SLAVUART PAGE 1

1 ;====================================================================

2 ;

3 ; Author : ADI - Apps

4 ;

5 ; Date : October 2003

6 ;

7 ; File : SLAVuart.asm

8 ;

9 ; Hardware : ADuC842

10 ;

11 ; Description : This slave program transmits the numbers 11-20 in

12 ; binary form continuously down the SPI serial port

13 ; after receiving a clock signal.

14 ;

15 ; After the transmission of each byte the incoming

16 ; byte is saved in order at an internal RAM address

17 ; between #40h and #50h.

18 ;

19 ; This program can be used with the master program

20 ; MASTuart.asm (which generates a clock signal for

21 ; the slave)

22 ;

23 ; After the 16 input bytes have been stored in memory

24 ; the values in memory are outputted up the UART to

25 ; the PC where they can be viewed on screen by a

26 ; program such as Hyperterminal. After each

27 ; transmission up the UART the program is delayed for

28 ; 1s before returning from the interrupt. It then

29 ; waits for the next data byte from the SPI port

30 ; which will arrive about 4s later if used with the

31 ; Master program (MASTuart.asm).

32 ;

33 ; The Slave program (SLAVuart.asm) should be started

34 ; after the master program (MASTuart.asm) but within

35 ; the time delay of 2s in order that the slave

36 ; program is synchronised by the first outputted

37 ; clock of the master.

38 ;

39 ; The clock is inputted at sclock (pin 26)

40 ; The data is outputted at MISO (pin 14)

41 ; The data is inputted at sdata/MOSI (pin 27)

42 ;====================================================================

43 ;

44 $MOD842 ;Use 8052 predefined Symbols

45

00B4 46 LED EQU P3.4

0000 47 FLAG BIT 00H

48

49 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

50 ; BEGINNING OF CODE

---- 51 CSEG

0000 52 ORG 0000H

53

0000 020060 54 JMP MAIN

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

56 ; SPI INTERRUPT ROUTINE

003B 57 ORG 003BH

003B C200 58 CLR FLAG ; Clear flag to leave LOOP2

SLAVUART PAGE 2

59

003D A7F7 60 MOV @R1, SPIDAT ; move input into memory

003F 09 61 INC R1 ; increment memory location so new

62 ; data is stored in new address

63

0040 B95003 64 CJNE R1, #50H, CONT ; reset memory location to 40h when

65 ; memory location reaches 50h saving

66 ; 16 bytes of data

67

0043 120088 68 CALL SNDUART

69

0046 32 70 CONT: RETI

71

72

73 ;====================================================================

74

0060 75 ORG 0060H ; Start code at address above interrupts

76

0060 77 MAIN: ; Main program

0060 75D703 78 MOV PLLCON,#03H

0063 759E83 79 MOV T3CON,#083h

0066 759D2D 80 MOV T3FD,#02Dh

0069 759852 81 MOV SCON,#052h

82

006C 75F824 83 MOV SPICON,#24h ; Initialise SPICON to have

84 ; -Enable SPI serial port

85 ; -slave mode select

86 ; -CPOL=0, clk idling low

87 ; -CPHA=1

88 ; note: it is important to have CPHA in the master and the slave

89 ; program equal, otherwise uncertainty will exist, as the input

90 ; will be measued during its change of state, and not is at

91 ; its final value.

92

006F 75A901 93 MOV IEIP2, #01h ; Enable SPI interrupt

0072 D2AF 94 SETB EA ; Enable interrupts

95

0074 7940 96 MOV R1, #40h ; initialise R1 to 40 to store the

97 ; input data from memory location 40

0076 780A 98 MOV R0, #0AH ; initialise R0 to 10

99

0078 100 TRNSMT:

0078 08 101 INC R0

0079 88F7 102 MOV SPIDAT, R0 ; transmit the current value on R0

007B D200 103 SETB FLAG ; set flag so that we wait here until

104 ; the spi interrupt routine clears

105 ; the FLAG

106

007D 2000FD 107 JB FLAG, $ ; stay here until flag is cleared

108 ; by interrupt

109

110 ; check if R0 is equal to 20. If so the number 20 has been

111 ; transmitted and we should reset R0 to 10 to start transmission

112 ; from 11 again.

