# An alarm primer for the Maxim Integrated DS3231 and DS3232 Real-Time Clocks

Since I wrote the [DS3232RTC library](https://github.com/JChristensen/DS3232RTC), I frequently receive questions regarding setting the alarms on the DS323x RTCs. It's always best to read the datasheet ([DS3231](https://datasheets.maximintegrated.com/en/ds/DS3231.pdf), [DS3232](https://datasheets.maximintegrated.com/en/ds/DS3232.pdf)); the datasheets for these RTCs have an "Alarms" section that does a good job of explaining the alarms. However, reading datasheets may not be everyone's cup of tea; indeed the writing style can be challenging at times. Therefore I will try to provide some pointers here in what is hopefully a little plainer language.

It's probably natural that when we think about an "alarm" we think of the clock radio that we might have on our nightstand. Pretty much everyone is familiar with how these devices work but the first thing we need to do is forget that paradigm because the DS323x alarms aren't exactly like clock radio alarms. My advice is: Don't have any preconceived notions as to how an "alarm" works. First understand how the RTC alarms work, and only then, consider how to apply them to your particular application.

## DS323x alarm basics

1. Alarms are always on. Unlike our clock radio, there is no way to turn them off. However, when an alarm condition occurs, it just causes a flag to be set in the RTC. So, if we don't want an alarm, it's as easy as ignoring the flag. That's not quite the whole story, as setting the flag may only be the first step that leads to other effects. More on that later.

2. An alarm condition occurs when an alarm register matches the RTC time registers. That's not quite the whole story either, because there are mask bits that cause only certain parts of the alarm registers (e.g. minutes and seconds but not hours or day/date) to participate in the match.

3. Once an alarm match occurs and the flag is set, the flag remains set until we tell the RTC to clear it.

4. There are two sets of alarm registers. Alarm 1 consists of day or date, hours, minutes and seconds. Alarm 2 doesn't have seconds, only day or date, hours and minutes. "Day" means day of the week, i.e. a number 1–7. "Date" means day of the month, i.e. a number 1–31. Alarms can be set for day or date, but not both at the same time.

5. The RTC alarms are not infinitely flexible. There are many practical alarm situations that cannot be handled by RTC alarms alone. This is seldom a show-stopper as a little fairly straightforward application code will usually do the trick. For example, unlike the time keeping registers, the alarm registers do not contain month or year. So to create an alarm that only occurs for a certain month, some application code will be required. Perhaps a more common scenario that cannot be addressed directly with the RTC alarms is (for example) to have an alarm every 10 seconds. There are a limited number of alarm scenarios where we can just set the alarm registers once and then forget them. To implement other scenarios may require setting the alarm registers to a new value every time an alarm match occurs.

6. There are six ways to set Alarm 1 (see the [`ALARM_TYPES_t` enumeration](https://github.com/JChristensen/DS3232RTC#alarm_types_t) in the library):
    * Alarm once per second
    * Alarm when seconds match (i.e. once per minute)
    * Alarm when minutes and seconds match (i.e. once per hour)
    * Alarm when hours, minutes, and seconds match (i.e. once per day)
    * Alarm when day, hours, minutes, and seconds match (i.e. once per week)
    * Alarm when date, hours, minutes, and seconds match (i.e. once per month)

7. There are five ways to set Alarm 2 (see the [`ALARM_TYPES_t` enumeration](https://github.com/JChristensen/DS3232RTC#alarm_types_t) in the library):
    * Alarm once per minute (at 00 seconds of every minute)
    * Alarm when minutes match (i.e. once per hour)
    * Alarm when hours and minutes match (i.e. once per day)
    * Alarm when day, hours and minutes match (i.e. once per week)
    * Alarm when date, hours and minutes match (i.e. once per month)

## Alarm flags and interrupts
Earlier I mentioned that when an alarm match occurs, an internal flag is set in the RTC. There is a flag for each alarm; the datasheet calls them A1F and A2F. The library has an [`alarm()`](https://github.com/JChristensen/DS3232RTC#alarmbyte-alarmnumber) function that will return the value of the alarm flag for either alarm, and if it was set, the [`alarm()`](https://github.com/JChristensen/DS3232RTC#alarmbyte-alarmnumber) function will also reset it.

