Topic:

Setup the SAM3X embedded Real Time Clock (RTC) for operation on the Arduino DUE

The RTC is one of several embedded peripherals of the SAM3X Microcontrollers by Atmel®. A pre-defined number identifies each Peripheral. This number is used to control the Peripheral Interrupt and the Peripheral Clock. In the case of the RTC, its ID = 2. A Slow Clock (SLCLK) derived from an embedded 32.768 KHz Oscillator clocks the RTC and other Peripherals. There are two 32KHz Oscillators embedded:

RC_OSC32k
XTAL_OSC32k

An embedded RC circuit sets the frequency of the RC_OSC32k Oscillator. The XTAL_OSC32k Oscillator frequency is set by an external 32.768 KHz Quartz Crystal connected to the XIN32 and XOUT32 pins. The user may choose which Oscillator should source the SLCLK assuming that the external crystal is connected. By default, at power up or reset, the RC_OSC32k Oscillator is selected. It is a less accurate Oscillator as compared to the XTAL_OSC32k Crystal Oscillator. On the Arduino DUE, the SAM3X's XIN32 and XOUT32 pins (pins 48 and 49 respectively) are connected to such a crystal (Y2) and is assumed to be very accurate. In our exercise, the XTAL_OSC32k Oscillator will be used to clock the RTC and other peripherals. Note that the RTC and a few other Peripherals are continuously clocked.

To use the RTC, it should be initialized with these steps:

1. Select the Oscillator (XTAL or RC 32K) to clock the RTC.
2. Initialize the RTC Interrupt handlers.
3. Set the current Time and Date.

These steps require interaction with the embedded System Controller User Interface. The specific sub-systems are:

Supply Controller	(SUPC)
Power Management Controller	(PMC)
Real Time Clock	(RTC)

Their pertinent internal registers are:

SUPC_CR	Control
SUPC_MR	Mode
SUPC_SR	Status
RTC_CR	Control
RTC_MR	Mode
RTC_TIMR	Time
RTC_CALR	Calendar
RTC_SR	Status
RTC_SCCR	Status Clear
RTC_VER	Valid Entry
RTC_WPMR	Write Protect Mode

Oscillator Selection:

The SAM3X8 data sheets (Chapter 18) describers the Supply Controller (SUPC). The Slow Clock Generator is part of the SUPC and is covered in section (18.4.2).

The 32KHz Oscillator selection is done in two steps as follows:

1. Set the XTALSEL bit to 1 (bit 3) in the SUPC_CR register.
2. Wait for the OSCSEL bit (bit 7) in the SUPC_SR to indicate when the switch is complete.

These steps are accomplished by using the Atmel® Software Foundation (ASF) libraries. The specific lines of code are:

```
pmc_switch_sclk_to_32kxtal(PMC_OSC_XTAL);
while (pmc_osc_is_ready_32kxtal() == 0);
```

Initialize RTC Interrupt Handlers:

In addition, the Nested Vector interrupt Controller (NVIC) registers must be initialized by the following code:

```
NVIC_DisableIRQ(RTC_IRQn);
NVIC_ClearPendingIRQ(RTC_IRQn);
NVIC_SetPriority(RTC_IRQn, 0);
//   NVIC_EnableIRQ(RTC_IRQn);
```

Date and Time set:

The date and time settings are done with these lines of code:

```
uint8_t hr = 18, min = 29, sec = 0;
uint8_t mm = 4, dd = 3, wkd = 3;
uint16_t yyyy = 2013;

RTC_SetHourMode(RTC, 0);   // Set 24 hr mode
while (RTC_GetHourMode(RTC) == 1);
RTC_SetTime(RTC, hr, min, sec);
RTC_SetDate(RTC, yyyy, mm, dd, wkd);
```

Conclusion:

The setup is quite simple once you have a good understanding of C++ *pointers* and *struct*. The SAM3X data sheets are difficult to follow. I found that it couldn't be used simply as reference. In fact, it requires reading most of the documentation in order to start making sense of the µcontroller and its embedded Peripherals (all information is there but it is laid out across several sections that initially seem unrelated). One area I struggled with was which Peripherals are continuously clocked vs. which can be orchestrated. A table is outlined but not implicitly explained. The ASF provides a wealth of knowledge once you have struggled with the Data Sheet (and I am a hardware person).

References:

Atmel SAM3X Series Data Sheet

Atmel ASF 3.7.3