Есть ли у вас вопросы?

Запись одного бита

- Установка одного бита:
  - a |= 1<<7; // установить седьмой бит
- Сброс одного бита:
  - a &= ~(1<<3); // сбросить третий бит
- Инверсия одного бита:
  - a ^= 1<<5; // инверсия пятого бита

Чтение одного бита

if(a & (1<<7)) – это условие истинно, если седьмой бит равен единице.

Писать if(a & (1<<7) == 1<<7) можно, но бессмысленно.

Скобки лучше ставить. Серьезно.

Запись нескольких бит

- Объединение через ИЛИ:
  - a |= (1<<7) | (1<<8); // установить седьмой и восьмой биты
- Три волшебных числа:
  - a |= 0x3<<7; // установить седьмой и восьмой биты
  - a |= 0x7<<7; // установить седьмой, восьмой и девятый биты
  - a |= 0xF<<7; // установить 7,8,9 и 10й биты
- Шестнадцатеричный код без сдвигов:
  - 0_{16} === 0000_2. Т.е. 100_{16} == $1\ 0000\ 0000_2$

Чтение нескольких бит (аналогично)

- Объединение через ИЛИ:
  - a & (1<<7 | 1<<8)
- Три волшебных числа:
  - a & (3<<7)
- Шестнадцатеричный код без сдвигов:
  - a & 0x180

Как помигать светодиодом?

Что такое мигание?

Это когда какое-то время светодиод горит, а потом какое-то время не горит!

Значит, нужно:

1. Зажечь светодиод
2. Подождать
3. Погасить светодиод
4. Подождать
5. Повторить 1-4

Как подождать?

С помощью функции delay? Но как она работает?

Самый простой способ – пустой цикл for:

```
for(uint32_t i=0; i<1000; i++) {;}
```

Ничего не делать 1000 раз подряд.

Количество итераций выбирается либо на глаз, либо исходя из частоты процессора (у нас – 72 МГц)

А можно ли чуть попроще помигать?

Можно:

1. Подождать
2. Инвертировать состояние ножки
3. Повторить 1-2

Краткое содержание этой серии

- Как на микроконтроллере измерять время
- Что такое прерывания и зачем они нужны

Как измерять время?

Вопрос на засыпку: что значит «измерять время»?

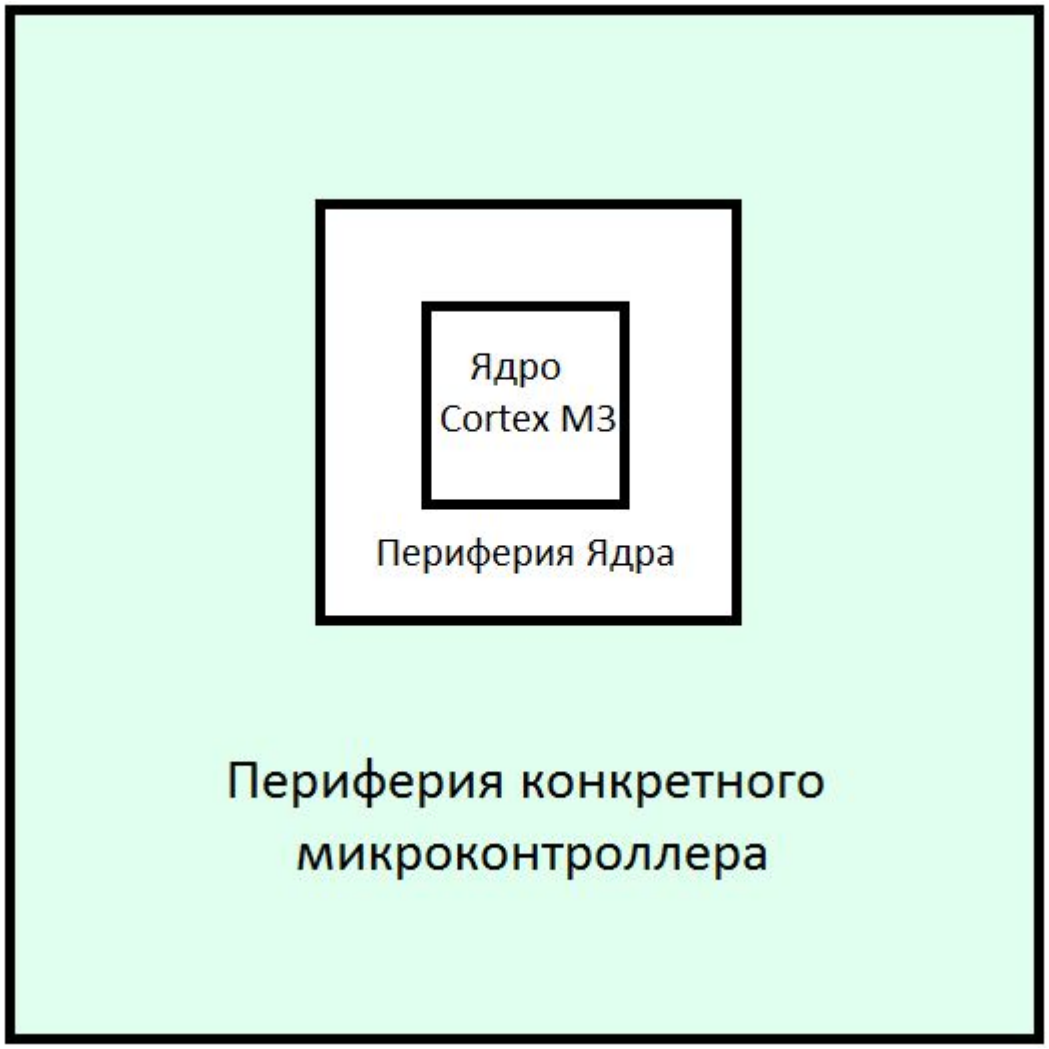
- Измерять, сколько времени прошло между событиями А и Б
- Отмерять необходимый промежуток времени и что-нибудь делать

