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

;

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

;

; Date : April 2002

;

; File : WDtimer.asm

;

; Hardware : ADuC832

;

; Description : Demonstrates use of the on-chip watchdog timer.

; In normal operation, WD timer is refreshed by code

; every 100ms, as indicated by a flashing LED (10 Hz).

; In a runaway code condition (which can be simulated

; here by pressing the INT0 button on the eval board)

; code fails to refresh WD bits, the LED stays in the

; off position, before the WD timer generates a

; reset after a user selected time-out period (from

; 15.6ms to 2000ms) has elapsed. The time-out period

; in this routine is 2.0s for visual clarity.

;

; After a reset the light blinks at a slower rate 2Hz.

; Future watchdog resets can be enabled in the same

; way. A hard reset will clear the WDS bit and the

; quicker flashes will occur again.

;

; note: be sure to remove the PSEN pull-down (LK2)

; before allowing watchdog to time-out, or

; you'll end up in serial download mode again

; (the LED will stay in the on condition)

; rather than recovering normal code execution.

;

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

$MOD832 ; Use 8052&ADuC832 predefined symbols

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

ERROR EQU F0 ; the 'ERROR' flag is used here to

; simulate an erroneous command that

; sends code into an unknown state

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

; BEGINNING OF CODE

CSEG

ORG 0000h

JMP MAIN ; jump to main program

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

; EXTERNAL INTERRUPT VECTOR SPACE

ORG 0003h ; (INT0 ISR)

SETB ERROR ; simulate an error condition..

; ..when INT0 button is pressed

RETI

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

; MAIN PROGRAM

ORG 0060h ; Start at address above interrupts

MAIN:

JB WDS, WDRESET

MOV R0, #01h ; this will blink the LED at 10Hz

JMP START

WDRESET: MOV R0, #05h ; this will blink the LED at 2Hz

; Enable external interupt to trigger simulated error condition...

START: SETB IT0 ; make INT0 edge triggered

SETB EX0 ; enable INT0 (button on eval board)

SETB EA

; Configure the Watchdog timer. It should be configured like this,

; with the global interrupts turned off and setting WDWR to allow

; writing to WDCON.

CLR EA

SETB WDWR

MOV WDCON, #72h ; Enable Watchdog timer to cause

; -2.0 second timeout period

; -enable WDIR bit to generate

; a reset and not an interrupt

SETB EA ; set global interrupts again

; from this point forward, watchdog bits must be refreshed every

; 2.0 seconds or less. if they are not, watchdog timer will

; generate a reset.

CLR ERROR ; simulate error free operation

; The below loop represents normal code execution...

FLASH: MOV A, R0

CALL DELAY ; delay by 100ms x R0

CPL LED ; blink (complement) the red LED

CLR EA ; refresh watchdog timer

SETB WDWR

SETB WDE

SETB EA

JNB ERROR, FLASH ; jump if 'ERROR' flag is not set

; The below endless loop represents run-away code execution...

SETB LED ; turn LED off during runaway code

JMP $ ; this endless loop is used to

; represent an unknown state of

; program execution

; program will sit in the above endless loop until the watchdog

; period (2000ms) has elapsed, at which time a reset will be

; generated by the watchdog timer, thereby recovering the chip to

; resume normal code execution.

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

DELAY: ; Delays by 100ms \* A

; 100mSec based on 2.097152MHZ

; Core Clock

; i.e. default ADuC832 Clock

MOV R5,A ; Acc holds delay variable

DLY0: MOV R6,#022h ; Set up delay loop0

DLY1: MOV R7,#0FFh ; Set up delay loop1

DJNZ R7,$ ; Dec R2 until R2 is zero

DJNZ R6,DLY1 ; Dec R1 & Jump DLY1 until R1 is 0

DJNZ R5,DLY0 ; Dec R0 & Jump DLY0 until R0 is 0

RET ; Return from subroutine

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

END