;====================================================================

;

; Author : ADI - Apps

;

; Date : November 2001

;

; File : UART.asm

;

; Hardware : ADuC836

;

; Description : This Program saves 16 numbers in order initially

; starting with 0 into memory locations 40h to 50h.

; When finished the values in these locations are

; transmitted down the UART in ASCII form to the PC

; where they can be viewed using the preconfigured

; Hyperterminal program. (c:\ADuC\9600com1.ht)

;

; After the transmission of the 16 bytes a 5 second

; delay is called and the process is repeated, this

; time starting with the saving of 10h to location

; 40h.

;

;====================================================================

;

$MOD836 ;Use 8052 predefined Symbols

LED EQU P3.4

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; BEGINNING OF CODE

CSEG

ORG 0000H

JMP MAIN

ORG 0060H ; Start code at address above interrupts

MAIN: ; Main program

; configure UART for 9600 using Timer3

MOV T3CON,#82h

MOV T3FD,#12h

MOV SCON,#52h

MOV R0, #00 ; start output data at 0

MOV R1, #40h ; initialise R1 to 40 to store the

; input data from memory location 40

SAVENOS:

MOV A,R0

MOV @R1, A ; move R0 into memory location R1

INC R1 ; increment memory location and data so

; new data is stored in new address

INC R0

CJNE R1, #50H, SAVENOS ; reset memory location to 40h

; when memory location reaches 50h

; saving 16 bytes of data

; Transmit the values in locations 40h->50h up the UART wait for

; 5 seconds and then repeat

START: CPL LED ;CPL LED with each transmission

MOV DPTR, #TITLE

CALL SENDSTRING ; write title block on screen

MOV R1, #40h ; move value at address 40 into R2

MOV A, @R1

MOV R2, A

NEXT: ; Put new value on a new line

MOV A, #10 ; Transmit a linefeed (= ASCII 10)

CALL SENDCHAR

MOV A, #13 ;Transmit a carriage return (=ASCII 13)

CALL SENDCHAR

MOV A, R2 ; Transmit R2 i.e. value @ address R1

CALL SENDVAL

INC R1 ; Increment address

MOV A, @R1

MOV R2, A ; R2 holds the value @ addrR1

MOV A, R1 ; Check if at address 50h

CJNE A, #50h, NEXT ; if not jump to Next

JMP WAIT5S ; if so wait 5s and repeat

WAIT5S: MOV A, #50

CALL DELAY ; Wait 5 seconds

MOV R1, #40h

JMP SAVENOS ; Resave new numbers to same addresses

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDSTRING

SENDSTRING: ; sends ASCII string to UART starting at location

; DPTR and ending with a null (0) value

PUSH ACC

PUSH B

CLR A

MOV B,A

IO0010: MOV A,B

INC B

MOVC A,@A+DPTR

JZ IO0020

CALL SENDCHAR

JMP IO0010

IO0020: POP B

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDCHAR

SENDCHAR: ; sends ASCII value contained in A to UART

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; SENDVAL

SENDVAL: ; converts the hex value of A into two ASCII chars,

; and then spits these two characters up the UART.

; does not change the value of A.

PUSH ACC

SWAP A

CALL HEX2ASCII

CALL SENDCHAR ; send high nibble

POP ACC

PUSH ACC

CALL HEX2ASCII

CALL SENDCHAR ; send low nibble

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; HEX2ASCII

HEX2ASCII: ; converts A into the hex character representing the

; value of A's least significant nibble

ANL A,#00Fh

CJNE A,#00Ah,$+3

JC IO0030

ADD A,#007h

IO0030: ADD A,#'0'

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; DELAY

DELAY: ; Delays by 100ms \* A

; 100mSec based on 1.5728MHZ

; Core Clock

; i.e. default ADuC824 Clock

MOV R1,A ; Acc holds delay variable

DLY0: MOV R2,#019h ; Set up delay loop0

DLY1: MOV R3,#0FEh ; Set up delay loop1

DJNZ R3,$ ; Dec R3 & Jump here until R2 is 0

DJNZ R2,DLY1 ; Dec R2 & Jump DLY1 until R1 is 0

DJNZ R1,DLY0 ; Dec R1 & Jump DLY0 until R0 is 0

RET ; Return from subroutine

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TITLE: DB 10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

DB 'Analog Devices MicroConverter ADuC836',10,13

DB ' UART Demo Routine',10,13

DB ' Data Stored in Memory in Hex Form',10,13,0

END