Podstawy techniki mikroprocesorowej

Sprawozdanie z laboratorium

Data	Tytuł zajęć	Uczestnicy
19.05.2017 14:15	Programowanie obsługi klawiatury	Iwo Bujkiewicz (226203)

Cel ćwiczenia

Celem zajęć było zaznajomienie się ze sposobem odczytywania kodów klawiszy wciskanych przez użytkownika na klawiaturze (typu numerycznego, 16-klawiszowej), konwertowania ich na kody znaków i wyświetlania owych znaków na ekranie.

Opisy i listingi programów

Zadaniem pierwszego programu było odczytywanie kodów wciśniętych na klawiaturze klawiszy i wyświetlanie tych kodów na bieżąco poprzez diody na porcie `P1`.

```
        LJMP START

        P5  EQU 0xF8    ; Port wyboru wiersza klawiatury
        P7  EQU 0xDB    ; Port odczytu kolumn klawiatury

        ORG 0x0100

RDrow MACRO             ; Makro odczytu i wyświetlania na diodach
        LOCAL next      ; kodów kolumn naciśniętych klawiszy z wybranych wierszy

        MOV A, R0       ; Wczytanie podanego w R0 kodu wierszy
        MOV P5, A       ; Wybranie wierszy do odczytu z klawiatury
        MOV A, P7       ; Odczytanie kodu kolumn
        ANL A, R0       ; Połączenie kodów wierszy i kolumn

        MOV R2, A       ; Zachowanie kopii uzyskanego kodu
        CLR C           ; Wyzerowanie flagi przeniesienia
        SUBB A, R0      ; Odjęcie kodu wierszy od uzyskanego kodu w celu
        JZ next         ; sprawdzenia, czy jakikolwiek klawisz został naciśnięty
        MOV A, R2       ; Jeśli tak, pobranie zachowanej kopii kodu
        MOV P1, A       ; i podanie go na port diód
next:
        ENDM

START:
        MOV R0, #0x7F   ; Zapętlone skanowanie kolejnych wierszy klawiatury
        RDrow           ; i wyświetlanie uzyskanych kodów na diodach
        MOV R0, #0xBF
        RDrow
        MOV R0, #0xDF
        RDrow
        MOV R0, #0xEF
        RDrow
        JMP START
        NOP
        NOP
        NOP
        JMP $
END START
```

