681.51(07)
М - 545

№5109

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра синергетики и процессов управления

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы №3
по дисциплине

МИКРОПРОЦЕССОРНЫЕ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Для студентов направления подготовки
27.03.03 Системный анализ и управление

Таганрог 2016

УДК 081.51.01(07.07)

Составитель: Скляров А.А.

Методические указания к выполнению лабораторных работ №3 по дисциплине «Микропроцессорные информационно-управляющие системы». – Таганрог: Изд-во ЮФУ, 2016. -11 с.

Представлен лабораторный практикум по ряду базовых разделов дисциплины «Микропроцессорные информационно-управляющие системы». К каждой работе прилагаются краткие теоретические сведения и рекомендации по использованию среды разработки IAR Embedded Workbench.

Указания предназначены для студентов направления подготовки 27.03.03 Системный анализ и управление.

Лабораторная работа № 3
Использование прерываний микроконтроллера.

Целью данной работы является изучение основ работы с прерываниями, возможностями, которые они предоставляют; научиться использовать таймеры для выполнения действий в определенные временные интервалы.

Теоретический материал

Прерывания – механизм, который позволяет аппаратному обеспечению сообщать о наступлении важных событий в своей работе. В момент, когда происходит прерывание, процессор переключается с выполнения основной программы на выполнение соответствующего обработчика прерываний. Как только выполнение обработчика завершено, продолжается выполнение основной программы с места, в котором она была прервана.

Для использования прерываний необходимо вначале настроить регистр, который называется Nested Vector Interrupt Controller (NVIC), встроенный контроллер вектора прерываний. Данный регистр является стандартной частью архитектуры ARM и встречается на всех процессорах, независимо от производителя. NVIC разработан таким образом, что задержка прерывания минимальна. NVIC поддерживает встороенные прерывания с 16-ю уровнями приоритета.

Микроконтроллеры новой линейки могут содержать до 17 таймеров:

- два 16-битных таймера с расширенными функциями,
- два 32-битных таймера общего назначения,
- восемь 16-битных таймеров общего назначения,
- два 16-битных базовых таймера,
- два сторожевых таймера (независимый и оконного типа),
- один 24-битный системный таймер.

Часть таймеров могут конфигурироваться на работу в мультирежимном формате, позволяющем строить системы из таймеров.

Производитель STM32 разделяет все таймеры на три типа:

Таймеры с расширенными функциями имеют очень широкий функционал – комплементарные выводы для поддержки трехфазных двигателей, поддержка режимов счета в прямом и обратном направлениях, генерация ШИМ, каналы захвата/сравнения сигнала, режим одиночного импульса, поддержка DMA, дополнительные функции безопасности в случае сбоев, поддержка интерфейса энкодера и датчика Холла.

Таймеры общего назначения аналогичны таймерам с расширенными функциями, за исключением функций, связанных с управлением двигателями.

Отличия между таймерами общего назначения заключаются в различном количестве каналов захвата/сравнения, поддержкой направления счета и поддержкой DMA или ее отсутствием.

Базовые таймеры имеют наименьший функционал – отсутствие каналов захвата/сравнения, поддержки функций управления двигателями, и, как правило, предназначены для использования в качестве обычных счетчиков.

В данной лабораторной работе будут использоваться базовые таймеры (TIM6 и TIM7), которые имеют следующие технические характеристики:

- 16-битный счётчик с автоперезагрузкой.
- 16-битный программируемый делитель частоты: с 1 по 65535.
- Схема синхронизации для запуска ЦАП.
- Генерация прерывания и/или запроса DMA по переполнению счётчика.

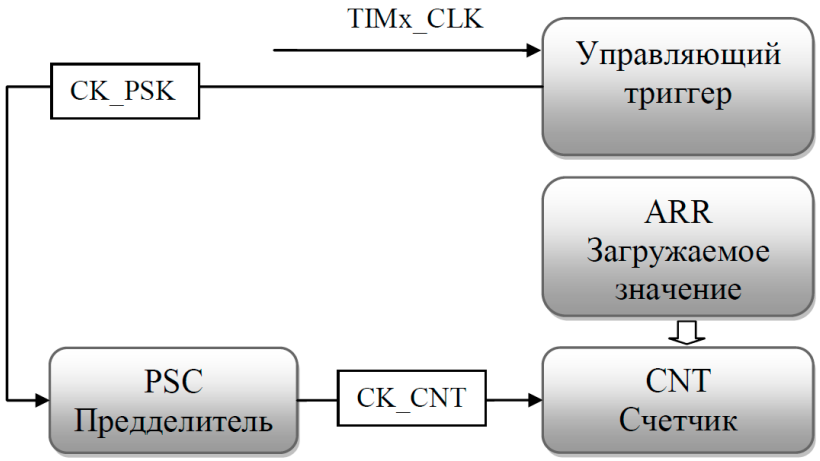


Рис. 1 Схема управления подсчетом импульсов

Конфигурирование базовых таймеров STM32F4xx

Для конфигурирования режимов базового таймера микроконтроллер имеет восемь регистров. Эти регистры имеют следующие названия и назначение:

- **CR1** – регистр основных настроек таймера.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

Биты 15:8 зарезервированы и недолжны изменяться.

Бит 7 (**ARPE**) изменяются для включения авто-перезагрузки:

0: Регистр автоперезагрузки TIMx_ARR работает без буфера. Если в программе содержимое регистра изменяется, то это происходит сразу же.

1: Для TIMx_ARR используется буфер. В этом случае новое значение в регистр автоперезагрузки заносится только в момент события – одновременно с вызовом прерывания или запросом DMA. Используется промежуточный регистр — “теневой”. То есть, когда счетный регистр достигнет предыдущего значения TIMx_ARR, и сбросится в “ноль”, только тогда и произойдет запись нового значения.

Биты 6:4 зарезервированы и недолжны изменяться.

Бит 3 (**OPM**) – режим одиночного импульса:

0: Счетчик не останавливается при появлении события,

1: Счетчик останавливается при появлении следующего события.

Бит 2 (**URS**) – источник возникновения событий:

0: Любое событие генерирует исключение или запрос ПДП:

- переполнение счетчика

- установка бита UG в регистре EGR (запуск сброса счетчика)

- сброс счетчика другим счетчиком.

1: Только переполнение счетчика генерирует исключение.

Бит 1 (**UDIS**) – отключение обновления счетчика:

0: Включение генераций события обновления (UEV). Событие UEV генерируется в следующих ситуациях:

- переполнение счетчика

- установка бита UG в регистре EGR (запуск сброса счетчика)

- сброс счетчика другим счетчиком.

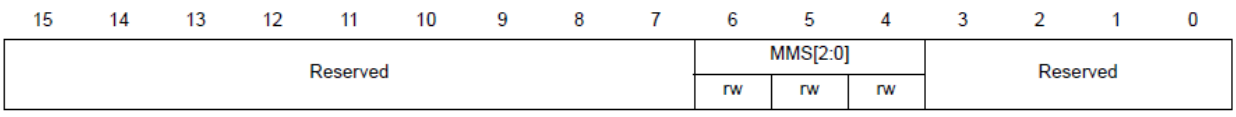
1: Отключение генераций события обновления (UEV).

Бит 0 (**CEN**) – включение/отключение счетчика

0: отключение счетчика

1: включение счетчика

- **CR2** – Регистр дополнительных настроек таймера



Биты 15:7 зарезервированы и недолжны изменяться.

Биты 6:4 (MMS) режим ведущего таймера, используется для выбора информации, которая будет передаваться ведомым таймерам для синхронизации:

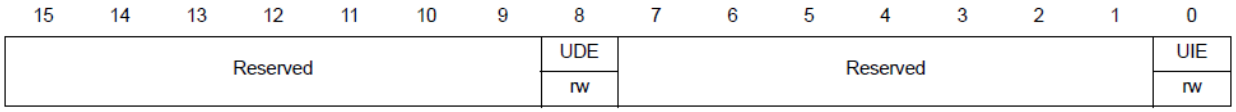
000: Сброс – бит UG в регистре EGR используется как внешний триггер.

001: Активация – запуск счетчиков сигналом CNT_EN. Применяться для активации нескольких ведомых таймеров.

010: Обновление – событие обновления используется как внешний триггер. В данной ситуации программный делитель ведущего таймера может быть использован для ведомых.

Биты 3:0 зарезервированы и недолжны изменяться.

- **DIER**– Регистр активации прерываний или запросов для прямого доступа к памяти.



Биты 15:9 зарезервированы и недолжны изменяться.

Бит 8 (**UDE**) Генерация запроса ПДП:

0: Генерация запроса ПДП отключена.

1: Генерация запроса ПДП включена.

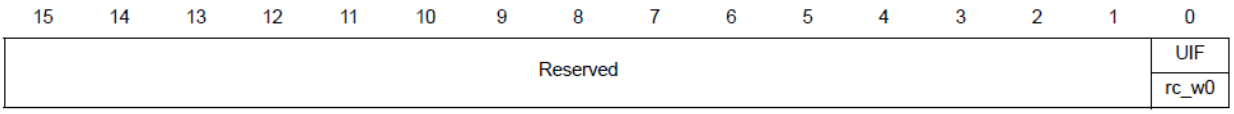
Биты 7:1 зарезервированы и недолжны изменяться.

Бит 0 (**UIE**) Генерация прерывания:

0: Генерация прерывания отключена.

1: Генерация прерывания включена.

- **SR** – регистр, с помощью которого определяется текущее состояние таймера;



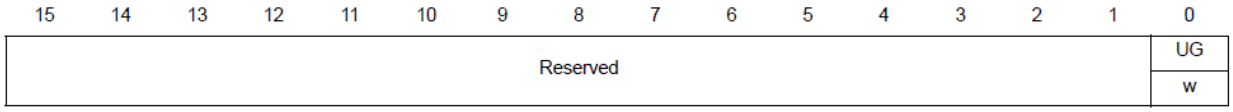
Биты 15:1 зарезервированы и недолжны изменяться.

Бит 0 (**UIF**) флаг возникновения прерывания:

0: Прерываний не возникало.

1: Существует прерывание ожидающее обработки.

- **EGR** – Регистр генерации событий;



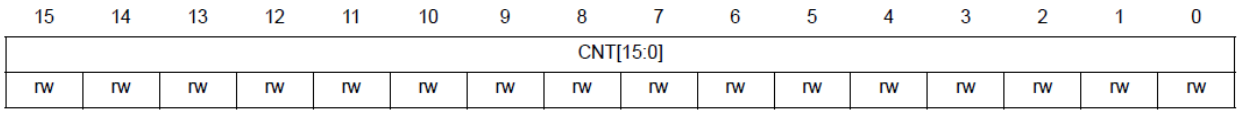
Биты 15:1 зарезервированы и недолжны изменяться.

Бит 0 (**UG**) генерация обновления. Данный бит может быть установлен программно, однако он автоматически обнуляется аппаратной частью процессора:

0: Не выполнять ни каких действий.

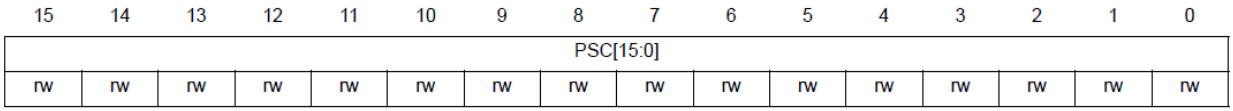
1: Переинициализировать счетчик таймера и сгенерировать сигнал обновления регистров. Следует отметить, что счетчик делителя также обновиться, однако значение величины делителя не измениться.

- **CNT**– Регистр значения счетчика



Биты 15:0 (**CNT**) – значение счетчика.

- **PSC**– Регистр программируемого делителя частоты;



Биты 15:0 (**PSC**) – значение программируемый делителя частоты. Содержит значение, которое записывается в регистр делителя при каждом запросе на обновление.

Частоту на выходе делителя можно рассчитать по следующей формуле:
Частота шины (в Гц) / (**PSC**[15:0] + 1) = Частота на выходе (в Гц)

- **ARR** – Регистр авто-перезагрузки.

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

Биты 15:0 (**ARR**) – значение регистра авто-перезагрузки. Счетчик блокируется, если значение равно 0.

Пример программы

Рассмотрим пример программы, которая демонстрирует работу прерываний базового таймера. Она переключает состояние светодиода LD4 после прохождения 1 секунды процессорного времени.

```
#include "stm32f4xx.h"

#define GPIO_OTYPER_PUSHPULL  0x00000000UL
#define GPIO_OTYPER_OPENDRAIN 0x00000001UL

#define GPIO_MODER_INPUT      0x00000000UL
#define GPIO_MODER_OUTPUT     0x00000001UL
#define GPIO_MODER_ALTER      0x00000002UL
#define GPIO_MODER_ANALOG     0x00000003UL

#define GPIO_OSPEEDER_LOW     0x00000000UL
#define GPIO_OSPEEDER_MEDIUM  0x00000001UL
#define GPIO_OSPEEDER_FAST    0x00000002UL
#define GPIO_OSPEEDER_HIGH    0x00000003UL

#define GPIO_PUPDR_NOPUPD     0x00000000UL
#define GPIO_PUPDR_PULLUP     0x00000001UL
#define GPIO_PUPDR_PULLDOWN   0x00000002UL

#define GPIO_AFR_AF0          0x00000000UL
#define GPIO_AFR_AF1          0x00000001UL
#define GPIO_AFR_AF2          0x00000002UL
#define GPIO_AFR_AF3          0x00000003UL
#define GPIO_AFR_AF4          0x00000004UL
#define GPIO_AFR_AF7          0x00000007UL
#define GPIO_AFR_AF9          0x00000009UL

int is_led_on = 0;

void SystemInit();
void Led4Init();
void TIM6Init(unsigned int time));

void TIM6_DAC_IRQHandler();
```

```
void main(void)
{
    SystemInit();

    Led4Init();

    NVIC_SetPriority(TIM6_DAC_IRQn, 1); //Приоритет прерывания
    NVIC_EnableIRQ(TIM6_DAC_IRQn); //Разрешаем обработку прерывания от таймера 6

    TIM6Init();

    while (1);
}

void Led4Init()
{
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOGEN; // подключение тактового генератора к
общей шине с портам

    unsigned int num_port = 14; // номер порта PG14 = RED_LED

    GPIOG->AFR[num_port >> 3] &= ~(0x0000000FUL << ((num_port & 0x07) * 4)); //
отчищаем биты от прежнего значения
    GPIOG->AFR[num_port >> 3] |= GPIO_AFR_AF0 << ((num_port & 0x07) * 4); // AFR =
AF0 или не использовать альтернативные функции

    GPIOG->MODER &= ~(0x00000003UL << (num_port * 2)); // отчищаем биты от прежнего
значения
    GPIOG->MODER |= GPIO_MODER_OUTPUT << (num_port * 2); // mode = output или
MODER14 = [29 бит = 0; 28 бит = 1]

    GPIOG->OSPEEDR &= ~(0x00000003UL << (num_port * 2)); // отчищаем биты от
прежнего значения
    GPIOG->OSPEEDR |= GPIO_OSPEEDER_HIGH << (num_port * 2); // OSPEEDER = HIGH или
OSPEEDER14 = [29 бит = 1; 28 бит = 1]

    GPIOG->OTYPER &= ~(0x00000001UL << num_port); // отчищаем биты от прежнего
значения
    GPIOG->OTYPER |= GPIO_OTYPER_PUSHPULL << num_port; // OTYPER = PUSHPULL или
OT14 = [14 бит = 0]

    GPIOG->PUPDR &= ~(0x00000003UL << (num_port * 2)); // отчищаем биты от прежнего
значения
    GPIOG->PUPDR |= GPIO_PUPDR_NOPUPD << (num_port * 2); // PUPDR = NOPUPD или
PUPDR14 = [29 бит = 0; 28 бит = 0]
}

void TIM6Init(unsigned int time)
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM6EN; // 42 МГц
    TIM6->PSC = 42000 - 1; // необходимо определить частоту
    TIM6->ARR = time; //Перезагружаемое значение
    TIM6->DIER |= TIM_DIER_UIE; //Разрешаем прерывание при переполнении счетчика
    TIM6->CR1 |= TIM_CR1_CEN;
}
void TIM6_DAC_IRQHandler() //Функция обработчика прерывания от таймера 6
{
    TIM6->SR &= ~TIM_SR_UIF; //Сбрасываем бит вызова прерывания.
    if (is_led_on == 0) //Проверяем текущее состояние светодиода - ON/OFF
        GPIOG->BSRRL = 1 << 14;  //Зажигаем LD4 светодиод
    else
        GPIOG->BSRRH = 1 << 14; // Гасим LD4 светодиод
    is_led_on = ~is_led_on; //Изменяем переменную, отображающую текущее состояние
светодиода
 }
```

Схемы подключения периферийных устройств необходимо брать из материалов лабораторной №2.

Порядок выполнения работы

1. На основе кода примера приложения создать новый проект в среде программирования IAR Embedded Workbench IDE.
2. Написать программу управления таймером с прерываниями в соответствии с вариантом задания.
3. Скомпилировать и отладить написанную программу в среде IAR Embedded Workbench IDE.
4. Провести тестирование работоспособности программы на отладочной плате STM32F4- DISCO.

Содержание отчета

1. Цель работы.
2. Описание последовательности действий для создания проекта.
3. Исходный текст программы с комментариями.
4. Выводы по работе.

Варианты заданий

№ варианта	Описание
1	Реализовать схему переменного включения/выключения светодиодов LD3 и LD4, расположенных на плате, с задержкой переключения равной 1 секунда.
2	Реализовать схему переменного включения и выключения светодиода LD3 с задержкой переключения равной 2 секунды и активируемого нажатием кнопки B1.
3	Реализовать схему переменного включения и выключения светодиода LD4 с задержкой переключения равной 3 секунды и активируемого нажатием кнопки B1.
4	Реализовать схему одновременного включения и выключения светодиодов LD3 и LD4 с задержкой переключения равной 1 секунда и активируемых нажатием кнопки B1.
5	Реализовать схему переменного включения и выключения светодиодов LD3 и LD4 с задержкой переключения равной 2 секунды и активируемых нажатием кнопки B1.
6	Реализовать схему одновременного включения/выключения светодиодов LD3 и LD4, расположенных на плате с задержкой переключения равной 3 секунды.
7[1]	Реализовать схему включения/выключения светодиода LD3, расположенного на плате, с задержкой переключения равной 3 секунды.

[1] Здание с минимальным уровнем сложности