Application Report
SPRA610 - December 1999

A Block Floating Point Implementation on the TMS320C54x DSP

Arun Chhabra and Ramesh Iyer — *Digital Signal Processing Solutions*

ABSTRACT

Block floating-point (BFP) implementation provides an innovative method of floating-point emulation. This application report implements the BFP algorithm for the Fast Fourier Transform (FFT) algorithm on a Texas Instruments (TI™) TMS320C54x DSP by taking advantage of the exponent encoder and normalization units on the DSP. The BFP algorithm as it applies to the FFT allows fractional signal gain adjustment in a fixed-point environment by using a block representation of input values of block size N to an N-point FFT. This algorithm is applied repetitively to all stages of the FFT. The elements within a block are further represented by their respective mantissas and a common exponent assigned to the block. This method allows for aggressive scaling with a single exponent while retaining greater dynamic range in the output. This application report discusses the BFP FFT and demonstrates its implementation in assembly language. The implementation is carried out on a fixed-point digital signal processor (DSP). The fixed-point BFP FFT results are contrasted with the results of a floating-point FFT of the same size implemented with MATLAB. For applications where the FFT is a core component of the overall algorithm, the BFP FFT can provide results approaching floating-point dynamic range on a low-cost fixed-point processor. Most DSP applications can be handled with fixed-point representation. However, for those applications that require extended dynamic range but do not warrant the cost of a floating-point chip, a block floating-point implementation on a fixed-point chip readily provides a cost-effective solution.

Contents

List of Figures

List of Tables

1 Fixed- and Floating-Point Representations

Fixed-point processors represent numbers either in fractional notation – used mostly in signal processing algorithms, or integer notation – primarily for control operations, address calculations and other non-signal processing operations. Clearly the term fixed-point representation is not synonymous with integer notation. In addition, the choice of fractional notation in digital signal processing algorithms is crucial to the implementation of a successful scaling strategy for fixed-point processors.

Integer representation encompasses numbers from zero to the largest whole number that can be represented using the available number of bits. Numbers can be represented in two's complement form with the most significant bit as the sign bit that is negatively weighted.

Fractional format is used to represent numbers between –1 and 1. A binary radix point is assumed to exist immediately after the sign bit that is also negatively weighted. For the purpose of this application report, the term fixed-point will imply use of the fractional notation.

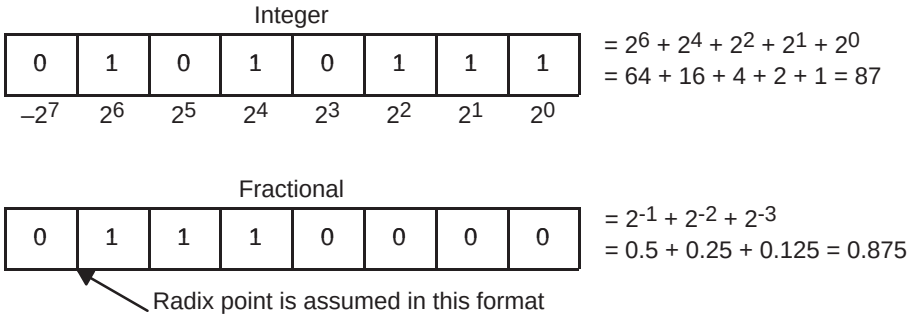
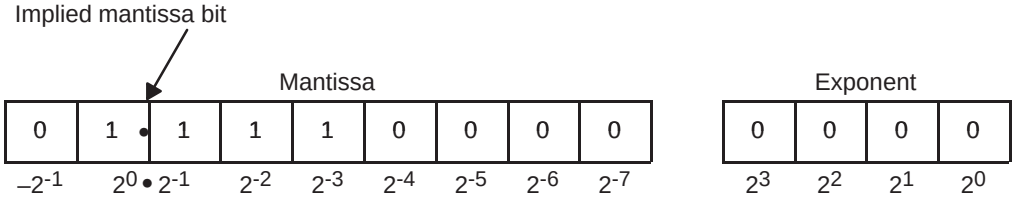


Figure 1. Diagram of Fixed-Point Representations – Integer and Fractional

Floating-point arithmetic consists of representing a number by way of two components – a mantissa and an exponent. The mantissa is generally a fractional value that can be viewed to be similar to the fixed-point component. The exponent is an integer that represents the number of places that the binary point of the mantissa must be shifted in either direction to obtain the original number. In floating point numbers, the binary point comes after the second most significant bit in the mantissa.



Mantissa = $2^0 + 2^{-1} + 2^{-2} + 2^{-3} = 1 + 0.5 + 0.25 + 0.125 = 1.875$

Exponent = $2^2 + 2^0 = 4 + 1 = 5$

Decimal value = $1.875 \times 2^5 = 60$

Figure 2. Diagram of Floating-Point Representation

2 Precision, Dynamic Range and Quantization Effects

Two primary means to gauge the performance of fixed-point and floating-point representations are dynamic range and precision.

Precision defines the resolution of a signal representation; it can be measured by the size of the least significant bit (LSB) of the fraction. In other words, the word-length of the fixed-point format governs precision. For floating-point format, the number of bits that make up the mantissa give the precision with which a number can be represented. Thus, for the floating-point case, precision would be the minimum difference between two numbers with a given common exponent. An added advantage of the floating-point processors is that the hardware automatically scales numbers to use the full range of the mantissa. If the number becomes too large for the available mantissa, the hardware scales it down by shifting it right. If the number consumes less space than the available word-length, the hardware scales it up by shifting it left. The exponent tracks the number of these shifts in either direction.

The dynamic range of a processor is the ratio between the smallest and largest number that can be represented. The dynamic range for a floating-point value is clearly determined by the size of the exponent. As a result, given the same word-length, a floating-point processor will always have a greater dynamic range than a fixed-point processor. On the other hand, given the same word-length, a fixed-point processor will always have greater precision than floating-point processors.

Quantization error also serves as a parameter by which the difference between fixed-point and floating-point representations can be measured. Quantization error is directly dependent on the size of the LSB. As the number of quantization levels increases, the difference between the original analog waveform and its quantized digital equivalent becomes less. As a result, the quantization error also decreases, thereby lowering the quantization noise. It is clear then that the quantization effect is directly dependent on the word-length of a given representation.

The increased dynamic range of a floating-point processor does come at a price. While providing increased dynamic range, floating-point processors also tend to cost more and dissipate more power than fixed-point processors, as more logic gates are required to implement floating-point operations.

3 The Block Floating Point Concept

At this point it is clear that fixed and floating-point implementations have their respective advantages. It is possible to achieve the dynamic range approaching that of floating-point arithmetic while working with fixed-point processors. This can be accomplished by using floating-point emulation software routines. Emulating floating-point behaviour on a fixed-point processor tends to be very cycle intensive, since the emulation routine must manipulate all arithmetic computations to artificially mimic floating-point math on a fixed-point device. This software emulation is only worthwhile if a small portion of the overall computation requires extended dynamic range. Clearly, a cost-effective alternative for floating-point dynamic range implemented on a fixed-point processor is needed.

The block floating point algorithm is based on the block automatic gain control (AGC) concept. Block AGC only scales values at the input stage of the FFT. It only adjusts the input signal power. The block floating point algorithm takes it a step further by tracking the signal strength from stage to stage to provide a more comprehensive scaling strategy and extended dynamic range.

The floating-point emulation scheme discussed here is the block floating-point algorithm. The primary benefit of the block floating-point algorithm emanates from the fact that operations are carried out on a block basis using a common exponent. Here, each value in the block can be expressed in two components – a mantissa and a common exponent. The common exponent is stored as a separate data word. This results in a minimum hardware implementation compared to that of a conventional floating-point implementation.

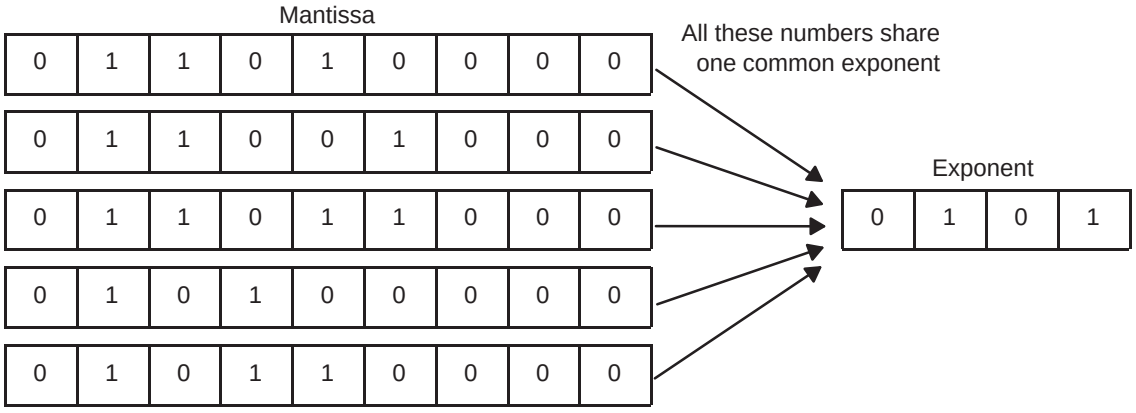


Figure 3. Diagram of Block Floating-Point Representation

The value of the common exponent is determined by the data element in the block with the largest amplitude. In order to compute the value of the exponent, the number of leading bits has to be determined. This is determined by the number of left shifts required for this data element to be normalized to the dynamic range of the processor. Certain DSP processors have specific instructions, such as exponent detection and normalization instructions, that perform this task. If a given block of data consists entirely of small values, a large common exponent can be used to shift the small data values left and provide more dynamic range. On the other hand, if a data block contains large data values, then a small common exponent will be applied. Whatever the case may be, once the common exponent is computed, all data elements in the block are shifted up by that amount, in order to make optimal use of the available dynamic range. The exponent computation does not consider the most significant bit, since that is reserved for the sign bit and is not considered to be part of the dynamic range.

As a result, block floating-point representation does provide an advantage over both, fixed and floating-point formats. Scaling each value up by the common exponent increases the dynamic range of data elements in comparison to that of a fixed-point implementation. At the same time, having a separate common exponent for all data values preserves the precision of a fixed-point processor. Therefore, the block floating-point algorithm is more economical than a conventional floating-point implementation.

4 The Block Floating Point for a Complex FFT

The block floating-point analysis presented here is based on its application to a 64-point complex decimation-in-time (DIT) Fast Fourier Transform (FFT). The assembly code that implements this FFT will be referred to as the “original code” through this application report. Block floating-point scaling is implemented by determining the input-scaling factor for each butterfly stage based on the actual bit growth of the previous stage. This can be implemented in a number of ways. One technique is as explained above by computing the number of leading bits and normalizing the whole array of input values by that amount. At the end of the computations of each stage, the output values are scaled down by the required amount such that they do not lead to overflow when used as the input to the next stage. This process will repeat itself so that maximum possible dynamic range is maintained while averting the possibility of an overflow. The original code only employs binary scaling, i.e., the output of every radix-2 stage is automatically scaled down by 2. It is also possible to perform non-binary scaling with BFP that allows for fractional gain adjustments. This application report employs both of the above scaling techniques.

The first stage of our FFT implementation is a radix-4 butterfly for code and execution speed optimization. The unique characteristic of this stage is that the magnitude growth can be no more than a factor of 4, since the value of theta (θ) can only be in increments of $\pi/2$. As a result, the scaling factor for that input array was chosen to be ¼, or a right shift of two bit places.

The subsequent stages of our FFT are radix-2 butterflies. The maximum theoretical magnitude growth possible for a general radix-2 FFT butterfly is a factor of 2.414. Given that a radix-2 butterfly can be expressed as $A' = A + (B*W)$, where all values are complex and W is the twiddle factor, we note that W will reach its maximum magnitude at $\pi/4$. Given this, it is obvious that A' will attain its maximum possible value when $A = 1+ j0$; $B = 1 – j1$; $W = 0.707 + j0.707$. This results in a maximum gain of 2.41421356. Thus the scaling based on this signal growth factor will be $1/2.414 \approx 0.4167$.

5 Implementing the Block Floating Point – Approaches Taken

Three different approaches were adopted in implementing the block floating-point concept for a 64-point complex FFT. The input to approaches I and II was a rail-to-rail complex random noise generated with MATLAB. Approach III uses the same complex random noise but scaled down by a factor of 2.

In the first approach, prior to processing any input values to the first radix-4 butterfly stage, all values are scaled up such that the signal occupies the entire dynamic range of the processor. This radix-4 stage is scaled by a factor of 4 to prevent overflow. Each subsequent radix-2 stage is automatically scaled by a factor of 2 in the original FFT code.

In the second approach, input values to the radix-4 stage are scaled up such that the signal occupies the entire dynamic range of the processor. In addition, scaling is done on a conditional basis in the block floating-point code when the maximum input value to each butterfly stage crosses a pre-determined threshold value. As mentioned before, the output of every radix-2 stage in the original FFT code is automatically scaled down by a factor of 2. Since the theoretical maximum growth for a radix-2 stage is 2.414, every radix-2 stage must be scaled down by this amount. To offset the original automatic scaling by 2, an effective scaling factor of (2/2.414) is introduced.

The third approach draws on the benefits of approaches I and II. Similar to approach I, the input set of values is scaled up to fully occupy the processor dynamic range. Drawing from approach II, the automatic scaling feature from the original fixed-point FFT code is disabled. This approach was an attempt to view the impact of an input signal of smaller amplitude on the results. The input used in this approach is a signal that is identical to the input signal of the previous steps in frequency distribution. However, in this case the amplitude of each of the frequency bins is half that of the previous approaches. Scaling is carried out similar to approach two.

6 Analysis of Results

The results of the block floating-point FFT are compared against two known good result sets – those of the floating-point MATLAB environment and of the pure fixed-point DSP environment.

The primary methods adopted to analyze the results of the block floating-point implementation are quantization error and signal-to-noise (SNR). The quantization error is a suitable study of the results since it compares the corresponding values between two types of signals. Calculating the total noise power for a given pair of signals results in the quantization error. Given two complex signals, A and B (where signal A is the reference signal against which comparison is carried out and signal B is the signal whose performance is under test), the total noise power computation for these two signals is found by

$$\sum\left(R_A - R_B\right)^2 + \sum\left(\mathrm{Im}_A - \mathrm{Im}_B\right)^2 \tag{1}$$

where ‘R’ and ‘Im’ denote real and imaginary quantities, respectively. This is the quantization error. Similarly, the total signal power can be computed by

$$\sum\left(R_A\right)^2 + \sum\left(\mathrm{Im}_A\right)^2 \tag{2}$$

The quantization error is computed for the signal under test (signal B) with respect to a signal considered to be the reference for comparison (signal A). As a result, in this analysis, the total signal power is always calculated with the reference signal – signal A in the equations above.

Armed with this knowledge, the computation for the SNR becomes relatively simple. It is the ratio of the total signal power to the total noise power. Using the equations above

$$SNR = \frac{\sum\left(R_A\right)^2 + \sum\left(\mathrm{Im}_A\right)^2}{\sum\left(R_A - R_B\right)^2 + \sum\left(\mathrm{Im}_A - \mathrm{Im}_B\right)^2} \tag{3}$$

Or in dB the SNR can be expressed as (10 log $SNR_{\text{power ratio}}$).

Provided that the block floating-point algorithm works as intended, its SNR results should be an improvement over the SNR for fixed-point results. The SNR in both these cases will be computed relative to the signal power for the reference MATLAB implementation. In addition, block floating-point will also promise an improved quantization error result when compared to that of the fixed-point implementation.

The results of the three different approaches are highlighted in the summary table of results below. It is clear that modifications made in the implementation of each case produced results that were better than those of the previous cases.

Table 1. Summary of Results

	Approach	SNR (dB)	Quantization Error Power	Total Signal Power
I				
	BFP	55.62	1.5893e–7	
	Fixed Pt FFT	53.35	2.6819e–7	0.0580
			41% improvement	
II				
	BFP	56.25	1.3738e–7	
	Fixed Pt FFT	53.35	2.6819e–7	0.0580
			49% improvement	
III				
	BFP	51.6	1.0056e–7	
	Fixed Pt FFT	47.9	2.3397e–7	0.0145
			58% improvement	

The results in Table 1 shows that the block floating-point implementation of approach I provides a 41% improvement over the fixed-point FFT implementation. This is a benefit arrived at by scaling up all values in the input such that they fully occupy the dynamic range available. As hypothesized, this case indicates a good result, since shifting all values to the left allows for increased precision.

The block floating-point implementation of approach II produces a 49% improvement when compared to the fixed-point FFT implementation. Recall that this technique is based on disabling the automatic scaling in butterfly stages that was present in the previous approach. Since bit growth of values between butterfly stages is a possibility but not a necessity, scaling down automatically during each stage can compromise the precision of results. Scaling down only makes sense if there is evidence of bit growth during butterfly computations. However, when bit growth is not expected, the results of that stage can be directly fed to the next stage while yielding full benefit of the available dynamic range. Automatically scaling down in this last instance would use less of the full dynamic range. In this manner, optimal scaling from stage-to-stage is used to prevent overflow while at the same time accuracy of the overall system is improved.

It is clear from the result table that approach III outputs the best quantization error amongst the three cases that were investigated. It is important to keep in mind that the reduced signal amplitude in the third approach will lead to a lower total signal power value. As a result, the SNR from this approach will not appear as high as the other cases. Thus, this approach is best suited for cases where a smaller quantization error is of primary interest, without concern for the slightly reduced SNR.

7 Conclusion

The benefits of the block floating point algorithm are apparent. From the results of our experiments, it is clear that the block floating-point implementation produces improved quantization error over the fixed-point implementation. It is important to note that the results in Table 1 reflect our interest to compare the relative performance between the fixed and block floating-point approaches. The test code used to produce these results is not optimized for SNR figures.

The separate common exponent is the key characteristic of the block floating point implementation. It increases the dynamic range of data elements of a fixed-point implementation by providing a dynamic range similar to that of a floating-point implementation. By using a separate memory word for the common exponent, the precision of the mantissa quantities is preserved as that of a fixed-point processor. By the same token, the block floating point algorithm is more economical than a conventional floating-point implementation.

The majority of applications are best suited for fixed-point processors. For those that require extended dynamic range but do not warrant the cost of a floating-point chip implementation, the block floating point implementation on a fixed-point chip readily provides a cost-effective solution.

8 Reference

1. Characteristics of DSP Processors, *Buyer's Guide to DSP Processors*, Berkeley Design Technology, Inc., 1994, pp. 33–41.
2. *Introduction to DSP – DSP processors: Data Formats*, Bores Signal Processing, http://www.bores.com/courses/intro/chips/6data.htm

Appendix A

Source code for the block floating-point implementation consists of the following files:

cfft64.asm	cfft_t.c	cbrev.asm	test.c	macros.asm
test.h	tms320.h	sintab.q15	dsplib.h	cfft.cmd

Cfft64.asm is the only file that has been modified from its original version present in the DSPLIB. These modifications were carried out in order to incorporate the changes necessary to implement the block floating point technique.

All the above named files are listed here in their entirety.

A.1 *Cfft64.asm*

```
;*****************************************************************
;  Function:  cfft64
;  Version :  1.00
;  Description:  complex FFT
;
;  Copyright Texas instruments Inc, 1998
;----------------------------------------------------------------
; Revision History:
;
; 0.00      M. Christ. Original code
;
; 0.01      M. Chishtie.12/96.
;- Improved radix-2 bfly code form 9 cycles to 8.
;- Combined bit-reversal in to COMBO5XX macro to save cycles.
;- Improved STAGE3 macro to 31 cycles
;
; 1.00Beta  R. Piedra, 8/31/98.
;- C-callable version.
;- Removed bit-reversing and made it a separate optional function
;  that also support in-place bit-reversing. In this way the FFT can
;  be computed 100% in-place (memory savings)
;- Modifed STAGE3 macro to correct functional problem
;- Modified order of xmem, ymem operands in butterfly code
;  to reduce number of cycles
;
; 1.00      A. Aboagye 10/15/98
; - added scale option as a parameter
;
; 1.00BFP   A. Chhabra 11/09/99
;- incorporated Block Floating Point concept
;- adjustable scaling factor dictated by "cmprval_2"
;
;*****************************************************************

N .set     64   ; NUMBER OF POINTS FOR FFT

 .include "macros.asm"
```

```
 .include “sintab.q15”

        .mmregs

; Far–mode adjustment
; ––––––––––––––––––

 .if __far_mode
offse  .set 1    ; far mode uses one extra location for ret addr ll
 .else
offset .set 0
 .endif

 .asg (0), DATA
 .asg (1), SIN45
 .asg (2), save_ar7  ; stack description
 .asg (3), save_ar6  ; stack description
 .asg (4), save_ar1
 .asg (5), ret_addr
 .asg (6+offset), scale
      ; x in A


;***********************************************************************
;Setting the bit growth test value.
;Depending on the bit growth quantity desired for implementation,
;include an appropriate “cmprval_2” value.
;***********************************************************************

;cmprval_2:    .equ 32768*5/10  ; 0.5 decimal
cmprval_2: .equ 32768*8284/10000 ; 0.8284 decimal


;***********************************************************************

  .ref InvYeTable
  .def _cfft64

  .text
_cfft64

; Preserve registers
; ––––––––––––––––––
 pshm ar1
 pshm ar6
 pshm ar7

; Preserve local variables
; ––––––––––––––––––––––––
 frame –2
 nop

; Get Arguments
; –––––––––––––
```

```
 stl a,*sp(DATA) ; DATA = *SP(DATA)

  .if   N>4  ;  ??? no need
 st    #5a82h,*sp(SIN45)
.endif

; Set modes
; ---------
 stm  #0100010101011110b,ST1 ; ASM=-2 , FRACT=1, sbx=1;CPL=1(compiler)
 stm  #0, *ar5               ; initialize the contents of AR5

; Execute
; -------

*********  Modifications by AC 03/25/99   **********
 mvdk *sp(DATA), ar1 ; Transfer first value of input buffer – which
                     ; currently
       ; contain the inputs to the first stage butterflies
       ; – into AR1.
       ; Further manipulation of input set can be done
       ; by addressing AR1

 call max_abs

 exp a               ; determine the scale up shift quantity

 stm #127, brc       ; scale for whole input array
 stm #0800h, ar1     ; Reset pointer to beginning of input array

 rptb  end_upscale-1  ; begin loop
 ld *ar1, a
 norm a              ; Scale up all input values
 sth a, *ar1+        ; Put rescaled value back into memory;
                     ; increment counter to shift next value
end_upscale:
 nop

 combo5xx            ; FFT CODE for STAGES 1 and 2


 stm #0800h, ar1     ; Reset pointer to beginning of input array
 call max_abs

 ld #cmprval_2, b    ; load threshold 0.4167 into Acc B
 max a               ; Acc A will contain the larger of the earlier
                     ; MaxAbs value and the current threshold value
                     ; of 0.4167
 sub #cmprval_2,a,b  ; If diff > 0, thenAcc A = maxabs value
       ; GOTO scaling_2
       ; If diff = 0, then Acc A = threshold value
       ; all values in input array are less than this...
       ; GOTO performing regular next stage of bfly

 .if cmprval_2 = (32768*8284/10000)
```

```
 cc scaling_2, bgt    ; perform scaling if MaxAbs > cmprval_2

 .else

 stm #127, brc        ; scale for whole input array
 stm #0800h, ar1      ; Reset pointer to beginning of input array

 bc loop1, beq        ; execute next 1 instruction if diff>0.

 rptb  loop1–1        ; begin loop
 ld *ar1, a
 sfta a, –1           ; shift down by factor 2
 stl a, *ar1+         ; restore new value into ar1

loop1: .endif

    stage3            ; MACRO WITH CODE FOR STAGE 3


 stm #0800h, ar1      ; Reset pointer to beginning of input array
 call max_abs

 ld #cmprval_2, b     ; load threshold cmprval_2 into Acc B
 max a                ; Acc A will contain the larger of the earlier
                      ; MaxAbs value and the current threshold value
                      ; of cmprval_2
 sub #cmprval_2,a,b   ; If diff > 0, then Acc A = maxabs value
       ; GOTO scaling_2
       ; If diff = 0, then Acc A = threshold value
       ; GOTO performing regular next stage of bfly

 .if cmprval_2 = (32768*8284/10000)

 cc scaling_2, bgt    ; perform scaling if MaxAbs > cmprval_2

 .else

 stm #127, brc        ; scale for whole input array
 stm #0800h, ar1      ; Reset pointer to beginning of input array

bc loop2, beq         ; execute next 1 instruction if diff>0.

 rptb  loop2–1        ; begin loop
 ld *ar1, a
 sfta a, –1           ; shift down by factor 2
 stl a, *ar1+         ; restore new value into ar1

loop2: .endif

 stdmacro 4,4,8,16,sin4,cos4  ; stage,outloopcnter,loopcnter,index

 stm #0800h, ar1      ; Reset pointer to beginning of input array
 call max_abs
```

```
 ld #cmprval_2, b     ; load threshold cmprval_2 into Acc B
 max a                ; Acc A will contain the larger of the earlier
                      ; MaxAbs value and the current threshold value
                      ; of cmprval_2

 sub #cmprval_2,a,b  ; If diff > 0, then  Acc A = maxabs value
        ; GOTO scaling_2
        ; If diff = 0, then Acc A = threshold value
        ; GOTO performing regular next stage of bfly

 .if cmprval_2 = (32768*8284/10000)

 cc scaling_2, bgt    ; perform scaling if MaxAbs > cmprval_2

 .else

 stm #127, brc        ; scale for whole input array
 stm #0800h, ar1      ; Reset pointer to beginning of input array

 bc loop3, beq        ; execute next 1 instruction if diff>0.

 rptb  loop3–1        ; begin loop
 ld *ar1, a
 sfta a, –1           ; shift down by factor 2
 stl a, *ar1+         ; restore new value into ar1

loop3: .endif

 stdmacro 5,2,16,32,sin5,cos5   ; stage,outloopcnter,loopcnter,index


 stm #0800h, ar1      ; Reset pointer to beginning of input array
 call max_abs

 ld #cmprval_2, b     ; load threshold cmprval_2 into Acc B
 max a                ; Acc A will contain the larger of the earlier
                      ; MaxAbs value and the current threshold value
                      ; of cmprval_2
 sub #cmprval_2,a,b   ; If diff > 0, then Acc A = maxabs value
        ; GOTO scaling_2
        ; If diff = 0, then Acc A = threshold value
        ; GOTO performing regular next stage of bfly

 .if cmprval_2 = (32768*8284/10000)

 cc scaling_2, bgt    ; perform scaling if MaxAbs > cmprval_2

 .else

 stm #127, brc        ; scale for whole input array
 stm #0800h, ar1      ; Reset pointer to beginning of input array

 bc loop4, beq        ; execute next 1 instruction if diff>0.

 rptb  loop4–1        ; begin loop
```

```
 ld *ar1, a
 sfta a, –1           ; shift down by factor 2
 stl a, *ar1+         ; restore new value into ar1

loop4:  .endif

 laststag 6,sin6,cos6       ; MACRO WITH CODE FOR STAGE 7

 bd end_lab
 nop
 nop

*********************************************************************
;MAX_ABS
;=======
;
;Perform comparison of consecutive values in order to obtain maximum
;absolute value in the array of inputs.  Steps to do this:
;
;(i)  Place consecutive values in acc A and B respectively
;(ii) Compute their absolute values
;(iii) Find the MAX of these two accumulators values
;(iv) Monitor the Carry bit and determine which acc contains MAX value.
;(v)  Store the max value in acc A
;(vi) Take in the next value in the input array and load into acc B.
;  Compute its absolute.
;(vii) Go back to step (ii)
;
;*** Steps (iv) and (v) above are performed in combination
;    as a result of the C54x “MAX” instruction.
*********************************************************************

max_abs:

    ; Set breakpoint to verify that AR1 does point to correct address

 ld *ar1+, a
    ld *ar1+, b
    absa        ; setup absolute value for max comparison
    absb

    stm #126, brc   ; 126 values remain to be read of the 128 value
                    ; input array.  The loop executes 127 times.
 rptb  find_max–1

 max a
 ld *ar1+, b   ; enter the next value in the input array into acc B
 abs b         ; setup next value for absolute max comparison
find_max
 ret           ; returns the maximum absolute value in the
               ; Acc A

********************** END of “MAX_ABS” routine ********************
```

```
*************************************************************************
;
;SCALING
;=======
;
;This routine performs the following in order:
;
;(i)  Now, since reaching this routine implies that bit growth is
;   likely at the output of this stage, scale the mantissa values.
;   Scaling factor is determined by the reciprocal of the expected
;   bit growth.
;   Expected bit growth = 2.4
;   Reciprocal = 0.4167
;   If maximum of input block to a stage is > 0.4167, then scale
;   down once by shifting right.
;   If maximum of input block is twice > 0.4167, then scale down
;   by shifting right twice.
;   Else ignore scaling and proceed as normal.
;
;Arriving at this routine implies that Acc A already contains
;the maximum absolute value of the input array.
;
*************************************************************************

scaling_2:              ; Acc A contains the MaxAbs value
 pshm ar1
 pshm ar2
 ssbx SXM
 rsbx FRCT
;rsbx TC   ; this is ID for div–by–1.207 selection
           ; in the reciprocal routine

 call reciprocal  ; Acc A contains MaxAbs

 ; COMPUTE SCALING FACTOR = (1/(MaxAbs*2.41421356))
 ; now, remember that the original DSPLIB FFT code contains
 ; automatic scaling down within each radix–2 stage
 ; by a factor of 2, i.e. one bit place.

 ; As a result, what we manually need to tweak with for
 ; scaling is a value = (2.41421356/2) = 1.207

 ; => Scaling factor needed here = (1/(MaxAbs * 1.207))

 ld a, b      ; temporarily store Acc A into B
 add *ar5, a  ; add scaling factor to Acc A
 stl a, *ar5  ; save the scaling factor into AR5

 stm #0800h, ar1  ; re–align to the beginning of the array
 stl a, *ar2
 ld *ar2, t
 stm #127, brc

 rptb end_scale2–1
```

```
 mpy *ar1, a
 sth a, *ar1+

end_scale2:

 popm ar2
 popm ar1

 ret
************************End of “SCALING” routine ********************


*************************************************************************
;
;RECIPROCAL
;==========
;
;This routine will perform the following:
;
;   1) Take the value in Acc A and compute its reciprocal
;2) The result is available in two portions, r and rexp.
;3) MPY r and rexp. however this may lead to a value greater
;  than 32768.
;4) Now MPY (3) by reciprocal of 4, i.e. by 0.25.  This is also
;  equivalent to right shifting by 2 bit locations.
;5) Since (3) will likely lead to a value greater than 32768, it
;  is probably better to perform (4) on value “r” from (2).
;  This way the value is decreased by a factor of 4.
;  Now, we can mutiply by rexp without exceeding 32768.
;
;NOTE: On entering this routine, the Acc A contains the MaxAbs value
;  on which reciprocal computation has to be performed.
;
*************************************************************************

reciprocal:

;-----------------------------------------------------------------
; Set offsets to local function variables defined on stack
;-----------------------------------------------------------------

 .asg 0, SP_INVYETABLE
 .asg 1, SP_XNORM
 .asg 2, SP_TEMP
 .asg 3, FRAME_SZ

;-----------------------------------------------------------------
; Assign registers to local variables
;-----------------------------------------------------------------
 .asg ar0, AR_X
 .asg ar1, AR_Z
;.asg brc, AR_N
 .asg ar3, AR_ZEXP
```

```
 .asg ar4, AR_TABLE

;----------------------------------------------------------------
; Process command–line arguments
;----------------------------------------------------------------
; stl a,*(AR_X) ; Acc A contains MaxAbs value

 pshm ar0
 pshm ar1
 pshm ar3
 pshm ar4
 pshm ar5

;----------------------------------------------------------------
; Initialize constants
;----------------------------------------------------------------

 st #InvYeTable,*sp(SP_INVYETABLE)
 ssbx  OVM   ; 1 cycle, MUST turn overflow mode on.
 stm #0040h, AR_X
 stl a, *AR_X
 ld *AR_X,16,a

;Acc A contains the MaxAbs value...so just start performing operation
 exp a   ; 1 cycle,  delay – slot

 nop     ; 1 cycle
 nop     ; 1 cycle
 norm a  ; 1 cycle

 st t,*sp(SP_TEMP)   ; store exponent computed by EXP instruction
earlier
 ld #InvYeTable,b    ; 2 cycles
 add *sp(SP_TEMP),b  ; 1 cycle
 stl b,*(AR_TABLE)   ; 1 cycle
 sth a,*sp(SP_XNORM) ; 1 cycle, AR2 points to appropriate Ye value in
table.
 sfta a,–1           ; 1 cycle, Estimate the first Ym value.
 xor #01FFFh,16,a    ; 2 cycles
 sth a,*AR_Z         ; store result in auxiliary register

;----------------------------------------------------------------
; First two iterations:
;----------------------------------------------------------------

.loop 2
 ld  *AR_Z,15,a      ; 2 cycles, Calculate Ym = 2*Ym – Ym^2*X
 ld  *AR_Z,t         ; 1 cycle
 mpy *sp(SP_XNORM),b ; 1 cycle
 sth b,1,*AR_Z       ; 2 cycles
 mpy *AR_Z,b         ; 1 cycle
 sub b,1,a           ; 1 cycle
 sth a,2,*AR_Z       ; 2 cycles
 .endloop
```

```
;-----------------------------------------------------------------
; Final iteration: – this code is same as above loop, except
; last instruction omitted
;-----------------------------------------------------------------

 ld  *AR_Z,15,a      ; 2 cycles, Calculate Ym = 2*Ym – Ym^2*X
 ld  *AR_Z,t         ; 1 cycle
 mpy  *sp(SP_XNORM),b  ; 1 cycle
 sth  b,1,*AR_Z      ; 2 cycles
 mpy  *AR_Z,b        ; 1 cycle
 sub  b,1,a          ; 1 cycle

 st #07000h,*AR_Z    ; 2 cycles, Make sure that 8000h <= Ym < 7FFFh
 add *AR_Z,16,a      ; 1 cycle
 sub *AR_Z,16,a      ; 1 cycle
 sub *AR_Z,16,a      ; 1 cycle
 add *AR_Z,16,a      ; 1 cycle
 sth a,3,*AR_Z       ; 2 cycles

 ld *AR_TABLE, t     ; setup for MPY
 bc div1207, ntc     ; if TC=0, then divide by 1.207

div4:
 ld *AR_Z, a         ; store the value of r into Acc A
 sfta a, –2          ; divide by 4 = r/4
 stl a, *AR_Z        ; r available at *AR_Z again
 bmultiply
;ld  *AR_TABLE,a   ; 1 cycle, Read exponent value from table.
;stl a,*AR_ZEXP    ; 1 cycle

div1207:
 stm #0500h, ar5
 st #cmprval_2, *ar5
 ld *ar5, t          ; load 0.8284 into Treg
 mpy *AR_Z, a        ; r * 0.8284
 ld *AR_TABLE, t     ; re–enter exponent
 stl a, *AR_Z        ; restore magnitude (r * 0.8284) to AR_Z
 nop

multiply:
 MPY *AR_Z, a        ; = {(r/4)*rexp} OR {(r * 0.8284)*rexp}

 rsbx ovm
 rsbx frct
 rsbx tc
 popm ar5
 popm ar4
 popm ar3
 popm ar2
 popm ar1
end_reciprocal:
 ret

********************* End of “RECIPROCAL” routine *******************
```

```
; Return
;--------
end_lab;

 frame +2
 popm ar7
 popm ar6
 popm ar1

 .if __far_mode
 fretd
 .else
 retd
 .endif
 rsbx frct
 rsbx ovm

    .def InvYeTable
 .data
InvYeTable:
 .word 0002h  ; Ye = 2^1
 .word 0004h  ; Ye = 2^2
 .word 0008h    Ye = 2^3
 .word 0010h  ; Ye = 2^4
 .word 0020h  ; Ye = 2^5
 .word 0040h  ; Ye = 2^6
 .word 0080h  ; Ye = 2^7
 .word 0100h  ; Ye = 2^8
 .word 0200h  ; Ye = 2^9
 .word 0400h  ; Ye = 2^10
 .word 0800h  ; Ye = 2^11
 .word 1000h  ; Ye = 2^12
 .word 2000h  ; Ye = 2^13
 .word 4000h  ; Ye = 2^14
 .word 8000h  ; Ye = 2^15


;end of file. please do not remove. it is left here to ensure that no
lines of code are removed by any editor
```

