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 №6

По дисциплине: «Аппаратные средства вычислительной техники»

Тема: «UART. Микроконтроллер PIC12F675»

Вариант №3

Выполнил: Березин М.А.
студент группы ИУ8-73

Проверил: Рафиков А. Г.,
Старший преподаватель кафедры
ИУ

г. Москва,

2021 г.

1. Цель работы

Изучить принципы построения систем взаимодействия нескольких микроконтроллеров PIC12F675 с использованием встроенного механизма передачи данных по протоколу UART.

2. Задание

Согласно варианту 3, данные представлены в таблице 1.

Параметр	Значение	Комментарий
Режим UART	2	Аппаратный
Режим передачи	Полудуплексный	Передача в одну сторону
Количество микроконтроллеров	4	
Архитектура сети	Общая Шина	
Режим UART	9N1	(·)
Скорость передачи данных	$\frac{19200}{2}$ бит/с	

Примечение (·): 9 бит данных, нет бита четности, 1 бит останова.

3. Теоретическая часть

$$F_{clk} = 4 \text{ кГц}$$

Для реализации аппаратного UART, требуется создать канал связи, реализовать синхронизацию отправляемого пакета и считывания пакета.

Чтобы реализовать скорость передачи в 9600 бит/с, надо добавить задержку с помощью таймера T0. $TMR0 = 255 - \frac{1}{9600} \cdot 10^6 = 255 - 104 = 151$.

Но есть одно но – надо добавить задержку на время выполнения обработки подпрограмм вызова прерывания таймера T0 и т.д. Итого, опытным путём было выяснено, что для задержки на 1 бит, надо установить TMR0 = 201 (десятично).

Для задержки на пол бита во время чтения вообще получилось так, что таймер не подошёл – была просто добавлена задержка с помощью NOP-ов (20 штук).

4. Практическая часть

Для реализации клавиатуры была добавлена кнопка INC – инкрементация значения. Кнопкой SEND мы фиксируем данные.

Как работает сам алгоритм схемы:

1. Изначально переключим ключ в режим набора сообщения.
2. Потом нажмём кнопку SEND – на индикаторе выведется «-» - ожидание набора данных.
3. Нажмём INC – выведется «0» - в регистре «адреса» будет записан «0». Потом нажимая INC раз за разом, будем увеличивать число. Когда перейдёт число через «9», будет снова «-» и потом снова «0».
4. Нажимаем кнопку SEND – фиксируется адрес и выведется сразу «0» - ожидание ввода ДАННЫх.
5. Нажимаем INC пока не будут нужные данные.
6. Нажимаем SEND – фиксируются данные. И отключаются все порты RX и TX.
7. Переводим ключ и нажимаем кнопку SEND – отправляются подряд два пакета (с перерывом в 255 мкс).
8. Когда происходит отправка – слушатель получает отрицательный фронт (стартовый бит), и затем начинается отсчёт на пол бита. Чтобы дальше попадать ровно на средину отправки и читать в этот момент бит.
9. Отсчёт на 1 бит – и считывание пакета в 8 бит.
10. Потом задержка и чтение 9 бита.
11. Если 9 бит = 1 – адрес пришёл. Если 9 бит = 0 – это данные.
12. Дальше необходима проверка, что ожидается – адрес/данные. Если совпадает – Ок. Иначе – пропуск сообщения.
13. Если адрес – надо проверить, что это число, младшие 4 бита, совпадают с индексом(номером) текущего микроконтроллера. Если нет – проверяем, что это число > 3, т.е. 3 – 9. Если да – то это попросту широковещательная рассылка.
14. В старших 4 битах отправляется всегда номер отправителя – чтобы определить от кого данные.
15. Потом происходит старт таймера T1 – для демонстрации сообщения. Но выключены изначально и вывод и прерывания по T1. Ожидается когда

будет произведено переключение ключа. Потом нажимается кнопка SEND – и происходит одно из.

16. Если ничего не было (не приходило, или мы НЕ знаем об этом, что логично в целом) – то мы нажимаем кнопку SEND для набора сообщения. Но если к этому моменту нам пришло сообщение – то по нажатию не будет включён режим «ввода», а будет выводиться сообщение – запустится таймера T1, и будет постепенно выводится динамически сообщение следующего формата: “A - <адрес> d - <данные>”.
17. По ещё одному нажатию кнопки SEND микроконтроллер вернётся в изначальное состояние.

Выше был описан цикл работы (алгоритм) программного UART. На рисунке 1 изображена схема.

