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 : ADuC824/ADuC816 (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 ;$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

43 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

44 ; BEGINNING OF CODE

---- 45 CSEG

0000 46 ORG 0000h

0000 020060 47 JMP MAIN

48

49

50 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

51 ; INT0 ISR

0003 52 ORG 0003h

0003 0533 53 INC OUTPUT

0005 32 54 RETI

55

56 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

57 ; MAIN PROGRAM

0060 58 ORG 0060h

I2CMSTR PAGE 2

0060 59 MAIN:

60

61 ; configure the UART ADuC812

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

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

64 ; MOV TH1,#-3

65 ; SETB TR1

66

67 ; configure the UART ADuC824/ADuC816

0060 75CBFF 68 MOV RCAP2H,#0FFh ; config UART for 9830baud

0063 75CAFB 69 MOV RCAP2L,#-5 ; (close enough to 9600baud)

0066 75CDFF 70 MOV TH2,#0FFh

0069 75CCFB 71 MOV TL2,#-5

006C 759852 72 MOV SCON,#52h

006F 75C834 73 MOV T2CON,#34h

74

75 ; configure & enable interrupts

0072 D2A8 76 SETB EX0 ; enable INT0

0074 D288 77 SETB IT0 ; INT0 edge triggered

0076 D2AF 78 SETB EA ; allow all the interrupts

79

80 ; initialise settings

0078 753188 81 MOV SLAVEADD,#88H ; clear RW bit

007B 75E8A8 82 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

83 ; selects master mode

007E 753300 84 MOV OUTPUT,#0 ; TX 0 as default

0081 C200 85 CLR NOACK

0083 C200 86 CLR ERR

87

0085 88 RXTXLOOP:

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

0085 1200CF 90 CALL RCVDATA ; sends start bit

91 ; sends address byte

92 ; checks acknowledge

93 ; receives byte into ACC

94 ; checks ACK

95 ; sends stop bit

96

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

0088 1200B6 98 CALL SENDDATA ; sends start bit

99 ; sends address byte

100 ; checks acknowledge

101 ; transmits ACC

102 ; checks ACK

103 ; sends stop bit

104

105 ; Check for Error message

008B 200008 106 JB ERR,SENDERR ; if error, send error message

107

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

008E E532 109 MOV A,INPUT ; put value received into ACC

0090 12015B 110 CALL SENDVAL ; send value received out the UART

0093 02009B 111 JMP SKIP

112

0096 113 SENDERR:

0096 120141 114 CALL ERROR ; send error message out the UART

0099 C200 115 CLR ERR ; clear error flag

116

I2CMSTR PAGE 3

009B 117 SKIP:

009B 740A 118 MOV A,#10 ; send LF+CR

009D 120147 119 CALL SENDCHAR

00A0 740D 120 MOV A,#13

00A2 120147 121 CALL SENDCHAR

122

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

00A5 740A 124 MOV A, #10

00A7 120135 125 CALL DELAY

00AA B2B4 126 CPL LED

127

128 ; Check for new OUTPUT

00AC 3098D6 129 JNB RI, RXTXLOOP ; repeat (unless UART data received)

130

131 ; If UART data received, then save to OUTPUT

00AF 859933 132 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00B2 C298 133 CLR RI ; must clear RI

00B4 80CF 134 JMP RXTXLOOP ; back to main loop

135

136

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

138 ; SUBROUTINES

139 ;====================================================================

140

141 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

142 ; SENDDATA

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

144

00B6 145 SENDDATA:

146 ; send start bit

00B6 1200EC 147 CALL STARTBIT ; acquire bus and send slave address

148

149 ; send slave address

00B9 E531 150 MOV A, SLAVEADD

00BB 1200FE 151 CALL SENDBYTE ; sets NOACK if NACK received

152

00BE 200005 153 JB NOACK, STOPSEND ; if no acknowledge send stop

154

155 ; send OUTPUT byte

00C1 E533 156 MOV A, OUTPUT

00C3 1200FE 157 CALL SENDBYTE ; sets NOACK if NACK received

158

00C6 159 STOPSEND:

00C6 1200F5 160 CALL STOPBIT ; sends stop bit

00C9 300002 161 JNB NOACK, SENDRET ; if slave sends NACK send error

00CC D200 162 SETB ERR ; sets the error flag

00CE 163 SENDRET:

00CE 22 164 RET

165

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

167 ; RCVDATA

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

169

00CF 170 RCVDATA:

00CF 0531 171 INC SLAVEADD ; Set RW for reception

172

173 ; send start bit

00D1 1200EC 174 CALL STARTBIT ; acquire bus and send slave address

I2CMSTR PAGE 4

175

176 ; send slave address

00D4 E531 177 MOV A, SLAVEADD

00D6 1200FE 178 CALL SENDBYTE ; sets NOACK if NACK received

179

00D9 1531 180 DEC SLAVEADD ; returns SLAVEADD to 88h (after INC)

181

00DB 200005 182 JB NOACK, STOPRCV ; Check for slave not responding.

00DE 12011B 183 CALL RCVBYTE ; Receive next data byte.

00E1 F532 184 MOV INPUT,A ; Save data byte in buffer.

185

00E3 186 STOPRCV:

00E3 1200F5 187 CALL STOPBIT

00E6 300002 188 JNB NOACK, RCVRET ; if slave sends NACK send error

00E9 D200 189 SETB ERR ; sets the error flag

00EB 190 RCVRET:

00EB 22 191 RET

192 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

193 ; STARTBIT

194 ; Sends the start bit to initiate an I2C communication

195

00EC 196 STARTBIT:

00EC D2EE 197 SETB MDE ; enable SDATA pin as an output

00EE C200 198 CLR NOACK

00F0 C2EF 199 CLR MDO ; low O/P on SDATA

00F2 C2ED 200 CLR MCO ; start bit

00F4 22 201 RET

202 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

203 ; STOPBIT

204 ; Sends the stop bit to end an I2C transmission

205

00F5 206 STOPBIT:

00F5 D2EE 207 SETB MDE ; to enable SDATA pin as an output

00F7 C2EF 208 CLR MDO ; get SDATA ready for stop

00F9 D2ED 209 SETB MCO ; set clock for stop

00FB D2EF 210 SETB MDO ; this is the stop bit

00FD 22 211 RET

212 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

213 ; SENDBYTE

214 ; Send 8-bits in ACC to the slave

00FE 215 SENDBYTE:

00FE 753008 216 MOV BITCNT,#8 ; 8 bits in a byte

0101 D2EE 217 SETB MDE ; to enable SDATA pin as an output

0103 C2ED 218 CLR MCO ; make sure that the clock line is low

0105 219 SENDBIT:

0105 33 220 RLC A ; put data bit to be sent into carry

0106 92EF 221 MOV MDO,C ; put data bit on SDATA line

0108 D2ED 222 SETB MCO ; clock to send bit

010A C2ED 223 CLR MCO ; clear clock

010C D530F6 224 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

225

010F C2EE 226 CLR MDE ; release data line for acknowledge

0111 D2ED 227 SETB MCO ; send clock for acknowledge

0113 30EC02 228 JNB MDI,NEXT ; this is a check for acknowledge

0116 D200 229 SETB NOACK ; no acknowledge, set flag

0118 C2ED 230 NEXT: CLR MCO ; clear clock

011A 22 231 RET

232 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I2CMSTR PAGE 5

233 ; RCVBYTE

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

235

011B 236 RCVBYTE:

011B 753008 237 MOV BITCNT,#8 ; Set bit count.

011E C2EE 238 CLR MDE ; to enable SDATA pin as an input

0120 C2ED 239 CLR MCO ; make sure the clock line is low

0122 240 RCVBIT:

0122 D2ED 241 SETB MCO ; clock to recieve bit

0124 C2ED 242 CLR MCO ; clear clock

0126 A2EC 243 MOV C,MDI ; read data bit into carry.

0128 33 244 RLC A ; Rotate bit into result byte.

