ADDING A 24C04 (OR 24C16) SERIAL EEPROM TO YOUR BOARD

A. Read the following data sheet and application notes to gain an understanding of the EEPROM and I^2C protocol. Most are available on the course web site.

1. Device Data Sheet – use the data sheet corresponding to your device.

   Comments: Read this to learn about the IIC (I^2C) interface protocol and the serial EEPROM device that you will be adding to your board. If you feel like you would like some more background on I^2C and EEPROM basics, you may want to also read application note 536 from microchip.com. Read the introduction sections and the '2-WIRE BUS OPERATION PRIMER'. Skip the 3-wire bus operation primer.

2. AN537 from Microchip

   Comments: Read this to learn about how EEPROM endurance figures in to some industry design examples. Also, get a little intuition about how programming works at the device level.

   Time: This should be a **quick** read. It is interesting to understand how endurance must be considered in products developed by industry. The device level programming of an EEPROM memory bit is also pretty fascinating.

3. Optional Application Notes Reading
   AN-794 (originally from http://www.fairchildsemi.com)
   AN-822 (originally from http://www.fairchildsemi.com)

   Comments: If you are still craving more information, read these app notes as well. However, they do **not** provide too much new information for the purposes of this class.

B. Add the EEPROM hardware to your board.
Comments: From a hardware standpoint, adding the EEPROM to your board is relatively simple. There are very few connections to be made and the wealth of information in the application notes and data sheets is helpful.

- Remember to use pull up resistors on SCL and SDA! (suggested value is 5.1 KΩ)
- If you want to leverage the 8051 code in AN614 (URL below in section C) be sure to read the app note PDF file on how to connect your hardware to match the code. **Note: the schematic shown in AN614 is missing a pull up resistor!
- Write protect is not needed (nor supported on the 24C04).
- Be sure to connect the address pins properly. For now you will only be adding one serial EEPROM to your board.

C. Software.

You can use C or assembly language code to access the EEPROM.
If you're using MICRO-C, you may use the eeread( ) and eewrite( ) library functions. You can analyze the assembly source code for these functions for understanding.

If you're using SDCC, note that there are library functions available on the web that can be leveraged. See the Note on SDCC document on the course web site for more information.

Leveraging 8051 assembly code from the WWW.
Comments: Microchip has a great application note (AN614) on interfacing a serial EEPROM with the 8051. The code that can be downloaded with the application note has been tested with success. After some minor changes, you will be able to assemble it. First, read through the code to gain an understanding of the following functions: BYTEW, OUTS, OUT, STOP, PAGEW, BLKRD, BYTERD, IN, ACKTST, CREAD. Second, you should verify that the code creates a 100kHz clock on SCL (make sure the clock frequency does not exceed the clock specification of the EEPROM that you're using). Third, verify that the correct I^2C protocol is being used by the functions. Debugging and testing will be much easier if the code makes complete sense.

The AN614 application note source code (in a ZIP file) can also be downloaded.

Code Changes: (Line numbers are given in the application note)
Since we are using a different assembler than what was used by the engineer that wrote the AN614 code, some modifications must be made before the source code can be assembled.

1. Delete the $MOD51 on line #1.
2. Delete the DSEG on line #34
3. Delete the CSEG on line #63
4. There is an ORG on line #35 and on line #64. Change or delete these to meet your needs. If you do not delete the ORGs, you will need to insert a space or TAB before each ORG in order to assemble the code.
5. Using an editor, replace all CALL instructions with ACALL or LCALL
6. Using an editor, replace all JMP instructions with AJMP or LJMP

After making these code changes, you should be able to assemble your code.

D. Testing/Debugging Tips

✓ To start, you might want to write a simple menu that will allow you to call the AN614 functions via the PAULMON or MON51 interface. However, if you do this, you may have to modify the AN614 functions so that they get their operands out of memory instead of registers. If you just use MON51 to change register values as a means of passing arguments, the data may get corrupted before your serial EEPROM function gets called.

✓ Using state analysis on the logic analyzers in the lab can be a great way to test that the proper data is being transmitted between the 8051 and the serial EEPROM. Simply attach the analyzer CLK probe to SCL and a data probe to SDA. Set the analyzer clock to trigger on a positive SCL edge so that you latch in valid data. Note, however, that you will not be able to observe the START bit with this scheme. Since there is no rising SCL clock edge for the START bit (see the timing diagrams in the 24C04 data sheet), the analyzer will not latch the bit in. Depending on the logic analyzer you use, you may be able to use multi-level triggering to capture both the start condition and the following data stream.