```
/**

  ******************************************************************************
  * @file    system_stm32f4xx.c

  * @author  MCD Application Team
  * @version V1.0.0RC1
  * @date    25-August-2011
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.
  *          This file contains the system clock configuration for STM32F4xx devices,
  *          and is generated by the clock configuration tool
  *          stm32f4xx_Clock_Configuration_V1.0.0.xls
  *
  * 1.  This file provides two functions and one global variable to be called from
  *     user application:
  *      - SystemInit(): Setups the system clock (System clock source, PLL Multiplier
  *                      and Divider factors, AHB/APBx prescalers and Flash settings),
  *                      depending on the configuration made in the clock xls tool.
  *                      This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it can be used
  *                                  by the user application to setup the SysTick
  *                                  timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must
  *                                 be called whenever the core clock is changed
  *                                 during program execution.
  *
  * 2. After each device reset the HSI (16 MHz) is used as system clock source.
  *    Then SystemInit() function is called, in "startup_stm32f4xx.s" file, to
  *    configure the system clock before to branch to main program.
  *
  * 3. If the system clock source selected by user fails to startup, the SystemInit()
  *    function will do nothing and HSI still used as system clock source. User can
  *    add some code to deal with this issue inside the SetSysClock() function.
  *
  * 4. The default value of HSE crystal is set to 25MHz, refer to "HSE_VALUE" define
  *    in "stm32f4xx.h" file. When HSE is used as system clock source, directly or
  *    through PLL, and you are using different crystal you have to adapt the HSE
  *    value to your own configuration.
  *
  * 5. This file configures the system clock as follows:
  *=============================================================================
  *=============================================================================
  *        Supported STM32F4xx device revision    | Rev A
```

```
*-----------------------------------------------------------------------
*        System Clock source                    | PLL (HSE)
*-----------------------------------------------------------------------
*        SYSCLK(Hz)                             | 168000000
*-----------------------------------------------------------------------
*        HCLK(Hz)                               | 168000000
*-----------------------------------------------------------------------
*        AHB Prescaler                          | 1
*-----------------------------------------------------------------------
*        APB1 Prescaler                         | 4
*-----------------------------------------------------------------------
*        APB2 Prescaler                         | 2
*-----------------------------------------------------------------------
*        HSE Frequency(Hz)                      | 25000000
*-----------------------------------------------------------------------
*        PLL_M                                  | 25
*-----------------------------------------------------------------------
*        PLL_N                                  | 336
*-----------------------------------------------------------------------
*        PLL_P                                  | 2
*-----------------------------------------------------------------------
*        PLL_Q                                  | 7
*-----------------------------------------------------------------------
*        PLLI2S_N                               | NA
*-----------------------------------------------------------------------
*        PLLI2S_R                               | NA
*-----------------------------------------------------------------------
*        I2S input clock                        | NA
*-----------------------------------------------------------------------
*        VDD(V)                                 | 3.3
*-----------------------------------------------------------------------
*        High Performance mode                  | Enabled
*-----------------------------------------------------------------------
*        Flash Latency(WS)                      | 5
*-----------------------------------------------------------------------
*        Prefetch Buffer                        | OFF
*-----------------------------------------------------------------------
*        Instruction cache                      | ON
*-----------------------------------------------------------------------
*        Data cache                             | ON
*-----------------------------------------------------------------------
*        Require 48MHz for USB OTG FS,          | Enabled
*        SDIO and RNG clock                     |
*-----------------------------------------------------------------------
*=======================================================================
*******************************************************************************
* @attention
*
```

```
  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS
  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING
  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE
  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
  *
  * <h2><center>&copy; COPYRIGHT 2011 STMicroelectronics</center></h2>
  ******************************************************************************
  */

/** @addtogroup CMSIS
  * @{
  */

/** @addtogroup stm32f4xx_system
  * @{
  */

/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */

#include "stm32f4xx.h"

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */

/*!< Uncomment the following line if you need to use external SRAM mounted
     on STM324xG_EVAL board as data memory  */
/* #define DATA_IN_ExtSRAM */

/*!< Uncomment the following line if you need to relocate your vector Table in
     Internal SRAM. */
/* #define VECT_TAB_SRAM */
```

```
#define VECT_TAB_OFFSET  0x00 /*!< Vector Table base offset field.