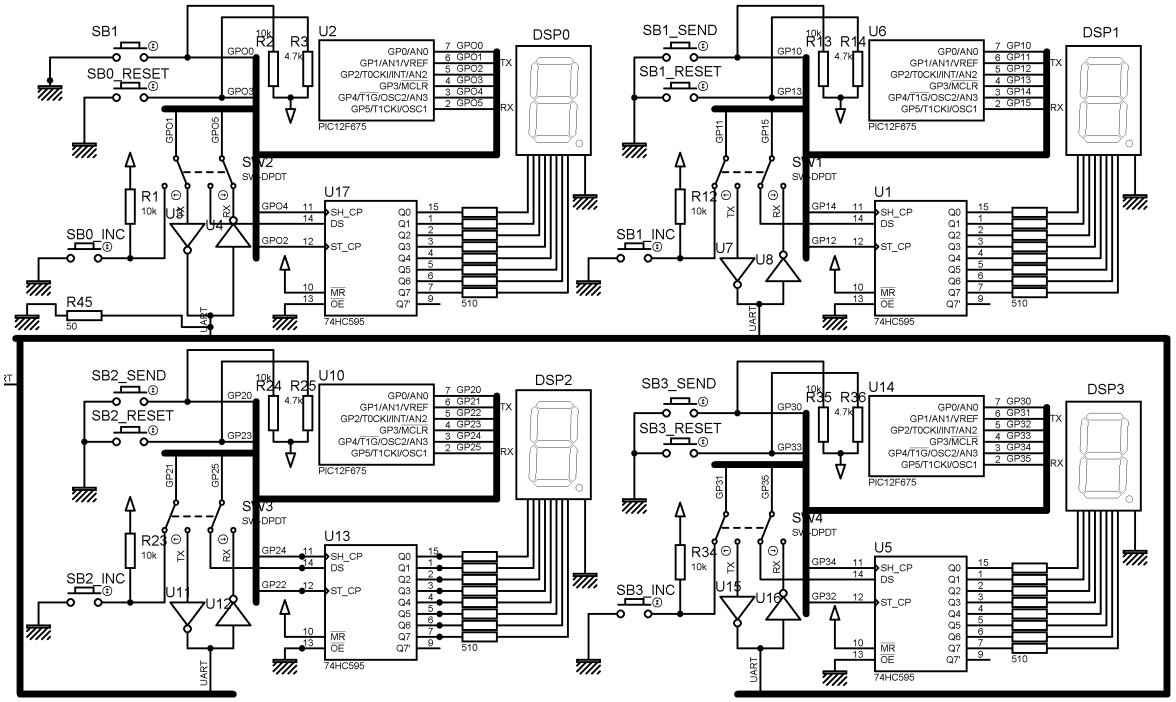

Рисунок 1 – Схема лабораторной работы

На рисунках 2-4 изображены скриншоты выводимых данных (индикации).

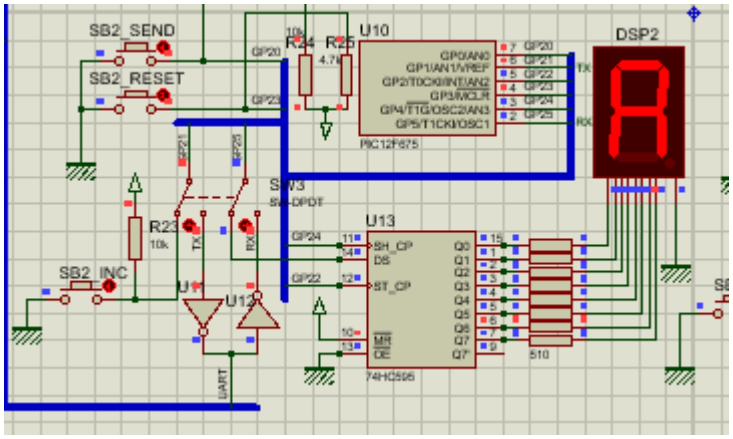

Рисунок 2 – Индикация входящего сообщения (часть 1 – «адрес»)

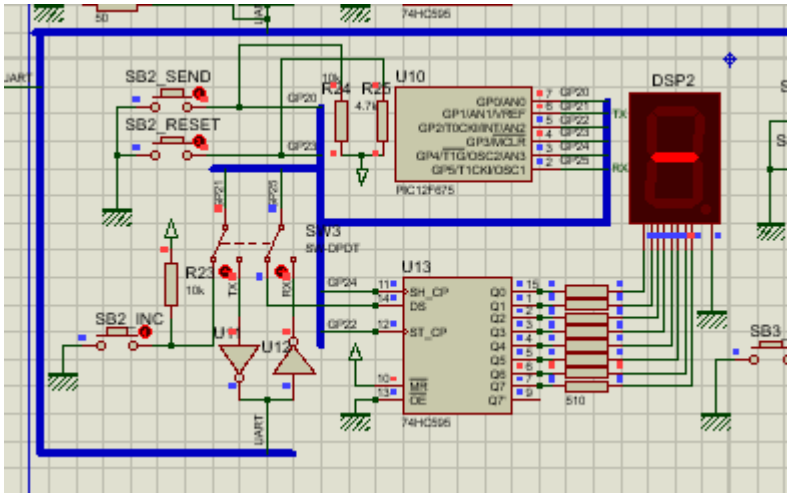

Рисунок 3 – Индикация входящего сообщения (часть 2)

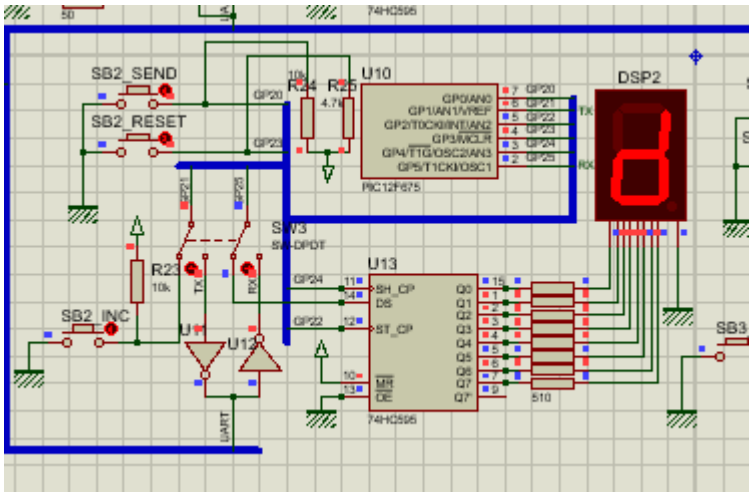

Рисунок 4 – Индикация входящего сообщения (часть 3 – «данные»)

