I2CMSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : Oct 2000

6 ;

7 ; File : i2Cmstr.asm

8 ;

9 ; Hardware : ADuC836 (commented out = ADuC812)

10 ;

11 ; Description : Code for a master in an I2C system. This code will

12 ; continuously receive and transmit a byte over the I2C

13 ; interface, then send the received byte out the UART,

14 ; then check if a character had been entered in the UART,

15 ; if so, it will send the ASCII value of the character

16 ; entered to the slave, the next time it transmits a byte.

17 ;

18 ; Reference : Tech Note, uC001: "MicroConverter I2C Compatible

19 ; Interface" find it at www.analog.com/microconverter

20

21 ;

22 ;======================================================================

23

24

25 $MOD836

26

27 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

28 ; DEFINE VARIABLES IN INTERNAL RAM

29

0030 30 BITCNT DATA 30h ; bit counter for I2C routines

0031 31 SLAVEADD DATA 31h ; slave address for I2C routines

0032 32 INPUT DATA 32h ; data recieved from the slave

0033 33 OUTPUT DATA 33h ; data to be transmitted to slave

34

0000 35 NOACK BIT 00h ; I2C no acknowledge flag

0000 36 ERR BIT 00h ; I2C error flag

37

00B4 38 LED EQU P3.4

39

40

41

42 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

43 ; BEGINNING OF CODE

---- 44 CSEG

0000 45 ORG 0000h

0000 020060 46 JMP MAIN

47

48

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

50 ; INT0 ISR

0003 51 ORG 0003h

0003 0533 52 INC OUTPUT

0005 32 53 RETI

54

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

56 ; MAIN PROGRAM

0060 57 ORG 0060h

0060 58 MAIN:

I2CMSTR PAGE 2

59

60 ; configure the UART ADuC812

61 ; MOV SCON,#52h ; configure UART for 9600baud..

62 ; MOV TMOD,#20h ; ..assuming 11.0592MHz crystal

63 ; MOV TH1,#-3

64 ; SETB TR1

65

66 ; configure the UART ADuC824/ADuC816

67 ; MOV RCAP2H,#0FFh ; config UART for 9830baud

68 ; MOV RCAP2L,#-5 ; (close enough to 9600baud)

69 ; MOV TH2,#0FFh

70 ; MOV TL2,#-5

71 ; MOV SCON,#52h

72 ; MOV T2CON,#34h

73

74 ; configure UART for 9600 using Timer3

0060 759E82 75 MOV T3CON,#82h

0063 759D12 76 MOV T3FD,#12h

0066 759852 77 MOV SCON,#52h

78

79 ; configure & enable interrupts

0069 D2A8 80 SETB EX0 ; enable INT0

006B D288 81 SETB IT0 ; INT0 edge triggered

006D D2AF 82 SETB EA ; allow all the interrupts

83

84 ; initialise settings

006F 753188 85 MOV SLAVEADD,#88H ; clear RW bit

0072 75E8A8 86 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

87 ; selects master mode

0075 753300 88 MOV OUTPUT,#0 ; TX 0 as default

0078 C200 89 CLR NOACK

007A C200 90 CLR ERR

91

007C 92 RXTXLOOP:

93 ; code for a read mode ( master recieves one byte from slave )

007C 1200C6 94 CALL RCVDATA ; sends start bit

95 ; sends address byte

96 ; checks acknowledge

97 ; receives byte into ACC

98 ; checks ACK

99 ; sends stop bit

100

101 ; code for write mode ( master transmits one byte to slave )

007F 1200AD 102 CALL SENDDATA ; sends start bit

103 ; sends address byte

104 ; checks acknowledge

105 ; transmits ACC

106 ; checks ACK

107 ; sends stop bit

108

109 ; Check for Error message

0082 200008 110 JB ERR,SENDERR ; if error, send error message

111

112 ; Transmit received byte (INPUT) up UART to PC (hyperterminal)

0085 E532 113 MOV A,INPUT ; put value received into ACC

0087 120152 114 CALL SENDVAL ; send value received out the UART

008A 020092 115 JMP SKIP

116

I2CMSTR PAGE 3

008D 117 SENDERR:

008D 120138 118 CALL ERROR ; send error message out the UART

0090 C200 119 CLR ERR ; clear error flag

120

0092 121 SKIP:

0092 740A 122 MOV A,#10 ; send LF+CR

0094 12013E 123 CALL SENDCHAR

0097 740D 124 MOV A,#13

0099 12013E 125 CALL SENDCHAR

126

127 ; Toggle LED (1s delay so that LED can be seen toggle)

009C 740A 128 MOV A, #10

009E 12012C 129 CALL DELAY

00A1 B2B4 130 CPL LED

131

132 ; Check for new OUTPUT

00A3 3098D6 133 JNB RI, RXTXLOOP ; repeat (unless UART data received)

134

135 ; If UART data received, then save to OUTPUT

00A6 859933 136 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00A9 C298 137 CLR RI ; must clear RI

00AB 80CF 138 JMP RXTXLOOP ; back to main loop

139

140

141 ;====================================================================

142 ; SUBROUTINES

143 ;====================================================================

144

145 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

146 ; SENDDATA

147 ; Send all the sequence to the slave (slave address + data (OUTPUT))

148

00AD 149 SENDDATA:

150 ; send start bit

00AD 1200E3 151 CALL STARTBIT ; acquire bus and send slave address

152

153 ; send slave address

00B0 E531 154 MOV A, SLAVEADD

00B2 1200F5 155 CALL SENDBYTE ; sets NOACK if NACK received

156

00B5 200005 157 JB NOACK, STOPSEND ; if no acknowledge send stop

158

159 ; send OUTPUT byte

00B8 E533 160 MOV A, OUTPUT

00BA 1200F5 161 CALL SENDBYTE ; sets NOACK if NACK received

162

00BD 163 STOPSEND:

00BD 1200EC 164 CALL STOPBIT ; sends stop bit

00C0 300002 165 JNB NOACK, SENDRET ; if slave sends NACK send error

00C3 D200 166 SETB ERR ; sets the error flag

00C5 167 SENDRET:

00C5 22 168 RET

169

170 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

171 ; RCVDATA

172 ; receives one or more bytes of data from an I2C slave device.

173

00C6 174 RCVDATA:

I2CMSTR PAGE 4

00C6 0531 175 INC SLAVEADD ; Set RW for reception

176

177 ; send start bit

00C8 1200E3 178 CALL STARTBIT ; acquire bus and send slave address

179

180 ; send slave address

00CB E531 181 MOV A, SLAVEADD

00CD 1200F5 182 CALL SENDBYTE ; sets NOACK if NACK received

183

00D0 1531 184 DEC SLAVEADD ; returns SLAVEADD to 88h (after INC)

185

00D2 200005 186 JB NOACK, STOPRCV ; Check for slave not responding.

00D5 120112 187 CALL RCVBYTE ; Receive next data byte.

00D8 F532 188 MOV INPUT,A ; Save data byte in buffer.

189

00DA 190 STOPRCV:

00DA 1200EC 191 CALL STOPBIT

00DD 300002 192 JNB NOACK, RCVRET ; if slave sends NACK send error

00E0 D200 193 SETB ERR ; sets the error flag

00E2 194 RCVRET:

00E2 22 195 RET

196 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

197 ; STARTBIT

198 ; Sends the start bit to initiate an I2C communication

199

00E3 200 STARTBIT:

00E3 D2EE 201 SETB MDE ; enable SDATA pin as an output

00E5 C200 202 CLR NOACK

00E7 C2EF 203 CLR MDO ; low O/P on SDATA

00E9 C2ED 204 CLR MCO ; start bit

00EB 22 205 RET

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

207 ; STOPBIT

208 ; Sends the stop bit to end an I2C transmission

209

00EC 210 STOPBIT:

00EC D2EE 211 SETB MDE ; to enable SDATA pin as an output

00EE C2EF 212 CLR MDO ; get SDATA ready for stop

00F0 D2ED 213 SETB MCO ; set clock for stop

00F2 D2EF 214 SETB MDO ; this is the stop bit

00F4 22 215 RET

216 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

217 ; SENDBYTE

218 ; Send 8-bits in ACC to the slave

00F5 219 SENDBYTE:

00F5 753008 220 MOV BITCNT,#8 ; 8 bits in a byte

00F8 D2EE 221 SETB MDE ; to enable SDATA pin as an output

00FA C2ED 222 CLR MCO ; make sure that the clock line is low

00FC 223 SENDBIT:

00FC 33 224 RLC A ; put data bit to be sent into carry

00FD 92EF 225 MOV MDO,C ; put data bit on SDATA line

00FF D2ED 226 SETB MCO ; clock to send bit

0101 C2ED 227 CLR MCO ; clear clock

0103 D530F6 228 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

229

0106 C2EE 230 CLR MDE ; release data line for acknowledge

0108 D2ED 231 SETB MCO ; send clock for acknowledge

010A 30EC02 232 JNB MDI,NEXT ; this is a check for acknowledge

I2CMSTR PAGE 5

010D D200 233 SETB NOACK ; no acknowledge, set flag

010F C2ED 234 NEXT: CLR MCO ; clear clock

0111 22 235 RET

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

237 ; RCVBYTE

238 ; receives one byte of data from an I2C slave device. Returns it in A

239

0112 240 RCVBYTE:

0112 753008 241 MOV BITCNT,#8 ; Set bit count.

0115 C2EE 242 CLR MDE ; to enable SDATA pin as an input

0117 C2ED 243 CLR MCO ; make sure the clock line is low

0119 244 RCVBIT:

0119 D2ED 245 SETB MCO ; clock to recieve bit

011B C2ED 246 CLR MCO ; clear clock

011D A2EC 247 MOV C,MDI ; read data bit into carry.

011F 33 248 RLC A ; Rotate bit into result byte.

249

0120 D530F6 250 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

251 ; recieved byte is in the accumulator

252

0123 D2EE 253 SETB MDE ; Data pin =Output for NACK

0125 D2EF 254 SETB MDO ; Send NACK (always send NACK for

255 ; last byte in transmission)

0127 D2ED 256 SETB MCO ; Send NACK clock.

0129 C2ED 257 CLR MCO

012B 22 258 RET

259

260 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

261 ; DELAY

262 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

012C 263 DELAY: ; Delays by 100ms \* A

264

265 ; ADuC812 100ms based on 11.0592MHz Core Clock

266 ; ADuC824 100ms based on 1.573MHz Core Clock

267

012C FA 268 MOV R2,A ; Acc holds delay variable

269 ;DLY0: MOV R3,#200 ; Set up delay loop0

270 ;DLY1: MOV R4,#229 ; Set up delay loop1

012D 7B32 271 DLY0: MOV R3,#50 ; Set up delay loop0

012F 7C83 272 DLY1: MOV R4,#131 ; Set up delay loop1

0131 DCFE 273 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

274 ; wait here for 131\*15.3us=2ms

0133 DBFA 275 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

276 ; Wait for 50\*2ms

0135 DAF6 277 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

278 ; wait for ACC\*100ms

0137 22 279 RET ; Return from subroutine

280 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

281 ; ERROR

282 ; this subroutine is run if a NACK is received from the slave

283

0138 284 ERROR:

0138 7445 285 MOV A,#45h

013A 12013E 286 CALL SENDCHAR ; send the letter E out the UART

013D 22 287 RET

288 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

289 ; SENDCHAR

290 ; sends ASCII value contained in A to UART

I2CMSTR PAGE 6

291

013E 292 SENDCHAR:

013E 3099FD 293 JNB TI,$ ; wait til present char gone

0141 C299 294 CLR TI ; must clear TI

0143 F599 295 MOV SBUF,A

0145 22 296 RET

297 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

298 ; HEX2ASCII

299 ; converts A into the hex character representing the value of A's

300 ; least significant nibble

301

0146 302 HEX2ASCII:

0146 540F 303 ANL A,#00Fh

0148 B40A00 304 CJNE A,#00Ah,$+3

014B 4002 305 JC IO0030

014D 2407 306 ADD A,#007h

014F 2430 307 IO0030: ADD A,#'0'

0151 22 308 RET

309 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

310 ; SENDVAL

311 ; converts the hex value of A into two ASCII chars, and then spits

312 ; these two characters up the UART. does not change the value of A.

313

0152 314 SENDVAL:

0152 C0E0 315 PUSH ACC

0154 C4 316 SWAP A

0155 3146 317 CALL HEX2ASCII

0157 313E 318 CALL SENDCHAR ; send high nibble

0159 D0E0 319 POP ACC

015B C0E0 320 PUSH ACC

015D 3146 321 CALL HEX2ASCII

015F 313E 322 CALL SENDCHAR ; send low nibble

0161 D0E0 323 POP ACC

0163 22 324 RET

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

326

327 END

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

I2CMSTR PAGE 7

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

BITCNT . . . . . . . . . . . . . D ADDR 0030H

DELAY. . . . . . . . . . . . . . C ADDR 012CH

DLY0 . . . . . . . . . . . . . . C ADDR 012DH

DLY1 . . . . . . . . . . . . . . C ADDR 012FH

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

ERR. . . . . . . . . . . . . . . B ADDR 0000H

ERROR. . . . . . . . . . . . . . C ADDR 0138H

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

HEX2ASCII. . . . . . . . . . . . C ADDR 0146H

I2CCON . . . . . . . . . . . . . D ADDR 00E8H PREDEFINED

INPUT. . . . . . . . . . . . . . D ADDR 0032H

IO0030 . . . . . . . . . . . . . C ADDR 014FH

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

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

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

MCO. . . . . . . . . . . . . . . B ADDR 00EDH PREDEFINED

MDE. . . . . . . . . . . . . . . B ADDR 00EEH PREDEFINED

MDI. . . . . . . . . . . . . . . B ADDR 00ECH PREDEFINED

MDO. . . . . . . . . . . . . . . B ADDR 00EFH PREDEFINED

NEXT . . . . . . . . . . . . . . C ADDR 010FH

NOACK. . . . . . . . . . . . . . B ADDR 0000H

OUTPUT . . . . . . . . . . . . . D ADDR 0033H

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

RCVBIT . . . . . . . . . . . . . C ADDR 0119H

RCVBYTE. . . . . . . . . . . . . C ADDR 0112H

RCVDATA. . . . . . . . . . . . . C ADDR 00C6H

RCVRET . . . . . . . . . . . . . C ADDR 00E2H

RI . . . . . . . . . . . . . . . B ADDR 0098H PREDEFINED

RXTXLOOP . . . . . . . . . . . . C ADDR 007CH

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

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

SENDBIT. . . . . . . . . . . . . C ADDR 00FCH

SENDBYTE . . . . . . . . . . . . C ADDR 00F5H

SENDCHAR . . . . . . . . . . . . C ADDR 013EH

SENDDATA . . . . . . . . . . . . C ADDR 00ADH

SENDERR. . . . . . . . . . . . . C ADDR 008DH

SENDRET. . . . . . . . . . . . . C ADDR 00C5H

SENDVAL. . . . . . . . . . . . . C ADDR 0152H

SKIP . . . . . . . . . . . . . . C ADDR 0092H

SLAVEADD . . . . . . . . . . . . D ADDR 0031H

STARTBIT . . . . . . . . . . . . C ADDR 00E3H

STOPBIT. . . . . . . . . . . . . C ADDR 00ECH

STOPRCV. . . . . . . . . . . . . C ADDR 00DAH

STOPSEND . . . . . . . . . . . . C ADDR 00BDH

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

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

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