Прерывания
таймеры
ПДП=DMA
WDT

Прерывания **Interrupt**

- **Прерывание-** это временная приостановка текущей программы и переход к подпрограмме обработки прерывания. После выполнения запрограммированных действий подпрограмма возвращает управление обычно в прерванную программу (адрес возврата можно изменить). В отличие от обычных подпрограмм, программа обработки прерывания вызывается аппаратно контроллером прерывания. Прерывания позволяют с минимальной задержкой реагировать на различные события (EVENT).

Внутреннее Событие

- Внутренние- изменения состояния проц ядра (деление на 0, обращение к несуществующему адресу, ресет и т.д.) или какого-либо ФМ (конец счёта таймера, готовность генератора в RCC и т.п)

Внешние события

- Внешние события обычно связывают с внешними сигналами: изменение логического уровня на выводе МК, изменение уровня аналогового сигнала, принятие байта (кадра) UART, I2C, SPI и по др. интерфейсам. Внешние сигналы обрабатываются каким либо ФМ и вызывают изменение его состояния и обычно в отражается в регистре типа XXXX_ISR Interrupt and status register- конец оцифровки в АЦП, прием или окончание передачи байта в USART и т.п)

- **Каждое прерывание вызывается событием,**
- **но не каждое событие вызывает прерывание.**

Подпрограмма обработки прерывания handler, irq_handler

- *// Int Vectors*
- *// ВСЕГДА void (void)*
- *//Ничего не возвращает (нет аргументов)*
- **void ADC_IRQHandler(void)**
- **{**
- **}**

Приоритет прерываний

- Устанавливает важность прерывания. Прерывания более высокого приоритета могут прервыть подпрограмму низкого приоритета. Уровней приоритета от 0 до 255. Чем меньше уровень (число) , тем выше приоритет.
- У CM0 используется всего 2 уровня =1 старш бит: 0-254 и -255 , у CM1-4 старш бита =16 уровней
- 0- самый высокий приоритет
- 255- самый низкий приоритет
- Уровень прерывания необходимо установить перед разрешением прерывания (при настройке прерываний)

Вектор прерывания

- Вектор прерывания – ячейка памяти, содержащая адрес начала подпрограммы прерывания.
- Все вектора прерывания располагаются в начальных адресах , определено в ассемблерном файле startup_stm32fxxxx.s

```
__Vectors       DCD     __initial_sp               ; Top of Stack //начало  памяти
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     0         ; Reserved
  . . .
                DCD     PendSV_Handler             ; PendSV Handler
                DCD     SysTick_Handler            ; SysTick Handle
 ; External Interrupts
                DCD     WWDG_IRQHandler            ; Window Watchdog
                DCD     0                          ; Reserved
                DCD     RTC_IRQHandler             ; RTC through EXTI Line
                DCD     FLASH_IRQHandler           ; FLASH
                DCD     RCC_IRQHandler             ; RCC
. . . .
                DCD     ADC1_IRQHandler            ; ADC1
. . . .
                DCD     USART1_IRQHandler          ; USART1
                DCD     USART2_IRQHandler          ; USART2
__Vectors_End
```

- При одинаковом приоритете и одновременном событии играет роль место вектора(=номер вектора) в таблице прерываний

Контроллер прерываний

- Nested vectored interrupt controller **NVIC**- встроенный контроллер векторных прерываний- часть ядра ARM
- регистры

Регистры NVIC

- **ISER** — Interrupt Set Enable Register. Запись бита в нужную позицию включает прерывание.
- **ICER** — Interrupt Clr Enable Register. Запись сюда наоборот выключает прерывание.
- **ISPR** — Interrupt Set Pending Register. Поставить прерывание в ожидание.
- **ICPR** — Interrupt Clr Pending Register. Сбросить прерывание с ожидания.
- **IABR** — Interrupt active bit registers. Регистр показывающий активно ли в данный момент прерывание
- **IPR**-приоритеты

