Лабораторная работа № 2

Контроллеры ARM: USART

Цель работы: изучение принципиального устройства и основ программирования USART у микроконтроллеров с архитектурой ARM7.

1. Устройство USART

USART — (последовательный) универсальный синхронно-асинхронный приемо-передатчик *(universal synchronous-asynchronous receiver-transmitter - USART)*. Передача данных в USART осуществляется через равные промежутки времени. Этот временной промежуток определяется заданной скоростью USART и указывается в бодах (для символов, которые могут принимать значения равные только нулю или единице, бод эквивалентен битам в секунду). Существует общепринятый ряд стандартных скоростей: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бод.

Помимо битов данных USART автоматически вставляет в поток синхронизирующие метки, так называемые **стартовый** и **стоповый биты**. При приёме эти лишние биты удаляются. Обычно стартовый и стоповый биты отделяют один байт информации (8 бит). Однако встречаются реализации USART, которые позволяют передавать по 5, 6, 7, 8 или 9 бит. Биты, отделённые стартовым и стоповым сигналами, являются минимальной посылкой. USART позволяет вставлять два стоповых бита при передаче для уменьшения вероятности рассинхронизации приёмника и передатчика при плотном трафике. Приёмник игнорирует второй стоповый бит, воспринимая его как короткую паузу на линии.

Принято соглашение, что **пассивным** (в отсутствие данных) состоянием входа и выхода USART является логическая «1» (рис. 1). Стартовый бит всегда логический «0». Поэтому приёмник USART ждёт перепада из «1» в «0» и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Если в этот момент на входе всё ещё «0», то запускается процесс приёма минимальной посылки. Для этого приёмник отсчитывает 9 битовых длительностей подряд (для 8-бит данных) и в каждый момент фиксирует состояние входа. Первые 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда «1», если реально принятое значение иное, USART фиксирует ошибку.

Рис. 1. Диаграмма обработки данных USART

Поскольку синхронизирующие биты занимают часть битового потока, то результирующая пропускная способность USART не равна скорости соединения. Например, для 8-битных посылок формата 8-N-1 синхронизирующие биты зани-

мают 20% потока, что для физической скорости 115 200 бод даёт битовую скорость данных 92160 бит/с или 11 520 байт/с.

Контроль чётности. В протоколе USART можно автоматически контролировать целостность данных методом контроля битовой чётности. Когда эта функция включена, последний бит данных («бит чётности») всегда принимает значение 1 или 0, так чтобы количество единиц в байте всегда было четным.

Управление потоком. На начальном этапе развития цифровой техники устройства с USART могли быть настолько медлительными, что не успевали обрабатывать поток принимаемых данных. Для решения этой проблемы модули USART снабжались отдельными выходами и входами управления потоком. При заполнении входного буфера логика принимающего USART выставляла на соответствующем выходе запрещающий уровень, и передающий USART приостанавливал передачу. Позже управление потоком возложили на коммуникационные протоколы, и надобность в отдельных линиях управления потоком постепенно исчезла.

Область применения. USART обычно применяется для проводной связи между двумя устройствами, расстояние между которыми не превышает единиц метров. Увеличение дальности связи приводит к зашумлению сигнала и уменьшению вероятности его правильного приёма.

USART в микроконтроллере STM32F072RBT

Микроконтроллер STM32F072RBT включает четыре модуля USART. Вход приемника и выход передатчика модулей USART1, USART2, USART3 и USART4 можно подключать к выводам разных GPIO с помощью системы альтернативных функций. На верхней частоте тактирования 48 МГц скорость обмена данными с помощью USART может достигать 6 Мбит/с. Приемник и передатчик USART могут работать независимо друг от друга и обеспечивать одновременный прием и передачу данных. Пакет передаваемых или принимаемых данных может составлять: 7, 8 или 9 бит. Количество стоп-битов в пакетах: 1 или 2. Для скоростной передачи данных каждый модуль USART может задействовать контроллер прямого доступа в память (DMA), а для обработки потока данных использовать систему управления событиями и прерываниями. Приемник и передатчик USART могут работать в синхронном режиме, использовать аппаратную поддержку управления потоком данных и выполнять функцию «пробуждения» контроллера по линиям связи.

Функциональная схема USART представлена на рисунке 2. Рассмотрим принципиальные основы работы передатчика (Tx) и приемника (Rx), которые на схеме выделены серым цветом.

Передатчик включает два регистра: регистр сдвига (Transmit Shift Register) и буферный регистр (Transmit Data Register - TDR). Программный доступ организован только к TDR. Данные, требующие передачи, следует загрузит в TDR. Если передача предыдущего битового пакета закончена и регистр сдвига пуст, то пакет данных из TDR автоматически копируются в сдвиговый регистр. Затем под действием тактовых импульсов данные сдвигаются и побитно поступают на выход TX. После копирования данных в сдвиговый регистр есть время на запись новых дан-

ных в буферный регистр, равное времени передачи. Если успевать заполнять буферный регистр, то данные будут передаваться сплошным потоком без пауз между битовыми пакетами. Так организована буферизация данных передатчика.

