842MSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : October 2003

6 ;

7 ; File : 832mstr.asm

8 ;

9 ; Hardware : ADuC842/ADuC843

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 ; '832slave.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/I2C 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

00B5 46 SS EQU P3.5 ; P3.5 drives slave device's SS pin

47

48 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

49 ; DEFINE VARIABLES IN INTERNAL RAM

---- 50 DSEG

0060 51 ORG 0060h

0060 52 INPUT: DS 1 ; data byte received by SPI

0061 53 OUTPUT: DS 1 ; data byte to send by SPI

54

55 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

56 ; BEGINNING OF CODE

---- 57 CSEG

58

842MSTR PAGE 2

0000 59 ORG 0000h

0000 02004B 60 JMP MAIN ; jump to main program

61

62 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

63 ; INTERRUPT VECTOR SPACE

0003 64 ORG 0003h ; (.................... INT0 ISR)

65

0003 0561 66 INC OUTPUT

0005 32 67 RETI

68

003B 69 ORG 003Bh ; (.................... SPI ISR)

70

003B D2B5 71 SETB SS ; pull slave's SS pin high

003D 85F760 72 MOV INPUT,SPIDAT

0040 32 73 RETI

74

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

76 ; MAIN PROGRAM

004B 77 ORG 004Bh

78

004B 79 MAIN:

80

004B 758107 81 MOV SP,#007h

82

83 ; CONFIGURE UART...

004E 759E83 84 MOV T3CON,#083h

0051 759D2D 85 MOV T3FD,#02Dh

0054 759852 86 MOV SCON,#052h

87

88 ; CONFIGURE SPI...

89

0057 75F837 90 MOV SPICON,#037h ; configure SPI port for:

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

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

93

94 ; CONFIGURE INTERRUPT 0...

95

005D D288 96 SETB IT0 ; INT0 edge triggered

005F D2A8 97 SETB EX0 ; enable INT0 interrupt

98

99 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

100

0061 756100 101 MOV OUTPUT,#0 ; set initial value for output byte

0064 D2AF 102 SETB EA ; enable inturrupts

103

0066 B2B4 104 LOOP: CPL LED ; flash the LED on the eval board

0068 E561 105 MOV A,OUTPUT ; byte to send via SPI into ACC

006A 12009D 106 CALL SENDSPI ; trigger SPI send/receive transfer

006D 1200A3 107 CALL DELAY ; pause 10ms

0070 1200A3 108 CALL DELAY ; pause 10ms

0073 1200A3 109 CALL DELAY ; pause 10ms

0076 1200A3 110 CALL DELAY ; pause 10ms

0079 1200A3 111 CALL DELAY ; pause 10ms

007C 1200A3 112 CALL DELAY ; pause 10ms

007F 1200A3 113 CALL DELAY ; pause 10ms

0082 1200A3 114 CALL DELAY ; pause 10ms

0085 1200A3 115 CALL DELAY ; pause 10ms

0088 E560 116 MOV A,INPUT ; send value received by SPI..

842MSTR PAGE 3

008A 1200D7 117 CALL SENDVAL ; ..out the UART as 2 ASCII chars

008D 900152 118 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

0090 1200B7 119 CALL SENDSTRING ; ..out the UART

0093 3098D0 120 JNB RI,LOOP ; repeat (unless UART data received)

121

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

123

0096 859961 124 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

0099 C298 125 CLR RI ; must clear RI

009B 80C9 126 JMP LOOP ; back to main loop

127

128 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

129 ; SUBROUTINES

130

009D 131 SENDSPI: ; sends the value in ACC out the SPI port. also

132 ; receives simultaneously into SPIDAT. SPI interrupt

133 ; is triggered when transfer is complete.

134

009D C2B5 135 CLR SS ; must pull slave's SS pin low first

009F 8561F7 136 MOV SPIDAT,OUTPUT ; trigger data transfer

00A2 22 137 RET

138

139 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

140

00A3 141 DELAY: ; delays approximately 10ms

142

00A3 C0E0 143 PUSH ACC

00A5 C0F0 144 PUSH B

00A7 741B 145 MOV A,#01Bh ;

00A9 75F0FF 146 DLY1: MOV B,#0FFh ;

00AC D5F0FD 147 DJNZ B,$ ;

00AF D5E0F7 148 DJNZ ACC,DLY1 ;

