Application Report
SPRA417A

EVM Application #8
Implementing a Spectrum Analyzer Using a 128-Point FFT and the TMS320F240 EVM

Dave Figoli *Digital Signal Processing Applications*

Abstract

Evaluation module (EVM) application #8 produces a spectrum analyzer using the digital-to-analog converter (DAC) of the Texas Instruments (TI™) TMS320F240 EVM. The spectrum analyzer is produced using a 128-point fast Fourier transform (FFT) with the values collected from the analog-to-digital converter (ADC). The output is produced using the DAC, and the results can be viewed using an oscilloscope and the XF pin as a trigger. The results of the FFT are normalized so that the maximum value obtained is equivalent to 5 V. This application is implemented using C2xx assembly code.

The specific topic discussed is FFT implementation.

Contents

Overview

This application produces a spectrum analyzer using the DAC of the TI TMS320F240 EVM. The spectrum analyzer is produced using a 128-point FFT with the values collected from the ADC. The output is produced using the DAC, and the results can be viewed using an oscilloscope and the XF pin as a trigger. The results of the FFT are normalized so that the maximum value obtained is equivalent to 5 V. This application is implemented using C2xx assembly code.

Modules Used

- Event manager
- General-purpose timer 1
- ADC
- DAC

Input

ADCIN0

Output

XF, DAC0

Background and Methodology

The initial setup of the program is similar to the finite-impulse-response (FIR) filter application. The phase-locked loop (PLL) module, digital I/O ports, and the event manager must be configured prior to capturing the input waveform.

The event manager provides the sampling time base via the general-purpose timer in this application. Without a known sampling frequency, the peaks appearing in the FFT result cannot be associated with any particular frequency. For example, if we knew that the sampling frequency was 2 kHz in a 128-point FFT, the first spike would be DC, then 1kHz/64, 2×1kHz/64, 3×1kHz/64, etc.

The PLL module is set up as in EVM application #1 (PWM0.ASM). (For more information on EVM application #1, see the application report, *Creation of Pulse Width Modulated Signal With a Fixed Duty Cycle*, SPRA410.) The frequency at which the CPUCLK runs is important to set up the timer period register of the event manager module to control the sampling frequency of the ADC. In this application, the CPUCLK is set to 20 MHz (that is, CLKIN = 10 MHz, PLL divide by 2 enabled, PLL multiplication ratio is 4).

As in EVM application #4, the output control register A must be configured so that the I/O port sharing its pin with the ADC is configured as an ADC input channel. (For more information on EVM application #4, see the application report, *Generating a PWM Signal Modulated by an Analog Input*, SPRA413.)

The ADC is set up as in EVM application #5 (FIR0.ASM). (For more information on EVM application #5, see the application report, *Detecting a Frequency Band Using a FIR Bandpass Filter*, SPRA414.) The event manager controls the sampling frequency. Because the event manager starts the ADC, the ADC is set up for a single conversion.

FFT Implementation

The FFT is implemented using a polling routine. All of the interrupts are masked in the setup of the event manager. However, because the interrupt flags are still set when an interrupt is supposed to occur, the flags still can be used in a polling routine.

A polling routine determines when the period register of the timer matches the counter. When the two values match, the timer interrupt flag is set. By using a polling routine to watch the flag, the routine that gathers the data can be executed each time the flag is set.

The routine that gathers the values from the ADCFIFO register:

1) Fetches the value
2) Stores the converted value to a location in memory using the bit-reversed addressing capabilities of the C2xx core
3) Changes the counter that tracks how many points have been gathered
4) Performs either one of the following:
   a) Restarts the ADC (if more points must be gathered), clears the flag register, and returns to the polling routine, or
   b) Implements the FFT with the 128 stored points

A counter keeps track of how many values the ADC converts. The polling routine waits until the flag register of the event manager is set before the value in the ADCFIFO register is read. Because the FFT cannot be performed until 128 samples are obtained, the auxiliary registers can be set before the first sample is obtained. The indirect addressing of the C2xx manipulates the auxiliary registers as the values are loaded. Once the values are loaded, the FFT can be performed.

The FFT results can be output to the DAC of the EVM; however, the results from the FFT are small and hard to visualize. As a result, the magnitudes of the FFT are normalized so that the maximum magnitude is the maximum voltage the DAC can produce.

To normalize the values of the magnitudes, the maximum magnitude of the FFT must be known. A simple pass through the memory locations containing the magnitudes locates the maximum value. Once the maximum value is obtained, the remaining values can be “divided” by the maximum value. To normalize the values to the maximum value, a routine that subtracts the maximum value from the magnitude and then shifts the remainder so that the maximum value can be subtracted again provides a Q15 result by repeating the routine 16 times.

For example, use the simple fractions ¼ and 2/3. The Q15 value for ¼ is 2000h and, the Q15 value for 2/3 is 5555h.

Q15 result		**Q15 result**	
	↓		↓
001 ÷ 100 =	**0** r 01 shift left 010	010 ÷ 011 =	**0** r 10 shift left 100
010 ÷ 100 =	**0** r 10 shift left 100	100 ÷ 011 =	**1** r 01 shift left 10
100 ÷ 100 =	**1** r 0 shift left 0	010 ÷ 011 =	**0** r 10 shift left 100
000 ÷ 100 =	**0** r 0 shift left 0	100 ÷ 011 =	**1** r 01 shift left 10
•		•	
•		•	
•		•	

Result 2000h (Q15 = 0.25) Result 5555h (Q15 = 0.666656)

The resulting Q15 values can then be multiplied by the maximum value the DAC of the EVM can output, which is FFFh. The result is that every magnitude is scaled in proportion to the maximum value.

The output of the FFT can be viewed through an oscilloscope. While the FFT is being output through the channel of the DAC, the XF pin is held high. After the FFT results have been output, the XF pin is brought low. As a result, the XF pin can be the trigger for the scope to view the results of the FFT.

