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

;

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

;

; Date : April 2002

;

; File : Boot831.asm

;

; Hardware : ADuC831

;

; Description : Example bootloader program that lies in upper

; 6kbytes of the 62kByte code space. This bootloader

; can be used to download to the bottom 56kBytes.

;

; NOTE: This program will only work if the option to

; always run code from E000H after download is

; selected.

;

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

$MOD831

ACK EQU 06H

NACK EQU 15H

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

; EXAMPLE USER CODE

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

CSEG

ORG 0000H

AJMP MAIN

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

; INT0 ISR

ORG 0003H

; wait to receive a character from the UART

JNB RI, $ ; wait for reception

CLR RI

MOV A, SBUF

CJNE A, #'U', ERROR

; plan to upgrade new code => set BOOTEN

MOV EADRH, #0

MOV EADRL, #0

MOV ECON, #1 ; read page

ORL EDATA1, #1 ; SET LSB

MOV ECON, #5 ; ERASE page

MOV ECON, #2 ; program page

MOV ECON, #4 ; verify page

MOV A, ECON

JNZ ERROR

; use the watchdog timer to reset part...run from E000H after reset

CLR EA ; disable interrupts for double

; write sequence

SETB WDWR

MOV WDCON, #82h

ERROR:

RETI

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

; MAIN

MAIN:

; enable INT0

SETB IT0 ; INT0 edge triggered

SETB EA ; enable inturrupts

SETB EX0 ; enable INT0

; configure UART for 115200

MOV T3CON, #81h

MOV T3FD, #20h

MOV SCON, #52H

; THIS SIMPLE BLINK ROUTINE REPRESENTS THE MAIN PROGRAM

BLINK:

CPL P3.4

CALL DELAY

AJMP BLINK

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

; DELAY

DELAY:

; 70ms DELAY

MOV R0,#255

DLY:

MOV R1,#255 ; 255 x 255 x 1.08us

DJNZ R1,$

DJNZ R0,DLY

RET

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

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

; EXAMPLE BOOTLOAGER CODE

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

CSEG

ORG 0E000h

; read BOOTEN

MOV EADRH, #0

MOV EADRl, #0

MOV ECON, #1 ; read page

MOV A, EDATA1

JB ACC.0, BOOTLOADER

; reset EDATA1-4 before running user code

CLR A

MOV EDATA1, A

MOV EDATA2, A

MOV EDATA3, A

MOV EDATA4, A

LJMP 0000H

BOOTLOADER:

; clear the deliberate WDT reset

SETB WDWR

MOV WDCON, #10H

; configure UART for 115200 baud

MOV T3CON, #81H

MOV T3FD, #20h

MOV SCON,#52h

; configure in ULOAD mode

MOV ECON, #0F0h

GETCOMMAND:

CALL RECBYTE

CJNE A, #'E', $+5

AJMP ERASECOMMAND

CJNE A, #'D', $+5

AJMP DOWNLOADCOMMAND

CJNE A, #'O', $+5

AJMP DOWNLOADOKCOMMAND

AJMP SENDNACK

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

; ERASE CODE

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

ERASECOMMAND:

; wait for erase command

; <'E'><CS>

CALL RECBYTE

ADD A,#'E'

JZ ERASEOK

AJMP SENDNACK

ERASEOK:

; erase 56kbytes of code space

MOV ECON, #6

; send ACK

AJMP SENDACK

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

; DOWNLOAD CODE

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

DOWNLOADCOMMAND:

; wait for download command

; <'D'><PAGEADD><DATA0->255><CS>

MOV R0, #'D'

; get page address

CALL RECBYTE

MOV EADRH, A

mov eadrl, #13

ADD A, R0

MOV R0, A

; check for page address >E0h

MOV A, EADRH

CJNE A, #0E0H, $+3

JC ADDRESSOK ; C=0 for EADRH < E0h

AJMP SENDNACK

ADDRESSOK:

MOV DPTR, #0

MOV R1, #0 ; count

ORL CFG831, #1 ; int XRAM

READDATA:

ACALL RECBYTE

MOVX @DPTR, A

INC DPTR

ADD A, R0

MOV R0, A

DJNZ R1, READDATA ; REPEAT 256 TIMES

; verify checksum

ACALL RECBYTE

ADD A, R0

MOV R0, A

JZ DOWNLOADCHECKSUMOK

AJMP SENDNACK

DOWNLOADCHECKSUMOK:

; program page

MOV ECON, #2

; verify download

MOV DPCON, #54H ; main DPTR in auto INC mode

; shadow DPTR in auto INC mode

; DPTR in aut toggle mode

MOV DPTR, #0 ; main DPTR=0 (XRAM)

INC DPCON ; select shadow DPTR

MOV DPH, EADRH ; shadow DPTR (CODE)

MOV DPL, #0

MOV R0, #0

VERIFYDOWNLOADLOOP:

; read code memory

CLR A

MOVC A, @A+DPTR ; swap to main DPTR

MOV B, A

MOVX A, @DPTR

CJNE A, B, JMPSENDNACK

DJNZ R0, VERIFYDOWNLOADLOOP

MOV DPCON, #0

AJMP SENDACK

JMPSENDNACK:

AJMP SENDNACK

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

; DOWNLOAD OK COMMAND

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

DOWNLOADOKCOMMAND:

; wait for Download OK command

; <'O'><CS>

CALL RECBYTE

ADD A,#'O'

JZ EXITULOADMODE

AJMP SENDNACK

EXITULOADMODE:

; exit ULOAD mode

MOV ECON, #0Fh

DOWNLOADOK:

; clear BOOTEN

MOV EADRH, #0

MOV EADRL, #0

MOV ECON, #1 ; read page

ANL EDATA1, #0FEh ; clear LSB

MOV ECON, #5 ; ERASE page

MOV ECON, #2 ; program page

MOV ECON, #4 ; verify page

MOV A, ECON

JZ BOOTENCLEAR

AJMP SENDNACK

BOOTENCLEAR:

; send an ACK

MOV A, #ACK

CALL SENDBYTE

JNB TI, $ ; disabling UART shortly

; => wait for char to send

RESETSFR:

; reset SFRs

CLR A

MOV B, A

MOV PSW, A

MOV EADRH, A

MOV EADRL, A

MOV EDATA1, A

MOV EDATA2, A

MOV EDATA3, A

MOV EDATA4, A

MOV DPCON, #1

MOV DPTR, #0 ; clear shadow DPTR

MOV DPCON, A

MOV DPTR, #0 ; clear main DPTR

XRL CFG831,#01H

MOV T3CON, A

MOV T3CON, A

MOV SCON, A

; jump to 0000H

LJMP 0000H

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

; FUNCTIONS

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

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

; RECBYTE

RECBYTE: ; waits for a single ASCII character to be received

; by the UART. places this character into A.

JNB RI,$

MOV A,SBUF

CLR RI

RET

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

; SENDBYTE

SENDBYTE: ; sends ASCII value contained in A to UART

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A

RET

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

; SENDACK/SENDNACK

SENDACK:

MOV A, #ACK

AJMP CONTSENDACK

SENDNACK:

MOV A, #NACK

CONTSENDACK:

ACALL SENDBYTE

AJMP GETCOMMAND

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

END