**Министерство науки и высшего образования Российской Федерации  
федеральное государственное автономное образовательное учреждение высшего образования  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»  
(Университет ИТМО)**

**Факультет программной инженерии и компьютерной техники**

**Лабораторная работа №2**

**Дисциплина «Встроенные системы»**

Выполнили: Калугина М.М.

Саржевский И.А.

Группа: P3402

Санкт-Петербург  
2020

# Описание cLab

Представленная система основана на sdk32.

Для работы предоставлен шаблон, в который можно писать пользовательский код. Исходный код шаблона находится по ссылке <https://github.com/lmtspbru/SDK_cLAB>.

Клиентская часть выполнена в виде веб-приложения, расположенного по адресу <https://sdk.ifmo.ru/remotesdk>

Для начала рассмотрим окружение:

В интерфейсе пользователя есть поле для загрузки bin-файла проекта и 2 окна: терминал, в котором отображается вывод программы по SDK\_TRACE\_Print и график прошедших событий.

Для работы необходимо написать исходный код на языке C, конфигурация портов ввода/вывода, а также различной перифирии производится в STM32CubeIDE (рекомендуется) или STM32CubeMX., собрать его и отправить бинарный файл на сервер. Для работы в системе был использован шаблон, предоставляемый создателями cLab для написания пользовательского кода.

Предоставленная нам система имеет 3 лампочки, возможность добавить 10 пользовательских событий и функции для вывода текста в консоль, при этом размер трассировочного буфера ограничен 256 байтами. Использование трассировочного буфера для вывода каких-либо данных в терминал и для отображения событий возможно между командами MX\_TRACE\_Init(); и SDK\_TRACE\_Stop();. При этом на выполнение пользовательского кода отведено 8 секунд.

Кроме того, в системе можно работать с прерываниями и таймером, хотя в шаблоне, предложенном для выполнения лабораторной настроенную в этом проекте периферию рекомендуется не менять

# Задание

На светодиодные индикаторы LED1 … LED8 должна выводиться анимация согласно варианту задания.

Скорость анимации задается с помощью переключателей SW. Если на переключателях выставлен код 0x0, то кадры анимации меняются каждые 500 мс. С увеличением значения, выставленном на переключателях SW анимация замедляется на T мс. Значение T задается вариантом задания.

Например, если по варианту задано, что T = 100 мс, это означает, что при установке переключателей в состояние SW = 0x1, кадры начинают меняться каждые 500+1\*100= 600 мс, если SW = 0x5, то кадры начинают меняться каждые 500+5\*100 = 1000 мс и т.д.

Все задержки должны быть реализованы с использованием прерываний от базовых таймеров.

# Задание с поправкой на cLab

Так как в нашей системе имеется только 3 лампочки, то анимация будет выводиться аналогично заданию, но на 3-х лампочках.

Анимация:

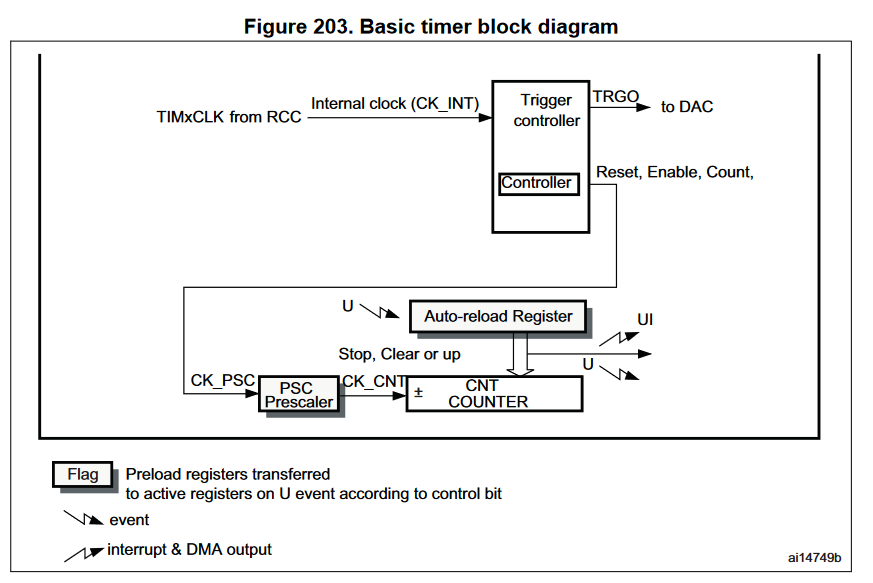
|  |  |  |  |
| --- | --- | --- | --- |
|  | LED1 | LED2 | LED3 |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

Так как в системе cLab не предусмотрены переключатели, то будем просто циклично менять значение интервала на каждое повторение анимации.

Начинать будем с 300 мс, затем увеличивать до 400 мс, затем 500 мс и снова 300.

# Ход работы

Было решено выполнить работу на основании прерываний таймера TIM7. Это 16-битный таймер, то есть может отсчитывать до 65535 клоков при частоте микроконтроллера, деленной на значение регистра Prescaler.



Таким образом, в main мы инициализируем значение Prescaler регистра таким образом, чтобы таймер считал на частоте примерно 10MHz. Это нужно для того, чтобы было удобно выставлять счетчик - в таком случае 300 мс будет равно 30000, 400 и 500 - 40000 и 50000 соответственно.

|  |
| --- |
| int main(void)  {  \_\_TIM6\_CLK\_ENABLE();  HAL\_Init();  SystemClock\_Config();  /\* Initialize all configured peripherals \*/  MX\_GPIO\_Init();  MX\_IWDG\_Init();  MX\_TIM7\_Init();  htim7.Instance->PSC=839; // 10mhz  htim7.Instance->ARR=30000; // 300ms  HAL\_TIM\_Base\_Start\_IT(&htim7);  MX\_TRACE\_Init();  SDK\_TRACE\_Start();  HAL\_Delay(8000);  SDK\_TRACE\_Stop();  while (1) { }  } |

В коде обработчика прерываний мы выставляем значения на пинах, отвечающих за нужные светодиоды и выводим в консоль информацию о том, какой светодиод сейчас горит и сколько миллисекунд прошло от начала работы, если все тайминги соблюдены верно. Дело в том, что если пытаться выводить информацию о горящих светодиодах в интерфейс cLab, то быстро происходит переполнение местного трассировочного буфера и мы не можем получить информацию даже о третьей итерации. Таким образом, был выбран следующий формат выведения информации в консоль:

|  |
| --- |
| TIMESTAMP: LED REFERENCE\_TIME |

Это удобно, потому что cLab записывает timestamp каждого лога, и можно легко сравнить REFERENCE\_TIME из вывода нашей программы с ним, чтобы убедиться, что все работает верно.

Код обработчика прерываний:

|  |
| --- |
| void TIM7\_IRQHandler(void)  {  if(led==1){  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_SET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_RESET);  SDK\_TRACE\_Timestamp(PRINT, 1);  SDK\_TRACE\_Print("%s %d","1", expected\_ms);  SDK\_TRACE\_Timestamp(PRINT, 0);  led=2;  }else if (led==2){  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_RESET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_RESET);  SDK\_TRACE\_Timestamp(PRINT, 1);  SDK\_TRACE\_Print("%s %d","2", expected\_ms);  SDK\_TRACE\_Timestamp(PRINT, 0);  led=3;  }else{  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, GPIO\_PIN\_RESET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);  HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, GPIO\_PIN\_SET);  SDK\_TRACE\_Timestamp(PRINT, 1);  SDK\_TRACE\_Print("%s %d","3", expected\_ms);  SDK\_TRACE\_Timestamp(PRINT, 0);  led=1;  }  counter++;  expected\_ms += iter\_ms;  if (counter == 3) {  htim7.Instance->ARR=40000; // 400ms  iter\_ms = 400;  }  if (counter == 6) {  htim7.Instance->ARR=50000; // 500ms  iter\_ms = 500;  }  if (counter == 9) {  htim7.Instance->ARR=30000; // 300ms  iter\_ms = 300;  counter = 0;  }  HAL\_TIM\_IRQHandler(&htim7);  } |

Вывод программы в терминал:

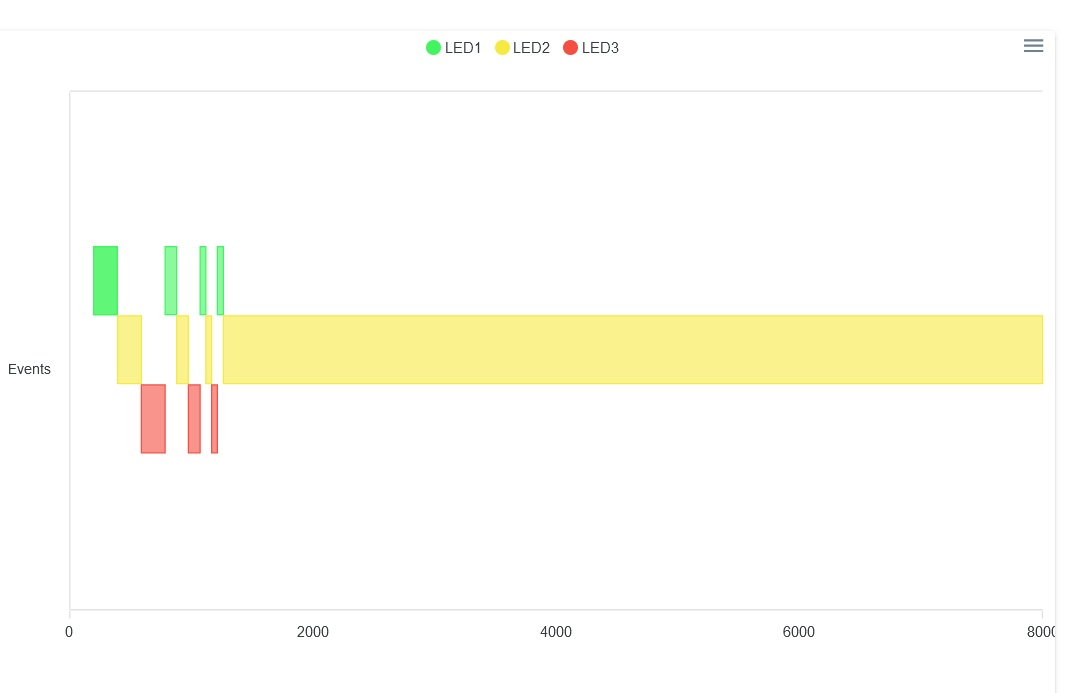
|  |
| --- |
| /\*\*\*\*\*\* SDK-1.1M Trace start (12/2/2020 1:41:15 AM) \*\*\*\*\*\*/  195.843: 1 195  495.813: 2 495  795.783: 3 795  1195.793: 1 1195  1595.803: 2 1595  1995.813: 3 1995  2495.823: 1 2495  2995.833: 2 2995  3495.843: 3 3495  3795.853: 1 3795  4095.863: 2 4095  4395.873: 3 4395  4795.883: 1 4795  5195.893: 2 5195  5595.903: 3 5595  6095.913: 1 6095  6595.923: 2 6595  7095.933: 3 7095  7395.943: 1 7395  7695.953: 2 7695  7995.963: 3 7995  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SDK-1.1M Trace stop \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

Как легко понять из вывода, нам удалось верно настроить чередование длин анимаций (300/400/500 мс).

Идея вывода анимации при помощи вывода команд в терминал была вынуждено применена при реализации программы, потому что трассировочного буфера, вмещающий в себя всего 256 байт данных для отображения хватает на 2.5 прокрутки анимации. Вывод отображения команд в графе событий выглядит нагляднее, но при этом не позволяет хранить много информации. На рисунке показан пример вывода анимации на графике событий:



При этом, попытка увеличения количества итераций приводит к переполнению и больше информации вывести на этот график невозможно:



# Вывод

В ходе данной лабораторной работы мы познакомились с системой cLab, а также настроили таймер для прерываний с динамической задержкой, в зависимости от которых выводили информацию на светодиоды.