Implementation of FIR/IIR Filters with the TMS32010/TMS32020

APPLICATION REPORT: SPRA003A

Authors: Al Lovrich and Ray Simar, Jr.
Digital Signal Processing – Semiconductor Group

Digital Signal Processing Solutions
1989

TEXAS INSTRUMENTS

INTRODUCTION

In many signal processing applications, it is advantageous to use digital filters in place of analog filters. Digital filters can meet tight specifications on magnitude and phase characteristics and eliminate voltage drift, temperature drift, and noise problems associated with analog filter components.

This application report describes a variety of methods for implementing Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters with the TMS320 family of digital signal processors. Emphasis is on minimizing both the execution time and the number of data memory locations required. Tradeoffs between several different structures of the two classes of digital filters are also discussed.

In this report, TMS320 source code examples are included for the implementation of two FIR filters and three IIR filters based on the techniques presented. Plots of magnitude response, log-magnitude response, unit-sample response, and other pertinent data accompany each of the filter implementations. Important performance considerations in digital filter design are also included. The methods presented for implementing the different types of filters can be readily extended to any desired order of filters.

Readers are assumed to have some familiarity with the basic concepts of digital signal processing theory.[1] The notation used in this report is consistent with that used in reference [1].

FILTERING WITH THE TMS320 FAMILY

Almost every field of science and engineering, such as acoustics, physics, telecommunications, data communications, control systems, and radar, deal with signals. In many applications, it is desirable that the frequency spectrum of a signal be modified, reshaped, or manipulated according to a desired specification. The process may include attenuating a range of frequency components and rejecting or isolating one specific frequency component.

Any system or network that exhibits such frequency-selective characteristics is called a filter. Several types of filters can be identified: lowpass filter (LPF) that passes only ''low'' frequencies, highpass filter (HPF) that passes ''high'' frequencies, bandpass filter (BPF) that passes a ''band'' of frequencies, and band-reject filter that rejects certain frequencies. Filters are used in a variety of applications, such as removing noise from a signal, removing signal distortion due to the transmission channel, separating two or more distinct signals that were mixed in order to maximize communication channel utilization, demodulating signals, and converting discrete-time signals into continuous-time signals.

Advantages of Digital Filtering

The term ''digital filter'' refers to the computational process or algorithm by which a digital signal or sequence of numbers (acting as input) is transformed into a second sequence of numbers termed the output digital signal. Digital filters involve signals in the digital domain (discrete-time signals), whereas analog filters relate signals in the analog domain (continuous-time signals). Digital filters are used extensively in applications, such as digital image processing, pattern recognition, and spectrum analysis. A band-limited continuous-time signal can be converted to a discrete-time signal by means of sampling. After processing, the discrete-time signal can be converted back to a continuous-time signal.Some of the advantages of using digital filters over their analog counterparts are:

1. High reliability
2. High accuracy
3. No effect of component drift on system performance
4. Component tolerances not critical.

Another important advantage of digital filters when implemented with a programmable processor such as the TMS320 is the ease of changing filter parameters to modify the filter characteristics. This feature allows the design engineer to effectively and easily upgrade or update the characteristics of the designed filter due to changes in the application environment.

Design of Digital Filters

The design of digital filters involves execution of the following steps:

1. Approximation
2. Realization
3. Study of arithmetic errors
4. Implementation.

Approximation is the process of generating a transfer function that satisfies a set of desired specifications, which may involve the time-domain response, frequency-domain response, or some combination of both responses of the filter.

Realization consists of the conversion of the desired transfer function into filter networks. Realization can be accomplished by using several network structures,[2,3] as listed below. Some of these structures are covered in detail in this report.

1. Direct
2. Direct canonic (direct-form II)
3. Cascade
4. Parallel
5. Wave[4]
6. Ladder.

Approximation and realization assume an infinite-precision device for implementation. However, implementation is concerned with the actual hardware circuit or software coding of the filter using a programmable processor. Since practical devices are of finite precision, it is necessary to study the effects of arithmetic errors on the filter response.

TMS320 Digital Signal Processors

Digital Signal Processing (DSP) is concerned with the representation of signals (and the information they contain) by sequences of numbers and with the transformation or processing of such signal representations by numeric-computational procedures. In the past, digital filters were implemented in software using mini- or main-frame computers for non-realtime operation or on specialized dedicated digital hardware for realtime processing of signals.

The recent advances in VLSI technology have resulted in the integration of these digital signal processing systems into small integrated circuits (ICs), such as the TMS320 family of digital signal processors from Texas Instruments. The TMS320 implementation of digital filters allows the filter to operate on realtime signals. This method combines the ease and flexibility of the software implemention of filters with reliable digital hardware. To further ease the design task, it is now possible for engineers to design and test filters using any one of the commercially available filter design packages, some of which create TMS320 code and decrease the design time.

The Texas Instruments TMS320 digital signal processing family contains two generations of digital signal processors. The TMS32010, the first-generation digital signal processor,[5] implements in hardware many functions that other processors typically perform in software. Some of the key features of the TMS32010 are:

- 200-ns instruction cycle
- 1.5K words (3K bytes) program ROM
- 144 words (288 bytes) data RAM
- External memory expansion to 4K words (8K bytes) at full speed
- 16 x 16-bit parallel multiplier
- Interrupt with context save
- Two parallel shifters
- On-chip clock
- Single 5-volt supply, NMOS technology, 40-pin DIP.

The TMS32020 is the second-generation processor[6] in the TMS320 DSP family. To maintain device compatibility, the TMS32020 architecture is based upon that of the TMS32010, the first member of the family, with emphasis on overall speed, communication, and flexibility in processor configuration. Some of the key features of the TMS32020 are:

- 544 words of on-chip data RAM, 256 words of which may be programmed as either data or program memory
- 128K words of data/program space
- Single-cycle multiply/accumulate instructions
- TMS32010 software upward compatibility
- 200-ns instruction cycle
- Sixteen input and sixteen output channels
- 16-bit parallel interface
- Directly accessible external data memory space
- Global data memory interface for multiprocessing
- Instruction set support for floating-point operations
- Block moves for data/program memory
- Serial port for multiprocessing or codec interface
- On-chip clock
- Single 5-volt supply, NMOS technology, 68-pin grid array package.

Because of their computational power, high I/O throughput, and realtime programming, the TMS320 processors have been widely adapted in telecommunication, data communication, and computer applications. In addition to the above features, the TMS320 has efficient DSP-oriented instructions and complete hardware/software development tools, thus making the TMS320 highly suitable for DSP applications.

DIGITAL FILTER IMPLEMENTATION ON THE TMS320

For a large variety of applications, digital filters are usually based on the following relationship between the filter input sequence x(n) and the filter output sequence y(n):

$$y(n) = \sum_{k=0}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k\, x(n-k) \qquad (1)$$

Equation (1) is referred to as a linear constant-coefficient difference equation. Two classes of filters can be represented by linear constant-coefficient difference equations:

1. Finite Impulse Response (FIR) filters, and
2. Infinite Impulse Response (IIR) filters.

The following sections describe the implementation of these classes of filters on the TMS32010 and TMS32020.

FIR Filters

For FIR filters, all of the a_k in (1) are zero. Therefore, (1) reduces to

$$y(n) = \sum_{k=0}^{M} b_k\, x(n-k) \qquad (2)$$

where (M + 1) is the length of the filter.

As a result, the output of the FIR filter is simply a finite-length weighted sum of the present and previous inputs to the filter. If the unit-sample response of the filter is denoted as h(n), then from (2), it is seen that h(n) = b(n). Therefore, (2) is sometimes written as

$$y(n) = \sum_{k=0}^{M} h(k)x(n-k) \qquad (3)$$

From (3), it can be seen that an FIR filter has, as the name implies, a finite-length response to a unit sample. Denoting the z transforms of x(n), y(n), and h(n) as X(z), Y(z), and H(z), respectively, then

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-1} = \sum_{k=0}^{M} h(k)z^{-k} \qquad (4)$$

Equations (3) and (4) may also be represented by the network structure shown in Figure 1. This structure is referred to as a direct-form realization of an FIR filter, because the filter coefficients can be identified directly from the difference equation (3). The branches labeled with z^{-1} in Figure 1 correspond to the delays in (3) and the multiplications by z^{-1} in (4). Equation (3) may be implemented in a straightforward and efficient manner on a TMS320 processor.

TMS32010 Implementation of FIR Filters

Figure 2 gives an example of a length-5 direct-form FIR filter, and Figure 3 shows a portion of the TMS32010 code for implementing this filter.

The notation developed in this section will be used throughout this application report. XN corresponds to x(n), XNM1 corresponds to x(n − 1), etc.

In the above implementation, the following three basic and important concepts for the implementation of FIR filters on the TMS320 should be understood:

1. The relationship between the unit-sample response of an FIR filter and the filter structure,
2. The power of the LTD and MPY instruction pair for this implementation, and
3. The ordering of the input samples in the data memory of the TMS320, which is critical for realtime signal processing.

The input sequence x(n) is stored as shown in Figure 4. In general, each of the multiplies and shifts of x(n) in (3) is implemented with an instruction pair of the form

```
LTD   XNM1
MPY   H1
```

The instruction LTD XNM1 loads the T register with the contents of address XNM1, adds the result of the previous multiply to the accumulator, and shifts the data at address XNM1 to the next higher address in data memory. Using the storage scheme in Figure 4, this corresponds to shifting the data at address XNM1 to address XNM2. The instruction MPY H1 multiplies the contents of the T register with the contents of address H1. The shifting is the reason for the storage scheme used in Figure 4. This scheme, critical for realtime digital signal processing, makes certain that the input sequence x(n) is in the correct location for the next pass through the filter.

By comparing (3) with the code in Figure 3, the reason for the ordering of the data and the importance of the shift implemented by the LTD instruction can be seen. To better

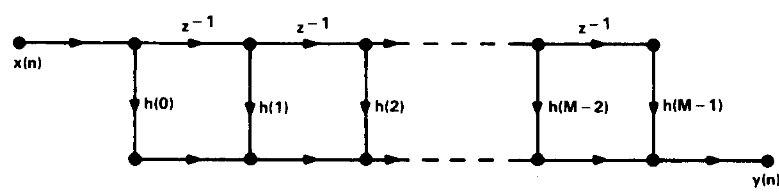


Figure 1. Direct-Form FIR Filter

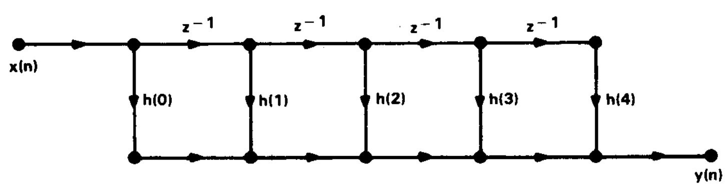


Figure 2. Length-5 Direct-Form FIR Filter

```
* THIS SECTION OF CODE IMPLEMENTS THE FOLLOWING EQUATION:                   *
* x(n-4)h(4) + x(n-3)h(3) + x(n-2)h(2) + x(n-1)h(1) + x(n)h(0) = y(n)      *
*
NXTPT   IN XN,PA2         * GET THE NEW INPUT VALUE XN FROM PORT PA0 *
*
        ZAC               * ZERO THE ACCUMULATOR *
*
        LT XNM4           * x(n-4)h(4) *
        MPY H4
*
        LTD XNM3          * x(n-4)h(4) + x(n-3)h(3) *
        MPY H3
*
        LTD XNM2          * SIMILAR TO THE PREVIOUS STEPS *
        MPY H2
*
        LTD XNM1
        MPY H1
*
        LTD XN
        MPY H0
*
        APAC              * ADD THE RESULT OF THE LAST MULTIPLY TO *
*                         * THE ACCUMULATOR                        *
*
        SACH YN,1         * STORE THE RESULT IN YN *
*
        OUT YN,PA2        * OUTPUT THE RESPONSE TO PORT PA1 *
*
        B NXTPT           * GO GET THE NEXT POINT *
```

Figure 3. TMS32010 Code for Implementing a Length-5 FIR Filter

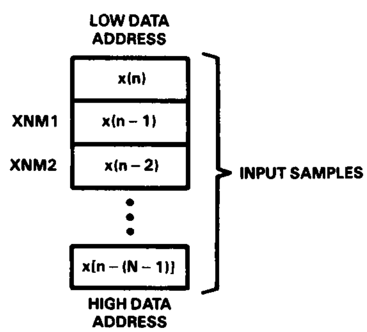


Figure 4. TMS32010 Input Sample Storage for a Length-N FIR Filter

understand the algorithm, the relationship between the input and output of the filter must be considered. Evaluating (3) for a particular value of n, for example, n_0, yields

$$y(n_0) = \sum_{k=0}^{N-1} h(k)\, x(n_0 - k) \tag{5}$$

If the next sample of the filter response $y(n_0 + 1)$ is needed, it is seen from (3) that

$$y(n_0 + 1) = \sum_{k=0}^{N-1} h(k)\, x(n_0 + 1 - k) \tag{6}$$

Equations (5) and (6) show that the samples of x(n) associated with particular values of h(k) in (5) have been shifted to the left (i.e., to a higher data address) by one in (6). This shifting of the input data, illustrated in Figure 5, corresponds to the shifting of the flipped input sequence in relation to the unit-sample response.

Depending on the system constraints, the designer may choose to reduce program memory size by taking advantage of indirect addressing capability provided by the TMS32010. Using either of the auxiliary registers along with the autoincrement or autodecrement feature, the FIR filter program can be rewritten in looped form as shown in Figure 6.

The input sequence x(n) is stored as shown in Figure 4, and the impulse response h(n) is stored as shown in Figure 7. In the looped version, the indirect addressing mode is used with the autodecrement feature and BANZ instruction to control the looping and address generation for data access. While the looped code requires less program memory than the straightline version, the straightline version runs more quickly than the looped code because of the overhead associated with loop control. This design tradeoff should be carefully considered by the design engineer.

It is also possible to use the LTD/MPYK instruction pair to implement each filter tap in straightline code. The MPYK instruction is used to multiply the contents of the T register by a signed 13-bit constant stored in the MPYK instruction word. For many applications, a 13-bit coefficient can adequately implement the filter without significant changes to the filter response. An advantage of using this approach is that the coefficients are stored in program memory and there is no need to transfer them to data memory. This reduces the amount of data memory locations required per filter tap from two to one.

The length-80 FIR filter program in Appendix A implements a linear-phase FIR filter in straightline code. The unit-sample response of the filter is symmetric in order to achieve linear phase. Because of the symmetry, it is necessary to store only 40 (rather than 80) of the samples of the impulse response. This symmetry can often be used to a designer's advantage since it significantly reduces the amount of storage space required to implement the filter.

In summary, by taking advantage of the TMS32010 features, a designer can implement a direct-form FIR filter, optimized for execution time, data memory, or program memory.

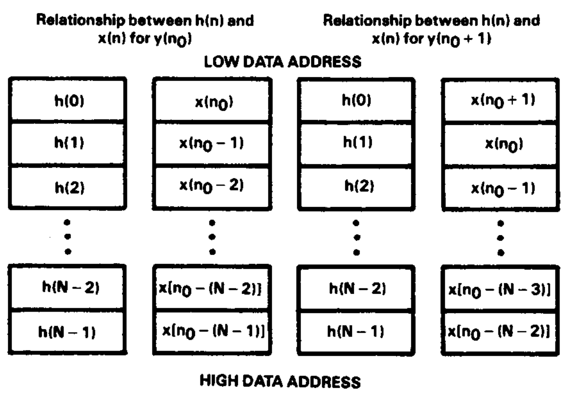


Figure 5. Relationship Between the Contents of Data Registers

```
* THIS SECTION OF CODE IMPLEMENTS THE EQUTION:                   *
* x(n-(N-1))h(N-1) + x(n-(N-2))h(N-2) + ... + x(n)h(0) = y(n) *
*
        LARP AR0        * AUXILIARY REGISTER POINTER SET TO AR0 *
*
NXTPT   IN XN,PA2       * PULL IN NEW INPUT FROM PORT PA0 *
*
        LARK AR0,XNMNM1 * AR0 POINTS TO X(n-(N-1)) *
        LARK AR1,HNM1   * AR1 POINTS TO H(N-1) *
*
        ZAC             * ZERO THE ACCUMULATOR *
*
        LT *-,AR1       * x(n-(N-1))h(N-1) *
        MPY *-,AR0
*
LOOP    LTD *,AR1       * x(n-(N-1))h(N-1)+x(n-(N-2))h(N-2)+...+x(n)h(0)=y(n)*
        MPY *-,AR0
*
        BANZ LOOP       * IF AR0 DOES NOT EQUAL ZERO,            *
*                       * THEN DECREMENT AR0 AND BRANCH TO LOOP *
*
        APAC            * ADD THE P REGISTER TO THE ACCUMULATOR *
*
        SACH YN,1       * STORE THE RESULT IN YN *
*
        OUT YN,PA2      * OUTPUT THE RESPONSE TO PORT PA1 *
*
        B NXTPT         * GO GET THE NEXT INPUT POINT *
```

Figure 6. TMS32010 Code for Implementing a Looped FIR Filter

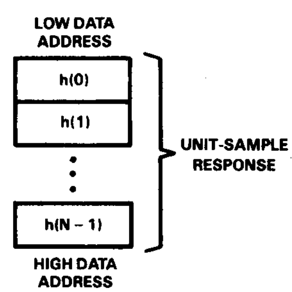


Figure 7. TMS32010 Unit-Sample Response Storage for a Looped FIR Filter

TMS32020 Implementation of FIR Filters

In many DSP applications, realtime processing of signals is very critical. Important choices must be made in selecting a DSP device capable of realtime filtering, For example, in a speech application, a sampling rate of 8 kHz is common, which corresponds to an interval of 125 μs between consecutive samples. This interval is the maximum allowable time for realtime operation, corresponding to 625 cycles on the TMS32010. In order to perform the required signal processing tasks in that interval, it is essential to reduce filter execution time. This can be accomplished by a single-cycle multiply/accumulate instruction. The TMS32020, the second-generation DSP device, is a processor with such a capability. A single-cycle multiply/accumulate with data-move instruction and larger on-chip RAM make it possible to implement each filter tap in approximately 200 ns.