Drugi program miał za zadanie oczekiwać na naciśnięcie klawisza, a następnie po puszczeniu klawisza przez użytkownika wyświetlić odpowiadający mu znak na LCD. Ekran wyświetlał wszystkie wprowadzone kolejno znaki, a po zapełnieniu obu linii był czyszczony i wprowadzanie rozpoczynało się na nowo.

```
#include    "LAB05_LCDM.A51"

LJMP START

P5  EQU 0xF8
P7  EQU 0xDB

ORG 0x0100

kcStor MACRO x, y       ; Makro zapisania w XRAM kodu znaku
        MOV DPL, x      ; odpowiadającego kodowi naciśniętego klawisza
        MOV A, y
        MOVX @DPTR, A
        ENDM

RDrow MACRO             ; Makro odczytu kodu klawisza z wybranego wiersza
        LOCAL save, next    ; i wypisania odpowiadającego mu znaku na LCD

        MOV A, R0       ; Wczytanie podanego w R0 kodu wierszy
        MOV P5, A       ; Wybranie wierszy do odczytu z klawiatury
        MOV A, P7       ; Odczytanie kodu kolumn
        ANL A, R0       ; Połączenie kodów wierszy i kolumn

        MOV R2, A       ; Zachowanie kopii uzyskanego kodu
        CLR C           ; Wyzerowanie flagi przeniesienia
        SUBB A, R0      ; Odjęcie kodu wierszy od uzyskanego kodu w
        JNZ save        ; sprawdzenia, czy jakikolwiek klawisz został naciśnięty
        MOV A, R4       ; Jeśli nie, sprawdzenie, czy znajdujemy się w wierszu,
        JNZ next        ; w którym ostatnio odczytaliśmy naciśnięcie klawisza

        MOV A, R3       ; Jeśli tak, oznacza to, że użytkownik puścił
        MOV DPH, #0x80  ; naciśnięty wcześniej klawisz, należy więc wypisać
        MOV DPL, A      ; odpowiedni znak na LCD
        MOVX A, @DPTR
        LCDcharWR
        DEC R1          ; Dekrementacja licznika wolnych znaków w linii
        MOV R4, #0xFF   ; Zresetowanie zapisanego kodu znaku

        JMP next
save:
        MOV A, R2       ; Jeśli zarejestrowano naciśnięty klawisz,
        MOV R3, A       ; zapisanie jego kodu
        MOV A, R0       ; oraz kodu aktualnego wiersza
        MOV R4, A       ; i oczekiwanie na zwolnienie lub naciśnięcie innego
next:
        ENDM

START:
        MOV DPH, #0x80
        MOV R0, #0x00
reset:                  ; Wyzerowanie zawartych w XRAM kodów znaków
        MOV A, R0       ; dla klawiszy
        kcStor A, #0x00
        DJNZ R0, reset

        kcStor #0x77, #0x31 ; Zapisanie odpowiednich kodów znaków w XRAM
        kcStor #0x7B, #0x32
        kcStor #0x7D, #0x33
        kcStor #0x7E, #0x41
        kcStor #0xB7, #0x34
        kcStor #0xBB, #0x35
        kcStor #0xBD, #0x36
        kcStor #0xBE, #0x42
        kcStor #0xD7, #0x37
        kcStor #0xDB, #0x38
```

```
        kcStor #0xDD, #0x39
        kcStor #0xDE, #0x43
        kcStor #0xE7, #0x45
        kcStor #0xEB, #0x30
        kcStor #0xED, #0x46
        kcStor #0xEE, #0x44

        init_LCD        ; Rozpoczęcie pracy LCD
        MOV R4, #0xFF
        JMP line1

scan:
        MOV R0, #0x7F   ; Zapętlone skanowanie kolejnych wierszy
        RDrow           ; i wyświetlanie uzyskanych znaków na ekranie
        MOV R0, #0xBF
        RDrow
        MOV R0, #0xDF
        RDrow
        MOV R0, #0xEF
        RDrow
        MOV A, R1       ; Sprawdzenie, czy wolne miejsce w aktualnej linii
        JZ line1        ; LCD się skończyło
        CLR C
        SUBB A, #0x10
        JZ line2
        JMP endline

line1:
        LCDcntrlWR #CLEAR   ; Wyczyszczenie LCD i ustawienie kursora
        LCDcntrlWR #HOME    ; w pierwszej linii
        MOV R1, #0x20
        JMP endline
line2:
        LCDcntrlWR #HOME2   ; Ustawienie kursora w drugiej linii LCD

endline:
        JMP scan

        NOP
        NOP
        NOP
        JMP $
END START
```

Kod zawarty w `LAB05_LCDM.A51` (makra do obsługi LCD)

```
        LCDstatus   EQU 0xFF2E
        LCDcontrol  EQU 0xFF2C
        LCDdataWR   EQU 0xFF2D
        LCDdataRD   EQU 0xFF2F

        #define HOME    0x80    // place caret in first line
        #define INIT    0x38    // LCD 8-bit init
        #define HOME2   0xC0    // place caret in second line
        #define LCDON   0x0E    // init caret, switch cursor off, switch blinking
off
        #define CLEAR   0x01    // clear LCD lines

LCDcntrlWR  MACRO x
        LOCAL loop
loop:
        MOV DPTR, #LCDstatus
        MOVX A,@DPTR
        JB ACC.7, loop          ; check if LCD busy

        MOV DPTR,#LCDcontrol    ; write to LCD control
        MOV A, x
        MOVX @DPTR,A
```

```
        ENDM

LCDcharWR   MACRO
        LOCAL loop1, loop2

        PUSH ACC
loop1:  MOV DPTR, #LCDstatus
        MOVX A, @DPTR
        JB ACC.7, loop1         ; check if LCD busy

loop2:  MOV DPTR, #LCDdataWR    ; write data to LCD
        POP ACC
        MOVX @DPTR,A
        ENDM

init_LCD    MACRO
        LCDcntrlWR #INIT
        LCDcntrlWR #CLEAR
        LCDcntrlWR #LCDON
        ENDM

charStor    MACRO x
        MOV A, x
        MOVX @DPTR, A
        INC DPTR
        INC R1
        ENDM

charSeqWr   MACRO
        LOCAL loop
loop:   MOVX A, @DPTR
        LCDcharWR
        INC DPTR
        DJNZ R1, loop
        ENDM
```