    SIM->SOPT2 = ((SIM->SOPT2 & ~SIM_SOPT2_RTCCLKOUTSEL_MASK) | SIM_SOPT2_RTCCLKOU
}

/*!
 * @brief
 * Set the SIM_CLKDIV1[OUTDIV1], SIM_CLKDIV1[OUTDIV4].
 */
static inline void CLOCK_SetOutDiv(uint32_t outdiv1, uint32_t outdiv4)
{
    SIM->CLKDIV1 = SIM_CLKDIV1_OUTDIV1(outdiv1) | SIM_CLKDIV1_OUTDIV4(outdiv4);
}

/*!
```

```
 * @brief Gets the clock frequency for a specific clock name.
 *
 * This function checks the current clock configurations and then calculates
 * the clock frequency for a specific clock name defined in clock_name_t.
 * The MCG must be properly configured before using this function.
 *
 * @param clockName Clock names defined in clock_name_t
 * @return Clock frequency value in Hertz
 */
uint32_t CLOCK_GetFreq(clock_name_t clockName);

/*!
 * @brief Get the core clock or system clock frequency.
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetCoreSysClkFreq(void);

/*!
 * @brief Get the platform clock frequency.
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetPlatClkFreq(void);

/*!
 * @brief Get the bus clock frequency.
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetBusClkFreq(void);

/*!
 * @brief Get the flash clock frequency.
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetFlashClkFreq(void);

/*!
 * @brief Get the output clock frequency selected by SIM[PLLFLLSEL].
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetPllFllSelClkFreq(void);

/*!
 * @brief Get the external reference 32K clock frequency (ERCLK32K).
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetEr32kClkFreq(void);
```

```
/*!
 * @brief Get the OSC0 external reference clock frequency (OSC0ERCLK).
 *
 * @return Clock frequency in Hz.
 */
uint32_t CLOCK_GetOsc0ErClkFreq(void);

/*!
 * @brief Set the clock configure in SIM module.
 *
 * This function sets system layer clock settings in SIM module.
 *
 * @param config Pointer to the configure structure.
 */
void CLOCK_SetSimConfig(sim_clock_config_t const *config);

/*!
 * @brief Set the system clock dividers in SIM to safe value.
 *
 * The system level clocks (core clock, bus clock, flexbus clock and flash clock)
 * must be in allowed ranges. During MCG clock mode switch, the MCG output clock
 * changes then the system level clocks may be out of range. This function could
 * be used before MCG mode change, to make sure system level clocks are in allowed
 * range.
 *
 * @param config Pointer to the configure structure.
 */
static inline void CLOCK_SetSimSafeDivs(void)
{
    SIM->CLKDIV1 = 0x10030000U;
}

/*! @name MCG frequency functions. */
/*@{*/

/*!
 * @brief Gets the MCG output clock (MCGOUTCLK) frequency.
 *
 * This function gets the MCG output clock frequency in Hz based on the current MCG
 * register value.
 *
 * @return The frequency of MCGOUTCLK.
 */
uint32_t CLOCK_GetOutClkFreq(void);

/*!
 * @brief Gets the MCG FLL clock (MCGFLLCLK) frequency.
 *
 * This function gets the MCG FLL clock frequency in Hz based on the current MCG
 * register value. The FLL is enabled in FEI/FBI/FEE/FBE mode and
 * disabled in low power state in other modes.
 *
 * @return The frequency of MCGFLLCLK.
 */
uint32_t CLOCK_GetFllFreq(void);

/*!
 * @brief Gets the MCG internal reference clock (MCGIRCLK) frequency.
 *
 * This function gets the MCG internal reference clock frequency in Hz based
 * on the current MCG register value.
 *
 * @return The frequency of MCGIRCLK.
 */
uint32_t CLOCK_GetInternalRefClkFreq(void);

/*!
 * @brief Gets the MCG fixed frequency clock (MCGFFCLK) frequency.
 *
 * This function gets the MCG fixed frequency clock frequency in Hz based
 * on the current MCG register value.
 *
 * @return The frequency of MCGFFCLK.
 */
uint32_t CLOCK_GetFixedFreqClkFreq(void);

/*!
 * @brief Gets the MCG PLL0 clock (MCGPLL0CLK) frequency.
 *
 * This function gets the MCG PLL0 clock frequency in Hz based on the current MCG
 * register value.
 *
 * @return The frequency of MCGPLL0CLK.
 */
uint32_t CLOCK_GetPll0Freq(void);

/*@}*/

/*! @name MCG clock configuration. */
/*@{*/

/*!
 * @brief Enables or disables the MCG low power.
 *
 * Enabling the MCG low power disables the PLL and FLL in bypass modes. In other words,
 * in FBE and PBE modes, enabling low power sets the MCG to BLPE mode. In FBI and
 * PBI modes, enabling low power sets the MCG to BLPI mode.
 * When disabling the MCG low power, the PLL or FLL are enabled based on MCG settings.
 *
 * @param enable True to enable MCG low power, false to disable MCG low power.
 */
static inline void CLOCK_SetLowPowerEnable(bool enable)
{
    if (enable)
    {
        MCG->C2 |= MCG_C2_LP_MASK;
    }
    else
    {
        MCG->C2 &= ~MCG_C2_LP_MASK;
    }
}

/*!
 * @brief Configures the Internal Reference clock (MCGIRCLK).
 *
 * This function sets the \c MCGIRCLK base on parameters. It also selects the IRC
 * source. If the fast IRC is used, this function sets the fast IRC divider.
 * This function also sets whether the \c MCGIRCLK is enabled in stop mode.
 * Calling this function in FBI/PBI/BLPI modes may change the system clock. As a result,
 * using the function in these modes it is not allowed.
 *
 * @param enableMode MCGIRCLK enable mode, OR'ed value of @ref _mcg_irclk_enable_mode.
 * @param ircs       MCGIRCLK clock source, choose fast or slow.
 * @param fcrdiv     Fast IRC divider setting (\c FCRDIV).
 * @retval kStatus_MCG_SourceUsed Because the internall reference clock is used as a clock source,
 * the confuration should not be changed. Otherwise, a glitch occurs.
 * @retval kStatus_Success MCGIRCLK configuration finished successfully.
 */
status_t CLOCK_SetInternalRefClkConfig(uint8_t enableMode, mcg_irc_mode_t ircs, uint8_t fcrdiv);

/*!
 * @brief Selects the MCG external reference clock.
 *
 * Selects the MCG external reference clock source, changes the MCG_C7[OSCSEL],
 * and waits for the clock source to be stable. Because the external reference
 * clock should not be changed in FEE/FBE/BLPE/PBE/PEE modes, do not call this function in these mode
 *
 * @param oscsel MCG external reference clock source, MCG_C7[OSCSEL].
 * @retval kStatus_MCG_SourceUsed Because the external reference clock is used as a clock source,
 * the confuration should not be changed. Otherwise, a glitch occurs.
 * @retval kStatus_Success External reference clock set successfully.
 */
status_t CLOCK_SetExternalRefClkConfig(mcg_oscsel_t oscsel);

/*!
 * @brief Set the FLL external reference clock divider value.
 *
 * Sets the FLL external reference clock divider value, the register MCG_C1[FRDIV].
 *
 * @param frdiv The FLL external reference clock divider value, MCG_C1[FRDIV].
 */
static inline void CLOCK_SetFllExtRefDiv(uint8_t frdiv)
{
    MCG->C1 = (MCG->C1 & ~MCG_C1_FRDIV_MASK) | MCG_C1_FRDIV(frdiv);
}

/*!
 * @brief Enables the PLL0 in FLL mode.
 *
 * This function sets us the PLL0 in FLL mode and reconfigures
 * the PLL0. Ensure that the PLL reference
 * clock is enabled before calling this function and that the PLL0 is not used as a clock source.
 * The function CLOCK_CalcPllDiv gets the correct PLL
 * divider values.
 *
 * @param config Pointer to the configuration structure.
 */
void CLOCK_EnablePll0(mcg_pll_config_t const *config);

/*!
 * @brief Disables the PLL0 in FLL mode.
 *
 * This function disables the PLL0 in FLL mode. It should be used together with the
 * @ref CLOCK_EnablePll0.
 */
static inline void CLOCK_DisablePll0(void)
{
    MCG->C5 &= ~(MCG_C5_PLLCLKEN0_MASK | MCG_C5_PLLSTEN0_MASK);
}

/*!
 * @brief Calculates the PLL divider setting for a desired output frequency.
 *
 * This function calculates the correct reference clock divider (\c PRDIV) and
 * VCO divider (\c VDIV) to generate a desired PLL output frequency. It returns the
 * closest frequency match with the corresponding \c PRDIV/VDIV
 * returned from parameters. If a desired frequency is not valid, this function
 * returns 0.
 *
 * @param refFreq    PLL reference clock frequency.
 * @param desireFreq Desired PLL output frequency.
 * @param prdiv      PRDIV value to generate desired PLL frequency.
 * @param vdiv       VDIV value to generate desired PLL frequency.
 * @return Closest frequency match that the PLL was able generate.
 */
uint32_t CLOCK_CalcPllDiv(uint32_t refFreq, uint32_t desireFreq, uint8_t *prdiv, uint8_t *vdiv);

/*@}*/

/*! @name MCG clock lock monitor functions. */
/*@{*/

/*!
 * @brief Sets the OSC0 clock monitor mode.
 *
 * This function sets the OSC0 clock monitor mode. See @ref mcg_monitor_mode_t for details.
 *
 * @param mode Monitor mode to set.
 */
void CLOCK_SetOsc0MonitorMode(mcg_monitor_mode_t mode);
```

```
/*!
 * @brief Sets the PLL0 clock monitor mode.
 *
 * This function sets the PLL0 clock monitor mode. See @ref mcg_monitor_mode_t for details.
 *
 * @param mode Monitor mode to set.
 */
void CLOCK_SetPll0MonitorMode(mcg_monitor_mode_t mode);

/*!
 * @brief Gets the MCG status flags.
 *
 * This function gets the MCG clock status flags. All status flags are
 * returned as a logical OR of the enumeration @ref _mcg_status_flags_t. To
 * check a specific flag, compare the return value with the flag.
 *
 * Example:
 * @code
   // To check the clock lost lock status of OSC0 and PLL0.
   uint32_t mcgFlags;

   mcgFlags = CLOCK_GetStatusFlags();

   if (mcgFlags & kMCG_Osc0LostFlag)
   {
       // OSC0 clock lock lost. Do something.
   }
   if (mcgFlags & kMCG_Pll0LostFlag)
   {
       // PLL0 clock lock lost. Do something.
   }
   @endcode
 *
 * @return  Logical OR value of the @ref _mcg_status_flags_t.
 */
uint32_t CLOCK_GetStatusFlags(void);

/*!
 * @brief Clears the MCG status flags.
 *
 * This function clears the MCG clock lock lost status. The parameter is a logical
 * OR value of the flags to clear. See @ref _mcg_status_flags_t.
 *
 * Example:
 * @code
   // To clear the clock lost lock status flags of OSC0 and PLL0.

   CLOCK_ClearStatusFlags(kMCG_Osc0LostFlag | kMCG_Pll0LostFlag);
   @endcode
 *
 * @param mask The status flags to clear. This is a logical OR of members of the
 *             enumeration @ref _mcg_status_flags_t.
 */
```

```
void CLOCK_ClearStatusFlags(uint32_t mask);

/*@}*/

/*!
 * @name OSC configuration
 * @{
 */

/*!
 * @brief Configures the OSC external reference clock (OSCERCLK).
 *
 * This function configures the OSC external reference clock (OSCERCLK).
 * This is an example to enable the OSCERCLK in normal and stop modes and also set
 * the output divider to 1:
 *
   @code
   oscer_config_t config =
   {
       .enableMode = kOSC_ErClkEnable | kOSC_ErClkEnableInStop,
       .erclkDiv   = 1U,
   };

   OSC_SetExtRefClkConfig(OSC, &config);
   @endcode
 *
 * @param base   OSC peripheral address.
 * @param config Pointer to the configuration structure.
 */
static inline void OSC_SetExtRefClkConfig(OSC_Type *base, oscer_config_t const *config)
{
    uint8_t reg = base->CR;

    reg &= ~(OSC_CR_ERCLKEN_MASK | OSC_CR_EREFSTEN_MASK);
    reg |= config->enableMode;

    base->CR = reg;
}

/*!
 * @brief Sets the capacitor load configuration for the oscillator.
 *
 * This function sets the specified capacitors configuration for the oscillator.
 * This should be done in the early system level initialization function call
 * based on the system configuration.
 *
 * @param base   OSC peripheral address.
 * @param capLoad OR'ed value for the capacitor load option, see \ref _osc_cap_load.
 *
 * Example:
   @code
   // To enable only 2 pF and 8 pF capacitor load, please use like this.
   OSC_SetCapLoad(OSC, kOSC_Cap2P | kOSC_Cap8P);
   @endcode
 */
static inline void OSC_SetCapLoad(OSC_Type *base, uint8_t capLoad)
{
    uint8_t reg = base->CR;

    reg &= ~(OSC_CR_SC2P_MASK | OSC_CR_SC4P_MASK | OSC_CR_SC8P_MASK | OSC_CR_SC16
    reg |= capLoad;

    base->CR = reg;
}

/*!
 * @brief Initializes the OSC0.
 *
 * This function initializes the OSC0 according to the board configuration.
 *
 * @param  config Pointer to the OSC0 configuration structure.
 */
void CLOCK_InitOsc0(osc_config_t const *config);

/*!
 * @brief Deinitializes the OSC0.
 *
 * This function deinitializes the OSC0.
 */
void CLOCK_DeinitOsc0(void);

/* @} */

/*!
 * @name External clock frequency
 * @{
 */

/*!
 * @brief Sets the XTAL0 frequency based on board settings.
 *
 * @param freq The XTAL0/EXTAL0 input clock frequency in Hz.
 */
static inline void CLOCK_SetXtal0Freq(uint32_t freq)
{
    g_xtal0Freq = freq;
}

/*!
 * @brief Sets the XTAL32/RTC_CLKIN frequency based on board settings.
 *
 * @param freq The XTAL32/EXTAL32/RTC_CLKIN input clock frequency in Hz.
 */
static inline void CLOCK_SetXtal32Freq(uint32_t freq)
{
    g_xtal32Freq = freq;
```

```
}
/* @} */

/*!
 * @name MCG auto-trim machine.
 * @{
 */

/*!
 * @brief Auto trims the internal reference clock.
 *
 * This function trims the internal reference clock by using the external clock. If
 * successful, it returns the kStatus_Success and the frequency after
 * trimming is received in the parameter @p actualFreq. If an error occurs,
 * the error code is returned.
 *
 * @param extFreq      External clock frequency, which should be a bus clock.
 * @param desireFreq   Frequency to trim to.
 * @param actualFreq   Actual frequency after trimming.
 * @param atms         Trim fast or slow internal reference clock.
 * @retval kStatus_Success ATM success.
 * @retval kStatus_MCG_AtmBusClockInvalid The bus clock is not in allowed range for the ATM.
 * @retval kStatus_MCG_AtmDesiredFreqInvalid MCGIRCLK could not be trimmed to the desired frequenc
 * @retval kStatus_MCG_AtmIrcUsed Could not trim because MCGIRCLK is used as a bus clock source.
 * @retval kStatus_MCG_AtmHardwareFail Hardware fails while trimming.
 */
status_t CLOCK_TrimInternalRefClk(uint32_t extFreq, uint32_t desireFreq, uint32_t *actualFreq, mcg_atm
/* @} */

/*! @name MCG mode functions. */
/*@{*/

/*!
 * @brief Gets the current MCG mode.
 *
 * This function checks the MCG registers and determines the current MCG mode.
 *
 * @return Current MCG mode or error code; See @ref mcg_mode_t.
 */
mcg_mode_t CLOCK_GetMode(void);

/*!
 * @brief Sets the MCG to FEI mode.
 *
 * This function sets the MCG to FEI mode. If setting to FEI mode fails
 * from the current mode, this function returns an error.
 *
 * @param       dmx32  DMX32 in FEI mode.
 * @param       drs The DCO range selection.
 * @param       fllStableDelay Delay function to  ensure that the FLL is stable. Passing
 *              NULL does not cause a delay.
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 * @note If @p dmx32 is set to kMCG_Dmx32Fine, the slow IRC must not be trimmed
 * to a frequency above 32768 Hz.
 */
status_t CLOCK_SetFeiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void));

/*!
 * @brief Sets the MCG to FEE mode.
 *
 * This function sets the MCG to FEE mode. If setting to FEE mode fails
 * from the current mode, this function returns an error.
 *
 * @param   frdiv  FLL reference clock divider setting, FRDIV.
 * @param   dmx32  DMX32 in FEE mode.
 * @param   drs    The DCO range selection.
 * @param   fllStableDelay Delay function to make sure FLL is stable. Passing
 *          NULL does not cause a delay.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_SetFeeMode(uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(vo

/*!
 * @brief Sets the MCG to FBI mode.
 *
 * This function sets the MCG to FBI mode. If setting to FBI mode fails
 * from the current mode, this function returns an error.
 *
 * @param  dmx32  DMX32 in FBI mode.
 * @param  drs  The DCO range selection.
 * @param  fllStableDelay Delay function to make sure FLL is stable. If the FLL
 *         is not used in FBI mode, this parameter can be NULL. Passing
 *         NULL does not cause a delay.
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 * @note If @p dmx32 is set to kMCG_Dmx32Fine, the slow IRC must not be trimmed
 * to frequency above 32768 Hz.
 */
status_t CLOCK_SetFbiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void));

/*!
 * @brief Sets the MCG to FBE mode.
 *
 * This function sets the MCG to FBE mode. If setting to FBE mode fails
 * from the current mode, this function returns an error.
 *
 * @param   frdiv  FLL reference clock divider setting, FRDIV.
 * @param   dmx32  DMX32 in FBE mode.
 * @param   drs    The DCO range selection.
 * @param   fllStableDelay Delay function to make sure FLL is stable. If the FLL
 *          is not used in FBE mode, this parameter can be NULL. Passing NULL
 *          does not cause a delay.
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_SetFbeMode(uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(vo

/*!
 * @brief Sets the MCG to BLPI mode.
 *
 * This function sets the MCG to BLPI mode. If setting to BLPI mode fails
 * from the current mode, this function returns an error.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_SetBlpiMode(void);

/*!
 * @brief Sets the MCG to BLPE mode.
 *
 * This function sets the MCG to BLPE mode. If setting to BLPE mode fails
 * from the current mode, this function returns an error.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_SetBlpeMode(void);

/*!
 * @brief Sets the MCG to PBE mode.
 *
 * This function sets the MCG to PBE mode. If setting to PBE mode fails
 * from the current mode, this function returns an error.
 *
 * @param   pllcs  The PLL selection, PLLCS.
 * @param   config Pointer to the PLL configuration.
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 *
 * @note
 * 1. The parameter \c pllcs selects the PLL. For platforms with
 * only one PLL, the parameter pllcs is kept for interface compatibility.
 * 2. The parameter \c config is the PLL configuration structure. On some
 * platforms,  it is possible to choose the external PLL directly, which renders the
 * configuration structure not necessary. In this case, pass in NULL.
 * For example: CLOCK_SetPbeMode(kMCG_OscselOsc, kMCG_PllClkSelExtPll, NULL);
 */
status_t CLOCK_SetPbeMode(mcg_pll_clk_select_t pllcs, mcg_pll_config_t const *config);

/*!
 * @brief Sets the MCG to PEE mode.
 *
 * This function sets the MCG to PEE mode.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 *
 * @note This function only changes the CLKS to use the PLL/FLL output. If the
 *       PRDIV/VDIV are different than in the PBE mode, set them up
 *       in PBE mode and wait. When the clock is stable, switch to PEE mode.
 */
status_t CLOCK_SetPeeMode(void);

/*!
 * @brief Switches the MCG to FBE mode from the external mode.
 *
 * This function switches the MCG from external modes (PEE/PBE/BLPE/FEE) to the FBE mode quickly.
 * The external clock is used as the system clock souce and PLL is disabled. However,
 * the FLL settings are not configured. This is a lite function with a small code size, which is useful
 * during the mode switch. For example, to switch from PEE mode to FEI mode:
 *
 * @code
 * CLOCK_ExternalModeToFbeModeQuick();
 * CLOCK_SetFeiMode(...);
 * @endcode
 *
 * @retval kStatus_Success Switched successfully.
 * @retval kStatus_MCG_ModeInvalid If the current mode is not an external mode, do not call this function
 */
status_t CLOCK_ExternalModeToFbeModeQuick(void);

/*!
 * @brief Switches the MCG to FBI mode from internal modes.
 *
 * This function switches the MCG from internal modes (PEI/PBI/BLPI/FEI) to the FBI mode quickly.
 * The MCGIRCLK is used as the system clock souce and PLL is disabled. However,
 * FLL settings are not configured. This is a lite function with a small code size, which is useful
 * during the mode switch. For example, to switch from PEI mode to FEE mode:
 *
 * @code
 * CLOCK_InternalModeToFbiModeQuick();
 * CLOCK_SetFeeMode(...);
 * @endcode
 *
 * @retval kStatus_Success Switched successfully.
 * @retval kStatus_MCG_ModeInvalid If the current mode is not an internal mode, do not call this function.
 */
status_t CLOCK_InternalModeToFbiModeQuick(void);

/*!
 * @brief Sets the MCG to FEI mode during system boot up.
 *
 * This function sets the MCG to FEI mode from the reset mode. It can also be used to
 * set up MCG during system boot up.
 *
 * @param  dmx32  DMX32 in FEI mode.
 * @param  drs The DCO range selection.
 * @param  fllStableDelay Delay function to ensure that the FLL is stable.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 * @note If @p dmx32 is set to kMCG_Dmx32Fine, the slow IRC must not be trimmed
 * to frequency above 32768 Hz.
 */
status_t CLOCK_BootToFeiMode(mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void));

/*!
 * @brief Sets the MCG to FEE mode during system bootup.
 *
 * This function sets MCG to FEE mode from the reset mode. It can also be used to
 * set up the MCG during system boot up.
 *
 * @param   oscsel OSC clock select, OSCSEL.
 * @param   frdiv  FLL reference clock divider setting, FRDIV.
 * @param   dmx32  DMX32 in FEE mode.
 * @param   drs    The DCO range selection.
 * @param   fllStableDelay Delay function to ensure that the FLL is stable.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_BootToFeeMode(
    mcg_oscsel_t oscsel, uint8_t frdiv, mcg_dmx32_t dmx32, mcg_drs_t drs, void (*fllStableDelay)(void));

/*!
 * @brief Sets the MCG to BLPI mode during system boot up.
 *
 * This function sets the MCG to BLPI mode from the reset mode. It can also be used to
 * set up the MCG during sytem boot up.
 *
 * @param  fcrdiv Fast IRC divider, FCRDIV.
 * @param  ircs   The internal reference clock to select, IRCS.
 * @param  ircEnableMode  The MCGIRCLK enable mode, OR'ed value of @ref _mcg_irclk_enable_mode
 *
 * @retval kStatus_MCG_SourceUsed Could not change MCGIRCLK setting.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_BootToBlpiMode(uint8_t fcrdiv, mcg_irc_mode_t ircs, uint8_t ircEnableMode);

/*!
 * @brief Sets the MCG to BLPE mode during sytem boot up.
 *
 * This function sets the MCG to BLPE mode from the reset mode. It can also be used to
 * set up the MCG during sytem boot up.
 *
 * @param  oscsel OSC clock select, MCG_C7[OSCSEL].
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_BootToBlpeMode(mcg_oscsel_t oscsel);
```

```
/*!
 * @brief Sets the MCG to PEE mode during system boot up.
 *
 * This function sets the MCG to PEE mode from reset mode. It can also be used to
 * set up the MCG during system boot up.
 *
 * @param   oscsel OSC clock select, MCG_C7[OSCSEL].
 * @param   pllcs  The PLL selection, PLLCS.
 * @param   config Pointer to the PLL configuration.
 *
 * @retval kStatus_MCG_ModeUnreachable Could not switch to the target mode.
 * @retval kStatus_Success Switched to the target mode successfully.
 */
status_t CLOCK_BootToPeeMode(mcg_oscsel_t oscsel, mcg_pll_clk_select_t pllcs, mcg_pll_config_t cons

/*!
 * @brief Sets the MCG to a target mode.
 *
 * This function sets MCG to a target mode defined by the configuration
 * structure. If switching to the target mode fails, this function
 * chooses the correct path.
 *
 * @param  config Pointer to the target MCG mode configuration structure.
 * @return Return kStatus_Success if switched successfully; Otherwise, it returns an error code #_mcg_sta
 *
 * @note If the external clock is used in the target mode, ensure that it is
 * enabled. For example, if the OSC0 is used, set up OSC0 correctly before calling this
 * function.
 */
status_t CLOCK_SetMcgConfig(mcg_config_t const *config);

/*@}*/

#if defined(__cplusplus)
}
#endif /* __cplusplus */

/*! @} */

#endif /* _FSL_CLOCK_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2013-2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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
```

```
#ifndef _FSL_FLASH_H_
#define _FSL_FLASH_H_

#if (defined(BL_TARGET_FLASH) || defined(BL_TARGET_ROM) || defined(BL_TARGET_RAM))
#include <assert.h>
#include <string.h>
#include "fsl_device_registers.h"
#include "bootloader_common.h"
#else
#include "fsl_common.h"
#endif

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*!
 * @addtogroup flash_driver
 * @{
 */

/*!
 * @name Flash version
 * @{
 */
/*! @brief Constructs the version number for drivers. */
#if !defined(MAKE_VERSION)
#define MAKE_VERSION(major, minor, bugfix) (((major) << 16) | ((minor) << 8) | (bugfix))
#endif

/*! @brief Flash driver version for SDK*/
#define FSL_FLASH_DRIVER_VERSION (MAKE_VERSION(2, 3, 1)) /*!< Version 2.3.1. */

/*! @brief Flash driver version for ROM*/
```

```
enum _flash_driver_version_constants
{
    kFLASH_DriverVersionName = 'F', /*!< Flash driver version name.*/
    kFLASH_DriverVersionMajor = 2,  /*!< Major flash driver version.*/
    kFLASH_DriverVersionMinor = 3,  /*!< Minor flash driver version.*/
    kFLASH_DriverVersionBugfix = 1  /*!< Bugfix for flash driver version.*/
};
/*@}*/

/*!
 * @name Flash configuration
 * @{
 */
/*! @brief Indicates whether to support FlexNVM in the Flash driver */
#if !defined(FLASH_SSD_CONFIG_ENABLE_FLEXNVM_SUPPORT)
#define FLASH_SSD_CONFIG_ENABLE_FLEXNVM_SUPPORT 1 /*!< Enables the FlexNVM support by d
#endif

/*! @brief Indicates whether the FlexNVM is enabled in the Flash driver */
#define FLASH_SSD_IS_FLEXNVM_ENABLED (FLASH_SSD_CONFIG_ENABLE_FLEXNVM_SUPPORT

/*! @brief Indicates whether to support Secondary flash in the Flash driver */
#if !defined(FLASH_SSD_CONFIG_ENABLE_SECONDARY_FLASH_SUPPORT)
#define FLASH_SSD_CONFIG_ENABLE_SECONDARY_FLASH_SUPPORT 1 /*!< Enables the secondar
#endif

/*! @brief Indicates whether the secondary flash is supported in the Flash driver */
#if defined(FSL_FEATURE_FLASH_HAS_MULTIPLE_FLASH) || defined(FSL_FEATURE_FLASH_PFLAS
#define FLASH_SSD_IS_SECONDARY_FLASH_ENABLED (FLASH_SSD_CONFIG_ENABLE_SECOND
#else
#define FLASH_SSD_IS_SECONDARY_FLASH_ENABLED (0)
#endif

/*! @brief Flash driver location. */
#if !defined(FLASH_DRIVER_IS_FLASH_RESIDENT)
#if (!defined(BL_TARGET_ROM) && !defined(BL_TARGET_RAM))
#define FLASH_DRIVER_IS_FLASH_RESIDENT 1 /*!< Used for the flash resident application. */
#else
#define FLASH_DRIVER_IS_FLASH_RESIDENT 0 /*!< Used for the non-flash resident application. */
#endif
#endif

/*! @brief Flash Driver Export option */
#if !defined(FLASH_DRIVER_IS_EXPORTED)
#if (defined(BL_TARGET_ROM) || defined(BL_TARGET_FLASH))
#define FLASH_DRIVER_IS_EXPORTED 1 /*!< Used for the ROM bootloader. */
#else
#define FLASH_DRIVER_IS_EXPORTED 0 /*!< Used for the MCUXpresso SDK application. */
#endif
#endif
/*@}*/

/*!
```

```
 * @name Flash status
 * @{
 */
/*! @brief Flash driver status group. */
#if defined(kStatusGroup_FlashDriver)
#define kStatusGroupGeneric kStatusGroup_Generic
#define kStatusGroupFlashDriver kStatusGroup_FlashDriver
#elif defined(kStatusGroup_FLASH)
#define kStatusGroupGeneric kStatusGroup_Generic
#define kStatusGroupFlashDriver kStatusGroup_FLASH
#else
#define kStatusGroupGeneric 0
#define kStatusGroupFlashDriver 1
#endif

/*! @brief Constructs a status code value from a group and a code number. */
#if !defined(MAKE_STATUS)
#define MAKE_STATUS(group, code) ((((group)*100) + (code)))
#endif

/*!
 * @brief Flash driver status codes.
 */
enum _flash_status
{
    kStatus_FLASH_Success = MAKE_STATUS(kStatusGroupGeneric, 0),         /*!< API is executed succes
    kStatus_FLASH_InvalidArgument = MAKE_STATUS(kStatusGroupGeneric, 4), /*!< Invalid argument*/
    kStatus_FLASH_SizeError = MAKE_STATUS(kStatusGroupFlashDriver, 0),   /*!< Error size*/
    kStatus_FLASH_AlignmentError =
        MAKE_STATUS(kStatusGroupFlashDriver, 1), /*!< Parameter is not aligned with the specified baselin
    kStatus_FLASH_AddressError = MAKE_STATUS(kStatusGroupFlashDriver, 2), /*!< Address is out of ra
    kStatus_FLASH_AccessError =
        MAKE_STATUS(kStatusGroupFlashDriver, 3), /*!< Invalid instruction codes and out-of bound address
    kStatus_FLASH_ProtectionViolation = MAKE_STATUS(
        kStatusGroupFlashDriver, 4), /*!< The program/erase operation is requested to execute on protected 
    kStatus_FLASH_CommandFailure =
        MAKE_STATUS(kStatusGroupFlashDriver, 5), /*!< Run-time error during command execution. */
    kStatus_FLASH_UnknownProperty = MAKE_STATUS(kStatusGroupFlashDriver, 6), /*!< Unknown prop
    kStatus_FLASH_EraseKeyError = MAKE_STATUS(kStatusGroupFlashDriver, 7),   /*!< API erase key is
    kStatus_FLASH_RegionExecuteOnly =
        MAKE_STATUS(kStatusGroupFlashDriver, 8), /*!< The current region is execute-only.*/
    kStatus_FLASH_ExecuteInRamFunctionNotReady =
        MAKE_STATUS(kStatusGroupFlashDriver, 9), /*!< Execute-in-RAM function is not available.*/
    kStatus_FLASH_PartitionStatusUpdateFailure =
        MAKE_STATUS(kStatusGroupFlashDriver, 10), /*!< Failed to update partition status.*/
    kStatus_FLASH_SetFlexramAsEepromError =
        MAKE_STATUS(kStatusGroupFlashDriver, 11), /*!< Failed to set FlexRAM as EEPROM.*/
    kStatus_FLASH_RecoverFlexramAsRamError =
        MAKE_STATUS(kStatusGroupFlashDriver, 12), /*!< Failed to recover FlexRAM as RAM.*/
    kStatus_FLASH_SetFlexramAsRamError = MAKE_STATUS(kStatusGroupFlashDriver, 13), /*!< Failed t
    kStatus_FLASH_RecoverFlexramAsEepromError =
        MAKE_STATUS(kStatusGroupFlashDriver, 14), /*!< Failed to recover FlexRAM as EEPROM.*/
    kStatus_FLASH_CommandNotSupported = MAKE_STATUS(kStatusGroupFlashDriver, 15), /*!< Flash A
    kStatus_FLASH_SwapSystemNotInUninitialized =
        MAKE_STATUS(kStatusGroupFlashDriver, 16), /*!< Swap system is not in an uninitialzed state.*/
    kStatus_FLASH_SwapIndicatorAddressError =
        MAKE_STATUS(kStatusGroupFlashDriver, 17), /*!< The swap indicator address is invalid.*/
    kStatus_FLASH_ReadOnlyProperty = MAKE_STATUS(kStatusGroupFlashDriver, 18), /*!< The flash pro
    kStatus_FLASH_InvalidPropertyValue =
        MAKE_STATUS(kStatusGroupFlashDriver, 19), /*!< The flash property value is out of range.*/
    kStatus_FLASH_InvalidSpeculationOption =
        MAKE_STATUS(kStatusGroupFlashDriver, 20), /*!< The option of flash prefetch speculation is invalid
};
/*@}*/

/*!
 * @name Flash API key
 * @{
 */
/*! @brief Constructs the four character code for the Flash driver API key. */
#if !defined(FOUR_CHAR_CODE)
#define FOUR_CHAR_CODE(a, b, c, d) (((d) << 24) | ((c) << 16) | ((b) << 8) | ((a)))
#endif

/*!
 * @brief Enumeration for Flash driver API keys.
 *
 * @note The resulting value is built with a byte order such that the string
 * being readable in expected order when viewed in a hex editor, if the value
 * is treated as a 32-bit little endian value.
 */
enum _flash_driver_api_keys
{
    kFLASH_ApiEraseKey = FOUR_CHAR_CODE('k', 'f', 'e', 'k') /*!< Key value used to validate all flash eras
};
/*@}*/

/*!
 * @brief Enumeration for supported flash margin levels.
 */
typedef enum _flash_margin_value
{
    kFLASH_MarginValueNormal,  /*!< Use the 'normal' read level for 1s.*/
    kFLASH_MarginValueUser,    /*!< Apply the 'User' margin to the normal read-1 level.*/
    kFLASH_MarginValueFactory, /*!< Apply the 'Factory' margin to the normal read-1 level.*/
    kFLASH_MarginValueInvalid  /*!< Not real margin level, Used to determine the range of valid margin lev
} flash_margin_value_t;

/*!
 * @brief Enumeration for the three possible flash security states.
 */
typedef enum _flash_security_state
{
    kFLASH_SecurityStateNotSecure,       /*!< Flash is not secure.*/
    kFLASH_SecurityStateBackdoorEnabled, /*!< Flash backdoor is enabled.*/
    kFLASH_SecurityStateBackdoorDisabled /*!< Flash backdoor is disabled.*/
```

```
} flash_security_state_t;

/*!
 * @brief Enumeration for the three possible flash protection levels.
 */
typedef enum _flash_protection_state
{
    kFLASH_ProtectionStateUnprotected, /*!< Flash region is not protected.*/
    kFLASH_ProtectionStateProtected,   /*!< Flash region is protected.*/
    kFLASH_ProtectionStateMixed        /*!< Flash is mixed with protected and unprotected region.*/
} flash_protection_state_t;

/*!
 * @brief Enumeration for the three possible flash execute access levels.
 */
typedef enum _flash_execute_only_access_state
{
    kFLASH_AccessStateUnLimited,   /*!< Flash region is unlimited.*/
    kFLASH_AccessStateExecuteOnly, /*!< Flash region is execute only.*/
    kFLASH_AccessStateMixed        /*!< Flash is mixed with unlimited and execute only region.*/
} flash_execute_only_access_state_t;

/*!
 * @brief Enumeration for various flash properties.
 */
typedef enum _flash_property_tag
{
    kFLASH_PropertyPflashSectorSize = 0x00U,         /*!< Pflash sector size property.*/
    kFLASH_PropertyPflashTotalSize = 0x01U,          /*!< Pflash total size property.*/
    kFLASH_PropertyPflashBlockSize = 0x02U,          /*!< Pflash block size property.*/
    kFLASH_PropertyPflashBlockCount = 0x03U,         /*!< Pflash block count property.*/
    kFLASH_PropertyPflashBlockBaseAddr = 0x04U,      /*!< Pflash block base address property.*/
    kFLASH_PropertyPflashFacSupport = 0x05U,         /*!< Pflash fac support property.*/
    kFLASH_PropertyPflashAccessSegmentSize = 0x06U,  /*!< Pflash access segment size property.*/
    kFLASH_PropertyPflashAccessSegmentCount = 0x07U, /*!< Pflash access segment count property.*/
    kFLASH_PropertyFlexRamBlockBaseAddr = 0x08U,     /*!< FlexRam block base address property.*/
    kFLASH_PropertyFlexRamTotalSize = 0x09U,         /*!< FlexRam total size property.*/
    kFLASH_PropertyDflashSectorSize = 0x10U,         /*!< Dflash sector size property.*/
    kFLASH_PropertyDflashTotalSize = 0x11U,          /*!< Dflash total size property.*/
    kFLASH_PropertyDflashBlockSize = 0x12U,          /*!< Dflash block size property.*/
    kFLASH_PropertyDflashBlockCount = 0x13U,         /*!< Dflash block count property.*/
    kFLASH_PropertyDflashBlockBaseAddr = 0x14U,      /*!< Dflash block base address property.*/
    kFLASH_PropertyEepromTotalSize = 0x15U,          /*!< EEPROM total size property.*/
    kFLASH_PropertyFlashMemoryIndex = 0x20U,         /*!< Flash memory index property.*/
    kFLASH_PropertyFlashCacheControllerIndex = 0x21U /*!< Flash cache controller index property.*/
} flash_property_tag_t;

/*!
 * @brief Constants for execute-in-RAM flash function.
 */
enum _flash_execute_in_ram_function_constants
{
    kFLASH_ExecuteInRamFunctionMaxSizeInWords = 16U, /*!< The maximum size of execute-in-RAM fur
    kFLASH_ExecuteInRamFunctionTotalNum = 2U         /*!< Total number of execute-in-RAM functions.*/
};

/*!
 * @brief Flash execute-in-RAM function information.
 */
typedef struct _flash_execute_in_ram_function_config
{
    uint32_t activeFunctionCount;      /*!< Number of available execute-in-RAM functions.*/
    uint32_t *flashRunCommand;         /*!< Execute-in-RAM function: flash_run_command.*/
    uint32_t *flashCommonBitOperation; /*!< Execute-in-RAM function: flash_common_bit_operation.*/
} flash_execute_in_ram_function_config_t;

/*!
 * @brief Enumeration for the two possible options of flash read resource command.
 */
typedef enum _flash_read_resource_option
{
    kFLASH_ResourceOptionFlashIfr =
        0x00U, /*!< Select code for Program flash 0 IFR, Program flash swap 0 IFR, Data flash 0 IFR */
    kFLASH_ResourceOptionVersionId = 0x01U /*!< Select code for the version ID*/
} flash_read_resource_option_t;

/*!
 * @brief Enumeration for the range of special-purpose flash resource
 */
enum _flash_read_resource_range
{
#if (FSL_FEATURE_FLASH_IS_FTFE == 1)
    kFLASH_ResourceRangePflashIfrSizeInBytes = 1024U,  /*!< Pflash IFR size in byte.*/
    kFLASH_ResourceRangeVersionIdSizeInBytes = 8U,     /*!< Version ID IFR size in byte.*/
    kFLASH_ResourceRangeVersionIdStart = 0x08U,        /*!< Version ID IFR start address.*/
    kFLASH_ResourceRangeVersionIdEnd = 0x0FU,          /*!< Version ID IFR end address.*/
    kFLASH_ResourceRangePflashSwapIfrStart = 0x40000U, /*!< Pflash swap IFR start address.*/
    kFLASH_ResourceRangePflashSwapIfrEnd =
        (kFLASH_ResourceRangePflashSwapIfrStart + 0x3FFU), /*!< Pflash swap IFR end address.*/
#else                                                     /* FSL_FEATURE_FLASH_IS_FTFL == 1 or FSL_FEATURE_FLAS
    kFLASH_ResourceRangePflashIfrSizeInBytes = 256U,  /*!< Pflash IFR size in byte.*/
    kFLASH_ResourceRangeVersionIdSizeInBytes = 8U,    /*!< Version ID IFR size in byte.*/
    kFLASH_ResourceRangeVersionIdStart = 0x00U,       /*!< Version ID IFR start address.*/
    kFLASH_ResourceRangeVersionIdEnd = 0x07U,         /*!< Version ID IFR end address.*/
#if 0x20000U == (FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT * FSL_FEATURE_FLASH_PFLASH
    kFLASH_ResourceRangePflashSwapIfrStart = 0x8000U, /*!< Pflash swap IFR start address.*/
#elif 0x40000U == (FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT * FSL_FEATURE_FLASH_PFLA
    kFLASH_ResourceRangePflashSwapIfrStart = 0x10000U, /*!< Pflash swap IFR start address.*/
#elif 0x80000U == (FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT * FSL_FEATURE_FLASH_PFLA
    kFLASH_ResourceRangePflashSwapIfrStart = 0x20000U, /*!< Pflash swap IFR start address.*/
#else
    kFLASH_ResourceRangePflashSwapIfrStart = 0,
#endif
    kFLASH_ResourceRangePflashSwapIfrEnd =
        (kFLASH_ResourceRangePflashSwapIfrStart + 0xFFU), /*!< Pflash swap IFR end address.*/
#endif
```

```
    kFLASH_ResourceRangeDflashIfrStart = 0x800000U, /*!< Dflash IFR start address.*/
    kFLASH_ResourceRangeDflashIfrEnd = 0x8003FFU,   /*!< Dflash IFR end address.*/
};

/*!
 * @brief Enumeration for the index of read/program once record
 */
enum _k3_flash_read_once_index
{
    kFLASH_RecordIndexSwapAddr = 0xA1U,    /*!< Index of Swap indicator address.*/
    kFLASH_RecordIndexSwapEnable = 0xA2U,  /*!< Index of Swap system enable.*/
    kFLASH_RecordIndexSwapDisable = 0xA3U, /*!< Index of Swap system disable.*/
};

/*!
 * @brief Enumeration for the two possilbe options of set FlexRAM function command.
 */
typedef enum _flash_flexram_function_option
{
    kFLASH_FlexramFunctionOptionAvailableAsRam = 0xFFU,    /*!< An option used to make FlexRAM ava
    kFLASH_FlexramFunctionOptionAvailableForEeprom = 0x00U /*!< An option used to make FlexRAM av
} flash_flexram_function_option_t;

/*!
 * @brief Enumeration for acceleration RAM property.
 */
enum _flash_acceleration_ram_property
{
    kFLASH_AccelerationRamSize = 0x400U
};

/*!
 * @brief Enumeration for the possible options of Swap function
 */
typedef enum _flash_swap_function_option
{
    kFLASH_SwapFunctionOptionEnable = 0x00U, /*!< An option used to enable the Swap function */
    kFLASH_SwapFunctionOptionDisable = 0x01U /*!< An option used to disable the Swap function */
} flash_swap_function_option_t;

/*!
 * @brief Enumeration for the possible options of Swap control commands
 */
typedef enum _flash_swap_control_option
{
    kFLASH_SwapControlOptionIntializeSystem = 0x01U,    /*!< An option used to initialize the Swap syster
    kFLASH_SwapControlOptionSetInUpdateState = 0x02U,   /*!< An option used to set the Swap in an upd
    kFLASH_SwapControlOptionSetInCompleteState = 0x04U, /*!< An option used to set the Swap in a com
    kFLASH_SwapControlOptionReportStatus = 0x08U,       /*!< An option used to report the Swap status */
    kFLASH_SwapControlOptionDisableSystem = 0x10U       /*!< An option used to disable the Swap status
} flash_swap_control_option_t;

/*!
```

```
 * @brief Enumeration for the possible flash Swap status.
 */
typedef enum _flash_swap_state
{
    kFLASH_SwapStateUninitialized = 0x00U, /*!< Flash Swap system is in an uninitialized state.*/
    kFLASH_SwapStateReady = 0x01U,         /*!< Flash Swap system is in a ready state.*/
    kFLASH_SwapStateUpdate = 0x02U,        /*!< Flash Swap system is in an update state.*/
    kFLASH_SwapStateUpdateErased = 0x03U,  /*!< Flash Swap system is in an updateErased state.*/
    kFLASH_SwapStateComplete = 0x04U,      /*!< Flash Swap system is in a complete state.*/
    kFLASH_SwapStateDisabled = 0x05U       /*!< Flash Swap system is in a disabled state.*/
} flash_swap_state_t;

/*!
 * @breif Enumeration for the possible flash Swap block status
 */
typedef enum _flash_swap_block_status
{
    kFLASH_SwapBlockStatusLowerHalfProgramBlocksAtZero =
        0x00U, /*!< Swap block status is that lower half program block at zero.*/
    kFLASH_SwapBlockStatusUpperHalfProgramBlocksAtZero =
        0x01U, /*!< Swap block status is that upper half program block at zero.*/
} flash_swap_block_status_t;

/*!
 * @brief Flash Swap information
 */
typedef struct _flash_swap_state_config
{
    flash_swap_state_t flashSwapState;                /*!<The current Swap system status.*/
    flash_swap_block_status_t currentSwapBlockStatus; /*!< The current Swap block status.*/
    flash_swap_block_status_t nextSwapBlockStatus;    /*!< The next Swap block status.*/
} flash_swap_state_config_t;

/*!
 * @brief Flash Swap IFR fields
 */
typedef struct _flash_swap_ifr_field_config
{
    uint16_t swapIndicatorAddress; /*!< A Swap indicator address field.*/
    uint16_t swapEnableWord;       /*!< A Swap enable word field.*/
    uint8_t reserved0[4];          /*!< A reserved field.*/
#if (FSL_FEATURE_FLASH_IS_FTFE == 1)
    uint8_t reserved1[2];     /*!< A reserved field.*/
    uint16_t swapDisableWord; /*!< A Swap disable word field.*/
    uint8_t reserved2[4];     /*!< A reserved field.*/
#endif
} flash_swap_ifr_field_config_t;

/*!
 * @brief Flash Swap IFR field data
 */
typedef union _flash_swap_ifr_field_data
{
    uint32_t flashSwapIfrData[2];                    /*!< A flash Swap IFR field data .*/
    flash_swap_ifr_field_config_t flashSwapIfrField; /*!< A flash Swap IFR field structure.*/
} flash_swap_ifr_field_data_t;

/*!
 * @brief PFlash protection status - low 32bit
 */
typedef union _pflash_protection_status_low
{
    uint32_t protl32b; /*!< PROT[31:0] .*/
    struct
    {
        uint8_t protsl; /*!< PROTS[7:0] .*/
        uint8_t protsh; /*!< PROTS[15:8] .*/
        uint8_t reserved[2];
    } prots16b;
} pflash_protection_status_low_t;

/*!
 * @brief PFlash protection status - full
 */
typedef struct _pflash_protection_status
{
    pflash_protection_status_low_t valueLow32b; /*!< PROT[31:0] or PROTS[15:0].*/
#if ((FSL_FEATURE_FLASH_IS_FTFA == 1) && (defined(FTFA_FPROTH0_PROT_MASK))) || \
    ((FSL_FEATURE_FLASH_IS_FTFE == 1) && (defined(FTFE_FPROTH0_PROT_MASK))) || \
    ((FSL_FEATURE_FLASH_IS_FTFL == 1) && (defined(FTFL_FPROTH0_PROT_MASK)))
    // uint32_t protHigh; /*!< PROT[63:32].*/
    struct
    {
        uint32_t proth32b;
    } valueHigh32b;
#endif
} pflash_protection_status_t;

/*!
 * @brief Enumeration for the FlexRAM load during reset option.
 */
typedef enum _flash_partition_flexram_load_option
{
    kFLASH_PartitionFlexramLoadOptionLoadedWithValidEepromData =
        0x00U, /*!< FlexRAM is loaded with valid EEPROM data during reset sequence.*/
    kFLASH_PartitionFlexramLoadOptionNotLoaded = 0x01U /*!< FlexRAM is not loaded during reset seque
} flash_partition_flexram_load_option_t;

/*!
 * @brief Enumeration for the flash memory index.
 */
typedef enum _flash_memory_index
{
    kFLASH_MemoryIndexPrimaryFlash = 0x00U,   /*!< Current flash memory is primary flash.*/
    kFLASH_MemoryIndexSecondaryFlash = 0x01U, /*!< Current flash memory is secondary flash.*/
} flash_memory_index_t;
```

```
/*!
 * @brief Enumeration for the flash cache controller index.
 */
typedef enum _flash_cache_controller_index
{
    kFLASH_CacheControllerIndexForCore0 = 0x00U, /*!< Current flash cache controller is for core 0.*/
    kFLASH_CacheControllerIndexForCore1 = 0x01U, /*!< Current flash cache controller is for core 1.*/
} flash_cache_controller_index_t;

/*! @brief A callback type used for the Pflash block*/
typedef void (*flash_callback_t)(void);

/*!
 * @brief Enumeration for the two possible options of flash prefetch speculation.
 */
typedef enum _flash_prefetch_speculation_option
{
    kFLASH_prefetchSpeculationOptionEnable = 0x00U,
    kFLASH_prefetchSpeculationOptionDisable = 0x01U
} flash_prefetch_speculation_option_t;

/*!
 * @brief Flash prefetch speculation status.
 */
typedef struct _flash_prefetch_speculation_status
{
    flash_prefetch_speculation_option_t instructionOption; /*!< Instruction speculation.*/
    flash_prefetch_speculation_option_t dataOption;        /*!< Data speculation.*/
} flash_prefetch_speculation_status_t;

/*!
 * @brief Flash cache clear process code.
 */
typedef enum _flash_cache_clear_process
{
    kFLASH_CacheClearProcessPre = 0x00U,  /*!< Pre flash cache clear process.*/
    kFLASH_CacheClearProcessPost = 0x01U, /*!< Post flash cache clear process.*/
} flash_cache_clear_process_t;

/*!
 * @brief Active flash protection information for the current operation.
 */
typedef struct _flash_protection_config
{
    uint32_t regionBase;  /*!< Base address of flash protection region.*/
    uint32_t regionSize;  /*!< size of flash protection region.*/
    uint32_t regionCount; /*!< flash protection region count.*/
} flash_protection_config_t;

/*!
 * @brief Active flash Execute-Only access information for the current operation.
 */
```

```
typedef struct _flash_access_config
{
    uint32_t SegmentBase;  /*!< Base address of flash Execute-Only segment.*/
    uint32_t SegmentSize;  /*!< size of flash Execute-Only segment.*/
    uint32_t SegmentCount; /*!< flash Execute-Only segment count.*/
} flash_access_config_t;

/*!
 * @brief Active flash information for the current operation.
 */
typedef struct _flash_operation_config
{
    uint32_t convertedAddress;           /*!< A converted address for the current flash type.*/
    uint32_t activeSectorSize;           /*!< A sector size of the current flash type.*/
    uint32_t activeBlockSize;            /*!< A block size of the current flash type.*/
    uint32_t blockWriteUnitSize;         /*!< The write unit size.*/
    uint32_t sectorCmdAddressAligment;   /*!< An erase sector command address alignment.*/
    uint32_t sectionCmdAddressAligment;  /*!< A program/verify section command address alignment.*/
    uint32_t resourceCmdAddressAligment; /*!< A read resource command address alignment.*/
    uint32_t checkCmdAddressAligment;    /*!< A program check command address alignment.*/
} flash_operation_config_t;

/*! @brief Flash driver state information.
 *
 * An instance of this structure is allocated by the user of the flash driver and
 * passed into each of the driver APIs.
 */
typedef struct _flash_config
{
    uint32_t PFlashBlockBase;                /*!< A base address of the first PFlash block */
    uint32_t PFlashTotalSize;                /*!< The size of the combined PFlash block. */
    uint8_t PFlashBlockCount;                /*!< A number of PFlash blocks. */
    uint8_t FlashMemoryIndex;                /*!< 0 - primary flash; 1 - secondary flash*/
    uint8_t FlashCacheControllerIndex;       /*!< 0 - Controller for core 0; 1 - Controller for core 1 */
    uint8_t Reserved0;                       /*!< Reserved field 0 */
    uint32_t PFlashSectorSize;               /*!< The size in bytes of a sector of PFlash. */
    flash_callback_t PFlashCallback;         /*!< The callback function for the flash API. */
    uint32_t PFlashAccessSegmentSize;        /*!< A size in bytes of an access segment of PFlash. */
    uint32_t PFlashAccessSegmentCount;       /*!< A number of PFlash access segments. */
    uint32_t *flashExecuteInRamFunctionInfo; /*!< An information structure of the flash execute-in-RAM func
    uint32_t FlexRAMBlockBase;               /*!< For the FlexNVM device, this is the base address of the FlexR
    /*!< For the non-FlexNVM device, this is the base address of the acceleration RAM memory */
    uint32_t FlexRAMTotalSize; /*!< For the FlexNVM device, this is the size of the FlexRAM */
                          /*!< For the non-FlexNVM device, this is the size of the acceleration RAM memory */
    uint32_t
        DFlashBlockBase; /*!< For the FlexNVM device, this is the base address of the D-Flash memory (Flex
                         /*!< For the non-FlexNVM device, this field is unused */
    uint32_t DFlashTotalSize; /*!< For the FlexNVM device, this is the total size of the FlexNVM memory; */
                              /*!< For the non-FlexNVM device, this field is unused */
    uint32_t EEpromTotalSize; /*!< For the FlexNVM device, this is the size in bytes of the EEPROM area w
                              partitioned from FlexRAM */
    /*!< For the non-FlexNVM device, this field is unused */
} flash_config_t;
```

```
/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif

/*!
 * @name Initialization
 * @{
 */

/*!
 * @brief Initializes the global flash properties structure members.
 *
 * This function checks and initializes the Flash module for the other Flash APIs.
 *
 * @param config Pointer to the storage for the driver runtime state.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_PartitionStatusUpdateFailure Failed to update the partition status.
 */
status_t FLASH_Init(flash_config_t *config);

/*!
 * @brief Sets the desired flash callback function.
 *
 * @param config Pointer to the storage for the driver runtime state.
 * @param callback A callback function to be stored in the driver.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 */
status_t FLASH_SetCallback(flash_config_t *config, flash_callback_t callback);

/*!
 * @brief Prepares flash execute-in-RAM functions.
 *
 * @param config Pointer to the storage for the driver runtime state.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 */
#if FLASH_DRIVER_IS_FLASH_RESIDENT
status_t FLASH_PrepareExecuteInRamFunctions(flash_config_t *config);
#endif

/*@}*/
```

```
/*!
 * @name Erasing
 * @{
 */

/*!
 * @brief Erases entire flash
 *
 * @param config Pointer to the storage for the driver runtime state.
 * @param key A value used to validate all flash erase APIs.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_EraseKeyError API erase key is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 * @retval #kStatus_FLASH_PartitionStatusUpdateFailure Failed to update the partition status.
 */
status_t FLASH_EraseAll(flash_config_t *config, uint32_t key);

/*!
 * @brief Erases the flash sectors encompassed by parameters passed into function.
 *
 * This function erases the appropriate number of flash sectors based on the
 * desired start address and length.
 *
 * @param config The pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be erased.
 *              The start address does not need to be sector-aligned but must be word-aligned.
 * @param lengthInBytes The length, given in bytes (not words or long-words)
 *                      to be erased. Must be word-aligned.
 * @param key The value used to validate all flash erase APIs.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError The parameter is not aligned with the specified baseline.
 * @retval #kStatus_FLASH_AddressError The address is out of range.
 * @retval #kStatus_FLASH_EraseKeyError The API erase key is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_Erase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key);

/*!
 * @brief Erases the entire flash, including protected sectors.
 *
 * @param config Pointer to the storage for the driver runtime state.
 * @param key A value used to validate all flash erase APIs.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_EraseKeyError API erase key is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 * @retval #kStatus_FLASH_PartitionStatusUpdateFailure Failed to update the partition status.
 */
#if defined(FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD) && FSL_FEATURE
status_t FLASH_EraseAllUnsecure(flash_config_t *config, uint32_t key);
#endif

/*!
 * @brief Erases all program flash execute-only segments defined by the FXACC registers.
 *
 * @param config Pointer to the storage for the driver runtime state.
 * @param key A value used to validate all flash erase APIs.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_EraseKeyError API erase key is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_EraseAllExecuteOnlySegments(flash_config_t *config, uint32_t key);

/*@}*/

/*!
 * @name Programming
 * @{
 */

/*!
 * @brief Programs flash with data at locations passed in through parameters.
 *
 * This function programs the flash memory with the desired data for a given
 * flash area as determined by the start address and the length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be programmed. Must be
 *              word-aligned.
 * @param src A pointer to the source buffer of data that is to be programmed
 *            into the flash.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *                      to be programmed. Must be word-aligned.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with the specified baseline.
 * @retval #kStatus_FLASH_AddressError Address is out of range.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_Program(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t lengthInBytes);

/*!
 * @brief Programs Program Once Field through parameters.
 *
 * This function programs the Program Once Field with the desired data for a given
 * flash area as determined by the index and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param index The index indicating which area of the Program Once Field to be programmed.
 * @param src A pointer to the source buffer of data that is to be programmed
 *            into the Program Once Field.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *                      to be programmed. Must be word-aligned.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_ProgramOnce(flash_config_t *config, uint32_t index, uint32_t *src, uint32_t lengthInBytes

/*!
 * @brief Programs flash with data at locations passed in through parameters via the Program Section com
 *
 * This function programs the flash memory with the desired data for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be programmed. Must be
 *              word-aligned.
 * @param src A pointer to the source buffer of data that is to be programmed
 *            into the flash.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *                      to be programmed. Must be word-aligned.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_AddressError Address is out of range.
 * @retval #kStatus_FLASH_SetFlexramAsRamError Failed to set flexram as RAM.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 * @retval #kStatus_FLASH_RecoverFlexramAsEepromError Failed to recover FlexRAM as EEPROM.
 */
#if defined(FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD) && FSL_FEATURE_FLASH_HA
status_t FLASH_ProgramSection(flash_config_t *config, uint32_t start, uint32_t *src, uint32_t lengthInByte
#endif

/*!
 * @brief Programs the EEPROM with data at locations passed in through parameters.
 *
 * This function programs the emulated EEPROM with the desired data for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be programmed. Must be
 *              word-aligned.
 * @param src A pointer to the source buffer of data that is to be programmed
 *            into the flash.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *                      to be programmed. Must be word-aligned.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AddressError Address is out of range.
 * @retval #kStatus_FLASH_SetFlexramAsEepromError Failed to set flexram as eeprom.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_RecoverFlexramAsRamError Failed to recover the FlexRAM as RAM.
 */
#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromWrite(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t lengthInBytes);
#endif

/*@}*/

/*!
 * @name Reading
 * @{
 */

/*!
 * @brief Reads the resource with data at locations passed in through parameters.
 *
 * This function reads the flash memory with the desired location for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be programmed. Must be
 *              word-aligned.
 * @param dst A pointer to the destination buffer of data that is used to store
 *        data to be read.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *        to be read. Must be word-aligned.
 * @param option The resource option which indicates which area should be read back.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with the specified baseline.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
#if defined(FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD) && FSL_FEATURE_FLASH_HAS_
status_t FLASH_ReadResource(
    flash_config_t *config, uint32_t start, uint32_t *dst, uint32_t lengthInBytes, flash_read_resource_option_
#endif

/*!
 * @brief Reads the Program Once Field through parameters.
 *
 * This function reads the read once feild with given index and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param index The index indicating the area of program once field to be read.
 * @param dst A pointer to the destination buffer of data that is used to store
 *        data to be read.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *        to be programmed. Must be word-aligned.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_ReadOnce(flash_config_t *config, uint32_t index, uint32_t *dst, uint32_t lengthInBytes);

/*@}*/

/*!
 * @name Security
 * @{
 */

/*!
 * @brief Returns the security state via the pointer passed into the function.
 *
 * This function retrieves the current flash security status, including the
 * security enabling state and the backdoor key enabling state.
 *
 * @param config A pointer to storage for the driver runtime state.
 * @param state A pointer to the value returned for the current security status code:
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 */
```

```
status_t FLASH_GetSecurityState(flash_config_t *config, flash_security_state_t *state);

/*!
 * @brief Allows users to bypass security with a backdoor key.
 *
 * If the MCU is in secured state, this function unsecures the MCU by
 * comparing the provided backdoor key with ones in the flash configuration
 * field.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param backdoorKey A pointer to the user buffer containing the backdoor key.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_SecurityBypass(flash_config_t *config, const uint8_t *backdoorKey);

/*@}*/

/*!
 * @name Verification
 * @{
 */

/*!
 * @brief Verifies erasure of the entire flash at a specified margin level.
 *
 * This function checks whether the flash is erased to the
 * specified read margin level.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param margin Read margin choice.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_VerifyEraseAll(flash_config_t *config, flash_margin_value_t margin);

/*!
 * @brief Verifies an erasure of the desired flash area at a specified margin level.
 *
 * This function checks the appropriate number of flash sectors based on
 * the desired start address and length to check whether the flash is erased
 * to the specified read margin level.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be verified.
 *       The start address does not need to be sector-aligned but must be word-aligned.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *       to be verified. Must be word-aligned.
 * @param margin Read margin choice.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_AddressError Address is out of range.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_VerifyErase(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, flash_margin_va

/*!
 * @brief Verifies programming of the desired flash area at a specified margin level.
 *
 * This function verifies the data programed in the flash memory using the
 * Flash Program Check Command and compares it to the expected data for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be verified. Must be word-aligned.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *       to be verified. Must be word-aligned.
 * @param expectedData A pointer to the expected data that is to be
 *       verified against.
 * @param margin Read margin choice.
 * @param failedAddress A pointer to the returned failing address.
 * @param failedData A pointer to the returned failing data.  Some derivatives do
 *       not include failed data as part of the FCCOBx registers.  In this
 *       case, zeros are returned upon failure.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_AddressError Address is out of range.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_VerifyProgram(flash_config_t *config,
                             uint32_t start,
                             uint32_t lengthInBytes,
                             const uint32_t *expectedData,
                             flash_margin_value_t margin,
                             uint32_t *failedAddress,
                             uint32_t *failedData);
```

```
/*!
 * @brief Verifies whether the program flash execute-only segments have been erased to
 *  the specified read margin level.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param margin Read margin choice.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
status_t FLASH_VerifyEraseAllExecuteOnlySegments(flash_config_t *config, flash_margin_value_t margi

/*@}*/

/*!
 * @name Protection
 * @{
 */

/*!
 * @brief Returns the protection state of the desired flash area via the pointer passed into the function.
 *
 * This function retrieves the current flash protect status for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be checked. Must be word-aligned.
 * @param lengthInBytes The length, given in bytes (not words or long-words)
 *        to be checked.  Must be word-aligned.
 * @param protection_state A pointer to the value returned for the current
 *        protection status code for the desired flash area.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_AddressError The address is out of range.
 */
status_t FLASH_IsProtected(flash_config_t *config,
                           uint32_t start,
                           uint32_t lengthInBytes,
                           flash_protection_state_t *protection_state);

/*!
 * @brief Returns the access state of the desired flash area via the pointer passed into the function.
 *
 * This function retrieves the current flash access status for a given
 * flash area as determined by the start address and length.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param start The start address of the desired flash memory to be checked. Must be word-aligned.
 * @param lengthInBytes The length, given in bytes (not words or long-words),
 *        to be checked.  Must be word-aligned.
 * @param access_state A pointer to the value returned for the current
 *        access status code for the desired flash area.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError The parameter is not aligned to the specified baseline.
 * @retval #kStatus_FLASH_AddressError The address is out of range.
 */
status_t FLASH_IsExecuteOnly(flash_config_t *config,
                             uint32_t start,
                             uint32_t lengthInBytes,
                             flash_execute_only_access_state_t *access_state);

/*@}*/

/*!
 * @name Properties
 * @{
 */

/*!
 * @brief Returns the desired flash property.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param whichProperty The desired property from the list of properties in
 *        enum flash_property_tag_t
 * @param value A pointer to the value returned for the desired flash property.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_UnknownProperty An unknown property tag.
 */
status_t FLASH_GetProperty(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t *value);

/*!
 * @brief Sets the desired flash property.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param whichProperty The desired property from the list of properties in
 *        enum flash_property_tag_t
 * @param value A to set for the desired flash property.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_UnknownProperty An unknown property tag.
 * @retval #kStatus_FLASH_InvalidPropertyValue An invalid property value.
 * @retval #kStatus_FLASH_ReadOnlyProperty An read-only property tag.
 */
status_t FLASH_SetProperty(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t value);
```

```
/*@}*/

/*!
 * @name FlexRAM
 * @{
 */

/*!
 * @brief Sets the FlexRAM function command.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param option The option used to set the work mode of FlexRAM.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
#if defined(FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD) && FSL_FEATURE_FLAS
status_t FLASH_SetFlexramFunction(flash_config_t *config, flash_flexram_function_option_t option);
#endif

/*@}*/

/*!
 * @name Swap
 * @{
 */

/*!
 * @brief Configures the Swap function or checks the the swap state of the Flash module.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param address Address used to configure the flash Swap function.
 * @param option The possible option used to configure Flash Swap function or check the flash Swap statu
 * @param returnInfo A pointer to the data which is used to return the information of flash Swap.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_SwapIndicatorAddressError Swap indicator address is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during the command execution.
 */
#if defined(FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD) && FSL_FEATURE_FLASH_HAS_S
status_t FLASH_SwapControl(flash_config_t *config,
                           uint32_t address,
                           flash_swap_control_option_t option,
                        flash_swap_state_config_t *returnInfo);
#endif

/*!
 * @brief Swaps the lower half flash with the higher half flash.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param address Address used to configure the flash swap function
 * @param option The possible option used to configure the Flash Swap function or check the flash Swap s
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_AlignmentError Parameter is not aligned with specified baseline.
 * @retval #kStatus_FLASH_SwapIndicatorAddressError Swap indicator address is invalid.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 * @retval #kStatus_FLASH_SwapSystemNotInUninitialized Swap system is not in an uninitialzed state.
 */
#if defined(FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP) && FSL_FEATURE_FLASH_HAS_P
status_t FLASH_Swap(flash_config_t *config, uint32_t address, flash_swap_function_option_t option);
#endif

/*!
 * @name FlexNVM
 * @{
 */

/*!
 * @brief Prepares the FlexNVM block for use as data flash, EEPROM backup, or a combination of both ar
 * FlexRAM.
 *
 * @param config Pointer to storage for the driver runtime state.
 * @param option The option used to set FlexRAM load behavior during reset.
 * @param eepromDataSizeCode Determines the amount of FlexRAM used in each of the available EEPR
 * @param flexnvmPartitionCode Specifies how to split the FlexNVM block between data flash memory and
 *        memory supporting EEPROM functions.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument Invalid argument is provided.
 * @retval #kStatus_FLASH_ExecuteInRamFunctionNotReady Execute-in-RAM function is not available.
 * @retval #kStatus_FLASH_AccessError Invalid instruction codes and out-of bounds addresses.
 * @retval #kStatus_FLASH_ProtectionViolation The program/erase operation is requested to execute on p
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 */
#if defined(FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD) && FSL_FEATURE_FLASH_H
status_t FLASH_ProgramPartition(flash_config_t *config,
                            flash_partition_flexram_load_option_t option,
                            uint32_t eepromDataSizeCode,
                            uint32_t flexnvmPartitionCode);
#endif
```

```
/*@}*/

/*!
* @name Flash Protection Utilities
* @{
*/

/*!
 * @brief Sets the PFlash Protection to the intended protection status.
 *
 * @param config A pointer to storage for the driver runtime state.
 * @param protectStatus The expected protect status to set to the PFlash protection register. Each bit is
 * corresponding to protection of 1/32(64) of the total PFlash. The least significant bit is corresponding to th
 * address area of PFlash. The most significant bit is corresponding to the highest address area of PFlash.
 * two possible cases as shown below:
 *       0: this area is protected.
 *       1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 */
status_t FLASH_PflashSetProtection(flash_config_t *config, pflash_protection_status_t *protectStatus);

/*!
 * @brief Gets the PFlash protection status.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param protectStatus  Protect status returned by the PFlash IP. Each bit is corresponding to the protect
 * 1/32(64)
 * of the
 * total PFlash. The least significant bit corresponds to the lowest address area of the PFlash. The most sig
 * bit corresponds to the highest address area of PFlash. There are two possible cases as shown below:
 *       0: this area is protected.
 *       1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 */
status_t FLASH_PflashGetProtection(flash_config_t *config, pflash_protection_status_t *protectStatus);

/*!
 * @brief Sets the DFlash protection to the intended protection status.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param protectStatus The expected protect status to set to the DFlash protection register. Each bit
 * corresponds to the protection of the 1/8 of the total DFlash. The least significant bit corresponds to the lo
 * address area of the DFlash. The most significant bit corresponds to the highest address area of  the DFl
 * are
 * two possible cases as shown below:
 *       0: this area is protected.
 *       1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_CommandNotSupported Flash API is not supported.
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 */
#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_DflashSetProtection(flash_config_t *config, uint8_t protectStatus);
#endif

/*!
 * @brief Gets the DFlash protection status.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param protectStatus  DFlash Protect status returned by the PFlash IP. Each bit corresponds to the pro
 * 1/8 of
 * the total DFlash. The least significant bit corresponds to the lowest address area of the DFlash. The mos
 * significant bit corresponds to the highest address area of the DFlash, and so on. There are two possible
 * below:
 *       0: this area is protected.
 *       1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_CommandNotSupported Flash API is not supported.
 */
#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_DflashGetProtection(flash_config_t *config, uint8_t *protectStatus);
#endif

/*!
 * @brief Sets the EEPROM protection to the intended protection status.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param protectStatus The expected protect status to set to the EEPROM protection register. Each bit
 * corresponds to the protection of the 1/8 of the total EEPROM. The least significant bit corresponds to the
 * address area of the EEPROM. The most significant bit corresponds to the highest address area of EEPR
 * There are two possible cases as shown below:
 *       0: this area is protected.
 *       1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_CommandNotSupported Flash API is not supported.
 * @retval #kStatus_FLASH_CommandFailure Run-time error during command execution.
 */
#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromSetProtection(flash_config_t *config, uint8_t protectStatus);
#endif

/*!
 * @brief Gets the DFlash protection status.
 *
 * @param config A pointer to the storage for the driver runtime state.
 * @param protectStatus  DFlash Protect status returned by the PFlash IP. Each bit corresponds to the pro
 * 1/8 of
 * the total EEPROM. The least significant bit corresponds to the lowest address area of the EEPROM. The
 * significant bit corresponds to the highest address area of the EEPROM. There are two possible cases as
 *      0: this area is protected.
 *      1: this area is unprotected.
 *
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidArgument An invalid argument is provided.
 * @retval #kStatus_FLASH_CommandNotSupported Flash API is not supported.
 */
#if FLASH_SSD_IS_FLEXNVM_ENABLED
status_t FLASH_EepromGetProtection(flash_config_t *config, uint8_t *protectStatus);
#endif

/*@}*/

/*@}*/

/*!
* @name Flash Speculation Utilities
* @{
*/

/*!
 * @brief Sets the PFlash prefetch speculation to the intended speculation status.
 *
 * @param speculationStatus The expected protect status to set to the PFlash protection register. Each bit
 * @retval #kStatus_FLASH_Success API was executed successfully.
 * @retval #kStatus_FLASH_InvalidSpeculationOption An invalid speculation option argument is provided.
 */
status_t FLASH_PflashSetPrefetchSpeculation(flash_prefetch_speculation_status_t *speculationStatus);

/*!
 * @brief Gets the PFlash prefetch speculation status.
 *
 * @param speculationStatus  Speculation status returned by the PFlash IP.
 * @retval #kStatus_FLASH_Success API was executed successfully.
 */
status_t FLASH_PflashGetPrefetchSpeculation(flash_prefetch_speculation_status_t *speculationStatus);

/*@}*/

#if defined(__cplusplus)
}
#endif

/*! @}*/

#endif /* _FSL_FLASH_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015-2016, Freescale Semiconductor, Inc.
```

```
#ifndef _FSL_COMMON_H_
#define _FSL_COMMON_H_

#include <assert.h>
#include <stdbool.h>
#include <stdint.h>
#include <string.h>

#if defined(__ICCARM__)
#include <stddef.h>
#endif

#include "fsl_device_registers.h"

/*!
 * @addtogroup ksdk_common
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @brief Construct a status code value from a group and code number. */
#define MAKE_STATUS(group, code) ((((group)*100) + (code)))
```

```
/*! @brief Construct the version number for drivers. */
#define MAKE_VERSION(major, minor, bugfix) (((major) << 16) | ((minor) << 8) | (bugfix))

/* Debug console type definition. */
#define DEBUG_CONSOLE_DEVICE_TYPE_NONE 0U     /*!< No debug console.             */
#define DEBUG_CONSOLE_DEVICE_TYPE_UART 1U     /*!< Debug console base on UART.   */
#define DEBUG_CONSOLE_DEVICE_TYPE_LPUART 2U   /*!< Debug console base on LPUART. */
#define DEBUG_CONSOLE_DEVICE_TYPE_LPSCI 3U    /*!< Debug console base on LPSCI.  */
#define DEBUG_CONSOLE_DEVICE_TYPE_USBCDC 4U   /*!< Debug console base on USBCDC. */
#define DEBUG_CONSOLE_DEVICE_TYPE_FLEXCOMM 5U /*!< Debug console base on USBCDC. */
#define DEBUG_CONSOLE_DEVICE_TYPE_IUART 6U    /*!< Debug console base on i.MX UART. */

/*! @brief Status group numbers. */
enum _status_groups
{
    kStatusGroup_Generic = 0,                 /*!< Group number for generic status codes. */
    kStatusGroup_FLASH = 1,                   /*!< Group number for FLASH status codes. */
    kStatusGroup_LPSPI = 4,                   /*!< Group number for LPSPI status codes. */
    kStatusGroup_FLEXIO_SPI = 5,              /*!< Group number for FLEXIO SPI status codes. */
    kStatusGroup_DSPI = 6,                    /*!< Group number for DSPI status codes. */
    kStatusGroup_FLEXIO_UART = 7,             /*!< Group number for FLEXIO UART status codes. */
    kStatusGroup_FLEXIO_I2C = 8,              /*!< Group number for FLEXIO I2C status codes. */
    kStatusGroup_LPI2C = 9,                   /*!< Group number for LPI2C status codes. */
    kStatusGroup_UART = 10,                   /*!< Group number for UART status codes. */
    kStatusGroup_I2C = 11,                    /*!< Group number for UART status codes. */
    kStatusGroup_LPSCI = 12,                  /*!< Group number for LPSCI status codes. */
    kStatusGroup_LPUART = 13,                 /*!< Group number for LPUART status codes. */
    kStatusGroup_SPI = 14,                    /*!< Group number for SPI status code.*/
    kStatusGroup_XRDC = 15,                   /*!< Group number for XRDC status code.*/
    kStatusGroup_SEMA42 = 16,                 /*!< Group number for SEMA42 status code.*/
    kStatusGroup_SDHC = 17,                   /*!< Group number for SDHC status code */
    kStatusGroup_SDMMC = 18,                  /*!< Group number for SDMMC status code */
    kStatusGroup_SAI = 19,                    /*!< Group number for SAI status code */
    kStatusGroup_MCG = 20,                    /*!< Group number for MCG status codes. */
    kStatusGroup_SCG = 21,                    /*!< Group number for SCG status codes. */
    kStatusGroup_SDSPI = 22,                  /*!< Group number for SDSPI status codes. */
    kStatusGroup_FLEXIO_I2S = 23,             /*!< Group number for FLEXIO I2S status codes */
    kStatusGroup_FLEXIO_MCULCD = 24,          /*!< Group number for FLEXIO LCD status codes */
    kStatusGroup_FLASHIAP = 25,               /*!< Group number for FLASHIAP status codes */
    kStatusGroup_FLEXCOMM_I2C = 26,           /*!< Group number for FLEXCOMM I2C status codes */
    kStatusGroup_I2S = 27,                    /*!< Group number for I2S status codes */
    kStatusGroup_IUART = 28,                  /*!< Group number for IUART status codes */
    kStatusGroup_SDRAMC = 35,                 /*!< Group number for SDRAMC status codes. */
    kStatusGroup_POWER = 39,                  /*!< Group number for POWER status codes. */
    kStatusGroup_ENET = 40,                   /*!< Group number for ENET status codes. */
    kStatusGroup_PHY = 41,                    /*!< Group number for PHY status codes. */
    kStatusGroup_TRGMUX = 42,                 /*!< Group number for TRGMUX status codes. */
    kStatusGroup_SMARTCARD = 43,              /*!< Group number for SMARTCARD status codes. */
    kStatusGroup_LMEM = 44,                   /*!< Group number for LMEM status codes. */
    kStatusGroup_QSPI = 45,                   /*!< Group number for QSPI status codes. */
    kStatusGroup_DMA = 50,                    /*!< Group number for DMA status codes. */
    kStatusGroup_EDMA = 51,                   /*!< Group number for EDMA status codes. */
    kStatusGroup_DMAMGR = 52,                 /*!< Group number for DMAMGR status codes. */
    kStatusGroup_FLEXCAN = 53,                /*!< Group number for FlexCAN status codes. */
    kStatusGroup_LTC = 54,                    /*!< Group number for LTC status codes. */
    kStatusGroup_FLEXIO_CAMERA = 55,          /*!< Group number for FLEXIO CAMERA status codes. */
    kStatusGroup_LPC_SPI = 56,                /*!< Group number for LPC_SPI status codes. */
    kStatusGroup_LPC_USART = 57,              /*!< Group number for LPC_USART status codes. */
    kStatusGroup_DMIC = 58,                   /*!< Group number for DMIC status codes. */
    kStatusGroup_SDIF = 59,                   /*!< Group number for SDIF status codes.*/
    kStatusGroup_SPIFI = 60,                  /*!< Group number for SPIFI status codes. */
    kStatusGroup_OTP = 61,                    /*!< Group number for OTP status codes. */
    kStatusGroup_MCAN = 62,                   /*!< Group number for MCAN status codes. */
    kStatusGroup_CAAM = 63,                   /*!< Group number for CAAM status codes. */
    kStatusGroup_ECSPI = 64,                  /*!< Group number for ECSPI status codes. */
    kStatusGroup_USDHC = 65,                  /*!< Group number for USDHC status codes.*/
    kStatusGroup_ESAI = 69,                   /*!< Group number for ESAI status codes. */
    kStatusGroup_FLEXSPI = 70,                /*!< Group number for FLEXSPI status codes. */
    kStatusGroup_NOTIFIER = 98,               /*!< Group number for NOTIFIER status codes. */
    kStatusGroup_DebugConsole = 99,           /*!< Group number for debug console status codes. */
    kStatusGroup_ApplicationRangeStart = 100, /*!< Starting number for application groups. */
};

/*! @brief Generic status return codes. */
enum _generic_status
{
    kStatus_Success = MAKE_STATUS(kStatusGroup_Generic, 0),
    kStatus_Fail = MAKE_STATUS(kStatusGroup_Generic, 1),
    kStatus_ReadOnly = MAKE_STATUS(kStatusGroup_Generic, 2),
    kStatus_OutOfRange = MAKE_STATUS(kStatusGroup_Generic, 3),
    kStatus_InvalidArgument = MAKE_STATUS(kStatusGroup_Generic, 4),
    kStatus_Timeout = MAKE_STATUS(kStatusGroup_Generic, 5),
    kStatus_NoTransferInProgress = MAKE_STATUS(kStatusGroup_Generic, 6),
};

/*! @brief Type used for all status and error return values. */
typedef int32_t status_t;

/*
 * The fsl_clock.h is included here because it needs MAKE_VERSION/MAKE_STATUS/status_t
 * defined in previous of this file.
 */
#include "fsl_clock.h"

/*
 * Chip level peripheral reset API, for MCUs that implement peripheral reset control external to a peripheral
 */
#if ((defined(FSL_FEATURE_SOC_SYSCON_COUNT) && (FSL_FEATURE_SOC_SYSCON_COUNT > 0
     (defined(FSL_FEATURE_SOC_ASYNC_SYSCON_COUNT) && (FSL_FEATURE_SOC_ASYNC_SYS
#include "fsl_reset.h"
#endif

/*! @name Min/max macros */
/* @{ */
#if !defined(MIN)
```

```
#define MIN(a, b) ((a) < (b) ? (a) : (b))
#endif

#if !defined(MAX)
#define MAX(a, b) ((a) > (b) ? (a) : (b))
#endif
/* @} */

/*! @brief Computes the number of elements in an array. */
#define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))

/*! @name UINT16_MAX/UINT32_MAX value */
/* @{ */
#if !defined(UINT16_MAX)
#define UINT16_MAX ((uint16_t)-1)
#endif

#if !defined(UINT32_MAX)
#define UINT32_MAX ((uint32_t)-1)
#endif
/* @} */

/*! @name Timer utilities */
/* @{ */
/*! Macro to convert a microsecond period to raw count value */
#define USEC_TO_COUNT(us, clockFreqInHz) (uint64_t)((uint64_t)us * clockFreqInHz / 1000000U)
/*! Macro to convert a raw count value to microsecond */
#define COUNT_TO_USEC(count, clockFreqInHz) (uint64_t)((uint64_t)count * 1000000U / clockFreqInHz)

/*! Macro to convert a millisecond period to raw count value */
#define MSEC_TO_COUNT(ms, clockFreqInHz) (uint64_t)((uint64_t)ms * clockFreqInHz / 1000U)
/*! Macro to convert a raw count value to millisecond */
#define COUNT_TO_MSEC(count, clockFreqInHz) (uint64_t)((uint64_t)count * 1000U / clockFreqInHz)
/* @} */

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif

/*!
 * @brief Enable specific interrupt.
 *
 * Enable the interrupt not routed from intmux.
 *
 * @param interrupt The IRQ number.
 */
static inline void EnableIRQ(IRQn_Type interrupt)
{
    if (NotAvail_IRQn == interrupt)
    {
        return;
    }

#if defined(FSL_FEATURE_SOC_INTMUX_COUNT) && (FSL_FEATURE_SOC_INTMUX_COUNT > 0)
    if (interrupt < FSL_FEATURE_INTMUX_IRQ_START_INDEX)
#endif
    {
#if defined(__GIC_PRIO_BITS)
        GIC_EnableIRQ(interrupt);
#else
        NVIC_EnableIRQ(interrupt);
#endif
    }
}

/*!
 * @brief Disable specific interrupt.
 *
 * Disable the interrupt not routed from intmux.
 *
 * @param interrupt The IRQ number.
 */
static inline void DisableIRQ(IRQn_Type interrupt)
{
    if (NotAvail_IRQn == interrupt)
    {
        return;
    }

#if defined(FSL_FEATURE_SOC_INTMUX_COUNT) && (FSL_FEATURE_SOC_INTMUX_COUNT > 0)
    if (interrupt < FSL_FEATURE_INTMUX_IRQ_START_INDEX)
#endif
    {
#if defined(__GIC_PRIO_BITS)
        GIC_DisableIRQ(interrupt);
#else
        NVIC_DisableIRQ(interrupt);
#endif
    }
}

/*!
 * @brief Disable the global IRQ
 *
 * Disable the global interrupt and return the current primask register. User is required to provided the prima
 * register for the EnableGlobalIRQ().
 *
 * @return Current primask value.
 */
static inline uint32_t DisableGlobalIRQ(void)
{
#if defined(CPSR_I_Msk)
```

```
    uint32_t cpsr = __get_CPSR() & CPSR_I_Msk;

    __disable_irq();

    return cpsr;
#else
    uint32_t regPrimask = __get_PRIMASK();

    __disable_irq();

    return regPrimask;
#endif
}

/*!
 * @brief Enaable the global IRQ
 *
 * Set the primask register with the provided primask value but not just enable the primask. The idea is for t
 * convinience of integration of RTOS. some RTOS get its own management mechanism of primask. User i
 * use the EnableGlobalIRQ() and DisableGlobalIRQ() in pair.
 *
 * @param primask value of primask register to be restored. The primask value is supposed to be provided
 * DisableGlobalIRQ().
 */
static inline void EnableGlobalIRQ(uint32_t primask)
{
#if defined(CPSR_I_Msk)
    __set_CPSR((__get_CPSR() & ~CPSR_I_Msk) | primask);
#else
    __set_PRIMASK(primask);
#endif
}

/*!
 * @brief install IRQ handler
 *
 * @param irq IRQ number
 * @param irqHandler IRQ handler address
 * @return The old IRQ handler address
 */
uint32_t InstallIRQHandler(IRQn_Type irq, uint32_t irqHandler);

#if (defined(FSL_FEATURE_SOC_SYSCON_COUNT) && (FSL_FEATURE_SOC_SYSCON_COUNT > 0)
/*!
 * @brief Enable specific interrupt for wake-up from deep-sleep mode.
 *
 * Enable the interrupt for wake-up from deep sleep mode.
 * Some interrupts are typically used in sleep mode only and will not occur during
 * deep-sleep mode because relevant clocks are stopped. However, it is possible to enable
 * those clocks (significantly increasing power consumption in the reduced power mode),
 * making these wake-ups possible.
 *
 * @note This function also enables the interrupt in the NVIC (EnableIRQ() is called internally).
 *
 * @param interrupt The IRQ number.
 */
void EnableDeepSleepIRQ(IRQn_Type interrupt);

/*!
 * @brief Disable specific interrupt for wake-up from deep-sleep mode.
 *
 * Disable the interrupt for wake-up from deep sleep mode.
 * Some interrupts are typically used in sleep mode only and will not occur during
 * deep-sleep mode because relevant clocks are stopped. However, it is possible to enable
 * those clocks (significantly increasing power consumption in the reduced power mode),
 * making these wake-ups possible.
 *
 * @note This function also disables the interrupt in the NVIC (DisableIRQ() is called internally).
 *
 * @param interrupt The IRQ number.
 */
void DisableDeepSleepIRQ(IRQn_Type interrupt);
#endif /* FSL_FEATURE_SOC_SYSCON_COUNT */

#if defined(__cplusplus)
}
#endif

/*! @} */

#endif /* _FSL_COMMON_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 * 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 _FSL_UART_H_
#define _FSL_UART_H_

#include "fsl_common.h"

/*!
 * @addtogroup uart_driver
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @name Driver version */
/*@{*/
/*! @brief UART driver version 2.1.4. */
#define FSL_UART_DRIVER_VERSION (MAKE_VERSION(2, 1, 4))
/*@}*/

/*! @brief Error codes for the UART driver. */
enum _uart_status
{
    kStatus_UART_TxBusy = MAKE_STATUS(kStatusGroup_UART, 0),              /*!< Transmitter is busy. */
    kStatus_UART_RxBusy = MAKE_STATUS(kStatusGroup_UART, 1),              /*!< Receiver is busy. */
    kStatus_UART_TxIdle = MAKE_STATUS(kStatusGroup_UART, 2),              /*!< UART transmitter is idle
    kStatus_UART_RxIdle = MAKE_STATUS(kStatusGroup_UART, 3),              /*!< UART receiver is idle. *
    kStatus_UART_TxWatermarkTooLarge = MAKE_STATUS(kStatusGroup_UART, 4), /*!< TX FIFO water
    kStatus_UART_RxWatermarkTooLarge = MAKE_STATUS(kStatusGroup_UART, 5), /*!< RX FIFO wate
    kStatus_UART_FlagCannotClearManually =
        MAKE_STATUS(kStatusGroup_UART, 6),                                /*!< UART flag can't be manually cleare
    kStatus_UART_Error = MAKE_STATUS(kStatusGroup_UART, 7),               /*!< Error happens on UART
    kStatus_UART_RxRingBufferOverrun = MAKE_STATUS(kStatusGroup_UART, 8), /*!< UART RX softwa
    kStatus_UART_RxHardwareOverrun = MAKE_STATUS(kStatusGroup_UART, 9),   /*!< UART RX recei
    kStatus_UART_NoiseError = MAKE_STATUS(kStatusGroup_UART, 10),         /*!< UART noise error. */
    kStatus_UART_FramingError = MAKE_STATUS(kStatusGroup_UART, 11),       /*!< UART framing error
    kStatus_UART_ParityError = MAKE_STATUS(kStatusGroup_UART, 12),        /*!< UART parity error. */
    kStatus_UART_BaudrateNotSupport =
        MAKE_STATUS(kStatusGroup_UART, 13), /*!< Baudrate is not support in current clock source */
};

/*! @brief UART parity mode. */
typedef enum _uart_parity_mode
{
    kUART_ParityDisabled = 0x0U, /*!< Parity disabled */
    kUART_ParityEven = 0x2U,     /*!< Parity enabled, type even, bit setting: PE|PT = 10 */
    kUART_ParityOdd = 0x3U,      /*!< Parity enabled, type odd,  bit setting: PE|PT = 11 */
} uart_parity_mode_t;
```

```
/*! @brief UART stop bit count. */
typedef enum _uart_stop_bit_count
{
    kUART_OneStopBit = 0U, /*!< One stop bit */
    kUART_TwoStopBit = 1U, /*!< Two stop bits */
} uart_stop_bit_count_t;

/*!
 * @brief UART interrupt configuration structure, default settings all disabled.
 *
 * This structure contains the settings for all of the UART interrupt configurations.
 */
enum _uart_interrupt_enable
{
#if defined(FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_UART_HAS_LIN_B
    kUART_LinBreakInterruptEnable = (UART_BDH_LBKDIE_MASK), /*!< LIN break detect interrupt. */
#endif
    kUART_RxActiveEdgeInterruptEnable = (UART_BDH_RXEDGIE_MASK),   /*!< RX active edge interrup
    kUART_TxDataRegEmptyInterruptEnable = (UART_C2_TIE_MASK << 8), /*!< Transmit data register er
    kUART_TransmissionCompleteInterruptEnable = (UART_C2_TCIE_MASK << 8), /*!< Transmission com
    kUART_RxDataRegFullInterruptEnable = (UART_C2_RIE_MASK << 8),        /*!< Receiver data register
    kUART_IdleLineInterruptEnable = (UART_C2_ILIE_MASK << 8),            /*!< Idle line interrupt. */
    kUART_RxOverrunInterruptEnable = (UART_C3_ORIE_MASK << 16),          /*!< Receiver overrun inter
    kUART_NoiseErrorInterruptEnable = (UART_C3_NEIE_MASK << 16),         /*!< Noise error flag interrup
    kUART_FramingErrorInterruptEnable = (UART_C3_FEIE_MASK << 16),       /*!< Framing error flag inte
    kUART_ParityErrorInterruptEnable = (UART_C3_PEIE_MASK << 16),        /*!< Parity error flag interrup
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
    kUART_RxFifoOverflowInterruptEnable = (UART_CFIFO_RXOFE_MASK << 24),  /*!< RX FIFO overflow
    kUART_TxFifoOverflowInterruptEnable = (UART_CFIFO_TXOFE_MASK << 24),  /*!< TX FIFO overflow
    kUART_RxFifoUnderflowInterruptEnable = (UART_CFIFO_RXUFE_MASK << 24), /*!< RX FIFO underfl
#endif
    kUART_AllInterruptsEnable =
#if defined(FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_UART_HAS_LIN_B
        kUART_LinBreakInterruptEnable |
#endif
        kUART_RxActiveEdgeInterruptEnable | kUART_TxDataRegEmptyInterruptEnable |
        kUART_TransmissionCompleteInterruptEnable | kUART_RxDataRegFullInterruptEnable | kUART_Idl
        kUART_RxOverrunInterruptEnable | kUART_NoiseErrorInterruptEnable | kUART_FramingErrorInterru
        kUART_ParityErrorInterruptEnable
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
        |
        kUART_RxFifoOverflowInterruptEnable | kUART_TxFifoOverflowInterruptEnable | kUART_RxFifoUnd
#endif
    ,
};

/*!
 * @brief UART status flags.
 *
 * This provides constants for the UART status flags for use in the UART functions.
 */
enum _uart_flags
```

```
{
   kUART_TxDataRegEmptyFlag = (UART_S1_TDRE_MASK),     /*!< TX data register empty flag. */
   kUART_TransmissionCompleteFlag = (UART_S1_TC_MASK), /*!< Transmission complete flag. */
   kUART_RxDataRegFullFlag = (UART_S1_RDRF_MASK),      /*!< RX data register full flag. */
   kUART_IdleLineFlag = (UART_S1_IDLE_MASK),           /*!< Idle line detect flag. */
   kUART_RxOverrunFlag = (UART_S1_OR_MASK),            /*!< RX overrun flag. */
   kUART_NoiseErrorFlag = (UART_S1_NF_MASK),           /*!< RX takes 3 samples of each received bit.
                                                If any of these samples differ, noise flag sets */
   kUART_FramingErrorFlag = (UART_S1_FE_MASK),         /*!< Frame error flag, sets if logic 0 was detec
                                                where stop bit expected */
   kUART_ParityErrorFlag = (UART_S1_PF_MASK),          /*!< If parity enabled, sets upon parity error dete
#if defined(FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_UART_HAS_LIN_B
   kUART_LinBreakFlag =
      (UART_S2_LBKDIF_MASK
       << 8), /*!< LIN break detect interrupt flag, sets when
                                             LIN break char detected and LIN circuit enabled */
#endif
   kUART_RxActiveEdgeFlag =
      (UART_S2_RXEDGIF_MASK << 8), /*!< RX pin active edge interrupt flag,
                                                          sets when active edge detected */
   kUART_RxActiveFlag =
      (UART_S2_RAF_MASK << 8), /*!< Receiver Active Flag (RAF),
                                                       sets at beginning of valid start bit */
#if defined(FSL_FEATURE_UART_HAS_EXTENDED_DATA_REGISTER_FLAGS) && FSL_FEATURE_U
   kUART_NoiseErrorInRxDataRegFlag = (UART_ED_NOISY_MASK << 16),    /*!< Noisy bit, sets if noise
   kUART_ParityErrorInRxDataRegFlag = (UART_ED_PARITYE_MASK << 16), /*!< Paritye bit, sets if pari
#endif
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
   kUART_TxFifoEmptyFlag = (UART_SFIFO_TXEMPT_MASK << 24),   /*!< TXEMPT bit, sets if TX buffer
   kUART_RxFifoEmptyFlag = (UART_SFIFO_RXEMPT_MASK << 24),   /*!< RXEMPT bit, sets if RX buffe
   kUART_TxFifoOverflowFlag = (UART_SFIFO_TXOF_MASK << 24),  /*!< TXOF bit, sets if TX buffer ove
   kUART_RxFifoOverflowFlag = (UART_SFIFO_RXOF_MASK << 24),  /*!< RXOF bit, sets if receive buffe
   kUART_RxFifoUnderflowFlag = (UART_SFIFO_RXUF_MASK << 24), /*!< RXUF bit, sets if receive buffe
#endif
};

/*! @brief UART configuration structure. */
typedef struct _uart_config
{
   uint32_t baudRate_Bps;         /*!< UART baud rate  */
   uart_parity_mode_t parityMode; /*!< Parity mode, disabled (default), even, odd */
#if defined(FSL_FEATURE_UART_HAS_STOP_BIT_CONFIG_SUPPORT) && FSL_FEATURE_UART_H
   uart_stop_bit_count_t stopBitCount; /*!< Number of stop bits, 1 stop bit (default) or 2 stop bits  */
#endif
#if defined(FSL_FEATURE_UART_HAS_FIFO) && FSL_FEATURE_UART_HAS_FIFO
   uint8_t txFifoWatermark; /*!< TX FIFO watermark */
   uint8_t rxFifoWatermark; /*!< RX FIFO watermark */
#endif
   bool enableTx; /*!< Enable TX */
   bool enableRx; /*!< Enable RX */
} uart_config_t;

/*! @brief UART transfer structure. */
```

```
typedef struct _uart_transfer
{
    uint8_t *data;   /*!< The buffer of data to be transfer.*/
    size_t dataSize; /*!< The byte count to be transfer. */
} uart_transfer_t;

/* Forward declaration of the handle typedef. */
typedef struct _uart_handle uart_handle_t;

/*! @brief UART transfer callback function. */
typedef void (*uart_transfer_callback_t)(UART_Type *base, uart_handle_t *handle, status_t status, void *u

/*! @brief UART handle structure. */
struct _uart_handle
{
    uint8_t *volatile txData;   /*!< Address of remaining data to send. */
    volatile size_t txDataSize; /*!< Size of the remaining data to send. */
    size_t txDataSizeAll;       /*!< Size of the data to send out. */
    uint8_t *volatile rxData;   /*!< Address of remaining data to receive. */
    volatile size_t rxDataSize; /*!< Size of the remaining data to receive. */
    size_t rxDataSizeAll;       /*!< Size of the data to receive. */

    uint8_t *rxRingBuffer;              /*!< Start address of the receiver ring buffer. */
    size_t rxRingBufferSize;            /*!< Size of the ring buffer. */
    volatile uint16_t rxRingBufferHead; /*!< Index for the driver to store received data into ring buffer. */
    volatile uint16_t rxRingBufferTail; /*!< Index for the user to get data from the ring buffer. */

    uart_transfer_callback_t callback; /*!< Callback function. */
    void *userData;                    /*!< UART callback function parameter.*/

    volatile uint8_t txState; /*!< TX transfer state. */
    volatile uint8_t rxState; /*!< RX transfer state */
};

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* _cplusplus */

/*!
 * @name Initialization and deinitialization
 * @{
 */

/*!
 * @brief Initializes a UART instance with a user configuration structure and peripheral clock.
 *
 * This function configures the UART module with the user-defined settings. The user can configure the co
 * structure and also get the default configuration by using the UART_GetDefaultConfig() function.
 * The example below shows how to use this API to configure UART.
 * @code
 *  uart_config_t uartConfig;
 *  uartConfig.baudRate_Bps = 115200U;
 *  uartConfig.parityMode = kUART_ParityDisabled;
 *  uartConfig.stopBitCount = kUART_OneStopBit;
 *  uartConfig.txFifoWatermark = 0;
 *  uartConfig.rxFifoWatermark = 1;
 *  UART_Init(UART1, &uartConfig, 20000000U);
 * @endcode
 *
 * @param base UART peripheral base address.
 * @param config Pointer to the user-defined configuration structure.
 * @param srcClock_Hz UART clock source frequency in HZ.
 * @retval kStatus_UART_BaudrateNotSupport Baudrate is not support in current clock source.
 * @retval kStatus_Success Status UART initialize succeed
 */
status_t UART_Init(UART_Type *base, const uart_config_t *config, uint32_t srcClock_Hz);

/*!
 * @brief Deinitializes a UART instance.
 *
 * This function waits for TX complete, disables TX and RX, and disables the UART clock.
 *
 * @param base UART peripheral base address.
 */
void UART_Deinit(UART_Type *base);

/*!
 * @brief Gets the default configuration structure.
 *
 * This function initializes the UART configuration structure to a default value. The default
 * values are as follows.
 *   uartConfig->baudRate_Bps = 115200U;
 *   uartConfig->bitCountPerChar = kUART_8BitsPerChar;
 *   uartConfig->parityMode = kUART_ParityDisabled;
 *   uartConfig->stopBitCount = kUART_OneStopBit;
 *   uartConfig->txFifoWatermark = 0;
 *   uartConfig->rxFifoWatermark = 1;
 *   uartConfig->enableTx = false;
 *   uartConfig->enableRx = false;
 *
 * @param config Pointer to configuration structure.
 */
void UART_GetDefaultConfig(uart_config_t *config);

/*!
 * @brief Sets the UART instance baud rate.
 *
 * This function configures the UART module baud rate. This function is used to update
 * the UART module baud rate after the UART module is initialized by the UART_Init.
 * @code
 *  UART_SetBaudRate(UART1, 115200U, 20000000U);
 * @endcode
 *
 * @param base UART peripheral base address.
 * @param baudRate_Bps UART baudrate to be set.
 * @param srcClock_Hz UART clock source freqency in Hz.
 * @retval kStatus_UART_BaudrateNotSupport Baudrate is not support in the current clock source.
 * @retval kStatus_Success Set baudrate succeeded.
 */
status_t UART_SetBaudRate(UART_Type *base, uint32_t baudRate_Bps, uint32_t srcClock_Hz);

/* @} */

/*!
 * @name Status
 * @{
 */

/*!
 * @brief Gets UART status flags.
 *
 * This function gets all UART status flags. The flags are returned as the logical
 * OR value of the enumerators @ref _uart_flags. To check a specific status,
 * compare the return value with enumerators in @ref _uart_flags.
 * For example, to check whether the TX is empty, do the following.
 * @code
 *     if (kUART_TxDataRegEmptyFlag & UART_GetStatusFlags(UART1))
 *     {
 *         ...
 *     }
 * @endcode
 *
 * @param base UART peripheral base address.
 * @return UART status flags which are ORed by the enumerators in the _uart_flags.
 */
uint32_t UART_GetStatusFlags(UART_Type *base);

/*!
 * @brief Clears status flags with the provided mask.
 *
 * This function clears UART status flags with a provided mask. An automatically cleared flag
 * can't be cleared by this function.
 * These flags can only be cleared or set by hardware.
 *    kUART_TxDataRegEmptyFlag, kUART_TransmissionCompleteFlag, kUART_RxDataRegFullFlag,
 *    kUART_RxActiveFlag, kUART_NoiseErrorInRxDataRegFlag, kUART_ParityErrorInRxDataRegFlag,
 *    kUART_TxFifoEmptyFlag,kUART_RxFifoEmptyFlag
 * Note that this API should be called when the Tx/Rx is idle. Otherwise it has no effect.
 *
 * @param base UART peripheral base address.
 * @param mask The status flags to be cleared; it is logical OR value of @ref _uart_flags.
 * @retval kStatus_UART_FlagCannotClearManually The flag can't be cleared by this function but
 *         it is cleared automatically by hardware.
 * @retval kStatus_Success Status in the mask is cleared.
 */
status_t UART_ClearStatusFlags(UART_Type *base, uint32_t mask);
```