The TMS32020 provides a total of 544 16-bit words of on-chip RAM, divided into three separate blocks of B0, B1, and B2. Of the 544 words, 288 words (blocks B1 and B2) are always data memory, and 256 words (block B0) are programmable as either data or program memory. The CNFD (configure block B0 as data memory) and CNFP (configure block B0 as program memory) instructions allow dynamic configuration of the memory maps through software, as illustrated in Figure 8. After execution of the CNFP instruction, block B0 is mapped into program memory, beginning with address 65280. To take advantage of the MACD (multiply and accumulate with data move) instruction, block B0 must be configured as program memory using the CNFP instruction. MACD only works with on-chip RAM. The use of the MACD instruction helps to speed

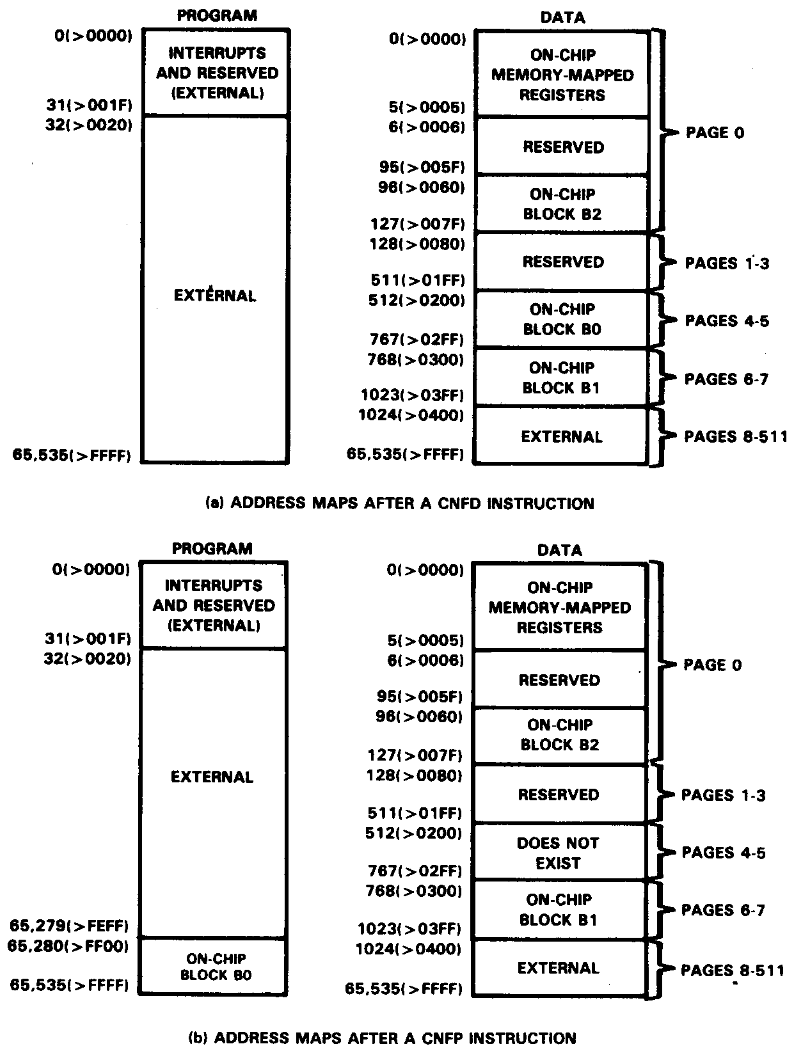


Figure 8. TMS32020 Memory Maps

the filter execution and allows the size of the FIR filter to expand to 256 taps.[6]

The TMS32020 implementation of (3) is made even more efficient with a repeat instruction, RPTK. It forms a useful instruction pair with MACD, such as

```
RPTK    NM1
MACD    (PMA),(DMA)
```

The RPTK NM1 instruction loads an immediate 8-bit value N-1 into the repeat counter. This causes the next instruction to be executed N times (N = the length of the filter). The instruction MACD (PMA),(DMA) performs the following functions:

1. Loads the program counter with PMA,
2. Multiplies the value in data memory location DMA (on-chip, block B1) by the

value in program memory location PMA (on-chip, block B0),
3. Adds the previous product to the accumulator,
4. Copies the data memory value (block B0) to the next higher on-chip RAM location. The data move is the mechanism by which the z^{-1} delay can be implemented, and
5. Increments the program counter with each multiply/accumulate to point to the next sample of the unit-sample response.

In other words, the MACD instruction combines the LTD/MPY instruction pair into one. With the proper storage of the input samples and the filter unit-sample response, one can take advantage of the power of the MACD instruction. Figure 9 is a data storage scheme that provides the correct sequence of inputs for the next pass through the filter.

In the TMS32020 code example of Figure 10, data memory values are accessed indirectly through auxiliary register 1 (AR1) when the MACD instruction is implemented. For low-order filters (second-order), using the MACD instruction in conjunction with the RPTK instruction is less effective due to the overhead associated with the MACD instruction in setting up the repeat construct. To take advantage of the MACD instruction, the filter order must be greater than three. For lower-order filters, it is recommended to use the LTD/MPY instruction pair in place of RPT/MACD.

Writing looped code for the TMS32020 implementation of an FIR filter gives no further advantage. Since the MACD instruction already uses less program memory, looped code in this case does not reduce program memory size. Implementing FIR filters of length-3 or higher requires the same amount of program memory (excluding coefficient storage). For example, an FIR filter of length-256 takes the same amount of program memory space as a FIR filter of length-4.

Since the TMS32020 instruction set is upward-compatible with the TMS32010 instruction set, it is possible to use the LTD/MPYK instruction pair to implement the filter. With the TMS32020, the designer can use either RPTK/MACD or LTD/MPY(K) where appropriate. Depending on the application and the data memory constraints, the use of the LTD/MPYK instruction pair results in less data memory usage at the cost of increasing the program memory storage.

The FIR filter program of Appendix A is an implementation of the same length-80 FIR filter used in the TMS32010 example. In this implementation, it can be seen that the TMS32020 uses less program memory than the TMS32010 with the tradeoff of using more data memory words. The increase in data memory size is indirectly related to the MACD instruction; i.e., in order to take full advantage of the instruction, it is necessary to keep the multiplier pipeline as busy as possible. Therefore, the filter will execute faster when all 80 coefficients are provided in block B0.

The TMS32020 provides a solution for the faster execution of FIR filters. The combination of the RPTK/MACD instructions provides for a minimum program memory and high-speed execution of an FIR filter. If data memory is a concern, the designer can use the LTD/MPYK instruction pair at the cost of increasing program memory and using 13-bit filter coefficients.

IIR Filters

The concepts introduced for the implementation of FIR filters can be extended to the implementation of IIR filters. However, for an IIR filter, at least one of the a_k in (1) is

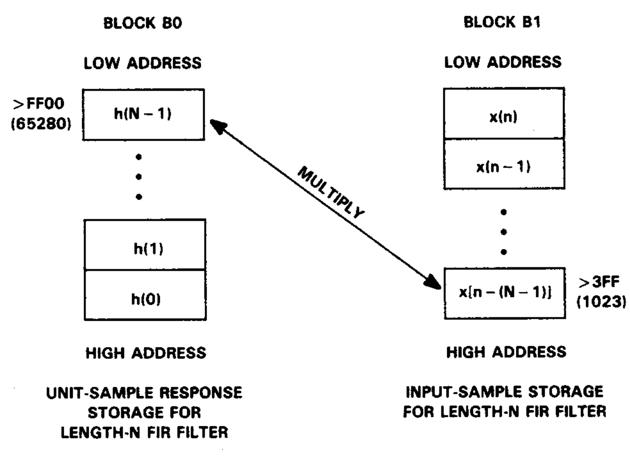


Figure 9. TMS32020 Memory Storage Scheme

```
* THIS SECTION OF CODE IMPLEMENTS THE EQUATION:
* x(n-(N-1))h(N-1) + x(n-(N-2))h(N-2) + ... + x(n)h(0) = y(n)
*
        CNFP                    * USE BLOCK B0 AS PROGRAM AREA
*
NXTPT   IN      XN,PA0          * BRING IN THE NEW SAMPLE XN
*
        LRLK    AR1,>3FF        * POINT TO THE BOTTOM OF BLOCK B1
        LARP    AR1
*
        MPYK    0               * SET P REGISTER TO ZERO
        ZAC                     * CLEAR THE ACCUMULATOR
*
        RPTK    NM1             * REPEAT N-1 TIMES
        MACD    >FF00,*-        * MULTIPLY/ACCUMULATE
*
        APAC
        SACH    YN,1
*
        OUT     YN,PA1          * OUTPUT THE FILTER RESPONSE y(n)
*
        B       NXTPNT          * GET THE NEXT POINT
```

Figure 10. TMS32020 Code for Implementing a Length-5 FIR Filter

nonzero. It has been shown[1] that the z transform of the unit-sample response of an IIR filter corresponding to (1) is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \tag{7}$$

where H(z), Y(z), and X(z) are the z transforms of h(n), y(n), and x(n), respectively. Three different network structures often used to implement (7) are the direct form, the cascade form, and the parallel form. Implementation of these structures is discussed in the following sections.

Direct-Form IIR Filter

Equations (1) and (7) may also be represented by the network structure shown in Figure 11. For convenience, it is assumed that M = N. This network structure is referred to as the direct-form I realization of an Nth-order difference equation. As was the case for the direct-form FIR filter, the structure in Figure 11 is called direct-form since the coefficients of the network can be obtained directly from the difference equation describing the network. Again, the branches associated with the z^{-1} correspond to the delays in (1) and the multiplications in (7).

The following difference equation:

$$y(n) = \sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k\, x(n-k) \tag{8}$$

shows that the output of the filter is a weighted sum of past values of the input to the filter and of the output of the filter. Using techniques similar to those for an FIR filter, this realization can be implemented in a straightforward and efficient way on the TMS32010 and TMS32020.

A network flowgraph equivalent to that in Figure 11 is shown in Figure 12. This system is referred to as the direct-form II structure. Since the direct-form II has the minimum number of delays (branches labeled z^{-1}), it requires the minimum number of storage registers for computation. This structure is advantageous for minimizing the amount of data memory used in the implementation of IIR filters.

In Figures 13 through 17, a second-order direct-form II IIR filter is used as an example for the TMS320 implementation of the IIR filter. The network structure is shown in Figure 13.

The difference equation for this network is

$$\begin{aligned} d(n) &= x(n) + a_1\, d(n-1) + a_2\, d(n-2) \\ y(n) &= b_0\, d(n) + b_1\, d(n-1) + b_2\, d(n-2) \end{aligned} \tag{9}$$

In this case, d(n), shown in (9) and Figure 13, corresponds to the network value at the different delay nodes. The zero-delay register corresponds to d(n); d(n − 1) is the register for the delay of one; and d(n − 2) is the register for the delay of two. A portion of the TMS32010 code necessary to implement (9) is shown in Figure 14. Initially all d(n − i) for i = 0,1,2 are set to zero.

The delay-node values of the filter are stored in data memory as shown in Figure 15. At each major step of the algorithm, a multiply is done, and the result from the previous multiply is added to the accumulator. Also, the past delay-node values are shifted to the next higher location in

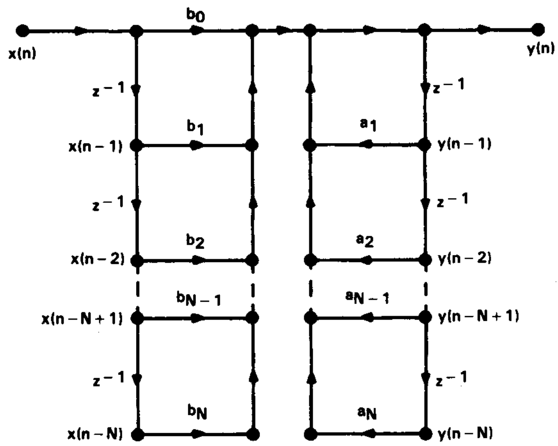


Figure 11. Direct-Form I IIR Filter

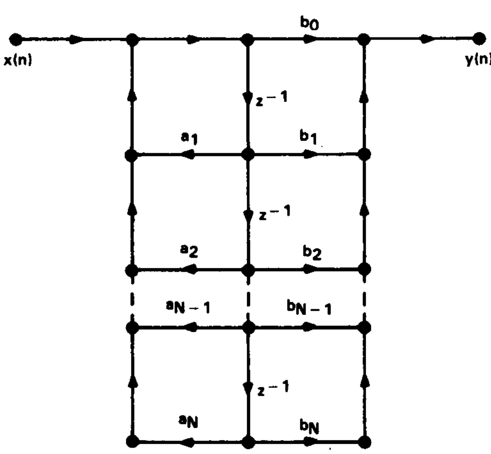


Figure 12. Direct-Form II IIR Filter

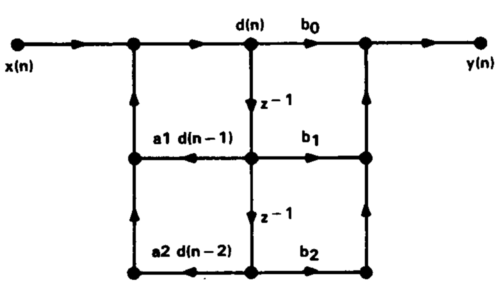


Figure 13. Second-Order Direct-Form II IIR Filter

data memory, thus placing them in the correct position for the next pass through the filter. All of these operations are carried out with instruction pairs, such as

```
LTD    DNM1
MPY    B1
```

where DNM1 corresponds to $d(n-1)$ and B1 corresponds to b_1 as in (9).

When the last multiplication is performed and the result is added to the accumulator, the accumulator contains the result of (9), which is $y(n)$. From (9) and Figure 13, it is evident that the delay-node value $d(n)$ depends on several of the previous delay-node values. This feedback is illustrated by the instruction

```
SACH   DN,1
```

and the use of the statements

```
LTD    DNM1
 .
 .
 .
LTD    DN
```

The ordering of the delay-node values, shown in Figure 15, allows for a simple program structure with minimal computations and minimal data locations. It also accommodates the shifting of the delay-node values in a straightforward way. The feedback of DN makes apparent the underlying structure of the direct-form II filter and (10). This form of the algorithm is flexible and can be extended to higher-order direct-form filters in a straightforward way.

```
*  THIS SECTION OF CODE IMPLEMENTS THE EQUATIONS:   *
*  d(n) = x(n) + d(n-1)a  + d(n-2)a                 *
*                       1          2                *
*  y(n) = d(n)b  + d(n-1)b  + d(n-2)b               *
*              0          1          2              *
*
*
          IN XN,PA0         * NEW INPUT VALUE XN *
*
          LAC XN,15         * LOAD ACCUMULATOR WITH XN *
*
          LT DNM1
          MPY A1
*
          LTA DNM2
          MPY A2
*
          APAC
*
          SACH DN,1         * d(n) = x(n) + d(n-1)a  + d(n-2)a  *
*                           *                      1          2 *
          ZAC
*
          MPY B2
*
          LTD DNM1
          MPY B1
*
          LTD DN
          MPY B0
*
          APAC
*
          SACH YN,1         * y(n) = d(n)b  + d(n-1)b  + d(n-2)b  *
*                           *             0          1          2 *
*
          OUT YN,PA1        * YN IS THE OUTPUT OF THE FILTER *
```

Figure 14. TMS32010 Code for Implementing a Second-Order Direct-Form II IIR Filter

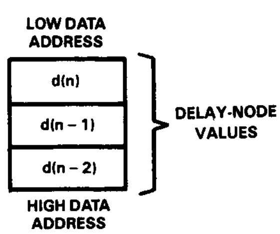


Figure 15. Delay-Node Value Storage for a Second-Order Direct-Form IIR Filter

Figure 16 shows the necessary ordering of the delay-node values for a general direct-form II structure for the case M ≥ N. Filter order is determined by M or N, whichever is greater.