A.2 *Cbrev.asm*

```
;***********************************************************************
;  Function: cbrev
;  Description: complex bit-reverse routine (C54x)
;  Version: 1.00
;
;  Copyright Texas instruments Inc, 1998
;-----------------------------------------------------------------------
;  Revision History:
;  1.00  R. Piedra, 8/31/98. Original release.
;***********************************************************************

        .mmregs
 .if __far_mode
offset .set 1
 .else
offset .set 0
 .endif
      ; stack description
 .asg (0), ret_addr

      ; x in A
 .asg (1+ offset), arg_y
 .asg (2+ offset), arg_n

      ; register usage
      ; ar0 : bit reversing idx
 .asg ar2,ar_dst
 .asg ar3,ar_src

 .global     _cbrev
        .text

_cbrev
 ssbx frct   ; fractional mode is on  (1)
 ssbx sxm    ;    (1)

; Get arguments
; -------------
 stlm a, ar_src             ; pointer to src  (1)
 mvdk *sp(arg_y), *(ar_dst) ; pointer to dst (temporary) (2)
 ld *sp(arg_n), a           ; a = n  (1)
 stlm a, AR0                ; AR0 = n = 1/2 size of circ buffer (1)
 sub #3,a                   ; a = n-3(by pass 1st and last elem)(2)

; Select in-place or off-place bit-reversing
; ------------------------------------------
 ldm ar_src,b               ; b = src_addr   (1)
 sub *sp(arg_y),b           ; b = src_addr - dst_addr (1)
 bcd in_place, beq          ; if (ar_src==ar_dst)then in_place (2)
 stlm a, brc                ; brc = n-3   (1)
 nop                        ;    (1)
```

```
; Off–place bit–reversing
; ----------------------

off_place:
_start1:

                        ; unroll to fill delayed slots
 rptbd off_place_end–1  ;    (2)
 mvdd *ar_src+,*ar_dst+ ; move real component  (1)
 mvdd *ar_src–,*ar_dst+ ; move Im component  (1)

 mar *ar_src+0B         ;    (1)
 mvdd *ar_src+,*ar_dst+ ; move real component  (1)
 mvdd *ar_src–,*ar_dst+ ; move Im component  (1)

off_place_end:
 mar *ar_src+0B         ;     (1)
 bd end                 ;     (2)
 mvdd *ar_src+,*ar_dst+ ; move real component  (1)
 mvdd *ar_src–,*ar_dst+ ; move Im component  (1)


; In–place bit–reversing
; ----------------------

in_place:

 mar *ar_src+0B         ; bypass first and last element (1)
 mar *+ar_dst(2)        ;     (1)
_start2:
 rptbd in_place_end–1   ;     (2)
 ldm ar_src,a           ; b = src_addr   (1)
 ldm ar_dst, b          ; a = dst_addr   (1)



 sub b,a                ; a =  src_addr  – dst_addr (1)
                        ; if >=0  bypass move just increment
 bcd bypass, ageq       ; if (src_addr>=dst_addr) then skip(2)
 ld *ar_dst+, a         ; a = Re dst element (preserve) (1)
 ld *ar_dst–, b         ; b = Im dst element (preserve) (1)

 mvdd *ar_src+, *ar_dst+ ; Re dst = Re src  (1)
 mvdd *ar_src , *ar_dst– ; Im dst = Im src;point to Re (1)
 stl b, *ar_src–         ; Im src = b = Im dst;point to Re (1)
 stl a, *ar_src          ; Re src = a = Re dst  (1)

bypass
 mar *ar_src+0B          ;    (1)
 mar *+ar_dst(2)         ;    (1)

 ldm ar_src,a            ; b = src_addr   (1)
 ldm ar_dst, b           ; a = dst_addr   (1)
```

```
in_place_end



; Return
; ------

_end:
end
 .if __far_mode
 fretd
 .else
 retd
 .endif
 rsbx frct
 rsbx ovm

;end of file. please do not remove. it is left here to ensure that no
lines of code are removed by any editor
```

A.3 *Macros.asm*

```
;***********************************************************************
;  Filename:  macros.asm
;  Version :  1.00
;  Description:  collections of macros for cfft
;----------------------------------------------------------------------
;  Description: Contains the following macros
;----------------------------------------------------------------------
; Revision History:
;
; 0.00 M. Christ/M. Chishtie. Original code
; 1.00 R./ Piedra, 8/31/98
;     – Modifed stage3 macro to correct functional problem
;     – Modified order of xmem, ymem operands in butterfly code
;       to reduce number of cycles from 10 to 8
;
;***********************************************************************
;
;Variation from macros.asm in fft_approach2.mak.  Here the
;auto scaling has been disabled in:
; stage3, stdmacro and laststag
;
;***********************************************************************

 .mmregs
;***********************************************************************
; macro : combo5xx
;
; COMBO5xx macro implements a bit reversal stage and the first two FFT
; stages (radix–4 implementation). Bit reversal is now done in the same
; loop
; thereby saving cycles. Circular addressing is used to access INPUT
;  buffer and
; bit–reversed addressing is used to implement the DATA buffer.
;  Therefore INPUT
; buffer must now be aligned at 4*N and DATA buffer at 2*N boundary.
;  (MCHI)
;----------------------------------------------------------------------
combo5xx .macro        ; REPEAT MACRO ‘combo5xx’: N/4 times
; .global STAGE1,COMBO1,COMBO2,end1,end2,end?

*
* R1  := [(R1+R2)+(R3+R4)]/4  INPUT               OUTPUT
*
* R2  := [(R1–R2)+(I3–I4)]/4  ------------------  ------------------
*
* R3  := [(R1+R2)–(R3+R4)]/4  AR0 =  7
*
* R4  := [(R1–R2)–(I3–I4)]/4  AR1 –> R1,I1        AR1 – > R5,I5
*
* I1  := [(I1+I2)+(I3+I4)]/4  AR2 –> R2,I2        AR2 – > R6,I6
*
* I2  := [(I1–I2)–(R3–R4)]/4  ARP–> AR3 –> R3,I3  ARP – > AR3 – > R7,I7
*
```

```
* I3  := [(I1+I2)–(I3+I4)]/4  AR4 –> R4,I4        AR4 – > R8,I8
*
* I4  := [(I1–I2)+(R3–R4)]/4
*

;
STAGE1:
 mvdk *sp(DATA),ar2; (RMP) pointer to DATA   r1,i1
 mvdk *sp(DATA),ar3

 mvmm ar3,ar4
 mvmm ar3,ar5

 mar *+ar3(2)  ; pointer to DATA + 2  r2,i2
 mar *+ar4(4)  ; pointer to DATA + 4  r3,i3
 mar *+ar5(6)  ; pointer to DATA + 6  r4,i4

 ld *sp(scale), a
 bcd COMBO2, AEQ
 ld      #0,ASM            ; ASM=0
 nop

 ld #–2, ASM
        .if     N>4
        stm     #7,ar0        ; index
        stm     #0,BK         ; blocksize to zero!
        stm     #N/4–1,BRC    ; execute N/4–1 times ‘combo5xx’
 rptb end1
 .endif
;                                                  AR2 AR3 AR4 AR5
;                              ;                   --- --- --- ---
COMBO1  sub     *ar2,*ar3,B    ; B  := (R1–R2)      R1  R2  R3  R4
        add     *ar2,*ar3,A    ; A  := (R1+R2)      R1  R2  R3  R4
        sth     B,ASM,*ar3     ; R2’:= (R1–R2)/4    R1  R2  R3  R4
        add     *ar4,*ar5,B    ; B  := (R3+R4)      R1  R2  R3  R4
        add     B,A            ; A  := (R1+R2) + (R3+R4)
                               ;                   R1  R2  R3  R4
        sth     A,ASM,*ar2+    ; R1’:=((R1+R2) + (R3+R4))/4
                               ;                   I1  R2  R3  R4
        sub     B,1,A          ; B  :=((R1+R2) – (R3+R4))
                               ;                   I1  R2  R3  R4
        sub    *ar4,*ar5,B     ; B  := (R3–R4)
                               ;                   I1  R2  R3  R4
        st      A,*ar4+ ;ASM   ; R3’:=((R1+R2) – (R3+R4))/4
                               ;                   I1  R2  I3  R4
   ||   ld      *ar3,A ; 16    ; A  := (R1–R2)/4    I1  R2  I3  R4
        sth     B,ASM,*ar5+    ; R4’:= (R3–R4)/4    I1  R2  I3  I4
        sub    *ar4,*ar5–,B    ; B  := (I3–I4)      I1  R2  I3  R4
        add     B,ASM,A        ; A  := (R1–R2) + (I3 –I4)/4
                               ;                   I1  R2  I3  R4
        sth     A,*ar3+        ; R2’:= (R1–R2) + (I3 –I4)/4
                               ;                   I1  I2  I3  R4
        sub     B,–1,A         ; A  :=((R1–R2) – (I3–I4))
                               ;                   I1  I2  I3  R4
```

```
        ld      *ar5,16,B       ; B=R3–R4
        sth     A,*ar5+         ; R4’:=((R1–R2) – (I3–I4))/4
                                ;                      I1  I2  I3  I4
        add     *ar4,*ar5,A     ; A  := (I3+I4)         I1  I2  I3  I4
        sth     A,ASM,*ar4      ; I3’:= (I3+I4)/4       I1  I2  I3  I4
        sub     *ar2,*ar3,A     ; A  := (I1–I2)         I1  I2  I3  I4
        add     B,2,A           ; A  := (I1–I2)+ (r3–r4)
                                ;                       I1  I2  I3  I4
        sth     A,ASM,*ar5+0    ; I4’:= (I1–I2)+ (r3–r4)/4
                                ;                       I1  I2  I3  R4’
        sub     B,3,A           ; A  := (I1–I2)– (r3–r4)
                                ;                       I1  I2  I3  R4’
        add     *ar2,*ar3,B     ; B  := (I1+I2)         I1  I2  I3  R4’
        st      A,*ar3+0% ;asm; I2’:= (I1–I2)–(R3–R4)/4
                                ;              I1  R2’ I3  R4’
   ||   ld      *ar4,A    ;16 ; A  := (I3+I4)/4       I1  R2’ I3  R4’
        add     A,2,B           ; B  := (I1+I2)+(I3+I4)
                                ;                       I1  R2’ I3  R4’
        sth     B,ASM,*ar2+0    ; I1’:= (I1+I2)+(I3+I4)/4
                                ;                       R1’ R2’ I3  R4’
        sub     A,3,B           ; B  := (I1+I2)–(I3+I4)/4
                                ;                       R1’ R2’ I3  R4’
end1    sth     B,ASM,*ar4+0    ; I3’:= (I1+I2)–(I3+I4)/4
                                ;                       R1’ R2’ R3’ R4’
 bend?


COMBO2
        .if     N>4
        stm     #7,ar0          ; index
        stm     #0,BK           ; blocksize to zero!
        stm     #N/4–1,BRC      ; execute N/4–1 times ’combo5xx’
 rptb end2       ;
        .endif
                                          ;         AR2 AR3 AR4 AR5
;                                         ;         --- --- --- ---
 sub     *ar2,*ar3,B            ; B  := (R1–R2)       R1  R2  R3  R4
        add     *ar2,*ar3,A     ; A  := (R1+R2)       R1  R2  R3  R4
        sth     B,ASM,*ar3      ; R2’:= (R1–R2)       R1  R2  R3  R4
        add     *ar4,*ar5,B     ; B  := (R3+R4)       R1  R2  R3  R4
        add     B,A             ; A  := (R1+R2) + (R3+R4)
                                ;                       R1  R2  R3  R4
        sth     A,ASM,*ar2+     ; R1’:= (R1+R2) + (R3+R4)
                                ;                       I1  R2  R3  R4
        sub     B,1,A           ; A  := (R1+R2) – (R3+R4)
                                ;                       I1  R2  R3  R4
        sub    *ar4,*ar5,B      ; B  := (R3–R4)       I1  R2  R3  R4
        st      A,*ar4+ ;ASM    ; R3’:= (R1+R2) – (R3+R4)
                                ;                       I1  R2  I3  R4
   ||   ld      *ar3,A ; 16     ; A  := (R1–R2)        I1  R2  I3  R4
        sth     B,ASM,*ar5+     ; R4’:= (R3–R4)        I1  R2  I3  I4
        sub    *ar4,*ar5–,B     ; B  := (I3–I4)        I1  R2  I3  R4
        add     B,ASM,A         ; A  := (R1–R2) + (I3–I4)
                                ;                       I1  R2  I3  R4
        sth     A,*ar3+         ; R2’:= (R1–R2) + (I3–I4)
```

```
                              ;I1  I2  I3  R4
        sub     B,1,A         ; A  := (R1-R2) - (I3-I4)
                              ;                    I1  I2  I3  R4
        ld      *ar5,16,B     ; B=R3-R4
        sth     A,*ar5+       ; R4':= (R1-R2) - (I3-I4)
                              ;                    I1  I2  I3  I4
        add     *ar4,*ar5,A   ; A  := (I3+I4)      I1  I2  I3  I4
        sth     A,ASM,*ar4    ; I3':= (I3+I4)      I1  I2  I3  I4
        sub     *ar2,*ar3,A   ; A  := (I1-I2)      I1  I2  I3  I4
        add     B,A           ; A  := (I1-I2)+ (r3-r4)
                              ;                    I1  I2  I3  I4
        sth     A,ASM,*ar5+0  ; I4':= (I1-I2)+ (r3-r4)
                              ;                    I1  I2  I3  R4'
        sub     B,1,A         ; A  := (I1-I2)- (r3-r4)
                              ;                    I1  I2  I3  R4'
        add     *ar2,*ar3,B   ; B  := (I1+I2)      I1  I2  I3  R4'
        st      A,*ar3+0% ;asm; I2':= (I1-I2)-(R3-R4)
                              ;                    I1  R2' I3  R4'
   ||   ld      *ar4,A    ;16 ; A  := (I3+I4)      I1  R2' I3  R4'
        add     A,B           ; B  := (I1+I2)+(I3+I4)
                              ;I1  R2' I3  R4'
        sth     B,ASM,*ar2+0  ; I1':= (I1+I2)+(I3+I4)
                              ;                    R1' R2' I3  R4'
        sub     A,1,B         ; B  := (I1+I2)-(I3+I4)
                              ;                    R1' R2' I3  R4'
end2    sth     B,ASM,*ar4+0  ; I3':= (I1+I2)-(I3+I4)
                             ;                    R1' R2' R3' R4'
end? .endm
;***********************************************************************
; macro: stage3
;
; STAGE3 macro is improved such that it now takes only 31 cycles per
;iteration.
; It uses two additional auxiliary registers(AR1,AR4) to support
;indexing.(MCHI)
;----------------------------------------------------------------------

stage3      .macro

;        .global STAGE3,MCR3,end?

        .asg    AR2,P
        .asg    AR3,Q

STAGE3:
 ld #0, ASM ; Introduced by AC 06/06/99 to bypass autoscaling
            ; and scale only when required within the file
            ; cfft64_2.asm
 ld *sp(DATA),a    ; a = DATA
 stlm a, P         ; pointer to DATA pr,pi
 add #8,a          ; a = DATA + #8
 stlm a, Q         ; pointer to DATA + 8 qr,qi


 ld *sp(scale),a
```

```
        STM     #9,AR1
        STM     #2,AR4
 xc 1,ANEQ
 ld #-1,ASM

        .if     N>8
        STM     #N/8-1,BRC  ; execute N/8-1 times '4 macros'
        RPTBD   end?        ;
        .endif              ;
 LD *sp(SIN45),T            ; load to sin(45)
 nop
*************************************************************************
*
*
*     MACRO requires   number of words/number of cycles: 6.5
*
*        PR'=(PR+QR)/2      PI'=(PI+QI)/2
*
*        QR'=(PR-QR)/2      QI'=(PI-QI)/2
*
*  version 0.99    from Manfred Christ     update:  2. May. 94
*
*************************************************************************
;                               (contents of register after exec.)
;                                                          AR2  AR3
;                                                          ---  ---
MCR3  LD    *P,16,A         ; A :=       PR                PR   QR
      SUB   *Q,16,A,B       ; B :        PR-QR             PR   QR
      ST    B,*Q            ; QR:= (1/2)(PR-QR)
||    ADD   *Q+,B           ; B :=      (PR+QR)            PR   QI
      ST    B,*P+           ; PR:= (1/2)(PR+QR)
||    LD    *Q,A            ; A :=          QI             PI   QI
      ST    A,*Q            ; Dummy write
||    SUB   *P,B            ; B :=      (PI-QI)            PI   QI
      ST    B,*Q+           ; QI:= (1/2)(PI-QI)            PI   QR+1
||    ADD   *P,B            ; B :=      (PI+QI)
      ST    B,*P+           ; PI:= (1/2)(PI+QI)            PR+1 QR+1


*************************************************************************
*
*     MACRO requires   number of words/number of cycles: 9
*
*  T=SIN(45)=COS(45)=W45
*
*        PR'= PR + (W*QI + W*QR) = PR + W * QI + W * QR    (<- AR2)
*
*        QR'= PR - (W*QI + W*QR) = PR - W * QI - W * QR    (<- AR3)
*
*        PI'= PI + (W*QI - W*QR) = PI + W * QI - W * QR    (<- AR2+1)
*
*        QI'= PI - (W*QI - W*QR) = PI - W * QI + W * QR    (<- AR3+2)
*
```

```
*
*
*        PR'= PR +  W * (QI + QR)    (<- AR2)
*
*        QR'= PR -  W * (QI + QR)    (<- AR3)
*
*        PI'= PI +  W * (QI - QR)    (<- AR2+1)
*
*        QI'= PI -  W * (QI - QR)    (<- AR3+1)
*
*  version 0.99    from Manfred Christ     update: 2. May. 94
*
*
*********************************************************************

||    MPY   *Q+,A          ;A   =  QR*W                    PR   QI
      MVMM  AR4,AR0        ;Index = 2
      MAC   *Q-,A          ;A  :=     (QR*W +QI*W)         PR   QR
      ADD   *P,16,A,B      ;B  := (PR+(QR*W +QI*W ))       PR   QR
      ST    B,*P    ;<<ASM;PR':= (PR+(QR*W +QI*W ))/2     PI   QR
||    SUB   *P+,B          ;B  := (PR-(QR*W +QI*W ))       PI   QR
      ST    B,*Q    ;<<ASM;QR':= (PR-(QR*W +QI*W ))/2
||    MPY   *Q+,A          ;A  :=      QR*W                PI   QI
      MAS   *Q,A           ;A  := (   (QR*W -QI*W ))       PI   QI
      ADD   *P,16,A,B      ;B  := (PI+(QR*W -QI*W ))       PI   QI
      ST    B,*Q+0%        ;QI':= (PI+(QR*W -QI*W ))/2     PI   QI+1
||    SUB   *P,B           ;B  := (PI-(QR*W -QI*W ))       PI   QI+1
      ST    B,*P+          ;PI':= (PI-(QR*W -QI*W ))/2     PR+1 QI+1
*********************************************************************
*
*
*     MACRO 'PBY2I'    number of words/number of cycles: 6
*
*        PR'=(PR+QI)/2      PI'=(PI-QR)/2
*
*        QR'=(PR-QI)/2      QI'=(PI+QR)/2
*
*  version 0.99    from Manfred Christ     update:  2. May. 94
*
*********************************************************************
;                                (contents of register after exec.)
;                                                       AR2      AR3
;                                                       ---      ---
||    LD    *Q-,A          ; A   :=     QI             PR       QR
; rmp ADD   *P,A,B         ; B   := (PR+QI)            PR       QR
; rmp: 8/31/98 corrected following ADD instruction
      ADD   *P,16,A,B      ; B   := (PR+QI)            PR       QR
      ST    B,*P           ; PR' := (PR+QI)/2
||    SUB   *P+,B          ; B   := (PR-QI)            PI       QR
      ST    B,*Q           ; QR' := (PR-QI)/2
||    LD    *Q+,A          ; A   :=     QR             PI       QI
; rmp ADD   *P,A,B         ; B   := (PI+QR)            PI       QI
; rmp 8/31/98 corrected following ADD instruction
```

```
      ADD   *P,16,A,B       ; B   := (PI+QR)          PI       QI
      ST    B,*Q+           ; QI' := (PI+QR)/2        PI       QR+1
||    SUB   *P,B            ; B   := (PI-QR)
      ST    B,*P+           ; PI' := (PI-QR)/2        PR+1     QR+1

*****************************************************************
*
*     MACRO requires   number of words/number of cycles: 9.5
*
*  version 0.99    from: Manfred Christ   update:  2. May. 94
*
*   ENTRANCE IN THE MACRO: AR2->PR,PI
*
*                          AR3->QR,QI
*
*                          TREG=W=COS(45)=SIN(45)
*
*       EXIT OF THE MACRO: AR2->PR+1,PI+1
*
*                          AR3->QR+1,QI+1
*
*        PR'= PR + (W*QI - W*QR) = PR + W * QI - W * QR    (<- AR1)
*
*        QR'= PR - (W*QI - W*QR) = PR - W * QI + W * QR    (<- AR2)
*
*        PI'= PI - (W*QI + W*QR) = PI - W * QI - W * QR    (<- AR1+1)
*
*        QI'= PI + (W*QI + W*QR) = PI + W * QI + W * QR    (<- AR1+2)
*
*        PR'= PR + W*(QI - QR)  = PR - W *(QR -QI)   (<- AR2)
*
*        QR'= PR - W*(QI - QR)  = PR - W *(QR -QI)   (<- AR3)
*
*        PI'= PI - W*(QI + QR)     (<- AR2+1)
*
*        QI'= PI + W*(QI + QR)     (<- AR3+1)
*
* BK==0 !!!!!
*
*****************************************************************
;                                                        AR2 AR3
;                                                        --- ---
||    MPY   *Q+,A           ;A  :=  QR*W                PR  QI
      MVMM  AR1,AR0         ;Index = 9
      MAS   *Q-,A           ;A  := (QR*W -QI*W )        PR  QR
      ADD   *P,16,A,B       ;B  := (PR+(QR*W -QI*W ))   PR  QR
      ST    B,*Q+    ;<<ASM;QR':= (PR+(QR*W -QI*W ))/2  PR  QI
||    SUB   *P,B            ;B  := (PR-(QR*W -QI*W ))
      ST    B,*P+    ;<<ASM;PR':= (PR-(QR*W -QI*W ))/2
||    MAC   *Q,A            ;A  := QR*W                 PI  QI

     MAC  *Q,A              ;A  := (   (QR*W +QI*W ))   PI  QI
      ADD   *P,16,A,B       ;B  := (PI+(QR*W +QI*W ))   PI  QI
```

```
      ST    B,*Q+0%   ;<ASM;QI'= (PI+(QR*W +QI*W ))/2      PI  QR+1
||    SUB   *P,B           ;B  := (PI-(QR*W +QI*W ))
      STH   B,ASM,*P+0%    ;PI'= (PI-(QR*W +QI*W ))/2      PR+1QR+1
end?  .set  $-1

      STM   #-2,AR0        ;Index used in stdmacro macro
      .endm

;*********************************************************************
; macro : laststag
;---------------------------------------------------------------------

laststag .macro  stage,sin,cos
;         .global STAGE:stage:,end?
STAGE:stage: .set $

 ld #0, ASM ; Introduced by AC 06/06/99 to bypass autoscaling
      ; and scale only when required within the file
      ; cfft64_2.asm
 ld *sp(DATA),a
 stlm a, ar2  ; ar2 -> DATA
 add #N,a
 stlm a, ar3  ; ar3 -> DATA+(offset=N)
      stm   #cos,ar4         ; start of cosine in stage 'stg'
      stm   #sin,ar5         ; start of sine in stage   'stg'
      buttfly N/2            ; execute N/2 butterflies
      .endm

;*********************************************************************
; macro : stdmacro
;---------------------------------------------------------------------

stdmacro .macro  stage,l1,l2,idx,sin,cos
;         .global STAGE:stage:,end?
STAGE:stage: .set $

 ld #0, ASM ; Introduced by AC 06/06/99 to bypass autoscaling
            ; and scale only when required within the file
            ; cfft64_2.asm
 ld *sp(DATA),a
 stl a,ar2   ; ar2 -> DATA
 add #idx,a  ; ar3 -> DATA+(offset=idx)
 stlm a,ar3

      stm   #l1-1,ar1        ; outer loop counter
      stm   #cos,ar6         ; start of cosine in stage 'stg'
      stm   #sin,ar7         ; start of sine in stage 'stg'

loop? mvmm  ar6,ar4          ; start of cosine in stage 'stg'
      mvmm  ar7,ar5          ; start of sine in stage 'stg'

      buttfly l2             ; execute l2 butterflies

      mar   *+ar2(idx)
      banzd loop?,*ar1-
```

```
      mar   *+ar3(idx)
      .endm

;***********************************************************************
; macro: buttfly
;
; Improved radix–2 butterfly code from 9 to 8 cycles per iteration. The
; new butterfly uses AR0 for indexing and the loop is unrolled such
; that one butterfly is implemented outside the loop.
;-----------------------------------------------------------------------

buttfly .macro num          ;      (contents of register after exec.)

      .asg  AR2, P
      .asg  AR3, Q
      .asg  AR4,WR
      .asg  AR5,WI

 ld #0, ASM ; Introduced by AC 06/06/99 to bypass autoscaling
      ; and scale only when required within the file
      ; cfft64_2.asm
      ; it should already be disabled by this point, since
      ; this has already been invoked in stdmacro and
      ; laststag.

;X    STM  #–2,AR0           ; index = –2
      STM  #:num:–3,BRC      ; execute startup + num–3 times general
BUTTFLY
;                                                      AR2  AR3  AR4  AR5
; takes 17 words–/cycles (including RPTB)              ---  ---  ---  ---
      LD   *P,16,A         ;A  :=  PR                PR   QR   WR   WI
      SUB  *Q,16,A,B       ;B  :   PR–QR             PR   QR   WR   WI
      ST   B,*Q     ;<<ASM;QR’:= (PR–QR)/2
||    ADD  *Q+,B           ;B  := (PR+QR)            PR   QI   WR   WI
      ST   B,*P+    ;<<ASM;PR’:= (PR+QR)/2
||    LD   *Q,A      ;<<16 ;A  :=  QI                PI   QI   WR   WI
      ADD  *P,16,A,B       ;B  := (PI+QI)            PI   QI   WR   WI
      ST   B,*P     ;<<ASM;PI’:= (PI+QI)/2
||    SUB  *P+,B           ;B  := (PI–QI)            PR+1 QR   WR   WI
      STH   B,ASM,*Q+      ;QI’:= (PI–QI)/2          PR+1 QR+1 WR   WI

      MPY   *WR,*Q+,A      ;A  := QR*WR              PR+1 QI+1 WR   WI
      MAC   *WI+,*Q–,A     ;A  := (QR*WR+QI*WI) || T=WI
                           ;                         PR+1 QR+1 WR   WI+1
      ADD   *P,16,A,B      ;B  := (PR+(QR*WR+QI*WI)) PR+1 QR+1 WR   WI+1
      ST    B,*P    ;<<ASM;PR’:= (PR+(QR*WR+QI*WI))/2
||    SUB   *P+,B          ;B  := (PR–(QR*WR+QI*WI)) PI+1 QR+1 WR   WI+1
      ST    B,*Q    ;<<ASM;QR’:= (PR–(QR*WR+QI*WI))/2
||    MPY   *Q+,A          ;A  := QR*WI     [t=WI]   PI+1 QI+1 WR   WI+1
      MAS   *WR+,*Q,A      ;A  := (   (QR*WI–QI*WR)) PI+1 QI+1 WR+1 WI+1

      RPTBD end?–1         ;delayed block repeat
      ST    A,*Q+          ;dummy write
||    SUB   *P,B           ;B  := (PI–(QR*WI–QI*WR)) PI+1 QR+2 WR+1 WI+1
      ST    B,*P    ;<<ASM;PI’:= (PI–(QR*WI–QI*WR))/2
```

```
||    ADD   *P+,B         ;B  := (PI+(QR*WI–QI*WR)) PR+2 QR+2 WR+1 WI+1
;
; Butterfly kernal with 8 instructions / 8 cycles
;
; rmp MPY   *WR,*Q+,A  ;A  :=  QR*WR      PR+2 QI+2 WR+1 WI+1
; rmp reversed order in following MPY instruction
      MPY   *Q+,*WR,A  ;A  :=  QR*WR      PR+2 QI+2 WR+1 WI+1
      MAC   *WI+,*Q+0%,A  ;A  := (QR*WR+QI*WI) || T=WI
                     ;                    PR+2 QI+1 WR+1 WI+2
      ST    B,*Q+   ;<<ASM;QI’:= (PI+(QR*WI–QI*WR))/2
||    ADD   *P,B          ;B  := (PR+(QR*WR+QI*WI))
                          ;               PR+2 QR+2 WR+1 WI+2
      ST    B,*P    ;<<ASM;PR’:= (PR+(QR*WR+QI*WI))/2
||    SUB   *P+,B         ;B  := (PR–(QR*WR+QI*WI))
                          ;               PI+2 QR+2 WR+1 WI+2
      ST    B,*Q    ;<<ASM;QR’:= (PR–(QR*WR+QI*WI))/2
||    MPY   *Q+,A         ;A  := QR*WI     [t=WI]
                          ;               PI+2 QI+2 WR+1 WI+2
; rmp MAS   *WR+,*Q,A   ;A  := (   (QR*WI–QI*WR))
                          ;               PI+2 QI+2 WR+2 WI+2
; rmp reversed order in following MPY instruction
     MAS   *Q,*WR+,A   ;A  := (   (QR*WI–QI*WR))
                          ;               PI+2 QI+2 WR+2 WI+2
     ST    A,*Q+         ;dummy write
||   SUB   *P,B           ;B  := (PI–(QR*WI–QI*WR))
                          ;               PI+2 QR+3 WR+2 WI+2
      ST   B,*P    ;<<ASM;PI’:= (PI–(QR*WI–QI*WR))/2
||    ADD  *P+,B          ;B  := (PI+(QR*WI–QI*WR))
                          ;               PR+3 QR+3 WR+2 WI+2
end?
      MAR  *Q–
      STH   B,ASM,*Q+     ;QI’:= (PI+(QR*WI–QI*WR))/2
                         ;               PR+3 QR+3 WR+2 WI+2
      .endm

;end of file. please do not remove. it is left here to ensure that no
lines of code are removed by any editor
```

