Лабораторная работа № 1: Прерывания и их использование. Использование таймеров

Цель работы: ознакомится с понятием прерывания, возможностями, которые они предоставляют; научиться использовать таймеры для выполнения действий в опреленные временные интервалы.

Оборудование и программное обеспечение: плата STM32F4 Discovery, среда разработки CooCox CoIDE 1.7, инструменты построения проекта GNU Toolchain for ARM Embedded Processors, библиотека CMSIS при необходимости самостоятельного подключения файлов к проекту.

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 встроенные прерывания с 16-ю уровнями приоритета. Микроконтроллер STM32F407VG содержит 14 таймеров. В общем виде схема управления подсчетом импульсов может быть представлена следующим образом (рис. 3.4):

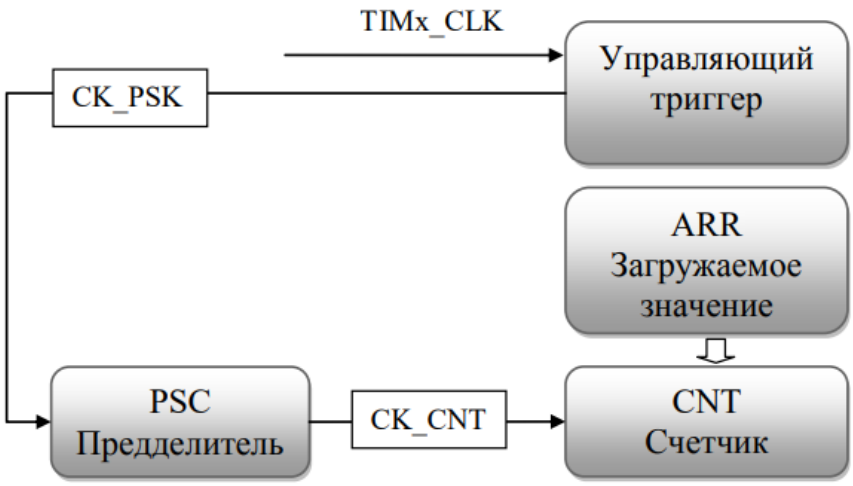

Рис. 3.4. Схема управления подсчетом импульсов

Производитель разделяет все таймеры на три типа: 1) с расширенными возможностями; 2) общего назначения; 3) базовые. Каждый таймер может иметь до 4 линий захвата/сравнения (именно они используются в режиме генерации ШИМ). Пример программы Данная программа демонстрирует работу с прерываниями и с таймерами. В ней реализовано переключение светодиода, который подключен к порту ввода/вывода, через определенные интервалы времени.

```
#include «stm32f4xx.h»

#include "stm32f4xx_gpio.h"

#include "stm32f4xx_rcc.h"

#include "stm32f4xx_tim.h"
```

```
#include "misc.h"
void INTTIM_Config(void);
void GPIO_Config(void);
int main(void)
{
GPIO_Config();
INTTIM_Config();
 while(1)
 {
 }
}
void TIM2_IRQHandler(void) {
if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET) {
TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
GPIOD->ODR ^= GPIO_Pin_13;
}
}
void GPIO_Config(void) {
GPIO_InitTypeDef gpio_struct;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
gpio_struct.GPIO_Pin = GPIO_Pin_13;
gpio_struct.GPIO_Mode = GPIO_Mode_OUT;
gpio_struct.GPIO_OType = GPIO_OType_PP;
gpio_struct.GPIO_PuPd = GPIO_PuPd_NOPULL;
gpio_struct.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_Init(GPIOD, &gpio_struct);
}
void INTTIM_Config(void)
{
NVIC_InitTypeDef nvic_struct;
nvic_struct.NVIC_IRQChannel = TIM2_IRQn;
nvic_struct.NVIC_IRQChannelPreemptionPriority = 0;
```

```
nvic_struct.NVIC_IRQChannelSubPriority = 1;
nvic_struct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&nvic_struct);
TIM_TimeBaseInitTypeDef tim_struct;
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
tim_struct.TIM_Period = 10000 - 1;
tim_struct.TIM_Prescaler = 168 - 1;
tim_struct.TIM_ClockDivision = 0;
tim_struct.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM2, &tim_struct);
TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
TIM_Cmd(TIM2, ENABLE);
}
```

Обращает на себя внимание включение дополнительного заголовочного файла – misc.h. Именно в нем описаны функции, перечисления, структуры для работы с прерываниями.

Ход работы

1. На основе кода примера приложения создать свой проект в среде разработки и проверить его работоспособность.

2. Ознакомиться с работой используемых функций, просмотрев исходный код, а также комментарии в исходных файлах библиотеки и в режиме отладки.

3. Создать новый проект в среде разработки для выполнения индивидуального задания, полученного от преподавателя.

4. Реализовать необходимую функциональность.

5. Запрограммировать плату и продемонстрировать работу программы.

Индивидуальные задания

1. С помощью одного из таймеров общего назначения сгенерировать задержку длительностью 1 с (на основе данных о рабочей частоте). Задержку генерировать на основе прерываний таймера. Продемонстрировать расчеты, подтверждающие правильность задания задержки.

2. Реализовать с помощью прерывания по переполнению таймера временную задержку с поочередным включением светодиодов на плате по кругу.