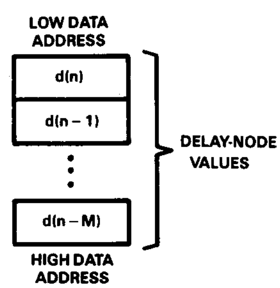


Figure 16. Delay-Node Value Storage for a Direct-Form II IIR Filter

Figure 17 shows a portion of the TMS32020 code for implementing the same second-order direct-form II IIR filter using the MACD instruction. As discussed in the section on FIR filters, using the RPTK/MACD instruction pair is most effective when the filter order is three or higher. The use of the MACD instruction allows the designer to save one word of program memory over the LTD/MPY implementation. The TMS32020 code in Figure 17 is provided only as an example. For a biquad implementation (second-order direct-form II IIR filter), the TMS32010 code and TMS32020 code for the filter implementation are identical. Note that due to larger on-chip RAM of the TMS32020, higher-order IIR filters or sections of IIR filters can be implemented. For the rest of the IIR filter structures, the same discussion applies to both processors.

An example of a TMS32010/TMS32020 program implementing a fourth-order direct-form II structure can be found in Appendix C.

Cascade-Form IIR Filter

In this section, the realization and implementation of cascade-form IIR filters are discussed. The implementation of a cascade-form IIR filter is an extension of the results of the implementation of the direct-form IIR filter.

The z transform of the unit-sample response of an IIR filter

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} \tag{10}$$

```
* THIS SECTION OF CODE IMPLEMENTS A SECOND-ORDER DIRECT-FORM II IIR FILTER
* d(n) = x(n) + d(n-1)a    + d(n-2)a
*                      1            2
* y(n) = d(n)b   + d(n-1)b   + d(n-2)b
*             0           1            2
*
NEXT     IN     XN,PA2          * NEW INPUT VALUE XN
*
         LAC    XN
         MPYK   0               * CLEAR P REGISTER
*
         LARP   AR1
         LRLK   AR1,>03FF
         CNFP                   * USE BLOCK B0 AS PROGRAM AREA
*
*  d(n) = x(n) + d(n-1)a   + d(n-2)a
*                       1           2
*
         RPTK   1               * REPEAT 2 TIMES
         MACD   >FF00,*+
*
         APAC
         SACH   DN,1            * d(n)
*
*  y(n) = d(n)b   + d(n-1)b   + d(n-10)b
*              0           1             2
*
         ZAC
         MPYK   0               * CLEAR P REGISTER
*
         MPY    >FF02
*
         RPTK   1
         MACD   >FF03,*-
*
         APAC
         SACH   YN,1            * SAVE FILTERED OUTPUT
*
         OUT    YN,PA2          * YN IS THE OUTPUT OF THE FILTER
         B      NEXT
```

Figure 17. TMS32020 Code for Implementing a Second-Order Direct-Form IIR Filter with MACD

may also be written in the equivalent form

$$H(z) = \prod_{k=1}^{N/2} \frac{\beta_{0k} + \beta_{1k}z^{-1} + \beta_{2k}z^{-2}}{1 - \alpha_{1k}z^{-1} - \alpha_{2k}z^{-2}} \tag{11}$$

where the filter is realized as a series of biquads. Therefore, this realization is referred to as the cascade form. Figure 18 shows a fourth-order IIR filter implemented in cascade structure, where the subsections are implemented as direct-form II sections. Each subsection corresponds to one of the terms in the product in (11). Note that any single cascade section is identical to the second-order direct-form II IIR filter described previously.

The difference equation for cascade section i can be written as

$$d_i(n) = y_{i-1}(n) + \alpha_{1i}\, d_i(n-1) + \alpha_{2i}\, d_i(n-2) \tag{12}$$

$$y_i(n) = \beta_{0i}\, d_i(n) + \beta_{1i}\, d_i(n-1) + \beta_{2i}\, d_i(n-2)$$

where

i	=	1,2,...,N/2.
$y_{i-1}(n)$	=	input to section i.
$d_i(n)$	=	value at a particular delay node in section 1.
$y_i(n)$	=	output of section i.
$y_0(n) = x(n)$	=	sample input to the filter.
$y_{N/2} = y(n)$	=	output of the filter.

For the IIR filter consisting of the two cascaded sections shown in Figure 18, there are two sets of equations describing the relationship between the input and output of the filter. The delay-node values for each section are stored as shown in Figure 19. The same indexing scheme used previously

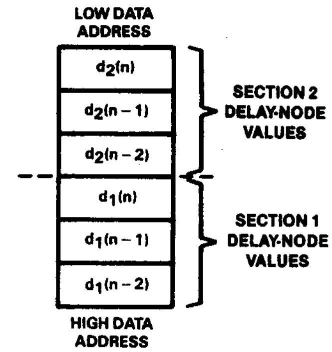


Figure 19. Delay-Node Storage for Cascaded IIR Filter Subsections

is used here (i.e., from the higher address in data memory to the lower address in data memory). In this case, the algorithm can be structured so that the 32-bit accumulator of the TMS320 acts as a storage register and carries the output of one of the second-order subsections to the input of the next second-order subsection. This avoids unnecessary truncation of the intermediate filter values into 16-bit words, and therefore provides better accuracy in the final output.

The implementation of the cascaded fourth-order IIR filter can be summarized as follows:

1. Load the new input value x(n).
2. Operate on the first section as outlined in Figure 12.
3. Leave the output of the first section in the accumulator (i.e., the SACH YN can be omitted for the first-section implementation since the accumulator links the output of one section to the input of the following section).
4. Operate on the second section in the same way as the first section, remembering that

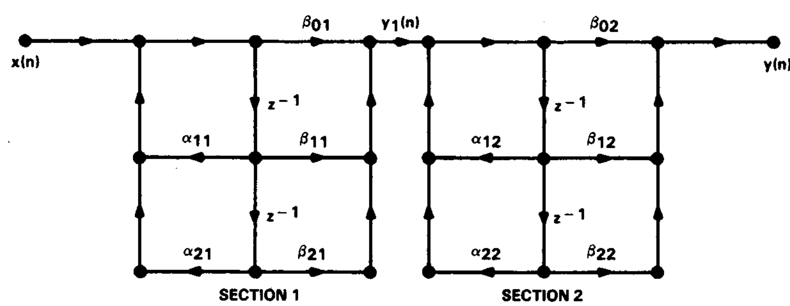


Figure 18. Fourth-Order Cascaded IIR Filter

the accumulator already contains the output of the previous section.

5. The output of the second section is the filter output y(n).

The above procedures can be applied to the IIR filter implementation of higher orders. It can be shown[3] that with proper ordering of the second-order cascades, the resulting filter has better immunity to quantization noise than the direct-form implementation, as will be discussed later.

An example of a TMS32010/TMS32020 program that implements a fourth-order IIR cascaded structure is contained in Appendix C.

Parallel-Form IIR Filter

The third form of an IIR filter is referred to as the parallel form. In this case, H(z) is written as

$$H(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{N/2} \frac{\gamma_{0k} + \gamma_{1k} z^{-1}}{1 - \alpha_{1k} z^{-1} - \alpha_{2k} z^{-2}} \tag{13}$$

If $M < N$, then the term $(C_k z^{-k}) = 0$. The network form is shown in Figure 20, where it is assumed that $M = N = 4$. The multiplication of the input by C (a constant) is trivial. However, for one of the parallel branches of this structure, the difference equation is

$$d_i(n) = x(n) + \alpha_{1i}\, d_i(n-1) + \alpha_{2i}\, d_i(n-2) \tag{14}$$

$$p_i(n) = \gamma_{0i}\, d_i(n) + \gamma_{1i}\, d_i(n-1)$$

where $i = 1,2,\ldots,N/2$, and $p_i(n)$ = the present output of a parallel branch.

The similarity to the second-order direct-form II network and the single parallel section is apparent. However, in this case, the outputs of all sections are summed to give the output y(n), i.e.,

$$y(n) = Cx(n) + \sum_{i=1}^{N/2} p_i(n) \tag{15}$$

if $M = N$. For the parallel implementation, the delay-node values are also structured in data memory, as shown in Figure 21, thus allowing for an implementation similar to that used previously. After the output of each section stored in the 32-bit accumulator is determined, these outputs are summed to yield the filter output y(n). An example of a TMS32010/TMS32020 program to implement a parallel structure can be found in Appendix C.

PERFORMANCE CONSIDERATIONS IN DIGITAL FILTER DESIGN

In the previous sections, different realizations of the FIR and IIR digital filters were discussed. This section is mainly concerned with the effects of finite wordlength on filter performance.

Some features of FIR and IIR filters, which distinguish them from each other and need special considerations when they are implemented, include phase characteristics, stability, and coefficient quantization effects.

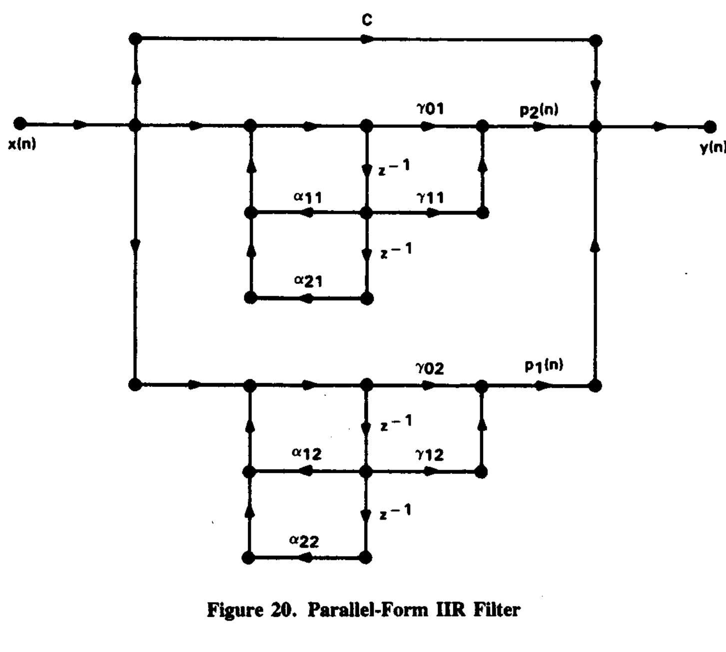


Figure 20. Parallel-Form IIR Filter

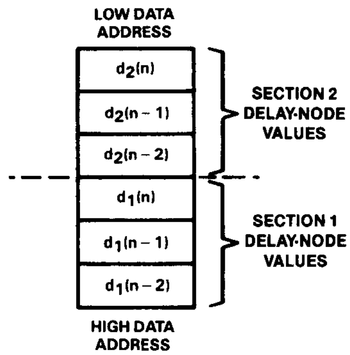


Figure 21. Delay-Node Value Storage for a Parallel IIR Filter

Given a set of frequency-response characteristics, typically a higher-order FIR filter is required to match these characteristics to a corresponding IIR filter. However, this does not imply that IIR filters should be used in all cases. In some applications, it is important that the filter have linear phase, and only FIR filters can be designed to have linear phase.

Another important consideration is the stability of the filter. Since the unit-sample response of an FIR filter is of finite length, FIR filters are inherently stable (i.e., a bounded input always produces a bounded output). This can be seen from (5) where the output of an FIR filter is a weighted finite sum of previous inputs. On the other hand, IIR filters may or may not be stable, depending on the locations of the poles of the filter.

Digital filters are designed with the assumption that the filter will be implemented on an infinite precision device. However, since all processors are of finite precision, it is necessary to approximate the "ideal" filter coefficients. This approximation introduces coefficient quantization error. The net result due to imprecise coefficient representations is a deviation of the resultant filter frequency response from the ideal one. For narrowband IIR filters with poles close to the unit circle, longer wordlengths may be required. The worst effect of coefficient quantization is instability resulting from poles being moved outside the unit circle.

The effect of coefficient quantization is highly dependent on the structure of the filter and the wordlength of the implementation hardware. Since the poles and zeroes for a filter implemented with finite wordlength arithmetic are not necessarily the same as the poles and zeroes of a filter implemented on an infinite precision device, the difference may affect the performance of the filter.

In the IIR filter, the cascade and parallel forms implement each pair of complex-conjugate poles separately. As a result, the coefficient quantization effect for each pair of complex-conjugate poles is independent of the other pairs of complex-conjugate poles. This is generally not true for direct-form filters. Therefore, the cascade and parallel forms of IIR filters are more commonly used than the direct form.

Another problem in implementing a digital filter is the quantization error due to the finite wordlength effect in the hardware. Sources of error arising from the use of finite wordlength include the following:

1. I/O signal quantization
2. Filter coefficient quantization
3. Uncorrelated roundoff (or truncation) noise
4. Correlated roundoff (or truncation) noise
5. Dynamic range constraints.

These problems are addressed in the following paragraphs in more detail.

Representing instantaneous values of a continuous-time signal in digital form introduces errors that are associated with I/O quantization. Input signals are subjected to A/D quantization noise while output signals are subjected to D/A quantization noise. Although output D/A noise is less detrimental, input A/D quantization noise is the more dominant factor in most systems. This is due to the fact that input noise "circulates" within IIR filters and can be "regenerative" while output noise normally just "propagates" off-stage.

The filter coefficients in all of the routines described in this report are initially stored in program memory, and then moved to data memory. These coefficients are represented in Q15 format; i.e., the binary point (represented in two's-complement form) is assumed to follow the most-significant bit. This gives a coefficient range of 0.999969 to -1.0 with increments of 0.000031. The input is also in Q15 format so that when two Q15 numbers are multiplied, the result is a number in Q30 format. When the Q30 number resides in the 32-bit accumulator of the TMS320, the binary point follows the second most-significant bit. Since the output of the filter is assumed to be in Q15 format, the Q30 number must be adjusted by left-shifting by one while maintaining the most-significant 16 bits of the result. This is accomplished with the step SACH YN,1, which shifts the Q30 number to the left by one and stores the upper sixteen bits of the accumulator following the shift. The result YN is in Q15 format. Note that it is important to keep intermediate values in the accumulator as long as possible to maintain the 32-bit accuracy.

Uncorrelated roundoff (or truncation) noise may occur in multiplications. Even though the input to the digital filter is represented with finite wordlength, the result of processing leads to values requiring additional bits for their representation. For example, a b-bit data sample, multiplied by a b-bit coefficient, results in a product that is 2b bits long. In a recursive filter realization, 2b bits are required after the first iteration, 3b bits after the second iteration, and so on. The fact that multiplication results have to be truncated means that every "multiplier" in a digital structure can be regarded as a noise source. The combined effects of various noise sources degrade system performance.

Truncation or rounding off the products formed within the digital filter is referred to as correlated roundoff noise. The result of correlated roundoff (or truncation) noise, including overflow oscillations, is that filters suffer from "limit-cycle effect" (small-amplitude oscillations). For systems with adequate coefficient wordlength and dynamic range, this problem is usually negligible. Overflows are generated by additions resulting in undesirable large-amplitude oscillations. Both limit cycles and overflow oscillations force the digital filter into nonlinear operations. Although limit cycles are difficult to eliminate, saturation arithmetic can be used to reduce overflow oscillations. The overflow mode of operation on the TMS320 family is accomplished with the SOVM (set overflow mode) instruction, which sets the accumulator to the largest representable 32-bit positive (>7FFFFFFF hex) or negative (>80000000 hex) number according to the direction of overflow.

