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

;

; Author : ADI - Apps

;

; Date : January 2001

;

; File : MASTuart.asm

;

; Hardware ; ADuC816

;

; Description : This Program transmits the numbers 1-10 in binary

; form continuously down the SPI serial port.

; After the transmission of each byte the incoming

; byte is saved in order between internal RAM

; addresses 40h and 50h.

;

; After the 16 bytes have been writen into memory

; the program outputs the received data up the UART

; where it can be viewed using Hyperterminal.

;

; An SPI slave program can be run on a second device

; (ADuC816/ADuC824/ADuC812/other) to communicate

; with this master code.

;

; The Slave program (SLAVuart.asm in the SPI\SLAVE

; directory) should be started after the master

; program (MASTuart.asm) but within the time delay

; of 5s in order that the slave program is

; synchronised by the first outputted clock of the

; master.

;

; The clock is outputted at sclock (pin 26)

; The data is outputted at sdata/MOSI (pin 27)

; The data is inputted at MISO (pin 14)

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

;

$MOD816 ; Use 8052/ADuC816 predefined Symbols

LED EQU P3.4

FLAG BIT 00H

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

; BEGINNING OF CODE

CSEG

ORG 0000H

JMP MAIN

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

; SPI INTERRUPT ROUTINE

ORG 003BH

SETB P3.5 ; set the SS bit after transmission

CLR FLAG ; Clear flag to leave loop

MOV @R1, SPIDAT ; move input into memory

INC R1 ; increment memory location so new

; data is stored in new address

CJNE R1, #50H, CONT ; reset memory location to 40h when

; memory location reaches 50h saving

; 16 bytes of data

CALL SNDUART ; send the data up the UART

CONT: RETI

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

; MAIN PROGRAM

ORG 0060H ; Start code at address above interrupts

MAIN: ; Main program

MOV RCAP2H,#0FFh ; config UART for 9830baud

MOV RCAP2L,#-5 ; (close enough to 9600baud)

MOV TH2,#0FFh

MOV TL2,#-5

MOV SCON,#52h

MOV T2CON,#34h

MOV SPICON,#037h ; Initialise SPICON to have

; -bitrate=fosc/64

; -CPHA=1

; -CPOL=0, sclk idling low

; -master mode select

; -Enable SPI serial port

MOV IEIP2, #01h ; Enable SPI interrrupt

SETB EA ; Enable interrupts

MOV R1, #40h ; initialise R1 to 40 to store the

; input data from memory location 40

MOV R0, #00H ; initialise R0 to 0 to start

; transmissions from 1

; Delay the output of data by 5.0s in order that the slave program

; can be easily synchronised with the master program.

MOV A, #50

CALL DELAY

TRNSMT:

INC R0

CLR P3.5 ; clear the SS bit during transmission

MOV SPIDAT, R0 ; transmit the current value on R0

SETB FLAG ; set flag so that we wait here until

; the spi interrupt routine clears

; the FLAG

JB FLAG, $ ; stay here until flag is cleared

; by interrupt

; check if R0 is equal to 10. If so the number 10 has been

; transmitted and we should reset R0 to 0 to start transmission

; from 1 again

MOV A, R0

CJNE A, #0AH, TRNSMT ; if R0 is not 10, jump to TRNSMT

MOV R0, #00H ; if R0=10 make R0=0 & jump to TRNSMT

JMP TRNSMT

; Transmit the values in locations 40h->50h up the UART wait for

; 5 seconds and then transmit and receive values to/from the slave

; again down the SPI port.

SNDUART:

CPL LED ;CPL LED with each transmission

MOV DPTR, #TITLE

CALL SENDSTRING ; write title block on screen

MOV R1, #40h ; move value at address 40 into R2

MOV A, @R1

MOV R2, A

NEXT: ; Put new value on a new line

MOV A, #10 ; Transmit a linefeed (= ASCII 10)

CALL SENDCHAR

MOV A, #13 ;Transmit a carriage return (=ASCII 13)

CALL SENDCHAR

MOV A, R2 ; Transmit R2 i.e. value @ address R1

CALL SENDVAL

INC R1 ; Increment address

MOV A, @R1

MOV R2, A ; R2 holds the value @ addrR1

MOV A, R1 ; Check if at address 50h

CJNE A, #50h, NEXT ; if not jump to Next

JMP WAIT5S ; if so wait 5s and repeat

WAIT5S: MOV A, #50 ; wait 5s before sending down the

; SPI port again for ease of viewing

; on screen and to allow the slave

; synchronise itself with the master

CALL DELAY

MOV R1, #40h ; store new inputs at address 40h again

RETI

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

; SENDSTRING

SENDSTRING: ; sends ASCII string to UART starting at location

; DPTR and ending with a null (0) value

PUSH ACC

PUSH B

CLR A

MOV B,A

IO0010: MOV A,B

INC B

MOVC A,@A+DPTR

JZ IO0020

CALL SENDCHAR

JMP IO0010

IO0020: POP B

POP ACC

RET

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

; SENDCHAR

SENDCHAR: ; sends ASCII value contained in A to UART

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A

RET

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

; SENDVAL

SENDVAL: ; converts the hex value of A into two ASCII chars,

; and then spits these two characters up the UART.

; does not change the value of A.

PUSH ACC

SWAP A

CALL HEX2ASCII

CALL SENDCHAR ; send high nibble

POP ACC

PUSH ACC

CALL HEX2ASCII

CALL SENDCHAR ; send low nibble

POP ACC

RET

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

; HEX2ASCII

HEX2ASCII: ; converts A into the hex character representing the

; value of A's least significant nibble

ANL A,#00Fh

CJNE A,#00Ah,$+3

JC IO0030

ADD A,#007h

IO0030: ADD A,#'0'

RET

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

; DELAY

DELAY: ; Delays by 100ms \* A

; 100mSec based on 1.573MHZ Core Clock

MOV R2,A ; Acc holds delay variable

DLY0: MOV R3,#50 ; Set up delay loop0

DLY1: MOV R4,#131 ; Set up delay loop1

DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

; wait here for 131\*15.3us=2ms

DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

; Wait for 50\*2ms

DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

; wait for ACC\*100ms

RET ; Return from subroutine

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

TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

DB 'Analog Devices MicroConverter ADuC824',10,13

DB ' SPI MASTER Demo Routine',10,13

DB ' Data Stored in Memory in Hex Form',10,13,0

END