SLAVUART PAGE 1

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

2 ;

3 ; Author : ADI - Apps

4 ;

5 ; Date : June 2002

6 ;

7 ; File : SLAVuart.asm

8 ;

9 ; Hardware : ADuC831

10 ;

11 ; Description : This slave program transmits the numbers 1-10 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 5s 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 19)

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

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

43 ;

44 $MOD831 ;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 120085 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

78

0060 759E85 79 MOV T3CON,#085h

0063 759D08 80 MOV T3FD,#08h

0066 759852 81 MOV SCON,#052h

82

0069 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

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

006F D2AF 94 SETB EA ; Enable interrupts

95

0071 7940 96 MOV R1, #40h ; initialise R1 to 40 to store the

97 ; input data from memory location 40

0073 780A 98 MOV R0, #0AH ; initialise R0 to 10

99

0075 100 TRNSMT:

0075 08 101 INC R0

0076 88F7 102 MOV SPIDAT, R0 ; transmit the current value on R0

0078 D200 103 SETB FLAG ; set flag so that we wait here until

104 ; the spi interrupt routine clears

105 ; the FLAG

106

007A 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.

007D E8 113 MOV A, R0

007E B414F4 114 CJNE A, #14H, TRNSMT ; if R0 is not 20, jump to TRNSMT

0081 780A 115 MOV R0, #0AH ; if R0=20 make R0=10 & jump to TRNSMT

0083 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

0085 123 SNDUART:

0085 B2B4 124 CPL LED ;CPL LED with each transmission

0087 9000FD 125 MOV DPTR, #TITLE

008A 1200B1 126 CALL SENDSTRING ; write title block on screen

127

008D 7940 128 MOV R1, #40h ; move value at address 40 into R2

008F E7 129 MOV A, @R1

0090 FA 130 MOV R2, A

0091 131 NEXT: ; Put new value on a new line

0091 740A 132 MOV A, #10 ; Transmit a linefeed (= ASCII 10)

0093 1200C9 133 CALL SENDCHAR

0096 740D 134 MOV A, #13 ;Transmit a carriage return (=ASCII 13)

0098 1200C9 135 CALL SENDCHAR

136

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

009C 1200D1 138 CALL SENDVAL

009F 09 139 INC R1 ; Increment address

00A0 E7 140 MOV A, @R1

00A1 FA 141 MOV R2, A ; R2 holds the value @ addrR1

142

00A2 E9 143 MOV A, R1 ; Check if at address 50h

00A3 B450EB 144 CJNE A, #50h, NEXT ; if not jump to Next

00A6 0200A9 145 JMP WAIT1S ; if so wait 1s and repeat

146

00A9 740A 147 WAIT1S: MOV A, #10 ; wait for time less than master to

148 ; synchronise with the master

00AB 1200F1 149 CALL DELAY

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

00B0 22 151 RET

152

153

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

155 ; SENDSTRING

156

00B1 157 SENDSTRING: ; sends ASCII string to UART starting at location

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

159

00B1 C0E0 160 PUSH ACC

00B3 C0F0 161 PUSH B

00B5 E4 162 CLR A

00B6 F5F0 163 MOV B,A

00B8 E5F0 164 IO0010: MOV A,B

00BA 05F0 165 INC B

00BC 93 166 MOVC A,@A+DPTR

00BD 6005 167 JZ IO0020

00BF 1200C9 168 CALL SENDCHAR

00C2 80F4 169 JMP IO0010

00C4 D0F0 170 IO0020: POP B

00C6 D0E0 171 POP ACC

172

00C8 22 173 RET

174

SLAVUART PAGE 4

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

176 ; SENDCHAR

177

00C9 178 SENDCHAR: ; sends ASCII value contained in A to UART

179

00C9 3099FD 180 JNB TI,$ ; wait til present char gone

00CC C299 181 CLR TI ; must clear TI

00CE F599 182 MOV SBUF,A

183

00D0 22 184 RET

185

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

187 ; SENDVAL

188

00D1 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

00D1 C0E0 193 PUSH ACC

00D3 C4 194 SWAP A

00D4 1200E5 195 CALL HEX2ASCII

00D7 11C9 196 CALL SENDCHAR ; send high nibble

00D9 D0E0 197 POP ACC

00DB C0E0 198 PUSH ACC

00DD 1200E5 199 CALL HEX2ASCII

00E0 11C9 200 CALL SENDCHAR ; send low nibble

00E2 D0E0 201 POP ACC

202

00E4 22 203 RET

204

205

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

207 ; HEX2ASCII

208

00E5 209 HEX2ASCII: ; converts A into the hex character representing the

210 ; value of A's least significant nibble

211

00E5 540F 212 ANL A,#00Fh

00E7 B40A00 213 CJNE A,#00Ah,$+3

00EA 4002 214 JC IO0030

00EC 2407 215 ADD A,#007h

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

217

00F0 22 218 RET

219

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

221 ; DELAY

222

00F1 223 DELAY:

224

00F1 FA 225 MOV R2,A ; Acc holds delay variable

00F2 7BC8 226 DLY0: MOV R3,#200 ; Set up delay loop0

00F4 7CE5 227 DLY1: MOV R4,#229 ; Set up delay loop1

00F6 DCFE 228 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

229 ;

00F8 DBFA 230 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

231 ;

00FA DAF6 232 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

SLAVUART PAGE 5

233 ; wait for ACC\*100ms

00FC 22 234 RET ; Return from subroutine

235

236

237 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

238

239

00FD 0A0A0D5F 240 TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

0101 5F5F5F5F

0105 5F5F5F5F

0109 5F5F5F5F

010D 5F5F5F5F

0111 5F5F5F5F

0115 5F5F5F5F

0119 5F5F5F5F

011D 5F5F5F5F

0121 5F5F5F0A

0125 0D

0126 416E616C 241 DB 'Analog Devices MicroConverter ADuC831',10,13

012A 6F672044

012E 65766963

0132 6573204D

0136 6963726F

013A 436F6E76

013E 65727465

0142 72204144

0146 75433833

014A 310A0D

014D 20202020 242 DB ' SPI SLAVE Demo Routine',10,13

0151 20202020

0155 53504920

0159 534C4156

015D 45204465

0161 6D6F2052

0165 6F757469

0169 6E650A0D

016D 20204461 243 DB ' Data Stored in Memory in Hex Form',10,13,0

0171 74612053

0175 746F7265

0179 6420696E

017D 204D656D

0181 6F727920

0185 696E2048

0189 65782046

018D 6F726D0A

0191 0D00

244

245 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

246

247

248 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 00F1H

DLY0 . . . . . . . . . . . . . . C ADDR 00F2H

DLY1 . . . . . . . . . . . . . . C ADDR 00F4H

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

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

HEX2ASCII. . . . . . . . . . . . C ADDR 00E5H

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

IO0010 . . . . . . . . . . . . . C ADDR 00B8H

IO0020 . . . . . . . . . . . . . C ADDR 00C4H

IO0030 . . . . . . . . . . . . . C ADDR 00EEH

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

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

NEXT . . . . . . . . . . . . . . C ADDR 0091H

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

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 00C9H

SENDSTRING . . . . . . . . . . . C ADDR 00B1H

SENDVAL. . . . . . . . . . . . . C ADDR 00D1H

SNDUART. . . . . . . . . . . . . C ADDR 0085H

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 00FDH

TRNSMT . . . . . . . . . . . . . C ADDR 0075H

WAIT1S . . . . . . . . . . . . . C ADDR 00A9H