Dynamic range constraints, such as scaling of parameters, can be used to prevent overflows and underflows of the finite wordlength registers. The dynamic range is the ratio between the largest and smallest signals that can be represented in a filter. For an FIR filter, an overflow of the output results in an error in the output sample. If the input sample has a maximum magnitude of unity, then the worstcase output is

$$y(n) = \sum_{n=0}^{N-1} h(n) = s \tag{16}$$

To guarantee y(n) to be a fraction, either the filter gain or the input x(n) has to be scaled down by a factor "s". Reducing the filter gain implies scaling down the filter coefficients so that the 16-bit coefficient is no longer used effectively. An implication of this scaling is a degradation of the filter frequency response due to higher quantization errors. As an alternative, the input signal may be scaled, resulting in a reduction in signal-to-noise ratio (SNR). In practice, the second approach is preferred since the scaling factor is normally less than two and does not change the SNR drastically. The required scaling on a TMS32020 is achieved by using the SPM (set P register output shift mode) instruction to invoke a right-shift by six bits to implement up to 128 multiply/accumulates without overflow occurring.

For an IIR filter, an overflow can cause an oscillation with full-scale amplitude, thus rendering the filter useless. In general, if the input signal x(n) is sinusoidal, the reciprocal of the gain "s" of the IIR filter is used to prevent output overflows.

For the TMS320 implementation with its double-precision accumulator and P register, scaling down the input sequence by the scaling factor "s" while maintaining a 16-bit accuracy for the coefficients can accomplish the task. For this reason, use of the MPYK instruction for IIR filter implementation is not recommended. Scaling the input signal by a factor "s" results in a degradation in the overall system SNR. Therefore, for IIR filters, it is important to keep the coefficient quantization errors as small as possible since less accurate coefficients may cause an unstable filter if the poles are moved outside the unit circle. The LAC (load accumulator with shift) instruction on the TMS320 processors easily accomplishes input signal scaling.

In the previous paragraphs, finite wordlength problems associated with digital filter implementation on programmable devices were discussed. The 16-bit coefficients and the 32-bit accumulator of the TMS320 processor help minimize the quantization effects. Special instructions also help overcome problems in the accumulator. These features, in addition to a powerful instruction set, make the TMS32010 and TMS32020 ideal programmable processors for filtering applications.

SOURCE CODE USING THE TMS320

Examples of TMS320 source code for the implementation of two FIR filters and three IIR filters, based on the techniques described in this application report, are contained in the appendixes. Plots of the magnitude response, log-magnitude response, unit-sample response, and other pertinent data precede the filter programs.

Five filter types are presented in the three appendixes as follows:

Appendix A	Length-80 bandpass FIR filter (TMS32010 and TMS32020)
Appendix B	Length-60 FIR differentiator (TMS32010/TMS32020)
Appendix C	Fourth-order lowpass IIR filters: direct-form, cascade, and parallel types (TMS32010/TMS32020)

The purpose of the source code is to further illustrate the use of the TMS320 devices for filtering applications and to allow implemention and analysis of these filters. The code is based on the programming techniques discussed earlier in this report.

TMS32020 source code is listed in the appendix for a length-80 FIR filter. The TMS32020 source code for the rest of the filter programs is identical to the TMS32010 code, as explained earlier. TMS32010 and TMS32020 instructions are compatible only at the mnemonic level. TMS32010 source programs should be reassembled using a TMS32020 assembler before execution. For more detail about code migration, refer to the TMS32020 User's Guide appendix, "TMS32010/TMS32020 System Migration," for detailed information.[6]

These filters were designed using the Digital Filter Design Package (DFDP) developed by Atlanta Signal Processors Incorporated (ASPI).[7] This package runs on either a Texas Instruments Professional Computer or an IBM Personal Computer and can generate TMS320 code for the filter designed. DFDP was used to design the FIR filters with the Remez exchange algorithm developed by Parks and McClellan, and to design the IIR filters by bilinear transformation of an elliptic analog prototype. All plots supplied with the filter programs were produced by DFDP.

Filter design packages, such as DFDP, make the design and implementation of digital filters straightforward. They allow the DSP engineer to quickly examine a variety of filters and understand the tradeoffs involved in varying the characteristics of the filters. Several digital filter design packages and other useful software support from third parties are described in the TMS32010 Development Support Reference Guide.[8]

All of the TMS320 source code examples have several features in common that depend on the implementation and application. These features include the moving of filter coefficients from storage in program memory to data memory, their representation in Q15 format, and the instructions that control the analog interface used for testing.

The hardware configuration that was used to test these filters included a Texas Instruments analog interface board (AIB) to provide an analog-to-digital and digital-to-analog interface. The sampling rate was 10 kHz in all cases. The filters were driven by a white-noise source, and the frequency response was estimated by a spectrum analyzer. Each filter routine contains several lines of code to initialize the analog interface board. The AIB signals the TMS320 that another input sample is available by pulling the $\overline{\text{BIO}}$ pin low. The TMS320 polls this pin using the BIOZ instruction. The AIB houses a TMS32010 device. In order to use the TMS32020 with the AIB (PN: RTC/EVM320C-06), a specially designed adaptor (PN: RTC/ADP320A-06) must be inserted to convert TMS32020 signals to TMS32010 signals. All of these implementation- and application-dependent sections of code are labeled.

Appendix A provides programs for the implementation of a length-80 linear-phase bandpass FIR filter on the TMS32010 and the TMS32020. The filter has been designed using the Parks-McClellan algorithm. Pertinent data for this filter is as follows:

Passband	1.375	-	3.625	kHz
Stopbands	0.0	-	1.0	kHz
	4.0	-	5.0	kHz
Attenuation in stopbands			-68.4	dB
Transition regions	1.0	-	1.375	kHz
	3.625	-	4.0	kHz

The figures preceding the program show the magnitude response using a linear scale, the log-magnitude response, and the unit-sample response. Both the magnitude response and the log-magnitude response illustrate the equiripple response expected from using the Parks-McClellan algorithm. The unit-sample response possesses the symmetry that is characteristic of linear-phase FIR filters.

A length-60 FIR differentiator, shown in Appendix B, is also designed using the Parks-McClellan algorithm. Characteristics for the FIR differentiator are listed below.

Lower band edge	0.0	kHz
Upper band edge	5.0	kHz
Desired slope	0.4800	
Maximum deviation	0.3172 percent	

The log-magnitude resonse is illustrated as well as the unit-sample response, which is antisymmetric for an FIR differentiator. Because the code is written in looped form, there is a dramatic reduction in the amount of program space necessary to implement this filter.

The three filters in Appendix C are fourth-order lowpass IIR filters, designed using the bilinear-transform technique. The first filter is based on a direct-form II structure, the second filter is based on a cascade structure with two second-order direct-form II subsections, and the third filter is based on a parallel structure. These three IIR filters are identical in terms of their frequency response and have the following characteristics:

Passband	0.0	-	2.5	kHz
Transition region	2.5	-	2.75	kHz
Stopband	2.75	-	5.0	kHz
Attenuation in stopband		-25.17		dB

The figures that show the magnitude response, log-magnitude response, phase response, group delay, and the unit-sample response for the three IIR filters are treated as a group and precede the three programs for filter implementation.

Table 1 is a summary of information about the five digital filters that are implemented in the appendixes.

An examination of the length-80 FIR filter implementation reveals the advantages of using a TMS32020 over the TMS32010. The program memory size is reduced by a factor of 15 (11 words vs. 163 words) while execution speed is improved by a factor of 1.8. Since the other filter types do not take advantage of the RPTK/MACD instruction pair, the performance results are the same. For example, a fourth-order cascade-form IIR filter executes at 5.4 μs using only 27 program memory words.

When implementing linear-phase FIR filters, the designer must choose the right device for the application. If fast execution time and less program memory are essential, then the TMS32020 is the right choice.

The IIR filters are direct transformations of analog filters, exhibiting the same amplitude and phase characteristics as their analog counterparts. IIR filters tend to be more efficient than FIR filters with respect to transitionband sharpness and filter orders required. Although they require less code for implementation than the FIR filters (TMS32010 straightline code), they show great nonlinearity in phase, which limits their use in some applications.

By far the most commonly used IIR structure is the cascade-form realization. It has been shown that proper ordering of the poles and zeroes results in less sensitivity to quantization noise. The Digital Filter Design Package designs IIR filters in cascade form only.

By using a TMS32020 for both FIR and IIR filter implementations, it is possible to design a higher-order filter

Table 1. Summary Table of Filter Programs

LENGTH-80 LINEAR-PHASE BANDPASS FIR (STRAIGHT-LINE CODE)				
CODE	CYCLES	EXECUTION TIME (MICROSECONDS)	PROGRAM MEMORY (WORDS)	DATA MEMORY (WORDS)
Straight Line:				
TMS32010	163	32.6	163	120
TMS32020 (with RPTK)	90	18	11	161
LENGTH-60 FIR DIFFERENTIATOR (LOOPED CODE)				
CODE	CYCLES	EXECUTION TIME (MICROSECONDS)	PROGRAM MEMORY (WORDS)	DATA MEMORY (WORDS)
Looped: TMS32010/20	243	48.6	11	120
FOURTH-ORDER LOWPASS IIR FILTERS				
STRUCTURE	CYCLES	EXECUTION TIME (MICROSECONDS)	PROGRAM MEMORY (WORDS)	DATA MEMORY (WORDS)
Direct-Form II: TMS32010/20	24	4.8	24	16
Cascade: TMS32010/20	27	5.4	27	18
Parallel: TMS3210/20	28	5.6	28	18

NOTE: The above performance figures are only given as a reference. They should not be taken as benchmarks since programs can always be improved for better speed and memory efficiency.

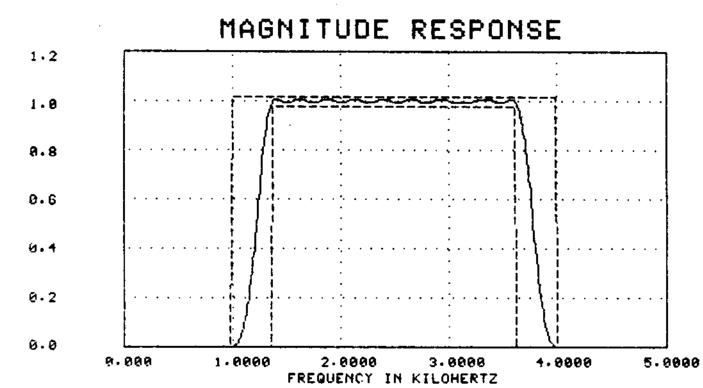
than with the TMS32010. The TMS32020 is also ideal for higher-order FIR filters that require single-cycle multiply/accumulate operations.

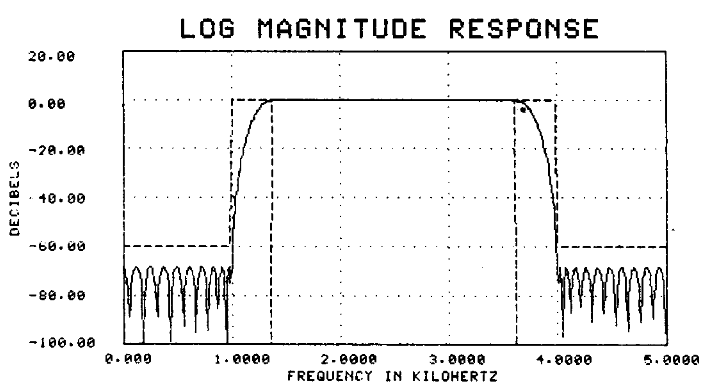
SUMMARY

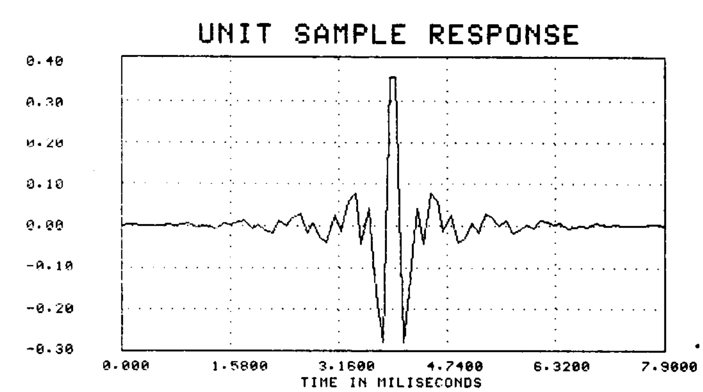
A brief review of FIR and IIR digital filters has been given to assist in understanding the fundamentals of digital filter structure and their implementations using a digital signal processor. Many design examples have also been included to show the tradeoffs between FIR and IIR structures.

This application report has also described methods for implementing FIR and IIR filters with the TMS32010 and TMS32020. The design engineer can now choose between the two devices, depending on the application.

