836MSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : November 2001

6 ;

7 ; File : 834mstr.asm

8 ;

9 ; Hardware : ADuC834/ADuC824/ADuC836/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 ; '834slave.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 ADuC834 eval board, you can

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

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

23 ; also ensure that LK10 ('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 ('836mstr' to the master,

28 ; '836slave' 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 $MOD836 ; Use 8052 & ADuC836 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

836MSTR 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 759E82 87 MOV T3CON,#82h

0051 759D12 88 MOV T3FD,#12h

0054 759852 89 MOV SCON,#52h

90

91 ; CONFIGURE SPI...

92

0057 75F837 93 MOV SPICON,#037h ; configure SPI port for:

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

005A 75A901 95 MOV IEIP2,#1 ; enable SPI interrupt

96

97 ; CONFIGURE INTERRUPT 0...

98

005D D288 99 SETB IT0 ; INT0 edge triggered

005F D2A8 100 SETB EX0 ; enable INT0 interrupt

101

102 ; ENABLE INTERRUPTS & ENTER MAIN LOOP...

103

0061 756100 104 MOV OUTPUT,#0 ; set initial value for output byte

0064 D2AF 105 SETB EA ; enable inturrupts

106

0066 B2B4 107 LOOP: CPL LED ; flash the LED on the eval board

0068 E561 108 MOV A,OUTPUT ; byte to send via SPI into ACC

006A 120085 109 CALL SENDSPI ; trigger SPI send/receive transfer

006D 12008B 110 CALL DELAY ; pause 100ms

0070 E560 111 MOV A,INPUT ; send value received by SPI..

0072 1200BF 112 CALL SENDVAL ; ..out the UART as 2 ASCII chars

0075 90013A 113 MOV DPTR,#SEPERATOR ; send line-feed & crdg-return..

0078 12009F 114 CALL SENDSTRING ; ..out the UART

007B 3098E8 115 JNB RI,LOOP ; repeat (unless UART data received)

116

836MSTR PAGE 3

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

118

007E 859961 119 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

0081 C298 120 CLR RI ; must clear RI

0083 80E1 121 JMP LOOP ; back to main loop

122

123 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

124 ; SUBROUTINES

125

0085 126 SENDSPI: ; sends the value in ACC out the SPI port. also

127 ; receives simultaneously into SPIDAT. SPI interrupt

128 ; is triggered when transfer is complete.

129

0085 C2B5 130 CLR SS ; must pull slave's SS pin low first

0087 8561F7 131 MOV SPIDAT,OUTPUT ; trigger data transfer

008A 22 132 RET

133

134 ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

135

008B 136 DELAY: ; delays approximately 100ms

137

008B C0E0 138 PUSH ACC

008D C0F0 139 PUSH B

008F 74C8 140 MOV A,#200 ; 100 \* 1ms = 100ms

0091 75F041 141 DLY1: MOV B,#65 ; 65 \* 15.26us = 1ms

0094 D5F0FD 142 DJNZ B,$ ; sit here for 1ms

0097 D5E0F7 143 DJNZ ACC,DLY1 ; repeat 100 times (100ms delay)

009A D0F0 144 POP B

009C D0E0 145 POP ACC

009E 22 146 RET

147

148 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

149 ; SUBROUTINE INCLUDE FILE

150

=1 151 $INCLUDE(UARTIO.asm)

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

=1 153 ;

=1 154 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 155 ;

=1 156 ; Date : January 2001

=1 157 ;

=1 158 ; File : UARTIO.asm

=1 159 ;

=1 160 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 161 ;

=1 162 ; Description : standard UART I/O subroutines. total size of this

=1 163 ; code when assembled is 155 bytes. routines for use

=1 164 ; external to this file are:

=1 165 ;

=1 166 ; SENDSTRING - sends a string of characters

=1 167 ; SENDCHAR - sends a single character

=1 168 ; SENDVAL - sends a byte as 2 ASCII characters

=1 169 ; HEX2ASCII - converts from HEX to ASCII

=1 170 ; ASCII2HEX - converts from ASCII to HEX

=1 171 ; GETCHAR - gets a single character

=1 172 ; GETVAL - gets a byte as 2 ASCII characters

=1 173 ;

