Programming Arduino
Why does this loop never end

Have you ever written a loop that decrements its index, or traverses an array in reverse order only to be stumped why it does not work?

If so, it is likely a result of whats called an **unsigned *integer overflow***. Take a look at the example below.

```
for( byte index = 10 ; index >= 0 ; --index ){

  //loop contents.
}
```

This loop seems fairly straight forward, however it is a typical example of integer overflow. The data type byte is an Arduino defined type which is really an unsigned char in disguise. The difference between a signed value and an unsigned value is that only signed values can be negative.

On what should be the last iteration of this loop, index is decremented. Being an unsigned value, it wraps around to its maximum value of 255 or 0xFF. This may seem confusing at first, but the hidden reason is a signed value of -1 is represented in binary the same as an unsigned value of 255. Signed and unsigned values are not convertible, so when cast or treated as its alternative, the data is used unchanged.

This means the loop never ends and will block the Arduino until a power cycle, upload, or an error caused by something in the loop running too many times (which will still block execution). If the index comes from elsewhere and should be unsigned, you can modify the loop to avoid the overflow.

```
byte index = 10;

for( ++index ; index ; --index ){

  const byte idx = index - 1;
  //loop contents use idx instead of index.
}
```

In this version, index is initially incremented one so the check can be index >= 1, which as you can see is optimised to simply converting index implcitly to a bool. And this prevents an overflow on the --index. Inside the loop the real index is recovered and can be used.

Unique solution ID: #1030
Author: Christopher Andrews
Last update: 2014-05-29 16:56