0080 E8 113 MOV A, R0

0081 B414F4 114 CJNE A, #14H, TRNSMT ; if R0 is not 20, jump to TRNSMT

0084 780A 115 MOV R0, #0AH ; if R0=20 make R0=10 & jump to TRNSMT

0086 80F0 116 JMP TRNSMT

SLAVUART PAGE 3

117

118

119 ; Transmit the values in locations 40h->50h up the UART wait for

120 ; 1 seconds and then transmit and receive values to/from the Master

121 ; again down the SPI port.

122

0088 123 SNDUART:

0088 B2B4 124 CPL LED ;CPL LED with each transmission

008A 900100 125 MOV DPTR, #TITLE

008D 1200B4 126 CALL SENDSTRING ; write title block on screen

127

0090 7940 128 MOV R1, #40h ; move value at address 40 into R2

0092 E7 129 MOV A, @R1

0093 FA 130 MOV R2, A

0094 131 NEXT: ; Put new value on a new line

0094 740A 132 MOV A, #10 ; Transmit a linefeed (= ASCII 10)

0096 1200CC 133 CALL SENDCHAR

0099 740D 134 MOV A, #13 ;Transmit a carriage return (=ASCII 13)

009B 1200CC 135 CALL SENDCHAR

136

009E EA 137 MOV A, R2 ; Transmit R2 i.e. value @ address R1

009F 1200D4 138 CALL SENDVAL

00A2 09 139 INC R1 ; Increment address

00A3 E7 140 MOV A, @R1

00A4 FA 141 MOV R2, A ; R2 holds the value @ addrR1

142

00A5 E9 143 MOV A, R1 ; Check if at address 50h

00A6 B450EB 144 CJNE A, #50h, NEXT ; if not jump to Next

00A9 0200AC 145 JMP WAIT1S ; if so wait 1s and repeat

146

00AC 7464 147 WAIT1S: MOV A, #100 ; wait for time less than master to

148 ; synchronise with the master

00AE 1200F4 149 CALL DELAY

00B1 7940 150 MOV R1, #40h ; store new inputs at address 40h again

00B3 22 151 RET

152

153

154 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

155 ; SENDSTRING

156

00B4 157 SENDSTRING: ; sends ASCII string to UART starting at location

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

159

00B4 C0E0 160 PUSH ACC

00B6 C0F0 161 PUSH B

00B8 E4 162 CLR A

00B9 F5F0 163 MOV B,A

00BB E5F0 164 IO0010: MOV A,B

00BD 05F0 165 INC B

00BF 93 166 MOVC A,@A+DPTR

00C0 6005 167 JZ IO0020

00C2 1200CC 168 CALL SENDCHAR

00C5 80F4 169 JMP IO0010

00C7 D0F0 170 IO0020: POP B

00C9 D0E0 171 POP ACC

172

00CB 22 173 RET

174

SLAVUART PAGE 4

175 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

176 ; SENDCHAR

177

00CC 178 SENDCHAR: ; sends ASCII value contained in A to UART

179

00CC 3099FD 180 JNB TI,$ ; wait til present char gone

00CF C299 181 CLR TI ; must clear TI

00D1 F599 182 MOV SBUF,A

183

00D3 22 184 RET

185

186 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

187 ; SENDVAL

188

00D4 189 SENDVAL: ; converts the hex value of A into two ASCII chars,

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

191 ; does not change the value of A.

