ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

канд. техн. наук		Т. Н. Соловьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

РАЗРАБОТКА МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ПОСЛЕДОВАТЕЛЬНЫХ ИНТЕРФЕЙСОВ

по курсу: Микропроцессорные системы

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4143		Д. В. Пономарев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Вариант №4

Цель работы

Изучение принципов последовательной передачи данных, приобретение навыков разработки микроконтроллерных систем, использующих последовательные интерфейсы.

Индивидуальное задание

Требуется разработать микроконтроллерную систему «Калькулятор», включающую в себя микроконтроллер семейства MCS-51 и виртуальный терминал.

При включении системы на экране терминала выводится ФИО автора работы, после чего система переходит в состояние ожидания ввода данных (например, 15/3=). После ввода пользователем символа «=» система выводит результат или сообщение об ошибке, если таковая обнаружена.

Обмен данными с виртуальным терминалом осуществляется в формате 8-bit UART. Скорость обмена данными, а также операция, выполняемая калькулятором, указаны в разделе «Варианты заданий». Операнды являются целыми однобайтными числами без знака.

Операция: /

Скорость: 2400

Таймер: 2

Описание решения

Сначала рассчитаем RCAP2 и частоту.

RCAP2 = FF70h

F = 11.0592 МГц

Сначала выводим ФИО, потом ожидаем на ввод строку формата (15/3=), как только пользователь вводит «=», то рассчитываем результат и выводим его или выводим ошибку, если она возникла.

Для определения наличия слагаемых при обработке слагаемых инкрементируем значение r6 и r7 соответственно, и потом делаем проверку

на то, не равны ли регистры 0, если равны, то данное слагаемое не было введено.

Для определения ошибок при вводе слагаемых при обработке нового символа смотрим, был ли уже «/», если был, то ошибка во втором слагаемом, иначе в первом. Для отметки обнуляем регистр r0 или r3 соответственно. Также есть проверка на деление на 0. На данный момент в результат выводиться целая часть от деления.

