Программирование микроконтроллеров STM32

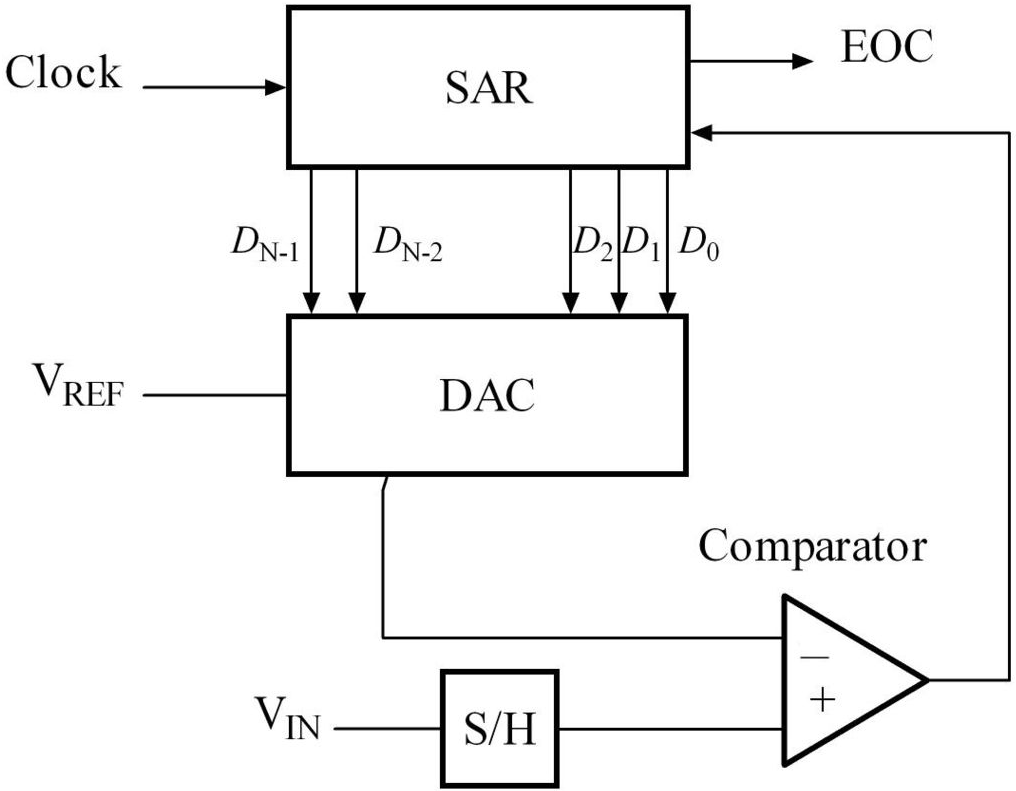
ADC & DAC

МФТИ, 2019

Введение

- АЦП используется для преобразование аналогового сигнала в цифровой код (микрофон, матрица камер и т.д.)
- Основные параметры
  - Скорость (кол-во семплов в секунду)
  - Разрешение (разрядность)
- Типы
  - АЦП прямого счета (послед. счета, сигма-дельта, двойн. интегр.)
  - АЦП поразрядного уравновешивания (successive approximation) (в STM32)
  - Параллельные АЦП (компараторные, конвейерные)