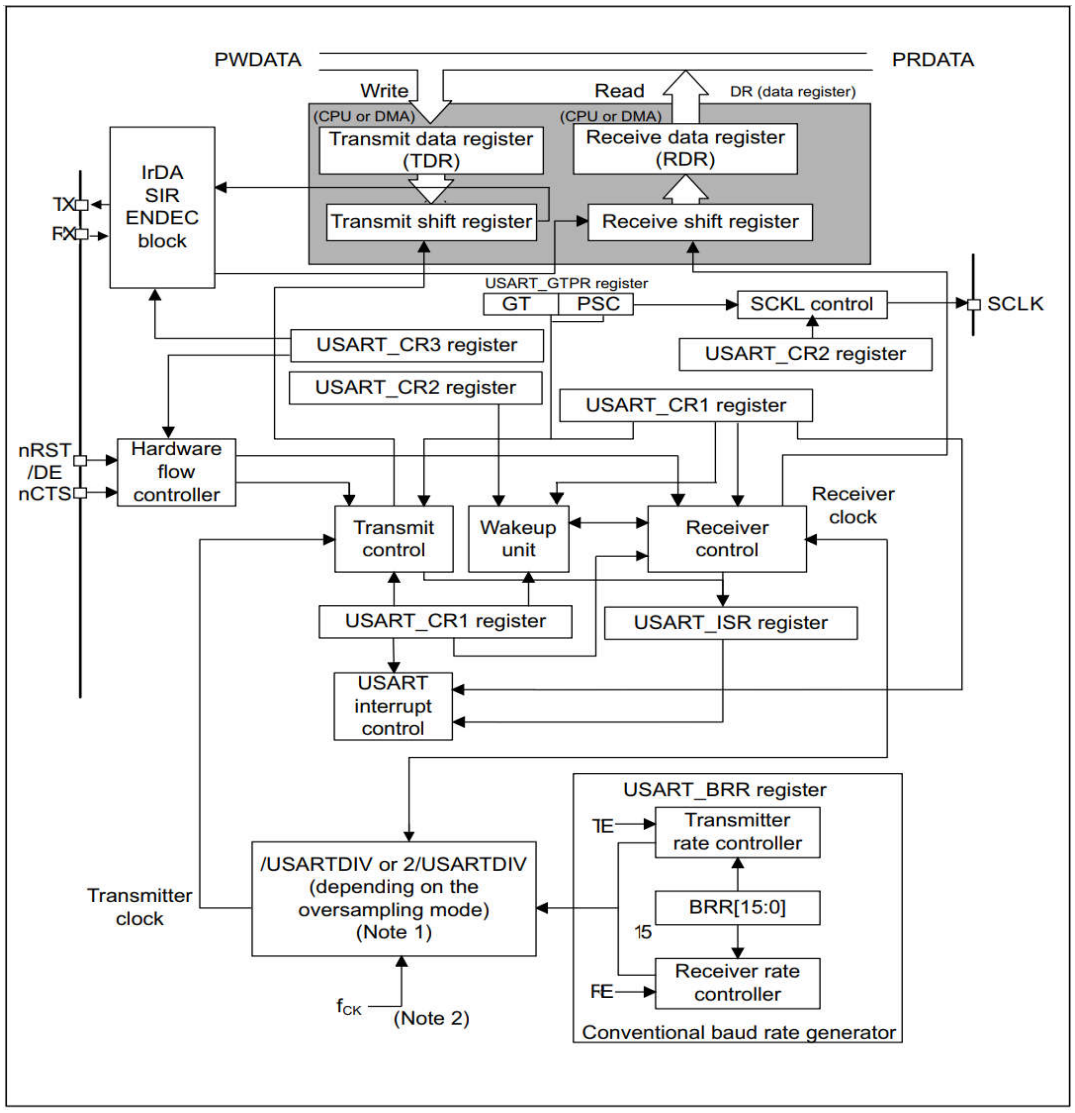

Рис. 2. Функциональная схема USART

Об окончании передачи данных регистром сдвига и освобождении буферного регистра сообщают специальные флаги – **TC** и **TXE** соответственно. По ним могут генерироваться прерывания.

Приемная часть устройства также состоит из двух регистров: регистра сдвига (Receive Shift Register) и буферного регистра (Receive Data Register - RDR). С входа Rx биты поступают на сдвиговый регистр, сдвигаются под действием тактовых импульсов и, после формирования полного битового пакета, автоматически копируются в буферный регистр. Буферный регистр имеет программный доступ. О поступлении в него новых данных сообщает специальный флаг - **RXNE**, по которому может генерироваться прерывание. Если данные поступают сплошным потоком, без пауз, то полученный пакет должен быть считан из буферного регистра за время меньшее, чем время приема следующего пакета. Иначе, следующий пакет при автоматическом копировании из сдвигового регистра в буферный уничтожит ранее принятые данные.

Регистры для конфигурирования USART

Ниже приведено описание регистров, полей и флагов, которые не были рассмотрены ранее, но являются значимыми для решения задач настоящей лабораторной работы. Подробное описание всех регистров, обеспечивающих конфигурирование USART приведено в «Reference manual STM32F0x1/STM32F0x2/ STM32F0x8».

Регистр RCC_APB2ENR (программный доступ: **RCC-> APB2ENR**) используется для разрешения тактирования модуля USART1 (рис. 3).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGM CUEN	Res.	Res.	Res.	TIM17 EN	TIM16 EN	TIM15 EN
									rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI1 EN	TIM1 EN	Res.	ADC EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFG COMP EN
	rw		rw	rw		rw									rw

Рис. 3. Структура регистра RCC_APB2ENR

В регистре RCC_APB2ENR за разрешение тактирования модуля USART1 отвечает 14-й бит. Его установка выполняется командой:

RCC->APB2ENR |= RCC_AHBENR_GPIOAEN;

где константа RCC_AHBENR_GPIOAEN = 0x0000 4000h. Сброс этого бита можно выполнить командой:

RCC->APB2ENR &=~ RCC_AHBENR_GPIOAEN;

После перезагрузки контроллера RCC_APB2ENR=0x0000 0000h.

Регистр GPIOx_AFRH (программный доступ: **GPIOA-> AFR[1]**) (**x** – буква порта, изменяется от A до F) используется для выбора вида альтернативной функции на выводах 8 – 15 порта A (рис. 4).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 4. Структура регистра GPIOx_AFRH

Для кодирования вида альтернативных функций используется 4 бита (рис. 5).

0000: AF0	1000: Reserved
0001: AF1	1001: Reserved
0010: AF2	1010: Reserved
0011: AF3	1011: Reserved
0100: AF4	1100: Reserved
0101: AF5	1101: Reserved
0110: AF6	1110: Reserved
0111: AF7	1111: Reserved

Рис. 5. Коды альтернативных функций

По умолчанию GPIOx_AFRH = 0x0000 0000h.

Регистр USARTx_CR1 (программный доступ: **USART1->CR1**) (**x** – буква порта, изменяется от A до F) используется для разрешения работы модуля USART, включения приемника и передатчика, настройки длины битового пакета, разрешения прерываний по флагам и др. (рис. 6).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 6. Структура регистра USARTx_CR1

Бит **UE** разрешает работу модуля USART.

Внимание! *Перед настройкой USART (изменением данных в регистрах USARTx_CR1, USARTx_CR2, USARTx_CR3, USARTx_BRR, USARTx_GTPR) бит UE нужно сбросить.*

Бит **TE** разрешает работу передатчика.

Бит **RE** разрешает работу приемника.

Биты **M1** и **M0** вместе определяют длину битового пакета, которым будут оперировать передатчик и приемник USART. Используется следующий битовый код (M1:M0):

- 00 – 1 старт-бит, 8 бит данных, число стоп-бит задается в USARTx_CR2;
- 01 – 1 старт-бит, 9 бит данных, число стоп-бит задается в USARTx_CR2;
- 10 – 1 старт-бит, 7 бит данных, число стоп-бит задается в USARTx_CR2;

Бит **PCE** разрешает аппаратный контроль четности в принимаемых и передаваемых битовых пакетах. При PCE=1 перед стоп-битом в пакете будет вставляться или анализироваться дополнительный бит – бит контроля четности, значение которого зависит от селектора четности (бит **PS**).

Бит **PS** выбирает тип контроля четности в соответствие со следующими кодами:

- 0 – четный паритет, т.е. значение бита контроля четности (0 или 1) будет выбираться так, чтобы вместе с ним количество единичных битов в пакете было четным;
- 1 - нечетный паритет, т.е. значение бита контроля четности (0 или 1) будет выбираться так, чтобы вместе с ним количество единичных битов в пакете было нечетным;

Значение регистра Регистр USARTx_CR1 по умолчанию: 0x0000 0000h.

Регистр USARTx_CR2 (программный доступ: **USART1->CR2**) (**x** – буква порта, изменяется от A до F) используется, чтобы задать число стоп-бит в пакетах, включить режим автоопределения скорости обмена данными и настроить его параметры, поменять местами выводы приемника и передатчика и др. (рис. 7).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

Рис. 7. Структура регистра USARTx_CR2

Биты **STOP[1:0]** (13 и 12 биты регистра) кодирую число стоп-бит, которые завершают передаваемые и принимаемые данные:

- 00 – 1 стоп-бит;
- 01 – зарезервировано (не используется);
- 10 – 2 стоп-бита;
- 11 – 1,5 стоп-бита.

Значение регистра Регистр USARTx_CR2 по умолчанию: 0x0000 0000h.

Регистр USARTx_CR3 (программный доступ: **USART1->CR3**) (**x** – буква порта, изменяется от A до F) используется для разрешения сигналов аппаратного управления потоком данных, разрешения работы DMA при передаче и приеме данных и др. (рис. 8).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS[2:0]		SCARCNT2:0]			Res.
									rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 8. Структура регистра USARTx_CR3

Бит **DMAT** разрешает работу DMA с передатчиком USART.