```
/* @} */

/*!
 * @name Interrupts
 * @{
 */

/*!
 * @brief Enables UART interrupts according to the provided mask.
 *
 * This function enables the UART interrupts according to the provided mask. The mask
 * is a logical OR of enumeration members. See @ref _uart_interrupt_enable.
 * For example, to enable TX empty interrupt and RX full interrupt, do the following.
 * @code
 *     UART_EnableInterrupts(UART1,kUART_TxDataRegEmptyInterruptEnable | kUART_RxDataRegFullIn
 * @endcode
 *
 * @param base UART peripheral base address.
 * @param mask The interrupts to enable. Logical OR of @ref _uart_interrupt_enable.
 */
void UART_EnableInterrupts(UART_Type *base, uint32_t mask);

/*!
 * @brief Disables the UART interrupts according to the provided mask.
 *
 * This function disables the UART interrupts according to the provided mask. The mask
 * is a logical OR of enumeration members. See @ref _uart_interrupt_enable.
 * For example, to disable TX empty interrupt and RX full interrupt do the following.
 * @code
 *     UART_DisableInterrupts(UART1,kUART_TxDataRegEmptyInterruptEnable | kUART_RxDataRegFullI
 * @endcode
 *
 * @param base UART peripheral base address.
 * @param mask The interrupts to disable. Logical OR of @ref _uart_interrupt_enable.
 */
void UART_DisableInterrupts(UART_Type *base, uint32_t mask);

/*!
 * @brief Gets the enabled UART interrupts.
 *
 * This function gets the enabled UART interrupts. The enabled interrupts are returned
 * as the logical OR value of the enumerators @ref _uart_interrupt_enable. To check
 * a specific interrupts enable status, compare the return value with enumerators
 * in @ref _uart_interrupt_enable.
 * For example, to check whether TX empty interrupt is enabled, do the following.
 * @code
 *     uint32_t enabledInterrupts = UART_GetEnabledInterrupts(UART1);
 *
 *     if (kUART_TxDataRegEmptyInterruptEnable & enabledInterrupts)
 *     {
 *         ...
 *     }
 * @endcode
 *
 * @param base UART peripheral base address.
 * @return UART interrupt flags which are logical OR of the enumerators in @ref _uart_interrupt_enable.
 */
uint32_t UART_GetEnabledInterrupts(UART_Type *base);

/* @} */

#if defined(FSL_FEATURE_UART_HAS_DMA_SELECT) && FSL_FEATURE_UART_HAS_DMA_SELEC
/*!
 * @name DMA Control
 * @{
 */

/*!
 * @brief Gets the UART data register address.
 *
 * This function returns the UART data register address, which is mainly used by DMA/eDMA.
 *
 * @param base UART peripheral base address.
 * @return UART data register addresses which are used both by the transmitter and the receiver.
 */
static inline uint32_t UART_GetDataRegisterAddress(UART_Type *base)
{
    return (uint32_t) & (base->D);
}

/*!
 * @brief Enables or disables the UART transmitter DMA request.
 *
 * This function enables or disables the transmit data register empty flag, S1[TDRE], to generate the DMA r
 *
 * @param base UART peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void UART_EnableTxDMA(UART_Type *base, bool enable)
{
    if (enable)
    {
#if (defined(FSL_FEATURE_UART_IS_SCI) && FSL_FEATURE_UART_IS_SCI)
        base->C4 |= UART_C4_TDMAS_MASK;
#else
        base->C5 |= UART_C5_TDMAS_MASK;
#endif
        base->C2 |= UART_C2_TIE_MASK;
    }
    else
    {
#if (defined(FSL_FEATURE_UART_IS_SCI) && FSL_FEATURE_UART_IS_SCI)
        base->C4 &= ~UART_C4_TDMAS_MASK;
#else
        base->C5 &= ~UART_C5_TDMAS_MASK;
#endif
        base->C2 &= ~UART_C2_TIE_MASK;
    }
}

/*!
 * @brief Enables or disables the UART receiver DMA.
 *
 * This function enables or disables the receiver data register full flag, S1[RDRF], to generate DMA request
 *
 * @param base UART peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void UART_EnableRxDMA(UART_Type *base, bool enable)
{
    if (enable)
    {
#if (defined(FSL_FEATURE_UART_IS_SCI) && FSL_FEATURE_UART_IS_SCI)
        base->C4 |= UART_C4_RDMAS_MASK;
#else
        base->C5 |= UART_C5_RDMAS_MASK;
#endif
        base->C2 |= UART_C2_RIE_MASK;
    }
    else
    {
#if (defined(FSL_FEATURE_UART_IS_SCI) && FSL_FEATURE_UART_IS_SCI)
        base->C4 &= ~UART_C4_RDMAS_MASK;
#else
        base->C5 &= ~UART_C5_RDMAS_MASK;
#endif
        base->C2 &= ~UART_C2_RIE_MASK;
    }
}

/* @} */
#endif /* FSL_FEATURE_UART_HAS_DMA_SELECT */

/*!
 * @name Bus Operations
 * @{
 */

/*!
 * @brief Enables or disables the UART transmitter.
 *
 * This function enables or disables the UART transmitter.
 *
 * @param base UART peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void UART_EnableTx(UART_Type *base, bool enable)
{
    if (enable)
    {
        base->C2 |= UART_C2_TE_MASK;
    }
    else
    {
        base->C2 &= ~UART_C2_TE_MASK;
    }
}

/*!
 * @brief Enables or disables the UART receiver.
 *
 * This function enables or disables the UART receiver.
 *
 * @param base UART peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void UART_EnableRx(UART_Type *base, bool enable)
{
    if (enable)
    {
        base->C2 |= UART_C2_RE_MASK;
    }
    else
    {
        base->C2 &= ~UART_C2_RE_MASK;
    }
}

/*!
 * @brief Writes to the TX register.
 *
 * This function writes data to the TX register directly. The upper layer must ensure
 * that the TX register is empty or TX FIFO has empty room before calling this function.
 *
 * @param base UART peripheral base address.
 * @param data The byte to write.
 */
static inline void UART_WriteByte(UART_Type *base, uint8_t data)
{
    base->D = data;
}

/*!
 * @brief Reads the RX register directly.
 *
 * This function reads data from the RX register directly. The upper layer must
 * ensure that the RX register is full or that the TX FIFO has data before calling this function.
 *
 * @param base UART peripheral base address.
 * @return The byte read from UART data register.
 */
```

```
static inline uint8_t UART_ReadByte(UART_Type *base)
{
    return base->D;
}

/*!
 * @brief Writes to the TX register using a blocking method.
 *
 * This function polls the TX register, waits for the TX register to be empty or for the TX FIFO
 * to have room and writes data to the TX buffer.
 *
 * @note This function does not check whether all data is sent out to the bus.
 * Before disabling the TX, check kUART_TransmissionCompleteFlag to ensure that the TX is
 * finished.
 *
 * @param base UART peripheral base address.
 * @param data Start address of the data to write.
 * @param length Size of the data to write.
 */
void UART_WriteBlocking(UART_Type *base, const uint8_t *data, size_t length);

/*!
 * @brief Read RX data register using a blocking method.
 *
 * This function polls the RX register, waits for the RX register to be full or for RX FIFO to
 * have data, and reads data from the TX register.
 *
 * @param base UART peripheral base address.
 * @param data Start address of the buffer to store the received data.
 * @param length Size of the buffer.
 * @retval kStatus_UART_RxHardwareOverrun Receiver overrun occurred while receiving data.
 * @retval kStatus_UART_NoiseError A noise error occurred while receiving data.
 * @retval kStatus_UART_FramingError A framing error occurred while receiving data.
 * @retval kStatus_UART_ParityError A parity error occurred while receiving data.
 * @retval kStatus_Success Successfully received all data.
 */
status_t UART_ReadBlocking(UART_Type *base, uint8_t *data, size_t length);

/* @} */

/*!
 * @name Transactional
 * @{
 */

/*!
 * @brief Initializes the UART handle.
 *
 * This function initializes the UART handle which can be used for other UART
 * transactional APIs. Usually, for a specified UART instance,
 * call this API once to get the initialized handle.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param callback The callback function.
 * @param userData The parameter of the callback function.
 */
void UART_TransferCreateHandle(UART_Type *base,
                               uart_handle_t *handle,
                               uart_transfer_callback_t callback,
                               void *userData);

/*!
 * @brief Sets up the RX ring buffer.
 *
 * This function sets up the RX ring buffer to a specific UART handle.
 *
 * When the RX ring buffer is used, data received are stored into the ring buffer even when the
 * user doesn't call the UART_TransferReceiveNonBlocking() API. If data is already received
 * in the ring buffer, the user can get the received data from the ring buffer directly.
 *
 * @note When using the RX ring buffer, one byte is reserved for internal use. In other
 * words, if @p ringBufferSize is 32, only 31 bytes are used for saving data.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param ringBuffer Start address of the ring buffer for background receiving. Pass NULL to disable the ri
 * @param ringBufferSize Size of the ring buffer.
 */
void UART_TransferStartRingBuffer(UART_Type *base, uart_handle_t *handle, uint8_t *ringBuffer, size_t

/*!
 * @brief Aborts the background transfer and uninstalls the ring buffer.
 *
 * This function aborts the background transfer and uninstalls the ring buffer.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 */
void UART_TransferStopRingBuffer(UART_Type *base, uart_handle_t *handle);

/*!
 * @brief Transmits a buffer of data using the interrupt method.
 *
 * This function sends data using an interrupt method. This is a non-blocking function, which
 * returns directly without waiting for all data to be written to the TX register. When
 * all data is written to the TX register in the ISR, the UART driver calls the callback
 * function and passes the @ref kStatus_UART_TxIdle as status parameter.
 *
 * @note The kStatus_UART_TxIdle is passed to the upper layer when all data is written
 * to the TX register. However, it does not ensure that all data is sent out. Before disabling the TX,
 * check the kUART_TransmissionCompleteFlag to ensure that the TX is finished.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param xfer UART transfer structure. See  #uart_transfer_t.
 * @retval kStatus_Success Successfully start the data transmission.
 * @retval kStatus_UART_TxBusy Previous transmission still not finished; data not all written to TX registe
 * @retval kStatus_InvalidArgument Invalid argument.
 */
status_t UART_TransferSendNonBlocking(UART_Type *base, uart_handle_t *handle, uart_transfer_t *xfer

/*!
 * @brief Aborts the interrupt-driven data transmit.
 *
 * This function aborts the interrupt-driven data sending. The user can get the remainBytes to find out
 * how many bytes are not sent out.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 */
void UART_TransferAbortSend(UART_Type *base, uart_handle_t *handle);

/*!
 * @brief Gets the number of bytes written to the UART TX register.
 *
 * This function gets the number of bytes written to the UART TX
 * register by using the interrupt method.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param count Send bytes count.
 * @retval kStatus_NoTransferInProgress No send in progress.
 * @retval kStatus_InvalidArgument The parameter is invalid.
 * @retval kStatus_Success Get successfully through the parameter \p count;
 */
status_t UART_TransferGetSendCount(UART_Type *base, uart_handle_t *handle, uint32_t *count);

/*!
 * @brief Receives a buffer of data using an interrupt method.
 *
 * This function receives data using an interrupt method. This is a non-blocking function, which
 *  returns without waiting for all data to be received.
 * If the RX ring buffer is used and not empty, the data in the ring buffer is copied and
 * the parameter @p receivedBytes shows how many bytes are copied from the ring buffer.
 * After copying, if the data in the ring buffer is not enough to read, the receive
 * request is saved by the UART driver. When the new data arrives, the receive request
 * is serviced first. When all data is received, the UART driver notifies the upper layer
 * through a callback function and passes the status parameter @ref kStatus_UART_RxIdle.
 * For example, the upper layer needs 10 bytes but there are only 5 bytes in the ring buffer.
 * The 5 bytes are copied to the xfer->data and this function returns with the
 * parameter @p receivedBytes set to 5. For the left 5 bytes, newly arrived data is
 * saved from the xfer->data[5]. When 5 bytes are received, the UART driver notifies the upper layer.
 * If the RX ring buffer is not enabled, this function enables the RX and RX interrupt
 * to receive data to the xfer->data. When all data is received, the upper layer is notified.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param xfer UART transfer structure, see #uart_transfer_t.
 * @param receivedBytes Bytes received from the ring buffer directly.
 * @retval kStatus_Success Successfully queue the transfer into transmit queue.
 * @retval kStatus_UART_RxBusy Previous receive request is not finished.
 * @retval kStatus_InvalidArgument Invalid argument.
 */
status_t UART_TransferReceiveNonBlocking(UART_Type *base,
                                         uart_handle_t *handle,
                                         uart_transfer_t *xfer,
                                         size_t *receivedBytes);

/*!
 * @brief Aborts the interrupt-driven data receiving.
 *
 * This function aborts the interrupt-driven data receiving. The user can get the remainBytes to know
 * how many bytes are not received yet.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 */
void UART_TransferAbortReceive(UART_Type *base, uart_handle_t *handle);

/*!
 * @brief Gets the number of bytes that have been received.
 *
 * This function gets the number of bytes that have been received.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 * @param count Receive bytes count.
 * @retval kStatus_NoTransferInProgress No receive in progress.
 * @retval kStatus_InvalidArgument Parameter is invalid.
 * @retval kStatus_Success Get successfully through the parameter \p count;
 */
status_t UART_TransferGetReceiveCount(UART_Type *base, uart_handle_t *handle, uint32_t *count);

/*!
 * @brief UART IRQ handle function.
 *
 * This function handles the UART transmit and receive IRQ request.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 */
void UART_TransferHandleIRQ(UART_Type *base, uart_handle_t *handle);

/*!
 * @brief UART Error IRQ handle function.
 *
 * This function handles the UART error IRQ request.
 *
 * @param base UART peripheral base address.
 * @param handle UART handle pointer.
 */
```

```
void UART_TransferHandleErrorIRQ(UART_Type *base, uart_handle_t *handle);

/* @} */

#if defined(__cplusplus)
}
#endif

/*! @}*/

#endif /* _FSL_UART_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015-2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 _FSL_LPSCI_H_
#define _FSL_LPSCI_H_

#include "fsl_common.h"

/*!
 * @addtogroup lpsci_driver
 * @{
 */

/*******************************************************************************
 * Definitions
```

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

/*! @name Driver version */
/*@{*/
/*! @brief LPSCI driver version 2.0.3. */
#define FSL_LPSCI_DRIVER_VERSION (MAKE_VERSION(2, 0, 3))
/*@{*/

/*! @brief Error codes for the LPSCI driver. */
enum _lpsci_status
{
    kStatus_LPSCI_TxBusy = MAKE_STATUS(kStatusGroup_LPSCI, 0), /*!< Transmitter is busy. */
    kStatus_LPSCI_RxBusy = MAKE_STATUS(kStatusGroup_LPSCI, 1), /*!< Receiver is busy. */
    kStatus_LPSCI_TxIdle = MAKE_STATUS(kStatusGroup_LPSCI, 2), /*!< Transmitter is idle. */
    kStatus_LPSCI_RxIdle = MAKE_STATUS(kStatusGroup_LPSCI, 3), /*!< Receiver is idle. */
    kStatus_LPSCI_FlagCannotClearManually =
        MAKE_STATUS(kStatusGroup_LPSCI, 4), /*!< Status flag can't be manually cleared. */
    kStatus_LPSCI_BaudrateNotSupport =
        MAKE_STATUS(kStatusGroup_LPSCI, 5),                 /*!< Baudrate is not support in current clock so
    kStatus_LPSCI_Error = MAKE_STATUS(kStatusGroup_LPSCI, 6), /*!< Error happens on LPSCI */
    kStatus_LPSCI_RxRingBufferOverrun =
        MAKE_STATUS(kStatusGroup_LPSCI, 7),                              /*!< LPSCI RX software ring buffer over
    kStatus_LPSCI_RxHardwareOverrun = MAKE_STATUS(kStatusGroup_LPSCI, 8), /*!< LPSCI RX recei
    kStatus_LPSCI_NoiseError = MAKE_STATUS(kStatusGroup_LPSCI, 9),        /*!< LPSCI noise error. */
    kStatus_LPSCI_FramingError = MAKE_STATUS(kStatusGroup_LPSCI, 10),     /*!< LPSCI framing error
    kStatus_LPSCI_ParityError = MAKE_STATUS(kStatusGroup_LPSCI, 11),      /*!< LPSCI parity error. */
};

/*! @brief LPSCI parity mode.*/
typedef enum _lpsci_parity_mode
{
    kLPSCI_ParityDisabled = 0x0U, /*!< Parity disabled */
    kLPSCI_ParityEven = 0x2U,     /*!< Parity enabled, type even, bit setting: PE|PT = 10 */
    kLPSCI_ParityOdd = 0x3U,      /*!< Parity enabled, type odd,  bit setting: PE|PT = 11 */
} lpsci_parity_mode_t;

/*! @brief LPSCI stop bit count.*/
typedef enum _lpsci_stop_bit_count
{
    kLPSCI_OneStopBit = 0U, /*!< One stop bit */
    kLPSCI_TwoStopBit = 1U, /*!< Two stop bits */
} lpsci_stop_bit_count_t;

/*!
 * @brief LPSCI interrupt configuration structure, default settings all disabled.
 *
 * This structure contains the settings for all LPSCI interrupt configurations.
 */
enum _lpsci_interrupt_enable_t
{
#if defined(FSL_FEATURE_LPSCI_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_LPSCI_HAS_LIN_E
    kLPSCI_LinBreakInterruptEnable = (UART0_BDH_LBKDIE_MASK), /*!< LIN break detect interrupt. */
#endif
```

```
    kLPSCI_RxActiveEdgeInterruptEnable = (UART0_BDH_RXEDGIE_MASK),   /*!< RX Active Edge interru
    kLPSCI_TxDataRegEmptyInterruptEnable = (UART0_C2_TIE_MASK << 8), /*!< Transmit data register e
    kLPSCI_TransmissionCompleteInterruptEnable = (UART0_C2_TCIE_MASK << 8), /*!< Transmission co
    kLPSCI_RxDataRegFullInterruptEnable = (UART0_C2_RIE_MASK << 8),  /*!< Receiver data register fu
    kLPSCI_IdleLineInterruptEnable = (UART0_C2_ILIE_MASK << 8),      /*!< Idle line interrupt. */
    kLPSCI_RxOverrunInterruptEnable = (UART0_C3_ORIE_MASK << 16),    /*!< Receiver Overrun interru
    kLPSCI_NoiseErrorInterruptEnable = (UART0_C3_NEIE_MASK << 16),   /*!< Noise error flag interrupt. 
    kLPSCI_FramingErrorInterruptEnable = (UART0_C3_FEIE_MASK << 16), /*!< Framing error flag interru
    kLPSCI_ParityErrorInterruptEnable = (UART0_C3_PEIE_MASK << 16),  /*!< Parity error flag interrupt. *
    kLPSCI_AllInterruptsEnable = kLPSCI_RxActiveEdgeInterruptEnable | kLPSCI_TxDataRegEmptyInterru
                                 kLPSCI_TransmissionCompleteInterruptEnable | kLPSCI_RxDataRegFullInterruptEna
                                 kLPSCI_IdleLineInterruptEnable | kLPSCI_RxOverrunInterruptEnable |
                                 kLPSCI_NoiseErrorInterruptEnable | kLPSCI_FramingErrorInterruptEnable |
                                 kLPSCI_ParityErrorInterruptEnable
#if defined(FSL_FEATURE_LPSCI_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_LPSCI_HAS_LIN_E
                                 |
                                 kLPSCI_LinBreakInterruptEnable
#endif
    ,
};

/*!
 * @brief LPSCI status flags.
 *
 * This provides constants for the LPSCI status flags for use in the LPSCI functions.
 */
enum _lpsci_status_flag_t
{
    kLPSCI_TxDataRegEmptyFlag = (UART0_S1_TDRE_MASK), /*!< Tx data register empty flag, sets whe
    kLPSCI_TransmissionCompleteFlag =
        (UART0_S1_TC_MASK), /*!< Transmission complete flag, sets when transmission activity complete *
    kLPSCI_RxDataRegFullFlag =
        (UART0_S1_RDRF_MASK), /*!< Rx data register full flag, sets when the receive data buffer is full */
    kLPSCI_IdleLineFlag = (UART0_S1_IDLE_MASK), /*!< Idle line detect flag, sets when idle line detected
    kLPSCI_RxOverrunFlag =
        (UART0_S1_OR_MASK), /*!< Rx Overrun, sets when new data is received before data is read from re
    kLPSCI_NoiseErrorFlag = (UART0_S1_NF_MASK), /*!< Rx takes 3 samples of each received bit. If any
                                                 differ, noise flag sets */
    kLPSCI_FramingErrorFlag =
        (UART0_S1_FE_MASK), /*!< Frame error flag, sets if logic 0 was detected where stop bit expected */
    kLPSCI_ParityErrorFlag = (UART0_S1_PF_MASK), /*!< If parity enabled, sets upon parity error detectio
#if defined(FSL_FEATURE_LPSCI_HAS_LIN_BREAK_DETECT) && FSL_FEATURE_LPSCI_HAS_LIN_E
    kLPSCI_LinBreakFlag =
        (UART0_S2_LBKDIF_MASK
         << 8), /*!< LIN break detect interrupt flag, sets when LIN break char detected and LIN circuit enabled
#endif
    kLPSCI_RxActiveEdgeFlag =
        (UART0_S2_RXEDGIF_MASK << 8), /*!< Rx pin active edge interrupt flag, sets when active edge det
    kLPSCI_RxActiveFlag =
        (UART0_S2_RAF_MASK << 8), /*!< Receiver Active Flag (RAF), sets at beginning of valid start bit */
#if defined(FSL_FEATURE_LPSCI_HAS_EXTENDED_DATA_REGISTER_FLAGS) && FSL_FEATURE_L
    kLPSCI_NoiseErrorInRxDataRegFlag =
        (UART0_ED_NOISY_MASK << 16), /*!< NOISY bit, sets if noise detected in current data word */
    kLPSCI_ParityErrorInRxDataRegFlag =
        (UART0_ED_PARITYE_MASK << 16), /*!< PARITYE bit, sets if noise detected in current data word */
#endif
};

/*! @brief LPSCI configure structure.*/
typedef struct _lpsci_config
{
    uint32_t baudRate_Bps;          /*!< LPSCI baud rate  */
    lpsci_parity_mode_t parityMode; /*!< Parity mode, disabled (default), even, odd */
#if defined(FSL_FEATURE_LPSCI_HAS_STOP_BIT_CONFIG_SUPPORT) && FSL_FEATURE_LPSCI_H
    lpsci_stop_bit_count_t stopBitCount; /*!< Number of stop bits, 1 stop bit (default) or 2 stop bits  */
#endif
    bool enableTx; /*!< Enable TX */
    bool enableRx; /*!< Enable RX */
} lpsci_config_t;

/*! @brief LPSCI transfer structure. */
typedef struct _lpsci_transfer
{
    uint8_t *data;   /*!< The buffer of data to be transfer.*/
    size_t dataSize; /*!< The byte count to be transfer. */
} lpsci_transfer_t;

/* Forward declaration of the handle typedef. */
typedef struct _lpsci_handle lpsci_handle_t;

/*! @brief LPSCI transfer callback function. */
typedef void (*lpsci_transfer_callback_t)(UART0_Type *base, lpsci_handle_t *handle, status_t status, void

/*<! @brief LPSCI handle used for storing the state among transactional APIs' calling. This structure is only
 * transactional APIs. */
struct _lpsci_handle
{
    uint8_t *volatile txData;   /*!< Address of remaining data to send. */
    volatile size_t txDataSize; /*!< Size of the remaining data to send. */
    size_t txDataSizeAll;       /*!< Size of the data to send out. */
    uint8_t *volatile rxData;   /*!< Address of remaining data to receive. */
    volatile size_t rxDataSize; /*!< Size of the remaining data to receive. */
    size_t rxDataSizeAll;       /*!< Size of the data to receive. */

    uint8_t *rxRingBuffer;              /*!< Start address of the receiver ring buffer. */
    size_t rxRingBufferSize;            /*!< Size of the ring buffer. */
    volatile uint16_t rxRingBufferHead; /*!< Index for the driver to store received data into ring buffer. */
    volatile uint16_t rxRingBufferTail; /*!< Index for the user to get data from the ring buffer. */

    lpsci_transfer_callback_t callback; /*!< Callback function. */
    void *userData;                     /*!< LPSCI callback function parameter.*/

    volatile uint8_t txState; /*!< TX transfer state. */
    volatile uint8_t rxState; /*!< RX transfer state */
};
```

```
/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* _cplusplus */

/*!
 * @name Initialization and deinitialization
 * @{
 */

/*!
* @brief Initializes an LPSCI instance with the user configuration structure and the peripheral clock.
*
* This function configures the LPSCI module with user-defined settings. The user can configure the configu
* structure and can also get the default configuration by calling the LPSCI_GetDefaultConfig() function.
* Example below shows how to use this API to configure the LPSCI.
*  @code
*   lpsci_config_t lpsciConfig;
*   lpsciConfig.baudRate_Bps = 115200U;
*   lpsciConfig.parityMode = kLPSCI_ParityDisabled;
*   lpsciConfig.stopBitCount = kLPSCI_OneStopBit;
*   LPSCI_Init(UART0, &lpsciConfig, 20000000U);
*   @endcode
*
* @param base LPSCI peripheral base address.
* @param config Pointer to user-defined configuration structure.
* @param srcClock_Hz LPSCI clock source frequency in HZ.
* @retval kStatus_LPSCI_BaudrateNotSupport Baudrate is not support in current clock source.
* @retval kStatus_Success LPSCI initialize succeed
*/
status_t LPSCI_Init(UART0_Type *base, const lpsci_config_t *config, uint32_t srcClock_Hz);

/*!
 * @brief Deinitializes an LPSCI instance.
 *
 * This function waits for TX complete, disables TX and RX, and disables the LPSCI clock.
 *
 * @param base LPSCI peripheral base address.
 */
void LPSCI_Deinit(UART0_Type *base);

/*!
 * @brief Gets the default configuration structure and saves the configuration to a user-provided pointer.
 *
 * This function initializes the LPSCI configure structure to default value. the default
 * value are:
 *   lpsciConfig->baudRate_Bps = 115200U;
 *   lpsciConfig->parityMode = kLPSCI_ParityDisabled;
 *   lpsciConfig->stopBitCount = kLPSCI_OneStopBit;
 *   lpsciConfig->enableTx = false;
 *   lpsciConfig->enableRx = false;
 *
 * @param config Pointer to configuration structure.
 */
void LPSCI_GetDefaultConfig(lpsci_config_t *config);

/*!
 * @brief Sets the LPSCI instance baudrate.
 *
 * This function configures the LPSCI module baudrate. This function is used to update
 * the LPSCI module baudrate after the LPSCI module is initialized with the LPSCI_Init.
 * @code
 *  LPSCI_SetBaudRate(UART0, 115200U, 20000000U);
 * @endcode
 *
 * @param base LPSCI peripheral base address.
 * @param baudRate_Bps LPSCI baudrate to be set.
 * @param srcClock_Hz LPSCI clock source frequency in HZ.
 * @retval kStatus_LPSCI_BaudrateNotSupport Baudrate is not supported in the current clock source.
 * @retval kStatus_Success Set baudrate succeed
 */
status_t LPSCI_SetBaudRate(UART0_Type *base, uint32_t baudRate_Bps, uint32_t srcClock_Hz);

/* @} */

/*!
 * @name Status
 * @{
 */

/*!
 * @brief Gets LPSCI status flags.
 *
 * This function gets all LPSCI status flags. The flags are returned as the logical
 * OR value of the enumerators @ref _lpsci_flags. To check a specific status,
 * compare the return value to the enumerators in @ref _LPSCI_flags.
 * For example, to check whether the TX is empty:
  * @code
 *     if (kLPSCI_TxDataRegEmptyFlag | LPSCI_GetStatusFlags(UART0))
 *     {
 *         ...
 *     }
 * @endcode
 *
 * @param base LPSCI peripheral base address.
 * @return LPSCI status flags which are ORed by the enumerators in the _lpsci_flags.
 */
uint32_t LPSCI_GetStatusFlags(UART0_Type *base);

/* @brief Clears status flags with a provided mask.
*
* This function clears the LPSCI status flags with a provided mask. Automatically cleared flag
* can't be cleared by this function.
* Some flags can only be cleared or set by hardware. These flags are:
*    kLPSCI_TxDataRegEmptyFlag,kLPSCI_TransmissionCompleteFlag,kLPSCI_RxDataRegFullFlag,kLPS
*    kLPSCI_ParityErrorInRxDataRegFlag,kLPSCI_TxFifoEmptyFlag,kLPSCI_RxFifoEmptyFlag
* Note: This API should be called when the Tx/Rx is idle, otherwise it takes no effects.
*
* @param base LPSCI peripheral base address.
* @param mask The status flags to be cleared, it is logical OR value of @ref _LPSCI_flagss.
* @retval kStatus_LPSCI_FlagCannotClearManually  can't be cleared by this function but it is cleared
* automatically by hardware.
* @retval kStatus_Success Status in the mask are cleared.
*/
status_t LPSCI_ClearStatusFlags(UART0_Type *base, uint32_t mask);

/* @} */

/*!
 * @name Interrupts
 * @{
 */

/*!
* @brief Enables an LPSCI interrupt according to a provided mask.
*
* This function enables the LPSCI interrupts according to a provided mask. The mask
* is a logical OR of enumeration members. See @ref _lpsci_interrupt_enable.
* For example, to enable the TX empty interrupt and RX full interrupt:
* @code
*     LPSCI_EnableInterrupts(UART0,kLPSCI_TxDataRegEmptyInterruptEnable | kLPSCI_RxDataRegFullI
* @endcode
*
* @param base LPSCI peripheral base address.
* @param mask The interrupts to enable. Logical OR of @ref _lpsci_interrupt_enable.
*/
void LPSCI_EnableInterrupts(UART0_Type *base, uint32_t mask);

/*!
 * @brief Disables the LPSCI interrupt according to a provided mask.
 *
 * This function disables the LPSCI interrupts according to a provided mask. The mask
 * is a logical OR of enumeration members. See @ref _lpsci_interrupt_enable.
 * For example, to disable TX empty interrupt and RX full interrupt:
 * @code
 *     LPSCI_DisableInterrupts(UART0,kLPSCI_TxDataRegEmptyInterruptEnable | kLPSCI_RxDataRegFul
 * @endcode
 *
 * @param base LPSCI peripheral base address.
 * @param mask The interrupts to disable. Logical OR of @ref _LPSCI_interrupt_enable.
 */
void LPSCI_DisableInterrupts(UART0_Type *base, uint32_t mask);

/*!
 * @brief Gets the enabled LPSCI interrupts.
 *
 * This function gets the enabled LPSCI interrupts, which are returned
 * as the logical OR value of the enumerators @ref _lpsci_interrupt_enable. To check
 * a specific interrupts enable status, compare the return value to the enumerators
 * in @ref _LPSCI_interrupt_enable.
 * For example, to check whether TX empty interrupt is enabled:
 * @code
 *     uint32_t enabledInterrupts = LPSCI_GetEnabledInterrupts(UART0);
 *
 *     if (kLPSCI_TxDataRegEmptyInterruptEnable & enabledInterrupts)
 *     {
 *         ...
 *     }
 * @endcode
 *
 * @param base LPSCI peripheral base address.
 * @return LPSCI interrupt flags which are logical OR of the enumerators in @ref _LPSCI_interrupt_enable
 */
uint32_t LPSCI_GetEnabledInterrupts(UART0_Type *base);

/* @} */

#if defined(FSL_FEATURE_LPSCI_HAS_DMA_ENABLE) && FSL_FEATURE_LPSCI_HAS_DMA_ENABL

/*!
 * @name DMA Control
 * @{
 */

/*!
 * @brief Gets the LPSCI data register address.
 *
 * This function returns the LPSCI data register address, which is mainly used by DMA/eDMA case.
 *
 * @param base LPSCI peripheral base address.
 * @return LPSCI data register address which are used both by transmitter and receiver.
 */
static inline uint32_t LPSCI_GetDataRegisterAddress(UART0_Type *base)
{
    return (uint32_t) & (base->D);
}

/*!
 * @brief Enables or disable LPSCI transmitter DMA request.
 *
 * This function enables or disables the transmit data register empty flag, S1[TDRE], to generate DMA requ
 *
 * @param base LPSCI peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void LPSCI_EnableTxDMA(UART0_Type *base, bool enable)
{
    if (enable)
    {
        base->C5 |= UART0_C5_TDMAE_MASK;
        base->C2 |= UART0_C2_TIE_MASK;
    }
    else
    {
        base->C5 &= ~UART0_C5_TDMAE_MASK;
        base->C2 &= ~UART0_C2_TIE_MASK;
    }
}

/*!
 * @brief Enables or disables the LPSCI receiver DMA.
 *
 * This function enables or disables the receiver data register full flag, S1[RDRF], to generate DMA request
 *
 * @param base LPSCI peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void LPSCI_EnableRxDMA(UART0_Type *base, bool enable)
{
    if (enable)
    {
        base->C5 |= UART0_C5_RDMAE_MASK;
        base->C2 |= UART0_C2_RIE_MASK;
    }
    else
    {
        base->C5 &= ~UART0_C5_RDMAE_MASK;
        base->C2 &= ~UART0_C2_RIE_MASK;
    }
}

/* @} */
#endif /* defined(FSL_FEATURE_LPSCI_HAS_DMA_ENABLE) && FSL_FEATURE_LPSCI_HAS_DMA_E

/*!
 * @name Bus Operations
 * @{
 */

/*!
 * @brief Enables or disables the LPSCI transmitter.
 *
 * This function enables or disables the LPSCI transmitter.
 *
 * @param base LPSCI peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void LPSCI_EnableTx(UART0_Type *base, bool enable)
{
    if (enable)
    {
        base->C2 |= UART0_C2_TE_MASK;
    }
    else
    {
        base->C2 &= ~UART0_C2_TE_MASK;
    }
}

/*!
 * @brief Enables or disables the LPSCI receiver.
 *
 * This function enables or disables the LPSCI receiver.
 *
 * @param base LPSCI peripheral base address.
 * @param enable True to enable, false to disable.
 */
static inline void LPSCI_EnableRx(UART0_Type *base, bool enable)
{
    if (enable)
    {
        base->C2 |= UART0_C2_RE_MASK;
    }
    else
    {
        base->C2 &= ~UART0_C2_RE_MASK;
    }
}

/*!
 * @brief Writes to the TX register.
 *
 * This function writes data to the TX register directly. The upper layer must ensure
 * that the TX register is empty before calling this function.
 *
 * @param base LPSCI peripheral base address.
 * @param data Data write to TX register.
 */
static inline void LPSCI_WriteByte(UART0_Type *base, uint8_t data)
{
    base->D = data;
}

/*!
 * @brief Reads the RX data register.
 *
 * This function reads data from the RX register directly. The upper layer must
 * ensure that the RX register is full before calling this function.
 *
 * @param base LPSCI peripheral base address.
 * @return Data read from RX data register.
 */
static inline uint8_t LPSCI_ReadByte(UART0_Type *base)
{
    return base->D;
```

```
}

/*!
 * @brief Writes to the TX register using a blocking method.
 *
 * This function polls the TX register, waits for the TX register empty, and
 * writes data to the TX buffer.
 *
 * @note This function does not check whether all the data has been sent out to bus,
 * so before disable TX, check kLPSCI_TransmissionCompleteFlag to ensure the TX is
 * finished.
 *
 * @param base LPSCI peripheral base address.
 * @param data Start address of the data to write.
 * @param length Size of the data to write.
 */
void LPSCI_WriteBlocking(UART0_Type *base, const uint8_t *data, size_t length);

/*!
* @brief Reads the RX register using a blocking method.
*
* This function polls the RX register, waits for the RX register to be full, and
* reads data from the RX register.
*
* @param base LPSCI peripheral base address.
* @param data Start address of the buffer to store the received data.
* @param length Size of the buffer.
* @retval kStatus_LPSCI_RxHardwareOverrun Receiver overrun happened while receiving data.
* @retval kStatus_LPSCI_NoiseError Noise error happened while receiving data.
* @retval kStatus_LPSCI_FramingError Framing error happened while receiving data.
* @retval kStatus_LPSCI_ParityError Parity error happened while receiving data.
* @retval kStatus_Success Successfully received all data.
*/
status_t LPSCI_ReadBlocking(UART0_Type *base, uint8_t *data, size_t length);

/* @} */

/*!
 * @name Transactional
 * @{
 */

/*!
 * @brief Initializes the LPSCI handle.
 *
 * This function initializes the LPSCI handle, which can be used for other LPSCI
 * transactional APIs. Usually, for a specified LPSCI instance,
 * call this API once to get the initialized handle.
 *
 * LPSCI driver supports the "background" receiving, which means that the user can set up
 * an RX ring buffer optionally. Data received are stored into the ring buffer even when the
 * user doesn't call the LPSCI_TransferReceiveNonBlocking() API. If there is already data received
 * in the ring buffer, get the received data from the ring buffer directly.
 * The ring buffer is disabled if pass NULL as @p ringBuffer.
 *
 * @param handle LPSCI handle pointer.
 * @param base LPSCI peripheral base address.
 * @param ringBuffer Start address of ring buffer for background receiving. Pass NULL to disable the ring b
 * @param ringBufferSize size of the ring buffer.
 */
void LPSCI_TransferCreateHandle(UART0_Type *base,
                                lpsci_handle_t *handle,
                                lpsci_transfer_callback_t callback,
                                void *userData);

/*!
 * @brief Sets up the RX ring buffer.
 *
 * This function sets up the RX ring buffer to a specific LPSCI handle.
 *
 * When the RX ring buffer is used, data received is stored into the ring buffer even when
 * the user doesn't call the LPSCI_TransferReceiveNonBlocking() API. If there is already data received
 * in the ring buffer, the user can get the received data from the ring buffer directly.
 *
 * @note When using the RX ring buffer, one byte is reserved for internal use. In other
 * words, if @p ringBufferSize is 32, only 31 bytes are used for saving data.
 *
 * @param base LPSCI peripheral base address.
 * @param handle LPSCI handle pointer.
 * @param ringBuffer Start address of ring buffer for background receiving. Pass NULL to disable the ring b
 * @param ringBufferSize size of the ring buffer.
 */
void LPSCI_TransferStartRingBuffer(UART0_Type *base,
                                   lpsci_handle_t *handle,
                                   uint8_t *ringBuffer,
                                   size_t ringBufferSize);

/*!
 * @brief Aborts the background transfer and uninstalls the ring buffer.
 *
 * This function aborts the background transfer and uninstalls the ringbuffer.
 *
 * @param base LPSCI peripheral base address.
 * @param handle LPSCI handle pointer.
 */
void LPSCI_TransferStopRingBuffer(UART0_Type *base, lpsci_handle_t *handle);

/*!
 * @brief Transmits a buffer of data using the interrupt method.
 *
 * This function sends data using the interrupt method. This is a non-blocking function, which
 * returns directly without waiting for all data to be written to the TX register. When
 * all data is written to the TX register in ISR, LPSCI driver calls the callback
 * function and passes the @ref kStatus_LPSCI_TxIdle as status parameter.
 *
 * @note The kStatus_LPSCI_TxIdle is passed to the upper layer when all data is written
 * to the TX register. However, it does not ensure that all data is sent out. Before disabling the TX,
 * check the kLPSCI_TransmissionCompleteFlag to ensure that the TX is complete.
 *
 * @param handle LPSCI handle pointer.
 * @param xfer LPSCI transfer structure, refer to #LPSCI_transfer_t.
 * @retval kStatus_Success Successfully start the data transmission.
 * @retval kStatus_LPSCI_TxBusy Previous transmission still not finished, data not all written to the TX reg
 * @retval kStatus_InvalidArgument Invalid argument.
 */
status_t LPSCI_TransferSendNonBlocking(UART0_Type *base, lpsci_handle_t *handle, lpsci_transfer_t *x

/*!
 * @brief Aborts the interrupt-driven data transmit.
 *
 * This function aborts the interrupt driven data send.
 *
 * @param handle LPSCI handle pointer.
 */
void LPSCI_TransferAbortSend(UART0_Type *base, lpsci_handle_t *handle);

/*!
 * @brief Get the number of bytes that have been written to LPSCI TX register.
 *
 * This function gets the number of bytes that have been written to LPSCI TX
 * register by interrupt method.
 *
 * @param base LPSCI peripheral base address.
 * @param handle LPSCI handle pointer.
 * @param count Send bytes count.
 * @retval kStatus_NoTransferInProgress No send in progress.
 * @retval kStatus_InvalidArgument Parameter is invalid.
 * @retval kStatus_Success Get successfully through the parameter \p count;
 */
status_t LPSCI_TransferGetSendCount(UART0_Type *base, lpsci_handle_t *handle, uint32_t *count);

/*!
 * @brief Receives buffer of data using the interrupt method.
 *
 * This function receives data using the interrupt method. This is a non-blocking function
 * which returns without waiting for all data to be received.
 * If the RX ring buffer is used and not empty, the data in ring buffer is copied and
 * the parameter @p receivedBytes shows how many bytes are copied from the ring buffer.
 * After copying, if the data in ring buffer is not enough to read, the receive
 * request is saved by the LPSCI driver. When new data arrives, the receive request
 * is serviced first. When all data is received, the LPSCI driver notifies the upper layer
 * through a callback function and passes the status parameter @ref kStatus_LPSCI_RxIdle.
 * For example, the upper layer needs 10 bytes but there are only 5 bytes in the ring buffer.
 * The 5 bytes are copied to the xfer->data and the function returns with the
 * parameter @p receivedBytes set to 5. For the remaining 5 bytes, newly arrived data is
 * saved from the xfer->data[5]. When 5 bytes are received, the LPSCI driver notifies the upper layer.
 * If the RX ring buffer is not enabled, this function enables the RX and RX interrupt
 * to receive data to the xfer->data. When all data is received, the upper layer is notified.
 *
 * @param handle LPSCI handle pointer.
 * @param xfer lpsci transfer structure. See #lpsci_transfer_t.
 * @param receivedBytes Bytes received from the ring buffer directly.
 * @retval kStatus_Success Successfully queue the transfer into transmit queue.
 * @retval kStatus_LPSCI_RxBusy Previous receive request is not finished.
 * @retval kStatus_InvalidArgument Invalid argument.
 */
status_t LPSCI_TransferReceiveNonBlocking(UART0_Type *base,
                                          lpsci_handle_t *handle,
                                          lpsci_transfer_t *xfer,
                                          size_t *receivedBytes);

/*!
 * @brief Aborts interrupt driven data receiving.
 *
 * This function aborts interrupt driven data receiving.
 *
 * @param handle LPSCI handle pointer.
 */
void LPSCI_TransferAbortReceive(UART0_Type *base, lpsci_handle_t *handle);

/*!
 * @brief Get the number of bytes that have been received.
 *
 * This function gets the number of bytes that have been received.
 *
 * @param base LPSCI peripheral base address.
 * @param handle LPSCI handle pointer.
 * @param count Receive bytes count.
 * @retval kStatus_NoTransferInProgress No receive in progress.
 * @retval kStatus_InvalidArgument Parameter is invalid.
 * @retval kStatus_Success Get successfully through the parameter \p count;
 */
status_t LPSCI_TransferGetReceiveCount(UART0_Type *base, lpsci_handle_t *handle, uint32_t *count);

/*!
 * @brief LPSCI IRQ handle function.
 *
 * This function handles the LPSCI transmit and receive IRQ request.
 *
 * @param handle LPSCI handle pointer.
 */
void LPSCI_TransferHandleIRQ(UART0_Type *base, lpsci_handle_t *handle);

/*!
 * @brief LPSCI Error IRQ handle function.
 *
 * This function handle the LPSCI error IRQ request.
 *
 * @param handle LPSCI handle pointer.
 */
void LPSCI_TransferHandleErrorIRQ(UART0_Type *base, lpsci_handle_t *handle);
```

```
/* @} */

#if defined(__cplusplus)
}
#endif

/*! @}*/

#endif /* _FSL_LPSCI_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
#ifndef _FSL_GPIO_H_
#define _FSL_GPIO_H_

#include "fsl_common.h"

/*!
 * @addtogroup gpio
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/
```

```
/*! @name Driver version */
/*@{*/
/*! @brief GPIO driver version 2.1.1. */
#define FSL_GPIO_DRIVER_VERSION (MAKE_VERSION(2, 1, 1))
/*@}*/

/*! @brief GPIO direction definition */
typedef enum _gpio_pin_direction
{
   kGPIO_DigitalInput = 0U,  /*!< Set current pin as digital input*/
   kGPIO_DigitalOutput = 1U, /*!< Set current pin as digital output*/
} gpio_pin_direction_t;

#if defined(FSL_FEATURE_GPIO_HAS_ATTRIBUTE_CHECKER) && FSL_FEATURE_GPIO_HAS_ATTR
/*! @brief GPIO checker attribute */
typedef enum _gpio_checker_attribute
{
   kGPIO_UsernonsecureRWUsersecureRWPrivilegedsecureRW =
      0x00U, /*!< User nonsecure:Read+Write; User Secure:Read+Write; Privileged Secure:Read+Write */
   kGPIO_UsernonsecureRUsersecureRWPrivilegedsecureRW =
      0x01U, /*!< User nonsecure:Read;       User Secure:Read+Write; Privileged Secure:Read+Write */
   kGPIO_UsernonsecureNUsersecureRWPrivilegedsecureRW =
      0x02U, /*!< User nonsecure:None;       User Secure:Read+Write; Privileged Secure:Read+Write */
   kGPIO_UsernonsecureRUsersecureRPrivilegedsecureRW =
      0x03U, /*!< User nonsecure:Read;       User Secure:Read;       Privileged Secure:Read+Write */
   kGPIO_UsernonsecureNUsersecureRPrivilegedsecureRW =
      0x04U, /*!< User nonsecure:None;       User Secure:Read;       Privileged Secure:Read+Write */
   kGPIO_UsernonsecureNUsersecureNPrivilegedsecureRW =
      0x05U, /*!< User nonsecure:None;       User Secure:None;       Privileged Secure:Read+Write */
   kGPIO_UsernonsecureNUsersecureNPrivilegedsecureR =
      0x06U, /*!< User nonsecure:None;       User Secure:None;       Privileged Secure:Read */
   kGPIO_UsernonsecureNUsersecureNPrivilegedsecureN =
      0x07U, /*!< User nonsecure:None;       User Secure:None;       Privileged Secure:None */
   kGPIO_IgnoreAttributeCheck = 0x10U, /*!< Ignores the attribute check */
} gpio_checker_attribute_t;
#endif

/*!
 * @brief The GPIO pin configuration structure.
 *
 * Each pin can only be configured as either an output pin or an input pin at a time.
 * If configured as an input pin, leave the outputConfig unused.
 * Note that in some use cases, the corresponding port property should be configured in advance
 *        with the PORT_SetPinConfig().
 */
typedef struct _gpio_pin_config
{
   gpio_pin_direction_t pinDirection; /*!< GPIO direction, input or output */
   /* Output configurations; ignore if configured as an input pin */
   uint8_t outputLogic; /*!< Set a default output logic, which has no use in input */
} gpio_pin_config_t;
```

```
/*! @} */

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif

/*!
 * @addtogroup gpio_driver
 * @{
 */

/*! @name GPIO Configuration */
/*@{*/

/*!
 * @brief Initializes a GPIO pin used by the board.
 *
 * To initialize the GPIO, define a pin configuration, as either input or output, in the user file.
 * Then, call the GPIO_PinInit() function.
 *
 * This is an example to define an input pin or an output pin configuration.
 * @code
 * // Define a digital input pin configuration,
 * gpio_pin_config_t config =
 * {
 *   kGPIO_DigitalInput,
 *   0,
 * }
 * //Define a digital output pin configuration,
 * gpio_pin_config_t config =
 * {
 *   kGPIO_DigitalOutput,
 *   0,
 * }
 * @endcode
 *
 * @param base   GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param pin    GPIO port pin number
 * @param config GPIO pin configuration pointer
 */
void GPIO_PinInit(GPIO_Type *base, uint32_t pin, const gpio_pin_config_t *config);

/*@}*/

/*! @name GPIO Output Operations */
/*@{*/

/*!
 * @brief Sets the output level of the multiple GPIO pins to the logic 1 or 0.
 *
 * @param base    GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param pin     GPIO pin number
 * @param output  GPIO pin output logic level.
 *        - 0: corresponding pin output low-logic level.
 *        - 1: corresponding pin output high-logic level.
 */
static inline void GPIO_WritePinOutput(GPIO_Type *base, uint32_t pin, uint8_t output)
{
    if (output == 0U)
    {
        base->PCOR = 1U << pin;
    }
    else
    {
        base->PSOR = 1U << pin;
    }
}

/*!
 * @brief Sets the output level of the multiple GPIO pins to the logic 1.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_SetPinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PSOR = mask;
}

/*!
 * @brief Sets the output level of the multiple GPIO pins to the logic 0.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_ClearPinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PCOR = mask;
}

/*!
 * @brief Reverses the current output logic of the multiple GPIO pins.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
static inline void GPIO_TogglePinsOutput(GPIO_Type *base, uint32_t mask)
{
    base->PTOR = mask;
}
/*@}*/
```

```
/*! @name GPIO Input Operations */
/*@{*/

/*!
 * @brief Reads the current input value of the GPIO port.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param pin     GPIO pin number
 * @retval GPIO port input value
 *        - 0: corresponding pin input low-logic level.
 *        - 1: corresponding pin input high-logic level.
 */
static inline uint32_t GPIO_ReadPinInput(GPIO_Type *base, uint32_t pin)
{
    return (((base->PDIR) >> pin) & 0x01U);
}
/*@}*/

/*! @name GPIO Interrupt */
/*@{*/

/*!
 * @brief Reads the GPIO port interrupt status flag.
 *
 * If a pin is configured to generate the DMA request, the corresponding flag
 * is cleared automatically at the completion of the requested DMA transfer.
 * Otherwise, the flag remains set until a logic one is written to that flag.
 * If configured for a level sensitive interrupt that remains asserted, the flag
 * is set again immediately.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @retval The current GPIO port interrupt status flag, for example, 0x00010001 means the
 *         pin 0 and 17 have the interrupt.
 */
uint32_t GPIO_GetPinsInterruptFlags(GPIO_Type *base);

/*!
 * @brief Clears multiple GPIO pin interrupt status flags.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
void GPIO_ClearPinsInterruptFlags(GPIO_Type *base, uint32_t mask);

#if defined(FSL_FEATURE_GPIO_HAS_ATTRIBUTE_CHECKER) && FSL_FEATURE_GPIO_HAS_ATTR
/*!
 * @brief The GPIO module supports a device-specific number of data ports, organized as 32-bit
 * words. Each 32-bit data port includes a GACR register, which defines the byte-level
 * attributes required for a successful access to the GPIO programming model. The attribute controls for th
 * bytes in the GACR follow a standard little endian
 * data convention.
 *
 * @param base GPIO peripheral base pointer (GPIOA, GPIOB, GPIOC, and so on.)
 * @param mask GPIO pin number macro
 */
void GPIO_CheckAttributeBytes(GPIO_Type *base, gpio_checker_attribute_t attribute);
#endif

/*@}*/
/*! @} */

/*!
 * @addtogroup fgpio_driver
 * @{
 */

/*
 * Introduces the FGPIO feature.
 *
 * The FGPIO features are only support on some Kinetis MCUs. The FGPIO registers are aliased to the IOI
 * interface. Accesses via the IOPORT interface occur in parallel with any instruction fetches and
 * complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.
 */

#if defined(FSL_FEATURE_SOC_FGPIO_COUNT) && FSL_FEATURE_SOC_FGPIO_COUNT

/*! @name FGPIO Configuration */
/*@{*/

/*!
 * @brief Initializes a FGPIO pin used by the board.
 *
 * To initialize the FGPIO driver, define a pin configuration, as either input or output, in the user file.
 * Then, call the FGPIO_PinInit() function.
 *
 * This is an example to define an input pin or an output pin configuration:
 * @code
 * // Define a digital input pin configuration,
 * gpio_pin_config_t config =
 * {
 *   kGPIO_DigitalInput,
 *   0,
 * }
 * //Define a digital output pin configuration,
 * gpio_pin_config_t config =
 * {
 *   kGPIO_DigitalOutput,
 *   0,
 * }
 * @endcode
 *
 * @param base   FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param pin    FGPIO port pin number
 * @param config FGPIO pin configuration pointer
 */
void FGPIO_PinInit(FGPIO_Type *base, uint32_t pin, const gpio_pin_config_t *config);
```

```
/*@}*/

/*! @name FGPIO Output Operations */
/*@{*/

/*!
 * @brief Sets the output level of the multiple FGPIO pins to the logic 1 or 0.
 *
 * @param base    FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param pin     FGPIO pin number
 * @param output  FGPIOpin output logic level.
 *        - 0: corresponding pin output low-logic level.
 *        - 1: corresponding pin output high-logic level.
 */
static inline void FGPIO_WritePinOutput(FGPIO_Type *base, uint32_t pin, uint8_t output)
{
    if (output == 0U)
    {
        base->PCOR = 1 << pin;
    }
    else
    {
        base->PSOR = 1 << pin;
    }
}

/*!
 * @brief Sets the output level of the multiple FGPIO pins to the logic 1.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param mask FGPIO pin number macro
 */
static inline void FGPIO_SetPinsOutput(FGPIO_Type *base, uint32_t mask)
{
    base->PSOR = mask;
}

/*!
 * @brief Sets the output level of the multiple FGPIO pins to the logic 0.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param mask FGPIO pin number macro
 */
static inline void FGPIO_ClearPinsOutput(FGPIO_Type *base, uint32_t mask)
{
    base->PCOR = mask;
}

/*!
 * @brief Reverses the current output logic of the multiple FGPIO pins.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param mask FGPIO pin number macro
 */
static inline void FGPIO_TogglePinsOutput(FGPIO_Type *base, uint32_t mask)
{
    base->PTOR = mask;
}
/*@}*/

/*! @name FGPIO Input Operations */
/*@{*/

/*!
 * @brief Reads the current input value of the FGPIO port.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param pin  FGPIO pin number
 * @retval FGPIO port input value
 *        - 0: corresponding pin input low-logic level.
 *        - 1: corresponding pin input high-logic level.
 */
static inline uint32_t FGPIO_ReadPinInput(FGPIO_Type *base, uint32_t pin)
{
    return (((base->PDIR) >> pin) & 0x01U);
}
/*@}*/

/*! @name FGPIO Interrupt */
/*@{*/

/*!
 * @brief Reads the FGPIO port interrupt status flag.
 *
 * If a pin is configured to generate the DMA request, the corresponding flag
 * is cleared automatically at the completion of the requested DMA transfer.
 * Otherwise, the flag remains set until a logic one is written to that flag.
 * If configured for a level-sensitive interrupt that remains asserted, the flag
 * is set again immediately.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @retval The current FGPIO port interrupt status flags, for example, 0x00010001 means the
 *         pin 0 and 17 have the interrupt.
 */
uint32_t FGPIO_GetPinsInterruptFlags(FGPIO_Type *base);

/*!
 * @brief Clears the multiple FGPIO pin interrupt status flag.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param mask FGPIO pin number macro
 */
void FGPIO_ClearPinsInterruptFlags(FGPIO_Type *base, uint32_t mask);

#if defined(FSL_FEATURE_GPIO_HAS_ATTRIBUTE_CHECKER) && FSL_FEATURE_GPIO_HAS_ATTR
```

```
/*!
 * @brief The FGPIO module supports a device-specific number of data ports, organized as 32-bit
 * words. Each 32-bit data port includes a GACR register, which defines the byte-level
 * attributes required for a successful access to the GPIO programming model. The attribute controls for th
 * bytes in the GACR follow a standard little endian
 * data convention.
 *
 * @param base FGPIO peripheral base pointer (FGPIOA, FGPIOB, FGPIOC, and so on.)
 * @param mask FGPIO pin number macro
 */
void FGPIO_CheckAttributeBytes(FGPIO_Type *base, gpio_checker_attribute_t attribute);
#endif

/*@}*/

#endif /* FSL_FEATURE_SOC_FGPIO_COUNT */

#if defined(__cplusplus)
}
#endif

/*!
 * @}
 */

#endif /* _FSL_GPIO_H_*/
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright 2017-2019 NXP
 * 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 NXP 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" AN
 * 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 FO
 * 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.
 */

/**
 * @file    peripherals.h
 * @brief   Peripherals initialization header file.
 */

/* This is a template for board specific configuration created by MCUXpresso IDE Project Wizard.*/

#ifndef _PERIPHERALS_H_
#define _PERIPHERALS_H_

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/**
 * @brief  Initialize peripherals specific settings.
 */
void BOARD_InitBootPeripherals(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus */

#endif /* _PERIPHERALS_H_ */


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/***********************************************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits made to this file
 * will be overwritten if the respective MCUXpresso Config Tools is used to update this file.
 **********************************************************************************************************************/

#ifndef _CLOCK_CONFIG_H_
#define _CLOCK_CONFIG_H_

#include "fsl_common.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define BOARD_XTAL0_CLK_HZ                         8000000U  /*!< Board xtal0 frequency in Hz */

/*******************************************************************************
 ************************ BOARD_InitBootClocks function ************************
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/
```

```
/*!
 * @brief This function executes default configuration of clocks.
 *
 */
void BOARD_InitBootClocks(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus*/

/*******************************************************************************
 ********************** Configuration BOARD_BootClockRUN ***********************
 ******************************************************************************/
/*******************************************************************************
 * Definitions for BOARD_BootClockRUN configuration
 ******************************************************************************/
#define BOARD_BOOTCLOCKRUN_CORE_CLOCK             48000000U  /*!< Core clock frequency: 4800

/*! @brief MCG set for BOARD_BootClockRUN configuration.
 */
extern const mcg_config_t mcgConfig_BOARD_BootClockRUN;
/*! @brief SIM module set for BOARD_BootClockRUN configuration.
 */
extern const sim_clock_config_t simConfig_BOARD_BootClockRUN;
/*! @brief OSC set for BOARD_BootClockRUN configuration.
 */
extern const osc_config_t oscConfig_BOARD_BootClockRUN;

/*******************************************************************************
 * API for BOARD_BootClockRUN configuration
 ******************************************************************************/
#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/

/*!
 * @brief This function executes configuration of clocks.
 *
 */
void BOARD_BootClockRUN(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus*/

/*******************************************************************************
 ********************* Configuration BOARD_BootClockVLPR ***********************
 ******************************************************************************/
/*******************************************************************************
 * Definitions for BOARD_BootClockVLPR configuration
 ******************************************************************************/
#define BOARD_BOOTCLOCKVLPR_CORE_CLOCK             4000000U  /*!< Core clock frequency: 4000
```

```
/*! @brief MCG set for BOARD_BootClockVLPR configuration.
 */
extern const mcg_config_t mcgConfig_BOARD_BootClockVLPR;
/*! @brief SIM module set for BOARD_BootClockVLPR configuration.
 */
extern const sim_clock_config_t simConfig_BOARD_BootClockVLPR;
/*! @brief OSC set for BOARD_BootClockVLPR configuration.
 */
extern const osc_config_t oscConfig_BOARD_BootClockVLPR;

/*******************************************************************************
 * API for BOARD_BootClockVLPR configuration
 ******************************************************************************/
#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus*/

/*!
 * @brief This function executes configuration of clocks.
 *
 */
void BOARD_BootClockVLPR(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus*/

#endif /* _CLOCK_CONFIG_H_ */

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/***********************************************************************************************************************
 * This file was generated by the MCUXpresso Config Tools. Any manual edits made to this file
 * will be overwritten if the respective MCUXpresso Config Tools is used to update this file.
 **********************************************************************************************************************/

#ifndef _PIN_MUX_H_
#define _PIN_MUX_H_

/*!
 * @addtogroup pin_mux
 * @{
 */

/***********************************************************************************************************************
 * API
 **********************************************************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif

/*!
```

```
 * @brief Calls initialization functions.
 *
 */
void BOARD_InitBootPins(void);

#define SOPT5_UART0RXSRC_UART_RX 0x00u /*!<@brief UART0 receive data source select: UART0_
#define SOPT5_UART0TXSRC_UART_TX 0x00u /*!<@brief UART0 transmit data source select: UART0_

/*! @name PORTA2 (number 28), J1[4]/D1/UART0_TX
  @{ */
#define BOARD_INITPINS_DEBUG_UART_TX_PORT PORTA /*!<@brief PORT device name: PORTA */
#define BOARD_INITPINS_DEBUG_UART_TX_PIN 2U     /*!<@brief PORTA pin index: 2 */
                                        /* @} */

/*! @name PORTA1 (number 27), J1[2]/D0/UART0_RX
  @{ */
#define BOARD_INITPINS_DEBUG_UART_RX_PORT PORTA /*!<@brief PORT device name: PORTA */
#define BOARD_INITPINS_DEBUG_UART_RX_PIN 1U     /*!<@brief PORTA pin index: 1 */
                                        /* @} */

/*! @name PORTD1 (number 74), J2[12]/D3[3]/D13/LEDRGB_BLUE
  @{ */
#define BOARD_INITPINS_LED_BLUE_GPIO GPIOD /*!<@brief GPIO device name: GPIOD */
#define BOARD_INITPINS_LED_BLUE_PORT PORTD /*!<@brief PORT device name: PORTD */
#define BOARD_INITPINS_LED_BLUE_PIN 1U     /*!<@brief PORTD pin index: 1 */
                                    /* @} */

/*! @name PORTB18 (number 53), D3[1]/LEDRGB_RED
  @{ */
#define BOARD_INITPINS_LED_RED_GPIO GPIOB /*!<@brief GPIO device name: GPIOB */
#define BOARD_INITPINS_LED_RED_PORT PORTB /*!<@brief PORT device name: PORTB */
#define BOARD_INITPINS_LED_RED_PIN 18U    /*!<@brief PORTB pin index: 18 */
                                   /* @} */

/*! @name PORTB19 (number 54), D3[4]/LEDRGB_GREEN
  @{ */
#define BOARD_INITPINS_LED_GREEN_GPIO GPIOB /*!<@brief GPIO device name: GPIOB */
#define BOARD_INITPINS_LED_GREEN_PORT PORTB /*!<@brief PORT device name: PORTB */
#define BOARD_INITPINS_LED_GREEN_PIN 19U    /*!<@brief PORTB pin index: 19 */
                                     /* @} */

/*!
 * @brief Configures pin routing and optionally pin electrical features.
 *
 */
void BOARD_InitPins(void);

#if defined(__cplusplus)
}
#endif

/*!
 * @}
 */
#endif /* _PIN_MUX_H_ */

/*******************************************************************************************************************
 * EOF
 *******************************************************************************************************************/
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2015, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 *
 * 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 the copyright holder 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" AN
 * 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 FO
 * 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 _BOARD_H_
#define _BOARD_H_

#include "clock_config.h"
#include "fsl_gpio.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/* The board name */
#define BOARD_NAME "FRDM-KL25Z"

/* The LPSCI to use for debug messages. */
#define BOARD_DEBUG_UART_TYPE DEBUG_CONSOLE_DEVICE_TYPE_LPSCI
#define BOARD_DEBUG_UART_BASEADDR (uint32_t) UART0
```

```
#define BOARD_DEBUG_UART_CLKSRC kCLOCK_PllFllSelClk
#define BOARD_DEBUG_UART_CLK_FREQ CLOCK_GetPllFllSelClkFreq()
#define BOARD_UART_IRQ UART0_IRQn
#define BOARD_UART_IRQ_HANDLER UART0_IRQHandler

#ifndef BOARD_DEBUG_UART_BAUDRATE
#define BOARD_DEBUG_UART_BAUDRATE 115200
#endif /* BOARD_DEBUG_UART_BAUDRATE */

/*! @brief Indexes of the TSI channels for on board electrodes */
#define BOARD_TSI_ELECTRODE_1 9U
#define BOARD_TSI_ELECTRODE_2 10U

/* Board led color mapping */
#define LOGIC_LED_ON 0U
#define LOGIC_LED_OFF 1U
#define BOARD_LED_RED_GPIO GPIOB
#define BOARD_LED_RED_GPIO_PORT PORTB
#define BOARD_LED_RED_GPIO_PIN 18U
#define BOARD_LED_GREEN_GPIO GPIOB
#define BOARD_LED_GREEN_GPIO_PORT PORTB
#define BOARD_LED_GREEN_GPIO_PIN 19U
#define BOARD_LED_BLUE_GPIO GPIOD
#define BOARD_LED_BLUE_GPIO_PORT PORTD
#define BOARD_LED_BLUE_GPIO_PIN 1U

#define LED_RED_INIT(output)                                              \
    GPIO_WritePinOutput(BOARD_LED_RED_GPIO, BOARD_LED_RED_GPIO_PIN, output); \
    BOARD_LED_RED_GPIO->PDDR |= (1U << BOARD_LED_RED_GPIO_PIN) /*!< Enable target LED_R
#define LED_RED_ON() \
    GPIO_ClearPinsOutput(BOARD_LED_RED_GPIO, 1U << BOARD_LED_RED_GPIO_PIN) /*!< Turn on
#define LED_RED_OFF() \
    GPIO_SetPinsOutput(BOARD_LED_RED_GPIO, 1U << BOARD_LED_RED_GPIO_PIN) /*!< Turn off ta
#define LED_RED_TOGGLE() \
    GPIO_TogglePinsOutput(BOARD_LED_RED_GPIO, 1U << BOARD_LED_RED_GPIO_PIN) /*!< Toggle

#define LED_GREEN_INIT(output)                                              \
    GPIO_WritePinOutput(BOARD_LED_GREEN_GPIO, BOARD_LED_GREEN_GPIO_PIN, output); \
    BOARD_LED_GREEN_GPIO->PDDR |= (1U << BOARD_LED_GREEN_GPIO_PIN) /*!< Enable target I
#define LED_GREEN_ON() \
    GPIO_ClearPinsOutput(BOARD_LED_GREEN_GPIO, 1U << BOARD_LED_GREEN_GPIO_PIN) /*!< T
#define LED_GREEN_OFF() \
    GPIO_SetPinsOutput(BOARD_LED_GREEN_GPIO, 1U << BOARD_LED_GREEN_GPIO_PIN) /*!< Tur
#define LED_GREEN_TOGGLE() \
    GPIO_TogglePinsOutput(BOARD_LED_GREEN_GPIO, 1U << BOARD_LED_GREEN_GPIO_PIN) /*!<

#define LED_BLUE_INIT(output)                                              \
    GPIO_WritePinOutput(BOARD_LED_BLUE_GPIO, BOARD_LED_BLUE_GPIO_PIN, output); \
    BOARD_LED_BLUE_GPIO->PDDR |= (1U << BOARD_LED_BLUE_GPIO_PIN) /*!< Enable target LED_
#define LED_BLUE_ON() \
    GPIO_ClearPinsOutput(BOARD_LED_BLUE_GPIO, 1U << BOARD_LED_BLUE_GPIO_PIN) /*!< Turn o
#define LED_BLUE_OFF() \
    GPIO_SetPinsOutput(BOARD_LED_BLUE_GPIO, 1U << BOARD_LED_BLUE_GPIO_PIN) /*!< Turn off
```

```
#define LED_BLUE_TOGGLE() \
    GPIO_TogglePinsOutput(BOARD_LED_BLUE_GPIO, 1U << BOARD_LED_BLUE_GPIO_PIN) /*!< Tog

#define BOARD_ACCEL_I2C_BASEADDR I2C0

/* ERPC SPI configuration */
#define ERPC_BOARD_SPI_BASEADDR SPI0
#define ERPC_BOARD_SPI_BAUDRATE 500000U
#define ERPC_BOARD_SPI_CLKSRC SPI0_CLK_SRC
#define ERPC_BOARD_SPI_CLK_FREQ CLOCK_GetFreq(SPI0_CLK_SRC)
#define ERPC_BOARD_SPI_INT_GPIO GPIOD
#define ERPC_BOARD_SPI_INT_PORT PORTD
#define ERPC_BOARD_SPI_INT_PIN 0U
#define ERPC_BOARD_SPI_INT_PIN_IRQ PORTD_IRQn
#define ERPC_BOARD_SPI_INT_PIN_IRQ_HANDLER PORTD_IRQHandler

/* DAC base address */
#define BOARD_DAC_BASEADDR DAC0

/* Board accelerometer driver */
#define BOARD_ACCEL_MMA

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*******************************************************************************
 * API
 ******************************************************************************/

void BOARD_InitDebugConsole(void);

#if defined(__cplusplus)
}
#endif /* __cplusplus */

#endif /* _BOARD_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file delay.h
 * @brief Project 2
 *
 * @details This file contains prototypes for calculating a spin-wait
 *         on various platforms, used for delaying LED state changes.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *        PC Linker: GNU ld 2.32
 *        PC Debugger: GNU gdb 8.2.91.20190405-git
 *        ARM Compiler: GNU gcc version 8.2.1 20181213
 *        ARM Linker: GNU ld 2.31.51.20181213
 *        ARM Debugger: GNU gdb 8.2.50.20181213-git
 */
```

```
#ifndef PES_PROJECT_2_DELAY_H
#define PES_PROJECT_2_DELAY_H

#include <stdint.h>

/**
 * delay
 *
 * @brief Blocks execution for the specified time.
 * @param inDelayMs Then time in milliseconds to block.
 */
void delay(uint64_t inDelayMs);

#endif //PES_PROJECT_2_DELAY_H
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file led_types.h
 * @brief Project 2
 *
 * @details Defines enumerations and constants used to describe colors and
 *          on/off states for LEDs.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */

#ifndef PES_PROJECT_2_LED_TYPES_H
#define PES_PROJECT_2_LED_TYPES_H

/**
 * COLOR
 *
 * @brief The possible color values of the LED.
 */
enum COLOR
{
    RED = 0,
    GREEN,
    BLUE,
    NUM_COLORS
};

/**
 * COLOR_STRINGS
 *
 * @brief String representations of the COLOR enum, used for printing.
 */
static const char * const COLOR_STRINGS[3] = {
```

```
        "RED",
        "GREEN",
        "BLUE"
};

#endif //PES_PROJECT_2_LED_TYPES_H
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file handle_led.h
 * @brief Project 2
 *
 * @details Contains the prototype for handling LEDs on various platforms.
 *          This may be actually turning an LED on and off or just printing
 *          what the LED state would be, in the absence of LEDs.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
 *         PC Debugger: GNU gdb 8.2.91.20190405-git
 *         ARM Compiler: GNU gcc version 8.2.1 20181213
 *         ARM Linker: GNU ld 2.31.51.20181213
 *         ARM Debugger: GNU gdb 8.2.50.20181213-git
 */
#ifndef PES_PROJECT_2_HANDLE_LED_H
#define PES_PROJECT_2_HANDLE_LED_H

#include <stdint.h>
#include "led_types.h"

/**
 * set_led
 *
 * @brief Sets the LED state.
 * @details This function, depending on platform, may or may not
 *          control a physical LED. On PC, it will simply print the
 *          state of what the LED would be.
 * @param inValue The on/off state of the LED to set.
 * @param inColor The color of the LED to set.
 */
void set_led(uint8_t inValue, enum COLOR inColor);

#endif //PES_PROJECT_2_HANDLE_LED_H
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * @file setup_teardown.h
 * @brief Project 2
 *
 * @details Contains the setup and cleanup prototypes to be implemented
 *          both for the FB and PC variants of the build.
 *
 * @author Jack Campbell
 * @tools  PC Compiler: GNU gcc 8.3.0
 *         PC Linker: GNU ld 2.32
```

```
 *        PC Debugger: GNU gdb 8.2.91.20190405-git
 *        ARM Compiler: GNU gcc version 8.2.1 20181213
 *        ARM Linker: GNU ld 2.31.51.20181213
 *        ARM Debugger: GNU gdb 8.2.50.20181213-git
 */

#ifndef PES_PROJECT_2_SETUP_TEARDOWN_H
#define PES_PROJECT_2_SETUP_TEARDOWN_H

/**
 * initialize
 *
 * @details Initializes components needed by a particular platform,
 *         such as LEDs.
 *
 */
void initialize(void);


/**
 * terminate
 *
 * @details Cleans up any required components on a particular platform.
 *
 */
void terminate(void);

#endif //PES_PROJECT_2_SETUP_TEARDOWN_H
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     core_cmSimd.h
 * @brief    CMSIS Cortex-M SIMD Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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
```

```
#if   defined ( __ICCARM__ )
 #pragma system_include         /* treat file as system include file for MISRA check */
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #pragma clang system_header   /* treat file as system include file */
#endif

#ifndef __CORE_CMSIMD_H
#define __CORE_CMSIMD_H

#ifdef __cplusplus
 extern "C" {
#endif


/* ###################  Compiler specific Intrinsics  ########################### */
/** \defgroup CMSIS_SIMD_intrinsics CMSIS SIMD Intrinsics
  Access to dedicated SIMD instructions
  @{
*/

/*------------------ RealView Compiler -----------------*/
#if   defined ( __CC_ARM )
  #include "cmsis_armcc.h"

/*------------------ ARM Compiler V6 -------------------*/
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #include "cmsis_armcc_V6.h"

/*------------------ GNU Compiler ----------------------*/
#elif defined ( __GNUC__ )
  #include "cmsis_gcc.h"

/*------------------ ICC Compiler ----------------------*/
#elif defined ( __ICCARM__ )
  #include <cmsis_iar.h>

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

/*------------------ TASKING Compiler ------------------*/
#elif defined ( __TASKING__ )
  /*
```

```
   * 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.
   */

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

#endif

/*@} end of group CMSIS_SIMD_intrinsics */


#ifdef __cplusplus
}
#endif

#endif /* __CORE_CMSIMD_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     core_cmFunc.h
 * @brief    CMSIS Cortex-M Core Function Access Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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 */
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #pragma clang system_header   /* treat file as system include file */
#endif

#ifndef __CORE_CMFUNC_H
#define __CORE_CMFUNC_H


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

/*------------------ RealView Compiler -----------------*/
#if   defined ( __CC_ARM )
  #include "cmsis_armcc.h"

/*------------------ ARM Compiler V6 -------------------*/
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #include "cmsis_armcc_V6.h"

/*------------------ GNU Compiler ----------------------*/
#elif defined ( __GNUC__ )
  #include "cmsis_gcc.h"

/*------------------ ICC Compiler ----------------------*/
#elif defined ( __ICCARM__ )
  #include <cmsis_iar.h>

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

/*------------------ TASKING Compiler ------------------*/
#elif defined ( __TASKING__ )
  /*
   * 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.
   */

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

#endif

/*@} end of CMSIS_Core_RegAccFunctions */
```

```
#endif /* __CORE_CMFUNC_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     cmsis_armcc.h
 * @brief    CMSIS Cortex-M Core Function/Instruction Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
```

```


#ifndef __CMSIS_ARMCC_H
#define __CMSIS_ARMCC_H


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

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

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

```
/**
  \brief   Get Control Register
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U)

/**
  \brief   Enable FIQ
  \details 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
  \details 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
  \details 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
  \details 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 & 0xFFU);
}


/**
  \brief   Set Base Priority with condition
  \details 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 & 0xFFU);
}


/**
  \brief   Get Fault Mask
  \details 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
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U) */
```

```
#if       (__CORTEX_M == 0x04U) || (__CORTEX_M == 0x07U)

/**
  \brief   Get FPSCR
  \details 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 == 1U) && (__FPU_USED == 1U)
  register uint32_t __regfpscr         __ASM("fpscr");
  return(__regfpscr);
#else
   return(0U);
#endif
}


/**
  \brief   Set FPSCR
  \details 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 == 1U) && (__FPU_USED == 1U)
  register uint32_t __regfpscr         __ASM("fpscr");
  __regfpscr = (fpscr);
#endif
}

#endif /* (__CORTEX_M == 0x04U) || (__CORTEX_M == 0x07U) */



/*@} end of CMSIS_Core_RegAccFunctions */


/* ##########################  Core Instruction Access  ######################### */
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface
  Access to dedicated instructions
  @{
*/

/**
  \brief   No Operation
  \details No Operation does nothing. This instruction can be used for code alignment purposes.
 */
#define __NOP                             __nop


/**
```

```
  \brief   Wait For Interrupt
  \details Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occ
 */
#define __WFI                       __wfi


/**
  \brief   Wait For Event
  \details 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
  \details Send Event is a hint instruction. It causes an event to be signaled to the CPU.
 */
#define __SEV                       __sev


/**
  \brief   Instruction Synchronization Barrier
  \details 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 (0U)

/**
  \brief   Data Synchronization Barrier
  \details 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 (0U)

/**
  \brief   Data Memory Barrier
  \details 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 (0U)

/**
  \brief   Reverse byte order (32 bit)
  \details 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)
  \details 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
  \details 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)
  \details Rotate Right (immediate) provides the value of the contents of a register rotated by a variable nur
  \param [in]    value  Value to rotate
  \param [in]    value  Number of Bits to rotate
  \return               Rotated value
 */
#define __ROR                             __ror


/**
  \brief   Breakpoint
  \details Causes the processor to enter Debug state.
```

```
           Debug tools can use this to investigate system state when the instruction at a particular address is r
  \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
  \details Reverses the bit order of the given value.
  \param [in]    value  Value to reverse
  \return               Reversed value
 */
#if       (__CORTEX_M >= 0x03U) || (__CORTEX_SC >= 300U)
  #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 >>= 1U; value; value >>= 1U)
  {
    result <<= 1U;
    result |= value & 1U;
    s--;
  }
  result <<= s;                        /* shift when v's highest bits are zero */
  return(result);
}
#endif


/**
  \brief   Count leading zeros
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U)

/**
  \brief   LDR Exclusive (8 bit)
  \details Executes a exclusive LDR instruction for 8 bit value.
  \param [in]    ptr  Pointer to data
  \return             value of type uint8_t at (*ptr)
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __LDREXB(ptr)                                                     ((uint8_t ) __ldrex(ptr))
```

```
#else
  #define __LDREXB(ptr)          _Pragma("push") _Pragma("diag_suppress 3731") ((uint8_t ) __ldrex(ptr))
#endif


/**
  \brief   LDR Exclusive (16 bit)
  \details Executes a exclusive LDR instruction for 16 bit values.
  \param [in]    ptr  Pointer to data
  \return        value of type uint16_t at (*ptr)
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __LDREXH(ptr)                                                        ((uint16_t) __ldrex(ptr))
#else
  #define __LDREXH(ptr)          _Pragma("push") _Pragma("diag_suppress 3731") ((uint16_t) __ldrex(ptr))
#endif


/**
  \brief   LDR Exclusive (32 bit)
  \details Executes a exclusive LDR instruction for 32 bit values.
  \param [in]    ptr  Pointer to data
  \return        value of type uint32_t at (*ptr)
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __LDREXW(ptr)                                                        ((uint32_t ) __ldrex(ptr))
#else
  #define __LDREXW(ptr)          _Pragma("push") _Pragma("diag_suppress 3731") ((uint32_t ) __ldrex(ptr)
#endif


/**
  \brief   STR Exclusive (8 bit)
  \details 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
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __STREXB(value, ptr)                                                 __strex(value, ptr)
#else
  #define __STREXB(value, ptr)   _Pragma("push") _Pragma("diag_suppress 3731") __strex(value, ptr)
#endif


/**
  \brief   STR Exclusive (16 bit)
  \details 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
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __STREXH(value, ptr)                                              __strex(value, ptr)
#else
  #define __STREXH(value, ptr)   _Pragma("push") _Pragma("diag_suppress 3731") __strex(value, ptr)
#endif


/**
  \brief   STR Exclusive (32 bit)
  \details 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
 */
#if defined(__ARMCC_VERSION) && (__ARMCC_VERSION < 5060020)
  #define __STREXW(value, ptr)                                              __strex(value, ptr)
#else
  #define __STREXW(value, ptr)   _Pragma("push") _Pragma("diag_suppress 3731") __strex(value, ptr)
#endif


/**
  \brief   Remove the exclusive lock
  \details Removes the exclusive lock which is created by LDREX.
 */
#define __CLREX                           __clrex


/**
  \brief   Signed Saturate
  \details 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
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U) */

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */


/* ###################  Compiler specific Intrinsics  ########################### */
/** \defgroup CMSIS_SIMD_intrinsics CMSIS SIMD Intrinsics
  Access to dedicated SIMD instructions
  @{
*/

#if (__CORTEX_M >= 0x04U)  /* only for Cortex-M4 and above */

#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 __SMLSLD                    __smlsld
#define __SMLSLDX                   __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) << 32U)     ) >> 32U))

#endif /* (__CORTEX_M >= 0x04) */
/*@} end of group CMSIS_SIMD_intrinsics */


#endif /* __CMSIS_ARMCC_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
** ###################################################################
**     Processors:          MKL25Z128VFM4
**                          MKL25Z128VFT4
**                          MKL25Z128VLH4
```

```
**                     MKL25Z128VLK4
**                     MKL25Z32VFM4
**                     MKL25Z32VFT4
**                     MKL25Z32VLH4
**                     MKL25Z32VLK4
**                     MKL25Z64VFM4
**                     MKL25Z64VFT4
**                     MKL25Z64VLH4
**                     MKL25Z64VLK4
**
**     Compilers:          Keil ARM C/C++ Compiler
**                     Freescale C/C++ for Embedded ARM
**                     GNU C Compiler
**                     IAR ANSI C/C++ Compiler for ARM
**                     MCUXpresso Compiler
**
**     Reference manual:    KL25P80M48SF0RM, Rev.3, Sep 2012
**     Version:            rev. 2.5, 2015-02-19
**     Build:              b170112
**
**     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) 2016 Freescale Semiconductor, Inc.
**     Copyright 2016 - 2017 NXP
**     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 the copyright holder 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" 
**     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
**     ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGE
**     (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
**     LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND 
**     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.nxp.com
**     mail:                 support@nxp.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



/* Define clock source values */

#define CPU_XTAL_CLK_HZ               8000000u           /* Value of the external crystal or oscillator clock f
#define CPU_INT_SLOW_CLK_HZ           32768u             /* Value of the slow internal oscillator clock fre
#define CPU_INT_FAST_CLK_HZ           4000000u           /* Value of the fast internal oscillator clock freq

/* RTC oscillator setting */

/* Low power mode enable */
/* SMC_PMPROT: AVLP=1,ALLS=1,AVLLS=1 */
#define SYSTEM_SMC_PMPROT_VALUE       0x2AU              /* SMC_PMPROT */

#define DEFAULT_SYSTEM_CLOCK          20971520u          /* Default System clock value */


/**
 * @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  /* _SYSTEM_MKL25Z4_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* ----------------------------------------------------------------------
* Copyright (C) 2010-2014 ARM Limited. All rights reserved.
*
* $Date:        19. March 2015
* $Revision:  V.1.4.5
*
* Project:      CMSIS DSP Library
* Title:      arm_const_structs.h
*
* Description: This file has constant structs that are initialized for
*              user convenience.  For example, some can be given as
*              arguments to the arm_cfft_f32() function.
*
* Target Processor: Cortex-M4/Cortex-M3
*
* 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 LIMITED 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 OWNER 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.
* -------------------------------------------------------------------- */

#ifndef _ARM_CONST_STRUCTS_H
#define _ARM_CONST_STRUCTS_H

#include "arm_math.h"
```

```
#include "arm_common_tables.h"

   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len16;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len32;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len64;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len128;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len256;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len512;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len1024;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len2048;
   extern const arm_cfft_instance_f32 arm_cfft_sR_f32_len4096;

   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len16;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len32;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len64;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len128;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len256;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len512;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len1024;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len2048;
   extern const arm_cfft_instance_q31 arm_cfft_sR_q31_len4096;

   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len16;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len32;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len64;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len128;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len256;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len512;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len1024;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len2048;
   extern const arm_cfft_instance_q15 arm_cfft_sR_q15_len4096;

#endif
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     core_cmInstr.h
 * @brief    CMSIS Cortex-M Core Instruction Access Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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.
```

```
#if   defined ( __ICCARM__ )
 #pragma system_include         /* treat file as system include file for MISRA check */
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #pragma clang system_header   /* treat file as system include file */
#endif

#ifndef __CORE_CMINSTR_H
#define __CORE_CMINSTR_H


/* ##########################  Core Instruction Access  ######################### */
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface
  Access to dedicated instructions
  @{
*/

/*------------------ RealView Compiler -----------------*/
#if   defined ( __CC_ARM )
  #include "cmsis_armcc.h"

/*------------------ ARM Compiler V6 -------------------*/
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #include "cmsis_armcc_V6.h"

/*------------------ GNU Compiler ----------------------*/
#elif defined ( __GNUC__ )
  #include "cmsis_gcc.h"

/*------------------ ICC Compiler ----------------------*/
#elif defined ( __ICCARM__ )
  #include <cmsis_iar.h>

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

/*------------------ TASKING Compiler ------------------*/
#elif defined ( __TASKING__ )
```

```
  /*
   * 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.
   */

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

#endif

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */

#endif /* __CORE_CMINSTR_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
** ###################################################################
**     Processors:          MKL25Z128VFM4
**                          MKL25Z128VFT4
**                          MKL25Z128VLH4
**                          MKL25Z128VLK4
**                          MKL25Z32VFM4
**                          MKL25Z32VFT4
**                          MKL25Z32VLH4
**                          MKL25Z32VLK4
**                          MKL25Z64VFM4
**                          MKL25Z64VFT4
**                          MKL25Z64VLH4
**                          MKL25Z64VLK4
**
**     Compilers:           Keil ARM C/C++ Compiler
**                          Freescale C/C++ for Embedded ARM
**                          GNU C Compiler
**                          IAR ANSI C/C++ Compiler for ARM
**                          MCUXpresso Compiler
**
**     Reference manual:    KL25P80M48SF0RM, Rev.3, Sep 2012
**     Version:             rev. 2.5, 2015-02-19
**     Build:               b170112
**
**     Abstract:
**         CMSIS Peripheral Access Layer for MKL25Z4
**
**     Copyright (c) 1997 - 2016 Freescale Semiconductor, Inc.
**     Copyright 2016 - 2017 NXP
**     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 the copyright holder 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" A
**   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
**   ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGE
**   (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
**   LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND O
**   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.nxp.com
**   mail:                 support@nxp.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
 */

#ifndef _MKL25Z4_H_
#define _MKL25Z4_H_                          /**< Symbol preventing repeated inclusion */

/** 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 {
  /* Auxiliary constants */
  NotAvail_IRQn                = -128,             /**< Not available device specific interrupt */

  /* 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 VTOR 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 defi

#include "core_cm0plus.h"              /* Core Peripheral Access Layer */
#include "system_MKL25Z4.h"            /* Device specific configuration file */

/*!
 * @}
 */ /* end of group Cortex_Core_Configuration */
```

```
/* ----------------------------------------------------------------------------
   -- Mapping Information
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup Mapping_Information Mapping Information
 * @{
 */

/** Mapping Information */
/*!
 * @addtogroup edma_request
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*!
 * @brief Structure for the DMA hardware request
 *
 * Defines the structure for the DMA hardware request collections. The user can configure the
 * hardware request into DMAMUX to trigger the DMA transfer accordingly. The index
 * of the hardware request varies according  to the to SoC.
 */
typedef enum _dma_request_source
{
    kDmaRequestMux0Disable          = 0|0x100U,    /**< DMAMUX TriggerDisabled. */
    kDmaRequestMux0Reserved1        = 1|0x100U,    /**< Reserved1 */
    kDmaRequestMux0UART0Rx          = 2|0x100U,    /**< UART0 Receive. */
    kDmaRequestMux0LPSCI0Rx         = 2|0x100U,    /**< UART0 Receive. */
    kDmaRequestMux0UART0Tx          = 3|0x100U,    /**< UART0 Transmit. */
    kDmaRequestMux0LPSCI0Tx         = 3|0x100U,    /**< UART0 Transmit. */
    kDmaRequestMux0UART1Rx          = 4|0x100U,    /**< UART1 Receive. */
    kDmaRequestMux0UART1Tx          = 5|0x100U,    /**< UART1 Transmit. */
    kDmaRequestMux0UART2Rx          = 6|0x100U,    /**< UART2 Receive. */
    kDmaRequestMux0UART2Tx          = 7|0x100U,    /**< UART2 Transmit. */
    kDmaRequestMux0Reserved8        = 8|0x100U,    /**< Reserved8 */
    kDmaRequestMux0Reserved9        = 9|0x100U,    /**< Reserved9 */
    kDmaRequestMux0Reserved10       = 10|0x100U,   /**< Reserved10 */
    kDmaRequestMux0Reserved11       = 11|0x100U,   /**< Reserved11 */
    kDmaRequestMux0Reserved12       = 12|0x100U,   /**< Reserved12 */
    kDmaRequestMux0Reserved13       = 13|0x100U,   /**< Reserved13 */
    kDmaRequestMux0Reserved14       = 14|0x100U,   /**< Reserved14 */
    kDmaRequestMux0Reserved15       = 15|0x100U,   /**< Reserved15 */
    kDmaRequestMux0SPI0Rx           = 16|0x100U,   /**< SPI0 Receive. */
    kDmaRequestMux0SPI0Tx           = 17|0x100U,   /**< SPI0 Transmit. */
    kDmaRequestMux0SPI1Rx           = 18|0x100U,   /**< SPI1 Receive. */
    kDmaRequestMux0SPI1Tx           = 19|0x100U,   /**< SPI1 Transmit. */
    kDmaRequestMux0Reserved20       = 20|0x100U,   /**< Reserved20 */
    kDmaRequestMux0Reserved21       = 21|0x100U,   /**< Reserved21 */
    kDmaRequestMux0I2C0             = 22|0x100U,   /**< I2C0. */
    kDmaRequestMux0I2C1             = 23|0x100U,   /**< I2C1. */
    kDmaRequestMux0TPM0Channel0     = 24|0x100U,   /**< TPM0 C0V. */
    kDmaRequestMux0TPM0Channel1     = 25|0x100U,   /**< TPM0 C1V. */
    kDmaRequestMux0TPM0Channel2     = 26|0x100U,   /**< TPM0 C2V. */
    kDmaRequestMux0TPM0Channel3     = 27|0x100U,   /**< TPM0 C3V. */
    kDmaRequestMux0TPM0Channel4     = 28|0x100U,   /**< TPM0 C4V. */
    kDmaRequestMux0TPM0Channel5     = 29|0x100U,   /**< TPM0 C5V. */
    kDmaRequestMux0Reserved30       = 30|0x100U,   /**< Reserved30 */
    kDmaRequestMux0Reserved31       = 31|0x100U,   /**< Reserved31 */
    kDmaRequestMux0TPM1Channel0     = 32|0x100U,   /**< TPM1 C0V. */
    kDmaRequestMux0TPM1Channel1     = 33|0x100U,   /**< TPM1 C1V. */
    kDmaRequestMux0TPM2Channel0     = 34|0x100U,   /**< TPM2 C0V. */
    kDmaRequestMux0TPM2Channel1     = 35|0x100U,   /**< TPM2 C1V. */
    kDmaRequestMux0Reserved36       = 36|0x100U,   /**< Reserved36 */
    kDmaRequestMux0Reserved37       = 37|0x100U,   /**< Reserved37 */
    kDmaRequestMux0Reserved38       = 38|0x100U,   /**< Reserved38 */
    kDmaRequestMux0Reserved39       = 39|0x100U,   /**< Reserved39 */
    kDmaRequestMux0ADC0             = 40|0x100U,   /**< ADC0. */
    kDmaRequestMux0Reserved41       = 41|0x100U,   /**< Reserved41 */
    kDmaRequestMux0CMP0             = 42|0x100U,   /**< CMP0. */
    kDmaRequestMux0Reserved43       = 43|0x100U,   /**< Reserved43 */
    kDmaRequestMux0Reserved44       = 44|0x100U,   /**< Reserved44 */
    kDmaRequestMux0DAC0             = 45|0x100U,   /**< DAC0. */
    kDmaRequestMux0Reserved46       = 46|0x100U,   /**< Reserved46 */
    kDmaRequestMux0Reserved47       = 47|0x100U,   /**< Reserved47 */
    kDmaRequestMux0Reserved48       = 48|0x100U,   /**< Reserved48 */
    kDmaRequestMux0PortA            = 49|0x100U,   /**< PTA. */
    kDmaRequestMux0Reserved50       = 50|0x100U,   /**< Reserved50 */
    kDmaRequestMux0Reserved51       = 51|0x100U,   /**< Reserved51 */
    kDmaRequestMux0PortD            = 52|0x100U,   /**< PTD. */
    kDmaRequestMux0Reserved53       = 53|0x100U,   /**< Reserved53 */
    kDmaRequestMux0TPM0Overflow     = 54|0x100U,   /**< TPM0. */
    kDmaRequestMux0TPM1Overflow     = 55|0x100U,   /**< TPM1. */
    kDmaRequestMux0TPM2Overflow     = 56|0x100U,   /**< TPM2. */
    kDmaRequestMux0TSI0             = 57|0x100U,   /**< TSI0. */
    kDmaRequestMux0Reserved58       = 58|0x100U,   /**< Reserved58 */
    kDmaRequestMux0Reserved59       = 59|0x100U,   /**< Reserved59 */
    kDmaRequestMux0AlwaysOn60       = 60|0x100U,   /**< DMAMUX Always Enabled slot. */
    kDmaRequestMux0AlwaysOn61       = 61|0x100U,   /**< DMAMUX Always Enabled slot. */
    kDmaRequestMux0AlwaysOn62       = 62|0x100U,   /**< DMAMUX Always Enabled slot. */
    kDmaRequestMux0AlwaysOn63       = 63|0x100U,   /**< DMAMUX Always Enabled slot. */
} dma_request_source_t;

/* @} */


/*!
 * @}
 */ /* end of group Mapping_Information */


/* ----------------------------------------------------------------------------
```

```
   -- 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 s
  __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: 0x
  __IO uint32_t CLPS;                            /**< ADC Plus-Side General Calibration Value Register, offset: 0x
  __IO uint32_t CLP4;                            /**< ADC Plus-Side General Calibration Value Register, offset: 0x3
  __IO uint32_t CLP3;                            /**< ADC Plus-Side General Calibration Value Register, offset: 0x4
  __IO uint32_t CLP2;                        /**< ADC Plus-Side General Calibration Value Register, offset: 0x4
  __IO uint32_t CLP1;                        /**< ADC Plus-Side General Calibration Value Register, offset: 0x4
  __IO uint32_t CLP0;                        /**< ADC Plus-Side General Calibration Value Register, offset: 0x4
       uint8_t RESERVED_0[4];
  __IO uint32_t CLMD;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLMS;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLM4;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLM3;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLM2;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLM1;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
  __IO uint32_t CLM0;                        /**< ADC Minus-Side General Calibration Value Register, offset: 0
} ADC_Type;

/* ----------------------------------------------------------------------------
   -- ADC Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup ADC_Register_Masks ADC Register Masks
 * @{
 */

/*! @name SC1 - ADC Status and Control Registers 1 */
#define ADC_SC1_ADCH_MASK                        (0x1FU)
#define ADC_SC1_ADCH_SHIFT                       (0U)
#define ADC_SC1_ADCH(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC1_ADCH_SHIFT)) & A
#define ADC_SC1_DIFF_MASK                        (0x20U)
#define ADC_SC1_DIFF_SHIFT                       (5U)
#define ADC_SC1_DIFF(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC1_DIFF_SHIFT)) & ADC
#define ADC_SC1_AIEN_MASK                        (0x40U)
#define ADC_SC1_AIEN_SHIFT                       (6U)
#define ADC_SC1_AIEN(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC1_AIEN_SHIFT)) & ADC
#define ADC_SC1_COCO_MASK                        (0x80U)
#define ADC_SC1_COCO_SHIFT                       (7U)
#define ADC_SC1_COCO(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC1_COCO_SHIFT)) & A

/* The count of ADC_SC1 */
#define ADC_SC1_COUNT                            (2U)

/*! @name CFG1 - ADC Configuration Register 1 */
#define ADC_CFG1_ADICLK_MASK                     (0x3U)
#define ADC_CFG1_ADICLK_SHIFT                    (0U)
#define ADC_CFG1_ADICLK(x)                       (((uint32_t)(((uint32_t)(x)) << ADC_CFG1_ADICLK_SHIFT))
#define ADC_CFG1_MODE_MASK                       (0xCU)
#define ADC_CFG1_MODE_SHIFT                      (2U)
#define ADC_CFG1_MODE(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_CFG1_MODE_SHIFT)) 
#define ADC_CFG1_ADLSMP_MASK                     (0x10U)
#define ADC_CFG1_ADLSMP_SHIFT                    (4U)
#define ADC_CFG1_ADLSMP(x)                       (((uint32_t)(((uint32_t)(x)) << ADC_CFG1_ADLSMP_SHIF
#define ADC_CFG1_ADIV_MASK                       (0x60U)
#define ADC_CFG1_ADIV_SHIFT                      (5U)
#define ADC_CFG1_ADIV(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_CFG1_ADIV_SHIFT)) & A
#define ADC_CFG1_ADLPC_MASK                      (0x80U)
```

```
#define ADC_CFG1_ADLPC_SHIFT                    (7U)
#define ADC_CFG1_ADLPC(x)                       (((uint32_t)(((uint32_t)(x)) << ADC_CFG1_ADLPC_SHIFT))

/*! @name CFG2 - ADC Configuration Register 2 */
#define ADC_CFG2_ADLSTS_MASK                    (0x3U)
#define ADC_CFG2_ADLSTS_SHIFT                   (0U)
#define ADC_CFG2_ADLSTS(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CFG2_ADLSTS_SHIFT
#define ADC_CFG2_ADHSC_MASK                     (0x4U)
#define ADC_CFG2_ADHSC_SHIFT                    (2U)
#define ADC_CFG2_ADHSC(x)                       (((uint32_t)(((uint32_t)(x)) << ADC_CFG2_ADHSC_SHIFT)
#define ADC_CFG2_ADACKEN_MASK                   (0x8U)
#define ADC_CFG2_ADACKEN_SHIFT                  (3U)
#define ADC_CFG2_ADACKEN(x)                     (((uint32_t)(((uint32_t)(x)) << ADC_CFG2_ADACKEN_SH
#define ADC_CFG2_MUXSEL_MASK                    (0x10U)
#define ADC_CFG2_MUXSEL_SHIFT                   (4U)
#define ADC_CFG2_MUXSEL(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CFG2_MUXSEL_SHIF

/*! @name R - ADC Data Result Register */
#define ADC_R_D_MASK                            (0xFFFFU)
#define ADC_R_D_SHIFT                           (0U)
#define ADC_R_D(x)                              (((uint32_t)(((uint32_t)(x)) << ADC_R_D_SHIFT)) & ADC_R_D_M

/* The count of ADC_R */
#define ADC_R_COUNT                             (2U)

/*! @name CV1 - Compare Value Registers */
#define ADC_CV1_CV_MASK                         (0xFFFFU)
#define ADC_CV1_CV_SHIFT                        (0U)
#define ADC_CV1_CV(x)                           (((uint32_t)(((uint32_t)(x)) << ADC_CV1_CV_SHIFT)) & ADC_C

/*! @name CV2 - Compare Value Registers */
#define ADC_CV2_CV_MASK                         (0xFFFFU)
#define ADC_CV2_CV_SHIFT                        (0U)
#define ADC_CV2_CV(x)                           (((uint32_t)(((uint32_t)(x)) << ADC_CV2_CV_SHIFT)) & ADC_C

/*! @name SC2 - Status and Control Register 2 */
#define ADC_SC2_REFSEL_MASK                     (0x3U)
#define ADC_SC2_REFSEL_SHIFT                    (0U)
#define ADC_SC2_REFSEL(x)                       (((uint32_t)(((uint32_t)(x)) << ADC_SC2_REFSEL_SHIFT)) &
#define ADC_SC2_DMAEN_MASK                      (0x4U)
#define ADC_SC2_DMAEN_SHIFT                     (2U)
#define ADC_SC2_DMAEN(x)                        (((uint32_t)(((uint32_t)(x)) << ADC_SC2_DMAEN_SHIFT)) &
#define ADC_SC2_ACREN_MASK                      (0x8U)
#define ADC_SC2_ACREN_SHIFT                     (3U)
#define ADC_SC2_ACREN(x)                        (((uint32_t)(((uint32_t)(x)) << ADC_SC2_ACREN_SHIFT)) &
#define ADC_SC2_ACFGT_MASK                      (0x10U)
#define ADC_SC2_ACFGT_SHIFT                     (4U)
#define ADC_SC2_ACFGT(x)                        (((uint32_t)(((uint32_t)(x)) << ADC_SC2_ACFGT_SHIFT)) &
#define ADC_SC2_ACFE_MASK                       (0x20U)
#define ADC_SC2_ACFE_SHIFT                      (5U)
#define ADC_SC2_ACFE(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_SC2_ACFE_SHIFT)) & AD
#define ADC_SC2_ADTRG_MASK                      (0x40U)
#define ADC_SC2_ADTRG_SHIFT                     (6U)
```

```
#define ADC_SC2_ADTRG(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_SC2_ADTRG_SHIFT)) &
#define ADC_SC2_ADACT_MASK                          (0x80U)
#define ADC_SC2_ADACT_SHIFT                         (7U)
#define ADC_SC2_ADACT(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_SC2_ADACT_SHIFT)) &

/*! @name SC3 - Status and Control Register 3 */
#define ADC_SC3_AVGS_MASK                           (0x3U)
#define ADC_SC3_AVGS_SHIFT                          (0U)
#define ADC_SC3_AVGS(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC3_AVGS_SHIFT)) & AI
#define ADC_SC3_AVGE_MASK                           (0x4U)
#define ADC_SC3_AVGE_SHIFT                          (2U)
#define ADC_SC3_AVGE(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC3_AVGE_SHIFT)) & AI
#define ADC_SC3_ADCO_MASK                           (0x8U)
#define ADC_SC3_ADCO_SHIFT                          (3U)
#define ADC_SC3_ADCO(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC3_ADCO_SHIFT)) & A
#define ADC_SC3_CALF_MASK                           (0x40U)
#define ADC_SC3_CALF_SHIFT                          (6U)
#define ADC_SC3_CALF(x)                          (((uint32_t)(((uint32_t)(x)) << ADC_SC3_CALF_SHIFT)) & AD
#define ADC_SC3_CAL_MASK                            (0x80U)
#define ADC_SC3_CAL_SHIFT                           (7U)
#define ADC_SC3_CAL(x)                           (((uint32_t)(((uint32_t)(x)) << ADC_SC3_CAL_SHIFT)) & ADC_