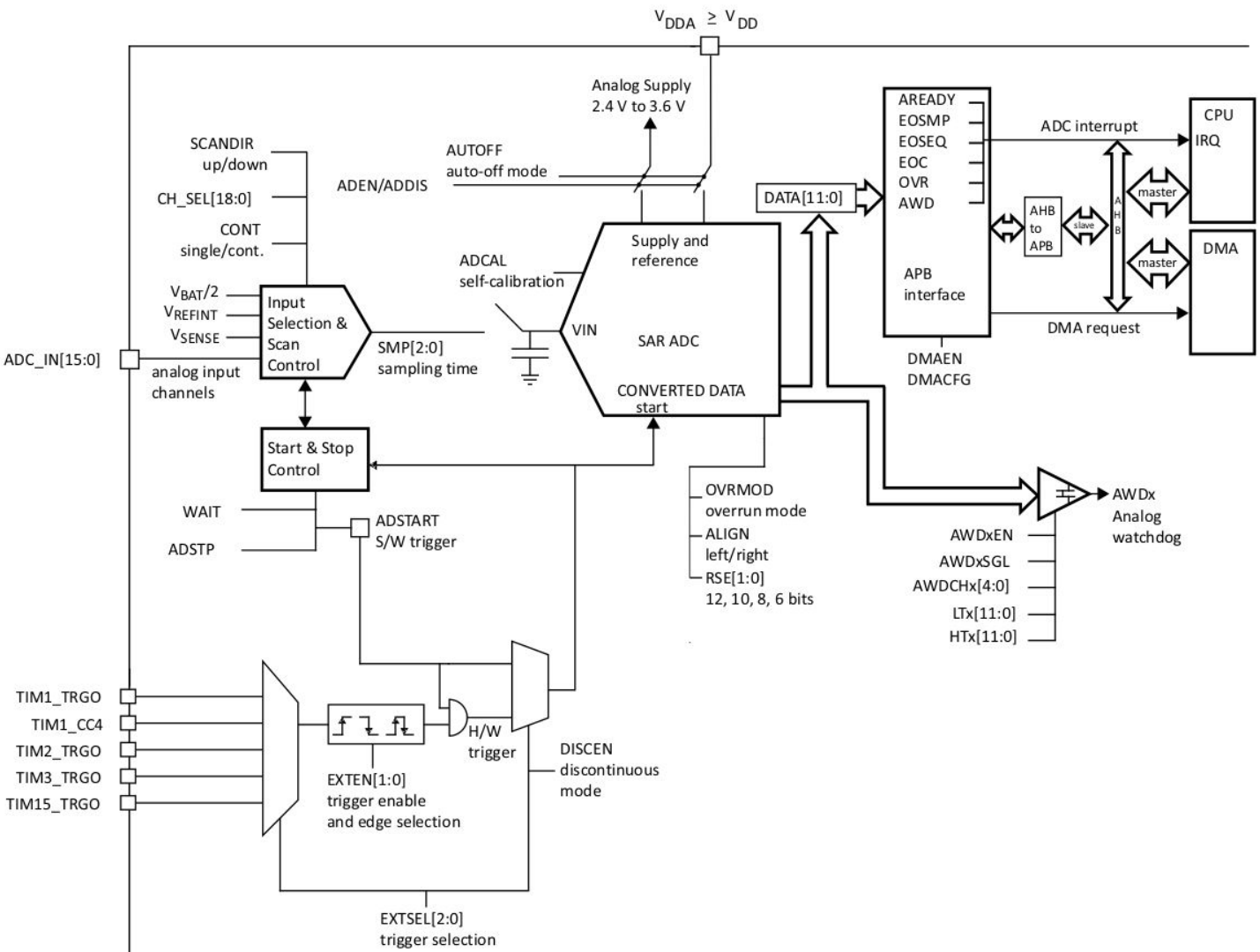
АЦП последовательного преобразования



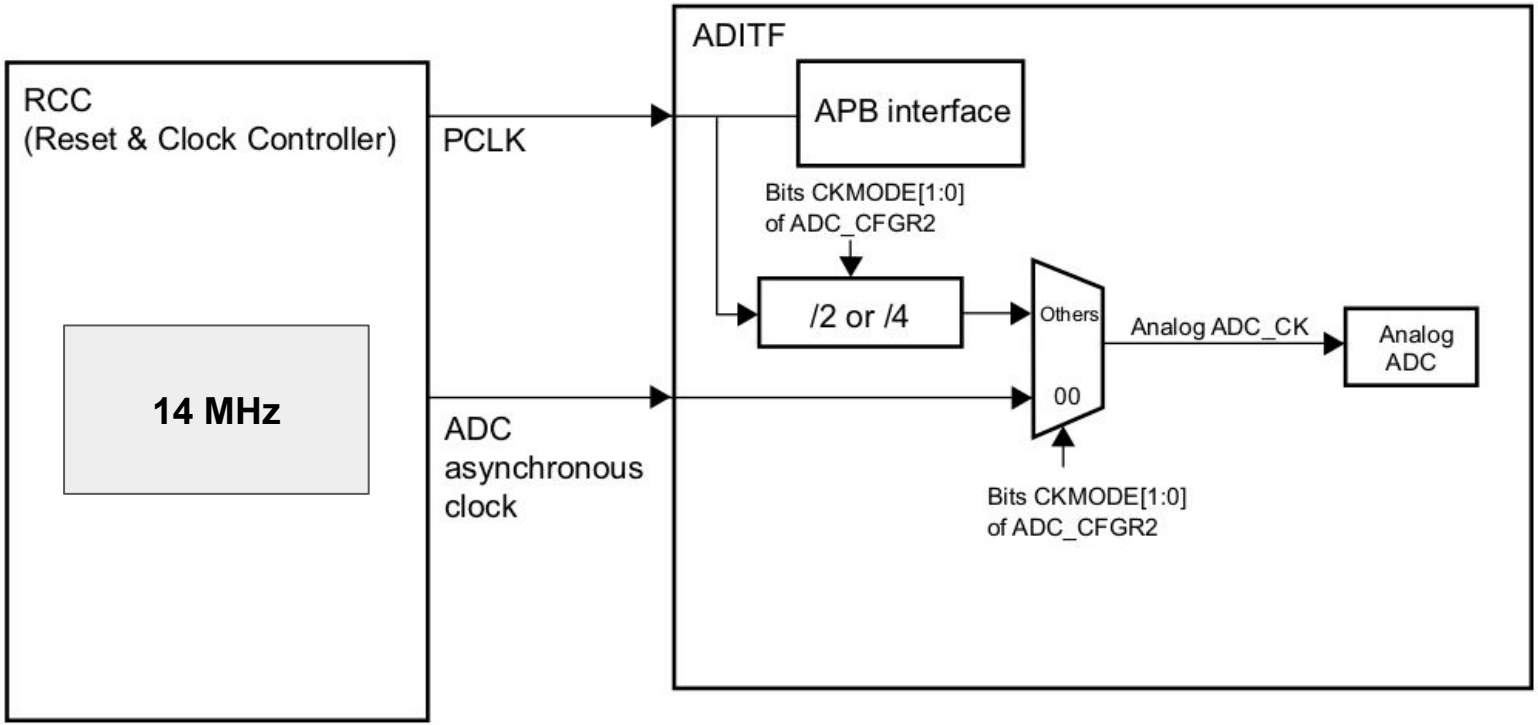
Credits: wikipedia

АЦП в STM32

- Конфигурируемая разрядность (12, 10, 8 или 6 бит)
- Калибровка
- Поддержка DMA
- 16 внешних каналов (ADC_INx)
- 3 внутренних
  - Температурный сенсор (ADC_IN16)
  - Опорное напряжение (Vrefint, ADC_IN17)
  - Напряжение батареи (Vbat, ADC_IN18)
- Поддержка связки с таймерами
- Поддержка прерываний


$V_{DDA} \geq V_{DD}$
Analog Supply
2.4 V to 3.6 V
SCANDIR
up/down
CH_SEL[18:0]
CONT
single/cont.
ADEN/ADDIS
AUTOFF
auto-off mode
ADCAL
self-calibration
Supply and
reference
$V_{BAT}/2$
V_{REFINT}
V_{SENSE}
Input
Selection &
Scan
Control
VIN
SAR ADC
SMP[2:0]
sampling time
ADC_IN[15:0]
analog input
channels
CONVERTED DATA
start
DATA[11:0]
AREADY
EOSMP
EOSEQ
EOC
OVR
AWD
APB
interface
ADC interrupt
CPU
IRQ
AHB
to
APB
slave
AHB
master
master
DMA
DMA request
DMAEN
DMACFG
Start & Stop
Control
WAIT
ADSTP
ADSTART
S/W trigger
OVRMOD
overrun mode
ALIGN
left/right
RSE[1:0]
12, 10, 8, 6 bits
AWDx
Analog
watchdog
AWDxEN
AWDxSGL
AWDCHx[4:0]
LTx[11:0]
HTx[11:0]
TIM1_TRGO
TIM1_CC4
TIM2_TRGO
TIM3_TRGO
TIM15_TRGO
H/W
trigger
DISCEN
discontinuous
mode
EXTEN[1:0]
trigger enable
and edge selection
EXTSEL[2:0]
trigger selection

