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

25 ;$MOD816

26 ;$MOD824

27 $MOD834

28

29 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

30 ; DEFINE VARIABLES IN INTERNAL RAM

31

0030 32 BITCNT DATA 30h ; bit counter for I2C routines

0031 33 SLAVEADD DATA 31h ; slave address for I2C routines

0032 34 INPUT DATA 32h ; data recieved from the slave

0033 35 OUTPUT DATA 33h ; data to be transmitted to slave

36

0000 37 NOACK BIT 00h ; I2C no acknowledge flag

0000 38 ERR BIT 00h ; I2C error flag

39

00B4 40 LED EQU P3.4

41

42

43

44 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

45 ; BEGINNING OF CODE

---- 46 CSEG

0000 47 ORG 0000h

0000 020060 48 JMP MAIN

49

50

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

52 ; INT0 ISR

0003 53 ORG 0003h

0003 0533 54 INC OUTPUT

0005 32 55 RETI

56

57 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

58 ; MAIN PROGRAM

I2CMSTR PAGE 2

0060 59 ORG 0060h

0060 60 MAIN:

61

62 ; configure the UART ADuC812

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

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

65 ; MOV TH1,#-3

66 ; SETB TR1

67

68 ; configure the UART ADuC824/ADuC816

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

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

71 ; MOV TH2,#0FFh

72 ; MOV TL2,#-5

73 ; MOV SCON,#52h

74 ; MOV T2CON,#34h

75

76 ; configure UART for 9600 using Timer3

0060 759E82 77 MOV T3CON,#82h

0063 759D12 78 MOV T3FD,#12h

0066 759852 79 MOV SCON,#52h

80

81 ; configure & enable interrupts

0069 D2A8 82 SETB EX0 ; enable INT0

006B D288 83 SETB IT0 ; INT0 edge triggered

006D D2AF 84 SETB EA ; allow all the interrupts

85

86 ; initialise settings

006F 753188 87 MOV SLAVEADD,#88H ; clear RW bit

0072 75E8A8 88 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

89 ; selects master mode

0075 753300 90 MOV OUTPUT,#0 ; TX 0 as default

0078 C200 91 CLR NOACK

007A C200 92 CLR ERR

93

007C 94 RXTXLOOP:

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

007C 1200C6 96 CALL RCVDATA ; sends start bit

97 ; sends address byte

98 ; checks acknowledge

99 ; receives byte into ACC

100 ; checks ACK

101 ; sends stop bit

102

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

007F 1200AD 104 CALL SENDDATA ; sends start bit

105 ; sends address byte

106 ; checks acknowledge

107 ; transmits ACC

108 ; checks ACK

109 ; sends stop bit

110

111 ; Check for Error message

0082 200008 112 JB ERR,SENDERR ; if error, send error message

113

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

0085 E532 115 MOV A,INPUT ; put value received into ACC

0087 120152 116 CALL SENDVAL ; send value received out the UART

I2CMSTR PAGE 3

008A 020092 117 JMP SKIP

118

008D 119 SENDERR:

008D 120138 120 CALL ERROR ; send error message out the UART

0090 C200 121 CLR ERR ; clear error flag

122

0092 123 SKIP:

0092 740A 124 MOV A,#10 ; send LF+CR

0094 12013E 125 CALL SENDCHAR

0097 740D 126 MOV A,#13

0099 12013E 127 CALL SENDCHAR

128

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

009C 740A 130 MOV A, #10

009E 12012C 131 CALL DELAY

00A1 B2B4 132 CPL LED

133

134 ; Check for new OUTPUT

00A3 3098D6 135 JNB RI, RXTXLOOP ; repeat (unless UART data received)

136

137 ; If UART data received, then save to OUTPUT

00A6 859933 138 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00A9 C298 139 CLR RI ; must clear RI

00AB 80CF 140 JMP RXTXLOOP ; back to main loop

141

142

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

144 ; SUBROUTINES

145 ;====================================================================

146

147 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

148 ; SENDDATA

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

150

00AD 151 SENDDATA:

152 ; send start bit

00AD 1200E3 153 CALL STARTBIT ; acquire bus and send slave address

154

155 ; send slave address

00B0 E531 156 MOV A, SLAVEADD

00B2 1200F5 157 CALL SENDBYTE ; sets NOACK if NACK received

158

00B5 200005 159 JB NOACK, STOPSEND ; if no acknowledge send stop

160

161 ; send OUTPUT byte

00B8 E533 162 MOV A, OUTPUT

00BA 1200F5 163 CALL SENDBYTE ; sets NOACK if NACK received

164

00BD 165 STOPSEND:

00BD 1200EC 166 CALL STOPBIT ; sends stop bit

00C0 300002 167 JNB NOACK, SENDRET ; if slave sends NACK send error

00C3 D200 168 SETB ERR ; sets the error flag

00C5 169 SENDRET:

00C5 22 170 RET

171

172 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

173 ; RCVDATA

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

I2CMSTR PAGE 4

175

00C6 176 RCVDATA:

00C6 0531 177 INC SLAVEADD ; Set RW for reception

178

179 ; send start bit

00C8 1200E3 180 CALL STARTBIT ; acquire bus and send slave address

181

182 ; send slave address

00CB E531 183 MOV A, SLAVEADD

00CD 1200F5 184 CALL SENDBYTE ; sets NOACK if NACK received

185

00D0 1531 186 DEC SLAVEADD ; returns SLAVEADD to 88h (after INC)

187

00D2 200005 188 JB NOACK, STOPRCV ; Check for slave not responding.

00D5 120112 189 CALL RCVBYTE ; Receive next data byte.

