Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

АППАРАТНЫЕ СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

«Обработка сигналов клавиатуры, передача данных по UART»

Преподаватель:

Рафиков А.Г. _____
(подпись, дата)

Студент:

Девяткин Е.Д., группа ИУ8-74 (4 курс) _____
(подпись, дата)

Москва, 2025

Содержание

Цель работы

Изучение микроконтроллера 8051; работы с клавиатурой; подключение внешних устройств к микроконтроллеру и управление ими.

Задание

Организовать передачу данных по интерфейсу UART FD между 4 МК. Режим 1. Пос 8P1 (возможно ошибка). Скорость передачи данных 57600 бод. Топология сети передачи данных – звезда.

Теория

UART (Universal Asynchronous Receiver-Transmitter) – это аппаратный интерфейс для асинхронной последовательной передачи данных между устройствами.

Каждый бит каждого байта передаётся в равный отведённый промежуток времени (фактически, тайм-слот). Стандартным размером данных в посылке является 8 байт, но помимо данных каждый пакет несёт и служебную информацию, а именно:

- стартовый бит (Обязателен)
- стоповый бит (Также обязателен, возможно использование 1, 1.5, 2 стоповых битов)
- бит чётности (Необязателен. Бывает типов Odd, Even)

Так как интерфейс асинхронный, то большую значимость имеет скорость передачи данных – и у приёмника, и у передатчика она должна быть одинаковой.

Скорость измеряется в битах в секунду, или коротко – в бодах. Стандарт RS232 подразумевает скорости от 1200 до 115200 бод, хотя по факту существуют скорости и ниже, и выше, причём до десятков мегабод.

«Режим» UART 8051: Mode 1 – асинхронный 8-битовый UART с переменной скоростью, формат кадра старт-бит, 8 данных, стоп-бит; задаётся SCON с SM0=0, SM1=1 и включённым приёмом REN=1. В коде:

```
MOV SCON, #01010000b
```

В работе использовался тип передачи FD – реализуется с помощью 2 линий star0 и star1.

Кадровый формат – запись вида 8N1/8E1/7E1 и т.п. Число данных-бит, чётность, стоп-биты. 8P1 означает 8 данных, есть чётность, 1 стоп. На 8051 в Mode 1 «аппаратной» чётности нет, поэтому стандартно используют 8N1; если нужна чётность, применяют Mode 3.

Настройка скорость передачи настраивается с помощью бита SMOD и Таймера 1 по следующей формуле:

$f = 2^{SMOD} f_{РЕЗ}/(32*12*(256 – TH1))$.

Векторы: перенаправляют Reset на Init, INT0 на чтение кнопок, Timer0 на SERVE_TM, UART на SERVE_TR/CLR RI, и обслуживают TI.

Init: обнуляет флаги, готовит буферы, настраивает UART в режиме 1 (SM0=0, SM1=1, SM2=1, REN=1), включает прерывания и запускает таймеры.

SERVE_INT: читает снэпшот порт-кнопок в BUTTON и по нажатиям формирует DATA_ и флаги DATA_F/ADDR_F.

SERVE_TM: мультиплексирует индикацию семисегментников, выводя по очереди OUT_DATA, ADDR_OUT, SELF_ADDR_OUT на порт P2.

SERVE_TR: принимает байт из SBUF, разбирает адрес/данные по полубайтам, конвертирует в семисегментный код через таблицу OUTM, и кладёт в OUT_DATA/ADDR_OUT.

Start (главный цикл): опрашивает флаги, при BUTTON_F вызывает SERVE_INT, при DATA_F собирает байт к отправке и пишет его в SBUF, при ADDR_F обновляет целевой адрес, затем обновляет SELF_ADDR с P0 и производный SELF_ADDR_OUT.

Loop: считывает P0, инвертирует и берёт младшие 2 бита как SELF_ADDR, конвертирует адрес в семисегментный код через OUTM в SELF_ADDR_OUT, затем безусловно прыгает на Start для следующего цикла опроса и обработки флагов.

Таблица OUTM: соответствие 0…F → кодам семисегментного индикатора.

СХЕМА

Принцип работы схемы. Нужно нажать:

Адрес (0/1/2/3/9) → * → Число (0/1/2/3/4/5/6/7/8/9) → #

Получаем: Адрес блока, Число, Адрес откуда отправляли число.

Пояснение: Адрес 9 – чтобы на всех отобразить.

Выполнение работы

Задание 1

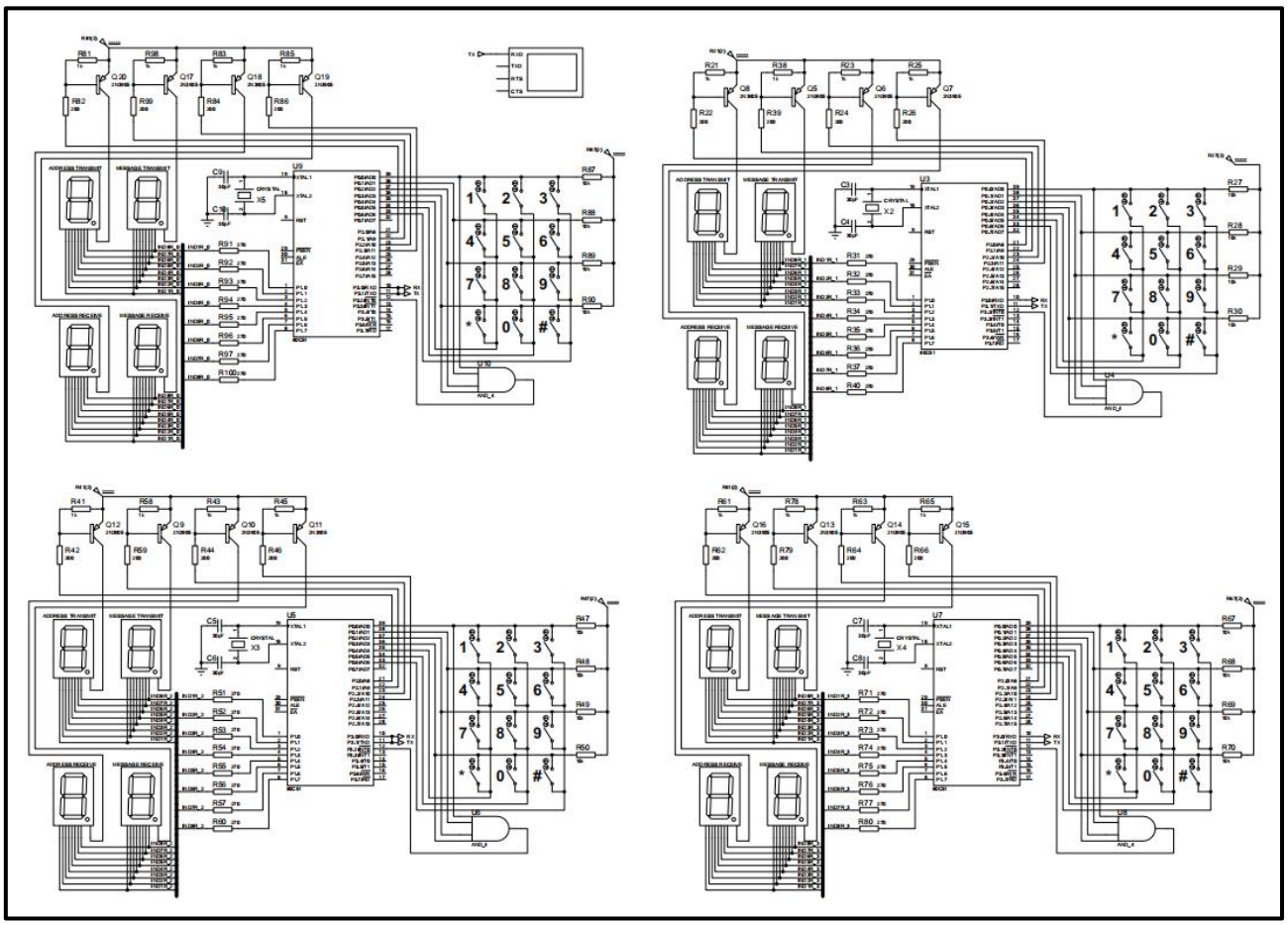

Рис. 1 - Схема.

```
$NOMOD51
$INCLUDE (8051.MCU)

;===================================================================
; DEFINITIONS
;===================================================================

; R0  ñîáñòâåííûé       àäðåñ
; R1  àäðåñ             èñõîäÿùèé (íà èíäèêàòîð)
; R2  ñîîáùåíèå   èñõîäÿùåå (íà èíäèêàòîð)
; R3  àäðåñ             ïðèíÿòûé (íà èíäèêàòîð)
; R4  ñîîáùåíèå   ïðèíÿòîå (íà èíäèêàòîð)
; R5  ïîñûëêà           èñõîäÿùàÿ (ñîîáùåíèå || àäð. îòïðàâèòåëÿ || àäð.
ïîëó÷àòåëÿ), ãää
;                               ñîîáùåíèå â äâîè÷íîì âèäå     = 4 áèòà
;                               àäð. îòïð. â äâîè÷íîì âèäå    = 2 áèòà - 00, 01, 10,
11
;                               àäð. ïîëó÷àòåëÿ â äâîè÷íîì âèäå = 2 áèòà - 01, 10,
11 èëè broadcast=ñîáñòâåííûé àäðåñ=00
; R6  ïîñûëêà           âõîäÿùàÿ (ñòðóêòóðà àíàëîãè÷íà R5)

;===================================================================
```

```
; VARIABLES
;=====================================================================

row1  BIT   P0.0
row2  BIT   P0.1
row3  BIT   P0.2
row4  BIT   P0.3

col1  BIT   P0.4
col2  BIT   P0.5
col3  BIT   P0.6

is_ad BIT   25H.0

cnt   DATA  20H
cnt1    DATA    22H
buff  DATA  21H
broad   DATA    23H
;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

ORG   0000H
JMP   Init

ORG   0003H
CLR   EX0
CALL  Check_keyboard
JB    IE0, $
SETB  EX0
RETI

ORG   0023H
CALL  Receive
RETI

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

ORG   0100H
Init:
      MOV   R0, #00000000B
      MOV   R1, #11111111B
      MOV   R2, #11111111B
      MOV   R3, #11111111B
      MOV   R4, #11111111B
      MOV   R5, #00000000B
      MOV   R6, #00000000B

      MOV   buff, #00000000B

      CLR   is_ad
      ; ðàçðåøèëè ïðåðûâàíèÿ
      SETB  EA
      SETB  EX0
      SETB  EX1

      MOV   SCON, #11010000B
      ;MOV  PCON, #10000000B
```

```
      ; set timer is second mode
      MOV   TMOD, #00100000B
      MOV   TH1, #11111111B
      SETB  TR1
      ; âîçìîæíîñòü ïðåðûâàíèé ñåðèàë
      SETB  ES

      CLR   col1
      CLR   col2
      CLR   col3

Start:
      CALL  Display

Loop:
      JMP   Start

Display:
      MOV   P2, #11111110B    ; àäðåñ èñòî÷íèêà
      CALL  Delay
      MOV   P1, #00000000B
      MOV   P1, R1

      MOV   P2, #11111101B    ; ñîîáùåíèå èñòî÷íèêà
      CALL  Delay
      MOV   P1, #00000000B
      MOV   P1, R2

      MOV   P2, #11111011B    ; àäðåñ ïðèíèìàòåëÿ
      CALL  Delay
      MOV   P1, #00000000B
      MOV   P1, R3

      MOV   P2, #11110111B    ; ñîîáùåíèå ïðèíÿòîå
      CALL  Delay
      MOV   P1, #00000000B
      MOV   P1, R4
      RET

Delay:
      MOV cnt, #0FH
L1:
      MOV cnt1, #0FH
L2:
      DJNZ cnt1, L2
      DJNZ cnt, L1
      RET

Check_keyboard:
      CLR   col1
      SETB  col2
      SETB  col3
      JNB   row1, Set_val_1             ; 1
      JB    row1, Set_not_val_1
      Set_val_1:
            MOV   A, #11111001B
            CALL  Handle
            RETI
      Set_not_val_1:                    ; 4