A.4 Cfft_t.c

```
//*********************************************************************/
//  Filename: cfft_t.c
//  Version:  0.01
//  Description: test for cfft routine
//---------------------------------------------------------------------
//  Revision History:
//    0.01, R. Piedra, 06/15/98, – Original release
//*********************************************************************/

#include “math.h”
#include “tms320.h”
#include “dsplib.h”

#include “test.h”

short i;
short eflagf= PASS;
short eflagi= PASS;

short scale = 1;
short noscale = 0;

short x1[2*NX];

main()
{
 for (i=0; i<2*NX; i++)
 {
/* x[i] = 32767; */
  x1[i] = x[i];
 }

/*compute */
 cbrev(x,x,NX);
 cfft(x,NX,scale);
/*test fft */
 eflagf  = test(x, rtest, NX, MAXERROR); /* for r */

/*for (i=0; i<2*NX; i++)
 {
  x[i] = x[i]>>3;
 }

 cbrev(x,x,NX);
 cifft(x,NX,noscale);*/
/*test ifft */
/*eflagi  = test(x, x1, NX, MAXERROR); /* for r */
 return;
}
```

A.5 Test.c

```
//*****************************************************************/
/*  Filename: test.c
//  Version:  0.01
// Description: test r against rtest (array of n elements)
//   Returns eflag
//-----------------------------------------------------------------
//  Revision History:
//    0.01, R. Piedra, 06/15/98, – Original release
//*****************************************************************/

#include “tms320.h”

short test(DATA *r, DATA *rtest, short n, DATA maxerror)

{
short i;
short eflag = PASS;  /* error flag or index into r vector where error */
DATA elevel = 0;     /* error level at failing eflag index location  */
DATA emax = 0;       /* max error level detected across when NOERROR */

for (i=0;i<n;i++)
    {
    if ( (elevel = ABSVAL(rtest[i] – r[i])) > maxerror)
       {
       eflag =i; /* if error –-> eflag = index and emax= max error */
       emax = elevel;
                 /* if no error –-> eflag = –1 and emax = max error */
       break;
       }
    else
    if (elevel>emax) emax = elevel;
   }
    /* Pass to Host: eflag and emax */
return(eflag);
}
```

A.6 Dsplib.h

```
#ifndef _DSPLIB
#define _DSPLIB

#include “tms320.h”

/* fft */

short cfft8 (DATA *x, DATA scale);
short cfft16 (DATA *x, DATA scale);
short cfft32 (DATA *x, DATA scale);
short cfft64 (DATA *x, DATA scale);
short cfft128 (DATA *x, DATA scale);
short cfft256 (DATA *x, DATA scale);
short cfft512 (DATA *x, DATA scale);
short cfft1024 (DATA *x, DATA scale);

short rfft16 (DATA *x, DATA scale);
short rfft32 (DATA *x, DATA scale);
short rfft64 (DATA *x, DATA scale);
short rfft128 (DATA *x, DATA scale);
short rfft256 (DATA *x, DATA scale);
short rfft512 (DATA *x, DATA scale);
short rfft1024 (DATA *x, DATA scale);

/* ifft */

short cifft8 (DATA *x, DATA scale);
short cifft16 (DATA *x, DATA scale);
short cifft32 (DATA *x, DATA scale);
short cifft64 (DATA *x, DATA scale);
short cifft128 (DATA *x, DATA scale);
short cifft256 (DATA *x, DATA scale);
short cifft512 (DATA *x, DATA scale);
short cifft1024 (DATA *x, DATA scale);

short rifft16 (DATA *x, DATA scale);
short rifft32 (DATA *x, DATA scale);
short rifft64 (DATA *x, DATA scale);
short rifft128 (DATA *x, DATA scale);
short rifft256 (DATA *x, DATA scale);
short rifft512 (DATA *x, DATA scale);
short rifft1024 (DATA *x, DATA scale);

short cbrev (DATA *x, DATA *y, ushort n);

/* correlations */

short acorr_raw (DATA *x, DATA *r, ushort nx, ushort nr);
short acorr_bias(DATA *x, DATA *r, ushort nx, ushort nr);
short acorr_unbias(DATA *x, DATA *r, ushort nx, ushort nr);

short corr_raw (DATA *x, DATA *y, DATA *r, ushort nx, ushort ny);
short corr_bias  (DATA *x, DATA *y, DATA *r, ushort nx, ushort ny);
```

```
short corr_unbias  (DATA *x, DATA *y, DATA *r, ushort nx, ushort ny);

/* filtering and convolution */

short convol (DATA *x, DATA *y, DATA *r, ushort ny, ushort nr);
short fir(DATA *x, DATA *h, DATA *r,DATA **d, ushort nh, ushort nx);
short firs(DATA *x, DATA *r,DATA **d, ushort nh, ushort nx);
short firs2(DATA *x, DATA *h, DATA *r,DATA **d, ushort nh, ushort nx);
short cfir(DATA *x, DATA *h, DATA *r,DATA **d, ushort nh, ushort nx);

short iircas4(DATA *x,DATA *h,DATA *r,DATA **d, ushort nbiq, ushort
nx);
short iircas5(DATA *x,DATA *h,DATA *r,DATA **d, ushort nbiq, ushort
nx);
short iircas51(DATA *x,DATA *h,DATA *r,DATA **d, ushort nbiq, ushort
nx);
short iir32(DATA *x,LDATA *h,DATA *r,LDATA **d, ushort nbiq, ushort
nx);

short firdec(DATA *x, DATA *h, DATA *r,DATA **d, ushort nh, ushort nx,
ushort D);
short firinterp(DATA *x,DATA *h,DATA *r,DATA **db,ushort nh,ushort
nx,ushort I);

short latfor (DATA *x, DATA *h, DATA *r, DATA *d, ushort nx, ushort
nh);

/* adaptive filtering */

short dlms(DATA *x,DATA *h,DATA *r, DATA **d, DATA *des, DATA step,
ushort nh, ushort nx);
short nblms (DATA *x,DATA *h,DATA *r, DATA **d, DATA *des, ushort nh,
ushort nx, ushort nb, DATA **norm_e, int l_tau, int cutoff, int gain);
short ndlms (DATA *x, DATA *h, DATA *r, DATA *d, DATA *des, ushort nh,
ushort nx, int l_tau, int cutoff, int gain, DATA *norm_d);

/* math */

short add (DATA *x, DATA *y, DATA *r, ushort nx, short scale);
short sub(DATA *x, DATA *y,  DATA *r, ushort nx, ushort scale);
short neg(DATA *x, DATA *r, ushort nx);

void recip16 (DATA *x, DATA *z, DATA *zexp, ushort n);


short expn(DATA *x, DATA *r, ushort nx);
short logn(DATA *x, LDATA *r, ushort nx);
short log_2(DATA *x, LDATA *r, ushort nx);
short log_16(DATA *x, LDATA *r, ushort nx);

short sqrt_16(DATA *x, DATA *y, short nx);

short maxidx (DATA *x, ushort nx);
short maxval (DATA *x, ushort nx);
short minidx (DATA *x, ushort nx);
```

```
short minval (DATA *x, ushort nx);

short rand16(DATA *r, ushort nr);
void rand16init(void);

short mul32(LDATA *x, LDATA *y,  LDATA *r, ushort nx);
short neg32(LDATA *x, LDATA *r, ushort nx);
short power(DATA *x, LDATA *r, ushort nx);

/* matrix */

short mmul(DATA *x1,short row1,short col1,DATA *x2,short row2,short
col2,DATA *r);
short mtrans(DATA *x, short row, short col, DATA *r);

/* trigonometric */

short sine (DATA *x, DATA *r, ushort nx);

/* miscellaneous */

short fltoq15(float *x, DATA *r, ushort nx);
short  q15tofl (DATA *x, float *r, ushort nx);


/* macro definition */

#define acorr(n1, n2, n3, n4, type) acorr_##type(n1, n2, n3, n4)
#define corr(n1, n2, n3, n4, n5, type) corr_##type(n1, n2, n3, n4, n5)
#define dummy(x,n,scale  cfft##n(x,scale) // macro for generic FFT look
#define cfft(x,n,scale)  dummy(x,n,scale)

#define dummy2(x,n,scale)rfft##n(x,scale) // macro for generic FFT look
#define rfft(x,n,scale)  dummy2(x,n,scale)

#define dummy3(x,n,scale) cifft##n(x,scale) // macro for generic IFFT
look
#define cifft(x,n,scale) dummy3(x,n,scale)

#define dummy4(x,n,scale) rifft##n(x,scale) // macro for generic IFFT
look
#define rifft(x,n,scale) dummy4(x,n,scale)

#endif
```

A.7 Tms320.h

```
#ifndef _TMS320
#define _TMS320

typedef unsigned int   uint;
typedef unsigned short ushort;
#define PASS –1


  typedef short DATA;
  typedef long LDATA;
  #define ABSVAL abs
  #define SHIFT15 >>15
  #define SHIFT1  /2
  #define ROUND 0x400
  #define DIV2 >>1

#endif
```