Бит **DMAR** разрешает работу DMA с приемником USART.

Значение регистра Регистр USARTx_CR3 по умолчанию: 0x0000 0000h.

Регистр USARTx_BRR (программный доступ: **USART1->BRR**) (**x** – буква порта, изменяется от A до F) задает коэффициент деления тактовой частоты и определяет скорость приема и передачи битовых пакетов (рис. 9).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 9. Структура регистра USARTx_BRR

USARTDIV[0:15] – 16-ти разрядный коэффициент деления частоты тактового генератора. Биты 16 – 31 не используются и должны быть равны нулю. Для битов 4

– 15: BRR[4:15] = USARTDIV[4:15]. Для битов 0 – 3 действует следующее правило:

- если бит OVER8=0 (регистр USARTx_CR1), то BRR[3:0] = USARTDIV[3:0];
- если бит OVER8=1, то BRR[2:0]=USARTDIV[3:0] (сдвиг на 1 бит вправо). При этом BRR[3] должен быть сброшен.

Коэффициент деления тактовой частоты рассчитывается по формуле

$$USARTDIV = round(F_{clk}/BAUD),$$

где F_{clk} – частота тактирования микроконтроллера, BAUD – скорость обмена битами по USART, бит/с, round() – оператор округления.

Значение регистра Регистр USARTx_BRR по умолчанию: 0x0000 0000h.

Регистр USARTx_ISR (программный доступ: **USART1->ISR**) (**x** – буква порта, изменяется от A до F) содержит набор флагов, отражающих текущее состояние работы USART (статусный регистр USART) (рис. 10).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Рис. 10. Структура регистра USARTx_ISR

Флаг **TXE** автоматически устанавливается в единицу, когда завершается копирования битового пакета из TDR в сдвиговый регистр передатчика, и в буферный регистр передатчика можно помещать новые данные. Этот флаг показывает, что данные поступили в передатчик для отправки в линию и в буферном регистре можно размещать новые данные. Сброс флага TXE происходит при записи в регистр TDR нового битового пакета.

Флаг **TC** автоматически устанавливается в единицу, когда завершается передача битового пакета и на выходе Tx устанавливается логическая «1». Сброс флага происходит при копировании данных из регистра TDR в сдвиговый регистр. Этот флаг, как правило, используется когда нужно выключать модуль USART после завершения передачи данных.

Флаг **RXNE** автоматически устанавливается в единицу, когда завершается копирование данных из сдвигового регистра приемника в буферный регистр RDR. Этот флаг показывает, что полученные приемником из линии битовые данные перемещены в буферный регистр RDR и их можно считать в программу. Сброс флага происходит при последовательном чтении регистров ISR и RDR.

Значение регистра Регистр USARTx_ISR по умолчанию: 0x0000 00C0h.

Регистр USARTx_RDR (программный доступ: **USART1->RDR**) (**x** – буква порта, изменяется от A до F) является буфером приемника. В нем размещается принятый битовый пакет. Данные из RDR должны быть считаны за время приема

следующего пакета. Если этого не происходит, то следующий битовый пакет уничтожит предыдущий при копировании из сдвигового регистра приемника. Количество значащих бит в регистре RDR определяется длиной данных, заданной с помощью регистра CR1 (рис. 11).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Рис. 11. Структура регистра USARTx_RDR

Значение регистра Регистр USARTx_ISR по умолчанию: не определено.

Регистр USARTx_TDR (программный доступ: **USART1->TDR**) (**x** – буква порта, изменяется от A до F) является буфером передатчика. В нем размещается битовый пакет, который требуется передать через интерфейс. Данные в TDR из программы можно копировать только когда установлен флаг TXE. В противном случае возможно наложения битовых пакетов и их уничтожение.

Структура интерфейса «STM_01 - ПК» на основе USART и его использование

Учебный лабораторный комплекс STM_01 для взаимодействия с ПК по USART использует посредника – аппаратный драйвер CH340N. Этот драйвер отображает USART1 микроконтроллера STM32F072RBT6 на виртуальный COM-порт ПК через интерфейс USB. Микросхема CH340N в лабораторном комплексе STM_01 подключена к выводам микроконтроллера PA9 (TX_1 – выход передатчика) и PA10 (RX_1 – вход приемника) (рис. 12).

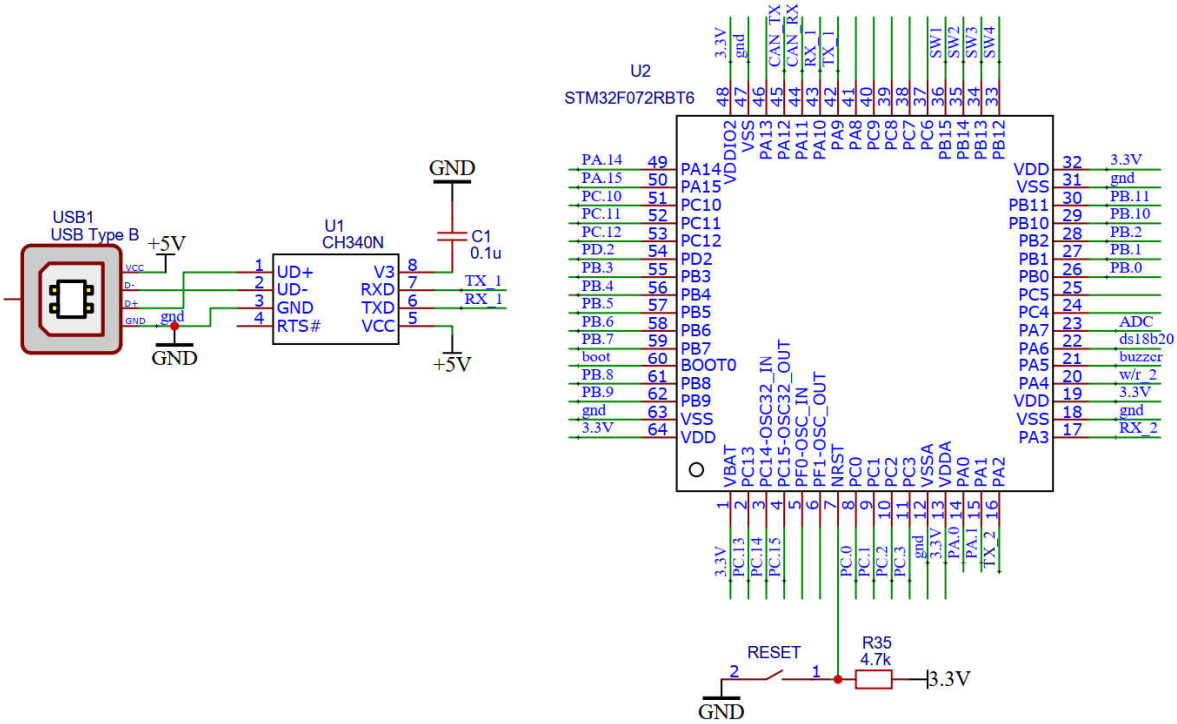

Рис. 12. Интерфейс на основе USART1 стенда STM_01

Алгоритм инициализации USART1 на микроконтроллере STM32F072RBT включает следующие действия:

1. Включить тактирование модуля USART1 в микроконтроллере (регистр **RCC_APB2ENR**);
2. Включить тактирование порта A (регистр **RCC_AHBENR**);
3. Перевести линии PA9 и PA10 в режим альтернативной функции (регистр **GPIOA_MODER**);
4. Указать альтернативную функцию **AF1** (регистр **GPIOA_AFRH)** (рис. 13).

Pin name	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
PA0	-	USART2_CTS	TIM2_CH1_ETR	TSC_G1_IO1	USART4_TX	-	-	COMP1_OUT
PA1	EVENTOUT	USART2_RTS	TIM2_CH2	TSC_G1_IO2	USART4_RX	TIM15_CH1N	-	-
PA2	TIM15_CH1	USART2_TX	TIM2_CH3	TSC_G1_IO3	-	-	-	COMP2_OUT
PA3	TIM15_CH2	USART2_RX	TIM2_CH4	TSC_G1_IO4	-	-	-	-
PA4	SPI1_NSS, I2S1_WS	USART2_CK	-	TSC_G2_IO1	TIM14_CH1	-	-	-
PA5	SPI1_SCK, I2S1_CK	CEC	TIM2_CH1_ETR	TSC_G2_IO2	-	-	-	-
PA6	SPI1_MISO, I2S1_MCK	TIM3_CH1	TIM1_BKIN	TSC_G2_IO3	USART3_CTS	TIM16_CH1	EVENTOUT	COMP1_OUT
PA7	SPI1_MOSI, I2S1_SD	TIM3_CH2	TIM1_CH1N	TSC_G2_IO4	TIM14_CH1	TIM17_CH1	EVENTOUT	COMP2_OUT
PA8	MCO	USART1_CK	TIM1_CH1	EVENTOUT	CRS_SYNC	-	-	-
PA9	TIM15_BKIN	USART1_TX	TIM1_CH2	TSC_G4_IO1	-	-	-	-
PA10	TIM17_BKIN	USART1_RX	TIM1_CH3	TSC_G4_IO2	-	-	-	-
PA11	EVENTOUT	USART1_CTS	TIM1_CH4	TSC_G4_IO3	CAN_RX	-	-	COMP1_OUT
PA12	EVENTOUT	USART1_RTS	TIM1_ETR	TSC_G4_IO4	CAN_TX	-	-	COMP2_OUT
PA13	SWDIO	IR_OUT	USB_NOE	-	-	-	-	-
PA14	SWCLK	USART2_TX	-	-	-	-	-	-
PA15	SPI1_NSS, I2S1_WS	USART2_RX	TIM2_CH1_ETR	EVENTOUT	USART4_RTS	-	-	-

Рис. 13. Альтернативные функции порта A

5. Настроить режим работы линии передатчика (PA9) следующим образом:
   - включить режим активного выхода (push-pull) (регистр **GPIOA_OTYPER**);
   - отключить все подтягивающие резисторы (регистр **GPIOA_PUPDR**);
   - включить высокую скорость синхронизации (регистр **GPIOA_OSPEEDR**);
6. Включить на линии приемника (PA10) подтягивающий резистор pull-up (регистр **GPIOA_PUPDR**);
7. Сбросить бит **UE** регистра **USART1_CR1** (запретить работу USART1).
8. С помощью регистра **USART1_CR1** разрешить работу передатчика (если нужно) и приемника (если нужно), определить количество бит данных в пакете (обязательно), задать контроль четности (если нужно), разрешить прерывания по флагам (если нужно).
9. С помощью регистра **USART1_CR2** задать число стоп-бит (обязательно).
10. С помощью регистра **USART1_CR3** разрешить работу DMA с приемником (если нужно) и с передатчиком (если нужно).
11. По завершении конфигурирования USART1 установить бит **UE** регистра **USART1_CR1** (разрешить работу USART1).

При приеме и передаче данных с помощью USART следует придерживаться следующих правил:

- Перед отправкой данные в USART следует дождаться, когда флаг TXE установиться в «1». После этого следует скопировать данные из программной переменной в регистр TDR;
- Перед считыванием данных из USART следует дождаться момента, когда флаг RXNE установиться в «1». После этого следует скопировать данные из регистра RDR в программную переменную.
- Перед отключением тактирования модуля USART следует дождаться момента, когда флаг TC установиться в «1».

2. Задание

1. Создайте папку «LabHard(<фамилия на латинице>-<номер группы>)-2». Например, студент Иванов из группы 1191б должен создать папку: LabHard(Ivanov-1191)-2.
2. Внутри папки «LabHard…» создайте папки: «Task2_1», «Task2_2» и «Task2_3». В них далее следует размещать проекты среды Keil µVision, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

2.1. Обучающая часть

Задача «USART – Эхо»: разработать программу для микроконтроллера лабораторного комплекса STM_01, которая на основе интерфейса USART1 будет принимать символы от ПК и ретранслировать их обратно.

1. В Keil µVision создайте новый проект и сохраните его в папке Task2_1. Название проекта задайте по следующему шаблону: «<инициалы ФИО на латинице>2_1». Например, проект Сидорова Ивана Петровича должен иметь название: **sip2_1**.
2. Добавьте в проект заголовочный файл **main.h** и файл программного кода **main.c**.
3. В заголовочный файл добавьте код:

```
#include "stm32f0xx.h"        //Подключение библиотеки с моделью stm32f0xx
void InitUSART1(void);        //Декларация функции инициализации USART1
```

4. В начале файла **main.c** подключите заголовочный файл **main.h** и опишите основную функцию:

```
int main(void)
{
}
```

5. Запрограммируем в функции main() первичную настройку модуля USART1, и реализуем алгоритм приема символа от ПК с его возвращением. Для инициализации USART используем команду:

```
InitUSART1();
```

Она вызовет продекларированную нами функцию для инициализации USART1. Реализацию этой функции сделаем после программного кода функции main(). Далее в тело функции main() добавьте строки:

```
while ((USART1->ISR & USART_ISR_RXNE) == 0) { }
```

Этот цикл блокирует исполнение дальнейших команд, пока флаг RXNE не будет установлен в «1». Как только приемник USART получил битовый пакет от ПК, разместил его в регистре RDR и установил флаг RXNE, то следует незамедлительно считать данные из RDR в программную переменную. Делается это командой

```
uint8_t d = (uint8_t)USART1->RDR;
```

которая размещается сразу за циклом while. Принятый от ПК символ нужно отправить обратно с помощью передатчика USART1. Первым действием по отправке символа является проверка готовности передатчика. В программе ее можно реализовать такой командой

```
while ((USART1->ISR & USART_ISR_TXE) == 0) { }
```

В цикле while осуществляется чтение статусного регистра ISR и анализ его бита TXE, который является одним из флагов передатчика. Пока флаг TXE==0, передатчик не готов принять новые данные в буферный регистр TDR и программа должна ждать. Это делает цикл while, который блокирует выполнение следующей за ним команды до установки TXE в «1». По готовности передатчика выполняется команда

```
USART1->TDR = d;
```

которая копирует символ из программной переменной в буферный регистр TDR и тем самым запускает процесс его отправки на ПК. Для того, чтобы прием символов от ПК и их отправка обратно происходили непрерывно нужно сделать последние 4 команды телом цикла

```
while(true) {
… //Команды приема и отправки символа
}
```

Полный код реализации функции main() приведен на рисунке 14.

6. Реализуем настройку USART1 в функции InitUSART1(). В соответствие с алгоритмом инициализации USART1, описанном в п.1, сначала требуется включить тактирование модуля USART1 в микроконтроллере STM32F072. Делается это командой

```
RCC->APB2ENR |= RCC_APB2ENR_USART1EN;
```

В ней используется константа, определенная в заголовочном файле stm32f0xx.h и содержащая значение 0x0000 4000h. С помощью этого значения устанавливается бит **USART1EN** регистра RCC_APB2ENR, отвечающий за разрешение тактирования USART1.

7. Приемник и передатчик USART1 использует выводы порта A. Поэтому следует активировать его, разрешив тактирование в микроконтроллере. В функции InitUSART1() для этого следует вызвать команду

   RCC->AHBENR |= RCC_AHBENR_GPIOAEN;

8. Для подключения передатчика (Tx) и приемника (Rx) к выводам PA9 и PA10 соответственно используем команды

   GPIOA->MODER |= 0x00280000;

   GPIOA->AFR[1] |= 0x00000110;

   Первая из них переключает выводы 9 и 10 порта A в режим альтернативной функции, а вторая – включает на этих выводах альтернативную функцию AF1 (рис. 13) (см. описание регистра GPIOx_AFRH).
9. Настройку режима работы выхода передатчика выполним с помощью трех команд:

   GPIOA->OTYPER &= ~GPIO_OTYPER_OT_9;

   GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR9;

   GPIOA->OSPEEDR |= GPIO_OSPEEDR_OSPEEDR9;

   Первая команда сбрасывает 9 бит регистра GPIOA_OTYPER, переключая его в режим «Push-Pull» (активный выход). Вторая команда отключает подтягивающие резисторы от вывода 9, сбрасывая оба бита для линии 9 в регистре GPIOA_PUPDR. Третья команда устанавливает максимальную скорость при синхронизации данных на выходе 9 порта A.

10. Настройку режима работы входа приемника выполним двумя командами:

    GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR10;

    GPIOA->PUPDR |= GPIO_PUPDR_PUPDR10_0;

    Первая команда сбрасывает оба бита, которые отвечают за режим работы подтягивающих резисторов на линии 10 порта A (оба подтягивающих резистора отключаются). Вторая команда включает на входе 10 порта A подтягивающий резистор по питанию. Включение подтягивающего резистора гарантирует, что вход приемника не окажется в «плавающем режиме», и так как в режиме ожидания на линии приемника должен быть уровень логической единице, то в качестве подтягивающего нужно выбрать резистор «pull-up».
11. Перед настройкой модуля USART1 необходимо сбросить бит UE в регистре USART1_CR1. Это действие запретит работу модуля и сделает возможным изменение параметров его конфигурации. Сброс бита выполняется командой

    USART1->CR1 &= ~USART_CR1_UE;

    Константа USART_CR1_UE определена в заголовочном файле stm32f0xx.h и имеет значение 0x1h.

12. Следующий шаг - настройка скорости работы приемника и передатчика выполняется в соответствие с тактовой частотой микроконтроллера 8 МГц и скорость обмена данными 115200 бит/с.Определим коэффициент деления частоты тактового генератора для модуля USART1: USARTDIV = 8000000 / 115200 = 69,(4). Округление до целого даст значение 69. Его нужно поместить в регистр USART1_BRR командой

    USART1->BRR=69;

13. Далее добавьте команду

    USART1->CR1 = USART_CR1_TE | USART_CR1_RE;

    Она разрешает работу приемника и передатчика в модуле USART1. Константы USART_CR1_TE и USART_CR1_RE определенны в заголовочном файле stm32f0xx.h и имеют значения 0x8h и 0x4h соответственно. Кроме того, эта команда сбрасывает остальные биты регистра USART1_CR1, что приводит к настройке следующих параметров:

    - количество бит данных – 8;
    - контроль четности – отключен;
    - прерывания по любым флагам USART - запрещены.

14. Чтобы задать количество стоп-бит в пакете следует использовать регистр USART1_CR2. Сброс всех установит стандартный режим работы USART, в котором число стоп-бит равно 1. Такой результат достигается командой

    USART1->CR2 = 0;

15. Завершите конфигурирование USART1 командой

    USART1->CR3 = 0;

    Она отключит использование контроллера прямого доступа к памяти при обмене данными посредством модуля USART1.

16. Последней командой в функции InitUSART1() нужно разрешить работу модуля USART1

    USART1->CR1 |= USART_CR1_UE;

    установив бит UE регистра USART1_CR1 в «1».

17. Откомпилируйте программу и постройте исполняемый hex-файл.
18. Загрузите hex-файл на микроконтроллер.

Полный код реализации функции InitUSART1() приведен на рисунке 14.

```c
/*-----------------------------------------------------------------------------------------------------------------------
**Проект: "USART-Эхо".
**Назначение программы: принять по USART пакеты данных от ПК и вернуть их обратно
**Разработчик: Долматов Алексей Викторович - ИЦ9
**Цель: продемонстрировать основы программирования USART микроконтроллера STM32F072RBT
**Решаемые задачи:
**        1. Конфигурирование линий GPIO на выполнение альтернативных функций;
**        2. Конфигурирование USART;
**        3. Демонстрация базовых принципов приема и передачи данных с помощью интерфейса USART (без использования прерываний).
**-----------------------------------------------------------------------------------------------------------------------*/

#include "main.h"

int main(void)
{
  InitUSART1();  //Инициализация USART1 лабораторного комплекса STM_01

  while(1)
  {
    //Получение данных из USART
    while ((USART1->ISR & USART_ISR_RXNE) == 0) {} //Чтение регистра состояния ISR и анализ флага RXNE: RXNE выставляется USART в 1, когда новый пакет данных получен приемником Rx и скопирован в RDR
    uint8_t d = (uint8_t)USART1->RDR;            //Копирование двнных из USART (регистр RDR) в программную переменную.
    //Чтение регистра RDR приводит к сбросу флага RXNE=0. После чего USART снова получает возможность просигналить программе о получении нового пакета от ПК.

    //Отправка данных в USART
    while ((USART1->ISR & USART_ISR_TXE) == 0) {}  /*Чтение регистра состояния ISR и анализ флага TXE:
                                                     TXE=0, пока данные из регистра TDR не скопированы в сдвиговый регистр передатчика Tx.
                                                     TXE автоматически выставляется USART в 1, когда копирование из TDR в сдвиговый регистр завершено
                                                     и регистр TDR готов к получению следующего пакета*/
    USART1->TDR = d;  //Передача данных из программной переменной в USART (регистр TDR).
    //Запись в регистр TDR приводит к сбросу флага TXE=0.
  }
}

//Функция инициализации USART лабораторного комплекса STM_01
void InitUSART1(void)
{
  //Включение тактирования USART1
  RCC->APB2ENR |= RCC_APB2ENR_USART1EN;  //RCC_APB2ENR_USART1EN=0x00004000;
  //Включение тактирования порта A
  RCC->AHBENR |= RCC_AHBENR_GPIOAEN;     //RCC_AHBENR_GPIOAEN=0x00020000;
  //Настройка линий порта A: PA9(TX_1) - выход передатчика; PA10(RX_1) - вход приемника
  GPIOA->MODER |= 0x00280000;            //Перевести линии PA9 и PA10 в режим альтернативной функции
  GPIOA->AFR[1] |= 0x00000110;           //Включить на линиях PA9 и PA10 альтернативную функцию AF1
  //Настройка линии передатчика Tx (PA9)
  GPIOA->OTYPER &= ~GPIO_OTYPER_OT_9;          //Сбросить 9 бит GPIOA->OTYPER - переключение в режим push-pull для линии PA9 (активный выход)
  GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR9;          //Отключение подтягивающих резисторов для линии PA9
  GPIOA->OSPEEDR |= GPIO_OSPEEDR_OSPEEDR9;     //Установка высокой скорости синхронизации линии PA9
  //Настройка линии приемника Rx (PA10)
  GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR10;         //Сброс режима подтягивающих резисторов для линии PA10
  GPIOA->PUPDR |= GPIO_PUPDR_PUPDR10_0;        //Включение подтягивающего резистора pull-up на входной линии PA10 (вход приемника Rx)
  //Конфигурирование USART
  USART1->CR1 &= ~USART_CR1_UE;//Запрещение работы модуля USART1 для изменения параметров его конфигурации
  USART1->BRR = 69;            /*Настройка делителя частоты, тактирующего USART и задающего скорость приема и передачи данных на уровне 115200 бит/с: Частота тактирующего генератора = 8 МГц;
                                 Скорость обмена по USART = 115200 бит/с; коэффициент деления = 8000000 / 115200 = 69,4444(4); Округленное значение = 69*/
  USART1->CR1 = USART_CR1_TE | USART_CR1_RE;   /*Разрешить работу приемника и передатчика USART. Остальные биты этого регистра сброшены, что обеспечивает: количество бит данных в пакете = 8;
                                                 контроль четности  - отключен; прерывания по любым флагам USART  - запрещены; состояние USART - отключен*/
  USART1->CR2 = 0;                             //Количество стоповых бит - 1
  USART1->CR3 = 0;                             //DMA1 - отключен
  USART1->CR1 |= USART_CR1_UE;                 //По завершении конфигурирования USART разрешить его работу (биту UE регистра CR1 присвоить 1).
  //Примечание: конфигурирование USART можно выполнять только в не активном состоянии: UE=0.
}
/*-----------------------------------------------------------------------------------------------------------------------
**Руководство пользователя:
**       1. Запустите программу на лабораторном комплексе STM_01;
**       2. На компьютере запустите любую программу для мониторинга последовательных портов ПК (Teminal, PuTTY или др.) и подключитесь к соответствующему COM-порту на скорости 115200 бит/с;
**       3. С помощью программы мониторинга отправьте на лабораторный стенд произвольный набор символов и отследите эхо-символы, вернувшиеся обратно
**-----------------------------------------------------------------------------------------------------------------------*/

```

Рис. 14. Программная реализация проекта «USART - Эхо»

Тестирование программы «USART - Эхо» с помощью PuTTY

19. Запустите микроконтроллер в режиме выполнения загруженной программы.
20. Запустите на ПК приложение PuTTY, выберите подключение «Serial», укажите COM-порт, к которому подключен стенд STM_01, и скорость обмена данными 115200 бит/с (рис. 15).

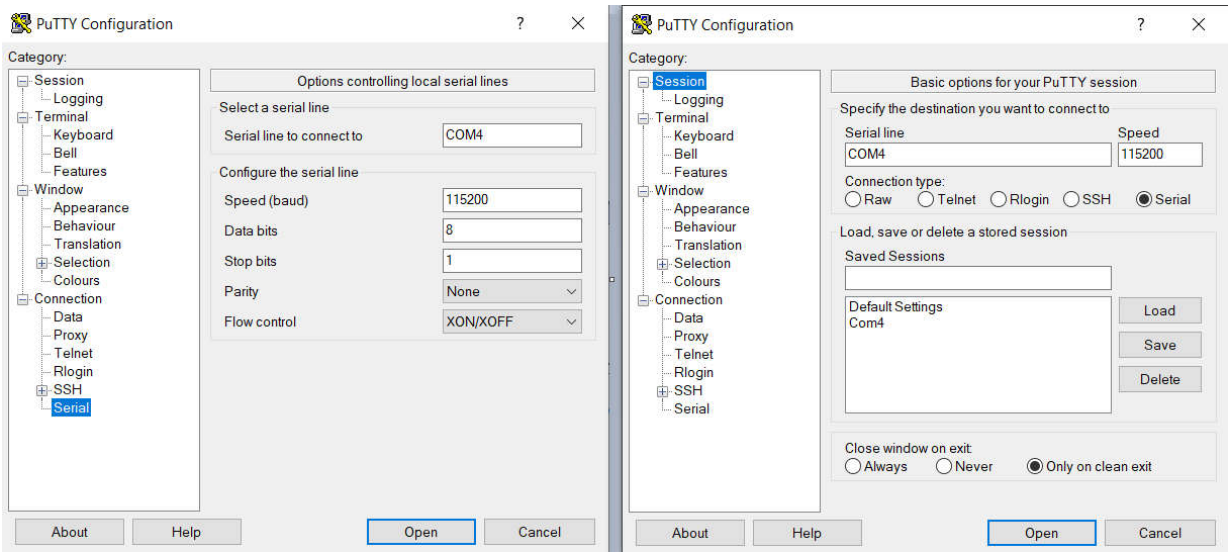

Рис. 15. Окна PuTTY для настройки параметров соединения

После нажатия кнопки «Open» откроется окно терминала.

21. Если теперь нажимать кнопки клавиатуры с буквами или цифрами PuTTY будет отправлять их через «Serial» на STM_01 и отображать символы, которые приходят в ПК через это соединение. Курсор в окне терминала показывает позицию, где будет отображен следующий символ, полученный из порта «Serial». Позицией курсора можно управлять с помощью стрелок на клавиатуре, а стирать символы в окне можно с помощью клавиши «backspace». В результате взаимодействия программы «USART - Эхо» и приложения PuTTY в окне терминала должны появиться эхо-символы, которые вводились с клавиатуры ПК (рис. 16).

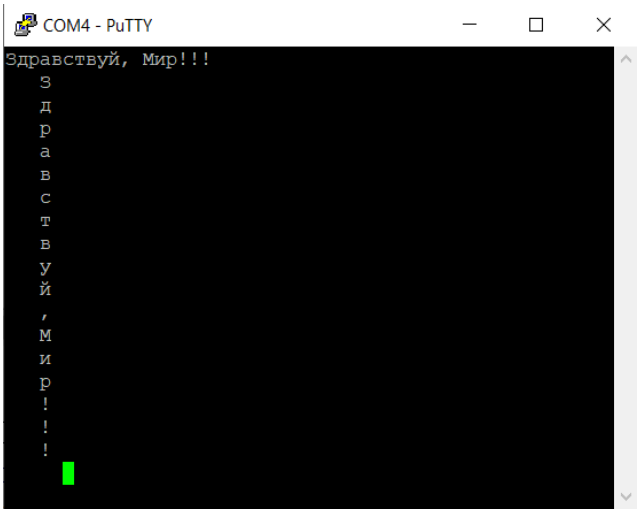

Рис. 16. Терминальное окно PuTTY

Если выключить программу на контроллере (например, перевести контроллер в состояние загрузки), то вывод символов в окно PuTTY прекратиться. При перезапуске программы на стенде STM_01 вывод символов в окне терминала должен возобновиться.

22. Протестируйте работу программы «USART – Эхо» на скорости соединения 9600 и 38400 бит/с.
23. Протестируйте работу программы «USART – Эхо» с включенным контролем четности.

Внимание! При выполнении п. 22 и 23 не забывайте правильно настраивать соединение как на стороне STM_01, так и на стороне ПК. Пример конфигурирования последовательного соединения в PuTTY показан на рисунке 15.

2.2. Самостоятельная работа

Цель Task2_2: сервер удаленного контроля светодиодов на стенде STM_01.

В процессе разработки программы микроконтроллера (сервера) для взаимодействия с терминалом PuTTY(клиентом) необходимо решить следующие задачи:

1. Нажатие кнопок на клавиатуре ПК должно приводить к изменению состояния светодиодов на стенде STM_01. При этом клавиша с цифрой «1» должна включать и выключать левый светодиод, «2» - изменять состояние следующего за ним светодиода и т.д. («8» соответствует крайне правому светодиоду). Нажатие кнопки «0» должно приводить к выключения всех светодиодов одновременно. Коды символов, полученные от ПК по USART, обратно не ретранслируются.
2. Разработать подпрограмму для отправки строки символов из буферного массива в ПК по USART. Признаком окончания строки в буферном массиве считать значение 0x0h.
3. После прием по USART символа из диапазона «1» - «8» и переключения состояния соответствующего светодиода нужно построить в буферном массиве текстовое сообщение, включающее обозначение светодиода и его текущее состояние, и отправить сообщение на ПК с помощью подпрограммы (п. 2). Сообщение должно заканчиваться символами LF(0x0Ah) – новая строка, и CR (0x0Dh) – возврат каретки. При нажатии «0» или «9» нужно последовательно отправить сообщения о состоянии всех светодиодов.
4. Протестировать взаимодействие сервера и клиента на режимах:
    - скорость обмена данными – 57600 бит/с, паритет – «четное», количество стоп-бит – 2;
    - скорость обмена данными – 19200 бит/с, паритет – «нечетное», количество стоп-бит – 1.

2.3. Индивидуальное задание

Цель Task2_3: сервер для определения ASCII-кодов клавиатуры с представлением их в заданной системе счисления.

В процессе разработки программы микроконтроллера (сервера) для взаимодействия с терминалом PuTTY(клиентом) необходимо решить следующие задачи:

1. Получение символа из порта удаленного соединения;
2. Вычисление кода символа в системе счисления, указанной в индивидуальном задании (см. **Приложение 1**);
3. Построение сообщения для клиента по следующему шаблону: «ФИО студента на латинице – основание системы счисления>код символа». Например, Сидоров Иван Петрович получил вариант задания, в котором основание системы счисления равно 11, тогда его серверная программа должна построить такое сообщение для клиента, пославшего символ «6»:

   sip-11>4A

   Терминал PuTTY при нажатии кнопки клавиатуры отправляет на STM_01 ASCII-код символа. Символу «6» соответствует код 54 (см. **Приложение 2**). В системе счисления с основанием 11 этот код представлен числом 4A.
4. Стирание предыдущего сообщения в терминале PuTTY и отображение на его месте следующего;
5. Отправка сообщения с помощью USART клиенту (терминалу PuTTY на ПК).
6. Ожидание следующего символа от клиента.
7. Тестирование взаимодействия сервера и клиента в режиме, который указан в варианте индивидуального задания (см. **Приложение 1**).

3. Подготовка отчета, представление и оценка работы

Комментарии в программном коде в качестве «Отчета о выполнении задания»

1. Комментарии в программном коде каждого проекта выступают в роли отчета и оцениваются в процессе защиты работы.
2. Структура комментариев, описывающих программный код должна быть следующей:
   - В начале файла «main.c» должны присутствовать сведения о разработчике (ФИО - группа), цели выполнения работы и задачах, которые решались в ходе создания программы;
   - Перед реализацией каждой функции следует указать ее назначение и привести список входных/выходных параметров с описанием;
   - Внутри тела функции следует выделить отдельные блоки, реализующие единую комплексную операцию обработки данных, и вынести в их заголовок сведения о назначении блока;
   - Каждая элементарная операция в строке кода должна быть снабжена комментарием;

- В конце файла «main.c» нужно привести комментарии, содержащие краткое руководство использования программы на стенде STM_01.

3. В заголовочных файлах (с расширением «h»), созданных разработчиком проекта, следует прокомментировать все строки кода. Однако можно опустить заголовочный и финальный комментарии со сведениями о проекте и руководстве пользователя.

Представление и защита работы

4. Для каждой части **Задания** нужно построить отдельный проект Keil µVision, протестировать и отладить его работу на стенде STM_01, снабдить программный код комментариями в соответствие с п. 3.1- 3.3.
5. Представление работы начинается с демонстрации преподавателю работоспособности программного кода проекта на стенде STM_01. Каждый проект (часть задания лабораторной работы) может быть представлена отдельно. В ходе представления преподаватель оценивает функциональность программы и качество написания программного кода. Студент должен быть готов ответить на вопросы преподавателя по использованным в проекте алгоритмам и их программной реализации.
6. После получения оценки за все проекты лабораторной работы следует создать единый архив (со всеми проектами), и загрузить его на сайт Eluniver.
7. По комментариям к проекту преподаватель оценивает качество отчета по каждому из выполненных заданий.
8. После представления всех заданий лабораторной работы студент получает право ее защиты, которая заключается в ответе на 2 контрольных вопроса из обозначенного в **разделе 4** списка. Каждый контрольный вопрос оценивается отдельно.

Структура оценки лабораторной работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
	Итого:	**100**

4. Контрольные вопросы

1. Как расшифровывается аббревиатура USART?
2. Что такое бод? Какие стандартные режимы работы USART, измеряемые в бодах, Вам известны?
3. Какова структура битового пакета, передаваемого или принимаемого USART?
4. Какова длина битового пакета, используемого для обмена данными с помощью USART?
5. Как рассчитать битовую скорость данных для заданной конфигурации USART? Приведите пример.
6. Каким образом осуществляется синхронизация передатчика и приемника?
7. Какой уровень сигнала соответствует стоп-биту?
8. Как приемник определяет количество стоп-бит в полученном пакете?
9. Как опознается старт-бит в линии приемника?
10. Что понимается под контролем четности передаваемых данных?
11. Какова максимальная скорость передачи может быть достигнута при обмене данными с помощью USART микроконтроллера STM32F072RBT, тактируемого с частотой 8 МГц?
12. Какие регистры используются передатчиком для отправки данных в линию?
13. Как организована буферизация данных в приемнике?
14. Каково назначение регистра TDR?
15. Каким образом задается скорость обмена данными в модуле USART?
16. Поясните назначение подтягивающего резистора на входной линии приемника USART?
17. Какой регистр изучаемого микроконтроллера позволяет управлять контролем четности USART?
18. Что произойдет с конфигурацией USART, если в регистр USARTx_CR2 поместить значение 0x00003000h?
19. Какой флаг USART показывает, что передача данных в линию завершена?
20. Назовите номер бита статусного регистра USART, содержащего флаг TXE?
21. Когда происходит сброс флага RXNE?
22. По какой причине возможна потеря данных в приемнике USART?
23. При каком условии изменение конфигурации USART невозможно?
24. Каким образом можно подключить модуль USART1 изучаемого микроконтроллера к порту B? Какие настройки нужно для этого сделать?
25. Каково назначение команды *USART1->ISR & USART_ISR_TXE* ?
26. Какая команда включает тактирование модуля USART3?
27. Какой ASCII-код, принимаемый через порт «Serial» в PuTTY приводит к стиранию символов в окне этого приложения?
28. Как можно определить значение цифры по ее ASCII-коду?
29. Каким образом в PuTTY осуществляется настройка параметров соединения с лабораторным комплексом STM_01?
30. Каким образом организуется отправка целой строки символов из STM_01 в PuTTY?

Приложение 1. Варианты индивидуальных заданий

№ вари-анта	Режим работы USART (скорость обмена, контроль четности, кол-во стоп-бит)	Основание системы счисления
1.	9600 бит/с, четное, 1 стоп-бит	5
2.	230400 бит/с, нечетное, 2 стоп-бита	13
3.	57600 бит/с, нет, 2 стоп-бита	9
4.	19200 бит/с, четное, 1,5 стоп-бита	4
5.	38400 бит/с, нечетное, 1 стоп-бит	11
6.	115200 бит/с, четное, 1,5 стоп-бита	8
7.	460800 бит/с, нет, 1 стоп-бит	13
8.	57600 бит/с, нечетное, 1,5 стоп-бита	12
9.	9600 бит/с, нет, 1,5 стоп-бита	7
10.	38400 бит/с, четное, 2 стоп-бита	11
11.	230400 бит/с, четное, 1 стоп-бит	4
12.	115200 бит/с, нет, 1 стоп-бит	9
13.	19200 бит/с, нечетное, 1,5 стоп-бита	6
14.	460800 бит/с, нечетное, 2 стоп-бита	16
15.	57600 бит/с, четное, 1 стоп-бит	14
16.	38400 бит/с, нет, 1,5 стоп-бита	13
17.	9600 бит/с, нечетное, 2 стоп-бита	4
18.	230400 бит/с, нет, 1,5 стоп-бита	3
19.	19200 бит/с, четное, 1 стоп-бит	11
20.	115200 бит/с, четное, 2 стоп-бита	5
21.	57600 бит/с, нет, 2 стоп-бита	15
22.	460800 бит/с, нечетное, 1 стоп-бит	6
23.	38400 бит/с, нечетное, 2 стоп-бита	12
24.	9600 бит/с, нет, 1,5 стоп-бита	7
25.	4800 бит/с, нет, 2 стоп-бита	11
26.	230400 бит/с, четное, 1 стоп-бит	4
27.	115200 бит/с, нечетное, 1 стоп-бит	15
28.	460800 бит/с, нет, 1,5 стоп-бита	7
29.	57600 бит/с, четное, 1,5 стоп-бита	3
30.	19200 бит/с, нечетное, 2 стоп-бита	5

Приложение 2. Таблица кодов ASCII Windows (Win-1251)

Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ	Dec	Hex	Символ
0	0	спец. NOP	32	20	спец. SP (Пробел)	64	40	@	96	60	`	128	80	Ђ	160	A0		192	C0	А	224	E0	а
1	1	спец. SOH	33	21	!	65	41	A	97	61	a	129	81	Ѓ	161	A1	Ў	193	C1	Б	225	E1	б
2	2	спец. STX	34	22	"	66	42	B	98	62	b	130	82	‚	162	A2	ў	194	C2	В	226	E2	в
3	3	спец. ETX	35	23	#	67	43	C	99	63	c	131	83	ѓ	163	A3	J	195	C3	Г	227	E3	г
4	4	спец. EOT	36	24	$	68	44	D	100	64	d	132	84	„	164	A4	¤	196	C4	Д	228	E4	д
5	5	спец. ENQ	37	25	%	69	45	E	101	65	e	133	85	…	165	A5	Ґ	197	C5	Е	229	E5	е
6	6	спец. ACK	38	26	&	70	46	F	102	66	f	134	86	†	166	A6	¦	198	C6	Ж	230	E6	ж
7	7	спец. BEL	39	27	'	71	47	G	103	67	g	135	87	‡	167	A7	§	199	C7	З	231	E7	з
8	8	спец. BS	40	28	(	72	48	H	104	68	h	136	88	€	168	A8	Ё	200	C8	И	232	E8	и
9	9	спец. Табуляция	41	29	)	73	49	I	105	69	i	137	89	‰	169	A9	©	201	C9	Й	233	E9	й
10	0A	спец. LF (Возвр. каретки)	42	2A	*	74	4A	J	106	6A	j	138	8A	Љ	170	AA	Є	202	CA	К	234	EA	к
11	0B	спец. VT	43	2B	+	75	4B	K	107	6B	k	139	8B	‹	171	AB	«	203	CB	Л	235	EB	л
12	0C	спец. FF	44	2C	,	76	4C	L	108	6C	l	140	8C	Њ	172	AC	¬	204	CC	М	236	EC	м
13	0D	спец. CR (Новая строка)	45	2D	-	77	4D	M	109	6D	m	141	8D	Ќ	173	AD	-	205	CD	Н	237	ED	н
14	0E	спец. SO	46	2E	.	78	4E	N	110	6E	n	142	8E	Ћ	174	AE	®	206	CE	О	238	EE	о
15	0F	спец. SI	47	2F	/	79	4F	O	111	6F	o	143	8F	Џ	175	AF	Ї	207	CF	П	239	EF	п
16	10	спец. DLE	48	30	0	80	50	P	112	70	p	144	90	ђ	176	B0	°	208	D0	Р	240	F0	р
17	11	спец. DC1	49	31	1	81	51	Q	113	71	q	145	91	‘	177	B1	±	209	D1	С	241	F1	с
18	12	спец. DC2	50	32	2	82	52	R	114	72	r	146	92	’	178	B2	І	210	D2	Т	242	F2	т
19	13	спец. DC3	51	33	3	83	53	S	115	73	s	147	93	“	179	B3	і	211	D3	У	243	F3	у
20	14	спец. DC4	52	34	4	84	54	T	116	74	t	148	94	”	180	B4	ґ	212	D4	Ф	244	F4	ф
21	15	спец. NAK	53	35	5	85	55	U	117	75	u	149	95	•	181	B5	µ	213	D5	Х	245	F5	х
22	16	спец. SYN	54	36	6	86	56	V	118	76	v	150	96	–	182	B6	¶	214	D6	Ц	246	F6	ц
23	17	спец. ETB	55	37	7	87	57	W	119	77	w	151	97	—	183	B7	·	215	D7	Ч	247	F7	ч
24	18	спец. CAN	56	38	8	88	58	X	120	78	x	152	98	[illegible]	184	B8	ё	216	D8	Ш	248	F8	ш
25	19	спец. EM	57	39	9	89	59	Y	121	79	y	153	99	™	185	B9	№	217	D9	Щ	249	F9	щ
26	1A	спец. SUB	58	3A	:	90	5A	Z	122	7A	z	154	9A	љ	186	BA	є	218	DA	Ъ	250	FA	ъ
27	1B	спец. ESC	59	3B	;	91	5B	[	123	7B	{	155	9B	›	187	BB	»	219	DB	Ы	251	FB	ы
28	1C	спец. FS	60	3C	<	92	5C	\	124	7C	|	156	9C	њ	188	BC	j	220	DC	Ь	252	FC	ь
29	1D	спец. GS	61	3D	=	93	5D	]	125	7D	}	157	9D	ќ	189	BD	S	221	DD	Э	253	FD	э
30	1E	спец. RS	62	3E	>	94	5E	^	126	7E	~	158	9E	ћ	190	BE	s	222	DE	Ю	254	FE	ю
31	1F	спец. US	63	3F	?	95	5F	_	127	7F		159	9F	џ	191	BF	ї	223	DF	Я	255	FF	я