

```

/*****
/*  Program shortevm.c
/*      This program provides a starting point for C541 EVM
/*  programs.  It is a simplified version of c541evm.c.  You can
/*  Use this basic file if you just want to set the AIC's A and
/*  B registers and accept the reset defaults for the other AIC
/*  registers.
/*
/*      Author: S.A. Tretter
/*      Date:   March 30, 1999
/*      Affiliation: Department of Electrical Engineering
/*                  University of Maryland, College Park
*****/

#include "c54xregs.h"

#include <math.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <stdio.h>

/*  Declare all function prototypes */
void init_aic(void);
void init_c541(void);
void inline disable_ints(void);
void inline enable_ints(void);

/*  The EVM uses address 14 (bit 15) of IO space for AIC reset */
volatile ioport unsigned port14;

void main(void)
{
    int received_sample;
    init_c541();
    disable_ints(); /* Turn off all maskable interrupts */
/*****
/*  Warning:  Initialize any long arrays before turning on the
/*  AIC.  The AIC can't wait for samples to be written into
/*  its DXR.  They must be ready when requested.  The initiali-
/*  zation can be done here.
*****/
    init_aic();
    IFR = 0xFFFF; /* Clear any pending interrupt requests */

```

```

    received_sample = DRR1; /* Dummy read to clear DRR1 */
    enable_ints(); /* Allow interrupts to be serviced */

/*****
/* Now the C541 and AIC are initialized to your desired state. */
/* Include the program to accomplish your desired task below. */
*****/

    for(;;); /* A dummy do nothing infinite loop */

} /* End of main() */

/*****
/* init_aic() initializes the TLC320AC01C analog interface */
/* circuit control register. */
/*-----*/
void init_aic(void) {

/*****
/* Formulas for the AIC sampling rate, fs, lowpass filter */
/* cutoff frequency, f(LP), highpass filter cutoff frequency, */
/* f(HP), and serial data shift clock rate, SCLK. */
/* */
/* f(MCLK) = 10.368 MHz = input master clock frequency from */
/* a crystal oscillator */
/* */
/* SCLK = serial shift clock= f(MCLK)/4 = 2.592 MHz */
/* */
/* FCLK = switched capacitor filter clock = f(MCLK)/(2xA) */
/* */
/* With no advance/retard, i.e., control bits = 00 */
/* fs = sampling frequency = f(MCLK)/ (2xAxB) */
/* With control bits = 01 */
/* fs = f(MCLK)/(2xAxB + A') */
/* With control bits = 10 */
/* fs = f(MCLK)/(2xAxB - A') */
/* */
/* f(LP) = lowpass filter cutoff = FCLK/40 = f(MCLK)/(80xA) */
/* */
/* f(HP) = highpass filter cutoff = fs/200 = f(MCLK)/(400xAxB) */
*****/

int A = 36; /* A = 36 -> FCLK = 144 kHz, f(LP) = 3.6 kHz */
int B = 18; /* B = 18 -> fs = 8 kHz, f(HP) = 40 Hz */

```

```

/* Put AIC in reset by setting bit 15 of IO port 14 to 0 */
port14 = 0x0;

/* Initialize Serial Port Control Register 1 (SPC1) as follows: */
/* F0 = 0,      16-bit mode                bit 2 */
/* FSM = 1,     Burst Mode                 bit 3 */
/* MCM = 0,     External Clock Source      bit 4 */
/* TXM = 0,     External Frame Sync        bit 5 */
/* ~XRST = 0,   Reset Transmitter and halt bit 6 */
/* ~RRST = 0,   Reset Receiver and halt    bit 7 */
/* SOFT = 1,    On halt, stop clock after  */
/*              current word sent          bit 14 */
/* all other bits are 0                    */
/* SPC1 = [0100 0000 0000 1000] = 0x4008  */

    SPC1 = 0x4008; /* Halt, reset, set mode as above */

/* Now set ~XRST = ~RRST = 1 to start serial port */
/* SPC1 = [0100 0000 1100 1000] = 0x40C8  */

    SPC1 = 0x40C8; /* Start serial port 1 */
    DXR1 = 0; /* Write a dummy word to serial port */

/* Pull AIC out of reset by setting bit 15 to 1 */
port14 = 0x8000;

/* Program the AIC registers */
while(!(SPC1 & 0x800)); /* Wait for XRDY bit = 1 */
DXR1 = 0x0003; /* Tell AIC to request a secondary command word */
while(!(SPC1 & 0x800)); /* Wait for XRDY bit = 1 */
DXR1 = 0x0100 | A; /* Set A Register */

while(!(SPC1 & 0x800)); /* Wait for XRDY = 1 */
DXR1 = 0x0003; /* Request a command word */
while(!(SPC1 & 0x800)); /* Wait for XRDY bit = 1 */
DXR1 = 0x0200 | B; /* Set B Register */
/*****
/* All the remaining AIC registers are left at their default */
/* reset values. */
*****/

    IMR = 0x40; /* Enable Serial Port 1 receiver interrupts by */
                /* setting bit 6 (RINT1) in the Interrupt Mask */

```

```

        /* Register (IMR) */

} /* end of init_aic() */

void init_c541(){
/* Set 0 wait states for external memory and 2 for IO */
/* Software Wait-State Register (SWWSR) Bit Summary */
/* Reset */
/* Bit Name Value Function */
/* 15 Reserved 0 */
/* 14-12 I/O 1 Wait states for I/O space 0000-FFFFh */
/* 11-9 Data 1 Wait states for data space 8000-FFFFh */
/* 8-6 Data 1 Wait states for data space 0000-7FFFh */
/* 5-3 Program 1 Wait states for program space 8000-FFFFh */
/* 2-0 Program 1 Wait states for program space 0000-7FFFh */

    SWWSR = 0x2000;

/* Set OVLY bit to overlay onchip RAM onto program space. */
/* Set bit 5 of Processor Mode Status Register (PMST) */
    PMST |= 0x0020;

/* Clear all flags in the Interrupt Flags Register (IFR) */
    IFR = 0xFFFF;

} /* End of init_c541() */

/* Function to disable all maskable interrupts */
void inline disable_ints()
{
    asm( " ssbx INTM" );
}

/* Function to enable all maskable interrupts */
void inline enable_ints()
{
    asm( " rsbx INTM" );
}

```