Application Report
SPRA708 - November 2000

TMS320C62x Algorithm: Sine Wave Generation

Yao-Ting Cheng *C6000 Applications*

ABSTRACT

This application report shows how to implement the 2nd-order IIR filter that generates a sinusoid signal on the TMS320C62x™ DSP.

Contents

List of Figures

List of Examples

1 Design Problem

This application report shows how to implement the 2nd-order IIR filter that generates a sinusoid signal on TMS320C62x DSP.

2 Solution

There are several ways to implement the sine wave generator on DSP processor such as a lookup table, interpolation, polynomials, etc. One efficient technique is using an IIR filter, making it oscillating by locating its poles in the unit circle of the Argand diagram. A typical 2nd order IIR filter can be established as illustrated in Figure 1.

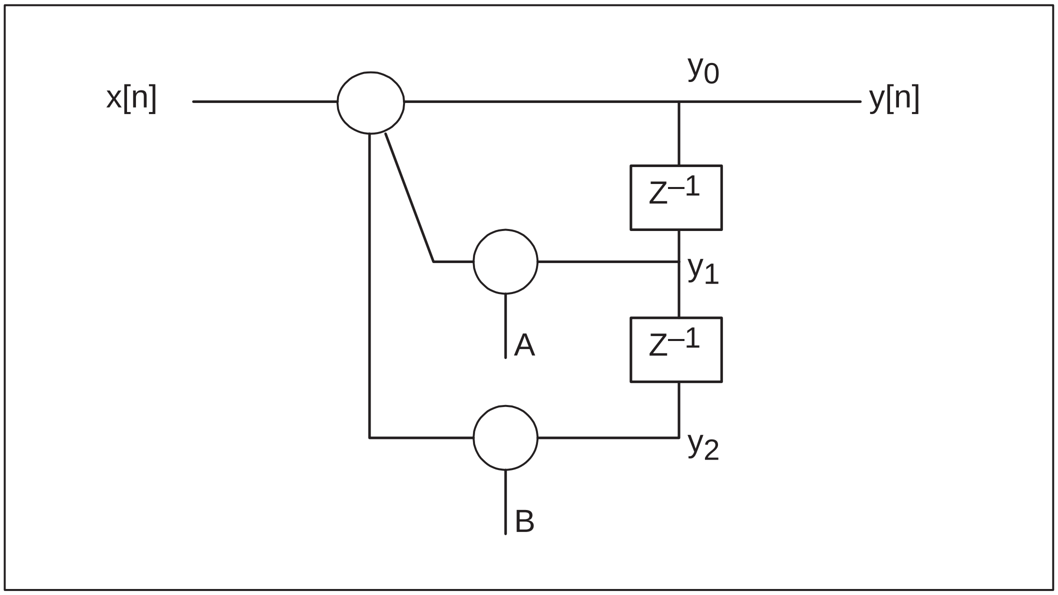

Figure 1. 2nd Order IIR Filter for Generating Sine Wave

Give this IIR two initial values as below based on the assumption of 40 samples to make up a complete sine wave, then disconnect the x[n] from the input. At time interval n=2,

$$y_2 = x[0] = \sin\left(\frac{2\pi}{40} \cdot 0\right) = 0$$

$$y_1 = x[1] = \sin\left(\frac{2\pi}{40} \cdot 1\right) \cong 0.1564.$$

Properly choose filter coefficients A and B, so that this IIR will oscillate by itself. The formal proof can be found in the DSP related text book. You can take a short cut to find the value of A and B simply by solving the difference equations:

y[n]=A•y[n–1]+B•y[n–2]

For example:

y[2] = A•y[1] + B•y[0],

y[3] = A•y[2] + B•y[1].

Substitute y[n] with known values as below:

$$y[0] = \sin\left(\frac{2\pi}{40} \cdot 0\right) = 0$$

$$y[1] = \sin\left(\frac{2\pi}{40} \cdot 1\right) \cong 0.1564$$

$$y[2] = \sin\left(\frac{2\pi}{40} \cdot 2\right) \cong 0.3090$$

$$y[3] = \sin\left(\frac{2\pi}{40} \cdot 3\right) \cong 0.4540.$$

That is

$0.3090 = A \times 0.1564 + B \times 0,$

$0.4540 = A \times 0.3090 + B \times 0.1564,$

therefore A=1.9754 and B=–1. Examining the behavior of this IIR filter by its transfer function as below:

y[n] = 1.9754•y[n–1] – y[n–2] + x[n]

Take a Z-transform:

Y[Z](1 – 1.9754Z–1 + Z–2) = X[Z].

The transfer function is

$$H(Z) = \frac{Y[Z]}{X[Z]} = \frac{1}{1 - 1.9754Z^{-1} + Z^{-2}}$$

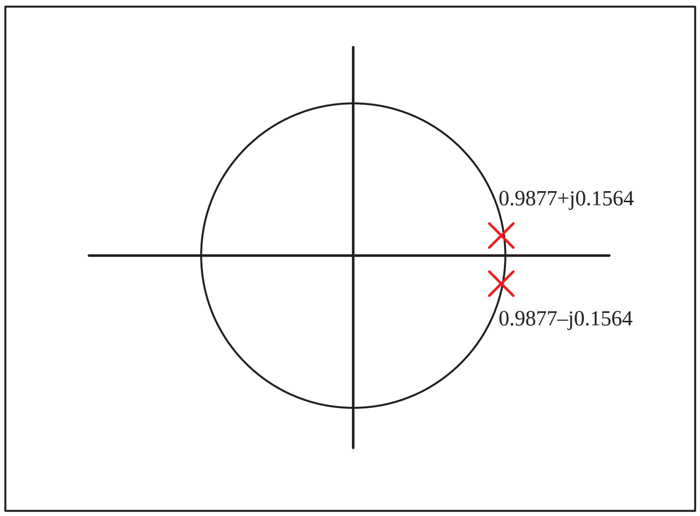

Figure 2. Location of the IIR Poles

Its has two poles Z=0.9877+j0.1564 and Z=0.9877–j0.1564 and are located in the unit circle as shown above. The following program codes show how to program the TMS320C6x using Assembly landuage and C language to implement the IIR Sine wave generator. You can utilize the *Probe point* feature available in the Code Composer Studio by connecting the varying “output” of the sine wave to a graphical display.

