SLAVUART PAGE 1

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

2 ;

3 ; Author : ADI - Apps

4 ;

5 ; Date : January 2001

6 ;

7 ; File : SLAVuart.asm

8 ;

9 ; Hardware : ADuC816

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 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 14)

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

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

43 ;

44 $MOD816 ;Use 8052 / ADuC816 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 12008E 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 75CBFF 79 MOV RCAP2H,#0FFh ; config UART for 9830baud

0063 75CAFB 80 MOV RCAP2L,#-5 ; (close enough to 9600baud)

0066 75CDFF 81 MOV TH2,#0FFh

0069 75CCFB 82 MOV TL2,#-5

006C 759852 83 MOV SCON,#52h

006F 75C834 84 MOV T2CON,#34h

85

0072 75F824 86 MOV SPICON,#24h ; Initialise SPICON to have

87 ; -Enable SPI serial port

88 ; -slave mode select

89 ; -CPOL=0, clk idling low

90 ; -CPHA=1

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

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

93 ; will be measued during its change of state, and not when at

94 ; its final value.

95

0075 75A901 96 MOV IEIP2, #01h ; Enable SPI interrupt

0078 D2AF 97 SETB EA ; Enable interrupts

98

007A 7940 99 MOV R1, #40h ; initialise R1 to 40 to store the

100 ; input data from memory location 40

007C 780A 101 MOV R0, #0AH ; initialise R0 to 10

102

007E 103 TRNSMT:

007E 08 104 INC R0

007F 88F7 105 MOV SPIDAT, R0 ; transmit the current value on R0

0081 D200 106 SETB FLAG ; set flag so that we wait here until

107 ; the spi interrupt routine clears

108 ; the FLAG

109

0083 2000FD 110 JB FLAG, $ ; stay here until flag is cleared

111 ; by interrupt

112

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

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

115 ; from 11 again.

0086 E8 116 MOV A, R0

SLAVUART PAGE 3

0087 B414F4 117 CJNE A, #14H, TRNSMT ; if R0 is not 20, jump to TRNSMT

008A 780A 118 MOV R0, #0AH ; if R0=20 make R0=10 & jump to TRNSMT

008C 80F0 119 JMP TRNSMT

120

121

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

123 ; 1 seconds and then transmit and receive values to/from the Master

124 ; again down the SPI port.

125

008E 126 SNDUART:

008E B2B4 127 CPL LED ;CPL LED with each transmission

0090 900106 128 MOV DPTR, #TITLE

0093 1200BA 129 CALL SENDSTRING ; write title block on screen

130

0096 7940 131 MOV R1, #40h ; move value at address 40 into R2

0098 E7 132 MOV A, @R1

0099 FA 133 MOV R2, A

009A 134 NEXT: ; Put new value on a new line

009A 740A 135 MOV A, #10 ; Transmit a linefeed (= ASCII 10)

009C 1200D2 136 CALL SENDCHAR

009F 740D 137 MOV A, #13 ;Transmit a carriage return (=ASCII 13)

00A1 1200D2 138 CALL SENDCHAR

139

00A4 EA 140 MOV A, R2 ; Transmit R2 i.e. value @ address R1

00A5 1200DA 141 CALL SENDVAL

00A8 09 142 INC R1 ; Increment address

00A9 E7 143 MOV A, @R1

00AA FA 144 MOV R2, A ; R2 holds the value @ addrR1

145

00AB E9 146 MOV A, R1 ; Check if at address 50h

00AC B450EB 147 CJNE A, #50h, NEXT ; if not jump to Next

00AF 0200B2 148 JMP WAIT1S ; if so wait 1s and repeat

149

00B2 740A 150 WAIT1S: MOV A, #10 ; wait for time less than master to

151 ; synchronise with the master

00B4 1200FA 152 CALL DELAY

00B7 7940 153 MOV R1, #40h ; store new inputs at address 40h again

00B9 22 154 RET

155

156

157 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

158 ; SENDSTRING

159

00BA 160 SENDSTRING: ; sends ASCII string to UART starting at location

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

162

00BA C0E0 163 PUSH ACC

00BC C0F0 164 PUSH B

00BE E4 165 CLR A

00BF F5F0 166 MOV B,A

00C1 E5F0 167 IO0010: MOV A,B

00C3 05F0 168 INC B

00C5 93 169 MOVC A,@A+DPTR

00C6 6005 170 JZ IO0020

00C8 1200D2 171 CALL SENDCHAR

00CB 80F4 172 JMP IO0010

00CD D0F0 173 IO0020: POP B

00CF D0E0 174 POP ACC

SLAVUART PAGE 4

175

00D1 22 176 RET

177

178 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

179 ; SENDCHAR

180

00D2 181 SENDCHAR: ; sends ASCII value contained in A to UART

182

00D2 3099FD 183 JNB TI,$ ; wait til present char gone

00D5 C299 184 CLR TI ; must clear TI

00D7 F599 185 MOV SBUF,A

186

00D9 22 187 RET

188

189 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

190 ; SENDVAL

191

00DA 192 SENDVAL: ; converts the hex value of A into two ASCII chars,

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

194 ; does not change the value of A.

