ADCSINGL PAGE 1

1 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 ;

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

4 ;

5 ; Date : October 2003

6 ;

7 ; File : ADCsingl.asm

8 ;

9 ; Hardware : ADuC842/ADuC843

10 ;

11 ; Description : Performs repeated single ADC conversions and moves

12 ; results to UART. Sets the red LED on the eval

13 ; board upon completion of each conversion. A new

14 ; conversion is innitiated every 200ms.

15 ; All rate calculations assume an 2.097152MHz Mclk.

16 ;

17 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18

19 $MOD842 ; Use 8052&ADuC832 predefined symbols

20

00B4 21 LED EQU P3.4 ; P3.4 drives red LED on eval board

0000 22 CHAN EQU 0 ; convert this ADC input channel..

23 ; ..chan values can be 0 thru 6

24 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

25 ; BEGINNING OF CODE

---- 26 CSEG

27

0000 28 ORG 0000h

29

0000 02004B 30 JMP MAIN ; jump to main program

31 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

32

33

34 ;====================================================================

35 ; MAIN PROGRAM

004B 36 ORG 004Bh

37

004B 38 MAIN:

39

40 ; Set up UART

004B 759E83 41 MOV T3CON,#083H

004E 759D2D 42 MOV T3FD,#02DH

0051 759852 43 MOV SCON,#52h

44

45 ; PRECONFIGURE...

46

0054 75EFAC 47 MOV ADCCON1,#0ACh ; power up ADC

0057 75D800 48 MOV ADCCON2,#CHAN ; select channel to convert

49

50 ; PERFORM REPEATED SINGLE CONVERSIONS...

51

005A B2B4 52 AGAIN: CPL LED ; turn the LED off

005C 7414 53 MOV A,#020 ; Delay length

005E 120072 54 CALL DELAY ; delay 200ms

55

0061 D2DC 56 SETB SCONV ; innitiate single ADC conversion

57 ; ADC ISR is called upon completion

0063 30DFFD 58 JNB ADCI,$

ADCSINGL PAGE 2

59

0066 E5DA 60 MOV A,ADCDATAH

0068 12009E 61 CALL SENDVAL

006B E5D9 62 MOV A,ADCDATAL

006D 12009E 63 CALL SENDVAL

64

0070 80E8 65 JMP AGAIN ; repeat

66

67 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

68 ; SUBROUTINE

0072 69 DELAY: ; Delays by 10ms \* A

70 ; 25mSec based on 2.09MHZ

71 ; Core Clock

72 ; i.e. default ADuC842 Clock

73

0072 F9 74 MOV R1,A ; Acc holds delay variable (1 clock)

0073 7A1B 75 DLY0: MOV R2,#01Bh ; Set up delay loop0 (2 clocks)

0075 7BFF 76 DLY1: MOV R3,#0FFh ; Set up delay loop1 (2 clocks)

0077 DBFE 77 DJNZ R3,$ ; Dec R3 & Jump here until R3 is 0 (3 clocks)

0079 DAFA 78 DJNZ R2,DLY1 ; Dec R2 & Jump DLY1 until R2 is 0 (3 clocks)

007B D9F6 79 DJNZ R1,DLY0 ; Dec R1 & Jump DLY0 until R1 is 0 (3 clocks)

007D 22 80 RET ; Return from subroutine

81

82

83 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

84

85

=1 86 $INCLUDE(UARTIO.ASM)

=1 87 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 88 ;

=1 89 ; Author : ADI - Apps www.analog.com/MicroConverter

=1 90 ;

=1 91 ; Date : 12 October 1999

=1 92 ;

=1 93 ; File : UARTIO.hex

=1 94 ;

=1 95 ; Hardware : any 8051 based microcontroller or MicroConverter

=1 96 ;

=1 97 ; Description : standard UART I/O subroutines. total size of this

=1 98 ; code when assembled is 155 bytes. routines for use

=1 99 ; external to this file are:

=1 100 ;

=1 101 ; SENDSTRING - sends a string of characters

=1 102 ; SENDCHAR - sends a single character

=1 103 ; SENDVAL - sends a byte as 2 ASCII characters

=1 104 ; HEX2ASCII - converts from HEX to ASCII

=1 105 ; ASCII2HEX - converts from ASCII to HEX

=1 106 ; GETCHAR - gets a single character

=1 107 ; GETVAL - gets a byte as 2 ASCII characters

=1 108 ;

