

**525.446**  
**DSP Hardware Laboratory**  
**Assignment #8**

**FFT with EDMA**

**Lab Description**

For this lab assignment you will modify the FFT lab (Lab 7) to work in real-time. In this lab you will automatically capture data from the serial port using EDMA (so we won't call `input_sample()` in this lab). Each 8192 point FFT will be processed while the next 8192 data points are collected. This will be accomplished by "linking" the EDMA channel to two stored sets of EDMA configurations in parameter RAM. Note that this means the FFTs in this lab are not 50% overlapped.

For this lab you can feel free to use the provided code sample that outlines an approach to configuring your software. This provided sample mainly requires you to determine the configuration for the Parameter RAM and to port your Lab7 FFT and bit-reverse into the code.

Your program should calculate 50 FFTs and store them as an STFT in external memory. Inject a simple dual-tone signal (or other interesting signal with more than a single frequency of content, such as music) into your audio input. Dump the data into MATLAB and turn in a MATLAB plot of the STFT.

**525.446**  
**DSP Hardware Laboratory**  
**Assignment #8, Grade Sheet**

**FFT, Lab2 ( \_\_\_\_/100)**

- ( \_\_\_\_/60) **EDMA PING/PONG configured correctly and functioning.**
- ( \_\_\_\_/20) **FFT from Lab 7 (including bit reversal) implemented.**
- ( \_\_\_\_/10) **MATLAB plot of STFT.**
- ( \_\_\_\_/2) **LED0 indicates if PING or PONG is being processed.**
- ( \_\_\_\_/3) **LED1 indicates how long the processing is taking.**
- ( \_\_\_\_/5) **Well Documented Code**

```

//lab8_fftEDMA.c Performs Lab7 functionality, except data is copied to external
//memory automatically using EDMA.

#include <math.h>
#include "csl_edma.h"
#include "dsk6713_aic23.h" //support file for codec,DSK
#include "DSPF_sp_cfftr2_dit.h"
#include "DSPF_sp_bitrev_cplx.h"

#define FS          48000
#define COMPLEX          2
#define FFTLEN          8192
#define FFTLEN_T_COMPLEX 2*8192
#define TCC_PING        1
#define TCC_PONG        2
#define NUM_FFT         50
#define MCBSP1_DRR      0x34000000
#define PING_PARAM      0
#define PONG_PARAM      1

Uint32 fs = DSK6713_AIC23_FREQ_48KHZ;//set sampling rate

//external memory
#pragma DATA_SECTION(stft, ".EXT_RAM")
float stft[FFTLEN*COMPLEX*NUM_FFT];

//internal memory
#pragma DATA_ALIGN (ping, 8);
#pragma DATA_ALIGN (pong, 8);
#pragma DATA_ALIGN (w, 8);
float ping[FFTLEN*COMPLEX];
float pong[FFTLEN*COMPLEX];
float w[FFTLEN/2*COMPLEX]; //twiddle factors

//EDMA handles
EDMA_Handle hEDMAstart, hEDMA_PING, hEDMA_PONG;

void
init_input_dma_channel (EDMA_Handle* hEDMA, Uint32 tcc, float* destination, Uint32 linkAddress)
{
    Uint32 regVal;

    //set OPT, SRC, CNT, DST, RLD in here

        regVal = EDMA_FMK(OPT, [field to set], [val to set]) //high
        .
        .
        .
        | EDMA_FMK(OPT, [field to set], [val to set]);
        EDMA_RSETH(*hEDMA, OPT, regVal);
        .
        .
        .
}

void
start_edma_input ()
{
    Uint32 regVal;

    //setup EDMA handles
    EDMA_clearPram(0);

    //EDMA channel and link for loading McBSP data
    hEDMAstart = EDMA_open(EDMA_CHA_REVT1, 0);
    hEDMA_PING = EDMA_allocTable(PING_PARAM);
    hEDMA_PONG = EDMA_allocTable(PONG_PARAM);
}

```

```

//ensure that the McBSP/EDMA is currently disabled
//by internal clock and frame (keep in reset)
MCBSP_FSET(SPCR1,RRST,0);

//set the main EDMA registers to turn off the event enable
EDMA_RSET(EER, 0);

//init the EDMA channel and the link sections
init_input_dma_channel(&hEDMAStart, TCC_PING, ping, EDMA_getTableAddress(hEDMA_PONG));
init_input_dma_channel(&hEDMA_PING, TCC_PING, ping, EDMA_getTableAddress(hEDMA_PONG));
init_input_dma_channel(&hEDMA_PONG, TCC_PONG, pong, EDMA_getTableAddress(hEDMA_PING));

//start the McBSP and the EDMA
//set the main EDMA registers
regVal = EDMA_RGET(EER);
regVal = regVal | (1 << EDMA_CHA_REVT1);
EDMA_RSET(EER, regVal);
MCBSP_FSET(SPCR1,RRST,1);
}

void
stop_edma_input()
{
    EDMA_RSET(EER, 0);
}

Uint32
waitfordma(Uint32 transferCompleteCode)
{
    //write a function that reads the CIPR like
    //regval = EDMA_RGET(CIPR) & mask;
    //and returns when the transferCompleteCode is set
    //
    //return a count of how many "spins" waitfordma makes
}

void
main()
{
    int i,j;
    short* x;
    float* y;
    float* stftAddr;
    Uint32 loadingPing, cnt;

    stftAddr = stft;

    comm_poll(); //init DSK,codec,McBSP
    DSK6713_LED_init(); //init LED from BSL
    DSK6713_DIP_init(); //init DIP from BSL

    loadingPing = 1;
    start_edma_input();
    cnt = waitfordma(TCC_PING);
    loadingPing = 0;
    for(i = 0; i < NUM_FFT; i++) //infinite loop executed each sample
// while(1)
    {
        if (loadingPing)
        {
            cnt = waitfordma(TCC_PING);
            loadingPing = 0;
            DSK6713_LED_on(0);
        }
        else
        {
            cnt = waitfordma(TCC_PONG);
            loadingPing = 1;
            DSK6713_LED_off(0);
        }
        if (cnt == 0) DSK6713_LED_on(3);
    }
}

```

```
DSK6713_LED_off(1);  
//unpack, fft, and QDMA data back to external memory  
DSK6713_LED_on(1);  
}  
  
stop_edma_input();  
while(1);  
}
```