/*! @name OFS - ADC Offset Correction Register */
#define ADC_OFS_OFS_MASK                            (0xFFFFU)
#define ADC_OFS_OFS_SHIFT                           (0U)
#define ADC_OFS_OFS(x)                           (((uint32_t)(((uint32_t)(x)) << ADC_OFS_OFS_SHIFT)) & ADC

/*! @name PG - ADC Plus-Side Gain Register */
#define ADC_PG_PG_MASK                              (0xFFFFU)
#define ADC_PG_PG_SHIFT                             (0U)
#define ADC_PG_PG(x)                             (((uint32_t)(((uint32_t)(x)) << ADC_PG_PG_SHIFT)) & ADC_PC

/*! @name MG - ADC Minus-Side Gain Register */
#define ADC_MG_MG_MASK                              (0xFFFFU)
#define ADC_MG_MG_SHIFT                             (0U)
#define ADC_MG_MG(x)                             (((uint32_t)(((uint32_t)(x)) << ADC_MG_MG_SHIFT)) & ADC_M

/*! @name CLPD - ADC Plus-Side General Calibration Value Register */
#define ADC_CLPD_CLPD_MASK                          (0x3FU)
#define ADC_CLPD_CLPD_SHIFT                         (0U)
#define ADC_CLPD_CLPD(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_CLPD_CLPD_SHIFT)) &

/*! @name CLPS - ADC Plus-Side General Calibration Value Register */
#define ADC_CLPS_CLPS_MASK                          (0x3FU)
#define ADC_CLPS_CLPS_SHIFT                         (0U)
#define ADC_CLPS_CLPS(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_CLPS_CLPS_SHIFT)) & /

/*! @name CLP4 - ADC Plus-Side General Calibration Value Register */
#define ADC_CLP4_CLP4_MASK                          (0x3FFU)
#define ADC_CLP4_CLP4_SHIFT                         (0U)
#define ADC_CLP4_CLP4(x)                         (((uint32_t)(((uint32_t)(x)) << ADC_CLP4_CLP4_SHIFT)) & A

/*! @name CLP3 - ADC Plus-Side General Calibration Value Register */
```

```
#define ADC_CLP3_CLP3_MASK                    (0x1FFU)
#define ADC_CLP3_CLP3_SHIFT                   (0U)
#define ADC_CLP3_CLP3(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLP3_CLP3_SHIFT)) & A

/*! @name CLP2 - ADC Plus-Side General Calibration Value Register */
#define ADC_CLP2_CLP2_MASK                    (0xFFU)
#define ADC_CLP2_CLP2_SHIFT                   (0U)
#define ADC_CLP2_CLP2(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLP2_CLP2_SHIFT)) & A

/*! @name CLP1 - ADC Plus-Side General Calibration Value Register */
#define ADC_CLP1_CLP1_MASK                    (0x7FU)
#define ADC_CLP1_CLP1_SHIFT                   (0U)
#define ADC_CLP1_CLP1(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLP1_CLP1_SHIFT)) & A

/*! @name CLP0 - ADC Plus-Side General Calibration Value Register */
#define ADC_CLP0_CLP0_MASK                    (0x3FU)
#define ADC_CLP0_CLP0_SHIFT                   (0U)
#define ADC_CLP0_CLP0(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLP0_CLP0_SHIFT)) & A

/*! @name CLMD - ADC Minus-Side General Calibration Value Register */
#define ADC_CLMD_CLMD_MASK                    (0x3FU)
#define ADC_CLMD_CLMD_SHIFT                   (0U)
#define ADC_CLMD_CLMD(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLMD_CLMD_SHIFT)) 

/*! @name CLMS - ADC Minus-Side General Calibration Value Register */
#define ADC_CLMS_CLMS_MASK                    (0x3FU)
#define ADC_CLMS_CLMS_SHIFT                   (0U)
#define ADC_CLMS_CLMS(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLMS_CLMS_SHIFT)) &

/*! @name CLM4 - ADC Minus-Side General Calibration Value Register */
#define ADC_CLM4_CLM4_MASK                    (0x3FFU)
#define ADC_CLM4_CLM4_SHIFT                   (0U)
#define ADC_CLM4_CLM4(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLM4_CLM4_SHIFT)) &

/*! @name CLM3 - ADC Minus-Side General Calibration Value Register */
#define ADC_CLM3_CLM3_MASK                    (0x1FFU)
#define ADC_CLM3_CLM3_SHIFT                   (0U)
#define ADC_CLM3_CLM3(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLM3_CLM3_SHIFT)) &

/*! @name CLM2 - ADC Minus-Side General Calibration Value Register */
#define ADC_CLM2_CLM2_MASK                    (0xFFU)
#define ADC_CLM2_CLM2_SHIFT                   (0U)
#define ADC_CLM2_CLM2(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLM2_CLM2_SHIFT)) &

/*! @name CLM1 - ADC Minus-Side General Calibration Value Register */
#define ADC_CLM1_CLM1_MASK                    (0x7FU)
#define ADC_CLM1_CLM1_SHIFT                   (0U)
#define ADC_CLM1_CLM1(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLM1_CLM1_SHIFT)) &

/*! @name CLM0 - ADC Minus-Side General Calibration Value Register */
#define ADC_CLM0_CLM0_MASK                    (0x3FU)
#define ADC_CLM0_CLM0_SHIFT                   (0U)
#define ADC_CLM0_CLM0(x)                      (((uint32_t)(((uint32_t)(x)) << ADC_CLM0_CLM0_SHIFT)) &
/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the ADC peripheral type */
#define ADC_IRQS                                 { ADC0_IRQn }

/*!
 * @}
 */ /* 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 CR0;                                /**< 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup CMP_Register_Masks CMP Register Masks
 * @{
 */

/*! @name CR0 - CMP Control Register 0 */
```

```
#define CMP_CR0_HYSTCTR_MASK                    (0x3U)
#define CMP_CR0_HYSTCTR_SHIFT                   (0U)
#define CMP_CR0_HYSTCTR(x)                      (((uint8_t)(((uint8_t)(x)) << CMP_CR0_HYSTCTR_SHIFT)
#define CMP_CR0_FILTER_CNT_MASK                 (0x70U)
#define CMP_CR0_FILTER_CNT_SHIFT                (4U)
#define CMP_CR0_FILTER_CNT(x)                   (((uint8_t)(((uint8_t)(x)) << CMP_CR0_FILTER_CNT_SH

/*! @name CR1 - CMP Control Register 1 */
#define CMP_CR1_EN_MASK                         (0x1U)
#define CMP_CR1_EN_SHIFT                        (0U)
#define CMP_CR1_EN(x)                           (((uint8_t)(((uint8_t)(x)) << CMP_CR1_EN_SHIFT)) & CMP_CF
#define CMP_CR1_OPE_MASK                        (0x2U)
#define CMP_CR1_OPE_SHIFT                       (1U)
#define CMP_CR1_OPE(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_CR1_OPE_SHIFT)) & CMP_
#define CMP_CR1_COS_MASK                        (0x4U)
#define CMP_CR1_COS_SHIFT                       (2U)
#define CMP_CR1_COS(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_CR1_COS_SHIFT)) & CMP_
#define CMP_CR1_INV_MASK                        (0x8U)
#define CMP_CR1_INV_SHIFT                       (3U)
#define CMP_CR1_INV(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_CR1_INV_SHIFT)) & CMP_C
#define CMP_CR1_PMODE_MASK                      (0x10U)
#define CMP_CR1_PMODE_SHIFT                     (4U)
#define CMP_CR1_PMODE(x)                        (((uint8_t)(((uint8_t)(x)) << CMP_CR1_PMODE_SHIFT)) &
#define CMP_CR1_TRIGM_MASK                      (0x20U)
#define CMP_CR1_TRIGM_SHIFT                     (5U)
#define CMP_CR1_TRIGM(x)                        (((uint8_t)(((uint8_t)(x)) << CMP_CR1_TRIGM_SHIFT)) & CM
#define CMP_CR1_WE_MASK                         (0x40U)
#define CMP_CR1_WE_SHIFT                        (6U)
#define CMP_CR1_WE(x)                           (((uint8_t)(((uint8_t)(x)) << CMP_CR1_WE_SHIFT)) & CMP_C
#define CMP_CR1_SE_MASK                         (0x80U)
#define CMP_CR1_SE_SHIFT                        (7U)
#define CMP_CR1_SE(x)                           (((uint8_t)(((uint8_t)(x)) << CMP_CR1_SE_SHIFT)) & CMP_CF

/*! @name FPR - CMP Filter Period Register */
#define CMP_FPR_FILT_PER_MASK                   (0xFFU)
#define CMP_FPR_FILT_PER_SHIFT                  (0U)
#define CMP_FPR_FILT_PER(x)                     (((uint8_t)(((uint8_t)(x)) << CMP_FPR_FILT_PER_SHIFT))

/*! @name SCR - CMP Status and Control Register */
#define CMP_SCR_COUT_MASK                       (0x1U)
#define CMP_SCR_COUT_SHIFT                      (0U)
#define CMP_SCR_COUT(x)                         (((uint8_t)(((uint8_t)(x)) << CMP_SCR_COUT_SHIFT)) & CM
#define CMP_SCR_CFF_MASK                        (0x2U)
#define CMP_SCR_CFF_SHIFT                       (1U)
#define CMP_SCR_CFF(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_SCR_CFF_SHIFT)) & CMP_
#define CMP_SCR_CFR_MASK                        (0x4U)
#define CMP_SCR_CFR_SHIFT                       (2U)
#define CMP_SCR_CFR(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_SCR_CFR_SHIFT)) & CMP_
#define CMP_SCR_IEF_MASK                        (0x8U)
#define CMP_SCR_IEF_SHIFT                       (3U)
#define CMP_SCR_IEF(x)                          (((uint8_t)(((uint8_t)(x)) << CMP_SCR_IEF_SHIFT)) & CMP_S
#define CMP_SCR_IER_MASK                        (0x10U)
#define CMP_SCR_IER_SHIFT                       (4U)
```

```
#define CMP_SCR_IER(x)                        (((uint8_t)(((uint8_t)(x)) << CMP_SCR_IER_SHIFT)) & CMP_S
#define CMP_SCR_DMAEN_MASK                    (0x40U)
#define CMP_SCR_DMAEN_SHIFT                   (6U)
#define CMP_SCR_DMAEN(x)                      (((uint8_t)(((uint8_t)(x)) << CMP_SCR_DMAEN_SHIFT)) &

/*! @name DACCR - DAC Control Register */
#define CMP_DACCR_VOSEL_MASK                  (0x3FU)
#define CMP_DACCR_VOSEL_SHIFT                 (0U)
#define CMP_DACCR_VOSEL(x)                    (((uint8_t)(((uint8_t)(x)) << CMP_DACCR_VOSEL_SHIFT)
#define CMP_DACCR_VRSEL_MASK                  (0x40U)
#define CMP_DACCR_VRSEL_SHIFT                 (6U)
#define CMP_DACCR_VRSEL(x)                    (((uint8_t)(((uint8_t)(x)) << CMP_DACCR_VRSEL_SHIFT)
#define CMP_DACCR_DACEN_MASK                  (0x80U)
#define CMP_DACCR_DACEN_SHIFT                 (7U)
#define CMP_DACCR_DACEN(x)                    (((uint8_t)(((uint8_t)(x)) << CMP_DACCR_DACEN_SHIFT

/*! @name MUXCR - MUX Control Register */
#define CMP_MUXCR_MSEL_MASK                   (0x7U)
#define CMP_MUXCR_MSEL_SHIFT                  (0U)
#define CMP_MUXCR_MSEL(x)                     (((uint8_t)(((uint8_t)(x)) << CMP_MUXCR_MSEL_SHIFT))
#define CMP_MUXCR_PSEL_MASK                   (0x38U)
#define CMP_MUXCR_PSEL_SHIFT                  (3U)
#define CMP_MUXCR_PSEL(x)                     (((uint8_t)(((uint8_t)(x)) << CMP_MUXCR_PSEL_SHIFT)) &
#define CMP_MUXCR_PSTM_MASK                   (0x80U)
#define CMP_MUXCR_PSTM_SHIFT                  (7U)
#define CMP_MUXCR_PSTM(x)                     (((uint8_t)(((uint8_t)(x)) << CMP_MUXCR_PSTM_SHIFT))


/*!
 * @}
 */ /* end of group CMP_Register_Masks */


/* CMP - Peripheral instance base addresses */
/** Peripheral CMP0 base address */
#define CMP0_BASE                             (0x40073000u)
/** Peripheral CMP0 base pointer */
#define CMP0                                  ((CMP_Type *)CMP0_BASE)
/** Array initializer of CMP peripheral base addresses */
#define CMP_BASE_ADDRS                        { CMP0_BASE }
/** Array initializer of CMP peripheral base pointers */
#define CMP_BASE_PTRS                         { CMP0 }
/** Interrupt vectors for the CMP peripheral type */
#define CMP_IRQS                              { CMP0_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup DAC_Register_Masks DAC Register Masks
 * @{
 */

/*! @name DATL - DAC Data Low Register */
#define DAC_DATL_DATA0_MASK                      (0xFFU)
#define DAC_DATL_DATA0_SHIFT                     (0U)
#define DAC_DATL_DATA0(x)                        (((uint8_t)(((uint8_t)(x)) << DAC_DATL_DATA0_SHIFT)) & D

/* The count of DAC_DATL */
#define DAC_DATL_COUNT                           (2U)

/*! @name DATH - DAC Data High Register */
#define DAC_DATH_DATA1_MASK                      (0xFU)
#define DAC_DATH_DATA1_SHIFT                     (0U)
#define DAC_DATH_DATA1(x)                        (((uint8_t)(((uint8_t)(x)) << DAC_DATH_DATA1_SHIFT)) &

/* The count of DAC_DATH */
#define DAC_DATH_COUNT                           (2U)

/*! @name SR - DAC Status Register */
#define DAC_SR_DACBFRPBF_MASK                    (0x1U)
#define DAC_SR_DACBFRPBF_SHIFT                   (0U)
#define DAC_SR_DACBFRPBF(x)                      (((uint8_t)(((uint8_t)(x)) << DAC_SR_DACBFRPBF_SHIF
#define DAC_SR_DACBFRPTF_MASK                    (0x2U)
#define DAC_SR_DACBFRPTF_SHIFT                   (1U)
#define DAC_SR_DACBFRPTF(x)                      (((uint8_t)(((uint8_t)(x)) << DAC_SR_DACBFRPTF_SHIF

/*! @name C0 - DAC Control Register */
```

```
#define DAC_C0_DACBBIEN_MASK                    (0x1U)
#define DAC_C0_DACBBIEN_SHIFT                   (0U)
#define DAC_C0_DACBBIEN(x)                      (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACBBIEN_SHIFT)) &
#define DAC_C0_DACBTIEN_MASK                    (0x2U)
#define DAC_C0_DACBTIEN_SHIFT                   (1U)
#define DAC_C0_DACBTIEN(x)                      (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACBTIEN_SHIFT)) &
#define DAC_C0_LPEN_MASK                        (0x8U)
#define DAC_C0_LPEN_SHIFT                       (3U)
#define DAC_C0_LPEN(x)                          (((uint8_t)(((uint8_t)(x)) << DAC_C0_LPEN_SHIFT)) & DAC_C
#define DAC_C0_DACSWTRG_MASK                    (0x10U)
#define DAC_C0_DACSWTRG_SHIFT                   (4U)
#define DAC_C0_DACSWTRG(x)                      (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACSWTRG_SHIFT
#define DAC_C0_DACTRGSEL_MASK                   (0x20U)
#define DAC_C0_DACTRGSEL_SHIFT                  (5U)
#define DAC_C0_DACTRGSEL(x)                     (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACTRGSEL_SHIF
#define DAC_C0_DACRFS_MASK                      (0x40U)
#define DAC_C0_DACRFS_SHIFT                     (6U)
#define DAC_C0_DACRFS(x)                        (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACRFS_SHIFT)) & D
#define DAC_C0_DACEN_MASK                       (0x80U)
#define DAC_C0_DACEN_SHIFT                      (7U)
#define DAC_C0_DACEN(x)                         (((uint8_t)(((uint8_t)(x)) << DAC_C0_DACEN_SHIFT)) & DAC

/*! @name C1 - DAC Control Register 1 */
#define DAC_C1_DACBFEN_MASK                     (0x1U)
#define DAC_C1_DACBFEN_SHIFT                    (0U)
#define DAC_C1_DACBFEN(x)                       (((uint8_t)(((uint8_t)(x)) << DAC_C1_DACBFEN_SHIFT)) &
#define DAC_C1_DACBFMD_MASK                     (0x4U)
#define DAC_C1_DACBFMD_SHIFT                    (2U)
#define DAC_C1_DACBFMD(x)                       (((uint8_t)(((uint8_t)(x)) << DAC_C1_DACBFMD_SHIFT)) &
#define DAC_C1_DMAEN_MASK                       (0x80U)
#define DAC_C1_DMAEN_SHIFT                      (7U)
#define DAC_C1_DMAEN(x)                         (((uint8_t)(((uint8_t)(x)) << DAC_C1_DMAEN_SHIFT)) & DAC

/*! @name C2 - DAC Control Register 2 */
#define DAC_C2_DACBFUP_MASK                     (0x1U)
#define DAC_C2_DACBFUP_SHIFT                    (0U)
#define DAC_C2_DACBFUP(x)                       (((uint8_t)(((uint8_t)(x)) << DAC_C2_DACBFUP_SHIFT)) &
#define DAC_C2_DACBFRP_MASK                     (0x10U)
#define DAC_C2_DACBFRP_SHIFT                    (4U)
#define DAC_C2_DACBFRP(x)                       (((uint8_t)(((uint8_t)(x)) << DAC_C2_DACBFRP_SHIFT)) &


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the DAC peripheral type */
#define DAC_IRQS                                { DAC0_IRQn }

/*!
 * @}
 */ /* 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: 0x
    __IO uint32_t DAR;                               /**< Destination Address Register, array offset: 0x104, array step
    union {                                          /* offset: 0x108, array step: 0x10 */
      __IO uint32_t DSR_BCR;                           /**< DMA Status Register / Byte Count Register, array offset
      struct {                                         /* offset: 0x108, array step: 0x10 */
             uint8_t RESERVED_0[3];
        __IO uint8_t DSR;                                /**< DMA_DSR0 register...DMA_DSR3 register., array offset: 0
      } DMA_DSR_ACCESS8BIT;
    };
    __IO uint32_t DCR;                               /**< DMA Control Register, array offset: 0x10C, array step: 0x10
  } DMA[4];
} DMA_Type;

/* ----------------------------------------------------------------------------
   -- DMA Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup DMA_Register_Masks DMA Register Masks
 * @{
 */

/*! @name SAR - Source Address Register */
#define DMA_SAR_SAR_MASK                         (0xFFFFFFFFU)
#define DMA_SAR_SAR_SHIFT                        (0U)
#define DMA_SAR_SAR(x)                           (((uint32_t)(((uint32_t)(x)) << DMA_SAR_SAR_SHIFT)) & DM

/* The count of DMA_SAR */
#define DMA_SAR_COUNT                            (4U)
```

```
/*! @name DAR - Destination Address Register */
#define DMA_DAR_DAR_MASK                         (0xFFFFFFFFU)
#define DMA_DAR_DAR_SHIFT                        (0U)
#define DMA_DAR_DAR(x)                           (((uint32_t)(((uint32_t)(x)) << DMA_DAR_DAR_SHIFT)) & DM

/* The count of DMA_DAR */
#define DMA_DAR_COUNT                            (4U)

/*! @name DSR_BCR - DMA Status Register / Byte Count Register */
#define DMA_DSR_BCR_BCR_MASK                     (0xFFFFFFU)
#define DMA_DSR_BCR_BCR_SHIFT                    (0U)
#define DMA_DSR_BCR_BCR(x)                       (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_BCR_SHIF
#define DMA_DSR_BCR_DONE_MASK                    (0x1000000U)
#define DMA_DSR_BCR_DONE_SHIFT                   (24U)
#define DMA_DSR_BCR_DONE(x)                      (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_DONE_S
#define DMA_DSR_BCR_BSY_MASK                     (0x2000000U)
#define DMA_DSR_BCR_BSY_SHIFT                    (25U)
#define DMA_DSR_BCR_BSY(x)                       (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_BSY_SHIF
#define DMA_DSR_BCR_REQ_MASK                     (0x4000000U)
#define DMA_DSR_BCR_REQ_SHIFT                    (26U)
#define DMA_DSR_BCR_REQ(x)                       (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_REQ_SHI
#define DMA_DSR_BCR_BED_MASK                     (0x10000000U)
#define DMA_DSR_BCR_BED_SHIFT                    (28U)
#define DMA_DSR_BCR_BED(x)                       (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_BED_SHIF
#define DMA_DSR_BCR_BES_MASK                     (0x20000000U)
#define DMA_DSR_BCR_BES_SHIFT                    (29U)
#define DMA_DSR_BCR_BES(x)                       (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_BES_SHIF
#define DMA_DSR_BCR_CE_MASK                      (0x40000000U)
#define DMA_DSR_BCR_CE_SHIFT                     (30U)
#define DMA_DSR_BCR_CE(x)                        (((uint32_t)(((uint32_t)(x)) << DMA_DSR_BCR_CE_SHIFT)

/* The count of DMA_DSR_BCR */
#define DMA_DSR_BCR_COUNT                        (4U)

/* The count of DMA_DSR */
#define DMA_DSR_COUNT                            (4U)

/*! @name DCR - DMA Control Register */
#define DMA_DCR_LCH2_MASK                        (0x3U)
#define DMA_DCR_LCH2_SHIFT                       (0U)
#define DMA_DCR_LCH2(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_LCH2_SHIFT)) & D
#define DMA_DCR_LCH1_MASK                        (0xCU)
#define DMA_DCR_LCH1_SHIFT                       (2U)
#define DMA_DCR_LCH1(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_LCH1_SHIFT)) & D
#define DMA_DCR_LINKCC_MASK                      (0x30U)
#define DMA_DCR_LINKCC_SHIFT                     (4U)
#define DMA_DCR_LINKCC(x)                        (((uint32_t)(((uint32_t)(x)) << DMA_DCR_LINKCC_SHIFT))
#define DMA_DCR_D_REQ_MASK                       (0x80U)
#define DMA_DCR_D_REQ_SHIFT                      (7U)
#define DMA_DCR_D_REQ(x)                         (((uint32_t)(((uint32_t)(x)) << DMA_DCR_D_REQ_SHIFT))
#define DMA_DCR_DMOD_MASK                        (0xF00U)
#define DMA_DCR_DMOD_SHIFT                       (8U)
```

```
#define DMA_DCR_DMOD(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_DMOD_SHIFT)) &
#define DMA_DCR_SMOD_MASK                        (0xF000U)
#define DMA_DCR_SMOD_SHIFT                       (12U)
#define DMA_DCR_SMOD(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_SMOD_SHIFT)) &
#define DMA_DCR_START_MASK                       (0x10000U)
#define DMA_DCR_START_SHIFT                      (16U)
#define DMA_DCR_START(x)                         (((uint32_t)(((uint32_t)(x)) << DMA_DCR_START_SHIFT)) &
#define DMA_DCR_DSIZE_MASK                       (0x60000U)
#define DMA_DCR_DSIZE_SHIFT                      (17U)
#define DMA_DCR_DSIZE(x)                         (((uint32_t)(((uint32_t)(x)) << DMA_DCR_DSIZE_SHIFT)) &
#define DMA_DCR_DINC_MASK                        (0x80000U)
#define DMA_DCR_DINC_SHIFT                       (19U)
#define DMA_DCR_DINC(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_DINC_SHIFT)) & D
#define DMA_DCR_SSIZE_MASK                       (0x300000U)
#define DMA_DCR_SSIZE_SHIFT                      (20U)
#define DMA_DCR_SSIZE(x)                         (((uint32_t)(((uint32_t)(x)) << DMA_DCR_SSIZE_SHIFT)) &
#define DMA_DCR_SINC_MASK                        (0x400000U)
#define DMA_DCR_SINC_SHIFT                       (22U)
#define DMA_DCR_SINC(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_SINC_SHIFT)) & DM
#define DMA_DCR_EADREQ_MASK                      (0x800000U)
#define DMA_DCR_EADREQ_SHIFT                     (23U)
#define DMA_DCR_EADREQ(x)                        (((uint32_t)(((uint32_t)(x)) << DMA_DCR_EADREQ_SHIFT
#define DMA_DCR_AA_MASK                          (0x10000000U)
#define DMA_DCR_AA_SHIFT                         (28U)
#define DMA_DCR_AA(x)                            (((uint32_t)(((uint32_t)(x)) << DMA_DCR_AA_SHIFT)) & DMA_
#define DMA_DCR_CS_MASK                          (0x20000000U)
#define DMA_DCR_CS_SHIFT                         (29U)
#define DMA_DCR_CS(x)                            (((uint32_t)(((uint32_t)(x)) << DMA_DCR_CS_SHIFT)) & DMA_
#define DMA_DCR_ERQ_MASK                         (0x40000000U)
#define DMA_DCR_ERQ_SHIFT                        (30U)
#define DMA_DCR_ERQ(x)                           (((uint32_t)(((uint32_t)(x)) << DMA_DCR_ERQ_SHIFT)) & DM
#define DMA_DCR_EINT_MASK                        (0x80000000U)
#define DMA_DCR_EINT_SHIFT                       (31U)
#define DMA_DCR_EINT(x)                          (((uint32_t)(((uint32_t)(x)) << DMA_DCR_EINT_SHIFT)) & DM

/* The count of DMA_DCR */
#define DMA_DCR_COUNT                            (4U)


/*!
 * @}
 */ /* 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)
/** 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 }
```

```
/** Interrupt vectors for the DMA peripheral type */
#define DMA_CHN_IRQS                            { { DMA0_IRQn, DMA1_IRQn, DMA2_IRQn, DMA3_IRQn } }

/*!
 * @}
 */ /* 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
} DMAMUX_Type;

/* ----------------------------------------------------------------------------
   -- DMAMUX Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup DMAMUX_Register_Masks DMAMUX Register Masks
 * @{
 */

/*! @name CHCFG - Channel Configuration register */
#define DMAMUX_CHCFG_SOURCE_MASK                 (0x3FU)
#define DMAMUX_CHCFG_SOURCE_SHIFT                (0U)
#define DMAMUX_CHCFG_SOURCE(x)                   (((uint8_t)(((uint8_t)(x)) << DMAMUX_CHCFG_SOUR
#define DMAMUX_CHCFG_TRIG_MASK                   (0x40U)
#define DMAMUX_CHCFG_TRIG_SHIFT                  (6U)
#define DMAMUX_CHCFG_TRIG(x)                     (((uint8_t)(((uint8_t)(x)) << DMAMUX_CHCFG_TRIG_SH
#define DMAMUX_CHCFG_ENBL_MASK                   (0x80U)
#define DMAMUX_CHCFG_ENBL_SHIFT                  (7U)
#define DMAMUX_CHCFG_ENBL(x)                     (((uint8_t)(((uint8_t)(x)) << DMAMUX_CHCFG_ENBL_S

/* The count of DMAMUX_CHCFG */
#define DMAMUX_CHCFG_COUNT                       (4U)


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup FGPIO_Register_Masks FGPIO Register Masks
 * @{
 */

/*! @name PDOR - Port Data Output Register */
#define FGPIO_PDOR_PDO_MASK                      (0xFFFFFFFFU)
#define FGPIO_PDOR_PDO_SHIFT                     (0U)
#define FGPIO_PDOR_PDO(x)                        (((uint32_t)(((uint32_t)(x)) << FGPIO_PDOR_PDO_SHIFT))

/*! @name PSOR - Port Set Output Register */
#define FGPIO_PSOR_PTSO_MASK                     (0xFFFFFFFFU)
#define FGPIO_PSOR_PTSO_SHIFT                    (0U)
#define FGPIO_PSOR_PTSO(x)                       (((uint32_t)(((uint32_t)(x)) << FGPIO_PSOR_PTSO_SHIFT

/*! @name PCOR - Port Clear Output Register */
#define FGPIO_PCOR_PTCO_MASK                     (0xFFFFFFFFU)
```

```
#define FGPIO_PCOR_PTCO_SHIFT                    (0U)
#define FGPIO_PCOR_PTCO(x)                       (((uint32_t)(((uint32_t)(x)) << FGPIO_PCOR_PTCO_SHIFT

/*! @name PTOR - Port Toggle Output Register */
#define FGPIO_PTOR_PTTO_MASK                     (0xFFFFFFFFU)
#define FGPIO_PTOR_PTTO_SHIFT                    (0U)
#define FGPIO_PTOR_PTTO(x)                       (((uint32_t)(((uint32_t)(x)) << FGPIO_PTOR_PTTO_SHIFT)

/*! @name PDIR - Port Data Input Register */
#define FGPIO_PDIR_PDI_MASK                      (0xFFFFFFFFU)
#define FGPIO_PDIR_PDI_SHIFT                     (0U)
#define FGPIO_PDIR_PDI(x)                        (((uint32_t)(((uint32_t)(x)) << FGPIO_PDIR_PDI_SHIFT)) & F

/*! @name PDDR - Port Data Direction Register */
#define FGPIO_PDDR_PDD_MASK                      (0xFFFFFFFFU)
#define FGPIO_PDDR_PDD_SHIFT                     (0U)
#define FGPIO_PDDR_PDD(x)                        (((uint32_t)(((uint32_t)(x)) << FGPIO_PDDR_PDD_SHIFT))


/*!
 * @}
 */ /* 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)
/** Peripheral FGPIOB base address */
#define FGPIOB_BASE                              (0xF80FF040u)
/** Peripheral FGPIOB base pointer */
#define FGPIOB                                   ((FGPIO_Type *)FGPIOB_BASE)
/** Peripheral FGPIOC base address */
#define FGPIOC_BASE                              (0xF80FF080u)
/** Peripheral FGPIOC base pointer */
#define FGPIOC                                   ((FGPIO_Type *)FGPIOC_BASE)
/** Peripheral FGPIOD base address */
#define FGPIOD_BASE                              (0xF80FF0C0u)
/** Peripheral FGPIOD base pointer */
#define FGPIOD                                   ((FGPIO_Type *)FGPIOD_BASE)
/** Peripheral FGPIOE base address */
#define FGPIOE_BASE                              (0xF80FF100u)
/** Peripheral FGPIOE base pointer */
#define FGPIOE                                   ((FGPIO_Type *)FGPIOE_BASE)
/** Array initializer of FGPIO peripheral base addresses */
#define FGPIO_BASE_ADDRS                         { FGPIOA_BASE, FGPIOB_BASE, FGPIOC_BASE, FGPIO
/** Array initializer of FGPIO peripheral base pointers */
#define FGPIO_BASE_PTRS                          { FGPIOA, FGPIOB, FGPIOC, FGPIOD, FGPIOE }

/*!
 * @}
 */ /* 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 FPROT0;                             /**< Program Flash Protection Registers, offset: 0x13 */
} FTFA_Type;

/* ----------------------------------------------------------------------------
   -- FTFA Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup FTFA_Register_Masks FTFA Register Masks
 * @{
 */

/*! @name FSTAT - Flash Status Register */
#define FTFA_FSTAT_MGSTAT0_MASK                  (0x1U)
#define FTFA_FSTAT_MGSTAT0_SHIFT                 (0U)
#define FTFA_FSTAT_MGSTAT0(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FSTAT_MGSTAT0_SH
#define FTFA_FSTAT_FPVIOL_MASK                   (0x10U)
#define FTFA_FSTAT_FPVIOL_SHIFT                  (4U)
#define FTFA_FSTAT_FPVIOL(x)                     (((uint8_t)(((uint8_t)(x)) << FTFA_FSTAT_FPVIOL_SHIFT))
#define FTFA_FSTAT_ACCERR_MASK                   (0x20U)
#define FTFA_FSTAT_ACCERR_SHIFT                  (5U)
```

```
#define FTFA_FSTAT_ACCERR(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FSTAT_ACCERR_SHIF
#define FTFA_FSTAT_RDCOLERR_MASK                 (0x40U)
#define FTFA_FSTAT_RDCOLERR_SHIFT                (6U)
#define FTFA_FSTAT_RDCOLERR(x)                  (((uint8_t)(((uint8_t)(x)) << FTFA_FSTAT_RDCOLERR_
#define FTFA_FSTAT_CCIF_MASK                     (0x80U)
#define FTFA_FSTAT_CCIF_SHIFT                    (7U)
#define FTFA_FSTAT_CCIF(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FSTAT_CCIF_SHIFT)) & F

/*! @name FCNFG - Flash Configuration Register */
#define FTFA_FCNFG_ERSSUSP_MASK                  (0x10U)
#define FTFA_FCNFG_ERSSUSP_SHIFT                 (4U)
#define FTFA_FCNFG_ERSSUSP(x)                   (((uint8_t)(((uint8_t)(x)) << FTFA_FCNFG_ERSSUSP_S
#define FTFA_FCNFG_ERSAREQ_MASK                  (0x20U)
#define FTFA_FCNFG_ERSAREQ_SHIFT                 (5U)
#define FTFA_FCNFG_ERSAREQ(x)                   (((uint8_t)(((uint8_t)(x)) << FTFA_FCNFG_ERSAREQ_S
#define FTFA_FCNFG_RDCOLLIE_MASK                 (0x40U)
#define FTFA_FCNFG_RDCOLLIE_SHIFT                (6U)
#define FTFA_FCNFG_RDCOLLIE(x)                  (((uint8_t)(((uint8_t)(x)) << FTFA_FCNFG_RDCOLLIE_S
#define FTFA_FCNFG_CCIE_MASK                     (0x80U)
#define FTFA_FCNFG_CCIE_SHIFT                    (7U)
#define FTFA_FCNFG_CCIE(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FCNFG_CCIE_SHIFT)) &

/*! @name FSEC - Flash Security Register */
#define FTFA_FSEC_SEC_MASK                       (0x3U)
#define FTFA_FSEC_SEC_SHIFT                      (0U)
#define FTFA_FSEC_SEC(x)                        (((uint8_t)(((uint8_t)(x)) << FTFA_FSEC_SEC_SHIFT)) & FTF
#define FTFA_FSEC_FSLACC_MASK                    (0xCU)
#define FTFA_FSEC_FSLACC_SHIFT                   (2U)
#define FTFA_FSEC_FSLACC(x)                     (((uint8_t)(((uint8_t)(x)) << FTFA_FSEC_FSLACC_SHIFT)
#define FTFA_FSEC_MEEN_MASK                      (0x30U)
#define FTFA_FSEC_MEEN_SHIFT                     (4U)
#define FTFA_FSEC_MEEN(x)                       (((uint8_t)(((uint8_t)(x)) << FTFA_FSEC_MEEN_SHIFT)) &
#define FTFA_FSEC_KEYEN_MASK                     (0xC0U)
#define FTFA_FSEC_KEYEN_SHIFT                    (6U)
#define FTFA_FSEC_KEYEN(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FSEC_KEYEN_SHIFT)) &

/*! @name FOPT - Flash Option Register */
#define FTFA_FOPT_OPT_MASK                       (0xFFU)
#define FTFA_FOPT_OPT_SHIFT                      (0U)
#define FTFA_FOPT_OPT(x)                        (((uint8_t)(((uint8_t)(x)) << FTFA_FOPT_OPT_SHIFT)) & FTF

/*! @name FCCOB3 - Flash Common Command Object Registers */
#define FTFA_FCCOB3_CCOBn_MASK                   (0xFFU)
#define FTFA_FCCOB3_CCOBn_SHIFT                  (0U)
#define FTFA_FCCOB3_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB3_CCOBn_SHI

/*! @name FCCOB2 - Flash Common Command Object Registers */
#define FTFA_FCCOB2_CCOBn_MASK                   (0xFFU)
#define FTFA_FCCOB2_CCOBn_SHIFT                  (0U)
#define FTFA_FCCOB2_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB2_CCOBn_SHI

/*! @name FCCOB1 - Flash Common Command Object Registers */
#define FTFA_FCCOB1_CCOBn_MASK                   (0xFFU)
```

```
#define FTFA_FCCOB1_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB1_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB1_CCOBn_SHI

/*! @name FCCOB0 - Flash Common Command Object Registers */
#define FTFA_FCCOB0_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB0_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB0_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB0_CCOBn_SHI

/*! @name FCCOB7 - Flash Common Command Object Registers */
#define FTFA_FCCOB7_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB7_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB7_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB7_CCOBn_SHI

/*! @name FCCOB6 - Flash Common Command Object Registers */
#define FTFA_FCCOB6_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB6_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB6_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB6_CCOBn_SHI

/*! @name FCCOB5 - Flash Common Command Object Registers */
#define FTFA_FCCOB5_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB5_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB5_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB5_CCOBn_SHI

/*! @name FCCOB4 - Flash Common Command Object Registers */
#define FTFA_FCCOB4_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB4_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB4_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB4_CCOBn_SHI

/*! @name FCCOBB - Flash Common Command Object Registers */
#define FTFA_FCCOBB_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOBB_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOBB_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOBB_CCOBn_SH

/*! @name FCCOBA - Flash Common Command Object Registers */
#define FTFA_FCCOBA_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOBA_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOBA_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOBA_CCOBn_SH

/*! @name FCCOB9 - Flash Common Command Object Registers */
#define FTFA_FCCOB9_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB9_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB9_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB9_CCOBn_SHI

/*! @name FCCOB8 - Flash Common Command Object Registers */
#define FTFA_FCCOB8_CCOBn_MASK                  (0xFFU)
#define FTFA_FCCOB8_CCOBn_SHIFT                 (0U)
#define FTFA_FCCOB8_CCOBn(x)                    (((uint8_t)(((uint8_t)(x)) << FTFA_FCCOB8_CCOBn_SHI

/*! @name FPROT3 - Program Flash Protection Registers */
#define FTFA_FPROT3_PROT_MASK                   (0xFFU)
#define FTFA_FPROT3_PROT_SHIFT                  (0U)
#define FTFA_FPROT3_PROT(x)                     (((uint8_t)(((uint8_t)(x)) << FTFA_FPROT3_PROT_SHIFT)
/*! @name FPROT2 - Program Flash Protection Registers */
#define FTFA_FPROT2_PROT_MASK                    (0xFFU)
#define FTFA_FPROT2_PROT_SHIFT                   (0U)
#define FTFA_FPROT2_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FPROT2_PROT_SHIFT)

/*! @name FPROT1 - Program Flash Protection Registers */
#define FTFA_FPROT1_PROT_MASK                    (0xFFU)
#define FTFA_FPROT1_PROT_SHIFT                   (0U)
#define FTFA_FPROT1_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FPROT1_PROT_SHIFT)

/*! @name FPROT0 - Program Flash Protection Registers */
#define FTFA_FPROT0_PROT_MASK                    (0xFFU)
#define FTFA_FPROT0_PROT_SHIFT                   (0U)
#define FTFA_FPROT0_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << FTFA_FPROT0_PROT_SHIFT)


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the FTFA peripheral type */
#define FTFA_COMMAND_COMPLETE_IRQS               { FTFA_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup GPIO_Register_Masks GPIO Register Masks
 * @{
 */

/*! @name PDOR - Port Data Output Register */
#define GPIO_PDOR_PDO_MASK                       (0xFFFFFFFFU)
#define GPIO_PDOR_PDO_SHIFT                      (0U)
#define GPIO_PDOR_PDO(x)                         (((uint32_t)(((uint32_t)(x)) << GPIO_PDOR_PDO_SHIFT)) &

/*! @name PSOR - Port Set Output Register */
#define GPIO_PSOR_PTSO_MASK                      (0xFFFFFFFFU)
#define GPIO_PSOR_PTSO_SHIFT                     (0U)
#define GPIO_PSOR_PTSO(x)                        (((uint32_t)(((uint32_t)(x)) << GPIO_PSOR_PTSO_SHIFT))

/*! @name PCOR - Port Clear Output Register */
#define GPIO_PCOR_PTCO_MASK                      (0xFFFFFFFFU)
#define GPIO_PCOR_PTCO_SHIFT                     (0U)
#define GPIO_PCOR_PTCO(x)                        (((uint32_t)(((uint32_t)(x)) << GPIO_PCOR_PTCO_SHIFT))

/*! @name PTOR - Port Toggle Output Register */
#define GPIO_PTOR_PTTO_MASK                      (0xFFFFFFFFU)
#define GPIO_PTOR_PTTO_SHIFT                     (0U)
#define GPIO_PTOR_PTTO(x)                        (((uint32_t)(((uint32_t)(x)) << GPIO_PTOR_PTTO_SHIFT))

/*! @name PDIR - Port Data Input Register */
#define GPIO_PDIR_PDI_MASK                       (0xFFFFFFFFU)
#define GPIO_PDIR_PDI_SHIFT                      (0U)
#define GPIO_PDIR_PDI(x)                         (((uint32_t)(((uint32_t)(x)) << GPIO_PDIR_PDI_SHIFT)) & GPI

/*! @name PDDR - Port Data Direction Register */
#define GPIO_PDDR_PDD_MASK                       (0xFFFFFFFFU)
#define GPIO_PDDR_PDD_SHIFT                      (0U)
#define GPIO_PDDR_PDD(x)                         (((uint32_t)(((uint32_t)(x)) << GPIO_PDDR_PDD_SHIFT)) &


/*!
 * @}
 */ /* 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)
/** Peripheral GPIOB base address */
#define GPIOB_BASE                                  (0x400FF040u)
/** Peripheral GPIOB base pointer */
#define GPIOB                                  ((GPIO_Type *)GPIOB_BASE)
/** Peripheral GPIOC base address */
#define GPIOC_BASE                                  (0x400FF080u)
/** Peripheral GPIOC base pointer */
#define GPIOC                                  ((GPIO_Type *)GPIOC_BASE)
/** Peripheral GPIOD base address */
#define GPIOD_BASE                                  (0x400FF0C0u)
/** Peripheral GPIOD base pointer */
#define GPIOD                                  ((GPIO_Type *)GPIOD_BASE)
/** Peripheral GPIOE base address */
#define GPIOE_BASE                                  (0x400FF100u)
/** Peripheral GPIOE base pointer */
#define GPIOE                                  ((GPIO_Type *)GPIOE_BASE)
/** Array initializer of GPIO peripheral base addresses */
#define GPIO_BASE_ADDRS                          { GPIOA_BASE, GPIOB_BASE, GPIOC_BASE, GPIOD_BA
/** Array initializer of GPIO peripheral base pointers */
#define GPIO_BASE_PTRS                           { GPIOA, GPIOB, GPIOC, GPIOD, GPIOE }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup I2C_Register_Masks I2C Register Masks
 * @{
 */

/*! @name A1 - I2C Address Register 1 */
#define I2C_A1_AD_MASK                           (0xFEU)
#define I2C_A1_AD_SHIFT                          (1U)
#define I2C_A1_AD(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_A1_AD_SHIFT)) & I2C_A1_AD_M

/*! @name F - I2C Frequency Divider register */
#define I2C_F_ICR_MASK                           (0x3FU)
#define I2C_F_ICR_SHIFT                          (0U)
#define I2C_F_ICR(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_F_ICR_SHIFT)) & I2C_F_ICR_MA
#define I2C_F_MULT_MASK                          (0xC0U)
#define I2C_F_MULT_SHIFT                         (6U)
#define I2C_F_MULT(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_F_MULT_SHIFT)) & I2C_F_MUL

/*! @name C1 - I2C Control Register 1 */
#define I2C_C1_DMAEN_MASK                        (0x1U)
#define I2C_C1_DMAEN_SHIFT                       (0U)
#define I2C_C1_DMAEN(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_C1_DMAEN_SHIFT)) & I2C_C
#define I2C_C1_WUEN_MASK                         (0x2U)
#define I2C_C1_WUEN_SHIFT                        (1U)
#define I2C_C1_WUEN(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C1_WUEN_SHIFT)) & I2C_C1_
#define I2C_C1_RSTA_MASK                         (0x4U)
#define I2C_C1_RSTA_SHIFT                        (2U)
#define I2C_C1_RSTA(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C1_RSTA_SHIFT)) & I2C_C1_R
#define I2C_C1_TXAK_MASK                         (0x8U)
#define I2C_C1_TXAK_SHIFT                        (3U)
#define I2C_C1_TXAK(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C1_TXAK_SHIFT)) & I2C_C1_T
#define I2C_C1_TX_MASK                           (0x10U)
#define I2C_C1_TX_SHIFT                          (4U)
#define I2C_C1_TX(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_C1_TX_SHIFT)) & I2C_C1_TX_M
#define I2C_C1_MST_MASK                          (0x20U)
#define I2C_C1_MST_SHIFT                         (5U)
#define I2C_C1_MST(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_C1_MST_SHIFT)) & I2C_C1_MS
#define I2C_C1_IICIE_MASK                        (0x40U)
#define I2C_C1_IICIE_SHIFT                       (6U)
#define I2C_C1_IICIE(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_C1_IICIE_SHIFT)) & I2C_C1_IICI
#define I2C_C1_IICEN_MASK                        (0x80U)
#define I2C_C1_IICEN_SHIFT                       (7U)
#define I2C_C1_IICEN(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_C1_IICEN_SHIFT)) & I2C_C1_II

/*! @name S - I2C Status register */
#define I2C_S_RXAK_MASK                          (0x1U)
#define I2C_S_RXAK_SHIFT                         (0U)
#define I2C_S_RXAK(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_S_RXAK_SHIFT)) & I2C_S_RXA
#define I2C_S_IICIF_MASK                         (0x2U)
#define I2C_S_IICIF_SHIFT                        (1U)
```

```
#define I2C_S_IICIF(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_S_IICIF_SHIFT)) & I2C_S_IICIF_M
#define I2C_S_SRW_MASK                           (0x4U)
#define I2C_S_SRW_SHIFT                          (2U)
#define I2C_S_SRW(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_S_SRW_SHIFT)) & I2C_S_SRW
#define I2C_S_RAM_MASK                           (0x8U)
#define I2C_S_RAM_SHIFT                          (3U)
#define I2C_S_RAM(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_S_RAM_SHIFT)) & I2C_S_RAM_
#define I2C_S_ARBL_MASK                          (0x10U)
#define I2C_S_ARBL_SHIFT                         (4U)
#define I2C_S_ARBL(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_S_ARBL_SHIFT)) & I2C_S_ARB
#define I2C_S_BUSY_MASK                          (0x20U)
#define I2C_S_BUSY_SHIFT                         (5U)
#define I2C_S_BUSY(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_S_BUSY_SHIFT)) & I2C_S_BUS
#define I2C_S_IAAS_MASK                          (0x40U)
#define I2C_S_IAAS_SHIFT                         (6U)
#define I2C_S_IAAS(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_S_IAAS_SHIFT)) & I2C_S_IAAS_
#define I2C_S_TCF_MASK                           (0x80U)
#define I2C_S_TCF_SHIFT                          (7U)
#define I2C_S_TCF(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_S_TCF_SHIFT)) & I2C_S_TCF_M

/*! @name D - I2C Data I/O register */
#define I2C_D_DATA_MASK                          (0xFFU)
#define I2C_D_DATA_SHIFT                         (0U)
#define I2C_D_DATA(x)                            (((uint8_t)(((uint8_t)(x)) << I2C_D_DATA_SHIFT)) & I2C_D_DAT

/*! @name C2 - I2C Control Register 2 */
#define I2C_C2_AD_MASK                           (0x7U)
#define I2C_C2_AD_SHIFT                          (0U)
#define I2C_C2_AD(x)                             (((uint8_t)(((uint8_t)(x)) << I2C_C2_AD_SHIFT)) & I2C_C2_AD_M
#define I2C_C2_RMEN_MASK                         (0x8U)
#define I2C_C2_RMEN_SHIFT                        (3U)
#define I2C_C2_RMEN(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C2_RMEN_SHIFT)) & I2C_C2_
#define I2C_C2_SBRC_MASK                         (0x10U)
#define I2C_C2_SBRC_SHIFT                        (4U)
#define I2C_C2_SBRC(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C2_SBRC_SHIFT)) & I2C_C2_S
#define I2C_C2_HDRS_MASK                         (0x20U)
#define I2C_C2_HDRS_SHIFT                        (5U)
#define I2C_C2_HDRS(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_C2_HDRS_SHIFT)) & I2C_C2_
#define I2C_C2_ADEXT_MASK                        (0x40U)
#define I2C_C2_ADEXT_SHIFT                       (6U)
#define I2C_C2_ADEXT(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_C2_ADEXT_SHIFT)) & I2C_C2
#define I2C_C2_GCAEN_MASK                        (0x80U)
#define I2C_C2_GCAEN_SHIFT                       (7U)
#define I2C_C2_GCAEN(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_C2_GCAEN_SHIFT)) & I2C_C

/*! @name FLT - I2C Programmable Input Glitch Filter register */
#define I2C_FLT_FLT_MASK                         (0x1FU)
#define I2C_FLT_FLT_SHIFT                        (0U)
#define I2C_FLT_FLT(x)                           (((uint8_t)(((uint8_t)(x)) << I2C_FLT_FLT_SHIFT)) & I2C_FLT_F
#define I2C_FLT_STOPIE_MASK                      (0x20U)
#define I2C_FLT_STOPIE_SHIFT                     (5U)
#define I2C_FLT_STOPIE(x)                        (((uint8_t)(((uint8_t)(x)) << I2C_FLT_STOPIE_SHIFT)) & I2C_
#define I2C_FLT_STOPF_MASK                       (0x40U)
```

```
#define I2C_FLT_STOPF_SHIFT                    (6U)
#define I2C_FLT_STOPF(x)                       (((uint8_t)(((uint8_t)(x)) << I2C_FLT_STOPF_SHIFT)) & I2C_F
#define I2C_FLT_SHEN_MASK                      (0x80U)
#define I2C_FLT_SHEN_SHIFT                     (7U)
#define I2C_FLT_SHEN(x)                        (((uint8_t)(((uint8_t)(x)) << I2C_FLT_SHEN_SHIFT)) & I2C_FL

/*! @name RA - I2C Range Address register */
#define I2C_RA_RAD_MASK                        (0xFEU)
#define I2C_RA_RAD_SHIFT                       (1U)
#define I2C_RA_RAD(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_RA_RAD_SHIFT)) & I2C_RA_R

/*! @name SMB - I2C SMBus Control and Status register */
#define I2C_SMB_SHTF2IE_MASK                   (0x1U)
#define I2C_SMB_SHTF2IE_SHIFT                  (0U)
#define I2C_SMB_SHTF2IE(x)                     (((uint8_t)(((uint8_t)(x)) << I2C_SMB_SHTF2IE_SHIFT)) & I2
#define I2C_SMB_SHTF2_MASK                     (0x2U)
#define I2C_SMB_SHTF2_SHIFT                    (1U)
#define I2C_SMB_SHTF2(x)                       (((uint8_t)(((uint8_t)(x)) << I2C_SMB_SHTF2_SHIFT)) & I2C_
#define I2C_SMB_SHTF1_MASK                     (0x4U)
#define I2C_SMB_SHTF1_SHIFT                    (2U)
#define I2C_SMB_SHTF1(x)                       (((uint8_t)(((uint8_t)(x)) << I2C_SMB_SHTF1_SHIFT)) & I2C_
#define I2C_SMB_SLTF_MASK                      (0x8U)
#define I2C_SMB_SLTF_SHIFT                     (3U)
#define I2C_SMB_SLTF(x)                        (((uint8_t)(((uint8_t)(x)) << I2C_SMB_SLTF_SHIFT)) & I2C_SM
#define I2C_SMB_TCKSEL_MASK                    (0x10U)
#define I2C_SMB_TCKSEL_SHIFT                   (4U)
#define I2C_SMB_TCKSEL(x)                      (((uint8_t)(((uint8_t)(x)) << I2C_SMB_TCKSEL_SHIFT)) & I2
#define I2C_SMB_SIICAEN_MASK                   (0x20U)
#define I2C_SMB_SIICAEN_SHIFT                  (5U)
#define I2C_SMB_SIICAEN(x)                     (((uint8_t)(((uint8_t)(x)) << I2C_SMB_SIICAEN_SHIFT)) & I2
#define I2C_SMB_ALERTEN_MASK                   (0x40U)
#define I2C_SMB_ALERTEN_SHIFT                  (6U)
#define I2C_SMB_ALERTEN(x)                     (((uint8_t)(((uint8_t)(x)) << I2C_SMB_ALERTEN_SHIFT)) &
#define I2C_SMB_FACK_MASK                      (0x80U)
#define I2C_SMB_FACK_SHIFT                     (7U)
#define I2C_SMB_FACK(x)                        (((uint8_t)(((uint8_t)(x)) << I2C_SMB_FACK_SHIFT)) & I2C_S

/*! @name A2 - I2C Address Register 2 */
#define I2C_A2_SAD_MASK                        (0xFEU)
#define I2C_A2_SAD_SHIFT                       (1U)
#define I2C_A2_SAD(x)                          (((uint8_t)(((uint8_t)(x)) << I2C_A2_SAD_SHIFT)) & I2C_A2_SA

/*! @name SLTH - I2C SCL Low Timeout Register High */
#define I2C_SLTH_SSLT_MASK                     (0xFFU)
#define I2C_SLTH_SSLT_SHIFT                    (0U)
#define I2C_SLTH_SSLT(x)                       (((uint8_t)(((uint8_t)(x)) << I2C_SLTH_SSLT_SHIFT)) & I2C_S

/*! @name SLTL - I2C SCL Low Timeout Register Low */
#define I2C_SLTL_SSLT_MASK                     (0xFFU)
#define I2C_SLTL_SSLT_SHIFT                    (0U)
#define I2C_SLTL_SSLT(x)                       (((uint8_t)(((uint8_t)(x)) << I2C_SLTL_SSLT_SHIFT)) & I2C_SL
/*!
 * @}
 */ /* 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)
/** Peripheral I2C1 base address */
#define I2C1_BASE                                (0x40067000u)
/** Peripheral I2C1 base pointer */
#define I2C1                                     ((I2C_Type *)I2C1_BASE)
/** 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 }
/** Interrupt vectors for the I2C peripheral type */
#define I2C_IRQS                                 { I2C0_IRQn, I2C1_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */
```

```
/*!
 * @addtogroup LLWU_Register_Masks LLWU Register Masks
 * @{
 */

/*! @name PE1 - LLWU Pin Enable 1 register */
#define LLWU_PE1_WUPE0_MASK                      (0x3U)
#define LLWU_PE1_WUPE0_SHIFT                     (0U)
#define LLWU_PE1_WUPE0(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE1_WUPE0_SHIFT)) &
#define LLWU_PE1_WUPE1_MASK                      (0xCU)
#define LLWU_PE1_WUPE1_SHIFT                     (2U)
#define LLWU_PE1_WUPE1(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE1_WUPE1_SHIFT)) &
#define LLWU_PE1_WUPE2_MASK                      (0x30U)
#define LLWU_PE1_WUPE2_SHIFT                     (4U)
#define LLWU_PE1_WUPE2(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE1_WUPE2_SHIFT)) &
#define LLWU_PE1_WUPE3_MASK                      (0xC0U)
#define LLWU_PE1_WUPE3_SHIFT                     (6U)
#define LLWU_PE1_WUPE3(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE1_WUPE3_SHIFT)) &

/*! @name PE2 - LLWU Pin Enable 2 register */
#define LLWU_PE2_WUPE4_MASK                      (0x3U)
#define LLWU_PE2_WUPE4_SHIFT                     (0U)
#define LLWU_PE2_WUPE4(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE2_WUPE4_SHIFT)) &
#define LLWU_PE2_WUPE5_MASK                      (0xCU)
#define LLWU_PE2_WUPE5_SHIFT                     (2U)
#define LLWU_PE2_WUPE5(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE2_WUPE5_SHIFT)) &
#define LLWU_PE2_WUPE6_MASK                      (0x30U)
#define LLWU_PE2_WUPE6_SHIFT                     (4U)
#define LLWU_PE2_WUPE6(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE2_WUPE6_SHIFT)) &
#define LLWU_PE2_WUPE7_MASK                      (0xC0U)
#define LLWU_PE2_WUPE7_SHIFT                     (6U)
#define LLWU_PE2_WUPE7(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE2_WUPE7_SHIFT)) &

/*! @name PE3 - LLWU Pin Enable 3 register */
#define LLWU_PE3_WUPE8_MASK                      (0x3U)
#define LLWU_PE3_WUPE8_SHIFT                     (0U)
#define LLWU_PE3_WUPE8(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE3_WUPE8_SHIFT)) &
#define LLWU_PE3_WUPE9_MASK                      (0xCU)
#define LLWU_PE3_WUPE9_SHIFT                     (2U)
#define LLWU_PE3_WUPE9(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_PE3_WUPE9_SHIFT)) &
#define LLWU_PE3_WUPE10_MASK                     (0x30U)
#define LLWU_PE3_WUPE10_SHIFT                    (4U)
#define LLWU_PE3_WUPE10(x)                       (((uint8_t)(((uint8_t)(x)) << LLWU_PE3_WUPE10_SHIFT))
#define LLWU_PE3_WUPE11_MASK                     (0xC0U)
#define LLWU_PE3_WUPE11_SHIFT                    (6U)
#define LLWU_PE3_WUPE11(x)                       (((uint8_t)(((uint8_t)(x)) << LLWU_PE3_WUPE11_SHIFT))

/*! @name PE4 - LLWU Pin Enable 4 register */
#define LLWU_PE4_WUPE12_MASK                     (0x3U)
#define LLWU_PE4_WUPE12_SHIFT                    (0U)
#define LLWU_PE4_WUPE12(x)                       (((uint8_t)(((uint8_t)(x)) << LLWU_PE4_WUPE12_SHIFT))
#define LLWU_PE4_WUPE13_MASK                     (0xCU)
#define LLWU_PE4_WUPE13_SHIFT                    (2U)
```

```
#define LLWU_PE4_WUPE13(x)                   (((uint8_t)(((uint8_t)(x)) << LLWU_PE4_WUPE13_SHIFT))
#define LLWU_PE4_WUPE14_MASK                     (0x30U)
#define LLWU_PE4_WUPE14_SHIFT                    (4U)
#define LLWU_PE4_WUPE14(x)                   (((uint8_t)(((uint8_t)(x)) << LLWU_PE4_WUPE14_SHIFT))
#define LLWU_PE4_WUPE15_MASK                     (0xC0U)
#define LLWU_PE4_WUPE15_SHIFT                    (6U)
#define LLWU_PE4_WUPE15(x)                   (((uint8_t)(((uint8_t)(x)) << LLWU_PE4_WUPE15_SHIFT))

/*! @name ME - LLWU Module Enable register */
#define LLWU_ME_WUME0_MASK                       (0x1U)
#define LLWU_ME_WUME0_SHIFT                      (0U)
#define LLWU_ME_WUME0(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME0_SHIFT)) &
#define LLWU_ME_WUME1_MASK                       (0x2U)
#define LLWU_ME_WUME1_SHIFT                      (1U)
#define LLWU_ME_WUME1(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME1_SHIFT)) &
#define LLWU_ME_WUME2_MASK                       (0x4U)
#define LLWU_ME_WUME2_SHIFT                      (2U)
#define LLWU_ME_WUME2(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME2_SHIFT)) &
#define LLWU_ME_WUME3_MASK                       (0x8U)
#define LLWU_ME_WUME3_SHIFT                      (3U)
#define LLWU_ME_WUME3(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME3_SHIFT)) &
#define LLWU_ME_WUME4_MASK                       (0x10U)
#define LLWU_ME_WUME4_SHIFT                      (4U)
#define LLWU_ME_WUME4(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME4_SHIFT)) &
#define LLWU_ME_WUME5_MASK                       (0x20U)
#define LLWU_ME_WUME5_SHIFT                      (5U)
#define LLWU_ME_WUME5(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME5_SHIFT)) &
#define LLWU_ME_WUME6_MASK                       (0x40U)
#define LLWU_ME_WUME6_SHIFT                      (6U)
#define LLWU_ME_WUME6(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME6_SHIFT)) &
#define LLWU_ME_WUME7_MASK                       (0x80U)
#define LLWU_ME_WUME7_SHIFT                      (7U)
#define LLWU_ME_WUME7(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_ME_WUME7_SHIFT)) &

/*! @name F1 - LLWU Flag 1 register */
#define LLWU_F1_WUF0_MASK                        (0x1U)
#define LLWU_F1_WUF0_SHIFT                       (0U)
#define LLWU_F1_WUF0(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF0_SHIFT)) & LLW
#define LLWU_F1_WUF1_MASK                        (0x2U)
#define LLWU_F1_WUF1_SHIFT                       (1U)
#define LLWU_F1_WUF1(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF1_SHIFT)) & LLW
#define LLWU_F1_WUF2_MASK                        (0x4U)
#define LLWU_F1_WUF2_SHIFT                       (2U)
#define LLWU_F1_WUF2(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF2_SHIFT)) & LLW
#define LLWU_F1_WUF3_MASK                        (0x8U)
#define LLWU_F1_WUF3_SHIFT                       (3U)
#define LLWU_F1_WUF3(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF3_SHIFT)) & LLW
#define LLWU_F1_WUF4_MASK                        (0x10U)
#define LLWU_F1_WUF4_SHIFT                       (4U)
#define LLWU_F1_WUF4(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF4_SHIFT)) & LLW
#define LLWU_F1_WUF5_MASK                        (0x20U)
#define LLWU_F1_WUF5_SHIFT                       (5U)
#define LLWU_F1_WUF5(x)                      (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF5_SHIFT)) & LLW
#define LLWU_F1_WUF6_MASK                    (0x40U)
#define LLWU_F1_WUF6_SHIFT                  (6U)
#define LLWU_F1_WUF6(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF6_SHIFT)) & LLW
#define LLWU_F1_WUF7_MASK                    (0x80U)
#define LLWU_F1_WUF7_SHIFT                  (7U)
#define LLWU_F1_WUF7(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F1_WUF7_SHIFT)) & LLW

/*! @name F2 - LLWU Flag 2 register */
#define LLWU_F2_WUF8_MASK                    (0x1U)
#define LLWU_F2_WUF8_SHIFT                  (0U)
#define LLWU_F2_WUF8(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF8_SHIFT)) & LLW
#define LLWU_F2_WUF9_MASK                    (0x2U)
#define LLWU_F2_WUF9_SHIFT                  (1U)
#define LLWU_F2_WUF9(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF9_SHIFT)) & LLW
#define LLWU_F2_WUF10_MASK                    (0x4U)
#define LLWU_F2_WUF10_SHIFT                  (2U)
#define LLWU_F2_WUF10(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF10_SHIFT)) & LL
#define LLWU_F2_WUF11_MASK                    (0x8U)
#define LLWU_F2_WUF11_SHIFT                  (3U)
#define LLWU_F2_WUF11(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF11_SHIFT)) & LL
#define LLWU_F2_WUF12_MASK                    (0x10U)
#define LLWU_F2_WUF12_SHIFT                  (4U)
#define LLWU_F2_WUF12(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF12_SHIFT)) & LL
#define LLWU_F2_WUF13_MASK                    (0x20U)
#define LLWU_F2_WUF13_SHIFT                  (5U)
#define LLWU_F2_WUF13(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF13_SHIFT)) & LL
#define LLWU_F2_WUF14_MASK                    (0x40U)
#define LLWU_F2_WUF14_SHIFT                  (6U)
#define LLWU_F2_WUF14(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF14_SHIFT)) & LL
#define LLWU_F2_WUF15_MASK                    (0x80U)
#define LLWU_F2_WUF15_SHIFT                  (7U)
#define LLWU_F2_WUF15(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F2_WUF15_SHIFT)) & LL

/*! @name F3 - LLWU Flag 3 register */
#define LLWU_F3_MWUF0_MASK                    (0x1U)
#define LLWU_F3_MWUF0_SHIFT                  (0U)
#define LLWU_F3_MWUF0(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF0_SHIFT)) & L
#define LLWU_F3_MWUF1_MASK                    (0x2U)
#define LLWU_F3_MWUF1_SHIFT                  (1U)
#define LLWU_F3_MWUF1(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF1_SHIFT)) & L
#define LLWU_F3_MWUF2_MASK                    (0x4U)
#define LLWU_F3_MWUF2_SHIFT                  (2U)
#define LLWU_F3_MWUF2(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF2_SHIFT)) & L
#define LLWU_F3_MWUF3_MASK                    (0x8U)
#define LLWU_F3_MWUF3_SHIFT                  (3U)
#define LLWU_F3_MWUF3(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF3_SHIFT)) & L
#define LLWU_F3_MWUF4_MASK                    (0x10U)
#define LLWU_F3_MWUF4_SHIFT                  (4U)
#define LLWU_F3_MWUF4(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF4_SHIFT)) & L
#define LLWU_F3_MWUF5_MASK                    (0x20U)
#define LLWU_F3_MWUF5_SHIFT                  (5U)
#define LLWU_F3_MWUF5(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF5_SHIFT)) & L
#define LLWU_F3_MWUF6_MASK                    (0x40U)
```

```
#define LLWU_F3_MWUF6_SHIFT                     (6U)
#define LLWU_F3_MWUF6(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF6_SHIFT)) & LL
#define LLWU_F3_MWUF7_MASK                      (0x80U)
#define LLWU_F3_MWUF7_SHIFT                     (7U)
#define LLWU_F3_MWUF7(x)                        (((uint8_t)(((uint8_t)(x)) << LLWU_F3_MWUF7_SHIFT)) & LL

/*! @name FILT1 - LLWU Pin Filter 1 register */
#define LLWU_FILT1_FILTSEL_MASK                 (0xFU)
#define LLWU_FILT1_FILTSEL_SHIFT                (0U)
#define LLWU_FILT1_FILTSEL(x)                   (((uint8_t)(((uint8_t)(x)) << LLWU_FILT1_FILTSEL_SHIFT))
#define LLWU_FILT1_FILTE_MASK                   (0x60U)
#define LLWU_FILT1_FILTE_SHIFT                  (5U)
#define LLWU_FILT1_FILTE(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_FILT1_FILTE_SHIFT)) & L
#define LLWU_FILT1_FILTF_MASK                   (0x80U)
#define LLWU_FILT1_FILTF_SHIFT                  (7U)
#define LLWU_FILT1_FILTF(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_FILT1_FILTF_SHIFT)) & L

/*! @name FILT2 - LLWU Pin Filter 2 register */
#define LLWU_FILT2_FILTSEL_MASK                 (0xFU)
#define LLWU_FILT2_FILTSEL_SHIFT                (0U)
#define LLWU_FILT2_FILTSEL(x)                   (((uint8_t)(((uint8_t)(x)) << LLWU_FILT2_FILTSEL_SHIFT))
#define LLWU_FILT2_FILTE_MASK                   (0x60U)
#define LLWU_FILT2_FILTE_SHIFT                  (5U)
#define LLWU_FILT2_FILTE(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_FILT2_FILTE_SHIFT)) & L
#define LLWU_FILT2_FILTF_MASK                   (0x80U)
#define LLWU_FILT2_FILTF_SHIFT                  (7U)
#define LLWU_FILT2_FILTF(x)                     (((uint8_t)(((uint8_t)(x)) << LLWU_FILT2_FILTF_SHIFT)) & L


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the LLWU peripheral type */
#define LLWU_IRQS                                { LLWU_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   -------------------------------------------------------------------------- */

/*!
 * @addtogroup LPTMR_Register_Masks LPTMR Register Masks
 * @{
 */

/*! @name CSR - Low Power Timer Control Status Register */
#define LPTMR_CSR_TEN_MASK                       (0x1U)
#define LPTMR_CSR_TEN_SHIFT                      (0U)
#define LPTMR_CSR_TEN(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TEN_SHIFT)) &
#define LPTMR_CSR_TMS_MASK                       (0x2U)
#define LPTMR_CSR_TMS_SHIFT                      (1U)
#define LPTMR_CSR_TMS(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TMS_SHIFT)) &
#define LPTMR_CSR_TFC_MASK                       (0x4U)
#define LPTMR_CSR_TFC_SHIFT                      (2U)
#define LPTMR_CSR_TFC(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TFC_SHIFT)) &
#define LPTMR_CSR_TPP_MASK                       (0x8U)
#define LPTMR_CSR_TPP_SHIFT                      (3U)
#define LPTMR_CSR_TPP(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TPP_SHIFT)) &
#define LPTMR_CSR_TPS_MASK                       (0x30U)
#define LPTMR_CSR_TPS_SHIFT                      (4U)
#define LPTMR_CSR_TPS(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TPS_SHIFT)) &
#define LPTMR_CSR_TIE_MASK                       (0x40U)
#define LPTMR_CSR_TIE_SHIFT                      (6U)
#define LPTMR_CSR_TIE(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TIE_SHIFT)) & L
#define LPTMR_CSR_TCF_MASK                       (0x80U)
#define LPTMR_CSR_TCF_SHIFT                      (7U)
#define LPTMR_CSR_TCF(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_CSR_TCF_SHIFT)) &

/*! @name PSR - Low Power Timer Prescale Register */
#define LPTMR_PSR_PCS_MASK                       (0x3U)
#define LPTMR_PSR_PCS_SHIFT                      (0U)
#define LPTMR_PSR_PCS(x)                         (((uint32_t)(((uint32_t)(x)) << LPTMR_PSR_PCS_SHIFT)) &
#define LPTMR_PSR_PBYP_MASK                      (0x4U)
#define LPTMR_PSR_PBYP_SHIFT                     (2U)
```

```
#define LPTMR_PSR_PBYP(x)                        (((uint32_t)(((uint32_t)(x)) << LPTMR_PSR_PBYP_SHIFT))
#define LPTMR_PSR_PRESCALE_MASK                  (0x78U)
#define LPTMR_PSR_PRESCALE_SHIFT                 (3U)
#define LPTMR_PSR_PRESCALE(x)                    (((uint32_t)(((uint32_t)(x)) << LPTMR_PSR_PRESCALE

/*! @name CMR - Low Power Timer Compare Register */
#define LPTMR_CMR_COMPARE_MASK                   (0xFFFFU)
#define LPTMR_CMR_COMPARE_SHIFT                  (0U)
#define LPTMR_CMR_COMPARE(x)                     (((uint32_t)(((uint32_t)(x)) << LPTMR_CMR_COMPARE

/*! @name CNR - Low Power Timer Counter Register */
#define LPTMR_CNR_COUNTER_MASK                   (0xFFFFU)
#define LPTMR_CNR_COUNTER_SHIFT                  (0U)
#define LPTMR_CNR_COUNTER(x)                     (((uint32_t)(((uint32_t)(x)) << LPTMR_CNR_COUNTER_


/*!
 * @}
 */ /* end of group LPTMR_Register_Masks */


/* LPTMR - Peripheral instance base addresses */
/** Peripheral LPTMR0 base address */
#define LPTMR0_BASE                              (0x40040000u)
/** Peripheral LPTMR0 base pointer */
#define LPTMR0                                   ((LPTMR_Type *)LPTMR0_BASE)
/** Array initializer of LPTMR peripheral base addresses */
#define LPTMR_BASE_ADDRS                         { LPTMR0_BASE }
/** Array initializer of LPTMR peripheral base pointers */
#define LPTMR_BASE_PTRS                          { LPTMR0 }
/** Interrupt vectors for the LPTMR peripheral type */
#define LPTMR_IRQS                               { LPTMR0_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup MCG_Register_Masks MCG Register Masks
 * @{
 */

/*! @name C1 - MCG Control 1 Register */
#define MCG_C1_IREFSTEN_MASK                     (0x1U)
#define MCG_C1_IREFSTEN_SHIFT                    (0U)
#define MCG_C1_IREFSTEN(x)                       (((uint8_t)(((uint8_t)(x)) << MCG_C1_IREFSTEN_SHIFT)) &
#define MCG_C1_IRCLKEN_MASK                      (0x2U)
#define MCG_C1_IRCLKEN_SHIFT                     (1U)
#define MCG_C1_IRCLKEN(x)                        (((uint8_t)(((uint8_t)(x)) << MCG_C1_IRCLKEN_SHIFT)) & M
#define MCG_C1_IREFS_MASK                        (0x4U)
#define MCG_C1_IREFS_SHIFT                       (2U)
#define MCG_C1_IREFS(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_C1_IREFS_SHIFT)) & MCG_
#define MCG_C1_FRDIV_MASK                        (0x38U)
#define MCG_C1_FRDIV_SHIFT                       (3U)
#define MCG_C1_FRDIV(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_C1_FRDIV_SHIFT)) & MCG_
#define MCG_C1_CLKS_MASK                         (0xC0U)
#define MCG_C1_CLKS_SHIFT                        (6U)
#define MCG_C1_CLKS(x)                           (((uint8_t)(((uint8_t)(x)) << MCG_C1_CLKS_SHIFT)) & MCG_

/*! @name C2 - MCG Control 2 Register */
#define MCG_C2_IRCS_MASK                         (0x1U)
#define MCG_C2_IRCS_SHIFT                        (0U)
#define MCG_C2_IRCS(x)                           (((uint8_t)(((uint8_t)(x)) << MCG_C2_IRCS_SHIFT)) & MCG_C
#define MCG_C2_LP_MASK                           (0x2U)
#define MCG_C2_LP_SHIFT                          (1U)
#define MCG_C2_LP(x)                             (((uint8_t)(((uint8_t)(x)) << MCG_C2_LP_SHIFT)) & MCG_C2_L
#define MCG_C2_EREFS0_MASK                       (0x4U)
#define MCG_C2_EREFS0_SHIFT                      (2U)
#define MCG_C2_EREFS0(x)                         (((uint8_t)(((uint8_t)(x)) << MCG_C2_EREFS0_SHIFT)) & M
#define MCG_C2_HGO0_MASK                         (0x8U)
#define MCG_C2_HGO0_SHIFT                        (3U)
#define MCG_C2_HGO0(x)                           (((uint8_t)(((uint8_t)(x)) << MCG_C2_HGO0_SHIFT)) & MCG_
#define MCG_C2_RANGE0_MASK                    (0x30U)
#define MCG_C2_RANGE0_SHIFT                   (4U)
#define MCG_C2_RANGE0(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C2_RANGE0_SHIFT)) & M
#define MCG_C2_LOCRE0_MASK                    (0x80U)
#define MCG_C2_LOCRE0_SHIFT                   (7U)
#define MCG_C2_LOCRE0(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C2_LOCRE0_SHIFT)) & M

/*! @name C3 - MCG Control 3 Register */
#define MCG_C3_SCTRIM_MASK                    (0xFFU)
#define MCG_C3_SCTRIM_SHIFT                   (0U)
#define MCG_C3_SCTRIM(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C3_SCTRIM_SHIFT)) & MC

/*! @name C4 - MCG Control 4 Register */
#define MCG_C4_SCFTRIM_MASK                   (0x1U)
#define MCG_C4_SCFTRIM_SHIFT                  (0U)
#define MCG_C4_SCFTRIM(x)                     (((uint8_t)(((uint8_t)(x)) << MCG_C4_SCFTRIM_SHIFT)) &
#define MCG_C4_FCTRIM_MASK                    (0x1EU)
#define MCG_C4_FCTRIM_SHIFT                   (1U)
#define MCG_C4_FCTRIM(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C4_FCTRIM_SHIFT)) & MC
#define MCG_C4_DRST_DRS_MASK                  (0x60U)
#define MCG_C4_DRST_DRS_SHIFT                 (5U)
#define MCG_C4_DRST_DRS(x)                    (((uint8_t)(((uint8_t)(x)) << MCG_C4_DRST_DRS_SHIFT)
#define MCG_C4_DMX32_MASK                     (0x80U)
#define MCG_C4_DMX32_SHIFT                    (7U)
#define MCG_C4_DMX32(x)                       (((uint8_t)(((uint8_t)(x)) << MCG_C4_DMX32_SHIFT)) & MC

/*! @name C5 - MCG Control 5 Register */
#define MCG_C5_PRDIV0_MASK                    (0x1FU)
#define MCG_C5_PRDIV0_SHIFT                   (0U)
#define MCG_C5_PRDIV0(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C5_PRDIV0_SHIFT)) & MC
#define MCG_C5_PLLSTEN0_MASK                  (0x20U)
#define MCG_C5_PLLSTEN0_SHIFT                 (5U)
#define MCG_C5_PLLSTEN0(x)                    (((uint8_t)(((uint8_t)(x)) << MCG_C5_PLLSTEN0_SHIFT)) &
#define MCG_C5_PLLCLKEN0_MASK                 (0x40U)
#define MCG_C5_PLLCLKEN0_SHIFT                (6U)
#define MCG_C5_PLLCLKEN0(x)                   (((uint8_t)(((uint8_t)(x)) << MCG_C5_PLLCLKEN0_SHIFT

/*! @name C6 - MCG Control 6 Register */
#define MCG_C6_VDIV0_MASK                     (0x1FU)
#define MCG_C6_VDIV0_SHIFT                    (0U)
#define MCG_C6_VDIV0(x)                       (((uint8_t)(((uint8_t)(x)) << MCG_C6_VDIV0_SHIFT)) & MCG_
#define MCG_C6_CME0_MASK                      (0x20U)
#define MCG_C6_CME0_SHIFT                     (5U)
#define MCG_C6_CME0(x)                        (((uint8_t)(((uint8_t)(x)) << MCG_C6_CME0_SHIFT)) & MCG_
#define MCG_C6_PLLS_MASK                      (0x40U)
#define MCG_C6_PLLS_SHIFT                     (6U)
#define MCG_C6_PLLS(x)                        (((uint8_t)(((uint8_t)(x)) << MCG_C6_PLLS_SHIFT)) & MCG_C
#define MCG_C6_LOLIE0_MASK                    (0x80U)
#define MCG_C6_LOLIE0_SHIFT                   (7U)
#define MCG_C6_LOLIE0(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_C6_LOLIE0_SHIFT)) & MCG

/*! @name S - MCG Status Register */
#define MCG_S_IRCST_MASK                      (0x1U)
```

```
#define MCG_S_IRCST_SHIFT                       (0U)
#define MCG_S_IRCST(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_S_IRCST_SHIFT)) & MCG_S
#define MCG_S_OSCINIT0_MASK                     (0x2U)
#define MCG_S_OSCINIT0_SHIFT                    (1U)
#define MCG_S_OSCINIT0(x)                       (((uint8_t)(((uint8_t)(x)) << MCG_S_OSCINIT0_SHIFT)) & M
#define MCG_S_CLKST_MASK                        (0xCU)
#define MCG_S_CLKST_SHIFT                       (2U)
#define MCG_S_CLKST(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_S_CLKST_SHIFT)) & MCG_
#define MCG_S_IREFST_MASK                       (0x10U)
#define MCG_S_IREFST_SHIFT                      (4U)
#define MCG_S_IREFST(x)                         (((uint8_t)(((uint8_t)(x)) << MCG_S_IREFST_SHIFT)) & MCG_
#define MCG_S_PLLST_MASK                        (0x20U)
#define MCG_S_PLLST_SHIFT                       (5U)
#define MCG_S_PLLST(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_S_PLLST_SHIFT)) & MCG_S
#define MCG_S_LOCK0_MASK                        (0x40U)
#define MCG_S_LOCK0_SHIFT                       (6U)
#define MCG_S_LOCK0(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_S_LOCK0_SHIFT)) & MCG_
#define MCG_S_LOLS0_MASK                        (0x80U)
#define MCG_S_LOLS0_SHIFT                       (7U)
#define MCG_S_LOLS0(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_S_LOLS0_SHIFT)) & MCG_S

/*! @name SC - MCG Status and Control Register */
#define MCG_SC_LOCS0_MASK                       (0x1U)
#define MCG_SC_LOCS0_SHIFT                      (0U)
#define MCG_SC_LOCS0(x)                         (((uint8_t)(((uint8_t)(x)) << MCG_SC_LOCS0_SHIFT)) & MC
#define MCG_SC_FCRDIV_MASK                      (0xEU)
#define MCG_SC_FCRDIV_SHIFT                     (1U)
#define MCG_SC_FCRDIV(x)                        (((uint8_t)(((uint8_t)(x)) << MCG_SC_FCRDIV_SHIFT)) & M
#define MCG_SC_FLTPRSRV_MASK                    (0x10U)
#define MCG_SC_FLTPRSRV_SHIFT                   (4U)
#define MCG_SC_FLTPRSRV(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_SC_FLTPRSRV_SHIFT)
#define MCG_SC_ATMF_MASK                        (0x20U)
#define MCG_SC_ATMF_SHIFT                       (5U)
#define MCG_SC_ATMF(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_SC_ATMF_SHIFT)) & MCG_
#define MCG_SC_ATMS_MASK                        (0x40U)
#define MCG_SC_ATMS_SHIFT                       (6U)
#define MCG_SC_ATMS(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_SC_ATMS_SHIFT)) & MCG_
#define MCG_SC_ATME_MASK                        (0x80U)
#define MCG_SC_ATME_SHIFT                       (7U)
#define MCG_SC_ATME(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_SC_ATME_SHIFT)) & MCG_

/*! @name ATCVH - MCG Auto Trim Compare Value High Register */
#define MCG_ATCVH_ATCVH_MASK                    (0xFFU)
#define MCG_ATCVH_ATCVH_SHIFT                   (0U)
#define MCG_ATCVH_ATCVH(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_ATCVH_ATCVH_SHIFT)

/*! @name ATCVL - MCG Auto Trim Compare Value Low Register */
#define MCG_ATCVL_ATCVL_MASK                    (0xFFU)
#define MCG_ATCVL_ATCVL_SHIFT                   (0U)
#define MCG_ATCVL_ATCVL(x)                      (((uint8_t)(((uint8_t)(x)) << MCG_ATCVL_ATCVL_SHIFT))

/*! @name C8 - MCG Control 8 Register */
#define MCG_C8_LOLRE_MASK                       (0x40U)
```

```
#define MCG_C8_LOLRE_SHIFT                       (6U)
#define MCG_C8_LOLRE(x)                          (((uint8_t)(((uint8_t)(x)) << MCG_C8_LOLRE_SHIFT)) & MC


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the MCG peripheral type */
#define MCG_IRQS                                 { MCG_IRQn }
/* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup MCM_Register_Masks MCM Register Masks
 * @{
 */

/*! @name PLASC - Crossbar Switch (AXBS) Slave Configuration */
#define MCM_PLASC_ASC_MASK                       (0xFFU)
#define MCM_PLASC_ASC_SHIFT                      (0U)
#define MCM_PLASC_ASC(x)                         (((uint16_t)(((uint16_t)(x)) << MCM_PLASC_ASC_SHIFT)) &

/*! @name PLAMC - Crossbar Switch (AXBS) Master Configuration */
#define MCM_PLAMC_AMC_MASK                       (0xFFU)
#define MCM_PLAMC_AMC_SHIFT                      (0U)
#define MCM_PLAMC_AMC(x)                         (((uint16_t)(((uint16_t)(x)) << MCM_PLAMC_AMC_SHIFT))

/*! @name PLACR - Platform Control Register */
#define MCM_PLACR_ARB_MASK                       (0x200U)
#define MCM_PLACR_ARB_SHIFT                      (9U)
#define MCM_PLACR_ARB(x)                         (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_ARB_SHIFT)) &
#define MCM_PLACR_CFCC_MASK                      (0x400U)
#define MCM_PLACR_CFCC_SHIFT                     (10U)
#define MCM_PLACR_CFCC(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_CFCC_SHIFT
#define MCM_PLACR_DFCDA_MASK                     (0x800U)
#define MCM_PLACR_DFCDA_SHIFT                    (11U)
#define MCM_PLACR_DFCDA(x)                       (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_DFCDA_SHI
#define MCM_PLACR_DFCIC_MASK                     (0x1000U)
#define MCM_PLACR_DFCIC_SHIFT                    (12U)
#define MCM_PLACR_DFCIC(x)                       (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_DFCIC_SHIFT
#define MCM_PLACR_DFCC_MASK                      (0x2000U)
#define MCM_PLACR_DFCC_SHIFT                     (13U)
#define MCM_PLACR_DFCC(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_DFCC_SHIFT
#define MCM_PLACR_EFDS_MASK                      (0x4000U)
#define MCM_PLACR_EFDS_SHIFT                     (14U)
#define MCM_PLACR_EFDS(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_EFDS_SHIFT)
#define MCM_PLACR_DFCS_MASK                      (0x8000U)
#define MCM_PLACR_DFCS_SHIFT                     (15U)
#define MCM_PLACR_DFCS(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_DFCS_SHIFT)
#define MCM_PLACR_ESFC_MASK                      (0x10000U)
#define MCM_PLACR_ESFC_SHIFT                     (16U)
```

```
#define MCM_PLACR_ESFC(x)                       (((uint32_t)(((uint32_t)(x)) << MCM_PLACR_ESFC_SHIFT)

/*! @name CPO - Compute Operation Control Register */
#define MCM_CPO_CPOREQ_MASK                      (0x1U)
#define MCM_CPO_CPOREQ_SHIFT                     (0U)
#define MCM_CPO_CPOREQ(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_CPO_CPOREQ_SHIF
#define MCM_CPO_CPOACK_MASK                      (0x2U)
#define MCM_CPO_CPOACK_SHIFT                     (1U)
#define MCM_CPO_CPOACK(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_CPO_CPOACK_SHIF
#define MCM_CPO_CPOWOI_MASK                      (0x4U)
#define MCM_CPO_CPOWOI_SHIFT                     (2U)
#define MCM_CPO_CPOWOI(x)                        (((uint32_t)(((uint32_t)(x)) << MCM_CPO_CPOWOI_SHIF


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 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
} MTB_Type;

/* ----------------------------------------------------------------------------
   -- MTB Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup MTB_Register_Masks MTB Register Masks
 * @{
 */

/*! @name POSITION - MTB Position Register */
#define MTB_POSITION_WRAP_MASK                   (0x4U)
#define MTB_POSITION_WRAP_SHIFT                  (2U)
#define MTB_POSITION_WRAP(x)                     (((uint32_t)(((uint32_t)(x)) << MTB_POSITION_WRAP_SH
#define MTB_POSITION_POINTER_MASK                (0xFFFFFFF8U)
#define MTB_POSITION_POINTER_SHIFT               (3U)
#define MTB_POSITION_POINTER(x)                  (((uint32_t)(((uint32_t)(x)) << MTB_POSITION_POINTE

/*! @name MASTER - MTB Master Register */
#define MTB_MASTER_MASK_MASK                     (0x1FU)
#define MTB_MASTER_MASK_SHIFT                    (0U)
#define MTB_MASTER_MASK(x)                       (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_MASK_SHIF
#define MTB_MASTER_TSTARTEN_MASK                 (0x20U)
#define MTB_MASTER_TSTARTEN_SHIFT                (5U)
#define MTB_MASTER_TSTARTEN(x)                   (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_TSTART
#define MTB_MASTER_TSTOPEN_MASK                  (0x40U)
#define MTB_MASTER_TSTOPEN_SHIFT                 (6U)
#define MTB_MASTER_TSTOPEN(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_TSTOPEN
#define MTB_MASTER_SFRWPRIV_MASK                 (0x80U)
#define MTB_MASTER_SFRWPRIV_SHIFT                (7U)
#define MTB_MASTER_SFRWPRIV(x)                   (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_SFRWPR
#define MTB_MASTER_RAMPRIV_MASK                  (0x100U)
#define MTB_MASTER_RAMPRIV_SHIFT                 (8U)
#define MTB_MASTER_RAMPRIV(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_RAMPRIV
#define MTB_MASTER_HALTREQ_MASK                 (0x200U)
#define MTB_MASTER_HALTREQ_SHIFT                (9U)
#define MTB_MASTER_HALTREQ(x)                   (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_HALTREQ
#define MTB_MASTER_EN_MASK                      (0x80000000U)
#define MTB_MASTER_EN_SHIFT                     (31U)
#define MTB_MASTER_EN(x)                        (((uint32_t)(((uint32_t)(x)) << MTB_MASTER_EN_SHIFT)) &

/*! @name FLOW - MTB Flow Register */
#define MTB_FLOW_AUTOSTOP_MASK                  (0x1U)
#define MTB_FLOW_AUTOSTOP_SHIFT                 (0U)
#define MTB_FLOW_AUTOSTOP(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_FLOW_AUTOSTOP_
#define MTB_FLOW_AUTOHALT_MASK                  (0x2U)
#define MTB_FLOW_AUTOHALT_SHIFT                 (1U)
#define MTB_FLOW_AUTOHALT(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_FLOW_AUTOHALT_
#define MTB_FLOW_WATERMARK_MASK                 (0xFFFFFFF8U)
#define MTB_FLOW_WATERMARK_SHIFT                (3U)
#define MTB_FLOW_WATERMARK(x)                   (((uint32_t)(((uint32_t)(x)) << MTB_FLOW_WATERMA

/*! @name BASE - MTB Base Register */
#define MTB_BASE_BASEADDR_MASK                  (0xFFFFFFFFU)
#define MTB_BASE_BASEADDR_SHIFT                 (0U)
#define MTB_BASE_BASEADDR(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_BASE_BASEADDR_

/*! @name MODECTRL - Integration Mode Control Register */
#define MTB_MODECTRL_MODECTRL_MASK              (0xFFFFFFFFU)
#define MTB_MODECTRL_MODECTRL_SHIFT             (0U)
#define MTB_MODECTRL_MODECTRL(x)                (((uint32_t)(((uint32_t)(x)) << MTB_MODECTRL_MO

/*! @name TAGSET - Claim TAG Set Register */
#define MTB_TAGSET_TAGSET_MASK                  (0xFFFFFFFFU)
#define MTB_TAGSET_TAGSET_SHIFT                 (0U)
#define MTB_TAGSET_TAGSET(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_TAGSET_TAGSET_S

/*! @name TAGCLEAR - Claim TAG Clear Register */
#define MTB_TAGCLEAR_TAGCLEAR_MASK              (0xFFFFFFFFU)
#define MTB_TAGCLEAR_TAGCLEAR_SHIFT             (0U)
#define MTB_TAGCLEAR_TAGCLEAR(x)                (((uint32_t)(((uint32_t)(x)) << MTB_TAGCLEAR_TAG

/*! @name LOCKACCESS - Lock Access Register */
#define MTB_LOCKACCESS_LOCKACCESS_MASK          (0xFFFFFFFFU)
#define MTB_LOCKACCESS_LOCKACCESS_SHIFT         (0U)
#define MTB_LOCKACCESS_LOCKACCESS(x)            (((uint32_t)(((uint32_t)(x)) << MTB_LOCKACCES

/*! @name LOCKSTAT - Lock Status Register */
#define MTB_LOCKSTAT_LOCKSTAT_MASK              (0xFFFFFFFFU)
#define MTB_LOCKSTAT_LOCKSTAT_SHIFT             (0U)
#define MTB_LOCKSTAT_LOCKSTAT(x)                (((uint32_t)(((uint32_t)(x)) << MTB_LOCKSTAT_LOCK

/*! @name AUTHSTAT - Authentication Status Register */
#define MTB_AUTHSTAT_BIT0_MASK                  (0x1U)
#define MTB_AUTHSTAT_BIT0_SHIFT                 (0U)
#define MTB_AUTHSTAT_BIT0(x)                    (((uint32_t)(((uint32_t)(x)) << MTB_AUTHSTAT_BIT0_SHI
#define MTB_AUTHSTAT_BIT1_MASK                  (0x2U)
```

```
#define MTB_AUTHSTAT_BIT1_SHIFT                  (1U)
#define MTB_AUTHSTAT_BIT1(x)                     (((uint32_t)(((uint32_t)(x)) << MTB_AUTHSTAT_BIT1_SHI
#define MTB_AUTHSTAT_BIT2_MASK                   (0x4U)
#define MTB_AUTHSTAT_BIT2_SHIFT                  (2U)
#define MTB_AUTHSTAT_BIT2(x)                     (((uint32_t)(((uint32_t)(x)) << MTB_AUTHSTAT_BIT2_SHI
#define MTB_AUTHSTAT_BIT3_MASK                   (0x8U)
#define MTB_AUTHSTAT_BIT3_SHIFT                  (3U)
#define MTB_AUTHSTAT_BIT3(x)                     (((uint32_t)(((uint32_t)(x)) << MTB_AUTHSTAT_BIT3_SHI

/*! @name DEVICEARCH - Device Architecture Register */
#define MTB_DEVICEARCH_DEVICEARCH_MASK           (0xFFFFFFFFU)
#define MTB_DEVICEARCH_DEVICEARCH_SHIFT          (0U)
#define MTB_DEVICEARCH_DEVICEARCH(x)             (((uint32_t)(((uint32_t)(x)) << MTB_DEVICEARCH_

/*! @name DEVICECFG - Device Configuration Register */
#define MTB_DEVICECFG_DEVICECFG_MASK             (0xFFFFFFFFU)
#define MTB_DEVICECFG_DEVICECFG_SHIFT            (0U)
#define MTB_DEVICECFG_DEVICECFG(x)               (((uint32_t)(((uint32_t)(x)) << MTB_DEVICECFG_DE

/*! @name DEVICETYPID - Device Type Identifier Register */
#define MTB_DEVICETYPID_DEVICETYPID_MASK         (0xFFFFFFFFU)
#define MTB_DEVICETYPID_DEVICETYPID_SHIFT        (0U)
#define MTB_DEVICETYPID_DEVICETYPID(x)           (((uint32_t)(((uint32_t)(x)) << MTB_DEVICETYPID_

/*! @name PERIPHID4 - Peripheral ID Register */
#define MTB_PERIPHID4_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID4_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID4_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID4_PERIPH

/*! @name PERIPHID5 - Peripheral ID Register */
#define MTB_PERIPHID5_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID5_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID5_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID5_PERIPH

/*! @name PERIPHID6 - Peripheral ID Register */
#define MTB_PERIPHID6_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID6_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID6_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID6_PERIPH

/*! @name PERIPHID7 - Peripheral ID Register */
#define MTB_PERIPHID7_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID7_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID7_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID7_PERIPH

/*! @name PERIPHID0 - Peripheral ID Register */
#define MTB_PERIPHID0_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID0_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID0_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID0_PERIPH

/*! @name PERIPHID1 - Peripheral ID Register */
#define MTB_PERIPHID1_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID1_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID1_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID1_PERIPH
/*! @name PERIPHID2 - Peripheral ID Register */
#define MTB_PERIPHID2_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID2_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID2_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID2_PERIPI

/*! @name PERIPHID3 - Peripheral ID Register */
#define MTB_PERIPHID3_PERIPHID_MASK              (0xFFFFFFFFU)
#define MTB_PERIPHID3_PERIPHID_SHIFT             (0U)
#define MTB_PERIPHID3_PERIPHID(x)                (((uint32_t)(((uint32_t)(x)) << MTB_PERIPHID3_PERIPI

/*! @name COMPID - Component ID Register */
#define MTB_COMPID_COMPID_MASK                   (0xFFFFFFFFU)
#define MTB_COMPID_COMPID_SHIFT                  (0U)
#define MTB_COMPID_COMPID(x)                     (((uint32_t)(((uint32_t)(x)) << MTB_COMPID_COMPID_S

/* The count of MTB_COMPID */
#define MTB_COMPID_COUNT                         (4U)


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 s
    __IO uint32_t MASK;                           /**< MTB_DWT Comparator Mask Register, array offset: 0x24, a
    __IO uint32_t FCT;                            /**< MTB_DWT Comparator Function Register 0..MTB_DWT Cor
         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 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
} MTBDWT_Type;

/* ----------------------------------------------------------------------------
   -- MTBDWT Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup MTBDWT_Register_Masks MTBDWT Register Masks
 * @{
 */

/*! @name CTRL - MTB DWT Control Register */
#define MTBDWT_CTRL_DWTCFGCTRL_MASK              (0xFFFFFFFU)
#define MTBDWT_CTRL_DWTCFGCTRL_SHIFT             (0U)
#define MTBDWT_CTRL_DWTCFGCTRL(x)                (((uint32_t)(((uint32_t)(x)) << MTBDWT_CTRL_DW
#define MTBDWT_CTRL_NUMCMP_MASK                  (0xF0000000U)
#define MTBDWT_CTRL_NUMCMP_SHIFT                 (28U)
#define MTBDWT_CTRL_NUMCMP(x)                    (((uint32_t)(((uint32_t)(x)) << MTBDWT_CTRL_NUMC

/*! @name COMP - MTB_DWT Comparator Register */
#define MTBDWT_COMP_COMP_MASK                    (0xFFFFFFFFU)
#define MTBDWT_COMP_COMP_SHIFT                   (0U)
#define MTBDWT_COMP_COMP(x)                      (((uint32_t)(((uint32_t)(x)) << MTBDWT_COMP_COMP_

/* The count of MTBDWT_COMP */
#define MTBDWT_COMP_COUNT                        (2U)

/*! @name MASK - MTB_DWT Comparator Mask Register */
#define MTBDWT_MASK_MASK_MASK                    (0x1FU)
#define MTBDWT_MASK_MASK_SHIFT                   (0U)
#define MTBDWT_MASK_MASK(x)                      (((uint32_t)(((uint32_t)(x)) << MTBDWT_MASK_MASK_S

/* The count of MTBDWT_MASK */
```

```
#define MTBDWT_MASK_COUNT                        (2U)

/*! @name FCT - MTB_DWT Comparator Function Register 0..MTB_DWT Comparator Function Register 1
#define MTBDWT_FCT_FUNCTION_MASK                 (0xFU)
#define MTBDWT_FCT_FUNCTION_SHIFT                (0U)
#define MTBDWT_FCT_FUNCTION(x)                   (((uint32_t)(((uint32_t)(x)) << MTBDWT_FCT_FUNCTIO
#define MTBDWT_FCT_DATAVMATCH_MASK               (0x100U)
#define MTBDWT_FCT_DATAVMATCH_SHIFT              (8U)
#define MTBDWT_FCT_DATAVMATCH(x)                 (((uint32_t)(((uint32_t)(x)) << MTBDWT_FCT_DATAV
#define MTBDWT_FCT_DATAVSIZE_MASK                (0xC00U)
#define MTBDWT_FCT_DATAVSIZE_SHIFT               (10U)
#define MTBDWT_FCT_DATAVSIZE(x)                  (((uint32_t)(((uint32_t)(x)) << MTBDWT_FCT_DATAVS
#define MTBDWT_FCT_DATAVADDR0_MASK               (0xF000U)
#define MTBDWT_FCT_DATAVADDR0_SHIFT              (12U)
#define MTBDWT_FCT_DATAVADDR0(x)                 (((uint32_t)(((uint32_t)(x)) << MTBDWT_FCT_DATAV
#define MTBDWT_FCT_MATCHED_MASK                  (0x1000000U)
#define MTBDWT_FCT_MATCHED_SHIFT                 (24U)
#define MTBDWT_FCT_MATCHED(x)                    (((uint32_t)(((uint32_t)(x)) << MTBDWT_FCT_MATCHE

/* The count of MTBDWT_FCT */
#define MTBDWT_FCT_COUNT                         (2U)

/*! @name TBCTRL - MTB_DWT Trace Buffer Control Register */
#define MTBDWT_TBCTRL_ACOMP0_MASK                (0x1U)
#define MTBDWT_TBCTRL_ACOMP0_SHIFT               (0U)
#define MTBDWT_TBCTRL_ACOMP0(x)                  (((uint32_t)(((uint32_t)(x)) << MTBDWT_TBCTRL_AC
#define MTBDWT_TBCTRL_ACOMP1_MASK                (0x2U)
#define MTBDWT_TBCTRL_ACOMP1_SHIFT               (1U)
#define MTBDWT_TBCTRL_ACOMP1(x)                  (((uint32_t)(((uint32_t)(x)) << MTBDWT_TBCTRL_AC
#define MTBDWT_TBCTRL_NUMCOMP_MASK               (0xF0000000U)
#define MTBDWT_TBCTRL_NUMCOMP_SHIFT              (28U)
#define MTBDWT_TBCTRL_NUMCOMP(x)                 (((uint32_t)(((uint32_t)(x)) << MTBDWT_TBCTRL_N

/*! @name DEVICECFG - Device Configuration Register */
#define MTBDWT_DEVICECFG_DEVICECFG_MASK          (0xFFFFFFFFU)
#define MTBDWT_DEVICECFG_DEVICECFG_SHIFT         (0U)
#define MTBDWT_DEVICECFG_DEVICECFG(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_DEVICEC

/*! @name DEVICETYPID - Device Type Identifier Register */
#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK      (0xFFFFFFFFU)
#define MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT     (0U)
#define MTBDWT_DEVICETYPID_DEVICETYPID(x)        (((uint32_t)(((uint32_t)(x)) << MTBDWT_DEVICE

/*! @name PERIPHID4 - Peripheral ID Register */
#define MTBDWT_PERIPHID4_PERIPHID_MASK           (0xFFFFFFFFU)
#define MTBDWT_PERIPHID4_PERIPHID_SHIFT          (0U)
#define MTBDWT_PERIPHID4_PERIPHID(x)             (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID4

/*! @name PERIPHID5 - Peripheral ID Register */
#define MTBDWT_PERIPHID5_PERIPHID_MASK           (0xFFFFFFFFU)
#define MTBDWT_PERIPHID5_PERIPHID_SHIFT          (0U)
#define MTBDWT_PERIPHID5_PERIPHID(x)             (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID5
/*! @name PERIPHID6 - Peripheral ID Register */
#define MTBDWT_PERIPHID6_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID6_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID6_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID6

/*! @name PERIPHID7 - Peripheral ID Register */
#define MTBDWT_PERIPHID7_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID7_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID7_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID7

/*! @name PERIPHID0 - Peripheral ID Register */
#define MTBDWT_PERIPHID0_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID0_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID0_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID0

/*! @name PERIPHID1 - Peripheral ID Register */
#define MTBDWT_PERIPHID1_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID1_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID1_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID1

/*! @name PERIPHID2 - Peripheral ID Register */
#define MTBDWT_PERIPHID2_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID2_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID2_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID2

/*! @name PERIPHID3 - Peripheral ID Register */
#define MTBDWT_PERIPHID3_PERIPHID_MASK          (0xFFFFFFFFU)
#define MTBDWT_PERIPHID3_PERIPHID_SHIFT         (0U)
#define MTBDWT_PERIPHID3_PERIPHID(x)            (((uint32_t)(((uint32_t)(x)) << MTBDWT_PERIPHID3

