What we know about the Atmel SAM3X Low-power modes

**SAM3X has 3 low-power modes- Wait, Backup, and Sleep**. Sleep and Wait had a negligible reduction in power consumption, but Backup mode conserves large amounts of energy. The tests done can be accessed [here](https://docs.google.com/document/d/1C3eieFO_A10apTFMJ8oy9yceoFG9tSve6rcSzVF03Qw/edit?usp=sharing).

Backup mode is much easier to get into than it is to get out of, it seems. Using the function pmc\_enable\_backupmode() turns off everything in the CPU except the RTT(Real-Time-Timer), the RTC(Real-Time-Clock), and the SUPC(Supply Controller).

**Backup mode can be waken up by any of these 3 peripherals**. However, this is easier said than done. Documentation of these peripherals is listed below, courtesy of Atmel.

[RTT](http://asf.atmel.com/docs/latest/sam3x/html/group__sam__drivers__rtt__group.html#func-members)

[RTC](http://asf.atmel.com/docs/latest/xmegac/html/group__rtc__group.html)

[SUPC](http://asf.atmel.com/docs/latest/sam3n/html/group__sam__drivers__supc__group.html)

Atmel also has examples of these wakeup methods on their website(Examples [1](http://asf.atmel.com/docs/latest/sam.applications.low_power.sam3x_ek/html/index.html) and [2](http://asf.atmel.com/docs/3.15.0/avr32.drivers.rtc.example.evk1100/html/index.html)). However, you cannot directly view the source code for these examples, you can only view it in Atmel Software Framework(ASF), Atmel’s IDE. ASF is only available on Windows, so Linux and macOS users(Like me) will not have access to the examples.

**More info on the RTC/RTT wakeup method**

While no example code has been provided, everyone on the Arduino and Atmel forums agrees that the best way to wake from backup mode periodically is to set an RTC/RTT alarm, chain the alarm to an interrupt, and send the CPU into backup mode. Then, when the alarm goes off as you set it, wakes up the CPU with the chained interrupt and runs whatever code you want periodically. Illustrated below. 

**Working wakeup method(RTT)**

The code below, found on the Arduino forums [here](https://forum.arduino.cc/index.php?topic=469943.0) wakes up the Due after a certain amount of seconds, using the internal RTT.

|  |
| --- |
| #define **SUPC\_KEY** (0xA5) const uint32\_t AL = 5; // Alarm Timestamp = AL \* T seconds  void **setup**() {  Serial.begin(250000);  RTT->RTT\_MR = RTT\_MR\_RTTRST  | RTT\_MR\_RTPRES (0x8000) // T = 1 second  | RTT\_MR\_ALMIEN;    RTT->RTT\_AR = RTT\_AR\_ALMV(AL);    NVIC\_DisableIRQ ( RTT\_IRQn );  NVIC\_ClearPendingIRQ ( RTT\_IRQn );  NVIC\_SetPriority ( RTT\_IRQn, 0 );  NVIC\_EnableIRQ ( RTT\_IRQn );   SUPC->SUPC\_WUMR |= SUPC\_WUMR\_RTTEN;   Serial.println("\n\rHELLO ALL !!");  Serial.print("GPBR[0] = ");Serial.println(GPBR->SYS\_GPBR[0]);  delay(1000); }  void **loop**() {   GPBR->SYS\_GPBR[0] += 1; // Increment one of the 8 General Purpose Backup Registers  if (GPBR->SYS\_GPBR[0] < 4) {  Serial.print("Entering backup mode : Do NOT stop the software now |:"); Serial.end(); // Exit from Serial  //alternate way to enter backup mode  SUPC->SUPC\_CR = SUPC\_CR\_VROFF\_STOP\_VREG | SUPC\_CR\_KEY(SUPC\_KEY);   }  Serial.println("Definitive Exit from backup mode"); Serial.println("Now you can stop the software"); Serial.println(GPBR->SYS\_GPBR[0]);  delay(5000);  }  void **RTT\_Handler** () {  RTT->RTT\_SR; } |