**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), всторенный 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. Микроконтроллер STM32F407VG содержит 14 таймеров. В общем виде схема управления подсчетом импульсов может быть представлена следующим образом (рис. 3.4):

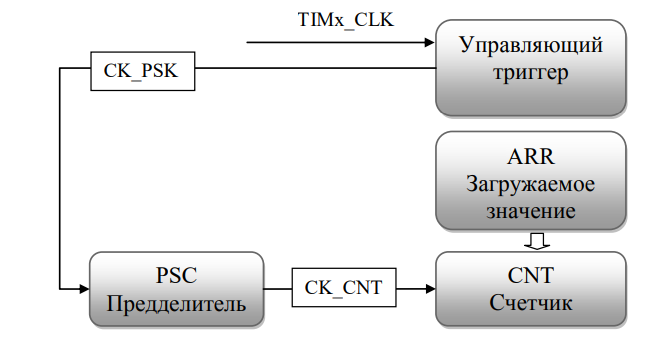


Рис. 3.4. Схема управления подсчетом импульсов

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.