;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

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

;

; Date : May 2002

;

; File : DualDPTR.asm

;

; Hardware : ADuC831

;

; Description : Sample Program to show the new ADuC831 feature

; of the extended 11-bit Stack Pointer

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$MOD831 ; Use 8052&ADuC831 predefined symbols

LED EQU P3.4 ; P3.4 drives red LED on eval board

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

; BEGINNING OF CODE

CSEG

ORG 0000h

; Configure UART

MOV T3CON,#85h

MOV T3FD,#08h

MOV SCON,#52h

; enable the intenal On-Chip XRAM, keep Flash Timing correct

ORL CFG831, #01h

MOV DPTR, #STACK8MSG

CALL SENDSTRING

; initialise SP

MOV SP, #0F0h

CALL SENDSP

CALL LEVEL1

CALL SENDSP ; stack should be back at F0 here

; (it will print as F2 because calling SENDSP

; will increment the stack twice)

; now enable the stack to rollover into XRAM

ORL CFG831, #80h

MOV DPTR, #STACK11MSG

CALL SENDSTRING

; print out new stack tree

CALL LEVEL1

CALL SENDSP

JMP $ ; will eventually return to here

LEVEL1:

CALL SENDSP

CALL LEVEL2

CALL SENDSP

RET

LEVEL2:

CALL SENDSP

CALL LEVEL3

CALL SENDSP

RET

LEVEL3:

CALL SENDSP

CALL LEVEL4

CALL SENDSP

RET

LEVEL4:

CALL SENDSP

CALL LEVEL5

CALL SENDSP

RET

LEVEL5:

CALL SENDSP

CALL LEVEL6

CALL SENDSP

RET

LEVEL6:

CALL SENDSP

CALL LEVEL7

CALL SENDSP

RET

LEVEL7:

CALL SENDSP

CALL LEVEL8

CALL SENDSP

RET

LEVEL8:

CALL SENDSP

CALL LEVEL9

CALL SENDSP

RET

LEVEL9:

CALL SENDSP

CALL LEVEL10

CALL SENDSP

RET

LEVEL10:

CALL SENDSP

CALL LEVEL11

CALL SENDSP

RET

LEVEL11:

CALL SENDSP

CALL LEVEL12

CALL SENDSP

RET

LEVEL12:

CALL SENDSP

RET

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

; SENDSP

SENDSP:

; send line feed

MOV A, #10

CALL SENDCHAR

MOV A, #13

CALL SENDCHAR

; send SP

MOV A, SPH

CALL SENDVAL

MOV A, SP

CALL SENDVAL

RET

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

; 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

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

; 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

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

; 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

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

; 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

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

STACK8MSG: DB 10,10,13,'8-BIT STACK POINTER',0

STACK11MSG: DB 10,10,13,'11-BIT STACK POINTER',0

END