Основные ф-ии CMSIS

void NVIC_SetPriorityGrouping(uint32_t priority_grouping)
// Задать группы/подгруппы приоритетов

void NVIC_EnableIRQ(IRQn_t IRQn) *// Включить IRQn*

void NVIC_DisableIRQ(IRQn_t IRQn) *// Выключить IRQn*

uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn) *// Вернуть true (точнее IRQ-Number) если IRQn в ожидании*

void NVIC_SetPendingIRQ (IRQn_t IRQn) *// Поставить IRQn в ожидание*

void NVIC_ClearPendingIRQ (IRQn_tIRQn) *// Выкинуть из очереди на ожидание IRQn*

uint32_t NVIC_GetActive (IRQn_t IRQn) *// Функция "где это я?" Возвращает номер текущего активного прерывания если такое имеется*

void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority) *// Задать приоритет IRQn*
uint32_t NVIC_GetPriority (IRQn_t IRQn) *// Считать приоритет IRQn* void NVIC_SystemReset (void) *// Reset the system*

Макросы = функции CMSIS для разрешения/запрета прерываний

- глобальные

__enable_irq (); //разрешить разрешённые

__disable_irq(); // запретить все

- Локальные для каждого прерывания

void NVIC_EnableIRQ(IRQn_t IRQn) *// Enable IRQn*

void NVIC_DisableIRQ(IRQn_t IRQn) *// Disable IRQn*

IRQn- номер вектора прерывания в списке = таблице векторов прерываний в начале памяти

Для разрешения прерываний:

- Разрешить глобальные прерывания
- Разрешить нужное прерывание в NVIC
- Настроить и разрешить конкретные прерывания непосредственно в периферии- в регистрах ФМ. Обычно в регистрах с ,с буквами в имени IR, ISR, SIR, IER, DIER.