245

0129 D530F6 246 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

247 ; recieved byte is in the accumulator

248

012C D2EE 249 SETB MDE ; Data pin =Output for NACK

012E D2EF 250 SETB MDO ; Send NACK (always send NACK for

251 ; last byte in transmission)

0130 D2ED 252 SETB MCO ; Send NACK clock.

0132 C2ED 253 CLR MCO

0134 22 254 RET

255

256 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

257 ; DELAY

258 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

0135 259 DELAY: ; Delays by 100ms \* A

260

261 ; ADuC812 100ms based on 11.0592MHz Core Clock

262 ; ADuC824 100ms based on 1.573MHz Core Clock

263

0135 FA 264 MOV R2,A ; Acc holds delay variable

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

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

0136 7B32 267 DLY0: MOV R3,#50 ; Set up delay loop0

0138 7C83 268 DLY1: MOV R4,#131 ; Set up delay loop1

013A DCFE 269 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

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

013C DBFA 271 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

272 ; Wait for 50\*2ms

013E DAF6 273 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

274 ; wait for ACC\*100ms

0140 22 275 RET ; Return from subroutine

276 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

277 ; ERROR

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

279

0141 280 ERROR:

0141 7445 281 MOV A,#45h

0143 120147 282 CALL SENDCHAR ; send the letter E out the UART

0146 22 283 RET

284 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

285 ; SENDCHAR

286 ; sends ASCII value contained in A to UART

287

0147 288 SENDCHAR:

0147 3099FD 289 JNB TI,$ ; wait til present char gone

014A C299 290 CLR TI ; must clear TI

I2CMSTR PAGE 6

014C F599 291 MOV SBUF,A

014E 22 292 RET

293 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

294 ; HEX2ASCII

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

296 ; least significant nibble

297

014F 298 HEX2ASCII:

014F 540F 299 ANL A,#00Fh

0151 B40A00 300 CJNE A,#00Ah,$+3

0154 4002 301 JC IO0030

0156 2407 302 ADD A,#007h

0158 2430 303 IO0030: ADD A,#'0'

015A 22 304 RET

305 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

306 ; SENDVAL

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

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

309

015B 310 SENDVAL:

015B C0E0 311 PUSH ACC

015D C4 312 SWAP A

015E 314F 313 CALL HEX2ASCII

0160 3147 314 CALL SENDCHAR ; send high nibble

0162 D0E0 315 POP ACC

0164 C0E0 316 PUSH ACC

0166 314F 317 CALL HEX2ASCII

0168 3147 318 CALL SENDCHAR ; send low nibble

016A D0E0 319 POP ACC

016C 22 320 RET

321 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

322

323 END

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

I2CMSTR PAGE 7

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

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

DELAY. . . . . . . . . . . . . . C ADDR 0135H

DLY0 . . . . . . . . . . . . . . C ADDR 0136H

DLY1 . . . . . . . . . . . . . . C ADDR 0138H

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

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

ERROR. . . . . . . . . . . . . . C ADDR 0141H

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

HEX2ASCII. . . . . . . . . . . . C ADDR 014FH

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

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

IO0030 . . . . . . . . . . . . . C ADDR 0158H

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 0118H

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 0122H

RCVBYTE. . . . . . . . . . . . . C ADDR 011BH

RCVDATA. . . . . . . . . . . . . C ADDR 00CFH

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

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

RXTXLOOP . . . . . . . . . . . . C ADDR 0085H

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

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

SENDBIT. . . . . . . . . . . . . C ADDR 0105H

SENDBYTE . . . . . . . . . . . . C ADDR 00FEH

SENDCHAR . . . . . . . . . . . . C ADDR 0147H

SENDDATA . . . . . . . . . . . . C ADDR 00B6H

SENDERR. . . . . . . . . . . . . C ADDR 0096H

SENDRET. . . . . . . . . . . . . C ADDR 00CEH

SENDVAL. . . . . . . . . . . . . C ADDR 015BH

SKIP . . . . . . . . . . . . . . C ADDR 009BH

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

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

STOPBIT. . . . . . . . . . . . . C ADDR 00F5H

STOPRCV. . . . . . . . . . . . . C ADDR 00E3H

STOPSEND . . . . . . . . . . . . C ADDR 00C6H

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

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

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

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