Modifications to LiquidCrystal for the Arduino

This version of LiquidCrystal supports the API of LiquidCrystal in Arduino 17 fully but does not actually ever use 8 data pins. This change makes the code smaller, saving flash RAM and also less complex, making it easier to read.

Linewrap

When you declare the dimensions of the LCD in your begin call, the LiquidCrystal library remembers how long the lines are. Now when it reaches the end of line 1, text wraps onto line 2 (not line 3 as previously).

println

Although print has worked properly in the past, println has not. Now the '\r' and '\n' characters are not sent to the screen as though they were visible characters and the '\r' resets the character position to the top of the next line.

16x4 LCDs

The begin statement also correctly positions text at the beginning of the line on 16x4 LCDs, which were not correctly handled before.

setCursor

In the past setCursor selected a location in the HD44780's RAM not actually a screen location. If you use any of the commands that shift the display left or right with the previous routines, then setCursor and print, text appears in an unexpected location on the screen. With the new software, if you call either scrollDisplayLeft() or scrollDisplayRight(), the LiquidCrystal package keeps track of the relationship between RAM and the LCD so that setCursor coordinates are pegged to a specific spot on the screen, rather than a spot in RAM. The sotware does not handle autoScroll, however. Call home() after autoScroll to restore the expected relationship between setCursor and the LCD screen.

Testing the LCD Busy Flag

Previous versions of LiquidCrystal always used timed delays on the Arduino side of the interface to give the LCD module enough time to complete its operation. This version still does that if you tie the RW pin to ground and do not tell LiquidCrystal what that pin number is or pass it the address of a user routine to test the busy flag. If you do specify RW now, however, the software will poll the busy flag on the LCD module. Arduino operations may thus overlap LCD operations and potentially things may go a little faster.

Syntactic Sugar

#include <Streaming.h> from http://arduiniana.org/2009/04/new-streaming-library/
Then you can combine that with an overloading of the () operator in this code. This lets you specify screen location and chain print commands together by writing:
lcd(column,line)<<"a="<<a;

Streaming.h is so efficient you may actually save a few bytes of memory!

Speed testing
All of the interface modes go faster than the eye can follow. This version of the software is even slower than the previous version when using timed delays. I found that my LCD (Axman) needed an even longer delay than before. In the interests of making the code foolproof, I lengthened the delays to make that LCD work. However Paul Stoffregen has significantly speeded up the code when testing the busy flag and so those options run significantly faster than before. I compared the speeds of the different interfaces--writing 80 characters to the screen then 80 blanks and looping through that 20 times. The results on a Mega are:

```
Axman 4 data pins no RW 1313 milliseconds  |  nonAxman 1313
Axman 4 data pins  + RW  561 milliseconds  |  nonAxman  455
```

Crazy 8 Addressing
16x1 LCDs often have an unusual address layout; these modules often have two 8 character halves and work best with this software if you declare them as lcd.begin(8,2); if you do that, then you can print("abcdefghilklmno"); and have all the characters appear as you would like across the screen. This works because of the linewrap fix. If you use any of the scrolling commands, the bizarre addressing of these modules will manifest itself. For details follow the _LCD Addressing_ link at web.alfredstate.edu/weimandn

Disadvantages
The only real disadvantage I can see to the changes I have made is the possibility that someone with a little 16x2 LCD is using the scrollDisplayLeft() or scrollDisplayRight() instructions to move data across the screen, but wants to write the data 40 characters at a time with a print statement. This version really does not let the user write data to the HD44780 DDRAM which is not visible. To accomplish a scrolling display with 40 characters per line, you would now need to write 16 characters, scroll the display, write a little more and so on.

There are going to be some incompatibilities between code that assumed that line 1 wrapped onto line 3, etc.

Directions for the future
Paul Stoffregen suggested making the internal function send() public rather than private so that classes that inherit from LiquidCrystal can provide an alternate function, presumably hard coding pins and getting still more speed. I made all of the 'private' functions and instance variables in LiquidCrystal 'protected'. I think this allows the same functionality, but have not yet tested this.

Thanks
Certainly my efforts would not have been possible without the help and prior efforts of David Mellis, Limor Friede, and Donald Weiman. Don was particularly patient in

guiding me through the idiosyncracies of the HD44780 based LCDs and especially in supplying an example of how the busy flag could be tested. Paul Stoffregen contributed significant optimization of the code to make it faster.