Analog and digital peripherals

nRF52 Global Tech Tour

Outline

- **Peripheral use**
- Timers and RTC
- TWI
- UART
- SPI
- Comparators
- ADC
- Demo

How to START a peripheral

- CONFIGURE → ENABLE → START
  - HW design is based on this principle
  - Avoid glitches or strange behavior on pins
  - Exceptions:
    - Peripherals w/o START task (or similar)
    - If abnormal behavior on pins is accepted

How to STOP a peripheral

- STOP → DISABLE (→ RECONFIGURE)
  - HW design is based on this principle
  - Avoid glitches or strange behavior on pins
  - Ensure lowest power consumption
  - Exceptions:
    - Peripherals w/o STOP task (or similar)
    - Configure on-the-fly (e.g. double-buffered memory pointers)
    - If abnormal behavior on pins is accepted

Outline

- Peripheral use
- **Timers and RTC**
- TWI
- UART
- SPI
- Comparators
- ADC
- Demo

Timers

Three type of timers:

- RTC
- TIMER
- M4 sys-tick timer

RTC

- Running at 32kHz.
- Pre-scale by dividing clock by (n + 1) where n is 0-(2^{12}-1)
- Internal RC or XO as source. XO is optional, but will require external components and use two GPIO's.
- RC with 250PPM accuracy with automatic calibration
- XO accuracy dictated by external crystal

RTC timers (32kHz or lower)

RTC#	Parameter	nRF51	nRF52
RTC0	Compare registers	3	3
	Bit width	24	24
RTC1	Compare registers	4	4
	Bit width	24	24
RTC2	Compare registers	-	4
	Bit width	-	24

RTC Tasks and Events

TASK/EVENT	Description
TASKS_START	Start RTC COUNTER
TASKS_STOP	Stop RTC COUNTER
TASKS_CLEAR	Clear RTC COUNTER
TASKS_TRIGOVRFLW	Set COUNTER to 0xFFFFF0
EVENTS_TICK	Event on COUNTER increment
EVENTS_OVRFLW	Event on COUNTER overflow
EVENTS_COMPARE[0-3]	Compare event on CC[0-3] match

RTC timers does not use SHORT's

TIMER

- Running at 16MHz or 1MHz.
- Pre-scale by dividing clock by 2^n where n is 0-9
- Clock source is free running or locked PLL clock, dictated by system state
- Accuracy dictated by system clock
- Can be used as timers or counters

TIMER timers (16MHz or lower)

Timer#	Parameter	nRF51	nRF52
TIMER0	Capture/Compare registers	4	4
	Bit width	8/16/24/32	8/16/24/32
TIMER1	Capture/Compare registers	4	4
	Bit width	8/16	8/16/24/32
TIMER2	Capture/Compare registers	4	4
	Bit width	8/16	8/16/24/32
TIMER3	Capture/Compare registers	-	6
	Bit width	-	8/16/24/32
TIMER4	Capture/Compare registers	-	6
	Bit width	-	8/16/24/32

TIMER Tasks and Events

TASK/EVENT	Description
TASKS_START	Start TIMER
TASKS_STOP	Stop TIMER, but keep timer value Not recommended , will prevent system on idle mode Use TASK_SHUTDOWN instead
TASKS_COUNT	Increment timer by 1 when in counter mode
TASKS_CLEAR	Clear TIMER
TASKS_CAPTURE[0-5]	Capture task for each CC[0-5]
TASKS_SHUTDOWN	Different behaviours in Timer, Counter or Low power counter mode
EVENTS_COMPARE[0-5]	Compare event on CC[0-5] match

TIMER Shortcuts

Shortcut	Description
COMPARE*n*_CLEAR	Clear of timer on CC[*n*] match
COMPARE*n*_STOP	Stop timer on CC[*n*] match

M4 sys-tick timer

- Not recommended for general use as it stops when the CPU stops
- Can be used for performance analysis as well as by some RTOS for timing control