                                   This value must be a multiple of 0x200. */


/* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N */
#define PLL_M      8 // is the quartz frequency mounted on the board
#define PLL_N      336

/* SYSCLK = PLL_VCO / PLL_P */
#define PLL_P      2

/* USB OTG FS, SDIO and RNG Clock =  PLL_VCO / PLLQ */
#define PLL_Q      7

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */

  uint32_t SystemCoreClock = 168000000;

  __I uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */

static void SetSysClock(void);
#ifdef DATA_IN_ExtSRAM
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM */

/**
```

```
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */

/**
  * @brief  Setup the microcontroller system
  *         Initialize the Embedded Flash Interface, the PLL and update the
  *         SystemFrequency variable.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* Reset the RCC clock configuration to the default reset state ------------*/
  /* Set HSION bit */
  RCC->CR |= (uint32_t)0x00000001;

  /* Reset CFGR register */
  RCC->CFGR = 0x00000000;

  /* Reset HSEON, CSSON and PLLON bits */
  RCC->CR &= (uint32_t)0xFEF6FFFF;

  /* Reset PLLCFGR register */
  RCC->PLLCFGR = 0x24003010;

  /* Reset HSEBYP bit */
  RCC->CR &= (uint32_t)0xFFFBFFFF;

  /* Disable all interrupts */
  RCC->CIR = 0x00000000;

#ifdef DATA_IN_ExtSRAM
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM */

  /* Configure the System clock source, PLL Multiplier and Divider factors,
     AHB/APBx prescalers and Flash settings ----------------------------------*/
  SetSysClock();

  /* Configure the Vector Table location add offset address ------------------*/
#ifdef VECT_TAB_SRAM
  SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH */
```

```
#endif
}

/**
   * @brief  Update SystemCoreClock variable according to Clock Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK), it can
  *         be used by the user application to setup the SysTick timer or configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function must be called
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx.h file (default value
  *             16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
  *
  *         (**) HSE_VALUE is a constant defined in stm32f4xx.h file (default value
  *              25 MHz), user has to ensure that HSE_VALUE is same as the real
  *              frequency of the crystal used. Otherwise, this function may
  *              have wrong result.
  *
  *         - The result of this function could be not correct when using fractional
  *           value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;

  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;

  switch (tmp)
  {
```

```
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */

      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >> 22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;

      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
        /* HSI used as PLL clock source */
        pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
      }

      pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 ) *2;
      SystemCoreClock = pllvco/pllp;
      break;
    default:
      SystemCoreClock = HSI_VALUE;
      break;
  }
  /* Compute HCLK frequency --------------------------------------------------*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}

/**
  * @brief  Configures the System clock source, PLL Multiplier and Divider factors,
  *         AHB/APBx prescalers and Flash settings
  * @Note   This function should be called only once the RCC clock configuration
  *         is reset to the default reset state (done in SystemInit() function).
  * @param  None
  * @retval None
  */
static void SetSysClock(void)
```

```
{
/******************************************************************************/
/*            PLL (clocked by HSE) used as System clock source                */
/******************************************************************************/
  __IO uint32_t StartUpCounter = 0, HSEStatus = 0;

  /* Enable HSE */
  RCC->CR |= ((uint32_t)RCC_CR_HSEON);

  /* Wait till HSE is ready and if Time out is reached exit */
  do
  {
    HSEStatus = RCC->CR & RCC_CR_HSERDY;
    StartUpCounter++;
  } while((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

  if ((RCC->CR & RCC_CR_HSERDY) != RESET)
  {
    HSEStatus = (uint32_t)0x01;
  }
  else
  {
    HSEStatus = (uint32_t)0x00;
  }

  if (HSEStatus == (uint32_t)0x01)
  {
    /* Enable high performance mode, System frequency up to 168 MHz */
    RCC->APB1ENR |= RCC_APB1ENR_PWREN;
    PWR->CR |= PWR_CR_PMODE;

    /* HCLK = SYSCLK / 1*/
    RCC->CFGR |= RCC_CFGR_HPRE_DIV1;

    /* PCLK2 = HCLK / 2*/
    RCC->CFGR |= RCC_CFGR_PPRE2_DIV2;

    /* PCLK1 = HCLK / 4*/
    RCC->CFGR |= RCC_CFGR_PPRE1_DIV4;

    /* Configure the main PLL */
    RCC->PLLCFGR = PLL_M | (PLL_N << 6) | (((PLL_P >> 1) -1) << 16) |
                   (RCC_PLLCFGR_PLLSRC_HSE) | (PLL_Q << 24);

    /* Enable the main PLL */
    RCC->CR |= RCC_CR_PLLON;

    /* Wait till the main PLL is ready */
```

```
    while((RCC->CR & RCC_CR_PLLRDY) == 0)
    {
    }

    /* Configure Flash prefetch, Instruction cache, Data cache and wait state */
    FLASH->ACR = FLASH_ACR_ICEN |FLASH_ACR_DCEN |FLASH_ACR_LATENCY_5WS;

