```c
/* SAM4S4B_rtc.h
 *
 * cferrarin@g.hmc.edu
 * kpezeshki@g.hmc.edu
 * 12/11/2018
 *
 * Contains base address locations, register structs, definitions, and
  functions for the RTC (Real-
 * Time Clock) peripheral of the SAM4S4B microcontroller. */

#ifndef SAM4S4B_RTC_H
#define SAM4S4B_RTC_H

#include <stdint.h>

////////////////////////////////////////////////////////////////////////////////
 ///////////////////
// RTC Base Address Definitions
////////////////////////////////////////////////////////////////////////////////
 ///////////////////

#define RTC_BASE   (0x400E1460U) // RTC Base Address


////////////////////////////////////////////////////////////////////////////////
 ///////////////////
// RTC Registers
////////////////////////////////////////////////////////////////////////////////
 ///////////////////

// Bit field struct for the RTC_CR register
typedef struct {
    volatile uint32_t UPDTIM   : 1;
    volatile uint32_t UPDCAL   : 1;
    volatile uint32_t          : 6;
    volatile uint32_t TIMEVSEL : 2;
    volatile uint32_t          : 6;
    volatile uint32_t CALEVSEL : 2;
    volatile uint32_t          : 14;
} RTC_CR_bits;

// Bit field struct for the RTC_MR register
typedef struct {
    volatile uint32_t HRMOD : 1;
    volatile uint32_t       : 31;
} RTC_MR_bits;

// Bit field struct for the RTC_TIMR register
typedef struct {
    volatile uint32_t SEC  : 7;
    volatile uint32_t      : 1;
```

```
    volatile uint32_t MIN  : 7;
    volatile uint32_t      : 1;
    volatile uint32_t HOUR : 6;
    volatile uint32_t AMPM : 1;
    volatile uint32_t      : 9;
} RTC_TIMR_bits;

// Bit field struct for the RTC_CALR register
typedef struct {
    volatile uint32_t CENT  : 7;
    volatile uint32_t       : 1;
    volatile uint32_t YEAR  : 8;
    volatile uint32_t MONTH : 5;
    volatile uint32_t DAY   : 3;
    volatile uint32_t DATE  : 6;
    volatile uint32_t       : 2;
} RTC_CALR_bits;

// Bit field struct for the RTC_SR register
typedef struct {
    volatile uint32_t ACKUPD : 1;
    volatile uint32_t ALARM  : 1;
    volatile uint32_t SEC    : 1;
    volatile uint32_t TIMEV  : 1;
    volatile uint32_t CALEV  : 1;
    volatile uint32_t        : 27;
} RTC_SR_bits;

// Bit field struct for the RTC_SCCR register
typedef struct {
    volatile uint32_t ACKCLR : 1;
    volatile uint32_t ALRCLR : 1;
    volatile uint32_t SECCLR : 1;
    volatile uint32_t TIMCLR : 1;
    volatile uint32_t CALCLR : 1;
    volatile uint32_t        : 27;
} RTC_SCCR_bits;

// Peripheral struct for the RTC peripheral
typedef struct {
    volatile RTC_CR_bits   RTC_CR;     // (Rtc Offset: 0x00) Control Register
    volatile RTC_MR_bits   RTC_MR;     // (Rtc Offset: 0x04) Mode Register
    volatile RTC_TIMR_bits RTC_TIMR;   // (Rtc Offset: 0x08) Time Register
    volatile RTC_CALR_bits RTC_CALR;   // (Rtc Offset: 0x0C) Calendar Register
    volatile uint32_t      RTC_TIMALR; // (Rtc Offset: 0x10) Time Alarm
     Register
    volatile uint32_t      RTC_CALALR; // (Rtc Offset: 0x14) Calendar Alarm
     Register
    volatile RTC_SR_bits   RTC_SR;     // (Rtc Offset: 0x18) Status Register
    volatile RTC_SCCR_bits RTC_SCCR;   // (Rtc Offset: 0x1C) Status Clear
     Command Register
```

```
    volatile uint32_t      RTC_IER;    // (Rtc Offset: 0x20) Interrupt Enable
     Register
    volatile uint32_t      RTC_IDR;    // (Rtc Offset: 0x24) Interrupt Disable
     Register
    volatile uint32_t      RTC_IMR;    // (Rtc Offset: 0x28) Interrupt Mask
     Register
    volatile uint32_t      RTC_VER;    // (Rtc Offset: 0x2C) Valid Entry
     Register
} Rtc;

// Pointer to an Rtc-sized chunk of memory at the RTC peripheral
#define RTC ((Rtc*) RTC_BASE)


////////////////////////////////////////////////////////////////////////////////
 ////////////////////
// RTC Definitions
////////////////////////////////////////////////////////////////////////////////
 ////////////////////

// Values which the HRMOD bit in the RTC_SR register can take on
#define RTC_MR_HRMOD_24HR 0 // 24-hour mode (not supported by this driver)
#define RTC_MR_HRMOD_12HR 1 // 12-hour mode (used exclusively by this driver)

// (RTC does not have write protection).