Как измерять время?

И как же это сделать?

- Использовать процессорные такты как «тики» - например, с помощью пустых циклов
- Использовать специализированное периферийное устройство - таймер

Микропроцессор и его ядро



Ядро Cortex M3

Что общего у микроконтроллеров на таком ядре?

- Система команд и время их выполнения
- Одинаковая периферия ядра (с вариациями)
- Контроллер прерываний

Периферия уровня ядра описана в документе про ядро — cortex m3 user guide.

Периферия конкретного МК — в документе про конкретный МК (напри, stm32f10x reference manual)

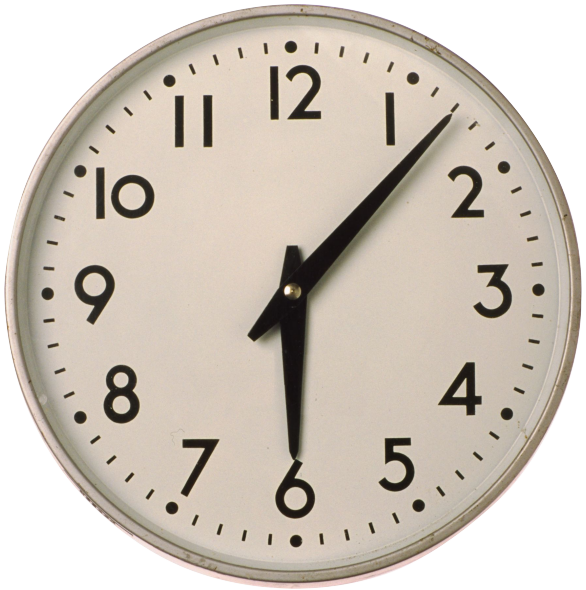
Так как же измерять время?

Мы хотим:

- Измерять время между событиями
- Отмерять интервалы времени

Так как же измерять время?

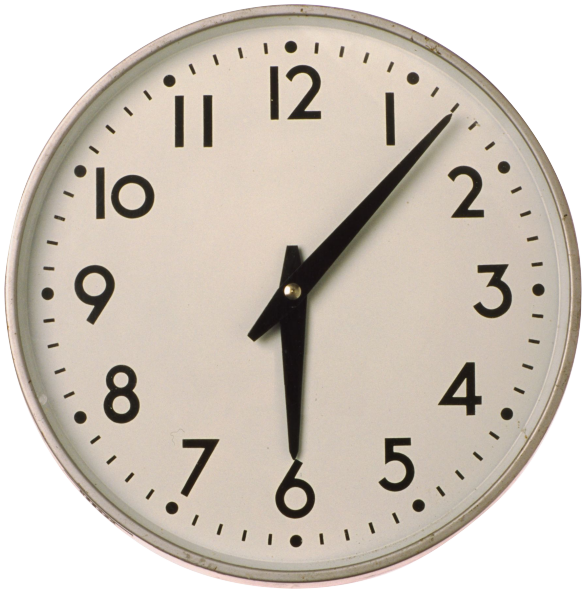
Допустим, у нас есть ЧАСЫ.
Как с их помощью отмерить интервал?



