**#include** "DSP28x\_Project.h" // Device Headerfile and Examples Include File

**#include** "math.h"

**#include** "float.h"

**#include** "FPU.h"

**#define** RFFT\_STAGES 9

**#define** RFFT\_SIZE (1 << RFFT\_STAGES)

**#define** ADC\_MODCLK 0x4

**#define** ADC\_BUF\_LEN RFFT\_SIZE // ADC buffer length

**#define** ADC\_SAMPLE\_PERIOD 3124 // 3124 = (3125-1) = 48 KHz sampling w/ 150 MHz SYSCLKOUT

**#define** F\_PER\_SAMPLE 48000.0L/(**float**)RFFT\_SIZE //Internal sampling rate is 48kHz

RFFT\_ADC\_F32\_STRUCT rfft\_adc;

RFFT\_F32\_STRUCT rfft;

float32 RFFToutBuff[RFFT\_SIZE]; //Calculated FFT result

float32 RFFTF32Coef[RFFT\_SIZE]; //Coefficient table buffer

float32 RFFTmagBuff[RFFT\_SIZE/2+1]; //Magnitude of frequency spectrum

//--- Global Variables

Uint16 AdcBuf[ADC\_BUF\_LEN]; // ADC buffer allocation

**volatile** Uint16 FFTStartFlag = 0; // One frame data ready flag

Uint16 DEBUG\_TOGGLE = 1; // Used in realtime mode investigation

float32 freq; // Frequency of single-frequency-component signal

// Prototype statements for functions found within this file.

**interrupt** **void** **adc\_isr**(**void**);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function: main()

\*

\* Description: Main function for C2833x Real-time RFFT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **main**(**void**)

{

 Uint16 i,j;

//--- CPU Initialization

 **InitSysCtrl**(); // Initialize the CPU (FILE: SysCtrl.c)

 **InitGpio**(); // Initialize the shared GPIO pins (FILE: Gpio.c)

 **InitPieCtrl**(); // Initialize and enable the PIE (FILE: PieCtrl.c)

//--- Peripheral Initialization

 **InitAdc**(); // Initialize the ADC (FILE: Adc.c)

 **InitPieVectTable**();

 EALLOW;

 SysCtrlRegs.HISPCP.all = ADC\_MODCLK;

 EDIS;

// Interrupts that are used in this example are re-mapped to

// ISR functions found within this file.

 EALLOW; // This is needed to write to EALLOW protected register

 PieVectTable.ADCINT = &adc\_isr;

 GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;

 GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;

 EDIS; // This is needed to disable write to EALLOW protected registers

 AdcRegs.ADCMAXCONV.all = 0x0001; // Setup 2 conv's on SEQ1

 AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0; // Setup ADCINA3 as 1st SEQ1 conv.

 AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1; // Setup ADCINA2 as 2nd SEQ1 conv.

 AdcRegs.ADCTRL2.bit.EPWM\_SOCA\_SEQ1 = 1;// Enable SOCA from ePWM to start SEQ1

 AdcRegs.ADCTRL2.bit.INT\_ENA\_SEQ1 = 1; // Enable SEQ1 interrupt (every EOS)

 // Assumes ePWM1 clock is already enabled in InitSysCtrl();

 EPwm1Regs.ETSEL.bit.SOCAEN = 1; // Enable SOC on A group

 EPwm1Regs.ETSEL.bit.SOCASEL = 4; // Select SOC from from CPMA on upcount

 EPwm1Regs.ETPS.bit.SOCAPRD = 1; // Generate pulse on 1st event

 // EPwm1Regs.CMPA.half.CMPA = 0x0080; // Set compare A value

 EPwm1Regs.TBPRD = 0xFFFF; // Set period for ePWM1

 EPwm1Regs.TBCTL.bit.CTRMODE = 0; // count up and start

 PieCtrlRegs.PIEIER1.bit.INTx6 = 1;

 IER |= M\_INT1; // Enable CPU Interrupt 1

 EINT;