A.8 Test.h

```
/* Test for cfft: both */
#define NX 64
#define FNAME “t12”
#define MAXERROR 10

#pragma DATA_SECTION (x,“.input”)

DATA x[2*NX] ={
3627, –2284,
–2245, –5,
–7049, 1744,
6912, 1802,
9108, 1123,
–7037, 9666,
–1825, –7164,
1298, –4957,
–230, –719,
9222, –7479,
–6005, –3615,
2585, –7466,
3025, 2433,
6061, –5043,
–471, –2214,
–5935, –9432,
8033, –1470,
–7160, 8950,
–1794, –7376,
7713, –8157,
–6756, –8579,
–2693, –4939,
–7298, 5663,
–894, –3010,
–954, 6179,
8633, 3033,
–5695, 3592,
8178, –4997,
7217, –575,
119, 2008,
6351, 5117,
–755, 9027,
2655, –1213,
6494, 3780,
4044, 9743,
9088, 7025,
–4214, 749,
289, –7931,
–1719, 1534,
7531, –1199,
4595, 7385,
4313, 6014,
4131, 4834,
–9618, 7721,
500, –734,
```

```
–8696, 4268,
–221, 3354,
3641, –6009,
8333, 7318,
7800, 879,
–7216, –993,
9787, –5689,
–1080, –3685,
293, 7630,
–1205, –649,
6133, –2697,
–5770, 9982,
–6928, 2610,
2327, –9988,
–9982, 5467,
4547, –3616,
–1646, 3650,
3611, –5895,
6728, 4178,
};

#pragma DATA_SECTION (rtest,“.test”)

DATA rtest[2*NX] ={
–1, –1, /* 0 */
3861, 3861,  /* 1 */
3861, 3861,  /* 2 */
3861, 3861,  /* 3 */
0, 0,  /* 4 */
0,–1,  /* 5 */
0,–1,  /* 6 */
0,0   /* 7 */
0,–1,  /* 8 */
0,–1,  /* 9 */
0,–1,  /* 10 */
0,0,  /* 11 */
0,0,  /* 12 */
0,0,  /* 13 */
0,0,  /* 14 */
0,0,  /* 15 */
–1,0,  /* 16 */
0,0,  /* 17 */
–1,0,  /* 18 */
–1,0,  /* 19 */
–1,–1,  /* 20 */
–1,–1,  /* 21 */
0,0,  /* 22 */
–1,0,  /* 23 */
–1,0,  /* 24 */
–1,–1,  /* 25 */
0,0,  /* 26 */
–1,0,  /* 27 */
–1,0,  /* 28 */
–1,0,  /* 29 */
–1,–1,  /* 30 */
```

```
0,0,  /* 31 */
-1,-1,  /* 32 */
0,0,  /* 33 */
-1,0,  /* 34 */
-1,-1,  /* 35 */
0,-1,  /* 36 */
0,-1,  /* 37 */
0,0,  /* 38 */
-1,-1,  /* 39 */
0,-1,  /* 40 */
0,-1,  /* 41 */
-1,0,  /* 42 */
-1,-1,  /* 43 */
-1,-1,  /* 44 */
-1,-1,  /* 45 */
0,0,  /* 46 */
0,0,  /* 47 */
0,-1,  /* 48 */
0,0,  /* 49 */
0,0,  /* 50 */
0,0,  /* 51 */
0,0,  /* 52 */
0,0,  /* 53 */
-1,0,  /* 54 */
-1,0,  /* 55 */
-1,0,  /* 56 */
0,-1,  /* 57 */
0,0,  /* 58 */
-1,0,  /* 59 */
0,0,  /* 60 */
3861, 3861,  /* 61 */
3861, 3861,  /* 62 */
3861, 3861,  /* 63 */
};

DATA error;
```

A.9 Sintab.q15

```
;***********************************************************************
;  Filename:  sintab.q15
;  Version :  Prod1.00
;  Description:  twiddle table to include for CFFT
;
;  Copyright Texas instruments Inc, 1998
;-----------------------------------------------------------------------
;  Description: a separate sine table is provided for each stage to
increase
;  FFT speed. This is at the expense of an increased Data Memory
;  size.
;  Format: a 1/4–cycle sine values followed by a 1/2–cycle cosine
;  values for a total of (3/4 * FFTSIZE –1) values
;-----------------------------------------------------------------------
; Revision History:
;
; 1.00Beta  M. Christ/M. Chishtie. 1996, Original code
;
;***********************************************************************

     .sect   ".sintab"
TWIDSTRT
     .if N>8
; STAGE 4
sin4 .word 030fch     ;   22.500 ø
     .word 05a82h     ;   45.000 ø
     .word 07642h     ;   67.500 ø
     .word 07fffh     ;   90.000 ø
cos4 .word 07642h     ;  112.500 ø
     .word 05a82h     ;  135.000 ø
     .word 030fch     ;  157.500 ø
     .word 00000h     ;  180.000 ø
     .word 0cf04h     ;  202.500 ø
     .word 0a57eh     ;  225.000 ø
     .word 089beh     ;  247.500 ø
; Numbers in stage 4 = 11
     .endif
     .if N>16
; STAGE 5
sin5 .word 018f9h     ;   11.250 ø
     .word 030fch     ;   22.500 ø
     .word 0471dh     ;   33.750 ø
     .word 05a82h     ;   45.000 ø
     .word 06a6eh     ;   56.250 ø
     .word 07642h     ;   67.500 ø
     .word 07d8ah     ;   78.750 ø
     .word 07fffh     ;   90.000 ø
cos5 .word 07d8ah     ;  101.250 ø
     .word 07642h     ;  112.500 ø
     .word 06a6eh     ;  123.750 ø
     .word 05a82h     ;  135.000 ø
     .word 0471dh     ;  146.250 ø
     .word 030fch     ;  157.500 ø
```

```
      .word 018f9h     ;  168.750 ø
      .word 00000h     ;  180.000 ø
      .word 0e707h     ;  191.250 ø
      .word 0cf04h     ;  202.500 ø
      .word 0b8e3h     ;  213.750 ø
      .word 0a57eh     ;  225.000 ø
      .word 09592h     ;  236.250 ø
      .word 089beh     ;  247.500 ø
      .word 08276h     ;  258.750 ø
; Numbers in stage 5 = 23
      .endif
      .if N>32
; STAGE 6
sin6  .word 00c8ch     ;    5.625 ø
      .word 018f9h     ;   11.250 ø
      .word 02528h     ;   16.875 ø
      .word 030fch     ;   22.500 ø
      .word 03c57h     ;   28.125 ø
      .word 0471dh     ;   33.750 ø
      .word 05134h     ;   39.375 ø
      .word 05a82h     ;   45.000 ø
      .word 062f2h     ;   50.625 ø
      .word 06a6eh     ;   56.250 ø
      .word 070e3h     ;   61.875 ø
      .word 07642h     ;   67.500 ø
      .word 07a7dh     ;   73.125 ø
      .word 07d8ah     ;   78.750 ø
      .word 07f62h     ;   84.375 ø
      .word 07fffh     ;   90.000 ø
cos6  .word 07f62h     ;   95.625 ø
      .word 07d8ah     ;  101.250 ø
      .word 07a7dh     ;  106.875 ø
      .word 07642h     ;  112.500 ø
      .word 070e3h     ;  118.125 ø
      .word 06a6eh     ;  123.750 ø
      .word 062f2h     ;  129.375 ø
      .word 05a82h     ;  135.000 ø
      .word 05134h     ;  140.625 ø
      .word 0471dh     ;  146.250 ø
      .word 03c57h     ;  151.875 ø
      .word 030fch     ;  157.500 ø
      .word 02528h     ;  163.125 ø
      .word 018f9h     ;  168.750 ø
      .word 00c8ch     ;  174.375 ø
      .word 00000h     ;  180.000 ø
      .word 0f374h     ;  185.625 ø
      .word 0e707h     ;  191.250 ø
      .word 0dad8h     ;  196.875 ø
      .word 0cf04h     ;  202.500 ø
      .word 0c3a9h     ;  208.125 ø
      .word 0b8e3h     ;  213.750 ø
      .word 0aecch     ;  219.375 ø
      .word 0a57eh     ;  225.000 ø
      .word 09d0eh     ;  230.625 ø
      .word 09592h     ;  236.250 ø
```

```
        .word 08f1dh     ;  241.875 ø
        .word 089beh     ;  247.500 ø
        .word 08583h     ;  253.125 ø
        .word 08276h     ;  258.750 ø
        .word 0809eh     ;  264.375 ø
; Numbers in stage 6 = 47
        .endif
        .if N>64
; STAGE 7
sin7    .word 00648h     ;    2.812 ø
        .word 00c8ch     ;    5.625 ø
        .word 012c8h     ;    8.438 ø
        .word 018f9h     ;   11.250 ø
        .word 01f1ah     ;   14.062 ø
        .word 02528h     ;   16.875 ø
        .word 02b1fh     ;   19.688 ø
        .word 030fch     ;   22.500 ø
        .word 036bah     ;   25.312 ø
        .word 03c57h     ;   28.125 ø
        .word 041ceh     ;   30.938 ø
        .word 0471dh     ;   33.750 ø
        .word 04c40h     ;   36.562 ø
        .word 05134h     ;   39.375 ø
        .word 055f6h     ;   42.188 ø
        .word 05a82h     ;   45.000 ø
        .word 05ed7h     ;   47.812 ø
        .word 062f2h     ;   50.625 ø
        .word 066d0h     ;   53.438 ø
        .word 06a6eh     ;   56.250 ø
        .word 06dcah     ;   59.062 ø
        .word 070e3h     ;   61.875 ø
        .word 073b6h     ;   64.688 ø
        .word 07642h     ;   67.500 ø
        .word 07885h     ;   70.312 ø
        .word 07a7dh     ;   73.125 ø
        .word 07c2ah     ;   75.938 ø
        .word 07d8ah     ;   78.750 ø
        .word 07e9dh     ;   81.562 ø
        .word 07f62h     ;   84.375 ø
        .word 07fd9h     ;   87.188 ø
        .word 07fffh     ;   90.000 ø
cos7    .word 07fd9h     ;   92.812 ø
        .word 07f62h     ;   95.625 ø
        .word 07e9dh     ;   98.438 ø
        .word 07d8ah     ;  101.250 ø
        .word 07c2ah     ;  104.062 ø
        .word 07a7dh     ;  106.875 ø
        .word 07885h     ;  109.688 ø
        .word 07642h     ;  112.500 ø
        .word 073b6h     ;  115.312 ø
        .word 070e3h     ;  118.125 ø
        .word 06dcah     ;  120.938 ø
        .word 06a6eh     ;  123.750 ø
        .word 066d0h     ;  126.562 ø
        .word 062f2h     ;  129.375 ø
```

```
      .word 05ed7h     ;  132.188 ø
      .word 05a82h     ;  135.000 ø
      .word 055f6h     ;  137.812 ø
      .word 05134h     ;  140.625 ø
      .word 04c40h     ;  143.438 ø
      .word 0471dh     ;  146.250 ø
      .word 041ceh     ;  149.062 ø
      .word 03c57h     ;  151.875 ø
      .word 036bah     ;  154.688 ø
      .word 030fch     ;  157.500 ø
      .word 02b1fh     ;  160.312 ø
      .word 02528h     ;  163.125 ø
      .word 01f1ah     ;  165.938 ø
      .word 018f9h     ;  168.750 ø
      .word 012c8h     ;  171.562 ø
      .word 00c8ch     ;  174.375 ø
      .word 00648h     ;  177.188 ø
      .word 00000h     ;  180.000 ø
      .word 0f9b8h     ;  182.812 ø
      .word 0f374h     ;  185.625 ø
      .word 0ed38h     ;  188.438 ø
      .word 0e707h     ;  191.250 ø
      .word 0e0e6h     ;  194.062 ø
      .word 0dad8h     ;  196.875 ø
      .word 0d4e1h     ;  199.688 ø
      .word 0cf04h     ;  202.500 ø
      .word 0c946h     ;  205.312 ø
      .word 0c3a9h     ;  208.125 ø
      .word 0be32h     ;  210.938 ø
      .word 0b8e3h     ;  213.750 ø
      .word 0b3c0h     ;  216.562 ø
      .word 0aecch     ;  219.375 ø
      .word 0aa0ah     ;  222.188 ø
      .word 0a57eh     ;  225.000 ø
      .word 0a129h     ;  227.812 ø
      .word 09d0eh     ;  230.625 ø
      .word 09930h     ;  233.438 ø
      .word 09592h     ;  236.250 ø
      .word 09236h     ;  239.062 ø
      .word 08f1dh     ;  241.875 ø
      .word 08c4ah     ;  244.688 ø
      .word 089beh     ;  247.500 ø
      .word 0877bh     ;  250.312 ø
      .word 08583h     ;  253.125 ø
      .word 083d6h     ;  255.938 ø
      .word 08276h     ;  258.750 ø
      .word 08163h     ;  261.562 ø
      .word 0809eh     ;  264.375 ø
      .word 08027h     ;  267.188 ø
; Numbers in stage 7 = 95
      .endif
      .if N>128
; STAGE 8
sin8  .word 00324h     ;    1.406 ø
      .word 00648h     ;    2.812 ø
```

```
        .word 0096bh     ;    4.219 ø
        .word 00c8ch     ;    5.625 ø
        .word 00fabh     ;    7.031 ø
        .word 012c8h     ;    8.438 ø
        .word 015e2h     ;    9.844 ø
        .word 018f9h     ;   11.250 ø
        .word 01c0ch     ;   12.656 ø
        .word 01f1ah     ;   14.062 ø
        .word 02224h     ;   15.469 ø
        .word 02528h     ;   16.875 ø
        .word 02827h     ;   18.281 ø
        .word 02b1fh     ;   19.688 ø
        .word 02e11h     ;   21.094 ø
        .word 030fch     ;   22.500 ø
        .word 033dfh     ;   23.906 ø
        .word 036bah     ;   25.312 ø
        .word 0398dh     ;   26.719 ø
        .word 03c57h     ;   28.125 ø
        .word 03f17h     ;   29.531 ø
        .word 041ceh     ;   30.938 ø
        .word 0447bh     ;   32.344 ø
        .word 0471dh     ;   33.750 ø
        .word 049b4h     ;   35.156 ø
        .word 04c40h     ;   36.562 ø
        .word 04ec0h     ;   37.969 ø
        .word 05134h     ;   39.375 ø
        .word 0539bh     ;   40.781 ø
        .word 055f6h     ;   42.188 ø
        .word 05843h     ;   43.594 ø
        .word 05a82h     ;   45.000 ø
        .word 05cb4h     ;   46.406 ø
        .word 05ed7h     ;   47.812 ø
        .word 060ech     ;   49.219 ø
        .word 062f2h     ;   50.625 ø
        .word 064e9h     ;   52.031 ø
        .word 066d0h     ;   53.438 ø
        .word 068a7h     ;   54.844 ø
        .word 06a6eh     ;   56.250 ø
        .word 06c24h     ;   57.656 ø
        .word 06dcah     ;   59.062 ø
        .word 06f5fh     ;   60.469 ø
        .word 070e3h     ;   61.875 ø
        .word 07255h     ;   63.281 ø
        .word 073b6h     ;   64.688 ø
        .word 07505h     ;   66.094 ø
        .word 07642h     ;   67.500 ø
        .word 0776ch     ;   68.906 ø
        .word 07885h     ;   70.312 ø
        .word 0798ah     ;   71.719 ø
        .word 07a7dh     ;   73.125 ø
        .word 07b5dh     ;   74.531 ø
        .word 07c2ah     ;   75.938 ø
        .word 07ce4h     ;   77.344 ø
        .word 07d8ah     ;   78.750 ø
        .word 07e1eh     ;   80.156 ø
```

```
        .word 07e9dh     ;   81.562 ø
        .word 07f0ah     ;   82.969 ø
        .word 07f62h     ;   84.375 ø
        .word 07fa7h     ;   85.781 ø
        .word 07fd9h     ;   87.188 ø
        .word 07ff6h     ;   88.594 ø
        .word 07fffh     ;   90.000 ø
cos8    .word 07ff6h     ;   91.406 ø
        .word 07fd9h     ;   92.812 ø
        .word 07fa7h     ;   94.219 ø
        .word 07f62h     ;   95.625 ø
        .word 07f0ah     ;   97.031 ø
        .word 07e9dh     ;   98.438 ø
        .word 07e1eh     ;   99.844 ø
        .word 07d8ah     ;  101.250 ø
        .word 07ce4h     ;  102.656 ø
        .word 07c2ah     ;  104.062 ø
        .word 07b5dh     ;  105.469 ø
        .word 07a7dh     ;  106.875 ø
        .word 0798ah     ;  108.281 ø
        .word 07885h     ;  109.688 ø
        .word 0776ch     ;  111.094 ø
        .word 07642h     ;  112.500 ø
        .word 07505h     ;  113.906 ø
        .word 073b6h     ;  115.312 ø
        .word 07255h     ;  116.719 ø
        .word 070e3h     ;  118.125 ø
        .word 06f5fh     ;  119.531 ø
        .word 06dcah     ;  120.938 ø
        .word 06c24h     ;  122.344 ø
        .word 06a6eh     ;  123.750 ø
        .word 068a7h     ;  125.156 ø
        .word 066d0h     ;  126.562 ø
        .word 064e9h     ;  127.969 ø
        .word 062f2h     ;  129.375 ø
        .word 060ech     ;  130.781 ø
        .word 05ed7h     ;  132.188 ø
        .word 05cb4h     ;  133.594 ø
        .word 05a82h     ;  135.000 ø
        .word 05843h     ;  136.406 ø
        .word 055f6h     ;  137.812 ø
        .word 0539bh     ;  139.219 ø
        .word 05134h     ;  140.625 ø
        .word 04ec0h     ;  142.031 ø
        .word 04c40h     ;  143.438 ø
        .word 049b4h     ;  144.844 ø
        .word 0471dh     ;  146.250 ø
        .word 0447bh     ;  147.656 ø
        .word 041ceh     ;  149.062 ø
        .word 03f17h     ;  150.469 ø
        .word 03c57h     ;  151.875 ø
        .word 0398dh     ;  153.281 ø
        .word 036bah     ;  154.688 ø
        .word 033dfh     ;  156.094 ø
        .word 030fch     ;  157.500 ø
```

```
        .word 02e11h     ;  158.906 ø
        .word 02b1fh     ;  160.312 ø
        .word 02827h     ;  161.719 ø
        .word 02528h     ;  163.125 ø
        .word 02224h     ;  164.531 ø
        .word 01f1ah     ;  165.938 ø
        .word 01c0ch     ;  167.344 ø
        .word 018f9h     ;  168.750 ø
        .word 015e2h     ;  170.156 ø
        .word 012c8h     ;  171.562 ø
        .word 00fabh     ;  172.969 ø
        .word 00c8ch     ;  174.375 ø
        .word 0096bh     ;  175.781 ø
        .word 00648h     ;  177.188 ø
        .word 00324h     ;  178.594 ø
        .word 00000h     ;  180.000 ø
        .word 0fcdch     ;  181.406 ø
        .word 0f9b8h     ;  182.812 ø
        .word 0f695h     ;  184.219 ø
        .word 0f374h     ;  185.625 ø
        .word 0f055h     ;  187.031 ø
        .word 0ed38h     ;  188.438 ø
        .word 0ea1eh     ;  189.844 ø
        .word 0e707h     ;  191.250 ø
        .word 0e3f4h     ;  192.656 ø
        .word 0e0e6h     ;  194.062 ø
        .word 0dddch     ;  195.469 ø
        .word 0dad8h     ;  196.875 ø
        .word 0d7d9h     ;  198.281 ø
        .word 0d4e1h     ;  199.688 ø
        .word 0d1efh     ;  201.094 ø
        .word 0cf04h     ;  202.500 ø
        .word 0cc21h     ;  203.906 ø
        .word 0c946h     ;  205.312 ø
        .word 0c673h     ;  206.719 ø
        .word 0c3a9h     ;  208.125 ø
        .word 0c0e9h     ;  209.531 ø
        .word 0be32h     ;  210.938 ø
        .word 0bb85h     ;  212.344 ø
        .word 0b8e3h     ;  213.750 ø
        .word 0b64ch     ;  215.156 ø
        .word 0b3c0h     ;  216.562 ø
        .word 0b140h     ;  217.969 ø
        .word 0aecch     ;  219.375 ø
        .word 0ac65h     ;  220.781 ø
        .word 0aa0ah     ;  222.188 ø
        .word 0a7bdh     ;  223.594 ø
        .word 0a57eh     ;  225.000 ø
        .word 0a34ch     ;  226.406 ø
        .word 0a129h     ;  227.812 ø
        .word 09f14h     ;  229.219 ø
        .word 09d0eh     ;  230.625 ø
        .word 09b17h     ;  232.031 ø
        .word 09930h     ;  233.438 ø
        .word 09759h     ;  234.844 ø
```

```
      .word 09592h     ;  236.250 ø
      .word 093dch     ;  237.656 ø
      .word 09236h     ;  239.062 ø
      .word 090a1h     ;  240.469 ø
      .word 08f1dh     ;  241.875 ø
      .word 08dabh     ;  243.281 ø
      .word 08c4ah     ;  244.688 ø
      .word 08afbh     ;  246.094 ø
      .word 089beh     ;  247.500 ø
      .word 08894h     ;  248.906 ø
      .word 0877bh     ;  250.312 ø
      .word 08676h     ;  251.719 ø
      .word 08583h     ;  253.125 ø
      .word 084a3h     ;  254.531 ø
      .word 083d6h     ;  255.938 ø
      .word 0831ch     ;  257.344 ø
      .word 08276h     ;  258.750 ø
      .word 081e2h     ;  260.156 ø
      .word 08163h     ;  261.562 ø
      .word 080f6h     ;  262.969 ø
      .word 0809eh     ;  264.375 ø
      .word 08059h     ;  265.781 ø
      .word 08027h     ;  267.188 ø
      .word 0800ah     ;  268.594 ø
; Numbers in stage 8 = 191
      .endif
      .if N>256
; STAGE 9
sin9  .word 00192h     ;    0.703 ø
      .word 00324h     ;    1.406 ø
      .word 004b6h     ;    2.109 ø
      .word 00648h     ;    2.812 ø
      .word 007d9h     ;    3.516 ø
      .word 0096bh     ;    4.219 ø
      .word 00afbh     ;    4.922 ø
      .word 00c8ch     ;    5.625 ø
      .word 00e1ch     ;    6.328 ø
      .word 00fabh     ;    7.031 ø
      .word 0113ah     ;    7.734 ø
      .word 012c8h     ;    8.438 ø
      .word 01455h     ;    9.141 ø
      .word 015e2h     ;    9.844 ø
      .word 0176eh     ;   10.547 ø
      .word 018f9h     ;   11.250 ø
      .word 01a83h     ;   11.953 ø
      .word 01c0ch     ;   12.656 ø
      .word 01d93h     ;   13.359 ø
      .word 01f1ah     ;   14.062 ø
      .word 0209fh     ;   14.766 ø
      .word 02224h     ;   15.469 ø
      .word 023a7h     ;   16.172 ø
      .word 02528h     ;   16.875 ø
      .word 026a8h     ;   17.578 ø
      .word 02827h     ;   18.281 ø
      .word 029a4h     ;   18.984 ø
```

```
        .word 02b1fh     ;   19.688 ø
        .word 02c99h     ;   20.391 ø
        .word 02e11h     ;   21.094 ø
        .word 02f87h     ;   21.797 ø
        .word 030fch     ;   22.500 ø
        .word 0326eh     ;   23.203 ø
        .word 033dfh     ;   23.906 ø
        .word 0354eh     ;   24.609 ø
        .word 036bah     ;   25.312 ø
        .word 03825h     ;   26.016 ø
        .word 0398dh     ;   26.719 ø
        .word 03af3h     ;   27.422 ø
        .word 03c57h     ;   28.125 ø
        .word 03db8h     ;   28.828 ø
        .word 03f17h     ;   29.531 ø
        .word 04074h     ;   30.234 ø
        .word 041ceh     ;   30.938 ø
        .word 04326h     ;   31.641 ø
        .word 0447bh     ;   32.344 ø
        .word 045cdh     ;   33.047 ø
        .word 0471dh     ;   33.750 ø
        .word 0486ah     ;   34.453 ø
        .word 049b4h     ;   35.156 ø
        .word 04afbh     ;   35.859 ø
        .word 04c40h     ;   36.562 ø
        .word 04d81h     ;   37.266 ø
        .word 04ec0h     ;   37.969 ø
        .word 04ffbh     ;   38.672 ø
        .word 05134h     ;   39.375 ø
        .word 05269h     ;   40.078 ø
        .word 0539bh     ;   40.781 ø
        .word 054cah     ;   41.484 ø
        .word 055f6h     ;   42.188 ø
        .word 0571eh     ;   42.891 ø
        .word 05843h     ;   43.594 ø
        .word 05964h     ;   44.297 ø
        .word 05a82h     ;   45.000 ø
        .word 05b9dh     ;   45.703 ø
        .word 05cb4h     ;   46.406 ø
        .word 05dc8h     ;   47.109 ø
        .word 05ed7h     ;   47.812 ø
        .word 05fe4h     ;   48.516 ø
        .word 060ech     ;   49.219 ø
        .word 061f1h     ;   49.922 ø
        .word 062f2h     ;   50.625 ø
        .word 063efh     ;   51.328 ø
        .word 064e9h     ;   52.031 ø
        .word 065deh     ;   52.734 ø
        .word 066d0h     ;   53.438 ø
        .word 067bdh     ;   54.141 ø
        .word 068a7h     ;   54.844 ø
        .word 0698ch     ;   55.547 ø
        .word 06a6eh     ;   56.250 ø
        .word 06b4bh     ;   56.953 ø
        .word 06c24h     ;   57.656 ø
```

```
        .word 06cf9h     ;   58.359 ø
        .word 06dcah     ;   59.062 ø
        .word 06e97h     ;   59.766 ø
        .word 06f5fh     ;   60.469 ø
        .word 07023h     ;   61.172 ø
        .word 070e3h     ;   61.875 ø
        .word 0719eh     ;   62.578 ø
        .word 07255h     ;   63.281 ø
        .word 07308h     ;   63.984 ø
        .word 073b6h     ;   64.688 ø
        .word 07460h     ;   65.391 ø
        .word 07505h     ;   66.094 ø
        .word 075a6h     ;   66.797 ø
        .word 07642h     ;   67.500 ø
        .word 076d9h     ;   68.203 ø
        .word 0776ch     ;   68.906 ø
        .word 077fbh     ;   69.609 ø
        .word 07885h     ;   70.312 ø
        .word 0790ah     ;   71.016 ø
        .word 0798ah     ;   71.719 ø
        .word 07a06h     ;   72.422 ø
        .word 07a7dh     ;   73.125 ø
        .word 07aefh     ;   73.828 ø
        .word 07b5dh     ;   74.531 ø
        .word 07bc6h     ;   75.234 ø
        .word 07c2ah     ;   75.938 ø
        .word 07c89h     ;   76.641 ø
        .word 07ce4h     ;   77.344 ø
        .word 07d3ah     ;   78.047 ø
        .word 07d8ah     ;   78.750 ø
        .word 07dd6h     ;   79.453 ø
        .word 07e1eh     ;   80.156 ø
        .word 07e60h     ;   80.859 ø
        .word 07e9dh     ;   81.562 ø
        .word 07ed6h     ;   82.266 ø
        .word 07f0ah     ;   82.969 ø
        .word 07f38h     ;   83.672 ø
        .word 07f62h     ;   84.375 ø
        .word 07f87h     ;   85.078 ø
        .word 07fa7h     ;   85.781 ø
        .word 07fc2h     ;   86.484 ø
        .word 07fd9h     ;   87.188 ø
        .word 07feah     ;   87.891 ø
        .word 07ff6h     ;   88.594 ø
        .word 07ffeh     ;   89.297 ø
        .word 07fffh     ;   90.000 ø
cos9    .word 07ffeh     ;   90.703 ø
        .word 07ff6h     ;   91.406 ø
        .word 07feah     ;   92.109 ø
        .word 07fd9h     ;   92.812 ø
        .word 07fc2h     ;   93.516 ø
        .word 07fa7h     ;   94.219 ø
        .word 07f87h     ;   94.922 ø
        .word 07f62h     ;   95.625 ø
        .word 07f38h     ;   96.328 ø
```

```
        .word 07f0ah     ;   97.031 ø
        .word 07ed6h     ;   97.734 ø
        .word 07e9dh     ;   98.438 ø
        .word 07e60h     ;   99.141 ø
        .word 07e1eh     ;   99.844 ø
        .word 07dd6h     ;  100.547 ø
        .word 07d8ah     ;  101.250 ø
        .word 07d3ah     ;  101.953 ø
        .word 07ce4h     ;  102.656 ø
        .word 07c89h     ;  103.359 ø
        .word 07c2ah     ;  104.062 ø
        .word 07bc6h     ;  104.766 ø
        .word 07b5dh     ;  105.469 ø
        .word 07aefh     ;  106.172 ø
        .word 07a7dh     ;  106.875 ø
        .word 07a06h     ;  107.578 ø
        .word 0798ah     ;  108.281 ø
        .word 0790ah     ;  108.984 ø
        .word 07885h     ;  109.688 ø
        .word 077fbh     ;  110.391 ø
        .word 0776ch     ;  111.094 ø
        .word 076d9h     ;  111.797 ø
        .word 07642h     ;  112.500 ø
        .word 075a6h     ;  113.203 ø
        .word 07505h     ;  113.906 ø
        .word 07460h     ;  114.609 ø
        .word 073b6h     ;  115.312 ø
        .word 07308h     ;  116.016 ø
        .word 07255h     ;  116.719 ø
        .word 0719eh     ;  117.422 ø
        .word 070e3h     ;  118.125 ø
        .word 07023h     ;  118.828 ø
        .word 06f5fh     ;  119.531 ø
        .word 06e97h     ;  120.234 ø
        .word 06dcah     ;  120.938 ø
        .word 06cf9h     ;  121.641 ø
        .word 06c24h     ;  122.344 ø
        .word 06b4bh     ;  123.047 ø
        .word 06a6eh     ;  123.750 ø
        .word 0698ch     ;  124.453 ø
        .word 068a7h     ;  125.156 ø
        .word 067bdh     ;  125.859 ø
        .word 066d0h     ;  126.562 ø
        .word 065deh     ;  127.266 ø
        .word 064e9h     ;  127.969 ø
        .word 063efh     ;  128.672 ø
        .word 062f2h     ;  129.375 ø
        .word 061f1h     ;  130.078 ø
        .word 060ech     ;  130.781 ø
        .word 05fe4h     ;  131.484 ø
        .word 05ed7h     ;  132.188 ø
        .word 05dc8h     ;  132.891 ø
        .word 05cb4h     ;  133.594 ø
        .word 05b9dh     ;  134.297 ø
        .word 05a82h     ;  135.000 ø
```

```
.word 05964h     ;  135.703 ø
.word 05843h     ;  136.406 ø
.word 0571eh     ;  137.109 ø
.word 055f6h     ;  137.812 ø
.word 054cah     ;  138.516 ø
.word 0539bh     ;  139.219 ø
.word 05269h     ;  139.922 ø
.word 05134h     ;  140.625 ø
.word 04ffbh     ;  141.328 ø
.word 04ec0h     ;  142.031 ø
.word 04d81h     ;  142.734 ø
.word 04c40h     ;  143.438 ø
.word 04afbh     ;  144.141 ø
.word 049b4h     ;  144.844 ø
.word 0486ah     ;  145.547 ø
.word 0471dh     ;  146.250 ø
.word 045cdh     ;  146.953 ø
.word 0447bh     ;  147.656 ø
.word 04326h     ;  148.359 ø
.word 041ceh     ;  149.062 ø
.word 04074h     ;  149.766 ø
.word 03f17h     ;  150.469 ø
.word 03db8h     ;  151.172 ø
.word 03c57h     ;  151.875 ø
.word 03af3h     ;  152.578 ø
.word 0398dh     ;  153.281 ø
.word 03825h     ;  153.984 ø
.word 036bah     ;  154.688 ø
.word 0354eh     ;  155.391 ø
.word 033dfh     ;  156.094 ø
.word 0326eh     ;  156.797 ø
.word 030fch     ;  157.500 ø
.word 02f87h     ;  158.203 ø
.word 02e11h     ;  158.906 ø
.word 02c99h     ;  159.609 ø
.word 02b1fh     ;  160.312 ø
.word 029a4h     ;  161.016 ø
.word 02827h     ;  161.719 ø
.word 026a8h     ;  162.422 ø
.word 02528h     ;  163.125 ø
.word 023a7h     ;  163.828 ø
.word 02224h     ;  164.531 ø
.word 0209fh     ;  165.234 ø
.word 01f1ah     ;  165.938 ø
.word 01d93h     ;  166.641 ø
.word 01c0ch     ;  167.344 ø
.word 01a83h     ;  168.047 ø
.word 018f9h     ;  168.750 ø
.word 0176eh     ;  169.453 ø
.word 015e2h     ;  170.156 ø
.word 01455h     ;  170.859 ø
.word 012c8h     ;  171.562 ø
.word 0113ah     ;  172.266 ø
.word 00fabh     ;  172.969 ø
.word 00e1ch     ;  173.672 ø
```

```
        .word 00c8ch     ;  174.375 ø
        .word 00afbh     ;  175.078 ø
        .word 0096bh     ;  175.781 ø
        .word 007d9h     ;  176.484 ø
        .word 00648h     ;  177.188 ø
        .word 004b6h     ;  177.891 ø
        .word 00324h     ;  178.594 ø
        .word 00192h     ;  179.297 ø
        .word 00000h     ;  180.000 ø
        .word 0fe6eh     ;  180.703 ø
        .word 0fcdch     ;  181.406 ø
        .word 0fb4ah     ;  182.109 ø
        .word 0f9b8h     ;  182.812 ø
        .word 0f827h     ;  183.516 ø
        .word 0f695h     ;  184.219 ø
        .word 0f505h     ;  184.922 ø
        .word 0f374h     ;  185.625 ø
        .word 0f1e4h     ;  186.328 ø
        .word 0f055h     ;  187.031 ø
        .word 0eec6h     ;  187.734 ø
        .word 0ed38h     ;  188.438 ø
        .word 0ebabh     ;  189.141 ø
        .word 0ea1eh     ;  189.844 ø
        .word 0e892h     ;  190.547 ø
        .word 0e707h     ;  191.250 ø
        .word 0e57dh     ;  191.953 ø
        .word 0e3f4h     ;  192.656 ø
        .word 0e26dh     ;  193.359 ø
        .word 0e0e6h     ;  194.062 ø
        .word 0df61h     ;  194.766 ø
        .word 0dddch     ;  195.469 ø
        .word 0dc59h     ;  196.172 ø
        .word 0dad8h     ;  196.875 ø
        .word 0d958h     ;  197.578 ø
        .word 0d7d9h     ;  198.281 ø
        .word 0d65ch     ;  198.984 ø
        .word 0d4e1h     ;  199.688 ø
        .word 0d367h     ;  200.391 ø
        .word 0d1efh     ;  201.094 ø
        .word 0d079h     ;  201.797 ø
        .word 0cf04h     ;  202.500 ø
        .word 0cd92h     ;  203.203 ø
        .word 0cc21h     ;  203.906 ø
        .word 0cab2h     ;  204.609 ø
        .word 0c946h     ;  205.312 ø
        .word 0c7dbh     ;  206.016 ø
        .word 0c673h     ;  206.719 ø
        .word 0c50dh     ;  207.422 ø
        .word 0c3a9h     ;  208.125 ø
        .word 0c248h     ;  208.828 ø
        .word 0c0e9h     ;  209.531 ø
        .word 0bf8ch     ;  210.234 ø
        .word 0be32h     ;  210.938 ø
        .word 0bcdah     ;  211.641 ø
        .word 0bb85h     ;  212.344 ø
```

```
.word 0ba33h     ;  213.047 ø
.word 0b8e3h     ;  213.750 ø
.word 0b796h     ;  214.453 ø
.word 0b64ch     ;  215.156 ø
.word 0b505h     ;  215.859 ø
.word 0b3c0h     ;  216.562 ø
.word 0b27fh     ;  217.266 ø
.word 0b140h     ;  217.969 ø
.word 0b005h     ;  218.672 ø
.word 0aecch     ;  219.375 ø
.word 0ad97h     ;  220.078 ø
.word 0ac65h     ;  220.781 ø
.word 0ab36h     ;  221.484 ø
.word 0aa0ah     ;  222.188 ø
.word 0a8e2h     ;  222.891 ø
.word 0a7bdh     ;  223.594 ø
.word 0a69ch     ;  224.297 ø
.word 0a57eh     ;  225.000 ø
.word 0a463h     ;  225.703 ø
.word 0a34ch     ;  226.406 ø
.word 0a238h     ;  227.109 ø
.word 0a129h     ;  227.812 ø
.word 0a01ch     ;  228.516 ø
.word 09f14h     ;  229.219 ø
.word 09e0fh     ;  229.922 ø
.word 09d0eh     ;  230.625 ø
.word 09c11h     ;  231.328 ø
.word 09b17h     ;  232.031 ø
.word 09a22h     ;  232.734 ø
.word 09930h     ;  233.438 ø
.word 09843h     ;  234.141 ø
.word 09759h     ;  234.844 ø
.word 09674h     ;  235.547 ø
.word 09592h     ;  236.250 ø
.word 094b5h     ;  236.953 ø
.word 093dch     ;  237.656 ø
.word 09307h     ;  238.359 ø
.word 09236h     ;  239.062 ø
.word 09169h     ;  239.766 ø
.word 090a1h     ;  240.469 ø
.word 08fddh     ;  241.172 ø
.word 08f1dh     ;  241.875 ø
.word 08e62h     ;  242.578 ø
.word 08dabh     ;  243.281 ø
.word 08cf8h     ;  243.984 ø
.word 08c4ah     ;  244.688 ø
.word 08ba0h     ;  245.391 ø
.word 08afbh     ;  246.094 ø
.word 08a5ah     ;  246.797 ø
.word 089beh     ;  247.500 ø
.word 08927h     ;  248.203 ø
.word 08894h     ;  248.906 ø
.word 08805h     ;  249.609 ø
.word 0877bh     ;  250.312 ø
.word 086f6h     ;  251.016 ø
```

```
      .word 08676h     ;  251.719 ø
      .word 085fah     ;  252.422 ø
