```
/* SAM4S4B_pmc.h
 *
 * cferrarin@g.hmc.edu
 * kpezeshki@g.hmc.edu
 * 12/11/2018
 *
 * Contains base address locations, register structs, definitions, and
  functions for the PMC
 * peripheral (Power Management Controller) of the SAM4S4B microcontroller. */

#ifndef SAM4S4B_PMC_H
#define SAM4S4B_PMC_H

#include <stdint.h>

//////////////////////////////////////////////////////////////////////////////
 ///////////////////
// PMC Base Address Definitions
//////////////////////////////////////////////////////////////////////////////
 ///////////////////

#define PMC_BASE   (0x400E0400U) // PMC Base Address


//////////////////////////////////////////////////////////////////////////////
 ///////////////////
// PMC Registers
//////////////////////////////////////////////////////////////////////////////
 ///////////////////

// Bit field struct for the PMC_SCER register
typedef struct {
    volatile uint32_t      : 7;
    volatile uint32_t UDP  : 1;
    volatile uint32_t PCK0 : 1;
    volatile uint32_t PCK1 : 1;
    volatile uint32_t PCK2 : 1;
    volatile uint32_t      : 21;
} PMC_SCER_bits;

// Bit field struct for the PMC_MCFR register
typedef struct {
    volatile uint32_t MAINF    : 16;
    volatile uint32_t MAINFRDY : 1;
    volatile uint32_t          : 15;
} PMC_MCFR_bits;

// Bit field struct fo the PMC_PCK register
typedef struct {
    volatile uint32_t CSS  : 3;
    volatile uint32_t      : 1;
```

```
    volatile uint32_t PRES : 3;
    volatile uint32_t      : 25;
} PMC_PCK_bits;

// Peripheral struct for a PMC peripheral
typedef struct {
    volatile PMC_SCER_bits PMC_SCER;      // (Pmc Offset: 0x0000) System Clock
     Enable Register
    volatile uint32_t      PMC_SCDR;      // (Pmc Offset: 0x0004) System Clock
     Disable Register
    volatile uint32_t      PMC_SCSR;      // (Pmc Offset: 0x0008) System Clock
     Status Register
    volatile uint32_t      Reserved1[1];
    volatile uint32_t      PMC_PCER0;     // (Pmc Offset: 0x0010) Peripheral
     Clock Enable Register 0
    volatile uint32_t      PMC_PCDR0;     // (Pmc Offset: 0x0014) Peripheral
     Clock Disable Register 0
    volatile uint32_t      PMC_PCSR0;     // (Pmc Offset: 0x0018) Peripheral
     Clock Status Register 0
    volatile uint32_t      Reserved2[1];
    volatile uint32_t      CKGR_MOR;      // (Pmc Offset: 0x0020) Main
     Oscillator Register
    volatile PMC_MCFR_bits CKGR_MCFR;     // (Pmc Offset: 0x0024) Main Clock
     Frequency Register
    volatile uint32_t      CKGR_PLLAR;    // (Pmc Offset: 0x0028) PLLA Register
    volatile uint32_t      CKGR_PLLBR;    // (Pmc Offset: 0x002C) PLLB Register
    volatile uint32_t      PMC_MCKR;      // (Pmc Offset: 0x0030) Master Clock
     Register
    volatile uint32_t      Reserved3[1];
    volatile uint32_t      PMC_USB;       // (Pmc Offset: 0x0038) USB Clock
     Register
    volatile uint32_t      Reserved4[1];
    volatile PMC_PCK_bits  PMC_PCK[3];    // (Pmc Offset: 0x0040) Programmable
     Clock 0 Register
    volatile uint32_t      Reserved5[5];
    volatile uint32_t      PMC_IER;       // (Pmc Offset: 0x0060) Interrupt
     Enable Register
    volatile uint32_t      PMC_IDR;       // (Pmc Offset: 0x0064) Interrupt
     Disable Register
    volatile uint32_t      PMC_SR;        // (Pmc Offset: 0x0068) Status
     Register
    volatile uint32_t      PMC_IMR;       // (Pmc Offset: 0x006C) Interrupt
     Mask Register
    volatile uint32_t      PMC_FSMR;      // (Pmc Offset: 0x0070) Fast Startup
     Mode Register
    volatile uint32_t      PMC_FSPR;      // (Pmc Offset: 0x0074) Fast Startup
     Polarity Register
    volatile uint32_t      PMC_FOCR;      // (Pmc Offset: 0x0078) Fault Output
     Clear Register
    volatile uint32_t      Reserved6[26];
```

```
    volatile uint32_t      PMC_WPMR;      // (Pmc Offset: 0x00E4) Write Protect
     Mode Register
    volatile uint32_t      PMC_WPSR;      // (Pmc Offset: 0x00E8) Write Protect
     Status Register
    volatile uint32_t      Reserved7[5];
    volatile uint32_t      PMC_PCER1;     // (Pmc Offset: 0x0100) Peripheral
     Clock Enable Register 1
    volatile uint32_t      PMC_PCDR1;     // (Pmc Offset: 0x0104) Peripheral
     Clock Disable Register 1
