```
/* SAM4S4B.h
 *
 * cferrarin@g.hmc.edu
 * kpezeshki@g.hmc.edu
 * 12/11/2018
 *
 * Top-level device driver for the SAM4S4B microcontroller.
 *
 * It is recommended to read the SAM4SB datasheet to understand the peripherals
  in this device
 * driver:
 * http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11100-32-bit%20Cortex-
  M4-Microcontroller-SAM4S_Datasheet.pdf
 *
 * This device driver provides minimal working support for the following
  peripherals:
 *   -- PMC  (Power Management Controller):
 *      -- For clock multiplexing to peripherals and controlling programmable
  clocks.
 *   -- PIO  (Parallel Input/Output Controller):
 *      -- For peripheral function pin multiplexing and reading and writing
  digital values from pins.
 *   -- TC   (Timer Counter):
 *      -- For system delays and counting and triggering at various clock
  speeds.
 *   -- SPI  (Serial Peripheral Interface):
 *      -- For serial communication with external devices that support SPI.
 *   -- UART (Universal Asynchronous Receiver-Transmitter):
 *      -- For serial communication with external devices that support UART.
 *   -- PWM  (Pulse Width Modulation Controller):
 *      -- For generating square waves of various frequencies and duty cycles.
 *   -- ADC  (Analog-to-Digital Converter):
 *      -- For reading analog voltages.
 *   -- RTC  (Real Time Clock):
 *      -- For automatic tracking of the time and date.
 *
 * Registers in this file are organized into structs in the following chain:
 *   -- Peripheral Struct (e.g. PCM, SPI) (in the case of PIO and TC, there are
  multiple)
 *      -- Channel Struct (e.g. TC_CH[k], PWM_CH[k]) (not always defined)
 *         -- Bit Field Struct (of type <Peripheral>_<Register>_bits struct)
  (not always defined)
 *         -- Register Struct (of type uint32_t struct)
 * The following are examples of how to access members of these structs:
 *   -- Access a register of a peripheral with no channels:
 *      <Peripheral Struct>-><Register Struct>
 *      Example: PIOA->PIO_PER
 *   -- Access a register of a peripheral with channels:
 *      <Peripheral Struct>->><Channel Struct[<Channel Number>]>.<Register
  Struct>
 *      Example: TC->TC_CH[2].TC_CV
 *   -- Access a bit of a peripheral with no channels:
 *      <Peripheral Struct>-><Bit Field Struct>.<Bit Name>
 *      Example: PMC->PMC_SCER.PCK2
 *   -- Access a bit of a peripheral with channels:
 *      <Peripheral Struct>-><Channel Struct[<Channel Number>]>.<Bit Field
  Struct>.<Bit Name>
 *      Example: TC->TC_CH.TC_CCR.CLKEN
 *
 * The main clock for peripherals is rated at 4 MHz but utilizes an RC
  oscillator, which is cheap
 * and consumes little power but can be inaccurate. As such, it is necessary to
  verify the clock's
 * frequency. This can be done by running the FPGA clock with samInit():
 *    #include "SAM4S4B.h"
 *    int main() {
 *        samInit();
 *    }
 * Observe pin PIO_PA31 and record its frequency. This will be MCK_FREQ divided
  by four, so multiply
 * the value by 4 and record this accurate MCK frequency in the #define
  directive in SAM4S4B_sys.h.
 *
 * Start your main.c file with the following lines:
 *    #include "SAM4S4B.h"
 *    int main() {
 *        samInit();
 *        // Your code goes here
 *    }
 * Remember to intialize each peripheral with its init function before using it
  (although PIO is
 * intialized automatically through samInit()), and enjoy!
 */

#ifndef SAM4S4B_H
#define SAM4S4B_H

#include "SAM4S4B_sys.h"
#include "SAM4S4B_pmc.h"
#include "SAM4S4B_pio.h"
#include "SAM4S4B_tc.h"
#include "SAM4S4B_spi.h"
#include "SAM4S4B_uart.h"
#include "SAM4S4B_pwm.h"
#include "SAM4S4B_adc.h"
#include "SAM4S4B_rtc.h"

////////////////////////////////////////////////////////////////////////////////
 /////////////////////
// Top-Level Functions
////////////////////////////////////////////////////////////////////////////////
 /////////////////////
```

```c
// Sets up the clock for the FPGA at 1 MHz
void samInit() {
    pioInit();
    pioPinMode(PIO_PA31, PIO_PERIPH_B);
    pmcPCK2Init();
}


#endif
```