

ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

канд. техн. наук

должность, уч. степень, звание

подпись, дата

Т.Н.Соловьёва

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

РАЗРАБОТКА МИКРОКОНТРОЛЛЕРНОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ПОСЛЕДОВАТЕЛЬНЫХ ИНТЕРФЕЙСОВ

по курсу: МИКРОКОНТРОЛЛЕРНЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4143

подпись, дата

Е.Д.Тегай

инициалы, фамилия

Санкт-Петербург 2024

Цель работы

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

Задание по работе

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, $15 \cdot 3 =$). 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. Разработка и моделирование системы производится в САПР Proteus.

Задание индивидуального варианта №9 продемонстрировано на рисунке 1, где для удобства восприятия необходимые данные выделены жёлтым цветом.

Номер варианта	Скорость UART, бит/с	Таймер	Операция
1	110	1	+
2	300	2	-
3	1200	1	*
4	2400	2	/
5	4800	1	
6	9600	2	&
7	19200	1	^
8	38400	2	+
9	57600	1	-

Рисунок 1 – Индивидуальное задание

Разработка схемы микроконтроллерной системы

Разработанная схема микроконтроллерной системы продемонстрирована на рисунке 2.

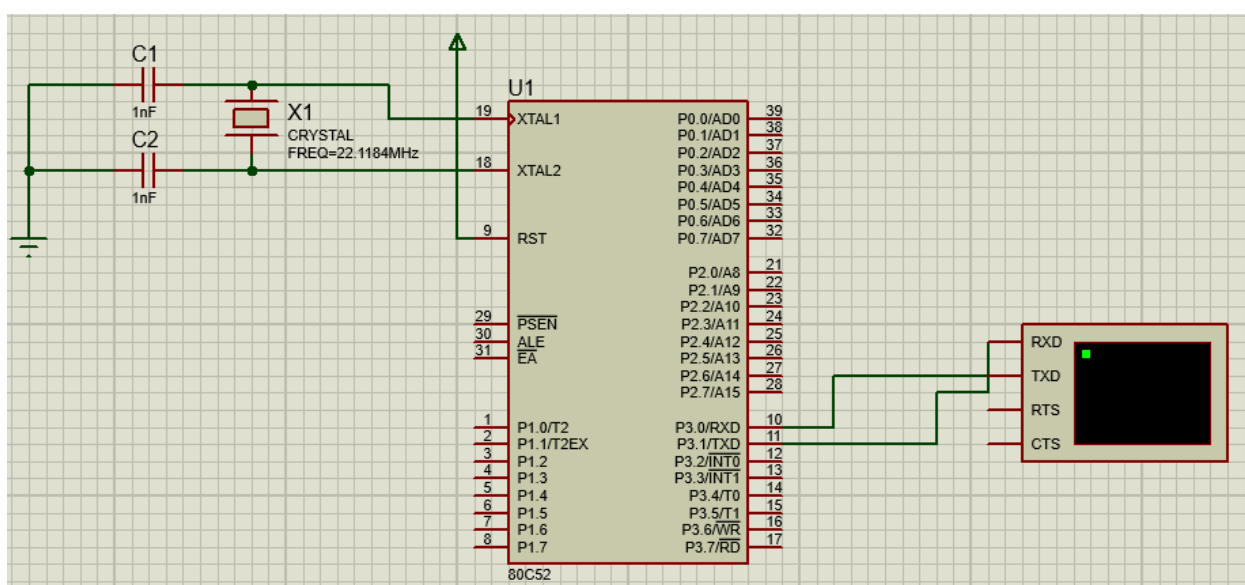
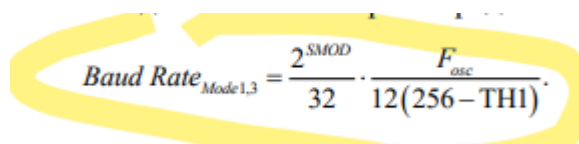


Рисунок 2 – Схема системы

Разработка программы

В начале необходимо привести некоторые вычисления, чтобы код и схема работали корректно. Как видно из рисунка 1, дана используемая скорость и таймер.

Далее нужно высчитать значение на TH1. Для этого используется формула, показанная на рисунке 3.



$$\text{Baud Rate}_{\text{Mode 1,3}} = \frac{2^{\text{SMOD}}}{32} \cdot \frac{F_{\text{osc}}}{12(256 - \text{TH1})}$$

Рисунок 3 – Используемая формула

При подставлении указанных значений получилось нецелое число (за частоту было взято число 12 МГц). Взяв целую часть, было получено число 255 (это FF в 16-ой системе счисления). Итого на TH1 будет передаваться число FF. В силу того, что число получилось нецелым, необходимо заново пересчитать по той же формуле эту частоту. Итого и была получена частота, равная 22.1184, которая указывается в свойствах микроконтроллера и кварцевого резонатора, тогда как скорость указывается в свойствах терминала.

Теперь рассмотрим более подробно разработанную программу. Для начала идёт указание директив препроцессора для управления процессом компиляции программы. Метка основной программы имеет название MAIN, она расположена по адресу 0000h. По адресу 23h расположен блок кода, который отвечает за загрузку данных из памяти в терминал. Данные загружаются, пока не будет обнаружен 0. Метка finish обозначает конец выполнения какой-либо процедуры. Метка hold предназначена для обработки прерывания и передачи содержимого регистра A через последовательный порт.

