836SLAVE PAGE 1

1 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 ;

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

4 ;

5 ; Date : 7 March 2000

6 ;

7 ; File : SPIslave.asm

8 ;

9 ; Hardware : ADuC834/ADuC824/ADuC836/ADuC816

10 ;

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

12 ;

13 ; Description : Demonstrates an example slave mode SPI interface.

14 ; Code is intended for use with companion code file

15 ; '836mstr.asm' running on a second MicroConverter

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

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

18 ; connect to SS pin on slave.

19 ;

20 ; If using the ADuC836 eval board, you can

21 ; simply connect the 10-pin SPI/I2C header directly

22 ; to that of the master board. However, you must

23 ; also ensure that LK10 ('SS master') is REMOVED on

24 ; the slave board, and INSERTED on the master board.

25 ;

26 ; Once hardware is connected, download code to both

27 ; master & slave devices ('824mstr' to the master,

28 ; '836slave' to the slave). Reset the slave first,

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

30 ; LED off until the master starts exchanging data

31 ; with it at which time its LED will start blinking

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

33 ; master. When first launched, both master and slave

34 ; are transmitting zeros repeatedly on the SPI port.

35 ; Pressing the INT0 button on either master or slave

36 ; increments the value it is transmitting. Received

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

38 ; on any VT100 terminal or terminal emulator at

39 ; 9600baud/8bits/noparity/1stopbit. Characters sent

40 ; from the terminal to the MicroConverter will update

41 ; the value being transmitted by SPI.

42 ;

43 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

44

45 $MOD836 ; Use 8052 & ADuC836 predefined symbols

46

00B4 47 LED EQU P3.4 ; P3.4 drives red LED on eval board

48

49 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

50 ; DEFINE VARIABLES IN INTERNAL RAM

---- 51 DSEG

0060 52 ORG 0060h

0060 53 INPUT: DS 1 ; data byte received by SPI

0061 54 OUTPUT: DS 1 ; data byte to send by SPI

55

56 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

57 ; BEGINNING OF CODE

---- 58 CSEG

836SLAVE PAGE 2

59

0000 60 ORG 0000h

0000 02004B 61 JMP MAIN ; jump to main program

62

63 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

64 ; INTERRUPT VECTOR SPACE

0003 65 ORG 0003h ; (.................... INT0 ISR)

66

0003 0561 67 INC OUTPUT

0005 32 68 RETI

69

003B 70 ORG 003Bh ; (.................... SPI ISR)

71

003B 85F760 72 MOV INPUT,SPIDAT ; get data just received by SPI

003E 8561F7 73 MOV SPIDAT,OUTPUT ; update next byte to transmit

0041 C3 74 CLR C ; clear C indicates transfer complete

0042 32 75 RETI

76

77 ;====================================================================

78 ; MAIN PROGRAM

004B 79 ORG 004Bh

80

004B 81 MAIN:

82

004B 758107 83 MOV SP,#007h

84

85 ; CONFIGURE UART...

86

004E 759E82 87 MOV T3CON,#82h

0051 759D12 88 MOV T3FD,#12h

0054 759852 89 MOV SCON,#52h

90

91 ; CONFIGURE SPI...

92

0057 75F824 93 MOV SPICON,#024h ; configure SPI port for:

94 ; CPHA=1, CPOL=0, slave

005A 75A901 95 MOV IEIP2,#1 ; enable SPI interrupt

96

97 ; CONFIGURE INTERRUPT 0...

98

005D D288 99 SETB IT0 ; INT0 edge triggered

005F D2A8 100 SETB EX0 ; enable INT0 interrupt

101

102 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

103

0061 756100 104 MOV OUTPUT,#0 ; set initial value for output byte..

0064 75F700 105 MOV SPIDAT,#0 ; ..including very fisrt output byte

0067 D2AF 106 SETB EA ; enable inturrupts

107

0069 B2B4 108 LOOP: CPL LED ; flash the LED on the eval board

006B D3 109 SETB C

006C 40FE 110 JC $ ; wait here to receive SPI transfer

006E E560 111 MOV A,INPUT ; send value received by SPI..

0070 1200A3 112 CALL SENDVAL ; ..out the UART as 2 hex chars

0073 90011E 113 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

0076 120083 114 CALL SENDSTRING ; ..out the UART

0079 3098ED 115 JNB RI,LOOP ; repeat (unless UART data received)

116

836SLAVE PAGE 3

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

118

007C 859961 119 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

007F C298 120 CLR RI ; must clear RI

0081 80E6 121 JMP LOOP ; back to main loop

122

123 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

124 ; SUBROUTINE INCLUDE FILE

125

=1 126 $INCLUDE(UARTIO.asm)

=1 127 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 128 ;

=1 129 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 130 ;

=1 131 ; Date : January 2001

=1 132 ;

=1 133 ; File : UARTIO.asm

=1 134 ;

=1 135 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 136 ;

=1 137 ; Description : standard UART I/O subroutines. total size of this

=1 138 ; code when assembled is 155 bytes. routines for use

=1 139 ; external to this file are:

=1 140 ;

=1 141 ; SENDSTRING - sends a string of characters

=1 142 ; SENDCHAR - sends a single character

=1 143 ; SENDVAL - sends a byte as 2 ASCII characters

=1 144 ; HEX2ASCII - converts from HEX to ASCII

=1 145 ; ASCII2HEX - converts from ASCII to HEX

=1 146 ; GETCHAR - gets a single character

=1 147 ; GETVAL - gets a byte as 2 ASCII characters

=1 148 ;

=1 149 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 150

=1 151 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 152 ; SENDSTRING

=1 153

0083 =1 154 SENDSTRING: ; sends ASCII string to UART starting at location

=1 155 ; DPTR and ending with a null (0) value

=1 156

0083 C0E0 =1 157 PUSH ACC

0085 C0F0 =1 158 PUSH B

0087 E4 =1 159 CLR A

0088 F5F0 =1 160 MOV B,A

008A E5F0 =1 161 IO0010: MOV A,B

008C 05F0 =1 162 INC B

008E 93 =1 163 MOVC A,@A+DPTR

008F 6005 =1 164 JZ IO0020

0091 12009B =1 165 CALL SENDCHAR

0094 80F4 =1 166 JMP IO0010

0096 D0F0 =1 167 IO0020: POP B

0098 D0E0 =1 168 POP ACC

=1 169

009A 22 =1 170 RET

=1 171

=1 172 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 173 ; SENDCHAR

=1 174

836SLAVE PAGE 4

009B =1 175 SENDCHAR: ; sends ASCII value contained in A to UART

=1 176

009B 3099FD =1 177 JNB TI,$ ; wait til present char gone

009E C299 =1 178 CLR TI ; must clear TI

00A0 F599 =1 179 MOV SBUF,A

=1 180

00A2 22 =1 181 RET

=1 182

=1 183 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 184 ; SENDVAL

=1 185

00A3 =1 186 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 187 ; and then spits these two characters up the UART.

=1 188 ; does not change the value of A.

=1 189

00A3 C0E0 =1 190 PUSH ACC

00A5 C4 =1 191 SWAP A

00A6 1200B7 =1 192 CALL HEX2ASCII

00A9 119B =1 193 CALL SENDCHAR ; send high nibble

00AB D0E0 =1 194 POP ACC

00AD C0E0 =1 195 PUSH ACC

00AF 1200B7 =1 196 CALL HEX2ASCII

00B2 119B =1 197 CALL SENDCHAR ; send low nibble

00B4 D0E0 =1 198 POP ACC

=1 199

00B6 22 =1 200 RET

=1 201

=1 202 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 203 ; HEX2ASCII

=1 204

00B7 =1 205 HEX2ASCII: ; converts A into the hex character representing the

=1 206 ; value of A's least significant nibble

=1 207

00B7 540F =1 208 ANL A,#00Fh

00B9 B40A00 =1 209 CJNE A,#00Ah,$+3

00BC 4002 =1 210 JC IO0030

00BE 2407 =1 211 ADD A,#007h

00C0 2430 =1 212 IO0030: ADD A,#'0'

=1 213

00C2 22 =1 214 RET

=1 215

=1 216 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 217 ; ASCII2HEX

=1 218

00C3 =1 219 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 220 ; into the corresponding number (0-15). returns C=1

=1 221 ; when input is other than an ASCII digit,

=1 222 ; indicating invalid output (returned as 255).

=1 223

00C3 C3 =1 224 CLR C

00C4 9430 =1 225 SUBB A,#'0'

00C6 B40A00 =1 226 CJNE A,#10,$+3

00C9 401B =1 227 JC IO0050 ; if '0'<=char<='9', return OK

00CB B41100 =1 228 CJNE A,#17,$+3

00CE 4013 =1 229 JC IO0040 ; if '9'<char<'A', return FAIL

00D0 9407 =1 230 SUBB A,#7

00D2 B41000 =1 231 CJNE A,#10h,$+3

00D5 400F =1 232 JC IO0050 ; if 'A'<=char<='F', return OK

836SLAVE PAGE 5

00D7 B42A00 =1 233 CJNE A,#42,$+3

00DA 4007 =1 234 JC IO0040 ; if 'F'<char<'a', return FAIL

00DC 9420 =1 235 SUBB A,#20h

00DE B41000 =1 236 CJNE A,#10h,$+3

00E1 4003 =1 237 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 238

00E3 C3 =1 239 IO0040: CLR C ; ..else return FAIL

00E4 74FF =1 240 MOV A,#0FFh

=1 241

00E6 B3 =1 242 IO0050: CPL C

00E7 22 =1 243 RET

=1 244

=1 245 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 246 ; GETCHAR

=1 247

00E8 =1 248 GETCHAR: ; waits for a single ASCII character to be received

=1 249 ; by the UART. places this character into A.

=1 250

00E8 3098FD =1 251 JNB RI,$

00EB E599 =1 252 MOV A,SBUF

00ED C298 =1 253 CLR RI

=1 254

00EF 22 =1 255 RET

=1 256

=1 257 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 258 ; GETVAL

=1 259

00F0 =1 260 GETVAL: ; waits for two ASCII hex digits to be received by

=1 261 ; the UART. returns the hex value in A.

=1 262

00F0 C0F0 =1 263 PUSH B

00F2 C000 =1 264 PUSH 0

00F4 C298 =1 265 IO0060: CLR RI

00F6 11E8 =1 266 CALL GETCHAR ; first nibble

00F8 F500 =1 267 MOV 0,A ; store received char

00FA 11C3 =1 268 CALL ASCII2HEX

00FC 40F6 =1 269 JC IO0060 ; if not '0' thru 'F', don't accept

00FE C4 =1 270 SWAP A ; swap nibbles

00FF F5F0 =1 271 MOV B,A ; store nibble in B

0101 E500 =1 272 MOV A,0 ; echo received char

0103 119B =1 273 CALL SENDCHAR

0105 C298 =1 274 IO0070: CLR RI

0107 11E8 =1 275 CALL GETCHAR ; second nibble

0109 F500 =1 276 MOV 0,A ; store received char

010B 11C3 =1 277 CALL ASCII2HEX

010D 40F6 =1 278 JC IO0070 ; if not '0' thru 'F', don't accept

010F 45F0 =1 279 ORL A,B ; combine nibbles

0111 F5F0 =1 280 MOV B,A ; store results in B

0113 E500 =1 281 MOV A,0 ; echo received char

0115 119B =1 282 CALL SENDCHAR

0117 E5F0 =1 283 MOV A,B ; final result

0119 D000 =1 284 POP 0

011B D0F0 =1 285 POP B

=1 286

011D 22 =1 287 RET

=1 288

289

290 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

836SLAVE PAGE 6

291 ; TEXT DATA TABLES

292

011E 0A0D00 293 SEPERATOR: DB 10,13,0

294

295 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

296

297 END

298

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

836SLAVE PAGE 7

ACC. . . . . . . . . . . . . . . D ADDR 00E0H PREDEFINED

ASCII2HEX. . . . . . . . . . . . C ADDR 00C3H

B. . . . . . . . . . . . . . . . D ADDR 00F0H PREDEFINED

EA . . . . . . . . . . . . . . . B ADDR 00AFH PREDEFINED

EX0. . . . . . . . . . . . . . . B ADDR 00A8H PREDEFINED

GETCHAR. . . . . . . . . . . . . C ADDR 00E8H

GETVAL . . . . . . . . . . . . . C ADDR 00F0H NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00B7H

IEIP2. . . . . . . . . . . . . . D ADDR 00A9H PREDEFINED

INPUT. . . . . . . . . . . . . . D ADDR 0060H

IO0010 . . . . . . . . . . . . . C ADDR 008AH

IO0020 . . . . . . . . . . . . . C ADDR 0096H

IO0030 . . . . . . . . . . . . . C ADDR 00C0H

IO0040 . . . . . . . . . . . . . C ADDR 00E3H

IO0050 . . . . . . . . . . . . . C ADDR 00E6H

IO0060 . . . . . . . . . . . . . C ADDR 00F4H

IO0070 . . . . . . . . . . . . . C ADDR 0105H

IT0. . . . . . . . . . . . . . . B ADDR 0088H PREDEFINED

LED. . . . . . . . . . . . . . . NUMB 00B4H

LOOP . . . . . . . . . . . . . . C ADDR 0069H

MAIN . . . . . . . . . . . . . . C ADDR 004BH

OUTPUT . . . . . . . . . . . . . D ADDR 0061H

P3 . . . . . . . . . . . . . . . D ADDR 00B0H PREDEFINED

RI . . . . . . . . . . . . . . . B ADDR 0098H PREDEFINED

SBUF . . . . . . . . . . . . . . D ADDR 0099H PREDEFINED

SCON . . . . . . . . . . . . . . D ADDR 0098H PREDEFINED

SENDCHAR . . . . . . . . . . . . C ADDR 009BH

SENDSTRING . . . . . . . . . . . C ADDR 0083H

SENDVAL. . . . . . . . . . . . . C ADDR 00A3H

SEPERATOR. . . . . . . . . . . . C ADDR 011EH

SP . . . . . . . . . . . . . . . D ADDR 0081H PREDEFINED

SPICON . . . . . . . . . . . . . D ADDR 00F8H PREDEFINED

SPIDAT . . . . . . . . . . . . . D ADDR 00F7H PREDEFINED

T3CON. . . . . . . . . . . . . . D ADDR 009EH PREDEFINED

T3FD . . . . . . . . . . . . . . D ADDR 009DH PREDEFINED

TI . . . . . . . . . . . . . . . B ADDR 0099H PREDEFINED