EVM Application #9

Controlling a Sine Wave Generator with the Serial Port Using the TMS320F240 EVM

APPLICATION REPORT: SPRA418

David Figoli

Digital Signal Processing Solutions
January 1999

TEXAS INSTRUMENTS

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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

EVM Application #9

Controlling a Sine Wave Generator with the Serial Port Using the TMS320F240 EVM

Abstract

This application report explains how the EVM Application #9 creates a 2 channel sine wave generator using the 12 bit digital-to-analog converter (DAC) of the Texas Instruments (TI™) TMS320F240 Evaluation Module (EVM). The document contains:

- An overview that includes information on the commands that control the sine waves
- Information on each of the three modules used
- Details on how to interpret transmitted ASCII values
- Commands to modify the frequency, magnitude and phase difference of the DAC0 and DAC1 channels of the sine wave generator
- The C2xx Assembly code which implements the application

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to dsph@ti.com. Questions receive prompt attention and are usually answered within one business day.

Overview

This application creates a 2 channel sine wave generator using the 12 bit digital-to-analog converter (DAC) of the TI TMS320F240 EVM. The generator modifies the frequency of the sine waves, the phase difference between the two waves, and the magnitude (peak to peak) of the waves according to a user's input commands.

Each channel of the sine wave generator is controlled using a terminal program (e.g., Windows® Terminal.exe) that uses a serial port to communicate with the serial port of the EVM. The serial port allows the frequency and magnitude of each channel to be controlled and the phase difference between the sine waves to be changed by entering commands in the terminal.

The commands control the sine waves:

First Letter	Second Letter	Third Value (if a or b)
f - frequency change	a - DAC0 Sine wave	xxxx - value
m - magnitude change	b - DAC1 Sine wave	
p - phase change	r - reset to initial values	

<CR> - enter needs to be pressed after each command

The application is implemented using C2xx Assembly code.

Module(s) Used

- Event Manager module
- General Purpose Timer 1
- SCI module

Input

PC serial port (SCI)

Output

DAC0OUT

DAC1OUT

Background and Methodology

The implementation of a dual channel sine wave generator with the DAC controlled via a terminal program is a modification of Application #3 (DAC0.ASM). To control the sine waves of the DAC through the serial port of the EVM, the SCI module of the F240 needs to be used. The sine wave values can be controlled using a terminal program and connecting the serial port of the EVM to the serial port of a PC.

The sine waves in this application are created using the same method described in Application #3. The sine waves are generated using the modulo counting register, interpolation, and a 256 point sine look up table.

A rolling 16 bit counter is used to determine the location of the value to be placed in the DAC. A step value is added to the counter every time a new value from the sine table is to be loaded. By changing the value of the step, one can reasonably control the frequency of the sine wave. However, to be able to calculate the frequency that will be produced, a “sampling” method is used as in the previous application. The “sampling” is accomplished using a polling routine.

A value is set in the period register of GP Timer 1. Because the flags of the interrupt register in the Event Manager are set regardless of whether the interrupts are masked or unmasked, the flag register can be watched until the period flag is set. Once the period flag is set, then the program can determine the next value to load into the output register of the DAC. Once the values have been updated, then the flag register is cleared, and the program returns to the polling routine to await for the next flag to be set.

Since a polling routine is used for the Event Manager module, only one interrupt routine needs to be made to service interrupts generated by the SCI module of the F240.

SCI Module

The SCI module of the F240 uses SYSCLK to derive its own clock, as a result, one needs to know the settings of the PLL module. By knowing what the PLL module is set up to generate, then baud rate of the SCI module can be selected. In this application the PLL generates a CPUCLK of 20MHz and a SYSCLK of 10MHz.

Since the controls for the sine waves will only be ASCII (7 bits) values, the data stream of the SCI can be set up to use 7 character bits, 1 stop bit, and odd parity. The SCI is set up to be able to receive and transmit data to the terminal using the internal clock as its clock source. Additionally, since the receiving of data is what causes the program to change the values in the sine registers, the receive interrupt is enabled whereas, the transmit interrupt is disabled. The baud rate of the SCI is set to 19200.[1]

Interpreting Transmitted ASCII Values

The values transmitted to the DSP are the corresponding ASCII values for each character entered into the terminal. Before the interpretation of the ASCII values can be performed, certain design aspects need to be considered before the algorithm can be developed.

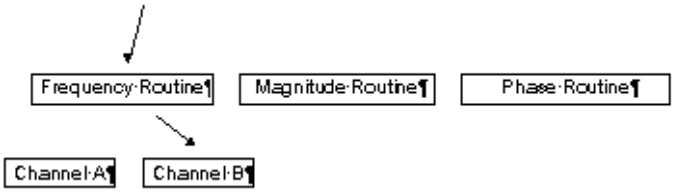
In this application, all entries are terminated by a carriage return (ASCII value - 0Dh). As a result, the values that are entered into the terminal need to be stored into a memory location until the carriage return is entered into the terminal. As a result, before anything is stored into the buffer, it is cleared to null values (ASCII value - 00h).

Once the buffer is cleared, then the data from the terminal can be input. When a carriage return is entered, then the program starts interpreting the input values. The first value that can be input is either an ‘f’, ‘m’, or ‘p’. The entered values can either be lowercase or uppercase because the program is case insensitive. The ASCII values for ‘f’, ‘m’, and ‘p’, are 66h, 6Dh, and 70h respectively. The program changes the case of the values to ‘F’ (46h), ‘M’ (4Dh), and ‘P’ (50h), by ANDing the input ASCII values with 5Fh. To prevent changing the values of the input numbers with the AND operation, the AND operation is only performed on values that have a 1 in the 6^{th} bit position. If the value is a 1 then the ASCII equivalent is a letter, if the value is a 0 then the ASCII equivalent is a terminal command of a number.

[1] Note: The SCI configuration should not be set or altered unless the SW RESET bit is cleared. Set up all configuration registers before setting SW RESET; otherwise, unpredictable behavior can occur.

Once the first value is interpreted by the program, the appropriate routine is executed to modify the frequency, phase, or magnitude. If the first value is not a valid input value, then the routine terminates, outputs an error message, and awaits another for another carriage return to interpret another set of input data from the terminal. If the first value is a valid value, then the next value is interpreted to determine which sine value to manipulate. Once the first two values have been processed, the remaining values are numerical.

```
Input value:      fb250<CR>
                  f  b  2  5  0  <CR>
Input buffer:     0066  0062  0032  0035  0030  000d
AND with 5F
                  F  B  2  5  0  <CR>
Input buffer:     0046  0042  0032  0035  0030  000d
```



In ASCII, the character values that correspond to 30h to 39h are the numbers 0 to 9. As a result, to convert the ASCII values, 30h needs to be subtracted. Once the ASCII values have been converted to digits, then the actual input number value can be determined. To convert the values, a routine multiplies a digit by 10 and then adds the next digit; this process continues until a carriage return is encountered. Once the carriage return is encountered, the input value has been completely converted from the input decimal value to the equivalent hexadecimal value.

```
ACC (hex)   Next Digit  x10      + Next Digit
00000000 2     00000000 00000002
00000002 5     00000014 00000019
00000019 0     000000FA 000000FA
000000FA <CR>

FAh = 250
```

This application uses the same equations as Application #3 to determine the value to load into the appropriate registers with values to control the sine waves. However, instead of entering the step value, displacement, and Q15 magnitude, the user can input the actual frequency, phase difference, and magnitude desired.

Frequency Manipulation

To modify the frequency of the DAC0 channel of the sine wave generator, the user enters

fa*frequency*<CR>

or

fb*frequency*<CR>

to modify DAC1. The entered *frequency* should be in Hertz. The application uses the following equation to determine the step size that should be stored into the step registers of the sine waves.

$$step = f\left(T_S \times 2^n\right)$$

where

f = desired frequency

T_S = the period of the frequency at which the DAC register values are updated

step = the step increment that is added to the counter to change the frequency

n = the number of bits in the counting register

To return the sine waves to the same frequency that the program started with, the user should enter

fr<CR>

In this application, the value for T_S is equivalent to the Q15 value 0001h. The smallest Q15 value was chosen to allow for the greatest range of allowable steps that can be used for manipulating the frequencies. A Q15 period of 0001h seconds equates to a 0.000031 second period which is 32.768kHz. As a result, the timers in the Event Manager are set up to update the DAC values at a rate of 32.768kHz.