      .word 08583h     ;  253.125 ø
      .word 08511h     ;  253.828 ø
      .word 084a3h     ;  254.531 ø
      .word 0843ah     ;  255.234 ø
      .word 083d6h     ;  255.938 ø
      .word 08377h     ;  256.641 ø
      .word 0831ch     ;  257.344 ø
      .word 082c6h     ;  258.047 ø
      .word 08276h     ;  258.750 ø
      .word 0822ah     ;  259.453 ø
      .word 081e2h     ;  260.156 ø
      .word 081a0h     ;  260.859 ø
      .word 08163h     ;  261.562 ø
      .word 0812ah     ;  262.266 ø
      .word 080f6h     ;  262.969 ø
      .word 080c8h     ;  263.672 ø
      .word 0809eh     ;  264.375 ø
      .word 08079h     ;  265.078 ø
      .word 08059h     ;  265.781 ø
      .word 0803eh     ;  266.484 ø
      .word 08027h     ;  267.188 ø
      .word 08016h     ;  267.891 ø
      .word 0800ah     ;  268.594 ø
      .word 08002h     ;  269.297 ø
; Numbers in stage 9 = 383
      .endif
      .if N>512
; STAGE 10
      .def SINE10
SINE10 .set $
sina .word 000c9h      ;    0.352 ø
      .word 00192h     ;    0.703 ø
      .word 0025bh     ;    1.055 ø
      .word 00324h     ;    1.406 ø
      .word 003edh     ;    1.758 ø
      .word 004b6h     ;    2.109 ø
      .word 0057fh     ;    2.461 ø
      .word 00648h     ;    2.812 ø
      .word 00711h     ;    3.164 ø
      .word 007d9h     ;    3.516 ø
      .word 008a2h     ;    3.867 ø
      .word 0096bh     ;    4.219 ø
      .word 00a33h     ;    4.570 ø
      .word 00afbh     ;    4.922 ø
      .word 00bc4h     ;    5.273 ø
      .word 00c8ch     ;    5.625 ø
      .word 00d54h     ;    5.977 ø
      .word 00e1ch     ;    6.328 ø
      .word 00ee4h     ;    6.680 ø
      .word 00fabh     ;    7.031 ø
      .word 01073h     ;    7.383 ø
      .word 0113ah     ;    7.734 ø
      .word 01201h     ;    8.086 ø
```

```
.word 012c8h     ;    8.438 ø
.word 0138fh     ;    8.789 ø
.word 01455h     ;    9.141 ø
.word 0151ch     ;    9.492 ø
.word 015e2h     ;    9.844 ø
.word 016a8h     ;   10.195 ø
.word 0176eh     ;   10.547 ø
.word 01833h     ;   10.898 ø
.word 018f9h     ;   11.250 ø
.word 019beh     ;   11.602 ø
.word 01a83h     ;   11.953 ø
.word 01b47h     ;   12.305 ø
.word 01c0ch     ;   12.656 ø
.word 01cd0h     ;   13.008 ø
.word 01d93h     ;   13.359 ø
.word 01e57h     ;   13.711 ø
.word 01f1ah     ;   14.062 ø
.word 01fddh     ;   14.414 ø
.word 0209fh     ;   14.766 ø
.word 02162h     ;   15.117 ø
.word 02224h     ;   15.469 ø
.word 022e5h     ;   15.820 ø
.word 023a7h     ;   16.172 ø
.word 02467h     ;   16.523 ø
.word 02528h     ;   16.875 ø
.word 025e8h     ;   17.227 ø
.word 026a8h     ;   17.578 ø
.word 02768h     ;   17.930 ø
.word 02827h     ;   18.281 ø
.word 028e5h     ;   18.633 ø
.word 029a4h     ;   18.984 ø
.word 02a62h     ;   19.336 ø
.word 02b1fh     ;   19.688 ø
.word 02bdch     ;   20.039 ø
.word 02c99h     ;   20.391 ø
.word 02d55h     ;   20.742 ø
.word 02e11h     ;   21.094 ø
.word 02ecch     ;   21.445 ø
.word 02f87h     ;   21.797 ø
.word 03042h     ;   22.148 ø
.word 030fch     ;   22.500 ø
.word 031b5h     ;   22.852 ø
.word 0326eh     ;   23.203 ø
.word 03327h     ;   23.555 ø
.word 033dfh     ;   23.906 ø
.word 03497h     ;   24.258 ø
.word 0354eh     ;   24.609 ø
.word 03604h     ;   24.961 ø
.word 036bah     ;   25.312 ø
.word 03770h     ;   25.664 ø
.word 03825h     ;   26.016 ø
.word 038d9h     ;   26.367 ø
.word 0398dh     ;   26.719 ø
.word 03a40h     ;   27.070 ø
.word 03af3h     ;   27.422 ø
```

```
        .word 03ba5h     ;   27.773 ø
        .word 03c57h     ;   28.125 ø
        .word 03d08h     ;   28.477 ø
        .word 03db8h     ;   28.828 ø
        .word 03e68h     ;   29.180 ø
        .word 03f17h     ;   29.531 ø
        .word 03fc6h     ;   29.883 ø
        .word 04074h     ;   30.234 ø
        .word 04121h     ;   30.586 ø
        .word 041ceh     ;   30.938 ø
        .word 0427ah     ;   31.289 ø
        .word 04326h     ;   31.641 ø
        .word 043d1h     ;   31.992 ø
        .word 0447bh     ;   32.344 ø
        .word 04524h     ;   32.695 ø
        .word 045cdh     ;   33.047 ø
        .word 04675h     ;   33.398 ø
        .word 0471dh     ;   33.750 ø
        .word 047c4h     ;   34.102 ø
        .word 0486ah     ;   34.453 ø
        .word 0490fh     ;   34.805 ø
        .word 049b4h     ;   35.156 ø
        .word 04a58h     ;   35.508 ø
        .word 04afbh     ;   35.859 ø
        .word 04b9eh     ;   36.211 ø
        .word 04c40h     ;   36.562 ø
        .word 04ce1h     ;   36.914 ø
        .word 04d81h     ;   37.266 ø
        .word 04e21h     ;   37.617 ø
        .word 04ec0h     ;   37.969 ø
        .word 04f5eh     ;   38.320 ø
        .word 04ffbh     ;   38.672 ø
        .word 05098h     ;   39.023 ø
        .word 05134h     ;   39.375 ø
        .word 051cfh     ;   39.727 ø
        .word 05269h     ;   40.078 ø
        .word 05303h     ;   40.430 ø
        .word 0539bh     ;   40.781 ø
        .word 05433h     ;   41.133 ø
        .word 054cah     ;   41.484 ø
        .word 05560h     ;   41.836 ø
        .word 055f6h     ;   42.188 ø
        .word 0568ah     ;   42.539 ø
        .word 0571eh     ;   42.891 ø
        .word 057b1h     ;   43.242 ø
        .word 05843h     ;   43.594 ø
        .word 058d4h     ;   43.945 ø
        .word 05964h     ;   44.297 ø
        .word 059f4h     ;   44.648 ø
        .word 05a82h     ;   45.000 ø
        .word 05b10h     ;   45.352 ø
        .word 05b9dh     ;   45.703 ø
        .word 05c29h     ;   46.055 ø
        .word 05cb4h     ;   46.406 ø
        .word 05d3eh     ;   46.758 ø
```

```
.word 05dc8h     ;   47.109 ø
.word 05e50h     ;   47.461 ø
.word 05ed7h     ;   47.812 ø
.word 05f5eh     ;   48.164 ø
.word 05fe4h     ;   48.516 ø
.word 06068h     ;   48.867 ø
.word 060ech     ;   49.219 ø
.word 0616fh     ;   49.570 ø
.word 061f1h     ;   49.922 ø
.word 06272h     ;   50.273 ø
.word 062f2h     ;   50.625 ø
.word 06371h     ;   50.977 ø
.word 063efh     ;   51.328 ø
.word 0646ch     ;   51.680 ø
.word 064e9h     ;   52.031 ø
.word 06564h     ;   52.383 ø
.word 065deh     ;   52.734 ø
.word 06657h     ;   53.086 ø
.word 066d0h     ;   53.438 ø
.word 06747h     ;   53.789 ø
.word 067bdh     ;   54.141 ø
.word 06832h     ;   54.492 ø
.word 068a7h     ;   54.844 ø
.word 0691ah     ;   55.195 ø
.word 0698ch     ;   55.547 ø
.word 069fdh     ;   55.898 ø
.word 06a6eh     ;   56.250 ø
.word 06addh     ;   56.602 ø
.word 06b4bh     ;   56.953 ø
.word 06bb8h     ;   57.305 ø
.word 06c24h     ;   57.656 ø
.word 06c8fh     ;   58.008 ø
.word 06cf9h     ;   58.359 ø
.word 06d62h     ;   58.711 ø
.word 06dcah     ;   59.062 ø
.word 06e31h     ;   59.414 ø
.word 06e97h     ;   59.766 ø
.word 06efbh     ;   60.117 ø
.word 06f5fh     ;   60.469 ø
.word 06fc2h     ;   60.820 ø
.word 07023h     ;   61.172 ø
.word 07083h     ;   61.523 ø
.word 070e3h     ;   61.875 ø
.word 07141h     ;   62.227 ø
.word 0719eh     ;   62.578 ø
.word 071fah     ;   62.930 ø
.word 07255h     ;   63.281 ø
.word 072afh     ;   63.633 ø
.word 07308h     ;   63.984 ø
.word 0735fh     ;   64.336 ø
.word 073b6h     ;   64.688 ø
.word 0740bh     ;   65.039 ø
.word 07460h     ;   65.391 ø
.word 074b3h     ;   65.742 ø
.word 07505h     ;   66.094 ø
```

```
        .word 07556h     ;   66.445 ø
        .word 075a6h     ;   66.797 ø
        .word 075f4h     ;   67.148 ø
        .word 07642h     ;   67.500 ø
        .word 0768eh     ;   67.852 ø
        .word 076d9h     ;   68.203 ø
        .word 07723h     ;   68.555 ø
        .word 0776ch     ;   68.906 ø
        .word 077b4h     ;   69.258 ø
        .word 077fbh     ;   69.609 ø
        .word 07840h     ;   69.961 ø
        .word 07885h     ;   70.312 ø
        .word 078c8h     ;   70.664 ø
        .word 0790ah     ;   71.016 ø
        .word 0794ah     ;   71.367 ø
        .word 0798ah     ;   71.719 ø
        .word 079c9h     ;   72.070 ø
        .word 07a06h     ;   72.422 ø
        .word 07a42h     ;   72.773 ø
        .word 07a7dh     ;   73.125 ø
        .word 07ab7h     ;   73.477 ø
        .word 07aefh     ;   73.828 ø
        .word 07b27h     ;   74.180 ø
        .word 07b5dh     ;   74.531 ø
        .word 07b92h     ;   74.883 ø
        .word 07bc6h     ;   75.234 ø
        .word 07bf9h     ;   75.586 ø
        .word 07c2ah     ;   75.938 ø
        .word 07c5ah     ;   76.289 ø
        .word 07c89h     ;   76.641 ø
        .word 07cb7h     ;   76.992 ø
        .word 07ce4h     ;   77.344 ø
        .word 07d0fh     ;   77.695 ø
        .word 07d3ah     ;   78.047 ø
        .word 07d63h     ;   78.398 ø
        .word 07d8ah     ;   78.750 ø
        .word 07db1h     ;   79.102 ø
        .word 07dd6h     ;   79.453 ø
        .word 07dfbh     ;   79.805 ø
        .word 07e1eh     ;   80.156 ø
        .word 07e3fh     ;   80.508 ø
        .word 07e60h     ;   80.859 ø
        .word 07e7fh     ;   81.211 ø
        .word 07e9dh     ;   81.562 ø
        .word 07ebah     ;   81.914 ø
        .word 07ed6h     ;   82.266 ø
        .word 07ef0h     ;   82.617 ø
        .word 07f0ah     ;   82.969 ø
        .word 07f22h     ;   83.320 ø
        .word 07f38h     ;   83.672 ø
        .word 07f4eh     ;   84.023 ø
        .word 07f62h     ;   84.375 ø
        .word 07f75h     ;   84.727 ø
        .word 07f87h     ;   85.078 ø
        .word 07f98h     ;   85.430 ø
```

```
        .word 07fa7h     ;   85.781 ø
        .word 07fb5h     ;   86.133 ø
        .word 07fc2h     ;   86.484 ø
        .word 07fceh     ;   86.836 ø
        .word 07fd9h     ;   87.188 ø
        .word 07fe2h     ;   87.539 ø
        .word 07feah     ;   87.891 ø
        .word 07ff1h     ;   88.242 ø
        .word 07ff6h     ;   88.594 ø
        .word 07ffah     ;   88.945 ø
        .word 07ffeh     ;   89.297 ø
        .word 07fffh     ;   89.648 ø
        .word 07fffh     ;   90.000 ø
cosa    .word 07fffh     ;   90.352 ø
        .word 07ffeh     ;   90.703 ø
        .word 07ffah     ;   91.055 ø
        .word 07ff6h     ;   91.406 ø
        .word 07ff1h     ;   91.758 ø
        .word 07feah     ;   92.109 ø
        .word 07fe2h     ;   92.461 ø
        .word 07fd9h     ;   92.812 ø
        .word 07fceh     ;   93.164 ø
        .word 07fc2h     ;   93.516 ø
        .word 07fb5h     ;   93.867 ø
        .word 07fa7h     ;   94.219 ø
        .word 07f98h     ;   94.570 ø
        .word 07f87h     ;   94.922 ø
        .word 07f75h     ;   95.273 ø
        .word 07f62h     ;   95.625 ø
        .word 07f4eh     ;   95.977 ø
        .word 07f38h     ;   96.328 ø
        .word 07f22h     ;   96.680 ø
        .word 07f0ah     ;   97.031 ø
        .word 07ef0h     ;   97.383 ø
        .word 07ed6h     ;   97.734 ø
        .word 07ebah     ;   98.086 ø
        .word 07e9dh     ;   98.438 ø
        .word 07e7fh     ;   98.789 ø
        .word 07e60h     ;   99.141 ø
        .word 07e3fh     ;   99.492 ø
        .word 07e1eh     ;   99.844 ø
        .word 07dfbh     ;  100.195 ø
        .word 07dd6h     ;  100.547 ø
        .word 07db1h     ;  100.898 ø
        .word 07d8ah     ;  101.250 ø
        .word 07d63h     ;  101.602 ø
        .word 07d3ah     ;  101.953 ø
        .word 07d0fh     ;  102.305 ø
        .word 07ce4h     ;  102.656 ø
        .word 07cb7h     ;  103.008 ø
        .word 07c89h     ;  103.359 ø
        .word 07c5ah     ;  103.711 ø
        .word 07c2ah     ;  104.062 ø
        .word 07bf9h     ;  104.414 ø
        .word 07bc6h     ;  104.766 ø
```

```
        .word 07b92h     ;  105.117 ø
        .word 07b5dh     ;  105.469 ø
        .word 07b27h     ;  105.820 ø
        .word 07aefh     ;  106.172 ø
        .word 07ab7h     ;  106.523 ø
        .word 07a7dh     ;  106.875 ø
        .word 07a42h     ;  107.227 ø
        .word 07a06h     ;  107.578 ø
        .word 079c9h     ;  107.930 ø
        .word 0798ah     ;  108.281 ø
        .word 0794ah     ;  108.633 ø
        .word 0790ah     ;  108.984 ø
        .word 078c8h     ;  109.336 ø
        .word 07885h     ;  109.688 ø
        .word 07840h     ;  110.039 ø
        .word 077fbh     ;  110.391 ø
        .word 077b4h     ;  110.742 ø
        .word 0776ch     ;  111.094 ø
        .word 07723h     ;  111.445 ø
        .word 076d9h     ;  111.797 ø
        .word 0768eh     ;  112.148 ø
        .word 07642h     ;  112.500 ø
        .word 075f4h     ;  112.852 ø
        .word 075a6h     ;  113.203 ø
        .word 07556h     ;  113.555 ø
        .word 07505h     ;  113.906 ø
        .word 074b3h     ;  114.258 ø
        .word 07460h     ;  114.609 ø
        .word 0740bh     ;  114.961 ø
        .word 073b6h     ;  115.312 ø
        .word 0735fh     ;  115.664 ø
        .word 07308h     ;  116.016 ø
        .word 072afh     ;  116.367 ø
        .word 07255h     ;  116.719 ø
        .word 071fah     ;  117.070 ø
        .word 0719eh     ;  117.422 ø
        .word 07141h     ;  117.773 ø
        .word 070e3h     ;  118.125 ø
        .word 07083h     ;  118.477 ø
        .word 07023h     ;  118.828 ø
        .word 06fc2h     ;  119.180 ø
        .word 06f5fh     ;  119.531 ø
        .word 06efbh     ;  119.883 ø
        .word 06e97h     ;  120.234 ø
        .word 06e31h     ;  120.586 ø
        .word 06dcah     ;  120.938 ø
        .word 06d62h     ;  121.289 ø
        .word 06cf9h     ;  121.641 ø
        .word 06c8fh     ;  121.992 ø
        .word 06c24h     ;  122.344 ø
        .word 06bb8h     ;  122.695 ø
        .word 06b4bh     ;  123.047 ø
        .word 06addh     ;  123.398 ø
        .word 06a6eh     ;  123.750 ø
        .word 069fdh     ;  124.102 ø
```

```
.word 0698ch     ;  124.453 ø
.word 0691ah     ;  124.805 ø
.word 068a7h     ;  125.156 ø
.word 06832h     ;  125.508 ø
.word 067bdh     ;  125.859 ø
.word 06747h     ;  126.211 ø
.word 066d0h     ;  126.562 ø
.word 06657h     ;  126.914 ø
.word 065deh     ;  127.266 ø
.word 06564h     ;  127.617 ø
.word 064e9h     ;  127.969 ø
.word 0646ch     ;  128.320 ø
.word 063efh     ;  128.672 ø
.word 06371h     ;  129.023 ø
.word 062f2h     ;  129.375 ø
.word 06272h     ;  129.727 ø
.word 061f1h     ;  130.078 ø
.word 0616fh     ;  130.430 ø
.word 060ech     ;  130.781 ø
.word 06068h     ;  131.133 ø
.word 05fe4h     ;  131.484 ø
.word 05f5eh     ;  131.836 ø
.word 05ed7h     ;  132.188 ø
.word 05e50h     ;  132.539 ø
.word 05dc8h     ;  132.891 ø
.word 05d3eh     ;  133.242 ø
.word 05cb4h     ;  133.594 ø
.word 05c29h     ;  133.945 ø
.word 05b9dh     ;  134.297 ø
.word 05b10h     ;  134.648 ø
.word 05a82h     ;  135.000 ø
.word 059f4h     ;  135.352 ø
.word 05964h     ;  135.703 ø
.word 058d4h     ;  136.055 ø
.word 05843h     ;  136.406 ø
.word 057b1h     ;  136.758 ø
.word 0571eh     ;  137.109 ø
.word 0568ah     ;  137.461 ø
.word 055f6h     ;  137.812 ø
.word 05560h     ;  138.164 ø
.word 054cah     ;  138.516 ø
.word 05433h     ;  138.867 ø
.word 0539bh     ;  139.219 ø
.word 05303h     ;  139.570 ø
.word 05269h     ;  139.922 ø
.word 051cfh     ;  140.273 ø
.word 05134h     ;  140.625 ø
.word 05098h     ;  140.977 ø
.word 04ffbh     ;  141.328 ø
.word 04f5eh     ;  141.680 ø
.word 04ec0h     ;  142.031 ø
.word 04e21h     ;  142.383 ø
.word 04d81h     ;  142.734 ø
.word 04ce1h     ;  143.086 ø
.word 04c40h     ;  143.438 ø
```

```
        .word 04b9eh     ;  143.789 ø
        .word 04afbh     ;  144.141 ø
        .word 04a58h     ;  144.492 ø
        .word 049b4h     ;  144.844 ø
        .word 0490fh     ;  145.195 ø
        .word 0486ah     ;  145.547 ø
        .word 047c4h     ;  145.898 ø
        .word 0471dh     ;  146.250 ø
        .word 04675h     ;  146.602 ø
        .word 045cdh     ;  146.953 ø
        .word 04524h     ;  147.305 ø
        .word 0447bh     ;  147.656 ø
        .word 043d1h     ;  148.008 ø
        .word 04326h     ;  148.359 ø
        .word 0427ah     ;  148.711 ø
        .word 041ceh     ;  149.062 ø
        .word 04121h     ;  149.414 ø
        .word 04074h     ;  149.766 ø
        .word 03fc6h     ;  150.117 ø
        .word 03f17h     ;  150.469 ø
        .word 03e68h     ;  150.820 ø
        .word 03db8h     ;  151.172 ø
        .word 03d08h     ;  151.523 ø
        .word 03c57h     ;  151.875 ø
        .word 03ba5h     ;  152.227 ø
        .word 03af3h     ;  152.578 ø
        .word 03a40h     ;  152.930 ø
        .word 0398dh     ;  153.281 ø
        .word 038d9h     ;  153.633 ø
        .word 03825h     ;  153.984 ø
        .word 03770h     ;  154.336 ø
        .word 036bah     ;  154.688 ø
        .word 03604h     ;  155.039 ø
        .word 0354eh     ;  155.391 ø
        .word 03497h     ;  155.742 ø
        .word 033dfh     ;  156.094 ø
        .word 03327h     ;  156.445 ø
        .word 0326eh     ;  156.797 ø
        .word 031b5h     ;  157.148 ø
        .word 030fch     ;  157.500 ø
        .word 03042h     ;  157.852 ø
        .word 02f87h     ;  158.203 ø
        .word 02ecch     ;  158.555 ø
        .word 02e11h     ;  158.906 ø
        .word 02d55h     ;  159.258 ø
        .word 02c99h     ;  159.609 ø
        .word 02bdch     ;  159.961 ø
        .word 02b1fh     ;  160.312 ø
        .word 02a62h     ;  160.664 ø
        .word 029a4h     ;  161.016 ø
        .word 028e5h     ;  161.367 ø
        .word 02827h     ;  161.719 ø
        .word 02768h     ;  162.070 ø
        .word 026a8h     ;  162.422 ø
        .word 025e8h     ;  162.773 ø
```

```
.word 02528h     ;  163.125 ø
.word 02467h     ;  163.477 ø
.word 023a7h     ;  163.828 ø
.word 022e5h     ;  164.180 ø
.word 02224h     ;  164.531 ø
.word 02162h     ;  164.883 ø
.word 0209fh     ;  165.234 ø
.word 01fddh     ;  165.586 ø
.word 01f1ah     ;  165.938 ø
.word 01e57h     ;  166.289 ø
.word 01d93h     ;  166.641 ø
.word 01cd0h     ;  166.992 ø
.word 01c0ch     ;  167.344 ø
.word 01b47h     ;  167.695 ø
.word 01a83h     ;  168.047 ø
.word 019beh     ;  168.398 ø
.word 018f9h     ;  168.750 ø
.word 01833h     ;  169.102 ø
.word 0176eh     ;  169.453 ø
.word 016a8h     ;  169.805 ø
.word 015e2h     ;  170.156 ø
.word 0151ch     ;  170.508 ø
.word 01455h     ;  170.859 ø
.word 0138fh     ;  171.211 ø
.word 012c8h     ;  171.562 ø
.word 01201h     ;  171.914 ø
.word 0113ah     ;  172.266 ø
.word 01073h     ;  172.617 ø
.word 00fabh     ;  172.969 ø
.word 00ee4h     ;  173.320 ø
.word 00e1ch     ;  173.672 ø
.word 00d54h     ;  174.023 ø
.word 00c8ch     ;  174.375 ø
.word 00bc4h     ;  174.727 ø
.word 00afbh     ;  175.078 ø
.word 00a33h     ;  175.430 ø
.word 0096bh     ;  175.781 ø
.word 008a2h     ;  176.133 ø
.word 007d9h     ;  176.484 ø
.word 00711h     ;  176.836 ø
.word 00648h     ;  177.188 ø
.word 0057fh     ;  177.539 ø
.word 004b6h     ;  177.891 ø
.word 003edh     ;  178.242 ø
.word 00324h     ;  178.594 ø
.word 0025bh     ;  178.945 ø
.word 00192h     ;  179.297 ø
.word 000c9h     ;  179.648 ø
.word 00000h     ;  180.000 ø
.word 0ff37h     ;  180.352 ø
.word 0fe6eh     ;  180.703 ø
.word 0fda5h     ;  181.055 ø
.word 0fcdch     ;  181.406 ø
.word 0fc13h     ;  181.758 ø
.word 0fb4ah     ;  182.109 ø
```

```
        .word 0fa81h     ;  182.461 ø
        .word 0f9b8h     ;  182.812 ø
        .word 0f8efh     ;  183.164 ø
        .word 0f827h     ;  183.516 ø
        .word 0f75eh     ;  183.867 ø
        .word 0f695h     ;  184.219 ø
        .word 0f5cdh     ;  184.570 ø
        .word 0f505h     ;  184.922 ø
        .word 0f43ch     ;  185.273 ø
        .word 0f374h     ;  185.625 ø
        .word 0f2ach     ;  185.977 ø
        .word 0f1e4h     ;  186.328 ø
        .word 0f11ch     ;  186.680 ø
        .word 0f055h     ;  187.031 ø
        .word 0ef8dh     ;  187.383 ø
        .word 0eec6h     ;  187.734 ø
        .word 0edffh     ;  188.086 ø
        .word 0ed38h     ;  188.438 ø
        .word 0ec71h     ;  188.789 ø
        .word 0ebabh     ;  189.141 ø
        .word 0eae4h     ;  189.492 ø
        .word 0ea1eh     ;  189.844 ø
        .word 0e958h     ;  190.195 ø
        .word 0e892h     ;  190.547 ø
        .word 0e7cdh     ;  190.898 ø
        .word 0e707h     ;  191.250 ø
        .word 0e642h     ;  191.602 ø
        .word 0e57dh     ;  191.953 ø
        .word 0e4b9h     ;  192.305 ø
        .word 0e3f4h     ;  192.656 ø
        .word 0e330h     ;  193.008 ø
        .word 0e26dh     ;  193.359 ø
        .word 0e1a9h     ;  193.711 ø
        .word 0e0e6h     ;  194.062 ø
        .word 0e023h     ;  194.414 ø
        .word 0df61h     ;  194.766 ø
        .word 0de9eh     ;  195.117 ø
        .word 0dddch     ;  195.469 ø
        .word 0dd1bh     ;  195.820 ø
        .word 0dc59h     ;  196.172 ø
        .word 0db99h     ;  196.523 ø
        .word 0dad8h     ;  196.875 ø
        .word 0da18h     ;  197.227 ø
        .word 0d958h     ;  197.578 ø
        .word 0d898h     ;  197.930 ø
        .word 0d7d9h     ;  198.281 ø
        .word 0d71bh     ;  198.633 ø
        .word 0d65ch     ;  198.984 ø
        .word 0d59eh     ;  199.336 ø
        .word 0d4e1h     ;  199.688 ø
        .word 0d424h     ;  200.039 ø
        .word 0d367h     ;  200.391 ø
        .word 0d2abh     ;  200.742 ø
        .word 0d1efh     ;  201.094 ø
        .word 0d134h     ;  201.445 ø
```

```
.word 0d079h     ;  201.797 ø
.word 0cfbeh     ;  202.148 ø
.word 0cf04h     ;  202.500 ø
.word 0ce4bh     ;  202.852 ø
.word 0cd92h     ;  203.203 ø
.word 0ccd9h     ;  203.555 ø
.word 0cc21h     ;  203.906 ø
.word 0cb69h     ;  204.258 ø
.word 0cab2h     ;  204.609 ø
.word 0c9fch     ;  204.961 ø
.word 0c946h     ;  205.312 ø
.word 0c890h     ;  205.664 ø
.word 0c7dbh     ;  206.016 ø
.word 0c727h     ;  206.367 ø
.word 0c673h     ;  206.719 ø
.word 0c5c0h     ;  207.070 ø
.word 0c50dh     ;  207.422 ø
.word 0c45bh     ;  207.773 ø
.word 0c3a9h     ;  208.125 ø
.word 0c2f8h     ;  208.477 ø
.word 0c248h     ;  208.828 ø
.word 0c198h     ;  209.180 ø
.word 0c0e9h     ;  209.531 ø
.word 0c03ah     ;  209.883 ø
.word 0bf8ch     ;  210.234 ø
.word 0bedfh     ;  210.586 ø
.word 0be32h     ;  210.938 ø
.word 0bd86h     ;  211.289 ø
.word 0bcdah     ;  211.641 ø
.word 0bc2fh     ;  211.992 ø
.word 0bb85h     ;  212.344 ø
.word 0badch     ;  212.695 ø
.word 0ba33h     ;  213.047 ø
.word 0b98bh     ;  213.398 ø
.word 0b8e3h     ;  213.750 ø
.word 0b83ch     ;  214.102 ø
.word 0b796h     ;  214.453 ø
.word 0b6f1h     ;  214.805 ø
.word 0b64ch     ;  215.156 ø
.word 0b5a8h     ;  215.508 ø
.word 0b505h     ;  215.859 ø
.word 0b462h     ;  216.211 ø
.word 0b3c0h     ;  216.562 ø
.word 0b31fh     ;  216.914 ø
.word 0b27fh     ;  217.266 ø
.word 0b1dfh     ;  217.617 ø
.word 0b140h     ;  217.969 ø
.word 0b0a2h     ;  218.320 ø
.word 0b005h     ;  218.672 ø
.word 0af68h     ;  219.023 ø
.word 0aecch     ;  219.375 ø
.word 0ae31h     ;  219.727 ø
.word 0ad97h     ;  220.078 ø
.word 0acfdh     ;  220.430 ø
.word 0ac65h     ;  220.781 ø
```

```
        .word 0abcdh     ;  221.133 ø
        .word 0ab36h     ;  221.484 ø
        .word 0aaa0h     ;  221.836 ø
        .word 0aa0ah     ;  222.188 ø
        .word 0a976h     ;  222.539 ø
        .word 0a8e2h     ;  222.891 ø
        .word 0a84fh     ;  223.242 ø
        .word 0a7bdh     ;  223.594 ø
        .word 0a72ch     ;  223.945 ø
        .word 0a69ch     ;  224.297 ø
        .word 0a60ch     ;  224.648 ø
        .word 0a57eh     ;  225.000 ø
        .word 0a4f0h     ;  225.352 ø
        .word 0a463h     ;  225.703 ø
        .word 0a3d7h     ;  226.055 ø
        .word 0a34ch     ;  226.406 ø
        .word 0a2c2h     ;  226.758 ø
        .word 0a238h     ;  227.109 ø
        .word 0a1b0h     ;  227.461 ø
        .word 0a129h     ;  227.812 ø
        .word 0a0a2h     ;  228.164 ø
        .word 0a01ch     ;  228.516 ø
        .word 09f98h     ;  228.867 ø
        .word 09f14h     ;  229.219 ø
        .word 09e91h     ;  229.570 ø
        .word 09e0fh     ;  229.922 ø
        .word 09d8eh     ;  230.273 ø
        .word 09d0eh     ;  230.625 ø
        .word 09c8fh     ;  230.977 ø
        .word 09c11h     ;  231.328 ø
        .word 09b94h     ;  231.680 ø
        .word 09b17h     ;  232.031 ø
        .word 09a9ch     ;  232.383 ø
        .word 09a22h     ;  232.734 ø
        .word 099a9h     ;  233.086 ø
        .word 09930h     ;  233.438 ø
        .word 098b9h     ;  233.789 ø
        .word 09843h     ;  234.141 ø
        .word 097ceh     ;  234.492 ø
        .word 09759h     ;  234.844 ø
        .word 096e6h     ;  235.195 ø
        .word 09674h     ;  235.547 ø
        .word 09603h     ;  235.898 ø
        .word 09592h     ;  236.250 ø
        .word 09523h     ;  236.602 ø
        .word 094b5h     ;  236.953 ø
        .word 09448h     ;  237.305 ø
        .word 093dch     ;  237.656 ø
        .word 09371h     ;  238.008 ø
        .word 09307h     ;  238.359 ø
        .word 0929eh     ;  238.711 ø
        .word 09236h     ;  239.062 ø
        .word 091cfh     ;  239.414 ø
        .word 09169h     ;  239.766 ø
        .word 09105h     ;  240.117 ø
```

```
.word 090a1h     ;  240.469 ø
.word 0903eh     ;  240.820 ø
.word 08fddh     ;  241.172 ø
.word 08f7dh     ;  241.523 ø
.word 08f1dh     ;  241.875 ø
.word 08ebfh     ;  242.227 ø
.word 08e62h     ;  242.578 ø
.word 08e06h     ;  242.930 ø
.word 08dabh     ;  243.281 ø
.word 08d51h     ;  243.633 ø
.word 08cf8h     ;  243.984 ø
.word 08ca1h     ;  244.336 ø
.word 08c4ah     ;  244.688 ø
.word 08bf5h     ;  245.039 ø
.word 08ba0h     ;  245.391 ø
.word 08b4dh     ;  245.742 ø
.word 08afbh     ;  246.094 ø
.word 08aaah     ;  246.445 ø
.word 08a5ah     ;  246.797 ø
.word 08a0ch     ;  247.148 ø
.word 089beh     ;  247.500 ø
.word 08972h     ;  247.852 ø
.word 08927h     ;  248.203 ø
.word 088ddh     ;  248.555 ø
.word 08894h     ;  248.906 ø
.word 0884ch     ;  249.258 ø
.word 08805h     ;  249.609 ø
.word 087c0h     ;  249.961 ø
.word 0877bh     ;  250.312 ø
.word 08738h     ;  250.664 ø
.word 086f6h     ;  251.016 ø
.word 086b6h     ;  251.367 ø
.word 08676h     ;  251.719 ø
.word 08637h     ;  252.070 ø
.word 085fah     ;  252.422 ø
.word 085beh     ;  252.773 ø
.word 08583h     ;  253.125 ø
.word 08549h     ;  253.477 ø
.word 08511h     ;  253.828 ø
.word 084d9h     ;  254.180 ø
.word 084a3h     ;  254.531 ø
.word 0846eh     ;  254.883 ø
.word 0843ah     ;  255.234 ø
.word 08407h     ;  255.586 ø
.word 083d6h     ;  255.938 ø
.word 083a6h     ;  256.289 ø
.word 08377h     ;  256.641 ø
.word 08349h     ;  256.992 ø
.word 0831ch     ;  257.344 ø
.word 082f1h     ;  257.695 ø
.word 082c6h     ;  258.047 ø
.word 0829dh     ;  258.398 ø
.word 08276h     ;  258.750 ø
.word 0824fh     ;  259.102 ø
.word 0822ah     ;  259.453 ø
```

```
        .word 08205h     ;  259.805 ø
        .word 081e2h     ;  260.156 ø
        .word 081c1h     ;  260.508 ø
        .word 081a0h     ;  260.859 ø
        .word 08181h     ;  261.211 ø
        .word 08163h     ;  261.562 ø
        .word 08146h     ;  261.914 ø
        .word 0812ah     ;  262.266 ø
        .word 08110h     ;  262.617 ø
        .word 080f6h     ;  262.969 ø
        .word 080deh     ;  263.320 ø
        .word 080c8h     ;  263.672 ø
        .word 080b2h     ;  264.023 ø
        .word 0809eh     ;  264.375 ø
        .word 0808bh     ;  264.727 ø
        .word 08079h     ;  265.078 ø
        .word 08068h     ;  265.430 ø
        .word 08059h     ;  265.781 ø
        .word 0804bh     ;  266.133 ø
        .word 0803eh     ;  266.484 ø
        .word 08032h     ;  266.836 ø
        .word 08027h     ;  267.188 ø
        .word 0801eh     ;  267.539 ø
        .word 08016h     ;  267.891 ø
        .word 0800fh     ;  268.242 ø
        .word 0800ah     ;  268.594 ø
        .word 08006h     ;  268.945 ø
        .word 08002h     ;  269.297 ø
        .word 08001h     ;  269.648 ø
; Numbers in stage 10 = 767
        .endif
TWIDEND  .set $
TWIDLEN  .set TWIDEND–TWIDSTRT
```

