МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине

«Программирование микроконтроллеров»

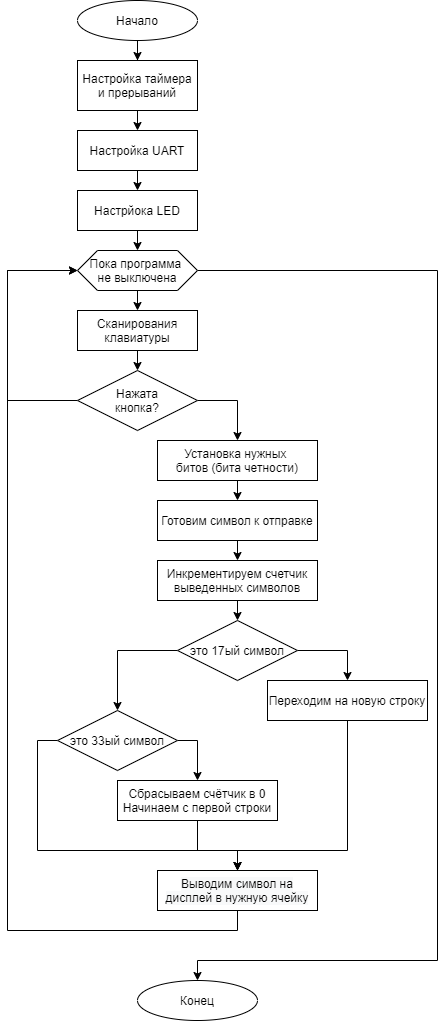
тема: «Работа с lcd дисплеем»

|  |  |
| --- | --- |
|  | Выполнил:  студент группы ВТ-32  Воскобойников Илья Сергеевич |
|  | Проверил:  Кижук А. С.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

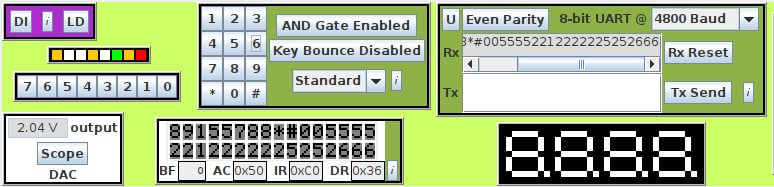
Белгород, 2021

**Выполнение:**

1. **Алгоритм**

****

1. **Пример использования**



1. **Программная реализация**

ORG 0

JMP mainLoop

org 13h

; прерывания int1 - чтение с клавиатуры

call scanKeyboardMarkup

mov p0, #0

reti

org 030h

mainLoop:

; Тут всё почти как всегда. Готовим нашу установку к работе

CLR SM0

SETB SM1 ; 2 режим, 8 бит, стоп/старт, четность

MOV A, PCON

SETB ACC.7

MOV PCON, A ; 7 бит в PCON установлен в 1, множитель скорости 2

; При переполнении записывается из TL1 в TH1 (M1 = 1, M0 = 0)

MOV TMOD, #20H

MOV TH1, #243

MOV TL1, #243

SETB TR1 ; Таймер №1

SETB IT1 ; Прерывания по срезу

SETB EX1 ; Прерывание от INT1

SETB EA ; Все прерывания

; Настраиваем дисплей.

call configForLCD

mov p0, #0

sjmp $ ; Бесконечный цикл ожидания пока юзер не грохнет программу

scanKeyboardMarkup:

; Получение нажатий с клавиатуры - практиески не меняется все 4 лабораторные

; поэтому тут комментариев будет мало, для чистоты

mov p0, #11111111b

clr p0.3

CALL scanKeyboardMarkupRow1

setb p0.3

clr p0.2

CALL scanKeyboardMarkupRow2

setb p0.2

clr p0.1

CALL scanKeyboardMarkupRow3

setb p0.1

clr p0.0

CALL scanKeyboardMarkupRow4

ret

scanKeyboardMarkupRow1:

MOV A, #31h

JNB P0.6, wasPressedButtonRow1

MOV A, #32h

JNB P0.5, wasPressedButtonRow1

MOV A, #33h

JNB P0.4, wasPressedButtonRow1

ret

wasPressedButtonRow1:

call displayScanedSymbolOnUART

ret

scanKeyboardMarkupRow2:

MOV A, #34h

JNB P0.6, wasPressedButtonRow2

MOV A, #35h

JNB P0.5, wasPressedButtonRow2

MOV A, #36h

JNB P0.4, wasPressedButtonRow2

ret

wasPressedButtonRow2:

call displayScanedSymbolOnUART

ret

scanKeyboardMarkupRow3:

MOV A, #37h

JNB P0.6, wasPressedButtonRow3

MOV A, #38h

JNB P0.5, wasPressedButtonRow3

MOV A, #39h

JNB P0.4, wasPressedButtonRow3

ret

wasPressedButtonRow3:

call displayScanedSymbolOnUART

ret

scanKeyboardMarkupRow4:

MOV A, #02ah

JNB P0.6, wasPressedButtonRow4

MOV A, #30h

JNB P0.5, wasPressedButtonRow4

MOV A, #23h

JNB P0.4, wasPressedButtonRow4

ret

wasPressedButtonRow4:

call displayScanedSymbolOnUART

ret

displayScanedSymbolOnUART:

MOV C, P

MOV ACC.7, C ; Ставим бит чётности

MOV SBUF, A ; printf(char)

JNB TI, $ ; Ожидание вывода

CLR TI ; clear flag TI

INC r0 ; Счётчик выведенных символов += 1

; Если символ №17 по порядку, то переходим на новую строку

cjne r0, #17, notMoveToNewLCDLine

CALL moveToNewLCDLine ; Перенос на новую строку

notMoveToNewLCDLine:

cjne r0, #33, skipRestart ; 33 символа => выводим в самый первый дисплей.

CALL restart ; "Перезапускаем" систему

CALL displayOnLCD ; Выводим на экран

skipRestart: ; Не рестартим

CALL displayOnLCD ; И тоже вывоим

ret

displayOnLCD:

; Вывод осуществляется переносом битов считанного символа

; и их помещением в порт передачи

; Старшие 4 бита "А" => порт передачи

MOV C, ACC.7

MOV P1.7, C

MOV C, ACC.6

MOV P1.6, C

MOV C, ACC.5

MOV P1.5, C

MOV C, ACC.4

MOV P1.4, C

; Перечадача

SETB P1.2

CLR P1.2

; Младшие 4 бита "А" => порт передачи

MOV C, ACC.3

MOV P1.7, C

MOV C, ACC.2

MOV P1.6, C

MOV C, ACC.1

MOV P1.5, C

MOV C, ACC.0

MOV P1.4, C

; Перечадача

SETB P1.2

CLR P1.2

; Ждём конца вывода символа

CALL pause

pause:

mov b, #20

djnz b, $

ret

configForLCD:

; Подготовка дисплея к использованию

; Очищаем бит RS для инструкции дисплея

CLR P1.3

; 2 строки. 4-битный режим.

mov p1, #00100000b

; Отправка половины инструкции

SETB P1.2

CLR P1.2

CALL pause

SETB P1.2

CLR P1.2

; Отправка второй половины инструкции

SETB P1.7 ; Работаем в 2 строчном режиме

; Отправка инструкции

SETB P1.2

CLR P1.2

CALL pause

; Подготовка к включению

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

; Отправка

SETB P1.2

CLR P1.2

SETB P1.7

SETB P1.6 ; Дисплей

SETB P1.5 ; Курсор подчеркивание

CLR P1.4 ; Выключаем другой курсор

SETB P1.2

; Отправляем инструкцию

CLR P1.2

CALL pause

; Активация режима вывода

SETB P1.3

; Инициализация счётчика чисел на LCD дисплее в данный момент

MOV R0, #0

ret

moveToNewLCDLine:

CLR P1.3 ; Инструкция

SETB P1.7 ; Перемещаем адреса записи

SETB P1.6 ; На вторую линию. Адрес 0x40

CLR P1.5

CLR P1.4 ; Инструкциия

SETB P1.2

; Отправка

CLR P1.2

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

SETB P1.2

; Отправка

CLR P1.2

CALL pause

; Выводим

SETB P1.3

ret

restart:

CLR P1.3

CLR P1.7 ; Смещаем указатель вывода в первый дисплей

CLR P1.6

CLR P1.5

CLR P1.4

SETB P1.2

CLR P1.2 ; Отправляем инструкцию

CLR P1.7

CLR P1.6

SETB P1.5 ; Код смещения в начало

CLR P1.4

SETB P1.2

CLR P1.2 ; Отправляем очередную инструкцию

CALL pause

MOV R0, #0

SETB P1.3 ; И мы снова готовы к выводу символов!

ret