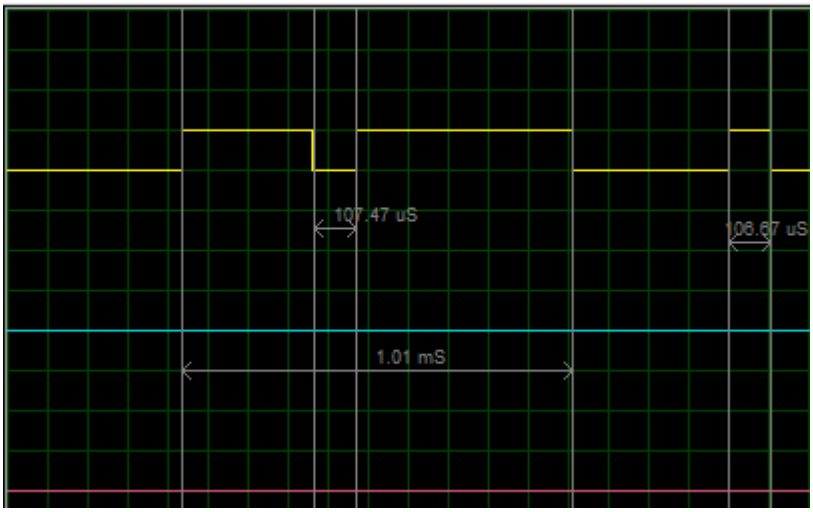

Рисунок 5 – Осциллограмма посылки

5. Выводы

Было изучено поведение прерываний в микроконтроллере PIC12F675. Получены практические навыки в области программной реализации протоколов данных или если быть правильнее, именно Шины.

Были выполнены все поставленные задания и подготовлена демонстрация схемы.

По выполненной работе был составлен отчёт.