Once the program is executed, it runs until the program is halted. Thus, the frequency input into the ADC can be modified, and the results can be viewed through an oscilloscope.

```
;***************************************************************************
; File Name: FFT0.ASM
; Project:   C240 EVM Test Platform
; Originator: Allister Chang (Texas Instruments)
;
; Description:The program perfoms a 128 point FFT.  The data samples
;   are gathered from the ADC of the C240.  The FFT is performed
;   and the magnitudes of the FFT are output via the DAC of the
;   EVM test board.  The results can then be viewed using
;   an oscilloscope.  The trigger for the FFT results on the
;   oscilloscope is XF.
;
;
;***************************************************************************
; Last Update:10 June 99
;
; Descriptions:   C/F240 on-chip 10bit ADC is connected to the 10 MSB of the
;   internal data bus[15:5].  Previous version assumed the ADC
;   is connected to the 10 LSB of the data bus.  As a result
;   all data collected with MSB 1 were interpreted as  neg
;   number and gave incorrect FFT result.
;
; Status:  Works
;
;
;===========================================================================
   .include  f240regs.h

;---------------------------------------------------------------------------
; Debug directives
;---------------------------------------------------------------------------
   .def GPR0  ;General purpose registers.
   .def GPR1
   .def GPR2
   .def GPR3

;---------------------------------------------------------------------------
; FFT specific constants
;---------------------------------------------------------------------------
N  .set 128   ;FFT length
TWID_TBL .set 08100h   ;Start of Twiddle table in B1
B0_SADR  .set 08000h   ;B0 start address  (changed to locate data at 0x8000
B0_EADR  .set 080FFh   ;B0 end address
B1_SADR  .set 08100h   ;B1 start address
B1_EADR  .set 081FFh   ;B1 end address
B2_SADR  .set 0060h  ;B2 start address
B2_EADR  .set 007Fh  ;B2 end address
COS45K .set 05A82h ;Constant for COS(45)

;-------------------------------------------------------------------
; I/O Mapped EVM Registers
;-------------------------------------------------------------------
DAC0  .set 0000h  ;Input Data Register for DAC0
DAC1  .set 0001h  ;Input Data Register for DAC1
DAC2  .set 0002h  ;Input Data Register for DAC2
DAC3  .set 0003h  ;Input Data Register for DAC3
DACUPDATE .set 0004h  ;DAC Update Register

;---------------------------------------------------------------------------
; Variable Declarations for on chip RAM Block B2 (DP=0)
;---------------------------------------------------------------------------
   .bss GPR0,1 ;General purpose registers.
   .bss GPR1,1
   .bss GPR2,1
   .bss GPR3,1
   .bss COS45,1  ;Value for COS(45)

   .bss DAC0VAL,1 ;DAC0 Channel Value
   .bss DAC1VAL,1 ;DAC1 Channel Value
   .bss DAC2VAL,1 ;DAC2 Channel Value
   .bss DAC3VAL,1 ;DAC3 Channel Value

   .bss COUNTER,1 ;COUNTER for obtaining 128 values

   .bss RPT_NUM,1
   .bss uSEC,1

;*************************************************************************
;                     *
;     MACRO 'ZEROI'    number of words/number of cycles: 10             *
;                     *
;  ARP=2 FOR INPUT AND OUTPUT              *
;  AR2 -> QR,QI,QR+1,...              *
;  AR1 -> PR,PI,PR+1,...              *
;                     *
;  CALCULATE Re[P+Q] AND Re[P-Q]        *
;  QR'=(PR-QR)/2                                          *
;  PR'=(PR+QR)/2                                         *
;  PI'=(PI+QI)/2                                         *
;  PI'=(PI-QI)/2                                          *
;                 *
;  version 1.00    from Manfred Christ update: 2. July 90  *
;                     *
;*************************************************************************
ZEROI  .macro         ;        AR1  AR2  ARP
         LACC    *,15,AR1  ; ACC := (1/2)(QR)           PR   QR    1
         ADD     *,15      ; ACC := (1/2)(PR+QR)        PR   QR    1
         SACH    *+,0,AR2  ; PR  := (1/2)(PR+QR)        PI   QR    2
         SUB     *,16      ; ACC := (1/2)(PR+QR)-(QR)   PI   QR    2
         SACH    *+        ; QR  := (1/2)(PR-QR)        PI   QI    2

         LACC    *,15,AR1  ; ACC := (1/2)(QI)           PI   QI    1
         ADD     *,15      ; ACC := (1/2)(PI+QI)        PI   QI    1
         SACH    *+,0,AR2  ; PI  := (1/2)(PI+QI)        PR+1 QI    2
         SUB     *,16      ; ACC := (1/2)(PI+QI)-(QI)   PR+1 QI    2
         SACH    *+        ; QI  := (1/2)(PI-QI)        PR+1 QR+1  2
   .endm

;*************************************************************************
;                     *
;     MACRO 'PBY2I'    number of words/number of cycles: 12             *
;                     *
;  ARP=2 on entry to macro              *
;  AR2 -> QR,QI,QR+1,...              *
;  AR1 -> PR,PI,PR+1,...              *
;                 *
;  PR'=(PR+QI)/2      PI'=(PI-QR)/2                              *
;  QR'=(PR-QI)/2      QI'=(PI+QR)/2                              *
;                     *
;  version 1.00    from Manfred Christ update: 02. July 90   *
;                     *
;*************************************************************************
PBY2I  .macro         ;        AR1  AR2  ARP
         LACC    *+,15,AR5 ;                            PR   QI    5
         SACH    *,1,AR2   ; TMP=QR                     PR   QI    2

         LACC    *,15,AR1  ; ACC := QI/2                PR   QI    1
         ADD     *,15      ; ACC := (PR+QI)/2           PR   QI    1
         SACH    *+,0,AR2  ; PR  := (PR+QI)/2           PI   QI    2
         SUB     *-,16     ; ACC := (PR-QI)/2           PI   QR    2
         SACH    *+,0,AR1  ; QR  := (PR-QI)/2           PI   QI    1

         LACC    *,15,AR5  ; ACC := (PI)/2              PI   QI    5
         SUB     *,15,AR1  ; ACC := (PI-QR)/2           PI   QI    1
         SACH    *+,0,AR5  ; PI  := (PI-QR)/2           PR+1 QI    5
         ADD     *,16,AR2  ; ACC := (PI+QR)/2           PR+1 QI    2