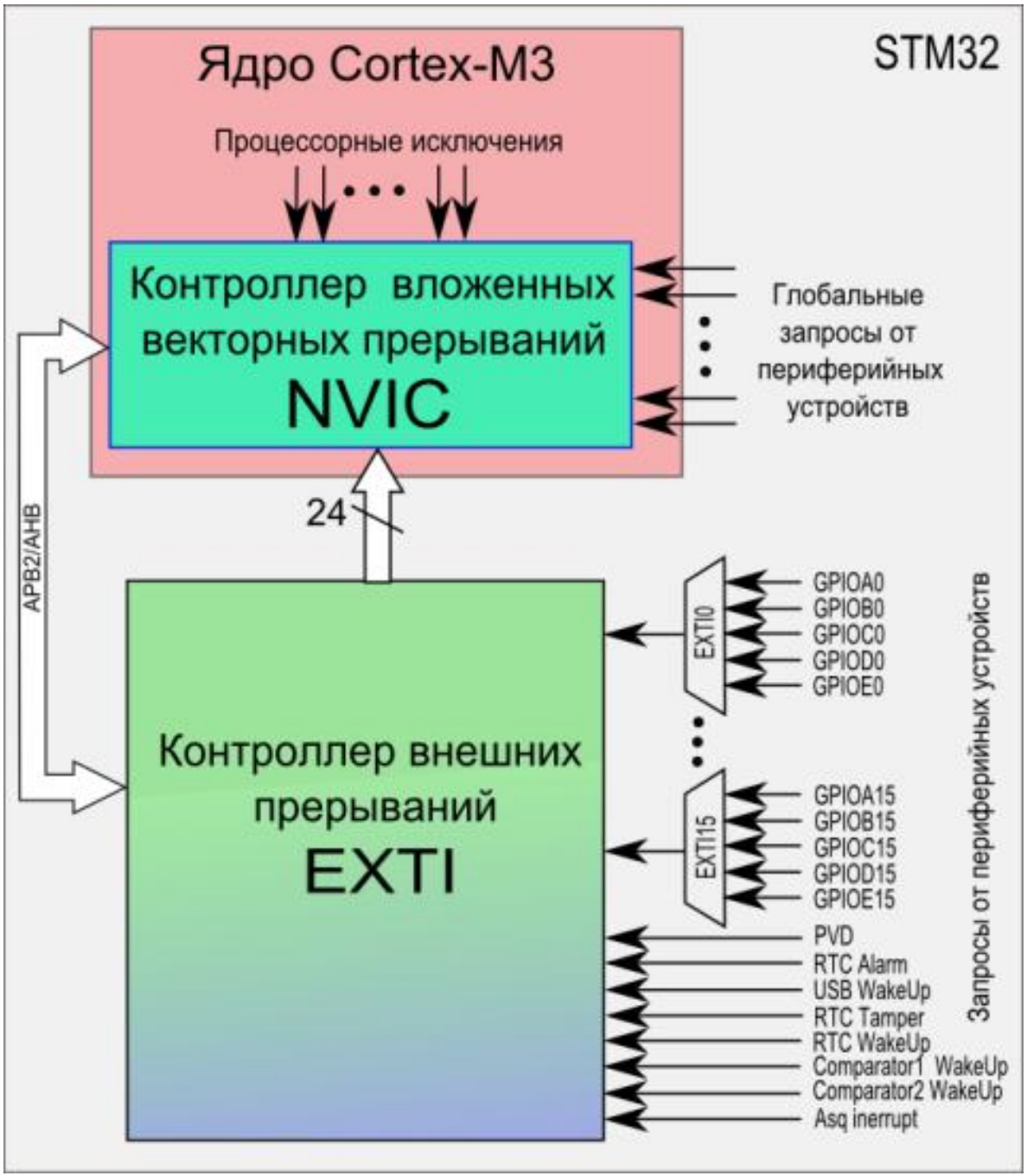
Модуль EXTI

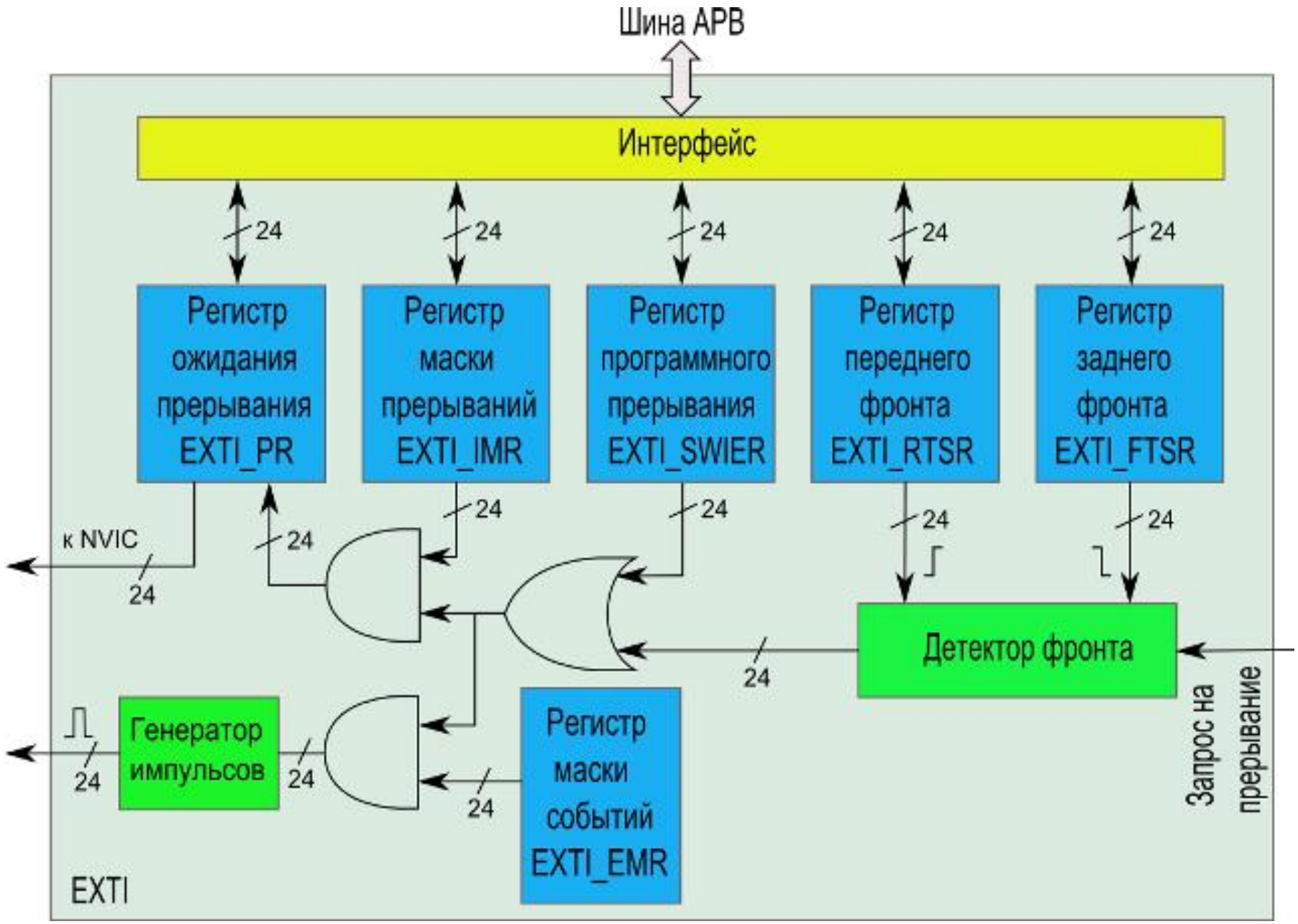
- Следит за логическими уровнями на выводах МК
- Может генерировать события и прерывания при смене уровней на выводах МК .
- 16 линий EXTI- сгруппированы по выводам GPIO
- 3 вектора прерываний:

EXTI0_1_IRQHandler ; EXTI Line 0 and 1

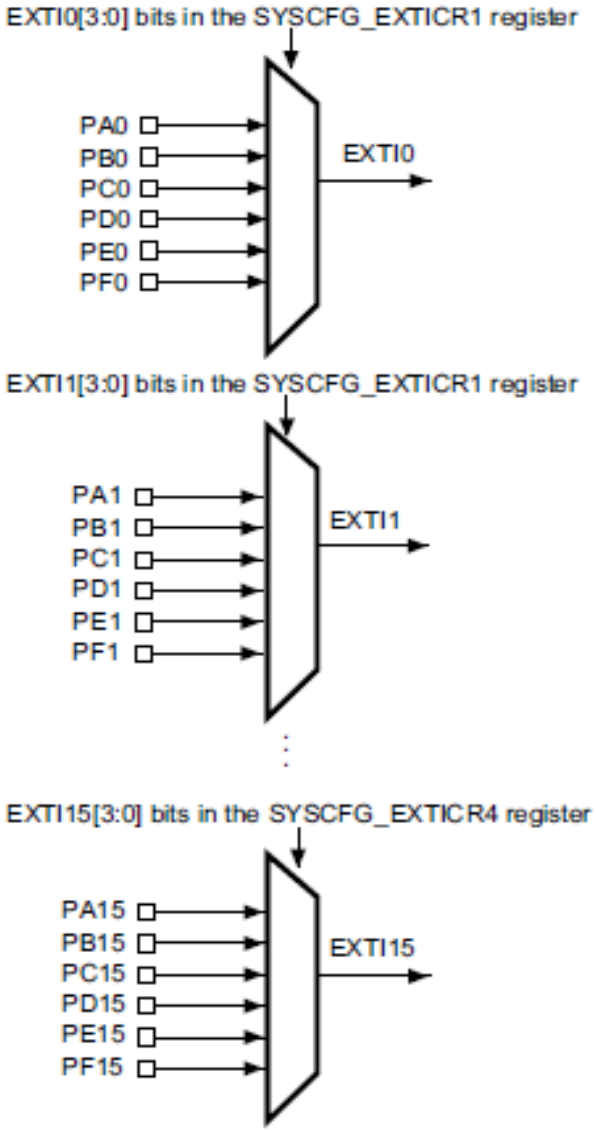
EXTI2_3_IRQHandler ; EXTI Line 2 and 3

EXTI4_15_IRQHandler ; EXTI Line 4 to 15


STM32
Ядро Cortex-M3
Процессорные исключения
Контроллер вложенных векторных прерываний
NVIC
Глобальные запросы от периферийных устройств
24
APB2/AHB
Контроллер внешних прерываний
EXTI
EXTI0
GPIOA0
GPIOB0
GPIOC0
GPIOD0
GPIOE0
EXTI15
GPIOA15
GPIOB15
GPIOC15
GPIOD15
GPIOE15
PVD
RTC Alarm
USB WakeUp
RTC Tamper
RTC WakeUp
Comparator1 WakeUp
Comparator2 WakeUp
Asq inerrupt
Запросы от периферийных устройств


Шина APB
Интерфейс
24
Регистр ожидания прерывания EXTI_PR
Регистр маски прерываний EXTI_IMR
Регистр программного прерывания EXTI_SWIER
Регистр переднего фронта EXTI_RTSR
Регистр заднего фронта EXTI_FTSR
к NVIC
Детектор фронта
Запрос на прерывание
Генератор импульсов
Регистр маски событий EXTI_EMR
EXTI

EXTI Линии 0-15



EXTI Линии 16-31

The remaining lines are connected as follow:

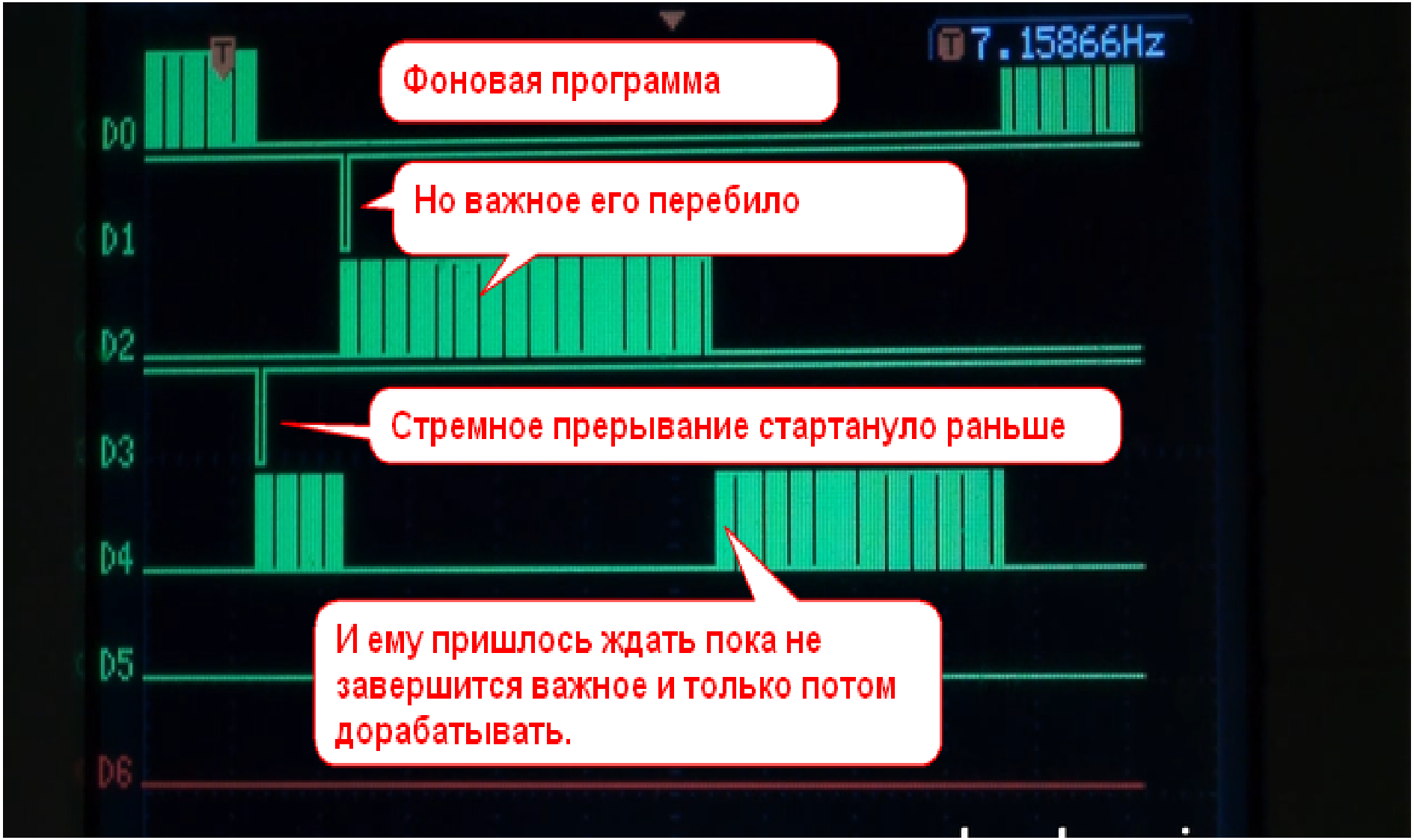
- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the internal USB wakeup event
- EXTI line 19 is connected to the RTC Tamper and TimeStamp events
- EXTI line 20 is connected to the RTC Wakeup event (available only on STM32F07x and STM32F09x devices)
- EXTI line 21 is connected to the Comparator 1 output
- EXTI line 22 is connected to the Comparator 2 output
- EXTI line 23 is connected to the internal I2C1 wakeup event
- EXTI line 24 is reserved (internally held low)
- EXTI line 25 is connected to the internal USART1 wakeup event
- EXTI line 26 is connected to the internal USART2 wakeup event (available only on STM32F07x and STM32F09x devices)
- EXTI line 27 is connected to the internal CEC wakeup event
- EXTI line 28 is connected to the internal USART3 wakeup event (available only on STM32F09x devices)
- EXTI line 29 is reserved (internally held low)
- EXTI line 30 is reserved (internally held low)
- EXTI line 31 is connected to the V_{DDIO2} supply comparator output (available only on STM32F04x, STM32F07x and STM32F09x devices