      JNB   row2, Set_val_4
```

```
      JB    row2, Set_not_val_4
      Set_val_4:
            MOV   A, #10011001B
            CALL  Handle
            RETI
      Set_not_val_4:                      ; 7
      JNB   row3, Set_val_7
      JB    row3, Set_not_val_7
      Set_val_7:
            MOV   A, #11111000B
            CALL  Handle
            RETI
      Set_not_val_7:                      ; *
      JNB   row4, Set_address
      JB    row4, Set_not_address
      Set_address:
            ; åñëè àäðåñ íå áûë ââåäåí, óñòàíîâèòü åãî êàê 0
            SETB  is_ad
            CJNE  R1, #11111111B, Skip_set_def_addr
            MOV   R1, #11000000B
            Skip_set_def_addr:
                  CALL  Return
                  RETI
      Set_not_address:
      SETB  col1
      CPL   col2
      SETB  col3
      JNB   row1, Set_val_2               ; 2
      JB    row1, Set_not_val_2
      Set_val_2:
            MOV   A, #10100100B
            CALL  Handle
            RETI
      Set_not_val_2:
      JNB   row2, Set_val_5               ; 5
      JB    row2, Set_not_val_5
      Set_val_5:
            MOV   A, #10010010B
            CALL  Handle
            RETI
      Set_not_val_5:
      JNB   row3, Set_val_8               ; 8
      JB    row3, Set_not_val_8
      Set_val_8:
            MOV   A, #10000000B
            CALL  Handle
            RETI
      Set_not_val_8:
      JNB   row4, Set_val_0               ; 0
      JB    row4, Set_not_val_0
      Set_val_0:
            MOV   A, #11000000B
            CALL  Handle
            RETI

      Set_not_val_0:
      SETB  col1
      SETB  col2
      CPL   col3
      JNB   row1, Set_val_3               ; 3
```

```
      JB    row1, Set_not_val_3
      Set_val_3:
            MOV   A, #10110000B
            CALL  Handle
            RETI
      Set_not_val_3:
      JNB   row2, Set_val_6               ; 6
      JB    row2, Set_not_val_6
      Set_val_6:
            MOV   A, #10000010B
            CALL  Handle
            RETI
      Set_not_val_6:
      JNB   row3, Set_val_9               ; 9
      JB    row3, Set_not_val_9
      Set_val_9:
            MOV   A, #10010000B
            CALL  Handle
            RETI
      Set_not_val_9:
      JNB   row4, Send_message            ; #
      JB    row4, Send_not_message
      Send_message:
            ; åñëè àäðåñ íå áûë óñòàíîâëåí, íå îòïðàâëÿòü
            ; åñëè ñîîáùåíèå íå áûëî óñòàíîâëåíî, óñòàíîâèòü åãî êàê 0
            JNB   is_ad, Skip_send_mes
            CJNE  R2, #11111111B, Skip_set_def_mess
            MOV   R2, #11000000B
            Skip_set_def_mess:
                  CLR   is_ad
                  CALL  Make_package
                  CALL  Send
                  MOV   R1, #11111111B
                  MOV   R2, #11111111B
            Skip_send_mes:
                  CALL  Return
                  RETI
      Send_not_message:
      RETI

Handle:
      JB    is_ad, Keep_message
      MOV   R1, A
      JMP   Return
      Keep_message:
              MOV R2, A
            JMP   Return

Return:
      CLR   col1
