Report 1 for Real-Time Signal Processing

Student: Yan Fei

Tutor: Dr. F.S. Schlindwein

# Exercise 1- familiarising with the DSK and the debugger / monitor program:

Through this exercise, I know some useful ways to test whether the DSK system works well and is connected correctly or not, when it has been initialized. There are three application programs within the TMS320C5x DSK software package for testing the DSK memory, setting up the AIC and testing out the host/target communications interface.

## SELFTEST.OUT

After connecting the DSK system, I copy SELFTEST.OUT to the working directory and test the DSK board with the following command:

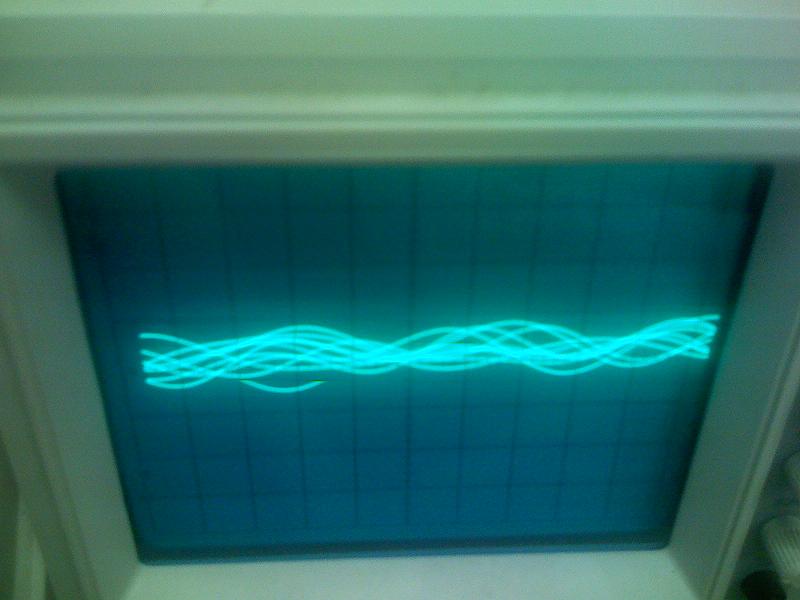
DSK5L -A –C1

The DSK responds with the messages as that showing in the handout, which tell me that the DSK has passed the test checking the DSK memory.

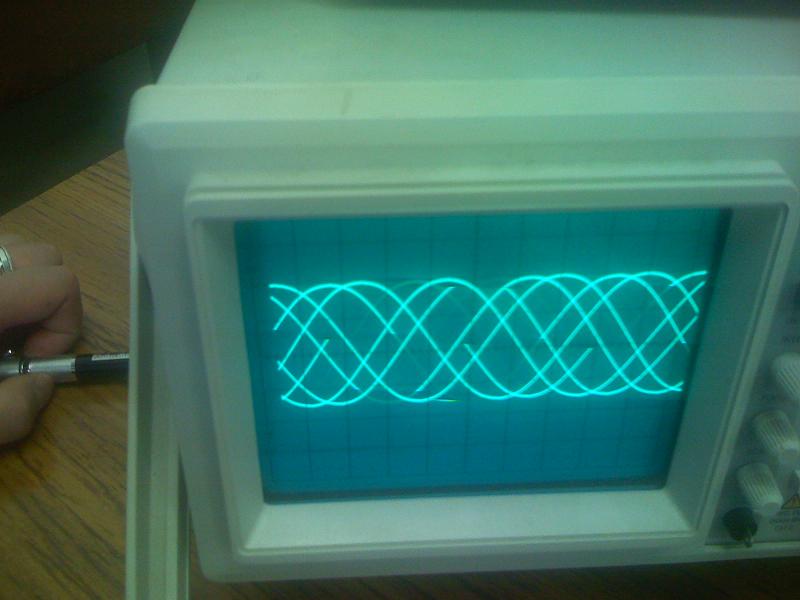
## FUNC.DSK

I connect an oscilloscope the DSK output RCA BNC and run the DSK debugger program with the following:

DSK5D C1

Then I see the debugger screen and load the FUNC.DSK file. The default FUNC.DSK is set up to generate random noise. After I execute this file, I see the random noise on the scope. 

Following the steps indicated in the handout, I change the vale at memory location 0xF0D to 1 and PC’s starting address to 0xA00. Finally, I run the program again and see the sine wave output on the scope.



Also, I can change the sampling rate to generate the different frequency of sinewave. So I change the values to TA=000Fh and RA-000Fh on data positions 0xF000 and 0xF01. I run the program from beginning and see the different frequency of sine wave.