Метка MAIN инициализирует различные компоненты микроконтроллера, такие как UART и таймер 1, а также загружает данные из памяти и передаёт их через UART. Метка waiting обозначает начало блока кода, который ожидает появления данных в регистре приема RI, а затем читает данные из регистра SBUF и выполняет дальнейшие действия в зависимости от прочитанных данных. Метка isNotSub описывает блок кода, который выполняет в случае, если прочитанный символ не является «-». Метка isNotSub_1 описывает блок кода, который выполняется в том случае, когда прочитанный символ не является «=». Метка uncorMinuend описывает код, который выполняется в случае, если ведено недопустимое значение

уменьшаемого. Ту же суть описывает метка uncorSubtrahend. Метка notZero обозначает тот случай, когда введенный символ не является числом.

Метка dig предназначена для обработки ввода в том случае, когда введенный символ является числом. Метка subR1 предназначена для выполнения вычислений при обработке ввода чисел. Метка minuendDig ввод цифры уменьшаемого. Метка subR4 обрабатывает выполнение операции над числами в случае, когда R4 не равен 0.

Далее идут метки, которые проверяют корректность или наличие ввода уменьшаемого, вычитаемого и знака «-». Меткой, описывающей вычитание, является subtraction. Метка print отвечает за вывод результата, а error – за вывод ошибочного сообщения. Завершающим блоком кода являются различные сообщения, выводимые в случае различных ошибок.

Текст 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
    INC DPTR
    CLR A
    MOVC A,@A+DPTR
```

```

        CJNE A,#0h, hold
finish:
        CLR ES
        CLR EA
        RETI
hold:
        MOV SBUF,A
        RETI

```

```

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

```

```

org 100h
MAIN:
    MOV SCON, #050H
    MOV TMOD, #020H
    MOV TH1, #0FFH
    MOV SP, #7FH
    MOV r2, #0h
    MOV r0, #1h
    MOV r3, #1h
    SETB TR1
    SETB ES
    SETB EA
    MOV DPTR,#400h
    CLR A
    MOVC A,@A+DPTR
    MOV SBUF,A

```

```

waiting:
    JNB RI, waiting
    clr RI
    MOV A, SBUF
    cjne a, #'-', isNotSub
    mov r5, #1h
    sjmp waiting

```

```

isNotSub:
    cjne a, #'=', isNotSub_1
    sjmp subCheck

```

```

isNotSub_1:

```

CJNE A, #2fh, notZero
SJMP waiting

uncorMinuend:
 cjne r5, #0h, uncorSubtrahend
 mov r0, #0h
 SJMP waiting

uncorSubtrahend:
 mov r3, #0h
 sjmp waiting

notZero:
 jc waiting
 CJNE A, #3ah, notDig
 SJMP waiting

notDig:
 Jnc uncorMinuend

dig:
 cjne r5, #0h, minuendDig
 inc r6
 subb a, #'0'
 mov r1, a
 cjne r2, #0h, subR1;
 mov r2, a
 mov a, r1
 SJMP waiting

subR1:
 mov a, r2
 mov b, #0Ah
 mul ab
 add a, r1
 mov r2, a
 mov a, r1
 SJMP waiting

minuendDig:
 inc r7
 subb a, #30h
 mov r1, a
 cjne r4, #0h, subR4;
 mov r4, a

SJMP waiting

subR4:

```
mov a, r4
mov b, #0Ah
mul ab
add a, r1
mov r4, a
SJMP waiting
```

checkMinuend:

```
cjne r6, #0h, checkSubtrahend
mov dptr, #500h
sjmp error
```

checkSubtrahend:

```
cjne r7, #0h, subCheck
mov dptr, #600h
sjmp error
```

subCheck:

```
cjne r5, #0h, corSubtrahend
mov dptr, #700h
sjmp error
```

corMinuend:

```
cjne r0, #0h, subtraction
mov dptr, #800h
sjmp error
```

corSubtrahend:

```
cjne r3, #0h, corMinuend
mov dptr, #900h
sjmp error
```

subtraction:

```
mov a, r2
subb a, r4
```

pushDig:

```
MOV B, #0Ah
DIV AB
push b
jnz pushDig
```



```
print:
    CLR TI
    pop acc
    mov a, acc
    add a, #'0'
    mov SBUF, a
    JNB TI, $
    mov a, sp
    CJNE a, #7Fh, print
    sjmp $
```

```
error:
    SETB ES;
    SETB EA
    CLR A
    MOVC A, @A+DPTR;
    MOV SBUF,A;
```

```
org 400h; ФИО
db 'Tegai Ekaterina Dmitrievna:',0h;
```

```
org 500h; отсутствует уменьшаемое
db 'Error! No minuend ',0h;
```

```
org 600h; Отсутствует вычитаемое
db 'Error! No subtrahend',0h;
```

```
org 700h; Недопустимый оператор
db 'Error! Unknown operator',0h;
```

```
org 800h; Неправильно введенное уменьшаемое
db 'Error! Unknown minuend',0h;
```

```
org 900h; Неправильно введенное вычитаемое
db 'Error! Unknown subtrahend',0h;
```

END

Результаты работы программы

Результаты работы программы продемонстрированы на рисунках 4



Рисунок 1 – Пример 1



Рисунок 2 – Пример 2



Рисунок 3 – Отсутствие вычитаемого

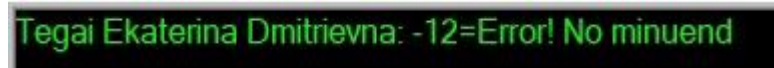


Рисунок 4 – Отсутствие уменьшаемого



Рисунок 5 – Некорректный ввод уменьшаемого

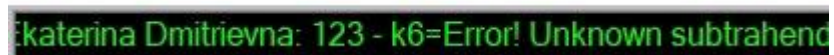


Рисунок 6 – Некорректный ввод вычитаемого



Рисунок 7 – Неизвестный оператор

Вывод

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