/*! @name COMPID - Component ID Register */
#define MTBDWT_COMPID_COMPID_MASK               (0xFFFFFFFFU)
#define MTBDWT_COMPID_COMPID_SHIFT              (0U)
#define MTBDWT_COMPID_COMPID(x)                 (((uint32_t)(((uint32_t)(x)) << MTBDWT_COMPID_CO

/* The count of MTBDWT_COMPID */
#define MTBDWT_COMPID_COUNT                     (4U)


/*!
 * @}
 */ /* 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)
/** 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 }
```

```
/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup NV_Register_Masks NV Register Masks
 * @{
 */

/*! @name BACKKEY3 - Backdoor Comparison Key 3. */
#define NV_BACKKEY3_KEY_MASK                     (0xFFU)
#define NV_BACKKEY3_KEY_SHIFT                    (0U)
#define NV_BACKKEY3_KEY(x)                       (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY3_KEY_SHIFT))

/*! @name BACKKEY2 - Backdoor Comparison Key 2. */
#define NV_BACKKEY2_KEY_MASK                     (0xFFU)
#define NV_BACKKEY2_KEY_SHIFT                    (0U)
#define NV_BACKKEY2_KEY(x)                       (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY2_KEY_SHIFT))

/*! @name BACKKEY1 - Backdoor Comparison Key 1. */
```

```
#define NV_BACKKEY1_KEY_MASK                    (0xFFU)
#define NV_BACKKEY1_KEY_SHIFT                  (0U)
#define NV_BACKKEY1_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY1_KEY_SHIFT))

/*! @name BACKKEY0 - Backdoor Comparison Key 0. */
#define NV_BACKKEY0_KEY_MASK                    (0xFFU)
#define NV_BACKKEY0_KEY_SHIFT                  (0U)
#define NV_BACKKEY0_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY0_KEY_SHIFT))

/*! @name BACKKEY7 - Backdoor Comparison Key 7. */
#define NV_BACKKEY7_KEY_MASK                    (0xFFU)
#define NV_BACKKEY7_KEY_SHIFT                  (0U)
#define NV_BACKKEY7_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY7_KEY_SHIFT))

/*! @name BACKKEY6 - Backdoor Comparison Key 6. */
#define NV_BACKKEY6_KEY_MASK                    (0xFFU)
#define NV_BACKKEY6_KEY_SHIFT                  (0U)
#define NV_BACKKEY6_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY6_KEY_SHIFT))

/*! @name BACKKEY5 - Backdoor Comparison Key 5. */
#define NV_BACKKEY5_KEY_MASK                    (0xFFU)
#define NV_BACKKEY5_KEY_SHIFT                  (0U)
#define NV_BACKKEY5_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY5_KEY_SHIFT))

/*! @name BACKKEY4 - Backdoor Comparison Key 4. */
#define NV_BACKKEY4_KEY_MASK                    (0xFFU)
#define NV_BACKKEY4_KEY_SHIFT                  (0U)
#define NV_BACKKEY4_KEY(x)                     (((uint8_t)(((uint8_t)(x)) << NV_BACKKEY4_KEY_SHIFT))

/*! @name FPROT3 - Non-volatile P-Flash Protection 1 - Low Register */
#define NV_FPROT3_PROT_MASK                     (0xFFU)
#define NV_FPROT3_PROT_SHIFT                   (0U)
#define NV_FPROT3_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << NV_FPROT3_PROT_SHIFT)) &

/*! @name FPROT2 - Non-volatile P-Flash Protection 1 - High Register */
#define NV_FPROT2_PROT_MASK                     (0xFFU)
#define NV_FPROT2_PROT_SHIFT                   (0U)
#define NV_FPROT2_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << NV_FPROT2_PROT_SHIFT)) &

/*! @name FPROT1 - Non-volatile P-Flash Protection 0 - Low Register */
#define NV_FPROT1_PROT_MASK                     (0xFFU)
#define NV_FPROT1_PROT_SHIFT                   (0U)
#define NV_FPROT1_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << NV_FPROT1_PROT_SHIFT)) &

/*! @name FPROT0 - Non-volatile P-Flash Protection 0 - High Register */
#define NV_FPROT0_PROT_MASK                     (0xFFU)
#define NV_FPROT0_PROT_SHIFT                   (0U)
#define NV_FPROT0_PROT(x)                      (((uint8_t)(((uint8_t)(x)) << NV_FPROT0_PROT_SHIFT)) &

/*! @name FSEC - Non-volatile Flash Security Register */
#define NV_FSEC_SEC_MASK                        (0x3U)
#define NV_FSEC_SEC_SHIFT                      (0U)
#define NV_FSEC_SEC(x)                         (((uint8_t)(((uint8_t)(x)) << NV_FSEC_SEC_SHIFT)) & NV_FS
#define NV_FSEC_FSLACC_MASK                    (0xCU)
#define NV_FSEC_FSLACC_SHIFT                   (2U)
#define NV_FSEC_FSLACC(x)                      (((uint8_t)(((uint8_t)(x)) << NV_FSEC_FSLACC_SHIFT)) & N
#define NV_FSEC_MEEN_MASK                      (0x30U)
#define NV_FSEC_MEEN_SHIFT                     (4U)
#define NV_FSEC_MEEN(x)                        (((uint8_t)(((uint8_t)(x)) << NV_FSEC_MEEN_SHIFT)) & NV_
#define NV_FSEC_KEYEN_MASK                     (0xC0U)
#define NV_FSEC_KEYEN_SHIFT                    (6U)
#define NV_FSEC_KEYEN(x)                       (((uint8_t)(((uint8_t)(x)) << NV_FSEC_KEYEN_SHIFT)) & NV

/*! @name FOPT - Non-volatile Flash Option Register */
#define NV_FOPT_LPBOOT0_MASK                   (0x1U)
#define NV_FOPT_LPBOOT0_SHIFT                  (0U)
#define NV_FOPT_LPBOOT0(x)                     (((uint8_t)(((uint8_t)(x)) << NV_FOPT_LPBOOT0_SHIFT)) &
#define NV_FOPT_NMI_DIS_MASK                   (0x4U)
#define NV_FOPT_NMI_DIS_SHIFT                  (2U)
#define NV_FOPT_NMI_DIS(x)                     (((uint8_t)(((uint8_t)(x)) << NV_FOPT_NMI_DIS_SHIFT)) & N
#define NV_FOPT_RESET_PIN_CFG_MASK             (0x8U)
#define NV_FOPT_RESET_PIN_CFG_SHIFT            (3U)
#define NV_FOPT_RESET_PIN_CFG(x)               (((uint8_t)(((uint8_t)(x)) << NV_FOPT_RESET_PIN_CF
#define NV_FOPT_LPBOOT1_MASK                   (0x10U)
#define NV_FOPT_LPBOOT1_SHIFT                  (4U)
#define NV_FOPT_LPBOOT1(x)                     (((uint8_t)(((uint8_t)(x)) << NV_FOPT_LPBOOT1_SHIFT)) &
#define NV_FOPT_FAST_INIT_MASK                 (0x20U)
#define NV_FOPT_FAST_INIT_SHIFT                (5U)
#define NV_FOPT_FAST_INIT(x)                   (((uint8_t)(((uint8_t)(x)) << NV_FOPT_FAST_INIT_SHIFT))


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup OSC_Register_Masks OSC Register Masks
 * @{
 */

/*! @name CR - OSC Control Register */
#define OSC_CR_SC16P_MASK                        (0x1U)
#define OSC_CR_SC16P_SHIFT                       (0U)
#define OSC_CR_SC16P(x)                          (((uint8_t)(((uint8_t)(x)) << OSC_CR_SC16P_SHIFT)) & OSC
#define OSC_CR_SC8P_MASK                         (0x2U)
#define OSC_CR_SC8P_SHIFT                        (1U)
#define OSC_CR_SC8P(x)                           (((uint8_t)(((uint8_t)(x)) << OSC_CR_SC8P_SHIFT)) & OSC_
#define OSC_CR_SC4P_MASK                         (0x4U)
#define OSC_CR_SC4P_SHIFT                        (2U)
#define OSC_CR_SC4P(x)                           (((uint8_t)(((uint8_t)(x)) << OSC_CR_SC4P_SHIFT)) & OSC_
#define OSC_CR_SC2P_MASK                         (0x8U)
#define OSC_CR_SC2P_SHIFT                        (3U)
#define OSC_CR_SC2P(x)                           (((uint8_t)(((uint8_t)(x)) << OSC_CR_SC2P_SHIFT)) & OSC_
#define OSC_CR_EREFSTEN_MASK                     (0x20U)
#define OSC_CR_EREFSTEN_SHIFT                    (5U)
#define OSC_CR_EREFSTEN(x)                       (((uint8_t)(((uint8_t)(x)) << OSC_CR_EREFSTEN_SHIFT)
#define OSC_CR_ERCLKEN_MASK                      (0x80U)
#define OSC_CR_ERCLKEN_SHIFT                     (7U)
#define OSC_CR_ERCLKEN(x)                        (((uint8_t)(((uint8_t)(x)) << OSC_CR_ERCLKEN_SHIFT)) &


/*!
 * @}
 */ /* 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 *)OSC0_BASE)
/** 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 }
```

```
/*!
 * @}
 */ /* 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:
    __I  uint32_t CVAL;                              /**< Current Timer Value Register, array offset: 0x104, array step:
    __IO uint32_t TCTRL;                             /**< Timer Control Register, array offset: 0x108, array step: 0x1
    __IO uint32_t TFLG;                              /**< Timer Flag Register, array offset: 0x10C, array step: 0x10 */
  } CHANNEL[2];
} PIT_Type;

/* ----------------------------------------------------------------------------
   -- PIT Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup PIT_Register_Masks PIT Register Masks
 * @{
 */

/*! @name MCR - PIT Module Control Register */
#define PIT_MCR_FRZ_MASK                         (0x1U)
#define PIT_MCR_FRZ_SHIFT                        (0U)
#define PIT_MCR_FRZ(x)                           (((uint32_t)(((uint32_t)(x)) << PIT_MCR_FRZ_SHIFT)) & PIT_M
#define PIT_MCR_MDIS_MASK                        (0x2U)
#define PIT_MCR_MDIS_SHIFT                       (1U)
#define PIT_MCR_MDIS(x)                          (((uint32_t)(((uint32_t)(x)) << PIT_MCR_MDIS_SHIFT)) & PIT_

/*! @name LTMR64H - PIT Upper Lifetime Timer Register */
#define PIT_LTMR64H_LTH_MASK                     (0xFFFFFFFFU)
#define PIT_LTMR64H_LTH_SHIFT                    (0U)
#define PIT_LTMR64H_LTH(x)                       (((uint32_t)(((uint32_t)(x)) << PIT_LTMR64H_LTH_SHIFT)) &

/*! @name LTMR64L - PIT Lower Lifetime Timer Register */
```

```
#define PIT_LTMR64L_LTL_MASK                   (0xFFFFFFFFU)
#define PIT_LTMR64L_LTL_SHIFT                  (0U)
#define PIT_LTMR64L_LTL(x)                     (((uint32_t)(((uint32_t)(x)) << PIT_LTMR64L_LTL_SHIFT)) &

/*! @name LDVAL - Timer Load Value Register */
#define PIT_LDVAL_TSV_MASK                     (0xFFFFFFFFU)
#define PIT_LDVAL_TSV_SHIFT                    (0U)
#define PIT_LDVAL_TSV(x)                       (((uint32_t)(((uint32_t)(x)) << PIT_LDVAL_TSV_SHIFT)) & PIT

/* The count of PIT_LDVAL */
#define PIT_LDVAL_COUNT                        (2U)

/*! @name CVAL - Current Timer Value Register */
#define PIT_CVAL_TVL_MASK                      (0xFFFFFFFFU)
#define PIT_CVAL_TVL_SHIFT                     (0U)
#define PIT_CVAL_TVL(x)                        (((uint32_t)(((uint32_t)(x)) << PIT_CVAL_TVL_SHIFT)) & PIT_C

/* The count of PIT_CVAL */
#define PIT_CVAL_COUNT                         (2U)

/*! @name TCTRL - Timer Control Register */
#define PIT_TCTRL_TEN_MASK                     (0x1U)
#define PIT_TCTRL_TEN_SHIFT                    (0U)
#define PIT_TCTRL_TEN(x)                       (((uint32_t)(((uint32_t)(x)) << PIT_TCTRL_TEN_SHIFT)) & PIT
#define PIT_TCTRL_TIE_MASK                     (0x2U)
#define PIT_TCTRL_TIE_SHIFT                    (1U)
#define PIT_TCTRL_TIE(x)                       (((uint32_t)(((uint32_t)(x)) << PIT_TCTRL_TIE_SHIFT)) & PIT_
#define PIT_TCTRL_CHN_MASK                     (0x4U)
#define PIT_TCTRL_CHN_SHIFT                    (2U)
#define PIT_TCTRL_CHN(x)                       (((uint32_t)(((uint32_t)(x)) << PIT_TCTRL_CHN_SHIFT)) & PI

/* The count of PIT_TCTRL */
#define PIT_TCTRL_COUNT                        (2U)

/*! @name TFLG - Timer Flag Register */
#define PIT_TFLG_TIF_MASK                      (0x1U)
#define PIT_TFLG_TIF_SHIFT                     (0U)
#define PIT_TFLG_TIF(x)                        (((uint32_t)(((uint32_t)(x)) << PIT_TFLG_TIF_SHIFT)) & PIT_TF

/* The count of PIT_TFLG */
#define PIT_TFLG_COUNT                         (2U)


/*!
 * @}
 */ /* 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)
```

```
/** 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 }
/** Interrupt vectors for the PIT peripheral type */
#define PIT_IRQS                                 { { PIT_IRQn, PIT_IRQn } }

/*!
 * @}
 */ /* 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: 0x
  __IO uint8_t LVDSC2;                             /**< Low Voltage Detect Status And Control 2 register, offset: 0x
  __IO uint8_t REGSC;                              /**< Regulator Status And Control register, offset: 0x2 */
} PMC_Type;

/* ----------------------------------------------------------------------------
   -- PMC Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup PMC_Register_Masks PMC Register Masks
 * @{
 */

/*! @name LVDSC1 - Low Voltage Detect Status And Control 1 register */
#define PMC_LVDSC1_LVDV_MASK                     (0x3U)
#define PMC_LVDSC1_LVDV_SHIFT                    (0U)
#define PMC_LVDSC1_LVDV(x)                       (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC1_LVDV_SHIFT))
#define PMC_LVDSC1_LVDRE_MASK                    (0x10U)
#define PMC_LVDSC1_LVDRE_SHIFT                   (4U)
#define PMC_LVDSC1_LVDRE(x)                      (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC1_LVDRE_SHIF
#define PMC_LVDSC1_LVDIE_MASK                    (0x20U)
#define PMC_LVDSC1_LVDIE_SHIFT                   (5U)
#define PMC_LVDSC1_LVDIE(x)                      (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC1_LVDIE_SHIFT))
#define PMC_LVDSC1_LVDACK_MASK                   (0x40U)
#define PMC_LVDSC1_LVDACK_SHIFT                  (6U)
#define PMC_LVDSC1_LVDACK(x)                     (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC1_LVDACK_SH
#define PMC_LVDSC1_LVDF_MASK                     (0x80U)
#define PMC_LVDSC1_LVDF_SHIFT                    (7U)
#define PMC_LVDSC1_LVDF(x)                       (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC1_LVDF_SHIFT))
/*! @name LVDSC2 - Low Voltage Detect Status And Control 2 register */
#define PMC_LVDSC2_LVWV_MASK                    (0x3U)
#define PMC_LVDSC2_LVWV_SHIFT                   (0U)
#define PMC_LVDSC2_LVWV(x)                      (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC2_LVWV_SHIFT)
#define PMC_LVDSC2_LVWIE_MASK                   (0x20U)
#define PMC_LVDSC2_LVWIE_SHIFT                  (5U)
#define PMC_LVDSC2_LVWIE(x)                     (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC2_LVWIE_SHIFT
#define PMC_LVDSC2_LVWACK_MASK                  (0x40U)
#define PMC_LVDSC2_LVWACK_SHIFT                 (6U)
#define PMC_LVDSC2_LVWACK(x)                    (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC2_LVWACK_SH
#define PMC_LVDSC2_LVWF_MASK                    (0x80U)
#define PMC_LVDSC2_LVWF_SHIFT                   (7U)
#define PMC_LVDSC2_LVWF(x)                      (((uint8_t)(((uint8_t)(x)) << PMC_LVDSC2_LVWF_SHIFT))

/*! @name REGSC - Regulator Status And Control register */
#define PMC_REGSC_BGBE_MASK                     (0x1U)
#define PMC_REGSC_BGBE_SHIFT                    (0U)
#define PMC_REGSC_BGBE(x)                       (((uint8_t)(((uint8_t)(x)) << PMC_REGSC_BGBE_SHIFT))
#define PMC_REGSC_REGONS_MASK                   (0x4U)
#define PMC_REGSC_REGONS_SHIFT                  (2U)
#define PMC_REGSC_REGONS(x)                     (((uint8_t)(((uint8_t)(x)) << PMC_REGSC_REGONS_SH
#define PMC_REGSC_ACKISO_MASK                   (0x8U)
#define PMC_REGSC_ACKISO_SHIFT                  (3U)
#define PMC_REGSC_ACKISO(x)                     (((uint8_t)(((uint8_t)(x)) << PMC_REGSC_ACKISO_SHIF
#define PMC_REGSC_BGEN_MASK                     (0x10U)
#define PMC_REGSC_BGEN_SHIFT                    (4U)
#define PMC_REGSC_BGEN(x)                       (((uint8_t)(((uint8_t)(x)) << PMC_REGSC_BGEN_SHIFT))


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the PMC peripheral type */
#define PMC_IRQS                                 { LVD_LVW_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup PORT_Register_Masks PORT Register Masks
 * @{
 */

/*! @name PCR - Pin Control Register n */
#define PORT_PCR_PS_MASK                         (0x1U)
#define PORT_PCR_PS_SHIFT                        (0U)
#define PORT_PCR_PS(x)                           (((uint32_t)(((uint32_t)(x)) << PORT_PCR_PS_SHIFT)) & POR
#define PORT_PCR_PE_MASK                         (0x2U)
#define PORT_PCR_PE_SHIFT                        (1U)
#define PORT_PCR_PE(x)                           (((uint32_t)(((uint32_t)(x)) << PORT_PCR_PE_SHIFT)) & POR
#define PORT_PCR_SRE_MASK                        (0x4U)
#define PORT_PCR_SRE_SHIFT                       (2U)
#define PORT_PCR_SRE(x)                          (((uint32_t)(((uint32_t)(x)) << PORT_PCR_SRE_SHIFT)) & P
#define PORT_PCR_PFE_MASK                        (0x10U)
#define PORT_PCR_PFE_SHIFT                       (4U)
#define PORT_PCR_PFE(x)                          (((uint32_t)(((uint32_t)(x)) << PORT_PCR_PFE_SHIFT)) & P
#define PORT_PCR_DSE_MASK                        (0x40U)
#define PORT_PCR_DSE_SHIFT                       (6U)
#define PORT_PCR_DSE(x)                          (((uint32_t)(((uint32_t)(x)) << PORT_PCR_DSE_SHIFT)) & P
#define PORT_PCR_MUX_MASK                        (0x700U)
#define PORT_PCR_MUX_SHIFT                       (8U)
#define PORT_PCR_MUX(x)                          (((uint32_t)(((uint32_t)(x)) << PORT_PCR_MUX_SHIFT)) & P
#define PORT_PCR_IRQC_MASK                       (0xF0000U)
#define PORT_PCR_IRQC_SHIFT                      (16U)
#define PORT_PCR_IRQC(x)                         (((uint32_t)(((uint32_t)(x)) << PORT_PCR_IRQC_SHIFT)) &
#define PORT_PCR_ISF_MASK                        (0x1000000U)
#define PORT_PCR_ISF_SHIFT                       (24U)
#define PORT_PCR_ISF(x)                          (((uint32_t)(((uint32_t)(x)) << PORT_PCR_ISF_SHIFT)) & POR

/* The count of PORT_PCR */
```

```
#define PORT_PCR_COUNT                          (32U)

/*! @name GPCLR - Global Pin Control Low Register */
#define PORT_GPCLR_GPWD_MASK                    (0xFFFFU)
#define PORT_GPCLR_GPWD_SHIFT                   (0U)
#define PORT_GPCLR_GPWD(x)                      (((uint32_t)(((uint32_t)(x)) << PORT_GPCLR_GPWD_SH
#define PORT_GPCLR_GPWE_MASK                    (0xFFFF0000U)
#define PORT_GPCLR_GPWE_SHIFT                   (16U)
#define PORT_GPCLR_GPWE(x)                      (((uint32_t)(((uint32_t)(x)) << PORT_GPCLR_GPWE_SH

/*! @name GPCHR - Global Pin Control High Register */
#define PORT_GPCHR_GPWD_MASK                    (0xFFFFU)
#define PORT_GPCHR_GPWD_SHIFT                   (0U)
#define PORT_GPCHR_GPWD(x)                      (((uint32_t)(((uint32_t)(x)) << PORT_GPCHR_GPWD_SH
#define PORT_GPCHR_GPWE_MASK                    (0xFFFF0000U)
#define PORT_GPCHR_GPWE_SHIFT                   (16U)
#define PORT_GPCHR_GPWE(x)                      (((uint32_t)(((uint32_t)(x)) << PORT_GPCHR_GPWE_SH

/*! @name ISFR - Interrupt Status Flag Register */
#define PORT_ISFR_ISF_MASK                      (0xFFFFFFFFU)
#define PORT_ISFR_ISF_SHIFT                     (0U)
#define PORT_ISFR_ISF(x)                        (((uint32_t)(((uint32_t)(x)) << PORT_ISFR_ISF_SHIFT)) & PO


/*!
 * @}
 */ /* 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)
/** Peripheral PORTB base address */
#define PORTB_BASE                               (0x4004A000u)
/** Peripheral PORTB base pointer */
#define PORTB                                    ((PORT_Type *)PORTB_BASE)
/** Peripheral PORTC base address */
#define PORTC_BASE                               (0x4004B000u)
/** Peripheral PORTC base pointer */
#define PORTC                                    ((PORT_Type *)PORTC_BASE)
/** Peripheral PORTD base address */
#define PORTD_BASE                               (0x4004C000u)
/** Peripheral PORTD base pointer */
#define PORTD                                    ((PORT_Type *)PORTD_BASE)
/** Peripheral PORTE base address */
#define PORTE_BASE                               (0x4004D000u)
/** Peripheral PORTE base pointer */
#define PORTE                                    ((PORT_Type *)PORTE_BASE)
/** Array initializer of PORT peripheral base addresses */
#define PORT_BASE_ADDRS                          { PORTA_BASE, PORTB_BASE, PORTC_BASE, PORTD_
/** Array initializer of PORT peripheral base pointers */
```

```
#define PORT_BASE_PTRS                           { PORTA, PORTB, PORTC, PORTD, PORTE }
/** Interrupt vectors for the PORT peripheral type */
#define PORT_IRQS                                { PORTA_IRQn, NotAvail_IRQn, NotAvail_IRQn, PORTD_IRQn,

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup RCM_Register_Masks RCM Register Masks
 * @{
 */

/*! @name SRS0 - System Reset Status Register 0 */
#define RCM_SRS0_WAKEUP_MASK                     (0x1U)
#define RCM_SRS0_WAKEUP_SHIFT                    (0U)
#define RCM_SRS0_WAKEUP(x)                       (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_WAKEUP_SHIFT
#define RCM_SRS0_LVD_MASK                        (0x2U)
#define RCM_SRS0_LVD_SHIFT                       (1U)
#define RCM_SRS0_LVD(x)                          (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_LVD_SHIFT)) & RCM
#define RCM_SRS0_LOC_MASK                        (0x4U)
#define RCM_SRS0_LOC_SHIFT                       (2U)
#define RCM_SRS0_LOC(x)                          (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_LOC_SHIFT)) & RCM
#define RCM_SRS0_LOL_MASK                        (0x8U)
#define RCM_SRS0_LOL_SHIFT                       (3U)
#define RCM_SRS0_LOL(x)                          (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_LOL_SHIFT)) & RCM
#define RCM_SRS0_WDOG_MASK                       (0x20U)
#define RCM_SRS0_WDOG_SHIFT                      (5U)
#define RCM_SRS0_WDOG(x)                         (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_WDOG_SHIFT)) &
#define RCM_SRS0_PIN_MASK                        (0x40U)
```

```
#define RCM_SRS0_PIN_SHIFT                       (6U)
#define RCM_SRS0_PIN(x)                          (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_PIN_SHIFT)) & RCM_
#define RCM_SRS0_POR_MASK                        (0x80U)
#define RCM_SRS0_POR_SHIFT                       (7U)
#define RCM_SRS0_POR(x)                          (((uint8_t)(((uint8_t)(x)) << RCM_SRS0_POR_SHIFT)) & RC

/*! @name SRS1 - System Reset Status Register 1 */
#define RCM_SRS1_LOCKUP_MASK                     (0x2U)
#define RCM_SRS1_LOCKUP_SHIFT                    (1U)
#define RCM_SRS1_LOCKUP(x)                       (((uint8_t)(((uint8_t)(x)) << RCM_SRS1_LOCKUP_SHIFT)
#define RCM_SRS1_SW_MASK                         (0x4U)
#define RCM_SRS1_SW_SHIFT                        (2U)
#define RCM_SRS1_SW(x)                           (((uint8_t)(((uint8_t)(x)) << RCM_SRS1_SW_SHIFT)) & RCM_
#define RCM_SRS1_MDM_AP_MASK                     (0x8U)
#define RCM_SRS1_MDM_AP_SHIFT                    (3U)
#define RCM_SRS1_MDM_AP(x)                       (((uint8_t)(((uint8_t)(x)) << RCM_SRS1_MDM_AP_SHIFT
#define RCM_SRS1_SACKERR_MASK                    (0x20U)
#define RCM_SRS1_SACKERR_SHIFT                   (5U)
#define RCM_SRS1_SACKERR(x)                      (((uint8_t)(((uint8_t)(x)) << RCM_SRS1_SACKERR_SHIF

/*! @name RPFC - Reset Pin Filter Control register */
#define RCM_RPFC_RSTFLTSRW_MASK                  (0x3U)
#define RCM_RPFC_RSTFLTSRW_SHIFT                 (0U)
#define RCM_RPFC_RSTFLTSRW(x)                    (((uint8_t)(((uint8_t)(x)) << RCM_RPFC_RSTFLTSRW_
#define RCM_RPFC_RSTFLTSS_MASK                   (0x4U)
#define RCM_RPFC_RSTFLTSS_SHIFT                  (2U)
#define RCM_RPFC_RSTFLTSS(x)                     (((uint8_t)(((uint8_t)(x)) << RCM_RPFC_RSTFLTSS_SHI

/*! @name RPFW - Reset Pin Filter Width register */
#define RCM_RPFW_RSTFLTSEL_MASK                  (0x1FU)
#define RCM_RPFW_RSTFLTSEL_SHIFT                 (0U)
#define RCM_RPFW_RSTFLTSEL(x)                    (((uint8_t)(((uint8_t)(x)) << RCM_RPFW_RSTFLTSEL_S


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup ROM_Register_Masks ROM Register Masks
 * @{
 */

/*! @name ENTRY - Entry */
#define ROM_ENTRY_ENTRY_MASK                     (0xFFFFFFFFU)
#define ROM_ENTRY_ENTRY_SHIFT                    (0U)
#define ROM_ENTRY_ENTRY(x)                       (((uint32_t)(((uint32_t)(x)) << ROM_ENTRY_ENTRY_SHI

/* The count of ROM_ENTRY */
#define ROM_ENTRY_COUNT                          (3U)

/*! @name TABLEMARK - End of Table Marker Register */
#define ROM_TABLEMARK_MARK_MASK                  (0xFFFFFFFFU)
#define ROM_TABLEMARK_MARK_SHIFT                 (0U)
#define ROM_TABLEMARK_MARK(x)                    (((uint32_t)(((uint32_t)(x)) << ROM_TABLEMARK_MAR

/*! @name SYSACCESS - System Access Register */
#define ROM_SYSACCESS_SYSACCESS_MASK             (0xFFFFFFFFU)
#define ROM_SYSACCESS_SYSACCESS_SHIFT            (0U)
#define ROM_SYSACCESS_SYSACCESS(x)               (((uint32_t)(((uint32_t)(x)) << ROM_SYSACCESS_
/*! @name PERIPHID4 - Peripheral ID Register */
#define ROM_PERIPHID4_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID4_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID4_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID4_PERIP

/*! @name PERIPHID5 - Peripheral ID Register */
#define ROM_PERIPHID5_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID5_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID5_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID5_PERIP

/*! @name PERIPHID6 - Peripheral ID Register */
#define ROM_PERIPHID6_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID6_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID6_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID6_PERIP

/*! @name PERIPHID7 - Peripheral ID Register */
#define ROM_PERIPHID7_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID7_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID7_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID7_PERIP

/*! @name PERIPHID0 - Peripheral ID Register */
#define ROM_PERIPHID0_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID0_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID0_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID0_PERIP

/*! @name PERIPHID1 - Peripheral ID Register */
#define ROM_PERIPHID1_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID1_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID1_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID1_PERIP

/*! @name PERIPHID2 - Peripheral ID Register */
#define ROM_PERIPHID2_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID2_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID2_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID2_PERIP

/*! @name PERIPHID3 - Peripheral ID Register */
#define ROM_PERIPHID3_PERIPHID_MASK            (0xFFFFFFFFU)
#define ROM_PERIPHID3_PERIPHID_SHIFT           (0U)
#define ROM_PERIPHID3_PERIPHID(x)              (((uint32_t)(((uint32_t)(x)) << ROM_PERIPHID3_PERIP

/*! @name COMPID - Component ID Register */
#define ROM_COMPID_COMPID_MASK                 (0xFFFFFFFFU)
#define ROM_COMPID_COMPID_SHIFT                (0U)
#define ROM_COMPID_COMPID(x)                   (((uint32_t)(((uint32_t)(x)) << ROM_COMPID_COMPID_

/* The count of ROM_COMPID */
#define ROM_COMPID_COUNT                       (4U)


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup RTC_Register_Masks RTC Register Masks
 * @{
 */

/*! @name TSR - RTC Time Seconds Register */
#define RTC_TSR_TSR_MASK                         (0xFFFFFFFFU)
#define RTC_TSR_TSR_SHIFT                        (0U)
#define RTC_TSR_TSR(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TSR_TSR_SHIFT)) & RTC_

/*! @name TPR - RTC Time Prescaler Register */
```

```
#define RTC_TPR_TPR_MASK                         (0xFFFFU)
#define RTC_TPR_TPR_SHIFT                        (0U)
#define RTC_TPR_TPR(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TPR_TPR_SHIFT)) & RTC_

/*! @name TAR - RTC Time Alarm Register */
#define RTC_TAR_TAR_MASK                         (0xFFFFFFFFU)
#define RTC_TAR_TAR_SHIFT                        (0U)
#define RTC_TAR_TAR(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TAR_TAR_SHIFT)) & RTC_

/*! @name TCR - RTC Time Compensation Register */
#define RTC_TCR_TCR_MASK                         (0xFFU)
#define RTC_TCR_TCR_SHIFT                        (0U)
#define RTC_TCR_TCR(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TCR_TCR_SHIFT)) & RTC
#define RTC_TCR_CIR_MASK                         (0xFF00U)
#define RTC_TCR_CIR_SHIFT                        (8U)
#define RTC_TCR_CIR(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TCR_CIR_SHIFT)) & RTC_
#define RTC_TCR_TCV_MASK                         (0xFF0000U)
#define RTC_TCR_TCV_SHIFT                        (16U)
#define RTC_TCR_TCV(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TCR_TCV_SHIFT)) & RTC
#define RTC_TCR_CIC_MASK                         (0xFF000000U)
#define RTC_TCR_CIC_SHIFT                        (24U)
#define RTC_TCR_CIC(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_TCR_CIC_SHIFT)) & RTC_

/*! @name CR - RTC Control Register */
#define RTC_CR_SWR_MASK                          (0x1U)
#define RTC_CR_SWR_SHIFT                         (0U)
#define RTC_CR_SWR(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_CR_SWR_SHIFT)) & RTC_
#define RTC_CR_WPE_MASK                          (0x2U)
#define RTC_CR_WPE_SHIFT                         (1U)
#define RTC_CR_WPE(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_CR_WPE_SHIFT)) & RTC_
#define RTC_CR_SUP_MASK                          (0x4U)
#define RTC_CR_SUP_SHIFT                         (2U)
#define RTC_CR_SUP(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_CR_SUP_SHIFT)) & RTC_
#define RTC_CR_UM_MASK                           (0x8U)
#define RTC_CR_UM_SHIFT                          (3U)
#define RTC_CR_UM(x)                             (((uint32_t)(((uint32_t)(x)) << RTC_CR_UM_SHIFT)) & RTC_C
#define RTC_CR_OSCE_MASK                         (0x100U)
#define RTC_CR_OSCE_SHIFT                        (8U)
#define RTC_CR_OSCE(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_CR_OSCE_SHIFT)) & RTC
#define RTC_CR_CLKO_MASK                         (0x200U)
#define RTC_CR_CLKO_SHIFT                        (9U)
#define RTC_CR_CLKO(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_CR_CLKO_SHIFT)) & RTC
#define RTC_CR_SC16P_MASK                        (0x400U)
#define RTC_CR_SC16P_SHIFT                       (10U)
#define RTC_CR_SC16P(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_CR_SC16P_SHIFT)) & RT
#define RTC_CR_SC8P_MASK                         (0x800U)
#define RTC_CR_SC8P_SHIFT                        (11U)
#define RTC_CR_SC8P(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_CR_SC8P_SHIFT)) & RTC
#define RTC_CR_SC4P_MASK                         (0x1000U)
#define RTC_CR_SC4P_SHIFT                        (12U)
#define RTC_CR_SC4P(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_CR_SC4P_SHIFT)) & RTC
#define RTC_CR_SC2P_MASK                         (0x2000U)
#define RTC_CR_SC2P_SHIFT                        (13U)
```

```
#define RTC_CR_SC2P(x)                           (((uint32_t)(((uint32_t)(x)) << RTC_CR_SC2P_SHIFT)) & RTC

/*! @name SR - RTC Status Register */
#define RTC_SR_TIF_MASK                           (0x1U)
#define RTC_SR_TIF_SHIFT                          (0U)
#define RTC_SR_TIF(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_SR_TIF_SHIFT)) & RTC_SR
#define RTC_SR_TOF_MASK                           (0x2U)
#define RTC_SR_TOF_SHIFT                          (1U)
#define RTC_SR_TOF(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_SR_TOF_SHIFT)) & RTC_S
#define RTC_SR_TAF_MASK                           (0x4U)
#define RTC_SR_TAF_SHIFT                          (2U)
#define RTC_SR_TAF(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_SR_TAF_SHIFT)) & RTC_S
#define RTC_SR_TCE_MASK                           (0x10U)
#define RTC_SR_TCE_SHIFT                          (4U)
#define RTC_SR_TCE(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_SR_TCE_SHIFT)) & RTC_S

/*! @name LR - RTC Lock Register */
#define RTC_LR_TCL_MASK                           (0x8U)
#define RTC_LR_TCL_SHIFT                          (3U)
#define RTC_LR_TCL(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_LR_TCL_SHIFT)) & RTC_LR
#define RTC_LR_CRL_MASK                           (0x10U)
#define RTC_LR_CRL_SHIFT                          (4U)
#define RTC_LR_CRL(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_LR_CRL_SHIFT)) & RTC_L
#define RTC_LR_SRL_MASK                           (0x20U)
#define RTC_LR_SRL_SHIFT                          (5U)
#define RTC_LR_SRL(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_LR_SRL_SHIFT)) & RTC_L
#define RTC_LR_LRL_MASK                           (0x40U)
#define RTC_LR_LRL_SHIFT                          (6U)
#define RTC_LR_LRL(x)                            (((uint32_t)(((uint32_t)(x)) << RTC_LR_LRL_SHIFT)) & RTC_L

/*! @name IER - RTC Interrupt Enable Register */
#define RTC_IER_TIIE_MASK                         (0x1U)
#define RTC_IER_TIIE_SHIFT                        (0U)
#define RTC_IER_TIIE(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_IER_TIIE_SHIFT)) & RTC_IE
#define RTC_IER_TOIE_MASK                         (0x2U)
#define RTC_IER_TOIE_SHIFT                        (1U)
#define RTC_IER_TOIE(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_IER_TOIE_SHIFT)) & RTC_
#define RTC_IER_TAIE_MASK                         (0x4U)
#define RTC_IER_TAIE_SHIFT                        (2U)
#define RTC_IER_TAIE(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_IER_TAIE_SHIFT)) & RTC_
#define RTC_IER_TSIE_MASK                         (0x10U)
#define RTC_IER_TSIE_SHIFT                        (4U)
#define RTC_IER_TSIE(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_IER_TSIE_SHIFT)) & RTC_
#define RTC_IER_WPON_MASK                         (0x80U)
#define RTC_IER_WPON_SHIFT                        (7U)
#define RTC_IER_WPON(x)                          (((uint32_t)(((uint32_t)(x)) << RTC_IER_WPON_SHIFT)) & R


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the RTC peripheral type */
#define RTC_IRQS                                { RTC_IRQn }
#define RTC_SECONDS_IRQS                        { RTC_Seconds_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup SIM_Register_Masks SIM Register Masks
 * @{
 */

/*! @name SOPT1 - System Options Register 1 */
#define SIM_SOPT1_OSC32KSEL_MASK                 (0xC0000U)
#define SIM_SOPT1_OSC32KSEL_SHIFT                (18U)
#define SIM_SOPT1_OSC32KSEL(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1_OSC32KSEL
#define SIM_SOPT1_USBVSTBY_MASK                  (0x20000000U)
#define SIM_SOPT1_USBVSTBY_SHIFT                 (29U)
#define SIM_SOPT1_USBVSTBY(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1_USBVSTBY_
#define SIM_SOPT1_USBSSTBY_MASK                  (0x40000000U)
#define SIM_SOPT1_USBSSTBY_SHIFT                 (30U)
#define SIM_SOPT1_USBSSTBY(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1_USBSSTBY_
#define SIM_SOPT1_USBREGEN_MASK                  (0x80000000U)
#define SIM_SOPT1_USBREGEN_SHIFT                 (31U)
#define SIM_SOPT1_USBREGEN(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1_USBREGEN_

/*! @name SOPT1CFG - SOPT1 Configuration Register */
#define SIM_SOPT1CFG_URWE_MASK                   (0x1000000U)
#define SIM_SOPT1CFG_URWE_SHIFT                  (24U)
#define SIM_SOPT1CFG_URWE(x)                     (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1CFG_URWE_
#define SIM_SOPT1CFG_UVSWE_MASK                  (0x2000000U)
#define SIM_SOPT1CFG_UVSWE_SHIFT                 (25U)
#define SIM_SOPT1CFG_UVSWE(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1CFG_UVSWE
#define SIM_SOPT1CFG_USSWE_MASK                  (0x4000000U)
#define SIM_SOPT1CFG_USSWE_SHIFT                 (26U)
#define SIM_SOPT1CFG_USSWE(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT1CFG_USSWE

/*! @name SOPT2 - System Options Register 2 */
#define SIM_SOPT2_RTCCLKOUTSEL_MASK              (0x10U)
#define SIM_SOPT2_RTCCLKOUTSEL_SHIFT             (4U)
#define SIM_SOPT2_RTCCLKOUTSEL(x)                (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_RTCCLKO
#define SIM_SOPT2_CLKOUTSEL_MASK                 (0xE0U)
#define SIM_SOPT2_CLKOUTSEL_SHIFT                (5U)
#define SIM_SOPT2_CLKOUTSEL(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_CLKOUTSEL
#define SIM_SOPT2_PLLFLLSEL_MASK                 (0x10000U)
#define SIM_SOPT2_PLLFLLSEL_SHIFT                (16U)
#define SIM_SOPT2_PLLFLLSEL(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_PLLFLLSEL_S
#define SIM_SOPT2_USBSRC_MASK                    (0x40000U)
#define SIM_SOPT2_USBSRC_SHIFT                   (18U)
#define SIM_SOPT2_USBSRC(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_USBSRC_SHIF
#define SIM_SOPT2_TPMSRC_MASK                    (0x3000000U)
```

```
#define SIM_SOPT2_TPMSRC_SHIFT                   (24U)
#define SIM_SOPT2_TPMSRC(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_TPMSRC_SHI
#define SIM_SOPT2_UART0SRC_MASK                  (0xC000000U)
#define SIM_SOPT2_UART0SRC_SHIFT                 (26U)
#define SIM_SOPT2_UART0SRC(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT2_UART0SRC_

/*! @name SOPT4 - System Options Register 4 */
#define SIM_SOPT4_TPM1CH0SRC_MASK                (0x40000U)
#define SIM_SOPT4_TPM1CH0SRC_SHIFT               (18U)
#define SIM_SOPT4_TPM1CH0SRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT4_TPM1CH0S
#define SIM_SOPT4_TPM2CH0SRC_MASK                (0x100000U)
#define SIM_SOPT4_TPM2CH0SRC_SHIFT               (20U)
#define SIM_SOPT4_TPM2CH0SRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT4_TPM2CH0S
#define SIM_SOPT4_TPM0CLKSEL_MASK                (0x1000000U)
#define SIM_SOPT4_TPM0CLKSEL_SHIFT               (24U)
#define SIM_SOPT4_TPM0CLKSEL(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT4_TPM0CLKS
#define SIM_SOPT4_TPM1CLKSEL_MASK                (0x2000000U)
#define SIM_SOPT4_TPM1CLKSEL_SHIFT               (25U)
#define SIM_SOPT4_TPM1CLKSEL(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT4_TPM1CLKS
#define SIM_SOPT4_TPM2CLKSEL_MASK                (0x4000000U)
#define SIM_SOPT4_TPM2CLKSEL_SHIFT               (26U)
#define SIM_SOPT4_TPM2CLKSEL(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT4_TPM2CLKS

/*! @name SOPT5 - System Options Register 5 */
#define SIM_SOPT5_UART0TXSRC_MASK                (0x3U)
#define SIM_SOPT5_UART0TXSRC_SHIFT               (0U)
#define SIM_SOPT5_UART0TXSRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART0TXS
#define SIM_SOPT5_UART0RXSRC_MASK                (0x4U)
#define SIM_SOPT5_UART0RXSRC_SHIFT               (2U)
#define SIM_SOPT5_UART0RXSRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART0RXS
#define SIM_SOPT5_UART1TXSRC_MASK                (0x30U)
#define SIM_SOPT5_UART1TXSRC_SHIFT               (4U)
#define SIM_SOPT5_UART1TXSRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART1TXS
#define SIM_SOPT5_UART1RXSRC_MASK                (0x40U)
#define SIM_SOPT5_UART1RXSRC_SHIFT               (6U)
#define SIM_SOPT5_UART1RXSRC(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART1RXS
#define SIM_SOPT5_UART0ODE_MASK                  (0x10000U)
#define SIM_SOPT5_UART0ODE_SHIFT                 (16U)
#define SIM_SOPT5_UART0ODE(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART0ODE_
#define SIM_SOPT5_UART1ODE_MASK                  (0x20000U)
#define SIM_SOPT5_UART1ODE_SHIFT                 (17U)
#define SIM_SOPT5_UART1ODE(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART1ODE_
#define SIM_SOPT5_UART2ODE_MASK                  (0x40000U)
#define SIM_SOPT5_UART2ODE_SHIFT                 (18U)
#define SIM_SOPT5_UART2ODE(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_SOPT5_UART2ODE_

/*! @name SOPT7 - System Options Register 7 */
#define SIM_SOPT7_ADC0TRGSEL_MASK                (0xFU)
#define SIM_SOPT7_ADC0TRGSEL_SHIFT               (0U)
#define SIM_SOPT7_ADC0TRGSEL(x)                  (((uint32_t)(((uint32_t)(x)) << SIM_SOPT7_ADC0TRGS
#define SIM_SOPT7_ADC0PRETRGSEL_MASK             (0x10U)
#define SIM_SOPT7_ADC0PRETRGSEL_SHIFT            (4U)
#define SIM_SOPT7_ADC0PRETRGSEL(x)               (((uint32_t)(((uint32_t)(x)) << SIM_SOPT7_ADC0PR
#define SIM_SOPT7_ADC0ALTTRGEN_MASK              (0x80U)
#define SIM_SOPT7_ADC0ALTTRGEN_SHIFT             (7U)
#define SIM_SOPT7_ADC0ALTTRGEN(x)                (((uint32_t)(((uint32_t)(x)) << SIM_SOPT7_ADC0ALT

/*! @name SDID - System Device Identification Register */
#define SIM_SDID_PINID_MASK                      (0xFU)
#define SIM_SDID_PINID_SHIFT                     (0U)
#define SIM_SDID_PINID(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SDID_PINID_SHIFT)) & SIM
#define SIM_SDID_DIEID_MASK                      (0xF80U)
#define SIM_SDID_DIEID_SHIFT                     (7U)
#define SIM_SDID_DIEID(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SDID_DIEID_SHIFT)) & SIM
#define SIM_SDID_REVID_MASK                      (0xF000U)
#define SIM_SDID_REVID_SHIFT                     (12U)
#define SIM_SDID_REVID(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SDID_REVID_SHIFT)) & S
#define SIM_SDID_SRAMSIZE_MASK                   (0xF0000U)
#define SIM_SDID_SRAMSIZE_SHIFT                  (16U)
#define SIM_SDID_SRAMSIZE(x)                     (((uint32_t)(((uint32_t)(x)) << SIM_SDID_SRAMSIZE_SHIF
#define SIM_SDID_SERIESID_MASK                   (0xF00000U)
#define SIM_SDID_SERIESID_SHIFT                  (20U)
#define SIM_SDID_SERIESID(x)                     (((uint32_t)(((uint32_t)(x)) << SIM_SDID_SERIESID_SHIFT)
#define SIM_SDID_SUBFAMID_MASK                   (0xF000000U)
#define SIM_SDID_SUBFAMID_SHIFT                  (24U)
#define SIM_SDID_SUBFAMID(x)                     (((uint32_t)(((uint32_t)(x)) << SIM_SDID_SUBFAMID_SHIF
#define SIM_SDID_FAMID_MASK                      (0xF0000000U)
#define SIM_SDID_FAMID_SHIFT                     (28U)
#define SIM_SDID_FAMID(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SDID_FAMID_SHIFT)) & S

/*! @name SCGC4 - System Clock Gating Control Register 4 */
#define SIM_SCGC4_I2C0_MASK                      (0x40U)
#define SIM_SCGC4_I2C0_SHIFT                     (6U)
#define SIM_SCGC4_I2C0(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_I2C0_SHIFT)) & S
#define SIM_SCGC4_I2C1_MASK                      (0x80U)
#define SIM_SCGC4_I2C1_SHIFT                     (7U)
#define SIM_SCGC4_I2C1(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_I2C1_SHIFT)) & S
#define SIM_SCGC4_UART0_MASK                     (0x400U)
#define SIM_SCGC4_UART0_SHIFT                    (10U)
#define SIM_SCGC4_UART0(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_UART0_SHIFT)
#define SIM_SCGC4_UART1_MASK                     (0x800U)
#define SIM_SCGC4_UART1_SHIFT                    (11U)
#define SIM_SCGC4_UART1(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_UART1_SHIFT)
#define SIM_SCGC4_UART2_MASK                     (0x1000U)
#define SIM_SCGC4_UART2_SHIFT                    (12U)
#define SIM_SCGC4_UART2(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_UART2_SHIFT)
#define SIM_SCGC4_USBOTG_MASK                    (0x40000U)
#define SIM_SCGC4_USBOTG_SHIFT                   (18U)
#define SIM_SCGC4_USBOTG(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_USBOTG_SHI
#define SIM_SCGC4_CMP_MASK                       (0x80000U)
#define SIM_SCGC4_CMP_SHIFT                      (19U)
#define SIM_SCGC4_CMP(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_CMP_SHIFT)) &
#define SIM_SCGC4_SPI0_MASK                      (0x400000U)
#define SIM_SCGC4_SPI0_SHIFT                     (22U)
#define SIM_SCGC4_SPI0(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_SPI0_SHIFT)) & S
#define SIM_SCGC4_SPI1_MASK                      (0x800000U)
```

```
#define SIM_SCGC4_SPI1_SHIFT                     (23U)
#define SIM_SCGC4_SPI1(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC4_SPI1_SHIFT)) & S

/*! @name SCGC5 - System Clock Gating Control Register 5 */
#define SIM_SCGC5_LPTMR_MASK                     (0x1U)
#define SIM_SCGC5_LPTMR_SHIFT                    (0U)
#define SIM_SCGC5_LPTMR(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_LPTMR_SHIFT
#define SIM_SCGC5_TSI_MASK                       (0x20U)
#define SIM_SCGC5_TSI_SHIFT                      (5U)
#define SIM_SCGC5_TSI(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_TSI_SHIFT)) & SIM
#define SIM_SCGC5_PORTA_MASK                     (0x200U)
#define SIM_SCGC5_PORTA_SHIFT                    (9U)
#define SIM_SCGC5_PORTA(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_PORTA_SHIFT
#define SIM_SCGC5_PORTB_MASK                     (0x400U)
#define SIM_SCGC5_PORTB_SHIFT                    (10U)
#define SIM_SCGC5_PORTB(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_PORTB_SHIFT
#define SIM_SCGC5_PORTC_MASK                     (0x800U)
#define SIM_SCGC5_PORTC_SHIFT                    (11U)
#define SIM_SCGC5_PORTC(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_PORTC_SHIFT
#define SIM_SCGC5_PORTD_MASK                     (0x1000U)
#define SIM_SCGC5_PORTD_SHIFT                    (12U)
#define SIM_SCGC5_PORTD(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_PORTD_SHIFT
#define SIM_SCGC5_PORTE_MASK                     (0x2000U)
#define SIM_SCGC5_PORTE_SHIFT                    (13U)
#define SIM_SCGC5_PORTE(x)                       (((uint32_t)(((uint32_t)(x)) << SIM_SCGC5_PORTE_SHIFT

/*! @name SCGC6 - System Clock Gating Control Register 6 */
#define SIM_SCGC6_FTF_MASK                       (0x1U)
#define SIM_SCGC6_FTF_SHIFT                      (0U)
#define SIM_SCGC6_FTF(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_FTF_SHIFT)) & S
#define SIM_SCGC6_DMAMUX_MASK                    (0x2U)
#define SIM_SCGC6_DMAMUX_SHIFT                   (1U)
#define SIM_SCGC6_DMAMUX(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_DMAMUX_SH
#define SIM_SCGC6_PIT_MASK                       (0x800000U)
#define SIM_SCGC6_PIT_SHIFT                      (23U)
#define SIM_SCGC6_PIT(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_PIT_SHIFT)) & SIM
#define SIM_SCGC6_TPM0_MASK                      (0x1000000U)
#define SIM_SCGC6_TPM0_SHIFT                     (24U)
#define SIM_SCGC6_TPM0(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_TPM0_SHIFT)) &
#define SIM_SCGC6_TPM1_MASK                      (0x2000000U)
#define SIM_SCGC6_TPM1_SHIFT                     (25U)
#define SIM_SCGC6_TPM1(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_TPM1_SHIFT)) &
#define SIM_SCGC6_TPM2_MASK                      (0x4000000U)
#define SIM_SCGC6_TPM2_SHIFT                     (26U)
#define SIM_SCGC6_TPM2(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_TPM2_SHIFT)) &
#define SIM_SCGC6_ADC0_MASK                      (0x8000000U)
#define SIM_SCGC6_ADC0_SHIFT                     (27U)
#define SIM_SCGC6_ADC0(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_ADC0_SHIFT)) &
#define SIM_SCGC6_RTC_MASK                       (0x20000000U)
#define SIM_SCGC6_RTC_SHIFT                      (29U)
#define SIM_SCGC6_RTC(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_RTC_SHIFT)) & S
#define SIM_SCGC6_DAC0_MASK                      (0x80000000U)
#define SIM_SCGC6_DAC0_SHIFT                     (31U)
```

```
#define SIM_SCGC6_DAC0(x)                        (((uint32_t)(((uint32_t)(x)) << SIM_SCGC6_DAC0_SHIFT)) &

/*! @name SCGC7 - System Clock Gating Control Register 7 */
#define SIM_SCGC7_DMA_MASK                       (0x100U)
#define SIM_SCGC7_DMA_SHIFT                      (8U)
#define SIM_SCGC7_DMA(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_SCGC7_DMA_SHIFT)) &

/*! @name CLKDIV1 - System Clock Divider Register 1 */
#define SIM_CLKDIV1_OUTDIV4_MASK                 (0x70000U)
#define SIM_CLKDIV1_OUTDIV4_SHIFT                (16U)
#define SIM_CLKDIV1_OUTDIV4(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_CLKDIV1_OUTDIV4_S
#define SIM_CLKDIV1_OUTDIV1_MASK                 (0xF0000000U)
#define SIM_CLKDIV1_OUTDIV1_SHIFT                (28U)
#define SIM_CLKDIV1_OUTDIV1(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_CLKDIV1_OUTDIV1_S

/*! @name FCFG1 - Flash Configuration Register 1 */
#define SIM_FCFG1_FLASHDIS_MASK                  (0x1U)
#define SIM_FCFG1_FLASHDIS_SHIFT                 (0U)
#define SIM_FCFG1_FLASHDIS(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_FCFG1_FLASHDIS_SH
#define SIM_FCFG1_FLASHDOZE_MASK                 (0x2U)
#define SIM_FCFG1_FLASHDOZE_SHIFT                (1U)
#define SIM_FCFG1_FLASHDOZE(x)                   (((uint32_t)(((uint32_t)(x)) << SIM_FCFG1_FLASHDOZE
#define SIM_FCFG1_PFSIZE_MASK                    (0xF000000U)
#define SIM_FCFG1_PFSIZE_SHIFT                   (24U)
#define SIM_FCFG1_PFSIZE(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_FCFG1_PFSIZE_SHIFT)

/*! @name FCFG2 - Flash Configuration Register 2 */
#define SIM_FCFG2_MAXADDR0_MASK                  (0x7F000000U)
#define SIM_FCFG2_MAXADDR0_SHIFT                 (24U)
#define SIM_FCFG2_MAXADDR0(x)                    (((uint32_t)(((uint32_t)(x)) << SIM_FCFG2_MAXADDR0_

/*! @name UIDMH - Unique Identification Register Mid-High */
#define SIM_UIDMH_UID_MASK                       (0xFFFFU)
#define SIM_UIDMH_UID_SHIFT                      (0U)
#define SIM_UIDMH_UID(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_UIDMH_UID_SHIFT)) & SIM

/*! @name UIDML - Unique Identification Register Mid Low */
#define SIM_UIDML_UID_MASK                       (0xFFFFFFFFU)
#define SIM_UIDML_UID_SHIFT                      (0U)
#define SIM_UIDML_UID(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_UIDML_UID_SHIFT)) & SIM

/*! @name UIDL - Unique Identification Register Low */
#define SIM_UIDL_UID_MASK                        (0xFFFFFFFFU)
#define SIM_UIDL_UID_SHIFT                       (0U)
#define SIM_UIDL_UID(x)                          (((uint32_t)(((uint32_t)(x)) << SIM_UIDL_UID_SHIFT)) & SIM_U

/*! @name COPC - COP Control Register */
#define SIM_COPC_COPW_MASK                       (0x1U)
#define SIM_COPC_COPW_SHIFT                      (0U)
#define SIM_COPC_COPW(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_COPC_COPW_SHIFT)) &
#define SIM_COPC_COPCLKS_MASK                    (0x2U)
#define SIM_COPC_COPCLKS_SHIFT                   (1U)
#define SIM_COPC_COPCLKS(x)                      (((uint32_t)(((uint32_t)(x)) << SIM_COPC_COPCLKS_SH
#define SIM_COPC_COPT_MASK                       (0xCU)
#define SIM_COPC_COPT_SHIFT                      (2U)
#define SIM_COPC_COPT(x)                         (((uint32_t)(((uint32_t)(x)) << SIM_COPC_COPT_SHIFT)) &

/*! @name SRVCOP - Service COP Register */
#define SIM_SRVCOP_SRVCOP_MASK                   (0xFFU)
#define SIM_SRVCOP_SRVCOP_SHIFT                  (0U)
#define SIM_SRVCOP_SRVCOP(x)                     (((uint32_t)(((uint32_t)(x)) << SIM_SRVCOP_SRVCOP_S


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup SMC_Register_Masks SMC Register Masks
 * @{
 */

/*! @name PMPROT - Power Mode Protection register */
#define SMC_PMPROT_AVLLS_MASK                    (0x2U)
#define SMC_PMPROT_AVLLS_SHIFT                   (1U)
#define SMC_PMPROT_AVLLS(x)                      (((uint8_t)(((uint8_t)(x)) << SMC_PMPROT_AVLLS_SHIF
#define SMC_PMPROT_ALLS_MASK                     (0x8U)
#define SMC_PMPROT_ALLS_SHIFT                    (3U)
#define SMC_PMPROT_ALLS(x)                       (((uint8_t)(((uint8_t)(x)) << SMC_PMPROT_ALLS_SHIFT))
#define SMC_PMPROT_AVLP_MASK                     (0x20U)
#define SMC_PMPROT_AVLP_SHIFT                    (5U)
#define SMC_PMPROT_AVLP(x)                       (((uint8_t)(((uint8_t)(x)) << SMC_PMPROT_AVLP_SHIFT)

/*! @name PMCTRL - Power Mode Control register */
#define SMC_PMCTRL_STOPM_MASK                    (0x7U)
#define SMC_PMCTRL_STOPM_SHIFT                   (0U)
#define SMC_PMCTRL_STOPM(x)                      (((uint8_t)(((uint8_t)(x)) << SMC_PMCTRL_STOPM_SHI
#define SMC_PMCTRL_STOPA_MASK                    (0x8U)
#define SMC_PMCTRL_STOPA_SHIFT                   (3U)
#define SMC_PMCTRL_STOPA(x)                      (((uint8_t)(((uint8_t)(x)) << SMC_PMCTRL_STOPA_SHIF
#define SMC_PMCTRL_RUNM_MASK                     (0x60U)
#define SMC_PMCTRL_RUNM_SHIFT                    (5U)
#define SMC_PMCTRL_RUNM(x)                       (((uint8_t)(((uint8_t)(x)) << SMC_PMCTRL_RUNM_SHIFT

/*! @name STOPCTRL - Stop Control Register */
#define SMC_STOPCTRL_VLLSM_MASK                  (0x7U)
#define SMC_STOPCTRL_VLLSM_SHIFT                 (0U)
#define SMC_STOPCTRL_VLLSM(x)                    (((uint8_t)(((uint8_t)(x)) << SMC_STOPCTRL_VLLSM_S
#define SMC_STOPCTRL_PORPO_MASK                  (0x20U)
#define SMC_STOPCTRL_PORPO_SHIFT                 (5U)
#define SMC_STOPCTRL_PORPO(x)                    (((uint8_t)(((uint8_t)(x)) << SMC_STOPCTRL_PORPO_
#define SMC_STOPCTRL_PSTOPO_MASK                 (0xC0U)
#define SMC_STOPCTRL_PSTOPO_SHIFT                (6U)
#define SMC_STOPCTRL_PSTOPO(x)                   (((uint8_t)(((uint8_t)(x)) << SMC_STOPCTRL_PSTOPO

/*! @name PMSTAT - Power Mode Status register */
#define SMC_PMSTAT_PMSTAT_MASK                   (0x7FU)
#define SMC_PMSTAT_PMSTAT_SHIFT                  (0U)
#define SMC_PMSTAT_PMSTAT(x)                     (((uint8_t)(((uint8_t)(x)) << SMC_PMSTAT_PMSTAT_SH


/*!
 * @}
 */ /* 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)
/** 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 }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup SPI_Register_Masks SPI Register Masks
 * @{
 */

/*! @name C1 - SPI control register 1 */
#define SPI_C1_LSBFE_MASK                        (0x1U)
#define SPI_C1_LSBFE_SHIFT                       (0U)
#define SPI_C1_LSBFE(x)                          (((uint8_t)(((uint8_t)(x)) << SPI_C1_LSBFE_SHIFT)) & SPI_C1
#define SPI_C1_SSOE_MASK                         (0x2U)
#define SPI_C1_SSOE_SHIFT                        (1U)
#define SPI_C1_SSOE(x)                           (((uint8_t)(((uint8_t)(x)) << SPI_C1_SSOE_SHIFT)) & SPI_C1_
#define SPI_C1_CPHA_MASK                         (0x4U)
#define SPI_C1_CPHA_SHIFT                        (2U)
#define SPI_C1_CPHA(x)                           (((uint8_t)(((uint8_t)(x)) << SPI_C1_CPHA_SHIFT)) & SPI_C1_
#define SPI_C1_CPOL_MASK                         (0x8U)
#define SPI_C1_CPOL_SHIFT                        (3U)
#define SPI_C1_CPOL(x)                           (((uint8_t)(((uint8_t)(x)) << SPI_C1_CPOL_SHIFT)) & SPI_C1_
#define SPI_C1_MSTR_MASK                         (0x10U)
```

```
#define SPI_C1_MSTR_SHIFT                    (4U)
#define SPI_C1_MSTR(x)                      (((uint8_t)(((uint8_t)(x)) << SPI_C1_MSTR_SHIFT)) & SPI_C1_
#define SPI_C1_SPTIE_MASK                    (0x20U)
#define SPI_C1_SPTIE_SHIFT                   (5U)
#define SPI_C1_SPTIE(x)                     (((uint8_t)(((uint8_t)(x)) << SPI_C1_SPTIE_SHIFT)) & SPI_C1_
#define SPI_C1_SPE_MASK                      (0x40U)
#define SPI_C1_SPE_SHIFT                     (6U)
#define SPI_C1_SPE(x)                       (((uint8_t)(((uint8_t)(x)) << SPI_C1_SPE_SHIFT)) & SPI_C1_SP
#define SPI_C1_SPIE_MASK                     (0x80U)
#define SPI_C1_SPIE_SHIFT                    (7U)
#define SPI_C1_SPIE(x)                      (((uint8_t)(((uint8_t)(x)) << SPI_C1_SPIE_SHIFT)) & SPI_C1_SP

/*! @name C2 - SPI control register 2 */
#define SPI_C2_SPC0_MASK                     (0x1U)
#define SPI_C2_SPC0_SHIFT                    (0U)
#define SPI_C2_SPC0(x)                      (((uint8_t)(((uint8_t)(x)) << SPI_C2_SPC0_SHIFT)) & SPI_C2_S
#define SPI_C2_SPISWAI_MASK                  (0x2U)
#define SPI_C2_SPISWAI_SHIFT                 (1U)
#define SPI_C2_SPISWAI(x)                   (((uint8_t)(((uint8_t)(x)) << SPI_C2_SPISWAI_SHIFT)) & SPI_
#define SPI_C2_RXDMAE_MASK                   (0x4U)
#define SPI_C2_RXDMAE_SHIFT                  (2U)
#define SPI_C2_RXDMAE(x)                    (((uint8_t)(((uint8_t)(x)) << SPI_C2_RXDMAE_SHIFT)) & SPI
#define SPI_C2_BIDIROE_MASK                  (0x8U)
#define SPI_C2_BIDIROE_SHIFT                 (3U)
#define SPI_C2_BIDIROE(x)                   (((uint8_t)(((uint8_t)(x)) << SPI_C2_BIDIROE_SHIFT)) & SPI_
#define SPI_C2_MODFEN_MASK                   (0x10U)
#define SPI_C2_MODFEN_SHIFT                  (4U)
#define SPI_C2_MODFEN(x)                    (((uint8_t)(((uint8_t)(x)) << SPI_C2_MODFEN_SHIFT)) & SPI
#define SPI_C2_TXDMAE_MASK                   (0x20U)
#define SPI_C2_TXDMAE_SHIFT                  (5U)
#define SPI_C2_TXDMAE(x)                    (((uint8_t)(((uint8_t)(x)) << SPI_C2_TXDMAE_SHIFT)) & SPI_
#define SPI_C2_SPMIE_MASK                    (0x80U)
#define SPI_C2_SPMIE_SHIFT                   (7U)
#define SPI_C2_SPMIE(x)                     (((uint8_t)(((uint8_t)(x)) << SPI_C2_SPMIE_SHIFT)) & SPI_C2_

/*! @name BR - SPI baud rate register */
#define SPI_BR_SPR_MASK                      (0xFU)
#define SPI_BR_SPR_SHIFT                     (0U)
#define SPI_BR_SPR(x)                       (((uint8_t)(((uint8_t)(x)) << SPI_BR_SPR_SHIFT)) & SPI_BR_SI
#define SPI_BR_SPPR_MASK                     (0x70U)
#define SPI_BR_SPPR_SHIFT                    (4U)
#define SPI_BR_SPPR(x)                      (((uint8_t)(((uint8_t)(x)) << SPI_BR_SPPR_SHIFT)) & SPI_BR_

/*! @name S - SPI status register */
#define SPI_S_MODF_MASK                      (0x10U)
#define SPI_S_MODF_SHIFT                     (4U)
#define SPI_S_MODF(x)                       (((uint8_t)(((uint8_t)(x)) << SPI_S_MODF_SHIFT)) & SPI_S_MO
#define SPI_S_SPTEF_MASK                     (0x20U)
#define SPI_S_SPTEF_SHIFT                    (5U)
#define SPI_S_SPTEF(x)                      (((uint8_t)(((uint8_t)(x)) << SPI_S_SPTEF_SHIFT)) & SPI_S_SF
#define SPI_S_SPMF_MASK                      (0x40U)
#define SPI_S_SPMF_SHIFT                     (6U)
#define SPI_S_SPMF(x)                       (((uint8_t)(((uint8_t)(x)) << SPI_S_SPMF_SHIFT)) & SPI_S_SPI
#define SPI_S_SPRF_MASK                          (0x80U)
#define SPI_S_SPRF_SHIFT                         (7U)
#define SPI_S_SPRF(x)                            (((uint8_t)(((uint8_t)(x)) << SPI_S_SPRF_SHIFT)) & SPI_S_SPF

/*! @name D - SPI data register */
#define SPI_D_Bits_MASK                          (0xFFU)
#define SPI_D_Bits_SHIFT                         (0U)
#define SPI_D_Bits(x)                            (((uint8_t)(((uint8_t)(x)) << SPI_D_Bits_SHIFT)) & SPI_D_Bits_MA

/*! @name M - SPI match register */
#define SPI_M_Bits_MASK                          (0xFFU)
#define SPI_M_Bits_SHIFT                         (0U)
#define SPI_M_Bits(x)                            (((uint8_t)(((uint8_t)(x)) << SPI_M_Bits_SHIFT)) & SPI_M_Bits_M


/*!
 * @}
 */ /* 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)
/** Peripheral SPI1 base address */
#define SPI1_BASE                                (0x40077000u)
/** Peripheral SPI1 base pointer */
#define SPI1                                     ((SPI_Type *)SPI1_BASE)
/** 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 }
/** Interrupt vectors for the SPI peripheral type */
#define SPI_IRQS                                 { SPI0_IRQn, SPI1_IRQn }

/*!
 * @}
 */ /* 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
    __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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup TPM_Register_Masks TPM Register Masks
 * @{
 */

/*! @name SC - Status and Control */
#define TPM_SC_PS_MASK                           (0x7U)
#define TPM_SC_PS_SHIFT                          (0U)
#define TPM_SC_PS(x)                             (((uint32_t)(((uint32_t)(x)) << TPM_SC_PS_SHIFT)) & TPM_SC
#define TPM_SC_CMOD_MASK                         (0x18U)
#define TPM_SC_CMOD_SHIFT                        (3U)
#define TPM_SC_CMOD(x)                           (((uint32_t)(((uint32_t)(x)) << TPM_SC_CMOD_SHIFT)) & TP
#define TPM_SC_CPWMS_MASK                        (0x20U)
#define TPM_SC_CPWMS_SHIFT                       (5U)
#define TPM_SC_CPWMS(x)                          (((uint32_t)(((uint32_t)(x)) << TPM_SC_CPWMS_SHIFT)) &
#define TPM_SC_TOIE_MASK                         (0x40U)
#define TPM_SC_TOIE_SHIFT                        (6U)
#define TPM_SC_TOIE(x)                           (((uint32_t)(((uint32_t)(x)) << TPM_SC_TOIE_SHIFT)) & TPM_
#define TPM_SC_TOF_MASK                          (0x80U)
#define TPM_SC_TOF_SHIFT                         (7U)
#define TPM_SC_TOF(x)                            (((uint32_t)(((uint32_t)(x)) << TPM_SC_TOF_SHIFT)) & TPM_S
#define TPM_SC_DMA_MASK                          (0x100U)
#define TPM_SC_DMA_SHIFT                         (8U)
#define TPM_SC_DMA(x)                            (((uint32_t)(((uint32_t)(x)) << TPM_SC_DMA_SHIFT)) & TPM_

/*! @name CNT - Counter */
#define TPM_CNT_COUNT_MASK                       (0xFFFFU)
#define TPM_CNT_COUNT_SHIFT                      (0U)
#define TPM_CNT_COUNT(x)                         (((uint32_t)(((uint32_t)(x)) << TPM_CNT_COUNT_SHIFT)) &

/*! @name MOD - Modulo */
#define TPM_MOD_MOD_MASK                         (0xFFFFU)
#define TPM_MOD_MOD_SHIFT                        (0U)
#define TPM_MOD_MOD(x)                           (((uint32_t)(((uint32_t)(x)) << TPM_MOD_MOD_SHIFT)) & T

/*! @name CnSC - Channel (n) Status and Control */
#define TPM_CnSC_DMA_MASK                        (0x1U)
```

```
#define TPM_CnSC_DMA_SHIFT                    (0U)
#define TPM_CnSC_DMA(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_DMA_SHIFT)) & T
#define TPM_CnSC_ELSA_MASK                    (0x4U)
#define TPM_CnSC_ELSA_SHIFT                   (2U)
#define TPM_CnSC_ELSA(x)                      (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_ELSA_SHIFT)) &
#define TPM_CnSC_ELSB_MASK                    (0x8U)
#define TPM_CnSC_ELSB_SHIFT                   (3U)
#define TPM_CnSC_ELSB(x)                      (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_ELSB_SHIFT)) &
#define TPM_CnSC_MSA_MASK                     (0x10U)
#define TPM_CnSC_MSA_SHIFT                    (4U)
#define TPM_CnSC_MSA(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_MSA_SHIFT)) & T
#define TPM_CnSC_MSB_MASK                     (0x20U)
#define TPM_CnSC_MSB_SHIFT                    (5U)
#define TPM_CnSC_MSB(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_MSB_SHIFT)) & T
#define TPM_CnSC_CHIE_MASK                    (0x40U)
#define TPM_CnSC_CHIE_SHIFT                   (6U)
#define TPM_CnSC_CHIE(x)                      (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_CHIE_SHIFT)) & T
#define TPM_CnSC_CHF_MASK                     (0x80U)
#define TPM_CnSC_CHF_SHIFT                    (7U)
#define TPM_CnSC_CHF(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CnSC_CHF_SHIFT)) & T

/* The count of TPM_CnSC */
#define TPM_CnSC_COUNT                        (6U)

/*! @name CnV - Channel (n) Value */
#define TPM_CnV_VAL_MASK                      (0xFFFFU)
#define TPM_CnV_VAL_SHIFT                     (0U)
#define TPM_CnV_VAL(x)                        (((uint32_t)(((uint32_t)(x)) << TPM_CnV_VAL_SHIFT)) & TPM_

/* The count of TPM_CnV */
#define TPM_CnV_COUNT                         (6U)

/*! @name STATUS - Capture and Compare Status */
#define TPM_STATUS_CH0F_MASK                  (0x1U)
#define TPM_STATUS_CH0F_SHIFT                 (0U)
#define TPM_STATUS_CH0F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH0F_SHIFT
#define TPM_STATUS_CH1F_MASK                  (0x2U)
#define TPM_STATUS_CH1F_SHIFT                 (1U)
#define TPM_STATUS_CH1F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH1F_SHIFT
#define TPM_STATUS_CH2F_MASK                  (0x4U)
#define TPM_STATUS_CH2F_SHIFT                 (2U)
#define TPM_STATUS_CH2F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH2F_SHIFT
#define TPM_STATUS_CH3F_MASK                  (0x8U)
#define TPM_STATUS_CH3F_SHIFT                 (3U)
#define TPM_STATUS_CH3F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH3F_SHIFT
#define TPM_STATUS_CH4F_MASK                  (0x10U)
#define TPM_STATUS_CH4F_SHIFT                 (4U)
#define TPM_STATUS_CH4F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH4F_SHIFT
#define TPM_STATUS_CH5F_MASK                  (0x20U)
#define TPM_STATUS_CH5F_SHIFT                 (5U)
#define TPM_STATUS_CH5F(x)                    (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_CH5F_SHIFT
#define TPM_STATUS_TOF_MASK                   (0x100U)
#define TPM_STATUS_TOF_SHIFT                  (8U)
```

```
#define TPM_STATUS_TOF(x)                      (((uint32_t)(((uint32_t)(x)) << TPM_STATUS_TOF_SHIFT))

/*! @name CONF - Configuration */
#define TPM_CONF_DOZEEN_MASK                     (0x20U)
#define TPM_CONF_DOZEEN_SHIFT                    (5U)
#define TPM_CONF_DOZEEN(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CONF_DOZEEN_SHIF
#define TPM_CONF_DBGMODE_MASK                    (0xC0U)
#define TPM_CONF_DBGMODE_SHIFT                   (6U)
#define TPM_CONF_DBGMODE(x)                      (((uint32_t)(((uint32_t)(x)) << TPM_CONF_DBGMODE_S
#define TPM_CONF_GTBEEN_MASK                     (0x200U)
#define TPM_CONF_GTBEEN_SHIFT                    (9U)
#define TPM_CONF_GTBEEN(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CONF_GTBEEN_SHIF
#define TPM_CONF_CSOT_MASK                       (0x10000U)
#define TPM_CONF_CSOT_SHIFT                      (16U)
#define TPM_CONF_CSOT(x)                         (((uint32_t)(((uint32_t)(x)) << TPM_CONF_CSOT_SHIFT)) &
#define TPM_CONF_CSOO_MASK                       (0x20000U)
#define TPM_CONF_CSOO_SHIFT                      (17U)
#define TPM_CONF_CSOO(x)                         (((uint32_t)(((uint32_t)(x)) << TPM_CONF_CSOO_SHIFT))
#define TPM_CONF_CROT_MASK                       (0x40000U)
#define TPM_CONF_CROT_SHIFT                      (18U)
#define TPM_CONF_CROT(x)                         (((uint32_t)(((uint32_t)(x)) << TPM_CONF_CROT_SHIFT)) &
#define TPM_CONF_TRGSEL_MASK                     (0xF000000U)
#define TPM_CONF_TRGSEL_SHIFT                    (24U)
#define TPM_CONF_TRGSEL(x)                       (((uint32_t)(((uint32_t)(x)) << TPM_CONF_TRGSEL_SHIF


/*!
 * @}
 */ /* 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)
/** Peripheral TPM1 base address */
#define TPM1_BASE                                (0x40039000u)
/** Peripheral TPM1 base pointer */
#define TPM1                                     ((TPM_Type *)TPM1_BASE)
/** Peripheral TPM2 base address */
#define TPM2_BASE                                (0x4003A000u)
/** Peripheral TPM2 base pointer */
#define TPM2                                     ((TPM_Type *)TPM2_BASE)
/** 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 }
/** Interrupt vectors for the TPM peripheral type */
#define TPM_IRQS                                 { TPM0_IRQn, TPM1_IRQn, TPM2_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup TSI_Register_Masks TSI Register Masks
 * @{
 */

/*! @name GENCS - TSI General Control and Status Register */
#define TSI_GENCS_CURSW_MASK                     (0x2U)
#define TSI_GENCS_CURSW_SHIFT                    (1U)
#define TSI_GENCS_CURSW(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_CURSW_SHIF
#define TSI_GENCS_EOSF_MASK                      (0x4U)
#define TSI_GENCS_EOSF_SHIFT                     (2U)
#define TSI_GENCS_EOSF(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_EOSF_SHIFT)) &
#define TSI_GENCS_SCNIP_MASK                     (0x8U)
#define TSI_GENCS_SCNIP_SHIFT                    (3U)
#define TSI_GENCS_SCNIP(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_SCNIP_SHIFT))
#define TSI_GENCS_STM_MASK                       (0x10U)
#define TSI_GENCS_STM_SHIFT                      (4U)
#define TSI_GENCS_STM(x)                         (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_STM_SHIFT)) & T
#define TSI_GENCS_STPE_MASK                      (0x20U)
#define TSI_GENCS_STPE_SHIFT                     (5U)
#define TSI_GENCS_STPE(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_STPE_SHIFT)) &
#define TSI_GENCS_TSIIEN_MASK                    (0x40U)
#define TSI_GENCS_TSIIEN_SHIFT                   (6U)
#define TSI_GENCS_TSIIEN(x)                      (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_TSIIEN_SHIFT))
#define TSI_GENCS_TSIEN_MASK                     (0x80U)
#define TSI_GENCS_TSIEN_SHIFT                    (7U)
#define TSI_GENCS_TSIEN(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_TSIEN_SHIFT)) &
#define TSI_GENCS_NSCN_MASK                      (0x1F00U)
#define TSI_GENCS_NSCN_SHIFT                     (8U)
#define TSI_GENCS_NSCN(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_NSCN_SHIFT)) &
#define TSI_GENCS_PS_MASK                        (0xE000U)
#define TSI_GENCS_PS_SHIFT                       (13U)
#define TSI_GENCS_PS(x)                          (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_PS_SHIFT)) & TSI_
#define TSI_GENCS_EXTCHRG_MASK                   (0x70000U)
#define TSI_GENCS_EXTCHRG_SHIFT                  (16U)
#define TSI_GENCS_EXTCHRG(x)                     (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_EXTCHRG_S
#define TSI_GENCS_DVOLT_MASK                     (0x180000U)
#define TSI_GENCS_DVOLT_SHIFT                    (19U)
#define TSI_GENCS_DVOLT(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_DVOLT_SHIFT)
#define TSI_GENCS_REFCHRG_MASK                   (0xE00000U)
#define TSI_GENCS_REFCHRG_SHIFT                  (21U)
#define TSI_GENCS_REFCHRG(x)                     (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_REFCHRG_S
#define TSI_GENCS_MODE_MASK                      (0xF000000U)
#define TSI_GENCS_MODE_SHIFT                     (24U)
#define TSI_GENCS_MODE(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_MODE_SHIFT))
#define TSI_GENCS_ESOR_MASK                      (0x10000000U)
#define TSI_GENCS_ESOR_SHIFT                     (28U)
#define TSI_GENCS_ESOR(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_ESOR_SHIFT)) &
#define TSI_GENCS_OUTRGF_MASK                    (0x80000000U)
#define TSI_GENCS_OUTRGF_SHIFT                   (31U)
#define TSI_GENCS_OUTRGF(x)                      (((uint32_t)(((uint32_t)(x)) << TSI_GENCS_OUTRGF_SHI

/*! @name DATA - TSI DATA Register */
#define TSI_DATA_TSICNT_MASK                     (0xFFFFU)
#define TSI_DATA_TSICNT_SHIFT                    (0U)
#define TSI_DATA_TSICNT(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_DATA_TSICNT_SHIFT)) &
#define TSI_DATA_SWTS_MASK                       (0x400000U)
#define TSI_DATA_SWTS_SHIFT                      (22U)
#define TSI_DATA_SWTS(x)                         (((uint32_t)(((uint32_t)(x)) << TSI_DATA_SWTS_SHIFT)) & T
#define TSI_DATA_DMAEN_MASK                      (0x800000U)
#define TSI_DATA_DMAEN_SHIFT                     (23U)
#define TSI_DATA_DMAEN(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_DATA_DMAEN_SHIFT)) &
#define TSI_DATA_TSICH_MASK                      (0xF0000000U)
#define TSI_DATA_TSICH_SHIFT                     (28U)
#define TSI_DATA_TSICH(x)                        (((uint32_t)(((uint32_t)(x)) << TSI_DATA_TSICH_SHIFT)) & T

/*! @name TSHD - TSI Threshold Register */
#define TSI_TSHD_THRESL_MASK                     (0xFFFFU)
#define TSI_TSHD_THRESL_SHIFT                    (0U)
#define TSI_TSHD_THRESL(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_TSHD_THRESL_SHIFT))
#define TSI_TSHD_THRESH_MASK                     (0xFFFF0000U)
#define TSI_TSHD_THRESH_SHIFT                    (16U)
#define TSI_TSHD_THRESH(x)                       (((uint32_t)(((uint32_t)(x)) << TSI_TSHD_THRESH_SHIFT)


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the TSI peripheral type */
#define TSI_IRQS                                 { TSI0_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup UART_Register_Masks UART Register Masks
 * @{
 */

/*! @name BDH - UART Baud Rate Register: High */
#define UART_BDH_SBR_MASK                        (0x1FU)
#define UART_BDH_SBR_SHIFT                       (0U)
#define UART_BDH_SBR(x)                          (((uint8_t)(((uint8_t)(x)) << UART_BDH_SBR_SHIFT)) & UAR
#define UART_BDH_SBNS_MASK                       (0x20U)
#define UART_BDH_SBNS_SHIFT                      (5U)
#define UART_BDH_SBNS(x)                         (((uint8_t)(((uint8_t)(x)) << UART_BDH_SBNS_SHIFT)) & U
#define UART_BDH_RXEDGIE_MASK                    (0x40U)
```

```
#define UART_BDH_RXEDGIE_SHIFT                   (6U)
#define UART_BDH_RXEDGIE(x)                      (((uint8_t)(((uint8_t)(x)) << UART_BDH_RXEDGIE_SHIFT
#define UART_BDH_LBKDIE_MASK                     (0x80U)
#define UART_BDH_LBKDIE_SHIFT                    (7U)
#define UART_BDH_LBKDIE(x)                       (((uint8_t)(((uint8_t)(x)) << UART_BDH_LBKDIE_SHIFT)) &

/*! @name BDL - UART Baud Rate Register: Low */
#define UART_BDL_SBR_MASK                        (0xFFU)
#define UART_BDL_SBR_SHIFT                       (0U)
#define UART_BDL_SBR(x)                          (((uint8_t)(((uint8_t)(x)) << UART_BDL_SBR_SHIFT)) & UAR

/*! @name C1 - UART Control Register 1 */
#define UART_C1_PT_MASK                          (0x1U)
#define UART_C1_PT_SHIFT                         (0U)
#define UART_C1_PT(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C1_PT_SHIFT)) & UART_C1
#define UART_C1_PE_MASK                          (0x2U)
#define UART_C1_PE_SHIFT                         (1U)
#define UART_C1_PE(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C1_PE_SHIFT)) & UART_C1
#define UART_C1_ILT_MASK                         (0x4U)
#define UART_C1_ILT_SHIFT                        (2U)
#define UART_C1_ILT(x)                           (((uint8_t)(((uint8_t)(x)) << UART_C1_ILT_SHIFT)) & UART_C1
#define UART_C1_WAKE_MASK                        (0x8U)
#define UART_C1_WAKE_SHIFT                       (3U)
#define UART_C1_WAKE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C1_WAKE_SHIFT)) & UAF
#define UART_C1_M_MASK                           (0x10U)
#define UART_C1_M_SHIFT                          (4U)
#define UART_C1_M(x)                             (((uint8_t)(((uint8_t)(x)) << UART_C1_M_SHIFT)) & UART_C1_
#define UART_C1_RSRC_MASK                        (0x20U)
#define UART_C1_RSRC_SHIFT                       (5U)
#define UART_C1_RSRC(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C1_RSRC_SHIFT)) & UAR
#define UART_C1_UARTSWAI_MASK                    (0x40U)
#define UART_C1_UARTSWAI_SHIFT                   (6U)
#define UART_C1_UARTSWAI(x)                      (((uint8_t)(((uint8_t)(x)) << UART_C1_UARTSWAI_SHIFT
#define UART_C1_LOOPS_MASK                       (0x80U)
#define UART_C1_LOOPS_SHIFT                      (7U)
#define UART_C1_LOOPS(x)                         (((uint8_t)(((uint8_t)(x)) << UART_C1_LOOPS_SHIFT)) & U

/*! @name C2 - UART Control Register 2 */
#define UART_C2_SBK_MASK                         (0x1U)
#define UART_C2_SBK_SHIFT                        (0U)
#define UART_C2_SBK(x)                           (((uint8_t)(((uint8_t)(x)) << UART_C2_SBK_SHIFT)) & UART_
#define UART_C2_RWU_MASK                         (0x2U)
#define UART_C2_RWU_SHIFT                        (1U)
#define UART_C2_RWU(x)                           (((uint8_t)(((uint8_t)(x)) << UART_C2_RWU_SHIFT)) & UART
#define UART_C2_RE_MASK                          (0x4U)
#define UART_C2_RE_SHIFT                         (2U)
#define UART_C2_RE(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C2_RE_SHIFT)) & UART_C2
#define UART_C2_TE_MASK                          (0x8U)
#define UART_C2_TE_SHIFT                         (3U)
#define UART_C2_TE(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C2_TE_SHIFT)) & UART_C2
#define UART_C2_ILIE_MASK                        (0x10U)
#define UART_C2_ILIE_SHIFT                       (4U)
#define UART_C2_ILIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C2_ILIE_SHIFT)) & UART_C
#define UART_C2_RIE_MASK                         (0x20U)
#define UART_C2_RIE_SHIFT                        (5U)
#define UART_C2_RIE(x)                           (((uint8_t)(((uint8_t)(x)) << UART_C2_RIE_SHIFT)) & UART_C
#define UART_C2_TCIE_MASK                        (0x40U)
#define UART_C2_TCIE_SHIFT                       (6U)
#define UART_C2_TCIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C2_TCIE_SHIFT)) & UART_
#define UART_C2_TIE_MASK                         (0x80U)
#define UART_C2_TIE_SHIFT                        (7U)
#define UART_C2_TIE(x)                           (((uint8_t)(((uint8_t)(x)) << UART_C2_TIE_SHIFT)) & UART_C2

/*! @name S1 - UART Status Register 1 */
#define UART_S1_PF_MASK                          (0x1U)
#define UART_S1_PF_SHIFT                         (0U)
#define UART_S1_PF(x)                            (((uint8_t)(((uint8_t)(x)) << UART_S1_PF_SHIFT)) & UART_S1_
#define UART_S1_FE_MASK                          (0x2U)
#define UART_S1_FE_SHIFT                         (1U)
#define UART_S1_FE(x)                            (((uint8_t)(((uint8_t)(x)) << UART_S1_FE_SHIFT)) & UART_S1_
#define UART_S1_NF_MASK                          (0x4U)
#define UART_S1_NF_SHIFT                         (2U)
#define UART_S1_NF(x)                            (((uint8_t)(((uint8_t)(x)) << UART_S1_NF_SHIFT)) & UART_S1
#define UART_S1_OR_MASK                          (0x8U)
#define UART_S1_OR_SHIFT                         (3U)
#define UART_S1_OR(x)                            (((uint8_t)(((uint8_t)(x)) << UART_S1_OR_SHIFT)) & UART_S
#define UART_S1_IDLE_MASK                        (0x10U)
#define UART_S1_IDLE_SHIFT                       (4U)
#define UART_S1_IDLE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_S1_IDLE_SHIFT)) & UART_
#define UART_S1_RDRF_MASK                        (0x20U)
#define UART_S1_RDRF_SHIFT                       (5U)
#define UART_S1_RDRF(x)                          (((uint8_t)(((uint8_t)(x)) << UART_S1_RDRF_SHIFT)) & UAR
#define UART_S1_TC_MASK                          (0x40U)
#define UART_S1_TC_SHIFT                         (6U)
#define UART_S1_TC(x)                            (((uint8_t)(((uint8_t)(x)) << UART_S1_TC_SHIFT)) & UART_S1
#define UART_S1_TDRE_MASK                        (0x80U)
#define UART_S1_TDRE_SHIFT                       (7U)
#define UART_S1_TDRE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_S1_TDRE_SHIFT)) & UAR

/*! @name S2 - UART Status Register 2 */
#define UART_S2_RAF_MASK                         (0x1U)
#define UART_S2_RAF_SHIFT                        (0U)
#define UART_S2_RAF(x)                           (((uint8_t)(((uint8_t)(x)) << UART_S2_RAF_SHIFT)) & UART_
#define UART_S2_LBKDE_MASK                       (0x2U)
#define UART_S2_LBKDE_SHIFT                      (1U)
#define UART_S2_LBKDE(x)                         (((uint8_t)(((uint8_t)(x)) << UART_S2_LBKDE_SHIFT)) & UA
#define UART_S2_BRK13_MASK                       (0x4U)
#define UART_S2_BRK13_SHIFT                      (2U)
#define UART_S2_BRK13(x)                         (((uint8_t)(((uint8_t)(x)) << UART_S2_BRK13_SHIFT)) & UAR
#define UART_S2_RWUID_MASK                       (0x8U)
#define UART_S2_RWUID_SHIFT                      (3U)
#define UART_S2_RWUID(x)                         (((uint8_t)(((uint8_t)(x)) << UART_S2_RWUID_SHIFT)) & UA
#define UART_S2_RXINV_MASK                       (0x10U)
#define UART_S2_RXINV_SHIFT                      (4U)
#define UART_S2_RXINV(x)                         (((uint8_t)(((uint8_t)(x)) << UART_S2_RXINV_SHIFT)) & UAR
#define UART_S2_RXEDGIF_MASK                     (0x40U)
```

```
#define UART_S2_RXEDGIF_SHIFT                    (6U)
#define UART_S2_RXEDGIF(x)                       (((uint8_t)(((uint8_t)(x)) << UART_S2_RXEDGIF_SHIFT)) &
#define UART_S2_LBKDIF_MASK                      (0x80U)
#define UART_S2_LBKDIF_SHIFT                     (7U)
#define UART_S2_LBKDIF(x)                        (((uint8_t)(((uint8_t)(x)) << UART_S2_LBKDIF_SHIFT)) & UA

/*! @name C3 - UART Control Register 3 */
#define UART_C3_PEIE_MASK                        (0x1U)
#define UART_C3_PEIE_SHIFT                       (0U)
#define UART_C3_PEIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C3_PEIE_SHIFT)) & UART_
#define UART_C3_FEIE_MASK                        (0x2U)
#define UART_C3_FEIE_SHIFT                       (1U)
#define UART_C3_FEIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C3_FEIE_SHIFT)) & UART_
#define UART_C3_NEIE_MASK                        (0x4U)
#define UART_C3_NEIE_SHIFT                       (2U)
#define UART_C3_NEIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C3_NEIE_SHIFT)) & UART_
#define UART_C3_ORIE_MASK                        (0x8U)
#define UART_C3_ORIE_SHIFT                       (3U)
#define UART_C3_ORIE(x)                          (((uint8_t)(((uint8_t)(x)) << UART_C3_ORIE_SHIFT)) & UART_
#define UART_C3_TXINV_MASK                       (0x10U)
#define UART_C3_TXINV_SHIFT                      (4U)
#define UART_C3_TXINV(x)                         (((uint8_t)(((uint8_t)(x)) << UART_C3_TXINV_SHIFT)) & UAR
#define UART_C3_TXDIR_MASK                       (0x20U)
#define UART_C3_TXDIR_SHIFT                      (5U)
#define UART_C3_TXDIR(x)                         (((uint8_t)(((uint8_t)(x)) << UART_C3_TXDIR_SHIFT)) & UAR
#define UART_C3_T8_MASK                          (0x40U)
#define UART_C3_T8_SHIFT                         (6U)
#define UART_C3_T8(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C3_T8_SHIFT)) & UART_C3_
#define UART_C3_R8_MASK                          (0x80U)
#define UART_C3_R8_SHIFT                         (7U)
#define UART_C3_R8(x)                            (((uint8_t)(((uint8_t)(x)) << UART_C3_R8_SHIFT)) & UART_C3

/*! @name D - UART Data Register */
#define UART_D_R0T0_MASK                         (0x1U)
#define UART_D_R0T0_SHIFT                        (0U)
#define UART_D_R0T0(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R0T0_SHIFT)) & UART_D
#define UART_D_R1T1_MASK                         (0x2U)
#define UART_D_R1T1_SHIFT                        (1U)
#define UART_D_R1T1(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R1T1_SHIFT)) & UART_D
#define UART_D_R2T2_MASK                         (0x4U)
#define UART_D_R2T2_SHIFT                        (2U)
#define UART_D_R2T2(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R2T2_SHIFT)) & UART_D
#define UART_D_R3T3_MASK                         (0x8U)
#define UART_D_R3T3_SHIFT                        (3U)
#define UART_D_R3T3(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R3T3_SHIFT)) & UART_D
#define UART_D_R4T4_MASK                         (0x10U)
#define UART_D_R4T4_SHIFT                        (4U)
#define UART_D_R4T4(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R4T4_SHIFT)) & UART_D
#define UART_D_R5T5_MASK                         (0x20U)
#define UART_D_R5T5_SHIFT                        (5U)
#define UART_D_R5T5(x)                           (((uint8_t)(((uint8_t)(x)) << UART_D_R5T5_SHIFT)) & UART_D
#define UART_D_R6T6_MASK                         (0x40U)
#define UART_D_R6T6_SHIFT                        (6U)
```

```
#define UART_D_R6T6(x)                          (((uint8_t)(((uint8_t)(x)) << UART_D_R6T6_SHIFT)) & UART_D
#define UART_D_R7T7_MASK                         (0x80U)
#define UART_D_R7T7_SHIFT                        (7U)
#define UART_D_R7T7(x)                          (((uint8_t)(((uint8_t)(x)) << UART_D_R7T7_SHIFT)) & UART_D

/*! @name C4 - UART Control Register 4 */
#define UART_C4_RDMAS_MASK                       (0x20U)
#define UART_C4_RDMAS_SHIFT                      (5U)
#define UART_C4_RDMAS(x)                        (((uint8_t)(((uint8_t)(x)) << UART_C4_RDMAS_SHIFT)) & U
#define UART_C4_TDMAS_MASK                       (0x80U)
#define UART_C4_TDMAS_SHIFT                      (7U)
#define UART_C4_TDMAS(x)                        (((uint8_t)(((uint8_t)(x)) << UART_C4_TDMAS_SHIFT)) & U


/*!
 * @}
 */ /* 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)
/** Peripheral UART2 base address */
#define UART2_BASE                               (0x4006C000u)
/** Peripheral UART2 base pointer */
#define UART2                                    ((UART_Type *)UART2_BASE)
/** Array initializer of UART peripheral base addresses */
#define UART_BASE_ADDRS                          { 0u, UART1_BASE, UART2_BASE }
/** Array initializer of UART peripheral base pointers */
#define UART_BASE_PTRS                           { (UART_Type *)0u, UART1, UART2 }
/** Interrupt vectors for the UART peripheral type */
#define UART_RX_TX_IRQS                          { NotAvail_IRQn, UART1_IRQn, UART2_IRQn }
#define UART_ERR_IRQS                            { NotAvail_IRQn, UART1_IRQn, UART2_IRQn }

/*!
 * @}
 */ /* 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 Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup UART0_Register_Masks UART0 Register Masks
 * @{
 */

/*! @name BDH - UART Baud Rate Register High */
#define UART0_BDH_SBR_MASK                       (0x1FU)
#define UART0_BDH_SBR_SHIFT                      (0U)
#define UART0_BDH_SBR(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_BDH_SBR_SHIFT)) & UA
#define UART0_BDH_SBNS_MASK                      (0x20U)
#define UART0_BDH_SBNS_SHIFT                     (5U)
#define UART0_BDH_SBNS(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_BDH_SBNS_SHIFT)) &
#define UART0_BDH_RXEDGIE_MASK                   (0x40U)
#define UART0_BDH_RXEDGIE_SHIFT                  (6U)
#define UART0_BDH_RXEDGIE(x)                     (((uint8_t)(((uint8_t)(x)) << UART0_BDH_RXEDGIE_SHIF
#define UART0_BDH_LBKDIE_MASK                    (0x80U)
#define UART0_BDH_LBKDIE_SHIFT                   (7U)
#define UART0_BDH_LBKDIE(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_BDH_LBKDIE_SHIFT))

/*! @name BDL - UART Baud Rate Register Low */
#define UART0_BDL_SBR_MASK                       (0xFFU)
#define UART0_BDL_SBR_SHIFT                      (0U)
#define UART0_BDL_SBR(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_BDL_SBR_SHIFT)) & UA

/*! @name C1 - UART Control Register 1 */
#define UART0_C1_PT_MASK                         (0x1U)
#define UART0_C1_PT_SHIFT                        (0U)
#define UART0_C1_PT(x)                           (((uint8_t)(((uint8_t)(x)) << UART0_C1_PT_SHIFT)) & UART0_
#define UART0_C1_PE_MASK                         (0x2U)
#define UART0_C1_PE_SHIFT                        (1U)
#define UART0_C1_PE(x)                           (((uint8_t)(((uint8_t)(x)) << UART0_C1_PE_SHIFT)) & UART0_
#define UART0_C1_ILT_MASK                        (0x4U)
#define UART0_C1_ILT_SHIFT                       (2U)
#define UART0_C1_ILT(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_C1_ILT_SHIFT)) & UART0_
#define UART0_C1_WAKE_MASK                       (0x8U)
#define UART0_C1_WAKE_SHIFT                      (3U)
```

```
#define UART0_C1_WAKE(x)                    (((uint8_t)(((uint8_t)(x)) << UART0_C1_WAKE_SHIFT)) & UA
#define UART0_C1_M_MASK                      (0x10U)
#define UART0_C1_M_SHIFT                     (4U)
#define UART0_C1_M(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_C1_M_SHIFT)) & UART0_C
#define UART0_C1_RSRC_MASK                   (0x20U)
#define UART0_C1_RSRC_SHIFT                  (5U)
#define UART0_C1_RSRC(x)                     (((uint8_t)(((uint8_t)(x)) << UART0_C1_RSRC_SHIFT)) & UA
#define UART0_C1_DOZEEN_MASK                 (0x40U)
#define UART0_C1_DOZEEN_SHIFT                (6U)
#define UART0_C1_DOZEEN(x)                   (((uint8_t)(((uint8_t)(x)) << UART0_C1_DOZEEN_SHIFT))
#define UART0_C1_LOOPS_MASK                  (0x80U)
#define UART0_C1_LOOPS_SHIFT                 (7U)
#define UART0_C1_LOOPS(x)                    (((uint8_t)(((uint8_t)(x)) << UART0_C1_LOOPS_SHIFT)) & U

/*! @name C2 - UART Control Register 2 */
#define UART0_C2_SBK_MASK                    (0x1U)
#define UART0_C2_SBK_SHIFT                   (0U)
#define UART0_C2_SBK(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C2_SBK_SHIFT)) & UART
#define UART0_C2_RWU_MASK                    (0x2U)
#define UART0_C2_RWU_SHIFT                   (1U)
#define UART0_C2_RWU(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C2_RWU_SHIFT)) & UAR
#define UART0_C2_RE_MASK                     (0x4U)
#define UART0_C2_RE_SHIFT                    (2U)
#define UART0_C2_RE(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_C2_RE_SHIFT)) & UART0_
#define UART0_C2_TE_MASK                     (0x8U)
#define UART0_C2_TE_SHIFT                    (3U)
#define UART0_C2_TE(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_C2_TE_SHIFT)) & UART0_
#define UART0_C2_ILIE_MASK                   (0x10U)
#define UART0_C2_ILIE_SHIFT                  (4U)
#define UART0_C2_ILIE(x)                     (((uint8_t)(((uint8_t)(x)) << UART0_C2_ILIE_SHIFT)) & UART0
#define UART0_C2_RIE_MASK                    (0x20U)
#define UART0_C2_RIE_SHIFT                   (5U)
#define UART0_C2_RIE(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C2_RIE_SHIFT)) & UART0
#define UART0_C2_TCIE_MASK                   (0x40U)
#define UART0_C2_TCIE_SHIFT                  (6U)
#define UART0_C2_TCIE(x)                     (((uint8_t)(((uint8_t)(x)) << UART0_C2_TCIE_SHIFT)) & UART
#define UART0_C2_TIE_MASK                    (0x80U)
#define UART0_C2_TIE_SHIFT                   (7U)
#define UART0_C2_TIE(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C2_TIE_SHIFT)) & UART0_

/*! @name S1 - UART Status Register 1 */
#define UART0_S1_PF_MASK                     (0x1U)
#define UART0_S1_PF_SHIFT                    (0U)
#define UART0_S1_PF(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_S1_PF_SHIFT)) & UART0_
#define UART0_S1_FE_MASK                     (0x2U)
#define UART0_S1_FE_SHIFT                    (1U)
#define UART0_S1_FE(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_S1_FE_SHIFT)) & UART0_
#define UART0_S1_NF_MASK                     (0x4U)
#define UART0_S1_NF_SHIFT                    (2U)
#define UART0_S1_NF(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_S1_NF_SHIFT)) & UART0_
#define UART0_S1_OR_MASK                     (0x8U)
#define UART0_S1_OR_SHIFT                    (3U)
#define UART0_S1_OR(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_S1_OR_SHIFT)) & UART0_
#define UART0_S1_IDLE_MASK                       (0x10U)
#define UART0_S1_IDLE_SHIFT                      (4U)
#define UART0_S1_IDLE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_S1_IDLE_SHIFT)) & UART
#define UART0_S1_RDRF_MASK                       (0x20U)
#define UART0_S1_RDRF_SHIFT                      (5U)
#define UART0_S1_RDRF(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_S1_RDRF_SHIFT)) & UA
#define UART0_S1_TC_MASK                         (0x40U)
#define UART0_S1_TC_SHIFT                        (6U)
#define UART0_S1_TC(x)                           (((uint8_t)(((uint8_t)(x)) << UART0_S1_TC_SHIFT)) & UART0_
#define UART0_S1_TDRE_MASK                       (0x80U)
#define UART0_S1_TDRE_SHIFT                      (7U)
#define UART0_S1_TDRE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_S1_TDRE_SHIFT)) & UA