The FUNC.DSK is used to check the functions of Analogue Interface Circuit (AIC).

## DSK\_SPEC.DSK

Finally, I connect the DSK system with an oscilloscope and a signal generator and copy the DSK\_SPEC.DSK to the working directory. Execute the file through the command:

DSK5L DSK\_SPEC.DSK –C1

By adjusting the settings I can see the spectrum of the DSK input signal.

# Exercise 2 - A/D, D/A and interrupts, ECHOINT.ASM & ECHOINT.DSK

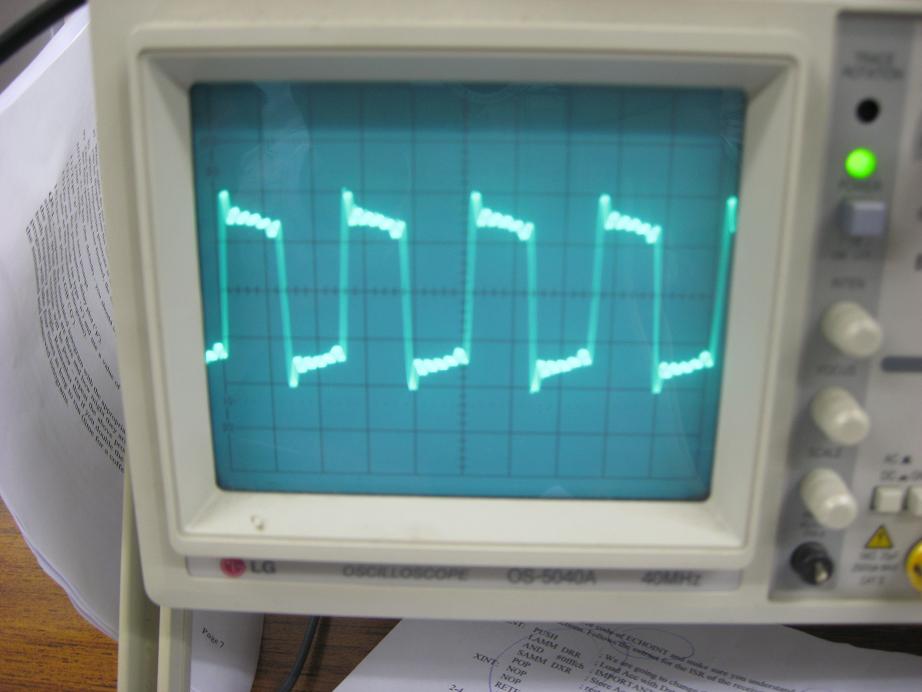
In this exercise, firstly, I assemble the program ECHOINT.ASM into ECHOINT.DSK with the following command:

DSK5A ECHOINT

Then run this program with:

DSK5L ECHOINT.DSK –C1

The input is the signal is the square wave and I get the result on the oscilloscope:



RINT: PUSH ; We are going to change the accumulator

LAMM DRR ; Load Acc with Data Rx Register (i.e. read A/D)

AND #0fffch ; IMPORTANT!! Make d00=d01=0

SAMM DXR ; Store Acc into Data Tx Register (echo to D/A)

POP ; restore Acc

XINT: NOP

NOP

RETE ; return from interrupt & re-enable interrupts

This part code is used to read a value from the A/D and echo the changed value to the D/A.

For the formula below:

∵ MCLK=10 MHz, RA=10 and RB=50

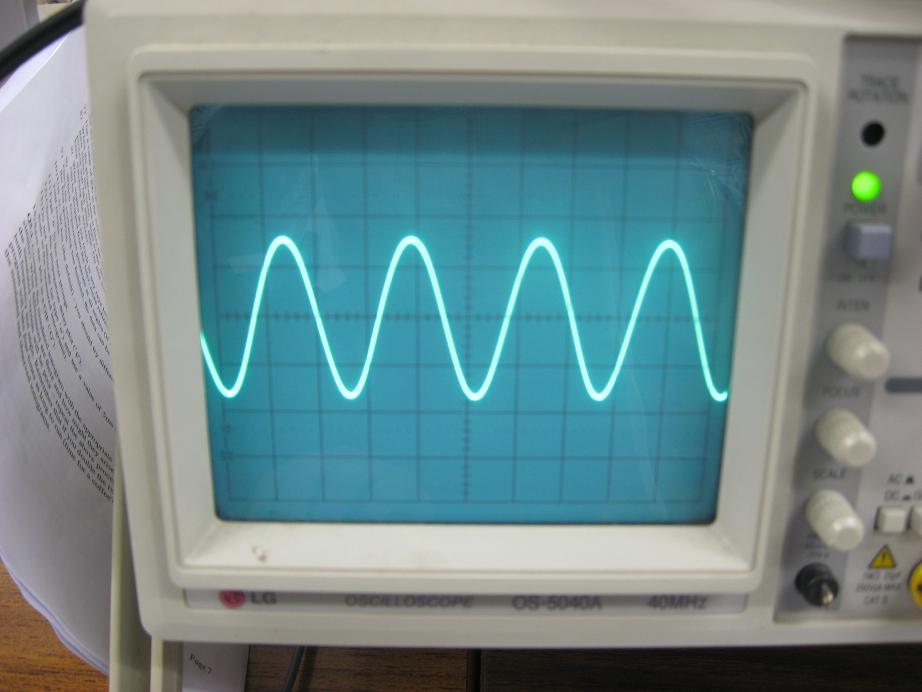
∴f­sam=10 kHz

# Exercise 3:

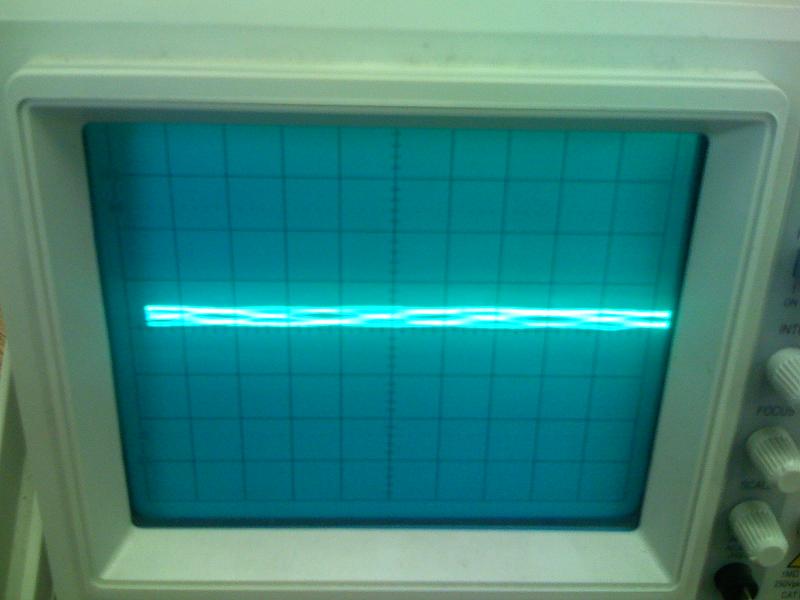
In the exercise 2, I had already modified the sampling frequency of ECHOINT to 10 kHz. Now I use this fixed sampling frequency and input signals with different frequency to investigate the effect of sampling frequency.

I use an analogue sinusoid as the input signal.

