;variable and constant definitions

keyBytesRAMaddress EQU 0x40 ; symbolic constant for base address of

; encryption key in RAM

keyLength EQU 0x30 ; variable to track length of key

keyvalIndex EQU 0xe0 ; variable to index the keyval constant array

; keyvalIndex is also an alias for accumulator

;begin section from lab 9

;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

jmp main ;jump past interrupt vector table

org 0x0030 ;put main program at rom location 0x0030

charIndex EQU 0xE0 ;alias variable for Accumulator

writeString:mov charIndex, #0x00 ;put char in serial buffer

again: PUSH charIndex

movc a, @a + DPTR

jnz not0

pop charIndex

ret

writeChar1: mov SBUF,a

jnb TI,$

clr TI

ret

not0: Call writeChar1

pop charIndex

inc charIndex

jmp again

main:

mov TMOD, #0x21 ;configure timer 0, mode 1

mov SCON, #0x50

mov TH1, #0xFD

SETB TR1

mov DPTR, #Welcome

Call writeString

mov R0, keyBytesRAMaddress

getKey: jnb RI, $

call getchar

clr RI

mov @R0, A

inc R0

cjne A, #0x00, getKey

mov DPTR, #Receive

Call writeString

;---------------- Initialization/configuration ----------------;

;keyval variable no longer used

; mov keyval, #0x23 ;load the keyval variable with encryption key

mov tmod, #0x20 ;config timer 1 mode 2

mov scon, #0x50 ;config serial 8-data, 1 start, 1 stop, no parity

mov th1, #0xFD ;9600 baud

setb tr1 ;start timer 1 to enable serial communication

;end section from lab 9

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;In the following section load the key bytes from ROM into RAM

mov r0, #keyBytesRAMaddress ;initialize RAM pointer

mov dptr, #keyvals2 ;initialize ROM pointer

mov keyvalIndex, #0x00 ;initialize keyvalIndex

LoadKey:

push keyvalIndex ;preserve keyvalIndex variable

movc a,@a+dptr;load byte of key from ROM

cjne a, #0x00, notNull ;check for null terminating character

jmp LoadDone ;if null is found, enter main\_loop

notNull:

mov @r0, a ;put byte of key into ram

pop keyvalIndex;restore keyvalIndex variable

inc keyvalIndex;increment keyvalIndex

inc r0;increment RAM pointer

jmp LoadKey;continue the loop

LoadDone:

mov @r0, #0x00 ;append null char to string

mov r0, #keyBytesRAMaddress;re-initialize RAM pointer

;---------------- END of Initialization/configuration ----------------;

;begin section from lab 9

;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

mainloop:

jnb ri, $ ;wait to receive a char

call getchar ;char received, get it!

; cjne a, #0x00, encrypt ;check for null character in string

cjne a, #0x00, checkKeyVal ;check for null character in string

jmp terminate ;terminate program if null character is recieved

;end section from lab 9

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

checkKeyVal:

cjne @r0, #0x00, Encrypt;go to Encrypt if keyVal is not null

mov r0, #keyBytesRAMaddress ;re-initialize RAM pointer

;begin section from lab 9

;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Encrypt:

xrl a, @r0 ;encrypt the character contained in the accumulator

call writechar ;write the encrypted character

inc R0

jmp mainloop

terminate:

mov a, #0x00 ;load null character into accumulator

call writechar ;append the null character to text output

sjmp $ ;halt

;----------- getchar ----------;

;subroutine receives nothing before it is called

;writes the character to the serial console

;returns a byte in the accumulator

getchar:

mov a, sbuf ;get serial data (char)

clr ri ;acknowledge data received

ret ;return from subroutine call

;----------- writechar ----------;

;receives byte or character

;reads a character that has been received serially

;returns the c

writechar:

mov sbuf, a ;send data (char) serially

jnb ti, $ ;wait until data is sent

clr ti ;acknowledge data has been sent

ret ;return from subroutine call

;end section from lab 9

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;multibyte keys are defined below, only one will be used at a time

org 0x200

Welcome: DB "Send a key value file: ",0

Receive: db "Key value file received ",0

keyvals: db '12345678',0

keyvals2: db 0x23, 0x34, 0x45, 0x56, 0x67, 0x78,0x89,0x90, 0x56, 0x3B, 0x00

DutyValues: DB 0x60, 0x70, 0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0

End