00B2 D0F0 149 POP B

00B4 D0E0 150 POP ACC

00B6 22 151 RET

152

153 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

154 ; SUBROUTINE INCLUDE FILE

155

=1 156 $INCLUDE(UARTIO.asm)

=1 157 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 158 ;

=1 159 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 160 ;

=1 161 ; Date : January 2001

=1 162 ;

=1 163 ; File : UARTIO.asm

=1 164 ;

=1 165 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 166 ;

=1 167 ; Description : standard UART I/O subroutines. total size of this

=1 168 ; code when assembled is 155 bytes. routines for use

=1 169 ; external to this file are:

=1 170 ;

=1 171 ; SENDSTRING - sends a string of characters

=1 172 ; SENDCHAR - sends a single character

=1 173 ; SENDVAL - sends a byte as 2 ASCII characters

=1 174 ; HEX2ASCII - converts from HEX to ASCII

842MSTR PAGE 4

=1 175 ; ASCII2HEX - converts from ASCII to HEX

=1 176 ; GETCHAR - gets a single character

=1 177 ; GETVAL - gets a byte as 2 ASCII characters

=1 178 ;

=1 179 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 180

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

=1 182 ; SENDSTRING

=1 183

00B7 =1 184 SENDSTRING: ; sends ASCII string to UART starting at location

=1 185 ; DPTR and ending with a null (0) value

=1 186

00B7 C0E0 =1 187 PUSH ACC

00B9 C0F0 =1 188 PUSH B

00BB E4 =1 189 CLR A

00BC F5F0 =1 190 MOV B,A

00BE E5F0 =1 191 IO0010: MOV A,B

00C0 05F0 =1 192 INC B

00C2 93 =1 193 MOVC A,@A+DPTR

00C3 6005 =1 194 JZ IO0020

00C5 1200CF =1 195 CALL SENDCHAR

00C8 80F4 =1 196 JMP IO0010

00CA D0F0 =1 197 IO0020: POP B

00CC D0E0 =1 198 POP ACC

=1 199

00CE 22 =1 200 RET

=1 201

=1 202 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 203 ; SENDCHAR

=1 204

00CF =1 205 SENDCHAR: ; sends ASCII value contained in A to UART

=1 206

00CF 3099FD =1 207 JNB TI,$ ; wait til present char gone

00D2 C299 =1 208 CLR TI ; must clear TI

00D4 F599 =1 209 MOV SBUF,A

=1 210

00D6 22 =1 211 RET

=1 212

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

=1 214 ; SENDVAL

=1 215

00D7 =1 216 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 217 ; and then spits these two characters up the UART.

=1 218 ; does not change the value of A.

=1 219

00D7 C0E0 =1 220 PUSH ACC

00D9 C4 =1 221 SWAP A

00DA 1200EB =1 222 CALL HEX2ASCII

00DD 11CF =1 223 CALL SENDCHAR ; send high nibble

00DF D0E0 =1 224 POP ACC

00E1 C0E0 =1 225 PUSH ACC

00E3 1200EB =1 226 CALL HEX2ASCII

00E6 11CF =1 227 CALL SENDCHAR ; send low nibble

00E8 D0E0 =1 228 POP ACC

=1 229

00EA 22 =1 230 RET

=1 231

=1 232 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

842MSTR PAGE 5

=1 233 ; HEX2ASCII

=1 234

00EB =1 235 HEX2ASCII: ; converts A into the hex character representing the

=1 236 ; value of A's least significant nibble

=1 237

00EB 540F =1 238 ANL A,#00Fh

00ED B40A00 =1 239 CJNE A,#00Ah,$+3

00F0 4002 =1 240 JC IO0030

00F2 2407 =1 241 ADD A,#007h

00F4 2430 =1 242 IO0030: ADD A,#'0'

=1 243

00F6 22 =1 244 RET

=1 245

=1 246 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 247 ; ASCII2HEX

=1 248

00F7 =1 249 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 250 ; into the corresponding number (0-15). returns C=1

=1 251 ; when input is other than an ASCII digit,

=1 252 ; indicating invalid output (returned as 255).