Код программы

```
;=========================================================
=========
; Main.asm file generated by New Project wizard
; Created: Ср янв 24 2024
; Processor: 80C52
; Compiler: ASEM-51 (Proteus)
;=========================================================
=========
$NOMOD51
$INCLUDE (80C52.MCU)
;=========================================================
=========
; RESET and INTERRUPT VECTORS
;=========================================================
=========
; Reset Vector
org 0000h
   jmp MAIN

org 23h
        CLR TI; сброс флага прерывания
        INC DPTR
```

```
        CLR A
        MOVC A,@A+DPTR; чтение из памяти очередного символа
        CJNE A,#0h, hold; если не конец строки
finish:
        CLR ES ; запрет прерываний от послед. порта
        CLR EA
        RETI
hold:
        MOV SBUF,A; запуск передачи
        RETI

;=========================================================
=========
; CODE SEGMENT
;=========================================================
=========

org 100h
MAIN:
    MOV SCON, #050H  ; Режим 1, 8-битные данные, прием включен
    MOV RCAP2H,#0FFh; Скорость приема 38400
    MOV RCAP2L,#070h
    MOV SP, #7FH
    mov r2, #0h
        mov r0, #1h
        mov r3, #1h
        SETB RCLK
    SETB TCLK; Таймер 2 в режиме генератора скорости передачи
    SETB TR2; Запуск таймера
        SETB ES; разрешение прерываний от послед. порта
```

```
        SETB EA
        SETB ET2
        MOV DPTR,#400h
        CLR A
        MOVC A,@A+DPTR; чтение из памяти первого символа
        MOV SBUF,A; запуск передачи

LOOP:
        JNB RI, loop  ; Ожидание приема данных (RI - флаг приема)
    clr RI
    MOV A, SBUF     ; Считывание символа из SBUF
    cjne a, #'/', isNotDel
    mov r5, #1h
    sjmp loop

isNotDel: ; символ не /
    cjne a, #'=', isNotDel1
    sjmp checkFirst

isNotDel1:  ; символ не = и не /
    CJNE A, #2fh, not_zero ; Проверка, меньше ли символ '0'
    SJMP Loop

uncorrectFirst: ; ошибка при вводе первого слагаемого
        cjne r5, #0h, uncorrectSecond
        mov r0, #0h
        SJMP Loop

uncorrectSecond: ; ошибка при вводе второго слагаемого
        mov r3, #0h
```

```
        sjmp loop

not_zero: ; если значение не 0
    jc loop ; если hex символа меньше hex 0
    CJNE A, #3ah, NOT_DIGIT ; Проверка, больше ли символ '9'
    SJMP Loop

not_digit: ;если у нас не число
    Jnc uncorrectFirst   ; если hex символа больше hex 9, переходим на обработку ошибок

digit: ; работаем с первым слагаемым
    cjne r5, #0h, secondDigit
        inc r6
    subb a, #'0'
    mov r1, a
    cjne r2, #0h, DelInR1;
    mov r2, a
    mov a, r1
    SJMP LOOP

DelInR1: ; расчитываем первое слагаемое
    mov a, r2
    mov b, #0Ah
    mul ab
    add a, r1
    mov r2, a
    mov a, r1
    SJMP LOOP
```

```
secondDigit: ; работаем со вторым слагаемым
        inc r7
    subb a, #'0'
    mov r1, a
    cjne r4, #0h, DelInR4;
    mov r4, a
    SJMP LOOP

DelInR4: ;расчитываем второе слагаемое
    mov a, r4
    mov b, #0Ah
    mul ab
    add a, r1
    mov r4, a
    SJMP LOOP

checkFirst: ; проверка на наличие первого слагаемого
        cjne r6, #0h, checkSecond
        mov dptr, #500h
        sjmp error

checkSecond: ; проверка на наличие второго слагаемого
        cjne r7, #0h, checkDel
        mov dptr, #600h
        sjmp error

checkDel: ; проверка на наличие /
        cjne r5, #0h, checkSecondIsCorrect
        mov dptr, #700h
        sjmp error
```

```
checkFirstIsCorrect: ; проверка на корректность первого слагаемого
        cjne r0, #0h, calculate
        mov dptr, #800h
        sjmp error

checkSecondIsCorrect: ; проверка на корректность второго слагаемого
        cjne r3, #0h, checkSecondNotZero ; Проверяем наличие второго слагаемого
        mov dptr, #900h ; Если второе слагаемое нулевое, выводим сообщение об ошибке
        sjmp error

checkSecondNotZero: ; проверка на то, что второе число не равно нулю
        cjne r4, #0h, checkFirstIsCorrect ; Если второе число не равно нулю, производим расчет
        mov dptr, #950h ; Если второе число ноль, выводим сообщение об ошибке
        sjmp error

calculate: ; расчет результата
    mov a, r2
    mov b, r4
    div ab

pushDigits: ; по 1 цифре числа записываем в стек
    MOV B, #0Ah
    DIV AB
    push b
    jnz pushDigits
```

```
print: ; вывод значения
        CLR TI
        pop acc ; достаем цифры из стека
    mov a, acc
        add a, #'0'
    mov SBUF, a; передача цифры на терминал
    JNB TI, $; ожидаем сброс флага TI
    mov a, sp; Проверка, достигнут ли конец стека
    CJNE a, #7Fh, print ; Если не достигнут, продолжаем вывод
        clr tr2
        sjmp $;

error: ; вывод ошибки
        SETB ES; разрешение прерываний от послед. порта
        SETB EA
        CLR A
        MOVC A,@A+DPTR; чтение из памяти первого символа
        MOV SBUF,A; запуск передачи данных на терминал
        sjmp $

org 400h; ФИО
db 'Ponomarev D.V.: ',0h;

org 500h; ошибка: отсутствует первое слагаемое
db 'Not enter first number',0h;

org 600h; ошибка: отсутствует второе слагаемое
db 'Not enter second number',0h;
```

```
org 700h; ошибка: неизвестный оператор
db 'Unknown operation',0h;

org 800h; ошибка: первое слагаемое некорректное
db 'First number is uncorrect',0h;

org 900h; ошибка: первое слагаемое некорректное
db 'Second number is uncorrect',0h;

org 950h; ошибка:
db 'Second number = 0',0h;
END
```

Скриншот выполнения программы

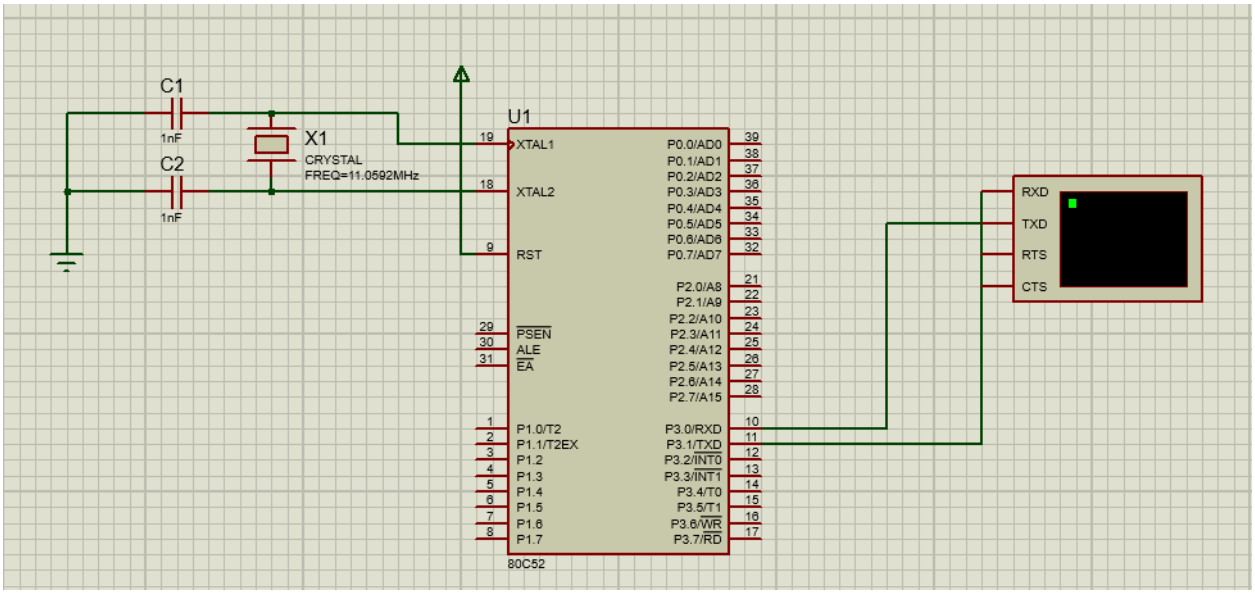

Рисунок 1 – Схема устройства

Рисунок 2 – Вывод ФИО

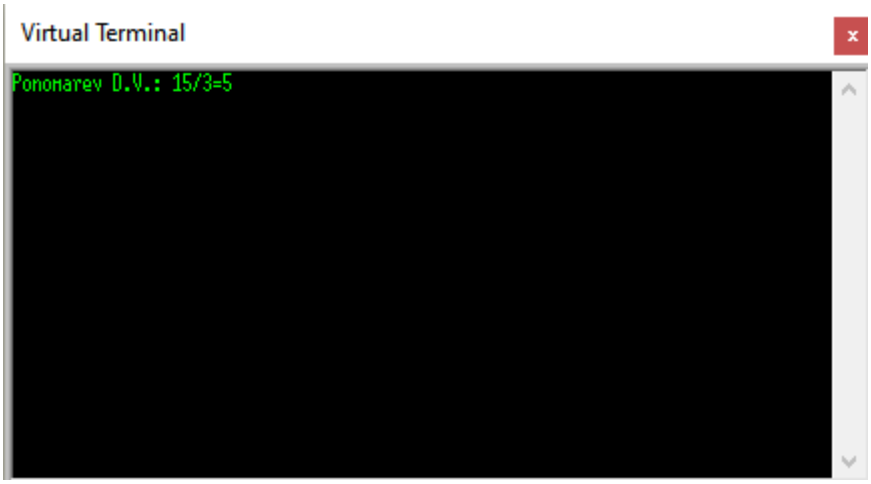

Рисунок 3 – Деление

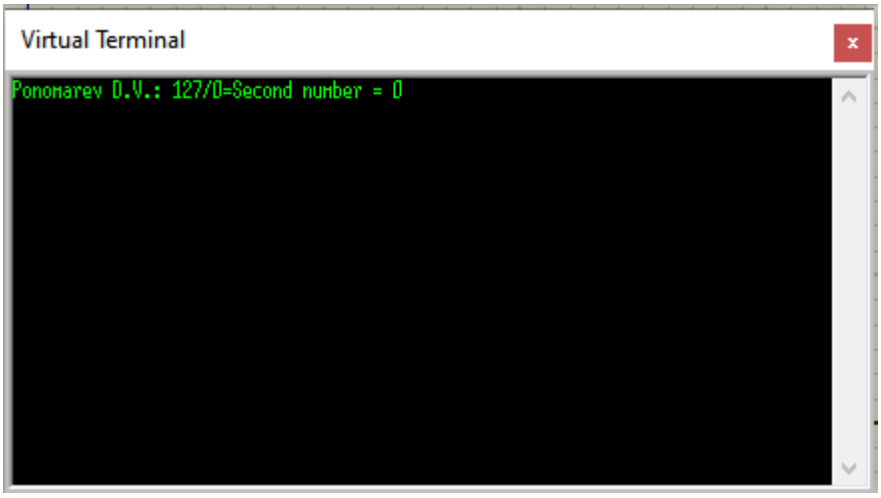

Рисунок 4 – Деление на 0