Outline

- Peripheral use
- Timers and RTC
- **TWI**
- **UART**
- **SPI**
- Comparators
- ADC
- Demo

Serial communication instances

- 4 serial communication instances: UART and SERIAL0, 1 and 2
- One function can be enabled at a time for each instance
- Supports nRF51 legacy serial (code-compatible, not recommended for new designs)

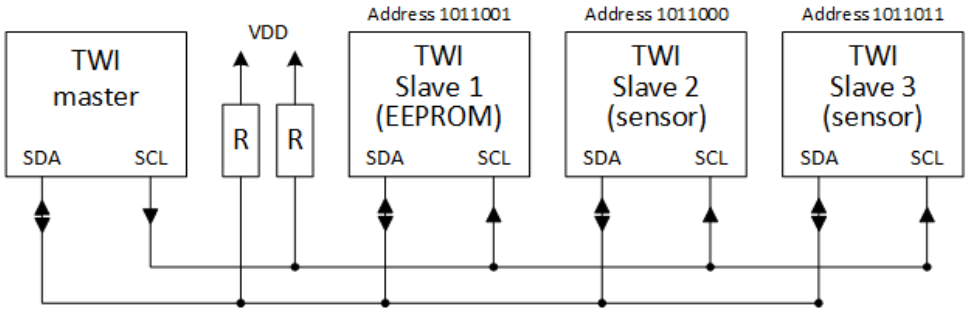
Instance:	UART	SERIAL0	SERIAL1	SERIAL2
UARTE	X			
UART	X			
SPIM		X	X	X
SPIS		X	X	X
SPI		X	X	X
TWIM		X	X	
TWIS		X	X	
TWI		X	X	

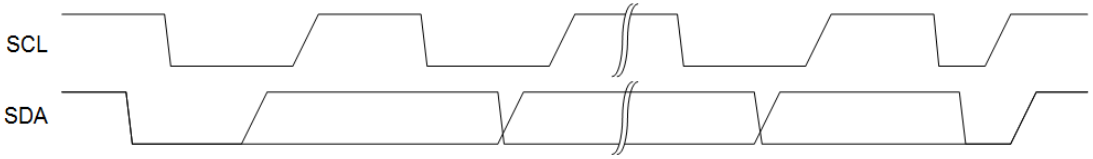
Outline

- Peripheral use
- Timers and RTC
- **TWI**
- UART
- SPI
- Comparators
- ADC
- Demo

