10.8 Triggered synchSMs

10.9 Reducing power consumption using a sleep function

Microcontrollers consume power while running a program. Commonly a microcontroller is executing a while loop that waits for some event to occur, such as `while (!TimerFlag) {}` or just `while (1) { }` . Executing that loop is somewhat wasteful because the programmer knows nothing will actually happen until the hardware generates an interrupt, causing an ISR to be called.

Thus, microcontrollers typically have a special low-power mode, known as ***sleep*** mode, in which the microcontroller stops the microcontroller's CPU and thus stops executing the program, but continues to operate other hardware like the timer and the UART. When in sleep mode, the microcontroller consumes substantially less power. When such hardware experiences a particular event, like the timer ticking or the UART receiving data, the hardware "wakes up" the microcontroller and typically calls an ISR, after which the microcontroller resumes executing the program after the instruction that put the processor to sleep. RIM can be put to sleep by calling a function named Sleep(). RIM will wake up when an ISR is automatically called.

In general, knowing when to call the sleep function in a program can be challenging to a programmer. Fortunately, the disciplined approach using synchSMs translated to C greatly simplifies the issue. The programmer merely need add a sleep call where the program waits for the TimerISR to be called. For an earlier version of C code with the scheduler loop in main(), the programmer merely need add a sleep call as follows.

Figure 10.9.1: Using sleep function to reduce power.

```
printf("Going to sleep...");
Sleep(); // Put processor in low-power mode waiting for wakeup event
// Below instructions won't execute until hw event wakes processor
// and calls appropriate ISR
printf("Waking up.\r\n");
while(!TimerFlag && !RxFlag);
```

If sporadic tasks are included (i.e., tasks that can enter a special Inactive state and thus stop ticking), the amount of time spent asleep could be further improved by dynamically adjusting the Timer's tick rate to be the GCD of the periods for all *active* tasks, rather than for all tasks.

Participation Activity 10.9.1: Low-power modes in microcontrollers.

#	Question	Your answer
1	Most microcontrollers can be put into a low-power mode. ✔ Low-power "sleep" modes can drastically reduce power consumption.	True False
2	A microcontroller usually wakes up from a low-power state when the program writes a certain value to a variable. ✔ Microcontroller hardware wakes up the processor when certain events occur, like receiving a UART byte.	True False
3	RIM can be put into a low-power mode via the Sleep() function. ✔ RIM will wake up if the timer interrupt or UART ISR handlers are executed.	True False

As microcontrollers are commonly powered by batteries, sleep modes are extensively used; a microcontroller may be asleep 99% of the time or more, waking up for brief periods, performing a burst of processing, and going back to sleep again. Assume a microcontroller consumes 1 milliWatts (mW) of power while running, and 1 microWatt when asleep (typical values for a low-cost low-power 8-bit microcontroller). Suppose a small battery stores about 1,000 Joules (J), where 1 W is 1 J/sec. Thus, the battery can power a running microcontroller for 1,000 J * (1 / 0.001 J/sec) = 1,000,000 seconds, meaning 277 hours, or 11 days. However, if the microcontroller is kept asleep 99% of the time (as is not uncommon if task periods are made as large as possible and task computations are short), then the battery can power the microcontroller for 1,000 J * (1%*(1/0.001) + 99%*(1/0.000001)) = 990,010,000 seconds, meaning 275002 hours, or 11,458 days.

An interesting power technology for embedded systems is that of ***supercapacitors***, which store energy using capacitance rather than typical battery technology. Supercapacitors charge more quickly, can be recharged many more times (typical rechargeable batteries may lose their ability to store energy after thousands of charging cycles), but presently can't store as much energy (perhaps 1/10th). However, modest-sized supercaps may store several thousand Joules, and thus could power a low-power microcontroller for days or years (depending on sleep time).

Participation Activity 10.9.2: Battery lifetime.

#	Question	Your answer
1	Suppose a battery stores 5,000 J. For how many seconds can that battery power a desktop processor running at 100 W? ✔ 50 5,000 J * (1 sec / 100 J) = 50 sec. That battery, about the capacity of a AAA or AA battery, can only power a desktop processor for less than a minute.	50 seconds
2	Suppose a battery stores 5,000 J. For how many seconds can that battery power a high-end embedded processor running at 1 W (typical for a 32-bit embedded processor like an ARM)? ✔ 5000 5,000 J * (1 sec / 1 J) = 5000 sec. That battery, about the capacity of a AAA or AA battery, can power the processor for about 5000 / 3600 = 1.4 hours.	5000 seconds
3	Suppose a battery stores 5,000 J. For how many seconds can that battery power a high-end embedded processor running at 1 W, if that processor is asleep 80% of the time and consumes 1 mW while asleep? ✔ 4001000 5,000 J * ((20% * 1s/1W) + (80% * 1s/0.001W)) = 4001000. That battery can power the processor for about 4001000 / 3600 = 1112 hours, or 46 days.	4001000 seconds

Exploring further:

- Joules in typical batteries
- Supercapacitors

10.10 Disciplined programming