=1 174 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

836MSTR PAGE 4

=1 175

=1 176 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 177 ; SENDSTRING

=1 178

009F =1 179 SENDSTRING: ; sends ASCII string to UART starting at location

=1 180 ; DPTR and ending with a null (0) value

=1 181

009F C0E0 =1 182 PUSH ACC

00A1 C0F0 =1 183 PUSH B

00A3 E4 =1 184 CLR A

00A4 F5F0 =1 185 MOV B,A

00A6 E5F0 =1 186 IO0010: MOV A,B

00A8 05F0 =1 187 INC B

00AA 93 =1 188 MOVC A,@A+DPTR

00AB 6005 =1 189 JZ IO0020

00AD 1200B7 =1 190 CALL SENDCHAR

00B0 80F4 =1 191 JMP IO0010

00B2 D0F0 =1 192 IO0020: POP B

00B4 D0E0 =1 193 POP ACC

=1 194

00B6 22 =1 195 RET

=1 196

=1 197 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 198 ; SENDCHAR

=1 199

00B7 =1 200 SENDCHAR: ; sends ASCII value contained in A to UART

=1 201

00B7 3099FD =1 202 JNB TI,$ ; wait til present char gone

00BA C299 =1 203 CLR TI ; must clear TI

00BC F599 =1 204 MOV SBUF,A

=1 205

00BE 22 =1 206 RET

=1 207

=1 208 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 209 ; SENDVAL

=1 210

00BF =1 211 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 212 ; and then spits these two characters up the UART.

=1 213 ; does not change the value of A.

=1 214

00BF C0E0 =1 215 PUSH ACC

00C1 C4 =1 216 SWAP A

00C2 1200D3 =1 217 CALL HEX2ASCII

00C5 11B7 =1 218 CALL SENDCHAR ; send high nibble

00C7 D0E0 =1 219 POP ACC

00C9 C0E0 =1 220 PUSH ACC

00CB 1200D3 =1 221 CALL HEX2ASCII

00CE 11B7 =1 222 CALL SENDCHAR ; send low nibble

00D0 D0E0 =1 223 POP ACC

=1 224

00D2 22 =1 225 RET

=1 226

=1 227 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 228 ; HEX2ASCII

=1 229

00D3 =1 230 HEX2ASCII: ; converts A into the hex character representing the

=1 231 ; value of A's least significant nibble

=1 232

836MSTR PAGE 5

00D3 540F =1 233 ANL A,#00Fh

00D5 B40A00 =1 234 CJNE A,#00Ah,$+3

00D8 4002 =1 235 JC IO0030

00DA 2407 =1 236 ADD A,#007h

00DC 2430 =1 237 IO0030: ADD A,#'0'

=1 238

00DE 22 =1 239 RET

=1 240

=1 241 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 242 ; ASCII2HEX

=1 243

00DF =1 244 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 245 ; into the corresponding number (0-15). returns C=1

=1 246 ; when input is other than an ASCII digit,

=1 247 ; indicating invalid output (returned as 255).

=1 248

00DF C3 =1 249 CLR C

00E0 9430 =1 250 SUBB A,#'0'

