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 : ADuC831

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 $MOD831

25

26 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

27 ; DEFINE VARIABLES IN INTERNAL RAM

28

0030 29 BITCNT DATA 30h ; bit counter for I2C routines

0031 30 SLAVEADD DATA 31h ; slave address for I2C routines

0032 31 INPUT DATA 32h ; data recieved from the slave

0033 32 OUTPUT DATA 33h ; data to be transmitted to slave

33

0000 34 NOACK BIT 00h ; I2C no acknowledge flag

0000 35 ERR BIT 00h ; I2C error flag

36

00B4 37 LED EQU P3.4

38

39

40

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

42 ; BEGINNING OF CODE

---- 43 CSEG

0000 44 ORG 0000h

0000 020060 45 JMP MAIN

46

47

48 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

49 ; INT0 ISR

0003 50 ORG 0003h

0003 0533 51 INC OUTPUT

0005 32 52 RETI

53

54 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

55 ; MAIN PROGRAM

0060 56 ORG 0060h

0060 57 MAIN:

58

I2CMSTR PAGE 2

0060 759E85 59 MOV T3CON,#085h

0063 759D08 60 MOV T3FD,#08h

0066 759852 61 MOV SCON,#52h

62

63 ; configure & enable interrupts

0069 D2A8 64 SETB EX0 ; enable INT0

006B D288 65 SETB IT0 ; INT0 edge triggered

006D D2AF 66 SETB EA ; allow all the interrupts

67

68 ; initialise settings

006F 753188 69 MOV SLAVEADD,#88H ; clear RW bit

0072 75E8A8 70 MOV I2CCON,#0A8h ; sets SDATA & SCLOCK, and

71 ; selects master mode

0075 753300 72 MOV OUTPUT,#0 ; TX 0 as default

0078 C200 73 CLR NOACK

007A C200 74 CLR ERR

75

007C 76 RXTXLOOP:

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

007C 1200C6 78 CALL RCVDATA ; sends start bit

79 ; sends address byte

80 ; checks acknowledge

81 ; receives byte into ACC

82 ; checks ACK

83 ; sends stop bit

84

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

007F 1200AD 86 CALL SENDDATA ; sends start bit

87 ; sends address byte

88 ; checks acknowledge

89 ; transmits ACC

90 ; checks ACK

91 ; sends stop bit

92

93 ; Check for Error message

0082 200008 94 JB ERR,SENDERR ; if error, send error message

95

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

0085 E532 97 MOV A,INPUT ; put value received into ACC

0087 120152 98 CALL SENDVAL ; send value received out the UART

008A 020092 99 JMP SKIP

100

008D 101 SENDERR:

008D 120138 102 CALL ERROR ; send error message out the UART

0090 C200 103 CLR ERR ; clear error flag

104

0092 105 SKIP:

0092 740A 106 MOV A,#10 ; send LF+CR

0094 12013E 107 CALL SENDCHAR

0097 740D 108 MOV A,#13

0099 12013E 109 CALL SENDCHAR

110

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

009C 740A 112 MOV A, #10

009E 12012C 113 CALL DELAY

00A1 B2B4 114 CPL LED

115

116 ; Check for new OUTPUT

I2CMSTR PAGE 3

00A3 3098D6 117 JNB RI, RXTXLOOP ; repeat (unless UART data received)

118

119 ; If UART data received, then save to OUTPUT

00A6 859933 120 MOV OUTPUT,SBUF ; update OUTPUT byte to new value

00A9 C298 121 CLR RI ; must clear RI

00AB 80CF 122 JMP RXTXLOOP ; back to main loop

123

124

125 ;====================================================================

126 ; SUBROUTINES

127 ;====================================================================

128

129 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

130 ; SENDDATA

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

132

00AD 133 SENDDATA:

134 ; send start bit

00AD 1200E3 135 CALL STARTBIT ; acquire bus and send slave address

136

137 ; send slave address

00B0 E531 138 MOV A, SLAVEADD

00B2 1200F5 139 CALL SENDBYTE ; sets NOACK if NACK received

140

00B5 200005 141 JB NOACK, STOPSEND ; if no acknowledge send stop

142

143 ; send OUTPUT byte

00B8 E533 144 MOV A, OUTPUT

00BA 1200F5 145 CALL SENDBYTE ; sets NOACK if NACK received

146

00BD 147 STOPSEND:

00BD 1200EC 148 CALL STOPBIT ; sends stop bit

00C0 300002 149 JNB NOACK, SENDRET ; if slave sends NACK send error

00C3 D200 150 SETB ERR ; sets the error flag

00C5 151 SENDRET:

00C5 22 152 RET

153

154 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

155 ; RCVDATA

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

157

00C6 158 RCVDATA:

00C6 0531 159 INC SLAVEADD ; Set RW for reception

160

161 ; send start bit

00C8 1200E3 162 CALL STARTBIT ; acquire bus and send slave address

163

164 ; send slave address

00CB E531 165 MOV A, SLAVEADD

00CD 1200F5 166 CALL SENDBYTE ; sets NOACK if NACK received

167

00D0 1531 168 DEC SLAVEADD ; returns SLAVEADD to 88h (after INC)

169

00D2 200005 170 JB NOACK, STOPRCV ; Check for slave not responding.

00D5 120112 171 CALL RCVBYTE ; Receive next data byte.

00D8 F532 172 MOV INPUT,A ; Save data byte in buffer.

173

00DA 174 STOPRCV:

I2CMSTR PAGE 4

00DA 1200EC 175 CALL STOPBIT

00DD 300002 176 JNB NOACK, RCVRET ; if slave sends NACK send error

00E0 D200 177 SETB ERR ; sets the error flag

00E2 178 RCVRET:

00E2 22 179 RET

180 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

181 ; STARTBIT

182 ; Sends the start bit to initiate an I2C communication

183

00E3 184 STARTBIT:

00E3 D2EE 185 SETB MDE ; enable SDATA pin as an output

00E5 C200 186 CLR NOACK

00E7 C2EF 187 CLR MDO ; low O/P on SDATA

00E9 C2ED 188 CLR MCO ; start bit

00EB 22 189 RET

190 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

191 ; STOPBIT

192 ; Sends the stop bit to end an I2C transmission

193

00EC 194 STOPBIT:

00EC D2EE 195 SETB MDE ; to enable SDATA pin as an output

00EE C2EF 196 CLR MDO ; get SDATA ready for stop

00F0 D2ED 197 SETB MCO ; set clock for stop

00F2 D2EF 198 SETB MDO ; this is the stop bit

00F4 22 199 RET

200 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

201 ; SENDBYTE

202 ; Send 8-bits in ACC to the slave

00F5 203 SENDBYTE:

00F5 753008 204 MOV BITCNT,#8 ; 8 bits in a byte

00F8 D2EE 205 SETB MDE ; to enable SDATA pin as an output

00FA C2ED 206 CLR MCO ; make sure that the clock line is low

00FC 207 SENDBIT:

00FC 33 208 RLC A ; put data bit to be sent into carry

00FD 92EF 209 MOV MDO,C ; put data bit on SDATA line

00FF D2ED 210 SETB MCO ; clock to send bit

0101 C2ED 211 CLR MCO ; clear clock

0103 D530F6 212 DJNZ BITCNT,SENDBIT ; jump back and send all eight bits

213

0106 C2EE 214 CLR MDE ; release data line for acknowledge

0108 D2ED 215 SETB MCO ; send clock for acknowledge

010A 30EC02 216 JNB MDI,NEXT ; this is a check for acknowledge

010D D200 217 SETB NOACK ; no acknowledge, set flag

010F C2ED 218 NEXT: CLR MCO ; clear clock

0111 22 219 RET

220 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

221 ; RCVBYTE

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

223

0112 224 RCVBYTE:

0112 753008 225 MOV BITCNT,#8 ; Set bit count.

0115 C2EE 226 CLR MDE ; to enable SDATA pin as an input

0117 C2ED 227 CLR MCO ; make sure the clock line is low

0119 228 RCVBIT:

0119 D2ED 229 SETB MCO ; clock to recieve bit

011B C2ED 230 CLR MCO ; clear clock

011D A2EC 231 MOV C,MDI ; read data bit into carry.

011F 33 232 RLC A ; Rotate bit into result byte.

I2CMSTR PAGE 5

233

0120 D530F6 234 DJNZ BITCNT,RCVBIT ; Repeat until all bits received.

235 ; recieved byte is in the accumulator

236

0123 D2EE 237 SETB MDE ; Data pin =Output for NACK

0125 D2EF 238 SETB MDO ; Send NACK (always send NACK for

239 ; last byte in transmission)

0127 D2ED 240 SETB MCO ; Send NACK clock.

0129 C2ED 241 CLR MCO

012B 22 242 RET

243

244 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

245 ; DELAY

246 ; DELAY ROUTINE FOR THE ADuC812/ADuC816/ADuC824

012C 247 DELAY: ; Delays by 100ms \* A

248

249 ; ADuC812 100ms based on 11.0592MHz Core Clock

250 ; ADuC824 100ms based on 1.573MHz Core Clock

251

012C FA 252 MOV R2,A ; Acc holds delay variable

012D 7BC8 253 DLY0: MOV R3,#200 ; Set up delay loop0

012F 7CE5 254 DLY1: MOV R4,#229 ; Set up delay loop1

0131 DCFE 255 DJNZ R4,$ ; Dec R4 & Jump here until R4 is 0

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

0133 DBFA 257 DJNZ R3,DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

258 ; Wait for 50\*2ms

0135 DAF6 259 DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

260 ; wait for ACC\*100ms

0137 22 261 RET ; Return from subroutine

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

263 ; ERROR

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

265

0138 266 ERROR:

0138 7445 267 MOV A,#45h

013A 12013E 268 CALL SENDCHAR ; send the letter E out the UART

013D 22 269 RET

270 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

271 ; SENDCHAR

272 ; sends ASCII value contained in A to UART

273

013E 274 SENDCHAR:

013E 3099FD 275 JNB TI,$ ; wait til present char gone

0141 C299 276 CLR TI ; must clear TI

0143 F599 277 MOV SBUF,A

0145 22 278 RET

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

280 ; HEX2ASCII

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

282 ; least significant nibble

283

0146 284 HEX2ASCII:

0146 540F 285 ANL A,#00Fh

0148 B40A00 286 CJNE A,#00Ah,$+3

014B 4002 287 JC IO0030

014D 2407 288 ADD A,#007h

014F 2430 289 IO0030: ADD A,#'0'

0151 22 290 RET

I2CMSTR PAGE 6

291 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

292 ; SENDVAL

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

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

295

0152 296 SENDVAL:

0152 C0E0 297 PUSH ACC

0154 C4 298 SWAP A

0155 3146 299 CALL HEX2ASCII

0157 313E 300 CALL SENDCHAR ; send high nibble

0159 D0E0 301 POP ACC

015B C0E0 302 PUSH ACC

015D 3146 303 CALL HEX2ASCII

015F 313E 304 CALL SENDCHAR ; send low nibble

0161 D0E0 305 POP ACC

0163 22 306 RET

307 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

308

309 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