- Запомнить момент начала измерения
- Вычислить момент, соответствующий концу интервала
- Ждать

А как не пропустить момент?

Периодически посматривать на часы!



«Основные свойства часов»

- Значение часов изменяется каждую секунду на фиксированную величину.
- Когда часы досчитывают до 23:59:59, они начинают отсчет от 00:00:00.

Таймер SysTick

Простейший таймер, периферия ядра Cortex M3 (поэтому его описание нужно искать в документе Cortex M3 user guide, глава 4, стих 4)

Как вы думаете, что он из себя представляет?

- Это регистр-счетчик. В начальный момент времени он равен какому-то числу
- Каждый «тик» таймера счетчик уменьшается на 1
- После достижения нуля, счет начинается заново

Таймер SysTick — особенности

- Счетчик 24-битный (макс. 16 777 216)
- Частота тиков не регулируется и всегда равна частоте ядра
- Направление счета не меняется — всегда к нулю
- По достижению нуля счет начинается заново сразу же
- Прерывание всегда разрешено!
- Регистры SysTick'a в мануале и в коде называются по-разному

Таймер SysTick — использование

Настройка производиться через структуру SysTick. В ней есть четыре регистра:

- CTRL – управление (битовое поле)
- LOAD – начальное значение счета (число)
  - Чтобы получить 100 тиков – нужно записать 99
- VAL – текущее значение счетчика (число)
- CALIB – калибровочный регистр

Таймер SysTick — использование

1. SysTick->LOAD = 1000 – 1; Задаем начало отсчета, таймер сделает 1000 тиков.

2. **Помните, максимальное значение LOAD = 2^{24} = 16 777 216**

3. SysTick->CTRL = 1<<2 | 1<< 0; - Выбираем источник частоты и запускаем счет

4. Опрос регистра SysTick->VAL — текущее значение счетчика..?

Постоянный опрос регистра называется «поллинг» - polling

Таймер SysTick — использование

Почему поллинг регистра SysTick->VAL работает не всегда?

Потому что счетчик меняется с частотой работы процессора! Он может не успеть «заметить» момент, когда счетчик равен нулю!

Что же делать?

Проверять бит COUNTFLAG в регистре SysTick->CTRL.
Он не сбрасывается, когда счет начинается заново (сбрасывается при чтении).

Или использовать прерывание!

Но этот флаг выставляется в момент изменения значения счетчика с 1 на 0. Именно поэтому я раньше писал про 99 тиков при значении RELOAD=100.

Таймер SysTick — использование

Для настройки таймера SysTick фирма ARM предоставляет библиотечную функцию SysTick_Config.

Она принимает 1 параметр – значение Reload. Всю остальную настройку она в себе прячет.

Однако, она в том числе разрешает прерывание!
Если оно вам не нужно, его приходится запрещать самостоятельно.

Таймер SysTick — правильное использование

1. Магическая строчка __disable_irq(); - запрет прерываний
2. Вызов SysTick_Config — задание reload value и включение счета
3. Опрос флага COUNTFLAG в регистре SysTick->CTRL
   - Это 16-й бит; для значения (1<<16) заведен псевдоним SysTick_CTRL_COUNTFLAG_Msk

Постоянный опрос регистра называется «поллинг» - polling

Как вычислить нужный reload?

Текущая частота ядра хранится в переменной SystemCoreClock (в герцах).
Я хочу, чтобы SysTick досчитал до нуля за 1 миллисекунду.
Какой нужно задать reload value?

SystemCoreClock / reload = 1000 (Герц) =>

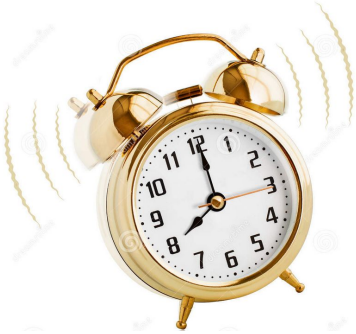
reload = SystemCoreClock / 1000

Помните, максимальное значение reload — 16,7 миллионов!