195

00DA C0E0 196 PUSH ACC

00DC C4 197 SWAP A

00DD 1200EE 198 CALL HEX2ASCII

00E0 11D2 199 CALL SENDCHAR ; send high nibble

00E2 D0E0 200 POP ACC

00E4 C0E0 201 PUSH ACC

00E6 1200EE 202 CALL HEX2ASCII

00E9 11D2 203 CALL SENDCHAR ; send low nibble

00EB D0E0 204 POP ACC

205

00ED 22 206 RET

207

208

209 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

210 ; HEX2ASCII

211

00EE 212 HEX2ASCII: ; converts A into the hex character representing the

213 ; value of A's least significant nibble

214

00EE 540F 215 ANL A,#00Fh

00F0 B40A00 216 CJNE A,#00Ah,$+3

00F3 4002 217 JC IO0030

00F5 2407 218 ADD A,#007h

00F7 2430 219 IO0030: ADD A,#'0'

220

00F9 22 221 RET

222

223 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

224 ; DELAY

00FA 225 DELAY: ; Delays by 13,000 \* A instructions (approx)

226 ; 100mSec based on 1.5728MHZ Core Clock (default)

227

00FA F8 228 MOV R0,A ; Acc holds delay variable

00FB 7941 229 DLY0: MOV R1,#65 ; Set up delay loop0

00FD 7AC8 230 DLY1: MOV R2,#200 ; Set up delay loop1

00FF DAFE 231 DJNZ R2,$ ; Dec R2 200 times

0101 D9FA 232 DJNZ R1,DLY1 ; Dec R1 65 times

SLAVUART PAGE 5

0103 D8F6 233 DJNZ R0,DLY0 ; Dec R0 #ACC times

0105 22 234 RET ; Return from subroutine

235

236 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

237

238

0106 0A0A0D5F 239 TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

010A 5F5F5F5F

010E 5F5F5F5F

0112 5F5F5F5F

0116 5F5F5F5F

011A 5F5F5F5F

011E 5F5F5F5F

0122 5F5F5F5F

0126 5F5F5F5F

012A 5F5F5F0A

012E 0D

012F 416E616C 240 DB 'Analog Devices MicroConverter ADuC824',10,13

0133 6F672044

0137 65766963

013B 6573204D

013F 6963726F

0143 436F6E76

0147 65727465

014B 72204144

014F 75433832

0153 340A0D

0156 20202020 241 DB ' SPI SLAVE Demo Routine',10,13

015A 20202020

015E 53504920

0162 534C4156

0166 45204465

016A 6D6F2052

016E 6F757469

0172 6E650A0D

0176 20204461 242 DB ' Data Stored in Memory in Hex Form',10,13,0

017A 74612053

017E 746F7265

0182 6420696E

0186 204D656D

018A 6F727920

018E 696E2048

0192 65782046

0196 6F726D0A

019A 0D00

243

244 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

245

246

247 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 00FAH

DLY0 . . . . . . . . . . . . . . C ADDR 00FBH

DLY1 . . . . . . . . . . . . . . C ADDR 00FDH

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

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

HEX2ASCII. . . . . . . . . . . . C ADDR 00EEH

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

IO0010 . . . . . . . . . . . . . C ADDR 00C1H

IO0020 . . . . . . . . . . . . . C ADDR 00CDH

IO0030 . . . . . . . . . . . . . C ADDR 00F7H

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

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

NEXT . . . . . . . . . . . . . . C ADDR 009AH

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

RCAP2H . . . . . . . . . . . . . D ADDR 00CBH PREDEFINED

RCAP2L . . . . . . . . . . . . . D ADDR 00CAH PREDEFINED

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

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

SENDCHAR . . . . . . . . . . . . C ADDR 00D2H

SENDSTRING . . . . . . . . . . . C ADDR 00BAH

SENDVAL. . . . . . . . . . . . . C ADDR 00DAH

SNDUART. . . . . . . . . . . . . C ADDR 008EH

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

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

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

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

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

TITLE. . . . . . . . . . . . . . C ADDR 0106H

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

TRNSMT . . . . . . . . . . . . . C ADDR 007EH

WAIT1S . . . . . . . . . . . . . C ADDR 00B2H