;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; Author : ADI - Apps www.analog.com/MicroConverter

;

; Date : 28 May 1999

;

; File : DACquadr.asm

;

; Hardware : ADuC812

;

; Description : Outputs sine waves on DAC0 and DAC1 at 400Hz.

; Output signals are in quadrature with eachother,

; DAC1 leading DAC0 by 90 degrees. since each DAC is

; updated when its DACxL register is written to, they

; are not updated at the exact same moment, and a

; phase error of (in this case) 0.625degrees results.

; to address this problem, see code: "DACsync.asm".

; Rate calculations assume an 11.0592MHz Mclk.

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$MOD812 ; Use 8052&ADuC812 predefined symbols

LED EQU P3.4 ; P3.4 drives red LED on eval board

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; BEGINNING OF CODE

CSEG

ORG 0000h

MOV DACCON,#01Fh ; both DACs on,12bit,asynchronous

MOV DAC0H,#008h

MOV DAC0L,#000h ; DAC0 to mid-scale

MOV DAC1H,#00Fh

MOV DAC1L,#0FFh ; DAC1 to full-scale

MOV DPTR,#TABLE

STEP: CLR A ; 1

MOVC A,@A+DPTR ; get high byte for mainDAC.. 2

MOV DAC0H,A ; ..and move it into DAC0 register 1

MOV A,#020h ; offset by 90deg for quadratureDAC 1

MOVC A,@A+DPTR ; get high byte for quadratureDAC.. 2

MOV DAC1H,A ; ..and move it into DAC1 register 1

INC DPTR ; move on to get low bytes 2

CLR A ; 1

MOVC A,@A+DPTR ; get low byte for mainDAC.. 2

MOV DAC0L,A ; ..and update DAC0 1

MOV A,#020h ; offset by 90deg for quadratureDAC 1

MOVC A,@A+DPTR ; get low byte for quadratureDAC.. 2

MOV DAC1L,A ; ..and update DAC1 1

INC DPTR ; move on for next data point 2

ANL DPL,#07Fh ; wrap around at end of table 2

MOV A,DAC0H ; 1

MOV C,ACC.3 ; MSB of DAC0 value 1

MOV LED,C ; LED = MSB of DAC0 2

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

JMP STEP ; 2

; numbers at right in the above loop represent the number of machine

; cycles for each instruction. the complete loop takes exactly 36

; machine cycles. with an 11.0592MHz master clock, a machine cycle

; is 1.085us, so the above loop takes 39.06us to update each data

; point. since there are 64 data points in the below sine lookup

; table, this results in a 2.50ms period, i.e. a 400Hz frequency.

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SINE LOOKUP TABLE

ORG 01000h

TABLE:

DB 007h, 0FFh

DB 008h, 0C8h

DB 009h, 08Eh

DB 00Ah, 051h

DB 00Bh, 00Fh

DB 00Bh, 0C4h

DB 00Ch, 071h

DB 00Dh, 012h

DB 00Dh, 0A7h

DB 00Eh, 02Eh

DB 00Eh, 0A5h

DB 00Fh, 00Dh

DB 00Fh, 063h

DB 00Fh, 0A6h

DB 00Fh, 0D7h

DB 00Fh, 0F5h

DB 00Fh, 0FFh

DB 00Fh, 0F5h

DB 00Fh, 0D7h

DB 00Fh, 0A6h

DB 00Fh, 063h

DB 00Fh, 00Dh

DB 00Eh, 0A5h

DB 00Eh, 02Eh

DB 00Dh, 0A7h

DB 00Dh, 012h

DB 00Ch, 071h

DB 00Bh, 0C4h

DB 00Bh, 00Fh

DB 00Ah, 051h

DB 009h, 08Eh

DB 008h, 0C8h

DB 007h, 0FFh

DB 007h, 036h

DB 006h, 070h

DB 005h, 0ADh

DB 004h, 0EFh

DB 004h, 03Ah

DB 003h, 08Dh

DB 002h, 0ECh

DB 002h, 057h

DB 001h, 0D0h

DB 001h, 059h

DB 000h, 0F1h

DB 000h, 09Bh

DB 000h, 058h

DB 000h, 027h

DB 000h, 009h

DB 000h, 000h

DB 000h, 009h

DB 000h, 027h

DB 000h, 058h

DB 000h, 09Bh

DB 000h, 0F1h

DB 001h, 059h

DB 001h, 0D0h

DB 002h, 057h

DB 002h, 0ECh

DB 003h, 08Dh

DB 004h, 03Ah

DB 004h, 0EFh

DB 005h, 0ADh

DB 006h, 070h

DB 007h, 036h ; end of table

DB 007h, 0FFh ; repeat first 90degrees for quadratureDAC

DB 008h, 0C8h

DB 009h, 08Eh

DB 00Ah, 051h

DB 00Bh, 00Fh

DB 00Bh, 0C4h

DB 00Ch, 071h

DB 00Dh, 012h

DB 00Dh, 0A7h

DB 00Eh, 02Eh

DB 00Eh, 0A5h

DB 00Fh, 00Dh

DB 00Fh, 063h

DB 00Fh, 0A6h

DB 00Fh, 0D7h

DB 00Fh, 0F5h

DB 00Fh, 0FFh

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

END