Once the input value is converted, the input value is multiplied by 0001h and 2^{16} to obtain the step value. Once the step value is obtained, the value is loaded into the appropriate channel and the program resumes back to the main line.

Phase Manipulation

To modify the phase difference between DAC0 and DAC1 of the sine wave generator, the user enters

pa*phase*<CR>

or

pb*phase*<CR>

The entered phase should be in degrees. The designation after the p(a or b), determines which sine wave will lead. If a is entered, then DAC0 will lead DAC1 by *phase* degrees; the same applies to if b is entered.

Once the value for the phase difference has been converted from the input value from the terminal, the converted phase in hex is checked to make sure it is an angle between 0 to 360 degrees. If the value is in between 0 and 360 degrees, then the program continues, if the value is outside the range, then the routine ends and nothing is changed.

To determine the offset value between the two registers, the input phase is divided by 360 degrees and then multiplied by 2^{16}. (Refer to Application #8 about division). The channel that is chosen determines the register used. The calculated displacement is added to whichever channel is to lag, and the new value is loaded into the channel that is supposed to lead.

To calculate the displacement value to load into the modulo register of the corresponding sine wave, the program uses the following formula

$$d(\phi) = \frac{\phi}{360^\circ} \times 2^n$$

where

$d(\phi)$ = displacement value for the modulo counter register

ϕ = desired phase difference in degrees

n = number of bits in the counter register

To return the sine waves to a phase difference of 0 degrees, the user should enter **pr**<CR>**.**

Magnitude Manipulation

To modify the magnitude of the sine waves output on DAC0 and DAC1, the user enters

ma*magnitude*<CR>

or

mb*magnitude*<CR>.

The entered peak to peak *magnitude* can only be in tenths of a volt. The value should be entered without the decimal point. For example, to obtain a 4.0V_{pp} value on DAC0, the user should enter

ma40<CR>

Once the value for the magnitude has been converted from the input value from the terminal, the converted phase in hex is checked to make sure that the magnitude is between 0 and 50. If the value is inside the limits, then the program continues, if the value is outside the range, then the routine ends and nothing is changed.

The value to load into the magnitude register is determined by the dividing the input value by 50 (See Application #8). The Q15 quotient is then multiplied by 2^{15}-1 (i.e. the most positive Q15 value). The resulting Q15 value is then stored into the appropriate sine wave's magnitude register.

The following formula provides the value to be used in the magnitude register of the appropriate sine wave.

$$A(m) = \frac{m}{50} \times \left(2^{Q} - 1\right)$$

where

A(m) = value for the magnitude register

m = desired peak to peak voltage

Q = Q format used (e.g. 16 bit word size, Q = 15)

To return the sine waves to a 5V_{pp} magnitude, the user should enter **mr**<CR>**.**

Examples of input commands:

fa100<CR>	Sets DAC0 sine wave to 100Hz
fb600<CR>	Sets DAC1 sine wave to 600Hz
fr<CR>	Resets DAC0 and DAC1 to original value - 500Hz
ma35<CR>	Sets DAC0 sine wave Vpp to 3.5 volts
mb23<CR>	Sets DAC1 sine wave Vpp to 2.3 volts
mr<CR>	Resets DAC0 and DAC1 to Vpp of 5V
	maximum magnitude = 50 = 5.0Vpp
pa45<CR>	Sets DAC0 sine wave to lead by 45 degrees
pb60<CR>	Sets DAC1 sine wave to lead by 60 degrees
pr<CR>	Resets DAC0 and DAC1 so phase difference is 0 degrees
	maximum phase difference = 360