Источник тактирования ADC



Калибровка АЦП

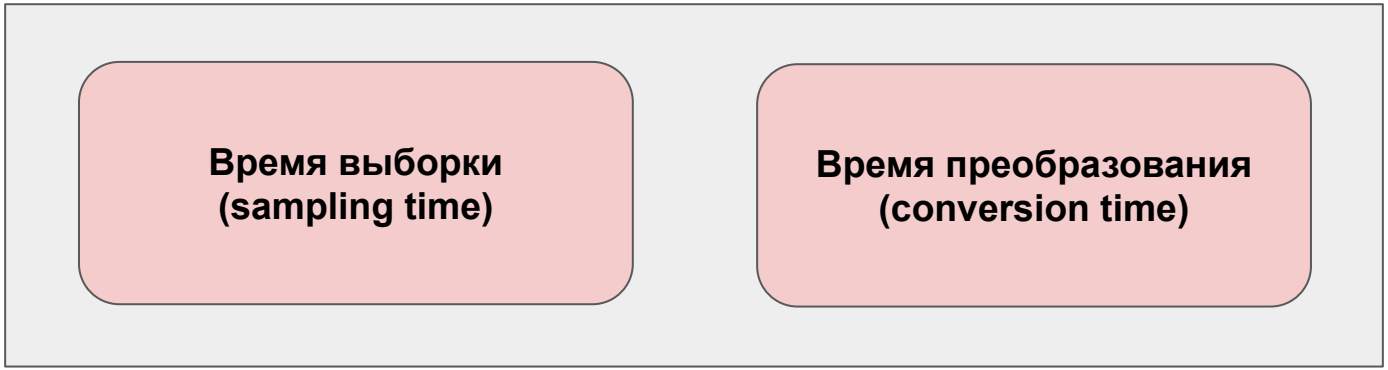
- Калибровка производится перед началом измерений
- Необходима для удаления погрешности смещения

Алгоритм калибровки:

1. Выключение ADC
2. Ожидание готовности выключения ADC
3. LL_ADC_StartCalibration() [Включение флага ADCAL]
4. Проверка LL_ADC_IsCalibrationOnGoing [проверка флага ADCAL]
5. Чтение калибровочного фактора из ADC_DR
6. Включение ADC + ожидание готовности

Время получения одного семпла

LL_ADC_SetSamplingTimeCommonChannels() [SMPR]



Время выборки задается в зависимости от выходного сопротивления источника сигнала

Время, необходимое для отработки алгоритма последовательного приближения (12.5 ADC тактов для 12 бит)

Выравнивание данных

- Данные могут быть выровнены слева и справа (*LL_ADC_SetDataAlignment*)
  - LSB - нулевой бит ADC_DR
  - MSB - старший бит ADC_DR
- Разрешение АЦП - 12, 10, 8 и 6 бит (*LL_ADC_SetResolution*)

Режимы работы ADC

LL_ADC_REG_SetContinuousMode(Single or Continuous)

LL_ADC_REG_SetSequencerDiscont()

Режим однократного преобразования (single conversion mode)	Режим постоянного преобразования (continuous conversion mode)	Режим с прерываниями (discontinuous mode)
Одиночное преобразование заданной последовательности каналов (по аппаратному или программному триггерам) **1**. 1-ый канал -> EOC … n. N-ый канал -> EOC EOSEQ	Постоянное преобразование заданной последовательности каналов (по аппаратному или программному триггерам) **1**. 1-ый канал -> EOC … n. N-ый канал -> EOC EOSEQ	Одиночное преобразование канала (по аппаратному или программному триггерам) **1**. 1-ый канал -> EOC … **n**. N-ый канал -> EOC EOSEQ