         SACH    *+        ; QI  := (PI+QR)/2           PR+1 QI+1  2
   .endm

;*************************************************************************
;                *
;     MACRO 'PBY4I'    number of words/number of cycles: 16   *
;                *
;  T=SIN(45)=COS(45)=W45           *
;                *
;  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) *
;                *
;                *
;  version 1.00    from Manfred Christ update: 02. July 90   *
;                *
;*************************************************************************
PBY4I  .macro         ; TREG= W        AR5   PREG   AR1 AR2 ARP
         MPY      *+,AR5   ; PREG= W*QR/2              -   W*QR/2 PR   QI   5
         SPH      *,AR1    ; TMP = W*QR/2          W*QR/2 W*QR/2 PR   QI   1
         LACC     *,15,AR2 ; ACC = PR/2            W*QR/2 W*QR/2 PR   QI   2
         MPYS     *-       ; ACC = (PR-W*QR)/2     W*QR/2 W*QI/2 PR   QR   2
         SPAC              ; ACC = (PR-W*QI-W*QR)/2  W*QR/2 W*QI/2 PR   QR   2
         SACH     *+,0,AR1 ; QR  = (PR-W*QI-W*QR)/2  W*QR/2 W*QI/2 PR   QI   1
         SUB      *,16     ; ACC = (-PR-W*QI-W*QR)/2 W*QR/2 W*QI/2 PR   QI   1
         NEG               ; ACC =  (PR+W*QI+W*QR)/2 W*QR/2 W*QI/2 PR   QI   1
         SACH     *+       ; QR  =  (PR+W*QI+W*QR)/2 W*QR/2 W*QI/2 PI   QI   1

         LACC     *,15,AR5 ; ACC = (PI)/2          W*QR/2 W*QI/2 PI   QI   5
         SPAC              ; ACC = (PI-W*QI)/2     W*QR/2   -    PI   QI   5
         ADD      *,16,AR2 ; ACC = (PI-W*QI+W*QR)/2    -      -    PI   QI   2
         SACH     *+,0,AR1 ; QI  = (PI-W*QI+W*QR)/2    -      -    PI   QR1  1
         SUB      *,16     ; ACCU= (-PI-W*QI+W*QR)/2   -      -    PI   QR1  1
         NEG               ; ACCU=  (PI+W*QI-W*QR)/2   -      -    PI   QR1  1
         SACH     *+,0,AR2 ; PI  =  (PI+W*QI-W*QR)/2   -      -    PR1  QR1  2
   .endm

;*************************************************************************
;                *
;     MACRO 'P3BY4I'   number of words/number of cycles: 16   *
;                *
;     version 1.00     from: Manfred Christ   update: 02. July 90   *
;                *
;  ENTRANCE IN THE MACRO: ARP=AR2       *
;     AR1->PR,PI          *
;     AR2->QR,QI          *
;     TREG=W=COS(45)=SIN(45)      *
;                *
;  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) *
;                *
;  EXIT OF THE MACRO: ARP=AR2       *
;     AR1->PR+1,PI+1       *
;     AR2->QR+1,QI+1       *
;                *
;*************************************************************************
```

```
P3BY4I .macro p,m    ; TREG= W           AR5   PREG   AR1  AR2 ARP
                     ;                         ------ ------ ---  --- ---
      MPY    *+,AR5      ; PREG= W*QR/2              -   W*QR/2 PR   QI   5
      SPH    *,AR1       ; TMP = W*QR/2           W*QR/2 W*QR/2 PR   QI   1
      LACC   *,15,AR2    ; ACC = PR/2             W*QR/2 W*QR/2 PR   QI   2
      MPYA   *-          ; ACC = (PR+W*QR)/2      W*QR/2 W*QI/2 PR   QR   2
      SPAC               ; ACC = (PR-W*QI+W*QR)/2  W*QR/2 W*QI/2 PR   QR   2
      SACH   *+,0,AR1    ; QR' = (PR-W*QI+W*QR)/2  W*QR/2 W*QI/2 PR   QI   1
      SUB    *,16        ; ACC = (-PR-W*QI+W*QR)/2 W*QR/2 W*QI/2 PR   QI   1
      NEG                ; ACC =  (PR+W*QI-W*QR)/2 W*QR/2 W*QI/2 PR   QI   1
      SACH   *+          ; PR' =  (PR+W*QI-W*QR)/2 W*QR/2 W*QI/2 PI   QI   1

      LACC   *,15,AR5    ; ACC = (PI)/2           W*QR/2 W*QI/2 PI   QI   5
      APAC               ; ACC = (PI+W*QI)/2      W*QR/2   -    PI   QI   5
      ADD    *,16,AR2    ; ACC = (PI+W*QI+W*QR)/2    -      -    PI   QI   2
      SACH   *:m:+,0,AR1 ; QI' = (PI+W*QI+W*QR)/2    -      -    PI   QR5  1
      SUB    *,16        ; ACCU= (-PI+W*QI+W*QR)/2   -      -    PI   QR5  1
      NEG                ; ACCU= (PI-W*QI-W*QR)/2    -      -    PI   QR5  1
      SACH   *:m:+,0,AR:p: ; PI' = (PI-W*QI-W*QR)/2  -     PR5   QR5  7
      .endm

;*****************************************************************************
;                  *
;  MACRO:  'BFLY'      general butterfly radix 2 for 320C2xx/5x    *
;                  *
;  version 1.00    from Manfred Christ update: 02. July 90    *
;                  *
;  THE MACRO 'BFLY' REQUIRES 18 WORDS AND 18 INSTRUCTIONS     *
;                  *
;  Definition: ARP -> AR2  (input)  ARP -> AR2    (output)    *
;                  *
;  Definition: AR1 -> QR   (input)  AR1 -> QR+1    (output)    *
;  Definition: AR2 -> PR   (input)  AR2 -> PR+1    (output)    *
;  Definition: AR3 -> Cxxx (input)  AR3 -> Cxxx+1   (output)  --> WR=cosine *
;  Definition: AR4 -> Sxxx (input)  AR4 -> Sxxx+1   (output)  --> WI=sine *
;  Definition: AR5 -> temporary variable (unchanged)      *
;                  *
;  uses index register             *
;                  *
; PR' = (PR+(QR*WR+QI*WI))/2       WR=COS(W)   WI=SIN(W)   *
; PI' = (PI+(QI*WR-QR*WI))/2           *
; QR' = (PR-(QR*WR+QI*WI))/2           *
; QI' = (PI-(QI*WR-QR*WI))/2           *
;                  *
;  Note: AR0 determines Twiddle Pointers (AR3 & AR4) step increments   *
;                  *
;*****************************************************************************
BFLY .macro p
;                                     (contents of register after exec.)
;                                     TREG AR1  AR2  AR3 AR4 ARP
;                  --- ---  ---  --- --- ---
  LT    *+,AR3    ;TREG:= QR                 QR PR   QI    C   S   3
  MPY   *,AR2     ;PREG:= QR*WR/2            QR PR   QI    C   S   2
  LTP   *-,AR4    ;ACC := QR*WR/2            QI PR   QR    C   S   4
  MPY   *,AR3     ;PREG:= QI*WI/2            QI PR   QR    C   S   3
  MPYA  *0+,AR2   ;ACC := (QR*WR+QI*WI)/2    QR PR   QR    C+n S   2
    ;PREG:= QI*WR
  LT   *,AR5;TREG = QR              QR PR   QR    C+n S   5
  SACH  *,1,AR1   ;TEMP:= (QR*WR+QI*WI)      QR PR   QR    C+n S   1

  ADD   *,15 ;ACC := (PR+(QR*WR+QI*WI))/2 QR PR   QR    C+n S   1
  SACH  *+,0,AR5  ;PR  := (PR+(QR*WR+QI*WI))/2 QR PI   QR    C+n S   5
  SUB   *,16,AR2  ;ACC := (PR-(QR*WR+QI*WI))/2 QR PI   QR    C+n S   2
  SACH  *+,0,AR1  ;QR  := (PR-(QR*WR+QI*WI))/2 QR PI   QI    C+n S   1
  LAC   *,15,AR4  ;ACC := PI  /PREG=QI*WR      QI PI   QI    C+n S   4
  MPYS  *0+,AR2   ;PREG:= QR*WI/2            QI PI   QI    C+n S+n 2
         ;ACC := (PI-QI*WR)/2
  APAC        ;ACC := (PI-(QI*WR-QR*WI))/2 QI PI   QI    C+n S+n 2
  SACH  *+,0,AR1  ;QI  := (PI-(QI*WR-QR*WI))/2 QI PI   QR+1  C+n S+n 1
  NEG     ;ACC :=(-PI+(QI*WR-QR*WI))/2 QI PI   QR+1  C+n S+n 1
  ADD   *,16   ;ACC := (PI+(QI*WR-QR*WI))/2 QI PI   QR+1  C+n S+n 1
  SACH  *+,0,AR:p: ;PI := (PI+(QI*WR-QR*WI))/2 QI PR+1 QR+1  C+n S+n 2
  .endm

;*****************************************************************************
; MACRO 'COMBO'                                                              *
;                  *
; 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                  *
;                      *
;*****************************************************************************
;                    ARP AR1 AR2 AR3 AR4 AR5
COMBO .macro   ;              --- --- --- --- --- ---
     LACC   *,14,AR4    ; ACC := (R3)/4              4  R1  R2  R3  R4  T1
     SUB    *,14,AR5    ; ACC := (R3+R4)/4           5  R1  R2  R3  R4  T1
     SACH   *+,1,AR4    ; T1   = (R3-R4)/2           4  R1  R2  I3  R4  T2

     ADD    *+,15,AR5   ; ACC := (R3+R4)/4           5  R1  R2  R3  I4  T2
     SACH   *,1,AR2     ; T2   = (R3+R4)/2           2  R1  R2  R3  I4  T2

     ADD    *,14,AR1    ; ACC := (R2+R3+R4)/4        1  R1  R2  R3  I4  T2
     ADD    *,14        ; ACC := (R1+R2+R3+R4)/4     1  R1  R2  R3  I4  T2
     SACH   *+,0,AR5    ; R1  := (R1+R2+R3+R4)/4     5  I1  R2  R3  I4  T2
     SUB    *,16,AR3    ; ACC := (R1+R2-(R3+R4))/4   3  I1  R2  R3  I4  T2
     SACH   *+,0,AR5    ; R3  := (R1+R2-(R3+R4))/4   5  I1  R2  I3  I4  T2

     ADD    *,15,AR2    ; ACC := (R1+R2)/4           2  I1  R2  I3  I4  T2
     SUB    *,15,AR3    ; ACC := (R1-R2)/4           3  I1  R2  I3  I4  T2
     ADD    *,14,AR4    ; ACC := ((R1-R2)+(I3))/4    4  I1  R2  I3  I4  T2
     SUB    *,14,AR2    ; ACC := ((R1-R2)+(I3-I4))/4  2  I1  R2  I3  I4  T2
     SACH   *+,0,AR4    ; R2  := ((R1-R2)+(I3-I4))/4  4  I1  I2  I3  I4  T2
     ADD    *-,15,AR3   ; ACC := ((R1-R2)+ I3+I4 )/4  3  I1  I2  I3  R4  T2
     SUB    *,15,AR4    ; ACC := ((R1-R2)-(I3-I4))/4  4  I1  I2  I3  R4  T2
     SACH   *+,0,AR1    ; R4  := ((R1-R2)-(I3-I4))/4  1  I1  I2  I3  I4  T2

     LACC   *,14,AR2    ; ACC := (I1)/4              2  I1  I2  I3  I4  T2
     SUB    *,14,AR5    ; ACC := (I1-I2)/4           5  I1  I2  I3  I4  T2
     SACH   *,1,AR2     ; T2  := (I1-I2)/2           2  I1  I2  I3  I4  T2
     ADD    *,15,AR3    ; ACC := ((I1+I2))/4         4  I1  I2  I3  I4  T2
     ADD    *,14,AR4    ; ACC := ((I1+I2)+(I3))/4    4  I1  I2  I3  I4  T2
     ADD    *,14,AR1    ; ACC := ((I1+I2)+(I3+I4))/4  1  I1  I2  I3  I4  T2
     SACH   *0+,0,AR3   ; I1  := ((I1+I2)+(I3+I4))/4  3  R5  I2  I3  I4  T2
     SUB    *,15,AR4    ; ACC := ((I1+I2)-(I3+I4))/4  4  R5  I2  I3  I4  T2
     SUB    *,15,AR3    ; ACC := ((I1+I2)-(I3+I4))/4  3  R5  I2  I3  I4  T2
     SACH   *0+,0,AR5   ; I3  := ((I1+I2)-(I3+I4))/4  5  R5  I2  R7  I4  T2

     LACC   *-,15       ; ACC := (I1-I2)/4           5  R5  I2  R7  I4  T1
     SUB    *,15,AR2    ; ACC := ((I1-I2)-(R3-R4))/4  2  R5  I2  R7  I4  T1
     SACH   *0+,0,AR5   ; I2  := ((I1-I2)-(R3-R4))/4  5  R5  R6  R7  I4  T1
     ADD    *,16,AR4    ; ACC := ((I1-I2)+(R3-R4))/4  4  R5  R6  R7  I4  T1
     SACH   *0+,0,AR7; I4  := ((I1-I2)+(R3-R4))/4  7  R5  R6  R7  R8  T1
 .endm
```

```
;-----------------------------------------------------------------------------
; Vector address declarations
;-----------------------------------------------------------------------------
    .sect  ".vectors"  ; use if Vectors are to be programmed in EPROM

RSVECT B    START   ; PM 0   Reset Vector    1
INT1   B    PHANTOM ; PM 2   Ext Int 1    4
INT2   B    PHANTOM ; PM 4   Ext Int 2    5
INT3   B    PHANTOM ; PM 6   Ext Int 3    6
INT4   B    PHANTOM ; PM 8   Ext Int 4    7
INT5   B    PHANTOM ; PM A   Ext Int 5    8
INT6   B    PHANTOM ; PM C   Ext Int 6    9
RESERVED    B    PHANTOM ; PM E   (Analysis Int)    10
SW_INT8     B    PHANTOM ; PM 10  User S/W int    -
SW_INT9     B    PHANTOM ; PM 12  User S/W int    -
SW_INT10    B    PHANTOM ; PM 14  User S/W int    -
SW_INT11    B    PHANTOM ; PM 16  User S/W int    -
SW_INT12    B    PHANTOM ; PM 18  User S/W int    -
SW_INT13    B    PHANTOM ; PM 1A  User S/W int    -
SW_INT14    B    PHANTOM ; PM 1C  User S/W int    -
SW_INT15    B    PHANTOM ; PM 1E  User S/W int    -
SW_INT16    B    PHANTOM ; PM 20  User S/W int    -
TRAP   B    PHANTOM ; PM 22  Trap vector     -
NMINT    B    PHANTOM ; PM 24  Non maskable Int   3
EMU_TRAP    B    PHANTOM ; PM 26  Emulator Trap     2
SW_INT20    B    PHANTOM ; PM 28  User S/W int    -
SW_INT21    B    PHANTOM ; PM 2A  User S/W int    -
SW_INT22    B    PHANTOM ; PM 2C  User S/W int    -
SW_INT23    B    PHANTOM ; PM 2E  User S/W int    -

;==============================================================================
; M A I N   C O D E  - starts here
;==============================================================================
   .text
   NOP
START:
   SETC INTM  ;Disable Interrupts
   SPLK #0000h,IMR ;Mask all core interrupts
   LACC IFR   ;Read Interrupt flags
   SACL IFR   ;Clear all interrupt flags
   NOP

   LDP #0h
   SPM 0  ;no shift from PREG-->ALU
   ROVM   ;dis overflow
   SSXM   ;allow sign extension
   CLRC CNF   ;Config Block B0 to Data mem.
   LACC #COS45K
   SACL COS45  ;Init location with constant.



;-----------------------------------
; Set up PLL Module
;-----------------------------------

   LDP   #00E0h  ;DP = 224; Address 7000h - 707Fh

;The following line is necessary if a previous program set the PLL to a different
;setting than the settings which the application uses.  By disabling the PLL, the
;CKCR1 register can be modified so that the PLL can run at the new settings when
;it is re-enabled.

;    5432109876543210
   SPLK #0000000001000001b,CKCR0 ;CLKMD=PLL Disable,SYSCLK=CPUCLK/2
```

```
;    5432109876543210
   SPLK #0000000010111011b,CKCR1 ;CLKIN(OSC)=10MHz,CPUCLK=20MHz

;CKCR1 - Clock Control Register 1
; Bits 7-4 (1011)CKINF(3)-CKINF(0) - Crystal or Clock-In Frequency
;      Frequency = 10MHz
; Bit 3  (1) PLLDIV(2) - PLL divide by 2 bit
;      Divide PLL input by 2
; Bits 2-0 (011) PLLFB(2)-PLLFB(0) - PLL multiplication ratio
;      PLL Multiplication Ratio = 4


;    5432109876543210
   SPLK #0000000011000001b,CKCR0 ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2

;CKCR0 - Clock Control Register 0
; Bits 7-6 (11) CLKMD(1),CLKMD(0) - Operational mode of Clock Module
;      PLL Enabled; Run on CLKIN on exiting low power mode
; Bits 5-4 (00) PLLOCK(1),PLLOCK(0) - PLL Status. READ ONLY
; Bits 3-2 (00) PLLPM(1),PLLPM(0) - Low Power Mode
;      LPM0
; Bit 1  (0) ACLKENA - 1MHz ACLK Enable
;      ACLK Disabled
; Bit 0  (1) PLLPS - System Clock Prescale Value
;      f(sysclk)=f(cpuclk)/2

;    5432109876543210
   SPLK #0100000011000000b,SYSCR ;CLKOUT=CPUCLK

;SYSCR - System Control Register
; Bit 15-14 (01) RESET1,RESET0 - Software Reset Bits
;      No Action
; Bits 13-8 (000000) Reserved
; Bit 7-6(11) CLKSRC1,CLKSRC0 - CLKOUT-Pin Source Select
;      CPUCLK: CPU clock output mode
; Bit 5-0(000000)  Reserved


   SPLK #006Fh, WDCR  ;Disable WD if VCCP=5V (JP5 in pos. 2-3)
   KICK_DOG     ;Reset Watchdog

;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
;- Event Manager Module Reset           *
;*                 -
;- This section resets all of the Event Manager Module Registers.*
;* This is necessary for silicon revsion 1.1; however, for   -
;- silicon revisions 2.0 and later, this is not necessary   *
;*                 -
;-                 *
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-

   LDP #232   ;DP=232 Data Page for the Event Manager
   SPLK #0000h,GPTCON ;Clear General Purpose Timer Control
   SPLK #0000h,T1CON ;Clear GP Timer 1 Control
   SPLK #0000h,T2CON ;Clear GP Timer 2 Control
   SPLK #0000h,T3CON ;Clear GP Timer 3 Control

   SPLK #0000h,COMCON ;Clear Compare Control
   SPLK #0000h,ACTR  ;Clear Full Compare Action Control Register
   SPLK #0000h,SACTR ;Clear Simple Compare Action Control Register
   SPLK #0000h,DBTCON ;Clear Dead-Band Timer Control Register

   SPLK #0FFFFh,EVIFRA;Clear Interrupt Flag Register A
   SPLK #0FFFFh,EVIFRB;Clear Interrupt Flag Register B
   SPLK #0FFFFh,EVIFRC;Clear Interrupt Flag Register C

   SPLK #0000h,CAPCON ;Clear Capture Control

   SPLK #0000h,EVIMRA ;Clear Event Manager Mask Register A
   SPLK #0000h,EVIMRB ;Clear Event Manager Mask Register B
   SPLK #0000h,EVIMRC ;Clear Event Manager Mask Register C

;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
;- End of RESET section for silicon revision 1.1     *
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-

;-----------------------------------
; Set up Event Manager Module
;-----------------------------------
T1COMPARE .set 78
T1PERIOD .set 156    ;Sets up period for 128kHz frequency

;T1COMPARE  .set 156
;T1PERIOD .set 313    ;Sets up period for 64kHz frequency

;T1COMPARE  .set 500
;T1PERIOD .set 1000   ;Sets up period for 20kHz frequency


   LDP #232     ;DP=232, Data Page for Event Manager Addresses
   SPLK #T1COMPARE,T1CMPR   ;T1CMPR set up for 50% duty cycle

;    2109876543210
   SPLK #0000001010101b,GPTCON

;GPTCON - GP Timer Control Register
;Bit 15 (0)  T3STAT - GP Timer 3 Status.  READ ONLY
;Bit 14 (0)  T2STAT - GP Timer 2 Status.  READ ONLY
;Bit 13 (0)  T1STAT - GP Timer 1 Status.  READ ONLY
;Bits 12-11 (00) T3TOADC - ADC start by event of GP Timer 3
;      No event starts ADC
;Bits 10-9 (00) T2TOADC - ADC start by event of GP Timer 2
;      No event starts ADC
;Bits 8-7 (00) T1TOADC - ADC start by event of GP Timer 1
;      No event starts ADC
;Bit 6  (1)  TCOMPOE - Compare output enable
;      Enable all three GP timer compare outputs
;Bits 5-4 (01) T3PIN - Polarity of GP Timer 3 compare output
;      Active Low
;Bits 3-2 (01) T2PIN - Polarity of GP Timer 2 compare output
;      Active Low
;Bits 1-0 (01) T1PIN - Polarity of GP Timer 1 compare output
;      Active Low

   SPLK #T1PERIOD,T1PR  ;T1PR = T1PERIOD value for Sampling Freq.
   SPLK #0000h,T1CNT  ;Clear GP Timer 1 Counter
   SPLK #0000h,T2CNT  ;Clear GP Timer 2 Counter
   SPLK #0000h,T3CNT  ;Clear GP Timer 3 Counter


;    5432109876543210
   SPLK #0001000001000010b,T1CON

;T1CON - GP Timer 1 Control Register
;Bits 15-14 (00) FREE,SOFT - Emulation Control Bits
;      Stop immediately on emulation suspend
;Bits 13-11 (010) TMODE2-TMODE0 - Count Mode Selection
;      Continuous-Up Count Mode
;Bits 10-8 (000) TPS2-TPS0 - Input Clock Prescaler
```

```
;      Divide by 1
;Bit 7  (0) Reserved
;Bit 6  (0) TENABLE - Timer Enable
;      Disable timer operations
;Bits 5-4 (00)TCLKS1,TCLKS0 - Clock Source Select
;      Internal Clock Source
;Bits 3-2 (00)TCLD1,TCLD0 - Timer Compare Register Reload Condition
;      When counter is 0
;Bit 1  (1) TECMPR - Timer compare enable
;      Enable timer compare operation
;Bit 0  (0) Reserved

;    5432109876543210
   SPLK #0000000000000000b,T2CON  ;GP Timer 2 - Not Used

;T2CON - GP Timer 2 Control Register
;Bits 15-14  (00)FREE,SOFT - Emulation Control Bits
;      Stop immediately on emulation suspend
;Bits 13-11  (000) TMODE2-TMODE0 - Count Mode Selection
;      Stop/Hold
;Bits 10-8 (000) TPS2-TPS0 - Input Clock Prescaler
;      Divide by 1
;Bit 7  (0) TSWT1 - GP Timer 1 timer enable bit
;      Use own TENABLE bit
;Bit 6  (0) TENABLE - Timer Enable
;      Disable timer operations
;Bits 5-4 (00)TCLKS1,TCLKS0 - Clock Source Select
;      Internal Clock Source
;Bits 3-2 (00)TCLD1,TCLD0 - Timer Compare Register Reload Condition
;      When counter is 0
;Bit 1  (0) TECMPR - Timer compare enable
;      Disable timer compare operation
;Bit 0  (0) SELT1PR - Period Register select
;      Use own period register

;    5432109876543210
   SPLK #0000000000000000b,T3CON  ;GP Timer 3 - Not Used

;T3CON - GP Timer 3 Control Register
;Bits 15-14  (00)FREE,SOFT - Emulation Control Bits
