812MSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : 7 March 2000

6 ;

7 ; File : SPImstr.asm

8 ;

9 ; Hardware : ADuC812

10 ;

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

12 ;

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

14 ; Code is intended for use with companion code file

15 ; 'SPIslave.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 slave board. However, on the master

23 ; board you must also ensure that LK5 is REMOVED, and

24 ; you'll have to manually jumper P3.5 to J3/pin7 (the

25 ; SS pin of the SPI/I2C header). To configure the

26 ; slave board, refer to 'SPIslave.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

00B5 50 SS EQU P3.5 ; P3.5 drives slave device's SS pin

51

52 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

53 ; DEFINE VARIABLES IN INTERNAL RAM

---- 54 DSEG

0060 55 ORG 0060h

0060 56 INPUT: DS 1 ; data byte received by SPI

0061 57 OUTPUT: DS 1 ; data byte to send by SPI

58

812MSTR PAGE 2

59 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

60 ; BEGINNING OF CODE

---- 61 CSEG

62

0000 63 ORG 0000h

0000 02004B 64 JMP MAIN ; jump to main program

65

66 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

67 ; INTERRUPT VECTOR SPACE

0003 68 ORG 0003h ; (.................... INT0 ISR)

69

0003 0561 70 INC OUTPUT

0005 32 71 RETI

72

003B 73 ORG 003Bh ; (.................... SPI ISR)

74

003B D2B5 75 SETB SS ; pull slave's SS pin high

003D 85F760 76 MOV INPUT,SPIDAT

0040 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 75F837 96 MOV SPICON,#037h ; configure SPI port for:

97 ; Fosc/64, CPHA=1, CPOL=0, master

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 75A901 103 MOV IE2,#1 ; enable SPI interrupt

104

105 ; CONFIGURE INTERRUPT 0...

106

005F D288 107 SETB IT0 ; INT0 edge triggered

0061 D2A8 108 SETB EX0 ; enable INT0 interrupt

109

110 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

111

0063 756100 112 MOV OUTPUT,#0 ; set initial value for output byte

0066 D2AF 113 SETB EA ; enable inturrupts

114

0068 B2B4 115 LOOP: CPL LED ; flash the LED on the eval board

006A E561 116 MOV A,OUTPUT ; byte to send via SPI into ACC

812MSTR PAGE 3

006C 120087 117 CALL SENDSPI ; trigger SPI send/receive transfer

006F 12008D 118 CALL DELAY ; pause 100ms

0072 E560 119 MOV A,INPUT ; send value received by SPI..

0074 1200C1 120 CALL SENDVAL ; ..out the UART as 2 hex chars

0077 90013C 121 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

007A 1200A1 122 CALL SENDSTRING ; ..out the UART

007D 3098E8 123 JNB RI,LOOP ; repeat (unless UART data received)

124

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

126

0080 859961 127 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

0083 C298 128 CLR RI ; must clear RI

0085 80E1 129 JMP LOOP ; back to main loop

130

131 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

132 ; SUBROUTINES

133

0087 134 SENDSPI: ; sends the value in ACC out the SPI port. also

135 ; receives simultaneously into SPIDAT. SPI interrupt

136 ; is triggered when transfer is complete.

137

0087 C2B5 138 CLR SS ; must pull slave's SS pin low first

0089 8561F7 139 MOV SPIDAT,OUTPUT ; trigger data transfer

008C 22 140 RET

141

142 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

143

008D 144 DELAY: ; delays approximately 100ms

145

008D C0E0 146 PUSH ACC

008F C0F0 147 PUSH B

0091 74C8 148 MOV A,#200 ; 200 \* 500us = 100ms

0093 75F0E5 149 DLY1: MOV B,#229 ; 229 \* 2.17us = 500us

0096 D5F0FD 150 DJNZ B,$ ; sit here for 500us

0099 D5E0F7 151 DJNZ ACC,DLY1 ; repeat 200 times (100ms delay)

009C D0F0 152 POP B

009E D0E0 153 POP ACC

00A0 22 154 RET

155

156 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

157 ; SUBROUTINE INCLUDE FILE

158

=1 159 $INCLUDE(UARTIO.asm)

=1 160 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 161 ;

=1 162 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 163 ;