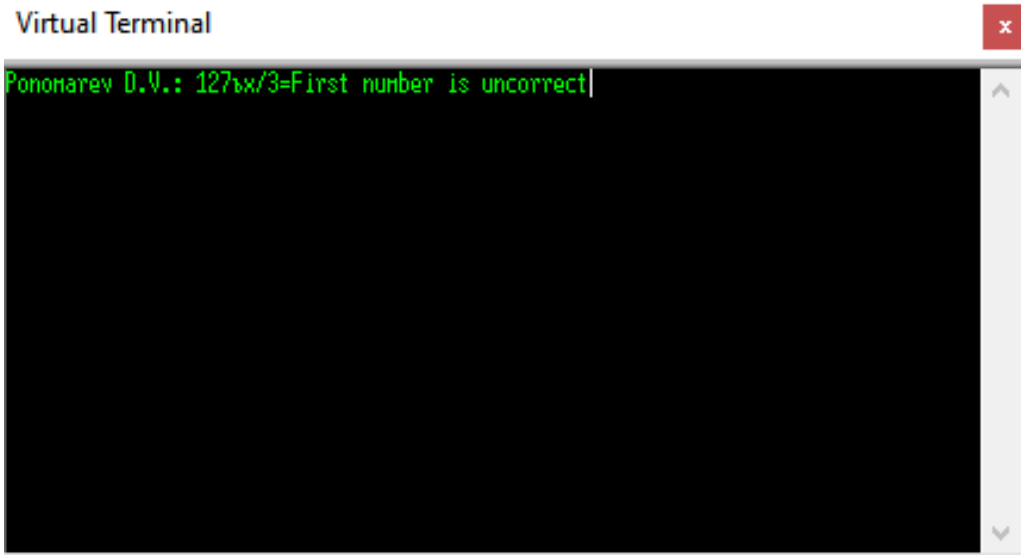

Рисунок 5 – Ошибка некорректный ввод слагаемых

Вывод

В результате выполнения лабораторной работы были изучены принципы последовательной передачи данных, приобретены навыки разработки микроконтроллерных систем, использующих последовательные интерфейсы.