;      Stop immediately on emulation suspend
;Bits 13-11  (000) TMODE2-TMODE0 - Count Mode Selection
;      Stop/Hold
;Bits 10-8 (000) TPS2-TPS0 - Input Clock Prescaler
;      Divide by 1
;Bit 7  (0) TSWT1 - GP Timer 1 timer enable bit
;      Use own TENABLE bit
;Bit 6  (0) TENABLE - Timer Enable
;      Disable timer operations
;Bits 5-4 (00)TCLKS1,TCLKS0 - Clock Source Select
;      Internal Clock Source
;Bits 3-2 (00)TCLD1,TCLD0 - Timer Compare Register Reload Condition
;      When counter is 0
;Bit 1  (0) TECMPR - Timer compare enable
;      Disable timer compare operation
;Bit 0  (0) SELT1PR - Period Register select
;      Use own period register


;-----------------------------------
; Set up Digital I/O Port
;-----------------------------------

   LDP #225    ;DP = 225; Addresses 7800h to 707Fh
;    5432109876543210
```

```
   SPLK#0011100000001111b,OCRA

;OCRA  - Output Control Register A
;Bit 15 (0) CRA.15 - IOPB7
;Bit 14 (0) CRA.14 - IOPB6
;Bit 13 (1) CRA.13 - T3PWM/T3CMP
;Bit 12 (1) CRA.12 - T2PWM/T2CMP
;Bit 11 (1) CRA.11 - T1PWM/T1CMP
;Bit 10 (0) CRA.10 - IOPB2
;Bit 9  (0) CRA.9 - IOPB1
;Bit 8  (0) CRA.8 - IOPB0
;Bits 7-4 (0000)Reserved
;Bit 3  (1) CRA.3 - ADCIN8
;Bit 2  (1) CRA.2 - ADCIN9
;Bit 1  (1) CRA.1 - ADCIN1
;Bit 0  (1) CRA.0 - ADCIN0

;    76543210
   SPLK#11110000b,OCRB

;OCRB - Output Control Register B
;Bit 7  (1) CRB.7 - CAP4
;Bit 6  (1) CRB.6 - CAP3
;Bit 5  (1) CRB.5 - CAP2/QEP2
;Bit 4  (1) CRB.4 - CAP1/QEP1
;Bit 3  (0) CRB.3 - BIO
;Bit 2  (0) CRB.2 - XF
;Bit 1  (0) CRB.1 - ?
;Bit 0  (0) CRB.0 - IOPC0


;-----------------------------------
; Set up ADC Module
;-----------------------------------
   LDP #224   ;DP = 224 Data Page for ADC Registers

;    5432109876543210
   SPLK#1000100100000000b,ADCTRL1


;ADCTRL1 - ADC Control Register 1
;Bit 15 (1) Suspend-SOFT -
;      Complete Conversion before halting emulator
;Bit 14 (0) Suspend-FREE -
;      Operations is determined by Suspend-SOFT
;Bit 13 (0) ADCIMSTART - ADC start converting immediately
;      Immediate Start of Conversion
;Bit 12 (0) ADC1EN - Enable/Disable ADC2
;      Disable ADC2
;Bit 11 (1) ADC2EN - Enable/Disable ADC1
;      Enable ADC1
;Bit 10 (0) ADCCONRUN - ADC Continuous Conversion Mode
;      Disable Continuous Conversion
;Bit 9  (0) ADCINTEN - Enable ADC Interrupt
;      No action when ADCINTFLAG is set
;Bit 8  (1) ADCINTFLAG - ADC Interrupt Flag
;      Clear Interrupt Flab Bit
;Bit 7  (0) ADCEOC - End of Conversion Bit  READ ONLY
;Bits 6-4 (000) ADC2CHSEL - ADC2 Channel Select
;      Channel 8
;Bits 3-1 (000) ADC1CHSEL - ADC1 Channel Select
;      Channel 0
;Bit 0  (0) ADCSOC - ADC Start of conversion bit
;      No Action
;    5432109876543210
   SPLK#0000000000000101b,ADCTRL2

;ADCTRL2 - ADC Control Register 2
;Bits 15-11 (00000)Reserved
;Bit 10 (0) ADCEVSOC - Event Manager SOC mask bit
;      Mask ADCEVSOC
;Bit 9  (0) ADCEXTSOC - External SOC mask bit
;      Mask ADCEXTSOC
;Bit 8  (0) Reserved
;Bits 7-6(00)ADCFIFO1 - Data Register FIFO1 Status  READ ONLY
;Bit 5  (0) Reserved
;Bits 4-3(00)ADCFIFO2 - Data Register FIFO2 Status  READ ONLY
;Bits 2-0(101) ADCPSCALE - ADC Input Clock Prescaler
;      Prescale Value 16
;      SYSCLK Period = 0.1usec
;      0.1usec x 16 x 6 = 9.6 usec >= 6usec


;The DAC module requires that wait states be generated for proper operation.

   LDP #0000h ;Set Data Page Pointer to 0000h, Block B2
   SPLK#4h,GPR0 ;Set Wait State Generator for
   OUT GPR0,WSGR ;Program Space, 0WS
       ;Date Space, 0WS
       ;I/O Space, 1WS

;===============================================================================
; Xfer the Twiddles to RAM Block B1 (# Twiddles = 96)
;===============================================================================
REPEAT LAR AR0, #95     ;set # of Twiddles to Xfer
   LAR AR1, #B1_SADR   ;AR1 points to 1st dest DM addr
   LACC#PS_TWID_STRT   ;ACC points to 1st Src PM addr

TWD_XFR  MAR *,1       ;select AR1
   TBLR*+, AR0      ;Mov Prog word --> Data mem
   ADD #1      ;inc ACC
   BANZTWD_XFR      ;continue Xfer until AR0=0

;===============================================================================
; Fetch 128 Data sample points from I/O port 0
;===============================================================================

FTCH_DATA LAR AR0, #128    ;AR0 = 128; # of Samples for Bit Reversed Addressing
   LAR AR1, #B0_SADR   ;AR1 = Start Address of B0; Data Buffer
   LAR AR2, #ADCFIFO1  ;AR2 = Value Converted from ADC
   LAR AR3, #(B1_SADR+128) ;AR3 = B1_SADR + 128
   LAR AR7, #127    ;AR7 = 128 - 1

   MAR *, AR2    ;ARP = AR2
   LDP #224     ;Set Bit 0 of ADCTRL1
   SBIT1 ADCTRL1,B0_MSK   ;Starts ADC converting
   LDP #232     ;Set Bit 6 of T1CON
   SBIT1 T1CON,B6_MSK  ;Starts the GP Timer 1

;==========================================================================
;Correction is made in the following data collection section
;     10 Jun 99
;==========================================================================

   CLRCSXM      ; Disable sign extension mode
          ; while reading in data from ADC
FTCH_LPBIT EVIFRA,BIT7   ;Polling routine to wait for
   BCNDFTCH_LP,NTC   ;T1PINT Flag to be Set
   LDP #224
   SBIT1 ADCTRL1,B0_MSK  ;Restart the ADC

       LACC  *,15,AR3  ;ACC = ADCFIFO1; ARP = AR3
   SACH *+,0,AR1    ;Value at location pointed by AR3
          ;= Value pointed to by AR2; ARP = AR1
   SACH *BR0+,0,AR7   ;Store sample into data buffer of B0 using
          ;bit reversed addressing; ARP = AR7
   LDP #232
   LACC EVIFRA    ;ACC = Event Manager Interrupt Flag Register
   SACL EVIFRA    ;Clear the flag register
   BANZ FTCH_LP,*-,AR2  ;Wait for next flag of T1PINT, or if 128
          ;samples have been collected, perform FFT.
          ;ARP = AR2

   SBIT0 T1CON,B6_MSK  ;Stop GP timer 1 - for FFT
   SPLK #0000h,T1CNT  ;Clear GP timer 1 counter

   SETC SXM      ; Enable sign extension mode

;===============================================================================
; Stages 1 & 2 - using the Radix 4 COMBO Macro
;===============================================================================
   MAR *, AR3