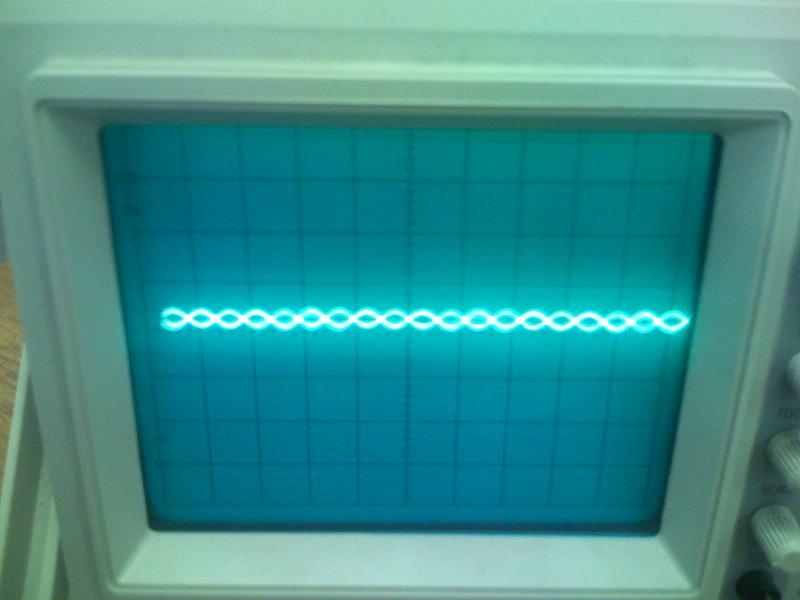
500 Hz



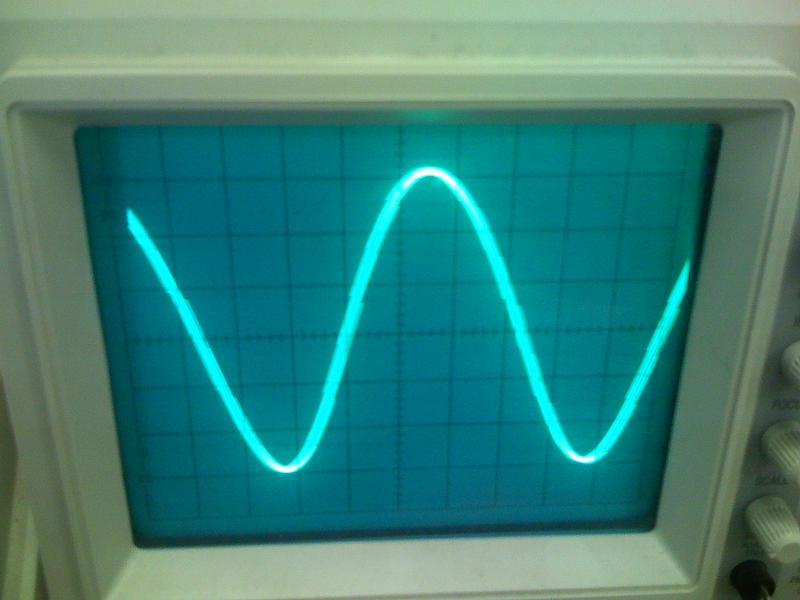
5000 Hz



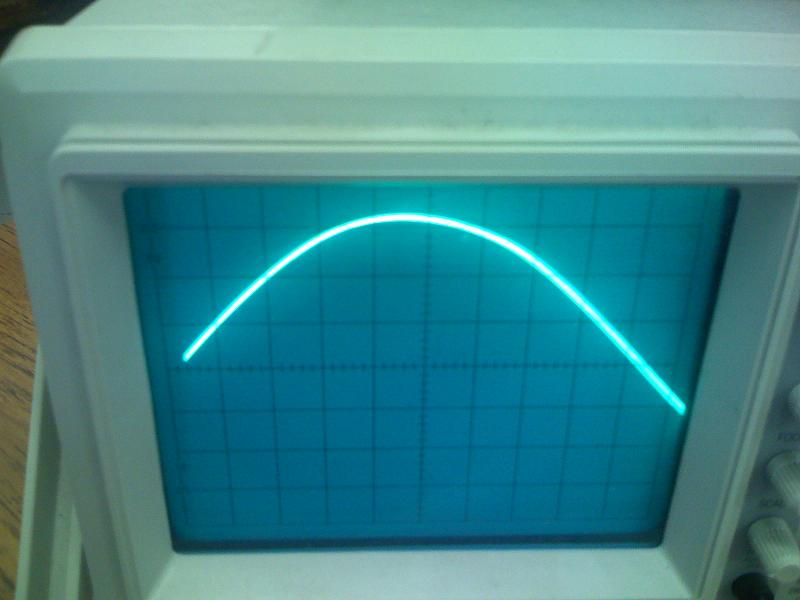
6000 Hz



9000 Hz



9500 Hz



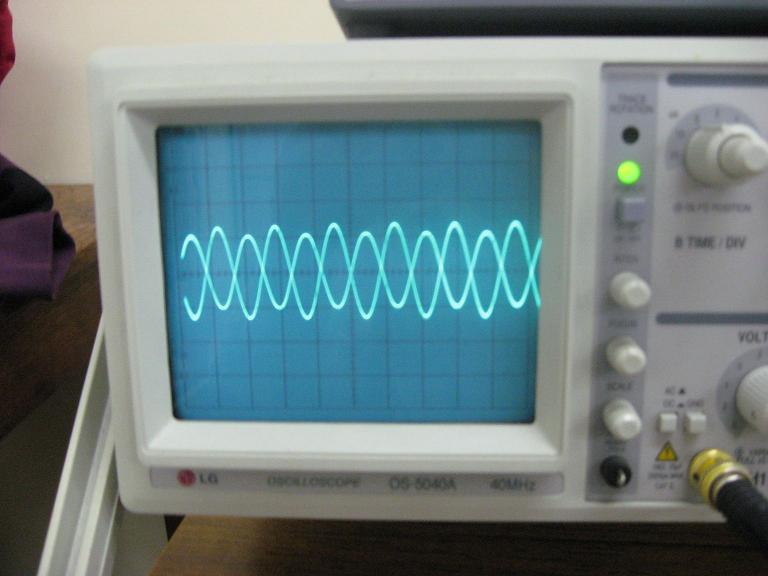
Through the graphs above, I can find that when the sampling frequency is twice bigger at least than the input frequency, the quality of the signal we caught can be accepted. When the sampling frequency is close to the frequency of original signal, it will lead to distortion. That is because this cannot guarantee that there is the action of sampling for the original signal during the unit time.

