842SLAVE PAGE 1

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

2 ;

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

4 ;

5 ; Date : October 2003

6 ;

7 ; File : SPIslave.asm

8 ;

9 ; Hardware : ADuC842/ADuC843

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 ; '842mstr.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 ADuC842 eval board, you can

21 ; simply connect the 5-pin SPI header directly

22 ; to that of the master board.

23 ;

24 ; Once hardware is connected, download code to both

25 ; master & slave devices ('842mstr' to the master,

26 ; '842slave' to the slave). Reset the slave first,

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

28 ; LED off until the master starts exchanging data

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

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

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

32 ; are transmitting zeros repeatedly on the SPI port.

33 ; Pressing the INT0 button on either master or slave

34 ; increments the value it is transmitting. Received

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

36 ; on any VT100 terminal or terminal emulator at

37 ; 9600baud/8bits/noparity/1stopbit. Characters sent

38 ; from the terminal to the MicroConverter will update

39 ; the value being transmitted by SPI.

40 ;

41 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

42

43 $MOD842 ; Use 8052 & ADuC842 predefined symbols

44

00B4 45 LED EQU P3.4 ; P3.4 drives red LED on eval board

46

47 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

48 ; DEFINE VARIABLES IN INTERNAL RAM

---- 49 DSEG

0060 50 ORG 0060h

0060 51 INPUT: DS 1 ; data byte received by SPI

0061 52 OUTPUT: DS 1 ; data byte to send by SPI

53

54 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

55 ; BEGINNING OF CODE

---- 56 CSEG

57

0000 58 ORG 0000h

842SLAVE PAGE 2

0000 02004B 59 JMP MAIN ; jump to main program

60

61 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

62 ; INTERRUPT VECTOR SPACE

0003 63 ORG 0003h ; (.................... INT0 ISR)

64

0003 0561 65 INC OUTPUT

0005 32 66 RETI

67

003B 68 ORG 003Bh ; (.................... SPI ISR)

69

003B 85F760 70 MOV INPUT,SPIDAT ; get data just received by SPI

003E 8561F7 71 MOV SPIDAT,OUTPUT ; update next byte to transmit

0041 C3 72 CLR C ; clear C indicates transfer complete

0042 32 73 RETI

74

75 ;====================================================================

76 ; MAIN PROGRAM

004B 77 ORG 004Bh

78

004B 79 MAIN:

004B 75D703 80 MOV PLLCON,#03H

004E 758107 81 MOV SP,#007h

82

83 ; CONFIGURE UART...

0051 759E83 84 MOV T3CON,#083h

0054 759D2D 85 Mov T3FD,#02Dh

0057 759852 86 MOV SCON,#052h

87

88 ; CONFIGURE SPI...

89

005A 75F824 90 MOV SPICON,#024h ; configure SPI port for:

91 ; CPHA=1, CPOL=0, slave

005D 75A901 92 MOV IEIP2,#1 ; enable SPI interrupt

93

94 ; CONFIGURE INTERRUPT 0...

95

0060 D288 96 SETB IT0 ; INT0 edge triggered

0062 D2A8 97 SETB EX0 ; enable INT0 interrupt

98

99 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

100

0064 756100 101 MOV OUTPUT,#0 ; set initial value for output byte..

0067 75F700 102 MOV SPIDAT,#0 ; ..including very fisrt output byte

006A D2AF 103 SETB EA ; enable inturrupts

104

006C B2B4 105 LOOP: CPL LED ; flash the LED on the eval board

006E D3 106 SETB C

006F 40FE 107 JC $ ; wait here to receive SPI transfer

0071 E560 108 MOV A,INPUT ; send value received by SPI..

0073 1200A6 109 CALL SENDVAL ; ..out the UART as 2 hex chars

0076 900121 110 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

0079 120086 111 CALL SENDSTRING ; ..out the UART

007C 3098ED 112 JNB RI,LOOP ; repeat (unless UART data received)

113

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

115

007F 859961 116 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

842SLAVE PAGE 3

0082 C298 117 CLR RI ; must clear RI

0084 80E6 118 JMP LOOP ; back to main loop

119

120 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

121 ; SUBROUTINE INCLUDE FILE

122

=1 123 $INCLUDE(UARTIO.asm)

=1 124 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 125 ;

=1 126 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 127 ;

=1 128 ; Date : January 2001

=1 129 ;

=1 130 ; File : UARTIO.asm

=1 131 ;

=1 132 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 133 ;

=1 134 ; Description : standard UART I/O subroutines. total size of this

=1 135 ; code when assembled is 155 bytes. routines for use

=1 136 ; external to this file are:

=1 137 ;

=1 138 ; SENDSTRING - sends a string of characters

=1 139 ; SENDCHAR - sends a single character

=1 140 ; SENDVAL - sends a byte as 2 ASCII characters

=1 141 ; HEX2ASCII - converts from HEX to ASCII

=1 142 ; ASCII2HEX - converts from ASCII to HEX

=1 143 ; GETCHAR - gets a single character

=1 144 ; GETVAL - gets a byte as 2 ASCII characters

=1 145 ;

=1 146 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 147

=1 148 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 149 ; SENDSTRING

=1 150

0086 =1 151 SENDSTRING: ; sends ASCII string to UART starting at location

=1 152 ; DPTR and ending with a null (0) value

=1 153

0086 C0E0 =1 154 PUSH ACC

0088 C0F0 =1 155 PUSH B

008A E4 =1 156 CLR A

008B F5F0 =1 157 MOV B,A

008D E5F0 =1 158 IO0010: MOV A,B

008F 05F0 =1 159 INC B

0091 93 =1 160 MOVC A,@A+DPTR

0092 6005 =1 161 JZ IO0020

0094 12009E =1 162 CALL SENDCHAR

0097 80F4 =1 163 JMP IO0010

0099 D0F0 =1 164 IO0020: POP B

009B D0E0 =1 165 POP ACC

=1 166

009D 22 =1 167 RET

=1 168

=1 169 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 170 ; SENDCHAR

=1 171

009E =1 172 SENDCHAR: ; sends ASCII value contained in A to UART

=1 173

009E 3099FD =1 174 JNB TI,$ ; wait til present char gone

842SLAVE PAGE 4

00A1 C299 =1 175 CLR TI ; must clear TI

00A3 F599 =1 176 MOV SBUF,A

=1 177

00A5 22 =1 178 RET

=1 179

=1 180 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 181 ; SENDVAL

=1 182

00A6 =1 183 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 184 ; and then spits these two characters up the UART.

=1 185 ; does not change the value of A.

=1 186

00A6 C0E0 =1 187 PUSH ACC

00A8 C4 =1 188 SWAP A

00A9 1200BA =1 189 CALL HEX2ASCII

00AC 119E =1 190 CALL SENDCHAR ; send high nibble

00AE D0E0 =1 191 POP ACC

00B0 C0E0 =1 192 PUSH ACC

00B2 1200BA =1 193 CALL HEX2ASCII

00B5 119E =1 194 CALL SENDCHAR ; send low nibble

00B7 D0E0 =1 195 POP ACC

=1 196

00B9 22 =1 197 RET

=1 198

=1 199 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 200 ; HEX2ASCII

=1 201

00BA =1 202 HEX2ASCII: ; converts A into the hex character representing the

=1 203 ; value of A's least significant nibble

=1 204

00BA 540F =1 205 ANL A,#00Fh

00BC B40A00 =1 206 CJNE A,#00Ah,$+3

00BF 4002 =1 207 JC IO0030

00C1 2407 =1 208 ADD A,#007h

00C3 2430 =1 209 IO0030: ADD A,#'0'

=1 210

00C5 22 =1 211 RET

=1 212

=1 213 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 214 ; ASCII2HEX

=1 215

00C6 =1 216 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 217 ; into the corresponding number (0-15). returns C=1

=1 218 ; when input is other than an ASCII digit,

=1 219 ; indicating invalid output (returned as 255).

=1 220

00C6 C3 =1 221 CLR C

00C7 9430 =1 222 SUBB A,#'0'

00C9 B40A00 =1 223 CJNE A,#10,$+3

00CC 401B =1 224 JC IO0050 ; if '0'<=char<='9', return OK

00CE B41100 =1 225 CJNE A,#17,$+3

00D1 4013 =1 226 JC IO0040 ; if '9'<char<'A', return FAIL

00D3 9407 =1 227 SUBB A,#7

00D5 B41000 =1 228 CJNE A,#10h,$+3

00D8 400F =1 229 JC IO0050 ; if 'A'<=char<='F', return OK

00DA B42A00 =1 230 CJNE A,#42,$+3

00DD 4007 =1 231 JC IO0040 ; if 'F'<char<'a', return FAIL

00DF 9420 =1 232 SUBB A,#20h

842SLAVE PAGE 5

00E1 B41000 =1 233 CJNE A,#10h,$+3

00E4 4003 =1 234 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 235

00E6 C3 =1 236 IO0040: CLR C ; ..else return FAIL

00E7 74FF =1 237 MOV A,#0FFh

=1 238

00E9 B3 =1 239 IO0050: CPL C

00EA 22 =1 240 RET

=1 241

=1 242 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 243 ; GETCHAR

=1 244

00EB =1 245 GETCHAR: ; waits for a single ASCII character to be received

=1 246 ; by the UART. places this character into A.

=1 247

00EB 3098FD =1 248 JNB RI,$

00EE E599 =1 249 MOV A,SBUF

00F0 C298 =1 250 CLR RI

=1 251

00F2 22 =1 252 RET

=1 253

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

=1 255 ; GETVAL

=1 256

00F3 =1 257 GETVAL: ; waits for two ASCII hex digits to be received by

=1 258 ; the UART. returns the hex value in A.

=1 259

00F3 C0F0 =1 260 PUSH B

00F5 C000 =1 261 PUSH 0

00F7 C298 =1 262 IO0060: CLR RI

00F9 11EB =1 263 CALL GETCHAR ; first nibble

00FB F500 =1 264 MOV 0,A ; store received char

00FD 11C6 =1 265 CALL ASCII2HEX

00FF 40F6 =1 266 JC IO0060 ; if not '0' thru 'F', don't accept

0101 C4 =1 267 SWAP A ; swap nibbles

0102 F5F0 =1 268 MOV B,A ; store nibble in B

0104 E500 =1 269 MOV A,0 ; echo received char

0106 119E =1 270 CALL SENDCHAR

0108 C298 =1 271 IO0070: CLR RI

010A 11EB =1 272 CALL GETCHAR ; second nibble

010C F500 =1 273 MOV 0,A ; store received char

010E 11C6 =1 274 CALL ASCII2HEX

0110 40F6 =1 275 JC IO0070 ; if not '0' thru 'F', don't accept

0112 45F0 =1 276 ORL A,B ; combine nibbles

0114 F5F0 =1 277 MOV B,A ; store results in B

0116 E500 =1 278 MOV A,0 ; echo received char

0118 119E =1 279 CALL SENDCHAR

011A E5F0 =1 280 MOV A,B ; final result

011C D000 =1 281 POP 0

011E D0F0 =1 282 POP B

=1 283

0120 22 =1 284 RET

=1 285

286

287 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

288 ; TEXT DATA TABLES

289

0121 0A0D00 290 SEPERATOR: DB 10,13,0

842SLAVE PAGE 6

291

292 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

293

294 END

295

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

842SLAVE PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00C6H

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

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 00EBH

GETVAL . . . . . . . . . . . . . C ADDR 00F3H NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00BAH

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

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

IO0010 . . . . . . . . . . . . . C ADDR 008DH

IO0020 . . . . . . . . . . . . . C ADDR 0099H

IO0030 . . . . . . . . . . . . . C ADDR 00C3H

IO0040 . . . . . . . . . . . . . C ADDR 00E6H

IO0050 . . . . . . . . . . . . . C ADDR 00E9H

IO0060 . . . . . . . . . . . . . C ADDR 00F7H

IO0070 . . . . . . . . . . . . . C ADDR 0108H

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

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

LOOP . . . . . . . . . . . . . . C ADDR 006CH

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

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

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

PLLCON . . . . . . . . . . . . . D ADDR 00D7H PREDEFINED

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

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 009EH

SENDSTRING . . . . . . . . . . . C ADDR 0086H

SENDVAL. . . . . . . . . . . . . C ADDR 00A6H

SEPERATOR. . . . . . . . . . . . C ADDR 0121H

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