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 : ADuC812 (commented out = ADuC824/ADuC816)

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 $MOD812 ; use ADuC812 & 8052 predefined symbols

25 ;$MOD816

26 ;$MOD824

27

28 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

29 ; DEFINE VARIABLES IN INTERNAL RAM

30

0030 31 BITCNT DATA 30h ; bit counter for I2C routines

0031 32 SLAVEADD DATA 31h ; slave address for I2C routines

0032 33 INPUT DATA 32h ; data recieved from the slave

0033 34 OUTPUT DATA 33h ; data to be transmitted to slave

35

0000 36 NOACK BIT 00h ; I2C no acknowledge flag

0000 37 ERR BIT 00h ; I2C error flag

38

00B4 39 LED EQU P3.4

40

41 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

42 ; BEGINNING OF CODE

---- 43 CSEG

0000 44 ORG 0000h

0000 020060 45 JMP MAIN

46 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

47 ; INT0 ISR

0003 48 ORG 0003h

0003 0533 49 INC OUTPUT

0005 32 50 RETI

51 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

52 ; MAIN PROGRAM

0060 53 ORG 0060h

0060 54 MAIN:

55

56 ; configure the UART ADuC812

0060 759852 57 MOV SCON,#52h ; configure UART for 9600baud..

0063 758920 58 MOV TMOD,#20h ; ..assuming 11.0592MHz crystal

I2CMSTR PAGE 2

0066 758DFD 59 MOV TH1,#-3

0069 D28E 60 SETB TR1

61

62 ; configure the UART ADuC824/ADuC816

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

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

65 ; MOV TH2,#0FFh

66 ; MOV TL2,#-5

67 ; MOV SCON,#52h

68 ; MOV T2CON,#34h

69

70 ; configure & enable interrupts

006B D2A8 71 SETB EX0 ; enable INT0

006D D288 72 SETB IT0 ; INT0 edge triggered

006F D2AF 73 SETB EA ; allow all the interrupts

74

75 ; initialise settings

0071 753188 76 MOV SLAVEADD,#88H ; clear RW bit

0074 75E8A8 77 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

78 ; selects master mode

0077 753300 79 MOV OUTPUT,#0 ; TX 0 as default

007A C200 80 CLR NOACK

007C C200 81 CLR ERR

82

83

007E 84 RXTXLOOP:

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

007E 1200CA 86 CALL RCVDATA ; sends start bit

87 ; sends address byte

88 ; checks acknowledge

89 ; receives byte into ACC

90 ; checks ACK

91 ; sends stop bit

92

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

0081 1200AF 94 CALL SENDDATA ; sends start bit

95 ; sends address byte

96 ; checks acknowledge

97 ; transmits ACC

98 ; checks ACK

99 ; sends stop bit

100

101 ; Check for Error message

0084 200008 102 JB ERR,SENDERR ; if error, send error message

103

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

0087 E532 105 MOV A,INPUT ; put value recieved into ACC

0089 120168 106 CALL SENDVAL ; send value recieved out the UART

008C 020094 107 JMP SKIP

108

008F 109 SENDERR:

008F 12014F 110 CALL ERROR ; send error message out the UART

0092 C200 111 CLR ERR ; clear error flag

112

0094 113 SKIP:

0094 740A 114 MOV A,#10 ; send LF+CR

0096 120154 115 CALL SENDCHAR

0099 740D 116 MOV A,#13

I2CMSTR PAGE 3

009B 120154 117 CALL SENDCHAR

118

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

009E 740A 120 MOV A, #10

00A0 120143 121 CALL DELAY

00A3 B2B4 122 CPL LED

123

124 ; Check for new OUTPUT

00A5 3098D6 125 JNB RI, RXTXLOOP ; repeat (unless UART data received)

126

127

128 ; If UART data received, then save to OUTPUT

00A8 859933 129 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00AB C298 130 CLR RI ; must clear RI

00AD 80CF 131 JMP RXTXLOOP ; back to main loop

132

133

134

135 ;====================================================================

136 ; SUBROUTINES

137 ;====================================================================

138

139 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

140 ; SENDDATA

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

142

00AF 143 SENDDATA:

144 ; send start bit

00AF 1200EC 145 CALL STARTBIT ; acquire bus and send slave address

146

147 ; send slave address

00B2 E531 148 MOV A, SLAVEADD

00B4 120104 149 CALL SENDBYTE ; sets NOACK if NACK received

150

00B7 200005 151 JB NOACK, STOPSEND ; if no acknowledge send stop

152

153 ; send OUTPUT byte

00BA E533 154 MOV A, OUTPUT

00BC 120104 155 CALL SENDBYTE ; sets NOACK if NACK received

156

00BF 157 STOPSEND:

00BF 1200F8 158 CALL STOPBIT ; sends stop bit

00C2 300004 159 JNB NOACK, SENDRET ; if slave sends no-acknowedge send error

00C5 D200 160 SETB ERR ; sets the error flag

00C7 D2EA 161 SETB I2CRS ; this resets the I2C interface

00C9 162 SENDRET:

00C9 22 163 RET

164

165

166 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

167 ; RCVDATA

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

169

00CA 170 RCVDATA:

00CA 0531 171 INC SLAVEADD ; Set RW for reception

172

173 ; send start bit

00CC 1200EC 174 CALL STARTBIT ; acquire bus and send slave address

I2CMSTR PAGE 4

175

176 ; send slave address

00CF E531 177 MOV A, SLAVEADD

00D1 120104 178 CALL SENDBYTE ; sets NOACK if NACK received

179

00D4 1531 180 DEC SLAVEADD ; put slave back in transmit mode

181

00D6 200008 182 JB NOACK, STOPRCV ; Check for slave not responding.

00D9 120141 183 CALL DELAY5 ; this lets slave get data ready

00DC 120124 184 CALL RCVBYTE ; Receive next data byte.

00DF F532 185 MOV INPUT,A ; Save data byte in buffer.

186

00E1 187 STOPRCV:

00E1 1200F8 188 CALL STOPBIT

00E4 300004 189 JNB NOACK, RCVRET ; if slave sends NACK send error

00E7 D200 190 SETB ERR ; sets the error flag

00E9 D2EA 191 SETB I2CRS ; this resets the I2C interface

00EB 192 RCVRET:

00EB 22 193 RET

194

195

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

197 ; STARTBIT

198 ; Sends the start bit to initiate an I2C communication

199

00EC 200 STARTBIT:

201

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

00EE C200 203 CLR NOACK

00F0 C2EF 204 CLR MDO ; low O/P on SDATA

00F2 120141 205 CALL DELAY5 ; delay 5 Machine cycles

00F5 C2ED 206 CLR MCO ; start bit

00F7 22 207 RET

208

209

210 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

211 ; STOPBIT

212 ; Sends the stop bit to end an I2C transmission

213

00F8 214 STOPBIT:

215

00F8 D2EE 216 SETB MDE ; to enable SDATA pin as an output

00FA C2EF 217 CLR MDO ; get SDATA ready for stop

00FC D2ED 218 SETB MCO ; set clock for stop

00FE 120141 219 CALL DELAY5

0101 D2EF 220 SETB MDO ; this is the stop bit

0103 22 221 RET

222

223

224

225 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

226 ; SENDBYTE

227 ; Send 8-bits in ACC to the slave

228

0104 229 SENDBYTE:

230

0104 753008 231 MOV BITCNT,#8 ; 8 bits in a byte

0107 D2EE 232 SETB MDE ; to enable SDATA pin as an output

I2CMSTR PAGE 5

0109 C2ED 233 CLR MCO ; make sure that the clock line is low

010B 234 SENDBIT:

010B 33 235 RLC A ; put data bit to be sent into carry

010C 92EF 236 MOV MDO,C ; put data bit on SDATA line

010E D2ED 237 SETB MCO ; clock to send bit

0110 C2ED 238 CLR MCO ; clear clock

0112 D530F6 239 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

240

0115 C2EE 241 CLR MDE ; release data line for acknowledge

0117 D2ED 242 SETB MCO ; send clock for acknowledge

0119 120141 243 CALL DELAY5

011C 30EC02 244 JNB MDI,NEXT ; this is a check for acknowledge

011F D200 245 SETB NOACK ; no acknowledge, set flag

0121 C2ED 246 NEXT: CLR MCO ; clear clock

0123 22 247 RET

248

249

250

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

252 ; RCVBYTE

253 ; receives one byte of data from an I2C slave device.

254

0124 255 RCVBYTE:

0124 753008 256 MOV BITCNT,#8 ; Set bit count.

0127 C2EE 257 CLR MDE ; to enable SDATA pin as an input

0129 C2ED 258 CLR MCO ; make sure the clock line is low

012B 259 RCVBIT:

012B D2ED 260 SETB MCO ; clock to recieve bit

012D C2ED 261 CLR MCO ; clear clock

012F A2EC 262 MOV C,MDI ; read data bit into carry.

0131 33 263 RLC A ; Rotate bit into result byte.

264

0132 D530F6 265 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

266 ; recieved byte is in the accumulator

267

0135 D2EE 268 SETB MDE ; Data pin of the master must be an..

269 ; ..output for the acknowledge

0137 D2EF 270 SETB MDO ; Send no acknowledge, last byte.

271

0139 D2ED 272 SETB MCO ; Send no-acknowledge clock.

013B 120141 273 CALL DELAY5

013E C2ED 274 CLR MCO ; clear clock

0140 22 275 RET

276

277

278 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

279 ; DELAY5

280 ; Short delay (5 machine cycles incl CALL time) for the main signals

281 ; (SCLOCK , SDATA)

282

0141 283 DELAY5:

0141 00 284 NOP

0142 22 285 RET

286

287 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

288 ; DELAY

289 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

0143 290 DELAY: ; Delays by 100ms \* A

I2CMSTR PAGE 6

291

292 ; ADuC812 100ms based on 11.0592MHz Core Clock

293 ; ADuC824 100ms based on 1.573MHz Core Clock

294

0143 FA 295 MOV R2,A ; Acc holds delay variable

0144 7BC8 296 DLY0: MOV R3,#200 ; Set up delay loop0

0146 7CE5 297 DLY1: MOV R4,#229 ; Set up delay loop1

298 ;DLY0: MOV R3,#50 ; Set up delay loop0

299 ;DLY1: MOV R4,#131 ; Set up delay loop1

0148 DCFE 300 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

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

014A DBFA 302 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

303 ; Wait for 50\*2ms

014C DAF6 304 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

305 ; wait for ACC\*100ms

014E 22 306 RET ; Return from subroutine

307

308 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

309 ; ERROR

310 ; this subroutine is run if a NACK is recieved from the slave

311

014F 312 ERROR:

313

014F 7445 314 MOV A,#45h

0151 3154 315 ACALL SENDCHAR ; send the letter E out the UART

0153 22 316 RET

317

318 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

319 ; SENDCHAR

320 ; sends ASCII value contained in A to UART

321

0154 322 SENDCHAR:

323

0154 3099FD 324 JNB TI,$ ; wait til present char gone

0157 C299 325 CLR TI ; must clear TI

0159 F599 326 MOV SBUF,A

015B 22 327 RET

328

329

330 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

331 ; HEX2ASCII

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

333 ; least significant nibble

334

015C 335 HEX2ASCII:

336

015C 540F 337 ANL A,#00Fh

015E B40A00 338 CJNE A,#00Ah,$+3

0161 4002 339 JC IO0030

0163 2407 340 ADD A,#007h

0165 2430 341 IO0030: ADD A,#'0'

0167 22 342 RET

343

344

345 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

346 ; SENDVAL

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

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

I2CMSTR PAGE 7

349

0168 350 SENDVAL:

0168 C0E0 351 PUSH ACC

016A C4 352 SWAP A

016B 315C 353 CALL HEX2ASCII

016D 3154 354 CALL SENDCHAR ; send high nibble

016F D0E0 355 POP ACC

0171 C0E0 356 PUSH ACC

0173 315C 357 CALL HEX2ASCII

0175 3154 358 CALL SENDCHAR ; send low nibble

0177 D0E0 359 POP ACC

360

0179 22 361 RET

362

363 END

364

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

I2CMSTR PAGE 8

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

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

DELAY. . . . . . . . . . . . . . C ADDR 0143H

DELAY5 . . . . . . . . . . . . . C ADDR 0141H

DLY0 . . . . . . . . . . . . . . C ADDR 0144H

DLY1 . . . . . . . . . . . . . . C ADDR 0146H

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

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

ERROR. . . . . . . . . . . . . . C ADDR 014FH

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

HEX2ASCII. . . . . . . . . . . . C ADDR 015CH

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

I2CRS. . . . . . . . . . . . . . B ADDR 00EAH PREDEFINED

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

IO0030 . . . . . . . . . . . . . C ADDR 0165H

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 0121H

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

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

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

RCVBIT . . . . . . . . . . . . . C ADDR 012BH

RCVBYTE. . . . . . . . . . . . . C ADDR 0124H

RCVDATA. . . . . . . . . . . . . C ADDR 00CAH

RCVRET . . . . . . . . . . . . . C ADDR 00EBH

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

RXTXLOOP . . . . . . . . . . . . C ADDR 007EH

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

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

SENDBIT. . . . . . . . . . . . . C ADDR 010BH

SENDBYTE . . . . . . . . . . . . C ADDR 0104H

SENDCHAR . . . . . . . . . . . . C ADDR 0154H

SENDDATA . . . . . . . . . . . . C ADDR 00AFH

SENDERR. . . . . . . . . . . . . C ADDR 008FH

SENDRET. . . . . . . . . . . . . C ADDR 00C9H

SENDVAL. . . . . . . . . . . . . C ADDR 0168H

SKIP . . . . . . . . . . . . . . C ADDR 0094H

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

STARTBIT . . . . . . . . . . . . C ADDR 00ECH

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

STOPRCV. . . . . . . . . . . . . C ADDR 00E1H

STOPSEND . . . . . . . . . . . . C ADDR 00BFH

TH1. . . . . . . . . . . . . . . D ADDR 008DH PREDEFINED

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

TMOD . . . . . . . . . . . . . . D ADDR 0089H PREDEFINED

TR1. . . . . . . . . . . . . . . B ADDR 008EH PREDEFINED