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

;

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

;

; Date : 28 May 1999

;

; File : FlashEE.asm

;

; Hardware : ADuC812

;

; Description : Demonstrates use of the on-chip read/write 640 byte

; FlashEE data memory space. Stores a sequence of

; button presses (INT0 button on eval board) in data

; FlashEE space. Replays sequence on LED when board

; is reset or power cycled. Will store the sequence

; until another is recorded with a new set of button

; presses. To record a new sequence, just wait until

; the current one finishes playing (LED is off) and

; enter new sequence via button (INT0).

;

; NOTE: : DO NOT write to FlashEE addresses above page 159!

; ----- The 640 bytes are stored in pages 0 thru 159 (9Fh)

; as four bytes per page. Writing to pages above

; those documented can permanently lock you out of

; the chip. See ADuC812 data sheet and errata sheet

; for details (www.analog.com/microconverter).

;

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

$MOD812 ; Use 8052&ADuC812 predefined symbols

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

BUTTON EQU P3.2 ; button on eval board drives P3.2

PREVIOUS EQU F0 ; flag to hold previous button value

READ EQU 01h ; FlashEE command: 'read page'

WRITE EQU 02h ; FlashEE command: 'write page'

VERIFY EQU 04h ; FlashEE command: 'verify page'

ERASE EQU 05h ; FlashEE command: 'erase page'

ERASEALL EQU 06h ; FlashEE command: 'erase all'

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

; BEGINNING OF CODE

CSEG

ORG 0000h

SETB LED ; turn LED off

MOV A,#15

CALL DELAY ; pause 1.5 seconds

MOV EADRL,#0 ; set data FlashEE address to page 0

; READ FLASH/EE DATA and indicate values via LED on and off times...

READPG: MOV ECON,#READ ; read current 4byte page of FlashEE

; into EDATA1,2,3,4

MOV A,#1

CJNE A,EDATA4,EMPTY ; if EDATA4 is 1, then page contains

; valid data from previous write

; otherwise, jump to EMPTY

CALL BLINK ; flash LED for period determined

; by FlashEE data just read

INC EADRL ; increment to next FlashEE page addr

MOV A,EADRL

CJNE A,#0A0h,CMPRR ; if address is less than 160..

CMPRR: JC READPG ; ..then jump to read the next page

; (DO NOT write to pages above 159!)

; WHEN "PLAY" SEQUENCE IS FINIISHED, wait for button press...

EMPTY: SETB LED

JB BUTTON,$ ; wait for first button press

MOV A,#1

CALL DELAY ; pause 100ms

JB BUTTON,EMPTY ; ensure button wasn't false trigger

; IF BUTTON PRESSED, then ERASE and go into "RECORD" mode...

MOV ECON,#ERASEALL ; clear all data FlashEE memory

MOV EADRL,#0

; CAPTURE AND RECORD in FLASH/EE space time between button edges...

GETNEW: CALL GETVAL

MOV EDATA1,DPL ; place DPTR in EDATA1,2,3

MOV EDATA2,DPH

MOV EDATA3,DPP

MOV EDATA4,#1 ; put 1 in EDATA4 as identifier

MOV ECON,#ERASE

MOV ECON,#WRITE ; write EDATA1-4 into current page of

; FlashEE data memory

MOV ECON,#VERIFY ; verify current page is same as..

MOV A,ECON ; ..EDATA1-4. if same, ECON <- 0

JNZ EMPTY ; if verify fails, jump to EMPTY

INC EADRL ; increment to next FlashEE page addr

MOV A,EADRL

CJNE A,#0A0h,CMPRG ; if EADRL<A0h..

CMPRG: JC GETNEW ; ..then jump to get the next value

; (DO NOT write to pages above 159!)

; WHEN FLASH/EE DATA SPACE IS FULL...

SETB LED ; code will end up here only after 160 button

; edges. no more can be recorded, so code

JMP $ ; just waits for a reset or power cycle.

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

; SUBROUTINES

BLINK: ; turn LED on or off for the duration

; based on the value in EDATA3/2/1

CPL LED

MOV DPL,#0

MOV DPH,#0 ; clear DPTR

MOV DPP,#0

AGAIN1: INC DPTR ; increment DPTR.. 2

MOV A,DPP ; 1

CJNE A,EDATA3,CMPR3 ; 2

MOV A,DPH ; 1

CJNE A,EDATA2,CMPR2 ; 2

MOV A,DPL ; 1

CJNE A,EDATA1,CMPR1 ; 2

CMPR3: NOP ; 1

NOP ; 1

NOP ; 1

CMPR2: NOP ; 1

NOP ; 1

NOP ; 1

CMPR1: JC AGAIN1 ; ..until DPTR>=EDATA3/2/1 2

RET

; this routine directly controls LED on and off times based on data

; previously stored by a similar routine (GETVAL) which measures

; BUTTON on and off times.

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

GETVAL: ; get a value based on duration of

; button press

MOV DPL,#0

MOV DPH,#0 ; clear DPTR

MOV DPP,#0

CPL LED

AGAIN2: INC DPTR ; keep incrementing DPTR.. 2

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

NOP ; 1

JNB LED,CHKB ; 2

JNB BUTTON,AGAIN2 ; 2

RET ; ..until the button changes state

CHKB: JB BUTTON,AGAIN2 ; 2

RET

; DPTR (DPP,DPH,DPL) now holds a number that represents the length of

; time between button edges. this data will be stored in FlashEE

; space for use in controlling LED on and off times in "play" mode.

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

DELAY: ; delay 100ms \* A

DLY0: MOV R7,#200 ; 200 \* 500us = 100ms

DLY1: MOV R6,#229 ; 229 \* 2.17us = 500us

DJNZ R6,$ ; sit here for 500us

DJNZ R7,DLY1 ; repeat 200 times (100ms total)

DJNZ ACC,DELAY ; repeat 100ms delay ACC times

RET

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

END