 ERTM;

 rfft\_adc.Tail = &rfft.OutBuf; //Link the RFFT\_ADC\_F32\_STRUCT to

 //RFFT\_F32\_STRUCT. Tail pointer of

 //RFFT\_ADC\_F32\_STRUCT is passed to

 //the OutBuf pointer of RFFT\_F32\_STRUCT

 rfft.FFTSize = RFFT\_SIZE; //Real FFT size

 rfft.FFTStages = RFFT\_STAGES; //Real FFT stages

 rfft\_adc.InBuf = &AdcBuf[0]; //Input buffer

 rfft.OutBuf = &RFFToutBuff[0]; //Output buffer

 rfft.CosSinBuf = &RFFTF32Coef[0]; //Twiddle factor

 rfft.MagBuf = &RFFTmagBuff[0]; //Magnitude output buffer

 **RFFT\_f32\_sincostable**(&rfft); //Calculate twiddle factor

 //Clean up output buffer

 **for** (i=0; i < RFFT\_SIZE; i++)

 {

 RFFToutBuff[i] = 0;

 }

 //Clean up magnitude buffer

 **for** (i=0; i < RFFT\_SIZE/2; i++)

 {

 RFFTmagBuff[i] = 0;

 }

//--- Enable global interrupts

 **asm**(" CLRC INTM, DBGM"); // Enable global interrupts and realtime debug

//--- Main Loop

 **while**(1) // endless loop - wait for an interrupt

 {

 **if**(FFTStartFlag) // If one frame data ready, then do FFT

 {

 **RFFT\_adc\_f32u**(&rfft\_adc); // This version of FFT doesn't need buffer alignment

 **RFFT\_f32\_mag**(&rfft); // Calculate spectrum amplitude

 j = 1;

 freq = RFFTmagBuff[1];

 **for**(i=2;i<RFFT\_SIZE/2+1;i++)

 {

 //Looking for the maximum valude of spectrum magnitude

 **if**(RFFTmagBuff[i] > freq)

 {

 j = i;

 freq = RFFTmagBuff[i];

 }

 }

 freq = F\_PER\_SAMPLE \* (**float**)j; //Convert normalized digital frequency to analog frequency

 FFTStartFlag = 0; //Start collecting the next frame of data

 }

 **asm**(" NOP");

 }

} //end of main()

**interrupt** **void** **adc\_isr**(**void**)

{

 **static** Uint16 \*AdcBufPtr = AdcBuf; // Pointer to ADC data buffer

 **static** **volatile** Uint16 GPIO34\_count = 0; // Counter for pin toggle

 PieCtrlRegs.PIEACK.all = PIEACK\_GROUP1; // Must acknowledge the PIE group

 GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // 880 micro sec

//--- Manage the ADC registers

 AdcRegs.ADCTRL2.bit.RST\_SEQ1 = 1; // Reset SEQ1 to CONV00 state

 AdcRegs.ADCST.bit.INT\_SEQ1\_CLR = 1; // Clear ADC SEQ1 interrupt flag

//--- Read the ADC result

 \*AdcBufPtr++ = AdcMirror.ADCRESULT0; // Read the result

//--- Brute-force the circular buffer

 **if**( AdcBufPtr == (AdcBuf + ADC\_BUF\_LEN) )

 {

 AdcBufPtr = AdcBuf; // Rewind the pointer to the beginning

 FFTStartFlag = 1; // One frame data ready

 }

//--- Example: Toggle GPIO18 so we can read it with the ADC

 **if**(DEBUG\_TOGGLE == 1)

 {

 GpioDataRegs.GPATOGGLE.bit.GPIO18 = 1; // Toggle the pin

 }

//--- Example: Toggle GPIO34 at a 0.5 sec rate (connected to the LED on the ControlCARD).

// (1/48000 sec/sample)\*(1 samples/int)\*(x interrupts/toggle) = (0.5 sec/toggle)

// ==> x = 24000

 **if**(GPIO34\_count++ > 24000) // Toggle slowly to see the LED blink

 {

 // GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Toggle the pin

 GPIO34\_count = 0; // Reset the counter

 }

 **return**;

} //end of main()

//===========================================================================

// End of File

//===========================================================================

ADC INPUT LEVEL SHIFTED BY 1.5 V DC MAX. 3 VOLT.

If plotted AdcBuff will get same sine wave of 50 HZ as observed on O’Scope



Result After giving break point on conversion flag =1;

X[0] bin we are getting max value .But should be 0 as it is a DC bias(please correct me if I am wrong)