   LAR AR0, #7h  ;Increment for Data pointers
   LAR AR1, #(B0_SADR)
   LAR AR2, #(B0_SADR+2)
   LAR AR3, #(B0_SADR+4)
   LAR AR4, #(B0_SADR+6)
   LAR AR5, #B2_SADR ;Gen purp reg @ 60h
   LAR AR7, #(N/4-1) ;Loop 32 times
STAGE1_2_LP:
   COMBO
   BANZ   STAGE1_2_LP,*-,AR3

;===============================================================================
; Stage  3 - using ZEROI, PBY4I, PBY2I, P3BY4I Macros
;===============================================================================
STAGE3:
   MAR *, AR2   ;ARP-->AR2
   LAR AR0, #9h  ;
   LAR AR1, #(B0_SADR)  ;-->P
   LAR AR2, #(B0_SADR+8) ;-->Q
   LAR AR5, #B2_SADR ;Gen purp reg @ 60h
   LAR AR7, #(N/8-1) ;Loop counter (32 times)
   LT COS45

STAGE3_LP:
   ZEROI
   PBY4I
   PBY2I
   P3BY4I 7,0    ;-->AR7 at end, use *0+ modify.
   BANZ   STAGE3_LP,*-,AR2

;===============================================================================
; Stage  4 - using ZEROI, PBY4I, PBY2I, P3BY4I, BFLY Macros
;===============================================================================
STAGE4:
   MAR *, AR2    ;ARP-->AR2
   LAR AR0, #16    ;Used to inc Twiddle pointers
   LAR AR1, #(B0_SADR)    ;-->P
   LAR AR2, #(B0_SADR+16) ;-->Q
   LAR AR5, #B2_SADR   ;Gen purp reg @ 60h
   LAR AR7, #(N/16-1)  ;Loop counter (8 times)
```

```
STAGE4_LP:
   ZEROI

   LAR AR3, #(TWID_TBL+8+N/4)
   LAR AR4, #(TWID_TBL+8)
STG4_B1  BFLY 2

   LT COS45
   PBY4I

STG4_B2  BFLY 2

   PBY2I

STG4_B3  BFLY 2

   LT COS45
   P3BY4I 2,     ;-->AR2 at end

STG4_B4  BFLY 1

   ADRK #16
   MAR *, AR2
   ADRK #16
   MAR *, AR7
   BANZ STAGE4_LP,*-,AR2

;===============================================================================
; Stage  5 - using BUTTFLYI Macro
;===============================================================================
STAGE5:
   MAR *, AR2    ;-->AR2
   LAR AR0, #4     ;Used to Inc Twiddle pointers
   LAR AR1, #(B0_SADR)    ;-->P (0-->15)
   LAR AR2, #(B0_SADR+32) ;-->Q (16-->31)
   LAR AR5, #B2_SADR   ;Gen purp reg @ 60h

;-----------------------------------------------------------------------------
STG5_BLK1 LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/8-1)   ;Loop counter (16 times)

STAGE51_LP:
   BFLY 7
   BANZ STAGE51_LP,*-,AR2

;-----------------------------------------------------------------------------
STG5_BLK2:
   LAR AR1, #(B0_SADR+64) ;-->P (32-->47)
   LAR AR2, #(B0_SADR+96) ;-->Q (48-->63)
   LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/8-1)   ;Loop counter (16 times)

STAGE52_LP:
   BFLY 7
   BANZ STAGE52_LP,*-,AR2

;-----------------------------------------------------------------------------
STG5_BLK3:
   LAR AR1, #(B0_SADR+128) ;-->P (64-->79)
   LAR AR2, #(B0_SADR+160) ;-->Q (80-->95)
   LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/8-1)   ;Loop counter (16 times)
```

```
STAGE53_LP:
   BFLY 7
   BANZ STAGE53_LP,*-,AR2

;-----------------------------------------------------------------------------
STG5_BLK4:
   LAR AR1, #(B0_SADR+192) ;-->P (96-->111)
   LAR AR2, #(B0_SADR+224) ;-->Q (112-->127)
   LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/8-1)   ;Loop counter (16 times)

STAGE54_LP:
   BFLY 7
   BANZ STAGE54_LP,*-,AR2


;================================================================================
; Stage  6 - Using BFLY Macro
;================================================================================
STAGE6:
   MAR *, AR2    ;-->AR2
   LAR AR0, #2     ;used to Inc Twiddle pointers
   LAR AR1, #(B0_SADR)    ;-->P (0-->31)
   LAR AR2, #(B0_SADR+64) ;-->Q (32-->63)
   LAR AR5, #B2_SADR   ;Gen purp reg @ 60h

;-----------------------------------------------------------------------------
STG6_BLK1 LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/4-1)   ;Loop counter (32 times)

STAGE61_LP:
   BFLY 7
   BANZ STAGE61_LP,*-,AR2

;-----------------------------------------------------------------------------
STG6_BLK2:
   LAR AR1, #(B0_SADR+128) ;-->P (64-->95)
   LAR AR2, #(B0_SADR+192) ;-->Q (96-->127)
   LAR AR3, #(TWID_TBL+N/4) ;COS(angle)
   LAR AR4, #(TWID_TBL)   ;COS(angle+pi/4)
   LAR AR7, #(N/4-1)   ;Loop counter (32 times)

STAGE62_LP:
   BFLY 7
   BANZ STAGE62_LP,*-,AR2


;================================================================================
; Stage  7 - Using BFLY Macro
;================================================================================
STAGE7:
   MAR *, AR2    ;-->AR2
   LAR AR0, #1      ;Used to Inc Twiddle pointers
   LAR AR1, #(B0_SADR)     ;-->P (0-->63)
   LAR AR2, #(B0_SADR+128) ;-->Q (64-->128)
   LAR AR5, #B2_SADR   ;Gen purp reg @ 60h

   LAR AR3, #(TWID_TBL+N/4)
   LAR AR4, #(TWID_TBL)
   LAR AR7, #(N/2-1)   ;Loop counter (64 times)

STG_7_LP BFLY 7
```

```
   BANZ   STG_7_LP,*-,AR2

;===============================================================================
; Convert Real/Img into Magnitude ( Mag = sqr(Xr[n]) + sqr(Xi[n]) )
;===============================================================================
   LAR AR1, #B0_SADR
   LAR AR2, #B1_SADR
   LARK AR3, #127    ;loop 128 times
   LAR AR4, #(B1_SADR+128)
   MAR *, AR1

MAG_LP ZAC
   MPYK 0
   SQRA *+     ;Xr[n]**2
   SQRA *+,AR2    ;Xi[n]**2
   APAC
   SACH *+,0,AR4
   SACH *+,0,AR3   ;XOUT(I) = X(I)**2 + Y(I)**2
   BANZ MAG_LP,*-,AR1

;===============================================================================
; Normalize Values of the FFT
;===============================================================================

   .bss MAX,1
   .bss QUOTIENT,1

   .text
   NOP
   LAR AR1,#(B1_SADR+128)
   LAR AR2,#127

   LDP #0
   SPLK #0,MAX    ;Initialize the variable MAX
   SPLK #0,QUOTIENT   ;Initizlize the variable QUOTIENT

;The following section finds the maximum value among the FFT magnitudes

   MAR *,AR1     ;ARP = AR1
FIND_MAX LACC *+,0,AR2   ;ACC = Value pointed by AR1; AR1 = AR1+1
   SUB MAX     ;Subtract MAX
   BCND RESUME,LEQ    ;If the result is less than zero, then
          ;the value is not larger than MAX, else
   ADD MAX
   SACL MAX     ;Store the new MAX value

RESUME BANZ FIND_MAX,*-,AR1  ;Go through all the magnitudes to find the