=1 109 ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=1 110

=1 111 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 112 ; SENDSTRING

=1 113

007E =1 114 SENDSTRING: ; sends ASCII string to UART starting at location

=1 115 ; DPTR and ending with a null (0) value

=1 116

ADCSINGL PAGE 3

007E C0E0 =1 117 PUSH ACC

0080 C0F0 =1 118 PUSH B

0082 E4 =1 119 CLR A

0083 F5F0 =1 120 MOV B,A

0085 E5F0 =1 121 IO0010: MOV A,B

0087 05F0 =1 122 INC B

0089 93 =1 123 MOVC A,@A+DPTR

008A 6005 =1 124 JZ IO0020

008C 120096 =1 125 CALL SENDCHAR

008F 80F4 =1 126 JMP IO0010

0091 D0F0 =1 127 IO0020: POP B

0093 D0E0 =1 128 POP ACC

=1 129

0095 22 =1 130 RET

=1 131

=1 132 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 133 ; SENDCHAR

=1 134

0096 =1 135 SENDCHAR: ; sends ASCII value contained in A to UART

=1 136

0096 3099FD =1 137 JNB TI,$ ; wait til present char gone

0099 C299 =1 138 CLR TI ; must clear TI

009B F599 =1 139 MOV SBUF,A

=1 140

009D 22 =1 141 RET

=1 142

=1 143 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 144 ; SENDVAL

=1 145

009E =1 146 SENDVAL: ; converts the hex value of A into two ASCII chars,

=1 147 ; and then spits these two characters up the UART.

=1 148 ; does not change the value of A.

=1 149

009E C0E0 =1 150 PUSH ACC

00A0 C4 =1 151 SWAP A

00A1 1200B2 =1 152 CALL HEX2ASCII

00A4 1196 =1 153 CALL SENDCHAR ; send high nibble

00A6 D0E0 =1 154 POP ACC

00A8 C0E0 =1 155 PUSH ACC

00AA 1200B2 =1 156 CALL HEX2ASCII

00AD 1196 =1 157 CALL SENDCHAR ; send low nibble

00AF D0E0 =1 158 POP ACC

=1 159

00B1 22 =1 160 RET

=1 161

=1 162 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 163 ; HEX2ASCII

=1 164

00B2 =1 165 HEX2ASCII: ; converts A into the hex character representing the

=1 166 ; value of A's least significant nibble

=1 167

00B2 540F =1 168 ANL A,#00Fh

00B4 B40A00 =1 169 CJNE A,#00Ah,$+3

00B7 4002 =1 170 JC IO0030

00B9 2407 =1 171 ADD A,#007h

00BB 2430 =1 172 IO0030: ADD A,#'0'

=1 173

00BD 22 =1 174 RET

ADCSINGL PAGE 4

=1 175

=1 176 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 177 ; ASCII2HEX

=1 178

00BE =1 179 ASCII2HEX: ; converts A from an ASCII digit ('0'-'9' or 'A'-'F')

=1 180 ; into the corresponding number (0-15). returns C=1

=1 181 ; when input is other than an ASCII digit,

=1 182 ; indicating invalid output (returned as 255).

=1 183

00BE C3 =1 184 CLR C

00BF 9430 =1 185 SUBB A,#'0'

00C1 B40A00 =1 186 CJNE A,#10,$+3

00C4 401B =1 187 JC IO0050 ; if '0'<=char<='9', return OK

00C6 B41100 =1 188 CJNE A,#17,$+3

00C9 4013 =1 189 JC IO0040 ; if '9'<char<'A', return FAIL

00CB 9407 =1 190 SUBB A,#7

00CD B41000 =1 191 CJNE A,#10h,$+3

00D0 400F =1 192 JC IO0050 ; if 'A'<=char<='F', return OK

00D2 B42A00 =1 193 CJNE A,#42,$+3

00D5 4007 =1 194 JC IO0040 ; if 'F'<char<'a', return FAIL

00D7 9420 =1 195 SUBB A,#20h

00D9 B41000 =1 196 CJNE A,#10h,$+3

00DC 4003 =1 197 JC IO0050 ; if 'a'<=char<='f', return OK..

=1 198

00DE C3 =1 199 IO0040: CLR C ; ..else return FAIL

00DF 74FF =1 200 MOV A,#0FFh

=1 201

00E1 B3 =1 202 IO0050: CPL C

00E2 22 =1 203 RET

=1 204

=1 205 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 206 ; GETCHAR

=1 207

00E3 =1 208 GETCHAR: ; waits for a single ASCII character to be received

=1 209 ; by the UART. places this character into A.