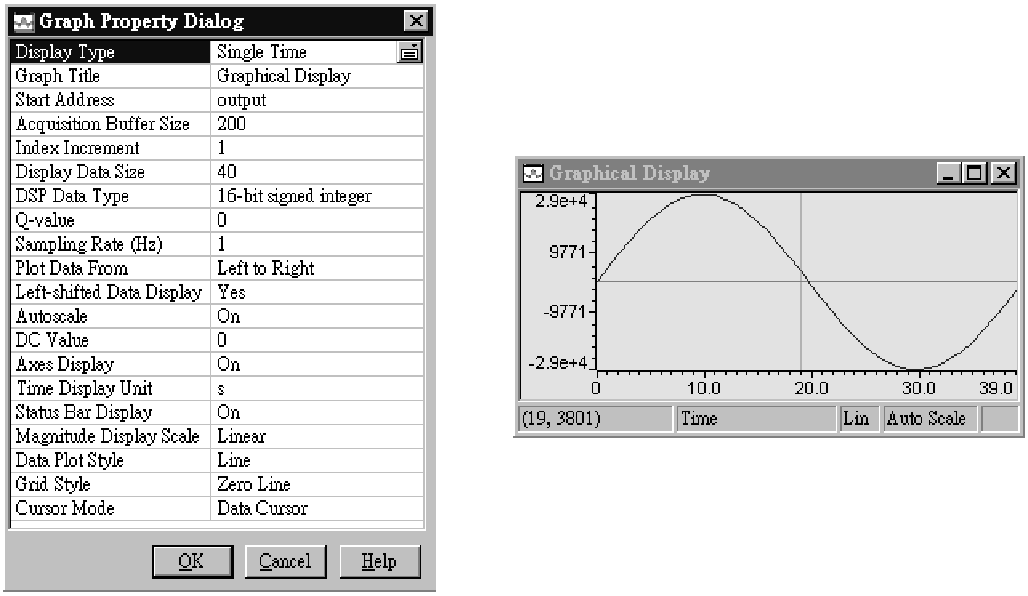

Figure 3. Graphical Display of the Sine Wave

Example 1. Code Listing in Assembly

```
        .title  ”fir.asm”       ;

        .def    init

int     .equ    4
half    .equ    2

        .data
a_half  .short  32768*1975/2000
y1      .short  32768*1409/10000
y2      .short  0

        .bss    output,40*half,half
        .bss    buffer,2*half,2*half

        .text

init:   MVK     .S1     a_half,A0
        MVKH    .S1     a_half,A0
        LDH     .D1     *A0,A2          ; load coeff A_HALF

        MVK     .S1     0x0001,A0       ; setup circular buf AMR=0001_0001h
        MVKLH   .S1     0x0001,A0       ; setup circular buf AMR=0001_0001h
        MVC     .S2X    A0,AMR         ; blk size is 4 bytes, pointer is A4

        MVK     .S1     output,A3       ; memory for store sine wave
        MVKH    .S1     output,A3
        MVK     .S1     0,A0
        STH     .D1     A0,*A3++        ; output y(0)=0 to IOPORT

        MVK     .S1     buffer,A4
        MVKH    .S1     buffer,A4
        MVK     .S1     y1,A0
        MVKH    .S1     y1,A0

        LDH     .D1     *A0++,A1        ; load y1
        NOP             4
        STH     .D1     A1,*A4++        ; y(n–1)=y1, point to y(n–2)
        STH     .D1     A1,*A3++        ; output y(1)=y1 to IOPORT

        LDH     .D1     *A0,A1          ; load y2
        NOP             4
        STH     .D1     A1,*A4++        ; y(n–2)=y2, point to y(n–1)

main:   MVK     .S1     40,A1           ; calculate 40 samples

loop:   LDH     .D1     *A4++,B1        ; ld y(n–1) to B1, point to y(n–2)
        LDH     .D1     *A4,B2          ; ld y(n–2) to B2, point to y(n–2)
        NOP             3               ; <–– try optimizing here
        SMPY    .M1X    A2,B1,A0        ; A_HALF*y(n–1)
        SUB     .L1     A1,1,A1
   [A1] B       .S1     loop
        SHR     .S1     A0,16,A0        ; <–– try optimizing here
        SADD    .L1     A0,A0,A0        ; A_HALF*y(n–1)*2
        SSUB    .L2X    A0,B2,B0        ; A_HALF*y(n–1)*2–y(n–2)
        STH     .D1     B0,*A4          ; st y(n) to y(n–2) as y(n–1)
        STH     .D1     B0,*A3++        ; st y(n) to Sine wave IOPORT

end:    B       .S1     $

        NOP             5
```

Example 2. Code Listing in C

```
short output;

main()
{

    int i;
    const short A=0x7e66;       /* A=(1.975/2 * 32768) */
    short y[3]={0,0x1209,0};    /* (y0,y1,y2), y1=(0.1409*32768) */

    for (i=0; i<40; i++) {
        y[0] = (((A*y[1])>>15) + ((A*y[1])>>15)) – y[2];
        y[2] = y[1];        /* y2 <–– y1 */
        y[1] = y[0];        /* y1 <–– y0 */
        output = y[0];

    }
}
```

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.