812SLAVE 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 : ADuC812

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 ; 'SPImstr.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 ADuC812 eval board, you can simply

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

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

23 ; board you must also remove R6 & C6 to disconnect

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

25 ; ensure that LK5 is INSERTED. To configure the

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

27 ;

28 ; Once hardware is connected, download code to both

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

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

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

32 ; LED off until the master starts exchanging data

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

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

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

36 ; are transmitting zeros repeatedly on the SPI port.

37 ; Pressing the INT0 button on either master or slave

38 ; increments the value it is transmitting. Received

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

40 ; on any VT100 terminal or terminal emulator at

41 ; 9600baud/8bits/noparity/1stopbit. Characters sent

42 ; from the terminal to the MicroConverter will update

43 ; the value being transmitted by SPI.

44 ;

45 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

46

47 $MOD812 ; Use 8052 & ADuC812 predefined symbols

48

00B4 49 LED EQU P3.4 ; P3.4 drives red LED on eval board

50

51 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

52 ; DEFINE VARIABLES IN INTERNAL RAM

---- 53 DSEG

0060 54 ORG 0060h

0060 55 INPUT: DS 1 ; data byte received by SPI

0061 56 OUTPUT: DS 1 ; data byte to send by SPI

57

58 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

812SLAVE PAGE 2

59 ; BEGINNING OF CODE

---- 60 CSEG

61

0000 62 ORG 0000h

0000 02004B 63 JMP MAIN ; jump to main program

64

65 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

66 ; INTERRUPT VECTOR SPACE

0003 67 ORG 0003h ; (.................... INT0 ISR)

68

0003 0561 69 INC OUTPUT

0005 32 70 RETI

71

003B 72 ORG 003Bh ; (.................... SPI ISR)

73

003B 85F760 74 MOV INPUT,SPIDAT ; get data just received by SPI

003E 8561F7 75 MOV SPIDAT,OUTPUT ; update next byte to transmit

0041 C3 76 CLR C ; clear C indicates transfer complete

0042 32 77 RETI

78

79 ;====================================================================

80 ; MAIN PROGRAM

004B 81 ORG 004Bh

82

004B 83 MAIN:

84

004B 758107 85 MOV SP,#007h

86

87 ; CONFIGURE UART...

88

004E 759852 89 MOV SCON,#52h ; configure UART for 9600baud..

0051 758920 90 MOV TMOD,#20h ; ..assuming 11.0592MHz crystal

0054 758DFD 91 MOV TH1,#-3

0057 D28E 92 SETB TR1

93

94 ; CONFIGURE SPI...

95

0059 75F824 96 MOV SPICON,#024h ; configure SPI port for:

97 ; CPHA=1, CPOL=0, slave

98

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

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

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

102

005C C295 103 CLR P1.5 ; enable SS pin as input

005E 75A901 104 MOV IE2,#1 ; enable SPI interrupt

105

106 ; CONFIGURE INTERRUPT 0...

107

0061 D288 108 SETB IT0 ; INT0 edge triggered

0063 D2A8 109 SETB EX0 ; enable INT0 interrupt

110

111 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

112

0065 756100 113 MOV OUTPUT,#0 ; set initial value for output byte..

0068 75F700 114 MOV SPIDAT,#0 ; ..including very fisrt output byte

006B D2AF 115 SETB EA ; enable inturrupts

116

812SLAVE PAGE 3

006D B2B4 117 LOOP: CPL LED ; flash the LED on the eval board

006F D3 118 SETB C

0070 40FE 119 JC $ ; wait here to receive SPI transfer

0072 E560 120 MOV A,INPUT ; send value received by SPI..

0074 1200A7 121 CALL SENDVAL ; ..out the UART as 2 hex chars

0077 900122 122 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

007A 120087 123 CALL SENDSTRING ; ..out the UART

007D 3098ED 124 JNB RI,LOOP ; repeat (unless UART data received)

125

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

127

0080 859961 128 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

0083 C298 129 CLR RI ; must clear RI

0085 80E6 130 JMP LOOP ; back to main loop

131

132 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

133 ; SUBROUTINE INCLUDE FILE

134

=1 135 $INCLUDE(UARTIO.asm)

=1 136 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 137 ;

=1 138 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 139 ;

=1 140 ; Date : 12 October 1999

=1 141 ;

=1 142 ; File : UARTIO.hex

=1 143 ;

=1 144 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 145 ;

=1 146 ; Description : standard UART I/O subroutines. total size of this

=1 147 ; code when assembled is 155 bytes. routines for use

=1 148 ; external to this file are:

=1 149 ;

=1 150 ; SENDSTRING - sends a string of characters

=1 151 ; SENDCHAR - sends a single character

=1 152 ; SENDVAL - sends a byte as 2 ASCII characters

=1 153 ; HEX2ASCII - converts from HEX to ASCII

=1 154 ; ASCII2HEX - converts from ASCII to HEX

=1 155 ; GETCHAR - gets a single character

=1 156 ; GETVAL - gets a byte as 2 ASCII characters

=1 157 ;

=1 158 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 159

=1 160 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 161 ; SENDSTRING

=1 162

0087 =1 163 SENDSTRING: ; sends ASCII string to UART starting at location

=1 164 ; DPTR and ending with a null (0) value

=1 165

0087 C0E0 =1 166 PUSH ACC

0089 C0F0 =1 167 PUSH B

008B E4 =1 168 CLR A

008C F5F0 =1 169 MOV B,A

008E E5F0 =1 170 IO0010: MOV A,B

0090 05F0 =1 171 INC B

0092 93 =1 172 MOVC A,@A+DPTR

0093 6005 =1 173 JZ IO0020

0095 12009F =1 174 CALL SENDCHAR

812SLAVE PAGE 4

0098 80F4 =1 175 JMP IO0010

009A D0F0 =1 176 IO0020: POP B

009C D0E0 =1 177 POP ACC

=1 178

009E 22 =1 179 RET

=1 180

=1 181 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 182 ; SENDCHAR

=1 183

009F =1 184 SENDCHAR: ; sends ASCII value contained in A to UART

=1 185

009F 3099FD =1 186 JNB TI,$ ; wait til present char gone

00A2 C299 =1 187 CLR TI ; must clear TI

00A4 F599 =1 188 MOV SBUF,A

=1 189

00A6 22 =1 190 RET

=1 191

=1 192 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 193 ; SENDVAL

=1 194

00A7 =1 195 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 196 ; and then spits these two characters up the UART.

=1 197 ; does not change the value of A.

=1 198

00A7 C0E0 =1 199 PUSH ACC

00A9 C4 =1 200 SWAP A

00AA 1200BB =1 201 CALL HEX2ASCII

00AD 119F =1 202 CALL SENDCHAR ; send high nibble

00AF D0E0 =1 203 POP ACC

00B1 C0E0 =1 204 PUSH ACC

00B3 1200BB =1 205 CALL HEX2ASCII

00B6 119F =1 206 CALL SENDCHAR ; send low nibble

00B8 D0E0 =1 207 POP ACC

=1 208

00BA 22 =1 209 RET

=1 210

=1 211 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 212 ; HEX2ASCII

=1 213

00BB =1 214 HEX2ASCII: ; converts A into the hex character representing the

=1 215 ; value of A's least significant nibble

=1 216

00BB 540F =1 217 ANL A,#00Fh

00BD B40A00 =1 218 CJNE A,#00Ah,$+3

00C0 4002 =1 219 JC IO0030

00C2 2407 =1 220 ADD A,#007h

00C4 2430 =1 221 IO0030: ADD A,#'0'

=1 222

00C6 22 =1 223 RET

=1 224

=1 225 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 226 ; ASCII2HEX

=1 227

00C7 =1 228 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 229 ; into the corresponding number (0-15). returns C=1

=1 230 ; when input is other than an ASCII digit,

=1 231 ; indicating invalid output (returned as 255).

=1 232

812SLAVE PAGE 5

00C7 C3 =1 233 CLR C

00C8 9430 =1 234 SUBB A,#'0'

00CA B40A00 =1 235 CJNE A,#10,$+3

00CD 401B =1 236 JC IO0050 ; if '0'<=char<='9', return OK

00CF B41100 =1 237 CJNE A,#17,$+3

00D2 4013 =1 238 JC IO0040 ; if '9'<char<'A', return FAIL

00D4 9407 =1 239 SUBB A,#7

00D6 B41000 =1 240 CJNE A,#10h,$+3

00D9 400F =1 241 JC IO0050 ; if 'A'<=char<='F', return OK

00DB B42A00 =1 242 CJNE A,#42,$+3

00DE 4007 =1 243 JC IO0040 ; if 'F'<char<'a', return FAIL

00E0 9420 =1 244 SUBB A,#20h

00E2 B41000 =1 245 CJNE A,#10h,$+3

00E5 4003 =1 246 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 247

00E7 C3 =1 248 IO0040: CLR C ; ..else return FAIL

00E8 74FF =1 249 MOV A,#0FFh

=1 250

00EA B3 =1 251 IO0050: CPL C

00EB 22 =1 252 RET

=1 253

=1 254 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 255 ; GETCHAR

=1 256

00EC =1 257 GETCHAR: ; waits for a single ASCII character to be received

=1 258 ; by the UART. places this character into A.

=1 259

00EC 3098FD =1 260 JNB RI,$

00EF E599 =1 261 MOV A,SBUF

00F1 C298 =1 262 CLR RI

=1 263

00F3 22 =1 264 RET

=1 265

=1 266 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 267 ; GETVAL

=1 268

00F4 =1 269 GETVAL: ; waits for two ASCII hex digits to be received by

=1 270 ; the UART. returns the hex value in A.

=1 271

00F4 C0F0 =1 272 PUSH B

00F6 C000 =1 273 PUSH 0

00F8 C298 =1 274 IO0060: CLR RI

00FA 11EC =1 275 CALL GETCHAR ; first nibble

00FC F500 =1 276 MOV 0,A ; store received char

00FE 11C7 =1 277 CALL ASCII2HEX

0100 40F6 =1 278 JC IO0060 ; if not '0' thru 'F', don't accept

0102 C4 =1 279 SWAP A ; swap nibbles

0103 F5F0 =1 280 MOV B,A ; store nibble in B

0105 E500 =1 281 MOV A,0 ; echo received char

0107 119F =1 282 CALL SENDCHAR

0109 C298 =1 283 IO0070: CLR RI

010B 11EC =1 284 CALL GETCHAR ; second nibble

010D F500 =1 285 MOV 0,A ; store received char

010F 11C7 =1 286 CALL ASCII2HEX

0111 40F6 =1 287 JC IO0070 ; if not '0' thru 'F', don't accept

0113 45F0 =1 288 ORL A,B ; combine nibbles

0115 F5F0 =1 289 MOV B,A ; store results in B

0117 E500 =1 290 MOV A,0 ; echo received char

812SLAVE PAGE 6

0119 119F =1 291 CALL SENDCHAR

011B E5F0 =1 292 MOV A,B ; final result

011D D000 =1 293 POP 0

011F D0F0 =1 294 POP B

=1 295

0121 22 =1 296 RET

=1 297

298

299 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

300 ; TEXT DATA TABLES

301

0122 0A0D00 302 SEPERATOR: DB 10,13,0

303

304 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

305

306 END

307

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

812SLAVE PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00C7H

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

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 00ECH

GETVAL . . . . . . . . . . . . . C ADDR 00F4H NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00BBH

IE2. . . . . . . . . . . . . . . D ADDR 00A9H PREDEFINED

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

IO0010 . . . . . . . . . . . . . C ADDR 008EH

IO0020 . . . . . . . . . . . . . C ADDR 009AH

IO0030 . . . . . . . . . . . . . C ADDR 00C4H

IO0040 . . . . . . . . . . . . . C ADDR 00E7H

IO0050 . . . . . . . . . . . . . C ADDR 00EAH

IO0060 . . . . . . . . . . . . . C ADDR 00F8H

IO0070 . . . . . . . . . . . . . C ADDR 0109H

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

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

LOOP . . . . . . . . . . . . . . C ADDR 006DH

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

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

P1 . . . . . . . . . . . . . . . D ADDR 0090H PREDEFINED

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

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

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 009FH

SENDSTRING . . . . . . . . . . . C ADDR 0087H

SENDVAL. . . . . . . . . . . . . C ADDR 00A7H

SEPERATOR. . . . . . . . . . . . C ADDR 0122H

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

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

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

TH1. . . . . . . . . . . . . . . D ADDR 008DH PREDEFINED

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

TMOD . . . . . . . . . . . . . . D ADDR 0089H PREDEFINED

TR1. . . . . . . . . . . . . . . B ADDR 008EH PREDEFINED