Введение в микроконтроллеры

Домашнее задание №8

Евгений Зеленин

9 февраля 2025 г.

1. Постановка задачи

Условия задачи.

Объединим лекции 6,7,8: Соберите схему как в доп. ДЗ к занятию про таймеры, запустите ШИМ; снимайте данные АЦП каждые 100 мкс и запишите их в файл в ОЗУ вашего МК в виде электронной таблицы CSV: "%lu; %f\n time_ mcsec, u_ volts Предоставьте доступ к этой таблице по интерфейсу USB. Пришлите код и скриншот графика либо саму таблицу.

2. Описание устройства

Кратко сформулируем суть эксперимента. К микроконтроллеру подключен энкодер, вращение которого обрабатывается аппаратно с помощью таймера 1. Допустимый диапазон значений энкодера установлен от 0 до 99, а для управления зарядом конденсатора используется аппаратный ШИМ на таймере 2. Таймер 2 работает на тактовой частоте 1мГц, максимальное значение регистра счетчика установлено в 99, что соответствует частоте ШИМ в 10кГц (период 100мкс). Для установки нужного коэффициента заполнения, текущее положение энкодера [0-99] записывается в регистр CCR1 таймера 2. После включения, устройство каждые 100мкс записывает результаты преобразования АЦП в массив. Через 3500 циклов запись заканчивается, работа останавливается, производится инициализация USB устройства в режиме накопителя и массив данных выводится в файл.

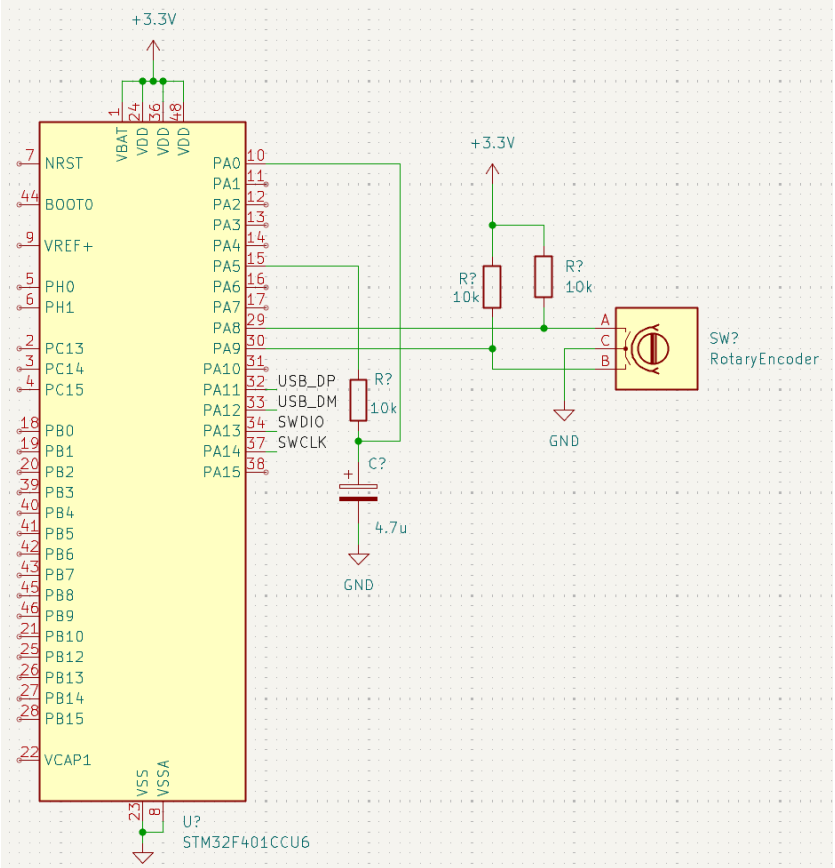

Рис. 1: Подключение rc-цепочки

К положительной обкладке конденсатора подсоединяется вывод АЦП. Запуск преобразования АЦП по регулярному каналу осуществляется по переднему фронту импульса ШИМ. Далее, после входа в основной цикл, программно запускается АЦП по инжектированному каналу для измерения Vref. Длительность преобразования для регулярного канала - около 25мкс, а длительность инжектированного - около 43мкс (56 циклов). Таким образом, оба преобразования укладываются в требуемый период измерений в 100мкс. Далее, осуществляется пересчет полученного значения с учетом Vref: $V_{dd} = 1.2 * 4095/V_{ref}$, $V_{adc} = adc * 1.2/Vref$

Для измерения длительности процессов использовался логический анализатор, а в ключевые моменты в программе на пинах PA1, PA2 изменялся уровень сигнала (рисунок 2).

