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

;

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

;

; Date : 7 March 2000

;

; File : SPIslave.asm

;

; Hardware : ADuC812

;

; Include File : UARTIO.asm - serial I/O routines

;

; Description : Demonstrates an example slave mode SPI interface.

; Code is intended for use with companion code file

; 'SPImstr.asm' running on a second MicroConverter

; chip. Chips must have SCLK, MOSI, MISO, & GND pins

; connected together, and P3.5 pin on master must

; connect to SS pin on slave.

;

; If using the ADuC812 eval board, you can simply

; connect the 10-pin J5 header (SPI/I2C) directly to

; that of the master board. However, on the slave

; board you must also remove R6 & C6 to disconnect

; an op amp output from the SS pin, and you must

; ensure that LK5 is INSERTED. To configure the

; master board, refer to 'SPImstr.asm'.

;

; Once hardware is connected, download code to both

; master & slave devices ('SPImstr' to the master,

; 'SPIslave' to the slave). Reset the slave first,

; and then the master. The slave will sit with the

; LED off until the master starts exchanging data

; with it at which time its LED will start blinking

; in sync (or 180°out of phase) with that of the

; master. When first launched, both master and slave

; are transmitting zeros repeatedly on the SPI port.

; Pressing the INT0 button on either master or slave

; increments the value it is transmitting. Received

; SPI data is relayed out the UART and can be viewed

; on any VT100 terminal or terminal emulator at

; 9600baud/8bits/noparity/1stopbit. Characters sent

; from the terminal to the MicroConverter will update

; the value being transmitted by SPI.

;

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

$MOD812 ; Use 8052 & ADuC812 predefined symbols

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

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

; DEFINE VARIABLES IN INTERNAL RAM

DSEG

ORG 0060h

INPUT: DS 1 ; data byte received by SPI

OUTPUT: DS 1 ; data byte to send by SPI

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

; BEGINNING OF CODE

CSEG

ORG 0000h

JMP MAIN ; jump to main program

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

; INTERRUPT VECTOR SPACE

ORG 0003h ; (.................... INT0 ISR)

INC OUTPUT

RETI

ORG 003Bh ; (.................... SPI ISR)

MOV INPUT,SPIDAT ; get data just received by SPI

MOV SPIDAT,OUTPUT ; update next byte to transmit

CLR C ; clear C indicates transfer complete

RETI

;====================================================================

; MAIN PROGRAM

ORG 004Bh

MAIN:

MOV SP,#007h

; CONFIGURE UART...

MOV SCON,#52h ; configure UART for 9600baud..

MOV TMOD,#20h ; ..assuming 11.0592MHz crystal

MOV TH1,#-3

SETB TR1

; CONFIGURE SPI...

MOV SPICON,#024h ; configure SPI port for:

; CPHA=1, CPOL=0, slave

;==> NOTE: it is important that CPHA and CPOL be the same for both

; the master and all slave devices. otherwise, data will

; be transitioning at the same time as it's being latched.

CLR P1.5 ; enable SS pin as input

MOV IE2,#1 ; enable SPI interrupt

; CONFIGURE INTERRUPT 0...

SETB IT0 ; INT0 edge triggered

SETB EX0 ; enable INT0 interrupt

; ENABLE INTERRUPTS & ENTER MAIN LOOP...

MOV OUTPUT,#0 ; set initial value for output byte..

MOV SPIDAT,#0 ; ..including very fisrt output byte

SETB EA ; enable inturrupts

LOOP: CPL LED ; flash the LED on the eval board

SETB C

JC $ ; wait here to receive SPI transfer

MOV A,INPUT ; send value received by SPI..

CALL SENDVAL ; ..out the UART as 2 hex chars

MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

CALL SENDSTRING ; ..out the UART

JNB RI,LOOP ; repeat (unless UART data received)

; WHEN UART DATA RECEIVED, MOVE DATA TO SPI OUTPUT...

MOV OUTPUT,SBUF ; update OUTPUT byte to new value

CLR RI ; must clear RI

JMP LOOP ; back to main loop

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

; SUBROUTINE INCLUDE FILE

$INCLUDE(UARTIO.asm)

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

; TEXT DATA TABLES

SEPERATOR: DB 10,13,0

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

END