Lab 4 Winter 2017 EECS 363 Digital Filtering Due February 15.

Tutorial for the DSPLIB's `fir` function

A major feature of a DSP chip is its MAC (multiplier accumulator), which in general computes a weighted sum, the fundamental computation in much of DSP, and which is exactly what is needed to compute one output y_n of a FIR filter:

$$y_n = \sum_{m=0}^{N} h_m x_{n-m}$$

The computation of a y_n can be done in C with a `for` loop but, according to the rules of C, each intermediate result is sent back to memory. The MAC, on the other hand, does not do that. Intermediate results are retained as the MAC accumulates.

Assembly language is required to access the MAC and TI provides a library of handwritten assembly language programs that can be called from a C program. It is the DSPLIB and is documented by TI's document `spru422j.pdf` . If you google that document number leave out the `j` because TI may have come out with a revision, a `spru422k` or whatever. In any case accept only what is on TI's website; you are likely to get many results linking to older versions of the document on other sites. If you find that it has been revised beyond `j` by TI, please tell me.

We will use the function `fir` in the DSPLIB.

Create your project smith_lab4 in the way Lab 2, was created, using the downloaded `lnkx.cmd` and `rts55h.lib`. However, you will not use the CSL and USBSTK5505 libraries.

Copy/paste the two source code files `FIR/fir_test_mono.c` and `FIR/fir_test_mono.h` from the folder `FIR` which was contained in my `363downloads.zip`; delete the existing `main.c` and C5535`.cmd` . This program tests the `fir.asm` program in the DSPLIB with an FIR filter with coefficients ¼,¼,¼,¼ using input ½,½,½,½,½,½,½,½,½,½,½,½,½,½,½,½,½,0,0,0,0 (length 20) . These values are specified in the header file in the Q15 format explained in class. The filter length `NH` is equivalent to the N+1 that I have been using in class. Of course the output should be 1/8,1/4,3/8,1/2,1/2,1/2,1/2, 1/2,1/2,1/2,1/2, 1/2,1/2,1/2,1/2,1/2,3/8,1/4,1/8,0 . The input signal is processed in blocks of length `NX` = 5.

The array `db` is the data buffer, which must store past values of the input and, in the computation of one output, supply the `NH` (= N+1 in the notation we have been using in class) input values required for the computation. Thus it may appear puzzling that its dimension is `NH`+2 rather than `NH`. The reason has to do with the "circular buffer" idea, which is discussed in class.

Another feature of the header is that the data buffer's memory locations are specified to be in a section called `.dbuffer` and the filter coefficients in a section called `.coeffs` .

In the Properties use the usual settings, as in Labs 2 and 3, but in the Include Options add only

C:\Code Composer v6\ccsv6\tools\compiler\c5500_4.4.1\dsplib_2.40.00\include

and add to the File Search Path only

C:\Code Composer v5\ccsv5\tools\compiler\c5500_4.4.1\dsplib_2.40.00\55xdsph.lib

You should get a successful build but with two new warnings about creating output sections `.coeffs` and `.dbuffer` without a SECTION specification. This may or may not be troublesome but fix it as follows. Open the file `lnkx.cmd` . There you will see a block of double access memory `DARAM0`, and three blocks of single access memory `SARAMn` declared. Then various sections are declared as being in one or another of these blocks: `.text` may be in any block of single access memory; `.stack` must be in double access memory, while `.data` may be in either single or double access memory.

Our data buffer `.dbuffer` must be in double access memory only but `.coeffs` may be in either. However I think it will be well to be guided by the `.cmd` file that came with the DSPLIB package, namely

C:\Code Composer v6\ccsv6\tools\compiler\c5500_4.4.1\dsplib_2.40.00\EXAMPLES\FIR\55x.cmd

A `.cmd` file can't be opened by double clicking in Windows, so open Notepad and then navigate to open `55x.cmd` as one of "All Files". This file may be looked at but don't use it otherwise. You will see that both `.coeffs` and `.dbuffer` are put in double access (DARAM) memory, so you should do the same in `lnkx.cmd`, using the syntax of `lnkx.cmd`. Within CCS, a `.cmd` file may be opened by double clicking and in any case you must make your change within CCS in order for it to be recognized. Rename the command file `lnkxA`.cmd, to avoid confusion.

Now when you build and Load you may or may not get a warning that variable `size_t`, which is in a `dsplib` header file, should be unsigned long. In either case that does not appear to be a serious problem; leave that variable declared as is. Execution should give you the above-mentioned result on the Console. In addition to giving your demo, you should hand in printouts of `lnkxA.cmd` and the output results from the Console window (Select All then copy/paste into a new Notepad document).

The `lnkxA.cmd` file you just created should be used in subsequent programs using the DSPLIB, i.e. labs 5 & 6.

2/7/2017