

Современные 32-разрядные ARM-микроконтроллеры серии STM32: система прерываний

Олег Вальпа (г. Миасс, Челябинская обл.)

В статье приведено описание системы прерываний 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрена архитектура и состав её регистров, а также приведены практические примеры программ

ВВЕДЕНИЕ

Понятие прерывания (от англ. Interrupt) появилось в микропроцессорной технике давно и с течением времени прочно в ней укрепились. Прерываниям в микроконтроллерах отведена особая роль. Они позволяют микроконтроллеру моментально реагировать на определённые события (от англ. Event), повышая тем самым его оперативность. Прерывание возникает в результате определённого ожидаемого события и временно прекращает выполнение основного кода программы, переходя к выполнению подпрограммы, которая находится в обработчике прерывания. После завершения этой подпрограммы продолжается выполнение основного кода программы.

Существует понятие внешних и внутренних прерываний. Внешнее прерывание инициируется внешним событием, например нажатием кнопки поль-

зователем или замыканием контактов датчика. Внутреннее прерывание обусловлено некоторым событием, связанным, например, с приёмом или передачей данных через интерфейс, завершением счёта таймера и тому подобным.

Различают аппаратное и программное прерывания. Аппаратное прерывание возникает в результате воздействия внешних сигналов или аппаратной установки регистровых флагов событий. Программное прерывание формируется с помощью кода программы путём программной установки флагов событий или непосредственного вызова подпрограммы обработки прерывания.

ОРГАНИЗАЦИЯ ПРЕРЫВАНИЙ В МИКРОКОНТРОЛЛЕРАХ STM32

В микроконтроллерах семейства STM32 [1] возможно возникновение множества событий, способных породить прерывание. Как правило, эти

события связаны с каким-либо блоком периферии. Например, если речь о таком блоке периферии, как USART, то в нём могут возникать такие события, как передача завершена, приём завершён, возникла ошибка чётности и тому подобные. Использование прерываний позволяет программе мгновенно реагировать на эти события.

В случае возникновения двух разных прерываний от одного блока периферии происходит одно и то же прерывание. Например, если возникает прерывание при приёме байта в USART и прерывание по завершению передачи через этот же блок USART, то в обоих случаях будет вызван один и тот же обработчик прерывания. Для того чтобы определить, какое из возможных прерываний произошло, необходимо проанализировать флаги состояния данного блока. Перед выходом из обработчика прерывания эти флаги следует сбросить. Когда обработчик прерывания отработает, управление будет передано той строке кода, во время выполнения которой наступило прерывание. То есть основная программа продолжит работу с места прерывания.

В микроконтроллерах STM32 имеется возможность независимо разрешать и запрещать прерывания, назначать им приоритет и, при необходимости, вызывать прерывания программно.

Если для возникшего прерывания в программе нет обработчика, то будет вызван системный обработчик по умолчанию.

Наличие механизма приоритетности прерываний позволяет выполнять их обработку в соответствии со значимостью. Более приоритетное прерывание в микроконтроллерах STM32 имеет меньший номер. Например, если возникнет прерывание с приоритетом 7, управление программы будет передано его обработчику. Если в процессе выполнения кода обработчика возникнет новое прерывание с приоритетом меньше 7, то исполнение текущего обработчика прервётся, и управление будет передано обработчику более высокоприоритетного прерывания.