      CLR   col2
      CLR   col3
      RET

Make_package:
      ; ********
      MOV   A, R0
      RL    A
      RL    A
      ; ****AA**
```

```
      MOV   R5, A

      CJNE  R1, #11111001B, Not_addr_1
      MOV   A, R5
      XRL   A, #00000001B                ; 1
      ; ****AABB
      MOV   R5, A
      JMP   Decode_message
      Not_addr_1:
            CJNE  R1, #10100100B, Not_addr_2
            MOV   A, R5
            XRL   A, #00000010B                ; 2
            MOV   R5, A
            JMP   Decode_message
      Not_addr_2:
            CJNE  R1, #10110000B, broadcast
            MOV   A, R5
            XRL   A, #00000011B                ; 3
            MOV   R5, A
            JMP   Decode_message
      broadcast:
            MOV   A, R5
            XRL   A, #00000000B                ; ØÈÐÎÊÎÂÅÙÀÒÅËÜÍÛÉ
            MOV   R5, A

      Decode_message:
      CJNE  R2, #11000000B, Not_mess_0
      MOV   A, R5
      XRL   A, #00000000B                      ; 0
      MOV   R5, A
      RET
      Not_mess_0:
            CJNE  R2, #11111001B, Not_mess_1
            MOV   A, R5
            XRL   A, #00010000B                ; 1
            MOV   R5, A
            RET
      Not_mess_1:
            CJNE  R2, #10100100B, Not_mess_2
            MOV   A, R5
            XRL   A, #00100000B                ; 2
            MOV   R5, A
            RET
      Not_mess_2:
            CJNE  R2, #10110000B, Not_mess_3
            MOV   A, R5
            XRL   A, #00110000B                ; 3
            MOV   R5, A
            RET
      Not_mess_3:
            CJNE  R2, #10011001B, Not_mess_4
            MOV   A, R5
            XRL   A, #01000000B                ; 4
            MOV   R5, A
            RET
      Not_mess_4:
            CJNE  R2, #10010010B, Not_mess_5
            MOV   A, R5
            XRL   A, #01010000B                ; 5
            MOV   R5, A
```

```
            RET
      Not_mess_5:
            CJNE  R2, #10000010B, Not_mess_6
            MOV   A, R5
            XRL   A, #01100000B                ; 6
            MOV   R5, A
            RET
      Not_mess_6:
            CJNE  R2, #11111000B, Not_mess_7
            MOV   A, R5
            XRL   A, #01110000B                ; 7
            MOV   R5, A
            RET
      Not_mess_7:
            CJNE  R2, #10000000B, Not_mess_8
            MOV   A, R5
            XRL   A, #10000000B                ; 8
            MOV   R5, A
            RET
      Not_mess_8:
            CJNE  R2, #10010000B, Not_mess_9
            MOV   A, R5
            XRL   A, #10010000B                ; 9
            MOV   R5, A
            RET
      Not_mess_9:
            MOV   A, R5
            XRL   A, #11100000B                ; E
            MOV   R5, A
      RET

Send:
      CLR   REN
      MOV   A, R5
      MOV   C, PSW.0
      MOV   TB8, C
      MOV   SBUF, A
      JNB   TI, $
      CLR   TI
      SETB  REN
      MOV   R5, #00000000B
      RET

Receive:
      CLR   RI
      MOV   A, SBUF
      MOV   C, RB8
      ; ïðîâåðêà ïîñûëêè ñ ïîìîùüþ áèòà ÷åòíîñòè