00D8 F532 190 MOV INPUT,A ; Save data byte in buffer.

191

00DA 192 STOPRCV:

00DA 1200EC 193 CALL STOPBIT

00DD 300002 194 JNB NOACK, RCVRET ; if slave sends NACK send error

00E0 D200 195 SETB ERR ; sets the error flag

00E2 196 RCVRET:

00E2 22 197 RET

198 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

199 ; STARTBIT

200 ; Sends the start bit to initiate an I2C communication

201

00E3 202 STARTBIT:

00E3 D2EE 203 SETB MDE ; enable SDATA pin as an output

00E5 C200 204 CLR NOACK

00E7 C2EF 205 CLR MDO ; low O/P on SDATA

00E9 C2ED 206 CLR MCO ; start bit

00EB 22 207 RET

208 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

209 ; STOPBIT

210 ; Sends the stop bit to end an I2C transmission

211

00EC 212 STOPBIT:

00EC D2EE 213 SETB MDE ; to enable SDATA pin as an output

00EE C2EF 214 CLR MDO ; get SDATA ready for stop

00F0 D2ED 215 SETB MCO ; set clock for stop

00F2 D2EF 216 SETB MDO ; this is the stop bit

00F4 22 217 RET

218 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

219 ; SENDBYTE

220 ; Send 8-bits in ACC to the slave

00F5 221 SENDBYTE:

00F5 753008 222 MOV BITCNT,#8 ; 8 bits in a byte

00F8 D2EE 223 SETB MDE ; to enable SDATA pin as an output

00FA C2ED 224 CLR MCO ; make sure that the clock line is low

00FC 225 SENDBIT:

00FC 33 226 RLC A ; put data bit to be sent into carry

00FD 92EF 227 MOV MDO,C ; put data bit on SDATA line

00FF D2ED 228 SETB MCO ; clock to send bit

0101 C2ED 229 CLR MCO ; clear clock

0103 D530F6 230 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

231

0106 C2EE 232 CLR MDE ; release data line for acknowledge

I2CMSTR PAGE 5

0108 D2ED 233 SETB MCO ; send clock for acknowledge

010A 30EC02 234 JNB MDI,NEXT ; this is a check for acknowledge

010D D200 235 SETB NOACK ; no acknowledge, set flag

010F C2ED 236 NEXT: CLR MCO ; clear clock

0111 22 237 RET

238 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

239 ; RCVBYTE

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

241

0112 242 RCVBYTE:

0112 753008 243 MOV BITCNT,#8 ; Set bit count.

0115 C2EE 244 CLR MDE ; to enable SDATA pin as an input

0117 C2ED 245 CLR MCO ; make sure the clock line is low

0119 246 RCVBIT:

0119 D2ED 247 SETB MCO ; clock to recieve bit

011B C2ED 248 CLR MCO ; clear clock

011D A2EC 249 MOV C,MDI ; read data bit into carry.

011F 33 250 RLC A ; Rotate bit into result byte.

251

0120 D530F6 252 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

253 ; recieved byte is in the accumulator

254

0123 D2EE 255 SETB MDE ; Data pin =Output for NACK

0125 D2EF 256 SETB MDO ; Send NACK (always send NACK for

257 ; last byte in transmission)

0127 D2ED 258 SETB MCO ; Send NACK clock.

0129 C2ED 259 CLR MCO

012B 22 260 RET

261

262 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

263 ; DELAY

264 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

012C 265 DELAY: ; Delays by 100ms \* A

266

267 ; ADuC812 100ms based on 11.0592MHz Core Clock

268 ; ADuC824 100ms based on 1.573MHz Core Clock

269

012C FA 270 MOV R2,A ; Acc holds delay variable

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

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

012D 7B32 273 DLY0: MOV R3,#50 ; Set up delay loop0

012F 7C83 274 DLY1: MOV R4,#131 ; Set up delay loop1

0131 DCFE 275 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

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

0133 DBFA 277 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

278 ; Wait for 50\*2ms

0135 DAF6 279 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

280 ; wait for ACC\*100ms

0137 22 281 RET ; Return from subroutine

282 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

283 ; ERROR

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

285

0138 286 ERROR:

0138 7445 287 MOV A,#45h

013A 12013E 288 CALL SENDCHAR ; send the letter E out the UART

013D 22 289 RET

290 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I2CMSTR PAGE 6

291 ; SENDCHAR

292 ; sends ASCII value contained in A to UART

293

013E 294 SENDCHAR:

013E 3099FD 295 JNB TI,$ ; wait til present char gone

0141 C299 296 CLR TI ; must clear TI

0143 F599 297 MOV SBUF,A

0145 22 298 RET

299 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

300 ; HEX2ASCII

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

302 ; least significant nibble

303

0146 304 HEX2ASCII:

0146 540F 305 ANL A,#00Fh

0148 B40A00 306 CJNE A,#00Ah,$+3

014B 4002 307 JC IO0030

014D 2407 308 ADD A,#007h

014F 2430 309 IO0030: ADD A,#'0'

0151 22 310 RET

311 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

312 ; SENDVAL

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

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

315

0152 316 SENDVAL:

0152 C0E0 317 PUSH ACC

0154 C4 318 SWAP A

0155 3146 319 CALL HEX2ASCII

0157 313E 320 CALL SENDCHAR ; send high nibble

0159 D0E0 321 POP ACC

015B C0E0 322 PUSH ACC

015D 3146 323 CALL HEX2ASCII

015F 313E 324 CALL SENDCHAR ; send low nibble

0161 D0E0 325 POP ACC

0163 22 326 RET

327 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

328

329 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