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

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

; Date : April 2004

; File : SPIslave.asm

; Hardware : ADuC845

; 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 '845mstr.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.

; Once hardware is connected, download code to both master & slave devices

; ('845mstr' to the master, '845slave' 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 (when slave is transmitting data to

; a PC via the UART) or slave (when Master is transmitting data to a PC via the

; UART) increments the value it is transmitting. Received (Master to slave or

; Slave to Master) SPI data is relayed out the UART and can be viewed on any

; VT100 terminal or terminal emulator at 9600baud/8bits/noparity/1stopbit.

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

$MOD845 ; Use 8052 & ADuC845 predefined symbols

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

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

DSEG ; DEFINE VARIABLES IN INTERNAL RAM

ORG 0060h

INPUT: DS 1 ; data byte received by SPI

OUTPUT: DS 1 ; data byte to send by SPI

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

CSEG ; BEGINNING OF CODE

ORG 0000h

JMP MAIN ; jump to main program

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

ORG 0003h ; (.................... INT0 ISR) ; INTERRUPT VECTOR SPACE

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

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

ORG 004Bh ; MAIN PROGRAM

MAIN:

MOV PLLCON,#03H

MOV SP,#007h

MOV R7, #00h ;Initialise R1 to 00. R1 will be incremented automatically

;to try to coincide with the incoming data

; CONFIGURE UART...

MOV T3CON,#083h

Mov T3FD,#012h

MOV SCON,#052h

; CONFIGURE SPI...

MOV SPICON,#024h ; configure SPI port for: CPHA=1, CPOL=0, slave

MOV IEIP2,#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

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

$INCLUDE(UARTIO.asm) ; SUBROUTINE INCLUDE FILE

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

SEPERATOR: DB 10,13,0 ; TEXT DATA TABLES

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

DELAY: ; Delays by 100ms \* A

MOV R0,100 ; Acc holds delay variable

DLY0: MOV R1,#019h ; Set up delay loop0

DLY1: MOV R2,#0FEh ; Set up delay loop1

DJNZ R2,$ ; Dec R2 & Jump here until R2 is 0

DJNZ R1,DLY1 ; Dec R1 & Jump DLY1 until R1 is 0

DJNZ R0,DLY0 ; Dec R0 & Jump DLY0 until R0 is 0

RET ; Return from subroutine

END