I2CMSTR PAGE 1

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

2 ;

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

4 ;

5 ; Date : May 2002

6 ;

7 ; File : i2Cmstr.asm

8 ;

9 ; Hardware : ADuC814

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 $MOD814

25 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

26 ; DEFINE VARIABLES IN INTERNAL RAM

27

0030 28 BITCNT DATA 30h ; bit counter for I2C routines

0031 29 SLAVEADD DATA 31h ; slave address for I2C routines

0032 30 INPUT DATA 32h ; data recieved from the slave

0033 31 OUTPUT DATA 33h ; data to be transmitted to slave

32

0000 33 NOACK BIT 00h ; I2C no acknowledge flag

0000 34 ERR BIT 00h ; I2C error flag

35

00B4 36 LED EQU P3.4

37

38

39

40 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

41 ; BEGINNING OF CODE

---- 42 CSEG

0000 43 ORG 0000h

0000 020060 44 JMP MAIN

45

46

47 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

48 ; INT0 ISR

0003 49 ORG 0003h

0003 0533 50 INC OUTPUT

0005 32 51 RETI

52

53 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

54 ; MAIN PROGRAM

0060 55 ORG 0060h

0060 56 MAIN:

57

58

I2CMSTR PAGE 2

59 ; configure the UART ADuC812s

0060 75CBFF 60 MOV RCAP2H,#0FFh ; config UART for 9830baud

0063 75CAF9 61 MOV RCAP2L,#-7 ; (close enough to 9600baud)

0066 75CDFF 62 MOV TH2,#0FFh

0069 75CCF9 63 MOV TL2,#-7

006C 759852 64 MOV SCON,#52h

006F 75C834 65 MOV T2CON,#34h

66

67 ; enable i2c pins on 812s

0072 759C01 68 MOV CFG814,#01H

69

70 ; configure & enable interrupts

0075 D2A8 71 SETB EX0 ; enable INT0

0077 D288 72 SETB IT0 ; INT0 edge triggered

0079 D2AF 73 SETB EA ; allow all the interrupts

74

75 ; initialise settings

007B 753188 76 MOV SLAVEADD,#88H ; clear RW bit

007E 75E8A8 77 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

78 ; selects master mode

0081 753300 79 MOV OUTPUT,#0 ; TX 0 as default

0084 C200 80 CLR NOACK

0086 C200 81 CLR ERR

82

0088 83 RXTXLOOP:

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

0088 1200D2 85 CALL RCVDATA ; sends start bit

86 ; sends address byte

87 ; checks acknowledge

88 ; receives byte into ACC

89 ; checks ACK

90 ; sends stop bit

91

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

008B 1200B9 93 CALL SENDDATA ; sends start bit

94 ; sends address byte

95 ; checks acknowledge

96 ; transmits ACC

97 ; checks ACK

98 ; sends stop bit

99

100 ; Check for Error message

008E 200008 101 JB ERR,SENDERR ; if error, send error message

102

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

0091 E532 104 MOV A,INPUT ; put value received into ACC

0093 12015E 105 CALL SENDVAL ; send value received out the UART

0096 02009E 106 JMP SKIP

107

0099 108 SENDERR:

0099 120144 109 CALL ERROR ; send error message out the UART

009C C200 110 CLR ERR ; clear error flag

111

009E 112 SKIP:

009E 740A 113 MOV A,#10 ; send LF+CR

00A0 12014A 114 CALL SENDCHAR

00A3 740D 115 MOV A,#13

00A5 12014A 116 CALL SENDCHAR

I2CMSTR PAGE 3

117

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

00A8 740A 119 MOV A, #10

00AA 120138 120 CALL DELAY

00AD B2B4 121 CPL LED

122

123 ; Check for new OUTPUT

00AF 3098D6 124 JNB RI, RXTXLOOP ; repeat (unless UART data received)

125

126 ; If UART data received, then save to OUTPUT

00B2 859933 127 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00B5 C298 128 CLR RI ; must clear RI

00B7 80CF 129 JMP RXTXLOOP ; back to main loop

130

131

132 ;====================================================================

133 ; SUBROUTINES

134 ;====================================================================

135

136 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

137 ; SENDDATA

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

139

00B9 140 SENDDATA:

141 ; send start bit

00B9 1200EF 142 CALL STARTBIT ; acquire bus and send slave address

143

144 ; send slave address

00BC E531 145 MOV A, SLAVEADD

00BE 120101 146 CALL SENDBYTE ; sets NOACK if NACK received

147

00C1 200005 148 JB NOACK, STOPSEND ; if no acknowledge send stop

149

150 ; send OUTPUT byte

00C4 E533 151 MOV A, OUTPUT

00C6 120101 152 CALL SENDBYTE ; sets NOACK if NACK received

153

00C9 154 STOPSEND:

00C9 1200F8 155 CALL STOPBIT ; sends stop bit

00CC 300002 156 JNB NOACK, SENDRET ; if slave sends NACK send error

00CF D200 157 SETB ERR ; sets the error flag

00D1 158 SENDRET:

00D1 22 159 RET

160

161 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

162 ; RCVDATA

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

164

00D2 165 RCVDATA:

00D2 0531 166 INC SLAVEADD ; Set RW for reception

167

168 ; send start bit

00D4 1200EF 169 CALL STARTBIT ; acquire bus and send slave address

170

171 ; send slave address

00D7 E531 172 MOV A, SLAVEADD

00D9 120101 173 CALL SENDBYTE ; sets NOACK if NACK received

174

I2CMSTR PAGE 4

00DC 1531 175 DEC SLAVEADD ; returns SLAVEADD to 88h (after INC)

176

00DE 200005 177 JB NOACK, STOPRCV ; Check for slave not responding.

00E1 12011E 178 CALL RCVBYTE ; Receive next data byte.

00E4 F532 179 MOV INPUT,A ; Save data byte in buffer.

180

00E6 181 STOPRCV:

00E6 1200F8 182 CALL STOPBIT

00E9 300002 183 JNB NOACK, RCVRET ; if slave sends NACK send error

00EC D200 184 SETB ERR ; sets the error flag

00EE 185 RCVRET:

00EE 22 186 RET

187 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

188 ; STARTBIT

189 ; Sends the start bit to initiate an I2C communication

190

00EF 191 STARTBIT:

00EF D2EE 192 SETB MDE ; enable SDATA pin as an output

00F1 C200 193 CLR NOACK

00F3 C2EF 194 CLR MDO ; low O/P on SDATA

00F5 C2ED 195 CLR MCO ; start bit

00F7 22 196 RET

197 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

198 ; STOPBIT

199 ; Sends the stop bit to end an I2C transmission

200

00F8 201 STOPBIT:

00F8 D2EE 202 SETB MDE ; to enable SDATA pin as an output

00FA C2EF 203 CLR MDO ; get SDATA ready for stop

00FC D2ED 204 SETB MCO ; set clock for stop

00FE D2EF 205 SETB MDO ; this is the stop bit

0100 22 206 RET

207 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

208 ; SENDBYTE

209 ; Send 8-bits in ACC to the slave

0101 210 SENDBYTE:

0101 753008 211 MOV BITCNT,#8 ; 8 bits in a byte

0104 D2EE 212 SETB MDE ; to enable SDATA pin as an output

0106 C2ED 213 CLR MCO ; make sure that the clock line is low

0108 214 SENDBIT:

0108 33 215 RLC A ; put data bit to be sent into carry

0109 92EF 216 MOV MDO,C ; put data bit on SDATA line

010B D2ED 217 SETB MCO ; clock to send bit

010D C2ED 218 CLR MCO ; clear clock

010F D530F6 219 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

220

0112 C2EE 221 CLR MDE ; release data line for acknowledge

0114 D2ED 222 SETB MCO ; send clock for acknowledge

0116 30EC02 223 JNB MDI,NEXT ; this is a check for acknowledge

0119 D200 224 SETB NOACK ; no acknowledge, set flag

011B C2ED 225 NEXT: CLR MCO ; clear clock

011D 22 226 RET

227 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

228 ; RCVBYTE

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

230

011E 231 RCVBYTE:

011E 753008 232 MOV BITCNT,#8 ; Set bit count.

I2CMSTR PAGE 5

0121 C2EE 233 CLR MDE ; to enable SDATA pin as an input

0123 C2ED 234 CLR MCO ; make sure the clock line is low

0125 235 RCVBIT:

0125 D2ED 236 SETB MCO ; clock to recieve bit

0127 C2ED 237 CLR MCO ; clear clock

0129 A2EC 238 MOV C,MDI ; read data bit into carry.

012B 33 239 RLC A ; Rotate bit into result byte.

