Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2
“Последовательный интерфейс UART”

Дисциплина: Проектирование вычислительных систем
Вариант 4

Выполнили: студенты группы
P34131

Бусыгин Дмитрий Алексеевич

Гиря Максим Дмитриевич

Преподаватель:

Пинкевич Василий Юрьевич

Санкт-Петербург
2024

Задание:

- Разработать и реализовать два варианта драйверов UART для стенда SDK-1.1M: с использованием и без использования прерываний.
- Драйверы, использующие прерывания, должны обеспечивать работу в «неблокирующем» режиме (возврат из функции происходит сразу же, без ожидания окончания приема/отправки), а также буферизацию данных для исключения случайной потери данных.
- В драйвере, не использующем прерывания, функция приема данных также должна быть «неблокирующей», то есть она не должна зависать до приема данных (которые могут никогда не поступить). При использовании режима «без прерываний» прерывания от соответствующего блока UART должны быть запрещены.
- Написать с использованием разработанных драйверов программу, которая выполняет определенную вариантом задачу.
- Для всех вариантов должно быть реализовано два режима работы программы: с использованием и без использования прерываний.
- Каждый принимаемый стендом символ должен отсылаться обратно, чтобы он был выведен в консоли (так называемое «эхо»).
- Каждое новое сообщение от стенда должно выводиться с новой строки.
- Если вариант предусматривает работу с командами, то на каждую команду должен выводиться ответ, определенный в задании или «ОК», если ответ не требуется.
- Если введена команда, которая не поддерживается, должно быть выведено сообщение об этом.
- Скорость работы интерфейса UART должна соответствовать указанной в варианте задания.

Вариант:

- Разработать программу-калькулятор. Ввод значений производится с компьютера через UART: xx…xyxx…x=, где x – десятичные цифры, y – знак (+, -, *, /). Ввод чисел завершается либо знаком операции (для первого числа), либо знаком «равно» (для второго числа), либо после ввода пяти цифр числа. Обратите внимание, что операнды не могут быть отрицательными, а ответ может.
- Размерность результата и обоих операндов должна быть short int (16-битовое знаковое число), и должна быть предусмотрена защита от переполнения. В случае выполнения недопустимых операций (ответ или вводимые числа больше, чем размер переменных в памяти) должен загораться красный светодиод, а в последовательный канал вместо ответа выводиться слово error.

- Включение/отключение прерываний должно осуществляться нажатием кнопки на стенде и сопровождаться отправкой в последовательный порт сообщения произвольного содержания, сообщающего, какой режим включен (с прерываниями или без прерываний).
- При смене режима необходимо сбрасывать все введенные пользователем данные и выводить приглашение начать ввод заново.
- Скорость обмена данными по UART – 19200 бит/с.

Описание работы алгоритма:

Получение и отправка сообщений в режиме прерываний и без них:

```
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
	HAL_UART_Transmit_IT(&huart6,(uint8_t*) &buf, write_len);
	write_len = 0;
}
void transmitIT(char* a, int len)
{
	if (write_len==0) HAL_UART_Transmit_IT(&huart6, (uint8_t*) a,  len);
	else strncpy((char*) buf+write_len, a, len);
	write_len += len;
}
void transmit(char* a, int len)
{
	interrupts_on
		? transmitIT(a, len)
		: HAL_UART_Transmit(&huart6,(uint8_t*) a, len, 10);
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
	transmitIT(input_line+input_line_size, 1);
	input_line_size++;
	process_line();
	ready_to_receive = 1;
}
void receive()
{
	if(interrupts_on)
	{
		if (ready_to_receive) HAL_UART_Receive_IT(&huart6,(uint8_t*)
input_line + input_line_size, 1);
		ready_to_receive = 0;
	}
	else if (HAL_UART_Receive(&huart6, (uint8_t *) input_line +
input_line_size, 1, 10) == HAL_OK)
	{
		transmit(input_line + input_line_size, 1);
```

```
                input_line_size++;
                process_line();
        }
}
```

Переключение режима работы с прерываниями:

```
void switch_interrupt_mode_if_needed()
{
        if (HAL_GetTick() - time < 100) return;
        int state = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
        if(state == 0 && buttonState == 1)
        {
                if (interrupts_on)
                {
                        interrupts_on = 0;
                        transmit("Interrupts off \n\r", strlen("Interrupts off \n\r"));
                }
                else
                {
                        interrupts_on = 1;
                        if (priority_mask) __set_PRIMASK(priority_mask);
                        transmit("Interrupts on \n\r", strlen("Interrupts on \n\r"));
                        ready_to_receive = 1;
                }
                input_line_size = 0;
        }
        buttonState = state;
        time = HAL_GetTick();
}
```

Ввод:

После получения данных:

- Анализируется размер вводимого числа.
- Проверяются уникальные командные символы (=, +, и другие).
- Выполняются вычисления (бизнес-логика)

```
void process_line()
{
        if ((input_line_size == 12 && input_line[input_line_size-1] != '=') ||
input_line[input_line_size-1] == '\r') goto err;
        if (input_line[input_line_size-1] == '=')
        {
                int b = 0;
                int (*func)(int, int);
                int i1=0, i2=0;
                for(int i=0; i<input_line_size-1; i++)
```

```
            {
                  if(input_line[i]>='0' && input_line[i]<='9')
                  {
                        if (b) i2=i2*10+input_line[i]-'0';
                        else i1=i1*10+input_line[i]-'0';
                  }
                  else if (!b) {switch(input_line[i])
                  {
                              case '+': func = func_sum; break;
                              case '-': func = func_sub; break;
                              case '*': func = func_mul; break;
                              case '/': func = func_dev; break;
                              default: goto err;
                        }
                        b++;
                  }
                  else goto err;
            }
            int answer = func(i1,i2);
            if(answer >= (2<<15) || i1 >= (2<<15) || i2 >= (2<<15)) goto err;
            char result[7] = {0};
            itoa(answer, result, 10);
            result[strlen(result)] = '\n';
            transmit(result, strlen(result));
            size = 0;
            return;
      }
      else return;
      err:
            transmit("\n error \n\r", strlen("\n error \n\r"));
            input_line_size=0;
            HAL_GPIO_WritePin(GPIOD,GPIO_PIN_15,1);
            time_red_starts=HAL_GetTick();
            red_should_blink=1;
            return;
}
```

Обработка ошибок:

- Если данные некорректны, выводится сообщение об ошибке (error).

Вычисления и вывод результата:

- Если данные корректны, выполняются вычисления.
- Результат отображается пользователю.

Основной цикл выполнения:

```
time = HAL_GetTick();
while (1)
{
	receive();
	switch_interrupt_mode_if_needed();
	if(red_should_blink && HAL_GetTick() - time_red_starts > 1000)
	{
		red_should_blink = 0;
		HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 0);
	}
}
```

Вывод:

В ходе выполнения лабораторной работы мы восстановили в памяти особенности работы с прерываниями, их виды и реализовали работу с ними на примере UART. Также поддержали режим работы с выключенными прерываниями