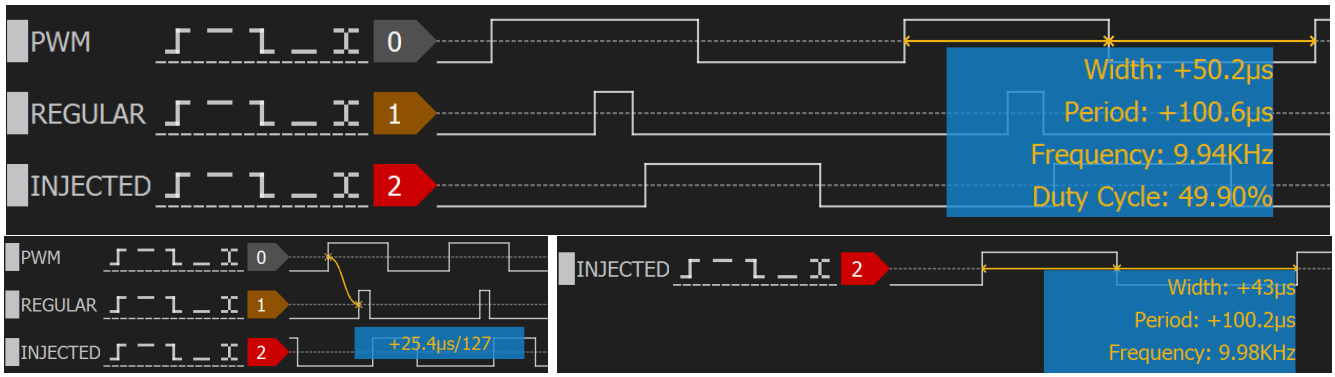

Рис. 2: Длительность процессов

Так как период измерения строго регламентирован в 100мкс, то в целях экономии места на накопителе в csv файл выводятся только полученные значения напряжения в вольтах.

Ниже приведено несколько примеров полученных измерений (рисунки 3 - 5):

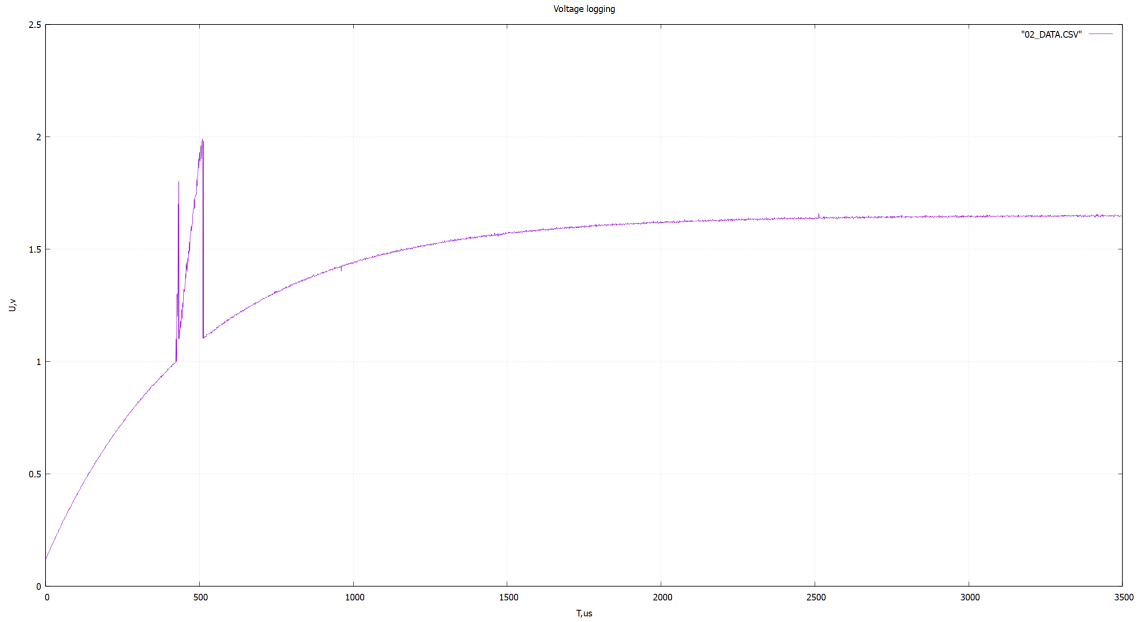

Рис. 3: Момент подключения устройства к USB-порту

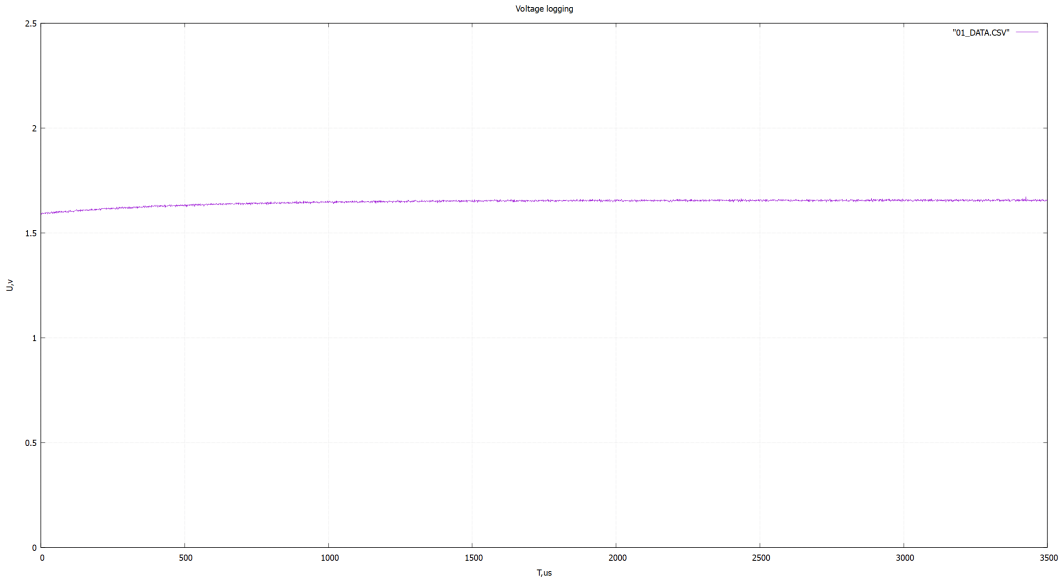

Рис. 4: Короткое нажатие на reset

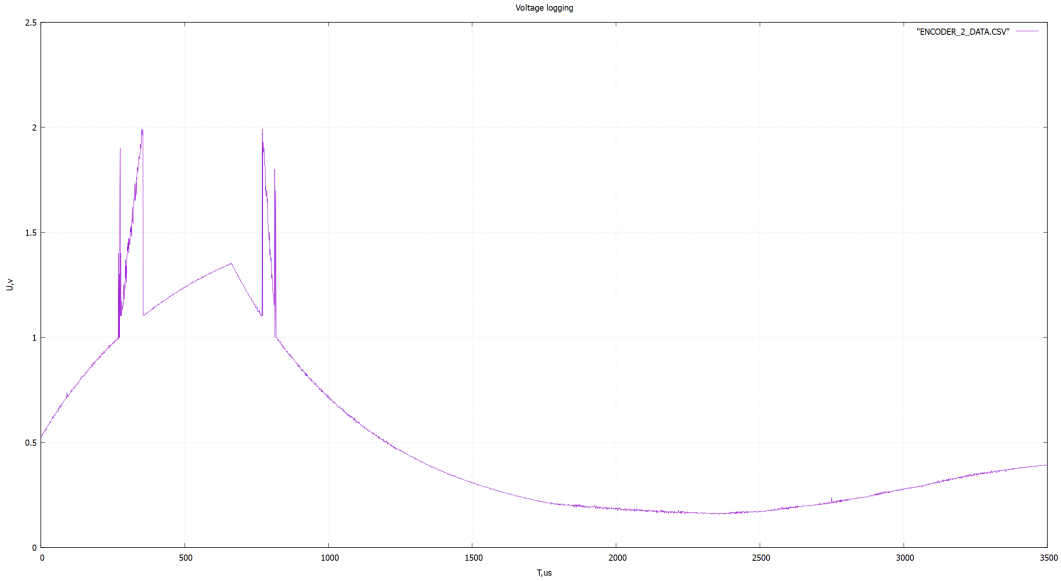

Рис. 5: Вращение энкодера в момент измерения

Что интересно, в момент подключения к USB всегда происходит выброс примерно через 100 мс (рисунок 3).

Если кратковременно нажать на reset, выброса нет, конденсатор не успевает разрядиться, можно видеть близкий к ровной линии график.

При вращении энкодера также наблюдаются всплески, это можно объяснить плохими контактами на dupont соединителях и помехами по USB.

3. Создание раздела FATfs

Чтобы создать раздел FATfs в ограниченном размере памяти, создадим диск с использованием ПК следующей командой: *mkfs.msdos -C fs.bin -F 12 -f 1 -r 16 -I -s 1 34*. В данном случае, создается диск размером 34кб (минимально возможный размер). После чего, с помощью STM32 Cube programmer файл "fs.bin"пересохраняется в ihex формат с нужным смещением начального адреса загрузки $0x8020000$. Далее, полученный образ файловой системы загружается во флеш мк по адресу $0x8020000$ через тот же STM32 Cube programmer.

Для инициализации RAMDISK используются следующие параметры в коде: размер сектора 512 байт, а количество секторов - 68. Адрес, по которому хранится образ чистой файловой системы - 0x8020000.

```
#define SECT_SZ (512U)
#define SECT_CNT (68U)
#define CLEAN_FS ((void *)0x8020000)
extern uint8_t ramdisk[SECT_CNT][SECT_SZ];
```

4. Исходный код проекта

Ниже можно ознакомиться фрагментами исходного кода: переопределение Sysinit, основной цикл, обработчики прерываний. Sysinit пришлось переопределить из-за особенностей операционной системы Windows:

- При подключении флеш накопителя, ОС Windows создает раздел System Volume и записывает туда служебную информацию, что съедает драгоценное место на виртуальном диске
- При отключении System Volume, остается проблема с обновлением содержимого диска, для чего диск нужно перемонтировать или изменить букву диска
- Следующая сложность в том, что ОС Windows каким-то образом препятствует записи содержимого в файл со стороны устройства. Возможно, это связано с одновременной записью на накопитель.
- Под ОС Linux таких проблем не наблюдается

Поэтому, в основном цикле сначала производится запись на RAMDISK через FATfs и только когда запись завершена производится инициализация USB-устройства в режиме накопителя.

Переопределение Sysinit

```
 /* USER CODE BEGIN SysInit */
 MX_GPIO_Init();
 MX_ADC1_Init();
 MX_TIM1_Init();
 MX_TIM2_Init();
 MX_FATFS_Init();
#if 0
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
```

```
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  MX_FATFS_Init();
  MX_USB_DEVICE_Init();
  /* USER CODE BEGIN 2 */
#endif
  /* USER CODE END 2 */
```

Основной цикл

```
 while (1) {
  //обработка вращения энкодера
  if (capture) {
   capture = 0;
   __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, count);
  }

  if (adc_ready && rec_count < MAX_REC) {
   HAL_GPIO_TogglePin(MARKER_1_GPIO_Port, MARKER_1_Pin);
            // Вывод сигнала на маркеры
   HAL_GPIO_TogglePin(MARKER_2_GPIO_Port, MARKER_2_Pin);
   HAL_ADCEx_InjectedStart(&hadc1);
   HAL_ADCEx_InjectedPollForConversion(&hadc1, 1);
   vref = HAL_ADCEx_InjectedGetValue(&hadc1, ADC_INJECTED_RANK_1);
   // Вывод сигнала на маркер
   HAL_GPIO_TogglePin(MARKER_2_GPIO_Port, MARKER_2_Pin);
            //пересчет результата измерений
   vcap[rec_count] = adc * 1200 / vref; // (adc * 3300) / 4095;
   adc = 0;
   adc_ready = 0;
   rec_count++;
  } else if (rec_count == MAX_REC) {
            //запись на RAMDISK
   HAL_ADC_Stop(&hadc1);
   HAL_ADCEx_InjectedStop(&hadc1);
   HAL_TIM_Encoder_Stop(&htim1, TIM_CHANNEL_1);
   HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_1);
   f_mount(&USERFatFS, USERPath, 1);
   f_open(&USERFile, "DATA.CSV", FA_CREATE_ALWAYS | FA_WRITE);
   for (uint16_t i = 0; i < MAX_REC; ++i) {
    f_printf(&USERFile, "%lu,%lu;\n", vcap[i] / 1000,
      vcap[i] % 1000);
   }
   f_close(&USERFile);
   rec_count++;
```

```
   //инициализация USB
   MX_USB_DEVICE_Init();
  }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
 }
  /* USER CODE END 3 */
```

Обработка прерываний

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
 if (htim->Instance == TIM1 && htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
  count = __HAL_TIM_GET_COUNTER(htim);
  direct = __HAL_TIM_IS_TIM_COUNTING_DOWN(htim);
  capture = 1;
 }
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc) {
 /* Prevent unused argument(s) compilation warning */
 if (hadc == &hadc1) {
  adc_ready = 1;
  adc = HAL_ADC_GetValue(&hadc1);
 }
 // Вывод сигнала на маркер
 HAL_GPIO_TogglePin(MARKER_1_GPIO_Port, MARKER_1_Pin);

 /* NOTE : This function Should not be modified, when the callback is needed,
  the HAL_ADC_ConvCpltCallback could be implemented in the user file
  */
}
```

Исходные коды проекта приложены к дополнительным материалам отчета.

5. Особенности работы

С этим заданием пришлось повозиться. Без логического анализатора и линукса на второй машине было бы совсем непросто разобраться с проблемами записи на USB-диск. Тот самый случай, когда WSL - не выход.

6. Заключение

В ходе выполнения работы получилось реализовать управление зарядом конденсатора с помощью ШИМ, собрать данные АЦП с периодом измерения в 100 мкс и осуществить передачу полученной информации на пк через виртуальный USB-диск.

7. Дополнительные материалы

Демонстрация работы и материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №08