Sketches

How to stop an Arduino sketch

There are a number of ways to stop an Arduino sketch from running, also this depends on what you expect 'stopped' to be. I say this as the Arduino is not using an operating system, your sketch can never really exit as it is all the AVR has to work with. The best we can do is pause it or put it into a stopped state. There are also better alternatives to actually halting the device.

Stop the Arduino manually.

When using the normal Arduino method of writing a sketch using setup() and loop(), your program will run forever, meaning the only time your Arduino can be considered stopped is when the power is removed. To put the Arduino into a stopped state, we can simply use an infinite loop to effectively lock the CPU, well almost...

```
while( true ){
  //An empty loop that never ends.
}
```

This is not quite the result we are looking for yet, as the Arduino uses a mechanism called interrupts to provide a large amount of functionality, and they pause the main program flow to run. Even basic sketches could use an interrupt behind the scenes. So to complete the code to 'stop' the CPU we can add the AVR provided function cli() to disable global (or all) interrupts.

```
cli();
while( true ); //An empty loop.
```

Stop the Arduino using built-ins.

The above is functionally equivalent to calling the C function abort(). Also if you override main()[1] and provide your own implementation, when it returns, abort() is implicitly called. Another possibility is a function called exit(), and is equivalent to calling abort(). So there are three methods to halt the Arduino, and each one does the same thing, however each compiled to a different size on testing. I'll leave it with you to find out which is best in your particular setup.

These are of course not the best methods. The sketch is still looping at full speed and is completely locked until the power is cycled, either physically or through use of the reset button. Advanced power management could be used to put the device to sleep and use less power, however if the device is recoverable without a reset, the device becomes far more user friendly.

Put the Arduino to sleep.

The AVR documentation for "Power Management and Sleep Modes" has some examples of sleeping the micro controller. Sleeping can be used for both a controlled pause, or a complete lock, similar to the infinite loop explained above. There are also different modes of sleep, each with different power saving potential[2].

Sketches

The most basic example of using sleep to halt a device until reset is shown below. The sleep_disable() function is not needed as the device will never wake up due to interrupts being turned off.

```
cli();
sleep_enable();
sleep_cpu();
```

Tag notes:

1. Arduino provides a version of main() in its core API, it calls setup() and repeatedly calls loop(), providing the basic Arduino program flow. You can override the main function and setup your Arduino however you like, see this FAQ for information on how to safely do this: Can I use int main() with Arduino?
2. Unfortunatly, the Arduino boards are quite power hungry and many power saving features of the AVR may not help much at all. That being said, porting your design to a custom and more efficient design could allow you to gain significant battery life using power saving techniques.

Unique solution ID: #1046
Author: Christopher Andrews
Last update: 2014-05-14 12:17