Университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №3
по дисциплине
«Встроенные системы»

Работу выполнил:
студент группы P3410
Глушков Дима

Санкт-Петербург
2020

Введение

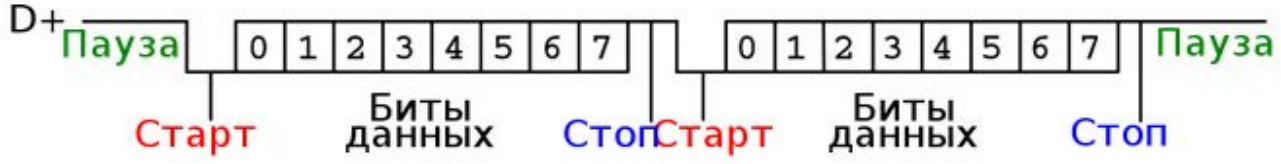
Данная лабораторная работа предназначена для получения практических навыков работы с UART(USART)на виртуальном стенде на базе микроконтроллера STM32F407с использованием HALфункцийописать и попробоватьразные способыпрограммнойреализации, а также поработатьс обработчиком прерываний USART.

Физические принципы работы

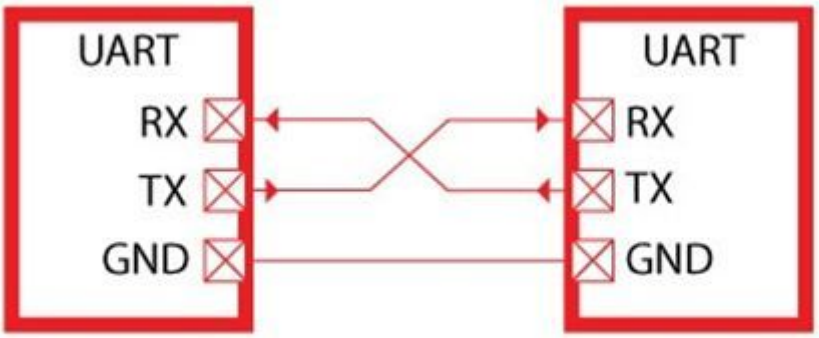
- UART (Universal Asynchronous Receiver/Transmitter) - универсальный асинхронный приёмопередатчик, интерфейс для связи цифровых устройств, предназначенный для передачи данных в последовательной форме.
- USART (Universal Synchronous-Asynchronous Receiver/Transmitter) -универсальный синхронно-асинхронный приёмопередатчик. Аналог UART, но дополнительно к возможностям UART, поддерживает режим синхронной передачи данных -с использованием дополнительной линии тактового сигнала.

Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной скоростью UART и для конкретного соединения указывается в бодах (что в данном случае соответствует битам в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод. Скорость (S, бод) и длительность бита (T, секунд) связаны соотношением T = 1 / S. Скорость в бодах иногда называют сленговым словом бодрейт или битрейт.

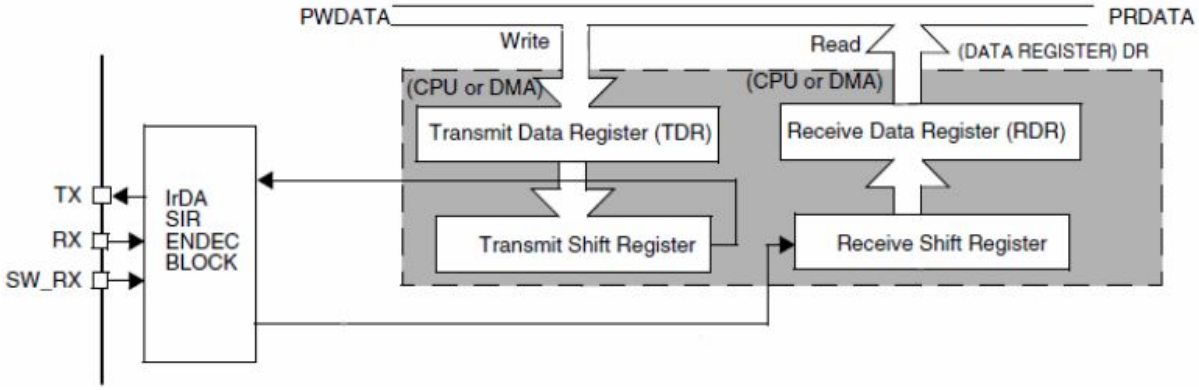
Поскольку синхронизирующие биты занимают часть битового потока, то результирующая пропускная способность UART меньше скорости соединения. Например, для 8-битных посылок формата 8-N-1 синхронизирующие биты занимают 20 % потока, что при физической скорости линии 115 200 бод означает полезную скорость передачи данных 92 160 бит/с или 11 520 байт/с.



UART является базой так широко используемого в прошлом интерфейса RS-232. В самом простом виде UART интерфейс представляет собой три провода: передача (RX), приём (TX) и земля (GND).



Основные регистры UART



Собственно передатчик состоит из 2 регистров: регистра сдвига (Transmit Shift Register) и буферного регистра (TDR).

Данное загружается в буферный регистр передатчика. Программно доступен только он. Если передача предыдущего данного закончена и регистр сдвига пуст, то данное перегружается в него, сдвигается и побитно поступает на выход TX. Как только данное перегружено в регистр сдвига, буферный регистр освобождается и в него может быть загружено новое слово. Оно будет ждать окончания передачи и автоматически перегрузится в сдвиговый регистр.

Приемная часть устройства такжесостоит из двух регистров: регистра сдвига (Receive Shift Register) и буферного регистра (RDR).

С входа RX данные поступают на сдвиговый регистр, сдвигаются и, после формирования полного слова, перегружаются в буферный регистр. Буферный регистр доступен программно.

Блокирующие и неблокирующие функции

Работа с UART может проимходить в двух режимах:

1. Блокирующий режим. Означает, что во время выполнения этих операций программа “зависает” в функциях HAL. Она ждет окончания передачи или приема данных. Дальнейший ход выполнения программы на время останавливается. Возможно только выполнение функций обработки прерываний, если они разрешены.

2. Неблокирующий режим. Для избежания блокировок программы, существуют функции с прерыванием по завершению передачи. У неблокирующей функции нет таймаута. Остальные аргументы те же, что и у блокирующей функции. По окончании отправки срабатывается прерывание и вызовается callback, в котором устанавливается флаг об окончании передачи.

Запись и чтение напрямую в DR

Подобный способ передачи по UART отличается от приведенных выше тем, что мы отказываемся от использования HAL-функций и осуществляем передачу путем записи напрямую в регистры UART.

Для передачи данных через порт USART необходимо записать передаваемый байт в регистр USART_DR конкретного порта, в прежде чем передавать следующий байт, необходимо дождаться окончания передачи предыдущего байта, анализируя состояние разряда TC регистра. Приём информации от порта можно осуществлять следующим образом: Дождаться приёма информации, считать принятый байт, сбросить флага окончания приёма автоматически после чтения регистра данных.

На рисунке ниже представлено графическое изображение регистра USART_DR.

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

Обработчик прерываний

Аппаратное прерывание это сигнал, сообщающий о каком-то событии. По его приходу выполнение программы приостанавливается, и управление переходит на функцию обработки прерывания (обработчик прерывания).

После отработки функции управление возвращается в прерванный код программы.
7RXT - прием байта
TXC - завершение отправки
UDRE - опустошение приемного буфера

Выводы

В результате выполнения данной лабораторной работы я ознакомился с основами передачи данных с использование UART и на практике реализовал разные способы передачи данных - в блокирующем режиме, в неблокирующем режиме, напрямую в DR, с использованием прерываний. В ходе выполнения и анализа полученных результатов было выявлено, что способ прямой записи в регистр DR был наиболее быстрым из всех описанных выше.

Приложение А. Исходный код программы

Блокирующий режим:

```
SDK_TRACE_Print("%s","Blocking test");
char str[] = "test start\n";
SDK_TRACE_Timestamp(P0, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
HAL_UART_Transmit(&huart4, (uint8_t*)str, sizeof(str)-1, 100);
SDK_TRACE_Print("%s","Test passed");
```

Неблокирующий режим:

```
SDK_TRACE_Print("%s","Unblocking start");
char str[] = "test start\n";
SDK_TRACE_Timestamp(P1, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
HAL_UART_Transmit_IT(&huart4, (uint8_t*)str, sizeof(str)-1);
SDK_TRACE_Print("%s","Test unblocking passed");
SDK_TRACE_Timestamp(PRINT, 0);
```

Работа с регистрами:

```
SDK_TRACE_Timestamp(PRINT, 1);
SDK_TRACE_Print("%s","Test register usartstart");
SDK_TRACE_Timestamp(P2, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 't';
SDK_TRACE_Timestamp(P2, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 'e';
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 's';
SDK_TRACE_Timestamp(P2, !HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 't';
while(!(UART4->SR& USART_SR_TC));
UART4->DR= ' ';
SDK_TRACE_Timestamp(P2, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 'U';
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 'A';
SDK_TRACE_Timestamp(P2, !HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 'R';
while(!(UART4->SR& USART_SR_TC));
UART4->DR= 'T';
SDK_TRACE_Timestamp(P2, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
while(!(UART4->SR& USART_SR_TC));
UART4->DR= '\r';
while(!(UART4->SR& USART_SR_TC));
```

```
UART4->DR= '\n';
SDK_TRACE_Print("%s","Test register passed");
SDK_TRACE_Timestamp(PRINT, 0);
```

Работа с прерываниями:

```
SDK_TRACE_Timestamp(PRINT, 1);
SDK_TRACE_Print("%s","Test interupt usart start");
SDK_TRACE_Timestamp(P3, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
HAL_UART_Transmit(&huart4, (uint8_t*)str, sizeof(str)-1,
100);
SDK_TRACE_Timestamp(P3, HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0));
SDK_TRACE_Print("%s","Test interupt passed");
SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Stop();
void UART4_IRQHandler(void)
{
  /* USER CODE BEGIN UART4_IRQn 0 */
  for (int i = 0; i < 10; i++)
  {
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
    SDK_TRACE_Timestamp(LED1, HAL_GPIO_ReadPin(GPIOD,
GPIO_PIN_13));
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
    SDK_TRACE_Timestamp(LED3, HAL_GPIO_ReadPin(GPIOD,
GPIO_PIN_15));
    HAL_Delay(250);
  }
  /* USER CODE END UART4_IRQn 0 */
  HAL_UART_IRQHandler(&huart4);
  /* USER CODE BEGIN UART4_IRQn 1 */
  /* USER CODE END UART4_IRQn 1 */
}
```