////////////////////////////////////////////////////////////////////////////////
 ////////////////////
// RTC Functions
////////////////////////////////////////////////////////////////////////////////
 ////////////////////

/* Initializes the RTC peripheral's mode to 12-hour mode (24-hour mode is
 default).
   Note: there is no need to enable the clock with PMC with this peripheral. */
void rtcInit() {
    RTC->RTC_MR.HRMOD = RTC_MR_HRMOD_12HR; // Selects 12-hour mode
}

/* Updates the current time (seconds, minutes, hour, AM/PM) according to user
 values.
 *    -- sec: The current number of seconds in the current minute (0-59, BCD)
 *    -- min: The current number of minutes in the current hour (0-59, BCD)
 *    -- hour: The current hour in the current half of the day (1012, BCD)
 *    -- ampm: the current half of the day (AM = 0, PM = 1) */
void rtcUpdateTime(uint32_t sec, uint32_t min, uint32_t hour, uint32_t ampm) {
    while ((!RTC->RTC_SR.SEC));
    RTC->RTC_CR.UPDTIM = 1;
    while (!(RTC->RTC_SR.ACKUPD));
    RTC->RTC_SCCR.ACKCLR = 1;
```

```
    RTC->RTC_TIMR.SEC = sec;
    RTC->RTC_TIMR.MIN = min;
    RTC->RTC_TIMR.HOUR = hour;
    RTC->RTC_TIMR.AMPM = ampm;

    RTC->RTC_CR.UPDTIM = 0;
    RTC->RTC_SR.SEC = 0;
}

/* Updates the current date (date, day, month, year, century) according to user
 values.
 *    -- cent: The current century (19-20, BCD)
 *    -- year: The current year in the current century (0-99, BCD)
 *    -- month: The current month in the current year (1-12, BCD)
 *    -- day: The current day in the current week (1-7, BCD, arbitrary coding)
 *    -- date: The current day in the current month (1-31, BCD) */
void rtcUpdateDate(uint32_t cent, uint32_t year, uint32_t month,
    uint32_t day, uint32_t date) {
    while ((!RTC->RTC_SR.SEC));
    RTC->RTC_CR.UPDCAL = 1;
    while (!(RTC->RTC_SR.ACKUPD));
    RTC->RTC_SCCR.ACKCLR = 1;

    RTC->RTC_CALR.CENT = cent;
    RTC->RTC_CALR.YEAR = year;
    RTC->RTC_CALR.MONTH = month;
    RTC->RTC_CALR.DAY = day;
    RTC->RTC_CALR.DATE = date;

    RTC->RTC_CR.UPDCAL = 0;
    RTC->RTC_SR.SEC = 0;
}

/* Reads the current second in the current minute.
 *    -- return: the current second, in decimal */
int rtcReadSec() {
    int units = (RTC->RTC_TIMR.SEC) & 0xF;
    int tens = (RTC->RTC_TIMR.SEC) >> 4;
    return 10*tens + units;
}

/* Reads the current minute in the current hour.
 *    -- return: the current minute, in decimal */
int rtcReadMin() {
    int units = (RTC->RTC_TIMR.MIN) & 0xF;
    int tens = (RTC->RTC_TIMR.MIN) >> 4;
    return 10*tens + units;
}

/* Reads the current hour in the current half of the day.
```

```
 *    -- return: the current hour, in decimal */
int rtcReadHour() {
    int units = (RTC->RTC_TIMR.HOUR) & 0xF;
    int tens = (RTC->RTC_TIMR.HOUR) >> 4;
    return 10*tens + units;
}

/* Reads the current half of the day (AM or PM).
 *    -- return: the current half of the day, in decimal */
int rtcReadAmPm() {
    return RTC->RTC_TIMR.AMPM;
}

/* Reads the current century.
 *    -- return: the current century, in decimal */
int rtcReadCent() {
    int units = (RTC->RTC_CALR.CENT) & 0xF;
    int tens = (RTC->RTC_CALR.CENT) >> 4;
    return 10*tens + units;
}

/* Reads the current year in the current century.
 *    -- return: the current year, in decimal */
int rtcReadYear() {
    int units = (RTC->RTC_CALR.YEAR) & 0xF;
    int tens = (RTC->RTC_CALR.YEAR) >> 4;
    return 10*tens + units;
}

/* Reads the current month in the current year.
 *    -- return: the current month, in decimal */
int rtcReadMonth() {
    int units = (RTC->RTC_CALR.MONTH) & 0xF;
    int tens = (RTC->RTC_CALR.MONTH) >> 4;
    return 10*tens + units;
}

/* Reads the current day in the current week.
 *    -- return: the current day, in decimal (arbitrary coding) */
int rtcReadDay() {
    return RTC->RTC_CALR.DAY;
}

/* Reads the current day in the current month.
 *    -- return: the current day, in decimal */
int rtcReadDate() {
    int units = (RTC->RTC_CALR.DATE) & 0xF;
    int tens = (RTC->RTC_CALR.DATE) >> 4;
    return 10*tens + units;
}
```

```
#endif
```