Регистры EXTI

- **Interrupt mask register (EXTI_IMR)**
- **Event mask register (EXTI_EMR)**
- **Rising trigger selection register (EXTI_RTSR)**
- **Falling trigger selection register (EXTI_FTSR)**
- **Software interrupt event register (EXTI_SWIER)**
- **Pending register (EXTI_PR)**

Равные приоритеты

Разные приоритеты



Таймеры

- Таймер- это цифровой счётчик. Разрядность 8, 16, 24, 32 бит, + предделитель+ регистры сравнения+ регистр автозагрузки+
- Считает тактовые импульсы от RCC или со входа МК (определяется управляющими регистрами).
- Разрешать счёт/сбрасывать можно тактовыми импульсами от RCC или со входа
- Результат счёта- событие /и/ прерывание при переполнении (переход через 0 или макс значение)
- Выход счётчика можно вывести на вывод МК
- Направление счёта (увеличение/уменьшение) и режим счёта определяется управляющими регистрами
- Количество таймеров от TIM1 до TIM17 с разными возможностями и разрядностью зависит от модели МК

Основные возможности

- Подсчёт импульсов,
- определение периода/частоты входного сигнала :

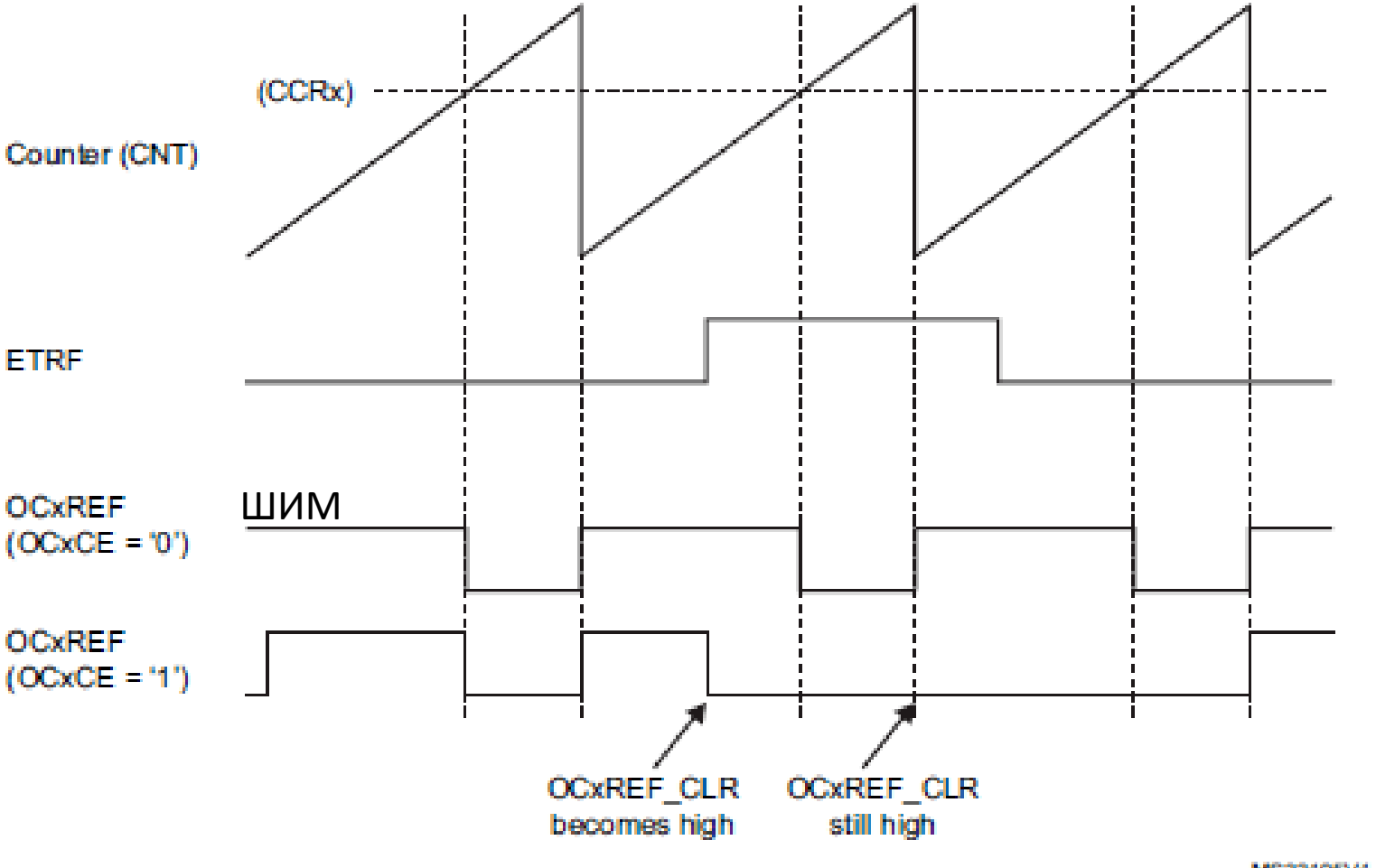
а)вход счётчика- с RCC- (t=1/f известно), «ворота» – вх сигнала (низк частота)