Диаграмма работы

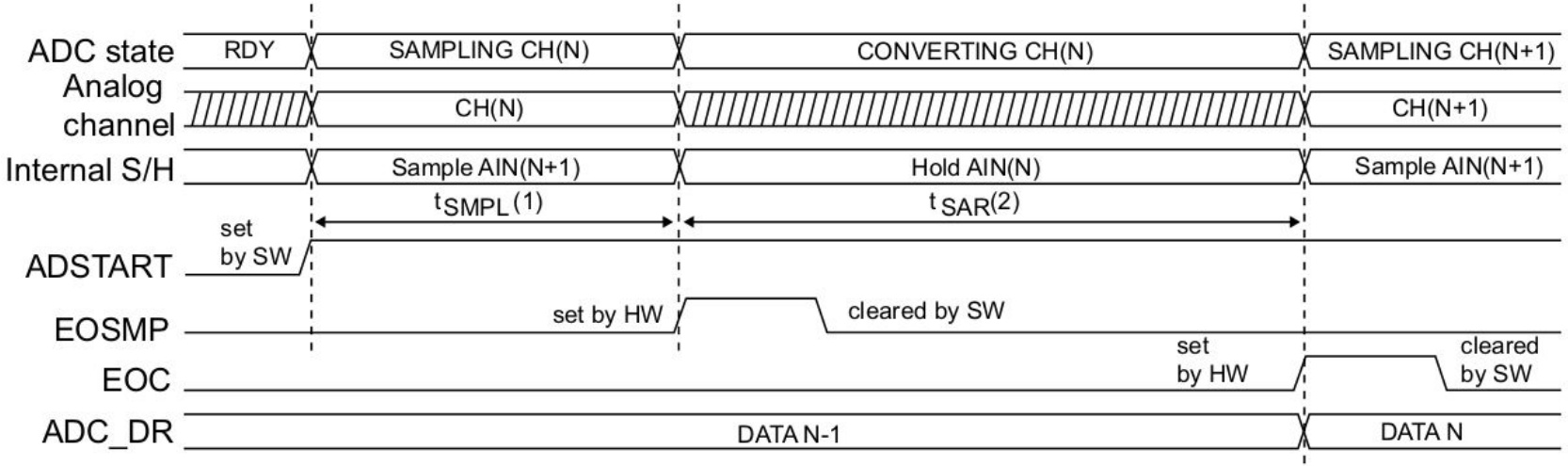


Диаграмма работы (single conv. mode)

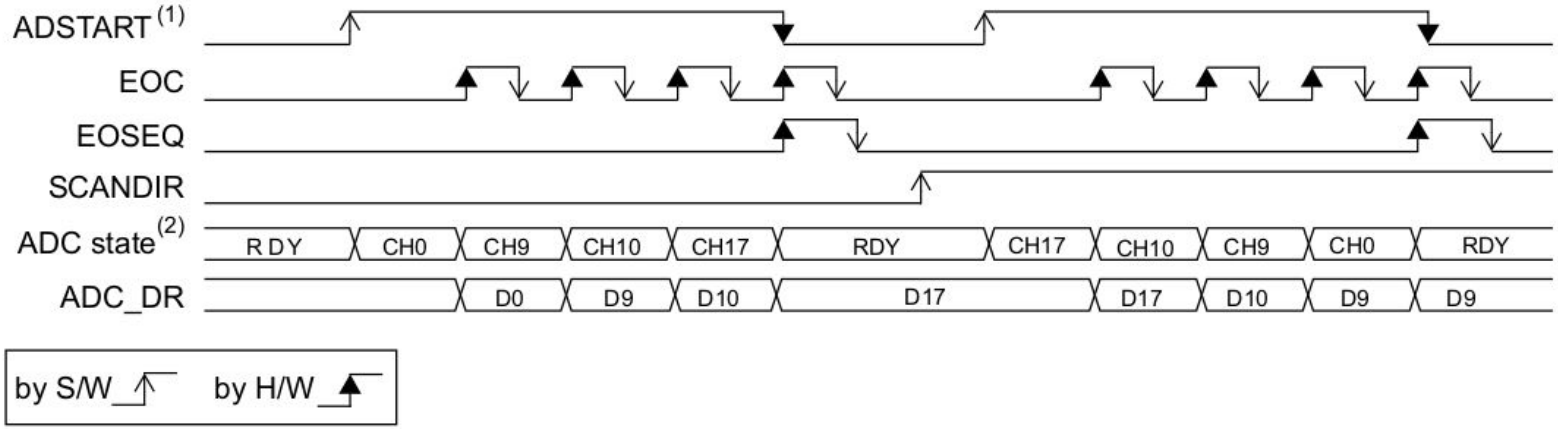
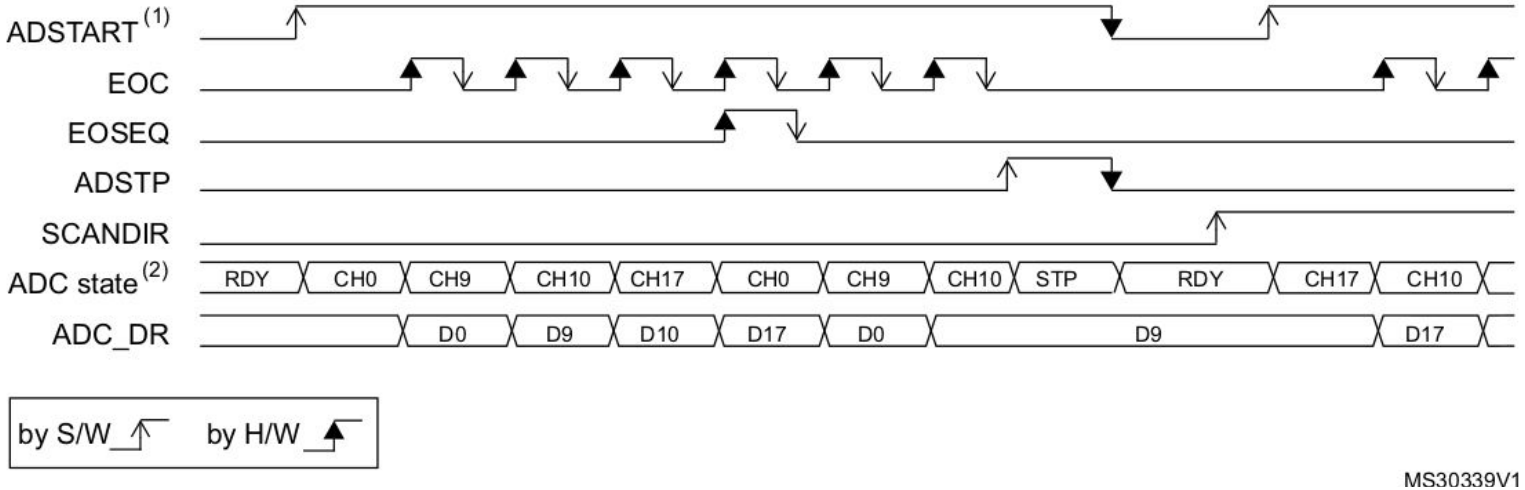