/*! @name S2 - UART Status Register 2 */
#define UART0_S2_RAF_MASK                        (0x1U)
#define UART0_S2_RAF_SHIFT                       (0U)
#define UART0_S2_RAF(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_S2_RAF_SHIFT)) & UART
#define UART0_S2_LBKDE_MASK                      (0x2U)
#define UART0_S2_LBKDE_SHIFT                     (1U)
#define UART0_S2_LBKDE(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_S2_LBKDE_SHIFT)) & U
#define UART0_S2_BRK13_MASK                      (0x4U)
#define UART0_S2_BRK13_SHIFT                     (2U)
#define UART0_S2_BRK13(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_S2_BRK13_SHIFT)) & U
#define UART0_S2_RWUID_MASK                      (0x8U)
#define UART0_S2_RWUID_SHIFT                     (3U)
#define UART0_S2_RWUID(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_S2_RWUID_SHIFT)) & U
#define UART0_S2_RXINV_MASK                      (0x10U)
#define UART0_S2_RXINV_SHIFT                     (4U)
#define UART0_S2_RXINV(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_S2_RXINV_SHIFT)) & UA
#define UART0_S2_MSBF_MASK                       (0x20U)
#define UART0_S2_MSBF_SHIFT                      (5U)
#define UART0_S2_MSBF(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_S2_MSBF_SHIFT)) & UA
#define UART0_S2_RXEDGIF_MASK                    (0x40U)
#define UART0_S2_RXEDGIF_SHIFT                   (6U)
#define UART0_S2_RXEDGIF(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_S2_RXEDGIF_SHIFT))
#define UART0_S2_LBKDIF_MASK                     (0x80U)
#define UART0_S2_LBKDIF_SHIFT                    (7U)
#define UART0_S2_LBKDIF(x)                       (((uint8_t)(((uint8_t)(x)) << UART0_S2_LBKDIF_SHIFT)) & U

/*! @name C3 - UART Control Register 3 */
#define UART0_C3_PEIE_MASK                       (0x1U)
#define UART0_C3_PEIE_SHIFT                      (0U)
#define UART0_C3_PEIE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_PEIE_SHIFT)) & UART
#define UART0_C3_FEIE_MASK                       (0x2U)
#define UART0_C3_FEIE_SHIFT                      (1U)
#define UART0_C3_FEIE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_FEIE_SHIFT)) & UART
#define UART0_C3_NEIE_MASK                       (0x4U)
#define UART0_C3_NEIE_SHIFT                      (2U)
#define UART0_C3_NEIE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_NEIE_SHIFT)) & UAR
#define UART0_C3_ORIE_MASK                       (0x8U)
#define UART0_C3_ORIE_SHIFT                      (3U)
#define UART0_C3_ORIE(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_ORIE_SHIFT)) & UAR
#define UART0_C3_TXINV_MASK                      (0x10U)
```

```
#define UART0_C3_TXINV_SHIFT                     (4U)
#define UART0_C3_TXINV(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_C3_TXINV_SHIFT)) & UA
#define UART0_C3_TXDIR_MASK                      (0x20U)
#define UART0_C3_TXDIR_SHIFT                     (5U)
#define UART0_C3_TXDIR(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_C3_TXDIR_SHIFT)) & UA
#define UART0_C3_R9T8_MASK                       (0x40U)
#define UART0_C3_R9T8_SHIFT                      (6U)
#define UART0_C3_R9T8(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_R9T8_SHIFT)) & UAR
#define UART0_C3_R8T9_MASK                       (0x80U)
#define UART0_C3_R8T9_SHIFT                      (7U)
#define UART0_C3_R8T9(x)                         (((uint8_t)(((uint8_t)(x)) << UART0_C3_R8T9_SHIFT)) & UAR

/*! @name D - UART Data Register */
#define UART0_D_R0T0_MASK                        (0x1U)
#define UART0_D_R0T0_SHIFT                       (0U)
#define UART0_D_R0T0(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R0T0_SHIFT)) & UART
#define UART0_D_R1T1_MASK                        (0x2U)
#define UART0_D_R1T1_SHIFT                       (1U)
#define UART0_D_R1T1(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R1T1_SHIFT)) & UART
#define UART0_D_R2T2_MASK                        (0x4U)
#define UART0_D_R2T2_SHIFT                       (2U)
#define UART0_D_R2T2(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R2T2_SHIFT)) & UART
#define UART0_D_R3T3_MASK                        (0x8U)
#define UART0_D_R3T3_SHIFT                       (3U)
#define UART0_D_R3T3(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R3T3_SHIFT)) & UART
#define UART0_D_R4T4_MASK                        (0x10U)
#define UART0_D_R4T4_SHIFT                       (4U)
#define UART0_D_R4T4(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R4T4_SHIFT)) & UART
#define UART0_D_R5T5_MASK                        (0x20U)
#define UART0_D_R5T5_SHIFT                       (5U)
#define UART0_D_R5T5(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R5T5_SHIFT)) & UART
#define UART0_D_R6T6_MASK                        (0x40U)
#define UART0_D_R6T6_SHIFT                       (6U)
#define UART0_D_R6T6(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R6T6_SHIFT)) & UART
#define UART0_D_R7T7_MASK                        (0x80U)
#define UART0_D_R7T7_SHIFT                       (7U)
#define UART0_D_R7T7(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_D_R7T7_SHIFT)) & UART

/*! @name MA1 - UART Match Address Registers 1 */
#define UART0_MA1_MA_MASK                        (0xFFU)
#define UART0_MA1_MA_SHIFT                       (0U)
#define UART0_MA1_MA(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_MA1_MA_SHIFT)) & UAR

/*! @name MA2 - UART Match Address Registers 2 */
#define UART0_MA2_MA_MASK                        (0xFFU)
#define UART0_MA2_MA_SHIFT                       (0U)
#define UART0_MA2_MA(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_MA2_MA_SHIFT)) & UAR

/*! @name C4 - UART Control Register 4 */
#define UART0_C4_OSR_MASK                        (0x1FU)
#define UART0_C4_OSR_SHIFT                       (0U)
#define UART0_C4_OSR(x)                          (((uint8_t)(((uint8_t)(x)) << UART0_C4_OSR_SHIFT)) & UAR
#define UART0_C4_M10_MASK                        (0x20U)
```

```
#define UART0_C4_M10_SHIFT                     (5U)
#define UART0_C4_M10(x)                        (((uint8_t)(((uint8_t)(x)) << UART0_C4_M10_SHIFT)) & UART
#define UART0_C4_MAEN2_MASK                    (0x40U)
#define UART0_C4_MAEN2_SHIFT                   (6U)
#define UART0_C4_MAEN2(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C4_MAEN2_SHIFT)) & U
#define UART0_C4_MAEN1_MASK                    (0x80U)
#define UART0_C4_MAEN1_SHIFT                   (7U)
#define UART0_C4_MAEN1(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C4_MAEN1_SHIFT)) & U

/*! @name C5 - UART Control Register 5 */
#define UART0_C5_RESYNCDIS_MASK                (0x1U)
#define UART0_C5_RESYNCDIS_SHIFT               (0U)
#define UART0_C5_RESYNCDIS(x)                  (((uint8_t)(((uint8_t)(x)) << UART0_C5_RESYNCDIS_SH
#define UART0_C5_BOTHEDGE_MASK                 (0x2U)
#define UART0_C5_BOTHEDGE_SHIFT                (1U)
#define UART0_C5_BOTHEDGE(x)                   (((uint8_t)(((uint8_t)(x)) << UART0_C5_BOTHEDGE_SH
#define UART0_C5_RDMAE_MASK                    (0x20U)
#define UART0_C5_RDMAE_SHIFT                   (5U)
#define UART0_C5_RDMAE(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C5_RDMAE_SHIFT)) &
#define UART0_C5_TDMAE_MASK                    (0x80U)
#define UART0_C5_TDMAE_SHIFT                   (7U)
#define UART0_C5_TDMAE(x)                      (((uint8_t)(((uint8_t)(x)) << UART0_C5_TDMAE_SHIFT)) &


/*!
 * @}
 */ /* 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)
/** 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 }
/** Interrupt vectors for the UART0 peripheral type */
#define UART0_RX_TX_IRQS                         { UART0_IRQn }
#define UART0_ERR_IRQS                           { UART0_IRQn }

/*!
 * @}
 */ /* 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: 0x
         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 USBTRC0;                          /**< USB Transceiver Control Register 0, offset: 0x10C */
       uint8_t RESERVED_25[7];
  __IO uint8_t USBFRMADJUST;                     /**< Frame Adjust Register, offset: 0x114 */
} USB_Type;

/* ----------------------------------------------------------------------------
   -- USB Register Masks
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup USB_Register_Masks USB Register Masks
 * @{
 */

/*! @name PERID - Peripheral ID register */
#define USB_PERID_ID_MASK                        (0x3FU)
#define USB_PERID_ID_SHIFT                       (0U)
#define USB_PERID_ID(x)                          (((uint8_t)(((uint8_t)(x)) << USB_PERID_ID_SHIFT)) & USB_P

/*! @name IDCOMP - Peripheral ID Complement register */
#define USB_IDCOMP_NID_MASK                      (0x3FU)
#define USB_IDCOMP_NID_SHIFT                     (0U)
#define USB_IDCOMP_NID(x)                        (((uint8_t)(((uint8_t)(x)) << USB_IDCOMP_NID_SHIFT)) & U

/*! @name REV - Peripheral Revision register */
#define USB_REV_REV_MASK                         (0xFFU)
#define USB_REV_REV_SHIFT                        (0U)
#define USB_REV_REV(x)                           (((uint8_t)(((uint8_t)(x)) << USB_REV_REV_SHIFT)) & USB_F

/*! @name ADDINFO - Peripheral Additional Info register */
#define USB_ADDINFO_IEHOST_MASK                  (0x1U)
#define USB_ADDINFO_IEHOST_SHIFT                 (0U)
#define USB_ADDINFO_IEHOST(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ADDINFO_IEHOST_SH
#define USB_ADDINFO_IRQNUM_MASK                  (0xF8U)
#define USB_ADDINFO_IRQNUM_SHIFT                 (3U)
#define USB_ADDINFO_IRQNUM(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ADDINFO_IRQNUM_S

/*! @name OTGISTAT - OTG Interrupt Status register */
#define USB_OTGISTAT_AVBUSCHG_MASK               (0x1U)
#define USB_OTGISTAT_AVBUSCHG_SHIFT              (0U)
#define USB_OTGISTAT_AVBUSCHG(x)                 (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_AVBUSC
#define USB_OTGISTAT_B_SESS_CHG_MASK             (0x4U)
#define USB_OTGISTAT_B_SESS_CHG_SHIFT            (2U)
#define USB_OTGISTAT_B_SESS_CHG(x)               (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_B_SES
#define USB_OTGISTAT_SESSVLDCHG_MASK             (0x8U)
#define USB_OTGISTAT_SESSVLDCHG_SHIFT            (3U)
#define USB_OTGISTAT_SESSVLDCHG(x)               (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_SESSV
#define USB_OTGISTAT_LINE_STATE_CHG_MASK         (0x20U)
#define USB_OTGISTAT_LINE_STATE_CHG_SHIFT        (5U)
#define USB_OTGISTAT_LINE_STATE_CHG(x)           (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_LINE
#define USB_OTGISTAT_ONEMSEC_MASK                (0x40U)
#define USB_OTGISTAT_ONEMSEC_SHIFT               (6U)
#define USB_OTGISTAT_ONEMSEC(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_ONEMSE
#define USB_OTGISTAT_IDCHG_MASK                  (0x80U)
#define USB_OTGISTAT_IDCHG_SHIFT                 (7U)
#define USB_OTGISTAT_IDCHG(x)                    (((uint8_t)(((uint8_t)(x)) << USB_OTGISTAT_IDCHG_SH

/*! @name OTGICR - OTG Interrupt Control Register */
#define USB_OTGICR_AVBUSEN_MASK                  (0x1U)
#define USB_OTGICR_AVBUSEN_SHIFT                 (0U)
#define USB_OTGICR_AVBUSEN(x)                    (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_AVBUSEN_S
#define USB_OTGICR_BSESSEN_MASK                  (0x4U)
#define USB_OTGICR_BSESSEN_SHIFT                 (2U)
#define USB_OTGICR_BSESSEN(x)                    (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_BSESSEN_S
#define USB_OTGICR_SESSVLDEN_MASK                (0x8U)
#define USB_OTGICR_SESSVLDEN_SHIFT               (3U)
#define USB_OTGICR_SESSVLDEN(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_SESSVLDE
#define USB_OTGICR_LINESTATEEN_MASK              (0x20U)
#define USB_OTGICR_LINESTATEEN_SHIFT             (5U)
#define USB_OTGICR_LINESTATEEN(x)                (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_LINESTATE
#define USB_OTGICR_ONEMSECEN_MASK                (0x40U)
#define USB_OTGICR_ONEMSECEN_SHIFT               (6U)
#define USB_OTGICR_ONEMSECEN(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_ONEMSEC
#define USB_OTGICR_IDEN_MASK                     (0x80U)
#define USB_OTGICR_IDEN_SHIFT                    (7U)
#define USB_OTGICR_IDEN(x)                       (((uint8_t)(((uint8_t)(x)) << USB_OTGICR_IDEN_SHIFT)) &

/*! @name OTGSTAT - OTG Status register */
#define USB_OTGSTAT_AVBUSVLD_MASK                (0x1U)
#define USB_OTGSTAT_AVBUSVLD_SHIFT               (0U)
#define USB_OTGSTAT_AVBUSVLD(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_AVBUSVL
#define USB_OTGSTAT_BSESSEND_MASK                (0x4U)
#define USB_OTGSTAT_BSESSEND_SHIFT               (2U)
#define USB_OTGSTAT_BSESSEND(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_BSESSE
#define USB_OTGSTAT_SESS_VLD_MASK                (0x8U)
#define USB_OTGSTAT_SESS_VLD_SHIFT               (3U)
#define USB_OTGSTAT_SESS_VLD(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_SESS_VL
#define USB_OTGSTAT_LINESTATESTABLE_MASK         (0x20U)
#define USB_OTGSTAT_LINESTATESTABLE_SHIFT        (5U)
#define USB_OTGSTAT_LINESTATESTABLE(x)           (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_LINE
#define USB_OTGSTAT_ONEMSECEN_MASK               (0x40U)
#define USB_OTGSTAT_ONEMSECEN_SHIFT              (6U)
#define USB_OTGSTAT_ONEMSECEN(x)                 (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_ONEMS
#define USB_OTGSTAT_ID_MASK                      (0x80U)
#define USB_OTGSTAT_ID_SHIFT                     (7U)
#define USB_OTGSTAT_ID(x)                        (((uint8_t)(((uint8_t)(x)) << USB_OTGSTAT_ID_SHIFT)) & U

/*! @name OTGCTL - OTG Control register */
#define USB_OTGCTL_OTGEN_MASK                    (0x4U)
#define USB_OTGCTL_OTGEN_SHIFT                   (2U)
```

```
#define USB_OTGCTL_OTGEN(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGCTL_OTGEN_SHIF
#define USB_OTGCTL_DMLOW_MASK                    (0x10U)
#define USB_OTGCTL_DMLOW_SHIFT                   (4U)
#define USB_OTGCTL_DMLOW(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGCTL_DMLOW_SHI
#define USB_OTGCTL_DPLOW_MASK                    (0x20U)
#define USB_OTGCTL_DPLOW_SHIFT                   (5U)
#define USB_OTGCTL_DPLOW(x)                  (((uint8_t)(((uint8_t)(x)) << USB_OTGCTL_DPLOW_SHIF
#define USB_OTGCTL_DPHIGH_MASK                   (0x80U)
#define USB_OTGCTL_DPHIGH_SHIFT                  (7U)
#define USB_OTGCTL_DPHIGH(x)                 (((uint8_t)(((uint8_t)(x)) << USB_OTGCTL_DPHIGH_SHI

/*! @name ISTAT - Interrupt Status register */
#define USB_ISTAT_USBRST_MASK                  (0x1U)
#define USB_ISTAT_USBRST_SHIFT                 (0U)
#define USB_ISTAT_USBRST(x)                (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_USBRST_SHIFT))
#define USB_ISTAT_ERROR_MASK                   (0x2U)
#define USB_ISTAT_ERROR_SHIFT                  (1U)
#define USB_ISTAT_ERROR(x)                 (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_ERROR_SHIFT)) &
#define USB_ISTAT_SOFTOK_MASK                  (0x4U)
#define USB_ISTAT_SOFTOK_SHIFT                 (2U)
#define USB_ISTAT_SOFTOK(x)                (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_SOFTOK_SHIFT))
#define USB_ISTAT_TOKDNE_MASK                  (0x8U)
#define USB_ISTAT_TOKDNE_SHIFT                 (3U)
#define USB_ISTAT_TOKDNE(x)                (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_TOKDNE_SHIFT))
#define USB_ISTAT_SLEEP_MASK                   (0x10U)
#define USB_ISTAT_SLEEP_SHIFT                  (4U)
#define USB_ISTAT_SLEEP(x)                 (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_SLEEP_SHIFT)) & U
#define USB_ISTAT_RESUME_MASK                  (0x20U)
#define USB_ISTAT_RESUME_SHIFT                 (5U)
#define USB_ISTAT_RESUME(x)                (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_RESUME_SHIFT)
#define USB_ISTAT_ATTACH_MASK                  (0x40U)
#define USB_ISTAT_ATTACH_SHIFT                 (6U)
#define USB_ISTAT_ATTACH(x)                (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_ATTACH_SHIFT))
#define USB_ISTAT_STALL_MASK                   (0x80U)
#define USB_ISTAT_STALL_SHIFT                  (7U)
#define USB_ISTAT_STALL(x)                 (((uint8_t)(((uint8_t)(x)) << USB_ISTAT_STALL_SHIFT)) & U

/*! @name INTEN - Interrupt Enable register */
#define USB_INTEN_USBRSTEN_MASK                  (0x1U)
#define USB_INTEN_USBRSTEN_SHIFT                 (0U)
#define USB_INTEN_USBRSTEN(x)                (((uint8_t)(((uint8_t)(x)) << USB_INTEN_USBRSTEN_SH
#define USB_INTEN_ERROREN_MASK                   (0x2U)
#define USB_INTEN_ERROREN_SHIFT                  (1U)
#define USB_INTEN_ERROREN(x)                 (((uint8_t)(((uint8_t)(x)) << USB_INTEN_ERROREN_SHI
#define USB_INTEN_SOFTOKEN_MASK                  (0x4U)
#define USB_INTEN_SOFTOKEN_SHIFT                 (2U)
#define USB_INTEN_SOFTOKEN(x)                (((uint8_t)(((uint8_t)(x)) << USB_INTEN_SOFTOKEN_SH
#define USB_INTEN_TOKDNEEN_MASK                  (0x8U)
#define USB_INTEN_TOKDNEEN_SHIFT                 (3U)
#define USB_INTEN_TOKDNEEN(x)                (((uint8_t)(((uint8_t)(x)) << USB_INTEN_TOKDNEEN_SH
#define USB_INTEN_SLEEPEN_MASK                   (0x10U)
#define USB_INTEN_SLEEPEN_SHIFT                  (4U)
#define USB_INTEN_SLEEPEN(x)                 (((uint8_t)(((uint8_t)(x)) << USB_INTEN_SLEEPEN_SHIF
#define USB_INTEN_RESUMEEN_MASK                 (0x20U)
#define USB_INTEN_RESUMEEN_SHIFT                (5U)
#define USB_INTEN_RESUMEEN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_INTEN_RESUMEEN_S
#define USB_INTEN_ATTACHEN_MASK                 (0x40U)
#define USB_INTEN_ATTACHEN_SHIFT                (6U)
#define USB_INTEN_ATTACHEN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_INTEN_ATTACHEN_SH
#define USB_INTEN_STALLEN_MASK                  (0x80U)
#define USB_INTEN_STALLEN_SHIFT                 (7U)
#define USB_INTEN_STALLEN(x)                    (((uint8_t)(((uint8_t)(x)) << USB_INTEN_STALLEN_SHIFT

/*! @name ERRSTAT - Error Interrupt Status register */
#define USB_ERRSTAT_PIDERR_MASK                 (0x1U)
#define USB_ERRSTAT_PIDERR_SHIFT                (0U)
#define USB_ERRSTAT_PIDERR(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_PIDERR_SH
#define USB_ERRSTAT_CRC5EOF_MASK                (0x2U)
#define USB_ERRSTAT_CRC5EOF_SHIFT               (1U)
#define USB_ERRSTAT_CRC5EOF(x)                  (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_CRC5EOF
#define USB_ERRSTAT_CRC16_MASK                  (0x4U)
#define USB_ERRSTAT_CRC16_SHIFT                 (2U)
#define USB_ERRSTAT_CRC16(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_CRC16_SHI
#define USB_ERRSTAT_DFN8_MASK                   (0x8U)
#define USB_ERRSTAT_DFN8_SHIFT                  (3U)
#define USB_ERRSTAT_DFN8(x)                     (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_DFN8_SHIFT
#define USB_ERRSTAT_BTOERR_MASK                 (0x10U)
#define USB_ERRSTAT_BTOERR_SHIFT                (4U)
#define USB_ERRSTAT_BTOERR(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_BTOERR_S
#define USB_ERRSTAT_DMAERR_MASK                 (0x20U)
#define USB_ERRSTAT_DMAERR_SHIFT                (5U)
#define USB_ERRSTAT_DMAERR(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_DMAERR_
#define USB_ERRSTAT_BTSERR_MASK                 (0x80U)
#define USB_ERRSTAT_BTSERR_SHIFT                (7U)
#define USB_ERRSTAT_BTSERR(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERRSTAT_BTSERR_S

/*! @name ERREN - Error Interrupt Enable register */
#define USB_ERREN_PIDERREN_MASK                 (0x1U)
#define USB_ERREN_PIDERREN_SHIFT                (0U)
#define USB_ERREN_PIDERREN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERREN_PIDERREN_S
#define USB_ERREN_CRC5EOFEN_MASK                (0x2U)
#define USB_ERREN_CRC5EOFEN_SHIFT               (1U)
#define USB_ERREN_CRC5EOFEN(x)                  (((uint8_t)(((uint8_t)(x)) << USB_ERREN_CRC5EOFE
#define USB_ERREN_CRC16EN_MASK                  (0x4U)
#define USB_ERREN_CRC16EN_SHIFT                 (2U)
#define USB_ERREN_CRC16EN(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ERREN_CRC16EN_SH
#define USB_ERREN_DFN8EN_MASK                   (0x8U)
#define USB_ERREN_DFN8EN_SHIFT                  (3U)
#define USB_ERREN_DFN8EN(x)                     (((uint8_t)(((uint8_t)(x)) << USB_ERREN_DFN8EN_SHIF
#define USB_ERREN_BTOERREN_MASK                 (0x10U)
#define USB_ERREN_BTOERREN_SHIFT                (4U)
#define USB_ERREN_BTOERREN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERREN_BTOERREN_
#define USB_ERREN_DMAERREN_MASK                 (0x20U)
#define USB_ERREN_DMAERREN_SHIFT                (5U)
#define USB_ERREN_DMAERREN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERREN_DMAERREN_
#define USB_ERREN_BTSERREN_MASK                 (0x80U)
```

```
#define USB_ERREN_BTSERREN_SHIFT                (7U)
#define USB_ERREN_BTSERREN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ERREN_BTSERREN_

/*! @name STAT - Status register */
#define USB_STAT_ODD_MASK                       (0x4U)
#define USB_STAT_ODD_SHIFT                      (2U)
#define USB_STAT_ODD(x)                         (((uint8_t)(((uint8_t)(x)) << USB_STAT_ODD_SHIFT)) & USB
#define USB_STAT_TX_MASK                        (0x8U)
#define USB_STAT_TX_SHIFT                       (3U)
#define USB_STAT_TX(x)                          (((uint8_t)(((uint8_t)(x)) << USB_STAT_TX_SHIFT)) & USB_S
#define USB_STAT_ENDP_MASK                      (0xF0U)
#define USB_STAT_ENDP_SHIFT                     (4U)
#define USB_STAT_ENDP(x)                        (((uint8_t)(((uint8_t)(x)) << USB_STAT_ENDP_SHIFT)) & US

/*! @name CTL - Control register */
#define USB_CTL_USBENSOFEN_MASK                 (0x1U)
#define USB_CTL_USBENSOFEN_SHIFT                (0U)
#define USB_CTL_USBENSOFEN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_CTL_USBENSOFEN_S
#define USB_CTL_ODDRST_MASK                     (0x2U)
#define USB_CTL_ODDRST_SHIFT                    (1U)
#define USB_CTL_ODDRST(x)                       (((uint8_t)(((uint8_t)(x)) << USB_CTL_ODDRST_SHIFT)) &
#define USB_CTL_RESUME_MASK                     (0x4U)
#define USB_CTL_RESUME_SHIFT                    (2U)
#define USB_CTL_RESUME(x)                       (((uint8_t)(((uint8_t)(x)) << USB_CTL_RESUME_SHIFT)) &
#define USB_CTL_HOSTMODEEN_MASK                 (0x8U)
#define USB_CTL_HOSTMODEEN_SHIFT                (3U)
#define USB_CTL_HOSTMODEEN(x)                   (((uint8_t)(((uint8_t)(x)) << USB_CTL_HOSTMODEEN_
#define USB_CTL_RESET_MASK                      (0x10U)
#define USB_CTL_RESET_SHIFT                     (4U)
#define USB_CTL_RESET(x)                        (((uint8_t)(((uint8_t)(x)) << USB_CTL_RESET_SHIFT)) & US
#define USB_CTL_TXSUSPENDTOKENBUSY_MASK         (0x20U)
#define USB_CTL_TXSUSPENDTOKENBUSY_SHIFT        (5U)
#define USB_CTL_TXSUSPENDTOKENBUSY(x)           (((uint8_t)(((uint8_t)(x)) << USB_CTL_TXSUSPE
#define USB_CTL_SE0_MASK                        (0x40U)
#define USB_CTL_SE0_SHIFT                       (6U)
#define USB_CTL_SE0(x)                          (((uint8_t)(((uint8_t)(x)) << USB_CTL_SE0_SHIFT)) & USB_CT
#define USB_CTL_JSTATE_MASK                     (0x80U)
#define USB_CTL_JSTATE_SHIFT                    (7U)
#define USB_CTL_JSTATE(x)                       (((uint8_t)(((uint8_t)(x)) << USB_CTL_JSTATE_SHIFT)) & U

/*! @name ADDR - Address register */
#define USB_ADDR_ADDR_MASK                      (0x7FU)
#define USB_ADDR_ADDR_SHIFT                     (0U)
#define USB_ADDR_ADDR(x)                        (((uint8_t)(((uint8_t)(x)) << USB_ADDR_ADDR_SHIFT)) & U
#define USB_ADDR_LSEN_MASK                      (0x80U)
#define USB_ADDR_LSEN_SHIFT                     (7U)
#define USB_ADDR_LSEN(x)                        (((uint8_t)(((uint8_t)(x)) << USB_ADDR_LSEN_SHIFT)) & US

/*! @name BDTPAGE1 - BDT Page Register 1 */
#define USB_BDTPAGE1_BDTBA_MASK                 (0xFEU)
#define USB_BDTPAGE1_BDTBA_SHIFT                (1U)
#define USB_BDTPAGE1_BDTBA(x)                   (((uint8_t)(((uint8_t)(x)) << USB_BDTPAGE1_BDTBA_S
/*! @name FRMNUML - Frame Number Register Low */
#define USB_FRMNUML_FRM_MASK                     (0xFFU)
#define USB_FRMNUML_FRM_SHIFT                    (0U)
#define USB_FRMNUML_FRM(x)                       (((uint8_t)(((uint8_t)(x)) << USB_FRMNUML_FRM_SHIFT

/*! @name FRMNUMH - Frame Number Register High */
#define USB_FRMNUMH_FRM_MASK                     (0x7U)
#define USB_FRMNUMH_FRM_SHIFT                    (0U)
#define USB_FRMNUMH_FRM(x)                       (((uint8_t)(((uint8_t)(x)) << USB_FRMNUMH_FRM_SHIF

/*! @name TOKEN - Token register */
#define USB_TOKEN_TOKENENDPT_MASK                (0xFU)
#define USB_TOKEN_TOKENENDPT_SHIFT               (0U)
#define USB_TOKEN_TOKENENDPT(x)                  (((uint8_t)(((uint8_t)(x)) << USB_TOKEN_TOKENEND
#define USB_TOKEN_TOKENPID_MASK                  (0xF0U)
#define USB_TOKEN_TOKENPID_SHIFT                 (4U)
#define USB_TOKEN_TOKENPID(x)                    (((uint8_t)(((uint8_t)(x)) << USB_TOKEN_TOKENPID_S

/*! @name SOFTHLD - SOF Threshold Register */
#define USB_SOFTHLD_CNT_MASK                     (0xFFU)
#define USB_SOFTHLD_CNT_SHIFT                    (0U)
#define USB_SOFTHLD_CNT(x)                       (((uint8_t)(((uint8_t)(x)) << USB_SOFTHLD_CNT_SHIFT))

/*! @name BDTPAGE2 - BDT Page Register 2 */
#define USB_BDTPAGE2_BDTBA_MASK                  (0xFFU)
#define USB_BDTPAGE2_BDTBA_SHIFT                 (0U)
#define USB_BDTPAGE2_BDTBA(x)                    (((uint8_t)(((uint8_t)(x)) << USB_BDTPAGE2_BDTBA_S

/*! @name BDTPAGE3 - BDT Page Register 3 */
#define USB_BDTPAGE3_BDTBA_MASK                  (0xFFU)
#define USB_BDTPAGE3_BDTBA_SHIFT                 (0U)
#define USB_BDTPAGE3_BDTBA(x)                    (((uint8_t)(((uint8_t)(x)) << USB_BDTPAGE3_BDTBA_S

/*! @name ENDPT - Endpoint Control register */
#define USB_ENDPT_EPHSHK_MASK                    (0x1U)
#define USB_ENDPT_EPHSHK_SHIFT                   (0U)
#define USB_ENDPT_EPHSHK(x)                      (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_EPHSHK_SHIF
#define USB_ENDPT_EPSTALL_MASK                   (0x2U)
#define USB_ENDPT_EPSTALL_SHIFT                  (1U)
#define USB_ENDPT_EPSTALL(x)                     (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_EPSTALL_SHIF
#define USB_ENDPT_EPTXEN_MASK                    (0x4U)
#define USB_ENDPT_EPTXEN_SHIFT                   (2U)
#define USB_ENDPT_EPTXEN(x)                      (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_EPTXEN_SHIFT
#define USB_ENDPT_EPRXEN_MASK                    (0x8U)
#define USB_ENDPT_EPRXEN_SHIFT                   (3U)
#define USB_ENDPT_EPRXEN(x)                      (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_EPRXEN_SHIF
#define USB_ENDPT_EPCTLDIS_MASK                  (0x10U)
#define USB_ENDPT_EPCTLDIS_SHIFT                 (4U)
#define USB_ENDPT_EPCTLDIS(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_EPCTLDIS_SH
#define USB_ENDPT_RETRYDIS_MASK                  (0x40U)
#define USB_ENDPT_RETRYDIS_SHIFT                 (6U)
#define USB_ENDPT_RETRYDIS(x)                    (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_RETRYDIS_SH
#define USB_ENDPT_HOSTWOHUB_MASK                 (0x80U)
```

```
#define USB_ENDPT_HOSTWOHUB_SHIFT                (7U)
#define USB_ENDPT_HOSTWOHUB(x)                   (((uint8_t)(((uint8_t)(x)) << USB_ENDPT_HOSTWOHU

/* The count of USB_ENDPT */
#define USB_ENDPT_COUNT                          (16U)

/*! @name USBCTRL - USB Control register */
#define USB_USBCTRL_PDE_MASK                     (0x40U)
#define USB_USBCTRL_PDE_SHIFT                    (6U)
#define USB_USBCTRL_PDE(x)                       (((uint8_t)(((uint8_t)(x)) << USB_USBCTRL_PDE_SHIFT))
#define USB_USBCTRL_SUSP_MASK                    (0x80U)
#define USB_USBCTRL_SUSP_SHIFT                   (7U)
#define USB_USBCTRL_SUSP(x)                      (((uint8_t)(((uint8_t)(x)) << USB_USBCTRL_SUSP_SHIFT

/*! @name OBSERVE - USB OTG Observe register */
#define USB_OBSERVE_DMPD_MASK                    (0x10U)
#define USB_OBSERVE_DMPD_SHIFT                   (4U)
#define USB_OBSERVE_DMPD(x)                      (((uint8_t)(((uint8_t)(x)) << USB_OBSERVE_DMPD_SHI
#define USB_OBSERVE_DPPD_MASK                    (0x40U)
#define USB_OBSERVE_DPPD_SHIFT                   (6U)
#define USB_OBSERVE_DPPD(x)                      (((uint8_t)(((uint8_t)(x)) << USB_OBSERVE_DPPD_SHIF
#define USB_OBSERVE_DPPU_MASK                    (0x80U)
#define USB_OBSERVE_DPPU_SHIFT                   (7U)
#define USB_OBSERVE_DPPU(x)                      (((uint8_t)(((uint8_t)(x)) << USB_OBSERVE_DPPU_SHIF

/*! @name CONTROL - USB OTG Control register */
#define USB_CONTROL_DPPULLUPNONOTG_MASK          (0x10U)
#define USB_CONTROL_DPPULLUPNONOTG_SHIFT         (4U)
#define USB_CONTROL_DPPULLUPNONOTG(x)            (((uint8_t)(((uint8_t)(x)) << USB_CONTROL_DP

/*! @name USBTRC0 - USB Transceiver Control Register 0 */
#define USB_USBTRC0_USB_RESUME_INT_MASK          (0x1U)
#define USB_USBTRC0_USB_RESUME_INT_SHIFT         (0U)
#define USB_USBTRC0_USB_RESUME_INT(x)            (((uint8_t)(((uint8_t)(x)) << USB_USBTRC0_USB_
#define USB_USBTRC0_SYNC_DET_MASK                (0x2U)
#define USB_USBTRC0_SYNC_DET_SHIFT               (1U)
#define USB_USBTRC0_SYNC_DET(x)                  (((uint8_t)(((uint8_t)(x)) << USB_USBTRC0_SYNC_DE
#define USB_USBTRC0_USBRESMEN_MASK               (0x20U)
#define USB_USBTRC0_USBRESMEN_SHIFT              (5U)
#define USB_USBTRC0_USBRESMEN(x)                 (((uint8_t)(((uint8_t)(x)) << USB_USBTRC0_USBRES
#define USB_USBTRC0_USBRESET_MASK                (0x80U)
#define USB_USBTRC0_USBRESET_SHIFT               (7U)
#define USB_USBTRC0_USBRESET(x)                  (((uint8_t)(((uint8_t)(x)) << USB_USBTRC0_USBRESE

/*! @name USBFRMADJUST - Frame Adjust Register */
#define USB_USBFRMADJUST_ADJ_MASK                (0xFFU)
#define USB_USBFRMADJUST_ADJ_SHIFT               (0U)
#define USB_USBFRMADJUST_ADJ(x)                  (((uint8_t)(((uint8_t)(x)) << USB_USBFRMADJUST_AD


/*!
 * @}
 */ /* end of group USB_Register_Masks */
```

```
/* USB - Peripheral instance base addresses */
/** Peripheral USB0 base address */
#define USB0_BASE                                (0x40072000u)
/** Peripheral USB0 base pointer */
#define USB0                                     ((USB_Type *)USB0_BASE)
/** Array initializer of USB peripheral base addresses */
#define USB_BASE_ADDRS                           { USB0_BASE }
/** Array initializer of USB peripheral base pointers */
#define USB_BASE_PTRS                            { USB0 }
/** Interrupt vectors for the USB peripheral type */
#define USB_IRQS                                 { USB0_IRQn }

/*!
 * @}
 */ /* 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 */


/* ----------------------------------------------------------------------------
   -- Macros for use with bit field definitions (xxx_SHIFT, xxx_MASK).
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup Bit_Field_Generic_Macros Macros for use with bit field definitions (xxx_SHIFT, xxx_MASK
 * @{
 */

#if defined(__ARMCC_VERSION)
  #if (__ARMCC_VERSION >= 6010050)
    #pragma clang system_header
  #endif
```

```
#elif defined(__IAR_SYSTEMS_ICC__)
  #pragma system_include
#endif

/**
 * @brief Mask and left-shift a bit field value for use in a register bit range.
 * @param field Name of the register bit field.
 * @param value Value of the bit field.
 * @return Masked and shifted value.
 */
#define NXP_VAL2FLD(field, value)    (((value) << (field ## _SHIFT)) & (field ## _MASK))
/**
 * @brief Mask and right-shift a register value to extract a bit field value.
 * @param field Name of the register bit field.
 * @param value Value of the register.
 * @return Masked and shifted bit field value.
 */
#define NXP_FLD2VAL(field, value)    (((value) & (field ## _MASK)) >> (field ## _SHIFT))

/*!
 * @}
 */ /* end of group Bit_Field_Generic_Macros */


/* ----------------------------------------------------------------------------
   -- SDK Compatibility
   ---------------------------------------------------------------------------- */

/*!
 * @addtogroup SDK_Compatibility_Symbols SDK 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 UARTLP_BDH_SBNS_SHIFT          UART0_BDH_SBNS_SHIFT
#define UARTLP_BDH_RXEDGIE_MASK        UART0_BDH_RXEDGIE_MASK
#define UARTLP_BDH_RXEDGIE_SHIFT       UART0_BDH_RXEDGIE_SHIFT
#define UARTLP_BDH_LBKDIE_MASK         UART0_BDH_LBKDIE_MASK
#define UARTLP_BDH_LBKDIE_SHIFT        UART0_BDH_LBKDIE_SHIFT
#define UARTLP_BDL_SBR_MASK            UART0_BDL_SBR_MASK
#define UARTLP_BDL_SBR_SHIFT           UART0_BDL_SBR_SHIFT
#define UARTLP_BDL_SBR(x)              UART0_BDL_SBR(x)
#define UARTLP_C1_PT_MASK              UART0_C1_PT_MASK
#define UARTLP_C1_PT_SHIFT             UART0_C1_PT_SHIFT
#define UARTLP_C1_PE_MASK              UART0_C1_PE_MASK
#define UARTLP_C1_PE_SHIFT             UART0_C1_PE_SHIFT
#define UARTLP_C1_ILT_MASK             UART0_C1_ILT_MASK
#define UARTLP_C1_ILT_SHIFT            UART0_C1_ILT_SHIFT
#define UARTLP_C1_WAKE_MASK            UART0_C1_WAKE_MASK
#define UARTLP_C1_WAKE_SHIFT           UART0_C1_WAKE_SHIFT
#define UARTLP_C1_M_MASK               UART0_C1_M_MASK
#define UARTLP_C1_M_SHIFT              UART0_C1_M_SHIFT
#define UARTLP_C1_RSRC_MASK            UART0_C1_RSRC_MASK
#define UARTLP_C1_RSRC_SHIFT           UART0_C1_RSRC_SHIFT
#define UARTLP_C1_DOZEEN_MASK          UART0_C1_DOZEEN_MASK
#define UARTLP_C1_DOZEEN_SHIFT         UART0_C1_DOZEEN_SHIFT
#define UARTLP_C1_LOOPS_MASK           UART0_C1_LOOPS_MASK
#define UARTLP_C1_LOOPS_SHIFT          UART0_C1_LOOPS_SHIFT
#define UARTLP_C2_SBK_MASK             UART0_C2_SBK_MASK
#define UARTLP_C2_SBK_SHIFT            UART0_C2_SBK_SHIFT
#define UARTLP_C2_RWU_MASK             UART0_C2_RWU_MASK
#define UARTLP_C2_RWU_SHIFT            UART0_C2_RWU_SHIFT
#define UARTLP_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 UARTLP_S2_LBKDE_MASK                   UART0_S2_LBKDE_MASK
#define UARTLP_S2_LBKDE_SHIFT                  UART0_S2_LBKDE_SHIFT
#define UARTLP_S2_BRK13_MASK                   UART0_S2_BRK13_MASK
#define UARTLP_S2_BRK13_SHIFT                  UART0_S2_BRK13_SHIFT
#define UARTLP_S2_RWUID_MASK                   UART0_S2_RWUID_MASK
#define UARTLP_S2_RWUID_SHIFT                  UART0_S2_RWUID_SHIFT
#define UARTLP_S2_RXINV_MASK                   UART0_S2_RXINV_MASK
#define UARTLP_S2_RXINV_SHIFT                  UART0_S2_RXINV_SHIFT
#define UARTLP_S2_MSBF_MASK                    UART0_S2_MSBF_MASK
#define UARTLP_S2_MSBF_SHIFT                   UART0_S2_MSBF_SHIFT
#define UARTLP_S2_RXEDGIF_MASK                 UART0_S2_RXEDGIF_MASK
#define UARTLP_S2_RXEDGIF_SHIFT                UART0_S2_RXEDGIF_SHIFT
#define UARTLP_S2_LBKDIF_MASK                  UART0_S2_LBKDIF_MASK
#define UARTLP_S2_LBKDIF_SHIFT                 UART0_S2_LBKDIF_SHIFT
#define UARTLP_C3_PEIE_MASK                    UART0_C3_PEIE_MASK
#define UARTLP_C3_PEIE_SHIFT                   UART0_C3_PEIE_SHIFT
#define UARTLP_C3_FEIE_MASK                    UART0_C3_FEIE_MASK
#define UARTLP_C3_FEIE_SHIFT                   UART0_C3_FEIE_SHIFT
#define UARTLP_C3_NEIE_MASK                    UART0_C3_NEIE_MASK
#define UARTLP_C3_NEIE_SHIFT                   UART0_C3_NEIE_SHIFT
#define UARTLP_C3_ORIE_MASK                    UART0_C3_ORIE_MASK
#define UARTLP_C3_ORIE_SHIFT                   UART0_C3_ORIE_SHIFT
#define UARTLP_C3_TXINV_MASK                   UART0_C3_TXINV_MASK
#define UARTLP_C3_TXINV_SHIFT                  UART0_C3_TXINV_SHIFT
#define UARTLP_C3_TXDIR_MASK                   UART0_C3_TXDIR_MASK
#define UARTLP_C3_TXDIR_SHIFT                  UART0_C3_TXDIR_SHIFT
#define UARTLP_C3_R9T8_MASK                    UART0_C3_R9T8_MASK
#define UARTLP_C3_R9T8_SHIFT                   UART0_C3_R9T8_SHIFT
#define UARTLP_C3_R8T9_MASK                    UART0_C3_R8T9_MASK
```

```
#define UARTLP_C3_R8T9_SHIFT                  UART0_C3_R8T9_SHIFT
#define UARTLP_D_R0T0_MASK                    UART0_D_R0T0_MASK
#define UARTLP_D_R0T0_SHIFT                   UART0_D_R0T0_SHIFT
#define UARTLP_D_R1T1_MASK                    UART0_D_R1T1_MASK
#define UARTLP_D_R1T1_SHIFT                   UART0_D_R1T1_SHIFT
#define UARTLP_D_R2T2_MASK                    UART0_D_R2T2_MASK
#define UARTLP_D_R2T2_SHIFT                   UART0_D_R2T2_SHIFT
#define UARTLP_D_R3T3_MASK                    UART0_D_R3T3_MASK
#define UARTLP_D_R3T3_SHIFT                   UART0_D_R3T3_SHIFT
#define UARTLP_D_R4T4_MASK                    UART0_D_R4T4_MASK
#define UARTLP_D_R4T4_SHIFT                   UART0_D_R4T4_SHIFT
#define UARTLP_D_R5T5_MASK                    UART0_D_R5T5_MASK
#define UARTLP_D_R5T5_SHIFT                   UART0_D_R5T5_SHIFT
#define UARTLP_D_R6T6_MASK                    UART0_D_R6T6_MASK
#define UARTLP_D_R6T6_SHIFT                   UART0_D_R6T6_SHIFT
#define UARTLP_D_R7T7_MASK                    UART0_D_R7T7_MASK
#define UARTLP_D_R7T7_SHIFT                   UART0_D_R7T7_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_RESYNCDIS_MASK              UART0_C5_RESYNCDIS_MASK
#define UARTLP_C5_RESYNCDIS_SHIFT             UART0_C5_RESYNCDIS_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 NV_FOPT_EZPORT_DIS_MASK               This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT              This_symbol_has_been_deprecated
#define FPTA_BASE                             FGPIOA_BASE
#define FPTA                                  FGPIOA
#define FPTB_BASE                             FGPIOB_BASE
#define FPTB                                  FGPIOB
#define FPTC_BASE                             FGPIOC_BASE
#define FPTC                                  FGPIOC
#define FPTD_BASE                             FGPIOD_BASE
#define FPTD                                  FGPIOD
#define FPTE_BASE                             FGPIOE_BASE
#define FPTE                                  FGPIOE
#define PTA_BASE                              GPIOA_BASE
```

```
#define PTA                            GPIOA
#define PTB_BASE                       GPIOB_BASE
#define PTB                            GPIOB
#define PTC_BASE                       GPIOC_BASE
#define PTC                            GPIOC
#define PTD_BASE                       GPIOD_BASE
#define PTD                            GPIOD
#define PTE_BASE                       GPIOE_BASE
#define PTE                            GPIOE
#define LPTimer_IRQn                   LPTMR0_IRQn
#define LPTimer_IRQHandler             LPTMR0_IRQHandler
#define LLW_IRQn                       LLWU_IRQn
#define LLW_IRQHandler                 LLWU_IRQHandler

/*!
 * @}
 */ /* end of group SDK_Compatibility_Symbols */


#endif  /* _MKL25Z4_H_ */

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     core_cm0plus.h
 * @brief    CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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 */
#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #pragma clang system_header   /* treat file as system include file */
#endif

#ifndef __CORE_CM0PLUS_H_GENERIC
#define __CORE_CM0PLUS_H_GENERIC

#include <stdint.h>

#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 CM0+ definitions */
#define __CM0PLUS_CMSIS_VERSION_MAIN (0x04U)                                   /*!< [31:16] CMSIS HAL ma
#define __CM0PLUS_CMSIS_VERSION_SUB  (0x1EU)                                   /*!< [15:0]  CMSIS HAL sub
#define __CM0PLUS_CMSIS_VERSION       ((__CM0PLUS_CMSIS_VERSION_MAIN << 16U) | \
                                       __CM0PLUS_CMSIS_VERSION_SUB           ) /*!< CMSIS HAL version number */

#define __CORTEX_M                (0x00U)                                      /*!< 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(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #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 i
  #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                                    /*!< inline keyword for COSMIC Compiler. Use -pc99 
  #define __STATIC_INLINE  static inline

#else
  #error Unknown compiler
#endif

/** __FPU_USED indicates whether an FPU is used or not.
    This core does not support an FPU at all
*/
#define __FPU_USED       0U

#if defined ( __CC_ARM )
  #if defined __TARGET_FPU_VFP
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
  #endif

#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #if defined __ARM_PCS_VFP
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
  #endif

#elif defined ( __GNUC__ )
  #if defined (__VFP_FP__) && !defined(__SOFTFP__)
```

```
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
  #endif

#elif defined ( __ICCARM__ )
  #if defined __ARMVFP__
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
  #endif

#elif defined ( __TMS470__ )
  #if defined __TI_VFP_SUPPORT__
    #error "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__ )
  #if ( __CSMC__ & 0x400U)
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
  #endif

#endif

#include "core_cmInstr.h"                /* Core Instruction Access */
#include "core_cmFunc.h"                 /* Core Function Access */

#ifdef __cplusplus
}
#endif

#endif /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

#ifdef __cplusplus
 extern "C" {
#endif

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
  #ifndef __CM0PLUS_REV
    #define __CM0PLUS_REV             0x0000U
    #warning "__CM0PLUS_REV not defined in device header file; using default!"
  #endif

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

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

  #ifndef __NVIC_PRIO_BITS
    #define __NVIC_PRIO_BITS          2U
    #warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
  #endif

  #ifndef __Vendor_SysTickConfig
    #define __Vendor_SysTickConfig    0U
    #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.
*/
#ifdef __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 */

/* following defines should be used for structure members */
#define     __IM     volatile const      /*! Defines 'read only' structure member permissions */
#define     __OM     volatile            /*! Defines 'write only' structure member permissions */
#define     __IOM    volatile            /*! Defines 'read / write' structure member 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                         31U                                            /*!< APSR: N Position */
#define APSR_N_Msk                         (1UL << APSR_N_Pos)                            /*!< APSR: N Mask */

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

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

#define APSR_V_Pos                         28U                                            /*!< 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                    0U                                    /*!< 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                      31U                                   /*!< xPSR: N Position */
#define xPSR_N_Msk                      (1UL << xPSR_N_Pos)                   /*!< xPSR: N Mask */

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

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

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

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

#define xPSR_ISR_Pos                    0U                                    /*!< 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                  1U                                            /*!< CONTROL: SPSEL Positio
#define CONTROL_SPSEL_Msk                 (1UL << CONTROL_SPSEL_Pos)                     /*!< CONTROL

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

/*@} 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
{
  __IOM uint32_t ISER[1U];               /*!< Offset: 0x000 (R/W)  Interrupt Set Enable Register */
        uint32_t RESERVED0[31U];
  __IOM uint32_t ICER[1U];               /*!< Offset: 0x080 (R/W)  Interrupt Clear Enable Register */
        uint32_t RSERVED1[31U];
  __IOM uint32_t ISPR[1U];               /*!< Offset: 0x100 (R/W)  Interrupt Set Pending Register */
        uint32_t RESERVED2[31U];
  __IOM uint32_t ICPR[1U];               /*!< Offset: 0x180 (R/W)  Interrupt Clear Pending Register */
        uint32_t RESERVED3[31U];
        uint32_t RESERVED4[64U];
  __IOM uint32_t IP[8U];                 /*!< 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
{
  __IM  uint32_t CPUID;                /*!< Offset: 0x000 (R/ )  CPUID Base Register */
  __IOM uint32_t ICSR;                 /*!< Offset: 0x004 (R/W)  Interrupt Control and State Register */
#if (__VTOR_PRESENT == 1U)
  __IOM uint32_t VTOR;                 /*!< Offset: 0x008 (R/W)  Vector Table Offset Register */
#else
        uint32_t RESERVED0;
#endif
  __IOM uint32_t AIRCR;                /*!< Offset: 0x00C (R/W)  Application Interrupt and Reset Control Regi
  __IOM uint32_t SCR;                  /*!< Offset: 0x010 (R/W)  System Control Register */
  __IOM uint32_t CCR;                  /*!< Offset: 0x014 (R/W)  Configuration Control Register */
        uint32_t RESERVED1;
  __IOM uint32_t SHP[2U];              /*!< Offset: 0x01C (R/W)  System Handlers Priority Registers. [0] is RE
  __IOM uint32_t SHCSR;                /*!< Offset: 0x024 (R/W)  System Handler Control and State Register
} SCB_Type;

/* SCB CPUID Register Definitions */
#define SCB_CPUID_IMPLEMENTER_Pos          24U                                            /*!< SCB CPUID: IMPLE
#define SCB_CPUID_IMPLEMENTER_Msk          (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)          /

#define SCB_CPUID_VARIANT_Pos              20U                                            /*!< SCB CPUID: VARIANT I
#define SCB_CPUID_VARIANT_Msk              (0xFUL << SCB_CPUID_VARIANT_Pos)               /*!< SCB

#define SCB_CPUID_ARCHITECTURE_Pos         16U                                            /*!< SCB CPUID: ARCH
#define SCB_CPUID_ARCHITECTURE_Msk         (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)

#define SCB_CPUID_PARTNO_Pos               4U                                             /*!< SCB CPUID: PARTNO P
#define SCB_CPUID_PARTNO_Msk               (0xFFFUL << SCB_CPUID_PARTNO_Pos)              /*!< SC

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

/* SCB Interrupt Control State Register Definitions */
#define SCB_ICSR_NMIPENDSET_Pos            31U                                            /*!< SCB ICSR: NMIPEND
#define SCB_ICSR_NMIPENDSET_Msk            (1UL << SCB_ICSR_NMIPENDSET_Pos)               /*!< SC

#define SCB_ICSR_PENDSVSET_Pos             28U                                            /*!< SCB ICSR: PENDSVS
#define SCB_ICSR_PENDSVSET_Msk             (1UL << SCB_ICSR_PENDSVSET_Pos)                /*!< SC

#define SCB_ICSR_PENDSVCLR_Pos             27U                                            /*!< SCB ICSR: PENDSVC
#define SCB_ICSR_PENDSVCLR_Msk             (1UL << SCB_ICSR_PENDSVCLR_Pos)                /*!< SC

#define SCB_ICSR_PENDSTSET_Pos             26U                                            /*!< SCB ICSR: PENDSTS
#define SCB_ICSR_PENDSTSET_Msk             (1UL << SCB_ICSR_PENDSTSET_Pos)                /*!< SC

#define SCB_ICSR_PENDSTCLR_Pos             25U                                            /*!< SCB ICSR: PENDSTC
#define SCB_ICSR_PENDSTCLR_Msk             (1UL << SCB_ICSR_PENDSTCLR_Pos)                /*!< SC

#define SCB_ICSR_ISRPREEMPT_Pos            23U                                            /*!< SCB ICSR: ISRPREEM
#define SCB_ICSR_ISRPREEMPT_Msk            (1UL << SCB_ICSR_ISRPREEMPT_Pos)               /*!< SC

#define SCB_ICSR_ISRPENDING_Pos            22U                                            /*!< SCB ICSR: ISRPENDI
```

```
#define SCB_ICSR_ISRPENDING_Msk           (1UL << SCB_ICSR_ISRPENDING_Pos)            /*!< SCB

#define SCB_ICSR_VECTPENDING_Pos          12U                                         /*!< SCB ICSR: VECTPEN
#define SCB_ICSR_VECTPENDING_Msk          (0x1FFUL << SCB_ICSR_VECTPENDING_Pos)       /*!

#define SCB_ICSR_VECTACTIVE_Pos           0U                                          /*!< SCB ICSR: VECTACTI
#define SCB_ICSR_VECTACTIVE_Msk           (0x1FFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/)    /*!<

#if (__VTOR_PRESENT == 1U)
/* SCB Interrupt Control State Register Definitions */
#define SCB_VTOR_TBLOFF_Pos               8U                                          /*!< SCB VTOR: TBLOFF Pos
#define SCB_VTOR_TBLOFF_Msk               (0xFFFFFFUL << SCB_VTOR_TBLOFF_Pos)         /*!< SC
#endif

/* SCB Application Interrupt and Reset Control Register Definitions */
#define SCB_AIRCR_VECTKEY_Pos             16U                                         /*!< SCB AIRCR: VECTKEY
#define SCB_AIRCR_VECTKEY_Msk             (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)         /*!< S

#define SCB_AIRCR_VECTKEYSTAT_Pos         16U                                         /*!< SCB AIRCR: VECTK
#define SCB_AIRCR_VECTKEYSTAT_Msk         (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)

#define SCB_AIRCR_ENDIANESS_Pos           15U                                         /*!< SCB AIRCR: ENDIAN
#define SCB_AIRCR_ENDIANESS_Msk           (1UL << SCB_AIRCR_ENDIANESS_Pos)            /*!< SC

#define SCB_AIRCR_SYSRESETREQ_Pos         2U                                          /*!< SCB AIRCR: SYSRE
#define SCB_AIRCR_SYSRESETREQ_Msk         (1UL << SCB_AIRCR_SYSRESETREQ_Pos)          /*

#define SCB_AIRCR_VECTCLRACTIVE_Pos       1U                                          /*!< SCB AIRCR: VECTC
#define SCB_AIRCR_VECTCLRACTIVE_Msk       (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)        /

/* SCB System Control Register Definitions */
#define SCB_SCR_SEVONPEND_Pos             4U                                          /*!< SCB SCR: SEVONPEN
#define SCB_SCR_SEVONPEND_Msk             (1UL << SCB_SCR_SEVONPEND_Pos)              /*!< SC

#define SCB_SCR_SLEEPDEEP_Pos             2U                                          /*!< SCB SCR: SLEEPDEE
#define SCB_SCR_SLEEPDEEP_Msk             (1UL << SCB_SCR_SLEEPDEEP_Pos)              /*!< SCB

#define SCB_SCR_SLEEPONEXIT_Pos           1U                                          /*!< SCB SCR: SLEEPONE
#define SCB_SCR_SLEEPONEXIT_Msk           (1UL << SCB_SCR_SLEEPONEXIT_Pos)            /*!< SC

/* SCB Configuration Control Register Definitions */
#define SCB_CCR_STKALIGN_Pos              9U                                          /*!< SCB CCR: STKALIGN Po
#define SCB_CCR_STKALIGN_Msk              (1UL << SCB_CCR_STKALIGN_Pos)               /*!< SCB C

#define SCB_CCR_UNALIGN_TRP_Pos           3U                                          /*!< SCB CCR: UNALIGN_
#define SCB_CCR_UNALIGN_TRP_Msk           (1UL << SCB_CCR_UNALIGN_TRP_Pos)            /*!< S

/* SCB System Handler Control and State Register Definitions */
#define SCB_SHCSR_SVCALLPENDED_Pos        15U                                         /*!< SCB SHCSR: SVC
#define SCB_SHCSR_SVCALLPENDED_Msk        (1UL << SCB_SHCSR_SVCALLPENDED_Pos)

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

/* SysTick Control / Status Register Definitions */
#define SysTick_CTRL_COUNTFLAG_Pos         16U                                            /*!< SysTick CTRL: COUN
#define SysTick_CTRL_COUNTFLAG_Msk         (1UL << SysTick_CTRL_COUNTFLAG_Pos)            /*!< S

#define SysTick_CTRL_CLKSOURCE_Pos          2U                                            /*!< SysTick CTRL: CLKS
#define SysTick_CTRL_CLKSOURCE_Msk         (1UL << SysTick_CTRL_CLKSOURCE_Pos)            /*!<

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

#define SysTick_CTRL_ENABLE_Pos             0U                                            /*!< SysTick CTRL: ENABLE
#define SysTick_CTRL_ENABLE_Msk            (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)           /*!< SysTic

/* SysTick Reload Register Definitions */
#define SysTick_LOAD_RELOAD_Pos             0U                                            /*!< SysTick LOAD: RELOAD
#define SysTick_LOAD_RELOAD_Msk            (0xFFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/)    /*!<

/* SysTick Current Register Definitions */
#define SysTick_VAL_CURRENT_Pos             0U                                            /*!< SysTick VAL: CURRENT
#define SysTick_VAL_CURRENT_Msk            (0xFFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/)    /*!<

/* SysTick Calibration Register Definitions */
#define SysTick_CALIB_NOREF_Pos            31U                                            /*!< SysTick CALIB: NOREF
#define SysTick_CALIB_NOREF_Msk            (1UL << SysTick_CALIB_NOREF_Pos)               /*!< SysTick

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

#define SysTick_CALIB_TENMS_Pos             0U                                            /*!< SysTick CALIB: TENMS P
#define SysTick_CALIB_TENMS_Msk            (0xFFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/)    /*!< Sy

/*@} end of group CMSIS_SysTick */

#if (__MPU_PRESENT == 1U)
```

```
/**
  \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
{
  __IM  uint32_t TYPE;                   /*!< Offset: 0x000 (R/ )  MPU Type Register */
  __IOM uint32_t CTRL;                   /*!< Offset: 0x004 (R/W)  MPU Control Register */
  __IOM uint32_t RNR;                    /*!< Offset: 0x008 (R/W)  MPU Region RNRber Register */
  __IOM uint32_t RBAR;                   /*!< Offset: 0x00C (R/W)  MPU Region Base Address Register */
  __IOM uint32_t RASR;                   /*!< Offset: 0x010 (R/W)  MPU Region Attribute and Size Register */
} MPU_Type;

/* MPU Type Register Definitions */
#define MPU_TYPE_IREGION_Pos              16U                                            /*!< MPU TYPE: IREGION Po
#define MPU_TYPE_IREGION_Msk              (0xFFUL << MPU_TYPE_IREGION_Pos)               /*!< MPU

#define MPU_TYPE_DREGION_Pos               8U                                            /*!< MPU TYPE: DREGION I
#define MPU_TYPE_DREGION_Msk              (0xFFUL << MPU_TYPE_DREGION_Pos)               /*!< MP

#define MPU_TYPE_SEPARATE_Pos              0U                                            /*!< MPU TYPE: SEPARAT
#define MPU_TYPE_SEPARATE_Msk             (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)             /*!< MP

/* MPU Control Register Definitions */
#define MPU_CTRL_PRIVDEFENA_Pos            2U                                            /*!< MPU CTRL: PRIVDEF
#define MPU_CTRL_PRIVDEFENA_Msk           (1UL << MPU_CTRL_PRIVDEFENA_Pos)               /*!< M

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

#define MPU_CTRL_ENABLE_Pos                0U                                            /*!< MPU CTRL: ENABLE Pos
#define MPU_CTRL_ENABLE_Msk               (1UL /*<< MPU_CTRL_ENABLE_Pos*/)               /*!< MPU C

/* MPU Region Number Register Definitions */
#define MPU_RNR_REGION_Pos                 0U                                            /*!< MPU RNR: REGION Posi
#define MPU_RNR_REGION_Msk                (0xFFUL /*<< MPU_RNR_REGION_Pos*/)             /*!< MPU

/* MPU Region Base Address Register Definitions */
#define MPU_RBAR_ADDR_Pos                  8U                                            /*!< MPU RBAR: ADDR Positio
#define MPU_RBAR_ADDR_Msk                 (0xFFFFFFUL << MPU_RBAR_ADDR_Pos)              /*!< MPU

#define MPU_RBAR_VALID_Pos                 4U                                            /*!< MPU RBAR: VALID Positio
#define MPU_RBAR_VALID_Msk                (1UL << MPU_RBAR_VALID_Pos)                    /*!< MPU RBA

