

EE 371 LAB 11 - F05
A/D Conversion

Schedule: Nov 30, 1 Dec: Do the lab
 Dec 6, 7: Last time to demo

Name _____
Partner _____
MEETING DAY _____ HR _____
Demo (1) _____

Your assignment this week is to use the A/D converter to display a voltage generated from the POT on the PC display.

Background: The version of the A/D in the HCS12 is essentially the same as presented in Chapter 12 of S&H except the register addresses are changed and there are some different bits that need to be programmed. I have put a new A/D chapter on the EE371 web page (<http://www.coe.montana.edu/ee/courses/ee/ee371/pdf/atd.pdf>). That chapter has a sample A/D program that we will go over in class on Monday the 28th.

The Student Learning Kit board has a small pot in the lower left hand corner. It is connected to PAD0 and to use it you must enable it by outputting a logic high on PT-3.

1. Write a program that converts the voltage of the pot once per second and prints out its value in the range of 0.0 to 5.0 on the screen of the PC. The display resolution is to be at least 0.1 volts. Each value is to be printed on a new line.

Helping Hand 1: The coward's solution to this problem is to use a look up table that contains the string to print for each A/D value \$00 to \$FF. The idea is to use the A/D value to calculate an offset into the table.

Helping Hand 2: The real embedded system macho programmer would never resort to something as pathetic as a look-up table¹. Another way to do this is to use the multiply and divide instructions to calculate the value to display (and then convert it to ASCII, or course, for printing.) It is a straightforward job to use MUL, IDIV and FDIV to compute this value. See S&H, chapter 4 starting at page 87.

Helping Hand 3: The conversion from the A/D value to a voltage is done by multiplying the A/D reading by 5 and then dividing by 255 or by dividing by 51.

Helping Hand 4: Doing the arithmetical conversion results in a quotient and a remainder. The quotient gives you the units digit to print. The remainder must be converted to a fractional representation. Doing a FDIV with the remainder results in a binary fraction. Here is an algorithm to convert a binary fraction to a binary coded decimal number.

You will need a two-digit BCD accumulator initialized to \$00. Assume a 4-bit fraction with the binary point left of the most significant bit. If the MSB is 1, add \$50 (this is 0.50) to the accumulator; if the next bit is 1, add \$25 (0.25); the next, add \$12 (0.125), and the last (LSB), add \$06 (0.0625). At each addition you will have to adjust for decimal addition (see the DAA instruction example on page 86 in S&H.) The result is a two-digit BCD number representing the binary fraction.

1. Grading: Program demo is 30 points, but if you take the coward's way out the maximum you can score is 20 points. You get 30 points if you are a real embedded system programmer.

G:\1WPDOCS\UNIV\DEPT\COURSES\EE371\371LAB\371L11F05.DOC

1 Unless, of course, it is the best way to do the job.