T=N*t; F=1/T

b) счёт с вх сигнала (выс. частота), «ворота» с RCC (t=1/f известно) T=t/N; F=1/T

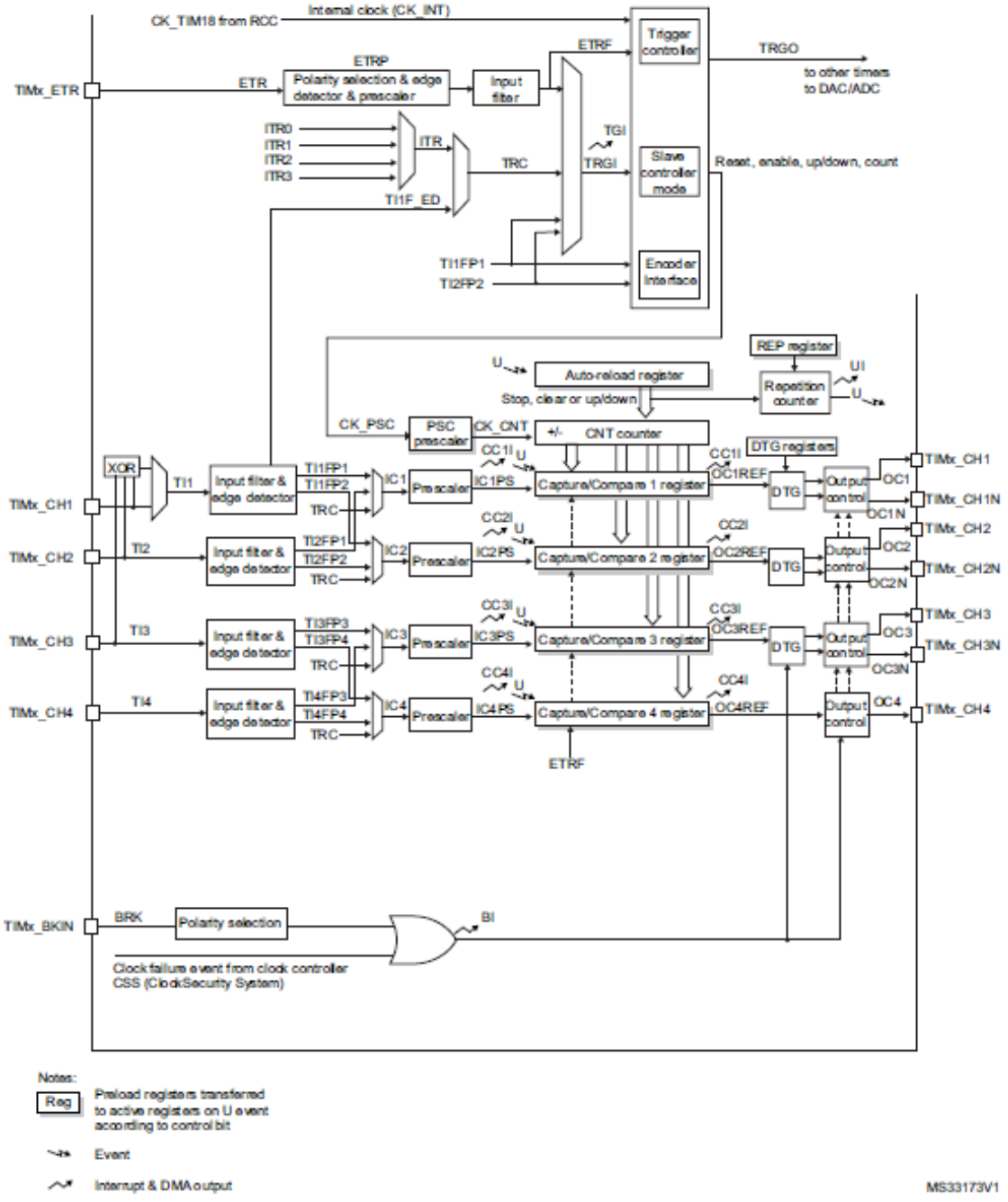
- Задание времени, периода выходной частоты T=N*t   (N- макс значение счёта)
- ШИМ (PWM) модуляция: управлять мощностью нагревателей, двигателей, меняя CCRx регистр