А что же такое прерывание?

Как засечь интервал времени с помощью часов?

- Постоянно смотреть на часы, не пора ли
- Завести будильник!



Прерывание — это как будильник. Оно произойдет в нужный момент «само».

А что же такое прерывание?

Прерывание (interrupt) — это аппаратный вызов функции по какому-то событию.
Такая функция называется «обработчик прерывания» (interrupt handler).

Вызывать ее программисту НЕ НУЖНО!
Ее вызовет контроллер прерываний, когда придет нужный момент.

Прерывания

Как вы думаете, какие бывают прерывания?

- Прерывания по исключительным ситуациям (исключения, exceptions) – например, деление на ноль, аппаратный отказ оперативной памяти и т.п.
- Прерывания от периферийных устройств – переполнение таймера, внешний сигнал на ножке, завершение работы АЦП и т.п.
- Программные прерывания (по приказу программиста)

Прерывания

В чем основное отличие функции-обработчика прерывания от обычной функции?
Ее вызов происходит аппаратно, в заранее неизвестный момент.

Какие параметры должны быть у функции-обработчика?
Никаких, ведь ее не вызывает программист.

А что она должна возвращать?
Ничего, неизвестно, куда возвращать.

Т.е. тип: void xxxx (void);

Прерывания

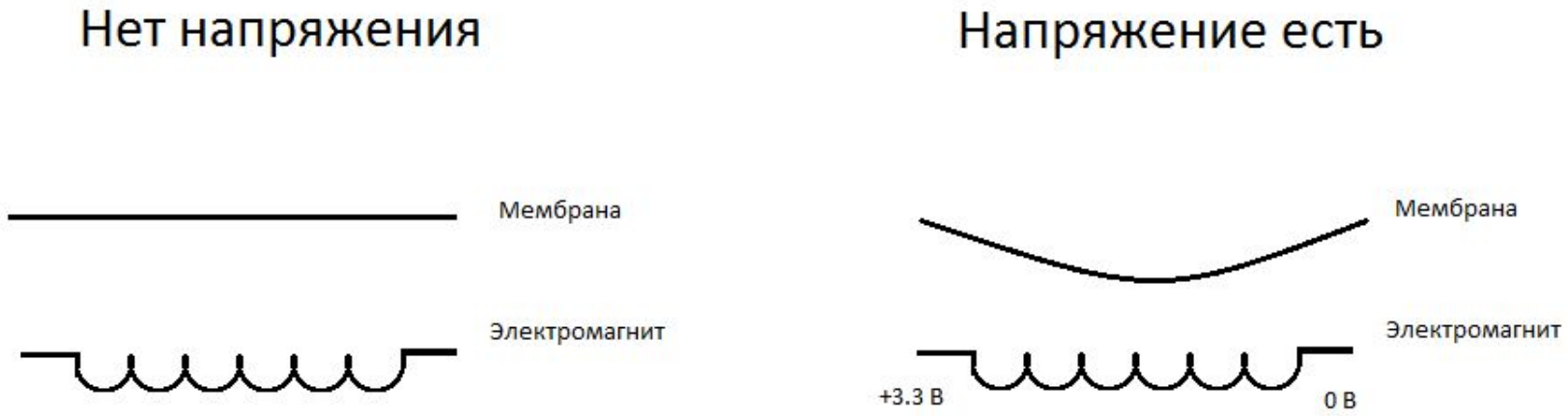
А что, если запретить все прерывания (вызвать __disable_irq())?
Прерывания не будут происходить.

А что, если разрешить прерывание, но не написать функцию-обработчик?
Вызовется обработчик по-умолчанию (из файла …startup.s), в котором просто бесконечный цикл.

Как использовать прерывания?

1. Написать функцию-обработчик (в Keil это функция с заранее определенным именем, например `SysTick_Handler`)
2. Запретить все прерывания - `__disable_irq();`
3. Настроить нужное прерывание
4. Разрешить все прерывания - `__enable_irq();`
5. Ждать, пока оно произойдет (можно делать что-то полезное, а не просто ждать)

Динамик



- При постоянном напряжении звука нет
- При **изменении** напряжения динамик издает щелчок
- При частоте выше ~50 Герц щелчки сливаются в единый звук
- Частота звука = частота изменения напряжения
- Диапазон человеческого слуха 20 Гц – 18 000 Гц