A.10 Cfft.cmd

```
/*********************************************************************/
/*    This is the Linker Command File for the TMS320C541             */
/*********************************************************************/
/*-c
-lrts.lib
-stack 0x200*/

MEMORY
{
  PAGE 0:
 INT_PM_DRAM: origin = 80h length = 1380h   /* DARAM if ovly=1 */
 EXT_PM_RAM: origin = 1400h length = 0ec00h   /* ext PM 0-wait */

 PAGE 1:
 SCRATCH: origin = 00060h  length = 00020h  /* page-0 scratch-pad */
 INT_DM_RAM: origin = 80h  length = 1380h   /* DARAM */
 EXT_DM_RAM: origin = 01400h  length = 0ec00h  /* ext DM 0-wait */
}

SECTIONS
{
    .text   : > INT_PM_DRAM   PAGE 0
    .cinit  : > INT_PM_DRAM   PAGE 0
    .sintab : > INT_DM_RAM    PAGE 1
    .input  : > INT_DM_RAM    PAGE 1,align(2048)
    .data : > INT_DM_RAM      PAGE 1
    .bss    : > INT_DM_RAM    PAGE 1
    .stack  : > INT_DM_RAM    PAGE 1
    .xref   : > INT_DM_RAM    PAGE 1
    .test   : > INT_DM_RAM    PAGE 1
}
```

IMPORTANT NOTICE

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

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

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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

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