816MSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : January 2001

6 ;

7 ; File : 816mstr.asm

8 ;

9 ; Hardware : ADuC816

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 ; '816slave.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

00B5 48 SS EQU P3.5 ; P3.5 drives slave device's SS pin

49

50 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

51 ; DEFINE VARIABLES IN INTERNAL RAM

---- 52 DSEG

0060 53 ORG 0060h

0060 54 INPUT: DS 1 ; data byte received by SPI

0061 55 OUTPUT: DS 1 ; data byte to send by SPI

56

57 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

58 ; BEGINNING OF CODE

816MSTR PAGE 2

---- 59 CSEG

60

0000 61 ORG 0000h

0000 02004B 62 JMP MAIN ; jump to main program

63

64 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

65 ; INTERRUPT VECTOR SPACE

0003 66 ORG 0003h ; (.................... INT0 ISR)

67

0003 0561 68 INC OUTPUT

0005 32 69 RETI

70

003B 71 ORG 003Bh ; (.................... SPI ISR)

72

003B D2B5 73 SETB SS ; pull slave's SS pin high

003D 85F760 74 MOV INPUT,SPIDAT

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

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

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 D2AF 108 SETB EA ; enable inturrupts

109

006F B2B4 110 LOOP: CPL LED ; flash the LED on the eval board

0071 E561 111 MOV A,OUTPUT ; byte to send via SPI into ACC

0073 12008E 112 CALL SENDSPI ; trigger SPI send/receive transfer

0076 120094 113 CALL DELAY ; pause 100ms

0079 E560 114 MOV A,INPUT ; send value received by SPI..

007B 1200C0 115 CALL SENDVAL ; ..out the UART as 2 ASCII chars

007E 90013B 116 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

816MSTR PAGE 3

0081 1200A0 117 CALL SENDSTRING ; ..out the UART

0084 3098E8 118 JNB RI,LOOP ; repeat (unless UART data received)

119

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

121

0087 859961 122 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

008A C298 123 CLR RI ; must clear RI

008C 80E1 124 JMP LOOP ; back to main loop

125

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

127 ; SUBROUTINES

128

008E 129 SENDSPI: ; sends the value in ACC out the SPI port. also

130 ; receives simultaneously into SPIDAT. SPI interrupt

131 ; is triggered when transfer is complete.

132

008E C2B5 133 CLR SS ; must pull slave's SS pin low first

0090 8561F7 134 MOV SPIDAT,OUTPUT ; trigger data transfer

0093 22 135 RET

136

137 ;--------------------------------------------------------------------

0094 138 DELAY: ; Delays by 13,000 \* A instructions (approx)

139 ; 100mSec based on 1.5728MHZ Core Clock (default)

140

0094 F8 141 MOV R0,A ; Acc holds delay variable

0095 7941 142 DLY0: MOV R1,#65 ; Set up delay loop0

0097 7AC8 143 DLY1: MOV R2,#200 ; Set up delay loop1

0099 DAFE 144 DJNZ R2,$ ; Dec R2 200 times

009B D9FA 145 DJNZ R1,DLY1 ; Dec R1 65 times

009D D8F6 146 DJNZ R0,DLY0 ; Dec R0 #ACC times

009F 22 147 RET ; Return from subroutine

148

149 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

150 ; SUBROUTINE INCLUDE FILE

151

=1 152 $INCLUDE(UARTIO.asm)

=1 153 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 154 ;

=1 155 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 156 ;

=1 157 ; Date : January 2001

=1 158 ;

=1 159 ; File : UARTIO.asm

=1 160 ;

=1 161 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 162 ;

=1 163 ; Description : standard UART I/O subroutines. total size of this

=1 164 ; code when assembled is 155 bytes. routines for use

=1 165 ; external to this file are:

=1 166 ;

