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)

**Лабораторная работа №1**

дисциплина «Программирование микроконтроллеров»

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил: Гольцов Ю. А.

Белгород 2020

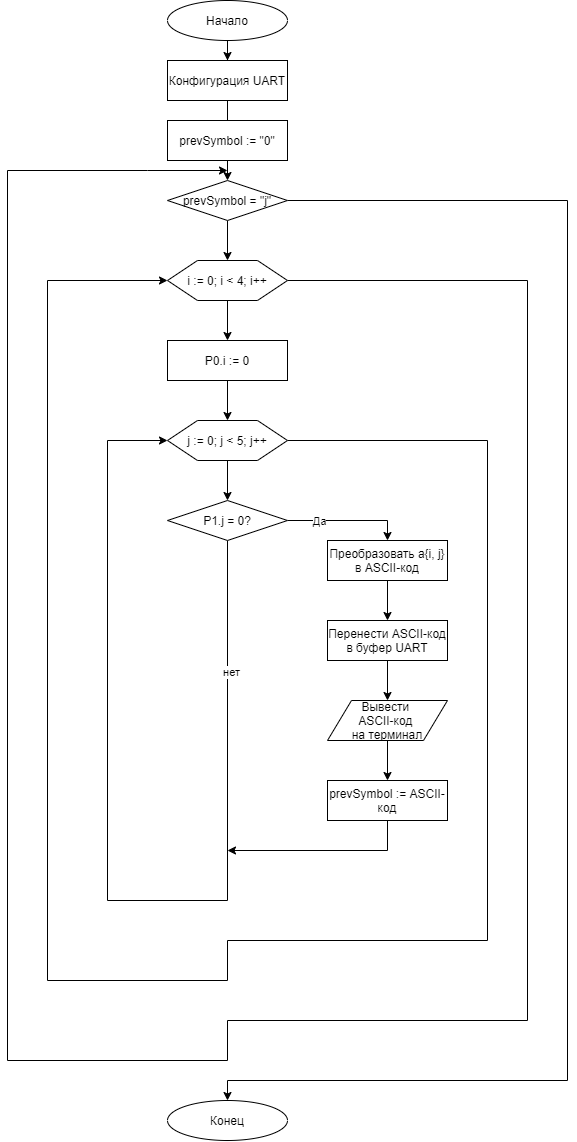
**Цель работы:** изучить принципы работы UART и организацию прерываний в микроконтроллерах серии MCS-51.

Задание к работе:

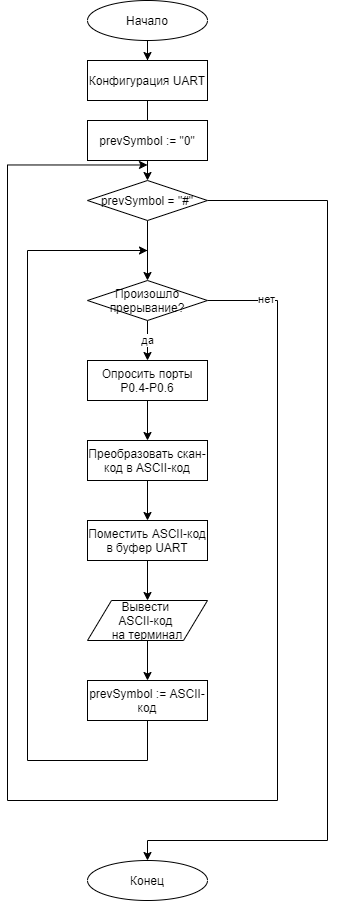
1. Создать программу для считывания скан-кода с клавиатуры, перевод его в ASCII код, и передачи полученного символа в терминал через UART путем постоянного опроса портов Р0, к которому подключена клавиатура.
2. Создать программу для считывания скан-кода с клавиатуры, перевод его в ASCII код, и передачи полученного символа в терминал через UART используя прерывания клавиатуры.