240

012C D530F6 241 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

242 ; recieved byte is in the accumulator

243

012F D2EE 244 SETB MDE ; Data pin =Output for NACK

0131 D2EF 245 SETB MDO ; Send NACK (always send NACK for

246 ; last byte in transmission)

0133 D2ED 247 SETB MCO ; Send NACK clock.

0135 C2ED 248 CLR MCO

0137 22 249 RET

250

251 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

252 ; DELAY

253 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

0138 254 DELAY: ; Delays by 100ms \* A

255

256 ; ADuC812 100ms based on 11.0592MHz Core Clock

257 ; ADuC824 100ms based on 1.573MHz Core Clock

258

0138 FA 259 MOV R2,A ; Acc holds delay variable

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

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

0139 7B22 262 DLY0: MOV R3,#22H ; Set up delay loop0

013B 7CFF 263 DLY1: MOV R4,#0FFH ; Set up delay loop1

013D DCFE 264 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

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

013F DBFA 266 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

267 ; Wait for 50\*2ms

0141 DAF6 268 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

269 ; wait for ACC\*100ms

0143 22 270 RET ; Return from subroutine

271 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

272 ; ERROR

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

274

0144 275 ERROR:

0144 7445 276 MOV A,#45h

0146 12014A 277 CALL SENDCHAR ; send the letter E out the UART

0149 22 278 RET

279 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

280 ; SENDCHAR

281 ; sends ASCII value contained in A to UART

282

014A 283 SENDCHAR:

014A 3099FD 284 JNB TI,$ ; wait til present char gone

014D C299 285 CLR TI ; must clear TI

014F F599 286 MOV SBUF,A

0151 22 287 RET

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

289 ; HEX2ASCII

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

I2CMSTR PAGE 6

291 ; least significant nibble

292

0152 293 HEX2ASCII:

0152 540F 294 ANL A,#00Fh

0154 B40A00 295 CJNE A,#00Ah,$+3

0157 4002 296 JC IO0030

0159 2407 297 ADD A,#007h

015B 2430 298 IO0030: ADD A,#'0'

015D 22 299 RET

300 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

301 ; SENDVAL

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

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

304

015E 305 SENDVAL:

015E C0E0 306 PUSH ACC

0160 C4 307 SWAP A

0161 3152 308 CALL HEX2ASCII

0163 314A 309 CALL SENDCHAR ; send high nibble

0165 D0E0 310 POP ACC

0167 C0E0 311 PUSH ACC

0169 3152 312 CALL HEX2ASCII

016B 314A 313 CALL SENDCHAR ; send low nibble

016D D0E0 314 POP ACC

016F 22 315 RET

316 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

317

318 END

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

I2CMSTR PAGE 7

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

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

CFG814 . . . . . . . . . . . . . D ADDR 009CH PREDEFINED

DELAY. . . . . . . . . . . . . . C ADDR 0138H

DLY0 . . . . . . . . . . . . . . C ADDR 0139H

DLY1 . . . . . . . . . . . . . . C ADDR 013BH

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

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

ERROR. . . . . . . . . . . . . . C ADDR 0144H

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

HEX2ASCII. . . . . . . . . . . . C ADDR 0152H

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

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

IO0030 . . . . . . . . . . . . . C ADDR 015BH

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 011BH

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

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

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

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

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

RCVBIT . . . . . . . . . . . . . C ADDR 0125H

RCVBYTE. . . . . . . . . . . . . C ADDR 011EH

RCVDATA. . . . . . . . . . . . . C ADDR 00D2H

RCVRET . . . . . . . . . . . . . C ADDR 00EEH

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

RXTXLOOP . . . . . . . . . . . . C ADDR 0088H

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

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

SENDBIT. . . . . . . . . . . . . C ADDR 0108H

SENDBYTE . . . . . . . . . . . . C ADDR 0101H

SENDCHAR . . . . . . . . . . . . C ADDR 014AH

SENDDATA . . . . . . . . . . . . C ADDR 00B9H

SENDERR. . . . . . . . . . . . . C ADDR 0099H

SENDRET. . . . . . . . . . . . . C ADDR 00D1H

SENDVAL. . . . . . . . . . . . . C ADDR 015EH

SKIP . . . . . . . . . . . . . . C ADDR 009EH

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

STARTBIT . . . . . . . . . . . . C ADDR 00EFH

STOPBIT. . . . . . . . . . . . . C ADDR 00F8H

STOPRCV. . . . . . . . . . . . . C ADDR 00E6H

STOPSEND . . . . . . . . . . . . C ADDR 00C9H

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

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

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

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