    /* Select the main PLL as system clock source */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
    RCC->CFGR |= RCC_CFGR_SW_PLL;

    /* Wait till the main PLL is used as system clock source */
    while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS ) != RCC_CFGR_SWS_PLL);
    {
    }
  }
  else
  { /* If HSE fails to start-up, the application will have wrong clock
         configuration. User can add here some code to deal with this error */
  }

}

/**
  * @brief  Setup the external memory controller. Called in startup_stm32f4xx.s
  *          before jump to __main
  * @param  None
  * @retval None
  */
#ifdef DATA_IN_ExtSRAM
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external SRAM mounted on STM324xG_EVAL board
  *         This SRAM will be used as program data memory (including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
/*-- GPIOs Configuration -----------------------------------------------------*/
/*
 +-------------------+--------------------+------------------+------------------+
 +                       SRAM pins assignment                                    +
 +-------------------+--------------------+------------------+------------------+
 | PD0  <-> FSMC_D2  | PE0  <-> FSMC_NBL0 | PF0  <-> FSMC_A0 | PG0 <-> FSMC_A10 |
 | PD1  <-> FSMC_D3  | PE1  <-> FSMC_NBL1 | PF1  <-> FSMC_A1 | PG1 <-> FSMC_A11 |
 | PD4  <-> FSMC_NOE | PE3  <-> FSMC_A19  | PF2  <-> FSMC_A2 | PG2 <-> FSMC_A12 |
```

```
 | PD5  <-> FSMC_NWE | PE4  <-> FSMC_A20  | PF3  <-> FSMC_A3 | PG3 <-> FSMC_A13 |
 | PD8  <-> FSMC_D13 | PE7  <-> FSMC_D4   | PF4  <-> FSMC_A4 | PG4 <-> FSMC_A14 |
 | PD9  <-> FSMC_D14 | PE8  <-> FSMC_D5   | PF5  <-> FSMC_A5 | PG5 <-> FSMC_A15 |
 | PD10 <-> FSMC_D15 | PE9  <-> FSMC_D6   | PF12 <-> FSMC_A6 | PG9 <-> FSMC_NE2 |
 | PD11 <-> FSMC_A16 | PE10 <-> FSMC_D7   | PF13 <-> FSMC_A7 |------------------+
 | PD12 <-> FSMC_A17 | PE11 <-> FSMC_D8   | PF14 <-> FSMC_A8 |
 | PD13 <-> FSMC_A18 | PE12 <-> FSMC_D9   | PF15 <-> FSMC_A9 |
 | PD14 <-> FSMC_D0  | PE13 <-> FSMC_D10  |------------------+
 | PD15 <-> FSMC_D1  | PE14 <-> FSMC_D11  |
 |                   | PE15 <-> FSMC_D12  |
 +-------------------+--------------------+
*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   = 0x00000078;

  /* Connect PDx pins to FSMC Alternate function */
  GPIOD->AFR[0]  = 0x00cc00cc;
  GPIOD->AFR[1]  = 0xcc0ccccc;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xaaaa0a0a;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xffff0f0f;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;

  /* Connect PEx pins to FSMC Alternate function */
  GPIOE->AFR[0]  = 0xc00cc0cc;
  GPIOE->AFR[1]  = 0xcccccccc;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xaaaa828a;
  /* Configure PEx pins speed to 100 MHz */
  GPIOE->OSPEEDR = 0xffffc3cf;
  /* Configure PEx pins Output type to push-pull */
  GPIOE->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PEx pins */
  GPIOE->PUPDR   = 0x00000000;

  /* Connect PFx pins to FSMC Alternate function */
  GPIOF->AFR[0]  = 0x00cccccc;
  GPIOF->AFR[1]  = 0xcccc0000;
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xaa000aaa;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xff000fff;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
```

```
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;

  /* Connect PGx pins to FSMC Alternate function */
  GPIOG->AFR[0]  = 0x00cccccc;
  GPIOG->AFR[1]  = 0x000000c0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00080aaa;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000c0fff;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;

/*-- FSMC Configuration ------------------------------------------------------*/
  /* Enable the FSMC interface clock */
  RCC->AHB3ENR         = 0x00000001;

  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001015;
  FSMC_Bank1->BTCR[3]  = 0x00010603;//0x00010400;
  FSMC_Bank1E->BWTR[2] = 0x0fffffff;
/*
  Bank1_SRAM2 is configured as follow:

  p.FSMC_AddressSetupTime = 3;//0;
  p.FSMC_AddressHoldTime = 0;
  p.FSMC_DataSetupTime = 6;//4;
  p.FSMC_BusTurnAroundDuration = 1;
  p.FSMC_CLKDivision = 0;
  p.FSMC_DataLatency = 0;
  p.FSMC_AccessMode = FSMC_AccessMode_A;

  FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM2;
  FSMC_NORSRAMInitStructure.FSMC_DataAddressMux = FSMC_DataAddressMux_Disable;
  FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_PSRAM;
  FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth = FSMC_MemoryDataWidth_16b;
  FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode = FSMC_BurstAccessMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_AsynchronousWait = FSMC_AsynchronousWait_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity = FSMC_WaitSignalPolarity_Low;
  FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive = FSMC_WaitSignalActive_BeforeWaitState;
  FSMC_NORSRAMInitStructure.FSMC_WriteOperation = FSMC_WriteOperation_Enable;
  FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ExtendedMode = FSMC_ExtendedMode_Disable;
  FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;
  FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &p;
```

```
  FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &p;
*/

}
#endif /* DATA_IN_ExtSRAM */


/**
  * @}
  */

/**
  * @}
  */

/**
  * @}
  */
/****************** (C) COPYRIGHT 2011 STMicroelectronics *****END OF FILE****/
```