APPENDIX A
LENGTH-80 LINEAR-PHASE PASSBAND FIR FILTER







```
                ****************************************************************
                *
                *                    LINEAR-PHASE FIR FILTER
                *                   LENGTH-80 BANDPASS FILTER
                *
                *       SAMPLING FREQUENCY = 10 KHZ
                *
                *                     FILTER CHARACTERISTICS
                *
                *                        BAND 1      BAND 2      BAND 3
                *
                *       LOWER BAND EDGE   0.0000      1.3750      4.0000
                *       UPPER BAND EDGE   1.0000      3.6250      5.0000
                *       NOMINAL GAIN      0.0000      1.0000      0.0000
                *       NOMINAL RIPPLE    0.0010      0.0200      0.0010
                *       MAXIMUM RIPPLE    0.0004      0.0076      0.0004
                *       RIPPLE IN DB    -68.3965      0.0657    -68.3997
                *
                *                        FILTER STRUCTURE
                *
                *
                *           -1       -1                          -1
                *          z        z                           z
                * o--->---o--->---o--->---o---> -  -  -o---->---o
                *  x(n)   |       |       |            |         |
                *         |       |       |            |         |
                *         v h(0)  v h(1)  v h(2)       v h(N-2)  v h(N-1)
                *         |       |       |            |         |
                *         |       |       |            |         |
                *         o--->---o--->---o---> -  -  -o--->----o--->---o
                *                                                   y(n)
                *
                ****************************************************************
                *
                *   CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
                *          | (MICROSECONDS) |     (WORDS)    |   (WORDS)
                *  --------+----------------+----------------+------------
                *          |                |                |
                *     90   |      18        |       10       |     161
                *          |                |                |
                *
                *   (EXCLUDING INITIALIZATION AND I/O)
                ****************************************************************
                *
                       IDT 'FIRBPASS'
          002D  YN     EQU     45
          002E  MODE   EQU     46
          002F  CLOCK  EQU     47
          0030  XN     EQU     48
                *
     0000              AORG 0
     0000 FF80         B START
     0001 0072
                *
     0020       CTABLE AORG    32
     0020 FFDC  CH0    DATA    >FFDC   * -0.107251E-02 *
     0021 001F  CH1    DATA    >001F   *  0.973976E-03 *
```



```
     0022 0051  CH2    DATA    >0051   *  0.249065E-02 *
     0023 FFE9  CH3    DATA    >FFE9   * -0.675043E-03 *
     0024 FFE6  CH4    DATA    >FFE6   * -0.771385E-03 *
     0025 FFBA  CH5    DATA    >FFBA   * -0.212256E-03 *
     0026 FFB4  CH6    DATA    >FFB4   * -0.229530E-02 *
     0027 004B  CH7    DATA    >004B   *  0.231021E-02 *
     0028 FFF9  CH8    DATA    >FFF9   * -0.194902E-03 *
     0029 0069  CH9    DATA    >0069   *  0.322896E-02 *
     002A 00A2  CH10   DATA    >00A2   *  0.496452E-02 *
     002B FF6F  CH11   DATA    >FF6F   * -0.440419E-02 *
     002C FFFE  CH12   DATA    >FFFE   * -0.314831E-04 *
     002D FF70  CH13   DATA    >FF70   * -0.438169E-02 *
     002E FEF4  CH14   DATA    >FEF4   * -0.815474E-02 *
     002F 00CB  CH15   DATA    >00CB   *  0.621682E-02 *
     0030 000B  CH16   DATA    >000B   *  0.342216E-03 *
     0031 00E6  CH17   DATA    >00E6   *  0.704627E-02 *
     0032 0187  CH18   DATA    >0187   *  0.119391E-01 *
     0033 FEE5  CH19   DATA    >FEE5   * -0.860811E-02 *
     0034 000B  CH20   DATA    >000B   *  0.346738E-03 *
     0035 FE7F  CH21   DATA    >FE7F   * -0.117293E-01 *
     0036 FDBF  CH22   DATA    >FDBF   * -0.175964E-01 *
     0037 0192  CH23   DATA    >0192   *  0.122947E-01 *
     0038 FFB5  CH24   DATA    >FFB5   * -0.227426E-02 *
     0039 026A  CH25   DATA    >026A   *  0.188796E-01 *
     003A 0368  CH26   DATA    >0368   *  0.266148E-01 *
     003B FDC2  CH27   DATA    >FDC2   * -0.175126E-01 *
     003C 00C0  CH28   DATA    >00C0   *  0.586574E-02 *
     003D FC0A  CH29   DATA    >FC0A   * -0.309240E-01 *
     003E FAA3  CH30   DATA    >FAA3   * -0.418954E-01 *
     003F 0347  CH31   DATA    >0347   *  0.256315E-01 *
     0040 FE3D  CH32   DATA    >FE3D   * -0.137498E-01 *
     0041 0747  CH33   DATA    >0747   *  0.568720E-01 *
     0042 09BB  CH34   DATA    >09BB   *  0.760286E-01 *
     0043 FA3D  CH35   DATA    >FA3D   * -0.450011E-01 *
     0044 052B  CH36   DATA    >052B   *  0.403853E-01 *
     0045 EB59  CH37   DATA    >EB59   * -0.161339E+00 *
     0046 DC2A  CH38   DATA    >DC2A   * -0.279963E+00 *
     0047 2D57  CH39   DATA    >2D57   *  0.352454E+00 *
     0048 2D57  CH40   DATA    >2D57   *  0.352454E+00 *
     0049 DC2A  CH41   DATA    >DC2A   * -0.279963E+00 *
     004A EB59  CH42   DATA    >EB59   * -0.161339E+00 *
     004B 052B  CH43   DATA    >052B   *  0.403853E-01 *
     004C FA3D  CH44   DATA    >FA3D   * -0.450011E-01 *
     004D 09BB  CH45   DATA    >09BB   *  0.760286E-01 *
     004E 0747  CH46   DATA    >0747   *  0.568720E-01 *
     004F FE3D  CH47   DATA    >FE3D   * -0.137498E-01 *
     0050 0347  CH48   DATA    >0347   *  0.256315E-01 *
     0051 FAA3  CH49   DATA    >FAA3   * -0.418954E-01 *
     0052 FC0A  CH50   DATA    >FC0A   * -0.309240E-01 *
     0053 00C0  CH51   DATA    >00C0   *  0.586574E-02 *
     0054 FDC2  CH52   DATA    >FDC2   * -0.175126E-01 *
     0055 0368  CH53   DATA    >0368   *  0.266148E-01 *
     0056 026A  CH54   DATA    >026A   *  0.188796E-01 *
     0057 FFB5  CH55   DATA    >FFB5   * -0.227426E-02 *
     0058 0192  CH56   DATA    >0192   *  0.122947E-01 *
     0059 FDBF  CH57   DATA    >FDBF   * -0.175964E-01 *
     005A FE7F  CH58   DATA    >FE7F   * -0.117293E-01 *
```

```
005B 000B  CH59   DATA   >000B    *  0.346738E-03 *
005C FEE5  CH60   DATA   >FEE5    * -0.860811E-02 *
005D 0187  CH61   DATA   >0187    *  0.119391E-01 *
005E 00E6  CH62   DATA   >00E6    *  0.704627E-02 *
005F 000B  CH63   DATA   >000B    *  0.342216E-03 *
0060 00CB  CH64   DATA   >00CB    *  0.621682E-02 *
0061 FEF4  CH65   DATA   >FEF4    * -0.815474E-02 *
0062 FF70  CH66   DATA   >FF70    * -0.438169E-02 *
0063 FFFE  CH67   DATA   >FFFE    * -0.314831E-04 *
0064 FF6F  CH68   DATA   >FF6F    * -0.440419E-02 *
0065 00A2  CH69   DATA   >00A2    *  0.496452E-02 *
0066 0069  CH70   DATA   >0069    *  0.322896E-02 *
0067 FFF9  CH71   DATA   >FFF9    * -0.194902E-03 *
0068 004B  CH72   DATA   >004B    *  0.231021E-02 *
0069 FFB4  CH73   DATA   >FFB4    * -0.229530E-02 *
006A FFBA  CH74   DATA   >FFBA    * -0.212256E-03 *
006B FFE6  CH75   DATA   >FFE6    * -0.771385E-03 *
006C FFE9  CH76   DATA   >FFE9    * -0.675043E-03 *
006D 0051  CH77   DATA   >0051    *  0.249065E-02 *
006E 001F  CH78   DATA   >001F    *  0.973976E-03 *
006F FFDC  CH79   DATA   >FFDC    * -0.107251E-02 *
                  *
0070 000A  MD     DATA   >000A
0071 01F3  SMP    DATA   >01F3           ; SAMPLING RATE OF 10 KHZ *
                  *
     0072  START  EQU    $
                  *
                  * INITIALIZATION OF THE ANALOG INTERFACE BOARD
                  *
0072 C807         LDPK   7
0073 CA70         LACK   MD
0074 582E         TBLR   MODE
0075 E02E         OUT    MODE,PA0
0076 CA71         LACK   SMP
0077 582F         TBLR   CLOCK
0078 E12F         OUT    CLOCK,PA1
                  *
                  * LOAD FILTER COEFFICIENTS
                  *
0079 5588         LARP   AR0             ; USE AR0 FOR INDIRECT ADDRESSING
007A D000         LRLK   AR0,>200        ; POINT TO BLOCK B0
007B 0200
007C CB4F         RPTK   >4F             ; 80 COEFFICIENTS
007D FCA0         BLKP   CTABLE,*+
007E 0020
                  *
007F CE05         CNFP                   ; USE BLOCK B0 AS PROGRAM AREA
                  *
0080 FA80  WAIT   BIOZ   NXTPT           ; BIO PIN GOES LOW WHEN A
0081 0084
0082 FF80         B      WAIT            ; NEW SAMPLE IS AVAILABLE
0083 0080
                  *
0084 8230  NXTPT  IN     XN,PA2          ; BRING IN THE NEW SAMPLE XN
                  *
0085 D100         LRLK   AR1,>3FF        ; POINT TO THE BOTTOM OF BLOCK B1
0086 03FF
```



```
0087 5589         LARP   AR1
                  *
0088 A000         MPYK   0
0089 CA00         ZAC
                  *
008A CB4F         RPTK   >4F
008B 5C90         MACD   >FF00,*-
008C FF00
                  *
008D CE15         APAC
008E 692D         SACH   YN,1
                  *
008F E22D         OUT    YN,PA2          ; OUTPUT THE FILTER RESPONSE y(n)
                  *
0090 FF80         B      WAIT            ; GO GET THE NEXT POINT
0091 0080
                  *
                  END
NO ERRORS, NO WARNINGS
```

```
        **************************************************************
        *
        *                    LINEAR-PHASE FIR FILTER
        *                   LENGTH-80 BANDPASS FILTER
        *
        *    SAMPLING FREQUENCY = 10 KHZ
        *
        *                      FILTER CHARACTERISTICS
        *
        *                         BAND 1      BAND 2      BAND 3
        *
        *    LOWER BAND EDGE     0.0000      1.3750      4.0000
        *    UPPER BAND EDGE     1.0000      3.6250      5.0000
        *    NOMINAL GAIN        0.0000      1.0000      0.0000
        *    NOMINAL RIPPLE      0.0010      0.0200      0.0010
        *    MAXIMUM RIPPLE      0.0004      0.0076      0.0004
        *    RIPPLE IN DB      -68.3965      0.0657    -68.3997
        *
        *                        FILTER STRUCTURE
        *
        *
        *              -1       -1                        -1
        *             z        z                         z
        * o--->---o--->---o--->---o---> -  -  -o---->---o
        *  x(n)   |       |       |            |        |
        *         |       |       |            |        |
        *         v h(0)  v h(1)  v h(2)       v h(N-2) v h(N-1)
        *         |       |       |            |        |
        *         |       |       |            |        |
        *         o--->---o--->---o---> -  -  -o--->----o--->---o
        *                                                    y(n)
        *
        **************************************************************
        *
        *  CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
        *         | (MICROSECONDS) |    (WORDS)     |   (WORDS)
        *  -------|----------------|----------------|-------------
        *         |                |                |
        *   163   |      32.6      |      163       |     120
        *         |                |                |
        *  ----------------------------------------------------------
        *
        *  (EXCLUDING INITIALIZATION AND I/O)
        *
        **************************************************************
        *
                IDT 'FIRBPASS'
0000    XN      EQU 0
0001    XNM1    EQU 1
0002    XNM2    EQU 2
0003    XNM3    EQU 3
0004    XNM4    EQU 4
0005    XNM5    EQU 5
0006    XNM6    EQU 6
0007    XNM7    EQU 7
0008    XNM8    EQU 8
0009    XNM9    EQU 9
```



```
000A    XNM10   EQU 10
000B    XNM11   EQU 11
000C    XNM12   EQU 12
000D    XNM13   EQU 13
000E    XNM14   EQU 14
000F    XNM15   EQU 15
0010    XNM16   EQU 16
0011    XNM17   EQU 17
0012    XNM18   EQU 18
0013    XNM19   EQU 19
0014    XNM20   EQU 20
0015    XNM21   EQU 21
0016    XNM22   EQU 22
0017    XNM23   EQU 23
0018    XNM24   EQU 24
0019    XNM25   EQU 25
001A    XNM26   EQU 26
001B    XN[illegible]   EQU 27
001C    X[illegible]28   EQU 28
001D    XNM29   EQU 29
001E    XNM30   EQU 30
001F    XNM31   EQU 31
0020    XNM32   EQU 32
0021    XNM33   EQU 33
0022    XNM34   EQU 34
0023    XNM35   EQU 35
0024    XNM36   EQU 36
0025    XNM3[illegible]   EQU 37
0026    XNM38   EQU 38
0027    XNM39   EQU 39
0028    XNM40   EQU 40
0029    XNM41   EQU 41
002A    XNM42   EQU 42
002B    XNM43   EQU 43
002C    XNM44   EQU 44
002D    XNM45   EQU 45
002E    XNM46   EQU 46
002F    XNM47   EQU 47
0030    XNM48   EQU 48
0031    XNM49   EQU 49
0032    XNM50   EQU 50
0033    XNM51   EQU 51
0034    XNM52   EQU 52
0035    XNM53   EQU 53
0036    XNM54   EQU 54
0037    XNM55   EQU 55
0038    XNM56   EQU 56
0039    XNM57   EQU 57
003A    XNM58   EQU 58
003B    XNM59   EQU 59
003C    XNM60   EQU 60
003D    XNM61   EQU 61
003E    XNM62   EQU 62
003F    XNM63   EQU 63
0040    XNM64   EQU 64
0041    XNM65   EQU 65
0042    XNM66   EQU 66
```

```
     0043  XNM67  EQU 67
     0044  XNM68  EQU 68
     0045  XNM69  EQU 69
     0046  XNM70  EQU 70
     0047  XNM71  EQU 71
     0048  XNM72  EQU 72
     0049  XNM73  EQU 73
     004A  XNM74  EQU 74
     004B  XNM75  EQU 75
     004C  XNM76  EQU 76
     004D  XNM77  EQU 77
     004E  XNM78  EQU 78
     004F  XNM79  EQU 79
           *
     0050  H0     EQU 80
     0051  H1     EQU 81
     0052  H2     EQU 82
     0053  H3     EQU 83
     0054  H4     EQU 84
     0055  H5     EQU 85
     0056  H6     EQU 86
     0057  H7     EQU 87
     0058  H8     EQU 88
     0059  H9     EQU 89
     005A  H10    EQU 90
     005B  H11    EQU 91
     005C  H12    EQU 92
     005D  H13    EQU 93
     005E  H14    EQU 94
     005F  H15    EQU 95
     0060  H16    EQU 96
     0061  H17    EQU 97
     0062  H18    EQU 98
     0063  H19    EQU 99
     0064  H20    EQU 100
     0065  H21    EQU 101
     0066  H22    EQU 102
     0067  H23    EQU 103
     0068  H24    EQU 104
     0069  H25    EQU 105
     006A  H26    EQU 106
     006B  H27    EQU 107
     006C  H28    EQU 108
     006D  H29    EQU 109
     006E  H30    EQU 110
     006F  H31    EQU 111
     0070  H32    EQU 112
     0071  H33    EQU 113
     0072  H34    EQU 114
     0073  H35    EQU 115
     0074  H36    EQU 116
     0075  H37    EQU 117
     0076  H38    EQU 118
     0077  H39    EQU 119
           *
     0078  MODE   EQU 120
     0079  CLOCK  EQU 121
```



```
     007A YN      EQU 122
     007B ONE     EQU 123
          *
0000              AORG 0
          *
0000 F900         B START
0001 002C
          *
          * COEFFICIENTS ARE INITIALLY *
          * STORED IN PROGRAM MEMORY   *
          *
          * DUE TO THE SYMMETRY OF THE IMPULSE RESPONSE *
          * ONLY HALF OF THE SAMPLES OF THE IMPULSE     *
          * RESPONSE ARE STORED. THIS MEANS THAT        *
          * h(N-1-n) = h(n).                            *
          *
0002 FFDC CH0     DATA    >FFDC   * -0.107251E-02 *
0003 001F CH1     DATA    >001F   *  0.973976E-03 *
0004 0051 CH2     DATA    >0051   *  0.249065E-02 *
0005 FFE9 CH3     DATA    >FFE9   * -0.675043E-03 *
0006 FFE6 CH4     DATA    >FFE6   * -0.771385E-03 *
0007 FFBA CH5     DATA    >FFBA   * -0.212256E-03 *
0008 FFB4 CH6     DATA    >FFB4   * -0.229530E-02 *
0009 004B CH7     DATA    >004B   *  0.231021E-02 *
000A FFF9 CH8     DATA    >FFF9   * -0.194902E-03 *
000B 0069 CH9     DATA    >0069   *  0.322896E-02 *
000C 00A2 CH10    DATA    >00A2   *  0.496452E-02 *
000D FF6F CH11    DATA    >FF6F   * -0.440419E-02 *
000E FFFE CH12    DATA    >FFFE   * -0.314831E-04 *
000F FF70 CH13    DATA    >FF70   * -0.438169E-02 *
0010 FEF4 CH14    DATA    >FEF4   * -0.815474E-02 *
0011 00CB CH15    DATA    >00CB   *  0.621682E-02 *
0012 000B CH16    DATA    >000B   *  0.342216E-03 *
0013 00E6 CH17    DATA    >00E6   *  0.704627E-02 *
0014 0187 CH18    DATA    >0187   *  0.119391E-01 *
0015 FEE5 CH19    DATA    >FEE5   * -0.860811E-02 *
0016 000B CH20    DATA    >000B   *  0.346738E-03 *
0017 FE7F CH21    DATA    >FE7F   * -0.117293E-01 *
0018 FDBF CH22    DATA    >FDBF   * -0.175964E-01 *
0019 0192 CH23    DATA    >0192   *  0.122947E-01 *
001A FFB5 CH24    DATA    >FFB5   * -0.227426E-02 *
001B 026A CH25    DATA    >026A   *  0.188796E-01 *
001C 0368 CH26    DATA    >0368   *  0.266148E-01 *
001D FDC2 CH27    DATA    >FDC2   * -0.175126E-01 *
001E 00C0 CH28    DATA    >00C0   *  0.586574E-02 *
001F FC0A CH29    DATA    >FC0A   * -0.309240E-01 *
0020 FAA3 CH30    DATA    >FAA3   * -0.418954E-01 *
0021 0347 CH31    DATA    >0347   *  0.256315E-01 *
0022 FE3D CH32    DATA    >FE3D   * -0.137498E-01 *
0023 0747 CH33    DATA    >0747   *  0.568720E-01 *
0024 09BB CH34    DATA    >09BB   *  0.760286E-01 *
0025 FA3D CH35    DATA    >FA3D   * -0.450011E-01 *
0026 052B CH36    DATA    >052B   *  0.403853E-01 *
0027 EB59 CH37    DATA    >EB59   * -0.161339E+00 *
0028 DC2A CH38    DATA    >DC2A   * -0.279963E+00 *
0029 2D57 CH39    DATA    >2D57   *  0.352454E+00 *
          *
```

```
002A 000A  MD     DATA    >000A
002B 01F3  SMP    DATA    >01F3    * SAMPLING RATE OF 10 KHZ *
                  *
002C 6E00  START  LDPK 0
                  *
002D 7E01         LACK 1
002E 507B         SACL ONE         * CONTENT OF ONE IS 1 *
                  *
002F 7079         LARK AR0,CLOCK   * THIS SECTION OF CODE LOADS  *
0030 7129         LARK AR1,>29     * THE FILTER COEFFICIENTS AND *
0031 7E2B         LACK SMP         * OTHER VALUES FROM PROGRAM   *
0032 6880  LOAD   LARP AR0         * MEMORY TO DATA MEMORY       *
0033 6791         TBLR *-,AR1
0034 107B         SUB ONE
0035 F400         BANZ LOAD
0036 0032
                  *
0037 4878         OUT MODE,PA0     * INITIALIZATION OF ANALOG *
0038 4979         OUT CLOCK,PA1    * INTERFACE BOARD          *
                  *
0039 F600  WAIT   BIOZ NXTPT       * BIO PIN GOES LOW WHEN A *
003A 003D
003B F900         B WAIT           * NEW SAMPLE IS AVAILABLE *
003C 0039
                  *
003D 4200  NXTPT  IN XN,PA2        * BRING IN THE NEW SAMPLE XN *
                  *
003E 7F89         ZAC
                  *
003F 6A4F         LT XNM79         * DUE TO SYMMETRY h(0) = h(79) *
0040 6D50         MPY H0           * x(n-79) * h(79) *
                  *
0041 6B4E         LTD XNM78
0042 6D51         MPY H1           * h(1) = h(78) *
                  *
0043 6B4D         LTD XNM77
0044 6D52         MPY H2
                  *
0045 6B4C         LTD XNM76
0046 6D53         MPY H3
                  *
0047 6B4B         LTD XNM75
0048 6D54         MPY H4
                  *
0049 6B4A         LTD XNM74
004A 6D55         MPY H5
                  *
004B 6B49         LTD XNM73
004C 6D56         MPY H6
                  *
004D 6B48         LTD XNM72
004E 6D57         MPY H7
                  *
004F 6B47         LTD XNM71
0050 6D58         MPY H8
                  *
0051 6B46         LTD XNM70
```



```
0052 6D59         MPY H9
                  *
0053 6B45         LTD XNM69
0054 6D5A         MPY H10
                  *
0055 6B44         LTD XNM68
0056 6D5B         MPY H11
                  *
0057 6B43         LTD XNM67
0058 6D5C         MPY H12
                  *
0059 6B42         LTD XNM66
005A 6D5D         MPY H13
                  *
005B 6B41         LTD XNM65
005C 6D5E         MPY H14
                  *
005D 6B40         LTD XNM64
005E 6D5F         MPY H15
                  *
005F 6B3F         LTD XNM63
0060 6D60         MPY H16
                  *
0061 6B3E         LTD XNM62
0062 6D61         MPY H17
                  *
0063 6B3D         LTD XNM61
0064 6D62         MPY H18
                  *
0065 6B3C         LTD XNM60
0066 6D63         MPY H19
                  *
0067 6B3B         LTD XNM59
0068 6D64         MPY H20
                  *
0069 6B3A         LTD XNM58
006A 6D65         MPY H21
                  *
006B 6B39         LTD XNM57
006C 6D66         MPY H22
                  *
006D 6B38         LTD XNM56
006E 6D67         MPY H23
                  *
006F 6B37         LTD XNM55
0070 6D68         MPY H24
                  *
0071 6B36         LTD XNM54
0072 6D69         MPY H25
                  *
0073 6B35         LTD XNM53
0074 6D6A         MPY H26
                  *
0075 6B34         LTD XNM52
0076 6D6B         MPY H27
                  *
0077 6B33         LTD XNM51
```