=1 164 ; Date : 12 October 1999

=1 165 ;

=1 166 ; File : UARTIO.hex

=1 167 ;

=1 168 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 169 ;

=1 170 ; Description : standard UART I/O subroutines. total size of this

=1 171 ; code when assembled is 155 bytes. routines for use

=1 172 ; external to this file are:

=1 173 ;

=1 174 ; SENDSTRING - sends a string of characters

812MSTR PAGE 4

=1 175 ; SENDCHAR - sends a single character

=1 176 ; SENDVAL - sends a byte as 2 ASCII characters

=1 177 ; HEX2ASCII - converts from HEX to ASCII

=1 178 ; ASCII2HEX - converts from ASCII to HEX

=1 179 ; GETCHAR - gets a single character

=1 180 ; GETVAL - gets a byte as 2 ASCII characters

=1 181 ;

=1 182 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 183

=1 184 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 185 ; SENDSTRING

=1 186

00A1 =1 187 SENDSTRING: ; sends ASCII string to UART starting at location

=1 188 ; DPTR and ending with a null (0) value

=1 189

00A1 C0E0 =1 190 PUSH ACC

00A3 C0F0 =1 191 PUSH B

00A5 E4 =1 192 CLR A

00A6 F5F0 =1 193 MOV B,A

00A8 E5F0 =1 194 IO0010: MOV A,B

00AA 05F0 =1 195 INC B

00AC 93 =1 196 MOVC A,@A+DPTR

00AD 6005 =1 197 JZ IO0020

00AF 1200B9 =1 198 CALL SENDCHAR

00B2 80F4 =1 199 JMP IO0010

00B4 D0F0 =1 200 IO0020: POP B

00B6 D0E0 =1 201 POP ACC

=1 202

00B8 22 =1 203 RET

=1 204

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

=1 206 ; SENDCHAR

=1 207

00B9 =1 208 SENDCHAR: ; sends ASCII value contained in A to UART

=1 209

00B9 3099FD =1 210 JNB TI,$ ; wait til present char gone

00BC C299 =1 211 CLR TI ; must clear TI

00BE F599 =1 212 MOV SBUF,A

=1 213

00C0 22 =1 214 RET

=1 215

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

=1 217 ; SENDVAL

=1 218

00C1 =1 219 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 220 ; and then spits these two characters up the UART.

=1 221 ; does not change the value of A.

=1 222

00C1 C0E0 =1 223 PUSH ACC

00C3 C4 =1 224 SWAP A

00C4 1200D5 =1 225 CALL HEX2ASCII

00C7 11B9 =1 226 CALL SENDCHAR ; send high nibble

00C9 D0E0 =1 227 POP ACC

00CB C0E0 =1 228 PUSH ACC

00CD 1200D5 =1 229 CALL HEX2ASCII

00D0 11B9 =1 230 CALL SENDCHAR ; send low nibble

00D2 D0E0 =1 231 POP ACC

=1 232

812MSTR PAGE 5

00D4 22 =1 233 RET

=1 234

=1 235 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 236 ; HEX2ASCII

=1 237

00D5 =1 238 HEX2ASCII: ; converts A into the hex character representing the

=1 239 ; value of A's least significant nibble

=1 240

00D5 540F =1 241 ANL A,#00Fh

00D7 B40A00 =1 242 CJNE A,#00Ah,$+3

00DA 4002 =1 243 JC IO0030

00DC 2407 =1 244 ADD A,#007h

00DE 2430 =1 245 IO0030: ADD A,#'0'

=1 246

00E0 22 =1 247 RET

=1 248

=1 249 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 250 ; ASCII2HEX

=1 251

00E1 =1 252 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 253 ; into the corresponding number (0-15). returns C=1

=1 254 ; when input is other than an ASCII digit,

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

=1 256

00E1 C3 =1 257 CLR C

00E2 9430 =1 258 SUBB A,#'0'