192

00D4 C0E0 193 PUSH ACC

00D6 C4 194 SWAP A

00D7 1200E8 195 CALL HEX2ASCII

00DA 11CC 196 CALL SENDCHAR ; send high nibble

00DC D0E0 197 POP ACC

00DE C0E0 198 PUSH ACC

00E0 1200E8 199 CALL HEX2ASCII

00E3 11CC 200 CALL SENDCHAR ; send low nibble

00E5 D0E0 201 POP ACC

202

00E7 22 203 RET

204

205

206 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

207 ; HEX2ASCII

208

00E8 209 HEX2ASCII: ; converts A into the hex character representing the

210 ; value of A's least significant nibble

211

00E8 540F 212 ANL A,#00Fh

00EA B40A00 213 CJNE A,#00Ah,$+3

00ED 4002 214 JC IO0030

00EF 2407 215 ADD A,#007h

00F1 2430 216 IO0030: ADD A,#'0'

217

00F3 22 218 RET

219

220 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

221 ; DELAY

222

00F4 223 DELAY: ; Delays by 10ms \* A

224 ;

225

226

00F4 FA 227 MOV R2,A ; Acc holds delay variable

00F5 7B1B 228 DLY0: MOV R3,#01Bh ; Set up delay loop0

00F7 7CFF 229 DLY1: MOV R4,#0FFh ; Set up delay loop1

00F9 DCFE 230 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

231 ;

00FB DBFA 232 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

SLAVUART PAGE 5

233 ;

00FD DAF6 234 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

235 ; wait for ACC\*100ms

00FF 22 236 RET ; Return from subroutine

237

238

239 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

240

241

0100 0A0A0D5F 242 TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

0104 5F5F5F5F

0108 5F5F5F5F

010C 5F5F5F5F

0110 5F5F5F5F

0114 5F5F5F5F

0118 5F5F5F5F

011C 5F5F5F5F

0120 5F5F5F5F

0124 5F5F5F0A

0128 0D

0129 416E616C 243 DB 'Analog Devices MicroConverter ADuC832',10,13

012D 6F672044

0131 65766963

0135 6573204D

0139 6963726F

013D 436F6E76

0141 65727465

0145 72204144

0149 75433833

014D 320A0D

0150 20202020 244 DB ' SPI SLAVE Demo Routine',10,13

0154 20202020

0158 53504920

015C 534C4156

0160 45204465

0164 6D6F2052

0168 6F757469

016C 6E650A0D

0170 20204461 245 DB ' Data Stored in Memory in Hex Form',10,13,0

0174 74612053

0178 746F7265

017C 6420696E

0180 204D656D

0184 6F727920

0188 696E2048

018C 65782046

0190 6F726D0A

0194 0D00

246

247 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

248

249

250 END

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

SLAVUART PAGE 6

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

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

CONT . . . . . . . . . . . . . . C ADDR 0046H

DELAY. . . . . . . . . . . . . . C ADDR 00F4H

DLY0 . . . . . . . . . . . . . . C ADDR 00F5H

DLY1 . . . . . . . . . . . . . . C ADDR 00F7H

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

FLAG . . . . . . . . . . . . . . B ADDR 0000H

HEX2ASCII. . . . . . . . . . . . C ADDR 00E8H

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

IO0010 . . . . . . . . . . . . . C ADDR 00BBH

IO0020 . . . . . . . . . . . . . C ADDR 00C7H

IO0030 . . . . . . . . . . . . . C ADDR 00F1H

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

MAIN . . . . . . . . . . . . . . C ADDR 0060H

NEXT . . . . . . . . . . . . . . C ADDR 0094H

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

PLLCON . . . . . . . . . . . . . D ADDR 00D7H PREDEFINED

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 00CCH

SENDSTRING . . . . . . . . . . . C ADDR 00B4H

SENDVAL. . . . . . . . . . . . . C ADDR 00D4H

SNDUART. . . . . . . . . . . . . C ADDR 0088H

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

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

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

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

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

TITLE. . . . . . . . . . . . . . C ADDR 0100H

TRNSMT . . . . . . . . . . . . . C ADDR 0078H

WAIT1S . . . . . . . . . . . . . C ADDR 00ACH