=1 167 ; SENDSTRING - sends a string of characters

=1 168 ; SENDCHAR - sends a single character

=1 169 ; SENDVAL - sends a byte as 2 ASCII characters

=1 170 ; HEX2ASCII - converts from HEX to ASCII

=1 171 ; ASCII2HEX - converts from ASCII to HEX

=1 172 ; GETCHAR - gets a single character

=1 173 ; GETVAL - gets a byte as 2 ASCII characters

=1 174 ;

816MSTR PAGE 4

=1 175 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 176

=1 177 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 178 ; SENDSTRING

=1 179

00A0 =1 180 SENDSTRING: ; sends ASCII string to UART starting at location

=1 181 ; DPTR and ending with a null (0) value

=1 182

00A0 C0E0 =1 183 PUSH ACC

00A2 C0F0 =1 184 PUSH B

00A4 E4 =1 185 CLR A

00A5 F5F0 =1 186 MOV B,A

00A7 E5F0 =1 187 IO0010: MOV A,B

00A9 05F0 =1 188 INC B

00AB 93 =1 189 MOVC A,@A+DPTR

00AC 6005 =1 190 JZ IO0020

00AE 1200B8 =1 191 CALL SENDCHAR

00B1 80F4 =1 192 JMP IO0010

00B3 D0F0 =1 193 IO0020: POP B

00B5 D0E0 =1 194 POP ACC

=1 195

00B7 22 =1 196 RET

=1 197

=1 198 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 199 ; SENDCHAR

=1 200

00B8 =1 201 SENDCHAR: ; sends ASCII value contained in A to UART

=1 202

00B8 3099FD =1 203 JNB TI,$ ; wait til present char gone

00BB C299 =1 204 CLR TI ; must clear TI

00BD F599 =1 205 MOV SBUF,A

=1 206

00BF 22 =1 207 RET

=1 208

=1 209 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 210 ; SENDVAL

=1 211

00C0 =1 212 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 213 ; and then spits these two characters up the UART.

=1 214 ; does not change the value of A.

=1 215

00C0 C0E0 =1 216 PUSH ACC

00C2 C4 =1 217 SWAP A

00C3 1200D4 =1 218 CALL HEX2ASCII

00C6 11B8 =1 219 CALL SENDCHAR ; send high nibble

00C8 D0E0 =1 220 POP ACC

00CA C0E0 =1 221 PUSH ACC

00CC 1200D4 =1 222 CALL HEX2ASCII

00CF 11B8 =1 223 CALL SENDCHAR ; send low nibble

00D1 D0E0 =1 224 POP ACC

=1 225

00D3 22 =1 226 RET

=1 227

=1 228 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 229 ; HEX2ASCII

=1 230

00D4 =1 231 HEX2ASCII: ; converts A into the hex character representing the

=1 232 ; value of A's least significant nibble

816MSTR PAGE 5

=1 233

00D4 540F =1 234 ANL A,#00Fh

00D6 B40A00 =1 235 CJNE A,#00Ah,$+3

00D9 4002 =1 236 JC IO0030

00DB 2407 =1 237 ADD A,#007h

00DD 2430 =1 238 IO0030: ADD A,#'0'

=1 239

00DF 22 =1 240 RET

=1 241

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

=1 243 ; ASCII2HEX

=1 244

00E0 =1 245 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 246 ; into the corresponding number (0-15). returns C=1

=1 247 ; when input is other than an ASCII digit,

=1 248 ; indicating invalid output (returned as 255).

=1 249

00E0 C3 =1 250 CLR C

00E1 9430 =1 251 SUBB A,#'0'