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

```
/* MPU Region Attribute and Size Register Definitions */
#define MPU_RASR_ATTRS_Pos              16U                                    /*!< MPU RASR: MPU Regior
#define MPU_RASR_ATTRS_Msk              (0xFFFFUL << MPU_RASR_ATTRS_Pos)              /*!< MPU

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

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

#define MPU_RASR_TEX_Pos                19U                                    /*!< MPU RASR: ATTRS.TEX P
#define MPU_RASR_TEX_Msk                (0x7UL << MPU_RASR_TEX_Pos)                   /*!< MPU RASI

#define MPU_RASR_S_Pos                  18U                                    /*!< MPU RASR: ATTRS.S Positi
#define MPU_RASR_S_Msk                  (1UL << MPU_RASR_S_Pos)                       /*!< MPU RASR: A

#define MPU_RASR_C_Pos                  17U                                    /*!< MPU RASR: ATTRS.C Posit
#define MPU_RASR_C_Msk                  (1UL << MPU_RASR_C_Pos)                       /*!< MPU RASR: A

#define MPU_RASR_B_Pos                  16U                                    /*!< MPU RASR: ATTRS.B Positi
#define MPU_RASR_B_Msk                  (1UL << MPU_RASR_B_Pos)                       /*!< MPU RASR: A

#define MPU_RASR_SRD_Pos                 8U                                    /*!< MPU RASR: Sub-Region D
#define MPU_RASR_SRD_Msk                (0xFFUL << MPU_RASR_SRD_Pos)                  /*!< MPU RA

#define MPU_RASR_SIZE_Pos                1U                                    /*!< MPU RASR: Region Size Fi
#define MPU_RASR_SIZE_Msk               (0x1FUL << MPU_RASR_SIZE_Pos)                 /*!< MPU RAS

#define MPU_RASR_ENABLE_Pos              0U                                    /*!< MPU RASR: Region ena
#define MPU_RASR_ENABLE_Msk             (1UL /*<< MPU_RASR_ENABLE_Pos*/)              /*!< MPU I

/*@} 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
            Therefore they are not covered by the Cortex-M0+ header file.
  @{
 */
/*@} end of group CMSIS_CoreDebug */


/**
  \ingroup    CMSIS_core_register
  \defgroup   CMSIS_core_bitfield     Core register bit field macros
  \brief      Macros for use with bit field definitions (xxx_Pos, xxx_Msk).
  @{
 */

/**
```

```
  \brief   Mask and shift a bit field value for use in a register bit range.
  \param[in] field  Name of the register bit field.
  \param[in] value  Value of the bit field.
  \return           Masked and shifted value.
*/
#define _VAL2FLD(field, value)    ((value << field ## _Pos) & field ## _Msk)

/**
  \brief     Mask and shift a register value to extract a bit filed value.
  \param[in] field  Name of the register bit field.
  \param[in] value  Value of register.
  \return           Masked and shifted bit field value.
*/
#define _FLD2VAL(field, value)    ((value & field ## _Msk) >> field ## _Pos)

/*@} end of group CMSIS_core_bitfield */



/**
  \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 Addres

#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 == 1U)
  #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
  \details 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[0U] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}


/**
  \brief   Disable External Interrupt
  \details 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[0U] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}


/**
  \brief   Get Pending Interrupt
  \details 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[0U] & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0U
}
```

```
/**
  \brief   Set Pending Interrupt
  \details 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[0U] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}


/**
  \brief   Clear Pending Interrupt
  \details 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[0U] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}


/**
  \brief   Set Interrupt Priority
  \details 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
       (((priority << (8U - __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 << (8U - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) << _BIT_SHIFT(IRQn)));
  }
}


/**
  \brief   Get Interrupt Priority
  \details 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) >> (8U -
  }
  else
  {
    return((uint32_t)(((NVIC->IP[ _IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & (uint32_t)0xFFUL) >> (8U - __N
  }
}


/**
  \brief   System Reset
  \details 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 */

  for(;;)                                                        /* wait until reset */
  {
    __NOP();
  }
}

/*@} 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 == 0U)

/**
  \brief   System Tick Configuration
  \details 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 Interrup
  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 */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* ----------------------------------------------------------------------
* Copyright (C) 2010-2015 ARM Limited. All rights reserved.
*
* $Date:        20. October 2015
* $Revision:    V1.4.5 b
*
* Project:      CMSIS DSP Library
* Title:        arm_math.h
*
* Description:  Public header file for CMSIS DSP Library
*
* Target Processor: Cortex-M7/Cortex-M4/Cortex-M3/Cortex-M0
*
* 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
```

```
/**
   \mainpage CMSIS DSP Software Library
   *
   * Introduction
   * ------------
   *
   * This user manual describes the CMSIS DSP software library,
   * a suite of common signal processing functions for use on Cortex-M processor based devices.
   *
   * The library is divided into a number of functions each covering a specific category:
   * - Basic math functions
   * - Fast math functions
   * - Complex math functions
   * - Filters
   * - Matrix functions
   * - Transforms
   * - Motor control functions
   * - Statistical functions
   * - Support functions
   * - Interpolation functions
   *
   * The library has separate functions for operating on 8-bit integers, 16-bit integers,
   * 32-bit integer and 32-bit floating-point values.
   *
   * Using the Library
   * ------------
   *
   * The library installer contains prebuilt versions of the libraries in the <code>Lib</code> folder.
   * - arm_cortexM7lfdp_math.lib (Little endian and Double Precision Floating Point Unit on Cortex-M7)
   * - arm_cortexM7bfdp_math.lib (Big endian and Double Precision Floating Point Unit on Cortex-M7)
   * - arm_cortexM7lfsp_math.lib (Little endian and Single Precision Floating Point Unit on Cortex-M7)
   * - arm_cortexM7bfsp_math.lib (Big endian and Single Precision Floating Point Unit on Cortex-M7)
   * - arm_cortexM7l_math.lib (Little endian on Cortex-M7)
   * - arm_cortexM7b_math.lib (Big endian on Cortex-M7)
   * - arm_cortexM4lf_math.lib (Little endian and Floating Point Unit on Cortex-M4)
   * - arm_cortexM4bf_math.lib (Big endian and Floating Point Unit on Cortex-M4)
   * - arm_cortexM4l_math.lib (Little endian on Cortex-M4)
   * - arm_cortexM4b_math.lib (Big endian on Cortex-M4)
   * - arm_cortexM3l_math.lib (Little endian on Cortex-M3)
   * - arm_cortexM3b_math.lib (Big endian on Cortex-M3)
   * - arm_cortexM0l_math.lib (Little endian on Cortex-M0 / CortexM0+)
   * - arm_cortexM0b_math.lib (Big endian on Cortex-M0 / CortexM0+)
   *
   * The library functions are declared in the public file <code>arm_math.h</code> which is placed in the <c
   * Simply include this file and link the appropriate library in the application and begin calling the library fun
   * public header file <code> arm_math.h</code> for Cortex-M7/M4/M3/M0/M0+ with little endian and big e
   * Define the appropriate pre processor MACRO ARM_MATH_CM7 or ARM_MATH_CM4 or  ARM_MATH
   * ARM_MATH_CM0 or ARM_MATH_CM0PLUS depending on the target processor in the application.
   *
   * Examples
   * --------
   *
   * The library ships with a number of examples which demonstrate how to use the library functions.
   *
   * Toolchain Support
   * ------------
   *
   * The library has been developed and tested with MDK-ARM version 5.14.0.0
   * The library is being tested in GCC and IAR toolchains and updates on this activity will be made availabl
   *
   * Building the Library
   * ------------
   *
   * The library installer contains a project file to re build libraries on MDK-ARM Tool chain in the <code>CM
   * - arm_cortexM_math.uvprojx
   *
   *
   * The libraries can be built by opening the arm_cortexM_math.uvprojx project in MDK-ARM, selecting a s
   *
   * Pre-processor Macros
   * ------------
   *
   * Each library project have differant pre-processor macros.
   *
   * - UNALIGNED_SUPPORT_DISABLE:
   *
   * Define macro UNALIGNED_SUPPORT_DISABLE, If the silicon does not support unaligned memory ac
   *
   * - ARM_MATH_BIG_ENDIAN:
   *
   * Define macro ARM_MATH_BIG_ENDIAN to build the library for big endian targets. By default library bu
   *
   * - ARM_MATH_MATRIX_CHECK:
      *
      * Define macro ARM_MATH_MATRIX_CHECK for checking on the input and output sizes of matrices
      *
      * - ARM_MATH_ROUNDING:
      *
      * Define macro ARM_MATH_ROUNDING for rounding on support functions
      *
      * - ARM_MATH_CMx:
      *
      * Define macro ARM_MATH_CM4 for building the library on Cortex-M4 target, ARM_MATH_CM3 for buil
      * and ARM_MATH_CM0 for building library on Cortex-M0 target, ARM_MATH_CM0PLUS for building lib
      * ARM_MATH_CM7 for building the library on cortex-M7.
      *
      * - __FPU_PRESENT:
      *
      * Initialize macro __FPU_PRESENT = 1 when building on FPU supported Targets. Enable this macro for
      *
      * <hr>
      * CMSIS-DSP in ARM::CMSIS Pack
      * -----------------------------
      *
      * The following files relevant to CMSIS-DSP are present in the <b>ARM::CMSIS</b> Pack directories:
      * |File/Folder                 |Content                                                  |
      * |------------------------------|------------------------------------------------------------------------|
      * |\b CMSIS\\Documentation\\DSP  | This documentation                                     |
      * |\b CMSIS\\DSP_Lib            | Software license agreement (license.txt)               |
      * |\b CMSIS\\DSP_Lib\\Examples   | Example projects demonstrating the usage of the library functions
      * |\b CMSIS\\DSP_Lib\\Source     | Source files for rebuilding the library                |
      *
      * <hr>
      * Revision History of CMSIS-DSP
      * ------------
      * Please refer to \ref ChangeLog_pg.
      *
      * Copyright Notice
      * ------------
      *
      * Copyright (C) 2010-2015 ARM Limited. All rights reserved.
      */


   /**
    * @defgroup groupMath Basic Math Functions
    */

   /**
    * @defgroup groupFastMath Fast Math Functions
    * This set of functions provides a fast approximation to sine, cosine, and square root.
    * As compared to most of the other functions in the CMSIS math library, the fast math functions
    * operate on individual values and not arrays.
    * There are separate functions for Q15, Q31, and floating-point data.
    *
    */
```

```
/**
 * @defgroup groupCmplxMath Complex Math Functions
 * This set of functions operates on complex data vectors.
 * The data in the complex arrays is stored in an interleaved fashion
 * (real, imag, real, imag, ...).
 * In the API functions, the number of samples in a complex array refers
 * to the number of complex values; the array contains twice this number of
 * real values.
 */

/**
 * @defgroup groupFilters Filtering Functions
 */

/**
 * @defgroup groupMatrix Matrix Functions
 *
 * This set of functions provides basic matrix math operations.
 * The functions operate on matrix data structures.  For example,
 * the type
 * definition for the floating-point matrix structure is shown
 * below:
 * <pre>
 *     typedef struct
 *     {
 *       uint16_t numRows;     // number of rows of the matrix.
 *       uint16_t numCols;     // number of columns of the matrix.
 *       float32_t *pData;     // points to the data of the matrix.
 *     } arm_matrix_instance_f32;
 * </pre>
 * There are similar definitions for Q15 and Q31 data types.
 *
 * The structure specifies the size of the matrix and then points to
 * an array of data.  The array is of size <code>numRows X numCols</code>
 * and the values are arranged in row order.  That is, the
 * matrix element (i, j) is stored at:
 * <pre>
 *     pData[i*numCols + j]
 * </pre>
 *
 * \par Init Functions
 * There is an associated initialization function for each type of matrix
 * data structure.
 * The initialization function sets the values of the internal structure fields.
 * Refer to the function <code>arm_mat_init_f32()</code>, <code>arm_mat_init_q31()</code>
 * and <code>arm_mat_init_q15()</code> for floating-point, Q31 and Q15 types,  respectively.
 *
 * \par
 * Use of the initialization function is optional. However, if initialization function is used
 * then the instance structure cannot be placed into a const data section.
 * To place the instance structure in a const data
 * section, manually initialize the data structure.  For example:
```

```
 * <pre>
 * <code>arm_matrix_instance_f32 S = {nRows, nColumns, pData};</code>
 * <code>arm_matrix_instance_q31 S = {nRows, nColumns, pData};</code>
 * <code>arm_matrix_instance_q15 S = {nRows, nColumns, pData};</code>
 * </pre>
 * where <code>nRows</code> specifies the number of rows, <code>nColumns</code>
 * specifies the number of columns, and <code>pData</code> points to the
 * data array.
 *
 * \par Size Checking
 * By default all of the matrix functions perform size checking on the input and
 * output matrices.  For example, the matrix addition function verifies that the
 * two input matrices and the output matrix all have the same number of rows and
 * columns.  If the size check fails the functions return:
 * <pre>
 *     ARM_MATH_SIZE_MISMATCH
 * </pre>
 * Otherwise the functions return
 * <pre>
 *     ARM_MATH_SUCCESS
 * </pre>
 * There is some overhead associated with this matrix size checking.
 * The matrix size checking is enabled via the \#define
 * <pre>
 *     ARM_MATH_MATRIX_CHECK
 * </pre>
 * within the library project settings.  By default this macro is defined
 * and size checking is enabled.  By changing the project settings and
 * undefining this macro size checking is eliminated and the functions
 * run a bit faster.  With size checking disabled the functions always
 * return <code>ARM_MATH_SUCCESS</code>.
 */

/**
 * @defgroup groupTransforms Transform Functions
 */

/**
 * @defgroup groupController Controller Functions
 */

/**
 * @defgroup groupStats Statistics Functions
 */
/**
 * @defgroup groupSupport Support Functions
 */

/**
 * @defgroup groupInterpolation Interpolation Functions
 * These functions perform 1- and 2-dimensional interpolation of data.
 * Linear interpolation is used for 1-dimensional data and
 * bilinear interpolation is used for 2-dimensional data.
 */

/**
 * @defgroup groupExamples Examples
 */
#ifndef _ARM_MATH_H
#define _ARM_MATH_H

/* ignore some GCC warnings */
#if defined ( __GNUC__ )
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wsign-conversion"
#pragma GCC diagnostic ignored "-Wconversion"
#pragma GCC diagnostic ignored "-Wunused-parameter"
#endif

#define __CMSIS_GENERIC         /* disable NVIC and Systick functions */

#if defined(ARM_MATH_CM7)
  #include "core_cm7.h"
#elif defined (ARM_MATH_CM4)
  #include "core_cm4.h"
#elif defined (ARM_MATH_CM3)
  #include "core_cm3.h"
#elif defined (ARM_MATH_CM0)
  #include "core_cm0.h"
  #define ARM_MATH_CM0_FAMILY
#elif defined (ARM_MATH_CM0PLUS)
  #include "core_cm0plus.h"
  #define ARM_MATH_CM0_FAMILY
#else
  #error "Define according the used Cortex core ARM_MATH_CM7, ARM_MATH_CM4, ARM_MATH_CM3
#endif

#undef  __CMSIS_GENERIC         /* enable NVIC and Systick functions */
#include "string.h"
#include "math.h"
#ifdef   __cplusplus
extern "C"
{
#endif


  /**
   * @brief Macros required for reciprocal calculation in Normalized LMS
   */

#define DELTA_Q31          (0x100)
#define DELTA_Q15          0x5
#define INDEX_MASK         0x0000003F
#ifndef PI
#define PI                 3.14159265358979f
#endif
```

```
  /**
   * @brief Macros required for SINE and COSINE Fast math approximations
   */

#define FAST_MATH_TABLE_SIZE  512
#define FAST_MATH_Q31_SHIFT   (32 - 10)
#define FAST_MATH_Q15_SHIFT   (16 - 10)
#define CONTROLLER_Q31_SHIFT  (32 - 9)
#define TABLE_SIZE  256
#define TABLE_SPACING_Q31     0x400000
#define TABLE_SPACING_Q15     0x80

  /**
   * @brief Macros required for SINE and COSINE Controller functions
   */
  /* 1.31(q31) Fixed value of 2/360 */
  /* -1 to +1 is divided into 360 values so total spacing is (2/360) */
#define INPUT_SPACING         0xB60B61

  /**
   * @brief Macro for Unaligned Support
   */
#ifndef UNALIGNED_SUPPORT_DISABLE
    #define ALIGN4
#else
  #if defined  (__GNUC__)
    #define ALIGN4 __attribute__((aligned(4)))
  #else
    #define ALIGN4 __align(4)
  #endif
#endif   /* #ifndef UNALIGNED_SUPPORT_DISABLE */

  /**
   * @brief Error status returned by some functions in the library.
   */

  typedef enum
  {
    ARM_MATH_SUCCESS = 0,                /**< No error */
    ARM_MATH_ARGUMENT_ERROR = -1,        /**< One or more arguments are incorrect */
    ARM_MATH_LENGTH_ERROR = -2,          /**< Length of data buffer is incorrect */
    ARM_MATH_SIZE_MISMATCH = -3,         /**< Size of matrices is not compatible with the operation. */
    ARM_MATH_NANINF = -4,                /**< Not-a-number (NaN) or infinity is generated */
    ARM_MATH_SINGULAR = -5,              /**< Generated by matrix inversion if the input matrix is singular a
    ARM_MATH_TEST_FAILURE = -6           /**< Test Failed  */
  } arm_status;

  /**
   * @brief 8-bit fractional data type in 1.7 format.
   */
  typedef int8_t q7_t;
  /**
   * @brief 16-bit fractional data type in 1.15 format.
   */
  typedef int16_t q15_t;

  /**
   * @brief 32-bit fractional data type in 1.31 format.
   */
  typedef int32_t q31_t;

  /**
   * @brief 64-bit fractional data type in 1.63 format.
   */
  typedef int64_t q63_t;

  /**
   * @brief 32-bit floating-point type definition.
   */
  typedef float float32_t;

  /**
   * @brief 64-bit floating-point type definition.
   */
  typedef double float64_t;

  /**
   * @brief definition to read/write two 16 bit values.
   */
#if defined __CC_ARM
  #define __SIMD32_TYPE int32_t __packed
  #define CMSIS_UNUSED __attribute__((unused))

#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #define __SIMD32_TYPE int32_t
  #define CMSIS_UNUSED __attribute__((unused))

#elif defined __GNUC__
  #define __SIMD32_TYPE int32_t
  #define CMSIS_UNUSED __attribute__((unused))

#elif defined __ICCARM__
  #define __SIMD32_TYPE int32_t __packed
  #define CMSIS_UNUSED

#elif defined __CSMC__
  #define __SIMD32_TYPE int32_t
  #define CMSIS_UNUSED

#elif defined __TASKING__
  #define __SIMD32_TYPE __unaligned int32_t
  #define CMSIS_UNUSED

#else
```

```
  #error Unknown compiler
#endif

#define __SIMD32(addr)        (*(__SIMD32_TYPE **) & (addr))
#define __SIMD32_CONST(addr)  ((__SIMD32_TYPE *)(addr))
#define _SIMD32_OFFSET(addr)  (*(__SIMD32_TYPE *)  (addr))
#define __SIMD64(addr)        (*(int64_t **) & (addr))

#if defined (ARM_MATH_CM3) || defined (ARM_MATH_CM0_FAMILY)
  /**
   * @brief definition to pack two 16 bit values.
   */
#define __PKHBT(ARG1, ARG2, ARG3)      ( (((int32_t)(ARG1) <<  0) & (int32_t)0x0000FFFF) | \
                                         (((int32_t)(ARG2) << ARG3) & (int32_t)0xFFFF0000)  )
#define __PKHTB(ARG1, ARG2, ARG3)      ( (((int32_t)(ARG1) <<  0) & (int32_t)0xFFFF0000) | \
                                         (((int32_t)(ARG2) >> ARG3) & (int32_t)0x0000FFFF)  )

#endif


   /**
   * @brief definition to pack four 8 bit values.
   */
#ifndef ARM_MATH_BIG_ENDIAN

#define __PACKq7(v0,v1,v2,v3) ( (((int32_t)(v0) <<  0) & (int32_t)0x000000FF) | \
                                (((int32_t)(v1) <<  8) & (int32_t)0x0000FF00) | \
                                (((int32_t)(v2) << 16) & (int32_t)0x00FF0000) | \
                                (((int32_t)(v3) << 24) & (int32_t)0xFF000000)  )
#else

#define __PACKq7(v0,v1,v2,v3) ( (((int32_t)(v3) <<  0) & (int32_t)0x000000FF) | \
                                (((int32_t)(v2) <<  8) & (int32_t)0x0000FF00) | \
                                (((int32_t)(v1) << 16) & (int32_t)0x00FF0000) | \
                                (((int32_t)(v0) << 24) & (int32_t)0xFF000000)  )

#endif



  /**
   * @brief Clips Q63 to Q31 values.
   */
  static __INLINE q31_t clip_q63_to_q31(
  q63_t x)
  {
    return ((q31_t) (x >> 32) != ((q31_t) x >> 31)) ?
      ((0x7FFFFFFF ^ ((q31_t) (x >> 63)))) : (q31_t) x;
  }

  /**
   * @brief Clips Q63 to Q15 values.
   */
  static __INLINE q15_t clip_q63_to_q15(
  q63_t x)
  {
    return ((q31_t) (x >> 32) != ((q31_t) x >> 31)) ?
      ((0x7FFF ^ ((q15_t) (x >> 63)))) : (q15_t) (x >> 15);
  }

  /**
   * @brief Clips Q31 to Q7 values.
   */
  static __INLINE q7_t clip_q31_to_q7(
  q31_t x)
  {
    return ((q31_t) (x >> 24) != ((q31_t) x >> 23)) ?
      ((0x7F ^ ((q7_t) (x >> 31)))) : (q7_t) x;
  }

  /**
   * @brief Clips Q31 to Q15 values.
   */
  static __INLINE q15_t clip_q31_to_q15(
  q31_t x)
  {
    return ((q31_t) (x >> 16) != ((q31_t) x >> 15)) ?
      ((0x7FFF ^ ((q15_t) (x >> 31)))) : (q15_t) x;
  }

  /**
   * @brief Multiplies 32 X 64 and returns 32 bit result in 2.30 format.
   */

  static __INLINE q63_t mult32x64(
  q63_t x,
  q31_t y)
  {
    return ((((q63_t) (x & 0x00000000FFFFFFFF) * y) >> 32) +
            (((q63_t) (x >> 32) * y)));
  }

/*
  #if defined (ARM_MATH_CM0_FAMILY) && defined ( __CC_ARM   )
  #define __CLZ __clz
  #endif
 */
/* note: function can be removed when all toolchain support __CLZ for Cortex-M0 */
#if defined (ARM_MATH_CM0_FAMILY) && ((defined (__ICCARM__))  )
  static __INLINE uint32_t __CLZ(
  q31_t data);

  static __INLINE uint32_t __CLZ(
  q31_t data)
  {
    uint32_t count = 0;
    uint32_t mask = 0x80000000;
    while((data & mask) == 0)
    {
      count += 1u;
      mask = mask >> 1u;
    }

    return (count);
  }
#endif

  /**
   * @brief Function to Calculates 1/in (reciprocal) value of Q31 Data type.
   */

  static __INLINE uint32_t arm_recip_q31(
  q31_t in,
  q31_t * dst,
  q31_t * pRecipTable)
  {
    q31_t out;
    uint32_t tempVal;
    uint32_t index, i;
    uint32_t signBits;

    if(in > 0)
    {
      signBits = ((uint32_t) (__CLZ( in) - 1));
    }
    else
    {
      signBits = ((uint32_t) (__CLZ(-in) - 1));
    }

    /* Convert input sample to 1.31 format */
    in = (in << signBits);

    /* calculation of index for initial approximated Val */
    index = (uint32_t)(in >> 24);
    index = (index & INDEX_MASK);

    /* 1.31 with exp 1 */
    out = pRecipTable[index];

    /* calculation of reciprocal value */
    /* running approximation for two iterations */
    for (i = 0u; i < 2u; i++)
    {
      tempVal = (uint32_t) (((q63_t) in * out) >> 31);
      tempVal = 0x7FFFFFFFu - tempVal;
      /*      1.31 with exp 1 */
      /* out = (q31_t) (((q63_t) out * tempVal) >> 30); */
      out = clip_q63_to_q31(((q63_t) out * tempVal) >> 30);
    }

    /* write output */
    *dst = out;

    /* return num of signbits of out = 1/in value */
    return (signBits + 1u);
  }


  /**
   * @brief Function to Calculates 1/in (reciprocal) value of Q15 Data type.
   */
  static __INLINE uint32_t arm_recip_q15(
  q15_t in,
  q15_t * dst,
  q15_t * pRecipTable)
  {
    q15_t out = 0;
    uint32_t tempVal = 0;
    uint32_t index = 0, i = 0;
    uint32_t signBits = 0;

    if(in > 0)
    {
      signBits = ((uint32_t)(__CLZ( in) - 17));
    }
    else
    {
      signBits = ((uint32_t)(__CLZ(-in) - 17));
    }

    /* Convert input sample to 1.15 format */
    in = (in << signBits);

    /* calculation of index for initial approximated Val */
    index = (uint32_t)(in >>  8);
    index = (index & INDEX_MASK);

    /*      1.15 with exp 1  */
    out = pRecipTable[index];

    /* calculation of reciprocal value */
    /* running approximation for two iterations */
    for (i = 0u; i < 2u; i++)
    {
      tempVal = (uint32_t) (((q31_t) in * out) >> 15);
      tempVal = 0x7FFFu - tempVal;
      /*      1.15 with exp 1 */
      out = (q15_t) (((q31_t) out * tempVal) >> 14);
      /* out = clip_q31_to_q15(((q31_t) out * tempVal) >> 14); */
    }
```

```
    /* write output */
    *dst = out;

    /* return num of signbits of out = 1/in value */
    return (signBits + 1);
  }



  /*
   * @brief C custom defined intrinisic function for only M0 processors
   */
#if defined(ARM_MATH_CM0_FAMILY)
  static __INLINE q31_t __SSAT(
  q31_t x,
  uint32_t y)
  {
    int32_t posMax, negMin;
    uint32_t i;

    posMax = 1;
    for (i = 0; i < (y - 1); i++)
    {
      posMax = posMax * 2;
    }

    if(x > 0)
    {
      posMax = (posMax - 1);

      if(x > posMax)
      {
        x = posMax;
      }
    }
    else
    {
      negMin = -posMax;

      if(x < negMin)
      {
        x = negMin;
      }
    }
    return (x);
  }
#endif /* end of ARM_MATH_CM0_FAMILY */



  /*
   * @brief C custom defined intrinsic function for M3 and M0 processors
   */
#if defined (ARM_MATH_CM3) || defined (ARM_MATH_CM0_FAMILY)
```

```
/*
 * @brief C custom defined QADD8 for M3 and M0 processors
 */
static __INLINE uint32_t __QADD8(
uint32_t x,
uint32_t y)
{
  q31_t r, s, t, u;

  r = __SSAT(((((q31_t)x << 24) >> 24) + (((q31_t)y << 24) >> 24)), 8) & (int32_t)0x000000FF;
  s = __SSAT(((((q31_t)x << 16) >> 24) + (((q31_t)y << 16) >> 24)), 8) & (int32_t)0x000000FF;
  t = __SSAT(((((q31_t)x <<  8) >> 24) + (((q31_t)y <<  8) >> 24)), 8) & (int32_t)0x000000FF;
  u = __SSAT(((((q31_t)x      ) >> 24) + (((q31_t)y      ) >> 24)), 8) & (int32_t)0x000000FF;

  return ((uint32_t)((u << 24) | (t << 16) | (s <<  8) | (r      )));
}


/*
 * @brief C custom defined QSUB8 for M3 and M0 processors
 */
static __INLINE uint32_t __QSUB8(
uint32_t x,
uint32_t y)
{
  q31_t r, s, t, u;

  r = __SSAT(((((q31_t)x << 24) >> 24) - (((q31_t)y << 24) >> 24)), 8) & (int32_t)0x000000FF;
  s = __SSAT(((((q31_t)x << 16) >> 24) - (((q31_t)y << 16) >> 24)), 8) & (int32_t)0x000000FF;
  t = __SSAT(((((q31_t)x <<  8) >> 24) - (((q31_t)y <<  8) >> 24)), 8) & (int32_t)0x000000FF;
  u = __SSAT(((((q31_t)x      ) >> 24) - (((q31_t)y      ) >> 24)), 8) & (int32_t)0x000000FF;

  return ((uint32_t)((u << 24) | (t << 16) | (s <<  8) | (r      )));
}


/*
 * @brief C custom defined QADD16 for M3 and M0 processors
 */
static __INLINE uint32_t __QADD16(
uint32_t x,
uint32_t y)
{
/*  q31_t r,     s;  without initialisation 'arm_offset_q15 test' fails  but 'intrinsic' tests pass! for armCC */
  q31_t r = 0, s = 0;

  r = __SSAT(((((q31_t)x << 16) >> 16) + (((q31_t)y << 16) >> 16)), 16) & (int32_t)0x0000FFFF;
  s = __SSAT(((((q31_t)x      ) >> 16) + (((q31_t)y      ) >> 16)), 16) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r      )));
}
```

```
/*
 * @brief C custom defined SHADD16 for M3 and M0 processors
 */
static __INLINE uint32_t __SHADD16(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = (((((q31_t)x << 16) >> 16) + (((q31_t)y << 16) >> 16)) >> 1) & (int32_t)0x0000FFFF;
  s = (((((q31_t)x       ) >> 16) + (((q31_t)y       ) >> 16)) >> 1) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r       )));
}


/*
 * @brief C custom defined QSUB16 for M3 and M0 processors
 */
static __INLINE uint32_t __QSUB16(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = __SSAT(((((q31_t)x << 16) >> 16) - (((q31_t)y << 16) >> 16)), 16) & (int32_t)0x0000FFFF;
  s = __SSAT(((((q31_t)x       ) >> 16) - (((q31_t)y       ) >> 16)), 16) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r       )));
}


/*
 * @brief C custom defined SHSUB16 for M3 and M0 processors
 */
static __INLINE uint32_t __SHSUB16(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = (((((q31_t)x << 16) >> 16) - (((q31_t)y << 16) >> 16)) >> 1) & (int32_t)0x0000FFFF;
  s = (((((q31_t)x       ) >> 16) - (((q31_t)y       ) >> 16)) >> 1) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r       )));
}


/*
 * @brief C custom defined QASX for M3 and M0 processors
 */
static __INLINE uint32_t __QASX(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = __SSAT(((((q31_t)x << 16) >> 16) - (((q31_t)y      ) >> 16)), 16) & (int32_t)0x0000FFFF;
  s = __SSAT(((((q31_t)x      ) >> 16) + (((q31_t)y << 16) >> 16)), 16) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r      )));
}


/*
 * @brief C custom defined SHASX for M3 and M0 processors
 */
static __INLINE uint32_t __SHASX(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = (((((q31_t)x << 16) >> 16) - (((q31_t)y      ) >> 16)) >> 1) & (int32_t)0x0000FFFF;
  s = (((((q31_t)x      ) >> 16) + (((q31_t)y << 16) >> 16)) >> 1) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r      )));
}


/*
 * @brief C custom defined QSAX for M3 and M0 processors
 */
static __INLINE uint32_t __QSAX(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = __SSAT(((((q31_t)x << 16) >> 16) + (((q31_t)y      ) >> 16)), 16) & (int32_t)0x0000FFFF;
  s = __SSAT(((((q31_t)x      ) >> 16) - (((q31_t)y << 16) >> 16)), 16) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r      )));
}


/*
 * @brief C custom defined SHSAX for M3 and M0 processors
 */
static __INLINE uint32_t __SHSAX(
uint32_t x,
uint32_t y)
{
  q31_t r, s;

  r = (((((q31_t)x << 16) >> 16) + (((q31_t)y      ) >> 16)) >> 1) & (int32_t)0x0000FFFF;
  s = (((((q31_t)x      ) >> 16) - (((q31_t)y << 16) >> 16)) >> 1) & (int32_t)0x0000FFFF;

  return ((uint32_t)((s << 16) | (r      )));
}


/*
 * @brief C custom defined SMUSDX for M3 and M0 processors
 */
static __INLINE uint32_t __SMUSDX(
uint32_t x,
uint32_t y)
{
  return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y      ) >> 16)) -
                     ((((q31_t)x      ) >> 16) * (((q31_t)y << 16) >> 16))   ));
}

/*
 * @brief C custom defined SMUADX for M3 and M0 processors
 */
static __INLINE uint32_t __SMUADX(
uint32_t x,
uint32_t y)
{
  return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y      ) >> 16)) +
                     ((((q31_t)x      ) >> 16) * (((q31_t)y << 16) >> 16))   ));
}


/*
 * @brief C custom defined QADD for M3 and M0 processors
 */
static __INLINE int32_t __QADD(
int32_t x,
int32_t y)
{
  return ((int32_t)(clip_q63_to_q31((q63_t)x + (q31_t)y)));
}


/*
 * @brief C custom defined QSUB for M3 and M0 processors
 */
static __INLINE int32_t __QSUB(
int32_t x,
int32_t y)
{
  return ((int32_t)(clip_q63_to_q31((q63_t)x - (q31_t)y)));
}


/*
 * @brief C custom defined SMLAD for M3 and M0 processors
```

```
   */
  static __INLINE uint32_t __SMLAD(
  uint32_t x,
  uint32_t y,
  uint32_t sum)
  {
    return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ((((q31_t)x      ) >> 16) * (((q31_t)y      ) >> 16)) +
                       ( ((q31_t)sum    )                                  )   ));
  }


  /*
   * @brief C custom defined SMLADX for M3 and M0 processors
   */
  static __INLINE uint32_t __SMLADX(
  uint32_t x,
  uint32_t y,
  uint32_t sum)
  {
    return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y      ) >> 16)) +
                       ((((q31_t)x      ) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ( ((q31_t)sum    )                                  )   ));
  }


  /*
   * @brief C custom defined SMLSDX for M3 and M0 processors
   */
  static __INLINE uint32_t __SMLSDX(
  uint32_t x,
  uint32_t y,
  uint32_t sum)
  {
    return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y      ) >> 16)) -
                       ((((q31_t)x      ) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ( ((q31_t)sum    )                                  )   ));
  }


  /*
   * @brief C custom defined SMLALD for M3 and M0 processors
   */
  static __INLINE uint64_t __SMLALD(
  uint32_t x,
  uint32_t y,
  uint64_t sum)
  {
/*  return (sum + ((q15_t) (x >> 16) * (q15_t) (y >> 16)) + ((q15_t) x * (q15_t) y)); */
    return ((uint64_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ((((q31_t)x      ) >> 16) * (((q31_t)y      ) >> 16)) +
                       ( ((q63_t)sum    )                                  )   ));
  }
  /*
   * @brief C custom defined SMLALDX for M3 and M0 processors
   */
  static __INLINE uint64_t __SMLALDX(
  uint32_t x,
  uint32_t y,
  uint64_t sum)
  {
/*  return (sum + ((q15_t) (x >> 16) * (q15_t) y)) + ((q15_t) x * (q15_t) (y >> 16)); */
    return ((uint64_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y       ) >> 16)) +
                       ((((q31_t)x       ) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ( ((q63_t)sum    )                                 )   ));
  }


  /*
   * @brief C custom defined SMUAD for M3 and M0 processors
   */
  static __INLINE uint32_t __SMUAD(
  uint32_t x,
  uint32_t y)
  {
    return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y << 16) >> 16)) +
                       ((((q31_t)x       ) >> 16) * (((q31_t)y       ) >> 16))   ));
  }


  /*
   * @brief C custom defined SMUSD for M3 and M0 processors
   */
  static __INLINE uint32_t __SMUSD(
  uint32_t x,
  uint32_t y)
  {
    return ((uint32_t)(((((q31_t)x << 16) >> 16) * (((q31_t)y << 16) >> 16)) -
                       ((((q31_t)x       ) >> 16) * (((q31_t)y       ) >> 16))   ));
  }


  /*
   * @brief C custom defined SXTB16 for M3 and M0 processors
   */
  static __INLINE uint32_t __SXTB16(
  uint32_t x)
  {
    return ((uint32_t)(((((q31_t)x << 24) >> 24) & (q31_t)0x0000FFFF) |
                       ((((q31_t)x <<  8) >>  8) & (q31_t)0xFFFF0000)  ));
  }

#endif /* defined (ARM_MATH_CM3) || defined (ARM_MATH_CM0_FAMILY) */
```

```
/**
 * @brief Instance structure for the Q7 FIR filter.
 */
typedef struct
{
  uint16_t numTaps;        /**< number of filter coefficients in the filter. */
  q7_t *pState;            /**< points to the state variable array. The array is of length numTaps+blockSize-1.
  q7_t *pCoeffs;           /**< points to the coefficient array. The array is of length numTaps.*/
} arm_fir_instance_q7;

/**
 * @brief Instance structure for the Q15 FIR filter.
 */
typedef struct
{
  uint16_t numTaps;         /**< number of filter coefficients in the filter. */
  q15_t *pState;            /**< points to the state variable array. The array is of length numTaps+blockSize-1
  q15_t *pCoeffs;           /**< points to the coefficient array. The array is of length numTaps.*/
} arm_fir_instance_q15;

/**
 * @brief Instance structure for the Q31 FIR filter.
 */
typedef struct
{
  uint16_t numTaps;         /**< number of filter coefficients in the filter. */
  q31_t *pState;            /**< points to the state variable array. The array is of length numTaps+blockSize-1
  q31_t *pCoeffs;           /**< points to the coefficient array. The array is of length numTaps. */
} arm_fir_instance_q31;

/**
 * @brief Instance structure for the floating-point FIR filter.
 */
typedef struct
{
  uint16_t numTaps;     /**< number of filter coefficients in the filter. */
  float32_t *pState;    /**< points to the state variable array. The array is of length numTaps+blockSize-1. *
  float32_t *pCoeffs;   /**< points to the coefficient array. The array is of length numTaps. */
} arm_fir_instance_f32;


/**
 * @brief Processing function for the Q7 FIR filter.
 * @param[in]  S          points to an instance of the Q7 FIR filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_q7(
const arm_fir_instance_q7 * S,
q7_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q7 FIR filter.
 * @param[in,out] S          points to an instance of the Q7 FIR structure.
 * @param[in]     numTaps    Number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of samples that are processed.
 */
void arm_fir_init_q7(
arm_fir_instance_q7 * S,
uint16_t numTaps,
q7_t * pCoeffs,
q7_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q15 FIR filter.
 * @param[in]  S          points to an instance of the Q15 FIR structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_q15(
const arm_fir_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the fast Q15 FIR filter for Cortex-M3 and Cortex-M4.
 * @param[in]  S          points to an instance of the Q15 FIR filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_fast_q15(
const arm_fir_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q15 FIR filter.
 * @param[in,out] S          points to an instance of the Q15 FIR filter structure.
 * @param[in]     numTaps    Number of filter coefficients in the filter. Must be even and greater than or eq
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of samples that are processed at a time.
 * @return The function returns ARM_MATH_SUCCESS if initialization was successful or ARM_MATH_A
 * <code>numTaps</code> is not a supported value.
 */
arm_status arm_fir_init_q15(
arm_fir_instance_q15 * S,
uint16_t numTaps,
q15_t * pCoeffs,
q15_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 FIR filter.
 * @param[in]  S          points to an instance of the Q31 FIR filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_q31(
const arm_fir_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the fast Q31 FIR filter for Cortex-M3 and Cortex-M4.
 * @param[in]  S          points to an instance of the Q31 FIR structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_fast_q31(
const arm_fir_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q31 FIR filter.
 * @param[in,out] S          points to an instance of the Q31 FIR structure.
 * @param[in]     numTaps    Number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of samples that are processed at a time.
 */
void arm_fir_init_q31(
arm_fir_instance_q31 * S,
uint16_t numTaps,
q31_t * pCoeffs,
q31_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the floating-point FIR filter.
 * @param[in]  S          points to an instance of the floating-point FIR structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_f32(
const arm_fir_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point FIR filter.
 * @param[in,out] S          points to an instance of the floating-point FIR filter structure.
 * @param[in]     numTaps    Number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of samples that are processed at a time.
 */
void arm_fir_init_f32(
arm_fir_instance_f32 * S,
uint16_t numTaps,
float32_t * pCoeffs,
float32_t * pState,
uint32_t blockSize);


/**
 * @brief Instance structure for the Q15 Biquad cascade filter.
 */
typedef struct
{
  int8_t numStages;        /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  q15_t *pState;          /**< Points to the array of state coefficients.  The array is of length 4*numStages. */
  q15_t *pCoeffs;          /**< Points to the array of coefficients.  The array is of length 5*numStages. */
  int8_t postShift;        /**< Additional shift, in bits, applied to each output sample. */
} arm_biquad_casd_df1_inst_q15;

/**
 * @brief Instance structure for the Q31 Biquad cascade filter.
 */
typedef struct
{
  uint32_t numStages;      /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  q31_t *pState;          /**< Points to the array of state coefficients.  The array is of length 4*numStages. */
  q31_t *pCoeffs;          /**< Points to the array of coefficients.  The array is of length 5*numStages. */
  uint8_t postShift;       /**< Additional shift, in bits, applied to each output sample. */
```

```
} arm_biquad_casd_df1_inst_q31;

/**
 * @brief Instance structure for the floating-point Biquad cascade filter.
 */
typedef struct
{
  uint32_t numStages;      /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  float32_t *pState;       /**< Points to the array of state coefficients.  The array is of length 4*numStages. *
  float32_t *pCoeffs;      /**< Points to the array of coefficients.  The array is of length 5*numStages. */
} arm_biquad_casd_df1_inst_f32;


/**
 * @brief Processing function for the Q15 Biquad cascade filter.
 * @param[in]  S          points to an instance of the Q15 Biquad cascade structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df1_q15(
const arm_biquad_casd_df1_inst_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q15 Biquad cascade filter.
 * @param[in,out] S          points to an instance of the Q15 Biquad cascade structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     postShift  Shift to be applied to the output. Varies according to the coefficients format
 */
void arm_biquad_cascade_df1_init_q15(
arm_biquad_casd_df1_inst_q15 * S,
uint8_t numStages,
q15_t * pCoeffs,
q15_t * pState,
int8_t postShift);


/**
 * @brief Fast but less precise processing function for the Q15 Biquad cascade filter for Cortex-M3 and C
 * @param[in]  S          points to an instance of the Q15 Biquad cascade structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df1_fast_q15(
const arm_biquad_casd_df1_inst_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 Biquad cascade filter
 * @param[in]  S         points to an instance of the Q31 Biquad cascade structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df1_q31(
const arm_biquad_casd_df1_inst_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Fast but less precise processing function for the Q31 Biquad cascade filter for Cortex-M3 and C
 * @param[in]  S         points to an instance of the Q31 Biquad cascade structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df1_fast_q31(
const arm_biquad_casd_df1_inst_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q31 Biquad cascade filter.
 * @param[in,out] S          points to an instance of the Q31 Biquad cascade structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     postShift  Shift to be applied to the output. Varies according to the coefficients format
 */
void arm_biquad_cascade_df1_init_q31(
arm_biquad_casd_df1_inst_q31 * S,
uint8_t numStages,
q31_t * pCoeffs,
q31_t * pState,
int8_t postShift);


/**
 * @brief Processing function for the floating-point Biquad cascade filter.
 * @param[in]  S         points to an instance of the floating-point Biquad cascade structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df1_f32(
const arm_biquad_casd_df1_inst_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point Biquad cascade filter.
 * @param[in,out] S          points to an instance of the floating-point Biquad cascade structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 */
void arm_biquad_cascade_df1_init_f32(
arm_biquad_casd_df1_inst_f32 * S,
uint8_t numStages,
float32_t * pCoeffs,
float32_t * pState);


/**
 * @brief Instance structure for the floating-point matrix structure.
 */
typedef struct
{
  uint16_t numRows;     /**< number of rows of the matrix.     */
  uint16_t numCols;     /**< number of columns of the matrix.  */
  float32_t *pData;     /**< points to the data of the matrix. */
} arm_matrix_instance_f32;


/**
 * @brief Instance structure for the floating-point matrix structure.
 */
typedef struct
{
  uint16_t numRows;     /**< number of rows of the matrix.     */
  uint16_t numCols;     /**< number of columns of the matrix.  */
  float64_t *pData;     /**< points to the data of the matrix. */
} arm_matrix_instance_f64;

/**
 * @brief Instance structure for the Q15 matrix structure.
 */
typedef struct
{
  uint16_t numRows;     /**< number of rows of the matrix.     */
  uint16_t numCols;     /**< number of columns of the matrix.  */
  q15_t *pData;         /**< points to the data of the matrix. */
} arm_matrix_instance_q15;
```

```
/**
 * @brief Instance structure for the Q31 matrix structure.
 */
typedef struct
{
  uint16_t numRows;     /**< number of rows of the matrix.     */
  uint16_t numCols;     /**< number of columns of the matrix.  */
  q31_t *pData;         /**< points to the data of the matrix. */
} arm_matrix_instance_q31;


/**
 * @brief Floating-point matrix addition.
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_add_f32(
const arm_matrix_instance_f32 * pSrcA,
const arm_matrix_instance_f32 * pSrcB,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15 matrix addition.
 * @param[in]   pSrcA  points to the first input matrix structure
 * @param[in]   pSrcB  points to the second input matrix structure
 * @param[out]  pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_add_q15(
const arm_matrix_instance_q15 * pSrcA,
const arm_matrix_instance_q15 * pSrcB,
arm_matrix_instance_q15 * pDst);


/**
 * @brief Q31 matrix addition.
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_add_q31(
const arm_matrix_instance_q31 * pSrcA,
const arm_matrix_instance_q31 * pSrcB,
arm_matrix_instance_q31 * pDst);
```

```
/**
 * @brief Floating-point, complex, matrix multiplication.
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_cmplx_mult_f32(
const arm_matrix_instance_f32 * pSrcA,
const arm_matrix_instance_f32 * pSrcB,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15, complex,  matrix multiplication.
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_cmplx_mult_q15(
const arm_matrix_instance_q15 * pSrcA,
const arm_matrix_instance_q15 * pSrcB,
arm_matrix_instance_q15 * pDst,
q15_t * pScratch);


/**
 * @brief Q31, complex, matrix multiplication.
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_cmplx_mult_q31(
const arm_matrix_instance_q31 * pSrcA,
const arm_matrix_instance_q31 * pSrcB,
arm_matrix_instance_q31 * pDst);


/**
 * @brief Floating-point matrix transpose.
 * @param[in]  pSrc  points to the input matrix
 * @param[out] pDst  points to the output matrix
 * @return    The function returns either  <code>ARM_MATH_SIZE_MISMATCH</code>
 * or <code>ARM_MATH_SUCCESS</code> based on the outcome of size checking.
 */
arm_status arm_mat_trans_f32(
const arm_matrix_instance_f32 * pSrc,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15 matrix transpose.
 * @param[in]  pSrc  points to the input matrix
 * @param[out] pDst  points to the output matrix
 * @return    The function returns either  <code>ARM_MATH_SIZE_MISMATCH</code>
 * or <code>ARM_MATH_SUCCESS</code> based on the outcome of size checking.
 */
arm_status arm_mat_trans_q15(
const arm_matrix_instance_q15 * pSrc,
arm_matrix_instance_q15 * pDst);


/**
 * @brief Q31 matrix transpose.
 * @param[in]  pSrc  points to the input matrix
 * @param[out] pDst  points to the output matrix
 * @return    The function returns either  <code>ARM_MATH_SIZE_MISMATCH</code>
 * or <code>ARM_MATH_SUCCESS</code> based on the outcome of size checking.
 */
arm_status arm_mat_trans_q31(
const arm_matrix_instance_q31 * pSrc,
arm_matrix_instance_q31 * pDst);


/**
 * @brief Floating-point matrix multiplication
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_mult_f32(
const arm_matrix_instance_f32 * pSrcA,
const arm_matrix_instance_f32 * pSrcB,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15 matrix multiplication
 * @param[in]  pSrcA   points to the first input matrix structure
 * @param[in]  pSrcB   points to the second input matrix structure
 * @param[out] pDst    points to output matrix structure
 * @param[in]  pState  points to the array for storing intermediate results
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_mult_q15(
const arm_matrix_instance_q15 * pSrcA,
const arm_matrix_instance_q15 * pSrcB,
arm_matrix_instance_q15 * pDst,
q15_t * pState);


/**
 * @brief Q15 matrix multiplication (fast variant) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA   points to the first input matrix structure
 * @param[in]  pSrcB   points to the second input matrix structure
 * @param[out] pDst    points to output matrix structure
 * @param[in]  pState  points to the array for storing intermediate results
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_mult_fast_q15(
const arm_matrix_instance_q15 * pSrcA,
const arm_matrix_instance_q15 * pSrcB,
arm_matrix_instance_q15 * pDst,
q15_t * pState);


/**
 * @brief Q31 matrix multiplication
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_mult_q31(
const arm_matrix_instance_q31 * pSrcA,
const arm_matrix_instance_q31 * pSrcB,
arm_matrix_instance_q31 * pDst);


/**
 * @brief Q31 matrix multiplication (fast variant) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_mult_fast_q31(
const arm_matrix_instance_q31 * pSrcA,
const arm_matrix_instance_q31 * pSrcB,
arm_matrix_instance_q31 * pDst);


/**
 * @brief Floating-point matrix subtraction
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_sub_f32(
const arm_matrix_instance_f32 * pSrcA,
const arm_matrix_instance_f32 * pSrcB,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15 matrix subtraction
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_sub_q15(
const arm_matrix_instance_q15 * pSrcA,
const arm_matrix_instance_q15 * pSrcB,
arm_matrix_instance_q15 * pDst);


/**
 * @brief Q31 matrix subtraction
 * @param[in]  pSrcA  points to the first input matrix structure
 * @param[in]  pSrcB  points to the second input matrix structure
 * @param[out] pDst   points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_sub_q31(
const arm_matrix_instance_q31 * pSrcA,
const arm_matrix_instance_q31 * pSrcB,
arm_matrix_instance_q31 * pDst);


/**
 * @brief Floating-point matrix scaling.
 * @param[in]  pSrc   points to the input matrix
 * @param[in]  scale  scale factor
 * @param[out] pDst   points to the output matrix
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_scale_f32(
const arm_matrix_instance_f32 * pSrc,
float32_t scale,
arm_matrix_instance_f32 * pDst);


/**
 * @brief Q15 matrix scaling.
 * @param[in]  pSrc        points to input matrix
 * @param[in]  scaleFract  fractional portion of the scale factor
 * @param[in]  shift       number of bits to shift the result by
 * @param[out] pDst        points to output matrix
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_scale_q15(
const arm_matrix_instance_q15 * pSrc,
q15_t scaleFract,
int32_t shift,
arm_matrix_instance_q15 * pDst);


/**
 * @brief Q31 matrix scaling.
 * @param[in]  pSrc        points to input matrix
 * @param[in]  scaleFract  fractional portion of the scale factor
 * @param[in]  shift       number of bits to shift the result by
 * @param[out] pDst        points to output matrix structure
 * @return     The function returns either
 * <code>ARM_MATH_SIZE_MISMATCH</code> or <code>ARM_MATH_SUCCESS</code> based on t
 */
arm_status arm_mat_scale_q31(
const arm_matrix_instance_q31 * pSrc,
q31_t scaleFract,
int32_t shift,
arm_matrix_instance_q31 * pDst);


/**
 * @brief  Q31 matrix initialization.
 * @param[in,out] S         points to an instance of the floating-point matrix structure.
 * @param[in]     nRows     number of rows in the matrix.
 * @param[in]     nColumns  number of columns in the matrix.
 * @param[in]     pData     points to the matrix data array.
 */
void arm_mat_init_q31(
arm_matrix_instance_q31 * S,
uint16_t nRows,
uint16_t nColumns,
q31_t * pData);


/**
 * @brief  Q15 matrix initialization.
 * @param[in,out] S         points to an instance of the floating-point matrix structure.
 * @param[in]     nRows     number of rows in the matrix.
 * @param[in]     nColumns  number of columns in the matrix.
 * @param[in]     pData     points to the matrix data array.
 */
void arm_mat_init_q15(
arm_matrix_instance_q15 * S,
  uint16_t nRows,
  uint16_t nColumns,
  q15_t * pData);


  /**
   * @brief  Floating-point matrix initialization.
   * @param[in,out] S         points to an instance of the floating-point matrix structure.
   * @param[in]     nRows     number of rows in the matrix.
   * @param[in]     nColumns  number of columns in the matrix.
   * @param[in]     pData     points to the matrix data array.
   */
  void arm_mat_init_f32(
  arm_matrix_instance_f32 * S,
  uint16_t nRows,
  uint16_t nColumns,
  float32_t * pData);



  /**
   * @brief Instance structure for the Q15 PID Control.
   */
  typedef struct
  {
    q15_t A0;           /**< The derived gain, A0 = Kp + Ki + Kd . */
#ifdef ARM_MATH_CM0_FAMILY
    q15_t A1;
    q15_t A2;
#else
    q31_t A1;           /**< The derived gain A1 = -Kp - 2Kd | Kd.*/
#endif
    q15_t state[3];     /**< The state array of length 3. */
    q15_t Kp;           /**< The proportional gain. */
    q15_t Ki;           /**< The integral gain. */
    q15_t Kd;           /**< The derivative gain. */
  } arm_pid_instance_q15;

  /**
   * @brief Instance structure for the Q31 PID Control.
   */
  typedef struct
  {
    q31_t A0;            /**< The derived gain, A0 = Kp + Ki + Kd . */
    q31_t A1;            /**< The derived gain, A1 = -Kp - 2Kd. */
    q31_t A2;            /**< The derived gain, A2 = Kd . */
    q31_t state[3];      /**< The state array of length 3. */
    q31_t Kp;            /**< The proportional gain. */
    q31_t Ki;            /**< The integral gain. */
    q31_t Kd;            /**< The derivative gain. */
  } arm_pid_instance_q31;

  /**
```

```
 * @brief Instance structure for the floating-point PID Control.
 */
typedef struct
{
  float32_t A0;          /**< The derived gain, A0 = Kp + Ki + Kd . */
  float32_t A1;          /**< The derived gain, A1 = -Kp - 2Kd. */
  float32_t A2;          /**< The derived gain, A2 = Kd . */
  float32_t state[3];    /**< The state array of length 3. */
  float32_t Kp;          /**< The proportional gain. */
  float32_t Ki;         /**< The integral gain. */
  float32_t Kd;          /**< The derivative gain. */
} arm_pid_instance_f32;



/**
 * @brief  Initialization function for the floating-point PID Control.
 * @param[in,out] S               points to an instance of the PID structure.
 * @param[in]     resetStateFlag  flag to reset the state. 0 = no change in state 1 = reset the state.
 */
void arm_pid_init_f32(
arm_pid_instance_f32 * S,
int32_t resetStateFlag);


/**
 * @brief  Reset function for the floating-point PID Control.
 * @param[in,out] S  is an instance of the floating-point PID Control structure
 */
void arm_pid_reset_f32(
arm_pid_instance_f32 * S);


/**
 * @brief  Initialization function for the Q31 PID Control.
 * @param[in,out] S               points to an instance of the Q15 PID structure.
 * @param[in]     resetStateFlag  flag to reset the state. 0 = no change in state 1 = reset the state.
 */
void arm_pid_init_q31(
arm_pid_instance_q31 * S,
int32_t resetStateFlag);


/**
 * @brief  Reset function for the Q31 PID Control.
 * @param[in,out] S   points to an instance of the Q31 PID Control structure
 */

void arm_pid_reset_q31(
arm_pid_instance_q31 * S);


/**
```

```
 * @brief  Initialization function for the Q15 PID Control.
 * @param[in,out] S               points to an instance of the Q15 PID structure.
 * @param[in]     resetStateFlag  flag to reset the state. 0 = no change in state 1 = reset the state.
 */
void arm_pid_init_q15(
arm_pid_instance_q15 * S,
int32_t resetStateFlag);


/**
 * @brief  Reset function for the Q15 PID Control.
 * @param[in,out] S  points to an instance of the q15 PID Control structure
 */
void arm_pid_reset_q15(
arm_pid_instance_q15 * S);


/**
 * @brief Instance structure for the floating-point Linear Interpolate function.
 */
typedef struct
{
  uint32_t nValues;           /**< nValues */
  float32_t x1;               /**< x1 */
  float32_t xSpacing;         /**< xSpacing */
  float32_t *pYData;          /**< pointer to the table of Y values */
} arm_linear_interp_instance_f32;

/**
 * @brief Instance structure for the floating-point bilinear interpolation function.
 */
typedef struct
{
  uint16_t numRows;   /**< number of rows in the data table. */
  uint16_t numCols;   /**< number of columns in the data table. */
  float32_t *pData;   /**< points to the data table. */
} arm_bilinear_interp_instance_f32;

 /**
 * @brief Instance structure for the Q31 bilinear interpolation function.
 */
typedef struct
{
  uint16_t numRows;   /**< number of rows in the data table. */
  uint16_t numCols;   /**< number of columns in the data table. */
  q31_t *pData;       /**< points to the data table. */
} arm_bilinear_interp_instance_q31;

 /**
 * @brief Instance structure for the Q15 bilinear interpolation function.
 */
typedef struct
{
  uint16_t numRows;   /**< number of rows in the data table. */
  uint16_t numCols;   /**< number of columns in the data table. */
  q15_t *pData;       /**< points to the data table. */
} arm_bilinear_interp_instance_q15;

 /**
 * @brief Instance structure for the Q15 bilinear interpolation function.
 */
typedef struct
{
  uint16_t numRows;   /**< number of rows in the data table. */
  uint16_t numCols;   /**< number of columns in the data table. */
  q7_t *pData;        /**< points to the data table. */
} arm_bilinear_interp_instance_q7;


/**
 * @brief Q7 vector multiplication.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_mult_q7(
q7_t * pSrcA,
q7_t * pSrcB,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief Q15 vector multiplication.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_mult_q15(
q15_t * pSrcA,
q15_t * pSrcB,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Q31 vector multiplication.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_mult_q31(
q31_t * pSrcA,
  q31_t * pSrcB,
  q31_t * pDst,
  uint32_t blockSize);


  /**
   * @brief Floating-point vector multiplication.
   * @param[in]  pSrcA      points to the first input vector
   * @param[in]  pSrcB      points to the second input vector
   * @param[out] pDst       points to the output vector
   * @param[in]  blockSize  number of samples in each vector
   */
  void arm_mult_f32(
  float32_t * pSrcA,
  float32_t * pSrcB,
  float32_t * pDst,
  uint32_t blockSize);


  /**
   * @brief Instance structure for the Q15 CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0) b
    q15_t *pTwiddle;                 /**< points to the Sin twiddle factor table. */
    uint16_t *pBitRevTable;          /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;       /**< twiddle coefficient modifier that supports different size FFTs with the s
    uint16_t bitRevFactor;           /**< bit reversal modifier that supports different size FFTs with the same bit
  } arm_cfft_radix2_instance_q15;

/* Deprecated */
  arm_status arm_cfft_radix2_init_q15(
  arm_cfft_radix2_instance_q15 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

/* Deprecated */
  void arm_cfft_radix2_q15(
  const arm_cfft_radix2_instance_q15 * S,
  q15_t * pSrc);


  /**
   * @brief Instance structure for the Q15 CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0) b
    q15_t *pTwiddle;                 /**< points to the twiddle factor table. */
    uint16_t *pBitRevTable;          /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;       /**< twiddle coefficient modifier that supports different size FFTs with the s
    uint16_t bitRevFactor;           /**< bit reversal modifier that supports different size FFTs with the same bit
  } arm_cfft_radix4_instance_q15;

/* Deprecated */
  arm_status arm_cfft_radix4_init_q15(
  arm_cfft_radix4_instance_q15 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

/* Deprecated */
  void arm_cfft_radix4_q15(
  const arm_cfft_radix4_instance_q15 * S,
  q15_t * pSrc);

  /**
   * @brief Instance structure for the Radix-2 Q31 CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0) b
    q31_t *pTwiddle;                 /**< points to the Twiddle factor table. */
    uint16_t *pBitRevTable;          /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;       /**< twiddle coefficient modifier that supports different size FFTs with the s
    uint16_t bitRevFactor;           /**< bit reversal modifier that supports different size FFTs with the same bit
  } arm_cfft_radix2_instance_q31;

/* Deprecated */
  arm_status arm_cfft_radix2_init_q31(
  arm_cfft_radix2_instance_q31 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

/* Deprecated */
  void arm_cfft_radix2_q31(
  const arm_cfft_radix2_instance_q31 * S,
  q31_t * pSrc);

  /**
   * @brief Instance structure for the Q31 CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0) b
    q31_t *pTwiddle;                 /**< points to the twiddle factor table. */
    uint16_t *pBitRevTable;          /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;       /**< twiddle coefficient modifier that supports different size FFTs with the s
    uint16_t bitRevFactor;           /**< bit reversal modifier that supports different size FFTs with the same bit
  } arm_cfft_radix4_instance_q31;

/* Deprecated */
  void arm_cfft_radix4_q31(
  const arm_cfft_radix4_instance_q31 * S,
  q31_t * pSrc);

/* Deprecated */
  arm_status arm_cfft_radix4_init_q31(
  arm_cfft_radix4_instance_q31 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

  /**
   * @brief Instance structure for the floating-point CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0)
    float32_t *pTwiddle;             /**< points to the Twiddle factor table. */
    uint16_t *pBitRevTable;          /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;       /**< twiddle coefficient modifier that supports different size FFTs with the
    uint16_t bitRevFactor;           /**< bit reversal modifier that supports different size FFTs with the same b
    float32_t onebyfftLen;           /**< value of 1/fftLen. */
  } arm_cfft_radix2_instance_f32;

/* Deprecated */
  arm_status arm_cfft_radix2_init_f32(
  arm_cfft_radix2_instance_f32 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

/* Deprecated */
  void arm_cfft_radix2_f32(
  const arm_cfft_radix2_instance_f32 * S,
  float32_t * pSrc);

  /**
   * @brief Instance structure for the floating-point CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                 /**< length of the FFT. */
    uint8_t ifftFlag;                /**< flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform. */
    uint8_t bitReverseFlag;          /**< flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0)
    float32_t *pTwiddle;               /**< points to the Twiddle factor table. */
    uint16_t *pBitRevTable;            /**< points to the bit reversal table. */
    uint16_t twidCoefModifier;         /**< twiddle coefficient modifier that supports different size FFTs with the
    uint16_t bitRevFactor;             /**< bit reversal modifier that supports different size FFTs with the same b
    float32_t onebyfftLen;             /**< value of 1/fftLen. */
  } arm_cfft_radix4_instance_f32;

/* Deprecated */
  arm_status arm_cfft_radix4_init_f32(
  arm_cfft_radix4_instance_f32 * S,
  uint16_t fftLen,
  uint8_t ifftFlag,
  uint8_t bitReverseFlag);

/* Deprecated */
  void arm_cfft_radix4_f32(
  const arm_cfft_radix4_instance_f32 * S,
  float32_t * pSrc);

  /**
   * @brief Instance structure for the fixed-point CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                   /**< length of the FFT. */
    const q15_t *pTwiddle;             /**< points to the Twiddle factor table. */
    const uint16_t *pBitRevTable;      /**< points to the bit reversal table. */
    uint16_t bitRevLength;             /**< bit reversal table length. */
  } arm_cfft_instance_q15;

void arm_cfft_q15(
    const arm_cfft_instance_q15 * S,
    q15_t * p1,
    uint8_t ifftFlag,
    uint8_t bitReverseFlag);

  /**
   * @brief Instance structure for the fixed-point CFFT/CIFFT function.
   */
  typedef struct
  {
    uint16_t fftLen;                   /**< length of the FFT. */
    const q31_t *pTwiddle;             /**< points to the Twiddle factor table. */
    const uint16_t *pBitRevTable;      /**< points to the bit reversal table. */
    uint16_t bitRevLength;             /**< bit reversal table length. */
  } arm_cfft_instance_q31;

void arm_cfft_q31(
    const arm_cfft_instance_q31 * S,
    q31_t * p1,
    uint8_t ifftFlag,
    uint8_t bitReverseFlag);
```

```
/**
 * @brief Instance structure for the floating-point CFFT/CIFFT function.
 */
typedef struct
{
  uint16_t fftLen;                   /**< length of the FFT. */
  const float32_t *pTwiddle;         /**< points to the Twiddle factor table. */
  const uint16_t *pBitRevTable;      /**< points to the bit reversal table. */
  uint16_t bitRevLength;             /**< bit reversal table length. */
} arm_cfft_instance_f32;

void arm_cfft_f32(
const arm_cfft_instance_f32 * S,
float32_t * p1,
uint8_t ifftFlag,
uint8_t bitReverseFlag);

/**
 * @brief Instance structure for the Q15 RFFT/RIFFT function.
 */
typedef struct
{
  uint32_t fftLenReal;                        /**< length of the real FFT. */
  uint8_t ifftFlagR;                          /**< flag that selects forward (ifftFlagR=0) or inverse (ifftFlagR=1) transfo
  uint8_t bitReverseFlagR;                    /**< flag that enables (bitReverseFlagR=1) or disables (bitReverseF
  uint32_t twidCoefRModifier;                 /**< twiddle coefficient modifier that supports different size FFTs wi
  q15_t *pTwiddleAReal;                       /**< points to the real twiddle factor table. */
  q15_t *pTwiddleBReal;                       /**< points to the imag twiddle factor table. */
  const arm_cfft_instance_q15 *pCfft;         /**< points to the complex FFT instance. */
} arm_rfft_instance_q15;

arm_status arm_rfft_init_q15(
arm_rfft_instance_q15 * S,
uint32_t fftLenReal,
uint32_t ifftFlagR,
uint32_t bitReverseFlag);

void arm_rfft_q15(
const arm_rfft_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst);

/**
 * @brief Instance structure for the Q31 RFFT/RIFFT function.
 */
typedef struct
{
  uint32_t fftLenReal;                        /**< length of the real FFT. */
  uint8_t ifftFlagR;                          /**< flag that selects forward (ifftFlagR=0) or inverse (ifftFlagR=1) transf
  uint8_t bitReverseFlagR;                    /**< flag that enables (bitReverseFlagR=1) or disables (bitReverse
  uint32_t twidCoefRModifier;                 /**< twiddle coefficient modifier that supports different size FFTs w
  q31_t *pTwiddleAReal;                       /**< points to the real twiddle factor table. */
  q31_t *pTwiddleBReal;                       /**< points to the imag twiddle factor table. */
```

```
    const arm_cfft_instance_q31 *pCfft;         /**< points to the complex FFT instance. */
  } arm_rfft_instance_q31;

  arm_status arm_rfft_init_q31(
  arm_rfft_instance_q31 * S,
  uint32_t fftLenReal,
  uint32_t ifftFlagR,
  uint32_t bitReverseFlag);

  void arm_rfft_q31(
  const arm_rfft_instance_q31 * S,
  q31_t * pSrc,
  q31_t * pDst);

  /**
   * @brief Instance structure for the floating-point RFFT/RIFFT function.
   */
  typedef struct
  {
    uint32_t fftLenReal;                        /**< length of the real FFT. */
    uint16_t fftLenBy2;                         /**< length of the complex FFT. */
    uint8_t ifftFlagR;                          /**< flag that selects forward (ifftFlagR=0) or inverse (ifftFlagR=1) trans
    uint8_t bitReverseFlagR;                    /**< flag that enables (bitReverseFlagR=1) or disables (bitReverse
    uint32_t twidCoefRModifier;                 /**< twiddle coefficient modifier that supports different size FFT
    float32_t *pTwiddleAReal;                   /**< points to the real twiddle factor table. */
    float32_t *pTwiddleBReal;                   /**< points to the imag twiddle factor table. */
    arm_cfft_radix4_instance_f32 *pCfft;        /**< points to the complex FFT instance. */
  } arm_rfft_instance_f32;

  arm_status arm_rfft_init_f32(
  arm_rfft_instance_f32 * S,
  arm_cfft_radix4_instance_f32 * S_CFFT,
  uint32_t fftLenReal,
  uint32_t ifftFlagR,
  uint32_t bitReverseFlag);

  void arm_rfft_f32(
  const arm_rfft_instance_f32 * S,
  float32_t * pSrc,
  float32_t * pDst);

  /**
   * @brief Instance structure for the floating-point RFFT/RIFFT function.
   */
typedef struct
  {
    arm_cfft_instance_f32 Sint;      /**< Internal CFFT structure. */
    uint16_t fftLenRFFT;             /**< length of the real sequence */
    float32_t * pTwiddleRFFT;        /**< Twiddle factors real stage  */
  } arm_rfft_fast_instance_f32 ;

arm_status arm_rfft_fast_init_f32 (
   arm_rfft_fast_instance_f32 * S,
   uint16_t fftLen);

void arm_rfft_fast_f32(
  arm_rfft_fast_instance_f32 * S,
  float32_t * p, float32_t * pOut,
  uint8_t ifftFlag);

  /**
   * @brief Instance structure for the floating-point DCT4/IDCT4 function.
   */
  typedef struct
  {
    uint16_t N;                          /**< length of the DCT4. */
    uint16_t Nby2;                        /**< half of the length of the DCT4. */
    float32_t normalize;                  /**< normalizing factor. */
    float32_t *pTwiddle;                  /**< points to the twiddle factor table. */
    float32_t *pCosFactor;                /**< points to the cosFactor table. */
    arm_rfft_instance_f32 *pRfft;         /**< points to the real FFT instance. */
    arm_cfft_radix4_instance_f32 *pCfft; /**< points to the complex FFT instance. */
  } arm_dct4_instance_f32;


  /**
   * @brief  Initialization function for the floating-point DCT4/IDCT4.
   * @param[in,out] S          points to an instance of floating-point DCT4/IDCT4 structure.
   * @param[in]     S_RFFT     points to an instance of floating-point RFFT/RIFFT structure.
   * @param[in]     S_CFFT     points to an instance of floating-point CFFT/CIFFT structure.
   * @param[in]     N          length of the DCT4.
   * @param[in]     Nby2       half of the length of the DCT4.
   * @param[in]     normalize  normalizing factor.
   * @return      arm_status function returns ARM_MATH_SUCCESS if initialization is successful or ARM_M
   */
  arm_status arm_dct4_init_f32(
  arm_dct4_instance_f32 * S,
  arm_rfft_instance_f32 * S_RFFT,
  arm_cfft_radix4_instance_f32 * S_CFFT,
  uint16_t N,
  uint16_t Nby2,
  float32_t normalize);


  /**
   * @brief Processing function for the floating-point DCT4/IDCT4.
   * @param[in]     S             points to an instance of the floating-point DCT4/IDCT4 structure.
   * @param[in]     pState         points to state buffer.
   * @param[in,out] pInlineBuffer  points to the in-place input and output buffer.
   */
  void arm_dct4_f32(
  const arm_dct4_instance_f32 * S,
  float32_t * pState,
  float32_t * pInlineBuffer);
```

```
/**
 * @brief Instance structure for the Q31 DCT4/IDCT4 function.
 */
typedef struct
{
  uint16_t N;                          /**< length of the DCT4. */
  uint16_t Nby2;                       /**< half of the length of the DCT4. */
  q31_t normalize;                     /**< normalizing factor. */
  q31_t *pTwiddle;                     /**< points to the twiddle factor table. */
  q31_t *pCosFactor;                   /**< points to the cosFactor table. */
  arm_rfft_instance_q31 *pRfft;        /**< points to the real FFT instance. */
  arm_cfft_radix4_instance_q31 *pCfft; /**< points to the complex FFT instance. */
} arm_dct4_instance_q31;


/**
 * @brief  Initialization function for the Q31 DCT4/IDCT4.
 * @param[in,out] S          points to an instance of Q31 DCT4/IDCT4 structure.
 * @param[in]     S_RFFT     points to an instance of Q31 RFFT/RIFFT structure
 * @param[in]     S_CFFT     points to an instance of Q31 CFFT/CIFFT structure
 * @param[in]     N          length of the DCT4.
 * @param[in]     Nby2       half of the length of the DCT4.
 * @param[in]     normalize  normalizing factor.
 * @return      arm_status function returns ARM_MATH_SUCCESS if initialization is successful or ARM_M
 */
arm_status arm_dct4_init_q31(
arm_dct4_instance_q31 * S,
arm_rfft_instance_q31 * S_RFFT,
arm_cfft_radix4_instance_q31 * S_CFFT,
uint16_t N,
uint16_t Nby2,
q31_t normalize);


/**
 * @brief Processing function for the Q31 DCT4/IDCT4.
 * @param[in]     S             points to an instance of the Q31 DCT4 structure.
 * @param[in]     pState        points to state buffer.
 * @param[in,out] pInlineBuffer  points to the in-place input and output buffer.
 */
void arm_dct4_q31(
const arm_dct4_instance_q31 * S,
q31_t * pState,
q31_t * pInlineBuffer);


/**
 * @brief Instance structure for the Q15 DCT4/IDCT4 function.
 */
typedef struct
{
  uint16_t N;                          /**< length of the DCT4. */
  uint16_t Nby2;                       /**< half of the length of the DCT4. */
  q15_t normalize;                    /**< normalizing factor. */
  q15_t *pTwiddle;                    /**< points to the twiddle factor table. */
  q15_t *pCosFactor;                  /**< points to the cosFactor table. */
  arm_rfft_instance_q15 *pRfft;        /**< points to the real FFT instance. */
  arm_cfft_radix4_instance_q15 *pCfft; /**< points to the complex FFT instance. */
} arm_dct4_instance_q15;


/**
 * @brief  Initialization function for the Q15 DCT4/IDCT4.
 * @param[in,out] S          points to an instance of Q15 DCT4/IDCT4 structure.
 * @param[in]     S_RFFT     points to an instance of Q15 RFFT/RIFFT structure.
 * @param[in]     S_CFFT     points to an instance of Q15 CFFT/CIFFT structure.
 * @param[in]     N          length of the DCT4.
 * @param[in]     Nby2       half of the length of the DCT4.
 * @param[in]     normalize  normalizing factor.
 * @return      arm_status function returns ARM_MATH_SUCCESS if initialization is successful or ARM_M
 */
arm_status arm_dct4_init_q15(
arm_dct4_instance_q15 * S,
arm_rfft_instance_q15 * S_RFFT,
arm_cfft_radix4_instance_q15 * S_CFFT,
uint16_t N,
uint16_t Nby2,
q15_t normalize);


/**
 * @brief Processing function for the Q15 DCT4/IDCT4.
 * @param[in]     S             points to an instance of the Q15 DCT4 structure.
 * @param[in]     pState         points to state buffer.
 * @param[in,out] pInlineBuffer  points to the in-place input and output buffer.
 */
void arm_dct4_q15(
const arm_dct4_instance_q15 * S,
q15_t * pState,
q15_t * pInlineBuffer);


/**
 * @brief Floating-point vector addition.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_add_f32(
float32_t * pSrcA,
float32_t * pSrcB,
float32_t * pDst,
uint32_t blockSize);
```

```
/**
 * @brief Q7 vector addition.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_add_q7(
q7_t * pSrcA,
q7_t * pSrcB,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief Q15 vector addition.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_add_q15(
q15_t * pSrcA,
q15_t * pSrcB,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Q31 vector addition.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_add_q31(
q31_t * pSrcA,
q31_t * pSrcB,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Floating-point vector subtraction.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_sub_f32(
float32_t * pSrcA,
float32_t * pSrcB,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Q7 vector subtraction.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_sub_q7(
q7_t * pSrcA,
q7_t * pSrcB,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief Q15 vector subtraction.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_sub_q15(
q15_t * pSrcA,
q15_t * pSrcB,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Q31 vector subtraction.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_sub_q31(
q31_t * pSrcA,
q31_t * pSrcB,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Multiplies a floating-point vector by a scalar.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  scale      scale factor to be applied
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_scale_f32(
float32_t * pSrc,
float32_t scale,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Multiplies a Q7 vector by a scalar.
 * @param[in]  pSrc        points to the input vector
 * @param[in]  scaleFract  fractional portion of the scale value
 * @param[in]  shift       number of bits to shift the result by
 * @param[out] pDst        points to the output vector
 * @param[in]  blockSize   number of samples in the vector
 */
void arm_scale_q7(
q7_t * pSrc,
q7_t scaleFract,
int8_t shift,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief Multiplies a Q15 vector by a scalar.
 * @param[in]  pSrc        points to the input vector
 * @param[in]  scaleFract  fractional portion of the scale value
 * @param[in]  shift       number of bits to shift the result by
 * @param[out] pDst        points to the output vector
 * @param[in]  blockSize   number of samples in the vector
 */
void arm_scale_q15(
q15_t * pSrc,
q15_t scaleFract,
int8_t shift,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Multiplies a Q31 vector by a scalar.
 * @param[in]  pSrc        points to the input vector
 * @param[in]  scaleFract  fractional portion of the scale value
 * @param[in]  shift       number of bits to shift the result by
 * @param[out] pDst        points to the output vector
 * @param[in]  blockSize   number of samples in the vector
 */
void arm_scale_q31(
q31_t * pSrc,
q31_t scaleFract,
int8_t shift,
q31_t * pDst,
uint32_t blockSize);
```

```
/**
 * @brief Q7 vector absolute value.
 * @param[in]  pSrc       points to the input buffer
 * @param[out] pDst       points to the output buffer
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_abs_q7(
q7_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief Floating-point vector absolute value.
 * @param[in]  pSrc       points to the input buffer
 * @param[out] pDst       points to the output buffer
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_abs_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Q15 vector absolute value.
 * @param[in]  pSrc       points to the input buffer
 * @param[out] pDst       points to the output buffer
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_abs_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Q31 vector absolute value.
 * @param[in]  pSrc       points to the input buffer
 * @param[out] pDst       points to the output buffer
 * @param[in]  blockSize  number of samples in each vector
 */
void arm_abs_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Dot product of floating-point vectors.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[in]  blockSize  number of samples in each vector
 * @param[out] result     output result returned here
 */
void arm_dot_prod_f32(
float32_t * pSrcA,
float32_t * pSrcB,
uint32_t blockSize,
float32_t * result);


/**
 * @brief Dot product of Q7 vectors.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[in]  blockSize  number of samples in each vector
 * @param[out] result     output result returned here
 */
void arm_dot_prod_q7(
q7_t * pSrcA,
q7_t * pSrcB,
uint32_t blockSize,
q31_t * result);


/**
 * @brief Dot product of Q15 vectors.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[in]  blockSize  number of samples in each vector
 * @param[out] result     output result returned here
 */
void arm_dot_prod_q15(
q15_t * pSrcA,
q15_t * pSrcB,
uint32_t blockSize,
q63_t * result);


/**
 * @brief Dot product of Q31 vectors.
 * @param[in]  pSrcA      points to the first input vector
 * @param[in]  pSrcB      points to the second input vector
 * @param[in]  blockSize  number of samples in each vector
 * @param[out] result     output result returned here
 */
void arm_dot_prod_q31(
q31_t * pSrcA,
q31_t * pSrcB,
uint32_t blockSize,
q63_t * result);


/**
```

```
 * @brief  Shifts the elements of a Q7 vector a specified number of bits.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  shiftBits  number of bits to shift.  A positive value shifts left; a negative value shifts right.
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_shift_q7(
q7_t * pSrc,
int8_t shiftBits,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Shifts the elements of a Q15 vector a specified number of bits.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  shiftBits  number of bits to shift.  A positive value shifts left; a negative value shifts right.
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_shift_q15(
q15_t * pSrc,
int8_t shiftBits,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Shifts the elements of a Q31 vector a specified number of bits.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  shiftBits  number of bits to shift.  A positive value shifts left; a negative value shifts right.
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_shift_q31(
q31_t * pSrc,
int8_t shiftBits,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Adds a constant offset to a floating-point vector.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  offset     is the offset to be added
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_offset_f32(
float32_t * pSrc,
float32_t offset,
float32_t * pDst,
uint32_t blockSize);
```

```
/**
 * @brief  Adds a constant offset to a Q7 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  offset     is the offset to be added
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_offset_q7(
q7_t * pSrc,
q7_t offset,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Adds a constant offset to a Q15 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  offset     is the offset to be added
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_offset_q15(
q15_t * pSrc,
q15_t offset,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Adds a constant offset to a Q31 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[in]  offset     is the offset to be added
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_offset_q31(
q31_t * pSrc,
q31_t offset,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Negates the elements of a floating-point vector.
 * @param[in]  pSrc       points to the input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_negate_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Negates the elements of a Q7 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_negate_q7(
q7_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Negates the elements of a Q15 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_negate_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Negates the elements of a Q31 vector.
 * @param[in]  pSrc       points to the input vector
 * @param[out] pDst       points to the output vector
 * @param[in]  blockSize  number of samples in the vector
 */
void arm_negate_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Copies the elements of a floating-point vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_copy_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Copies the elements of a Q7 vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_copy_q7(
q7_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Copies the elements of a Q15 vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_copy_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Copies the elements of a Q31 vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_copy_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Fills a constant value into a floating-point vector.
 * @param[in]  value      input value to be filled
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_fill_f32(
float32_t value,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Fills a constant value into a Q7 vector.
 * @param[in]  value      input value to be filled
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_fill_q7(
  q7_t value,
  q7_t * pDst,
  uint32_t blockSize);


  /**
   * @brief  Fills a constant value into a Q15 vector.
   * @param[in]  value      input value to be filled
   * @param[out] pDst       output pointer
   * @param[in]  blockSize  number of samples to process
   */
  void arm_fill_q15(
  q15_t value,
  q15_t * pDst,
  uint32_t blockSize);


  /**
   * @brief  Fills a constant value into a Q31 vector.
   * @param[in]  value      input value to be filled
   * @param[out] pDst       output pointer
   * @param[in]  blockSize  number of samples to process
   */
  void arm_fill_q31(
  q31_t value,
  q31_t * pDst,
  uint32_t blockSize);


/**
 * @brief Convolution of floating-point sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the location where the output result is written.  Length srcALen+srcBLen-1
 */
  void arm_conv_f32(
  float32_t * pSrcA,
  uint32_t srcALen,
  float32_t * pSrcB,
  uint32_t srcBLen,
  float32_t * pDst);


  /**
   * @brief Convolution of Q15 sequences.
   * @param[in]  pSrcA      points to the first input sequence.
   * @param[in]  srcALen    length of the first input sequence.
   * @param[in]  pSrcB      points to the second input sequence.
   * @param[in]  srcBLen    length of the second input sequence.
   * @param[out] pDst       points to the block of output data  Length srcALen+srcBLen-1.
   * @param[in]  pScratch1  points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBL
   * @param[in]  pScratch2  points to scratch buffer of size min(srcALen, srcBLen).
   */
  void arm_conv_opt_q15(
  q15_t * pSrcA,
  uint32_t srcALen,
  q15_t * pSrcB,
  uint32_t srcBLen,
  q15_t * pDst,
  q15_t * pScratch1,
  q15_t * pScratch2);


/**
 * @brief Convolution of Q15 sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the location where the output result is written.  Length srcALen+srcBLen-1
 */
  void arm_conv_q15(
  q15_t * pSrcA,
  uint32_t srcALen,
  q15_t * pSrcB,
  uint32_t srcBLen,
  q15_t * pDst);


  /**
   * @brief Convolution of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4
   * @param[in]  pSrcA    points to the first input sequence.
   * @param[in]  srcALen  length of the first input sequence.
   * @param[in]  pSrcB    points to the second input sequence.
   * @param[in]  srcBLen  length of the second input sequence.
   * @param[out] pDst     points to the block of output data  Length srcALen+srcBLen-1.
   */
  void arm_conv_fast_q15(
          q15_t * pSrcA,
          uint32_t srcALen,
          q15_t * pSrcB,
          uint32_t srcBLen,
          q15_t * pDst);


  /**
   * @brief Convolution of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4
   * @param[in]  pSrcA      points to the first input sequence.
   * @param[in]  srcALen    length of the first input sequence.
   * @param[in]  pSrcB      points to the second input sequence.
   * @param[in]  srcBLen    length of the second input sequence.
   * @param[out] pDst       points to the block of output data  Length srcALen+srcBLen-1.
   * @param[in]  pScratch1  points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBL
   * @param[in]  pScratch2  points to scratch buffer of size min(srcALen, srcBLen).
   */
  void arm_conv_fast_opt_q15(
  q15_t * pSrcA,
  uint32_t srcALen,
  q15_t * pSrcB,
  uint32_t srcBLen,
  q15_t * pDst,
  q15_t * pScratch1,
  q15_t * pScratch2);


  /**
   * @brief Convolution of Q31 sequences.
   * @param[in]  pSrcA    points to the first input sequence.
   * @param[in]  srcALen  length of the first input sequence.
   * @param[in]  pSrcB    points to the second input sequence.
   * @param[in]  srcBLen  length of the second input sequence.
   * @param[out] pDst     points to the block of output data  Length srcALen+srcBLen-1.
   */
  void arm_conv_q31(
  q31_t * pSrcA,
  uint32_t srcALen,
  q31_t * pSrcB,
  uint32_t srcBLen,
  q31_t * pDst);


  /**
   * @brief Convolution of Q31 sequences (fast version) for Cortex-M3 and Cortex-M4
   * @param[in]  pSrcA    points to the first input sequence.
   * @param[in]  srcALen  length of the first input sequence.
   * @param[in]  pSrcB    points to the second input sequence.
   * @param[in]  srcBLen  length of the second input sequence.
   * @param[out] pDst     points to the block of output data  Length srcALen+srcBLen-1.
   */
  void arm_conv_fast_q31(
  q31_t * pSrcA,
  uint32_t srcALen,
  q31_t * pSrcB,
  uint32_t srcBLen,
  q31_t * pDst);


    /**
   * @brief Convolution of Q7 sequences.
   * @param[in]  pSrcA      points to the first input sequence.
   * @param[in]  srcALen    length of the first input sequence.
   * @param[in]  pSrcB      points to the second input sequence.
   * @param[in]  srcBLen    length of the second input sequence.
   * @param[out] pDst       points to the block of output data  Length srcALen+srcBLen-1.
   * @param[in]  pScratch1  points to scratch buffer(of type q15_t) of size max(srcALen, srcBLen) + 2*min(s
   * @param[in]  pScratch2  points to scratch buffer (of type q15_t) of size min(srcALen, srcBLen).
   */
```

```
void arm_conv_opt_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst,
q15_t * pScratch1,
q15_t * pScratch2);


/**
 * @brief Convolution of Q7 sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length srcALen+srcBLen-1.
 */
void arm_conv_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst);


/**
 * @brief Partial convolution of floating-point sequences.
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AF
 */
arm_status arm_conv_partial_f32(
float32_t * pSrcA,
uint32_t srcALen,
float32_t * pSrcB,
uint32_t srcBLen,
float32_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);


/**
 * @brief Partial convolution of Q15 sequences.
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @param[in]  pScratch1   points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBl
 * @param[in]  pScratch2   points to scratch buffer of size min(srcALen, srcBLen).
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AF
 */
arm_status arm_conv_partial_opt_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
uint32_t firstIndex,
uint32_t numPoints,
q15_t * pScratch1,
q15_t * pScratch2);


/**
 * @brief Partial convolution of Q15 sequences.
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AF
 */
arm_status arm_conv_partial_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);


/**
 * @brief Partial convolution of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AF
 */
arm_status arm_conv_partial_fast_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);


/**
 * @brief Partial convolution of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @param[in]  pScratch1   points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBl
 * @param[in]  pScratch2   points to scratch buffer of size min(srcALen, srcBLen).
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AR
 */
arm_status arm_conv_partial_fast_opt_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
uint32_t firstIndex,
uint32_t numPoints,
q15_t * pScratch1,
q15_t * pScratch2);


/**
 * @brief Partial convolution of Q31 sequences.
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AR
 */
arm_status arm_conv_partial_q31(
q31_t * pSrcA,
uint32_t srcALen,
q31_t * pSrcB,
uint32_t srcBLen,
q31_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);
```

```
/**
 * @brief Partial convolution of Q31 sequences (fast version) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AR
 */
arm_status arm_conv_partial_fast_q31(
q31_t * pSrcA,
uint32_t srcALen,
q31_t * pSrcB,
uint32_t srcBLen,
q31_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);


/**
 * @brief Partial convolution of Q7 sequences
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @param[in]  pScratch1   points to scratch buffer(of type q15_t) of size max(srcALen, srcBLen) + 2*min(
 * @param[in]  pScratch2   points to scratch buffer (of type q15_t) of size min(srcALen, srcBLen).
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AR
 */
arm_status arm_conv_partial_opt_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst,
uint32_t firstIndex,
uint32_t numPoints,
q15_t * pScratch1,
q15_t * pScratch2);


/**
 * @brief Partial convolution of Q7 sequences.
 * @param[in]  pSrcA       points to the first input sequence.
 * @param[in]  srcALen     length of the first input sequence.
 * @param[in]  pSrcB       points to the second input sequence.
 * @param[in]  srcBLen     length of the second input sequence.
 * @param[out] pDst        points to the block of output data
 * @param[in]  firstIndex  is the first output sample to start with.
 * @param[in]  numPoints   is the number of output points to be computed.
 * @return  Returns either ARM_MATH_SUCCESS if the function completed correctly or ARM_MATH_AR
 */
arm_status arm_conv_partial_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst,
uint32_t firstIndex,
uint32_t numPoints);


/**
 * @brief Instance structure for the Q15 FIR decimator.
 */
typedef struct
{
  uint8_t M;                  /**< decimation factor. */
  uint16_t numTaps;           /**< number of coefficients in the filter. */
  q15_t *pCoeffs;             /**< points to the coefficient array. The array is of length numTaps.*/
  q15_t *pState;              /**< points to the state variable array. The array is of length numTaps+blockSize-
} arm_fir_decimate_instance_q15;

/**
 * @brief Instance structure for the Q31 FIR decimator.
 */
typedef struct
{
  uint8_t M;                  /**< decimation factor. */
  uint16_t numTaps;           /**< number of coefficients in the filter. */
  q31_t *pCoeffs;             /**< points to the coefficient array. The array is of length numTaps.*/
  q31_t *pState;              /**< points to the state variable array. The array is of length numTaps+blockSize-
} arm_fir_decimate_instance_q31;

/**
 * @brief Instance structure for the floating-point FIR decimator.
 */
typedef struct
{
  uint8_t M;                  /**< decimation factor. */
  uint16_t numTaps;           /**< number of coefficients in the filter. */
  float32_t *pCoeffs;         /**< points to the coefficient array. The array is of length numTaps.*/
  float32_t *pState;          /**< points to the state variable array. The array is of length numTaps+blockSize
} arm_fir_decimate_instance_f32;


/**
 * @brief Processing function for the floating-point FIR decimator.
 * @param[in]  S         points to an instance of the floating-point FIR decimator structure.
 * @param[in]  pSrc      points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_decimate_f32(
const arm_fir_decimate_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point FIR decimator.
 * @param[in,out] S          points to an instance of the floating-point FIR decimator structure.
 * @param[in]     numTaps    number of coefficients in the filter.
 * @param[in]     M          decimation factor.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return    The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_LE
 * <code>blockSize</code> is not a multiple of <code>M</code>.
 */
arm_status arm_fir_decimate_init_f32(
arm_fir_decimate_instance_f32 * S,
uint16_t numTaps,
uint8_t M,
float32_t * pCoeffs,
float32_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q15 FIR decimator.
 * @param[in]  S          points to an instance of the Q15 FIR decimator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_decimate_q15(
const arm_fir_decimate_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the Q15 FIR decimator (fast variant) for Cortex-M3 and Cortex-M4.
 * @param[in]  S          points to an instance of the Q15 FIR decimator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_decimate_fast_q15(
const arm_fir_decimate_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q15 FIR decimator.
 * @param[in,out] S          points to an instance of the Q15 FIR decimator structure.
 * @param[in]     numTaps    number of coefficients in the filter.
 * @param[in]     M          decimation factor.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return    The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_LE
 * <code>blockSize</code> is not a multiple of <code>M</code>.
 */
arm_status arm_fir_decimate_init_q15(
arm_fir_decimate_instance_q15 * S,
uint16_t numTaps,
uint8_t M,
q15_t * pCoeffs,
q15_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 FIR decimator.
 * @param[in]  S     points to an instance of the Q31 FIR decimator structure.
 * @param[in]  pSrc  points to the block of input data.
 * @param[out] pDst  points to the block of output data
 * @param[in] blockSize number of input samples to process per call.
 */
void arm_fir_decimate_q31(
const arm_fir_decimate_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);

/**
 * @brief Processing function for the Q31 FIR decimator (fast variant) for Cortex-M3 and Cortex-M4.
 * @param[in]  S          points to an instance of the Q31 FIR decimator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_decimate_fast_q31(
arm_fir_decimate_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
```

```
 * @brief  Initialization function for the Q31 FIR decimator.
 * @param[in,out] S          points to an instance of the Q31 FIR decimator structure.
 * @param[in]     numTaps    number of coefficients in the filter.
 * @param[in]     M          decimation factor.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return    The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_LE
 * <code>blockSize</code> is not a multiple of <code>M</code>.
 */
arm_status arm_fir_decimate_init_q31(
arm_fir_decimate_instance_q31 * S,
uint16_t numTaps,
uint8_t M,
q31_t * pCoeffs,
q31_t * pState,
uint32_t blockSize);


/**
 * @brief Instance structure for the Q15 FIR interpolator.
 */
typedef struct
{
  uint8_t L;                      /**< upsample factor. */
  uint16_t phaseLength;           /**< length of each polyphase filter component. */
  q15_t *pCoeffs;                 /**< points to the coefficient array. The array is of length L*phaseLength. */
  q15_t *pState;                  /**< points to the state variable array. The array is of length blockSize+phaseL
} arm_fir_interpolate_instance_q15;

/**
 * @brief Instance structure for the Q31 FIR interpolator.
 */
typedef struct
{
  uint8_t L;                      /**< upsample factor. */
  uint16_t phaseLength;           /**< length of each polyphase filter component. */
  q31_t *pCoeffs;                 /**< points to the coefficient array. The array is of length L*phaseLength. */
  q31_t *pState;                  /**< points to the state variable array. The array is of length blockSize+phaseL
} arm_fir_interpolate_instance_q31;

/**
 * @brief Instance structure for the floating-point FIR interpolator.
 */
typedef struct
{
  uint8_t L;                     /**< upsample factor. */
  uint16_t phaseLength;          /**< length of each polyphase filter component. */
  float32_t *pCoeffs;            /**< points to the coefficient array. The array is of length L*phaseLength. */
  float32_t *pState;             /**< points to the state variable array. The array is of length phaseLength+num
} arm_fir_interpolate_instance_f32;
```

```
/**
 * @brief Processing function for the Q15 FIR interpolator.
 * @param[in]  S          points to an instance of the Q15 FIR interpolator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_interpolate_q15(
const arm_fir_interpolate_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q15 FIR interpolator.
 * @param[in,out] S          points to an instance of the Q15 FIR interpolator structure.
 * @param[in]     L          upsample factor.
 * @param[in]     numTaps    number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficient buffer.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return        The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_
 * the filter length <code>numTaps</code> is not a multiple of the interpolation factor <code>L</code>.
 */
arm_status arm_fir_interpolate_init_q15(
arm_fir_interpolate_instance_q15 * S,
uint8_t L,
uint16_t numTaps,
q15_t * pCoeffs,
q15_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 FIR interpolator.
 * @param[in]  S          points to an instance of the Q15 FIR interpolator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_interpolate_q31(
const arm_fir_interpolate_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q31 FIR interpolator.
 * @param[in,out] S          points to an instance of the Q31 FIR interpolator structure.
 * @param[in]     L          upsample factor.
 * @param[in]     numTaps    number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficient buffer.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return        The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_
 * the filter length <code>numTaps</code> is not a multiple of the interpolation factor <code>L</code>.
 */
arm_status arm_fir_interpolate_init_q31(
arm_fir_interpolate_instance_q31 * S,
uint8_t L,
uint16_t numTaps,
q31_t * pCoeffs,
q31_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the floating-point FIR interpolator.
 * @param[in]  S          points to an instance of the floating-point FIR interpolator structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of input samples to process per call.
 */
void arm_fir_interpolate_f32(
const arm_fir_interpolate_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point FIR interpolator.
 * @param[in,out] S          points to an instance of the floating-point FIR interpolator structure.
 * @param[in]     L          upsample factor.
 * @param[in]     numTaps    number of filter coefficients in the filter.
 * @param[in]     pCoeffs    points to the filter coefficient buffer.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     blockSize  number of input samples to process per call.
 * @return        The function returns ARM_MATH_SUCCESS if initialization is successful or ARM_MATH_
 * the filter length <code>numTaps</code> is not a multiple of the interpolation factor <code>L</code>.
 */
arm_status arm_fir_interpolate_init_f32(
arm_fir_interpolate_instance_f32 * S,
uint8_t L,
uint16_t numTaps,
float32_t * pCoeffs,
float32_t * pState,
uint32_t blockSize);


/**
 * @brief Instance structure for the high precision Q31 Biquad cascade filter.
 */
typedef struct
```

```
{
  uint8_t numStages;       /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  q63_t *pState;           /**< points to the array of state coefficients.  The array is of length 4*numStages. */
  q31_t *pCoeffs;          /**< points to the array of coefficients.  The array is of length 5*numStages. */
  uint8_t postShift;       /**< additional shift, in bits, applied to each output sample. */
} arm_biquad_cas_df1_32x64_ins_q31;


/**
 * @param[in]  S          points to an instance of the high precision Q31 Biquad cascade filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cas_df1_32x64_q31(
const arm_biquad_cas_df1_32x64_ins_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @param[in,out] S          points to an instance of the high precision Q31 Biquad cascade filter structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     postShift  shift to be applied to the output. Varies according to the coefficients format
 */
void arm_biquad_cas_df1_32x64_init_q31(
arm_biquad_cas_df1_32x64_ins_q31 * S,
uint8_t numStages,
q31_t * pCoeffs,
q63_t * pState,
uint8_t postShift);


/**
 * @brief Instance structure for the floating-point transposed direct form II Biquad cascade filter.
 */
typedef struct
{
  uint8_t numStages;         /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  float32_t *pState;         /**< points to the array of state coefficients.  The array is of length 2*numStages.
  float32_t *pCoeffs;        /**< points to the array of coefficients.  The array is of length 5*numStages. */
} arm_biquad_cascade_df2T_instance_f32;

/**
 * @brief Instance structure for the floating-point transposed direct form II Biquad cascade filter.
 */
typedef struct
{
  uint8_t numStages;         /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  float32_t *pState;         /**< points to the array of state coefficients.  The array is of length 4*numStages.
  float32_t *pCoeffs;        /**< points to the array of coefficients.  The array is of length 5*numStages. */
} arm_biquad_cascade_stereo_df2T_instance_f32;

/**
 * @brief Instance structure for the floating-point transposed direct form II Biquad cascade filter.
 */
typedef struct
{
  uint8_t numStages;         /**< number of 2nd order stages in the filter.  Overall order is 2*numStages. */
  float64_t *pState;         /**< points to the array of state coefficients.  The array is of length 2*numStages.
  float64_t *pCoeffs;        /**< points to the array of coefficients.  The array is of length 5*numStages. */
} arm_biquad_cascade_df2T_instance_f64;


/**
 * @brief Processing function for the floating-point transposed direct form II Biquad cascade filter.
 * @param[in]  S          points to an instance of the filter data structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df2T_f32(
const arm_biquad_cascade_df2T_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the floating-point transposed direct form II Biquad cascade filter. 2 chann
 * @param[in]  S          points to an instance of the filter data structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_stereo_df2T_f32(
const arm_biquad_cascade_stereo_df2T_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Processing function for the floating-point transposed direct form II Biquad cascade filter.
 * @param[in]  S          points to an instance of the filter data structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data
 * @param[in]  blockSize  number of samples to process.
 */
void arm_biquad_cascade_df2T_f64(
const arm_biquad_cascade_df2T_instance_f64 * S,
float64_t * pSrc,
float64_t * pDst,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point transposed direct form II Biquad cascade filter.
 * @param[in,out] S          points to an instance of the filter data structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 */
void arm_biquad_cascade_df2T_init_f32(
arm_biquad_cascade_df2T_instance_f32 * S,
uint8_t numStages,
float32_t * pCoeffs,
float32_t * pState);


/**
 * @brief  Initialization function for the floating-point transposed direct form II Biquad cascade filter.
 * @param[in,out] S          points to an instance of the filter data structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 */
void arm_biquad_cascade_stereo_df2T_init_f32(
arm_biquad_cascade_stereo_df2T_instance_f32 * S,
uint8_t numStages,
float32_t * pCoeffs,
float32_t * pState);


/**
 * @brief  Initialization function for the floating-point transposed direct form II Biquad cascade filter.
 * @param[in,out] S          points to an instance of the filter data structure.
 * @param[in]     numStages  number of 2nd order stages in the filter.
 * @param[in]     pCoeffs    points to the filter coefficients.
 * @param[in]     pState     points to the state buffer.
 */
void arm_biquad_cascade_df2T_init_f64(
arm_biquad_cascade_df2T_instance_f64 * S,
uint8_t numStages,
float64_t * pCoeffs,
float64_t * pState);


/**
 * @brief Instance structure for the Q15 FIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of filter stages. */
  q15_t *pState;                       /**< points to the state variable array. The array is of length numStages. */
  q15_t *pCoeffs;                      /**< points to the coefficient array. The array is of length numStages. */
```

```
} arm_fir_lattice_instance_q15;

/**
 * @brief Instance structure for the Q31 FIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of filter stages. */
  q31_t *pState;                       /**< points to the state variable array. The array is of length numStages. */
  q31_t *pCoeffs;                      /**< points to the coefficient array. The array is of length numStages. */
} arm_fir_lattice_instance_q31;

/**
 * @brief Instance structure for the floating-point FIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of filter stages. */
  float32_t *pState;                   /**< points to the state variable array. The array is of length numStages. */
  float32_t *pCoeffs;                  /**< points to the coefficient array. The array is of length numStages. */
} arm_fir_lattice_instance_f32;


/**
 * @brief Initialization function for the Q15 FIR lattice filter.
 * @param[in] S          points to an instance of the Q15 FIR lattice structure.
 * @param[in] numStages  number of filter stages.
 * @param[in] pCoeffs    points to the coefficient buffer.  The array is of length numStages.
 * @param[in] pState     points to the state buffer.  The array is of length numStages.
 */
void arm_fir_lattice_init_q15(
arm_fir_lattice_instance_q15 * S,
uint16_t numStages,
q15_t * pCoeffs,
q15_t * pState);


/**
 * @brief Processing function for the Q15 FIR lattice filter.
 * @param[in]  S          points to an instance of the Q15 FIR lattice structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_fir_lattice_q15(
const arm_fir_lattice_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Initialization function for the Q31 FIR lattice filter.
```

```
   * @param[in] S          points to an instance of the Q31 FIR lattice structure.
   * @param[in] numStages  number of filter stages.
   * @param[in] pCoeffs    points to the coefficient buffer.  The array is of length numStages.
   * @param[in] pState     points to the state buffer.   The array is of length numStages.
   */
  void arm_fir_lattice_init_q31(
  arm_fir_lattice_instance_q31 * S,
  uint16_t numStages,
  q31_t * pCoeffs,
  q31_t * pState);


  /**
   * @brief Processing function for the Q31 FIR lattice filter.
   * @param[in]  S          points to an instance of the Q31 FIR lattice structure.
   * @param[in]  pSrc       points to the block of input data.
   * @param[out] pDst       points to the block of output data
   * @param[in]  blockSize  number of samples to process.
   */
  void arm_fir_lattice_q31(
  const arm_fir_lattice_instance_q31 * S,
  q31_t * pSrc,
  q31_t * pDst,
  uint32_t blockSize);


/**
 * @brief Initialization function for the floating-point FIR lattice filter.
 * @param[in] S          points to an instance of the floating-point FIR lattice structure.
 * @param[in] numStages  number of filter stages.
 * @param[in] pCoeffs    points to the coefficient buffer.  The array is of length numStages.
 * @param[in] pState     points to the state buffer.  The array is of length numStages.
 */
  void arm_fir_lattice_init_f32(
  arm_fir_lattice_instance_f32 * S,
  uint16_t numStages,
  float32_t * pCoeffs,
  float32_t * pState);


  /**
   * @brief Processing function for the floating-point FIR lattice filter.
   * @param[in]  S          points to an instance of the floating-point FIR lattice structure.
   * @param[in]  pSrc       points to the block of input data.
   * @param[out] pDst       points to the block of output data
   * @param[in]  blockSize  number of samples to process.
   */
  void arm_fir_lattice_f32(
  const arm_fir_lattice_instance_f32 * S,
  float32_t * pSrc,
  float32_t * pDst,
  uint32_t blockSize);
```

```
/**
 * @brief Instance structure for the Q15 IIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of stages in the filter. */
  q15_t *pState;                      /**< points to the state variable array. The array is of length numStages+blo
  q15_t *pkCoeffs;                     /**< points to the reflection coefficient array. The array is of length numSta
  q15_t *pvCoeffs;                     /**< points to the ladder coefficient array. The array is of length numStages
} arm_iir_lattice_instance_q15;

/**
 * @brief Instance structure for the Q31 IIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of stages in the filter. */
  q31_t *pState;                      /**< points to the state variable array. The array is of length numStages+blo
  q31_t *pkCoeffs;                     /**< points to the reflection coefficient array. The array is of length numSta
  q31_t *pvCoeffs;                     /**< points to the ladder coefficient array. The array is of length numStages
} arm_iir_lattice_instance_q31;

/**
 * @brief Instance structure for the floating-point IIR lattice filter.
 */
typedef struct
{
  uint16_t numStages;                  /**< number of stages in the filter. */
  float32_t *pState;                  /**< points to the state variable array. The array is of length numStages+blo
  float32_t *pkCoeffs;                 /**< points to the reflection coefficient array. The array is of length numSta
  float32_t *pvCoeffs;                 /**< points to the ladder coefficient array. The array is of length numStage
} arm_iir_lattice_instance_f32;


/**
 * @brief Processing function for the floating-point IIR lattice filter.
 * @param[in]  S          points to an instance of the floating-point IIR lattice structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_iir_lattice_f32(
const arm_iir_lattice_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief Initialization function for the floating-point IIR lattice filter.
 * @param[in] S          points to an instance of the floating-point IIR lattice structure.
 * @param[in] numStages  number of stages in the filter.
 * @param[in] pkCoeffs   points to the reflection coefficient buffer.  The array is of length numStages.
 * @param[in] pvCoeffs   points to the ladder coefficient buffer.  The array is of length numStages+1.
 * @param[in] pState     points to the state buffer.  The array is of length numStages+blockSize-1.
 * @param[in] blockSize  number of samples to process.
 */
void arm_iir_lattice_init_f32(
arm_iir_lattice_instance_f32 * S,
uint16_t numStages,
float32_t * pkCoeffs,
float32_t * pvCoeffs,
float32_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 IIR lattice filter.
 * @param[in]  S          points to an instance of the Q31 IIR lattice structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_iir_lattice_q31(
const arm_iir_lattice_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Initialization function for the Q31 IIR lattice filter.
 * @param[in] S          points to an instance of the Q31 IIR lattice structure.
 * @param[in] numStages  number of stages in the filter.
 * @param[in] pkCoeffs   points to the reflection coefficient buffer.  The array is of length numStages.
 * @param[in] pvCoeffs   points to the ladder coefficient buffer.  The array is of length numStages+1.
 * @param[in] pState     points to the state buffer.  The array is of length numStages+blockSize.
 * @param[in] blockSize  number of samples to process.
 */
void arm_iir_lattice_init_q31(
arm_iir_lattice_instance_q31 * S,
uint16_t numStages,
q31_t * pkCoeffs,
q31_t * pvCoeffs,
q31_t * pState,
uint32_t blockSize);


/**
 * @brief Processing function for the Q15 IIR lattice filter.
 * @param[in]  S          points to an instance of the Q15 IIR lattice structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[out] pDst       points to the block of output data.
 * @param[in]  blockSize  number of samples to process.
 */
```

```
  void arm_iir_lattice_q15(
  const arm_iir_lattice_instance_q15 * S,
  q15_t * pSrc,
  q15_t * pDst,
  uint32_t blockSize);


/**
 * @brief Initialization function for the Q15 IIR lattice filter.
 * @param[in] S          points to an instance of the fixed-point Q15 IIR lattice structure.
 * @param[in] numStages  number of stages in the filter.
 * @param[in] pkCoeffs   points to reflection coefficient buffer.  The array is of length numStages.
 * @param[in] pvCoeffs   points to ladder coefficient buffer.  The array is of length numStages+1.
 * @param[in] pState     points to state buffer.  The array is of length numStages+blockSize.
 * @param[in] blockSize  number of samples to process per call.
 */
  void arm_iir_lattice_init_q15(
  arm_iir_lattice_instance_q15 * S,
  uint16_t numStages,
  q15_t * pkCoeffs,
  q15_t * pvCoeffs,
  q15_t * pState,
  uint32_t blockSize);


  /**
   * @brief Instance structure for the floating-point LMS filter.
   */
  typedef struct
  {
    uint16_t numTaps;    /**< number of coefficients in the filter. */
    float32_t *pState;   /**< points to the state variable array. The array is of length numTaps+blockSize-1. */
    float32_t *pCoeffs;  /**< points to the coefficient array. The array is of length numTaps. */
    float32_t mu;        /**< step size that controls filter coefficient updates. */
  } arm_lms_instance_f32;


  /**
   * @brief Processing function for floating-point LMS filter.
   * @param[in]  S          points to an instance of the floating-point LMS filter structure.
   * @param[in]  pSrc       points to the block of input data.
   * @param[in]  pRef       points to the block of reference data.
   * @param[out] pOut       points to the block of output data.
   * @param[out] pErr       points to the block of error data.
   * @param[in]  blockSize  number of samples to process.
   */
  void arm_lms_f32(
  const arm_lms_instance_f32 * S,
  float32_t * pSrc,
  float32_t * pRef,
  float32_t * pOut,
  float32_t * pErr,
  uint32_t blockSize);
```

```
/**
 * @brief Initialization function for floating-point LMS filter.
 * @param[in] S          points to an instance of the floating-point LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to the coefficient buffer.
 * @param[in] pState     points to state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 */
void arm_lms_init_f32(
arm_lms_instance_f32 * S,
uint16_t numTaps,
float32_t * pCoeffs,
float32_t * pState,
float32_t mu,
uint32_t blockSize);


/**
 * @brief Instance structure for the Q15 LMS filter.
 */
typedef struct
{
  uint16_t numTaps;    /**< number of coefficients in the filter. */
  q15_t *pState;       /**< points to the state variable array. The array is of length numTaps+blockSize-1. */
  q15_t *pCoeffs;      /**< points to the coefficient array. The array is of length numTaps. */
  q15_t mu;            /**< step size that controls filter coefficient updates. */
  uint32_t postShift;  /**< bit shift applied to coefficients. */
} arm_lms_instance_q15;


/**
 * @brief Initialization function for the Q15 LMS filter.
 * @param[in] S          points to an instance of the Q15 LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to the coefficient buffer.
 * @param[in] pState     points to the state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 * @param[in] postShift  bit shift applied to coefficients.
 */
void arm_lms_init_q15(