      ANL   C, PSW.0
      MOV   F0, C       ; RB8 /\ PSW.0
      MOV   C, PSW.0
      CPL   C
      ANL   C, /RB8           ; /RB8 /\ /PSW.0
      ORL   C, F0
      JNC   Recieve_error

      CALL  Parse_package

      RETI
      Recieve_error:
```

```
            MOV   R3, #10000110B
            MOV   R4, #10000110B
            RETI

Parse_package:
      MOV   R6, A
      MOV   buff, R0
      MOV     broad, R6
      ANL     broad, #00001100B
      ANL   A, #00000011B
      RL    A
      RL    A
      CJNE  A, broad, not_broad
      CALL  Get_fields_from_package
      ;MOV  A, R6
      ;ANL  A, #00000011B
      ;CJNE A, buff, Resend_package
      RET
not_broad:
      MOV   A, R6
      ANL   A, #00000011B
      CJNE  A, buff, Resend_package
      CALL  Get_fields_from_package
      RET
      Resend_package:
      RET

Get_fields_from_package:
      MOV   A, R6
      ANL   A, #00001100B
      CJNE  A, #00000000B, Not_sender_addr_0
      MOV   R3, #11000000B
      CALL  Get_message_from_package
      RET

      Not_sender_addr_0:
            CJNE  A, #00000100B, Not_sender_addr_1
            MOV   R3, #11111001B
            CALL  Get_message_from_package
            RET
      Not_sender_addr_1:
            CJNE  A, #00001000B, Not_sender_addr_2
            MOV   R3, #10100100B
            CALL  Get_message_from_package
            RET
      Not_sender_addr_2:
            CJNE  A, #00001100B, Not_sender_addr_3
            MOV   R3, #10110000B
            CALL  Get_message_from_package
            RET
      Not_sender_addr_3:
            MOV   R3, #10000110B
            RET

Get_message_from_package:
      MOV   A, R6
      ANL   A, #11110000B
      CJNE  A, #00000000B, Not_sent_message_0
      MOV   R4, #11000000B
      RET
```

```
      Not_sent_message_0:
            CJNE  A, #00010000B, Not_sent_message_1
            MOV   R4, #11111001B
            RET
      Not_sent_message_1:
            CJNE  A, #00100000B, Not_sent_message_2
            MOV   R4, #10100100B
            RET
      Not_sent_message_2:
            CJNE  A, #00110000B, Not_sent_message_3
            MOV   R4, #10110000B
            RET
      Not_sent_message_3:
            CJNE  A, #01000000B, Not_sent_message_4
            MOV   R4, #10011001B
            RET
      Not_sent_message_4:
            CJNE  A, #01010000B, Not_sent_message_5
            MOV   R4, #10010010B
            RET
      Not_sent_message_5:
            CJNE  A, #01100000B, Not_sent_message_6
            MOV   R4, #10000010B
            RET
      Not_sent_message_6:
            CJNE  A, #01110000B, Not_sent_message_7
            MOV   R4, #11111000B
            RET
      Not_sent_message_7:
            CJNE  A, #10000000B, Not_sent_message_8
            MOV   R4, #10000000B
            RET
      Not_sent_message_8:
            CJNE  A, #10010000B, Not_sent_message_9
            MOV   R4, #10010000B
            RET
      Not_sent_message_9:
            MOV   R4, #10000110B
              RET

;===================================================================
END
```