ПРИЛОЖЕНИЕ А

Листинг для задания (пример для 1 процессора)

```
;=====================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Вс ноя 14 2021
; Processor: PIC12F675
; Compiler:  MPASM (Proteus)
;=====================================================================

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

#include p12f675.inc                ; Include register definition file
__CONFIG b'11111111110100'

;=====================================================================
; CONSTANTS
;=====================================================================

GPIO_DATA equ b'000010'
INTERRUPT_INIT equ b'11001000'
PIE1_INIT equ b'00000000'
TMR1H_INIT equ 0x0
TMR1L_INIT equ 0x0
TMR0_INIT equ 0x0
WPU_INIT equ b'110100'   ;b'110100'                     ;GPIO1 И GPIO5 - мбб как READ так и WRITE режим. Для
кнопки (изначально!) и TX; Для RX и Data соответственно.
IOC_INIT equ b'100001'
T1CON_INIT equ b'00000000'
TRISIO_INT equ b'00101011' ; b'00101001'
;=====================================================================
; VARIABLES
;=====================================================================

UART_CON equ 20h                        ; ИЗНАЧАЛЬНО ДОЛЖЕН БЫТЬ РЕЖИМ
ПРОСЛУШИВАНИЯ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        BUSY_F equ 00h
        TR_F equ 01h
        RD_F equ 02h
        T0_MODE equ 03h
        DATA_INC_F equ 04h
        BIT9 equ 05h
        SEG7_Display_F equ 06h
        BanReceiveZ9Bit_F equ 07h;  Флаг запрета приёма сообщений,в котором 9 бит имеет значение 0 (т.е. если
это ДАННЫЕ)
        ;SaveDataOrAddr equ 07h         ; || 0 - Addr | 1 - Data ||

UART_t0 equ 21h
        WAIT_HalfBaud equ 00h
        WAIT_FullBaud equ 01h
        Simple_TMR equ 02h
        T1_Busy_on_display equ 03h
;LEDS_STATES equ 21h
;LED1 equ 00h
;LED2 equ 01h
;LED3 equ 02h



btn_SEND equ GPIO0
btn_RST equ GPIO3
btn_INC equ GPIO1

;pc_states equ 21h
;in_work_flag equ 00h
;btn_wait_flag equ 01h
```

```
;btn2_st equ 02h
;btn3_st equ 03h

T1H_REG equ 22h
T1L_REG equ 23h

T02SecScaler equ 24h

LastPortState equ 25h

Accum equ 26h

Lcd_data equ 27h

Loop_ident equ 28h

SERBUF equ 29h          ; TODO
TEMP equ 2Ah

TX equ GPIO1
RX equ GPIO5



LastPressedBtns equ 2Bh         ;  || 0 bit - SB_SEND | 1 bit - SB_INC/TX | 2 bit - RX/DATA_7_SEG ||
SB_SEND_Ch equ 00h
SB_INC_Ch equ 01h
;--

ADDR_reg equ 2Ch
DATA_reg equ 2Dh

ButtonModes equ 2Eh
        SB0_M0  equ 00h
        SB0_M1  equ 01h


SEV_SEGM_reg equ 2Fh

GotMsgDispMode equ 30h


;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

      ; Reset Vector
RST   code  0x0
      goto  START

INT code 0x4
        goto IINT_HNDL


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

PGM   code

Init

        banksel ANSEL
    clrf ANSEL

    banksel CMCON
    movlw 0x7
    movwf CMCON

          banksel GPIO
    ;movlw b'10' ; Default GPIO data
```

```
    movlw GPIO_DATA
movwf GPIO

     banksel TRISIO
movlw TRISIO_INT   ; GPIO Port options
movwf TRISIO

banksel OPTION_REG
movlw b'01001000'       ; Options
movwf OPTION_REG




banksel WPU
movlw WPU_INIT  ; Pull ups
movwf WPU

banksel INTCON
movlw INTERRUPT_INIT   ; Interrupts
movwf INTCON

banksel PIE1
movlw PIE1_INIT  ; Enables timer 1 (16-bit) INTERRRUPT (NO)
movwf PIE1

banksel TMR1H
movlw TMR1H_INIT
movwf TMR1H
movlw TMR1L_INIT
movwf TMR1L

banksel T1CON
movlw T1CON_INIT    ; Enables timer 1
movwf T1CON

banksel IOCB
movlw IOC_INIT
movwf IOCB

    ;CALL SavePortStates
    movf GPIO, w
    movwf LastPortState

    clrf UART_CON
    bsf UART_CON, BanReceiveZ9Bit_F

    clrf UART_t0

    clrf T02SecScaler

    clrf LastPortState

    clrf Accum

    clrf Lcd_data

    clrf Loop_ident

    clrf SERBUF
    clrf TEMP

    clrf LastPressedBtns

    clrf ADDR_reg
    clrf DATA_reg

    clrf ButtonModes

    clrf SEV_SEGM_reg
```

```
        clrf GotMsgDispMode


    return

START
        CALL Init
        MainLoop
                GOTO MainLoop



IINT_HNDL
        banksel INTCON
        btfss INTCON, GPIE
                GOTO Check_INT_T0
                btfsc INTCON,  GPIF
                        GOTO GPIO_INT
        Check_INT_T0
                banksel INTCON
                btfss INTCON, T0IE
                        GOTO Check_INT_T1
                        btfsc INTCON, T0IF
                                GOTO T0_INT
        Check_INT_T1
                banksel PIE1
                btfss PIE1, TMR1IE
                        return
                        banksel PIR1
                        btfsc PIR1, TMR1IF
                                GOTO T1_INT
                                retfie

FindPortDiff
        ; Допустим, состояние порта записывается в регистр LastPortState
        Port_GP0
                banksel IOCB
                btfss IOCB, GPIO0
                        GOTO Port_GP1
                banksel GPIO
                btfsc GPIO, btn_SEND; Если нажата кнопка - то (0)
                        GOTO Port_GP1 ; Если (1)
                        btfss LastPortState, btn_SEND; Если (0) (Если в LastPortState и был 0 - то забиваем,
иначе - нажата кнопка)
                                GOTO Port_GP1; Если остаётся быть нажатой
                                bsf LastPressedBtns, SB_SEND_Ch; Если произошло на самом деле нажатие
        Port_GP1 ; ТОЛЬКО В РЕЖИМЕ ДО отправки {TODO}
                banksel IOCB
                btfss IOCB, GPIO1
                        GOTO Port_GP5
                banksel GPIO
                btfsc UART_CON, DATA_INC_F
                        GOTO Check_SB_INC; Если 1
                        GOTO Port_GP5; Если 0
                Check_SB_INC
                        btfsc GPIO, btn_INC ; Если нажата кнопка - то (0)
                                GOTO Port_GP5
                                btfss LastPortState, btn_INC ; Если в LastPortState и был 0 - то забиваем,
иначе - нажатая кнопка
                                        GOTO Port_GP5 ; Если остаётся
                                        bsf LastPressedBtns, SB_INC_Ch ; Если на самом деле произошло нажатие
        Port_GP5
                banksel IOCB
                btfss IOCB, GPIO5