          ;maximum value, then normalize the values

;The following section makes everything a ratio according to the maximum value

   LAR AR1,#(B0_SADR+256) ;AR1 = B1_SADR
   LAR AR2,#127    ;AR2 = 128-1; Number of Magnitudes to Normalize
   LAR AR3,#(B1_SADR+128) ;AR3 = B1_SADR + 128
   LAR AR4,#MAX    ;AR4 = Address for MAX value
   LAR AR5,#QUOTIENT   ;AR5 = Address for Q15 quotient
   LAR AR6,#15   ;AR6 = 16 - 1; Number of times to subtract

   MAR *,AR5     ;ARP = AR5
DIVIDING LACC *     ;ACC = QUOTIENT
   SACL *,1,AR3   ;QUOTIENT = QUOTIENT * 2; 1st shift doesn't matter
          ;because QUOTIENT = 0
   LACC *,0,AR4   ;ACC = Value pointed by AR3
   SUB *     ;Subtract MAX
   BCND ADD_ONE,GEQ   ;If ACC is still positive, then increment
```

```
          ;the ones place of the quotient, and shift
          ;the remainder in the ACC, else
          ;shift the remainder in the ACC


;Check the following section to reduce unnecessary instructions

   ADD *,AR3

   SACL *,1,AR6   ;Store the remainder * 2 to location
          ;pointed by AR3
   BANZ DIVIDING,*-,AR5  ;AR6 = AR6 - 1; Repeat the dividing
   B NEXT_VALUE    ;After repeating 16 times, fetch another
          ;value

ADD_ONEMAR *,AR3     ;ARP = AR3
   SACL *,1,AR5   ;Store the Remainder shifted by 1 back
          ;into the buffer
   LACC *     ;ACC = Quotient
   ADD #1     ;Increment the quotient
   SACL *,0,AR6   ;Store the new value of quotient; ARP = AR6
   BANZ DIVIDING,*-,AR5  ;AR6 = AR6 - 1; Repeat the dividing
   B NEXT_VALUE

NEXT_VALUE  LAR AR6,#15   ;Reset AR6 to 15
   MAR *,AR3     ;ARP = AR3
   SACL *+,0,AR5   ;Store the quotient into the buffer
   SPLK #0,*,AR2   ;Clear the variable QUOTIENT
   BANZ DIVIDING,*-,AR5  ;Repeat the dividing routine for all 128 values

;The following section corrects the value obtained when the maximum value is
;divided by itself. Changes 8000h to max Q15 value of 7FFFh

   LAR AR1,#(B1_SADR+128)
   LAR AR2,#127

   MAR *,AR1     ;ARP = AR1
CHANGING LACC *     ;ACC = Value pointed by AR1
   SUB #8000h    ;Subtract 8000h
   BCND CONTINUE,NEQ  ;If equal to 8000h, then
   LACC #0FFFFh   ;ACC = FFFFh, else
CONTINUE ADD #8000h
   SACL *+,0,AR2   ;Check the next value
   BANZ CHANGING,*-,AR1  ;Go through the entire buffer


;The following section multiplies the Q15 numbers by the maximum value (FFFh)
;of the DAC on the EVM

   LAR AR1,#(B1_SADR+128)
   LAR AR2,#127

   MAR *,AR1     ;ARP = AR1
NORMALIZE LT *     ;TREG = Value pointed by AR1
   MPY #0FFFh    ;Multiply by FFFh
   PAC       ;ACC = PREG
   SACH *+,1,AR2   ;Store the high word, elimnate extra sign bit
   BANZ NORMALIZE,*-,AR1   ;Normalize all of the magnitudes

;==============================================================================
; Output the values on the DAC of the EVM
;==============================================================================

   LAR AR1, #B1_SADR ;AR1 = Beginning of Un-normalized magnitudes
   LAR AR2, #63  ;AR2 = 64 - 1; Display first 64 magnitudes
   MAR *,AR1    ;ARP = AR1
   LDP #0

   SPLK #2,uSEC  ;Variable uSEC = 2


;The following section outputs the un-normalized values

;OUTPUTSETC XF      ;Set XF high
;  LACC *+,AR2    ;ACC = value pointed by AR1
;  SACL DAC0VAL   ;DAC0VAL = ACC
;  OUT DAC0VAL,DAC0  ;Write value to channel 0 of DAC
;  OUT DAC0VAL,DACUPDATE   ;Output the value
;  CALL DELAY     ;Wait
;  MAR *,AR2     ;ARP = AR2
;  BANZ OUTPUT,*-,AR1   ;Output the remaining values
;  CLRC XF     ;Clear XF once all values have been output

   LAR AR1, #(B1_SADR+128) ;AR1 = Beginning of normalized magnitudes
   LAR AR2, #63    ;AR2 = 64 -1; Display first 64 magnitudes
   MAR *,AR1     ;ARP = AR1

;The following section outputs the normalized values

OUTPUT2SETC XF      ;Set XF high
   LACC *+,AR2    ;ACC = value pointed by AR1
   SACL DAC0VAL   ;DAC0VAL = ACC
   OUT DAC0VAL,DAC0  ;Write value to channel 0 of DAC
   OUT DAC0VAL,DACUPDATE   ;Output the value
   CALL DELAY     ;Wait
   MAR *,AR2     ;ARP = AR2
   BANZ OUTPUT2,*-,AR1  ;Output the remaining values


   SPLK  #0,DAC0VAL    ;DAC0VAL = 0
   OUT DAC0VAL,DAC0  ;Clear channel 0 of DAC
   OUT DAC0VAL,DACUPDATE   ;Output 0V

       CLRC XF      ;Clear XF
   B REPEAT     ;Go back and wait for next 128 samples

DEND  B DEND

;======================================================================
; Routine Name:  DELAY
;
; Description:Produces a multiple of 1.6uS delays using the RPT
;   instruction. The Delay produced is based on the
;   value loaded in uSEC (i.e. Delay = uSEC x 1.6mS).
;   Indirect addressing is used to count the number
;   of times the delay loop is repeated.
;
; Calling Convention:
;
; Variables   on Entry   on Exit
; ----------------------------------------------------------------------
;   DP    XX     0x0000
;   ARP   XX     AR7
;   ACC   XX     XX
;   uSEC       value in 1.6 uS  un-touched
; ----------------------------------------------------------------------
;======================================================================
DELAY: LDP #0h      ;DP-->0000h-007Fh
   LACC #32
   SACL RPT_NUM   ;RPT_NUM = 32
   LAR AR7,uSEC    ;Set AR0 to generate a
   MAR *,AR7     ;(AR0*0.1)mSEC delay loop

DELAY_LOOP: LDP #0h      ;DP-->0000h-007Fh
   RPT RPT_NUM   ;32 cycles = 1.6uS
   NOP       ;1 cycle
   BANZ DELAY_LOOP    ;Repeat DELAY_LOOP
   RET       ;Return from DELAY SR

;===============================================================================
; I S R  -  PHANTOM
;
; Description:Dummy ISR, used to trap spurious interrupts.
;
; Modifies:
;
; Last Update:16-06-95
;===============================================================================
PHANTOM  B PHANTOM

;===============================================================================
; TWIDDLES FOR N=128 FFT (96 entries)
;===============================================================================
PS_TWID_STRT:
     .word 00000h     ;    0.000 ø
     .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
     .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
PS_TWID_END:
     .word 08027h     ;  267.188 ø
```

TI Contact Numbers