# Exercise 4:

Run the program with

DSK5L FILT1.DSK –C2

And I find the result on the oscilloscope below:



Then I modify the codes in FILT1.ASM into FILT2.ASM with following changes:

.ds 0F00h

A0 .word 8192

A1 .word 8192

A2 .word 8192

A3 .word 8192

X0 .word 0

X1 .word 0

X2 .word 0

X3 .word 0

Y0 .word 0

PUSH

SPM 1 ; to correct for the multiplications (Q15\*Q15 = Q30)

LDP #X0 ; data memory base-address = F00h

FIR ZAC ; initialise Accumulator to zero (ZERO ACC)

LT X3 ; load T with X3

MPY A3 ; P = a3 x(n-3)

LTD X2 ; LTD = accumulate previous P into Acc so that Acc = a3 \* x(n-3),

; load T with X2, and DMOV X2 to X3, i.e., ‘age’ it (\* 1/z)

MPY A2 ; Acc = a3 x(n-3), P =a2 x(n-2)

LTD X1 ; Acc = a2 x(n-2) + a3 x(n-3), T = X1, X1 moved to X2

MPY A1 ; P = a1 x(n-1)

LTD X0 ; Acc = a1 x(n-1) +a2 x(n-2) + a3 x(n-3), T = X0, X0 'aged'

MPY A0 ; P = a0 x(n)

APAC ; Acc = a0 x(n) + a1 x(n-1) +a2 x(n-2) + a3 x(n-3)

SACH Y0 ; Save result onto y(n)

LAC Y0 ; bring it into low ACCU

AND #0fffch ;IMPORTANT!!. Make d00=d01=0

SAMM DXR ; Store Acc into Data Tx Register (echo to D/A)

LAMM DRR ; Load Acc with Data Rx Register (i.e. read A/D). Sample x(n)

SACL X0 ; Read new input value. Notice how all others 'aged' by falling 1 place

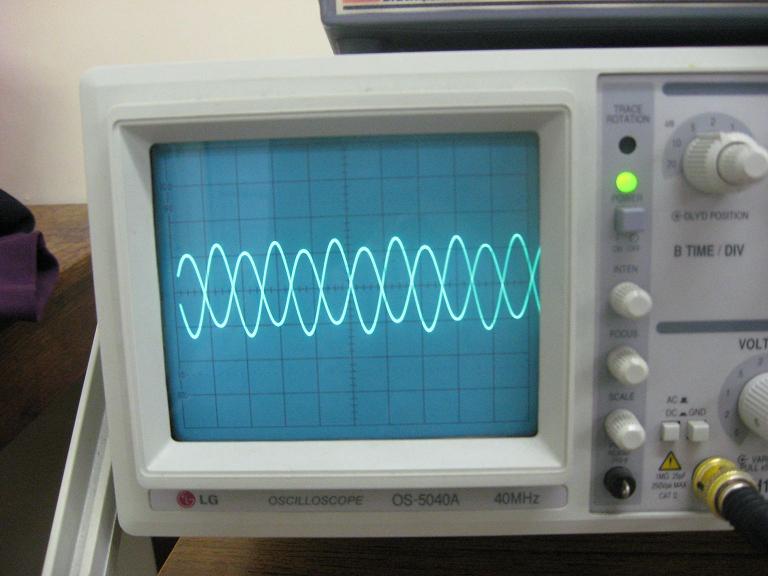
B FIR ; in memory

Assemble FILT2.ASM and RUN it with following commands:

DSK5A FILT2

DSK5L FILT2.DSK –C2

I see the result on the oscilloscope below:



Because we just change the sequence of calculation, the formula of calculation is not changed; the FILT1 and FILT2 show the same graph on the scope.

Finally, I make the sampling frequency equal to 10 kHz and check the zeroes of the filter by sweeping the frequency of the sinusoid

|  |  |
| --- | --- |
| Frequency(Hz) | Zero |
| 0-2500 | Pass |
| 2500-4000 | No Pass |
| 4000-6500 | Around 0 and No Pass |
| 6500 | Pass |
| 7300 | 0 |

# Exercise 5: