816SLAVE PAGE 1

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

2 ;

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

4 ;

5 ; Date : January 2001

6 ;

7 ; File : 816slave.asm

8 ;

9 ; Hardware : 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 ; '816mstr.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 ADuC816 eval board, you can simply

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

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

23 ; also ensure that LK1 ('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 ('816mstr' to the master,

28 ; '816slave' 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 $MOD816 ; Use 8052 & ADuC816 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

816SLAVE 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 75CBFF 87 MOV RCAP2H,#0FFh ; configure UART for 9830baud

0051 75CAFB 88 MOV RCAP2L,#-5 ; (close enough to 9600baud)

0054 75CDFF 89 MOV TH2,#0FFh

0057 75CCFB 90 MOV TL2,#-5

005A 759852 91 MOV SCON,#01010010b

005D 75C834 92 MOV T2CON,#00110100b

93

94 ; CONFIGURE SPI...

95

0060 75F824 96 MOV SPICON,#024h ; configure SPI port for:

97 ; CPHA=1, CPOL=0, slave

0063 75A901 98 MOV IEIP2,#1 ; enable SPI interrupt

99

100 ; CONFIGURE INTERRUPT 0...

101

0066 D288 102 SETB IT0 ; INT0 edge triggered

0068 D2A8 103 SETB EX0 ; enable INT0 interrupt

104

105 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

106

006A 756100 107 MOV OUTPUT,#0 ; set initial value for output byte..

006D 75F700 108 MOV SPIDAT,#0 ; ..including very fisrt output byte

0070 D2AF 109 SETB EA ; enable inturrupts

110

0072 B2B4 111 LOOP: CPL LED ; flash the LED on the eval board

0074 D3 112 SETB C

0075 40FE 113 JC $ ; wait here to receive SPI transfer

0077 E560 114 MOV A,INPUT ; send value received by SPI..

0079 1200AC 115 CALL SENDVAL ; ..out the UART as 2 hex chars

007C 900127 116 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

816SLAVE PAGE 3

007F 12008C 117 CALL SENDSTRING ; ..out the UART

0082 3098ED 118 JNB RI,LOOP ; repeat (unless UART data received)

119

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

121

0085 859961 122 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

0088 C298 123 CLR RI ; must clear RI

008A 80E6 124 JMP LOOP ; back to main loop

125

126 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

127 ; SUBROUTINE INCLUDE FILE

128

=1 129 $INCLUDE(UARTIO.asm)

=1 130 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 131 ;

=1 132 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 133 ;

=1 134 ; Date : January 2001

=1 135 ;

=1 136 ; File : UARTIO.hex

=1 137 ;

=1 138 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 139 ;

=1 140 ; Description : standard UART I/O subroutines. total size of this

=1 141 ; code when assembled is 155 bytes. routines for use

=1 142 ; external to this file are:

=1 143 ;

=1 144 ; SENDSTRING - sends a string of characters

=1 145 ; SENDCHAR - sends a single character

=1 146 ; SENDVAL - sends a byte as 2 ASCII characters

=1 147 ; HEX2ASCII - converts from HEX to ASCII

=1 148 ; ASCII2HEX - converts from ASCII to HEX

=1 149 ; GETCHAR - gets a single character

=1 150 ; GETVAL - gets a byte as 2 ASCII characters

=1 151 ;

=1 152 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 153

=1 154 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 155 ; SENDSTRING

=1 156

008C =1 157 SENDSTRING: ; sends ASCII string to UART starting at location

=1 158 ; DPTR and ending with a null (0) value

=1 159

008C C0E0 =1 160 PUSH ACC

008E C0F0 =1 161 PUSH B

0090 E4 =1 162 CLR A

0091 F5F0 =1 163 MOV B,A

0093 E5F0 =1 164 IO0010: MOV A,B

0095 05F0 =1 165 INC B

0097 93 =1 166 MOVC A,@A+DPTR

0098 6005 =1 167 JZ IO0020

009A 1200A4 =1 168 CALL SENDCHAR

009D 80F4 =1 169 JMP IO0010

009F D0F0 =1 170 IO0020: POP B

00A1 D0E0 =1 171 POP ACC

=1 172

00A3 22 =1 173 RET

=1 174

816SLAVE PAGE 4

=1 175 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 176 ; SENDCHAR

=1 177

00A4 =1 178 SENDCHAR: ; sends ASCII value contained in A to UART

=1 179

00A4 3099FD =1 180 JNB TI,$ ; wait til present char gone

00A7 C299 =1 181 CLR TI ; must clear TI

00A9 F599 =1 182 MOV SBUF,A

=1 183

00AB 22 =1 184 RET

=1 185

=1 186 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 187 ; SENDVAL

=1 188

00AC =1 189 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 190 ; and then spits these two characters up the UART.

=1 191 ; does not change the value of A.

=1 192

00AC C0E0 =1 193 PUSH ACC

00AE C4 =1 194 SWAP A

00AF 1200C0 =1 195 CALL HEX2ASCII

00B2 11A4 =1 196 CALL SENDCHAR ; send high nibble

00B4 D0E0 =1 197 POP ACC

00B6 C0E0 =1 198 PUSH ACC

00B8 1200C0 =1 199 CALL HEX2ASCII

00BB 11A4 =1 200 CALL SENDCHAR ; send low nibble

00BD D0E0 =1 201 POP ACC

=1 202

00BF 22 =1 203 RET

=1 204

=1 205 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 206 ; HEX2ASCII

=1 207

00C0 =1 208 HEX2ASCII: ; converts A into the hex character representing the

=1 209 ; value of A's least significant nibble

=1 210

00C0 540F =1 211 ANL A,#00Fh

00C2 B40A00 =1 212 CJNE A,#00Ah,$+3

00C5 4002 =1 213 JC IO0030

00C7 2407 =1 214 ADD A,#007h

00C9 2430 =1 215 IO0030: ADD A,#'0'

=1 216

00CB 22 =1 217 RET

=1 218

=1 219 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 220 ; ASCII2HEX

=1 221

00CC =1 222 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 223 ; into the corresponding number (0-15). returns C=1

=1 224 ; when input is other than an ASCII digit,

=1 225 ; indicating invalid output (returned as 255).

=1 226

00CC C3 =1 227 CLR C

00CD 9430 =1 228 SUBB A,#'0'

00CF B40A00 =1 229 CJNE A,#10,$+3

00D2 401B =1 230 JC IO0050 ; if '0'<=char<='9', return OK

00D4 B41100 =1 231 CJNE A,#17,$+3

00D7 4013 =1 232 JC IO0040 ; if '9'<char<'A', return FAIL

816SLAVE PAGE 5

00D9 9407 =1 233 SUBB A,#7

00DB B41000 =1 234 CJNE A,#10h,$+3

00DE 400F =1 235 JC IO0050 ; if 'A'<=char<='F', return OK

00E0 B42A00 =1 236 CJNE A,#42,$+3

00E3 4007 =1 237 JC IO0040 ; if 'F'<char<'a', return FAIL

00E5 9420 =1 238 SUBB A,#20h

00E7 B41000 =1 239 CJNE A,#10h,$+3

00EA 4003 =1 240 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 241

00EC C3 =1 242 IO0040: CLR C ; ..else return FAIL

00ED 74FF =1 243 MOV A,#0FFh

=1 244

00EF B3 =1 245 IO0050: CPL C

00F0 22 =1 246 RET

=1 247

=1 248 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 249 ; GETCHAR

=1 250

00F1 =1 251 GETCHAR: ; waits for a single ASCII character to be received

=1 252 ; by the UART. places this character into A.

=1 253

00F1 3098FD =1 254 JNB RI,$

00F4 E599 =1 255 MOV A,SBUF

00F6 C298 =1 256 CLR RI

=1 257

00F8 22 =1 258 RET

=1 259

=1 260 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 261 ; GETVAL

=1 262

00F9 =1 263 GETVAL: ; waits for two ASCII hex digits to be received by

=1 264 ; the UART. returns the hex value in A.

=1 265

00F9 C0F0 =1 266 PUSH B

00FB C000 =1 267 PUSH 0

00FD C298 =1 268 IO0060: CLR RI

00FF 11F1 =1 269 CALL GETCHAR ; first nibble

0101 F500 =1 270 MOV 0,A ; store received char

0103 11CC =1 271 CALL ASCII2HEX

0105 40F6 =1 272 JC IO0060 ; if not '0' thru 'F', don't accept

0107 C4 =1 273 SWAP A ; swap nibbles

0108 F5F0 =1 274 MOV B,A ; store nibble in B

010A E500 =1 275 MOV A,0 ; echo received char

010C 11A4 =1 276 CALL SENDCHAR

010E C298 =1 277 IO0070: CLR RI

0110 11F1 =1 278 CALL GETCHAR ; second nibble

0112 F500 =1 279 MOV 0,A ; store received char

0114 11CC =1 280 CALL ASCII2HEX

0116 40F6 =1 281 JC IO0070 ; if not '0' thru 'F', don't accept

0118 45F0 =1 282 ORL A,B ; combine nibbles

011A F5F0 =1 283 MOV B,A ; store results in B

011C E500 =1 284 MOV A,0 ; echo received char

011E 11A4 =1 285 CALL SENDCHAR

0120 E5F0 =1 286 MOV A,B ; final result

0122 D000 =1 287 POP 0

0124 D0F0 =1 288 POP B

=1 289

0126 22 =1 290 RET

816SLAVE PAGE 6

=1 291

292

293 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

294 ; TEXT DATA TABLES

295

0127 0A0D00 296 SEPERATOR: DB 10,13,0

297

298 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

299

300 END

301

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

816SLAVE PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00CCH

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

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 00F1H

GETVAL . . . . . . . . . . . . . C ADDR 00F9H NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00C0H

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

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

IO0010 . . . . . . . . . . . . . C ADDR 0093H

IO0020 . . . . . . . . . . . . . C ADDR 009FH

IO0030 . . . . . . . . . . . . . C ADDR 00C9H

IO0040 . . . . . . . . . . . . . C ADDR 00ECH

IO0050 . . . . . . . . . . . . . C ADDR 00EFH

IO0060 . . . . . . . . . . . . . C ADDR 00FDH

IO0070 . . . . . . . . . . . . . C ADDR 010EH

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

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

LOOP . . . . . . . . . . . . . . C ADDR 0072H

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

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

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

RCAP2H . . . . . . . . . . . . . D ADDR 00CBH PREDEFINED

RCAP2L . . . . . . . . . . . . . D ADDR 00CAH PREDEFINED

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

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 00A4H

SENDSTRING . . . . . . . . . . . C ADDR 008CH

SENDVAL. . . . . . . . . . . . . C ADDR 00ACH

SEPERATOR. . . . . . . . . . . . C ADDR 0127H

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

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

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

T2CON. . . . . . . . . . . . . . D ADDR 00C8H PREDEFINED

TH2. . . . . . . . . . . . . . . D ADDR 00CDH PREDEFINED

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

TL2. . . . . . . . . . . . . . . D ADDR 00CCH PREDEFINED