=1 253

00F7 C3 =1 254 CLR C

00F8 9430 =1 255 SUBB A,#'0'

00FA B40A00 =1 256 CJNE A,#10,$+3

00FD 401B =1 257 JC IO0050 ; if '0'<=char<='9', return OK

00FF B41100 =1 258 CJNE A,#17,$+3

0102 4013 =1 259 JC IO0040 ; if '9'<char<'A', return FAIL

0104 9407 =1 260 SUBB A,#7

0106 B41000 =1 261 CJNE A,#10h,$+3

0109 400F =1 262 JC IO0050 ; if 'A'<=char<='F', return OK

010B B42A00 =1 263 CJNE A,#42,$+3

010E 4007 =1 264 JC IO0040 ; if 'F'<char<'a', return FAIL

0110 9420 =1 265 SUBB A,#20h

0112 B41000 =1 266 CJNE A,#10h,$+3

0115 4003 =1 267 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 268

0117 C3 =1 269 IO0040: CLR C ; ..else return FAIL

0118 74FF =1 270 MOV A,#0FFh

=1 271

011A B3 =1 272 IO0050: CPL C

011B 22 =1 273 RET

=1 274

=1 275 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 276 ; GETCHAR

=1 277

011C =1 278 GETCHAR: ; waits for a single ASCII character to be received

=1 279 ; by the UART. places this character into A.

=1 280

011C 3098FD =1 281 JNB RI,$

011F E599 =1 282 MOV A,SBUF

0121 C298 =1 283 CLR RI

=1 284

0123 22 =1 285 RET

=1 286

=1 287 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 288 ; GETVAL

=1 289

0124 =1 290 GETVAL: ; waits for two ASCII hex digits to be received by

842MSTR PAGE 6

=1 291 ; the UART. returns the hex value in A.

=1 292

0124 C0F0 =1 293 PUSH B

0126 C000 =1 294 PUSH 0

0128 C298 =1 295 IO0060: CLR RI

012A 311C =1 296 CALL GETCHAR ; first nibble

012C F500 =1 297 MOV 0,A ; store received char

012E 11F7 =1 298 CALL ASCII2HEX

0130 40F6 =1 299 JC IO0060 ; if not '0' thru 'F', don't accept

0132 C4 =1 300 SWAP A ; swap nibbles

0133 F5F0 =1 301 MOV B,A ; store nibble in B

0135 E500 =1 302 MOV A,0 ; echo received char

0137 11CF =1 303 CALL SENDCHAR

0139 C298 =1 304 IO0070: CLR RI

013B 311C =1 305 CALL GETCHAR ; second nibble

013D F500 =1 306 MOV 0,A ; store received char

013F 11F7 =1 307 CALL ASCII2HEX

0141 40F6 =1 308 JC IO0070 ; if not '0' thru 'F', don't accept

0143 45F0 =1 309 ORL A,B ; combine nibbles

0145 F5F0 =1 310 MOV B,A ; store results in B

0147 E500 =1 311 MOV A,0 ; echo received char

0149 11CF =1 312 CALL SENDCHAR

014B E5F0 =1 313 MOV A,B ; final result

014D D000 =1 314 POP 0

014F D0F0 =1 315 POP B

=1 316

0151 22 =1 317 RET

=1 318

319

320 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

321 ; TEXT DATA TABLES

322

0152 0A0D00 323 SEPERATOR: DB 10,13,0

324

325 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

326

327 END

328

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

842MSTR PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00F7H

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

DELAY. . . . . . . . . . . . . . C ADDR 00A3H

DLY1 . . . . . . . . . . . . . . C ADDR 00A9H

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 011CH

GETVAL . . . . . . . . . . . . . C ADDR 0124H NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00EBH

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

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

IO0010 . . . . . . . . . . . . . C ADDR 00BEH

IO0020 . . . . . . . . . . . . . C ADDR 00CAH

IO0030 . . . . . . . . . . . . . C ADDR 00F4H

IO0040 . . . . . . . . . . . . . C ADDR 0117H

IO0050 . . . . . . . . . . . . . C ADDR 011AH

IO0060 . . . . . . . . . . . . . C ADDR 0128H

IO0070 . . . . . . . . . . . . . C ADDR 0139H

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

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

LOOP . . . . . . . . . . . . . . C ADDR 0066H

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 00CFH

SENDSPI. . . . . . . . . . . . . C ADDR 009DH

SENDSTRING . . . . . . . . . . . C ADDR 00B7H

SENDVAL. . . . . . . . . . . . . C ADDR 00D7H

SEPERATOR. . . . . . . . . . . . C ADDR 0152H

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

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

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

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

T3CON. . . . . . . . . . . . . . D ADDR 009EH PREDEFINED

T3FD . . . . . . . . . . . . . . D ADDR 009DH PREDEFINED

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