00E4 B40A00 =1 259 CJNE A,#10,$+3

00E7 401B =1 260 JC IO0050 ; if '0'<=char<='9', return OK

00E9 B41100 =1 261 CJNE A,#17,$+3

00EC 4013 =1 262 JC IO0040 ; if '9'<char<'A', return FAIL

00EE 9407 =1 263 SUBB A,#7

00F0 B41000 =1 264 CJNE A,#10h,$+3

00F3 400F =1 265 JC IO0050 ; if 'A'<=char<='F', return OK

00F5 B42A00 =1 266 CJNE A,#42,$+3

00F8 4007 =1 267 JC IO0040 ; if 'F'<char<'a', return FAIL

00FA 9420 =1 268 SUBB A,#20h

00FC B41000 =1 269 CJNE A,#10h,$+3

00FF 4003 =1 270 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 271

0101 C3 =1 272 IO0040: CLR C ; ..else return FAIL

0102 74FF =1 273 MOV A,#0FFh

=1 274

0104 B3 =1 275 IO0050: CPL C

0105 22 =1 276 RET

=1 277

=1 278 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 279 ; GETCHAR

=1 280

0106 =1 281 GETCHAR: ; waits for a single ASCII character to be received

=1 282 ; by the UART. places this character into A.

=1 283

0106 3098FD =1 284 JNB RI,$

0109 E599 =1 285 MOV A,SBUF

010B C298 =1 286 CLR RI

=1 287

010D 22 =1 288 RET

=1 289

=1 290 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

812MSTR PAGE 6

=1 291 ; GETVAL

=1 292

010E =1 293 GETVAL: ; waits for two ASCII hex digits to be received by

=1 294 ; the UART. returns the hex value in A.

=1 295

010E C0F0 =1 296 PUSH B

0110 C000 =1 297 PUSH 0

0112 C298 =1 298 IO0060: CLR RI

0114 3106 =1 299 CALL GETCHAR ; first nibble

0116 F500 =1 300 MOV 0,A ; store received char

0118 11E1 =1 301 CALL ASCII2HEX

011A 40F6 =1 302 JC IO0060 ; if not '0' thru 'F', don't accept

011C C4 =1 303 SWAP A ; swap nibbles

011D F5F0 =1 304 MOV B,A ; store nibble in B

011F E500 =1 305 MOV A,0 ; echo received char

0121 11B9 =1 306 CALL SENDCHAR

0123 C298 =1 307 IO0070: CLR RI

0125 3106 =1 308 CALL GETCHAR ; second nibble

0127 F500 =1 309 MOV 0,A ; store received char

0129 11E1 =1 310 CALL ASCII2HEX

012B 40F6 =1 311 JC IO0070 ; if not '0' thru 'F', don't accept

012D 45F0 =1 312 ORL A,B ; combine nibbles

012F F5F0 =1 313 MOV B,A ; store results in B

0131 E500 =1 314 MOV A,0 ; echo received char

0133 11B9 =1 315 CALL SENDCHAR

0135 E5F0 =1 316 MOV A,B ; final result

0137 D000 =1 317 POP 0

0139 D0F0 =1 318 POP B

=1 319

013B 22 =1 320 RET

=1 321

322

323 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

324 ; TEXT DATA TABLES

325

013C 0A0D00 326 SEPERATOR: DB 10,13,0

327

328 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

329

330 END

331

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

812MSTR PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00E1H

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

DELAY. . . . . . . . . . . . . . C ADDR 008DH

DLY1 . . . . . . . . . . . . . . C ADDR 0093H

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 0106H

GETVAL . . . . . . . . . . . . . C ADDR 010EH NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00D5H

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

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

IO0010 . . . . . . . . . . . . . C ADDR 00A8H

IO0020 . . . . . . . . . . . . . C ADDR 00B4H

IO0030 . . . . . . . . . . . . . C ADDR 00DEH

IO0040 . . . . . . . . . . . . . C ADDR 0101H

IO0050 . . . . . . . . . . . . . C ADDR 0104H

IO0060 . . . . . . . . . . . . . C ADDR 0112H

IO0070 . . . . . . . . . . . . . C ADDR 0123H

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

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

LOOP . . . . . . . . . . . . . . C ADDR 0068H

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 00B9H

SENDSPI. . . . . . . . . . . . . C ADDR 0087H

SENDSTRING . . . . . . . . . . . C ADDR 00A1H

SENDVAL. . . . . . . . . . . . . C ADDR 00C1H

SEPERATOR. . . . . . . . . . . . C ADDR 013CH

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

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

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

SS . . . . . . . . . . . . . . . NUMB 00B5H

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

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

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

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