ШИМ


(CCRx)
Counter (CNT)
ETRF
ШИМ
OCxREF
(OCxCE = '0')
OCxREF
(OCxCE = '1')
OCxREF_CLR
becomes high
OCxREF_CLR
still high
MS33105V1

- Структура

TIM1



регистры

- CNT- регистр счётчика
- TIM1_PSC **p**re**sc**aler =коэфф. предварительного деления
- ARR- autoReloadRegister- макс значение
- CCRx (x=1..4)- регистры сравнения
- REPr- счётчик повторений
- DTG- регистр «мёртвого» интервала
- CR1, CR2 –управляющие
- TIM1_SMCR slave mode control register
- TIM1_DIER DMA/interrupt enable register
- TIM1_SR status register
- TIM1_EGR event generation register
- другие

SYSTICK таймер

- Особый таймер, который включён в ядро ARM (другие таймеры- как отдельные модули), поэтому стандартный для всех ARM МК (одинаковые регистры у всех производителей)
- Предназначен для отсчёта времени для системных событий.
- Разрядность повышенная 24 бита (16 бит =65535, 24 бит=16776960)
- Структура намного проще и возможностей (режимов работы) меньше, чем у универсальных таймеров

ФМ DMA== ПДП

- DMA- direct memory access- прямой доступ к памяти
- Модуль МК, позволяющий без процессорного ядра копировать данные:

- память-память
- регистр ФМ-> память
- память -> регистр ФМ

- Всего в DMA может быть одновременно до 12 каналов передачи данных (7шт в DMA1 и 5 в DMA2)

В регистрах DMA для каждого канала:

- Адрес источника
- Режим источника (длина 8,16, 32, автоинкремент/декремент адреса есть/нет)
- Адрес приёмника
- Режим приёмника (длина 8,16, 32, автоинкремент/декремент адреса есть/нет)
- Количество пересылок данных (после пересылки уменьшается на 1 до нуля) Ноль= конец блока.

- автоинкремент/декремент увеличение/уменьшение адреса на N после пересылки одного элемента данных, где N –длина элемента в байтах
- После пересылки блока данных DMA может создавать событие/прерывание.

ФМ WDT, WWDG, IWDG

Модуль «сторожевого таймера» предотвращает зависание процессора. Содержит счётчик, отсчитывающий время (количество периодов собственного независимого генератора)

Процессор (программа) должен периодически читать (или писать требуемое значение в) регистр WDT. Если в течение определённого времени (от неск мс до десятков секунд) опроса не произошло, модуль WDT производит аппаратный сброс (reset) процессора и процессор с начала начинает выполнять программу из ПЗУ.

IWDG independent WDT- полностью независимый модуль, тактирование от LSI 40 кГц, счётчик 12 бит, предделитель 8 бит. Работает в SLEEP STOP режимах ядра.

WWDG- window WDT- задаётся окно времени для подтверждения нормальной работы. При меньшем или большем- reset. Тактирование от PCLK (модуля RCC)

Модуль RTC

- Real Time Clock- Модуль часов – содержит счётчики времени= регистры секунд, минут, дней недели, месяцев, лет.
- Регистры будильников и схемы сравнения с реальным временем. При равенстве- происходит событие/прерывание, пробуждение ядра из режима сна.
- Тактирование от LSE генератора 32767 Гц модуля RCC
- Входит в состав батарейного домена, Питание которого при пропадании основного питания переключается на батарейку.