Диаграмма работы (cont. conv. mode)



Использование DMA

- Данные могут быть потеряны из-за низкой скорости чтения
- Высокая скорость чтения может привести к большой загрузке CPU
- DMA позволяет существенно снизить нагрузку на CPU и не теряет данные

Для настройки DMA+ADC необходимо

1. Настроить DMA и в качестве адреса (PeriphAddress) указать ADC->DR
2. Включить DMA в конф. регистрах ADC (LL_ADC_REG_SetDMATransfer)
3. По окончанию передачи данные могут быть прочитаны в заданном последовательностью порядке (ch0, ch1, ch3, ch0, ch1, ch3 и т.д.)

Vref и температурный сенсор

- Напряжение Vdda может варьироваться
- В STM32 стоит точный источник опорного напряжения Vref, который может использоваться для определения Vdda

$$V_{DDA} = 3.3\ V \times \mathbf{V_{REFINT\_CAL}} / V_{REFINT\_DATA}$$

- Для расчета значения температурного сенсора:

$$\text{Temperature (in °C)} = \frac{110\ °C - 30\ °C}{TS\_CAL2 - TS\_CAL1} \times (TS\_DATA - TS\_CAL1) + 30\ °C$$

Пример инициализации

Задача:

1. Чтение двух каналов - Vref и TempSensor
2. Усреднение по 8 выборкам
3. Расчет текущего Vdda по значению Vref
4. Корректировка семпла с температурного сенсора
5. Расчет текущей температуры чипа

Решение:

ADC+DMA (обработка будет происходить в прерывании DMA по TC)

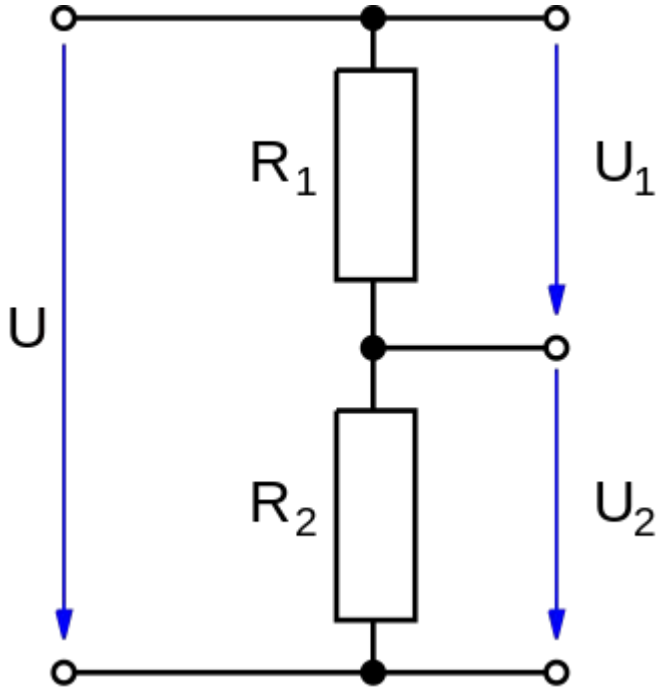
Пример инициализации

1. Включение тактирования
2. Включение источника на 14 МГц
3. Калибровка ADC
4. Включение ADC
5. *LL_ADC_SetResolution(ADC1, LL_ADC_RESOLUTION_12B)*
6. *LL_ADC_SetDataAlignment(ADC1, LL_ADC_DATA_ALIGN_RIGHT)*
7. *LL_ADC_SetSamplingTimeCommonChannels(ADC1, LL_ADC_SAMPLINGTIME_239CYCLES_5);*
8. *LL_ADC_REG_SetTriggerSource(ADC1, LL_ADC_REG_TRIG_SOFTWARE)*
9. *LL_ADC_REG_SetSequencerChannels(ADC1, LL_ADC_CHANNEL_TEMPSENSOR | LL_ADC_CHANNEL_VREFINT)*