    volatile uint32_t      PMC_PCSR1;     // (Pmc Offset: 0x0108) Peripheral
     Clock Status Register 1
    volatile uint32_t      Reserved8[1];
    volatile uint32_t      PMC_OCR;       // (Pmc Offset: 0x0110) Oscillator
     Calibration Register
} Pmc;

// Pointer to a Pmc-sized chunk of memory at the PMC peripheral
#define PMC ((Pmc *) PMC_BASE)


////////////////////////////////////////////////////////////////////////////
 ////////////////////
// PMC Definitions
////////////////////////////////////////////////////////////////////////////
 ////////////////////

// Values which "periph" can take on in pmcEnablePeriph()
#define PMC_ID_SUPC   ( 0) // Supply Controller (SUPC)
#define PMC_ID_RSTC   ( 1) // Reset Controller (RSTC)
#define PMC_ID_RTC    ( 2) // Real Time Clock (RTC)
#define PMC_ID_RTT    ( 3) // Real Time Timer (RTT)
#define PMC_ID_WDT    ( 4) // Watchdog Timer (WDT)
#define PMC_ID_PMC    ( 5) // Power Management Controller (PMC)
#define PMC_ID_EFC    ( 6) // Enhanced Embedded Flash Controller (EFC)
#define PMC_ID_UART0  ( 8) // UART 0 (UART0)
#define PMC_ID_UART1  ( 9) // UART 1 (UART1)
#define PMC_ID_PIOA   (11) // Parallel I/O Controller A (PIOA)
#define PMC_ID_PIOB   (12) // Parallel I/O Controller B (PIOB)
#define PMC_ID_USART0 (14) // USART 0 (USART0)
#define PMC_ID_USART1 (15) // USART 1 (USART1)
#define PMC_ID_HSMCI  (18) // Multimedia Card Interface (HSMCI)
#define PMC_ID_TWI0   (19) // Two Wire Interface 0 (TWI0)
#define PMC_ID_TWI1   (20) // Two Wire Interface 1 (TWI1)
#define PMC_ID_SPI    (21) // Serial Peripheral Interface (SPI)
#define PMC_ID_SSC    (22) // Synchronous Serial Controler (SSC)
#define PMC_ID_TC0    (23) // Timer/Counter 0 (TC0)
#define PMC_ID_TC1    (24) // Timer/Counter 1 (TC1)
#define PMC_ID_TC2    (25) // Timer/Counter 2 (TC2)
#define PMC_ID_ADC    (29) // Analog To Digital Converter (ADC)
#define PMC_ID_DACC   (30) // Digital To Analog Converter (DACC)
#define PMC_ID_PWM    (31) // Pulse Width Modulation (PWM)
```

```
#define PMC_ID_CRCCU  (32) // CRC Calculation Unit (CRCCU)
#define PMC_ID_ACC    (33) // Analog Comparator (ACC)
#define PMC_ID_UDP    (34) // USB Device Port (UDP)

// Values which the CSS bits in PMC_PCK[k] can take on; clock IDs
#define PMC_PCK_CSS_SLOW_CLK 0
#define PMC_PCK_CSS_MAIN_CLK 1
#define PMC_PCK_CSS_PLLA_CLK 2
#define PMC_PCK_CSS_PLLB_CLK 3
#define PMC_PCK_CSS_MCK      4

// Values which the PRES bits in PMC_PCK[k] can take on; clock division factors
#define PMC_PCK_PRES_CLK1  0
#define PMC_PCK_PRES_CLK2  1
#define PMC_PCK_PRES_CLK4  2
#define PMC_PCK_PRES_CLK8  3
#define PMC_PCK_PRES_CLK16 4
#define PMC_PCK_PRES_CLK32 5
#define PMC_PCK_PRES_CLK63 6

// Writing any other value in this field aborts the write operation of the WPEN
 bit.
// Always reads as 0.
#define PMC_WPMR_WPKEY_PASSWD (0x504D43U << 8)


////////////////////////////////////////////////////////////////////////////////
 ////////////////////
// PMC Functions
////////////////////////////////////////////////////////////////////////////////
 ////////////////////

/* Routes Master Clock to the desired peripheral, thereby enabling it.
 *    -- periphID: a PMC peripheral ID to enable, e.g. PMC_ID_PIOA */
void pmcEnablePeriph(int periphID) {
    PMC->PMC_PCER0 = 1 << periphID;
}

/* Returns the number of Main Clock cycles within 16 Slow Clock periods.
 *    -- return: the number of Main Clock cycles in 16 Slow Clock periods.
 * This is useful for calibrating the Main Clock (which the peripherals
  indirectly use) if using
 * a reliable crystal oscillator for the slow clock. Returns 0 if the master
  clock is disabled or
 * if the read value is invalid. The RC oscillator which the Slow Clock uses by
  default runs at
 * about 32 kHz, but this can vary fairly substantially between boards */
int pmcCheckMasterClk() {
    int valid = PMC->CKGR_MCFR.MAINFRDY; // Check if reported value is valid
    return valid ? PMC->CKGR_MCFR.MAINF : 0;
}
```

```
/* Initializes the programmable clock PCK2 for the FPGA to run at 1 MHz */
void pmcPCK2Init() {
    PMC->PMC_PCK[2].CSS = PMC_PCK_CSS_MCK; // Set clock source to master clock
    PMC->PMC_PCK[2].PRES = PMC_PCK_PRES_CLK4; // Divide clock by 4 for 1 MHz
    PMC->PMC_SCER.PCK2 = 1; // Enable programmable clock 0
}


#endif
```