                        return
                        ; Если GPIO5 в режиме чтения
                        banksel GPIO
                        btfsc GPIO, RX
                                return  ; 1 - сё ок
                                bsf UART_CON, RD_F      ; Пришёл отрицательный флаг
```

```
                        bsf UART_CON, BUSY_F
                        return

        ; Нужна проверка, что RX в режиме чтения (мб и не надо, так как не срабатывает прерывание по GPIO во время "вывода")
        ; По СПАДУ вызывается обработка приходящего сообщения
                ;btfsc UART_CON, SEG7_Display_F
                ;       return; Если происходит вывод - то не может быть никак подключен порт к RX
                ;       bsf UART_CON, RD_F; Если не вывод - то мб и RX (устанавливаем флаг чтения)
                return
        ;       btfsc UART_CON, SEG7_Display_F
        ;               return ; Если вывод на 7-сегментник
        ;               ; Если Чтение с RX

SB0_SEND_HNDL
        ; 4 нажатия
        ; ИЗНАЧАЛЬНО не горит (если не было присланных данных) и не ждёт данные никакие
        ; 1(00) - активирует считывание адреса (пользователь нажимает на кнопку INC для изменения адреса отправителя)

        ; Если включён таймер T0 - значит надо обрабатывать только нажатия на приходящие данные - то есть переключать когда требуется вывести приходящее сообщение.
        ; Изначально ключ настроен на UART. По нажатию на кнопке, если что-то пришло пользователю - ему высветится принятое сообщение. По второму нажатию выводиться перестанет и
        ; микроконтроллер снова вернётся в стандартный режим

        btfsc UART_CON, BUSY_F
                        GOTO SB0_ifBusy; Если по какой-то причине UART занят - или обработка входящего потока, либо уже обработка отправляемого сообщения
                        GOTO SB0_OK;
                SB0_ifBusy
                        banksel T1CON
                        btfss T1CON, TMR1ON
                                GOTO NormalHandle
                        ; Обработка вывода полученного сообщения
                        btfss UART_t0, T1_Busy_on_display
                                GOTO EnableDisplayingMassage
                                GOTO StopDisplayingMassage
                NormalHandle
                        banksel STATUS
                        movf ButtonModes, w
                        btfsc STATUS, Z
                                GOTO SB0_OK
                                btfss UART_CON, TR_F
                                        GOTO Fix_addr ; Если Z = 1
                                        GOTO Send_to_TX
                        ; ВХОДЯЩЕЕ СООБЩЕНИЕ ОБРЫВАЕТ ВВОД ДАННЫХ


                EnableDisplayingMassage
                        banksel TRISIO
                        bcf TRISIO, GPIO5 ; Делаем Порт 5 "Выводом"
                        banksel PIE1
                        bsf PIE1, TMR1IE                ; РАЗРЕШАЕМ ПРЕРЫВАНИЯ И ВЫВОД СООБЩЕНИЙ
                        bsf UART_CON, SEG7_Display_F
                        movlw b'00000010'
                        movwf Lcd_data
                        CALL Disp_Info_7seg
                        return

                StopDisplayingMassage
                        banksel T1CON
                        bcf T1CON, TMR1ON                       ; ВЫКЛЮЧАЕМ ТАЙМЕР №1
                        banksel PIE1
                        bcf PIE1, TMR1IE                ; ЗАПРЕЩАЕМ ПРЕРЫВАНИЯ И ВЫВОД СООБЩЕНИЙ

                        movlw b'00000010'
                        movwf Lcd_data
                        CALL Disp_Info_7seg

                        bcf UART_CON, SEG7_Display_F
```

```
                banksel TRISIO
                bsf TRISIO, RX
                bcf TRISIO, TX
                banksel IOCB
                movf IOC_INIT, w
                movwf IOCB
                bcf UART_CON, BUSY_F
                CALL Init
                return

SB0_OK

        banksel STATUS
        movf ButtonModes, w
        btfss STATUS, Z
                GOTO Fix_addr ; Если Z = 0

                bsf ButtonModes, SB0_M0  ;  Если Z = 0
                bsf UART_CON, DATA_INC_F ; Разрешаем чтение данных с SB_INC
                movlw 0FFh
                movwf SEV_SEGM_reg
                bsf UART_CON, BUSY_F
                banksel TRISIO
                bcf TRISIO, RX
                bsf TRISIO, TX
                banksel IOCB
                bsf IOCB, TX
                bcf IOCB, RX

                movlw b'00000010'
                movwf Lcd_data
                CALL Disp_Info_7seg
                ;CALL SB1_INC_HNDL
                return ; TODO

Fix_addr ; 2 (01) - фиксируется данный адрес. Переходит к считыванию данных отправляемых
        btfsc ButtonModes, SB0_M1
                GOTO Fix_data_TX

                movf SEV_SEGM_reg, w
                movwf ADDR_reg
                movlw 0FFh
                movwf SEV_SEGM_reg
                bcf ButtonModes, SB0_M0  ;  Если Z = 0
                bsf ButtonModes, SB0_M1  ;  Если Z = 0
                CALL SB1_INC_HNDL
                return

Fix_data_TX  ; 3 (10) - фиксируются отправляемые данные И отправка данных и адреса
        btfss ButtonModes, SB0_M1
                return                                  ; WTF (TODO)
                btfsc ButtonModes, SB0_M0
                        GOTO Send_to_TX         ; мб не так надо?? (TODO)
                movf SEV_SEGM_reg, w
                movwf DATA_reg
                bsf ButtonModes, SB0_M0
                bsf UART_CON, TR_F
                ;banksel IOCB
                ;bcf IOCB, TX
                ;bcf IOCB, RX
                return

Send_to_TX
                ;bcf UART_CON, TX
                ;banksel IOCB
                ;bcf IOCB, TX
                ;bcf IOCB, RX
                ;bcf IOCB, GPIO0
                banksel INTCON
                bcf INTCON, GPIE
                ;bcf LastPressedBtns, SB_SEND_Ch        ; Чтобы не было лишних вызовов
```

```
                        banksel GPIO
                        bsf GPIO, TX
                        ;bsf GPIO, TX
                        banksel TRISIO
                        bsf TRISIO, RX
                        bcf TRISIO, TX
                        banksel WPU
                        bsf WPU, TX

                        CALL TRSF_HNDL

                        clrf UART_CON
                        bsf UART_CON, BanReceiveZ9Bit_F                                 ; TODO (ПРАВИЛЬНО
ЛИ???)
                        clrf ButtonModes
                        banksel WPU
                        bcf WPU, TX
                        banksel TRISIO
                        movlw TRISIO_INT
                        movwf TRISIO
                        banksel IOCB
                        bcf IOCB, TX
                        bsf IOCB, RX
                        bsf IOCB, GPIO0
                        banksel INTCON
                        bsf INTCON, GPIE
                        ;clrf SEV_SEGM_reg

                        return

Select_Display_Info
        banksel STATUS
        If_0_on7segm
                movlw 00h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_1_on7segm; Не совпало с "0"
                        movlw b'11111100'
                        movwf Lcd_data
                        return

        If_1_on7segm
                movlw 01h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_2_on7segm; Не совпало с "1"
                        movlw b'01100000'
                        movwf Lcd_data
                        return

        If_2_on7segm
                movlw 02h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_3_on7segm; Не совпало с "2"
                        movlw b'11011010'
                        movwf Lcd_data
                        return

        If_3_on7segm
                movlw 03h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_4_on7segm; Не совпало с "3"
                        movlw b'11110010'
                        movwf Lcd_data
                        return

        If_4_on7segm
                movlw 04h
```

```
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_5_on7segm; Не совпало с "4"
                        movlw b'01100110'
                        movwf Lcd_data
                        return

        If_5_on7segm
                movlw 05h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_6_on7segm; Не совпало с "5"
                        movlw b'10110110'
                        movwf Lcd_data
                        return

        If_6_on7segm
                movlw 06h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_7_on7segm; Не совпало с "6"
                        movlw b'10111110'
                        movwf Lcd_data
                        return

        If_7_on7segm
                movlw 07h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_8_on7segm; Не совпало с "7"
                        movlw b'11100000'
                        movwf Lcd_data
                        return

        If_8_on7segm
                movlw 08h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_9_on7segm; Не совпало с "8"
                        movlw b'11111110'
                        movwf Lcd_data
                        return

        If_9_on7segm
                movlw 09h
                xorwf SEV_SEGM_reg, w
                btfss STATUS, Z
                        GOTO If_ERR; Не совпало с "9"
                        movlw b'11110110'
                        movwf Lcd_data
                        return

        If_ERR
                movlw b'00000010'
                movwf Lcd_data
                movlw 0FFh
                movwf SEV_SEGM_reg
                return



Disp_Info_7seg
        banksel GPIO
        movlw .9
        movwf Loop_ident
        Loop_Disp_bits
                decfsz Loop_ident
                        GOTO Fill_LCD_bit
                        GOTO Shift_reg
                        Fill_LCD_bit
                                btfss Lcd_data, 00h
```

```
                                        GOTO Conf_0bit
                                                bsf GPIO, GPIO5
                                                GOTO Transfer_bit_to_7seg

                                        Conf_0bit
                                                bcf GPIO, GPIO5

                                        Transfer_bit_to_7seg
                                                bcf GPIO, GPIO4
                                                bsf GPIO, GPIO4
                                                bcf GPIO, GPIO4
                                                banksel STATUS
                                                bcf STATUS, C
                                                rrf Lcd_data, f
                                                GOTO Loop_Disp_bits

                        Shift_reg
                                bsf GPIO, GPIO2
                                bcf GPIO, GPIO2
                                return

SB1_INC_HNDL
        btfss UART_CON, BUSY_F
                return ; Если не занят - нах надо что-то инкрементировать? Делать нехрен??
                incf SEV_SEGM_reg
        InfoDispayAlg
                CALL Select_Display_Info
                CALL Disp_Info_7seg
                ; TODO НАДО менять какие-то ФЛАГИ 100%
                return









GPIO_INT
        ;btfss UART_CON, BUSY_F
        ;       retfie   ; TODO Мб ЭТО НЕПРАВИЛЬНО
;        Если UART работает, то проверим дальше что было нажато (по режиму UART)
        CALL FindPortDiff
        btfsc UART_CON, RD_F; Если установился флаг RD_F по прерыванию, то значит пришёл start-бит ПОСЫЛКИ - и
на всё остальное по барабану.
                CALL GET_MSG
        btfsc LastPressedBtns, SB_INC_Ch
                CALL SB1_INC_HNDL; Если кнопка INC нажата
        btfsc LastPressedBtns, SB_SEND_Ch
                CALL SB0_SEND_HNDL; Если кнопка SEND нажата
        clrf LastPressedBtns
        banksel GPIO
        movf GPIO, w
        movwf LastPortState
        banksel INTCON
        bcf INTCON, GPIF
        retfie  ; TODO Проверить достаточно ли этого!


T0_INT  ; Скорость в 9600 бит/с
        btfss UART_CON, BUSY_F ; Проверка, что UART работает сейчас (а не простаивает)
                retfie  ; CHECK IT (ТАК КАК МОГУТ БЫТЬ ДРУГИЕ ПРИЧИНЫ РАБОТЫ ТАЙМЕРА 0!!!!!!!!!!!!!!!!!!)

        ;Если T0_mode  == 1, то это целый бит
        ;Если T0_mode  == 0, то это пол бита
        btfss UART_t0, Simple_TMR
                GOTO Baud_Control
                bcf UART_t0, Simple_TMR
```

```
                bcf INTCON, T0IE
                bcf INTCON, T0IF
                return

        Baud_Control
                banksel INTCON
                btfsc UART_CON, T0_MODE
                        GOTO T0_UPD_Baud
                        GOTO T0_UPD_HalfBaud

                T0_UPD_Baud
                        bcf UART_t0, WAIT_FullBaud

                        bcf INTCON, T0IE
                        bcf INTCON, T0IF
                        return
                T0_UPD_HalfBaud
                        bcf UART_t0, WAIT_HalfBaud
                        bcf INTCON, T0IE
                        bcf INTCON, T0IF
                        return



T1_INT
        btfss UART_CON, SEG7_Display_F
                GOTO UpdateTMR1 ; Если флаг почему-то не стоит (TODO)

                banksel STATUS
                If_Disp_A
                        movlw .0
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO If_Disp_dash
                                movlw b'11101110'
                                movwf Lcd_data
                                CALL Disp_Info_7seg
                                incf GotMsgDispMode, f
                                GOTO UpdateTMR1

                If_Disp_dash
                        movlw .1
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO If_Disp_GotAddr
                        Disp_dash
                                movlw b'00000010'
                                movwf Lcd_data
                                CALL Disp_Info_7seg
                                incf GotMsgDispMode, f
                                GOTO UpdateTMR1

                If_Disp_GotAddr
                        movlw .2
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO If_Disp_D
                                movf ADDR_reg, w
                                movwf SEV_SEGM_reg
                                CALL InfoDispayAlg
                                incf GotMsgDispMode, f
                                GOTO UpdateTMR1

                If_Disp_D
                        movlw .3
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO If_DispSecondDash
                                movlw b'01111010'
                                movwf Lcd_data
                                CALL Disp_Info_7seg
```

```
                        incf GotMsgDispMode, f
                        GOTO UpdateTMR1

                If_DispSecondDash
                        movlw .4
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO If_Disp_GotData
                                GOTO Disp_dash

                If_Disp_GotData
                        movlw .5
                        xorwf GotMsgDispMode, w
                        btfss STATUS, Z
                                GOTO SmthNotOKWithTMR1

                                movf DATA_reg, w
                                movwf SEV_SEGM_reg
                                CALL InfoDispayAlg

                                clrf GotMsgDispMode
                                bsf UART_t0, T1_Busy_on_display
                                banksel IOCB
                                clrf IOCB
                                bsf IOCB, GPIO0
                                banksel INTCON
                                bsf INTCON, GPIE
                                GOTO UpdateTMR1

        SmthNotOKWithTMR1
                NOP             ; Поставь точку остановы
        UpdateTMR1
                banksel TMR1L
                movlw 7Fh
                movwf TMR1L
                movwf TMR1H
                banksel PIR1
                bcf PIR1, TMR1IF
                retfie

TRSF_HNDL
        ; Вызывается, если режим отправки сообщений, UART свободен и была нажата кнопка SB0_SEND, для
инициализации отправки пакета
        ; Сначала выбирается адрес отправителя. Для этого вызывается функция постоянного обновления
        ;banksel TRISIO
        ;bcf TRISIO, TX


        movf ADDR_reg, w
        iorlw b'00000000'
        movwf SERBUF
        bsf UART_CON, BIT9
        CALL SEND_MSG   ; Далее отправляется данный бит в TX
        ; Далее снова режим выбора данных отправляемых
        ; По нажатию SB_SEND фиксируются данные

        CLRF TMR0
        CALL StartSimpleT0
        ; WAIT мб нужен!!!!!!!  TODO

        movf DATA_reg, w
        movwf SERBUF
        bcf UART_CON, BIT9
        CALL SEND_MSG
        movlw b'00001010'
        movwf Lcd_data
        CALL Disp_Info_7seg
        return
        ; Отправка данных в TX
        ; Сброс режимов работы в изначальное состояние. Ждём или нажатия кнопки SB_SEND, или прихода пакета.
```

```
StartSimpleT0
        banksel INTCON
        bsf INTCON, T0IE
        banksel INTCON
        bcf INTCON, GPIF
        bcf INTCON, T0IF
        bcf INTCON, PEIE
        bsf INTCON, GIE ; Разрешаем прерывания (Таймер)
        bsf UART_t0, Simple_TMR
        WAIT_SimpleTMR
                btfsc UART_t0, Simple_TMR
                        GOTO WAIT_SimpleTMR
                        bcf INTCON, GIE ; Разрешаем прерывания (Таймер)
                        bsf INTCON, PEIE
                        return

HalfBaud
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        return

OneBaud
        movlw .201
        banksel TMR0
        movwf TMR0
        banksel INTCON
        bsf INTCON, T0IE
        bsf UART_CON, T0_MODE
        bsf UART_t0, WAIT_FullBaud
        banksel INTCON
        bcf INTCON, GPIF
        bcf INTCON, T0IF
        bcf INTCON, PEIE
        bsf INTCON, GIE ; Разрешаем прерывания (Таймер)
        WAIT_forFullBaud
                btfsc UART_t0, WAIT_FullBaud
                        GOTO WAIT_forFullBaud
                        bcf INTCON, GIE ; Разрешаем прерывания (Таймер)
                        bsf INTCON, PEIE
                        return

Start_Displaying_GotMessage
        ; Если получили сообщение (сначала адрес), потом данные
        ; Запускается таймер Т1. Каждую примерно секунду будет отображаться то Адрес, от кого, то данные
        ; A - # d - #
        ; Выключаем прерывания по TX и RX

        banksel STATUS
        movf GotMsgDispMode, w
        btfss STATUS, Z
                GOTO OtherModes; Если НЕ режим 0

        ; Если режим 0
```

```
        banksel TMR1L
        movlw 7Fh
        movwf TMR1L
        movwf TMR1H

        banksel PIE1
        bcf PIE1, TMR1IE                ; СПЕЦИАЛЬНО ВЫКЛЮЧАЕМ ДО ТЕХ ПОР, ПОКА НЕ БУДЕТ НАЖАТА КНОПКА SEND
ПОЛЬЗОВАТЕЛЕМ!!!

        banksel TRISIO
        bsf TRISIO, GPIO5 ; Делаем Порт 5 "Вводом",пока не будет сделано переключение
        bsf TRISIO, TX
        bsf TRISIO, GPIO0
        banksel IOCB
        bcf IOCB, GPIO5 ;(запрещаем прерывания)
        bsf IOCB, GPIO0

        banksel T1CON
        bsf T1CON, T1CKPS1
        bsf T1CON, T1CKPS0

        bcf UART_CON, RD_F

        banksel INTCON
        bsf INTCON, PEIE
        bsf INTCON, GPIE
        bsf T1CON, TMR1ON               ; Старт таймера
        return

        OtherModes
                ; WHAT THE FUCK? Я сюда не должен был попадать!!
                return ; TODO Это что-то явно неправильное!



CheckIfForMe
        movf SERBUF, w
        movwf Accum
        movlw b'00001111'
        andwf Accum, f
        movlw b'00000000'               ; Проверяем, что пришло 0-му и это правильно!
        xorwf Accum, f
        movf Accum, f
        btfsc STATUS, Z
                return
        movf SERBUF, w
        movwf Accum
        movlw b'00001111'
        andwf Accum, f
        movlw b'11111100'
        addwf Accum, w
        banksel STATUS
        btfsc STATUS, C
                GOTO Wide
        movlw 0FFh
        movwf Accum
        return

        Wide
                clrf Accum
                return


GET_MSG
        banksel IOCB
        ;clrf IOCB
        banksel INTCON
        bcf INTCON, GPIE
        bcf INTCON, GPIF
        movlw .8
```

```
movwf TEMP
clrf SERBUF
CALL HalfBaud
banksel GPIO
Recieve_data
        CALL OneBaud
        bcf STATUS, C
        rrf SERBUF , f
        btfsc GPIO, RX
        bsf SERBUF, 7
        decfsz TEMP, f
        GOTO Recieve_data
CALL OneBaud
; Читаем 9 бит
btfsc GPIO, RX
        GOTO GetAddress; Если пришёл 9 бит == 1 (адрес)
        ; Если пришёл 9 бит == 0 (данные) - то ХЗ. Надо проверить!
        ; (9bit == 0)
        btfsc UART_CON, BanReceiveZ9Bit_F
                GOTO  NotForMe; ПРОПУСК ДАННОЙ ПОСЫЛКИ! (ожидали адрес - получили данные)

                CALL OneBaud
                banksel GPIO
                btfss GPIO, RX
                        GOTO NotForMe    ; Если успешно дошёл Последний бит
                ; ЕСЛИ ПОЛУЧИЛИ ДАННЫЕ - И ЭТО ОК
                movf SERBUF, w
                movwf DATA_reg
                        CALL Start_Displaying_GotMessage
                        return

GetAddress ; (9bit == 1)
        btfss UART_CON, BanReceiveZ9Bit_F
                return ; ПРОПУСК ДАННОЙ ПОСЫЛКИ! (ожидали данные - получили адрес)

                CALL OneBaud
                banksel GPIO
                btfss GPIO, RX
                        GOTO NotForMe    ; Если успешно дошёл Последний бит
                ; Адрес
                CALL CheckIfForMe
                banksel STATUS
                movf Accum, f
                btfss STATUS, Z
                        GOTO NotForMe;Если посылка НЕ нам
                movlw b'11110000'
                andwf SERBUF, f
                banksel STATUS
                bcf STATUS, C
                rrf SERBUF
                rrf SERBUF
                rrf SERBUF
                rrf SERBUF
                movf SERBUF, w                  ; Не знаю зачем
                movwf ADDR_reg
                bcf UART_CON, BanReceiveZ9Bit_F
                banksel IOCB
                bsf IOCB, RX
                bsf IOCB, GPIO0
                banksel INTCON
                bsf INTCON, GPIE
                bcf INTCON, GPIF
                return

        NotForMe
                bcf UART_CON, RD_F
                banksel IOCB
                bsf IOCB, RX
                bsf IOCB, GPIO0
                banksel INTCON
                bsf INTCON, GPIE
```

```
                        bcf INTCON, GPIF
                        return                  return

        NotOKAll
                banksel IOCB
                movf IOC_INIT, w
                movwf IOCB
                banksel INTCON
                bsf INTCON, GPIE
                bcf INTCON, GPIF
                return


SEND_MSG
        ; TODO Загрузка отправляемой константы  И загрузка 9 битого
        banksel GPIO
        movlw .8
        movwf Loop_ident
        bcf GPIO, TX
         CALL OneBaud
        Send_data
                banksel STATUS
                bcf STATUS, C
                rrf SERBUF, f
                banksel STATUS
                btfss STATUS, C
                GOTO Transfer_0
                GOTO Transfer_1
                Transfer_0
                        banksel GPIO
                        bcf GPIO, TX
                        GOTO BIT_ready
                Transfer_1
                        banksel GPIO
                        bsf GPIO, TX
                BIT_ready
                        CALL OneBaud
                        decfsz Loop_ident, f
                        GOTO Send_data
                        btfsc UART_CON, BIT9
                                GOTO Send_9one
                                GOTO Send_9zero
                        Send_9one
                                bsf GPIO, TX
                                GOTO Send_Stop
                        Send_9zero
                                bcf GPIO, TX
                        Send_Stop
                                CALL OneBaud
                                banksel STATUS
                                bcf STATUS, C
                                rrf  SERBUF, f          ; TODO ЗАЧЕМ?????
                                bsf GPIO, TX            ; СТОП-БИТ
                                retlw 0



END
```