The RTCs also have an "interrupt" option that causes a pin (called INT/SQW) on the RTC to be asserted when an alarm flag is set. It's just a logic output pin, high or low, but it's common to connect the INT/SQW pin to cause an interrupt on a microcontroller, hence the name.

The INT/SQW is a multi-function pin. It either outputs a continuous square wave (SQW) of various frequencies, or it acts as an interrupt output pin. In order to act as an interrupt output pin, the INTCN (interrupt control) bit in the RTC's control register needs to be set to 1. This is the default power-on state, but good programming practice would be to always initialize it according to the desired behavior.

Note also that the INT/SQW pin requires a pull-up resistor and that it is active low, i.e. it is driven to a low logic level when an alarm occurs, else it is high. To configure the INT/SQW pin to act as an interrupt output with the library, disable square wave output by calling [`squareWave(DS3232RTC::SQWAVE_NONE)`](https://github.com/JChristensen/DS3232RTC#squarewavesqwave_freqs_t-freq).

One final condition is required to cause an alarm match to assert the INT/SQW pin. The *alarm interrupt enable* bit in the RTC control register must be set to 1. There is a bit for each alarm, called A1IE and A2IE. Using the library, these bits can be set or cleared by calling the [`alarmInterrupt()`](https://github.com/JChristensen/DS3232RTC#alarminterruptbyte-alarmnumber-boolean-alarmenabled) function.

Since there is no guaranteed power-on state for the alarm registers, good practice would be to set both to known values. Also, when INT/SQW is configured as an alarm interrupt output, turn off the interrupt enable bit for an alarm that is not being used.

## Examples
The library contains several example sketches. Here are short descriptions of each.

### alarm_ex1
Sets Alarm 1 to occur once per minute at five seconds after the minute. Detects the alarm by polling the RTC alarm flag.

### alarm_ex2
Sets Alarm 1 to occur once a minute at 5 seconds after the minute. Configures the RTC INT/SQW pin to be asserted when alarm match occurs. Detects the alarm by polling the INT/SQW pin.

### alarm_ex3
Sets Alarm 1 to occur every 10 seconds. Detects the alarm by polling the RTC alarm flag. Note that the RTC does not have an alarm mode for every 10 seconds, so after an alarm occurs, we reset the alarm register to the current time plus ten seconds.

### alarm_ex4
Set Alarm 1 to occur every second. Detects the alarm by polling the RTC alarm flag.

### alarm_ex5
Set Alarm 2 to occur once per minute. Detects the alarm by polling the RTC alarm flag.

### alarm_ex6
Sets Alarm 1 for 20 seconds after each minute and Alarm 2 for each minute (at 00 seconds). Configures the INT/SQW pin to be asserted when either alarm is raised and uses this signal to generate an interrupt on the Arduino. When an interrupt occurs, the RTC alarm flags are checked to determine which alarm was raised.

### alarm_ex7
Sets Alarm 2 for a specific time of day, hh:mm. Configures the RTC INT/SQW pin to be asserted when alarm match occurs. Detects the alarm by polling the INT/SQW pin.

### alarm_ex8
Sets both alarms to occur once per day at different times. Configures the RTC INT/SQW pin to be asserted when alarm match occurs. Detects the alarms by polling the INT/SQW pin. Assumes the RTC time is already set.

### alarm_ex9
Sets Alarm 1 to turn on an LED at a given time, and Alarm 2 to turn it off at another time. Detects the alarms by polling the alarm flags.

### alarm_ex10
Sleeping MCU example. Wakes the MCU at regular intervals with an interrupt generated by an RTC alarm. After waking, the built-in LED is turned on for several seconds, then the MCU sleeps until the next alarm.