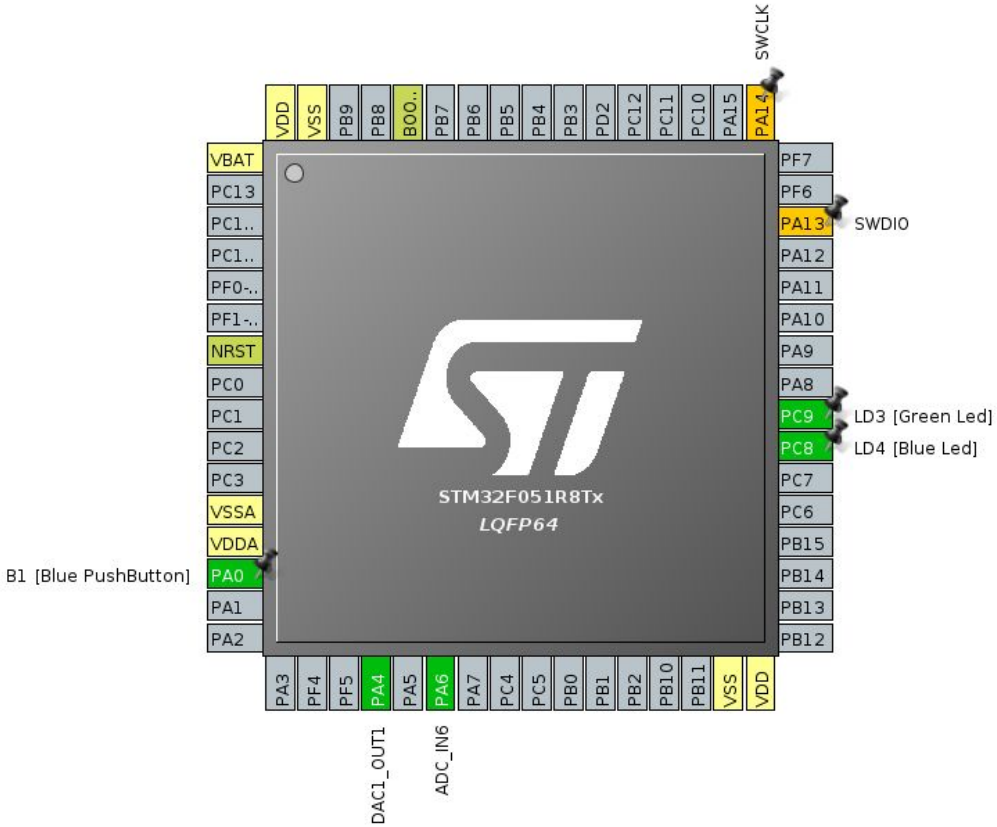
Пример инициализации

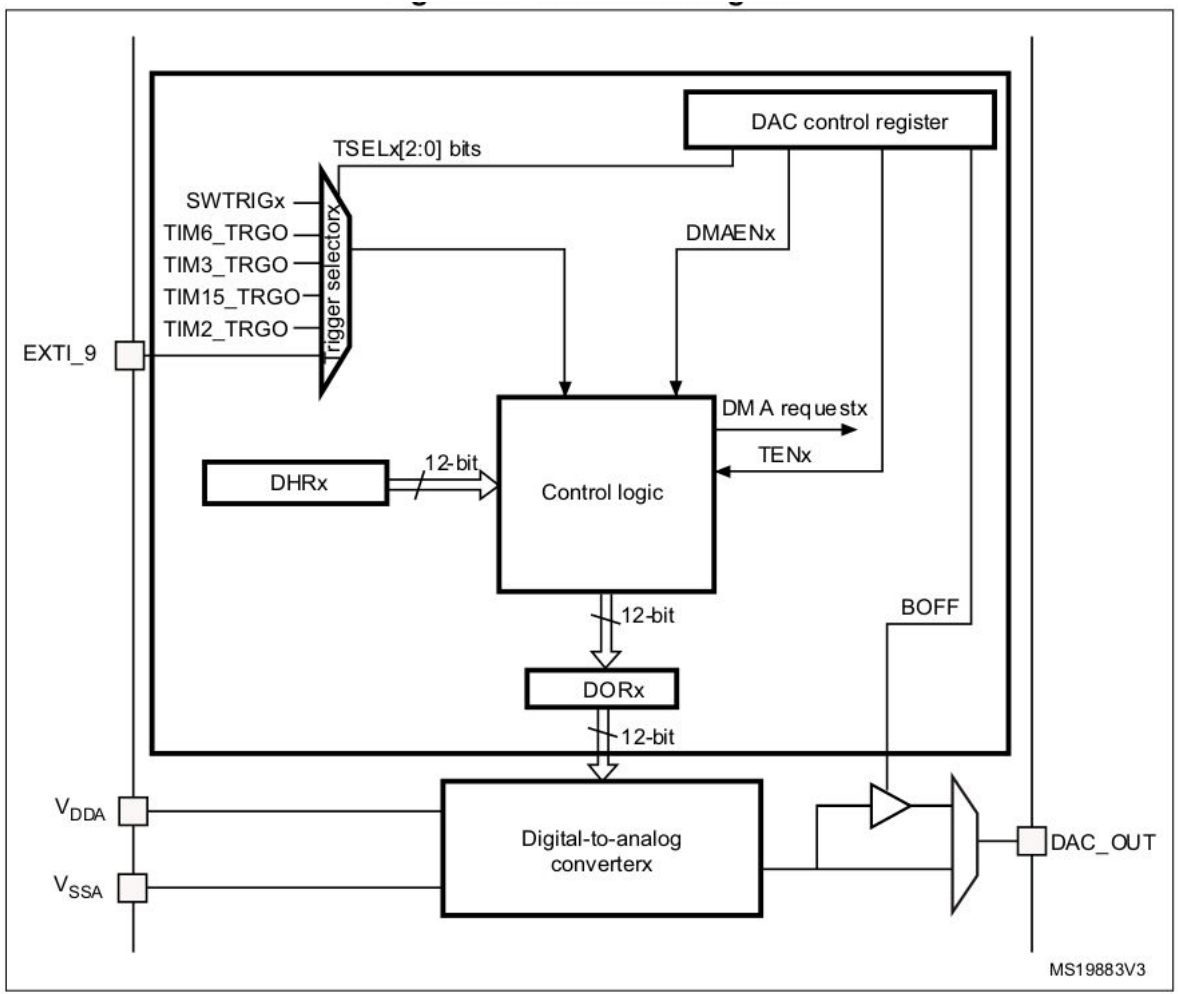
1. *LL_ADC_REG_SetContinuousMode(ADC1, LL_ADC_REG_CONV_CONTINUOUS)*
2. *LL_ADC_REG_SetDMATransfer(ADC1, LL_ADC_REG_DMA_TRANSFER_UNLIMITED)*
3. *LL_ADC_REG_SetOverrun(ADC1, LL_ADC_REG_OVR_DATA_PRESERVED)*
4. Настройка DMA + включение прерывания на канал 1 по TC
5. Настройка NVIC
6. *LL_ADC_SetCommonPathInternalCh(ADC, LL_ADC_PATH_INTERNAL_TEMPSENSOR | LL_ADC_PATH_INTERNAL_VREFINT)*
7. *LL_ADC_REG_StartConversion(ADC1)*

Делитель напряжения



Цифро-аналоговый преобразователь




DAC control register
TSELx[2:0] bits
SWTRIGx
TIM6_TRGO
TIM3_TRGO
TIM15_TRGO
TIM2_TRGO
trigger selectorx
EXTI_9
DMAENx
DMA requestx
TENx
DHRx
12-bit
Control logic
12-bit
BOFF
DORx
12-bit
VDDA
VSSA
Digital-to-analog converterx
DAC_OUT
MS19883V3

Настройка работы

- Настройка таймера (с включенным триггером) *LL_TIM_SetTriggerOutput*
- Настройка PA4 (аналоговый режим)
- Подача тактирования на модуль DAC
- *LL_DAC_EnableTrigger(DAC1, LL_DAC_CHANNEL_1)* [CR]
- *LL_DAC_SetTriggerSource* [CR]
- *LL_DAC_SetOutputBuffer* [CR]
- *LL_DAC_Enable* [CR]
- *LL_DAC_ConvertData12RightAligned* [DHR12R1]

255 / 4096 * 3V = 0.188V

Репозиторий

https://github.com/edosedgar/stm32f0_ARM