|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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.

1. **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 | бит/с |  |

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

1. **Теоретическая часть**

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.

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).

1. **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.

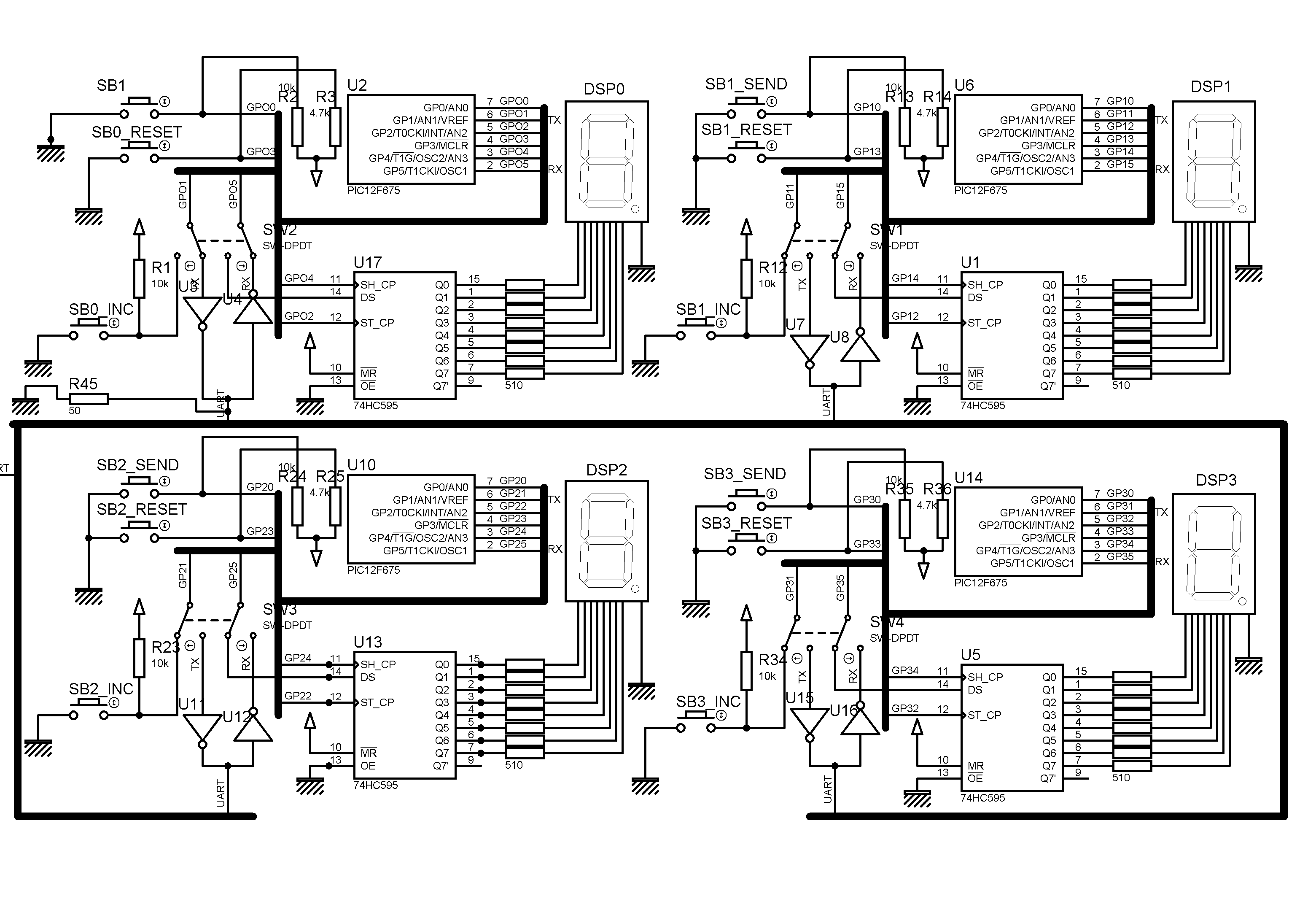


Рисунок 1 – Схема лабораторной работы

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

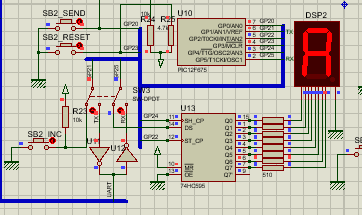


Рисунок 2 – Индикация входящего сообщения (часть 1 – «адрес»)

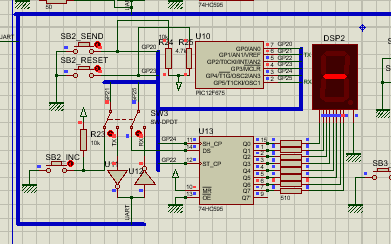


Рисунок 3 – Индикация входящего сообщения (часть 2)

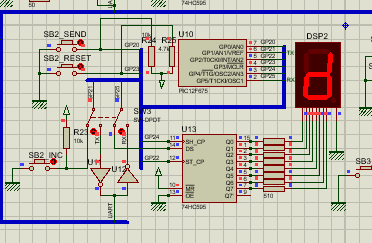


Рисунок 4 – Индикация входящего сообщения (часть 3 – «данные»)

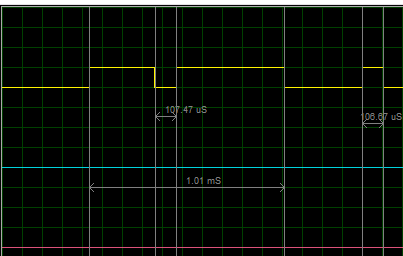


Рисунок 5 – Осциллограмма посылки

**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 T1. Каждую примерно секунду будет отображаться то Адрес, от кого, то данные

; А - # 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