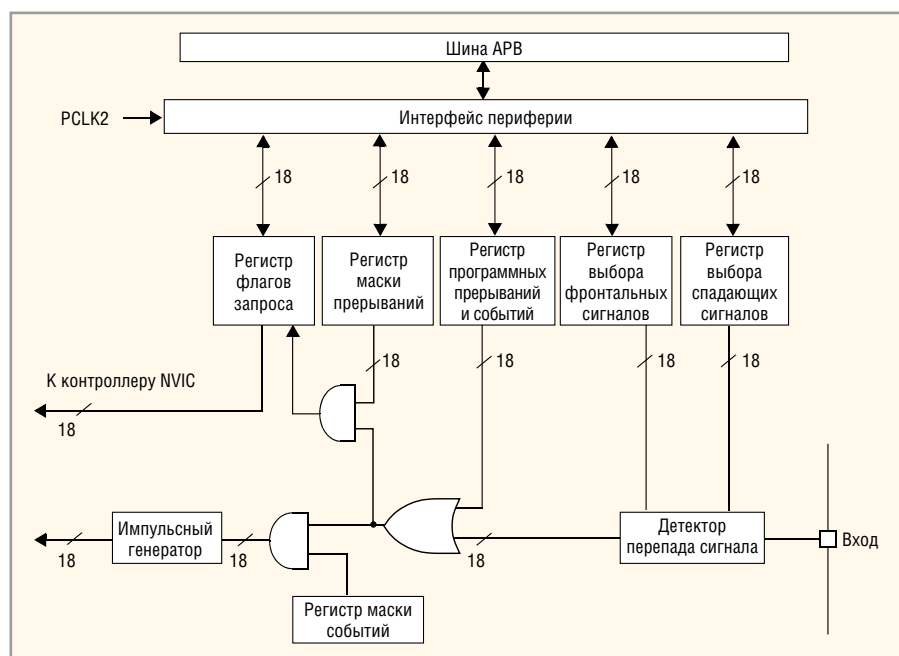


Рис. 1. Структурная схема блока прерываний

Для управления прерываниями в микроконтроллерах семейства STM32 существует специальный блок с названием NVIC, который представляет собой контроллер вложенных векторных прерываний (от англ. Nested Vectored Interrupt Controller, NVIC).

Контроллер NVIC поддерживает вложенность прерываний и их приоритетность. Каждому прерыванию при настройке NVIC присваивается свой приоритет. Если во время обработки низкоприоритетного прерывания возникает высокоприоритетное, то оно, в свою очередь, прервёт обработчик низкоприоритетного прерывания.

В зависимости от семейства микроконтроллера STM32 он может осуществлять обслуживание до 68 источников прерываний и выполняет следующие функции:

- разрешение и запрет прерывания;
- назначение и изменение приоритета прерывания от максимального приоритета 0 до минимального приоритета 15;
- автоматическое сохранение данных при выполнении одиночного или вложенного прерывания.

В библиотечном файле stm32f10x.h [1] содержится список всех прерываний для микроконтроллеров STM32 с указанием их приоритетности и адреса вектора подпрограммы обработки.

ФУНКЦИОНАЛЬНОЕ ОПИСАНИЕ

Как видно из структурной схемы, приведённой на рисунке 1, блок прерываний состоит из множества регистров конфигурации и логики формирования сигнала прерывания для 18 источников. В качестве этих источников прерываний могут выступать 16 линий GPIO, линия программируемого детектора напряжения PVD и линия тревоги часов реального времени RTC.

Для включения механизма прерываний линия прерывания должна быть сконфигурирована и разрешена. Это

выполняется программированием двух регистров запуска, где выбирается нужный перепад сигнала прерывания, и разрешением прерывания путём записи 1 в соответствующий бит регистра маскирования прерывания. Когда произойдёт выбранный перепад на линии внешнего прерывания, генерируется запрос на прерывание. Аналогично устанавливается флаг запроса от соответствующей линии прерывания. Данный флаг запроса можно сбросить записью 1 в регистр сброса прерывания.

Запрос прерывания можно также сгенерировать программно записью 1 в регистр программного прерывания.

Подключение любой из линий GPIO к 16 линиям внешних прерываний от EXTI0 до EXTI15 осуществляется в соответствии со схемой, приведённой на рисунке 2.

ОПИСАНИЕ РЕГИСТРОВ

Карта размещения регистров блока прерываний в пространстве памяти представлена в таблице 1. На карте также показано значение регистров после сброса.

Все эти регистры являются 32-разрядными, но в каждом из них используются только младшие 18 разрядов. Остальные разряды зарезервированы.

Регистр маскирования прерываний EXTI_IMR позволяет наложить битовую маску для внешних входов, разрешающую или запрещающую формирование прерываний от них. Значение 0 в разряде регистра маскирует запрос прерывания от соответствующей внешней линии, а значение 1 разрешает этот запрос.

Регистр маскирования событий EXTI_EMR позволяет выполнять операции маскирования для событий аналогично регистру EXTI_IMR.

Регистр EXTI_RTSR осуществляет задание линий, запускающихся по фронту сигнала.

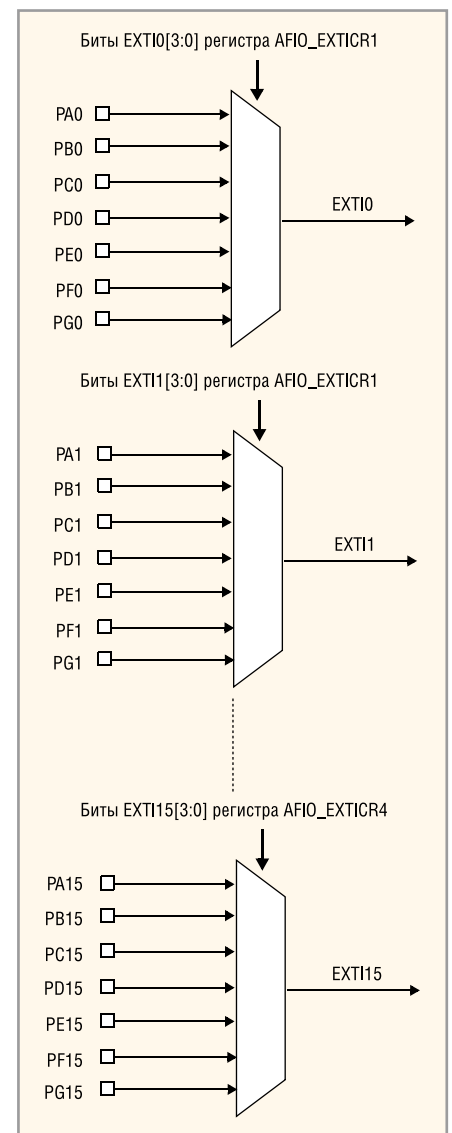


Рис. 2. Схема подключения к линиям прерываний

Регистр EXTI_FTSR позволяет задать линии, запускающиеся по срезу сигнала.

Регистр программных прерываний и событий EXTI_SWIER даёт возможность сформировать программное прерывание или событие. Запись 1 в биты этого регистра, имеющие нулевое значение, установит соответствующий флаг запроса в регистре EXTI_PR. Если разрешено прерывание от этой линии в регистрах EXTI_IMR и EXTI_EMR, то

Табл. 1. Карта регистров блока прерываний

Сдвиг	Регистр	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Резерв															MR[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Резерв															EMR[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Резерв															RTSR[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Резерв															FTSR[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Резерв															SWIER[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Резерв															PR[17:0]																
	Исх. значение																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

будет сгенерирован запрос на прерывание.

Регистр флагов запроса EXTI_PR позволяет определить источник формирования запроса на прерывание. Биты этого регистра устанавливаются,

когда на внешней линии прерывания произошло заданное условие запуска.

Для настройки выводов портов, подключаемых к линии внешних прерываний EXTI, используются регистры альтернативных функций ввода-вывода

AFIO_EXTICR1...AFIO_EXTICR4. Организация этих регистров представлена в таблице 2.

С их помощью можно назначить произвольный вывод «х» любого порта, как вход прерывания от 0 до 15, путём записи в соответствующую группу разрядов EXTIx. Эти разряды принадлежат регистрам AFIO_EXTICR1...AFIO_EXTICR4 и могут принимать следующие двоичные значения: 0000 для порта PAx, 0001 для PBx, 0010 для PCx и так далее до 0111 для PGx.

Например:

```
AFIO->EXTICR[0] |= 0x0000; //
Подключить EXTI0 к PA0
AFIO->EXTICR[0] |= 0x0001; //
Подключить EXTI0 к PB0
AFIO->EXTICR[4] |= 0x1000; //
Подключить EXTI15 к PA0
```

ПРИМЕР ПРОГРАММЫ

В листинге приведён пример программного кода, который будет обрабатывать прерывание, возникающее при нажатии на кнопку, и при этом инвертировать состояние индикатора. В программе предполагается, что кнопка подключена к порту PA0, а индикатор – к порту PC3.

Из этой программы видно, что операции по анализу состояния кнопки и изменению состояния индикатора возложены на прерывания. Поэтому данные процедуры будут выполняться автоматически, как в параллельном потоке для основного кода программы. Это наглядный пример одной из реализаций так называемой многозадачности систем.

В бесконечном цикле программы можно разместить код, предназначенный для выполнения других необходимых задач.

Подробнее познакомиться с системой прерываний поможет первоисточник [2].

ЛИТЕРАТУРА

1. www.st.com.
2. www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf.



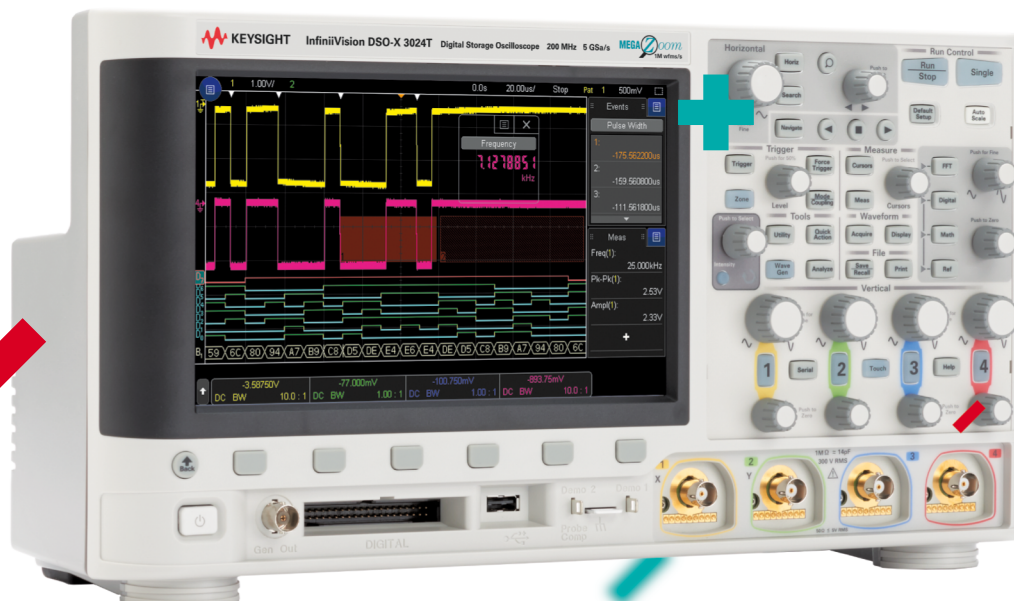
Листинг

```
// Подключение библиотечных функций
#include <stm32f10x.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_gpio.h>
// Подпрограмма инициализации кнопки
void init_key()
{
    RCC->APB2ENR |= RCC_APB2Periph_GPIOA; // Включить тактирование порта PA
    GPIOA->CRL &= ~GPIO_CRL_MODE0; // Обнулить разряды MODE
    GPIOA->CRL &= ~GPIO_CRL_CNF0; // Обнулить разряды CNF
    GPIOA->CRL |= GPIO_CRL_CNF0_1; // Вывод как дискретный вход
    GPIOA->BSRR = GPIO_BSRR_BS0; // Включить подтягивающий резистор
    AFIO->EXTICR[1] = AFIO_EXTICR1_EXTIO_PA; // Подключить вход EXTI
    EXTI->FTSR |= EXTI_FTSR_TR0;
    EXTI->IMR |= EXTI_IMR_MR0;
    NVIC_EnableIRQ (EXTIO_IRQn); // Разрешить прерывание
}
// Подпрограмма инициализации индикатора
void init_led()
{
    RCC->APB2ENR |= RCC_APB2Periph_GPIOC; // Включить тактирование порта PC
    GPIOC->CRL &= ~GPIO_CRL_MODE3; // Обнулить разряды MODE
    GPIOC->CRL &= ~GPIO_CRL_CNF3; // Обнулить разряды CNF
    GPIOC->CRL |= GPIO_CRL_MODE3_1; // Подать тактирование 2 МГц
    GPIOC->CRL &= ~GPIO_CRL_CNF3; // Выход общего назначения симметричный
}
// Обработчик прерывания
void EXTI0_IRQHandler(void)
{
    if(EXTI->PR & (1<<0)) // Проверить прерывание от EXTI0
    {
        GPIOC->ODR = GPIOC->ODR ^ (1<<3); // Изменить состояние индикатора
        EXTI->PR |= EXTI_PR_PR1; // Сбросить флаг прерывания
    }
}
// Главный модуль программы
int main(void)
{
    init_key();
    init_led();
    GPIOC->ODR = GPIO_ODR_ODR8;
    __enable_irq();
    while(1) // Бесконечный цикл
    {
        // Место для других операторов программы
    }
}
```

Табл. 2. Карта регистров AFIO_EXTICRx

Сдвиг	Регистр	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	AFIO_EXTICR1	Резерв																EXTI3[3:0]			EXTI2[3:0]			EXTI1[3:0]			EXTI0[3:0]						
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	AFIO_EXTICR2	Резерв																EXTI7[3:0]			EXTI6[3:0]			EXTI5[3:0]			EXTI4[3:0]						
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	AFIO_EXTICR3	Резерв																EXTI11[3:0]			EXTI10[3:0]			EXTI9[3:0]			EXTI8[3:0]						
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AFIO_EXTICR4	Резерв																EXTI15[3:0]			EXTI14[3:0]			EXTI13[3:0]			EXTI12[3:0]						
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ДИПОЛЬ



Новые осциллографы Keysight Technologies InfiniiVision 3000T серии X

Производительность старших серий осциллографов теперь доступна в сегменте среднего класса!

Революционная технология сенсорного запуска InfiniiScan Zone Trigger, емкостный сенсорный экран, специально разработанный пользовательский интерфейс, функциональность нескольких приборов в одном – и все это в сочетании с бескомпромиссной скоростью обновления более 1 млн. осциллограмм в секунду.

- Функциональность «6 приборов в 1»: осциллограф, частотомер, вольтметр, генератор, логический анализатор и анализатор протоколов
- Полоса пропускания до 1 ГГц
- Скорость обновления осциллограмм на экране – 1 млн. осцилл./с
- Аппаратное декодирование протоколов и тестирование по маске
- Расширенный математический анализ в базовой конфигурации, 38 автоматических измерений

Реклама



Инновационные решения для электронной промышленности

Санкт-Петербург / Москва / Нижний Новгород
www.dipaul.ru / info@dipaul.ru