Since the SCI is used to modify the values in the registers that control the sine waves and because the program uses an interrupt to update the values, the program can be run and modified indefinitely.

```
;*****************************************************************
; File Name:      scidac0.asm
; Originator:     Digital control systems Apps group - Houston
; Target System:  'C24x Evaluation Board
;
; Description:    Outputs 2 Sine Waves on the EVM DAC - DAC0 and DAC1
;                 Sine waves generated through a look up table and
;                 interpolation.
;
;                 The sine waves can be controlled through the
;                 computer by using a terminal program that uses the
;                 serial port of the computer to communicate with the
;                 DSP through the EVM's serial port.
;
;                 To control the sine waves, the terminal program
;                 needs to be set to a baud rate of 19200, 1 stop
;                 bit, 7 character  bits, and odd parity.
;
;                 Entering the following commands will allow control
;                 over the sine waves
;
;                 First letter:
;                 f-frequency    m-magnitude    p-phase
;                 Second letter:
;                 a-DAC0 sine wave  b-DAC1 sine wave  r-reset to
;                 initial value
;                 If second letter is an a or b:
;                 xxxx-value
;                 <CR>-enter needs to be pressed after each command
;
;                 Examples:
;
;                 fa100 - Sets DAC0 sine wave to 100Hz
;                 fb600 - Sets DAC1 sine wave to 600Hz
;                 fr - Resets DAC0 and DAC1 to original value - 500Hz
;
;                 ma35 - Sets DAC0 sine wave Vpp to 3.5 volts
;                 mb23 - Sets DAC1 sine wave Vpp to 2.3 volts
;                 mr - Resets DAC0 and DAC1 to Vpp of 5V
;                 maximum magnitude = 50 = 5.0Vpp
;
;                 pa45 - Sets DAC0 sine wave to lead by 45 degrees
;                 pb60 - Sets DAC1 sine wave to lead by 60 degrees
;                 pr - Resets DAC0 and DAC1 so phase difference is 0
;                 degrees
;                 maximum phase difference = 360
;
; Last Updated:   9 July 1997
;
;*****************************************************************
                  .include f240regs.h

;*****************************************************************
; File Name:   TMS320x240 SCI Idle-line Mode Example Code
```

```
;
; Description: This program uses the SCI module to implement a
; simple asynchronous communications routine.  The SCI is
; initialized to operate in idle-line mode with 7 character
; bits, 1 stop bit, and odd parity.  The SCI Baud Rate
; Registers (BRR) are set to transmit and receive data at
; 19200 baud. The SCI generates an interrupt every time a
; character is loaded into the receive buffer (SCIRXBUF).
; The interrupt service routine(ISR) reads the receive
; buffer and determines if the carriage return button, <CR>,
; has been pressed.  If so, the character string "Ready" is
; transmitted. If not, no character string is transmitted.
;*****************************************************************


;-----------------------------------------------------------------
; 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

;-----------------------------------------------------------------
; Constant definitions
;-----------------------------------------------------------------
LENGTH1              .set  00007h   ;Length of the data stream to be
                                    ;transmitted
LENGTH2              .set  16

;-----------------------------------------------------------------
; Variable definitions
;-----------------------------------------------------------------
;     bss DATA_OUT,LENGTH         ;Location of LENGTH byte character
                                  ;stream to be transmitted
                  .bss  GPR0,1         ;General purpose register.
                  .bss  DAC0VAL,1      ;DAC0 Channel Value
                  .bss  DAC1VAL,1      ;DAC1 Channel Value
                  .bss  DAC2VAL,1      ;DAC2 Channel Value
                  .bss  DAC3VAL,1      ;DAC3 Channel Value

;-----------------------------------------------------------------
; Initialized Transmit Data for Interrupt Service Routine
;-----------------------------------------------------------------
                     .data
READY             .word  0052h      ;Hex equivalent of ASCII character
'R'
                  .word  0065h      ;Hex equivalent of ASCII character
'e'
                  .word  0061h      ;Hex equivalent of ASCII character
'a'
                  .word  0064h      ;Hex equivalent of ASCII character
'd'
```

```
                .word  0079h      ;Hex equivalent of ASCII character
'y'
                   .word  000dh      ;Hex equivalent of ASCII <CR>
                   .word  0000h      ;Hex equivalent of ASCII NULL

INVALID_IP      .word 000ah    ;Hex equivalent of ASCII (NEW LINE)
                .word    0049h    ;Hex equivalent of ASCII character
'I'
                .word 006Eh    ;Hex equivalent of ASCII character 'n'
                .word 0076h    ;Hex equivalent of ASCII character 'v'
                .word 0061h    ;Hex equivalent of ASCII character 'a'
                .word 006Ch    ;Hex equivalent of ASCII character 'l'
                .word 0069h    ;Hex equivalent of ASCII character 'i'
                .word 0064h    ;Hex equivalent of ASCII character 'd'
                .word 0020h    ;Hex equivalent of ASCII (SPACE)
                .word 0069h    ;Hex equivalent of ASCII character 'i'
                .word 006Eh    ;Hex equivalent of ASCII character 'n'
                .word 0070h    ;Hex equivalent of ASCII character 'p'
                .word 0075h    ;Hex equivalent of ASCII character 'u'
                .word 0074h    ;Hex equivalent of ASCII character 't'
                .word 000dh    ;Hex equivalent of ASCII <CR>
                .word    0000h    ;Hex equivalent of ASCII NULL

;---------------------------------------------------------------------
; Vector address declarations
;---------------------------------------------------------------------
                .sect  ".vectors"
RSVECT          B    START     ; PM 0   Reset Vector   1
INT1            B    INT1_ISR  ; PM 2   Int level 1    4
INT2            B    PHANTOM   ; PM 4   Int level 2    5
INT3            B    PHANTOM   ; PM 6   Int level 3    6
INT4            B    PHANTOM   ; PM 8   Int level 4    7
INT5            B    PHANTOM   ; PM A   Int level 5    8
INT6            B    PHANTOM   ; PM C   Int level 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
                CLRC  SXM        ;Clear Sign Extension Mode
                CLRC  OVM        ;Reset Overflow Mode
                CLRC  CNF        ;Config Block B0 to Data mem.

SPLK            #0001h,IMR       ;Mask all core interrupts except
                                    ;INT1

                LACC  IFR         ;Read Interrupt flags
                ACL   IFR         ;Clear all interrupt flags

                LDP   #00E0h
                SPLK  #006Fh, WDCR ;Disable Watchdog if VCCP=5V
                KICK_DOG          ;Reset Watchdog counter

;=====================================================================
; Initialize B2 RAM to zero's.
;=====================================================================
            LAR   AR2,#B2_SADDR   ;AR2 -> B2 start address
            MAR   *,AR2           ;Set ARP=AR2
            ZAC                  ;Set ACC = 0
            RPT   #1fh            ;Set repeat cntr for 31+1 loops
            SACL  *+              ;Write zeros to B2 RAM
;=====================================================================
; Initialize B2 RAM to zero's.
;=====================================================================
            LAR   AR2,#B0_SADDR   ;AR2 -> B2 start address
            MAR   *,AR2           ;Set ARP=AR2
            ZAC                  ;Set ACC = 0
            RPT   #255            ;Set repeat cntr for 31+1 loops
            SACL  *+              ;Write zeros to B2 RAM
;=====================================================================
; Initialize B2 RAM to zero's.
;=====================================================================
            LAR   AR2,#B1_SADDR   ;AR2 -> B2 start address
            MAR   *,AR2           ;Set ARP=AR2
            ZAC                  ;Set ACC = 0
            RPT   #255            ;Set repeat cntr for 31+1 loops
            SACL  *+              ;Write zeros to B2 RAM

;=====================================================================
; Initialize DATAOUT with data to be transmitted.
;=====================================================================
            LAR   AR2,#B1_SADDR          ;Reset AR2 -> B1 start
                                         ;address
                  RPT   #(LENGTH1+LENGTH2-1);Set repeat counter for
                                         ;LENGTH1+LENGTH2 loops
            BLPD  #READY,*+              ;loads B2 with TXDATA
```

```
;=====================================================================
; INITIALIZATION OF INTERRUPT DRIVEN SCI ROUTINE
;=====================================================================
SCI_INIT:         LDP   #00E0h
                  SPLK  #0036h,SCICCR     ;1 stop bit, odd parity, 7
char
                                          ;bit
                                          ;async mode, idle-line protocol
                  SPLK  #0013h, SCICTL1   ;Enable TX, RX, internal
                                ; SCICLK,
                                ;Disable RX ERR, SLEEP, TXWAKE
                  SPLK  #0002h, SCICTL2   ;Enable RX INT,disable TX INT
                  SPLK  #0000h, SCIHBAUD
                  SPLK  #0040h, SCILBAUD  ;Baud Rate=19200 b/s (10 MHz
                                ;SYSCLK)
                  SPLK  #0022h, SCIPC2 ;Enable TXD & RXD pins
                  SPLK  #0033h, SCICTL1   ;Relinquish SCI from Reset.

                  LAR   AR0, #SCITXBUF ;Load AR0 with SCI_TX_BUF
                                ;address
                  LAR   AR1, #SCIRXBUF ;Load AR1 with SCI_RX_BUF
                                ;address
                  LAR   AR2, #B2_SADDR ;Load AR2 with TX data start
                                ;address

;=====================================================================
; INITIALIZATION OF PLL MODULE
;=====================================================================
                  LDP   #00E0h

;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.
```

```
                  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


;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;- 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  #0000h,CAPCON  ;Clear Capture Control

                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,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
*

;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

;=================================================================
; INITIALIZATION OF EVENT MANAGER MODULE
;=================================================================

T1COMPARE       .set  0        ;Compare value not necessary
T1PERIOD        .set  610      ;T1PERIOD set to value equivalent to
                              ;32.768kHz with CPULCLK = 20MHz
QT1PERIOD       .set  0001h    ;Q15 period value for period of
                              ;32.768kHz

                   .text
LDP             #232             ;DP=232, Data Page for Event Manager
                                 ;Addresses

SPLK            #T1COMPARE,T1CMPR    ;Initialize GP Timer 1 Compare
                                 ;Register

;               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    ;Initialize GP Timer 1 Period
Register
SPLK              #0000h,T1CNT         ;Initialize GP Timer 1
SPLK              #0000h,T2CNT         ;Initialize GP Timer 2
SPLK              #0000h,T3CNT         ;Initialize GP Timer 3


;                 5432109876543210
SPLK              #0001000000000100b,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         (01)  TCLD1,TCLD0 - Timer Compare Register Reload
;                       Condition
;                       When counter is 0 or equals period register
;                       value
;Bit 1            (0)   TECMPR - Timer compare enable
;                       Disable timer compare operation
;Bit 0            (0)   Reserved
;
;                 5432109876543210
SPLK              #0000000000000000b,T2CON   ;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  ;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

SBIT1              T1CON,B6_MSK      ;Sets Bit 6 of T1CON
                                ;Starts GP Timer 1

;T1CON - GP Timer 1 Control Register
;Bit 6            (1)   TENABLE - Timer Enable
;                       Enable Timer Operations

;---------------------------------------------------------------------
; Initialize Variables for Generation of Sine Wave on DAC
;---------------------------------------------------------------------
```

```
;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
                                          ;Data Space, 0WS
                                          ;I/O Space, 1WS

                    .bss  TABLE,1         ;Keeps address of the
                                          ;pointer in the SINE Table
                    .bss  TOPTABLE,1      ;Keeps the reset value for
                                          ;the pointer
                    .bss  COMPARET1,1     ;A register to do
                                          ;calculations since the
                                          ;T1CMPR register is double
                                          ;buffered
                    .bss  REMAINDER,1     ;Remainder of the MODREGx
                                          ;values
                    .bss  VALUE,1         ;SINE Table Value
                    .bss  NEXTVALUE,1     ;Next entry in the SINE Table
                    .bss  DIFFERENCE,1    ;Difference between Entries


                    .bss  FREQSTEP1,1     ;Frequency modulation of the
                                          ;1st sine wave
                    .bss  MODREG1,1       ;Rolling Modulo Register for
                                          ;1st sine wave
                    .bss  MAG1,1          ;Magnitude of the frequency
                                          ;for 1st sine wave

                    .bss  FREQSTEP2,1     ;Frequency modulation of the
                                          ;2nd sine wave
                    .bss  MODREG2,1       ;Rolling Modulo Register for
                                          ;2nd sine wave
                    .bss  MAG2,1          ;Magnitude of the frequency
                                          ;for 2nd sine wave

                    .bss  TEMP,1          ;Register to hold temporary
                                          ;values
                    .bss  NEW_VALUE,1     ;New value to load into
                                          ;the appropriate register

                    .bss  PREV_VALUE,1    ;Previous value before
                                          ;being changed

                    .bss  DIVISOR,1
                          .text
                    NOP
                    SPLK  #0000h,TABLE
                    SPLK  #STABLE,TOPTABLE

                    SPLK  #1000,FREQSTEP1   ;Controls the frequency
```

```
                              ;for DAC0
                    SPLK  #0000h,MODREG1    ;Sets the starting point
                    SPLK  #7FFFh,MAG1       ;Maximum value, Q15

                    SPLK  #1000,FREQSTEP2   ;Controls the frequency
                              ;for DAC1
                    SPLK  #0000h,MODREG2    ;Sets the starting point
                    SPLK  #7FFFh,MAG2       ;Maximum value, Q15

                    SPLK  #0000h,TEMP       ;Initialize temporary
                              ;register
                    SPLK  #0000h,NEW_VALUE  ;Initialize new value
                    SPLK  #0000h,PREV_VALUE ;Initialize previous
                              ;value

                    SPLK  #0000h,DIVISOR    ;Initialize the maximum
                              ;register
                    SPLK  #0000h,QUOTIENT   ;Initialize the quotient

               LAR   AR0, #SCITXBUF  ;Load AR0 with SCI_TX_BUF
                              ;address
               LAR      AR1, #SCIRXBUF   ;Load AR1 with SCI_RX_BUF
                              ;address
               LAR      AR2, #B1_SADDR       ;Load AR2 with TX data
                                 ;start address

                    LAR      AR3, #B0_SADDR

                    CLRC     INTM

WAITING             LDP   #232
                    BIT   EVIFRA,BIT7          ;Polling routine to wait
for
                    BCND  WAITING,NTC          ;T1PINT Flag to be Set

;---------------------------------------------------------------------
; Generate Sine Wave
;---------------------------------------------------------------------

;The following section performs the necessary calculations for the
;first sine wave

SINE                LDP   #0
                    LACC  MODREG1     ;ACC loaded with the counting
                              ;register
                    ADD   FREQSTEP1   ;Counting Register increased by
                              ;specific step
                    SACL  MODREG1     ;Store the updated counter value
                    LACC  MODREG1,8   ;Reload the new ctr value but
                              ;shift left by 8 bits
                    SACH  TABLE       ;Store the high bit pointer to
lookup
                              ;table
                    SFR               ;Shift the value to the right
                              ;convert to Q15
```

```
                AND   #07FFFh     ;Make sure the Q15 value is
                                  ;positive
                SACL  REMAINDER   ;Store the fractional value of
                                  ;the counting register
                LACC  TABLE       ;Load the accumulator with the
                                  ;proper index value
                ADD   TOPTABLE    ;Displace the ACC with the
                                  ;starting address
                TBLR  VALUE       ;Read the value from the table
                                  ;and store into VALUE
                ADD   #1          ;Increment the ACC to the next
                                  ;address
                TBLR  NEXTVALUE   ;Read the next value from the
                                  ;table and

                LACC  NEXTVALUE   ;Load the ACC with NEXTVALUE
                SUB   VALUE       ;Subtract the previous value
                SACL  DIFFERENCE  ;Store the difference between
                                  ;the values

                LT DIFFERENCE  ;Load the TREG with DIFFERENCE
                MPY   REMAINDER   ;Multiply the DIFFERENCE with
                                  ;REMAINDER
                PAC               ;Move the product to the ACC
                SACH  REMAINDER,1 ;Store the upper byte and shift
                                  ;left by 1, Q15
                LACC  REMAINDER   ;Load ACC with new REMAINDER
                ADD   VALUE       ;Add VALUE to get the new
                                  ;interpolated value
                SACL  VALUE       ;Store the interpolated value
                                  ;into VALUE

                LT VALUE       ;Load the TREG with the new
                               ;interpolated VALUE
                MPY   MAG1        ;Multiply VALUE by a magnitude
                PAC               ;Move the product to ACC
                SACH  DAC0VAL,1   ;Store the new value, shift to
                                  ;get Q15

;The following section performs the necessary calculations for the
;second sine wave

                LACC  MODREG2     ;ACC loaded with the counting
                                  ;register
                ADD   FREQSTEP2   ;Counting Register increased by
                                  ;specific step
                SACL  MODREG2     ;Store the updated the counter value
                LACC  MODREG2,8   ;Reload the new ctr value, shift
left
                                  ;by 8 bits
                SACH  TABLE       ;Store the high bit as pointer to
                                  ;lookup table
                SFR               ;Shift the value to the right
convert
                                  ;to Q15
```

```
                AND   #07FFFh     ;Make sure the Q15 value is positive
                SACL  REMAINDER   ;Store the fractional value of the
                                  ;counting register
                LACC  TABLE       ;Load the accumulator with the
proper
                                  ;index value
                ADD   TOPTABLE    ;Displace the ACC with the starting
                                  ;address
                TBLR  VALUE       ;Read the value from the table and
                                  ;store into VALUE
                ADD   #1          ;Increment the ACC to the next
                                  ;address
                TBLR  NEXTVALUE   ;Read the next value from the table
                                  ;and store

                LACC  NEXTVALUE   ;Load the ACC with NEXTVALUE
                SUB   VALUE       ;Subtract the previous value
                SACL  DIFFERENCE  ;Store the difference between the
                                  ;value

                LT DIFFERENCE  ;Load the TREG with DIFFERENCE
                MPY   REMAINDER   ;Multiply the DIFFERENCE with
                                  ;REMAINDER
                PAC               ;Move the product to the ACC
                SACH  REMAINDER,1 ;Store the upper byte and shift left
                                  ;by 1, Q15
                LACC  REMAINDER   ;Load ACC with new REMAINDER
                ADD   VALUE       ;Add VALUE to get the new
                                  ;interpolated value
                SACL  VALUE       ;Store the interpolated value into
                                  ;VALUE

                LT VALUE          ;Load the TREG with the new
                                  ;interpolated VALUE
                MPY      MAG2        ;Multiply VALUE by a magnitude
                PAC                  ;Move the product to ACC
                SACH     DAC1VAL,1   ;Store the new value, shift to
get
                                     ;Q15

                LDP   #0             ;This section outputs the SINE
wave
                                     ;to the DAC
                LACC  DAC0VAL        ;ACC = DAC0VAL - entry from the
                                     ;lookup table
                ADD   #8000h         ;Displace the value half the
maximum
                SFR                  ;Shift over 4 places since the DAC
is
                                  ;12bits
                SFR
                SFR
                SFR
                SACL  DAC0VAL     ;Store the new 12 bit value into
                                  ;DAC0VAL
```

```
                LACC  DAC1VAL     ;ACC = DAC0VAL - entry from the
                                ;lookup table
                ADD   #8000h      ;Displace the value half the maximum
                SFR         ;Shift over 4 places since the DAC is
                          ;12bits
                SFR
                SFR
                SFR
                SACL  DAC1VAL     ;Store the new 12 bit value into
                          ;DAC0VAL

                OUT   DAC0VAL,DAC0         ;Stores the 12 bit value
                                ;into DAC0 register
                OUT   DAC1VAL,DAC1         ;Stores the 12 bit value
                                ;into DAC1 register
                OUT   DAC0VAL,DACUPDATE ;Causes the DAC to output
                                ;the value


RESUME          LDP   #232        ;DP = 232 - DP for Event Manager
                LACC  EVIFRA      ;Load EVIFRA - Type A Interrupt
Flags
                SACL  EVIFRA      ;Clear the Interrupt Flags
                B  WAITING

;=====================================================================
; I S R  -  INT1_ISR
;
; Description:   The INT1_ISR first determines if the SCI RXINT
;                 caused the interrupt.  If so, the SCI Specific ISR
;                 reads the character in the RX buffer.  If the
;                 character received corresponds to a carriage
;                 return, <CR>, the character string "Ready" is
;                 transmitted. If the character received does NOT
;                 correspond to a carriage return, <CR>, then the ISR
;                 returns to the main program without transmitting a
;                 character string.  If the SCI RXINT did not cause
;                 an interrupt, then the value '0x0bad' is stored in
;                 the accumulator and program gets caught in the
;                 BAD_INT endless loop.
;=====================================================================

INT1_ISR:   LDP      #00E0h         ;DP = 224 Address 7000h-707Fh
            LACL     SYSIVR         ;Load peripheral INT vector
                                    ;address
            LDP      #0000h         ;DP = 0 Addresses 0000h-007Fh
            SUB      #0006h         ;Subtract RXINT offset from above
            BCND     RX_ISR,EQ      ;verify RXINT initiated interrupt
            B        BAD_INT        ;Else, bad interrupt occurred

RX_ISR          MAR   *,AR1          ;ARP = AR1
                LACC  *              ;Load ACC w/RX buffer character
                BIT   *,BIT6,AR3     ;Determine if the character is a
                                ;letter
```

```
                BCND  NUMBER,NTC
                AND   #01011111b     ;If a letter, capitalize the
                                     ;letter
NUMBER             SACL  *+          ;Store the character/number

IP_VALUE        SUB   #000Dh         ;Check to see if <CR>
                BCND  CHECK_IP,EQ    ;If value entered is a <CR>, then
                                     ;process input
                   B  NO_IP          ;else, wait until <CR> is pressed

CHECK_IP        LAR   AR3,#B0_SADDR     ;AR3 = Address of first value
                                        ;entered
                LACC  *+                ;ACC = ASCII equivalent of
                                        ;value

                SUB   #0046h            ;Check to see if ASCII letter
                                        ;'F'
                BCND  FREQ_CHG,EQ       ;If 'F' goto routine to change
                                        ;frequency
                ADD   #0046h

                SUB   #0050h            ;Check to see if ASCII letter
                                        ;'P'
                BCND  PHASE_CHG,EQ      ;If 'P' goto routine to change
                                        ;frequency
                ADD   #0050h

                SUB   #004Dh            ;Check to see if ASCII letter
                                        ;'M'
                BCND  MAG_CHG,EQ        ;If 'M' goto routine to change
                                        ;frequency
                ADD   #004Dh

                SUB   #000Dh            ;Check to see if ASCII <CR>
                BCND  SCI_ISR,EQ        ;If <CR>, output "Ready"
                B  BAD_IP
                B  ISR_END              ;If neither a
                                        ;'F','P','M',or<CR>,
                                        ;then do nothing

BAD_IP          LAR   AR2,#(B1_SADDR+LENGTH1) ;Address to output
                                              ;"Invalid Input"
SCI_ISR         MAR   *,AR2             ;ARP = AR2
                LDP   #00E0h            ;DP = 224 Addresses 7000h-
707Fh
XMIT_CHAR:  LACC  *+,AR0                ;Load char to be xmitted into
                                        ;ACC
            BCND  ISR_END,EQ            ;Check for Null Character
                                        ;YES? Return from INT1_ISR.
            SACL  *,AR2                 ;NO? Load char into xmit
                                        ;buffer.

XMIT_RDY:   BIT   SCICTL2, BIT7         ;Test TXRDY bit
            BCND  XMIT_RDY, NTC         ;If TXRDY=0,then repeat loop
            B     XMIT_CHAR             ;else xmit next character
```

```
ISR_END:    LAR   AR2, #B1_SADDR       ;Reload AR2 w/ TX data start
                                       ;address
                  LAR   AR3, #B0_SADDR    ;Reload AR3 w/ RX data start
                                       ;address
                  LDP   #00E0h            ;DP = 224 Addresses 7000h-
707Fh
                  LACC  #0Ah              ;Cause a line feed in the
                                       ;terminal
                  SACL  SCITXBUF          ;transmit the line feed
                  LDP   #0                ;DP = 0 Addresses 0000h-007Fh
                  CLRC  INTM              ;Enable Interrrupts
            RET                           ;Return from INT1_ISR

NO_IP       LAR       AR2, #B1_SADDR    ;Reload AR2 w/ TX data start
                                       ;address
                  LDP   #0                ;DP = 0 Addresses 0000h-007Fh
                  CLRC  INTM              ;Enable Interrupts
                  RET                     ;Return from interrupt

BAD_INT:    LACC      #0BADh            ;Load ACC with "bad"
            B         BAD_INT           ;Repeat loop

;---------------------------------------------------------------------
;The following section will modify the frequency of the specified
;channel
;---------------------------------------------------------------------

FREQ_CHG          LDP   #0                ;DP = 0 Addresses 0000h -
007Fh
                  SPLK  #0000h,NEW_VALUE  ;Initialize NEW_VALUE
                  LACC  *+
                  SUB   #0041h            ;Check whether to modify
                                       ;Channel A
                  BCND  FREQ_A,EQ         ;If 'A', goto part that
                                       ;modifies Chan. A

                  ADD   #0041h
                  SUB   #0042h            ;Check whether to modify
                                       ;Channel B
                  BCND  FREQ_B,EQ         ;If 'B', goto part that
                                       ;modifies Chan. B

                  ADD   #0042h
                  SUB   #0052h            ;Check whether to reset the
                                       ;Channels
                  BCND  FREQ_RESET,EQ
                  B  BAD_IP               ;Else, bad input so end routine

FREQ_B            LACC  FREQSTEP2         ;ACC = FREQSTEP2
                  SACL  PREV_VALUE        ;Keep the previous value in
                                       ;case
                                       ;the new value is invalid

                  CALL  WHAT_VALUE        ;Determine the entered
```

```
                                    ;frequency

                    LACC  NEW_VALUE         ;ACC = entered value
                    SUB   PREV_VALUE        ;If entered value is the same
                                    ;as the prev
                    BCND  NO_CHGB,EQ        ;then, the entered value was
                                    ;invalid, so
                                    ;nothing changes, else
                    CALL  FSTEP_VALUE       ;Determine the equivalent step
                                    ;value

NO_CHGB             LACC  NEW_VALUE         ;ACC = New step value
                    SACL  FREQSTEP2         ;FREQSTEP2 = New step value

                    B  ISR_END              ;End of the frequency change
                                    ;for Chan. B

FREQ_A              LACC  FREQSTEP1         ;ACC = FREQSTEP1
                    SACL  PREV_VALUE        ;Keep the previous value in
                                    ;case
                                    ;the new value is invalid

                    CALL  WHAT_VALUE        ;Determine the entered
                                    ;frequency

                    LACC  NEW_VALUE         ;ACC = entered value
                    SUB   PREV_VALUE        ;If entered value is the same
                                    ;as the prev
                    BCND  NO_CHGA,EQ        ;then, the entered value was
                                    ;invalid, so
                                    ;nothing changes, else
                    CALL  FSTEP_VALUE       ;Determine the equivalent step
                                    ;value

NO_CHGA             LACC  NEW_VALUE         ;ACC = New step value
                    SACL  FREQSTEP1         ;FREQSTEP1 = New Step Value

                    B  ISR_END              ;End of the frequency change
                                    ;for Chan. A

FREQ_RESET          SPLK  #1000,FREQSTEP1   ;Initialize Channel A to
                                    ;original value
                    SPLK  #1000,FREQSTEP2   ;Initialize Channel B to
                                    ;original value

                    B  ISR_END

;=====================================================================
; Converts the entered frequency into a step value
;=====================================================================

FSTEP_VALUE         LT NEW_VALUE            ;TREG = NEW_VALUE
                     MPY  #QT1PERIOD        ;Multiply by the "sampling
                                    ;period"
                     PAC                    ;ACC = PREG
```

```
                SACL NEW_VALUE,1        ;Store the new step value
                RET                     ;Return from routine


;---------------------------------------------------------------------
;The following section will modify the phase of the specified ;channel
;---------------------------------------------------------------------
PHASE_CHG       LDP   #0                ;DP = 0 Addresses 0000h -
007Fh
                SPLK  #360,DIVISOR      ;Initialize the max value to
                                       ;360 deg
                SPLK  #0000h,QUOTIENT   ;Initialize the quotient value
                SPLK  #0000h,NEW_VALUE  ;Initialize NEW_VALUE

                LACC  *+
                SUB   #0041h            ;Check whether to modify
                                       ;Channel A
                BCND  PHASE_A,EQ

                ADD   #0041h
                SUB   #0042h            ;Check whether to modify
                                       ;Channel B
                BCND  PHASE_B,EQ

                ADD   #0042h
                SUB   #0052h            ;Check whether to reset the
                                       ;Channels
                BCND  PHASE_RESET,EQ
                B  BAD_IP               ;Else, bad input so end routine

PHASE_B         LACC  MODREG2           ;Load the Modulo Register of
                                       ;Channel B
                SACL  PREV_VALUE        ;Save the value in case the
                                       ;entered
                                       ;value is invalid
                CALL  WHAT_VALUE        ;Determine the entered phase
                LACC  NEW_VALUE         ;Load the value

                SUB   PREV_VALUE        ;If input value is invalid,
                                       ;WHAT_VALUE
                BCND  ISR_END,EQ        ;sets NEXT_VALUE=PREV_VALUE
                ADD   PREV_VALUE

                SUB   #360              ;If input is a number, then
                BCND  BAD_IP,GT         ;Check if value is larger than
                                       ;360 degrees

                ADD   #360
                CALL  DIVIDE            ;If the value is okay, then
                                       ;determine
                                       ;what the Q15 fraction the
                                       ;value is of 360
                LT QUOTIENT             ;TREG = QUOTIENT
                MPY   #100h             ;PREG = QUOTIENT * 256
                PAC                     ;ACC = PREG
```

```
                SACH  NEW_VALUE,1       ;NEW_VALUE=QUOTIENT *256;
Shift
                                        ;by 1 to
                                        ;remove extra sign bit
                LACC  NEW_VALUE,8       ;ACC = NEW_VALUE * 256
                ADD   MODREG1           ;Add the Modulo Register of
                                        ;Channel A
                SACL  MODREG2           ;Store the new value for
                                        ;Channel B

                B  ISR_END              ;End of phase shift for Channel
                                        ;B

PHASE_A         LACC  MODREG1           ;Load the Modulo Register of
                                        ;Channel A
                SACL  PREV_VALUE        ;Save the value in case the
                                        ;entered value
                                      ;is invalid
                CALL  WHAT_VALUE        ;Determine the entered phase
                LACC  NEW_VALUE         ;Load the value

                SUB   PREV_VALUE        ;If input value is invalid,
                                      ;WHAT_VALUE
                BCND  ISR_END,EQ        ;sets NEXT_VALUE=PREV_VALUE
                ADD   PREV_VALUE

                SUB   #360              ;If input is a number, then
                BCND  BAD_IP,GT         ;Check if value is larger than
                                        ;360 degrees
                ADD   #360
                CALL  DIVIDE            ;If the value is okay, then
                                        ;determine
                                        ;what the Q15 fraction the
                                        ;value is of 360
                LT QUOTIENT             ;TREG = QUOTIENT
                MPY   #100h             ;PREG = QUOTIENT * 256
                PAC                     ;ACC = PREG
                SACH  NEW_VALUE,1       ;NEW_VALUE = QUOTIENT * 256;
                                        ;Shift by 1 to
                                        ;remove extra sign bit
                LACC  NEW_VALUE,8       ;ACC = NEW_VALUE * 256
                ADD   MODREG2           ;Add the Modulo Register of
                                        ;Channel B
                SACL  MODREG1           ;Store the new value for
                                        ;Channel A

                B  ISR_END              ;End of phase shift for Channel
                                        ;A

PHASE_RESET     SPLK  #0000h,MODREG1    ;Initialize Channel A to
                                        ;original value
                SPLK  #0000h,MODREG2    ;Initialize Channel B to
                                        ;original value

                B  ISR_END
```

```
;-----------------------------------------------------------------
;The following section will modify the magnitude of the specified
;channel
;-----------------------------------------------------------------

MAXMAG          .equ  50

MAG_CHG         LDP     #0              ;DP = 0 Addresses
                                      ;0000h - 007Fh
                SPLK  #MAXMAG,DIVISOR      ;Initialize DIVISOR
                SPLK  #7FFFh,TEMP          ;Initialize TEMP
                SPLK  #0000h,QUOTIENT      ;Initialize QUOTIENT
                SPLK  #0000h,NEW_VALUE     ;Initialize NEW_VALUE

                LACC  *+
                SUB   #0041h
                BCND  MAG_A,EQ             ;Check whether to modify
                                      ;Channel A

                ADD   #0041h
                SUB   #0042h
                BCND  MAG_B,EQ             ;Check whether to modify
                                      ;Channel B
                ADD   #0042h
                SUB   #0052h               ;Check whether to reset the
                                      ;Channels
                BCND  MAG_RESET,EQ
                B  BAD_IP             ;Else, bad input so end
                                      ;routine

MAG_B           LACC  MAG2                 ;ACC = MAG2
                SACL  PREV_VALUE           ;Store the current
magnitude
                                      ;in case
                                      ;the input value is invalid
                CALL  WHAT_VALUE           ;Determine the entered
                                      ;magnitude
                LACC  NEW_VALUE            ;ACC = Entered Value

                SUB   PREV_VALUE           ;If input value is invalid,
                                      ;WHAT_VALUE
                BCND  ISR_END,EQ           ;sets NEXT_VALUE=PREV_VALUE
                ADD   PREV_VALUE

                SUB   #MAXMAG              ;If input is a number, then
                BCND  BAD_IP,GT            ;Check if value is larger
                                      ;than MAXMAG
                ADD   #MAXMAG
                CALL  DIVIDE               ;If the value is okay,
                                      ;determine what
                                      ;proportion the entered
                                      ;value is of MAXMAG
                CALL  MAG_VALUE            ;Normalize the ratio to the
```

```
                                        ;maximum
                                        ;Q15 value
                LACC  NEW_VALUE
                AND   #7FFFh              ;Make sure the value is
                                        ;positive
                SACL  MAG2                ;Store the new magnitude
for
                                        ;Channel B

                B  ISR_END              ;End of modifying magnitude
                                        ;for Channel B

MAG_A           LACC  MAG1                ;ACC = MAG1
                SACL  PREV_VALUE          ;Store the current
magnitude
                                        ;in case
                                        ;the input value is invalid
                CALL  WHAT_VALUE          ;determine the entered
                                        ;magnitude
                LACC  NEW_VALUE           ;ACC = Entered value

                SUB   PREV_VALUE          ;If input value is invalid,
                                        ;WHAT_VALUE
                BCND  ISR_END,EQ          ;sets NEXT_VALUE=PREV_VALUE
                ADD   PREV_VALUE

                SUB   #MAXMAG             ;If input is a number, then
                BCND  BAD_IP,GT           ;Check if value is larger
                                        ;than MAXMAG
                ADD   #MAXMAG
                CALL  DIVIDE              ;if the value is okay,
                                        ;determine what
                                        ;proportion the entered
                                        ;value is of MAXMAG
                CALL  MAG_VALUE           ;Normalize the ratio to the
                                        ;maximum
                                        ;Q15 value
                LACC  NEW_VALUE
                AND   #7FFFh              ;Make sure the value is
                                        ;positive
                SACL  MAG1                ;Store the new magnitude
for
                                        ;Channel A

                B  ISR_END              ;End of modifying magnitude
                                        ;for Channel A

MAG_RESET       SPLK  #7FFFh,MAG1         ;Initialize Channel A to
                                        ;original value
                SPLK  #7FFFh,MAG2         ;Initialize Channel B to
                                        ;original value

                B  ISR_END

;=================================================================
```

```
; Converts the entered magnitude into a Q15 value
;=================================================================

MAG_VALUE         LACC  QUOTIENT
                  SUB   #08000h          ;If the DIVISOR=DIVIDENT, then
                  BCND  MAX_MAG,EQ       ;set quotient to maximum Q15
                                        ;value
                  LT TEMP               ;TREG = TEMP
                  MPY   QUOTIENT           ;PREG = TEMP * QUOTIENT
                  PAC                      ;ACC = PREG
                  SACH  NEW_VALUE,1        ;NEW_VALUE = TEMP * QUOTIENT;
                                        ;Shift by 1 to
                                        ;remove extra sign bit
                  RET                      ;Return from routine

MAX_MAG           SPLK  #7FFFh,NEW_VALUE  ;Set NEW_VALUE to maximum Q15
                                        ;value
                  RET                      ;Return from Routine

;=================================================================
;Routine to determine the next value
;=================================================================

WHAT_VALUE        LACC  *+                 ;ACC = First place

                  SUB   #000Dh             ;Check if the value is a <CR>
                  BCND  NO_VALUE,EQ

                  ADD   #000Dh
                  SUB   #0030h             ;Check if value is below ASCII
                                           ;0
                  BCND  NO_VALUE,LT

                  ADD   #0030h
                  SUB   #0039h             ;Check if value is above ASCII
                                        ;9
                  BCND  NO_VALUE,GT

                  ADD   #0039h

VALID_VALUE       SUB   #0030h             ;Make the value into a usable
                                        ;value
                  LT NEW_VALUE          ;TREG = Digit of entered value
                  MPY   #10                ;PREG = TREG * 10
                  APAC                     ;ACC = PREG + ACC
                  SACL  NEW_VALUE          ;Store the new value

                  LACC  *+                 ;Load the next digit
                  SUB   #000Dh
                  BCND  EOV,EQ             ;If it is <CR>, then End of
                                        ;Value
                  ADD   #000Dh
                  SUB   #0030h             ;Check is value is below ASCII
                                        ;0
                  BCND  NO_VALUE,LT
```

```
                ADD   #0030h
                SUB   #0039h            ;Check if value is above ASCII
                                       ;9
                BCND  NO_VALUE,GT
                ADD   #0039h
                B  VALID_VALUE

NO_VALUE        LACC  PREV_VALUE        ;If the entered value is bad,
                                       ;then
                SACL  NEW_VALUE         ;store the previous value as
                                       ;the new value
                CALL  BAD_IP
                RET                     ;Return from routine

EOV             SPLK  #0FFFFh,PREV_VALUE ;Valid Value, Discard
previous value
                RET                        ;Return from routine

;=====================================================================
; Divide Routine
;=====================================================================
DIVIDE          LAR   AR4,#DIVISOR         ;AR4 = Address for DIVISOR
                                       ;value
                LAR   AR5,#QUOTIENT        ;AR5 = Address for Q15
                                       ;quotient
                LAR   AR6,#15              ;AR6 = 16 - 1; Number of
                                       ;times to subtract
                LAR   AR7,#NEW_VALUE

                MAR   *,AR5                ;ARP = AR5
DIVIDING        LACC  *                    ;ACC = QUOTIENT
                ACL   *,1,AR7              ;QUOTIENT = QUOTIENT * 2;
                                       ;1st shift doesn't
                                       ;matter because QUOTIENT = 0
                LACC  *,0,AR4              ;ACC = Value pointed by AR7
                SUB   *                    ;Subtract DIVISOR
                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

ADD             *,AR7
                SACL  *,1,AR6              ;Store the remainder * 2 to
                                       ;location
                                       ;pointed by AR7
                BANZ  DIVIDING,*-,AR5      ;AR6 = AR6 - 1; Repeat the
                                       ;dividing
                RET

ADD_ONE         MAR   *,AR7                ;ARP = AR7
```

```
                  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
                  RET

;---------------------------------------------------------------------
; Sine look-up table
; No. Entries     : 256
; Angle Range     : 360 deg
; Number format : Q15 with range -1 < N < +1
;---------------------------------------------------------------------
                   ;SINVAL        ;        Index      Angle Sin(Angle)
STABLE             .word          0        ;          0           0
0.0000
        .word        804          ;         1        1.41     0.0245
        .word       1608          ;         2        2.81     0.0491
        .word       2410          ;         3        4.22     0.0736
        .word       3212          ;         4        5.63     0.0980
        .word       4011          ;         5        7.03     0.1224
        .word       4808          ;         6        8.44     0.1467
        .word       5602          ;         7        9.84     0.1710
        .word       6393          ;         8       11.25     0.1951
        .word       7179          ;         9       12.66     0.2191
        .word       7962          ;        10       14.06     0.2430
        .word       8739          ;        11       15.47     0.2667
        .word       9512          ;        12       16.88     0.2903
        .word       10278         ;        13       18.28     0.3137
        .word       11039         ;        14       19.69     0.3369
        .word       11793         ;        15       21.09     0.3599
        .word       12539         ;        16       22.50     0.3827
        .word       13279         ;        17       23.91     0.4052
        .word       14010         ;        18       25.31     0.4276
        .word       14732         ;        19       26.72     0.4496
        .word       15446         ;        20       28.13     0.4714
        .word       16151         ;        21       29.53     0.4929
        .word       16846         ;        22       30.94     0.5141
        .word       17530         ;        23       32.34     0.5350
        .word       18204         ;        24       33.75     0.5556
        .word       18868         ;        25       35.16     0.5758
        .word       19519         ;        26       36.56     0.5957
        .word       20159         ;        27       37.97     0.6152
        .word       20787         ;        28       39.38     0.6344
        .word       21403         ;        29       40.78     0.6532
        .word       22005         ;        30       42.19     0.6716
        .word       22594         ;        31       43.59     0.6895
        .word       23170         ;        32       45.00     0.7071
        .word       23731         ;        33       46.41     0.7242
        .word       24279         ;        34       47.81     0.7410
```

```
        .word   24811   ;       35      49.22   0.7572
        .word   25329   ;       36      50.63   0.7730
        .word   25832   ;       37      52.03   0.7883
        .word   26319   ;       38      53.44   0.8032
        .word   26790   ;       39      54.84   0.8176
        .word   27245   ;       40      56.25   0.8315
        .word   27683   ;       41      57.66   0.8449
        .word   28105   ;       42      59.06   0.8577
        .word   28510   ;       43      60.47   0.8701
        .word   28898   ;       44      61.88   0.8819
        .word   29268   ;       45      63.28   0.8932
        .word   29621   ;       46      64.69   0.9040
        .word   29956   ;       47      66.09   0.9142
        .word   30273   ;       48      67.50   0.9239
        .word   30571   ;       49      68.91   0.9330
        .word   30852   ;       50      70.31   0.9415
        .word   31113   ;       51      71.72   0.9495
        .word   31356   ;       52      73.13   0.9569
        .word   31580   ;       53      74.53   0.9638
        .word   31785   ;       54      75.94   0.9700
        .word   31971   ;       55      77.34   0.9757
        .word   32137   ;       56      78.75   0.9808
        .word   32285   ;       57      80.16   0.9853
        .word   32412   ;       58      81.56   0.9892
        .word   32521   ;       59      82.97   0.9925
        .word   32609   ;       60      84.38   0.9952
        .word   32678   ;       61      85.78   0.9973
        .word   32728   ;       62      87.19   0.9988
        .word   32757   ;       63      88.59   0.9997
        .word   32767   ;       64      90.00   1.0000
        .word   32757   ;       65      91.41   0.9997
        .word   32728   ;       66      92.81   0.9988
        .word   32678   ;       67      94.22   0.9973
        .word   32609   ;       68      95.63   0.9952
        .word   32521   ;       69      97.03   0.9925
        .word   32412   ;       70      98.44   0.9892
        .word   32285   ;       71      99.84   0.9853
        .word   32137   ;       72      101.25  0.9808
        .word   31971   ;       73      102.66  0.9757
        .word   31785   ;       74      104.06  0.9700
        .word   31580   ;       75      105.47  0.9638
        .word   31356   ;       76      106.88  0.9569
        .word   31113   ;       77      108.28  0.9495
        .word   30852   ;       78      109.69  0.9415
        .word   30571   ;       79      111.09  0.9330
        .word   30273   ;       80      112.50  0.9239
        .word   29956   ;       81      113.91  0.9142
        .word   29621   ;       82      115.31  0.9040
        .word   29268   ;       83      116.72  0.8932
        .word   28898   ;       84      118.13  0.8819
        .word   28510   ;       85      119.53  0.8701
        .word   28105   ;       86      120.94  0.8577
        .word   27683   ;       87      122.34  0.8449
        .word   27245   ;       88      123.75  0.8315
        .word   26790   ;       89      125.16  0.8176
```

```
.word       26319       ;       90      126.56    0.8032
.word       25832       ;       91      127.97    0.7883
.word       25329       ;       92      129.38    0.7730
.word       24811       ;       93      130.78    0.7572
.word       24279       ;       94      132.19    0.7410
.word       23731       ;       95      133.59    0.7242
.word       23170       ;       96      135.00    0.7071
.word       22594       ;       97      136.41    0.6895
.word       22005       ;       98      137.81    0.6716
.word       21403       ;       99      139.22    0.6532
.word       20787       ;       100     140.63    0.6344
.word       20159       ;       101     142.03    0.6152
.word       19519       ;       102     143.44    0.5957
.word       18868       ;       103     144.84    0.5758
.word       18204       ;       104     146.25    0.5556
.word       17530       ;       105     147.66    0.5350
.word       16846       ;       106     149.06    0.5141
.word       16151       ;       107     150.47    0.4929
.word       15446       ;       108     151.88    0.4714
.word       14732       ;       109     153.28    0.4496
.word       14010       ;       110     154.69    0.4276
.word       13279       ;       111     156.09    0.4052
.word       12539       ;       112     157.50    0.3827
.word       11793       ;       113     158.91    0.3599
.word       11039       ;       114     160.31    0.3369
.word       10278       ;       115     161.72    0.3137
.word       9512        ;       116     163.13    0.2903
.word       8739        ;       117     164.53    0.2667
.word       7962        ;       118     165.94    0.2430
.word       7179        ;       119     167.34    0.2191
.word       6393        ;       120     168.75    0.1951
.word       5602        ;       121     170.16    0.1710
.word       4808        ;       122     171.56    0.1467
.word       4011        ;       123     172.97    0.1224
.word       3212        ;       124     174.38    0.0980
.word       2410        ;       125     175.78    0.0736
.word       1608        ;       126     177.19    0.0491
.word        804        ;       127     178.59    0.0245
.word          0        ;       128     180.00    0.0000
.word      64731        ;       129     181.41   -0.0245
.word      63927        ;       130     182.81   -0.0491
.word      63125        ;       131     184.22   -0.0736
.word      62323        ;       132     185.63   -0.0980
.word      61524        ;       133     187.03   -0.1224
.word      60727        ;       134     188.44   -0.1467
.word      59933        ;       135     189.84   -0.1710
.word      59142        ;       136     191.25   -0.1951
.word      58356        ;       137     192.66   -0.2191
.word      57573        ;       138     194.06   -0.2430
.word      56796        ;       139     195.47   -0.2667
.word      56023        ;       140     196.88   -0.2903
.word      55257        ;       141     198.28   -0.3137
.word      54496        ;       142     199.69   -0.3369
.word      53742        ;       143     201.09   -0.3599
.word      52996        ;       144     202.50   -0.3827
```

```
        .word   52256   ;       145     203.91  -0.4052
        .word   51525   ;       146     205.31  -0.4276
        .word   50803   ;       147     206.72  -0.4496
        .word   50089   ;       148     208.13  -0.4714
        .word   49384   ;       149     209.53  -0.4929
        .word   48689   ;       150     210.94  -0.5141
        .word   48005   ;       151     212.34  -0.5350
        .word   47331   ;       152     213.75  -0.5556
        .word   46667   ;       153     215.16  -0.5758
        .word   46016   ;       154     216.56  -0.5957
        .word   45376   ;       155     217.97  -0.6152
        .word   44748   ;       156     219.38  -0.6344
        .word   44132   ;       157     220.78  -0.6532
        .word   43530   ;       158     222.19  -0.6716
        .word   42941   ;       159     223.59  -0.6895
        .word   42365   ;       160     225.00  -0.7071
        .word   41804   ;       161     226.41  -0.7242
        .word   41256   ;       162     227.81  -0.7410
        .word   40724   ;       163     229.22  -0.7572
        .word   40206   ;       164     230.63  -0.7730
        .word   39703   ;       165     232.03  -0.7883
        .word   39216   ;       166     233.44  -0.8032
        .word   38745   ;       167     234.84  -0.8176
        .word   38290   ;       168     236.25  -0.8315
        .word   37852   ;       169     237.66  -0.8449
        .word   37430   ;       170     239.06  -0.8577
        .word   37025   ;       171     240.47  -0.8701
        .word   36637   ;       172     241.88  -0.8819
        .word   36267   ;       173     243.28  -0.8932
        .word   35914   ;       174     244.69  -0.9040
        .word   35579   ;       175     246.09  -0.9142
        .word   35262   ;       176     247.50  -0.9239
        .word   34964   ;       177     248.91  -0.9330
        .word   34683   ;       178     250.31  -0.9415
        .word   34422   ;       179     251.72  -0.9495
        .word   34179   ;       180     253.13  -0.9569
        .word   33955   ;       181     254.53  -0.9638
        .word   33750   ;       182     255.94  -0.9700
        .word   33564   ;       183     257.34  -0.9757
        .word   33398   ;       184     258.75  -0.9808
        .word   33250   ;       185     260.16  -0.9853
        .word   33123   ;       186     261.56  -0.9892
        .word   33014   ;       187     262.97  -0.9925
        .word   32926   ;       188     264.38  -0.9952
        .word   32857   ;       189     265.78  -0.9973
        .word   32807   ;       190     267.19  -0.9988
        .word   32778   ;       191     268.59  -0.9997
        .word   32768   ;       192     270.00  -1.0000
        .word   32778   ;       193     271.41  -0.9997
        .word   32807   ;       194     272.81  -0.9988
        .word   32857   ;       195     274.22  -0.9973
        .word   32926   ;       196     275.63  -0.9952
        .word   33014   ;       197     277.03  -0.9925
        .word   33123   ;       198     278.44  -0.9892
        .word   33250   ;       199     279.84  -0.9853
```

```
.word   33398   ;   200   281.25   -0.9808
.word   33564   ;   201   282.66   -0.9757
.word   33750   ;   202   284.06   -0.9700
.word   33955   ;   203   285.47   -0.9638
.word   34179   ;   204   286.88   -0.9569
.word   34422   ;   205   288.28   -0.9495
.word   34683   ;   206   289.69   -0.9415
.word   34964   ;   207   291.09   -0.9330
.word   35262   ;   208   292.50   -0.9239
.word   35579   ;   209   293.91   -0.9142
.word   35914   ;   210   295.31   -0.9040
.word   36267   ;   211   296.72   -0.8932
.word   36637   ;   212   298.13   -0.8819
.word   37025   ;   213   299.53   -0.8701
.word   37430   ;   214   300.94   -0.8577
.word   37852   ;   215   302.34   -0.8449
.word   38290   ;   216   303.75   -0.8315
.word   38745   ;   217   305.16   -0.8176
.word   39216   ;   218   306.56   -0.8032
.word   39703   ;   219   307.97   -0.7883
.word   40206   ;   220   309.38   -0.7730
.word   40724   ;   221   310.78   -0.7572
.word   41256   ;   222   312.19   -0.7410
.word   41804   ;   223   313.59   -0.7242
.word   42365   ;   224   315.00   -0.7071
.word   42941   ;   225   316.41   -0.6895
.word   43530   ;   226   317.81   -0.6716
.word   44132   ;   227   319.22   -0.6532
.word   44748   ;   228   320.63   -0.6344
.word   45376   ;   229   322.03   -0.6152
.word   46016   ;   230   323.44   -0.5957
.word   46667   ;   231   324.84   -0.5758
.word   47331   ;   232   326.25   -0.5556
.word   48005   ;   233   327.66   -0.5350
.word   48689   ;   234   329.06   -0.5141
.word   49384   ;   235   330.47   -0.4929
.word   50089   ;   236   331.88   -0.4714
.word   50803   ;   237   333.28   -0.4496
.word   51525   ;   238   334.69   -0.4276
.word   52256   ;   239   336.09   -0.4052
.word   52996   ;   240   337.50   -0.3827
.word   53742   ;   241   338.91   -0.3599
.word   54496   ;   242   340.31   -0.3369
.word   55257   ;   243   341.72   -0.3137
.word   56023   ;   244   343.13   -0.2903
.word   56796   ;   245   344.53   -0.2667
.word   57573   ;   246   345.94   -0.2430
.word   58356   ;   247   347.34   -0.2191
.word   59142   ;   248   348.75   -0.1951
.word   59933   ;   249   350.16   -0.1710
.word   60727   ;   250   351.56   -0.1467
.word   61524   ;   251   352.97   -0.1224
.word   62323   ;   252   354.38   -0.0980
.word   63125   ;   253   355.78   -0.0736
.word   63927   ;   254   357.19   -0.0491
```

```
        .word       64731       ;         255      358.59   -0.0245
        .word       65535       ;         256      360.00    0.0000

;=================================================================
; I S R  -  PHANTOM
;
; Description:    Dummy ISR, used to trap spurious interrupts.
;=================================================================
PHANTOM:    KICK_DOG
                B    PHANTOM
```