TWIM, TWIS, TWI

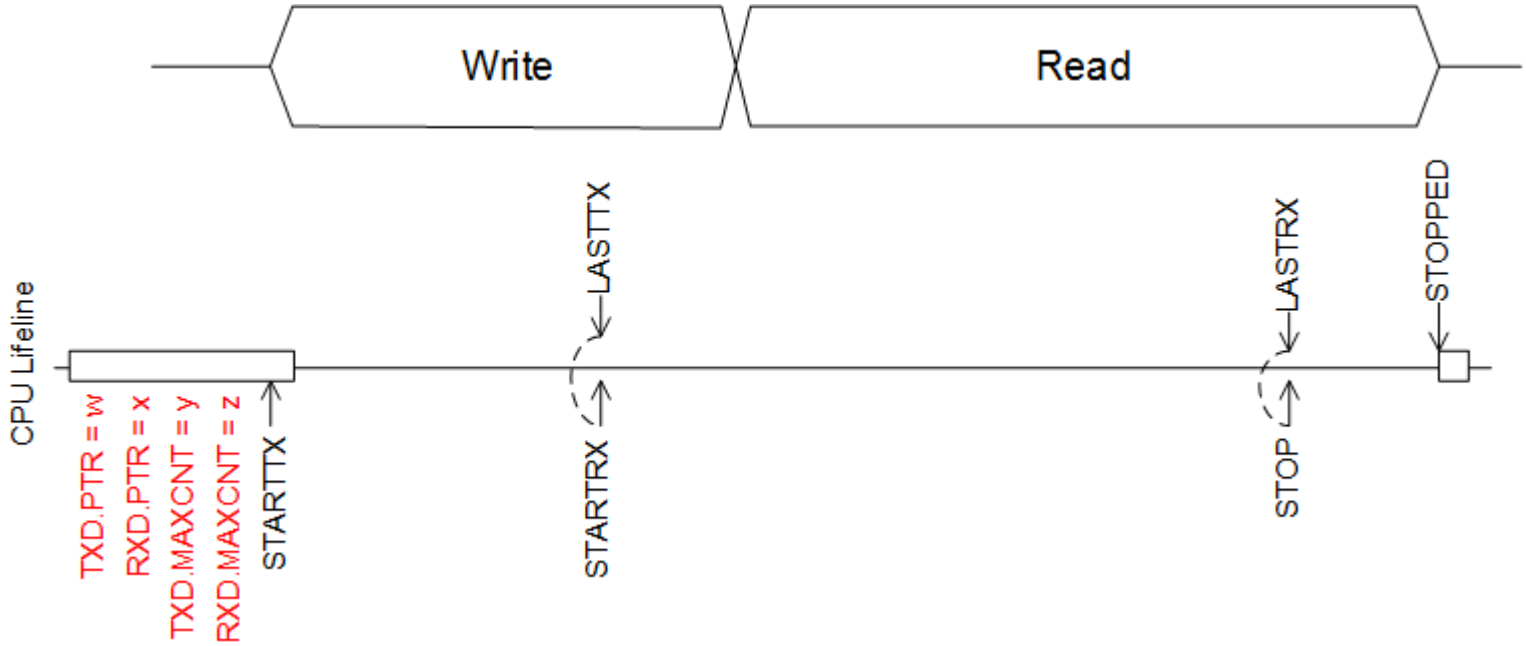
- 2 wire half duplex synchronous bus, I2C compatible
- 100kbps, 250kbps and 400kbps
- TWIM master with EasyDMA & Autolog
- TWIS slave with EasyDMA
- 2 instances in nRF52832
- Clock stretching supported
  - Master supports clock stretching
  - TWIS only stretches if no data loaded
- Single-master support