=1 210

00E3 3098FD =1 211 JNB RI,$

00E6 E599 =1 212 MOV A,SBUF

00E8 C298 =1 213 CLR RI

=1 214

00EA 22 =1 215 RET

=1 216

=1 217 ;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

=1 218 ; GETVAL

=1 219

00EB =1 220 GETVAL: ; waits for two ASCII hex digits to be received by

=1 221 ; the UART. returns the hex value in A.

=1 222

00EB C0F0 =1 223 PUSH B

00ED C000 =1 224 PUSH 0

00EF C298 =1 225 IO0060: CLR RI

00F1 11E3 =1 226 CALL GETCHAR ; first nibble

00F3 F500 =1 227 MOV 0,A ; store received char

00F5 11BE =1 228 CALL ASCII2HEX

00F7 40F6 =1 229 JC IO0060 ; if not '0' thru 'F', don't accept

00F9 C4 =1 230 SWAP A ; swap nibbles

00FA F5F0 =1 231 MOV B,A ; store nibble in B

00FC E500 =1 232 MOV A,0 ; echo received char

ADCSINGL PAGE 5

00FE 1196 =1 233 CALL SENDCHAR

0100 C298 =1 234 IO0070: CLR RI

0102 11E3 =1 235 CALL GETCHAR ; second nibble

0104 F500 =1 236 MOV 0,A ; store received char

0106 11BE =1 237 CALL ASCII2HEX

0108 40F6 =1 238 JC IO0070 ; if not '0' thru 'F', don't accept

010A 45F0 =1 239 ORL A,B ; combine nibbles

010C F5F0 =1 240 MOV B,A ; store results in B

010E E500 =1 241 MOV A,0 ; echo received char

0110 1196 =1 242 CALL SENDCHAR

0112 E5F0 =1 243 MOV A,B ; final result

0114 D000 =1 244 POP 0

0116 D0F0 =1 245 POP B

=1 246

0118 22 =1 247 RET

=1 248

249 END

VERSION 1.2h ASSEMBLY COMPLETE, 0 ERRORS FOUND

ADCSINGL PAGE 6

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

ADCCON1. . . . . . . . . . . . . D ADDR 00EFH PREDEFINED

ADCCON2. . . . . . . . . . . . . D ADDR 00D8H PREDEFINED

ADCDATAH . . . . . . . . . . . . D ADDR 00DAH PREDEFINED

ADCDATAL . . . . . . . . . . . . D ADDR 00D9H PREDEFINED

ADCI . . . . . . . . . . . . . . B ADDR 00DFH PREDEFINED

AGAIN. . . . . . . . . . . . . . C ADDR 005AH

ASCII2HEX. . . . . . . . . . . . C ADDR 00BEH

B. . . . . . . . . . . . . . . . D ADDR 00F0H PREDEFINED

CHAN . . . . . . . . . . . . . . NUMB 0000H

DELAY. . . . . . . . . . . . . . C ADDR 0072H

DLY0 . . . . . . . . . . . . . . C ADDR 0073H

DLY1 . . . . . . . . . . . . . . C ADDR 0075H

GETCHAR. . . . . . . . . . . . . C ADDR 00E3H

GETVAL . . . . . . . . . . . . . C ADDR 00EBH NOT USED

HEX2ASCII. . . . . . . . . . . . C ADDR 00B2H

IO0010 . . . . . . . . . . . . . C ADDR 0085H

IO0020 . . . . . . . . . . . . . C ADDR 0091H

IO0030 . . . . . . . . . . . . . C ADDR 00BBH

IO0040 . . . . . . . . . . . . . C ADDR 00DEH

IO0050 . . . . . . . . . . . . . C ADDR 00E1H

IO0060 . . . . . . . . . . . . . C ADDR 00EFH

IO0070 . . . . . . . . . . . . . C ADDR 0100H

LED. . . . . . . . . . . . . . . NUMB 00B4H

MAIN . . . . . . . . . . . . . . C ADDR 004BH

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

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

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

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

SCONV. . . . . . . . . . . . . . B ADDR 00DCH PREDEFINED

SENDCHAR . . . . . . . . . . . . C ADDR 0096H

SENDSTRING . . . . . . . . . . . C ADDR 007EH NOT USED

SENDVAL. . . . . . . . . . . . . C ADDR 009EH

T3CON. . . . . . . . . . . . . . D ADDR 009EH PREDEFINED

T3FD . . . . . . . . . . . . . . D ADDR 009DH PREDEFINED

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