00E3 B40A00 =1 252 CJNE A,#10,$+3

00E6 401B =1 253 JC IO0050 ; if '0'<=char<='9', return OK

00E8 B41100 =1 254 CJNE A,#17,$+3

00EB 4013 =1 255 JC IO0040 ; if '9'<char<'A', return FAIL

00ED 9407 =1 256 SUBB A,#7

00EF B41000 =1 257 CJNE A,#10h,$+3

00F2 400F =1 258 JC IO0050 ; if 'A'<=char<='F', return OK

00F4 B42A00 =1 259 CJNE A,#42,$+3

00F7 4007 =1 260 JC IO0040 ; if 'F'<char<'a', return FAIL

00F9 9420 =1 261 SUBB A,#20h

00FB B41000 =1 262 CJNE A,#10h,$+3

00FE 4003 =1 263 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 264

0100 C3 =1 265 IO0040: CLR C ; ..else return FAIL

0101 74FF =1 266 MOV A,#0FFh

=1 267

0103 B3 =1 268 IO0050: CPL C

0104 22 =1 269 RET

=1 270

=1 271 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 272 ; GETCHAR

=1 273

0105 =1 274 GETCHAR: ; waits for a single ASCII character to be received

=1 275 ; by the UART. places this character into A.

=1 276

0105 3098FD =1 277 JNB RI,$

0108 E599 =1 278 MOV A,SBUF

010A C298 =1 279 CLR RI

=1 280

010C 22 =1 281 RET

=1 282

=1 283 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 284 ; GETVAL

=1 285

010D =1 286 GETVAL: ; waits for two ASCII hex digits to be received by

=1 287 ; the UART. returns the hex value in A.

=1 288

010D C0F0 =1 289 PUSH B

010F C000 =1 290 PUSH 0

816MSTR PAGE 6

0111 C298 =1 291 IO0060: CLR RI

0113 3105 =1 292 CALL GETCHAR ; first nibble

0115 F500 =1 293 MOV 0,A ; store received char

0117 11E0 =1 294 CALL ASCII2HEX

0119 40F6 =1 295 JC IO0060 ; if not '0' thru 'F', don't accept

011B C4 =1 296 SWAP A ; swap nibbles

011C F5F0 =1 297 MOV B,A ; store nibble in B

011E E500 =1 298 MOV A,0 ; echo received char

0120 11B8 =1 299 CALL SENDCHAR

0122 C298 =1 300 IO0070: CLR RI

0124 3105 =1 301 CALL GETCHAR ; second nibble

0126 F500 =1 302 MOV 0,A ; store received char

0128 11E0 =1 303 CALL ASCII2HEX

012A 40F6 =1 304 JC IO0070 ; if not '0' thru 'F', don't accept

012C 45F0 =1 305 ORL A,B ; combine nibbles

012E F5F0 =1 306 MOV B,A ; store results in B

0130 E500 =1 307 MOV A,0 ; echo received char

0132 11B8 =1 308 CALL SENDCHAR

0134 E5F0 =1 309 MOV A,B ; final result

0136 D000 =1 310 POP 0

0138 D0F0 =1 311 POP B

=1 312

013A 22 =1 313 RET

=1 314

315

316 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

317 ; TEXT DATA TABLES

318

013B 0A0D00 319 SEPERATOR: DB 10,13,0

320

321 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

322

323 END

324

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

816MSTR PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00E0H

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

DELAY. . . . . . . . . . . . . . C ADDR 0094H

DLY0 . . . . . . . . . . . . . . C ADDR 0095H

DLY1 . . . . . . . . . . . . . . C ADDR 0097H

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 0105H

GETVAL . . . . . . . . . . . . . C ADDR 010DH NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00D4H

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

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

IO0010 . . . . . . . . . . . . . C ADDR 00A7H

IO0020 . . . . . . . . . . . . . C ADDR 00B3H

IO0030 . . . . . . . . . . . . . C ADDR 00DDH

IO0040 . . . . . . . . . . . . . C ADDR 0100H

IO0050 . . . . . . . . . . . . . C ADDR 0103H

IO0060 . . . . . . . . . . . . . C ADDR 0111H

IO0070 . . . . . . . . . . . . . C ADDR 0122H

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

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

LOOP . . . . . . . . . . . . . . C ADDR 006FH

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 00B8H

SENDSPI. . . . . . . . . . . . . C ADDR 008EH

SENDSTRING . . . . . . . . . . . C ADDR 00A0H

SENDVAL. . . . . . . . . . . . . C ADDR 00C0H

SEPERATOR. . . . . . . . . . . . C ADDR 013BH

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

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

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

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

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

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

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

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