00E2 B40A00 =1 251 CJNE A,#10,$+3

00E5 401B =1 252 JC IO0050 ; if '0'<=char<='9', return OK

00E7 B41100 =1 253 CJNE A,#17,$+3

00EA 4013 =1 254 JC IO0040 ; if '9'<char<'A', return FAIL

00EC 9407 =1 255 SUBB A,#7

00EE B41000 =1 256 CJNE A,#10h,$+3

00F1 400F =1 257 JC IO0050 ; if 'A'<=char<='F', return OK

00F3 B42A00 =1 258 CJNE A,#42,$+3

00F6 4007 =1 259 JC IO0040 ; if 'F'<char<'a', return FAIL

00F8 9420 =1 260 SUBB A,#20h

00FA B41000 =1 261 CJNE A,#10h,$+3

00FD 4003 =1 262 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 263

00FF C3 =1 264 IO0040: CLR C ; ..else return FAIL

0100 74FF =1 265 MOV A,#0FFh

=1 266

0102 B3 =1 267 IO0050: CPL C

0103 22 =1 268 RET

=1 269

=1 270 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 271 ; GETCHAR

=1 272

0104 =1 273 GETCHAR: ; waits for a single ASCII character to be received

=1 274 ; by the UART. places this character into A.

=1 275

0104 3098FD =1 276 JNB RI,$

0107 E599 =1 277 MOV A,SBUF

0109 C298 =1 278 CLR RI

=1 279

010B 22 =1 280 RET

=1 281

=1 282 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 283 ; GETVAL

=1 284

010C =1 285 GETVAL: ; waits for two ASCII hex digits to be received by

=1 286 ; the UART. returns the hex value in A.

=1 287

010C C0F0 =1 288 PUSH B

010E C000 =1 289 PUSH 0

0110 C298 =1 290 IO0060: CLR RI

836MSTR PAGE 6

0112 3104 =1 291 CALL GETCHAR ; first nibble

0114 F500 =1 292 MOV 0,A ; store received char

0116 11DF =1 293 CALL ASCII2HEX

0118 40F6 =1 294 JC IO0060 ; if not '0' thru 'F', don't accept

011A C4 =1 295 SWAP A ; swap nibbles

011B F5F0 =1 296 MOV B,A ; store nibble in B

011D E500 =1 297 MOV A,0 ; echo received char

011F 11B7 =1 298 CALL SENDCHAR

0121 C298 =1 299 IO0070: CLR RI

0123 3104 =1 300 CALL GETCHAR ; second nibble

0125 F500 =1 301 MOV 0,A ; store received char

0127 11DF =1 302 CALL ASCII2HEX

0129 40F6 =1 303 JC IO0070 ; if not '0' thru 'F', don't accept

012B 45F0 =1 304 ORL A,B ; combine nibbles

012D F5F0 =1 305 MOV B,A ; store results in B

012F E500 =1 306 MOV A,0 ; echo received char

0131 11B7 =1 307 CALL SENDCHAR

0133 E5F0 =1 308 MOV A,B ; final result

0135 D000 =1 309 POP 0

0137 D0F0 =1 310 POP B

=1 311

0139 22 =1 312 RET

=1 313

314

315 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

316 ; TEXT DATA TABLES

317

013A 0A0D00 318 SEPERATOR: DB 10,13,0

319

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

321

322 END

323

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

836MSTR PAGE 7

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

ASCII2HEX. . . . . . . . . . . . C ADDR 00DFH

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

DELAY. . . . . . . . . . . . . . C ADDR 008BH

DLY1 . . . . . . . . . . . . . . C ADDR 0091H

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

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

GETCHAR. . . . . . . . . . . . . C ADDR 0104H

GETVAL . . . . . . . . . . . . . C ADDR 010CH NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00D3H

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

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

IO0010 . . . . . . . . . . . . . C ADDR 00A6H

IO0020 . . . . . . . . . . . . . C ADDR 00B2H

IO0030 . . . . . . . . . . . . . C ADDR 00DCH

IO0040 . . . . . . . . . . . . . C ADDR 00FFH

IO0050 . . . . . . . . . . . . . C ADDR 0102H

IO0060 . . . . . . . . . . . . . C ADDR 0110H

IO0070 . . . . . . . . . . . . . C ADDR 0121H

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 00B7H

SENDSPI. . . . . . . . . . . . . C ADDR 0085H

SENDSTRING . . . . . . . . . . . C ADDR 009FH

SENDVAL. . . . . . . . . . . . . C ADDR 00BFH

SEPERATOR. . . . . . . . . . . . C ADDR 013AH

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