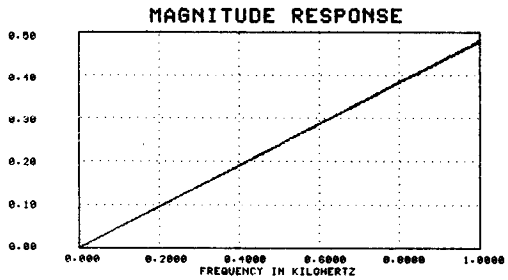
```
0339 0078 6D6C         MPY H28
0340            *
0341 0079 6B32         LTD XNM50
0342 007A 6D6D         MPY H29
0343            *
0344 007B 6B31         LTD XNM49
0345 007C 6D6E         MPY H30
0346            *
0347 007D 6B30         LTD XNM48
0348 007E 6D6F         MPY H31
0349            *
0350 007F 6B2F         LTD XNM47
0351 0080 6D70         MPY H32
0352            *
0353 0081 6B2E         LTD XNM46
0354 0082 6D71         MPY H33
0355            *
0356 0083 6B2D         LTD XNM45
0357 0084 6D72         MPY H34
0358            *
0359 0085 6B2C         LTD XNM44
0360 0086 6D73         MPY H35
0361            *
0362 0087 6B2B         LTD XNM43
0363 0088 6D74         MPY H36
0364            *
0365 0089 6B2A         LTD XNM42
0366 008A 6D75         MPY H37
0367            *
0368 008B 6B29         LTD XNM41
0369 008C 6D76         MPY H38
0370            *
0371 008D 6B28         LTD XNM40
0372 008E 6D77         MPY H39
0373            *
0374 008F 6B27         LTD XNM39
0375 0090 6D77         MPY H39
0376            *
0377 0091 6B26         LTD XNM38
0378 0092 6D76         MPY H38
0379            *
0380 0093 6B25         LTD XNM37
0381 0094 6D75         MPY H37
0382            *
0383 0095 6B24         LTD XNM36
0384 0096 6D74         MPY H36
0385            *
0386 0097 6B23         LTD XNM35
0387 0098 6D73         MPY H35
0388            *
0389 0099 6B22         LTD XNM34
0390 009A 6D72         MPY H34
0391            *
0392 009B 6B21         LTD XNM33
0393 009C 6D71         MPY H33
0394            *
0395 009D 6B20         LTD XNM32
```

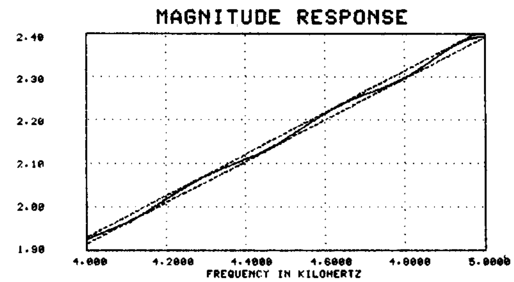


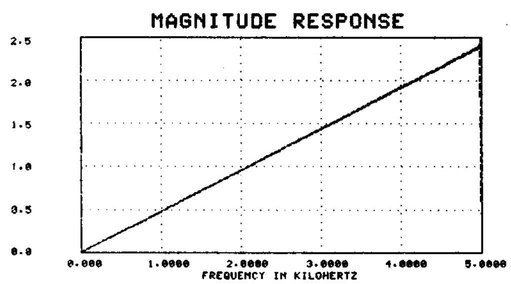
```
0396 009E 6D70         MPY H32
0397            *
0398 009F 6B1F         LTD XNM31
0399 00A0 6D6F         MPY H31
0400            *
0401 00A1 6B1E         LTD XNM30
0402 00A2 6D6E         MPY H30
0403            *
0404 00A3 6B1D         LTD XNM29
0405 00A4 6D6D         MPY H29
0406            *
0407 00A5 6B1C         LTD XNM28
0408 00A6 6D6C         MPY H28
0409            *
0410 00A7 6B1B         LTD XNM27
0411 00A8 6D6B         MPY H27
0412            *
0413 00A9 6B1A         LTD XNM26
0414 00AA 6D6A         MPY H26
0415            *
0416 00AB 6B19         LTD XNM25
0417 00AC 6D69         MPY H25
0418            *
0419 00AD 6B18         LTD XNM24
0420 00AE 6D68         MPY H24
0421            *
0422 00AF 6B17         LTD XNM23
0423 00B0 6D67         MPY H23
0424            *
0425 00B1 6B16         LTD XNM22
0426 00B2 6D66         MPY H22
0427            *
0428 00B3 6B15         LTD XNM21
0429 00B4 6D65         MPY H21
0430            *
0431 00B5 6B14         LTD XNM20
0432 00B6 6D64         MPY H20
0433            *
0434 00B7 6B13         LTD XNM19
0435 00B8 6D63         MPY H19
0436            *
0437 00B9 6B12         LTD XNM18
0438 00BA 6D62         MPY H18
0439            *
0440 00BB 6B11         LTD XNM17
0441 00BC 6D61         MPY H17
0442            *
0443 00BD 6B10         LTD XNM16
0444 00BE 6D60         MPY H16
0445            *
0446 00BF 6B0F         LTD XNM15
0447 00C0 6D5F         MPY H15
0448            *
0449 00C1 6B0E         LTD XNM14
0450 00C2 6D5E         MPY H14
0451            *
0452 00C3 6B0D         LTD XNM13
```

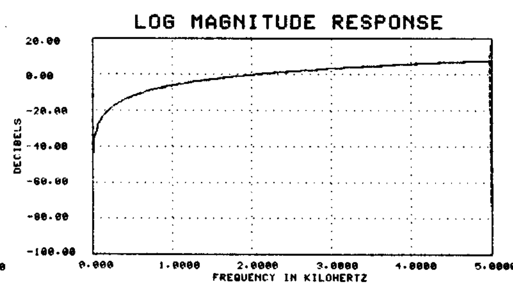
```
00C4 6D5D          MPY H13
               *
00C5 6B0C          LTD XNM12
00C6 6D5C          MPY H12
               *
00C7 6B0B          LTD XNM11
00C8 6D5B          MPY H11
               *
00C9 6B0A          LTD XNM10
00CA 6D5A          MPY H10
               *
00CB 6B09          LTD XNM9
00CC 6D59          MPY H9
               *
00CD 6B08          LTD XNM8
00CE 6D58          MPY H8
               *
00CF 6B07          LTD XNM7
00D0 6D57          MPY H7
               *
00D1 6B06          LTD XNM6
00D2 6D56          MPY H6
               *
00D3 6B05          LTD XNM5
00D4 6D55          MPY H5
               *
00D5 6B04          LTD XNM4
00D6 6D54          MPY H4
               *
00D7 6B03          LTD XNM3
00D8 6D53          MPY H3
               *
00D9 6B02          LTD XNM2
00DA 6D52          MPY H2
               *
00DB 6B01          LTD XNM1
00DC 6D51          MPY H1
               *
00DD 6B00          LTD XN
00DE 6D50          MPY H0
               *
00DF 7F8F          APAC
               *
00E0 597A          SACH YN,1
               *
00E1 4A7A          OUT YN,PA2      * OUTPUT THE FILTER RESPONSE y(n) *
               *
00E2 F900          B WAIT          * GO GET THE NEXT POINT *
00E3 0039
               *
                   END
NO ERRORS, NO WARNINGS
```

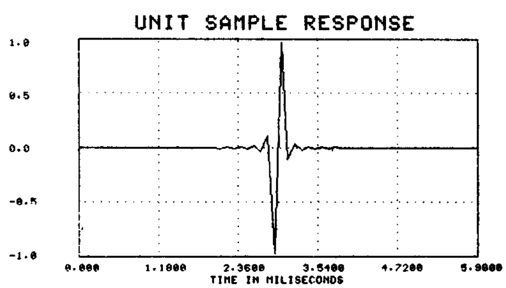
APPENDIX B
LENGTH-60 FIR DIFFERENTIATOR











```
              ***********************************************************
              *
              *                     FIR FILTER
              *               LENGTH-60 DIFFERENTIATOR
              *
              *                FILTER CHARACTERISTICS
              *
              *     SAMPLING FREQUENCY = 10 KHZ
              *
              *     LOWER BAND EDGE          0.0000
              *     UPPER BAND EDGE          5.0000
              *     DESIRED SLOPE            0.4800
              *     MAX % DEVIATION          0.3171
              *
              *                     FILTER STRUCTURE
              *
              *
              *              -1      -1                       -1
              *            z       z                        z
              * o--->---o--->---o--->---o---> -  -  -o---->---o
              *  x(n)   |       |       |               |         |
              *         |       |       |               |         |
              *         v h(0)  v h(1)  v h(2)          v h(N-2)  v h(N-1)
              *         |       |       |               |         |
              *         |       |       |               |         |
              *         o--->---o--->---o---> -  -  -o---->---o--->---o
              *                                                    y(n)
              *
              ***********************************************************
              *
              *   CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
              *          | (MICROSECONDS) |    (WORDS)     |   (WORDS)
              *   -------|----------------|----------------|------------
              *          |                |                |
              *     243  |     48.6       |       11       |    120
              *          |                |                |
              *   ------------------------------------------------------
              *
              *   (EXCLUDING I/O AND INITIALIZATION)
              *
              ***********************************************************
              *
                    IDT 'FIRDIF'
0000   XN           EQU 0
0001   XNM1         EQU 1
0002   XNM2         EQU 2
0003   XNM3         EQU 3
0004   XNM4         EQU 4
0005   XNM5         EQU 5
0006   XNM6         EQU 6
0007   XNM7         EQU 7
0008   XNM8         EQU 8
0009   XNM9         EQU 9
000A   XNM10        EQU 10
000B   XNM11        EQU 11
000C   XNM12        EQU 12
000D   XNM13        EQU 13
```



```
000E   XNM14   EQU 14
000F   XNM15   EQU 15
0010   XNM16   EQU 16
0011   XNM17   EQU 17
0012   XNM18   EQU 18
0013   XNM19   EQU 19
0014   XNM20   EQU 20
0015   XNM21   EQU 21
0016   XNM22   EQU 22
0017   XNM23   EQU 23
0018   XNM24   EQU 24
0019   XNM25   EQU 25
001A   XNM26   EQU 26
001B   XNM27   EQU 27
001C   XNM28   EQU 28
001D   XNM29   EQU 29
001E   XNM30   EQU 30
001F   XNM31   EQU 31
0020   XNM32   EQU 32
0021   XNM33   EQU 33
0022   XNM34   EQU 34
0023   XNM35   EQU 35
0024   XNM36   EQU 36
0025   XNM37   EQU 37
0026   XNM38   EQU 38
0027   XNM39   EQU 39
0028   XNM40   EQU 40
0029   XNM41   EQU 41
002A   XNM42   EQU 42
002B   XNM43   EQU 43
002C   XNM44   EQU 44
002D   XNM45   EQU 45
002E   XNM46   EQU 46
002F   XNM47   EQU 47
0030   XNM48   EQU 48
0031   XNM49   EQU 49
0032   XNM50   EQU 50
0033   XNM51   EQU 51
0034   XNM52   EQU 52
0035   XNM53   EQU 53
0036   XNM54   EQU 54
0037   XNM55   EQU 55
0038   XNM56   EQU 56
0039   XNM57   EQU 57
003A   XNM58   EQU 58
003B   XNM59   EQU 59
       *
003C   H0      EQU 60
003D   H1      EQU 61
003E   H2      EQU 62
003F   H3      EQU 63
0040   H4      EQU 64
0041   H5      EQU 65
0042   H6      EQU 66
0043   H7      EQU 67
0044   H8      EQU 68
0045   H9      EQU 69
```

```
0115      0046  H10    EQU 70
0116      0047  H11    EQU 71
0117      0048  H12    EQU 72
0118      0049  H13    EQU 73
0119      004A  H14    EQU 74
0120      004B  H15    EQU 75
0121      004C  H16    EQU 76
0122      004D  H17    EQU 77
0123      004E  H18    EQU 78
0124      004F  H19    EQU 79
0125      0050  H20    EQU 80
0126      0051  H21    EQU 81
0127      0052  H22    EQU 82
0128      0053  H23    EQU 83
0129      0054  H24    EQU 84
0130      0055  H25    EQU 85
0131      0056  H26    EQU 86
0132      0057  H27    EQU 87
0133      0058  H28    EQU 88
0134      0059  H29    EQU 89
0135      005A  H30    EQU 90
0136      005B  H31    EQU 91
0137      005C  H32    EQU 92
0138      005D  H33    EQU 93
0139      005E  H34    EQU 94
0140      005F  H35    EQU 95
0141      0060  H36    EQU 96
0142      0061  H37    EQU 97
0143      0062  H38    EQU 98
0144      0063  H39    EQU 99
0145      0064  H40    EQU 100
0146      0065  H41    EQU 101
0147      0066  H42    EQU 102
0148      0067  H43    EQU 103
0149      0068  H44    EQU 104
0150      0069  H45    EQU 105
0151      006A  H46    EQU 106
0152      006B  H47    EQU 107
0153      006C  H48    EQU 108
0154      006D  H49    EQU 109
0155      006E  H50    EQU 110
0156      006F  H51    EQU 111
0157      0070  H52    EQU 112
0158      0071  H53    EQU 113
0159      0072  H54    EQU 114
0160      0073  H55    EQU 115
0161      0074  H56    EQU 116
0162      0075  H57    EQU 117
0163      0076  H58    EQU 118
0164      0077  H59    EQU 119
0165            *
0166      0078  MODE   EQU 120
0167      0079  CLOCK  EQU 121
0168      007A  YN     EQU 122
0169      007B  ONE    EQU 123
0170            *
0171 0000       *      AORG 0
```



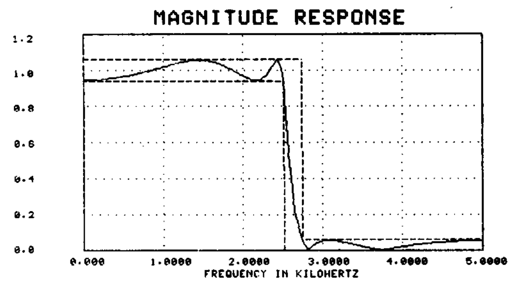
```
0172                 *
0173 0000 F900              B START
     0001 0040
0174                 *
0175                 * COEFFICIENTS ARE INITIALLY *
0176                 * STORED IN PROGRAM MEMORY   *
0177                 *
0178 0002 0030  CH0   DATA  >0030  *  0.146547E-02 *
0179 0003 FFC2  CH1   DATA  >FFC2  * -0.186717E-02 *
0180 0004 0015  CH2   DATA  >0015  *  0.670857E-03 *
0181 0005 FFEF  CH3   DATA  >FFEF  * -0.507893E-03 *
0182 0006 000F  CH4   DATA  >000F  *  0.476907E-03 *
0183 0007 FFF0  CH5   DATA  >FFF0  * -0.482679E-03 *
0184 0008 0010  CH6   DATA  >0010  *  0.505055E-03 *
0185 0009 FFEE  CH7   DATA  >FFEE  * -0.536698E-03 *
0186 000A 0012  CH8   DATA  >0012  *  0.576256E-03 *
0187 000B FFEB  CH9   DATA  >FFEB  * -0.624602E-03 *
0188 000C 0016  CH10  DATA  >0016  *  0.681939E-03 *
0189 000D FFE7  CH11  DATA  >FFE7  * -0.750338E-03 *
0190 000E 001B  CH12  DATA  >001B  *  0.831878E-03 *
0191 000F FFE1  CH13  DATA  >FFE1  * -0.929373E-03 *
0192 0010 0022  CH14  DATA  >0022  *  0.104702E-02 *
0193 0011 FFD8  CH15  DATA  >FFD8  * -0.119041E-02 *
0194 0012 002C  CH16  DATA  >002C  *  0.136731E-02 *
0195 0013 FFCB  CH17  DATA  >FFCB  * -0.158880E-02 *
0196 0014 003D  CH18  DATA  >003D  *  0.187070E-02 *
0197 0015 FFB6  CH19  DATA  >FFB6  * -0.223732E-02 *
0198 0016 0059  CH20  DATA  >0059  *  0.272579E-02 *
0199 0017 FF90  CH21  DATA  >FF90  * -0.339682E-02 *
0200 0018 008E  CH22  DATA  >008E  *  0.435422E-02 *
0201 0019 FF42  CH23  DATA  >FF42  * -0.578642E-02 *
0202 001A 0108  CH24  DATA  >0108  *  0.806880E-02 *
0203 001B FE75  CH25  DATA  >FE75  * -0.120382E-01 *
0204 001C 028B  CH26  DATA  >028B  *  0.198777E-01 *
0205 001D FB04  CH27  DATA  >FB04  * -0.389339E-01 *
0206 001E 0DD6  CH28  DATA  >0DD6  *  0.108105E+00 *
0207 001F 837E  CH29  DATA  >837E  * -0.972714E+00 *
0208 0020 7C81  CH30  DATA  >7C81  * -CH29 *
0209 0021 F229  CH31  DATA  >F229  * -CH28 *
0210 0022 04FB  CH32  DATA  >04FB  * -CH27 *
0211 0023 FD74  CH33  DATA  >FD74  * -CH26 *
0212 0024 018A  CH34  DATA  >018A  * -CH25 *
0213 0025 FEF7  CH35  DATA  >FEF7  * -CH24 *
0214 0026 00BD  CH36  DATA  >00BD  * -CH23 *
0215 0027 FF71  CH37  DATA  >FF71  * -CH22 *
0216 0028 006F  CH38  DATA  >006F  * -CH21 *
0217 0029 FFA6  CH39  DATA  >FFA6  * -CH20 *
0218 002A 0049  CH40  DATA  >0049  * -CH19 *
0219 002B FFC2  CH41  DATA  >FFC2  * -CH18 *
0220 002C 0034  CH42  DATA  >0034  * -CH17 *
0221 002D FFD3  CH43  DATA  >FFD3  * -CH16 *
0222 002E 0027  CH44  DATA  >0027  * -CH15 *
0223 002F FFDD  CH45  DATA  >FFDD  * -CH14 *
0224 0030 001E  CH46  DATA  >001E  * -CH13 *
0225 0031 FFE4  CH47  DATA  >FFE4  * -CH12 *
0226 0032 0018  CH48  DATA  >0018  * -CH11 *
0227 0033 FFE9  CH49  DATA  >FFE9  * -CH10 *
```

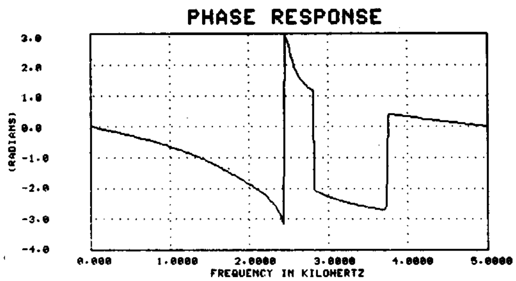
```
0034 0014 CH50   DATA   >0014      * -CH9 *
0035 FFED CH51   DATA   >FFED      * -CH8 *
0036 0011 CH52   DATA   >0011      * -CH7 *
0037 FFEF CH53   DATA   >FFEF      * -CH6 *
0038 000F CH54   DATA   >000F      * -CH5 *
0039 FFF0 CH55   DATA   >FFF0      * -CH4 *
003A 0010 CH56   DATA   >0010      * -CH3 *
003B FFEA CH57   DATA   >FFEA      * -CH2 *
003C 003D CH58   DATA   >003D      * -CH1 *
003D FFCF CH59   DATA   >FFCF      * -CH0 *
          *
003E 000A MD     DATA   >000A
003F 01F3 SMP    DATA   499        * SAMPLING RATE OF 10 KHZ *
          *
0040 6E00 START  LDPK 0
          *
0041 7E01        LACK 1
0042 507B        SACL ONE          * CONTENT OF ONE IS 1 *
          *
0043 7079        LARK AR0,CLOCK    * THIS SECTION OF CODE LOADS  *
0044 713C        LARK AR1,60       * THE FILTER COEFFICIENTS AND *
0045 7E3F        LACK SMP          * OTHER VALUES FROM PROGRAM   *
0046 6880 LOAD   LARP AR0          * MEMORY TO DATA MEMORY       *
0047 6791        TBLR *-,AR1
0048 107B        SUB ONE
0049 F400        BANZ LOAD
004A 0046
          *
004B 4878        OUT MODE,PA0      * INITIALIZATION OF ANALOG *
004C 4979        OUT CLOCK,PA1     * INTERFACE BOARD          *
          *
004D 6880        LARP AR0          * SET ARP TO AR0 *
          *
004E F600 WAIT   BIOZ NXTPT        * BIO PIN GOES LOW WHEN A *
004F 0052
0050 F900        B WAIT            * NEW SAMPLE IS AVAILABLE *
0051 004E
          *
0052 4200 NXTPT  IN XN,PA2         * BRING IN THE NEW SAMPLE XN *
          *
0053 703B        LARK AR0,XNM59    * AR0 POINTS TO THE INPUT SEQUENCE *
0054 7177        LARK AR1,H59      * AR1 POINTS TO THE IMPULSE RESPONSE *
          *
0055 7F89        ZAC
          *
0056 6A91        LT *-,AR1
0057 6D90        MPY *-,AR0
          *
0058 6B81 LOOP   LTD *,AR1
0059 6D90        MPY *-,AR0
          *
005A F400        BANZ LOOP
005B 0058
          *
005C 7F8F        APAC              * ACCUMULATE LAST MULTIPLY *
          *
005D 597A        SACH YN,1
```

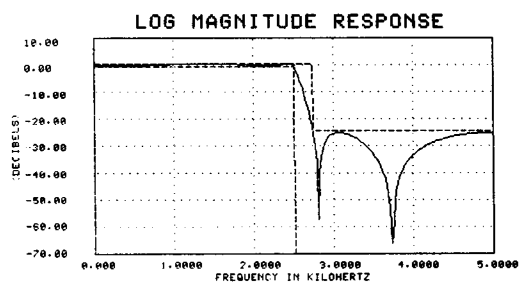


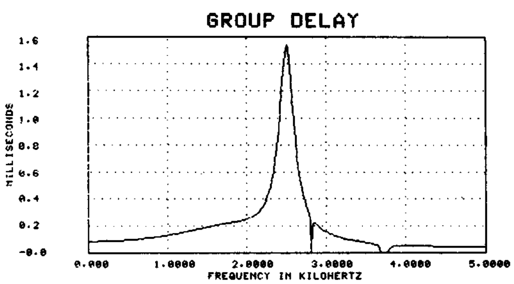
```
          *
005E 4A7A        OUT YN,PA2        * OUTPUT THE FILTER RESPONSE y(n) *
          *
005F F900        B WAIT            * GO GET THE NEXT POINT *
0060 004E
          *
                 END
NO ERRORS, NO WARNINGS
```

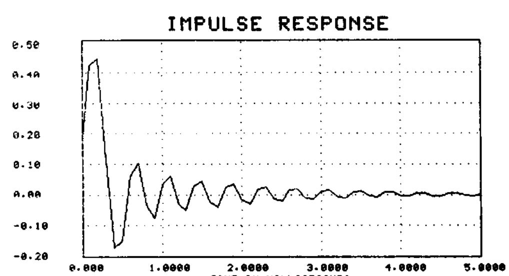
APPENDIX C
FOURTH-ORDER LOWPASS IIR FILTERS











```
          ************************************************************
          *
          *                      FOURTH-ORDER IIR
          *                  ELLIPTIC LOWPASS FILTER
          *
          *                  DIRECT-FORM II STRUCTURE
          *
          *                  FILTER CHARACTERISTICS
          *
          *     SAMPLING FREQUENCY = 10 KHZ
          *
          *                          BAND 1           BAND 2
          *
          *     LOWER BAND EDGE      0.00000          2.75000
          *     UPPER BAND EDGE      2.50000          5.00000
          *     NOMINAL GAIN         1.00000          0.00000
          *     NOMINAL RIPPLE       0.06000          0.06000
          *     MAXIMUM RIPPLE       0.05617          0.05514
          *     RIPPLE IN DB         0.47469        -25.17089
          *
          *
          *                       FILTER STRUCTURE
          *
          *
          *                             b
          *                              0
          *          o--->---o---->----o---->----o--->---o
          *          x(n)    |         |  -1     |       y(n)
          *                  ^    a    v  z    b ^
          *                  |     1   |        1 |
          *                  o----<----o---->----o
          *                  |         |  -1     |
          *                  ^    a    v  z    b ^
          *                  |     2   |        2 |
          *                  o----<----o---->----o
          *                  |         |  -1     |
          *                  ^    a    v  z    b ^
          *                  |     3   |        3 |
          *                  o----<----o---->----o
          *                  |         |  -1     |
          *                  ^    a    v  z    b ^
          *                  |     4   |        4 |
          *                  o----<----o---->----o
          *
          *
          *
          ************************************************************
          *
          *  CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
          *         | (MICROSECONDS) |    (WORDS)     |   (WORDS)
          *  -------+----------------+----------------+------------
          *         |                |                |
          *    24   |      4.8       |       24       |     16
          *         |                |                |
          *  -------------------------------------------------------
          *
          *  (EXCLUDING I/O AND INITIALIZATION)
```



```
                *
                *************************************************************
                *
                       IDT 'IIR4DIR'
     0000       DN     EQU 0
     0001       DNM1   EQU 1
     0002       DNM2   EQU 2
     0003       DNM3   EQU 3
     0004       DNM4   EQU 4
                *
     0005       A1     EQU 5
     0006       A2     EQU 6
     0007       A3     EQU 7
     0008       A4     EQU 8
                *
     0009       B0     EQU 9
     000A       B1     EQU 10
     000B       B2     EQU 11
     000C       B3     EQU 12
     000D       B4     EQU 13
                *
     000E       MODE   EQU 14
     000F       CLOCK  EQU 15
     0010       YN     EQU 16
     0011       XN     EQU 17
     0012       ONE    EQU 18
                *
0000                   AORG 0
                *
0000 F900              B START
0001 000D
                *
                * COEFFICIENTS ARE INITIALLY *
                * STORED IN PROGRAM MEMORY   *
                *
0002 3845       CA1    DATA    >3845   *  0.4396070 *
                *
                *                      * A2 = -1.172416.  THE -1.0 TERM *
0003 E9EE       CA2    DATA    >E9EE   * IS IMPLEMENTED WITH A SUB AND  *
                *                      * A2 CONTAINS -0.172416 = >E9EE. *
                *
0004 3167       CA3    DATA    >3167   *  0.3859772 *
0005 DDC1       CA4    DATA    >DDC1   * -0.2675277 *
                *
0006 17FA       CB0    DATA    >17FA   *  0.1873279 *
0007 2B02       CB1    DATA    >2B02   *  0.3360168 *
0008 3D1F       CB2    DATA    >3D1F   *  0.4775291 *
0009 2AFF       CB3    DATA    >2AFF   *  0.3359135 *
000A 17F3       CB4    DATA    >17F3   *  0.1871291 *
                *
000B 000A       MD     DATA    >000A
000C 01F3       SMP    DATA    499     * SAMPLING RATE OF 10 KHZ *
                *
000D 6E00       START  LDPK 0
                *
000E 7E01              LACK 1
000F 5012              SACL ONE        * CONTENT OF ONE IS 1 *
```

```
0114                 *
0115 0010 700F               LARK AR0,CLOCK  * THIS SECTION OF CODE LOADS  *
0116 0011 710A               LARK AR1,10     * THE FILTER COEFFICIENTS AND *
0117 0012 7E0C               LACK SMP        * OTHER VALUES FROM PROGRAM   *
0118 0013 6880       LOAD    LARP AR0        * MEMORY TO DATA MEMORY       *
0119 0014 6791               TBLR *-,AR1
0120 0015 1012               SUB ONE
0121 0016 F400               BANZ LOAD
     0017 0013
0122                 *
0123 0018 7F89               ZAC             * THIS SECTION SETS THE *
0124 0019 5000               SACL DN         * INITIAL STATE OF THE  *
0125 001A 5001               SACL DNM1       * FILTER TO ZERO        *
0126 001B 5002               SACL DNM2
0127 001C 5003               SACL DNM3
0128 001D 5004               SACL DNM4
0129                 *
0130 001E 480E               OUT MODE,PA0    * INITIALIZATION OF ANALOG *
0131 001F 490F               OUT CLOCK,PA1   * INTERFACE BOARD          *
0132                 *
0133 0020 F600       WAIT    BIOZ NXTPT      * BIO PIN GOES LOW WHEN A *
     0021 0024
0134 0022 F900               B WAIT          * NEW SAMPLE IS AVAILABLE *
     0023 0020
0135                 *
0136 0024 4211       NXTPT   IN XN,PA2       * BRING IN THE NEW SAMPLE XN *
0137                 *
0138                 *                       * IMPLEMENTATION OF SYSTEM POLES *
0139 0025 2F11               LAC XN,15
0140                 *
0141 0026 6A01               LT DNM1
0142 0027 6D05               MPY A1          * d(n-1) * a  *
0143                 *                                    1
0144 0028 6C02               LTA DNM2        * THIS SECTION IS EQUIVALENT TO *
0145 0029 6D06               MPY A2          * -1.172416 * DNM2.  THE -1.0   *
0146 002A 1F02               SUB DNM2,15     * TERM IS IMPLEMENTED WITH THE  *
0147                 *                       * SUB DNM2,15 AND A2 CONTAINS   *
0148                 *                       * -0.172416 = >E9EE.            *
0149                 *
0150 002B 6C03               LTA DNM3
0151 002C 6D07               MPY A3
0152                 *
0153 002D 6C04               LTA DNM4
0154 002E 6D08               MPY A4
0155                 *
0156 002F 7F8F               APAC
0157                 *
0158 0030 5900               SACH DN,1
0159                 *
0160 0031 7F89               ZAC
0161                 *
0162 0032 6D0D               MPY B4          * IMPLEMENTATION OF SYSTEM ZEROES *
0163                 *
0164 0033 6803               LTD DNM3
0165 0034 6D0C               MPY B3          * d(n-3) * b  *
0166                 *                                    3
0167 0035 6802               LTD DNM2
```



```
0168 0036 6D0B               MPY B2
0169                 *
0170 0037 6B01               LTD DNM1
0171 0038 6D0A               MPY B1
0172                 *
0173 0039 6B00               LTD DN
0174 003A 6D09               MPY B0
0175                 *
0176 003B 7F8F               APAC
0177                 *
0178 003C 5910               SACH YN,1       * FINISHED FILTER *
0179                 *
0180 003D 4A10               OUT YN,PA2      * OUTPUT THE FILTER RESPONSE y(n) *
0181                 *
0182 003E F900               B WAIT          * GO GET THE NEXT POINT *
     003F 0020
0183                 *
0184                         END
NO ERRORS, NO WARNINGS
```

```
           **************************************************************
           *
           *                      FOURTH-ORDER IIR
           *                   ELLIPTIC LOWPASS FILTER
           *
           *                   CASCADE STRUCTURE WITH
           *          SECOND-ORDER DIRECT-FORM II SUBSECTIONS
           *
           *                   FILTER CHARACTERISTICS
           *
           *      SAMPLING FREQUENCY = 10 KHZ
           *
           *                            BAND 1          BAND 2
           *
           *      LOWER BAND EDGE       0.00000         2.75000
           *      UPPER BAND EDGE       2.50000         5.00000
           *      NOMINAL GAIN          1.00000         0.00000
           *      NOMINAL RIPPLE        0.06000         0.06000
           *      MAXIMUM RIPPLE        0.05617         0.05514
           *      RIPPLE IN DB          0.47469       -25.17089
           *
           *
           *                      FILTER STRUCTURE
           *
           *
           *                 b      y (n)           b
           *                  01     1               02
           * o--->--o--->---o--->---o--->---o--->---o--->---o--->---o
           *  x(n)  |       |  -1   |       |       |  -1   |    y(n)
           *        ^   a   v  z  b ^       ^   a   v  z  b ^
           *        |   11  |     11|       |   12  |     12|
           *        o---<---o--->---o       o---<---o--->---o
           *        |       |  -1   |       |       |  -1   |
           *        ^   a   v  z  b ^       ^   a   v  z  b ^
           *        |   21  |     21|       |   22  |     22|
           *        o---<---o--->---o       o---<---o--->---o
           *
           *
           **************************************************************
           *
           *  CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
           *         | (MICROSECONDS) |    (WORDS)     |   (WORDS)
           * --------|----------------|----------------|-------------
           *         |                |                |
           *    27   |      5.4       |       27       |      18
           *         |                |                |
           * -----------------------------------------------------------
           *
           *  (EXCLUDING I/O AND INITIALIZATION)
           *
           **************************************************************
           *
                  IDT 'IIR4CAS'
      0000 D2N    EQU 0
      0001 D2NM1  EQU 1
      0002 D2NM2  EQU 2
           *
```



```
      0003 D1N    EQU 3
      0004 D1NM1  EQU 4
      0005 D1NM2  EQU 5
           *
      0006 B01    EQU 6
      0007 B11    EQU 7
      0008 B21    EQU 8
           *
      0009 A11    EQU 9
      000A A21    EQU 10
           *
      000B B02    EQU 11
      000C B12    EQU 12
      000D B22    EQU 13
           *
      000E A12    EQU 14
      000F A22    EQU 15
           *
      0010 MODE   EQU 16
      0011 CLOCK  EQU 17
      0012 YN     EQU 18
      0013 XN     EQU 19
      0014 ONE    EQU 20
           *
0000              AORG 0
           *
0000 F900         B START
0001 000E
           *
           * COEFFICIENTS ARE INITIALLY *
           * STORED IN PROGRAM MEMORY   *
           *
0002 1F05  CB01   DATA    >1F05    *  0.242342 *
0003 2B75  CB11   DATA    >2B75    *  0.339521 *
0004 1EFD  CB21   DATA    >1EFD    *  0.242117 *
           *
0005 394D  CA11   DATA    >394D    *  0.447687 *
0006 D889  CA21   DATA    >D889    * -0.308310 *
           *
0007 62F1  CB02   DATA    >62F1    *  0.772990 *
0008 26DB  CB12   DATA    >26DB    *  0.303581 *
0009 62ED  CB22   DATA    >62ED    *  0.772887 *
           *
000A FEF7  CA12   DATA    >FEF7    * -0.008080 *
000B 90EE  CA22   DATA    >90EE    * -0.867723 *
           *
000C 000A  MD     DATA    >000A
000D 01F3  SMP    DATA    499      * SAMPLING RATE OF 10 KHZ *
           *
000E 6E00  START  LDPK 0
           *
000F 7E01         LACK 1
0010 5014         SACL ONE         * CONTENT OF ONE IS 1 *
           *
0011 7011         LARK AR0,CLOCK   * THIS SECTION OF CODE LOADS  *
0012 710B         LARK AR1,11      * THE FILTER COEFFICIENTS AND *
0013 7E0D         LACK SMP         * OTHER VALUES FROM PROGRAM   *
```

```
0014 6880  LOAD   LARP AR0        * MEMORY TO DATA MEMORY       *
0015 6791         TBLR *-,AR1
0016 1014         SUB ONE
0017 F400         BANZ LOAD
0018 0014
           *
0019 7F89         ZAC             * THIS SECTION SETS THE *
001A 5000         SACL D2N        * INITIAL STATE OF THE  *
001B 5001         SACL D2NM1      * FILTER TO ZERO        *
001C 5002         SACL D2NM2
001D 5003         SACL D1N
001E 5004         SACL D1NM1
001F 5005         SACL D1NM2
           *
0020 4810         OUT MODE,PA0    * INITIALIZATION OF ANALOG *
0021 4911         OUT CLOCK,PA1   * INTERFACE BOARD          *
           *
0022 F600  WAIT   BIOZ NXTPT      * BIO PIN GOES LOW WHEN A *
0023 0026
0024 F900         B WAIT          * NEW SAMPLE IS AVAILABLE *
0025 0022
           *
0026 4213  NXTPT  IN XN,PA2       * BRING IN THE NEW SAMPLE XN *
           *
0027 2F13         LAC XN,15       * START FIRST CASCADE SECTION *
           *
0028 6A04         LT D1NM1
0029 6D09         MPY A11         * d (n-1) * a   *
           *                         1         11
002A 6C05         LTA D1NM2
002B 6D0A         MPY A21
           *
002C 7F8F         APAC
           *
002D 5903         SACH D1N,1
           *
002E 7F89         ZAC
           *
002F 6D08         MPY B21
           *
0030 6B04         LTD D1NM1
0031 6D07         MPY B11
           *
0032 6B03         LTD D1N
0033 6D06         MPY B01         * FINISHED FIRST CASCADE SECTION *
           *
           *                      * START SECOND CASCADE SECTION *
           *
0034 6C01         LTA D2NM1
0035 6D0E         MPY A12         * d (n-1) * a  *
           *                         2         12
0036 6C02         LTA D2NM2
0037 6D0F         MPY A22
           *
0038 7F8F         APAC
           *
0039 5900         SACH D2N,1
```



```
           *
003A 7F89         ZAC
           *
003B 6D0D         MPY B22
           *
003C 6B01         LTD D2NM1
003D 6D0C         MPY B12
           *
003E 6B00         LTD D2N
003F 6D0B         MPY B02
           *
0040 7F8F         APAC
           *
0041 5912         SACH YN,1       * FINISHED SECOND CASCADE SECTION *
           *                      * AND FILTER                      *
           *
0042 4A12         OUT YN,PA2      * OUTPUT THE FILTER RESPONSE y(n) *
           *
0043 F900         B WAIT          * GO GET THE NEXT POINT *
0044 0022
           *
                  END
NO ERRORS, NO WARNINGS
```

```
          **************************************************************
          *
          *                   FOURTH-ORDER IIR
          *                ELLIPTIC LOWPASS FILTER
          *
          *                  PARALLEL STRUCTURE
          *
          *                FILTER CHARACTERISTICS
          *
          *      SAMPLING FREQUENCY = 10 KHZ
          *
          *                          BAND 1          BAND 2
          *
          *      LOWER BAND EDGE     0.00000         2.75000
          *      UPPER BAND EDGE     2.50000         5.00000
          *      NOMINAL GAIN        1.00000         0.00000
          *      NOMINAL RIPPLE      0.06000         0.06000
          *      MAXIMUM RIPPLE      0.05617         0.05514
          *      RIPPLE IN DB        0.47469       -25.17089
          *
          *
          *                  FILTER STRUCTURE
          *
          *                       C
          *         +---------------->----------------+
          *         |                                 |
          *         ^                                 |
          *                               g      p (n) v
          *         |                      01     1   |
          *  o--->---o--->---o---->----o---->----o--->---o--->---o
          *  x(n)    |       |         |  -1     |       |    y(n)
          *         |       ^    a    v z   g   ^       |
          *         |       |     11  |      11 |       |
          *         |       o----<----o---->----o       |
          *         |       |         |  -1             |
          *         |       ^    a    v z               |
          *         v       |     21  |                 ^
          *         |       o----<----o                 |
          *         |                                   |
          *         |                                   |
          *         |                     g      p (n) |
          *         |                      02     2    |
          *         +--->---o---->----o---->----o--->---+
          *                 |         |  -1     |
          *                 ^    a    v z   g   ^
          *                 |     12  |      12 |
          *                 o----<----o---->----o
          *                 |         |  -1
          *                 ^    a    v z
          *                 |     22  |
          *                 o----<----o
          *
          *
          **************************************************************
          *
          *   CYCLES | EXECUTION TIME | PROGRAM MEMORY | DATA MEMORY
          *          | (MICROSECONDS) |    (WORDS)     |   (WORDS)
          *   -------+----------------+----------------+------------
```



```
                  *          |                |                |
                  *     28   |     5.6        |       28       |     18
                  *          |                |                |
                  *   -------------------------------------------------------
                  *
                  *   (EXCLUDING I/O AND INITIALIZATION)
                  *
                  **************************************************************
                  *
                  *
                         IDT 'IIR4PAR'
           0000   D2N    EQU 0
           0001   D2NM1  EQU 1
           0002   D2NM2  EQU 2
           0003   D1N    EQU 3
           0004   D1NM1  EQU 4
           0005   D1NM2  EQU 5
                  *
           0006   G01    EQU 6
           0007   G11    EQU 7
                  *
           0008   A11    EQU 8
           0009   A21    EQU 9
                  *
           000A   G02    EQU 10
           000B   G12    EQU 11
                  *
           000C   A12    EQU 12
           000D   A22    EQU 13
                  *
           000E   C      EQU 14
                  *
           000F   MODE   EQU 15
           0010   CLOCK  EQU 16
           0011   YN     EQU 17
           0012   XN     EQU 18
           0013   ONE    EQU 19
           0014   P1     EQU 20
                  *
      0000               AORG 0
                  *
      0000 F900          B START
      0001 000D
                  *
                  * COEFFICIENTS ARE INITIALLY *
                  * STORED IN PROGRAM MEMORY   *
                  *
                  *
      0002 C750   CG01   DATA >C750       * -0.4428446 *
      0003 5F2C   CG11   DATA >5F2C       *  0.7435591 *
                  *
      0004 394D   CA11   DATA >394D       *  0.447687 *
      0005 D889   CA21   DATA >D889       *  0.308310 *
                  *
      0006 F721   CG02   DATA >F721       * -0.0693031 *
      0007 EFAE   CG12   DATA >EFAE       * -0.1274959 *
                  *
```

```
0008 FEF7  CA12   DATA >FEF7      * -0.008080 *
0009 90EE  CA22   DATA >90EE      * -0.867723 *
           *
000A 5988  CC     DATA >5988      *  0.699476 *
           *
000B 000A  MD     DATA    >000A
000C 01F3  SMP    DATA    >01F3   * SAMPLING RATE OF 10 KHZ *
           *
000D 6E00  START  LDPK 0
           *
000E 7E01         LACK 1
000F 5013         SACL ONE        * CONTENT OF ONE IS 1 *
           *
0010 7010         LARK AR0,CLOCK  * THIS SECTION OF CODE LOADS  *
0011 710A         LARK AR1,10     * THE FILTER COEFFICIENTS AND *
0012 7E0C         LACK SMP        * OTHER VALUES FROM PROGRAM   *
0013 6880  LOAD   LARP AR0        * MEMORY TO DATA MEMORY       *
0014 6791         TBLR *-,AR1
0015 1013         SUB ONE
0016 F400         BANZ LOAD
0017 0013
           *
0018 7F89         ZAC             * THIS SECTION SETS THE *
0019 5000         SACL D2N        * INITIAL STATE OF THE  *
001A 5001         SACL D2NM1      * FILTER TO ZERO        *
001B 5002         SACL D2NM2
001C 5003         SACL D1N
001D 5004         SACL D1NM1
001E 5005         SACL D1NM2
           *
001F 480F         OUT MODE,PA0    * INITIALIZATION OF ANALOG *
0020 4910         OUT CLOCK,PA1   * INTERFACE BOARD          *
           *
0021 F600  WAIT   BIOZ NXTPT      * BIO PIN GOES LOW WHEN A *
0022 0025
0023 F900         B WAIT          * NEW SAMPLE IS AVAILABLE *
0024 0021
           *
0025 4212  NXTPT  IN XN,PA2       * BRING IN THE NEW SAMPLE XN *
           *
0026 2F12         LAC XN,15       * START FIRST PARALLEL SECTION *
           *
0027 6A05         LT D1NM2
0028 6D09         MPY A21         * d (n-2) * a   *
           *                         1         21
0029 6B04         LTD D1NM1
002A 6D08         MPY A11
           *
002B 7F8F         APAC
           *
002C 5903         SACH D1N,1
           *
002D 7F89         ZAC
           *
002E 6D07         MPY G11
           *
002F 6B03         LTD D1N
```



```
0030 6D06         MPY G01
           *
0031 7F8F         APAC
           *
0032 5914         SACH P1,1       * FINISHED FIRST PARALLEL SECTION *
           *
0033 2F12         LAC XN,15       * START SECOND PARALLEL SECTION *
           *
0034 6A02         LT D2NM2
0035 6D0D         MPY A22         * d (n-2) * a   *
           *                         2         22
0036 6B01         LTD D2NM1
0037 6D0C         MPY A12
           *
0038 7F8F         APAC
           *
0039 5900         SACH D2N,1
           *
003A 2F14         LAC P1,15
           *
003B 6D0B         MPY G12
           *
003C 6B00         LTD D2N
003D 6D0A         MPY G02
           *
003E 6C0E         LTA C
003F 6D12         MPY XN
           *
0040 7F8F         APAC
           *
0041 5911         SACH YN,1       * FINISHED SECOND PARALLEL SECTION *
           *                      * AND FINISHED FILTER              *
           *
0042 4A11         OUT YN,PA2      * OUTPUT THE FILTER RESPONSE y(n) *
           *
0043 F900         B WAIT          * GO GET THE NEXT POINT *
0044 0021
           *
                  END
NO ERRORS, NO WARNINGS
```

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall (1975).
2. Andreas Antoniou, *Digital Filters: Analysis and Design*, McGraw-Hill (1979).
3. C.S. Burrus and T.W. Parks, *Digital Filter Design*, John Wiley & Sons (1986).
4. U. Kaiser, "Wave Digital Filters and Their Significance for Customized Digital Signal Processing," *Texas Instruments Engineering Journal*, **Vol 2**, No. 5, 29-44 (September - October 1985).
5. *TMS32010 User's Guide* (SPRU001B), Texas Instruments (1985).
6. *TMS32020 User's Guide* (SPRU004A), Texas Instruments (1985).
7. *Digital Filter Design Package (DFDP)*, Atlanta Signal Processors Inc. (ASPI), 770 Spring St. NW, Suite 208, Atlanta, GA 30308, 404/892-7265 (1984).
8. *TMS32010 Development Support Reference Guide* (SPRU007), Texas Instruments (1984).