arm_lms_instance_q15 * S,
uint16_t numTaps,
q15_t * pCoeffs,
q15_t * pState,
q15_t mu,
uint32_t blockSize,
uint32_t postShift);
```

```
/**
 * @brief Processing function for Q15 LMS filter.
 * @param[in]  S         points to an instance of the Q15 LMS filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[in]  pRef       points to the block of reference data.
 * @param[out] pOut       points to the block of output data.
 * @param[out] pErr       points to the block of error data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_lms_q15(
const arm_lms_instance_q15 * S,
q15_t * pSrc,
q15_t * pRef,
q15_t * pOut,
q15_t * pErr,
uint32_t blockSize);


/**
 * @brief Instance structure for the Q31 LMS filter.
 */
typedef struct
{
  uint16_t numTaps;    /**< number of coefficients in the filter. */
  q31_t *pState;       /**< points to the state variable array. The array is of length numTaps+blockSize-1. */
  q31_t *pCoeffs;      /**< points to the coefficient array. The array is of length numTaps. */
  q31_t mu;            /**< step size that controls filter coefficient updates. */
  uint32_t postShift;  /**< bit shift applied to coefficients. */
} arm_lms_instance_q31;


/**
 * @brief Processing function for Q31 LMS filter.
 * @param[in]  S         points to an instance of the Q15 LMS filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[in]  pRef       points to the block of reference data.
 * @param[out] pOut       points to the block of output data.
 * @param[out] pErr       points to the block of error data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_lms_q31(
const arm_lms_instance_q31 * S,
q31_t * pSrc,
q31_t * pRef,
q31_t * pOut,
q31_t * pErr,
uint32_t blockSize);


/**
 * @brief Initialization function for Q31 LMS filter.
 * @param[in] S         points to an instance of the Q31 LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to coefficient buffer.
 * @param[in] pState     points to state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 * @param[in] postShift  bit shift applied to coefficients.
 */
void arm_lms_init_q31(
arm_lms_instance_q31 * S,
uint16_t numTaps,
q31_t * pCoeffs,
q31_t * pState,
q31_t mu,
uint32_t blockSize,
uint32_t postShift);


/**
 * @brief Instance structure for the floating-point normalized LMS filter.
 */
typedef struct
{
  uint16_t numTaps;     /**< number of coefficients in the filter. */
  float32_t *pState;    /**< points to the state variable array. The array is of length numTaps+blockSize-1. *
  float32_t *pCoeffs;   /**< points to the coefficient array. The array is of length numTaps. */
  float32_t mu;         /**< step size that control filter coefficient updates. */
  float32_t energy;     /**< saves previous frame energy. */
  float32_t x0;         /**< saves previous input sample. */
} arm_lms_norm_instance_f32;


/**
 * @brief Processing function for floating-point normalized LMS filter.
 * @param[in]  S          points to an instance of the floating-point normalized LMS filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[in]  pRef       points to the block of reference data.
 * @param[out] pOut       points to the block of output data.
 * @param[out] pErr       points to the block of error data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_lms_norm_f32(
arm_lms_norm_instance_f32 * S,
float32_t * pSrc,
float32_t * pRef,
float32_t * pOut,
float32_t * pErr,
uint32_t blockSize);


/**
 * @brief Initialization function for floating-point normalized LMS filter.
 * @param[in] S          points to an instance of the floating-point LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to coefficient buffer.
 * @param[in] pState     points to state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 */
void arm_lms_norm_init_f32(
arm_lms_norm_instance_f32 * S,
uint16_t numTaps,
float32_t * pCoeffs,
float32_t * pState,
float32_t mu,
uint32_t blockSize);


/**
 * @brief Instance structure for the Q31 normalized LMS filter.
 */
typedef struct
{
  uint16_t numTaps;     /**< number of coefficients in the filter. */
  q31_t *pState;        /**< points to the state variable array. The array is of length numTaps+blockSize-1. */
  q31_t *pCoeffs;       /**< points to the coefficient array. The array is of length numTaps. */
  q31_t mu;             /**< step size that controls filter coefficient updates. */
  uint8_t postShift;    /**< bit shift applied to coefficients. */
  q31_t *recipTable;    /**< points to the reciprocal initial value table. */
  q31_t energy;         /**< saves previous frame energy. */
  q31_t x0;             /**< saves previous input sample. */
} arm_lms_norm_instance_q31;


/**
 * @brief Processing function for Q31 normalized LMS filter.
 * @param[in]  S         points to an instance of the Q31 normalized LMS filter structure.
 * @param[in]  pSrc      points to the block of input data.
 * @param[in]  pRef      points to the block of reference data.
 * @param[out] pOut      points to the block of output data.
 * @param[out] pErr      points to the block of error data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_lms_norm_q31(
arm_lms_norm_instance_q31 * S,
q31_t * pSrc,
q31_t * pRef,
q31_t * pOut,
q31_t * pErr,
uint32_t blockSize);


/**
 * @brief Initialization function for Q31 normalized LMS filter.
 * @param[in] S         points to an instance of the Q31 normalized LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to coefficient buffer.
 * @param[in] pState     points to state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 * @param[in] postShift  bit shift applied to coefficients.
 */
void arm_lms_norm_init_q31(
arm_lms_norm_instance_q31 * S,
uint16_t numTaps,
q31_t * pCoeffs,
q31_t * pState,
q31_t mu,
uint32_t blockSize,
uint8_t postShift);


/**
 * @brief Instance structure for the Q15 normalized LMS filter.
 */
typedef struct
{
  uint16_t numTaps;     /**< Number of coefficients in the filter. */
  q15_t *pState;        /**< points to the state variable array. The array is of length numTaps+blockSize-1. *
  q15_t *pCoeffs;       /**< points to the coefficient array. The array is of length numTaps. */
  q15_t mu;             /**< step size that controls filter coefficient updates. */
  uint8_t postShift;    /**< bit shift applied to coefficients. */
  q15_t *recipTable;    /**< Points to the reciprocal initial value table. */
  q15_t energy;         /**< saves previous frame energy. */
  q15_t x0;             /**< saves previous input sample. */
} arm_lms_norm_instance_q15;


/**
 * @brief Processing function for Q15 normalized LMS filter.
 * @param[in]  S         points to an instance of the Q15 normalized LMS filter structure.
 * @param[in]  pSrc       points to the block of input data.
 * @param[in]  pRef       points to the block of reference data.
 * @param[out] pOut       points to the block of output data.
 * @param[out] pErr       points to the block of error data.
 * @param[in]  blockSize  number of samples to process.
 */
void arm_lms_norm_q15(
arm_lms_norm_instance_q15 * S,
q15_t * pSrc,
q15_t * pRef,
q15_t * pOut,
q15_t * pErr,
uint32_t blockSize);


/**
 * @brief Initialization function for Q15 normalized LMS filter.
 * @param[in] S         points to an instance of the Q15 normalized LMS filter structure.
 * @param[in] numTaps    number of filter coefficients.
 * @param[in] pCoeffs    points to coefficient buffer.
 * @param[in] pState     points to state buffer.
 * @param[in] mu         step size that controls filter coefficient updates.
 * @param[in] blockSize  number of samples to process.
 * @param[in] postShift  bit shift applied to coefficients.
 */
void arm_lms_norm_init_q15(
arm_lms_norm_instance_q15 * S,
uint16_t numTaps,
q15_t * pCoeffs,
q15_t * pState,
q15_t mu,
uint32_t blockSize,
uint8_t postShift);


/**
 * @brief Correlation of floating-point sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */
void arm_correlate_f32(
float32_t * pSrcA,
uint32_t srcALen,
float32_t * pSrcB,
uint32_t srcBLen,
float32_t * pDst);


 /**
 * @brief Correlation of Q15 sequences
 * @param[in]  pSrcA     points to the first input sequence.
 * @param[in]  srcALen   length of the first input sequence.
 * @param[in]  pSrcB     points to the second input sequence.
 * @param[in]  srcBLen   length of the second input sequence.
 * @param[out] pDst      points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 * @param[in]  pScratch  points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBLe
 */
void arm_correlate_opt_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
q15_t * pScratch);


/**
 * @brief Correlation of Q15 sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */

void arm_correlate_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst);


/**
 * @brief Correlation of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */

void arm_correlate_fast_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst);


/**
 * @brief Correlation of Q15 sequences (fast version) for Cortex-M3 and Cortex-M4.
 * @param[in]  pSrcA     points to the first input sequence.
 * @param[in]  srcALen   length of the first input sequence.
 * @param[in]  pSrcB     points to the second input sequence.
 * @param[in]  srcBLen   length of the second input sequence.
 * @param[out] pDst      points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 * @param[in]  pScratch  points to scratch buffer of size max(srcALen, srcBLen) + 2*min(srcALen, srcBLe
 */
void arm_correlate_fast_opt_q15(
q15_t * pSrcA,
uint32_t srcALen,
q15_t * pSrcB,
uint32_t srcBLen,
q15_t * pDst,
q15_t * pScratch);


/**
 * @brief Correlation of Q31 sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */
void arm_correlate_q31(
q31_t * pSrcA,
uint32_t srcALen,
q31_t * pSrcB,
uint32_t srcBLen,
q31_t * pDst);


/**
 * @brief Correlation of Q31 sequences (fast version) for Cortex-M3 and Cortex-M4
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */
void arm_correlate_fast_q31(
q31_t * pSrcA,
uint32_t srcALen,
q31_t * pSrcB,
uint32_t srcBLen,
q31_t * pDst);


/**
 * @brief Correlation of Q7 sequences.
 * @param[in]  pSrcA      points to the first input sequence.
 * @param[in]  srcALen    length of the first input sequence.
 * @param[in]  pSrcB      points to the second input sequence.
 * @param[in]  srcBLen    length of the second input sequence.
 * @param[out] pDst       points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 * @param[in]  pScratch1  points to scratch buffer(of type q15_t) of size max(srcALen, srcBLen) + 2*min(s
 * @param[in]  pScratch2  points to scratch buffer (of type q15_t) of size min(srcALen, srcBLen).
 */
void arm_correlate_opt_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst,
q15_t * pScratch1,
q15_t * pScratch2);


/**
 * @brief Correlation of Q7 sequences.
 * @param[in]  pSrcA    points to the first input sequence.
 * @param[in]  srcALen  length of the first input sequence.
 * @param[in]  pSrcB    points to the second input sequence.
 * @param[in]  srcBLen  length of the second input sequence.
 * @param[out] pDst     points to the block of output data  Length 2 * max(srcALen, srcBLen) - 1.
 */
void arm_correlate_q7(
q7_t * pSrcA,
uint32_t srcALen,
q7_t * pSrcB,
uint32_t srcBLen,
q7_t * pDst);



/**
 * @brief Instance structure for the floating-point sparse FIR filter.
 */
typedef struct
{
  uint16_t numTaps;             /**< number of coefficients in the filter. */
  uint16_t stateIndex;          /**< state buffer index.  Points to the oldest sample in the state buffer. */
  float32_t *pState;            /**< points to the state buffer array. The array is of length maxDelay+blockSize-
  float32_t *pCoeffs;           /**< points to the coefficient array. The array is of length numTaps.*/
  uint16_t maxDelay;            /**< maximum offset specified by the pTapDelay array. */
  int32_t *pTapDelay;           /**< points to the array of delay values.  The array is of length numTaps. */
} arm_fir_sparse_instance_f32;

/**
 * @brief Instance structure for the Q31 sparse FIR filter.
 */
typedef struct
{
  uint16_t numTaps;             /**< number of coefficients in the filter. */
  uint16_t stateIndex;          /**< state buffer index.  Points to the oldest sample in the state buffer. */
  q31_t *pState;                /**< points to the state buffer array. The array is of length maxDelay+blockSize-
  q31_t *pCoeffs;               /**< points to the coefficient array. The array is of length numTaps.*/
  uint16_t maxDelay;            /**< maximum offset specified by the pTapDelay array. */
  int32_t *pTapDelay;           /**< points to the array of delay values.  The array is of length numTaps. */
} arm_fir_sparse_instance_q31;

/**
 * @brief Instance structure for the Q15 sparse FIR filter.
 */
typedef struct
{
  uint16_t numTaps;             /**< number of coefficients in the filter. */
  uint16_t stateIndex;          /**< state buffer index.  Points to the oldest sample in the state buffer. */
  q15_t *pState;                /**< points to the state buffer array. The array is of length maxDelay+blockSize-
  q15_t *pCoeffs;               /**< points to the coefficient array. The array is of length numTaps.*/
  uint16_t maxDelay;            /**< maximum offset specified by the pTapDelay array. */
  int32_t *pTapDelay;           /**< points to the array of delay values.  The array is of length numTaps. */
} arm_fir_sparse_instance_q15;

/**
 * @brief Instance structure for the Q7 sparse FIR filter.
 */
typedef struct
```

```
{
  uint16_t numTaps;             /**< number of coefficients in the filter. */
  uint16_t stateIndex;          /**< state buffer index.  Points to the oldest sample in the state buffer. */
  q7_t *pState;                 /**< points to the state buffer array. The array is of length maxDelay+blockSize-1
  q7_t *pCoeffs;                /**< points to the coefficient array. The array is of length numTaps.*/
  uint16_t maxDelay;            /**< maximum offset specified by the pTapDelay array. */
  int32_t *pTapDelay;           /**< points to the array of delay values.  The array is of length numTaps. */
} arm_fir_sparse_instance_q7;


/**
 * @brief Processing function for the floating-point sparse FIR filter.
 * @param[in]  S          points to an instance of the floating-point sparse FIR structure.
 * @param[in]  pSrc        points to the block of input data.
 * @param[out] pDst        points to the block of output data
 * @param[in]  pScratchIn  points to a temporary buffer of size blockSize.
 * @param[in]  blockSize   number of input samples to process per call.
 */
void arm_fir_sparse_f32(
arm_fir_sparse_instance_f32 * S,
float32_t * pSrc,
float32_t * pDst,
float32_t * pScratchIn,
uint32_t blockSize);


/**
 * @brief  Initialization function for the floating-point sparse FIR filter.
 * @param[in,out] S          points to an instance of the floating-point sparse FIR structure.
 * @param[in]     numTaps    number of nonzero coefficients in the filter.
 * @param[in]     pCoeffs    points to the array of filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     pTapDelay  points to the array of offset times.
 * @param[in]     maxDelay   maximum offset time supported.
 * @param[in]     blockSize  number of samples that will be processed per block.
 */
void arm_fir_sparse_init_f32(
arm_fir_sparse_instance_f32 * S,
uint16_t numTaps,
float32_t * pCoeffs,
float32_t * pState,
int32_t * pTapDelay,
uint16_t maxDelay,
uint32_t blockSize);


/**
 * @brief Processing function for the Q31 sparse FIR filter.
 * @param[in]  S          points to an instance of the Q31 sparse FIR structure.
 * @param[in]  pSrc        points to the block of input data.
 * @param[out] pDst        points to the block of output data
 * @param[in]  pScratchIn  points to a temporary buffer of size blockSize.
 * @param[in]  blockSize   number of input samples to process per call.
 */
void arm_fir_sparse_q31(
arm_fir_sparse_instance_q31 * S,
q31_t * pSrc,
q31_t * pDst,
q31_t * pScratchIn,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q31 sparse FIR filter.
 * @param[in,out] S         points to an instance of the Q31 sparse FIR structure.
 * @param[in]     numTaps    number of nonzero coefficients in the filter.
 * @param[in]     pCoeffs    points to the array of filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     pTapDelay  points to the array of offset times.
 * @param[in]     maxDelay   maximum offset time supported.
 * @param[in]     blockSize  number of samples that will be processed per block.
 */
void arm_fir_sparse_init_q31(
arm_fir_sparse_instance_q31 * S,
uint16_t numTaps,
q31_t * pCoeffs,
q31_t * pState,
int32_t * pTapDelay,
uint16_t maxDelay,
uint32_t blockSize);


/**
 * @brief Processing function for the Q15 sparse FIR filter.
 * @param[in]  S           points to an instance of the Q15 sparse FIR structure.
 * @param[in]  pSrc        points to the block of input data.
 * @param[out] pDst        points to the block of output data
 * @param[in]  pScratchIn   points to a temporary buffer of size blockSize.
 * @param[in]  pScratchOut  points to a temporary buffer of size blockSize.
 * @param[in]  blockSize    number of input samples to process per call.
 */
void arm_fir_sparse_q15(
arm_fir_sparse_instance_q15 * S,
q15_t * pSrc,
q15_t * pDst,
q15_t * pScratchIn,
q31_t * pScratchOut,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q15 sparse FIR filter.
 * @param[in,out] S         points to an instance of the Q15 sparse FIR structure.
 * @param[in]     numTaps    number of nonzero coefficients in the filter.
 * @param[in]     pCoeffs    points to the array of filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     pTapDelay  points to the array of offset times.
 * @param[in]     maxDelay   maximum offset time supported.
 * @param[in]     blockSize  number of samples that will be processed per block.
 */
void arm_fir_sparse_init_q15(
arm_fir_sparse_instance_q15 * S,
uint16_t numTaps,
q15_t * pCoeffs,
q15_t * pState,
int32_t * pTapDelay,
uint16_t maxDelay,
uint32_t blockSize);


/**
 * @brief Processing function for the Q7 sparse FIR filter.
 * @param[in]  S           points to an instance of the Q7 sparse FIR structure.
 * @param[in]  pSrc         points to the block of input data.
 * @param[out] pDst         points to the block of output data
 * @param[in]  pScratchIn   points to a temporary buffer of size blockSize.
 * @param[in]  pScratchOut  points to a temporary buffer of size blockSize.
 * @param[in]  blockSize    number of input samples to process per call.
 */
void arm_fir_sparse_q7(
arm_fir_sparse_instance_q7 * S,
q7_t * pSrc,
q7_t * pDst,
q7_t * pScratchIn,
q31_t * pScratchOut,
uint32_t blockSize);


/**
 * @brief  Initialization function for the Q7 sparse FIR filter.
 * @param[in,out] S          points to an instance of the Q7 sparse FIR structure.
 * @param[in]     numTaps    number of nonzero coefficients in the filter.
 * @param[in]     pCoeffs    points to the array of filter coefficients.
 * @param[in]     pState     points to the state buffer.
 * @param[in]     pTapDelay  points to the array of offset times.
 * @param[in]     maxDelay   maximum offset time supported.
 * @param[in]     blockSize  number of samples that will be processed per block.
 */
void arm_fir_sparse_init_q7(
arm_fir_sparse_instance_q7 * S,
uint16_t numTaps,
q7_t * pCoeffs,
q7_t * pState,
int32_t * pTapDelay,
uint16_t maxDelay,
uint32_t blockSize);


/**
```

```
 * @brief  Floating-point sin_cos function.
 * @param[in]  theta   input value in degrees
 * @param[out] pSinVal  points to the processed sine output.
 * @param[out] pCosVal  points to the processed cos output.
 */
void arm_sin_cos_f32(
float32_t theta,
float32_t * pSinVal,
float32_t * pCosVal);


/**
 * @brief  Q31 sin_cos function.
 * @param[in]  theta    scaled input value in degrees
 * @param[out] pSinVal  points to the processed sine output.
 * @param[out] pCosVal  points to the processed cosine output.
 */
void arm_sin_cos_q31(
q31_t theta,
q31_t * pSinVal,
q31_t * pCosVal);


/**
 * @brief  Floating-point complex conjugate.
 * @param[in]  pSrc        points to the input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_conj_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t numSamples);

/**
 * @brief  Q31 complex conjugate.
 * @param[in]  pSrc        points to the input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_conj_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q15 complex conjugate.
 * @param[in]  pSrc        points to the input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_conj_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t numSamples);


/**
 * @brief  Floating-point complex magnitude squared
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_squared_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q31 complex magnitude squared
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_squared_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q15 complex magnitude squared
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_squared_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t numSamples);


/**
 * @ingroup groupController
 */

/**
 * @defgroup PID PID Motor Control
 *
 * A Proportional Integral Derivative (PID) controller is a generic feedback control
 * loop mechanism widely used in industrial control systems.
 * A PID controller is the most commonly used type of feedback controller.
 *
 * This set of functions implements (PID) controllers
 * for Q15, Q31, and floating-point data types.  The functions operate on a single sample
 * of data and each call to the function returns a single processed value.
 * <code>S</code> points to an instance of the PID control data structure.  <code>in</code>
 * is the input sample value. The functions return the output value.
 *
 * \par Algorithm:
 * <pre>
 *    y[n] = y[n-1] + A0 * x[n] + A1 * x[n-1] + A2 * x[n-2]
 *    A0 = Kp + Ki + Kd
 *    A1 = (-Kp ) - (2 * Kd )
 *    A2 = Kd  </pre>
 *
 * \par
 * where \c Kp is proportional constant, \c Ki is Integral constant and \c Kd is Derivative constant
 *
 * \par
 * \image html PID.gif "Proportional Integral Derivative Controller"
 *
 * \par
 * The PID controller calculates an "error" value as the difference between
 * the measured output and the reference input.
 * The controller attempts to minimize the error by adjusting the process control inputs.
 * The proportional value determines the reaction to the current error,
 * the integral value determines the reaction based on the sum of recent errors,
 * and the derivative value determines the reaction based on the rate at which the error has been changin
 *
 * \par Instance Structure
 * The Gains A0, A1, A2 and state variables for a PID controller are stored together in an instance data st
 * A separate instance structure must be defined for each PID Controller.
 * There are separate instance structure declarations for each of the 3 supported data types.
 *
 * \par Reset Functions
 * There is also an associated reset function for each data type which clears the state array.
 *
 * \par Initialization Functions
 * There is also an associated initialization function for each data type.
 * The initialization function performs the following operations:
 * - Initializes the Gains A0, A1, A2 from Kp,Ki, Kd gains.
 * - Zeros out the values in the state buffer.
 *
 * \par
 * Instance structure cannot be placed into a const data section and it is recommended to use the initializa
 *
 * \par Fixed-Point Behavior
 * Care must be taken when using the fixed-point versions of the PID Controller functions.
 * In particular, the overflow and saturation behavior of the accumulator used in each function must be co
 * Refer to the function specific documentation below for usage guidelines.
 */

/**
 * @addtogroup PID
 * @{
 */
```

```
/**
 * @brief  Process function for the floating-point PID Control.
 * @param[in,out] S   is an instance of the floating-point PID Control structure
 * @param[in]     in  input sample to process
 * @return out processed output sample.
 */
static __INLINE float32_t arm_pid_f32(
arm_pid_instance_f32 * S,
float32_t in)
{
  float32_t out;

  /* y[n] = y[n-1] + A0 * x[n] + A1 * x[n-1] + A2 * x[n-2]  */
  out = (S->A0 * in) +
    (S->A1 * S->state[0]) + (S->A2 * S->state[1]) + (S->state[2]);

  /* Update state */
  S->state[1] = S->state[0];
  S->state[0] = in;
  S->state[2] = out;

  /* return to application */
  return (out);

}

/**
 * @brief  Process function for the Q31 PID Control.
 * @param[in,out] S  points to an instance of the Q31 PID Control structure
 * @param[in]     in  input sample to process
 * @return out processed output sample.
 *
 * <b>Scaling and Overflow Behavior:</b>
 * \par
 * The function is implemented using an internal 64-bit accumulator.
 * The accumulator has a 2.62 format and maintains full precision of the intermediate multiplication results
 * Thus, if the accumulator result overflows it wraps around rather than clip.
 * In order to avoid overflows completely the input signal must be scaled down by 2 bits as there are four a
 * After all multiply-accumulates are performed, the 2.62 accumulator is truncated to 1.32 format and then
 */
static __INLINE q31_t arm_pid_q31(
arm_pid_instance_q31 * S,
q31_t in)
{
  q63_t acc;
  q31_t out;

  /* acc = A0 * x[n]  */
  acc = (q63_t) S->A0 * in;

  /* acc += A1 * x[n-1] */
  acc += (q63_t) S->A1 * S->state[0];
```

```
    /* acc += A2 * x[n-2]  */
    acc += (q63_t) S->A2 * S->state[1];

    /* convert output to 1.31 format to add y[n-1] */
    out = (q31_t) (acc >> 31u);

    /* out += y[n-1] */
    out += S->state[2];

    /* Update state */
    S->state[1] = S->state[0];
    S->state[0] = in;
    S->state[2] = out;

    /* return to application */
    return (out);
  }


  /**
   * @brief  Process function for the Q15 PID Control.
   * @param[in,out] S   points to an instance of the Q15 PID Control structure
   * @param[in]     in  input sample to process
   * @return out processed output sample.
   *
   * <b>Scaling and Overflow Behavior:</b>
   * \par
   * The function is implemented using a 64-bit internal accumulator.
   * Both Gains and state variables are represented in 1.15 format and multiplications yield a 2.30 result.
   * The 2.30 intermediate results are accumulated in a 64-bit accumulator in 34.30 format.
   * There is no risk of internal overflow with this approach and the full precision of intermediate multiplicatio
   * After all additions have been performed, the accumulator is truncated to 34.15 format by discarding low
   * Lastly, the accumulator is saturated to yield a result in 1.15 format.
   */
  static __INLINE q15_t arm_pid_q15(
  arm_pid_instance_q15 * S,
  q15_t in)
  {
    q63_t acc;
    q15_t out;

#ifndef ARM_MATH_CM0_FAMILY
    __SIMD32_TYPE *vstate;

    /* Implementation of PID controller */

    /* acc = A0 * x[n]  */
    acc = (q31_t) __SMUAD((uint32_t)S->A0, (uint32_t)in);

    /* acc += A1 * x[n-1] + A2 * x[n-2]  */
    vstate = __SIMD32_CONST(S->state);
    acc = (q63_t)__SMLALD((uint32_t)S->A1, (uint32_t)*vstate, (uint64_t)acc);
```

```
#else
    /* acc = A0 * x[n]  */
    acc = ((q31_t) S->A0) * in;

    /* acc += A1 * x[n-1] + A2 * x[n-2]  */
    acc += (q31_t) S->A1 * S->state[0];
    acc += (q31_t) S->A2 * S->state[1];
#endif

    /* acc += y[n-1] */
    acc += (q31_t) S->state[2] << 15;

    /* saturate the output */
    out = (q15_t) (__SSAT((acc >> 15), 16));

    /* Update state */
    S->state[1] = S->state[0];
    S->state[0] = in;
    S->state[2] = out;

    /* return to application */
    return (out);
  }

  /**
   * @} end of PID group
   */



  /**
   * @brief Floating-point matrix inverse.
   * @param[in]  src   points to the instance of the input floating-point matrix structure.
   * @param[out] dst   points to the instance of the output floating-point matrix structure.
   * @return The function returns ARM_MATH_SIZE_MISMATCH, if the dimensions do not match.
   * If the input matrix is singular (does not have an inverse), then the algorithm terminates and returns erro
   */
  arm_status arm_mat_inverse_f32(
  const arm_matrix_instance_f32 * src,
  arm_matrix_instance_f32 * dst);



  /**
   * @brief Floating-point matrix inverse.
   * @param[in]  src   points to the instance of the input floating-point matrix structure.
   * @param[out] dst   points to the instance of the output floating-point matrix structure.
   * @return The function returns ARM_MATH_SIZE_MISMATCH, if the dimensions do not match.
   * If the input matrix is singular (does not have an inverse), then the algorithm terminates and returns erro
   */
  arm_status arm_mat_inverse_f64(
  const arm_matrix_instance_f64 * src,
  arm_matrix_instance_f64 * dst);
```

```
/**
 * @ingroup groupController
 */

/**
 * @defgroup clarke Vector Clarke Transform
 * Forward Clarke transform converts the instantaneous stator phases into a two-coordinate time invariant
 * Generally the Clarke transform uses three-phase currents <code>Ia, Ib and Ic</code> to calculate curre
 * in the two-phase orthogonal stator axis <code>Ialpha</code> and <code>Ibeta</code>.
 * When <code>Ialpha</code> is superposed with <code>Ia</code> as shown in the figure below
 * \image html clarke.gif Stator current space vector and its components in (a,b).
 * and <code>Ia + Ib + Ic = 0</code>, in this condition <code>Ialpha</code> and <code>Ibeta</code>
 * can be calculated using only <code>Ia</code> and <code>Ib</code>.
 *
 * The function operates on a single sample of data and each call to the function returns the processed ou
 * The library provides separate functions for Q31 and floating-point data types.
 * \par Algorithm
 * \image html clarkeFormula.gif
 * where <code>Ia</code> and <code>Ib</code> are the instantaneous stator phases and
 * <code>pIalpha</code> and <code>pIbeta</code> are the two coordinates of time invariant vector.
 * \par Fixed-Point Behavior
 * Care must be taken when using the Q31 version of the Clarke transform.
 * In particular, the overflow and saturation behavior of the accumulator used must be considered.
 * Refer to the function specific documentation below for usage guidelines.
 */

/**
 * @addtogroup clarke
 * @{
 */

/**
 *
 * @brief  Floating-point Clarke transform
 * @param[in]  Ia       input three-phase coordinate <code>a</code>
 * @param[in]  Ib       input three-phase coordinate <code>b</code>
 * @param[out] pIalpha  points to output two-phase orthogonal vector axis alpha
 * @param[out] pIbeta   points to output two-phase orthogonal vector axis beta
 */
static __INLINE void arm_clarke_f32(
float32_t Ia,
float32_t Ib,
float32_t * pIalpha,
float32_t * pIbeta)
{
  /* Calculate pIalpha using the equation, pIalpha = Ia */
  *pIalpha = Ia;

  /* Calculate pIbeta using the equation, pIbeta = (1/sqrt(3)) * Ia + (2/sqrt(3)) * Ib */
  *pIbeta = ((float32_t) 0.57735026919 * Ia + (float32_t) 1.15470053838 * Ib);
}
/**
 * @brief  Clarke transform for Q31 version
 * @param[in]  Ia       input three-phase coordinate <code>a</code>
 * @param[in]  Ib       input three-phase coordinate <code>b</code>
 * @param[out] pIalpha  points to output two-phase orthogonal vector axis alpha
 * @param[out] pIbeta   points to output two-phase orthogonal vector axis beta
 *
 * <b>Scaling and Overflow Behavior:</b>
 * \par
 * The function is implemented using an internal 32-bit accumulator.
 * The accumulator maintains 1.31 format by truncating lower 31 bits of the intermediate multiplication in 2
 * There is saturation on the addition, hence there is no risk of overflow.
 */
static __INLINE void arm_clarke_q31(
q31_t Ia,
q31_t Ib,
q31_t * pIalpha,
q31_t * pIbeta)
{
  q31_t product1, product2;                    /* Temporary variables used to store intermediate results */

  /* Calculating pIalpha from Ia by equation pIalpha = Ia */
  *pIalpha = Ia;

  /* Intermediate product is calculated by (1/(sqrt(3)) * Ia) */
  product1 = (q31_t) (((q63_t) Ia * 0x24F34E8B) >> 30);

  /* Intermediate product is calculated by (2/sqrt(3) * Ib) */
  product2 = (q31_t) (((q63_t) Ib * 0x49E69D16) >> 30);

  /* pIbeta is calculated by adding the intermediate products */
  *pIbeta = __QADD(product1, product2);
}

/**
 * @} end of clarke group
 */

/**
 * @brief  Converts the elements of the Q7 vector to Q31 vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_q7_to_q31(
q7_t * pSrc,
q31_t * pDst,
uint32_t blockSize);




/**
```

```
 * @ingroup groupController
 */

/**
 * @defgroup inv_clarke Vector Inverse Clarke Transform
 * Inverse Clarke transform converts the two-coordinate time invariant vector into instantaneous stator pha
 *
 * The function operates on a single sample of data and each call to the function returns the processed ou
 * The library provides separate functions for Q31 and floating-point data types.
 * \par Algorithm
 * \image html clarkeInvFormula.gif
 * where <code>pIa</code> and <code>pIb</code> are the instantaneous stator phases and
 * <code>Ialpha</code> and <code>Ibeta</code> are the two coordinates of time invariant vector.
 * \par Fixed-Point Behavior
 * Care must be taken when using the Q31 version of the Clarke transform.
 * In particular, the overflow and saturation behavior of the accumulator used must be considered.
 * Refer to the function specific documentation below for usage guidelines.
 */

/**
 * @addtogroup inv_clarke
 * @{
 */

 /**
 * @brief  Floating-point Inverse Clarke transform
 * @param[in]  Ialpha  input two-phase orthogonal vector axis alpha
 * @param[in]  Ibeta   input two-phase orthogonal vector axis beta
 * @param[out] pIa     points to output three-phase coordinate <code>a</code>
 * @param[out] pIb     points to output three-phase coordinate <code>b</code>
 */
static __INLINE void arm_inv_clarke_f32(
float32_t Ialpha,
float32_t Ibeta,
float32_t * pIa,
float32_t * pIb)
{
  /* Calculating pIa from Ialpha by equation pIa = Ialpha */
  *pIa = Ialpha;

  /* Calculating pIb from Ialpha and Ibeta by equation pIb = -(1/2) * Ialpha + (sqrt(3)/2) * Ibeta */
  *pIb = -0.5f * Ialpha + 0.8660254039f * Ibeta;
}


/**
 * @brief  Inverse Clarke transform for Q31 version
 * @param[in]  Ialpha  input two-phase orthogonal vector axis alpha
 * @param[in]  Ibeta   input two-phase orthogonal vector axis beta
 * @param[out] pIa     points to output three-phase coordinate <code>a</code>
 * @param[out] pIb     points to output three-phase coordinate <code>b</code>
 *
 * <b>Scaling and Overflow Behavior:</b>
 * \par
 * The function is implemented using an internal 32-bit accumulator.
 * The accumulator maintains 1.31 format by truncating lower 31 bits of the intermediate multiplication in 2
 * There is saturation on the subtraction, hence there is no risk of overflow.
 */
static __INLINE void arm_inv_clarke_q31(
q31_t Ialpha,
q31_t Ibeta,
q31_t * pIa,
q31_t * pIb)
{
  q31_t product1, product2;                    /* Temporary variables used to store intermediate results */

  /* Calculating pIa from Ialpha by equation pIa = Ialpha */
  *pIa = Ialpha;

  /* Intermediate product is calculated by (1/(2*sqrt(3)) * Ia) */
  product1 = (q31_t) (((q63_t) (Ialpha) * (0x40000000)) >> 31);

  /* Intermediate product is calculated by (1/sqrt(3) * pIb) */
  product2 = (q31_t) (((q63_t) (Ibeta) * (0x6ED9EBA1)) >> 31);

  /* pIb is calculated by subtracting the products */
  *pIb = __QSUB(product2, product1);
}

/**
 * @} end of inv_clarke group
 */

/**
 * @brief  Converts the elements of the Q7 vector to Q15 vector.
 * @param[in]  pSrc       input pointer
 * @param[out] pDst       output pointer
 * @param[in]  blockSize  number of samples to process
 */
void arm_q7_to_q15(
q7_t * pSrc,
q15_t * pDst,
uint32_t blockSize);



/**
 * @ingroup groupController
 */

/**
 * @defgroup park Vector Park Transform
 *
 * Forward Park transform converts the input two-coordinate vector to flux and torque components.
 * The Park transform can be used to realize the transformation of the <code>Ialpha</code> and the <coc
 * from the stationary to the moving reference frame and control the spatial relationship between
 * the stator vector current and rotor flux vector.
 * If we consider the d axis aligned with the rotor flux, the diagram below shows the
 * current vector and the relationship from the two reference frames:
 * \image html park.gif "Stator current space vector and its component in (a,b) and in the d,q rotating refer
 *
 * The function operates on a single sample of data and each call to the function returns the processed ou
 * The library provides separate functions for Q31 and floating-point data types.
 * \par Algorithm
 * \image html parkFormula.gif
 * where <code>Ialpha</code> and <code>Ibeta</code> are the stator vector components,
 * <code>pId</code> and <code>pIq</code> are rotor vector components and <code>cosVal</code> and
 * cosine and sine values of theta (rotor flux position).
 * \par Fixed-Point Behavior
 * Care must be taken when using the Q31 version of the Park transform.
 * In particular, the overflow and saturation behavior of the accumulator used must be considered.
 * Refer to the function specific documentation below for usage guidelines.
 */

/**
 * @addtogroup park
 * @{
 */

/**
 * @brief Floating-point Park transform
 * @param[in]  Ialpha  input two-phase vector coordinate alpha
 * @param[in]  Ibeta   input two-phase vector coordinate beta
 * @param[out] pId     points to output   rotor reference frame d
 * @param[out] pIq     points to output   rotor reference frame q
 * @param[in]  sinVal  sine value of rotation angle theta
 * @param[in]  cosVal  cosine value of rotation angle theta
 *
 * The function implements the forward Park transform.
 *
 */
static __INLINE void arm_park_f32(
float32_t Ialpha,
float32_t Ibeta,
float32_t * pId,
float32_t * pIq,
float32_t sinVal,
float32_t cosVal)
{
  /* Calculate pId using the equation, pId = Ialpha * cosVal + Ibeta * sinVal */
  *pId = Ialpha * cosVal + Ibeta * sinVal;

  /* Calculate pIq using the equation, pIq = - Ialpha * sinVal + Ibeta * cosVal */
  *pIq = -Ialpha * sinVal + Ibeta * cosVal;
}


/**
 * @brief  Park transform for Q31 version
 * @param[in]  Ialpha  input two-phase vector coordinate alpha
 * @param[in]  Ibeta   input two-phase vector coordinate beta
 * @param[out] pId     points to output rotor reference frame d
 * @param[out] pIq     points to output rotor reference frame q
 * @param[in]  sinVal  sine value of rotation angle theta
 * @param[in]  cosVal  cosine value of rotation angle theta
 *
 * <b>Scaling and Overflow Behavior:</b>
 * \par
 * The function is implemented using an internal 32-bit accumulator.
 * The accumulator maintains 1.31 format by truncating lower 31 bits of the intermediate multiplication in 2
 * There is saturation on the addition and subtraction, hence there is no risk of overflow.
 */
static __INLINE void arm_park_q31(
q31_t Ialpha,
q31_t Ibeta,
q31_t * pId,
q31_t * pIq,
q31_t sinVal,
q31_t cosVal)
{
  q31_t product1, product2;                    /* Temporary variables used to store intermediate results */
  q31_t product3, product4;                    /* Temporary variables used to store intermediate results */

  /* Intermediate product is calculated by (Ialpha * cosVal) */
  product1 = (q31_t) (((q63_t) (Ialpha) * (cosVal)) >> 31);

  /* Intermediate product is calculated by (Ibeta * sinVal) */
  product2 = (q31_t) (((q63_t) (Ibeta) * (sinVal)) >> 31);


  /* Intermediate product is calculated by (Ialpha * sinVal) */
  product3 = (q31_t) (((q63_t) (Ialpha) * (sinVal)) >> 31);

  /* Intermediate product is calculated by (Ibeta * cosVal) */
  product4 = (q31_t) (((q63_t) (Ibeta) * (cosVal)) >> 31);

  /* Calculate pId by adding the two intermediate products 1 and 2 */
  *pId = __QADD(product1, product2);

  /* Calculate pIq by subtracting the two intermediate products 3 from 4 */
  *pIq = __QSUB(product4, product3);
}

/**
 * @} end of park group
 */

/**
 * @brief  Converts the elements of the Q7 vector to floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q7_to_float(
q7_t * pSrc,
float32_t * pDst,
uint32_t blockSize);



/**
 * @ingroup groupController
 */

/**
 * @defgroup inv_park Vector Inverse Park transform
 * Inverse Park transform converts the input flux and torque components to two-coordinate vector.
 *
 * The function operates on a single sample of data and each call to the function returns the processed ou
 * The library provides separate functions for Q31 and floating-point data types.
 * \par Algorithm
 * \image html parkInvFormula.gif
 * where <code>pIalpha</code> and <code>pIbeta</code> are the stator vector components,
 * <code>Id</code> and <code>Iq</code> are rotor vector components and <code>cosVal</code> and <
 * cosine and sine values of theta (rotor flux position).
 * \par Fixed-Point Behavior
 * Care must be taken when using the Q31 version of the Park transform.
 * In particular, the overflow and saturation behavior of the accumulator used must be considered.
 * Refer to the function specific documentation below for usage guidelines.
 */

/**
 * @addtogroup inv_park
 * @{
 */

 /**
 * @brief  Floating-point Inverse Park transform
 * @param[in]  Id       input coordinate of rotor reference frame d
 * @param[in]  Iq       input coordinate of rotor reference frame q
 * @param[out] pIalpha  points to output two-phase orthogonal vector axis alpha
 * @param[out] pIbeta   points to output two-phase orthogonal vector axis beta
 * @param[in]  sinVal   sine value of rotation angle theta
 * @param[in]  cosVal   cosine value of rotation angle theta
 */
static __INLINE void arm_inv_park_f32(
float32_t Id,
float32_t Iq,
float32_t * pIalpha,
float32_t * pIbeta,
float32_t sinVal,
float32_t cosVal)
{
  /* Calculate pIalpha using the equation, pIalpha = Id * cosVal - Iq * sinVal */
  *pIalpha = Id * cosVal - Iq * sinVal;
  /* Calculate pIbeta using the equation, pIbeta = Id * sinVal + Iq * cosVal */
  *pIbeta = Id * sinVal + Iq * cosVal;
}


/**
 * @brief  Inverse Park transform for   Q31 version
 * @param[in]  Id       input coordinate of rotor reference frame d
 * @param[in]  Iq       input coordinate of rotor reference frame q
 * @param[out] pIalpha  points to output two-phase orthogonal vector axis alpha
 * @param[out] pIbeta   points to output two-phase orthogonal vector axis beta
 * @param[in]  sinVal   sine value of rotation angle theta
 * @param[in]  cosVal   cosine value of rotation angle theta
 *
 * <b>Scaling and Overflow Behavior:</b>
 * \par
 * The function is implemented using an internal 32-bit accumulator.
 * The accumulator maintains 1.31 format by truncating lower 31 bits of the intermediate multiplication in 2
 * There is saturation on the addition, hence there is no risk of overflow.
 */
static __INLINE void arm_inv_park_q31(
q31_t Id,
q31_t Iq,
q31_t * pIalpha,
q31_t * pIbeta,
q31_t sinVal,
q31_t cosVal)
{
  q31_t product1, product2;                    /* Temporary variables used to store intermediate results */
  q31_t product3, product4;                    /* Temporary variables used to store intermediate results */

  /* Intermediate product is calculated by (Id * cosVal) */
  product1 = (q31_t) (((q63_t) (Id) * (cosVal)) >> 31);

  /* Intermediate product is calculated by (Iq * sinVal) */
  product2 = (q31_t) (((q63_t) (Iq) * (sinVal)) >> 31);


  /* Intermediate product is calculated by (Id * sinVal) */
  product3 = (q31_t) (((q63_t) (Id) * (sinVal)) >> 31);

  /* Intermediate product is calculated by (Iq * cosVal) */
  product4 = (q31_t) (((q63_t) (Iq) * (cosVal)) >> 31);

  /* Calculate pIalpha by using the two intermediate products 1 and 2 */
  *pIalpha = __QSUB(product1, product2);

  /* Calculate pIbeta by using the two intermediate products 3 and 4 */
  *pIbeta = __QADD(product4, product3);
}

/**
 * @} end of Inverse park group
 */


/**
 * @brief  Converts the elements of the Q31 vector to floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q31_to_float(
q31_t * pSrc,
float32_t * pDst,
uint32_t blockSize);

/**
 * @ingroup groupInterpolation
 */

/**
 * @defgroup LinearInterpolate Linear Interpolation
 *
 * Linear interpolation is a method of curve fitting using linear polynomials.
 * Linear interpolation works by effectively drawing a straight line between two neighboring samples and r
 *
 * \par
 * \image html LinearInterp.gif "Linear interpolation"
 *
 * \par
 * A  Linear Interpolate function calculates an output value(y), for the input(x)
 * using linear interpolation of the input values x0, x1( nearest input values) and the output values y0 and 
 *
 * \par Algorithm:
 * <pre>
 *       y = y0 + (x - x0) * ((y1 - y0)/(x1-x0))
 *       where x0, x1 are nearest values of input x
 *             y0, y1 are nearest values to output y
 * </pre>
 *
 * \par
 * This set of functions implements Linear interpolation process
 * for Q7, Q15, Q31, and floating-point data types.  The functions operate on a single
 * sample of data and each call to the function returns a single processed value.
 * <code>S</code> points to an instance of the Linear Interpolate function data structure.
 * <code>x</code> is the input sample value. The functions returns the output value.
 *
 * \par
 * if x is outside of the table boundary, Linear interpolation returns first value of the table
 * if x is below input range and returns last value of table if x is above range.
 */

/**
 * @addtogroup LinearInterpolate
 * @{
```

```
 */

/**
 * @brief  Process function for the floating-point Linear Interpolation Function.
 * @param[in,out] S  is an instance of the floating-point Linear Interpolation structure
 * @param[in]     x  input sample to process
 * @return y processed output sample.
 *
 */
static __INLINE float32_t arm_linear_interp_f32(
arm_linear_interp_instance_f32 * S,
float32_t x)
{
  float32_t y;
  float32_t x0, x1;                              /* Nearest input values */
  float32_t y0, y1;                              /* Nearest output values */
  float32_t xSpacing = S->xSpacing;            /* spacing between input values */
  int32_t i;                                   /* Index variable */
  float32_t *pYData = S->pYData;               /* pointer to output table */

  /* Calculation of index */
  i = (int32_t) ((x - S->x1) / xSpacing);

  if(i < 0)
  {
    /* Iniatilize output for below specified range as least output value of table */
    y = pYData[0];
  }
  else if((uint32_t)i >= S->nValues)
  {
    /* Iniatilize output for above specified range as last output value of table */
    y = pYData[S->nValues - 1];
  }
  else
  {
    /* Calculation of nearest input values */
    x0 = S->x1 +  i      * xSpacing;
    x1 = S->x1 + (i + 1) * xSpacing;

    /* Read of nearest output values */
    y0 = pYData[i];
    y1 = pYData[i + 1];

    /* Calculation of output */
    y = y0 + (x - x0) * ((y1 - y0) / (x1 - x0));

  }

  /* returns output value */
  return (y);
}
```

```
 /**
 *
 * @brief  Process function for the Q31 Linear Interpolation Function.
 * @param[in] pYData   pointer to Q31 Linear Interpolation table
 * @param[in] x        input sample to process
 * @param[in] nValues  number of table values
 * @return y processed output sample.
 *
 * \par
 * Input sample <code>x</code> is in 12.20 format which contains 12 bits for table index and 20 bits for fr
 * This function can support maximum of table size 2^12.
 *
 */
static __INLINE q31_t arm_linear_interp_q31(
q31_t * pYData,
q31_t x,
uint32_t nValues)
{
  q31_t y;                                  /* output */
  q31_t y0, y1;                               /* Nearest output values */
  q31_t fract;                               /* fractional part */
  int32_t index;                              /* Index to read nearest output values */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  index = ((x & (q31_t)0xFFF00000) >> 20);

  if(index >= (int32_t)(nValues - 1))
  {
    return (pYData[nValues - 1]);
  }
  else if(index < 0)
  {
    return (pYData[0]);
  }
  else
  {
    /* 20 bits for the fractional part */
    /* shift left by 11 to keep fract in 1.31 format */
    fract = (x & 0x000FFFFF) << 11;

    /* Read two nearest output values from the index in 1.31(q31) format */
    y0 = pYData[index];
    y1 = pYData[index + 1];

    /* Calculation of y0 * (1-fract) and y is in 2.30 format */
    y = ((q31_t) ((q63_t) y0 * (0x7FFFFFFF - fract) >> 32));

    /* Calculation of y0 * (1-fract) + y1 *fract and y is in 2.30 format */
    y += ((q31_t) (((q63_t) y1 * fract) >> 32));

    /* Convert y to 1.31 format */
    return (y << 1u);
  }
}


/**
 *
 * @brief  Process function for the Q15 Linear Interpolation Function.
 * @param[in] pYData   pointer to Q15 Linear Interpolation table
 * @param[in] x        input sample to process
 * @param[in] nValues  number of table values
 * @return y processed output sample.
 *
 * \par
 * Input sample <code>x</code> is in 12.20 format which contains 12 bits for table index and 20 bits for fr
 * This function can support maximum of table size 2^12.
 *
 */
static __INLINE q15_t arm_linear_interp_q15(
q15_t * pYData,
q31_t x,
uint32_t nValues)
{
  q63_t y;                                    /* output */
  q15_t y0, y1;                               /* Nearest output values */
  q31_t fract;                                /* fractional part */
  int32_t index;                              /* Index to read nearest output values */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  index = ((x & (int32_t)0xFFF00000) >> 20);

  if(index >= (int32_t)(nValues - 1))
  {
    return (pYData[nValues - 1]);
  }
  else if(index < 0)
  {
    return (pYData[0]);
  }
  else
  {
    /* 20 bits for the fractional part */
    /* fract is in 12.20 format */
    fract = (x & 0x000FFFFF);

    /* Read two nearest output values from the index */
    y0 = pYData[index];
    y1 = pYData[index + 1];

    /* Calculation of y0 * (1-fract) and y is in 13.35 format */
    y = ((q63_t) y0 * (0xFFFFF - fract));
    /* Calculation of (y0 * (1-fract) + y1 * fract) and y is in 13.35 format */
    y += ((q63_t) y1 * (fract));

    /* convert y to 1.15 format */
    return (q15_t) (y >> 20);
  }
}


/**
 *
 * @brief  Process function for the Q7 Linear Interpolation Function.
 * @param[in] pYData   pointer to Q7 Linear Interpolation table
 * @param[in] x        input sample to process
 * @param[in] nValues  number of table values
 * @return y processed output sample.
 *
 * \par
 * Input sample <code>x</code> is in 12.20 format which contains 12 bits for table index and 20 bits for fr
 * This function can support maximum of table size 2^12.
 */
static __INLINE q7_t arm_linear_interp_q7(
q7_t * pYData,
q31_t x,
uint32_t nValues)
{
  q31_t y;                                   /* output */
  q7_t y0, y1;                                 /* Nearest output values */
  q31_t fract;                                /* fractional part */
  uint32_t index;                               /* Index to read nearest output values */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  if (x < 0)
  {
    return (pYData[0]);
  }
  index = (x >> 20) & 0xfff;

  if(index >= (nValues - 1))
  {
    return (pYData[nValues - 1]);
  }
  else
  {
    /* 20 bits for the fractional part */
    /* fract is in 12.20 format */
    fract = (x & 0x000FFFFF);

    /* Read two nearest output values from the index and are in 1.7(q7) format */
    y0 = pYData[index];
    y1 = pYData[index + 1];

    /* Calculation of y0 * (1-fract ) and y is in 13.27(q27) format */
    y = ((y0 * (0xFFFFF - fract)));

    /* Calculation of y1 * fract + y0 * (1-fract) and y is in 13.27(q27) format */
    y += (y1 * fract);

    /* convert y to 1.7(q7) format */
    return (q7_t) (y >> 20);
  }
}

/**
 * @} end of LinearInterpolate group
 */

/**
 * @brief  Fast approximation to the trigonometric sine function for floating-point data.
 * @param[in] x  input value in radians.
 * @return  sin(x).
 */
float32_t arm_sin_f32(
float32_t x);


/**
 * @brief  Fast approximation to the trigonometric sine function for Q31 data.
 * @param[in] x  Scaled input value in radians.
 * @return  sin(x).
 */
q31_t arm_sin_q31(
q31_t x);


/**
 * @brief  Fast approximation to the trigonometric sine function for Q15 data.
 * @param[in] x  Scaled input value in radians.
 * @return  sin(x).
 */
q15_t arm_sin_q15(
q15_t x);


/**
 * @brief  Fast approximation to the trigonometric cosine function for floating-point data.
 * @param[in] x  input value in radians.
 * @return  cos(x).
 */
float32_t arm_cos_f32(
float32_t x);
```

```
/**
 * @brief Fast approximation to the trigonometric cosine function for Q31 data.
 * @param[in] x  Scaled input value in radians.
 * @return  cos(x).
 */
q31_t arm_cos_q31(
q31_t x);


/**
 * @brief  Fast approximation to the trigonometric cosine function for Q15 data.
 * @param[in] x  Scaled input value in radians.
 * @return  cos(x).
 */
q15_t arm_cos_q15(
q15_t x);


/**
 * @ingroup groupFastMath
 */


/**
 * @defgroup SQRT Square Root
 *
 * Computes the square root of a number.
 * There are separate functions for Q15, Q31, and floating-point data types.
 * The square root function is computed using the Newton-Raphson algorithm.
 * This is an iterative algorithm of the form:
 * <pre>
 *      x1 = x0 - f(x0)/f'(x0)
 * </pre>
 * where <code>x1</code> is the current estimate,
 * <code>x0</code> is the previous estimate, and
 * <code>f'(x0)</code> is the derivative of <code>f()</code> evaluated at <code>x0</code>.
 * For the square root function, the algorithm reduces to:
 * <pre>
 *     x0 = in/2                         [initial guess]
 *     x1 = 1/2 * ( x0 + in / x0)        [each iteration]
 * </pre>
 */


/**
 * @addtogroup SQRT
 * @{
 */

/**
 * @brief  Floating-point square root function.
 * @param[in]  in    input value.
 * @param[out] pOut  square root of input value.
```

```
   * @return The function returns ARM_MATH_SUCCESS if input value is positive value or ARM_MATH_A
   * <code>in</code> is negative value and returns zero output for negative values.
   */
  static __INLINE arm_status arm_sqrt_f32(
  float32_t in,
  float32_t * pOut)
  {
    if(in >= 0.0f)
    {

#if   (__FPU_USED == 1) && defined ( __CC_ARM   )
      *pOut = __sqrtf(in);
#elif (__FPU_USED == 1) && (defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050))
      *pOut = __builtin_sqrtf(in);
#elif (__FPU_USED == 1) && defined(__GNUC__)
      *pOut = __builtin_sqrtf(in);
#elif (__FPU_USED == 1) && defined ( __ICCARM__ ) && (__VER__ >= 6040000)
      __ASM("VSQRT.F32 %0,%1" : "=t"(*pOut) : "t"(in));
#else
      *pOut = sqrtf(in);
#endif

      return (ARM_MATH_SUCCESS);
    }
    else
    {
      *pOut = 0.0f;
      return (ARM_MATH_ARGUMENT_ERROR);
    }
  }


  /**
   * @brief Q31 square root function.
   * @param[in]  in    input value.  The range of the input value is [0 +1) or 0x00000000 to 0x7FFFFFFF.
   * @param[out] pOut  square root of input value.
   * @return The function returns ARM_MATH_SUCCESS if input value is positive value or ARM_MATH_A
   * <code>in</code> is negative value and returns zero output for negative values.
   */
  arm_status arm_sqrt_q31(
  q31_t in,
  q31_t * pOut);


  /**
   * @brief  Q15 square root function.
   * @param[in]  in    input value.  The range of the input value is [0 +1) or 0x0000 to 0x7FFF.
   * @param[out] pOut  square root of input value.
   * @return The function returns ARM_MATH_SUCCESS if input value is positive value or ARM_MATH_A
   * <code>in</code> is negative value and returns zero output for negative values.
   */
  arm_status arm_sqrt_q15(
  q15_t in,
q15_t * pOut);

/**
 * @} end of SQRT group
 */



/**
 * @brief floating-point Circular write function.
 */
static __INLINE void arm_circularWrite_f32(
int32_t * circBuffer,
int32_t L,
uint16_t * writeOffset,
int32_t bufferInc,
const int32_t * src,
int32_t srcInc,
uint32_t blockSize)
{
  uint32_t i = 0u;
  int32_t wOffset;

  /* Copy the value of Index pointer that points
   * to the current location where the input samples to be copied */
  wOffset = *writeOffset;

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the input sample to the circular buffer */
    circBuffer[wOffset] = *src;

    /* Update the input pointer */
    src += srcInc;

    /* Circularly update wOffset.  Watch out for positive and negative value */
    wOffset += bufferInc;
    if(wOffset >= L)
      wOffset -= L;

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *writeOffset = (uint16_t)wOffset;
}



/**
```

```
 * @brief floating-point Circular Read function.
 */
static __INLINE void arm_circularRead_f32(
int32_t * circBuffer,
int32_t L,
int32_t * readOffset,
int32_t bufferInc,
int32_t * dst,
int32_t * dst_base,
int32_t dst_length,
int32_t dstInc,
uint32_t blockSize)
{
  uint32_t i = 0u;
  int32_t rOffset, dst_end;

  /* Copy the value of Index pointer that points
   * to the current location from where the input samples to be read */
  rOffset = *readOffset;
  dst_end = (int32_t) (dst_base + dst_length);

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the sample from the circular buffer to the destination buffer */
    *dst = circBuffer[rOffset];

    /* Update the input pointer */
    dst += dstInc;

    if(dst == (int32_t *) dst_end)
    {
      dst = dst_base;
    }

    /* Circularly update rOffset.  Watch out for positive and negative value  */
    rOffset += bufferInc;

    if(rOffset >= L)
    {
      rOffset -= L;
    }

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *readOffset = rOffset;
}
```

```
/**
 * @brief Q15 Circular write function.
 */
static __INLINE void arm_circularWrite_q15(
q15_t * circBuffer,
int32_t L,
uint16_t * writeOffset,
int32_t bufferInc,
const q15_t * src,
int32_t srcInc,
uint32_t blockSize)
{
  uint32_t i = 0u;
  int32_t wOffset;

  /* Copy the value of Index pointer that points
   * to the current location where the input samples to be copied */
  wOffset = *writeOffset;

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the input sample to the circular buffer */
    circBuffer[wOffset] = *src;

    /* Update the input pointer */
    src += srcInc;

    /* Circularly update wOffset.  Watch out for positive and negative value */
    wOffset += bufferInc;
    if(wOffset >= L)
      wOffset -= L;

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *writeOffset = (uint16_t)wOffset;
}


/**
 * @brief Q15 Circular Read function.
 */
static __INLINE void arm_circularRead_q15(
q15_t * circBuffer,
int32_t L,
int32_t * readOffset,
int32_t bufferInc,
q15_t * dst,
q15_t * dst_base,
int32_t dst_length,
int32_t dstInc,
uint32_t blockSize)
{
  uint32_t i = 0;
  int32_t rOffset, dst_end;

  /* Copy the value of Index pointer that points
   * to the current location from where the input samples to be read */
  rOffset = *readOffset;

  dst_end = (int32_t) (dst_base + dst_length);

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the sample from the circular buffer to the destination buffer */
    *dst = circBuffer[rOffset];

    /* Update the input pointer */
    dst += dstInc;

    if(dst == (q15_t *) dst_end)
    {
      dst = dst_base;
    }

    /* Circularly update wOffset.  Watch out for positive and negative value */
    rOffset += bufferInc;

    if(rOffset >= L)
    {
      rOffset -= L;
    }

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *readOffset = rOffset;
}


/**
 * @brief Q7 Circular write function.
 */
static __INLINE void arm_circularWrite_q7(
q7_t * circBuffer,
int32_t L,
uint16_t * writeOffset,
int32_t bufferInc,
const q7_t * src,
int32_t srcInc,
uint32_t blockSize)
{
  uint32_t i = 0u;
  int32_t wOffset;

  /* Copy the value of Index pointer that points
   * to the current location where the input samples to be copied */
  wOffset = *writeOffset;

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the input sample to the circular buffer */
    circBuffer[wOffset] = *src;

    /* Update the input pointer */
    src += srcInc;

    /* Circularly update wOffset.  Watch out for positive and negative value */
    wOffset += bufferInc;
    if(wOffset >= L)
      wOffset -= L;

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *writeOffset = (uint16_t)wOffset;
}


/**
 * @brief Q7 Circular Read function.
 */
static __INLINE void arm_circularRead_q7(
q7_t * circBuffer,
int32_t L,
int32_t * readOffset,
int32_t bufferInc,
q7_t * dst,
q7_t * dst_base,
int32_t dst_length,
int32_t dstInc,
uint32_t blockSize)
{
  uint32_t i = 0;
  int32_t rOffset, dst_end;

  /* Copy the value of Index pointer that points
   * to the current location from where the input samples to be read */
  rOffset = *readOffset;

  dst_end = (int32_t) (dst_base + dst_length);

  /* Loop over the blockSize */
  i = blockSize;

  while(i > 0u)
  {
    /* copy the sample from the circular buffer to the destination buffer */
    *dst = circBuffer[rOffset];

    /* Update the input pointer */
    dst += dstInc;

    if(dst == (q7_t *) dst_end)
    {
      dst = dst_base;
    }

    /* Circularly update rOffset.  Watch out for positive and negative value */
    rOffset += bufferInc;

    if(rOffset >= L)
    {
      rOffset -= L;
    }

    /* Decrement the loop counter */
    i--;
  }

  /* Update the index pointer */
  *readOffset = rOffset;
}


/**
 * @brief  Sum of the squares of the elements of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_power_q31(
q31_t * pSrc,
uint32_t blockSize,
q63_t * pResult);
```

```
/**
 * @brief  Sum of the squares of the elements of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_power_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult);


/**
 * @brief  Sum of the squares of the elements of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_power_q15(
q15_t * pSrc,
uint32_t blockSize,
q63_t * pResult);


/**
 * @brief  Sum of the squares of the elements of a Q7 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_power_q7(
q7_t * pSrc,
uint32_t blockSize,
q31_t * pResult);


/**
 * @brief  Mean value of a Q7 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_mean_q7(
q7_t * pSrc,
uint32_t blockSize,
q7_t * pResult);


/**
 * @brief  Mean value of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_mean_q15(
q15_t * pSrc,
uint32_t blockSize,
q15_t * pResult);


/**
 * @brief  Mean value of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_mean_q31(
q31_t * pSrc,
uint32_t blockSize,
q31_t * pResult);


/**
 * @brief  Mean value of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_mean_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult);


/**
 * @brief  Variance of the elements of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_var_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult);


/**
 * @brief  Variance of the elements of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_var_q31(
q31_t * pSrc,
uint32_t blockSize,
q31_t * pResult);


/**
 * @brief  Variance of the elements of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_var_q15(
q15_t * pSrc,
uint32_t blockSize,
q15_t * pResult);


/**
 * @brief  Root Mean Square of the elements of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_rms_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult);


/**
 * @brief  Root Mean Square of the elements of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_rms_q31(
q31_t * pSrc,
uint32_t blockSize,
q31_t * pResult);


/**
 * @brief  Root Mean Square of the elements of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_rms_q15(
q15_t * pSrc,
uint32_t blockSize,
q15_t * pResult);


/**
 * @brief  Standard deviation of the elements of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_std_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult);


/**
 * @brief  Standard deviation of the elements of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_std_q31(
q31_t * pSrc,
uint32_t blockSize,
q31_t * pResult);


/**
 * @brief  Standard deviation of the elements of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output value.
 */
void arm_std_q15(
q15_t * pSrc,
uint32_t blockSize,
q15_t * pResult);


/**
 * @brief  Floating-point complex magnitude
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_f32(
float32_t * pSrc,
float32_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q31 complex magnitude
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_q31(
q31_t * pSrc,
q31_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q15 complex magnitude
 * @param[in]  pSrc        points to the complex input vector
 * @param[out] pDst        points to the real output vector
 * @param[in]  numSamples  number of complex samples in the input vector
 */
void arm_cmplx_mag_q15(
q15_t * pSrc,
q15_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q15 complex dot product
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[in]  numSamples  number of complex samples in each vector
 * @param[out] realResult  real part of the result returned here
 * @param[out] imagResult  imaginary part of the result returned here
 */
void arm_cmplx_dot_prod_q15(
q15_t * pSrcA,
q15_t * pSrcB,
uint32_t numSamples,
q31_t * realResult,
q31_t * imagResult);


/**
 * @brief  Q31 complex dot product
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[in]  numSamples  number of complex samples in each vector
 * @param[out] realResult  real part of the result returned here
 * @param[out] imagResult  imaginary part of the result returned here
 */
void arm_cmplx_dot_prod_q31(
q31_t * pSrcA,
q31_t * pSrcB,
uint32_t numSamples,
q63_t * realResult,
q63_t * imagResult);


/**
 * @brief  Floating-point complex dot product
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[in]  numSamples  number of complex samples in each vector
 * @param[out] realResult  real part of the result returned here
 * @param[out] imagResult  imaginary part of the result returned here
 */
void arm_cmplx_dot_prod_f32(
float32_t * pSrcA,
float32_t * pSrcB,
uint32_t numSamples,
float32_t * realResult,
float32_t * imagResult);


/**
 * @brief  Q15 complex-by-real multiplication
 * @param[in]  pSrcCmplx   points to the complex input vector
 * @param[in]  pSrcReal    points to the real input vector
 * @param[out] pCmplxDst   points to the complex output vector
 * @param[in]  numSamples  number of samples in each vector
 */
void arm_cmplx_mult_real_q15(
q15_t * pSrcCmplx,
q15_t * pSrcReal,
q15_t * pCmplxDst,
uint32_t numSamples);


/**
 * @brief  Q31 complex-by-real multiplication
 * @param[in]  pSrcCmplx   points to the complex input vector
 * @param[in]  pSrcReal    points to the real input vector
 * @param[out] pCmplxDst   points to the complex output vector
 * @param[in]  numSamples  number of samples in each vector
 */
void arm_cmplx_mult_real_q31(
q31_t * pSrcCmplx,
q31_t * pSrcReal,
q31_t * pCmplxDst,
uint32_t numSamples);


/**
 * @brief  Floating-point complex-by-real multiplication
 * @param[in]  pSrcCmplx   points to the complex input vector
 * @param[in]  pSrcReal    points to the real input vector
 * @param[out] pCmplxDst   points to the complex output vector
 * @param[in]  numSamples  number of samples in each vector
 */
void arm_cmplx_mult_real_f32(
float32_t * pSrcCmplx,
float32_t * pSrcReal,
float32_t * pCmplxDst,
uint32_t numSamples);
```

```
/**
 * @brief  Minimum value of a Q7 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] result     is output pointer
 * @param[in]  index      is the array index of the minimum value in the input buffer.
 */
void arm_min_q7(
q7_t * pSrc,
uint32_t blockSize,
q7_t * result,
uint32_t * index);


/**
 * @brief  Minimum value of a Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output pointer
 * @param[in]  pIndex     is the array index of the minimum value in the input buffer.
 */
void arm_min_q15(
q15_t * pSrc,
uint32_t blockSize,
q15_t * pResult,
uint32_t * pIndex);


/**
 * @brief  Minimum value of a Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output pointer
 * @param[out] pIndex     is the array index of the minimum value in the input buffer.
 */
void arm_min_q31(
q31_t * pSrc,
uint32_t blockSize,
q31_t * pResult,
uint32_t * pIndex);


/**
 * @brief  Minimum value of a floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[in]  blockSize  is the number of samples to process
 * @param[out] pResult    is output pointer
 * @param[out] pIndex     is the array index of the minimum value in the input buffer.
 */
void arm_min_f32(
float32_t * pSrc,
uint32_t blockSize,
  float32_t * pResult,
  uint32_t * pIndex);


/**
 * @brief Maximum value of a Q7 vector.
 * @param[in]  pSrc       points to the input buffer
 * @param[in]  blockSize  length of the input vector
 * @param[out] pResult    maximum value returned here
 * @param[out] pIndex     index of maximum value returned here
 */
  void arm_max_q7(
  q7_t * pSrc,
  uint32_t blockSize,
  q7_t * pResult,
  uint32_t * pIndex);


/**
 * @brief Maximum value of a Q15 vector.
 * @param[in]  pSrc       points to the input buffer
 * @param[in]  blockSize  length of the input vector
 * @param[out] pResult    maximum value returned here
 * @param[out] pIndex     index of maximum value returned here
 */
  void arm_max_q15(
  q15_t * pSrc,
  uint32_t blockSize,
  q15_t * pResult,
  uint32_t * pIndex);


/**
 * @brief Maximum value of a Q31 vector.
 * @param[in]  pSrc       points to the input buffer
 * @param[in]  blockSize  length of the input vector
 * @param[out] pResult    maximum value returned here
 * @param[out] pIndex     index of maximum value returned here
 */
  void arm_max_q31(
  q31_t * pSrc,
  uint32_t blockSize,
  q31_t * pResult,
  uint32_t * pIndex);


/**
 * @brief Maximum value of a floating-point vector.
 * @param[in]  pSrc       points to the input buffer
 * @param[in]  blockSize  length of the input vector
 * @param[out] pResult    maximum value returned here
 * @param[out] pIndex     index of maximum value returned here
 */
```

```
void arm_max_f32(
float32_t * pSrc,
uint32_t blockSize,
float32_t * pResult,
uint32_t * pIndex);


/**
 * @brief  Q15 complex-by-complex multiplication
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_mult_cmplx_q15(
q15_t * pSrcA,
q15_t * pSrcB,
q15_t * pDst,
uint32_t numSamples);


/**
 * @brief  Q31 complex-by-complex multiplication
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_mult_cmplx_q31(
q31_t * pSrcA,
q31_t * pSrcB,
q31_t * pDst,
uint32_t numSamples);


/**
 * @brief  Floating-point complex-by-complex multiplication
 * @param[in]  pSrcA       points to the first input vector
 * @param[in]  pSrcB       points to the second input vector
 * @param[out] pDst        points to the output vector
 * @param[in]  numSamples  number of complex samples in each vector
 */
void arm_cmplx_mult_cmplx_f32(
float32_t * pSrcA,
float32_t * pSrcB,
float32_t * pDst,
uint32_t numSamples);


/**
 * @brief Converts the elements of the floating-point vector to Q31 vector.
 * @param[in]  pSrc       points to the floating-point input vector
 * @param[out] pDst       points to the Q31 output vector
 * @param[in]  blockSize  length of the input vector
 */
void arm_float_to_q31(
float32_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief Converts the elements of the floating-point vector to Q15 vector.
 * @param[in]  pSrc       points to the floating-point input vector
 * @param[out] pDst       points to the Q15 output vector
 * @param[in]  blockSize  length of the input vector
 */
void arm_float_to_q15(
float32_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief Converts the elements of the floating-point vector to Q7 vector.
 * @param[in]  pSrc       points to the floating-point input vector
 * @param[out] pDst       points to the Q7 output vector
 * @param[in]  blockSize  length of the input vector
 */
void arm_float_to_q7(
float32_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Converts the elements of the Q31 vector to Q15 vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q31_to_q15(
q31_t * pSrc,
q15_t * pDst,
uint32_t blockSize);


/**
 * @brief  Converts the elements of the Q31 vector to Q7 vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q31_to_q7(
q31_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @brief  Converts the elements of the Q15 vector to floating-point vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q15_to_float(
q15_t * pSrc,
float32_t * pDst,
uint32_t blockSize);


/**
 * @brief  Converts the elements of the Q15 vector to Q31 vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q15_to_q31(
q15_t * pSrc,
q31_t * pDst,
uint32_t blockSize);


/**
 * @brief  Converts the elements of the Q15 vector to Q7 vector.
 * @param[in]  pSrc       is input pointer
 * @param[out] pDst       is output pointer
 * @param[in]  blockSize  is the number of samples to process
 */
void arm_q15_to_q7(
q15_t * pSrc,
q7_t * pDst,
uint32_t blockSize);


/**
 * @ingroup groupInterpolation
 */

/**
 * @defgroup BilinearInterpolate Bilinear Interpolation
 *
 * Bilinear interpolation is an extension of linear interpolation applied to a two dimensional grid.
 * The underlying function <code>f(x, y)</code> is sampled on a regular grid and the interpolation process
 * determines values between the grid points.
 * Bilinear interpolation is equivalent to two step linear interpolation, first in the x-dimension and then in th
 * Bilinear interpolation is often used in image processing to rescale images.
 * The CMSIS DSP library provides bilinear interpolation functions for Q7, Q15, Q31, and floating-point da
 *
 * <b>Algorithm</b>
 * \par
 * The instance structure used by the bilinear interpolation functions describes a two dimensional data tab
 * For floating-point, the instance structure is defined as:
 * <pre>
 *   typedef struct
 *   {
 *     uint16_t numRows;
 *     uint16_t numCols;
 *     float32_t *pData;
 * } arm_bilinear_interp_instance_f32;
 * </pre>
 *
 * \par
 * where <code>numRows</code> specifies the number of rows in the table;
 * <code>numCols</code> specifies the number of columns in the table;
 * and <code>pData</code> points to an array of size <code>numRows*numCols</code> values.
 * The data table <code>pTable</code> is organized in row order and the supplied data values fall on inte
 * That is, table element (x,y) is located at <code>pTable[x + y*numCols]</code> where x and y are integ
 *
 * \par
 * Let <code>(x, y)</code> specify the desired interpolation point.  Then define:
 * <pre>
 *     XF = floor(x)
 *     YF = floor(y)
 * </pre>
 * \par
 * The interpolated output point is computed as:
 * <pre>
 *  f(x, y) = f(XF, YF) * (1-(x-XF)) * (1-(y-YF))
 *           + f(XF+1, YF) * (x-XF)*(1-(y-YF))
 *           + f(XF, YF+1) * (1-(x-XF))*(y-YF)
 *           + f(XF+1, YF+1) * (x-XF)*(y-YF)
 * </pre>
 * Note that the coordinates (x, y) contain integer and fractional components.
 * The integer components specify which portion of the table to use while the
 * fractional components control the interpolation processor.
 *
 * \par
 * if (x,y) are outside of the table boundary, Bilinear interpolation returns zero output.
 */

/**
 * @addtogroup BilinearInterpolate
 * @{
 */



/**
*
* @brief  Floating-point bilinear interpolation.
* @param[in,out] S  points to an instance of the interpolation structure.
* @param[in]     X  interpolation coordinate.
* @param[in]     Y  interpolation coordinate.
* @return out interpolated value.
*/
static __INLINE float32_t arm_bilinear_interp_f32(
const arm_bilinear_interp_instance_f32 * S,
float32_t X,
float32_t Y)
{
  float32_t out;
  float32_t f00, f01, f10, f11;
  float32_t *pData = S->pData;
  int32_t xIndex, yIndex, index;
  float32_t xdiff, ydiff;
  float32_t b1, b2, b3, b4;

  xIndex = (int32_t) X;
  yIndex = (int32_t) Y;

  /* Care taken for table outside boundary */
  /* Returns zero output when values are outside table boundary */
  if(xIndex < 0 || xIndex > (S->numRows - 1) || yIndex < 0 || yIndex > (S->numCols - 1))
  {
    return (0);
  }

  /* Calculation of index for two nearest points in X-direction */
  index = (xIndex - 1) + (yIndex - 1) * S->numCols;


  /* Read two nearest points in X-direction */
  f00 = pData[index];
  f01 = pData[index + 1];

  /* Calculation of index for two nearest points in Y-direction */
  index = (xIndex - 1) + (yIndex) * S->numCols;


  /* Read two nearest points in Y-direction */
  f10 = pData[index];
  f11 = pData[index + 1];

  /* Calculation of intermediate values */
  b1 = f00;
  b2 = f01 - f00;
  b3 = f10 - f00;
  b4 = f00 - f01 - f10 + f11;

  /* Calculation of fractional part in X */
  xdiff = X - xIndex;

  /* Calculation of fractional part in Y */
  ydiff = Y - yIndex;
  /* Calculation of bi-linear interpolated output */
  out = b1 + b2 * xdiff + b3 * ydiff + b4 * xdiff * ydiff;

  /* return to application */
  return (out);
}



/**
*
* @brief  Q31 bilinear interpolation.
* @param[in,out] S  points to an instance of the interpolation structure.
* @param[in]     X  interpolation coordinate in 12.20 format.
* @param[in]     Y  interpolation coordinate in 12.20 format.
* @return out interpolated value.
*/
static __INLINE q31_t arm_bilinear_interp_q31(
arm_bilinear_interp_instance_q31 * S,
q31_t X,
q31_t Y)
{
  q31_t out;                                  /* Temporary output */
  q31_t acc = 0;                              /* output */
  q31_t xfract, yfract;                       /* X, Y fractional parts */
  q31_t x1, x2, y1, y2;                       /* Nearest output values */
  int32_t rI, cI;                             /* Row and column indices */
  q31_t *pYData = S->pData;                   /* pointer to output table values */
  uint32_t nCols = S->numCols;                /* num of rows */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  rI = ((X & (q31_t)0xFFF00000) >> 20);

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  cI = ((Y & (q31_t)0xFFF00000) >> 20);

  /* Care taken for table outside boundary */
  /* Returns zero output when values are outside table boundary */
  if(rI < 0 || rI > (S->numRows - 1) || cI < 0 || cI > (S->numCols - 1))
  {
    return (0);
  }

  /* 20 bits for the fractional part */
  /* shift left xfract by 11 to keep 1.31 format */
  xfract = (X & 0x000FFFFF) << 11u;

  /* Read two nearest output values from the index */
  x1 = pYData[(rI) + (int32_t)nCols * (cI)    ];
  x2 = pYData[(rI) + (int32_t)nCols * (cI) + 1];
  /* 20 bits for the fractional part */
  /* shift left yfract by 11 to keep 1.31 format */
  yfract = (Y & 0x000FFFFF) << 11u;

  /* Read two nearest output values from the index */
  y1 = pYData[(rI) + (int32_t)nCols * (cI + 1)    ];
  y2 = pYData[(rI) + (int32_t)nCols * (cI + 1) + 1];

  /* Calculation of x1 * (1-xfract ) * (1-yfract) and acc is in 3.29(q29) format */
  out = ((q31_t) (((q63_t) x1  * (0x7FFFFFFF - xfract)) >> 32));
  acc = ((q31_t) (((q63_t) out * (0x7FFFFFFF - yfract)) >> 32));

  /* x2 * (xfract) * (1-yfract)  in 3.29(q29) and adding to acc */
  out = ((q31_t) ((q63_t) x2 * (0x7FFFFFFF - yfract) >> 32));
  acc += ((q31_t) ((q63_t) out * (xfract) >> 32));

  /* y1 * (1 - xfract) * (yfract)  in 3.29(q29) and adding to acc */
  out = ((q31_t) ((q63_t) y1 * (0x7FFFFFFF - xfract) >> 32));
  acc += ((q31_t) ((q63_t) out * (yfract) >> 32));

  /* y2 * (xfract) * (yfract)  in 3.29(q29) and adding to acc */
  out = ((q31_t) ((q63_t) y2 * (xfract) >> 32));
  acc += ((q31_t) ((q63_t) out * (yfract) >> 32));

  /* Convert acc to 1.31(q31) format */
  return ((q31_t)(acc << 2));
}


/**
* @brief  Q15 bilinear interpolation.
* @param[in,out] S  points to an instance of the interpolation structure.
* @param[in]     X  interpolation coordinate in 12.20 format.
* @param[in]     Y  interpolation coordinate in 12.20 format.
* @return out interpolated value.
*/
static __INLINE q15_t arm_bilinear_interp_q15(
arm_bilinear_interp_instance_q15 * S,
q31_t X,
q31_t Y)
{
  q63_t acc = 0;                               /* output */
  q31_t out;                                   /* Temporary output */
  q15_t x1, x2, y1, y2;                        /* Nearest output values */
  q31_t xfract, yfract;                        /* X, Y fractional parts */
  int32_t rI, cI;                              /* Row and column indices */
  q15_t *pYData = S->pData;                    /* pointer to output table values */
  uint32_t nCols = S->numCols;                 /* num of rows */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  rI = ((X & (q31_t)0xFFF00000) >> 20);

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  cI = ((Y & (q31_t)0xFFF00000) >> 20);

  /* Care taken for table outside boundary */
  /* Returns zero output when values are outside table boundary */
  if(rI < 0 || rI > (S->numRows - 1) || cI < 0 || cI > (S->numCols - 1))
  {
    return (0);
  }

  /* 20 bits for the fractional part */
  /* xfract should be in 12.20 format */
  xfract = (X & 0x000FFFFF);

  /* Read two nearest output values from the index */
  x1 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI)    ];
  x2 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI) + 1];

  /* 20 bits for the fractional part */
  /* yfract should be in 12.20 format */
  yfract = (Y & 0x000FFFFF);

  /* Read two nearest output values from the index */
  y1 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI + 1)    ];
  y2 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI + 1) + 1];

  /* Calculation of x1 * (1-xfract ) * (1-yfract) and acc is in 13.51 format */

  /* x1 is in 1.15(q15), xfract in 12.20 format and out is in 13.35 format */
  /* convert 13.35 to 13.31 by right shifting  and out is in 1.31 */
  out = (q31_t) (((q63_t) x1 * (0xFFFFF - xfract)) >> 4u);
  acc = ((q63_t) out * (0xFFFFF - yfract));

  /* x2 * (xfract) * (1-yfract)  in 1.51 and adding to acc */
  out = (q31_t) (((q63_t) x2 * (0xFFFFF - yfract)) >> 4u);
  acc += ((q63_t) out * (xfract));

  /* y1 * (1 - xfract) * (yfract)  in 1.51 and adding to acc */
  out = (q31_t) (((q63_t) y1 * (0xFFFFF - xfract)) >> 4u);
  acc += ((q63_t) out * (yfract));

  /* y2 * (xfract) * (yfract)  in 1.51 and adding to acc */
  out = (q31_t) (((q63_t) y2 * (xfract)) >> 4u);
  acc += ((q63_t) out * (yfract));

  /* acc is in 13.51 format and down shift acc by 36 times */
  /* Convert out to 1.15 format */
  return ((q15_t)(acc >> 36));
}
```

```
/**
* @brief  Q7 bilinear interpolation.
* @param[in,out] S  points to an instance of the interpolation structure.
* @param[in]     X  interpolation coordinate in 12.20 format.
* @param[in]     Y  interpolation coordinate in 12.20 format.
* @return out interpolated value.
*/
static __INLINE q7_t arm_bilinear_interp_q7(
arm_bilinear_interp_instance_q7 * S,
q31_t X,
q31_t Y)
{
  q63_t acc = 0;                              /* output */
  q31_t out;                                  /* Temporary output */
  q31_t xfract, yfract;                       /* X, Y fractional parts */
  q7_t x1, x2, y1, y2;                        /* Nearest output values */
  int32_t rI, cI;                             /* Row and column indices */
  q7_t *pYData = S->pData;                    /* pointer to output table values */
  uint32_t nCols = S->numCols;                /* num of rows */

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  rI = ((X & (q31_t)0xFFF00000) >> 20);

  /* Input is in 12.20 format */
  /* 12 bits for the table index */
  /* Index value calculation */
  cI = ((Y & (q31_t)0xFFF00000) >> 20);

  /* Care taken for table outside boundary */
  /* Returns zero output when values are outside table boundary */
  if(rI < 0 || rI > (S->numRows - 1) || cI < 0 || cI > (S->numCols - 1))
  {
    return (0);
  }

  /* 20 bits for the fractional part */
  /* xfract should be in 12.20 format */
  xfract = (X & (q31_t)0x000FFFFF);

  /* Read two nearest output values from the index */
  x1 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI)    ];
  x2 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI) + 1];

  /* 20 bits for the fractional part */
  /* yfract should be in 12.20 format */
  yfract = (Y & (q31_t)0x000FFFFF);

  /* Read two nearest output values from the index */
  y1 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI + 1)    ];
```

```
    y2 = pYData[((uint32_t)rI) + nCols * ((uint32_t)cI + 1) + 1];

    /* Calculation of x1 * (1-xfract ) * (1-yfract) and acc is in 16.47 format */
    out = ((x1 * (0xFFFFF - xfract)));
    acc = (((q63_t) out * (0xFFFFF - yfract)));

    /* x2 * (xfract) * (1-yfract)  in 2.22 and adding to acc */
    out = ((x2 * (0xFFFFF - yfract)));
    acc += (((q63_t) out * (xfract)));

    /* y1 * (1 - xfract) * (yfract)  in 2.22 and adding to acc */
    out = ((y1 * (0xFFFFF - xfract)));
    acc += (((q63_t) out * (yfract)));

    /* y2 * (xfract) * (yfract)  in 2.22 and adding to acc */
    out = ((y2 * (yfract)));
    acc += (((q63_t) out * (xfract)));

    /* acc in 16.47 format and down shift by 40 to convert to 1.7 format */
    return ((q7_t)(acc >> 40));
  }

  /**
   * @} end of BilinearInterpolate group
   */


/* SMMLAR */
#define multAcc_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((((q63_t) a) << 32) + ((q63_t) x * y) + 0x80000000LL ) >> 32)

/* SMMLSR */
#define multSub_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((((q63_t) a) << 32) - ((q63_t) x * y) + 0x80000000LL ) >> 32)

/* SMMULR */
#define mult_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((q63_t) x * y + 0x80000000LL ) >> 32)

/* SMMLA */
#define multAcc_32x32_keep32(a, x, y) \
    a += (q31_t) (((q63_t) x * y) >> 32)

/* SMMLS */
#define multSub_32x32_keep32(a, x, y) \
    a -= (q31_t) (((q63_t) x * y) >> 32)

/* SMMUL */
#define mult_32x32_keep32(a, x, y) \
    a = (q31_t) (((q63_t) x * y ) >> 32)


#if defined ( __CC_ARM )
```

```
  /* Enter low optimization region - place directly above function definition */
  #if defined( ARM_MATH_CM4 ) || defined( ARM_MATH_CM7)
    #define LOW_OPTIMIZATION_ENTER \
       _Pragma ("push")         \
       _Pragma ("O1")
  #else
    #define LOW_OPTIMIZATION_ENTER
  #endif

  /* Exit low optimization region - place directly after end of function definition */
  #if defined( ARM_MATH_CM4 ) || defined( ARM_MATH_CM7)
    #define LOW_OPTIMIZATION_EXIT \
       _Pragma ("pop")
  #else
    #define LOW_OPTIMIZATION_EXIT
  #endif

  /* Enter low optimization region - place directly above function definition */
  #define IAR_ONLY_LOW_OPTIMIZATION_ENTER

  /* Exit low optimization region - place directly after end of function definition */
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#elif defined(__ARMCC_VERSION) && (__ARMCC_VERSION >= 6010050)
  #define LOW_OPTIMIZATION_ENTER
  #define LOW_OPTIMIZATION_EXIT
  #define IAR_ONLY_LOW_OPTIMIZATION_ENTER
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#elif defined(__GNUC__)
  #define LOW_OPTIMIZATION_ENTER __attribute__(( optimize("-O1") ))
  #define LOW_OPTIMIZATION_EXIT
  #define IAR_ONLY_LOW_OPTIMIZATION_ENTER
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#elif defined(__ICCARM__)
  /* Enter low optimization region - place directly above function definition */
  #if defined( ARM_MATH_CM4 ) || defined( ARM_MATH_CM7)
    #define LOW_OPTIMIZATION_ENTER \
       _Pragma ("optimize=low")
  #else
    #define LOW_OPTIMIZATION_ENTER
  #endif

  /* Exit low optimization region - place directly after end of function definition */
  #define LOW_OPTIMIZATION_EXIT

  /* Enter low optimization region - place directly above function definition */
  #if defined( ARM_MATH_CM4 ) || defined( ARM_MATH_CM7)
    #define IAR_ONLY_LOW_OPTIMIZATION_ENTER \
       _Pragma ("optimize=low")
  #else
    #define IAR_ONLY_LOW_OPTIMIZATION_ENTER
  #endif

  /* Exit low optimization region - place directly after end of function definition */
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#elif defined(__CSMC__)
  #define LOW_OPTIMIZATION_ENTER
  #define LOW_OPTIMIZATION_EXIT
  #define IAR_ONLY_LOW_OPTIMIZATION_ENTER
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#elif defined(__TASKING__)
  #define LOW_OPTIMIZATION_ENTER
  #define LOW_OPTIMIZATION_EXIT
  #define IAR_ONLY_LOW_OPTIMIZATION_ENTER
  #define IAR_ONLY_LOW_OPTIMIZATION_EXIT

#endif


#ifdef   __cplusplus
}
#endif


#if defined ( __GNUC__ )
#pragma GCC diagnostic pop
#endif

#endif /* _ARM_MATH_H */

/**
 *
 * End of file.
 */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     cmsis_gcc.h
 * @brief    CMSIS Cortex-M Core Function/Instruction Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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
```

```
#ifndef __CMSIS_GCC_H
#define __CMSIS_GCC_H

/* ignore some GCC warnings */
#if defined ( __GNUC__ )
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wsign-conversion"
#pragma GCC diagnostic ignored "-Wconversion"
#pragma GCC diagnostic ignored "-Wunused-parameter"
#endif


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

/**
  \brief   Enable IRQ Interrupts
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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 >= 0x03U)

/**
  \brief   Enable FIQ
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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 >= 0x03U) */


#if       (__CORTEX_M == 0x04U) || (__CORTEX_M == 0x07U)

/**
  \brief   Get FPSCR
  \details 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 == 1U) && (__FPU_USED == 1U)
  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
  \details 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 == 1U) && (__FPU_USED == 1U)
  /* Empty asm statement works as a scheduling barrier */
  __ASM volatile ("");
  __ASM volatile ("VMSR fpscr, %0" : : "r" (fpscr) : "vfpcc");
  __ASM volatile ("");
#endif
}

#endif /* (__CORTEX_M == 0x04U) || (__CORTEX_M == 0x07U) */



/*@} end of CMSIS_Core_RegAccFunctions */


/* ##########################  Core Instruction Access  ######################### */
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface
  Access to dedicated instructions
  @{
*/

/* 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
  \details 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
  \details Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occ
 */
__attribute__((always_inline)) __STATIC_INLINE void __WFI(void)
{
  __ASM volatile ("wfi");
}
```

```
/**
  \brief   Wait For Event
  \details 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
  \details 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
  \details 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
  \details 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
  \details 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)
  \details 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)
  \details 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 (valu
  return(result);
}


/**
  \brief   Reverse byte order in signed short value
  \details 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
  int32_t result;

  __ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (valu
  return(result);
#endif
}
```

```
/**
  \brief   Rotate Right in unsigned value (32 bit)
  \details Rotate Right (immediate) provides the value of the contents of a register rotated by a variable nur
  \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 << (32U - op2));
}


/**
  \brief   Breakpoint
  \details Causes the processor to enter Debug state.
           Debug tools can use this to investigate system state when the instruction at a particular address is r
  \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
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U)
   __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 >>= 1U; value; value >>= 1U)
  {
    result <<= 1U;
    result |= value & 1U;
    s--;
  }
  result <<= s;                        /* shift when v's highest bits are zero */
#endif
  return(result);
}
```

```
/**
  \brief   Count leading zeros
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U)

/**
  \brief   LDR Exclusive (8 bit)
  \details 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)
  \details 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 ("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)
  \details 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)
  \details 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 ("strexb %0, %2, %1" : "=&r" (result), "=Q" (*addr) : "r" ((uint32_t)value) );
   return(result);
}


/**
  \brief   STR Exclusive (16 bit)
  \details 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 *ad
{
   uint32_t result;

   __ASM volatile ("strexh %0, %2, %1" : "=&r" (result), "=Q" (*addr) : "r" ((uint32_t)value) );
   return(result);
}


/**
  \brief   STR Exclusive (32 bit)
  \details 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 *ad
{
   uint32_t result;

   __ASM volatile ("strex %0, %2, %1" : "=&r" (result), "=Q" (*addr) : "r" (value) );
   return(result);
}


/**
  \brief   Remove the exclusive lock
  \details Removes the exclusive lock which is created by LDREX.
 */
__attribute__((always_inline)) __STATIC_INLINE void __CLREX(void)
{
  __ASM volatile ("clrex" ::: "memory");
}


/**
  \brief   Signed Saturate
  \details 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
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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)
  \details 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 >= 0x03U) || (__CORTEX_SC >= 300U) */

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */


/* ###################  Compiler specific Intrinsics  ########################### */
/** \defgroup CMSIS_SIMD_intrinsics CMSIS SIMD Intrinsics
  Access to dedicated SIMD instructions
  @{
*/

#if (__CORTEX_M >= 0x04U)  /* only for Cortex-M4 and above */

__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
{
  uint32_t result;

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

#define __SSAT16(ARG1,ARG2) \
({                          \
  int32_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
{
  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, uint3
{
  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, uint6
{
  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.w3
#else               /* Big endian */
  __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w3
#endif

  return(llr.w64);
}

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

#ifndef __ARMEB__   /* Little endian */
  __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w
#else               /* Big endian */
  __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w
#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
{
  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, uint3
{
  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, uint6
{
  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.w3
#else               /* Big endian */
  __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w3
#endif

  return(llr.w64);
}

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

#ifndef __ARMEB__   /* Little endian */
  __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w
#else               /* Big endian */
  __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w
#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  int32_t __QADD( int32_t op1,  int32_t op2)
{
  int32_t result;

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

__attribute__( ( always_inline ) ) __STATIC_INLINE  int32_t __QSUB( int32_t op1,  int32_t op2)
{
  int32_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), "I" (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 o
{
 int32_t result;

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

#endif /* (__CORTEX_M >= 0x04) */
/*@} end of group CMSIS_SIMD_intrinsics */


#if defined ( __GNUC__ )
#pragma GCC diagnostic pop
#endif

#endif /* __CMSIS_GCC_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/**************************************************************************//**
 * @file     cmsis_armcc_V6.h
 * @brief    CMSIS Cortex-M Core Function/Instruction Header File
 * @version  V4.30
 * @date     20. October 2015
 ******************************************************************************/
/* 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"
```

```
#ifndef __CMSIS_ARMCC_V6_H
#define __CMSIS_ARMCC_V6_H


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

/**
  \brief   Enable IRQ Interrupts
  \details 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
  \details 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
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Control Register (non-secure)
  \details Returns the content of the non-secure Control Register when in secure mode.
  \return               non-secure Control Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_CONTROL_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, control_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Control Register
  \details 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");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Control Register (non-secure)
  \details Writes the given value to the non-secure Control Register when in secure state.
  \param [in]    control  Control Register value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_CONTROL_NS(uint32_t control)
{
  __ASM volatile ("MSR control_ns, %0" : : "r" (control) : "memory");
}
#endif


/**
  \brief   Get IPSR Register
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get IPSR Register (non-secure)
  \details Returns the content of the non-secure IPSR Register when in secure state.
  \return               IPSR Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_IPSR_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, ipsr_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Get APSR Register
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get APSR Register (non-secure)
  \details Returns the content of the non-secure APSR Register when in secure state.
  \return               APSR Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_APSR_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, apsr_ns" : "=r" (result) );
  return(result);
}
#endif
```

```
/**
  \brief   Get xPSR Register
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get xPSR Register (non-secure)
  \details Returns the content of the non-secure xPSR Register when in secure state.
  \return               xPSR Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_xPSR_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, xpsr_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Get Process Stack Pointer
  \details 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"  : "=r" (result) );
  return(result);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Process Stack Pointer (non-secure)
  \details Returns the current value of the non-secure Process Stack Pointer (PSP) when in secure state.
  \return               PSP Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_PSP_NS(void)
{
  register uint32_t result;
  __ASM volatile ("MRS %0, psp_ns"  : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Process Stack Pointer
  \details 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" : : "r" (topOfProcStack) : "sp");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Process Stack Pointer (non-secure)
  \details Assigns the given value to the non-secure Process Stack Pointer (PSP) when in secure state.
  \param [in]    topOfProcStack  Process Stack Pointer value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_PSP_NS(uint32_t topOfProcStack)
{
  __ASM volatile ("MSR psp_ns, %0" : : "r" (topOfProcStack) : "sp");
}
#endif


/**
  \brief   Get Main Stack Pointer
  \details 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" : "=r" (result) );
  return(result);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Main Stack Pointer (non-secure)
  \details Returns the current value of the non-secure Main Stack Pointer (MSP) when in secure state.
  \return               MSP Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_MSP_NS(void)
{
  register uint32_t result;

  __ASM volatile ("MRS %0, msp_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Main Stack Pointer
  \details 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" : : "r" (topOfMainStack) : "sp");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Main Stack Pointer (non-secure)
  \details Assigns the given value to the non-secure Main Stack Pointer (MSP) when in secure state.
  \param [in]    topOfMainStack  Main Stack Pointer value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_MSP_NS(uint32_t topOfMainStack)
{
  __ASM volatile ("MSR msp_ns, %0" : : "r" (topOfMainStack) : "sp");
}
#endif


/**
  \brief   Get Priority Mask
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Priority Mask (non-secure)
  \details Returns the current state of the non-secure priority mask bit from the Priority Mask Register when
  \return               Priority Mask value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_PRIMASK_NS(void)
```

```
{
  uint32_t result;

  __ASM volatile ("MRS %0, primask_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Priority Mask
  \details 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  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Priority Mask (non-secure)
  \details Assigns the given value to the non-secure Priority Mask Register when in secure state.
  \param [in]    priMask  Priority Mask
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_PRIMASK_NS(uint32_t priMask)
{
  __ASM volatile ("MSR primask_ns, %0" : : "r" (priMask) : "memory");
}
#endif


#if ((__ARM_ARCH_7M__ == 1U) || (__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U))  /* T

/**
  \brief   Enable FIQ
  \details 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
  \details 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
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Base Priority (non-secure)
  \details Returns the current value of the non-secure Base Priority register when in secure state.
  \return               Base Priority register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_BASEPRI_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, basepri_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Base Priority
  \details 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");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Base Priority (non-secure)
  \details Assigns the given value to the non-secure Base Priority register when in secure state.
  \param [in]    basePri  Base Priority value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_BASEPRI_NS(uint32_t value)
{
  __ASM volatile ("MSR basepri_ns, %0" : : "r" (value) : "memory");
}
#endif


/**
  \brief   Set Base Priority with condition
  \details 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");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Base Priority with condition (non_secure)
  \details Assigns the given value to the non-secure Base Priority register when in secure state only if BASE
         or the new value increases the BASEPRI priority level.
  \param [in]    basePri  Base Priority value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_BASEPRI_MAX_NS(uint32_t value)
{
  __ASM volatile ("MSR basepri_max_ns, %0" : : "r" (value) : "memory");
}
#endif


/**
  \brief   Get Fault Mask
  \details 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);
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get Fault Mask (non-secure)
  \details Returns the current value of the non-secure Fault Mask register when in secure state.
  \return               Fault Mask register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_FAULTMASK_NS(void)
{
  uint32_t result;

  __ASM volatile ("MRS %0, faultmask_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Fault Mask
  \details 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");
}


#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Set Fault Mask (non-secure)
  \details Assigns the given value to the non-secure Fault Mask register when in secure state.
  \param [in]    faultMask  Fault Mask value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_FAULTMASK_NS(uint32_t faultMask)
{
  __ASM volatile ("MSR faultmask_ns, %0" : : "r" (faultMask) : "memory");
}
#endif


#endif /* ((__ARM_ARCH_7M__ == 1U) || (__ARM_ARCH_8M__ == 1U)) */


#if (__ARM_ARCH_8M__ == 1U)

/**
  \brief   Get Process Stack Pointer Limit
  \details Returns the current value of the Process Stack Pointer Limit (PSPLIM).
  \return               PSPLIM Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __get_PSPLIM(void)
{
  register uint32_t result;

  __ASM volatile ("MRS %0, psplim"  : "=r" (result) );
  return(result);
}


#if  (__ARM_FEATURE_CMSE == 3U) && (__ARM_ARCH_PROFILE == 'M')     /* ToDo:  ARMCC_V6: ch
/**
```

```
  \brief   Get Process Stack Pointer Limit (non-secure)
  \details Returns the current value of the non-secure Process Stack Pointer Limit (PSPLIM) when in secure
  \return               PSPLIM Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_PSPLIM_NS(void)
{
  register uint32_t result;

  __ASM volatile ("MRS %0, psplim_ns"  : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Process Stack Pointer Limit
  \details Assigns the given value to the Process Stack Pointer Limit (PSPLIM).
  \param [in]    ProcStackPtrLimit  Process Stack Pointer Limit value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __set_PSPLIM(uint32_t ProcStackPtrLimit)
{
  __ASM volatile ("MSR psplim, %0" : : "r" (ProcStackPtrLimit));
}


#if  (__ARM_FEATURE_CMSE == 3U) && (__ARM_ARCH_PROFILE == 'M')     /* ToDo:  ARMCC_V6: ch
/**
  \brief   Set Process Stack Pointer (non-secure)
  \details Assigns the given value to the non-secure Process Stack Pointer Limit (PSPLIM) when in secure
  \param [in]    ProcStackPtrLimit  Process Stack Pointer Limit value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_PSPLIM_NS(uint32_t ProcStackPtrLimit)
{
  __ASM volatile ("MSR psplim_ns, %0\n" : : "r" (ProcStackPtrLimit));
}
#endif


/**
  \brief   Get Main Stack Pointer Limit
  \details Returns the current value of the Main Stack Pointer Limit (MSPLIM).
  \return               MSPLIM Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __get_MSPLIM(void)
{
  register uint32_t result;

  __ASM volatile ("MRS %0, msplim" : "=r" (result) );

  return(result);
}
```

```
#if  (__ARM_FEATURE_CMSE == 3U) && (__ARM_ARCH_PROFILE == 'M')     /* ToDo:  ARMCC_V6: ch
/**
  \brief   Get Main Stack Pointer Limit (non-secure)
  \details Returns the current value of the non-secure Main Stack Pointer Limit(MSPLIM) when in secure st
  \return               MSPLIM Register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_MSPLIM_NS(void)
{
  register uint32_t result;

  __ASM volatile ("MRS %0, msplim_ns" : "=r" (result) );
  return(result);
}
#endif


/**
  \brief   Set Main Stack Pointer Limit
  \details Assigns the given value to the Main Stack Pointer Limit (MSPLIM).
  \param [in]    MainStackPtrLimit  Main Stack Pointer Limit value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __set_MSPLIM(uint32_t MainStackPtrLimit)
{
  __ASM volatile ("MSR msplim, %0" : : "r" (MainStackPtrLimit));
}


#if  (__ARM_FEATURE_CMSE == 3U) && (__ARM_ARCH_PROFILE == 'M')     /* ToDo:  ARMCC_V6: ch
/**
  \brief   Set Main Stack Pointer Limit (non-secure)
  \details Assigns the given value to the non-secure Main Stack Pointer Limit (MSPLIM) when in secure sta
  \param [in]    MainStackPtrLimit  Main Stack Pointer value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_MSPLIM_NS(uint32_t MainStackPtrLimit
{
  __ASM volatile ("MSR msplim_ns, %0" : : "r" (MainStackPtrLimit));
}
#endif

#endif /* (__ARM_ARCH_8M__ == 1U) */


#if ((__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U))  /* ToDo:  ARMCC_V6: check if this i

/**
  \brief   Get FPSCR
  \details eturns the current value of the Floating Point Status/Control register.
  \return               Floating Point Status/Control register value
 */
#define __get_FPSCR      __builtin_arm_get_fpscr
#if 0
__attribute__((always_inline)) __STATIC_INLINE uint32_t __get_FPSCR(void)
{
```

```
#if (__FPU_PRESENT == 1U) && (__FPU_USED == 1U)
  uint32_t result;

  __ASM volatile ("");                                  /* Empty asm statement works as a scheduling barrier */
  __ASM volatile ("VMRS %0, fpscr" : "=r" (result) );
  __ASM volatile ("");
  return(result);
#else
   return(0);
#endif
}
#endif

#if  (__ARM_FEATURE_CMSE == 3U)
/**
  \brief   Get FPSCR (non-secure)
  \details Returns the current value of the non-secure Floating Point Status/Control register when in secure
  \return               Floating Point Status/Control register value
 */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __TZ_get_FPSCR_NS(void)
{
#if (__FPU_PRESENT == 1U) && (__FPU_USED == 1U)
  uint32_t result;

  __ASM volatile ("");                                  /* Empty asm statement works as a scheduling barrier */
  __ASM volatile ("VMRS %0, fpscr_ns" : "=r" (result) );
  __ASM volatile ("");
  return(result);
#else
   return(0);
#endif
}
#endif



/**
  \brief   Set FPSCR
  \details Assigns the given value to the Floating Point Status/Control register.
  \param [in]    fpscr  Floating Point Status/Control value to set
 */
#define __set_FPSCR      __builtin_arm_set_fpscr
#if 0
__attribute__((always_inline)) __STATIC_INLINE void __set_FPSCR(uint32_t fpscr)
{
#if (__FPU_PRESENT == 1U) && (__FPU_USED == 1U)
  __ASM volatile ("");                                  /* Empty asm statement works as a scheduling barrier */
  __ASM volatile ("VMSR fpscr, %0" : : "r" (fpscr) : "vfpcc");
  __ASM volatile ("");
#endif
}
#endif

#if  (__ARM_FEATURE_CMSE == 3U)
```

```
/**
  \brief   Set FPSCR (non-secure)
  \details Assigns the given value to the non-secure Floating Point Status/Control register when in secure s
  \param [in]    fpscr  Floating Point Status/Control value to set
 */
__attribute__((always_inline)) __STATIC_INLINE void __TZ_set_FPSCR_NS(uint32_t fpscr)
{
#if (__FPU_PRESENT == 1U) && (__FPU_USED == 1U)
  __ASM volatile ("");                                /* Empty asm statement works as a scheduling barrier */
  __ASM volatile ("VMSR fpscr_ns, %0" : : "r" (fpscr) : "vfpcc");
  __ASM volatile ("");
#endif
}
#endif

#endif /* ((__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U)) */



/*@} end of CMSIS_Core_RegAccFunctions */


/* ##########################  Core Instruction Access  ######################### */
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface
  Access to dedicated instructions
  @{
*/

/* 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
  \details No Operation does nothing. This instruction can be used for code alignment purposes.
 */
#define __NOP          __builtin_arm_nop

/**
  \brief   Wait For Interrupt
  \details Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occ
 */
#define __WFI          __builtin_arm_wfi


/**
```

```
  \brief   Wait For Event
  \details 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          __builtin_arm_wfe


/**
  \brief   Send Event
  \details Send Event is a hint instruction. It causes an event to be signaled to the CPU.
 */
#define __SEV          __builtin_arm_sev


/**
  \brief   Instruction Synchronization Barrier
  \details 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()        __builtin_arm_isb(0xF);

/**
  \brief   Data Synchronization Barrier
  \details Acts as a special kind of Data Memory Barrier.
           It completes when all explicit memory accesses before this instruction complete.
 */
#define __DSB()        __builtin_arm_dsb(0xF);


/**
  \brief   Data Memory Barrier
  \details Ensures the apparent order of the explicit memory operations before
           and after the instruction, without ensuring their completion.
 */
#define __DMB()        __builtin_arm_dmb(0xF);


/**
  \brief   Reverse byte order (32 bit)
  \details Reverses the byte order in integer value.
  \param [in]    value  Value to reverse
  \return               Reversed value
 */
#define __REV          __builtin_bswap32


/**
  \brief   Reverse byte order (16 bit)
  \details Reverses the byte order in two unsigned short values.
  \param [in]    value  Value to reverse
  \return               Reversed value
 */
```

```
#define __REV16          __builtin_bswap16                        /* ToDo:  ARMCC_V6: check if __builtin_bswa
#if 0
__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 (valu
  return(result);
}
#endif



/**
  \brief   Reverse byte order in signed short value
  \details Reverses the byte order in a signed short value with sign extension to integer.
  \param [in]    value  Value to reverse
  \return               Reversed value
 */
                                                  /* ToDo:  ARMCC_V6: check if __builtin_bswap16 could be used */
__attribute__((always_inline)) __STATIC_INLINE int32_t __REVSH(int32_t value)
{
  int32_t result;

  __ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (valu
  return(result);
}


/**
  \brief   Rotate Right in unsigned value (32 bit)
  \details Rotate Right (immediate) provides the value of the contents of a register rotated by a variable nur
  \param [in]    op1  Value to rotate
  \param [in]    op2  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 << (32U - op2));
}


/**
  \brief   Breakpoint
  \details Causes the processor to enter Debug state.
           Debug tools can use this to investigate system state when the instruction at a particular address is
    \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
  \details Reverses the bit order of the given value.
  \param [in]    value  Value to reverse
  \return               Reversed value
 */
                                                         /* ToDo:  ARMCC_V6: check if __builtin_arm_rbit is supported */
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t value)
{
  uint32_t result;

#if ((__ARM_ARCH_7M__ == 1U) || (__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U))  /* T
   __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 >>= 1U; value; value >>= 1U)
  {
    result <<= 1U;
    result |= value & 1U;
    s--;
  }
  result <<= s;                        /* shift when v's highest bits are zero */
#endif
  return(result);
}


/**
  \brief   Count leading zeros
  \details 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 ((__ARM_ARCH_7M__ == 1U) || (__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U))  /* T

/**
  \brief   LDR Exclusive (8 bit)
  \details 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        (uint8_t)__builtin_arm_ldrex


/**
  \brief   LDR Exclusive (16 bit)
  \details Executes a exclusive LDR instruction for 16 bit values.
  \param [in]    ptr  Pointer to data
  \return        value of type uint16_t at (*ptr)
 */
```

```
#define __LDREXH        (uint16_t)__builtin_arm_ldrex


/**
  \brief   LDR Exclusive (32 bit)
  \details 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        (uint32_t)__builtin_arm_ldrex


/**
  \brief   STR Exclusive (8 bit)
  \details 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        (uint32_t)__builtin_arm_strex


/**
  \brief   STR Exclusive (16 bit)
  \details 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        (uint32_t)__builtin_arm_strex


/**
  \brief   STR Exclusive (32 bit)
  \details 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        (uint32_t)__builtin_arm_strex


/**
  \brief   Remove the exclusive lock
  \details Removes the exclusive lock which is created by LDREX.
 */
#define __CLREX             __builtin_arm_clrex


/**
```

```
  \brief   Signed Saturate
  \details 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             __builtin_arm_ssat*/
#define __SSAT(ARG1,ARG2) \
({                          \
  int32_t __RES, __ARG1 = (ARG1); \
  __ASM ("ssat %0, %1, %2" : "=r" (__RES) :  "I" (ARG2), "r" (__ARG1) ); \
  __RES; \
 })


/**
  \brief   Unsigned Saturate
  \details 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             __builtin_arm_usat
#if 0
#define __USAT(ARG1,ARG2) \
({                          \
  uint32_t __RES, __ARG1 = (ARG1); \
  __ASM ("usat %0, %1, %2" : "=r" (__RES) :  "I" (ARG2), "r" (__ARG1) ); \
  __RES; \
 })
#endif


/**
  \brief   Rotate Right with Extend (32 bit)
  \details 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)
  \details 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 *ptr)
{
    uint32_t result;

   __ASM volatile ("ldrbt %0, %1" : "=r" (result) : "Q" (*ptr) );
   return ((uint8_t) result);    /* Add explicit type cast here */
}


/**
  \brief   LDRT Unprivileged (16 bit)
  \details 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 *ptr)
{
    uint32_t result;

   __ASM volatile ("ldrht %0, %1" : "=r" (result) : "Q" (*ptr) );
   return ((uint16_t) result);    /* Add explicit type cast here */
}


/**
  \brief   LDRT Unprivileged (32 bit)
  \details 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 *ptr)
{
    uint32_t result;

   __ASM volatile ("ldrt %0, %1" : "=r" (result) : "Q" (*ptr) );
   return(result);
}


/**
  \brief   STRT Unprivileged (8 bit)
  \details 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 *ptr)
{
   __ASM volatile ("strbt %1, %0" : "=Q" (*ptr) : "r" ((uint32_t)value) );
}
```

```
/**
  \brief   STRT Unprivileged (16 bit)
  \details 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 *ptr)
{
   __ASM volatile ("strht %1, %0" : "=Q" (*ptr) : "r" ((uint32_t)value) );
}


/**
  \brief   STRT Unprivileged (32 bit)
