Modifications to LiquidCrystal for the Arduino

I made several modifications to the LiquidCrystal library module from Arduino17:

40x4 LCDs
I added support for an LCD of 4 LInes and 40 characters. I think that if 24x4, 32x4 LCDs exist, they would also work with the software as I have modified it although I have not had the opportunity to test that. The 40x4 LCD (and any HD44780 based LCD with between 81 and 160 characters) will have 2 enable lines. To use an LCD with 4 lines and 40 columns you would declare your LiquidCrystal object as:
LiquidCrystal lcd(RS,RW,Enable1,Enable2, data3,data2,data1,data0); at this time I don't support 8 data lines. nor do I support eliminating the RW line in this option, although it may be possible to pass 255 as the RW item and get that behavior.
Then in the setup function you would call:
lcd.begin(40,4);

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).

16x4 LCDs
The begin statement also correctly positions text at the beginning of the line on 16x4 (and 40x4) 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. 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.

Speed testing
All of the interface modes go faster than the eye can follow. 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 are:
8 data pins + RW 431 milliseconds
4 data pins + RW 532 milliseconds
8 data pins - RW 641 milliseconds
4 data pins - RW 687 milliseconds

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. 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 two real disadvantages I can see to the changes I have made are:

1. The code is longer than before. Much of the increase is in checkLcdBusyFlag() and this could be fairly easily replaced with delayMicroseconds(100);

2. 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
I see little purpose to retaining the 8 pin interface options. Even in the slowest situation output to the LCD goes faster than the eye can follow. The slowest situation would be using the 40x4 LCD, doing a scrollDisplayLeft() or scrollDisplayRight() and then printing 160 characters. When the display has been scrolled, the software actually calls setCursor internally with every character sent to the LCD, so you would be communicating 480 items to the LCD in that instance. As I have tested the various interfaces with the various shapes of LCDs, I have reflected many times on the likelihood that no one else has tried the 8 bit interfaces for some time. Don Weiman argues that inclusion of both 4 and 8 bit interfaces makes the code harder to read. I am more motivated by the idea that the 8 bit interface takes up Arduino RAM and pins but contributes little additional function.

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.