**Блок-схема разработанных программ**

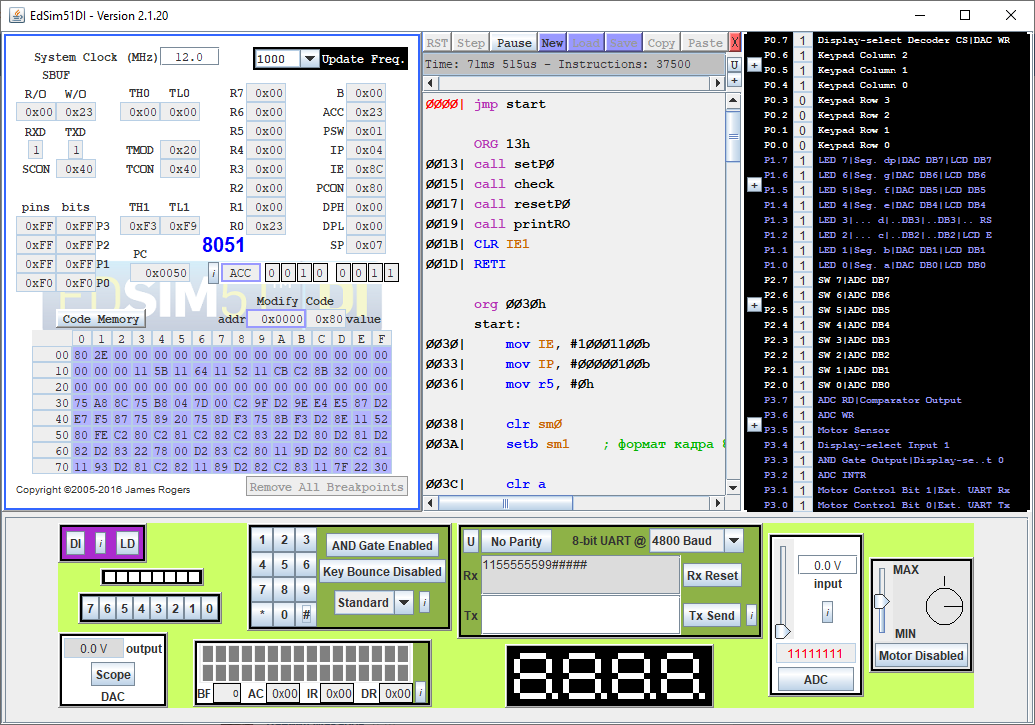
**Задача 1**

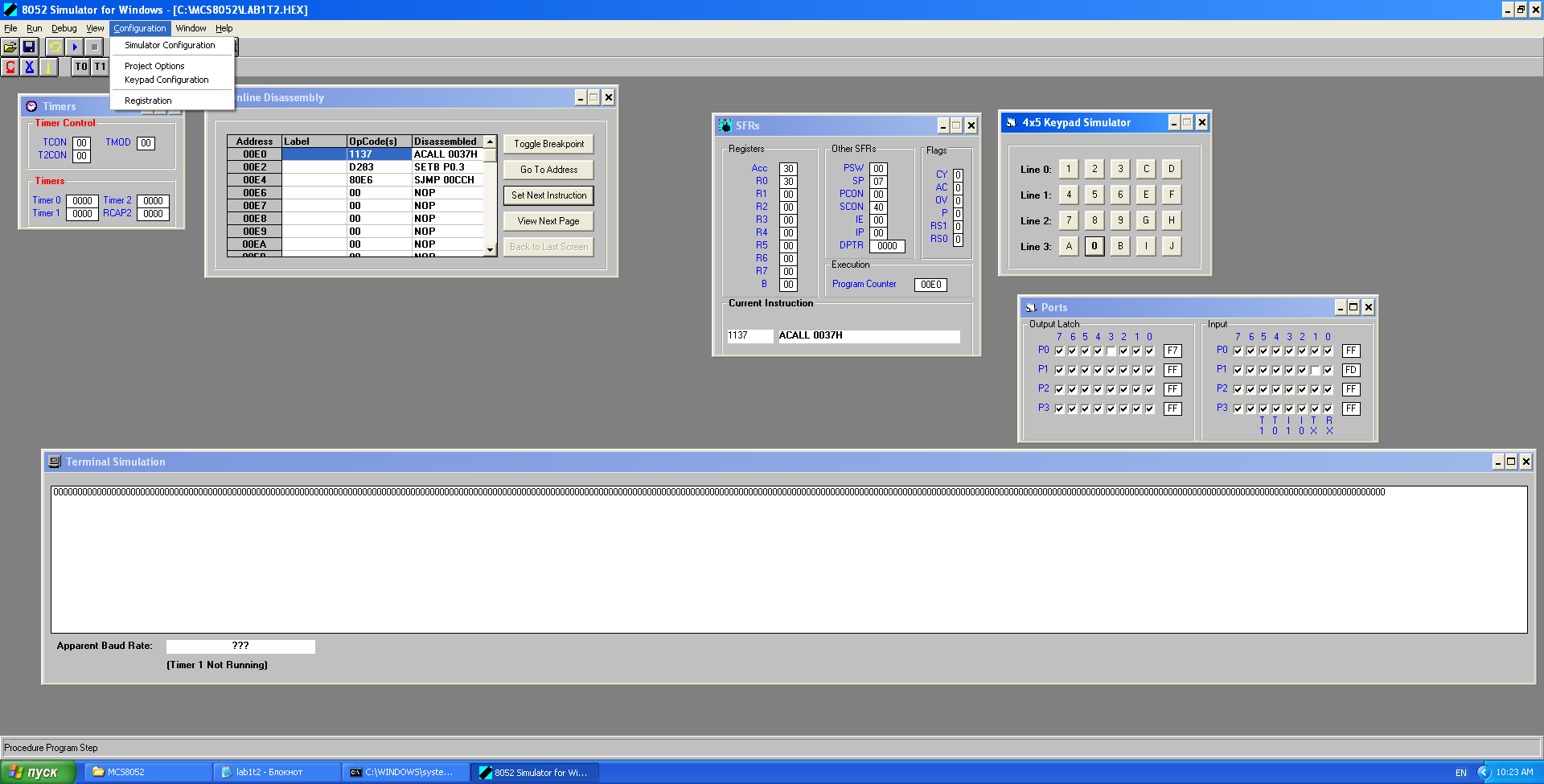


**Задача 2**



**Результаты работы программ:**





**Вывод:** В процессе выполнения лабораторной работы были изучены принципы работы UART, опроса клавиатуры методом полинга и организация прерываний в микроконтроллерах серии MCS-51.

**Задача №1:**

org 0h

CLR SM0

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

jmp start

start:

; Поочередно выставляем на каждой ноге 0 и сканируем столбец

; После каждой проверки возвращаем 1

CLR P0.0

CALL colScan0 ; вызываем опрос столбца 1

SETB P0.0

CLR P0.1

CALL colScan1 ; вызываем опрос столбца 2

SETB P0.1

CLR P0.2

CALL colScan2 ; вызываем опрос столбца 3

SETB P0.2

CLR P0.3

CALL colScan3 ; вызываем опрос столбца 4

SETB P0.3

jmp start ; И повторяем

end

; Поочередно сравниваем значения на каждой ноге P1.

; Если где то 0 - уже будем знать, какой символ нажат.

colScan0:

JNB P1.0, gotKey1

JNB P1.1, gotKey2

JNB P1.2, gotKey3

JNB P1.3, gotKeyC

JNB P1.4, gotKeyD

ret

colScan1:

JNB P1.0, gotKey4

JNB P1.1, gotKey5

JNB P1.2, gotKey6

JNB P1.3, gotKeyE

JNB P1.4, gotKeyF

ret

colScan2:

JNB P1.0, gotKey7

JNB P1.1, gotKey8

JNB P1.2, gotKey9

JNB P1.3, gotKeyG

JNB P1.4, gotKeyH

ret

colScan3:

JNB P1.0, gotKeyA

JNB P1.1, gotKey0

JNB P1.2, gotKeyB

JNB P1.3, gotKeyI

JNB P1.4, gotKeyJ

ret

; В r0 помещаем ASCII-код

gotKey0:

mov r0, #48

call print

ret

; Выводим ASCII-код, который записан в r0

print:

mov a, r0

MOV C, P

MOV ACC.7, C

MOV SBUF, A

JNB TI, $

CLR TI

ret

gotKey1:

mov r0, #49

call print

ret

gotKey2:

mov r0, #50

call print

ret

gotKey3:

mov r0, #51

call print

ret

gotKey4:

mov r0, #52

call print

ret

gotKey5:

mov r0, #53

call print

ret

gotKey6:

mov r0, #54

call print

ret

gotKey7:

mov r0, #55

call print

ret

gotKey8:

mov r0, #56

call print

ret

gotKey9:

mov r0, #57

call print

ret

gotKeyA:

mov r0, #65

call print

ret

gotKeyB:

mov r0, #66

call print

ret

gotKeyC:

mov r0, #67

call print

ret

gotKeyD:

mov r0, #68

call print

ret

gotKeyE:

mov r0, #69

call print

ret

gotKeyF:

mov r0, #70

call print

ret

gotKeyG:

mov r0, #71

call print

ret

gotKeyH:

mov r0, #72

call print

ret

gotKeyI:

mov r0, #73

call print

ret

gotKeyJ:

mov r0, #74

call print

ret

**Задача №2:**

ORG 0h

jmp start

; Вектор прерывания INT1

ORG 13h

call setP0 ; Устанавливаем p0.0-p0.4 в 1, чтобы опросить клавиатуру

call check ; Сканируем клавиатуру

call resetP0 ; Устанавливаем p0.0-p0.4 в 0, чтобы прерывания по INT1 вызывались

call printRO ; Выводим содержимое регистра r0, в котором хранится ASCII-код

CLR IE1

RETI

org 0030h

start:

; Настройка прерываний и UART

mov IE, #10001100b

mov IP, #00000100b

mov r5, #0h

clr sm0

setb sm1 ; формат кадра 8 бит

clr a

mov a, pcon

setb acc.7

mov pcon, a ; установка скорости

mov tmod, #20H ; восьмибитный таймер 1 с авто сбросом

mov th1, #243 ;

mov tl1, #243 ; переполнение таймера каждые 13 микросекунд

setb tr1

call resetP0

wait:

jmp wait

; Обнуляем биты p0.0-p0.3

resetP0:

CLR p0.0

CLR p0.1

CLR p0.2

CLR p0.3

ret

; Поднимаем биты p0.0-p0.3

setP0:

setb p0.0

setb p0.1

setb p0.2

setb p0.3

ret

check:

MOV R0, #0 ; Инициализируем r0, в который запишем ASCII-код нажатой кнопки

; Далее поочередно сканируем каждую строку

SETB P0.3

CLR P0.0

CALL colScan4

SETB P0.0

CLR P0.1

CALL colScan3

SETB P0.1

CLR P0.2

CALL colScan2

SETB P0.2

CLR P0.3

CALL colScan1

RET

; Выводим содержимое r0 в терминал

printRO:

mov a, r0

Jz printExit

mov sbuf, a

jnb ti, $

clr ti

printExit:

ret

colScan1:

JNB P0.4, gotKey3

JNB P0.5, gotKey2

JNB P0.6, gotKey1

RET

colScan2:

JNB P0.4, gotKey6

JNB P0.5, gotKey5

JNB P0.6, gotKey4

RET

colScan3:

JNB P0.4, gotKey9

JNB P0.5, gotKey8

JNB P0.6, gotKey7

RET

colScan4:

JNB P0.4, gotKeySharp

JNB P0.5, gotKey0

JNB P0.6, gotKeyAst

RET

gotKey0:

mov r0, #48

RET

gotKey1:

mov r0, #49

RET

gotKey2:

mov r0, #50

RET

gotKey3:

mov r0, #51

RET

gotKey4:

mov r0, #52

RET

gotKey5:

mov r0, #53

RET

gotKey6:

mov r0, #54

RET

gotKey7:

mov r0, #55

RET

gotKey8:

mov r0, #56

RET

gotKey9:

mov r0, #57

RET

gotKeyAst:

mov r0, #42

RET

gotKeySharp:

mov r0, #35

RET