  \details 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 *ptr)
{
   __ASM volatile ("strt %1, %0" : "=Q" (*ptr) : "r" (value) );
}

#endif /* ((__ARM_ARCH_7M__ == 1U) || (__ARM_ARCH_7EM__ == 1U) || (__ARM_ARCH_8M__ == 1U


#if (__ARM_ARCH_8M__ == 1U)

/**
  \brief   Load-Acquire (8 bit)
  \details Executes a LDAB 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 __LDAB(volatile uint8_t *ptr)
{
    uint32_t result;

   __ASM volatile ("ldab %0, %1" : "=r" (result) : "Q" (*ptr) );
   return ((uint8_t) result);
}


/**
  \brief   Load-Acquire (16 bit)
  \details Executes a LDAH 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 __LDAH(volatile uint16_t *ptr)
{
    uint32_t result;
  __ASM volatile ("ldah %0, %1" : "=r" (result) : "Q" (*ptr) );
  return ((uint16_t) result);
}


/**
  \brief   Load-Acquire (32 bit)
  \details Executes a LDA 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 __LDA(volatile uint32_t *ptr)
{
    uint32_t result;

  __ASM volatile ("lda %0, %1" : "=r" (result) : "Q" (*ptr) );
  return(result);
}


/**
  \brief   Store-Release (8 bit)
  \details Executes a STLB instruction for 8 bit values.
  \param [in]  value  Value to store
  \param [in]    ptr  Pointer to location
 */
__attribute__((always_inline)) __STATIC_INLINE void __STLB(uint8_t value, volatile uint8_t *ptr)
{
  __ASM volatile ("stlb %1, %0" : "=Q" (*ptr) : "r" ((uint32_t)value) );
}


/**
  \brief   Store-Release (16 bit)
  \details Executes a STLH instruction for 16 bit values.
  \param [in]  value  Value to store
  \param [in]    ptr  Pointer to location
 */
__attribute__((always_inline)) __STATIC_INLINE void __STLH(uint16_t value, volatile uint16_t *ptr)
{
  __ASM volatile ("stlh %1, %0" : "=Q" (*ptr) : "r" ((uint32_t)value) );
}


/**
  \brief   Store-Release (32 bit)
  \details Executes a STL instruction for 32 bit values.
  \param [in]  value  Value to store
  \param [in]    ptr  Pointer to location
 */
__attribute__((always_inline)) __STATIC_INLINE void __STL(uint32_t value, volatile uint32_t *ptr)
{
  __ASM volatile ("stl %1, %0" : "=Q" (*ptr) : "r" ((uint32_t)value) );
```

```
}


/**
  \brief   Load-Acquire Exclusive (8 bit)
  \details Executes a LDAB exclusive instruction for 8 bit value.
  \param [in]    ptr  Pointer to data
  \return             value of type uint8_t at (*ptr)
 */
#define     __LDAEXB                 (uint8_t)__builtin_arm_ldaex


/**
  \brief   Load-Acquire Exclusive (16 bit)
  \details Executes a LDAH exclusive instruction for 16 bit values.
  \param [in]    ptr  Pointer to data
  \return        value of type uint16_t at (*ptr)
 */
#define     __LDAEXH                 (uint16_t)__builtin_arm_ldaex


/**
  \brief   Load-Acquire Exclusive (32 bit)
  \details Executes a LDA exclusive instruction for 32 bit values.
  \param [in]    ptr  Pointer to data
  \return        value of type uint32_t at (*ptr)
 */
#define     __LDAEX                  (uint32_t)__builtin_arm_ldaex


/**
  \brief   Store-Release Exclusive (8 bit)
  \details Executes a STLB exclusive 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     __STLEXB                 (uint32_t)__builtin_arm_stlex


/**
  \brief   Store-Release Exclusive (16 bit)
  \details Executes a STLH exclusive 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     __STLEXH                 (uint32_t)__builtin_arm_stlex


/**
```

```
  \brief   Store-Release Exclusive (32 bit)
  \details Executes a STL exclusive 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     __STLEX                  (uint32_t)__builtin_arm_stlex

#endif /* (__ARM_ARCH_8M__ == 1U) */

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */


/* ###################  Compiler specific Intrinsics  ########################### */
/** \defgroup CMSIS_SIMD_intrinsics CMSIS SIMD Intrinsics
  Access to dedicated SIMD instructions
  @{
*/

#if (__ARM_FEATURE_DSP == 1U)        /* ToDo:  ARMCC_V6: This should be ARCH >= ARMv7-M + SIM

__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
{
  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
{
  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_
{
  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_
{
  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.w3
#else               /* Big endian */
  __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w3
#endif

  return(llr.w64);
}

__attribute__((always_inline)) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2, uint64
{
  union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
  } llr;
  llr.w64 = acc;

#ifndef __ARMEB__   /* Little endian */
  __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w
#else               /* Big endian */
  __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w
#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
{
  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_
{
  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_
{
  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.w3
#else               /* Big endian */
  __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w3
#endif

  return(llr.w64);
}

__attribute__((always_inline)) __STATIC_INLINE uint64_t __SMLSLDX (uint32_t op1, uint32_t op2, uint64
{
  union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
  } llr;
  llr.w64 = acc;

#ifndef __ARMEB__   /* Little endian */
  __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w
#else               /* Big endian */
  __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w
#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  int32_t __QADD( int32_t op1,  int32_t op2)
{
  int32_t result;

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

__attribute__((always_inline)) __STATIC_INLINE  int32_t __QSUB( int32_t op1,  int32_t op2)
{
  int32_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), "I" (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);
}

#endif /* (__ARM_FEATURE_DSP == 1U) */
/*@} end of group CMSIS_SIMD_intrinsics */


#endif /* __CMSIS_ARMCC_V6_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
** ###################################################################
**     Version:             rev. 2.10, 2015-05-27
**     Build:              b170228
**
**     Abstract:
**         Chip specific module features.
**
**     Copyright 2016 Freescale Semiconductor, Inc.
**     Copyright 2016-2017 NXP
**     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 the copyright holder 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" 
**    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
**    ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGE
**    (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
**    LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND 
**    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.nxp.com
**    mail:                 support@nxp.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. 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 (2014-01-30)
**        Added single maximum value generation and a constrain to varying feature values that only numbe
**    - rev. 2.2 (2014-07-16)
**        Module access macro module_BASES replaced by module_BASE_PTRS.
**        System initialization and startup updated.
**    - rev. 2.3 (2014-08-22)
**        System initialization updated - default clock config changed.
**    - rev. 2.4 (2014-08-28)
**        Update of startup files - possibility to override DefaultISR added.
**    - rev. 2.5 (2014-10-14)
**        Interrupt INT_LPTimer renamed to INT_LPTMR0.
**    - rev. 2.6 (2015-01-21)
**        Added FSL_FEATURE_SOC_peripheral_COUNT with number of peripheral instances
**    - rev. 2.7 (2015-02-19)
**        Renamed interrupt vector LLW to LLWU.
**    - rev. 2.8 (2015-05-19)
**        FSL_FEATURE_SOC_CAU_COUNT remamed to FSL_FEATURE_SOC_MMCAU_COUNT.
**        Added FSL_FEATURE_SOC_peripheral_COUNT for TRNG and HSADC.
**        Added features for PORT.
**    - rev. 2.9 (2015-05-25)
**        Added FSL_FEATURE_FLASH_PFLASH_START_ADDRESS
**     - rev. 2.10 (2015-05-27)
**        Several USB features added.
**
** ###################################################################
*/

#ifndef _MKL25Z4_FEATURES_H_
#define _MKL25Z4_FEATURES_H_

/* SOC module features */

/* @brief ACMP availability on the SoC. */
#define FSL_FEATURE_SOC_ACMP_COUNT (0)
/* @brief ADC16 availability on the SoC. */
#define FSL_FEATURE_SOC_ADC16_COUNT (1)
/* @brief ADC12 availability on the SoC. */
#define FSL_FEATURE_SOC_ADC12_COUNT (0)
/* @brief AFE availability on the SoC. */
#define FSL_FEATURE_SOC_AFE_COUNT (0)
/* @brief AIPS availability on the SoC. */
#define FSL_FEATURE_SOC_AIPS_COUNT (0)
/* @brief AOI availability on the SoC. */
#define FSL_FEATURE_SOC_AOI_COUNT (0)
/* @brief AXBS availability on the SoC. */
#define FSL_FEATURE_SOC_AXBS_COUNT (0)
/* @brief ASMC availability on the SoC. */
#define FSL_FEATURE_SOC_ASMC_COUNT (0)
/* @brief CADC availability on the SoC. */
#define FSL_FEATURE_SOC_CADC_COUNT (0)
/* @brief FLEXCAN availability on the SoC. */
#define FSL_FEATURE_SOC_FLEXCAN_COUNT (0)
/* @brief MMCAU availability on the SoC. */
#define FSL_FEATURE_SOC_MMCAU_COUNT (0)
/* @brief CMP availability on the SoC. */
#define FSL_FEATURE_SOC_CMP_COUNT (1)
/* @brief CMT availability on the SoC. */
#define FSL_FEATURE_SOC_CMT_COUNT (0)
/* @brief CNC availability on the SoC. */
#define FSL_FEATURE_SOC_CNC_COUNT (0)
/* @brief CRC availability on the SoC. */
#define FSL_FEATURE_SOC_CRC_COUNT (0)
/* @brief DAC availability on the SoC. */
#define FSL_FEATURE_SOC_DAC_COUNT (1)
/* @brief DAC32 availability on the SoC. */
#define FSL_FEATURE_SOC_DAC32_COUNT (0)
/* @brief DCDC availability on the SoC. */
#define FSL_FEATURE_SOC_DCDC_COUNT (0)
/* @brief DDR availability on the SoC. */
#define FSL_FEATURE_SOC_DDR_COUNT (0)
/* @brief DMA availability on the SoC. */
#define FSL_FEATURE_SOC_DMA_COUNT (1)
/* @brief EDMA availability on the SoC. */
```

```
#define FSL_FEATURE_SOC_EDMA_COUNT (0)
/* @brief DMAMUX availability on the SoC. */
#define FSL_FEATURE_SOC_DMAMUX_COUNT (1)
/* @brief DRY availability on the SoC. */
#define FSL_FEATURE_SOC_DRY_COUNT (0)
/* @brief DSPI availability on the SoC. */
#define FSL_FEATURE_SOC_DSPI_COUNT (0)
/* @brief EMVSIM availability on the SoC. */
#define FSL_FEATURE_SOC_EMVSIM_COUNT (0)
/* @brief ENC availability on the SoC. */
#define FSL_FEATURE_SOC_ENC_COUNT (0)
/* @brief ENET availability on the SoC. */
#define FSL_FEATURE_SOC_ENET_COUNT (0)
/* @brief EWM availability on the SoC. */
#define FSL_FEATURE_SOC_EWM_COUNT (0)
/* @brief FB availability on the SoC. */
#define FSL_FEATURE_SOC_FB_COUNT (0)
/* @brief FGPIO availability on the SoC. */
#define FSL_FEATURE_SOC_FGPIO_COUNT (0)
/* @brief FLEXIO availability on the SoC. */
#define FSL_FEATURE_SOC_FLEXIO_COUNT (0)
/* @brief FMC availability on the SoC. */
#define FSL_FEATURE_SOC_FMC_COUNT (0)
/* @brief FSKDT availability on the SoC. */
#define FSL_FEATURE_SOC_FSKDT_COUNT (0)
/* @brief FTFA availability on the SoC. */
#define FSL_FEATURE_SOC_FTFA_COUNT (1)
/* @brief FTFE availability on the SoC. */
#define FSL_FEATURE_SOC_FTFE_COUNT (0)
/* @brief FTFL availability on the SoC. */
#define FSL_FEATURE_SOC_FTFL_COUNT (0)
/* @brief FTM availability on the SoC. */
#define FSL_FEATURE_SOC_FTM_COUNT (0)
/* @brief FTMRA availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRA_COUNT (0)
/* @brief FTMRE availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRE_COUNT (0)
/* @brief FTMRH availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRH_COUNT (0)
/* @brief GPIO availability on the SoC. */
#define FSL_FEATURE_SOC_GPIO_COUNT (5)
/* @brief HSADC availability on the SoC. */
#define FSL_FEATURE_SOC_HSADC_COUNT (0)
/* @brief I2C availability on the SoC. */
#define FSL_FEATURE_SOC_I2C_COUNT (2)
/* @brief I2S availability on the SoC. */
#define FSL_FEATURE_SOC_I2S_COUNT (0)
/* @brief ICS availability on the SoC. */
#define FSL_FEATURE_SOC_ICS_COUNT (0)
/* @brief INTMUX availability on the SoC. */
#define FSL_FEATURE_SOC_INTMUX_COUNT (0)
/* @brief IRQ availability on the SoC. */
#define FSL_FEATURE_SOC_IRQ_COUNT (0)
```

```
/* @brief KBI availability on the SoC. */
#define FSL_FEATURE_SOC_KBI_COUNT (0)
/* @brief SLCD availability on the SoC. */
#define FSL_FEATURE_SOC_SLCD_COUNT (0)
/* @brief LCDC availability on the SoC. */
#define FSL_FEATURE_SOC_LCDC_COUNT (0)
/* @brief LDO availability on the SoC. */
#define FSL_FEATURE_SOC_LDO_COUNT (0)
/* @brief LLWU availability on the SoC. */
#define FSL_FEATURE_SOC_LLWU_COUNT (1)
/* @brief LMEM availability on the SoC. */
#define FSL_FEATURE_SOC_LMEM_COUNT (0)
/* @brief LPI2C availability on the SoC. */
#define FSL_FEATURE_SOC_LPI2C_COUNT (0)
/* @brief LPIT availability on the SoC. */
#define FSL_FEATURE_SOC_LPIT_COUNT (0)
/* @brief LPSCI availability on the SoC. */
#define FSL_FEATURE_SOC_LPSCI_COUNT (1)
/* @brief LPSPI availability on the SoC. */
#define FSL_FEATURE_SOC_LPSPI_COUNT (0)
/* @brief LPTMR availability on the SoC. */
#define FSL_FEATURE_SOC_LPTMR_COUNT (1)
/* @brief LPTPM availability on the SoC. */
#define FSL_FEATURE_SOC_LPTPM_COUNT (0)
/* @brief LPUART availability on the SoC. */
#define FSL_FEATURE_SOC_LPUART_COUNT (0)
/* @brief LTC availability on the SoC. */
#define FSL_FEATURE_SOC_LTC_COUNT (0)
/* @brief MC availability on the SoC. */
#define FSL_FEATURE_SOC_MC_COUNT (0)
/* @brief MCG availability on the SoC. */
#define FSL_FEATURE_SOC_MCG_COUNT (1)
/* @brief MCGLITE availability on the SoC. */
#define FSL_FEATURE_SOC_MCGLITE_COUNT (0)
/* @brief MCM availability on the SoC. */
#define FSL_FEATURE_SOC_MCM_COUNT (1)
/* @brief MMAU availability on the SoC. */
#define FSL_FEATURE_SOC_MMAU_COUNT (0)
/* @brief MMDVSQ availability on the SoC. */
#define FSL_FEATURE_SOC_MMDVSQ_COUNT (0)
/* @brief SYSMPU availability on the SoC. */
#define FSL_FEATURE_SOC_SYSMPU_COUNT (0)
/* @brief MSCAN availability on the SoC. */
#define FSL_FEATURE_SOC_MSCAN_COUNT (0)
/* @brief MSCM availability on the SoC. */
#define FSL_FEATURE_SOC_MSCM_COUNT (0)
/* @brief MTB availability on the SoC. */
#define FSL_FEATURE_SOC_MTB_COUNT (1)
/* @brief MTBDWT availability on the SoC. */
#define FSL_FEATURE_SOC_MTBDWT_COUNT (1)
/* @brief MU availability on the SoC. */
#define FSL_FEATURE_SOC_MU_COUNT (0)
/* @brief NFC availability on the SoC. */
```

```
#define FSL_FEATURE_SOC_NFC_COUNT (0)
/* @brief OPAMP availability on the SoC. */
#define FSL_FEATURE_SOC_OPAMP_COUNT (0)
/* @brief OSC availability on the SoC. */
#define FSL_FEATURE_SOC_OSC_COUNT (1)
/* @brief OSC32 availability on the SoC. */
#define FSL_FEATURE_SOC_OSC32_COUNT (0)
/* @brief OTFAD availability on the SoC. */
#define FSL_FEATURE_SOC_OTFAD_COUNT (0)
/* @brief PDB availability on the SoC. */
#define FSL_FEATURE_SOC_PDB_COUNT (0)
/* @brief PCC availability on the SoC. */
#define FSL_FEATURE_SOC_PCC_COUNT (0)
/* @brief PGA availability on the SoC. */
#define FSL_FEATURE_SOC_PGA_COUNT (0)
/* @brief PIT availability on the SoC. */
#define FSL_FEATURE_SOC_PIT_COUNT (1)
/* @brief PMC availability on the SoC. */
#define FSL_FEATURE_SOC_PMC_COUNT (1)
/* @brief PORT availability on the SoC. */
#define FSL_FEATURE_SOC_PORT_COUNT (5)
/* @brief PWM availability on the SoC. */
#define FSL_FEATURE_SOC_PWM_COUNT (0)
/* @brief PWT availability on the SoC. */
#define FSL_FEATURE_SOC_PWT_COUNT (0)
/* @brief QuadSPI availability on the SoC. */
#define FSL_FEATURE_SOC_QuadSPI_COUNT (0)
/* @brief RCM availability on the SoC. */
#define FSL_FEATURE_SOC_RCM_COUNT (1)
/* @brief RFSYS availability on the SoC. */
#define FSL_FEATURE_SOC_RFSYS_COUNT (0)
/* @brief RFVBAT availability on the SoC. */
#define FSL_FEATURE_SOC_RFVBAT_COUNT (0)
/* @brief RNG availability on the SoC. */
#define FSL_FEATURE_SOC_RNG_COUNT (0)
/* @brief RNGB availability on the SoC. */
#define FSL_FEATURE_SOC_RNGB_COUNT (0)
/* @brief ROM availability on the SoC. */
#define FSL_FEATURE_SOC_ROM_COUNT (1)
/* @brief RSIM availability on the SoC. */
#define FSL_FEATURE_SOC_RSIM_COUNT (0)
/* @brief RTC availability on the SoC. */
#define FSL_FEATURE_SOC_RTC_COUNT (1)
/* @brief SCG availability on the SoC. */
#define FSL_FEATURE_SOC_SCG_COUNT (0)
/* @brief SCI availability on the SoC. */
#define FSL_FEATURE_SOC_SCI_COUNT (0)
/* @brief SDHC availability on the SoC. */
#define FSL_FEATURE_SOC_SDHC_COUNT (0)
/* @brief SDRAM availability on the SoC. */
#define FSL_FEATURE_SOC_SDRAM_COUNT (0)
/* @brief SEMA42 availability on the SoC. */
#define FSL_FEATURE_SOC_SEMA42_COUNT (0)
```

```
/* @brief SIM availability on the SoC. */
#define FSL_FEATURE_SOC_SIM_COUNT (1)
/* @brief SMC availability on the SoC. */
#define FSL_FEATURE_SOC_SMC_COUNT (1)
/* @brief SPI availability on the SoC. */
#define FSL_FEATURE_SOC_SPI_COUNT (2)
/* @brief TMR availability on the SoC. */
#define FSL_FEATURE_SOC_TMR_COUNT (0)
/* @brief TPM availability on the SoC. */
#define FSL_FEATURE_SOC_TPM_COUNT (3)
/* @brief TRGMUX availability on the SoC. */
#define FSL_FEATURE_SOC_TRGMUX_COUNT (0)
/* @brief TRIAMP availability on the SoC. */
#define FSL_FEATURE_SOC_TRIAMP_COUNT (0)
/* @brief TRNG availability on the SoC. */
#define FSL_FEATURE_SOC_TRNG_COUNT (0)
/* @brief TSI availability on the SoC. */
#define FSL_FEATURE_SOC_TSI_COUNT (1)
/* @brief TSTMR availability on the SoC. */
#define FSL_FEATURE_SOC_TSTMR_COUNT (0)
/* @brief UART availability on the SoC. */
#define FSL_FEATURE_SOC_UART_COUNT (2)
/* @brief USB availability on the SoC. */
#define FSL_FEATURE_SOC_USB_COUNT (1)
/* @brief USBDCD availability on the SoC. */
#define FSL_FEATURE_SOC_USBDCD_COUNT (0)
/* @brief USBHS availability on the SoC. */
#define FSL_FEATURE_SOC_USBHS_COUNT (0)
/* @brief USBHSDCD availability on the SoC. */
#define FSL_FEATURE_SOC_USBHSDCD_COUNT (0)
/* @brief USBPHY availability on the SoC. */
#define FSL_FEATURE_SOC_USBPHY_COUNT (0)
/* @brief VREF availability on the SoC. */
#define FSL_FEATURE_SOC_VREF_COUNT (0)
/* @brief WDOG availability on the SoC. */
#define FSL_FEATURE_SOC_WDOG_COUNT (0)
/* @brief XBAR availability on the SoC. */
#define FSL_FEATURE_SOC_XBAR_COUNT (0)
/* @brief XBARA availability on the SoC. */
#define FSL_FEATURE_SOC_XBARA_COUNT (0)
/* @brief XBARB availability on the SoC. */
#define FSL_FEATURE_SOC_XBARB_COUNT (0)
/* @brief XCVR availability on the SoC. */
#define FSL_FEATURE_SOC_XCVR_COUNT (0)
/* @brief XRDC availability on the SoC. */
#define FSL_FEATURE_SOC_XRDC_COUNT (0)
/* @brief ZLL availability on the SoC. */
#define FSL_FEATURE_SOC_ZLL_COUNT (0)

/* ADC16 module features */

/* @brief Has Programmable Gain Amplifier (PGA) in ADC (register PGA). */
#define FSL_FEATURE_ADC16_HAS_PGA (0)
```

```
/* @brief Has PGA chopping control in ADC (bit PGA[PGACHPb] or PGA[PGACHP]). */
#define FSL_FEATURE_ADC16_HAS_PGA_CHOPPING (0)
/* @brief Has PGA offset measurement mode in ADC (bit PGA[PGAOFSM]). */
#define FSL_FEATURE_ADC16_HAS_PGA_OFFSET_MEASUREMENT (0)
/* @brief Has DMA support (bit SC2[DMAEN] or SC4[DMAEN]). */
#define FSL_FEATURE_ADC16_HAS_DMA (1)
/* @brief Has differential mode (bitfield SC1x[DIFF]). */
#define FSL_FEATURE_ADC16_HAS_DIFF_MODE (1)
/* @brief Has FIFO (bit SC4[AFDEP]). */
#define FSL_FEATURE_ADC16_HAS_FIFO (0)
/* @brief FIFO size if available (bitfield SC4[AFDEP]). */
#define FSL_FEATURE_ADC16_FIFO_SIZE (0)
/* @brief Has channel set a/b multiplexor (bitfield CFG2[MUXSEL]). */
#define FSL_FEATURE_ADC16_HAS_MUX_SELECT (1)
/* @brief Has HW trigger masking (bitfield SC5[HTRGMASKE]. */
#define FSL_FEATURE_ADC16_HAS_HW_TRIGGER_MASK (0)
/* @brief Has calibration feature (bit SC3[CAL] and registers CLPx, CLMx). */
#define FSL_FEATURE_ADC16_HAS_CALIBRATION (1)
/* @brief Has HW averaging (bit SC3[AVGE]). */
#define FSL_FEATURE_ADC16_HAS_HW_AVERAGE (1)
/* @brief Has offset correction (register OFS). */
#define FSL_FEATURE_ADC16_HAS_OFFSET_CORRECTION (1)
/* @brief Maximum ADC resolution. */
#define FSL_FEATURE_ADC16_MAX_RESOLUTION (16)
/* @brief Number of SC1x and Rx register pairs (conversion control and result registers). */
#define FSL_FEATURE_ADC16_CONVERSION_CONTROL_COUNT (2)

/* CMP module features */

/* @brief Has Trigger mode in CMP (register bit field CR1[TRIGM]). */
#define FSL_FEATURE_CMP_HAS_TRIGGER_MODE (1)
/* @brief Has Window mode in CMP (register bit field CR1[WE]). */
#define FSL_FEATURE_CMP_HAS_WINDOW_MODE (0)
/* @brief Has External sample supported in CMP (register bit field CR1[SE]). */
#define FSL_FEATURE_CMP_HAS_EXTERNAL_SAMPLE_SUPPORT (0)
/* @brief Has DMA support in CMP (register bit field SCR[DMAEN]). */
#define FSL_FEATURE_CMP_HAS_DMA (1)
/* @brief Has Pass Through mode in CMP (register bit field MUXCR[PSTM]). */
#define FSL_FEATURE_CMP_HAS_PASS_THROUGH_MODE (0)
/* @brief Has DAC Test function in CMP (register DACTEST). */
#define FSL_FEATURE_CMP_HAS_DAC_TEST (0)

/* COP module features */

/* @brief Has the COP Debug Enable bit (COPC[COPDBGEN]) */
#define FSL_FEATURE_COP_HAS_DEBUG_ENABLE (0)
/* @brief Has the COP Stop mode Enable bit (COPC[COPSTPEN]) */
#define FSL_FEATURE_COP_HAS_STOP_ENABLE (0)
/* @brief Has more clock sources like MCGIRC */
#define FSL_FEATURE_COP_HAS_MORE_CLKSRC (0)
/* @brief Has the timeout long and short mode bit (COPC[COPCLKSEL] and COPC[COPCLKS]) */
#define FSL_FEATURE_COP_HAS_LONGTIME_MODE (0)
```

```
/* DAC module features */

/* @brief Define the size of hardware buffer */
#define FSL_FEATURE_DAC_BUFFER_SIZE (2)
/* @brief Define whether the buffer supports watermark event detection or not. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_DETECTION (0)
/* @brief Define whether the buffer supports watermark selection detection or not. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_SELECTION (0)
/* @brief Define whether the buffer supports watermark event 1 word before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_1_WORD (0)
/* @brief Define whether the buffer supports watermark event 2 words before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_2_WORDS (0)
/* @brief Define whether the buffer supports watermark event 3 words before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_3_WORDS (0)
/* @brief Define whether the buffer supports watermark event 4 words before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_4_WORDS (0)
/* @brief Define whether FIFO buffer mode is available or not. */
#define FSL_FEATURE_DAC_HAS_BUFFER_FIFO_MODE (0)
/* @brief Define whether swing buffer mode is available or not.. */
#define FSL_FEATURE_DAC_HAS_BUFFER_SWING_MODE (0)

/* DMA module features */

/* @brief Number of DMA channels. */
#define FSL_FEATURE_DMA_MODULE_CHANNEL (4)
/* @brief Total number of DMA channels on all modules. */
#define FSL_FEATURE_DMA_DMAMUX_CHANNELS (FSL_FEATURE_SOC_DMA_COUNT * 4)

/* DMAMUX module features */

/* @brief Number of DMA channels (related to number of register CHCFGn). */
#define FSL_FEATURE_DMAMUX_MODULE_CHANNEL (4)
/* @brief Total number of DMA channels on all modules. */
#define FSL_FEATURE_DMAMUX_DMAMUX_CHANNELS (FSL_FEATURE_SOC_DMAMUX_COUNT *
/* @brief Has the periodic trigger capability for the triggered DMA channel (register bit CHCFG0[TRIG]). */
#define FSL_FEATURE_DMAMUX_HAS_TRIG (1)

/* FLASH module features */

#if defined(CPU_MKL25Z128VFM4) || defined(CPU_MKL25Z128VFT4) || defined(CPU_MKL25Z128VLH4
    /* @brief Is of type FTFA. */
    #define FSL_FEATURE_FLASH_IS_FTFA (1)
    /* @brief Is of type FTFE. */
    #define FSL_FEATURE_FLASH_IS_FTFE (0)
    /* @brief Is of type FTFL. */
    #define FSL_FEATURE_FLASH_IS_FTFL (0)
    /* @brief Has flags indicating the status of the FlexRAM (register bits FCNFG[EEERDY], FCNFG[RAMR
    #define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)
    /* @brief Has program flash swapping status flag (register bit FCNFG[SWAP]). */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
    /* @brief Has EEPROM region protection (register FEPROT). */
    #define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
    /* @brief Has data flash region protection (register FDPROT). */
```

```
#define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
/* @brief Has flash access control (registers XACCHn, SACCHn, where n is a number, FACSS and FAC
#define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
/* @brief Has flash cache control in FMC module. */
#define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
/* @brief Has flash cache control in MCM module. */
#define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
/* @brief Has flash cache control in MSCM module. */
#define FSL_FEATURE_FLASH_HAS_MSCM_FLASH_CACHE_CONTROLS (0)
/* @brief Has prefetch speculation control in flash, such as kv5x. */
#define FSL_FEATURE_FLASH_PREFETCH_SPECULATION_CONTROL_IN_FLASH (0)
/* @brief P-Flash start address. */
#define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
/* @brief P-Flash block count. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
/* @brief P-Flash block size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (131072)
/* @brief P-Flash sector size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
/* @brief P-Flash write unit size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
/* @brief P-Flash data path width. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
/* @brief P-Flash block swap feature. */
#define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
/* @brief P-Flash protection region count. */
#define FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT (32)
/* @brief Has FlexNVM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
/* @brief FlexNVM start address. (Valid only if FlexNVM is available.) */
#define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
/* @brief FlexNVM block count. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
/* @brief FlexNVM block size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
/* @brief FlexNVM sector size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)
/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)
/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
```

```
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (1)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (1)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x49 Erase All Blocks Unsecure command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD (0)
/* @brief Has 0x4A Read 1s All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x4B Erase All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Read resource command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Read resource command address alignment. */
```

```
#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111 (0xFFFFFFFF)
/* @brief Emulated eeprom size code 0000 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0000 (0xFFFF)
/* @brief Emulated eeprom size code 0001 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0001 (0xFFFF)
/* @brief Emulated eeprom size code 0010 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0010 (0xFFFF)
/* @brief Emulated eeprom size code 0011 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0011 (0xFFFF)
/* @brief Emulated eeprom size code 0100 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0100 (0xFFFF)
/* @brief Emulated eeprom size code 0101 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0101 (0xFFFF)
/* @brief Emulated eeprom size code 0110 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0110 (0xFFFF)
/* @brief Emulated eeprom size code 0111 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0111 (0xFFFF)
/* @brief Emulated eeprom size code 1000 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1000 (0xFFFF)
```

```
    /* @brief Emulated eeprom size code 1001 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1001 (0xFFFF)
    /* @brief Emulated eeprom size code 1010 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1010 (0xFFFF)
    /* @brief Emulated eeprom size code 1011 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1011 (0xFFFF)
    /* @brief Emulated eeprom size code 1100 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1100 (0xFFFF)
    /* @brief Emulated eeprom size code 1101 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1101 (0xFFFF)
    /* @brief Emulated eeprom size code 1110 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1110 (0xFFFF)
    /* @brief Emulated eeprom size code 1111 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1111 (0xFFFF)
#elif defined(CPU_MKL25Z32VFM4) || defined(CPU_MKL25Z32VFT4) || defined(CPU_MKL25Z32VLH4) |
    /* @brief Is of type FTFA. */
    #define FSL_FEATURE_FLASH_IS_FTFA (1)
    /* @brief Is of type FTFE. */
    #define FSL_FEATURE_FLASH_IS_FTFE (0)
    /* @brief Is of type FTFL. */
    #define FSL_FEATURE_FLASH_IS_FTFL (0)
    /* @brief Has flags indicating the status of the FlexRAM (register bits FCNFG[EEERDY], FCNFG[RAMR
    #define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)
    /* @brief Has program flash swapping status flag (register bit FCNFG[SWAP]). */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
    /* @brief Has EEPROM region protection (register FEPROT). */
    #define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
    /* @brief Has data flash region protection (register FDPROT). */
    #define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
    /* @brief Has flash access control (registers XACCHn, SACCHn, where n is a number, FACSS and FAC
    #define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
    /* @brief Has flash cache control in FMC module. */
    #define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
    /* @brief Has flash cache control in MCM module. */
    #define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
    /* @brief Has flash cache control in MSCM module. */
    #define FSL_FEATURE_FLASH_HAS_MSCM_FLASH_CACHE_CONTROLS (0)
    /* @brief Has prefetch speculation control in flash, such as kv5x. */
    #define FSL_FEATURE_FLASH_PREFETCH_SPECULATION_CONTROL_IN_FLASH (0)
    /* @brief P-Flash start address. */
    #define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
    /* @brief P-Flash block count. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
    /* @brief P-Flash block size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (32768)
    /* @brief P-Flash sector size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
    /* @brief P-Flash write unit size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
    /* @brief P-Flash data path width. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
    /* @brief P-Flash block swap feature. */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
```

```
/* @brief P-Flash protection region count. */
#define FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT (32)
/* @brief Has FlexNVM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
/* @brief FlexNVM start address. (Valid only if FlexNVM is available.) */
#define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
/* @brief FlexNVM block count. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
/* @brief FlexNVM block size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
/* @brief FlexNVM sector size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)
/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)
/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (1)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (1)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x49 Erase All Blocks Unsecure command. */
```

```
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD (0)
/* @brief Has 0x4A Read 1s All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x4B Erase All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Read resource command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Read resource command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010 (0xFFFFFFFF)
```

```
    /* @brief FlexNVM partition code 1011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
    #define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011 (0xFFFFFFFF)
    /* @brief FlexNVM partition code 1100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
    #define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100 (0xFFFFFFFF)
    /* @brief FlexNVM partition code 1101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
    #define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101 (0xFFFFFFFF)
    /* @brief FlexNVM partition code 1110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
    #define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110 (0xFFFFFFFF)
    /* @brief FlexNVM partition code 1111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
    #define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111 (0xFFFFFFFF)
    /* @brief Emulated eeprom size code 0000 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0000 (0xFFFF)
    /* @brief Emulated eeprom size code 0001 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0001 (0xFFFF)
    /* @brief Emulated eeprom size code 0010 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0010 (0xFFFF)
    /* @brief Emulated eeprom size code 0011 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0011 (0xFFFF)
    /* @brief Emulated eeprom size code 0100 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0100 (0xFFFF)
    /* @brief Emulated eeprom size code 0101 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0101 (0xFFFF)
    /* @brief Emulated eeprom size code 0110 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0110 (0xFFFF)
    /* @brief Emulated eeprom size code 0111 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0111 (0xFFFF)
    /* @brief Emulated eeprom size code 1000 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1000 (0xFFFF)
    /* @brief Emulated eeprom size code 1001 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1001 (0xFFFF)
    /* @brief Emulated eeprom size code 1010 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1010 (0xFFFF)
    /* @brief Emulated eeprom size code 1011 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1011 (0xFFFF)
    /* @brief Emulated eeprom size code 1100 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1100 (0xFFFF)
    /* @brief Emulated eeprom size code 1101 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1101 (0xFFFF)
    /* @brief Emulated eeprom size code 1110 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1110 (0xFFFF)
    /* @brief Emulated eeprom size code 1111 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1111 (0xFFFF)
#elif defined(CPU_MKL25Z64VFM4) || defined(CPU_MKL25Z64VFT4) || defined(CPU_MKL25Z64VLH4) |
    /* @brief Is of type FTFA. */
    #define FSL_FEATURE_FLASH_IS_FTFA (1)
    /* @brief Is of type FTFE. */
    #define FSL_FEATURE_FLASH_IS_FTFE (0)
    /* @brief Is of type FTFL. */
    #define FSL_FEATURE_FLASH_IS_FTFL (0)
    /* @brief Has flags indicating the status of the FlexRAM (register bits FCNFG[EEERDY], FCNFG[RAMR
    #define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)
    /* @brief Has program flash swapping status flag (register bit FCNFG[SWAP]). */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
```

```
/* @brief Has EEPROM region protection (register FEPROT). */
#define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
/* @brief Has data flash region protection (register FDPROT). */
#define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
/* @brief Has flash access control (registers XACCHn, SACCHn, where n is a number, FACSS and FAC
#define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
/* @brief Has flash cache control in FMC module. */
#define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
/* @brief Has flash cache control in MCM module. */
#define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
/* @brief Has flash cache control in MSCM module. */
#define FSL_FEATURE_FLASH_HAS_MSCM_FLASH_CACHE_CONTROLS (0)
/* @brief Has prefetch speculation control in flash, such as kv5x. */
#define FSL_FEATURE_FLASH_PREFETCH_SPECULATION_CONTROL_IN_FLASH (0)
/* @brief P-Flash start address. */
#define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
/* @brief P-Flash block count. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
/* @brief P-Flash block size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (65536)
/* @brief P-Flash sector size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
/* @brief P-Flash write unit size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
/* @brief P-Flash data path width. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
/* @brief P-Flash block swap feature. */
#define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
/* @brief P-Flash protection region count. */
#define FSL_FEATURE_FLASH_PFLASH_PROTECTION_REGION_COUNT (32)
/* @brief Has FlexNVM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
/* @brief FlexNVM start address. (Valid only if FlexNVM is available.) */
#define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
/* @brief FlexNVM block count. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
/* @brief FlexNVM block size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
/* @brief FlexNVM sector size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)
/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
```

```
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)
/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (1)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (1)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x49 Erase All Blocks Unsecure command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_UNSECURE_CMD (0)
/* @brief Has 0x4A Read 1s All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x4B Erase All Execute-only Segments command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_EXECUTE_ONLY_SEGMENTS_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Read resource command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGMENT (0)
```

```
/* @brief FlexNVM Rrogram/Verify section command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Read resource command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110 (0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1001 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1011 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1100 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1101 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1110 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110 (0xFFFFFFFF)
/* @brief FlexNVM partition code 1111 mapping to data flash size in bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111 (0xFFFFFFFF)
/* @brief Emulated eeprom size code 0000 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0000 (0xFFFF)
/* @brief Emulated eeprom size code 0001 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0001 (0xFFFF)
/* @brief Emulated eeprom size code 0010 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0010 (0xFFFF)
/* @brief Emulated eeprom size code 0011 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0011 (0xFFFF)
/* @brief Emulated eeprom size code 0100 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0100 (0xFFFF)
/* @brief Emulated eeprom size code 0101 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0101 (0xFFFF)
/* @brief Emulated eeprom size code 0110 mapping to emulated eeprom size in bytes (0xFFFF = reser
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0110 (0xFFFF)
/* @brief Emulated eeprom size code 0111 mapping to emulated eeprom size in bytes (0xFFFF = reser
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0111 (0xFFFF)
    /* @brief Emulated eeprom size code 1000 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1000 (0xFFFF)
    /* @brief Emulated eeprom size code 1001 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1001 (0xFFFF)
    /* @brief Emulated eeprom size code 1010 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1010 (0xFFFF)
    /* @brief Emulated eeprom size code 1011 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1011 (0xFFFF)
    /* @brief Emulated eeprom size code 1100 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1100 (0xFFFF)
    /* @brief Emulated eeprom size code 1101 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1101 (0xFFFF)
    /* @brief Emulated eeprom size code 1110 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1110 (0xFFFF)
    /* @brief Emulated eeprom size code 1111 mapping to emulated eeprom size in bytes (0xFFFF = reserv
    #define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1111 (0xFFFF)
#endif /* defined(CPU_MKL25Z128VFM4) || defined(CPU_MKL25Z128VFT4) || defined(CPU_MKL25Z128

/* GPIO module features */

/* @brief Has fast (single cycle) access capability via a dedicated memory region. */
#define FSL_FEATURE_GPIO_HAS_FAST_GPIO (1)
/* @brief Has port input disable register (PIDR). */
#define FSL_FEATURE_GPIO_HAS_INPUT_DISABLE (0)
/* @brief Has dedicated interrupt vector. */
#define FSL_FEATURE_GPIO_HAS_PORT_INTERRUPT_VECTOR (1)

/* I2C module features */

/* @brief Has System Management Bus support (registers SMB, A2, SLTL and SLTH). */
#define FSL_FEATURE_I2C_HAS_SMBUS (1)
/* @brief Maximum supported baud rate in kilobit per second. */
#define FSL_FEATURE_I2C_MAX_BAUD_KBPS (400)
/* @brief Is affected by errata with ID 6070 (repeat start cannot be generated if the F[MULT] bit field is set t
#define FSL_FEATURE_I2C_HAS_ERRATA_6070 (1)
/* @brief Has DMA support (register bit C1[DMAEN]). */
#define FSL_FEATURE_I2C_HAS_DMA_SUPPORT (1)
/* @brief Has I2C bus start and stop detection (register bits FLT[SSIE], FLT[STARTF] and FLT[STOPF]). *
#define FSL_FEATURE_I2C_HAS_START_STOP_DETECT (0)
/* @brief Has I2C bus stop detection (register bits FLT[STOPIE] and FLT[STOPF]). */
#define FSL_FEATURE_I2C_HAS_STOP_DETECT (1)
/* @brief Has I2C bus stop hold off (register bit FLT[SHEN]). */
#define FSL_FEATURE_I2C_HAS_STOP_HOLD_OFF (1)
/* @brief Maximum width of the glitch filter in number of bus clocks. */
#define FSL_FEATURE_I2C_MAX_GLITCH_FILTER_WIDTH (31)
/* @brief Has control of the drive capability of the I2C pins. */
#define FSL_FEATURE_I2C_HAS_HIGH_DRIVE_SELECTION (1)
/* @brief Has double buffering support (register S2). */
#define FSL_FEATURE_I2C_HAS_DOUBLE_BUFFERING (0)
/* @brief Has double buffer enable. */
#define FSL_FEATURE_I2C_HAS_DOUBLE_BUFFER_ENABLE (0)
```

```
/* LLWU module features */

/* @brief Maximum number of pins (maximal index plus one) connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN (16)
/* @brief Has pins 8-15 connected to LLWU device. */
#define FSL_FEATURE_LLWU_EXTERNAL_PIN_GROUP2 (1)
/* @brief Maximum number of internal modules connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE (8)
/* @brief Number of digital filters. */
#define FSL_FEATURE_LLWU_HAS_PIN_FILTER (2)
/* @brief Has MF register. */
#define FSL_FEATURE_LLWU_HAS_MF (0)
/* @brief Has PF register. */
#define FSL_FEATURE_LLWU_HAS_PF (0)
/* @brief Has possibility to enable reset in low leakage power mode and enable digital filter for RESET pin
#define FSL_FEATURE_LLWU_HAS_RESET_ENABLE (0)
/* @brief Has no internal module wakeup flag register. */
#define FSL_FEATURE_LLWU_HAS_NO_INTERNAL_MODULE_WAKEUP_FLAG_REG (0)
/* @brief Has external pin 0 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN0 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN0_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN0_GPIO_PIN (0)
/* @brief Has external pin 1 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN1 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN1_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN1_GPIO_PIN (0)
/* @brief Has external pin 2 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN2 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN2_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN2_GPIO_PIN (0)
/* @brief Has external pin 3 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN3 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN3_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN3_GPIO_PIN (0)
/* @brief Has external pin 4 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN4 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN4_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN4_GPIO_PIN (0)
/* @brief Has external pin 5 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN5 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN5_GPIO_IDX (GPIOB_IDX)
/* @brief Number of external pin port on specified port. */
```

```
#define FSL_FEATURE_LLWU_PIN5_GPIO_PIN (0)
/* @brief Has external pin 6 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN6 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN6_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN6_GPIO_PIN (1)
/* @brief Has external pin 7 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN7 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN7_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN7_GPIO_PIN (3)
/* @brief Has external pin 8 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN8 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN8_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN8_GPIO_PIN (4)
/* @brief Has external pin 9 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN9 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN9_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN9_GPIO_PIN (5)
/* @brief Has external pin 10 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN10 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN10_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN10_GPIO_PIN (6)
/* @brief Has external pin 11 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN11 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN11_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN11_GPIO_PIN (0)
/* @brief Has external pin 12 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN12 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN12_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN12_GPIO_PIN (0)
/* @brief Has external pin 13 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN13 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN13_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN13_GPIO_PIN (0)
/* @brief Has external pin 14 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN14 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN14_GPIO_IDX (GPIOD_IDX)
```

```
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN14_GPIO_PIN (4)
/* @brief Has external pin 15 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN15 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN15_GPIO_IDX (GPIOD_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN15_GPIO_PIN (6)
/* @brief Has external pin 16 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN16 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN16_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN16_GPIO_PIN (0)
/* @brief Has external pin 17 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN17 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN17_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN17_GPIO_PIN (0)
/* @brief Has external pin 18 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN18 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN18_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN18_GPIO_PIN (0)
/* @brief Has external pin 19 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN19 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN19_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN19_GPIO_PIN (0)
/* @brief Has external pin 20 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN20 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN20_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN20_GPIO_PIN (0)
/* @brief Has external pin 21 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN21 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN21_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN21_GPIO_PIN (0)
/* @brief Has external pin 22 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN22 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN22_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN22_GPIO_PIN (0)
/* @brief Has external pin 23 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN23 (0)
/* @brief Index of port of external pin. */
```

```
#define FSL_FEATURE_LLWU_PIN23_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN23_GPIO_PIN (0)
/* @brief Has external pin 24 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN24 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN24_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN24_GPIO_PIN (0)
/* @brief Has external pin 25 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN25 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN25_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN25_GPIO_PIN (0)
/* @brief Has external pin 26 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN26 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN26_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN26_GPIO_PIN (0)
/* @brief Has external pin 27 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN27 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN27_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN27_GPIO_PIN (0)
/* @brief Has external pin 28 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN28 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN28_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN28_GPIO_PIN (0)
/* @brief Has external pin 29 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN29 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN29_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN29_GPIO_PIN (0)
/* @brief Has external pin 30 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN30 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN30_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN30_GPIO_PIN (0)
/* @brief Has external pin 31 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN31 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN31_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN31_GPIO_PIN (0)
/* @brief Has internal module 0 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE0 (1)
```

```
/* @brief Has internal module 1 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE1 (1)
/* @brief Has internal module 2 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE2 (0)
/* @brief Has internal module 3 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE3 (0)
/* @brief Has internal module 4 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE4 (1)
/* @brief Has internal module 5 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE5 (1)
/* @brief Has internal module 6 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE6 (0)
/* @brief Has internal module 7 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE7 (1)
/* @brief Has Version ID Register (LLWU_VERID). */
#define FSL_FEATURE_LLWU_HAS_VERID (0)
/* @brief Has Parameter Register (LLWU_PARAM). */
#define FSL_FEATURE_LLWU_HAS_PARAM (0)
/* @brief Width of registers of the LLWU. */
#define FSL_FEATURE_LLWU_REG_BITWIDTH (8)
/* @brief Has DMA Enable register (LLWU_DE). */
#define FSL_FEATURE_LLWU_HAS_DMA_ENABLE_REG (0)

/* LPTMR module features */

/* @brief Has shared interrupt handler with another LPTMR module. */
#define FSL_FEATURE_LPTMR_HAS_SHARED_IRQ_HANDLER (0)
/* @brief Whether LPTMR counter is 32 bits width. */
#define FSL_FEATURE_LPTMR_CNR_WIDTH_IS_32B (0)
/* @brief Has timer DMA request enable (register bit CSR[TDRE]). */
#define FSL_FEATURE_LPTMR_HAS_CSR_TDRE (0)

/* MCG module features */

/* @brief PRDIV base value (divider of register bit field [PRDIV] zero value). */
#define FSL_FEATURE_MCG_PLL_PRDIV_BASE (1)
/* @brief Maximum PLL external reference divider value (max. value of register bit field C5[PRVDIV]). */
#define FSL_FEATURE_MCG_PLL_PRDIV_MAX (24)
/* @brief VCO divider base value (multiply factor of register bit field C6[VDIV] zero value). */
#define FSL_FEATURE_MCG_PLL_VDIV_BASE (24)
/* @brief PLL reference clock low range. OSCCLK/PLL_R. */
#define FSL_FEATURE_MCG_PLL_REF_MIN (2000000)
/* @brief PLL reference clock high range. OSCCLK/PLL_R. */
#define FSL_FEATURE_MCG_PLL_REF_MAX (4000000)
/* @brief The PLL clock is divided by 2 before VCO divider. */
#define FSL_FEATURE_MCG_HAS_PLL_INTERNAL_DIV (0)
/* @brief FRDIV supports 1280. */
#define FSL_FEATURE_MCG_FRDIV_SUPPORT_1280 (1)
/* @brief FRDIV supports 1536. */
#define FSL_FEATURE_MCG_FRDIV_SUPPORT_1536 (1)
/* @brief MCGFFCLK divider. */
#define FSL_FEATURE_MCG_FFCLK_DIV (1)
/* @brief Is PLL clock divided by 2 before MCG PLL/FLL clock selection in the SIM module. */
```

```
#define FSL_FEATURE_MCG_HAS_PLL_EXTRA_DIV (1)
/* @brief Has 32kHz RTC external reference clock (register bits C8[LOCS1], C8[CME1], C8[LOCRE1] and
#define FSL_FEATURE_MCG_HAS_RTC_32K (0)
/* @brief Has PLL1 external reference clock (registers C10, C11, C12, S2). */
#define FSL_FEATURE_MCG_HAS_PLL1 (0)
/* @brief Has 48MHz internal oscillator. */
#define FSL_FEATURE_MCG_HAS_IRC_48M (0)
/* @brief Has OSC1 external oscillator (registers C10, C11, C12, S2). */
#define FSL_FEATURE_MCG_HAS_OSC1 (0)
/* @brief Has fast internal reference clock fine trim (register bit C2[FCFTRIM]). */
#define FSL_FEATURE_MCG_HAS_FCFTRIM (0)
/* @brief Has PLL loss of lock reset (register bit C8[LOLRE]). */
#define FSL_FEATURE_MCG_HAS_LOLRE (1)
/* @brief Has MCG OSC clock selection (register bit C7[OSCSEL]). */
#define FSL_FEATURE_MCG_USE_OSCSEL (0)
/* @brief Has PLL external reference selection (register bits C5[PLLREFSEL0] and C11[PLLREFSEL1]). */
#define FSL_FEATURE_MCG_USE_PLLREFSEL (0)
/* @brief TBD */
#define FSL_FEATURE_MCG_USE_SYSTEM_CLOCK (0)
/* @brief Has phase-locked loop (PLL) (register C5 and bits C6[VDIV], C6[PLLS], C6[LOLIE0], S[PLLST], S
#define FSL_FEATURE_MCG_HAS_PLL (1)
/* @brief Has phase-locked loop (PLL) PRDIV (register C5[PRDIV]. */
#define FSL_FEATURE_MCG_HAS_PLL_PRDIV (1)
/* @brief Has phase-locked loop (PLL) VDIV (register C6[VDIV]. */
#define FSL_FEATURE_MCG_HAS_PLL_VDIV (1)
/* @brief PLL/OSC related register bit fields have PLL/OSC index in their name. */
#define FSL_FEATURE_MCG_HAS_PLL_OSC_INDEX (1)
/* @brief Has frequency-locked loop (FLL) (register ATCVH, ATCVL and bits C1[IREFS], C1[FRDIV]). */
#define FSL_FEATURE_MCG_HAS_FLL (1)
/* @brief Has PLL external to MCG (C9[PLL_CME], C9[PLL_LOCRE], C9[EXT_PLL_LOCS]). */
#define FSL_FEATURE_MCG_HAS_EXTERNAL_PLL (0)
/* @brief Has crystal oscillator or external reference clock low power controls (register bits C2[HGO], C2[R
#define FSL_FEATURE_MCG_HAS_EXT_REF_LOW_POWER_CONTROL (1)
/* @brief Has PLL/FLL selection as MCG output (register bit C6[PLLS]). */
#define FSL_FEATURE_MCG_HAS_PLL_FLL_SELECTION (1)
/* @brief Has PLL output selection (PLL0/PLL1, PLL/external PLL) (register bit C11[PLLCS]). */
#define FSL_FEATURE_MCG_HAS_PLL_OUTPUT_SELECTION (0)
/* @brief Has automatic trim machine (registers ATCVH, ATCVL and bits SC[ATMF], SC[ATMS], SC[ATME
#define FSL_FEATURE_MCG_HAS_AUTO_TRIM_MACHINE (1)
/* @brief Has external clock monitor (register bit C6[CME]). */
#define FSL_FEATURE_MCG_HAS_EXTERNAL_CLOCK_MONITOR (1)
/* @brief Has low frequency internal reference clock (IRC) (registers LTRIMRNG, LFRIM, LSTRIM and bit
#define FSL_FEATURE_MCG_HAS_LOW_FREQ_IRC (0)
/* @brief Has high frequency internal reference clock (IRC) (registers HCTRIM, HTTRIM, HFTRIM and bit
#define FSL_FEATURE_MCG_HAS_HIGH_FREQ_IRC (0)
/* @brief Has PEI mode or PBI mode. */
#define FSL_FEATURE_MCG_HAS_PLL_INTERNAL_MODE (0)
/* @brief Reset clock mode is BLPI. */
#define FSL_FEATURE_MCG_RESET_IS_BLPI (0)

/* interrupt module features */

/* @brief Lowest interrupt request number. */
```

```
#define FSL_FEATURE_INTERRUPT_IRQ_MIN (-14)
/* @brief Highest interrupt request number. */
#define FSL_FEATURE_INTERRUPT_IRQ_MAX (31)

/* OSC module features */

/* @brief Has OSC1 external oscillator. */
#define FSL_FEATURE_OSC_HAS_OSC1 (0)
/* @brief Has OSC0 external oscillator. */
#define FSL_FEATURE_OSC_HAS_OSC0 (1)
/* @brief Has OSC external oscillator (without index). */
#define FSL_FEATURE_OSC_HAS_OSC (0)
/* @brief Number of OSC external oscillators. */
#define FSL_FEATURE_OSC_OSC_COUNT (1)
/* @brief Has external reference clock divider (register bit field DIV[ERPS]). */
#define FSL_FEATURE_OSC_HAS_EXT_REF_CLOCK_DIVIDER (0)

/* PIT module features */

/* @brief Number of channels (related to number of registers LDVALn, CVALn, TCTRLn, TFLGn). */
#define FSL_FEATURE_PIT_TIMER_COUNT (2)
/* @brief Has lifetime timer (related to existence of registers LTMR64L and LTMR64H). */
#define FSL_FEATURE_PIT_HAS_LIFETIME_TIMER (1)
/* @brief Has chain mode (related to existence of register bit field TCTRLn[CHN]). */
#define FSL_FEATURE_PIT_HAS_CHAIN_MODE (1)
/* @brief Has shared interrupt handler (has not individual interrupt handler for each channel). */
#define FSL_FEATURE_PIT_HAS_SHARED_IRQ_HANDLER (1)

/* PMC module features */

/* @brief Has Bandgap Enable In VLPx Operation support. */
#define FSL_FEATURE_PMC_HAS_BGEN (1)
/* @brief Has Bandgap Buffer Enable. */
#define FSL_FEATURE_PMC_HAS_BGBE (1)
/* @brief Has Bandgap Buffer Drive Select. */
#define FSL_FEATURE_PMC_HAS_BGBDS (0)
/* @brief Has Low-Voltage Detect Voltage Select support. */
#define FSL_FEATURE_PMC_HAS_LVDV (1)
/* @brief Has Low-Voltage Warning Voltage Select support. */
#define FSL_FEATURE_PMC_HAS_LVWV (1)
/* @brief Has LPO. */
#define FSL_FEATURE_PMC_HAS_LPO (0)
/* @brief Has VLPx option PMC_REGSC[VLPO]. */
#define FSL_FEATURE_PMC_HAS_VLPO (0)
/* @brief Has acknowledge isolation support. */
#define FSL_FEATURE_PMC_HAS_ACKISO (1)
/* @brief Has Regulator In Full Performance Mode Status Bit PMC_REGSC[REGFPM]. */
#define FSL_FEATURE_PMC_HAS_REGFPM (0)
/* @brief Has Regulator In Run Regulation Status Bit PMC_REGSC[REGONS]. */
#define FSL_FEATURE_PMC_HAS_REGONS (1)
/* @brief Has PMC_HVDSC1. */
#define FSL_FEATURE_PMC_HAS_HVDSC1 (0)
/* @brief Has PMC_PARAM. */
```

```
#define FSL_FEATURE_PMC_HAS_PARAM (0)
/* @brief Has PMC_VERID. */
#define FSL_FEATURE_PMC_HAS_VERID (0)

/* PORT module features */

/* @brief Has control lock (register bit PCR[LK]). */
#define FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK (0)
/* @brief Has open drain control (register bit PCR[ODE]). */
#define FSL_FEATURE_PORT_HAS_OPEN_DRAIN (0)
/* @brief Has digital filter (registers DFER, DFCR and DFWR). */
#define FSL_FEATURE_PORT_HAS_DIGITAL_FILTER (0)
/* @brief Has DMA request (register bit field PCR[IRQC] values). */
#define FSL_FEATURE_PORT_HAS_DMA_REQUEST (1)
/* @brief Has pull resistor selection available. */
#define FSL_FEATURE_PORT_HAS_PULL_SELECTION (0)
/* @brief Has pull resistor enable (register bit PCR[PE]). */
#define FSL_FEATURE_PORT_HAS_PULL_ENABLE (1)
/* @brief Has slew rate control (register bit PCR[SRE]). */
#define FSL_FEATURE_PORT_HAS_SLEW_RATE (1)
/* @brief Has passive filter (register bit field PCR[PFE]). */
#define FSL_FEATURE_PORT_HAS_PASSIVE_FILTER (1)
/* @brief Has drive strength control (register bit PCR[DSE]). */
#define FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH (1)
/* @brief Has separate drive strength register (HDRVE). */
#define FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH_REGISTER (0)
/* @brief Has glitch filter (register IOFLT). */
#define FSL_FEATURE_PORT_HAS_GLITCH_FILTER (0)
/* @brief Defines width of PCR[MUX] field. */
#define FSL_FEATURE_PORT_PCR_MUX_WIDTH (3)
/* @brief Has dedicated interrupt vector. */
#define FSL_FEATURE_PORT_HAS_INTERRUPT_VECTOR (1)
/* @brief Has multiple pin IRQ configuration (register GICLR and GICHR). */
#define FSL_FEATURE_PORT_HAS_MULTIPLE_IRQ_CONFIG (0)
/* @brief Defines whether PCR[IRQC] bit-field has flag states. */
#define FSL_FEATURE_PORT_HAS_IRQC_FLAG (0)
/* @brief Defines whether PCR[IRQC] bit-field has trigger states. */
#define FSL_FEATURE_PORT_HAS_IRQC_TRIGGER (0)

/* RCM module features */

/* @brief Has Loss-of-Lock Reset support. */
#define FSL_FEATURE_RCM_HAS_LOL (1)
/* @brief Has Loss-of-Clock Reset support. */
#define FSL_FEATURE_RCM_HAS_LOC (1)
/* @brief Has JTAG generated Reset support. */
#define FSL_FEATURE_RCM_HAS_JTAG (0)
/* @brief Has EzPort generated Reset support. */
#define FSL_FEATURE_RCM_HAS_EZPORT (0)
/* @brief Has bit-field indicating EZP_MS_B pin state during last reset. */
#define FSL_FEATURE_RCM_HAS_EZPMS (0)
/* @brief Has boot ROM configuration, MR[BOOTROM], FM[FORCEROM] */
#define FSL_FEATURE_RCM_HAS_BOOTROM (0)
```

```
/* @brief Has sticky system reset status register RCM_SSRS0 and RCM_SSRS1. */
#define FSL_FEATURE_RCM_HAS_SSRS (0)
/* @brief Has Version ID Register (RCM_VERID). */
#define FSL_FEATURE_RCM_HAS_VERID (0)
/* @brief Has Parameter Register (RCM_PARAM). */
#define FSL_FEATURE_RCM_HAS_PARAM (0)
/* @brief Has Reset Interrupt Enable Register RCM_SRIE. */
#define FSL_FEATURE_RCM_HAS_SRIE (0)
/* @brief Width of registers of the RCM. */
#define FSL_FEATURE_RCM_REG_WIDTH (8)
/* @brief Has Core 1 generated Reset support RCM_SRS[CORE1] */
#define FSL_FEATURE_RCM_HAS_CORE1 (0)
/* @brief Has MDM-AP system reset support RCM_SRS1[MDM_AP] */
#define FSL_FEATURE_RCM_HAS_MDM_AP (1)
/* @brief Has wakeup reset feature. Register bit SRS[WAKEUP]. */
#define FSL_FEATURE_RCM_HAS_WAKEUP (1)

/* RTC module features */

/* @brief Has wakeup pin. */
#define FSL_FEATURE_RTC_HAS_WAKEUP_PIN (1)
/* @brief Has wakeup pin selection (bit field CR[WPS]). */
#define FSL_FEATURE_RTC_HAS_WAKEUP_PIN_SELECTION (0)
/* @brief Has low power features (registers MER, MCLR and MCHR). */
#define FSL_FEATURE_RTC_HAS_MONOTONIC (0)
/* @brief Has read/write access control (registers WAR and RAR). */
#define FSL_FEATURE_RTC_HAS_ACCESS_CONTROL (0)
/* @brief Has security features (registers TTSR, MER, MCLR and MCHR). */
#define FSL_FEATURE_RTC_HAS_SECURITY (0)
/* @brief Has RTC_CLKIN available. */
#define FSL_FEATURE_RTC_HAS_RTC_CLKIN (1)
/* @brief Has prescaler adjust for LPO. */
#define FSL_FEATURE_RTC_HAS_LPO_ADJUST (0)
/* @brief Has Clock Pin Enable field. */
#define FSL_FEATURE_RTC_HAS_CPE (0)
/* @brief Has Timer Seconds Interrupt Configuration field. */
#define FSL_FEATURE_RTC_HAS_TSIC (0)
/* @brief Has OSC capacitor setting RTC_CR[SC2P ~ SC16P] */
#define FSL_FEATURE_RTC_HAS_OSC_SCXP (1)

/* SIM module features */

/* @brief Has USB FS divider. */
#define FSL_FEATURE_SIM_USBFS_USE_SPECIAL_DIVIDER (0)
/* @brief Is PLL clock divided by 2 before MCG PLL/FLL clock selection. */
#define FSL_FEATURE_SIM_PLLCLK_USE_SPECIAL_DIVIDER (1)
/* @brief Has RAM size specification (register bit field SOPT1[RAMSIZE]). */
#define FSL_FEATURE_SIM_OPT_HAS_RAMSIZE (0)
/* @brief Has 32k oscillator clock output (register bit SOPT1[OSC32KOUT]). */
#define FSL_FEATURE_SIM_OPT_HAS_OSC32K_OUT (0)
/* @brief Has 32k oscillator clock selection (register bit field SOPT1[OSC32KSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_OSC32K_SELECTION (1)
/* @brief 32k oscillator clock selection width (width of register bit field SOPT1[OSC32KSEL]). */
```

```
#define FSL_FEATURE_SIM_OPT_OSC32K_SELECTION_WIDTH (2)
/* @brief Has RTC clock output selection (register bit SOPT2[RTCCLKOUTSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_RTC_CLOCK_OUT_SELECTION (1)
/* @brief Has USB voltage regulator (register bits SOPT1[USBVSTBY], SOPT1[USBSSTBY], SOPT1[USB
#define FSL_FEATURE_SIM_OPT_HAS_USB_VOLTAGE_REGULATOR (1)
/* @brief USB has integrated PHY (register bits USBPHYCTL[USBVREGSEL], USBPHYCTL[USBVREGP
#define FSL_FEATURE_SIM_OPT_HAS_USB_PHY (0)
/* @brief Has PTD7 pad drive strength control (register bit SOPT2[PTD7PAD]). */
#define FSL_FEATURE_SIM_OPT_HAS_PTD7PAD (0)
/* @brief Has FlexBus security level selection (register bit SOPT2[FBSL]). */
#define FSL_FEATURE_SIM_OPT_HAS_FBSL (0)
/* @brief Has number of FlexBus hold cycle before FlexBus can release bus (register bit SOPT6[PCR]). */
#define FSL_FEATURE_SIM_OPT_HAS_PCR (0)
/* @brief Has number of NFC hold cycle in case of FlexBus request (register bit SOPT6[MCC]). */
#define FSL_FEATURE_SIM_OPT_HAS_MCC (0)
/* @brief Has UART open drain enable (register bits UARTnODE, where n is a number, in register SOPT5)
#define FSL_FEATURE_SIM_OPT_HAS_ODE (1)
/* @brief Number of LPUART modules (number of register bits LPUARTn, where n is a number, in register
#define FSL_FEATURE_SIM_OPT_LPUART_COUNT (0)
/* @brief Number of UART modules (number of register bits UARTn, where n is a number, in register SCG
#define FSL_FEATURE_SIM_OPT_UART_COUNT (3)
/* @brief Has UART0 open drain enable (register bit SOPT5[UART0ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_ODE (1)
/* @brief Has UART1 open drain enable (register bit SOPT5[UART1ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_ODE (1)
/* @brief Has UART2 open drain enable (register bit SOPT5[UART2ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART2_ODE (1)
/* @brief Has LPUART0 open drain enable (register bit SOPT5[LPUART0ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_ODE (0)
/* @brief Has LPUART1 open drain enable (register bit SOPT5[LPUART1ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_ODE (0)
/* @brief Has CMT/UART pad drive strength control (register bit SOPT2[CMTUARTPAD]). */
#define FSL_FEATURE_SIM_OPT_HAS_CMTUARTPAD (0)
/* @brief Has LPUART0 transmit data source selection (register bit SOPT5[LPUART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_TX_SRC (0)
/* @brief Has LPUART0 receive data source selection (register bit SOPT5[LPUART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_RX_SRC (0)
/* @brief Has LPUART1 transmit data source selection (register bit SOPT5[LPUART1TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_TX_SRC (0)
/* @brief Has LPUART1 receive data source selection (register bit SOPT5[LPUART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_RX_SRC (0)
/* @brief Has UART0 transmit data source selection (register bit SOPT5[UART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_TX_SRC (1)
/* @brief UART0 transmit data source selection width (width of register bit SOPT5[UART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART0_TX_SRC_WIDTH (2)
/* @brief Has UART0 receive data source selection (register bit SOPT5[UART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_RX_SRC (1)
/* @brief UART0 receive data source selection width (width of register bit SOPT5[UART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART0_RX_SRC_WIDTH (1)
/* @brief Has UART1 transmit data source selection (register bit SOPT5[UART1TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_TX_SRC (1)
/* @brief Has UART1 receive data source selection (register bit SOPT5[UART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_RX_SRC (1)
```

```
/* @brief UART1 receive data source selection width (width of register bit SOPT5[UART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART1_RX_SRC_WIDTH (1)
/* @brief Has FTM module(s) configuration. */
#define FSL_FEATURE_SIM_OPT_HAS_FTM (0)
/* @brief Number of FTM modules. */
#define FSL_FEATURE_SIM_OPT_FTM_COUNT (0)
/* @brief Number of FTM triggers with selectable source. */
#define FSL_FEATURE_SIM_OPT_FTM_TRIGGER_COUNT (0)
/* @brief Has FTM0 triggers source selection (register bits SOPT4[FTM0TRGnSRC], where n is a number)
#define FSL_FEATURE_SIM_OPT_HAS_FTM0_TRIGGER (0)
/* @brief Has FTM3 triggers source selection (register bits SOPT4[FTM3TRGnSRC], where n is a number)
#define FSL_FEATURE_SIM_OPT_HAS_FTM3_TRIGGER (0)
/* @brief Has FTM1 channel 0 input capture source selection (register bit SOPT4[FTM1CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM1_CHANNELS (0)
/* @brief Has FTM2 channel 0 input capture source selection (register bit SOPT4[FTM2CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM2_CHANNELS (0)
/* @brief Has FTM3 channel 0 input capture source selection (register bit SOPT4[FTM3CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM3_CHANNELS (0)
/* @brief Has FTM2 channel 1 input capture source selection (register bit SOPT4[FTM2CH1SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM2_CHANNEL1 (0)
/* @brief Number of configurable FTM0 fault detection input (number of register bits SOPT4[FTM0FLTn], w
#define FSL_FEATURE_SIM_OPT_FTM0_FAULT_COUNT (0)
/* @brief Number of configurable FTM1 fault detection input (number of register bits SOPT4[FTM1FLTn], w
#define FSL_FEATURE_SIM_OPT_FTM1_FAULT_COUNT (0)
/* @brief Number of configurable FTM2 fault detection input (number of register bits SOPT4[FTM2FLTn], w
#define FSL_FEATURE_SIM_OPT_FTM2_FAULT_COUNT (0)
/* @brief Number of configurable FTM3 fault detection input (number of register bits SOPT4[FTM3FLTn], w
#define FSL_FEATURE_SIM_OPT_FTM3_FAULT_COUNT (0)
/* @brief Has FTM hardware trigger 0 software synchronization (register bit SOPT8[FTMnSYNCBIT], wher
#define FSL_FEATURE_SIM_OPT_HAS_FTM_TRIGGER_SYNC (0)
/* @brief Has FTM channels output source selection (register bit SOPT8[FTMxOCHnSRC], where x is a m
#define FSL_FEATURE_SIM_OPT_HAS_FTM_CHANNELS_OUTPUT_SRC (0)
/* @brief Has TPM module(s) configuration. */
#define FSL_FEATURE_SIM_OPT_HAS_TPM (1)
/* @brief The highest TPM module index. */
#define FSL_FEATURE_SIM_OPT_MAX_TPM_INDEX (2)
/* @brief Has TPM module with index 0. */
#define FSL_FEATURE_SIM_OPT_HAS_TPM0 (1)
/* @brief Has TPM0 clock selection (register bit field SOPT4[TPM0CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPM0_CLK_SEL (1)
/* @brief Is TPM channels configuration in the SOPT4 (not SOPT9) register (register bits TPMnCH0SRC, T
#define FSL_FEATURE_SIM_OPT_HAS_TPM_CHANNELS_CONFIG_IN_SOPT4_REG (1)
/* @brief Has TPM1 channel 0 input capture source selection (register bit field SOPT4[TPM1CH0SRC] or S
#define FSL_FEATURE_SIM_OPT_HAS_TPM1_CH0_SRC_SELECTION (1)
/* @brief Has TPM1 clock selection (register bit field SOPT4[TPM1CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPM1_CLK_SEL (1)
/* @brief TPM1 channel 0 input capture source selection width (width of register bit field SOPT4[TPM1CH0
#define FSL_FEATURE_SIM_OPT_TPM1_CH0_SRC_SELECTION_WIDTH (1)
/* @brief Has TPM2 channel 0 input capture source selection (register bit field SOPT4[TPM2CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPM2_CH0_SRC_SELECTION (1)
/* @brief Has TPM2 clock selection (register bit field SOPT4[TPM2CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPM2_CLK_SEL (1)
/* @brief Has PLL/FLL clock selection (register bit field SOPT2[PLLFLLSEL]). */
```

```
#define FSL_FEATURE_SIM_OPT_HAS_PLL_FLL_SELECTION (1)
/* @brief PLL/FLL clock selection width (width of register bit field SOPT2[PLLFLLSEL]). */
#define FSL_FEATURE_SIM_OPT_PLL_FLL_SELECTION_WIDTH (1)
/* @brief Has NFC clock source selection (register bit SOPT2[NFCSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_NFCSRC (0)
/* @brief Has eSDHC clock source selection (register bit SOPT2[ESDHCSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_ESDHCSRC (0)
/* @brief Has SDHC clock source selection (register bit SOPT2[SDHCSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_SDHCSRC (0)
/* @brief Has LCDC clock source selection (register bits SOPT2[LCDCSRC], SOPT2[LCDC_CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_LCDCSRC (0)
/* @brief Has ENET timestamp clock source selection (register bit SOPT2[TIMESRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_TIMESRC (0)
/* @brief Has ENET RMII clock source selection (register bit SOPT2[RMIISRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_RMIISRC (0)
/* @brief Has USB clock source selection (register bit SOPT2[USBSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_USBSRC (1)
/* @brief Has USB FS clock source selection (register bit SOPT2[USBFSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_USBFSRC (0)
/* @brief Has USB HS clock source selection (register bit SOPT2[USBHSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_USBHSRC (0)
/* @brief Has LPUART clock source selection (register bit SOPT2[LPUARTSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUARTSRC (0)
/* @brief Has LPUART0 clock source selection (register bit SOPT2[LPUART0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0SRC (0)
/* @brief Has LPUART1 clock source selection (register bit SOPT2[LPUART1SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1SRC (0)
/* @brief Has FLEXIOSRC clock source selection (register bit SOPT2[FLEXIOSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FLEXIOSRC (0)
/* @brief Has UART0 clock source selection (register bit SOPT2[UART0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0SRC (1)
/* @brief Has TPM clock source selection (register bit SOPT2[TPMSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPMSRC (1)
/* @brief Has debug trace clock selection (register bit SOPT2[TRACECLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_TRACE_CLKSEL (0)
/* @brief Number of ADC modules (register bits SOPT7[ADCnTRGSEL], SOPT7[ADCnPRETRGSEL], SO
#define FSL_FEATURE_SIM_OPT_ADC_COUNT (1)
/* @brief Has clock 2 output divider (register bit field CLKDIV1[OUTDIV2]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV2 (0)
/* @brief Has clock 3 output divider (register bit field CLKDIV1[OUTDIV3]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV3 (0)
/* @brief Has clock 4 output divider (register bit field CLKDIV1[OUTDIV4]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV4 (1)
/* @brief Clock 4 output divider width (width of register bit field CLKDIV1[OUTDIV4]). */
#define FSL_FEATURE_SIM_DIVIDER_OUTDIV4_WIDTH (3)
/* @brief Has clock 5 output divider (register bit field CLKDIV1[OUTDIV5]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV5 (0)
/* @brief Has USB clock divider (register bit field CLKDIV2[USBDIV] and CLKDIV2[USBFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBDIV (0)
/* @brief Has USB FS clock divider (register bit field CLKDIV2[USBFSDIV] and CLKDIV2[USBFSFRAC]).
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBFSDIV (0)
/* @brief Has USB HS clock divider (register bit field CLKDIV2[USBHSDIV] and CLKDIV2[USBHSFRAC]).
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBHSDIV (0)
```

```
/* @brief Has PLL/FLL clock divider (register bit field CLKDIV3[PLLFLLDIV] and CLKDIV3[PLLFLLFRAC]).
#define FSL_FEATURE_SIM_DIVIDER_HAS_PLLFLLDIV (0)
/* @brief Has LCDC clock divider (register bit field CLKDIV3[LCDCDIV] and CLKDIV3[LCDCFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_LCDCDIV (0)
/* @brief Has trace clock divider (register bit field CLKDIV4[TRACEDIV] and CLKDIV4[TRACEFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_TRACEDIV (0)
/* @brief Has NFC clock divider (register bit field CLKDIV4[NFCDIV] and CLKDIV4[NFCFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_NFCDIV (0)
/* @brief Has Kinetis family ID (register bit field SDID[FAMILYID]). */
#define FSL_FEATURE_SIM_SDID_HAS_FAMILYID (0)
/* @brief Has Kinetis family ID (register bit field SDID[FAMID]). */
#define FSL_FEATURE_SIM_SDID_HAS_FAMID (1)
/* @brief Has Kinetis sub-family ID (register bit field SDID[SUBFAMID]). */
#define FSL_FEATURE_SIM_SDID_HAS_SUBFAMID (1)
/* @brief Has Kinetis series ID (register bit field SDID[SERIESID]). */
#define FSL_FEATURE_SIM_SDID_HAS_SERIESID (1)
/* @brief Has device die ID (register bit field SDID[DIEID]). */
#define FSL_FEATURE_SIM_SDID_HAS_DIEID (1)
/* @brief Has system SRAM size specifier (register bit field SDID[SRAMSIZE]). */
#define FSL_FEATURE_SIM_SDID_HAS_SRAMSIZE (1)
/* @brief Has flash mode (register bit FCFG1[FLASHDOZE]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FLASHDOZE (1)
/* @brief Has flash disable (register bit FCFG1[FLASHDIS]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FLASHDIS (1)
/* @brief Has FTFE disable (register bit FCFG1[FTFDIS]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FTFDIS (0)
/* @brief Has FlexNVM size specifier (register bit field FCFG1[NVMSIZE]). */
#define FSL_FEATURE_SIM_FCFG_HAS_NVMSIZE (0)
/* @brief Has EEPROM size specifier (register bit field FCFG1[EESIZE]). */
#define FSL_FEATURE_SIM_FCFG_HAS_EESIZE (0)
/* @brief Has FlexNVM partition (register bit field FCFG1[DEPART]). */
#define FSL_FEATURE_SIM_FCFG_HAS_DEPART (0)
/* @brief Maximum flash address block 0 address specifier (register bit field FCFG2[MAXADDR0]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR0 (1)
/* @brief Maximum flash address block 1 address specifier (register bit field FCFG2[MAXADDR1]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR1 (0)
/* @brief Maximum flash address block 0 or 1 address specifier (register bit field FCFG2[MAXADDR01]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR01 (0)
/* @brief Maximum flash address block 2 or 3 address specifier (register bit field FCFG2[MAXADDR23]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR23 (0)
/* @brief Has program flash availability specifier (register bit FCFG2[PFLSH]). */
#define FSL_FEATURE_SIM_FCFG_HAS_PFLSH (0)
/* @brief Has program flash swapping (register bit FCFG2[SWAPPFLSH]). */
#define FSL_FEATURE_SIM_FCFG_HAS_PFLSH_SWAP (0)
/* @brief Has miscellanious control register (register MCR). */
#define FSL_FEATURE_SIM_HAS_MISC_CONTROLS (0)
/* @brief Has COP watchdog (registers COPC and SRVCOP). */
#define FSL_FEATURE_SIM_HAS_COP_WATCHDOG (1)
/* @brief Has COP watchdog stop (register bits COPC[COPSTPEN], COPC[COPDBGEN] and COPC[COP
#define FSL_FEATURE_SIM_HAS_COP_STOP (0)
/* @brief Has LLWU clock gate bit (e.g SIM_SCGC4). */
#define FSL_FEATURE_SIM_HAS_SCGC_LLWU (0)
```

```
/* SMC module features */

/* @brief Has partial stop option (register bit STOPCTRL[PSTOPO]). */
#define FSL_FEATURE_SMC_HAS_PSTOPO (1)
/* @brief Has LPO power option (register bit STOPCTRL[LPOPO]). */
#define FSL_FEATURE_SMC_HAS_LPOPO (0)
/* @brief Has POR power option (register bit STOPCTRL[PORPO] or VLLSCTRL[PORPO]). */
#define FSL_FEATURE_SMC_HAS_PORPO (1)
/* @brief Has low power wakeup on interrupt (register bit PMCTRL[LPWUI]). */
#define FSL_FEATURE_SMC_HAS_LPWUI (0)
/* @brief Has LLS or VLLS mode control (register bit STOPCTRL[LLSM]). */
#define FSL_FEATURE_SMC_HAS_LLS_SUBMODE (0)
/* @brief Has VLLS mode control (register bit VLLSCTRL[VLLSM]). */
#define FSL_FEATURE_SMC_USE_VLLSCTRL_REG (0)
/* @brief Has VLLS mode control (register bit STOPCTRL[VLLSM]). */
#define FSL_FEATURE_SMC_USE_STOPCTRL_VLLSM (1)
/* @brief Has RAM partition 2 power option (register bit STOPCTRL[RAM2PO]). */
#define FSL_FEATURE_SMC_HAS_RAM2_POWER_OPTION (0)
/* @brief Has high speed run mode (register bit PMPROT[AHSRUN]). */
#define FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE (0)
/* @brief Has low leakage stop mode (register bit PMPROT[ALLS]). */
#define FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE (1)
/* @brief Has very low leakage stop mode (register bit PMPROT[AVLLS]). */
#define FSL_FEATURE_SMC_HAS_VERY_LOW_LEAKAGE_STOP_MODE (1)
/* @brief Has stop submode. */
#define FSL_FEATURE_SMC_HAS_SUB_STOP_MODE (1)
/* @brief Has stop submode 0(VLLS0). */
#define FSL_FEATURE_SMC_HAS_STOP_SUBMODE0 (1)
/* @brief Has stop submode 2(VLLS2). */
#define FSL_FEATURE_SMC_HAS_STOP_SUBMODE2 (0)
/* @brief Has SMC_PARAM. */
#define FSL_FEATURE_SMC_HAS_PARAM (0)
/* @brief Has SMC_VERID. */
#define FSL_FEATURE_SMC_HAS_VERID (0)
/* @brief Has stop abort flag (register bit PMCTRL[STOPA]). */
#define FSL_FEATURE_SMC_HAS_PMCTRL_STOPA (1)
/* @brief Has tamper reset (register bit SRS[TAMPER]). */
#define FSL_FEATURE_SMC_HAS_SRS_TAMPER (0)
/* @brief Has security violation reset (register bit SRS[SECVIO]). */
#define FSL_FEATURE_SMC_HAS_SRS_SECVIO (0)

/* SPI module features */

/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL_FEATURE_SPI_HAS_FIFO (0)
/* @brief Has DMA support (register bit fields C2[RXDMAE] and C2[TXDMAE]). */
#define FSL_FEATURE_SPI_HAS_DMA_SUPPORT (1)
/* @brief Has separate DMA RX and TX requests. */
#define FSL_FEATURE_SPI_HAS_SEPARATE_DMA_RX_TX_REQn(x) (1)
/* @brief Receive/transmit FIFO size in number of 16-bit communication items. */
#define FSL_FEATURE_SPI_FIFO_SIZEn(x) (0)
/* @brief Maximum transfer data width in bits. */
#define FSL_FEATURE_SPI_MAX_DATA_WIDTH (8)
```

```
/* @brief The data register name has postfix (L as low and H as high). */
#define FSL_FEATURE_SPI_DATA_REGISTER_HAS_POSTFIX (0)
/* @brief Has separated TXDATA and CMD FIFOs (register SREX). */
#define FSL_FEATURE_SPI_HAS_SEPARATE_TXDATA_CMD_FIFO (0)
/* @brief Has 16-bit data transfer support. */
#define FSL_FEATURE_SPI_16BIT_TRANSFERS (0)

/* SysTick module features */

/* @brief Systick has external reference clock. */
#define FSL_FEATURE_SYSTICK_HAS_EXT_REF (1)
/* @brief Systick external reference clock is core clock divided by this value. */
#define FSL_FEATURE_SYSTICK_EXT_REF_CORE_DIV (16)

/* TPM module features */

/* @brief Bus clock is the source clock for the module. */
#define FSL_FEATURE_TPM_BUS_CLOCK (0)
/* @brief Number of channels. */
#define FSL_FEATURE_TPM_CHANNEL_COUNTn(x) \
    ((x) == TPM0 ? (6) : \
    ((x) == TPM1 ? (2) : \
    ((x) == TPM2 ? (2) : (-1))))
/* @brief Has counter reset by the selected input capture event (register bits C0SC[ICRST], C1SC[ICRST]
#define FSL_FEATURE_TPM_HAS_COUNTER_RESET_BY_CAPTURE_EVENT (0)
/* @brief Has TPM_PARAM. */
#define FSL_FEATURE_TPM_HAS_PARAM (0)
/* @brief Has TPM_VERID. */
#define FSL_FEATURE_TPM_HAS_VERID (0)
/* @brief Has TPM_GLOBAL. */
#define FSL_FEATURE_TPM_HAS_GLOBAL (0)
/* @brief Has TPM_TRIG. */
#define FSL_FEATURE_TPM_HAS_TRIG (0)
/* @brief Has counter pause on trigger. */
#define FSL_FEATURE_TPM_HAS_PAUSE_COUNTER_ON_TRIGGER (0)
/* @brief Has external trigger selection. */
#define FSL_FEATURE_TPM_HAS_EXTERNAL_TRIGGER_SELECTION (0)
/* @brief Has TPM_COMBINE register. */
#define FSL_FEATURE_TPM_HAS_COMBINE (0)
/* @brief Whether COMBINE register has effect. */
#define FSL_FEATURE_TPM_COMBINE_HAS_EFFECTn(x) (0)
/* @brief Has TPM_POL. */
#define FSL_FEATURE_TPM_HAS_POL (0)
/* @brief Has TPM_FILTER register. */
#define FSL_FEATURE_TPM_HAS_FILTER (0)
/* @brief Whether FILTER register has effect. */
#define FSL_FEATURE_TPM_FILTER_HAS_EFFECTn(x) (0)
/* @brief Has TPM_QDCTRL register. */
#define FSL_FEATURE_TPM_HAS_QDCTRL (0)
/* @brief Whether QDCTRL register has effect. */
#define FSL_FEATURE_TPM_QDCTRL_HAS_EFFECTn(x) (0)

/* TSI module features */
/* @brief TSI module version. */
#define FSL_FEATURE_TSI_VERSION (4)
/* @brief Has end-of-scan DMA transfer request enable (register bit GENCS[EOSDMEO]). */
#define FSL_FEATURE_TSI_HAS_END_OF_SCAN_DMA_ENABLE (0)
/* @brief Number of TSI channels. */
#define FSL_FEATURE_TSI_CHANNEL_COUNT (16)

/* LPSCI module features */

/* @brief Has receive FIFO overflow detection (bit field CFIFO[RXOFE]). */
#define FSL_FEATURE_LPSCI_HAS_IRQ_EXTENDED_FUNCTIONS (1)
/* @brief Has low power features (can be enabled in wait mode via register bit C1[DOZEEN] or CTRL[DOZ
#define FSL_FEATURE_LPSCI_HAS_LOW_POWER_UART_SUPPORT (1)
/* @brief Has extended data register ED (or extra flags in the DATA register if the registers are 32-bit wide
#define FSL_FEATURE_LPSCI_HAS_EXTENDED_DATA_REGISTER_FLAGS (0)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL_FEATURE_LPSCI_HAS_FIFO (0)
/* @brief Hardware flow control (RTS, CTS) is supported. */
#define FSL_FEATURE_LPSCI_HAS_MODEM_SUPPORT (0)
/* @brief Infrared (modulation) is supported. */
#define FSL_FEATURE_LPSCI_HAS_IR_SUPPORT (0)
/* @brief 2 bits long stop bit is available. */
#define FSL_FEATURE_LPSCI_HAS_STOP_BIT_CONFIG_SUPPORT (1)
/* @brief If 10-bit mode is supported. */
#define FSL_FEATURE_LPSCI_HAS_10BIT_DATA_SUPPORT (1)
/* @brief Baud rate fine adjustment is available. */
#define FSL_FEATURE_LPSCI_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT (0)
/* @brief Baud rate oversampling is available (has bit fields C4[OSR], C5[BOTHEDGE], C5[RESYNCDIS] 
#define FSL_FEATURE_LPSCI_HAS_BAUD_RATE_OVER_SAMPLING_SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_LPSCI_HAS_RX_RESYNC_SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_LPSCI_HAS_BOTH_EDGE_SAMPLING_SUPPORT (1)
/* @brief Peripheral type. */
#define FSL_FEATURE_LPSCI_IS_SCI (1)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL_FEATURE_LPSCI_FIFO_SIZE (0)
/* @brief Maximal data width without parity bit. */
#define FSL_FEATURE_LPSCI_MAX_DATA_WIDTH_WITH_NO_PARITY (10)
/* @brief Maximal data width with parity bit. */
#define FSL_FEATURE_LPSCI_MAX_DATA_WIDTH_WITH_PARITY (9)
/* @brief Supports two match addresses to filter incoming frames. */
#define FSL_FEATURE_LPSCI_HAS_ADDRESS_MATCHING (1)
/* @brief Has transmitter/receiver DMA enable bits C5[TDMAE]/C5[RDMAE] (or BAUD[TDMAE]/BAUD[RD
#define FSL_FEATURE_LPSCI_HAS_DMA_ENABLE (1)
/* @brief Has transmitter/receiver DMA select bits C4[TDMAS]/C4[RDMAS], resp. C5[TDMAS]/C5[RDMAS
#define FSL_FEATURE_LPSCI_HAS_DMA_SELECT (0)
/* @brief Data character bit order selection is supported (bit field S2[MSBF] or STAT[MSBF] if the registers
#define FSL_FEATURE_LPSCI_HAS_BIT_ORDER_SELECT (1)
/* @brief Has smart card (ISO7816 protocol) support and no improved smart card support. */
#define FSL_FEATURE_LPSCI_HAS_SMART_CARD_SUPPORT (0)
/* @brief Has improved smart card (ISO7816 protocol) support. */
```

```
#define FSL_FEATURE_LPSCI_HAS_IMPROVED_SMART_CARD_SUPPORT (0)
/* @brief Has local operation network (CEA709.1-B protocol) support. */
#define FSL_FEATURE_LPSCI_HAS_LOCAL_OPERATION_NETWORK_SUPPORT (0)
/* @brief Has 32-bit registers (BAUD, STAT, CTRL, DATA, MATCH, MODIR) instead of 8-bit (BDH, BDL, C
#define FSL_FEATURE_LPSCI_HAS_32BIT_REGISTERS (0)
/* @brief Lin break detect available (has bit BDH[LBKDIE]). */
#define FSL_FEATURE_LPSCI_HAS_LIN_BREAK_DETECT (1)
/* @brief UART stops in Wait mode available (has bit C1[UARTSWAI]). */
#define FSL_FEATURE_LPSCI_HAS_WAIT_MODE_OPERATION (0)
/* @brief Has separate DMA RX and TX requests. */
#define FSL_FEATURE_LPSCI_HAS_SEPARATE_DMA_RX_TX_REQn(x) (1)

/* UART module features */

/* @brief Has receive FIFO overflow detection (bit field CFIFO[RXOFE]). */
#define FSL_FEATURE_UART_HAS_IRQ_EXTENDED_FUNCTIONS (1)
/* @brief Has low power features (can be enabled in wait mode via register bit C1[DOZEEN] or CTRL[DOZ
#define FSL_FEATURE_UART_HAS_LOW_POWER_UART_SUPPORT (0)
/* @brief Has extended data register ED (or extra flags in the DATA register if the registers are 32-bit wide)
#define FSL_FEATURE_UART_HAS_EXTENDED_DATA_REGISTER_FLAGS (0)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL_FEATURE_UART_HAS_FIFO (0)
/* @brief Hardware flow control (RTS, CTS) is supported. */
#define FSL_FEATURE_UART_HAS_MODEM_SUPPORT (0)
/* @brief Infrared (modulation) is supported. */
#define FSL_FEATURE_UART_HAS_IR_SUPPORT (0)
/* @brief 2 bits long stop bit is available. */
#define FSL_FEATURE_UART_HAS_STOP_BIT_CONFIG_SUPPORT (1)
/* @brief If 10-bit mode is supported. */
#define FSL_FEATURE_UART_HAS_10BIT_DATA_SUPPORT (0)
/* @brief Baud rate fine adjustment is available. */
#define FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT (0)
/* @brief Baud rate oversampling is available (has bit fields C4[OSR], C5[BOTHEDGE], C5[RESYNCDIS] o
#define FSL_FEATURE_UART_HAS_BAUD_RATE_OVER_SAMPLING_SUPPORT (0)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_UART_HAS_RX_RESYNC_SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_UART_HAS_BOTH_EDGE_SAMPLING_SUPPORT (1)
/* @brief Peripheral type. */
#define FSL_FEATURE_UART_IS_SCI (1)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL_FEATURE_UART_FIFO_SIZE (0)
/* @brief Maximal data width without parity bit. */
#define FSL_FEATURE_UART_MAX_DATA_WIDTH_WITH_NO_PARITY (9)
/* @brief Maximal data width with parity bit. */
#define FSL_FEATURE_UART_MAX_DATA_WIDTH_WITH_PARITY (8)
/* @brief Supports two match addresses to filter incoming frames. */
#define FSL_FEATURE_UART_HAS_ADDRESS_MATCHING (0)
/* @brief Has transmitter/receiver DMA enable bits C5[TDMAE]/C5[RDMAE] (or BAUD[TDMAE]/BAUD[RD
#define FSL_FEATURE_UART_HAS_DMA_ENABLE (0)
/* @brief Has transmitter/receiver DMA select bits C4[TDMAS]/C4[RDMAS], resp. C5[TDMAS]/C5[RDMAS
#define FSL_FEATURE_UART_HAS_DMA_SELECT (1)
/* @brief Data character bit order selection is supported (bit field S2[MSBF] or STAT[MSBF] if the registers
#define FSL_FEATURE_UART_HAS_BIT_ORDER_SELECT (0)
/* @brief Has smart card (ISO7816 protocol) support and no improved smart card support. */
#define FSL_FEATURE_UART_HAS_SMART_CARD_SUPPORT (0)
/* @brief Has improved smart card (ISO7816 protocol) support. */
#define FSL_FEATURE_UART_HAS_IMPROVED_SMART_CARD_SUPPORT (0)
/* @brief Has local operation network (CEA709.1-B protocol) support. */
#define FSL_FEATURE_UART_HAS_LOCAL_OPERATION_NETWORK_SUPPORT (0)
/* @brief Has 32-bit registers (BAUD, STAT, CTRL, DATA, MATCH, MODIR) instead of 8-bit (BDH, BDL, C
#define FSL_FEATURE_UART_HAS_32BIT_REGISTERS (0)
/* @brief Lin break detect available (has bit BDH[LBKDIE]). */
#define FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT (1)
/* @brief UART stops in Wait mode available (has bit C1[UARTSWAI]). */
#define FSL_FEATURE_UART_HAS_WAIT_MODE_OPERATION (1)
/* @brief Has separate DMA RX and TX requests. */
#define FSL_FEATURE_UART_HAS_SEPARATE_DMA_RX_TX_REQn(x) (1)

/* USB module features */

/* @brief KHCI module instance count */
#define FSL_FEATURE_USB_KHCI_COUNT (1)
/* @brief HOST mode enabled */
#define FSL_FEATURE_USB_KHCI_HOST_ENABLED (1)
/* @brief OTG mode enabled */
#define FSL_FEATURE_USB_KHCI_OTG_ENABLED (1)
/* @brief Size of the USB dedicated RAM */
#define FSL_FEATURE_USB_KHCI_USB_RAM (0)
/* @brief Has KEEP_ALIVE_CTRL register */
#define FSL_FEATURE_USB_KHCI_KEEP_ALIVE_ENABLED (0)
/* @brief Has the Dynamic SOF threshold compare support */
#define FSL_FEATURE_USB_KHCI_DYNAMIC_SOF_THRESHOLD_COMPARE_ENABLED (0)
/* @brief Has the VBUS detect support */
#define FSL_FEATURE_USB_KHCI_VBUS_DETECT_ENABLED (0)
/* @brief Has the IRC48M module clock support */
#define FSL_FEATURE_USB_KHCI_IRC48M_MODULE_CLOCK_ENABLED (0)
/* @brief Number of endpoints supported */
#define FSL_FEATURE_USB_ENDPT_COUNT (16)

#endif /* _MKL25Z4_FEATURES_H_ */

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/*
 * Copyright (c) 2014 - 2016, Freescale Semiconductor, Inc.
 * Copyright 2016 - 2017 NXP
 * 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 the copyright holder 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" AN
 * 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 FC
 * 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 __FSL_DEVICE_REGISTERS_H__
#define __FSL_DEVICE_REGISTERS_H__

/*
 * Include the cpu specific register header files.
 *
 * The CPU macro should be declared in the project or makefile.
 */
#if (defined(CPU_MKL25Z128VFM4) || defined(CPU_MKL25Z128VFT4) || defined(CPU_MKL25Z128VLH4
    defined(CPU_MKL25Z128VLK4) || defined(CPU_MKL25Z32VFM4) || defined(CPU_MKL25Z32VFT4) || \
    defined(CPU_MKL25Z32VLH4) || defined(CPU_MKL25Z32VLK4) || defined(CPU_MKL25Z64VFM4) || \
    defined(CPU_MKL25Z64VFT4) || defined(CPU_MKL25Z64VLH4) || defined(CPU_MKL25Z64VLK4))

#define KL25Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL25Z4.h"
/* CPU specific feature definitions */
#include "MKL25Z4_features.h"

#else
    #error "No valid CPU defined!"
#endif

#endif /* __FSL_DEVICE_REGISTERS_H__ */

/*******************************************************************************
 * EOF
 ******************************************************************************/
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* ----------------------------------------------------------------------
* Copyright (C) 2010-2014 ARM Limited. All rights reserved.
*
* $Date:        19. October 2015
* $Revision:  V.1.4.5 a
*
```

```
* Project:      CMSIS DSP Library
* Title:     arm_common_tables.h
*
* Description: This file has extern declaration for common tables like Bitreverse, reciprocal etc which are us
*
* Target Processor: Cortex-M4/Cortex-M3
*
* 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 LIMITED 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 OWNER 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.
* -------------------------------------------------------------------- */

#ifndef _ARM_COMMON_TABLES_H
#define _ARM_COMMON_TABLES_H

#include "arm_math.h"

extern const uint16_t armBitRevTable[1024];
extern const q15_t armRecipTableQ15[64];
extern const q31_t armRecipTableQ31[64];
/* extern const q31_t realCoefAQ31[1024]; */
/* extern const q31_t realCoefBQ31[1024]; */
extern const float32_t twiddleCoef_16[32];
extern const float32_t twiddleCoef_32[64];
extern const float32_t twiddleCoef_64[128];
extern const float32_t twiddleCoef_128[256];
extern const float32_t twiddleCoef_256[512];
extern const float32_t twiddleCoef_512[1024];
extern const float32_t twiddleCoef_1024[2048];
extern const float32_t twiddleCoef_2048[4096];
extern const float32_t twiddleCoef_4096[8192];
```

```
#define twiddleCoef twiddleCoef_4096
extern const q31_t twiddleCoef_16_q31[24];
extern const q31_t twiddleCoef_32_q31[48];
extern const q31_t twiddleCoef_64_q31[96];
extern const q31_t twiddleCoef_128_q31[192];
extern const q31_t twiddleCoef_256_q31[384];
extern const q31_t twiddleCoef_512_q31[768];
extern const q31_t twiddleCoef_1024_q31[1536];
extern const q31_t twiddleCoef_2048_q31[3072];
extern const q31_t twiddleCoef_4096_q31[6144];
extern const q15_t twiddleCoef_16_q15[24];
extern const q15_t twiddleCoef_32_q15[48];
extern const q15_t twiddleCoef_64_q15[96];
extern const q15_t twiddleCoef_128_q15[192];
extern const q15_t twiddleCoef_256_q15[384];
extern const q15_t twiddleCoef_512_q15[768];
extern const q15_t twiddleCoef_1024_q15[1536];
extern const q15_t twiddleCoef_2048_q15[3072];
extern const q15_t twiddleCoef_4096_q15[6144];
extern const float32_t twiddleCoef_rfft_32[32];
extern const float32_t twiddleCoef_rfft_64[64];
extern const float32_t twiddleCoef_rfft_128[128];
extern const float32_t twiddleCoef_rfft_256[256];
extern const float32_t twiddleCoef_rfft_512[512];
extern const float32_t twiddleCoef_rfft_1024[1024];
extern const float32_t twiddleCoef_rfft_2048[2048];
extern const float32_t twiddleCoef_rfft_4096[4096];


/* floating-point bit reversal tables */
#define ARMBITREVINDEXTABLE__16_TABLE_LENGTH ((uint16_t)20  )
#define ARMBITREVINDEXTABLE__32_TABLE_LENGTH ((uint16_t)48  )
#define ARMBITREVINDEXTABLE__64_TABLE_LENGTH ((uint16_t)56  )
#define ARMBITREVINDEXTABLE_128_TABLE_LENGTH ((uint16_t)208 )
#define ARMBITREVINDEXTABLE_256_TABLE_LENGTH ((uint16_t)440 )
#define ARMBITREVINDEXTABLE_512_TABLE_LENGTH ((uint16_t)448 )
#define ARMBITREVINDEXTABLE1024_TABLE_LENGTH ((uint16_t)1800)
#define ARMBITREVINDEXTABLE2048_TABLE_LENGTH ((uint16_t)3808)
#define ARMBITREVINDEXTABLE4096_TABLE_LENGTH ((uint16_t)4032)

extern const uint16_t armBitRevIndexTable16[ARMBITREVINDEXTABLE__16_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable32[ARMBITREVINDEXTABLE__32_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable64[ARMBITREVINDEXTABLE__64_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable128[ARMBITREVINDEXTABLE_128_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable256[ARMBITREVINDEXTABLE_256_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable512[ARMBITREVINDEXTABLE_512_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable1024[ARMBITREVINDEXTABLE1024_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable2048[ARMBITREVINDEXTABLE2048_TABLE_LENGTH];
extern const uint16_t armBitRevIndexTable4096[ARMBITREVINDEXTABLE4096_TABLE_LENGTH];

/* fixed-point bit reversal tables */
#define ARMBITREVINDEXTABLE_FIXED___16_TABLE_LENGTH ((uint16_t)12  )
#define ARMBITREVINDEXTABLE_FIXED___32_TABLE_LENGTH ((uint16_t)24  )
```

```
#define ARMBITREVINDEXTABLE_FIXED___64_TABLE_LENGTH ((uint16_t)56  )
#define ARMBITREVINDEXTABLE_FIXED__128_TABLE_LENGTH ((uint16_t)112 )
#define ARMBITREVINDEXTABLE_FIXED__256_TABLE_LENGTH ((uint16_t)240 )
#define ARMBITREVINDEXTABLE_FIXED__512_TABLE_LENGTH ((uint16_t)480 )
#define ARMBITREVINDEXTABLE_FIXED_1024_TABLE_LENGTH ((uint16_t)992 )
#define ARMBITREVINDEXTABLE_FIXED_2048_TABLE_LENGTH ((uint16_t)1984)
#define ARMBITREVINDEXTABLE_FIXED_4096_TABLE_LENGTH ((uint16_t)4032)

extern const uint16_t armBitRevIndexTable_fixed_16[ARMBITREVINDEXTABLE_FIXED___16_TABLE_L
extern const uint16_t armBitRevIndexTable_fixed_32[ARMBITREVINDEXTABLE_FIXED___32_TABLE_L
extern const uint16_t armBitRevIndexTable_fixed_64[ARMBITREVINDEXTABLE_FIXED___64_TABLE_L
extern const uint16_t armBitRevIndexTable_fixed_128[ARMBITREVINDEXTABLE_FIXED__128_TABLE_
extern const uint16_t armBitRevIndexTable_fixed_256[ARMBITREVINDEXTABLE_FIXED__256_TABLE_
extern const uint16_t armBitRevIndexTable_fixed_512[ARMBITREVINDEXTABLE_FIXED__512_TABLE_
extern const uint16_t armBitRevIndexTable_fixed_1024[ARMBITREVINDEXTABLE_FIXED_1024_TABLE
extern const uint16_t armBitRevIndexTable_fixed_2048[ARMBITREVINDEXTABLE_FIXED_2048_TABLE
extern const uint16_t armBitRevIndexTable_fixed_4096[ARMBITREVINDEXTABLE_FIXED_4096_TABLE

/* Tables for Fast Math Sine and Cosine */
extern const float32_t sinTable_f32[FAST_MATH_TABLE_SIZE + 1];
extern const q31_t sinTable_q31[FAST_MATH_TABLE_SIZE + 1];
extern const q15_t sinTable_q15[FAST_MATH_TABLE_SIZE + 1];

#endif /*  ARM_COMMON_TABLES_H */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 *
 * Debug console shall provide input and output functions to scan and print formatted data.
 * o Support a format specifier for PRINTF follows this prototype "%[flags][width][.precision][length]specifier
 *   - [flags] :'-', '+', '#', ' ', '0'
 *   - [width]:  number (0,1...)
 *   - [.precision]: number (0,1...)
 *   - [length]: do not support
 *   - [specifier]: 'd', 'i', 'f', 'F', 'x', 'X', 'o', 'p', 'u', 'c', 's', 'n'
 * o Support a format specifier for SCANF follows this prototype " %[*][width][length]specifier"
 *   - [*]: is supported.
 *   - [width]: number (0,1...)
 *   - [length]: 'h', 'hh', 'l','ll','L'. ignore ('j','z','t')
 *   - [specifier]: 'd', 'i', 'u', 'f', 'F', 'e', 'E', 'g', 'G', 'a', 'A', 'o', 'c', 's'
 */

#ifndef _FSL_DEBUGCONSOLE_H_
#define _FSL_DEBUGCONSOLE_H_

#include "fsl_common.h"

/*
 * @addtogroup debugconsole
 * @{
 */

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*! @brief Definition to select sdk or toolchain printf, scanf. */
#ifndef SDK_DEBUGCONSOLE
#define SDK_DEBUGCONSOLE 1U
#endif

#if defined(SDK_DEBUGCONSOLE) && !(SDK_DEBUGCONSOLE)
#include <stdio.h>
#endif

/*! @brief Definition to printf the float number. */
#ifndef PRINTF_FLOAT_ENABLE
#define PRINTF_FLOAT_ENABLE 0U
#endif /* PRINTF_FLOAT_ENABLE */

/*! @brief Definition to scanf the float number. */
#ifndef SCANF_FLOAT_ENABLE
#define SCANF_FLOAT_ENABLE 0U
#endif /* SCANF_FLOAT_ENABLE */

/*! @brief Definition to support advanced format specifier for printf. */
#ifndef PRINTF_ADVANCED_ENABLE
#define PRINTF_ADVANCED_ENABLE 0U
#endif /* PRINTF_ADVANCED_ENABLE */
```

```
/*! @brief Definition to support advanced format specifier for scanf. */
#ifndef SCANF_ADVANCED_ENABLE
#define SCANF_ADVANCED_ENABLE 0U
#endif /* SCANF_ADVANCED_ENABLE */

#if SDK_DEBUGCONSOLE /* Select printf, scanf, putchar, getchar of SDK version. */
#define PRINTF DbgConsole_Printf
#define SCANF DbgConsole_Scanf
#define PUTCHAR DbgConsole_Putchar
#define GETCHAR DbgConsole_Getchar
#else /* Select printf, scanf, putchar, getchar of toolchain. */
#define PRINTF printf
#define SCANF scanf
#define PUTCHAR putchar
#define GETCHAR getchar
#endif /* SDK_DEBUGCONSOLE */

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif /* __cplusplus */

/*! @name Initialization*/
/* @{ */

/*!
 * @brief Initializes the the peripheral used for debug messages.
 *
 * Call this function to enable debug log messages to be output via the specified peripheral,
 * frequency of peripheral source clock, and base address at the specified baud rate.
 * After this function has returned, stdout and stdin are connected to the selected peripheral.
 *
 * @param baseAddr      Indicates the address of the peripheral used to send debug messages.
 * @param baudRate      The desired baud rate in bits per second.
 * @param device        Low level device type for the debug console, can be one of the following.
 *                      @arg DEBUG_CONSOLE_DEVICE_TYPE_UART,
 *                      @arg DEBUG_CONSOLE_DEVICE_TYPE_LPUART,
 *                      @arg DEBUG_CONSOLE_DEVICE_TYPE_LPSCI,
 *                      @arg DEBUG_CONSOLE_DEVICE_TYPE_USBCDC.
 * @param clkSrcFreq    Frequency of peripheral source clock.
 *
 * @return              Indicates whether initialization was successful or not.
 * @retval kStatus_Success          Execution successfully
 * @retval kStatus_Fail             Execution failure
 * @retval kStatus_InvalidArgument  Invalid argument existed
 */
status_t DbgConsole_Init(uint32_t baseAddr, uint32_t baudRate, uint8_t device, uint32_t clkSrcFreq);

/*!
 * @brief De-initializes the peripheral used for debug messages.
 *
 * Call this function to disable debug log messages to be output via the specified peripheral
 * base address and at the specified baud rate.
 *
 * @return Indicates whether de-initialization was successful or not.
 */
status_t DbgConsole_Deinit(void);

#if SDK_DEBUGCONSOLE
/*!
 * @brief Writes formatted output to the standard output stream.
 *
 * Call this function to write a formatted output to the standard output stream.
 *
 * @param   fmt_s Format control string.
 * @return  Returns the number of characters printed or a negative value if an error occurs.
 */
int DbgConsole_Printf(const char *fmt_s, ...);

/*!
 * @brief Writes a character to stdout.
 *
 * Call this function to write a character to stdout.
 *
 * @param   ch Character to be written.
 * @return  Returns the character written.
 */
int DbgConsole_Putchar(int ch);

/*!
 * @brief Reads formatted data from the standard input stream.
 *
 * Call this function to read formatted data from the standard input stream.
 *
 * @param   fmt_ptr Format control string.
 * @return  Returns the number of fields successfully converted and assigned.
 */
int DbgConsole_Scanf(char *fmt_ptr, ...);

/*!
 * @brief Reads a character from standard input.
 *
 * Call this function to read a character from standard input.
 *
 * @return Returns the character read.
 */
int DbgConsole_Getchar(void);

#endif /* SDK_DEBUGCONSOLE */

/*! @} */

#if defined(__cplusplus)
```

```
}
#endif /* __cplusplus */

/*! @} */

#endif /* _FSL_DEBUGCONSOLE_H_ */
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Top level makefile. Delves into the Release and Debug subdirs and runs make on the specific targets co

# Builds all targets.
all: fb_run fb_debug pc_run pc_debug

# Builds the PC release build.
pc_run:
 cd Release && $(MAKE) output/pc_run

# Builds the FB release build.
fb_run:
 cd Release && $(MAKE) output/pes_project2.axf

# Builds the PC debug build.
pc_debug:
 cd Debug && $(MAKE) output/pc_debug

# Builds the FB debug build.
fb_debug:
 cd Debug && $(MAKE) output/pes_project2.axf

# Cleans both Debug and Release areas.
clean:
 cd Debug && $(MAKE) clean
 cd Release && $(MAKE) clean
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# defines for the pc builds
PC_SRC_REL_ROOT = ../source/pc_implementation
PC_SRCS = $(wildcard $(PC_SRC_REL_ROOT)/*.c)
PC_OBJS = $(patsubst ../%.c, bin/%.o, $(PC_SRCS)) bin/source/pc_implementation/main.o
# compiler
CC = gcc
# flags
CFLAGS  = -g -Wall -Werror -I../include
RM := rm -rf

# These includes were generated by the IDE.
-include sources.mk
-include utilities/subdir.mk
-include startup/subdir.mk
-include source/fb_implementation/subdir.mk
-include source/subdir.mk
-include drivers/subdir.mk
-include board/subdir.mk
-include CMSIS/subdir.mk
-include subdir.mk
```

```
-include objects.mk

# runs all rules
all: pc_debug fb_debug

# dependencies create the output and build dirs and then compile/link all the code
pc_debug: output/pc_debug
fb_debug: output/pes_project2_debug.axf

# Create the output dir and place
output/pc_debug: $(PC_OBJS)
 mkdir -p $(dir $@)
 $(CC) $(CFLAGS) -DDEBUG -o $@ $^

# IDE generated FB build
output/pes_project2.axf: $(OBJS) $(USER_OBJS)
 mkdir -p $(dir $@)
 @echo 'Building target: $@'
 @echo 'Invoking: MCU Linker'
 arm-none-eabi-gcc -Werror -Wall -nostdlib -Xlinker --gc-sections -Xlinker -Map="pes_project2.map" -Xlinke
 @echo 'Finished building target: $@'
 @echo ' '
 $(MAKE) --no-print-directory post-build

# making object targets for all the source files
bin/source/pc_implementation/%.o: $(PC_SRC_REL_ROOT)/%.c
 mkdir -p $(dir $@)
 gcc -c -o $@ $< $(CFLAGS) -DDEBUG

# special case main, which is not in the platform specific dir
bin/source/pc_implementation/main.o: ../source/main.c
 mkdir -p $(dir $@)
 gcc -c -o $@ $< $(CFLAGS) -DDEBUG

# Other Targets
clean:
 -$(RM) $(EXECUTABLES)$(OBJS)$(C_DEPS)$(PC_OBJS) output/pc_debug output/pes_project2.axf sou
 -@echo ' '

post-build:
 -@echo 'Performing post-build steps'
 -arm-none-eabi-size "output/pes_project2.axf"; # arm-none-eabi-objcopy -v -O binary "pes_project2.axf" "p
 -@echo ' '

.PHONY: all clean dependents post-build

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# defines for the pc builds
PC_SRC_REL_ROOT = ../source/pc_implementation
PC_SRCS = $(wildcard $(PC_SRC_REL_ROOT)/*.c)
PC_OBJS = $(patsubst ../%.c, bin/%.o, $(PC_SRCS)) bin/source/pc_implementation/main.o
# compiler
CC = gcc
```

```
# flags
CFLAGS  = -g -Werror -Wall -I../include
RM := rm -rf

# All of the sources participating in the build are defined here
-include sources.mk
-include utilities/subdir.mk
-include startup/subdir.mk
-include source/pc_implementation/subdir.mk
-include source/fb_implementation/subdir.mk
-include source/subdir.mk
-include drivers/subdir.mk
-include board/subdir.mk
-include CMSIS/subdir.mk
-include subdir.mk
-include objects.mk

# runs all rules
all: pc_run fb_run

# dependencies create the output and build dirs and then compile/link all the code
pc_run: output/pc_run
fb_run: output/pes_project2.axf

output/pc_run: $(PC_OBJS)
 mkdir -p $(dir $@)
 $(CC) $(CFLAGS) -o $@ $^

# Tool invocations
output/pes_project2.axf: $(OBJS) $(USER_OBJS)
 mkdir -p $(dir $@)
 @echo 'Building target: $@'
 @echo 'Invoking: MCU Linker'
 arm-none-eabi-gcc -Werror -Wall -nostdlib -Xlinker --gc-sections -Xlinker -Map="pes_project2.map" -Xlinke
 @echo 'Finished building target: $@'
 @echo ' '
 $(MAKE) --no-print-directory post-build


# making object targets for all the source files
bin/source/pc_implementation/%.o: $(PC_SRC_REL_ROOT)/%.c
 mkdir -p $(dir $@)
 gcc -c -o $@ $< $(CFLAGS)

# special case main, which is not in the platform specific dir
bin/source/pc_implementation/main.o: ../source/main.c
 mkdir -p $(dir $@)
 gcc -c -o $@ $< $(CFLAGS)

# Other Targets
clean:
 -$(RM) $(EXECUTABLES)$(OBJS)$(C_DEPS) pes_project2.axf output/pc_debug */*.o */*/*.o */*.su */*/*.s
 -@echo ' '
```

```
post-build:
 -@echo 'Performing post-build steps'
 -arm-none-eabi-size "output/pes_project2.axf"; # arm-none-eabi-objcopy -v -O binary "pes_project2.axf" "p
 -@echo ' '

.PHONY: all clean dependents post-build

-include ../makefile.targets
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

USER_OBJS :=

LIBS :=

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

OBJ_SRCS :=
S_SRCS :=
ASM_SRCS :=
C_SRCS :=
S_UPPER_SRCS :=
O_SRCS :=
EXECUTABLES :=
OBJS :=
C_DEPS :=

# Every subdirectory with source files must be described here
SUBDIRS := \
CMSIS \
board \
drivers \
source/fb_implementation \
source \
startup \
utilities \

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../drivers/fsl_clock.c \
../drivers/fsl_common.c \
../drivers/fsl_flash.c \
```

```
../drivers/fsl_gpio.c \
../drivers/fsl_lpsci.c \
../drivers/fsl_smc.c \
../drivers/fsl_uart.c

OBJS += \
./drivers/fsl_clock.o \
./drivers/fsl_common.o \
./drivers/fsl_flash.o \
./drivers/fsl_gpio.o \
./drivers/fsl_lpsci.o \
./drivers/fsl_smc.o \
./drivers/fsl_uart.o

C_DEPS += \
./drivers/fsl_clock.d \
./drivers/fsl_common.d \
./drivers/fsl_flash.d \
./drivers/fsl_gpio.d \
./drivers/fsl_lpsci.d \
./drivers/fsl_smc.d \
./drivers/fsl_uart.d


# Each subdirectory must supply rules for building sources it contributes
drivers/%.o: ../drivers/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../startup/startup_mkl25z4.c

OBJS += \
./startup/startup_mkl25z4.o

C_DEPS += \
./startup/startup_mkl25z4.d


# Each subdirectory must supply rules for building sources it contributes
startup/%.o: ../startup/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
```

```
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../board/board.c \
../board/clock_config.c \
../board/peripherals.c \
../board/pin_mux.c

OBJS += \
./board/board.o \
./board/clock_config.o \
./board/peripherals.o \
./board/pin_mux.o

C_DEPS += \
./board/board.d \
./board/clock_config.d \
./board/peripherals.d \
./board/pin_mux.d


# Each subdirectory must supply rules for building sources it contributes
board/%.o: ../board/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../CMSIS/system_MKL25Z4.c

OBJS += \
./CMSIS/system_MKL25Z4.o

C_DEPS += \
./CMSIS/system_MKL25Z4.d
```

```
# Each subdirectory must supply rules for building sources it contributes
CMSIS/%.o: ../CMSIS/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../utilities/fsl_debug_console.c

OBJS += \
./utilities/fsl_debug_console.o

C_DEPS += \
./utilities/fsl_debug_console.d


# Each subdirectory must supply rules for building sources it contributes
utilities/%.o: ../utilities/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../source/main.c \
../source/semihost_hardfault.c

OBJS += \
./source/main.o \
./source/semihost_hardfault.o

C_DEPS += \
./source/main.d \
./source/semihost_hardfault.d
```

```
# Each subdirectory must supply rules for building sources it contributes
source/%.o: ../source/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../source/fb_implementation/delay.c \
../source/fb_implementation/handle_led.c \
../source/fb_implementation/setup_teardown.c

OBJS += \
./source/fb_implementation/delay.o \
./source/fb_implementation/handle_led.o \
./source/fb_implementation/setup_teardown.o

C_DEPS += \
./source/fb_implementation/delay.d \
./source/fb_implementation/handle_led.d \
./source/fb_implementation/setup_teardown.d


# Each subdirectory must supply rules for building sources it contributes
source/fb_implementation/%.o: ../source/fb_implementation/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -D__REDLIB__ -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../board/board.c \
../board/clock_config.c \
../board/peripherals.c \
../board/pin_mux.c

OBJS += \
```

```
./board/board.o \
./board/clock_config.o \
./board/peripherals.o \
./board/pin_mux.o

C_DEPS += \
./board/board.d \
./board/clock_config.d \
./board/peripherals.d \
./board/pin_mux.d


# Each subdirectory must supply rules for building sources it contributes
board/%.o: ../board/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -D__REDLIB__ -O0 -g3 -Wall -c -fmessage-length=0 -mcpu=cortex-m7 -mthumb -D__
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

USER_OBJS :=

LIBS :=

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

OBJ_SRCS :=
S_SRCS :=
ASM_SRCS :=
C_SRCS :=
S_UPPER_SRCS :=
O_SRCS :=
EXECUTABLES :=
OBJS :=
C_DEPS :=

# Every subdirectory with source files must be described here
SUBDIRS := \
CMSIS \
board \
drivers \
source/fb_implementation \
source \
source/pc_implementation \
```

```
startup \
utilities \

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../drivers/fsl_clock.c \
../drivers/fsl_common.c \
../drivers/fsl_flash.c \
../drivers/fsl_gpio.c \
../drivers/fsl_lpsci.c \
../drivers/fsl_smc.c \
../drivers/fsl_uart.c

OBJS += \
./drivers/fsl_clock.o \
./drivers/fsl_common.o \
./drivers/fsl_flash.o \
./drivers/fsl_gpio.o \
./drivers/fsl_lpsci.o \
./drivers/fsl_smc.o \
./drivers/fsl_uart.o

C_DEPS += \
./drivers/fsl_clock.d \
./drivers/fsl_common.d \
./drivers/fsl_flash.d \
./drivers/fsl_gpio.d \
./drivers/fsl_lpsci.d \
./drivers/fsl_smc.d \
./drivers/fsl_uart.d


# Each subdirectory must supply rules for building sources it contributes
drivers/%.o: ../drivers/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DS
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
```

```
../startup/startup_mkl25z4.c

OBJS += \
./startup/startup_mkl25z4.o

C_DEPS += \
./startup/startup_mkl25z4.d


# Each subdirectory must supply rules for building sources it contributes
startup/%.o: ../startup/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DS
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../board/board.c \
../board/clock_config.c \
../board/peripherals.c \
../board/pin_mux.c

OBJS += \
./board/board.o \
./board/clock_config.o \
./board/peripherals.o \
./board/pin_mux.o

C_DEPS += \
./board/board.d \
./board/clock_config.d \
./board/peripherals.d \
./board/pin_mux.d


# Each subdirectory must supply rules for building sources it contributes
board/%.o: ../board/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DS
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../CMSIS/system_MKL25Z4.c

OBJS += \
./CMSIS/system_MKL25Z4.o

C_DEPS += \
./CMSIS/system_MKL25Z4.d


# Each subdirectory must supply rules for building sources it contributes
CMSIS/%.o: ../CMSIS/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DS
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../utilities/fsl_debug_console.c

OBJS += \
./utilities/fsl_debug_console.o

C_DEPS += \
./utilities/fsl_debug_console.d


# Each subdirectory must supply rules for building sources it contributes
utilities/%.o: ../utilities/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DDEBUG -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DS
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################
```

```
# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../source/main.c \
../source/semihost_hardfault.c

OBJS += \
./source/main.o \
./source/semihost_hardfault.o

C_DEPS += \
./source/main.d \
./source/semihost_hardfault.d


# Each subdirectory must supply rules for building sources it contributes
source/%.o: ../source/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DSCANF_FLC
 @echo 'Finished building: $<'
 @echo ' '


  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
################################################################################
# Automatically-generated file. Do not edit!
################################################################################

# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../source/fb_implementation/delay.c \
../source/fb_implementation/handle_led.c \
../source/fb_implementation/setup_teardown.c

OBJS += \
./source/fb_implementation/delay.o \
./source/fb_implementation/handle_led.o \
./source/fb_implementation/setup_teardown.o

C_DEPS += \
./source/fb_implementation/delay.d \
./source/fb_implementation/handle_led.d \
./source/fb_implementation/setup_teardown.d


# Each subdirectory must supply rules for building sources it contributes
source/fb_implementation/%.o: ../source/fb_implementation/%.c
 @echo 'Building file: $<'
 @echo 'Invoking: MCU C Compiler'
 arm-none-eabi-gcc -std=gnu99 -DCPU_MKL25Z128VLK4 -DPRINTF_FLOAT_ENABLE=0 -DSCANF_FLC
 @echo 'Finished building: $<'
 @echo ' '
```

~~~~~~
# PES Project 2 Readme
Jack Campbell, Troy Davis

## Description
This repo contains custom sources and makefiles for Project 2 as well as adapted and generated code from MCUXpresso and the KL25Z SDK.

A global makefile lives at the project root that contains all the build targets:
fb_run, fb_debug, pc_run, pc_debug, and clean. The *_run variants go into the `Release`
directory and execute release-specific rules and make logic, whereas the *_debug variants
do the same thing in the `Debug` subdirectory.

The `*.mk` files were all generated by the IDE and saved and checked in so the project
could be loaded consistently from the repo.

## Installation/Execution Notes

These are the steps to build the project in MCUXpresso.

1) Clone the repo
2) In MCUXpresso, click `New > Project`.
3) Select `Makefile project with existin code...`
4) Unselect C++, enter a project name, browse to the directory of the repo, and select `NXP MCU Tools`, t
5) Now the project is active in the IDE.

### Adding build targets

To add the build targets specified in the makefile:
1) Open the Build Targets side window
2) Right click the project name
3) Select New
4) Enter the name of one of the build targets (fb_run, fb_debug, pc_run, pc_debug, and clean)
5) Repeat for all build targets
6) Double clicking on a target name will invoke that target

### Running the PC builds

After building the targets (either individually using the build target steps above or just using the hammer ico

1) Right click on the project name in the file hierarchy, select `Run as > Run configurations...`
2) Select `C/C++ Application`
3) Hit `New launch configuration`
4) Select a name for the output configuration (you need one for both Release and Debug)
5) Set the `C/C++ Application` field to the binary you want to run, either `Debug/output/pc_debug` for Debu
6) Hit Apply
7) Hit Run
8) The program should run in the console below.

### Running the FB builds
~~~~~~

1) Right click on the project name in the file hierarchy, select `Debug as > Debug configurations...`
2) Select `GDB PEMicro Interface Debugging`
3) Hit `New launch configuration`
4) Select a name for the output configuration (you need one for both Release and Debug)
5) Set the `C/C++ Application` field to the binary you want to run, either `Debug/output/pes_project2.axf` fo
6) Hit Apply
7) Hit Debug
8) The program should run in the console below, provided the board is connected successfully.