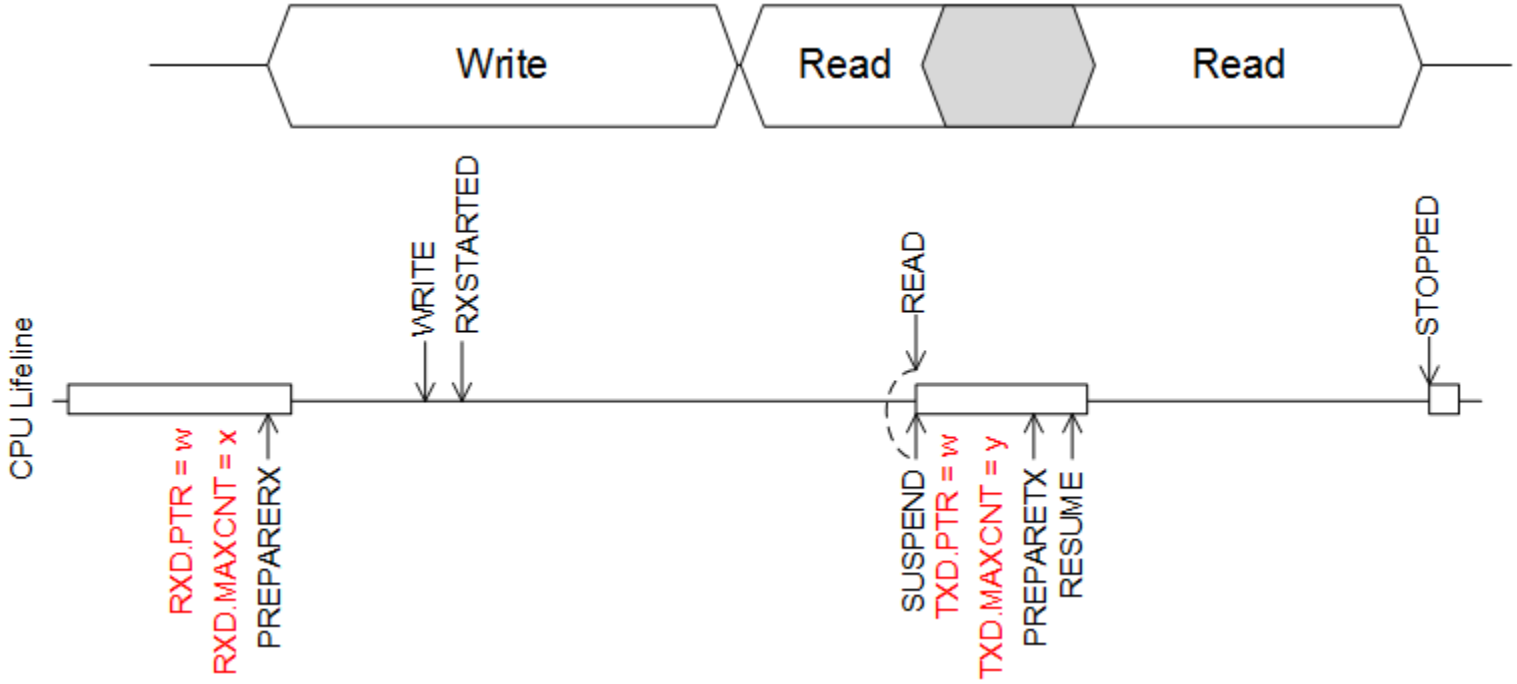
Automated communication, TWIM

- Minimizing CPU usage through EasyDMA and Shorts/PPI
  - Data buffers in RAM
  - Automatic change of mode through shorts
- Full master write&read supported without CPU use



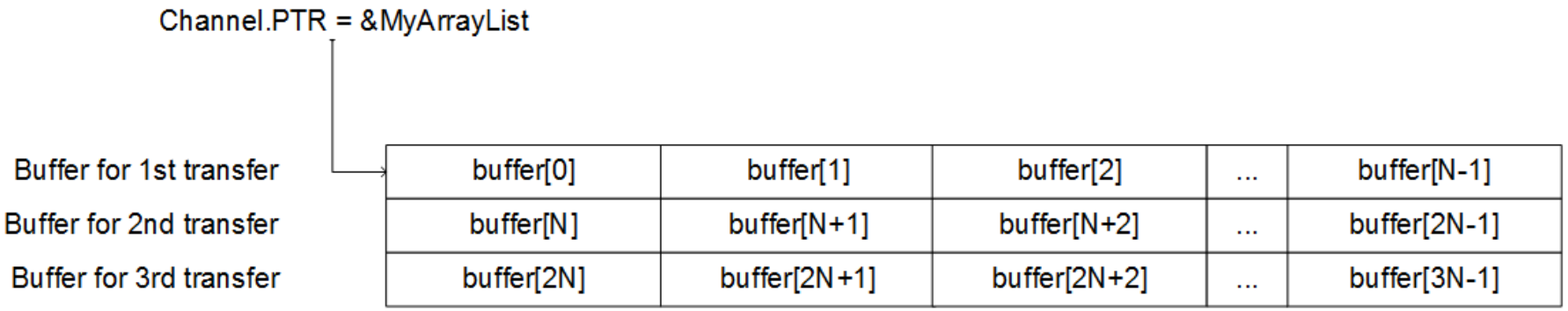
Automated communication, TWIS

- Full slave read or write supported without CPU use
- Event/Interrupt generated if SW needs to change out data, HW suspend of module while waiting for data update



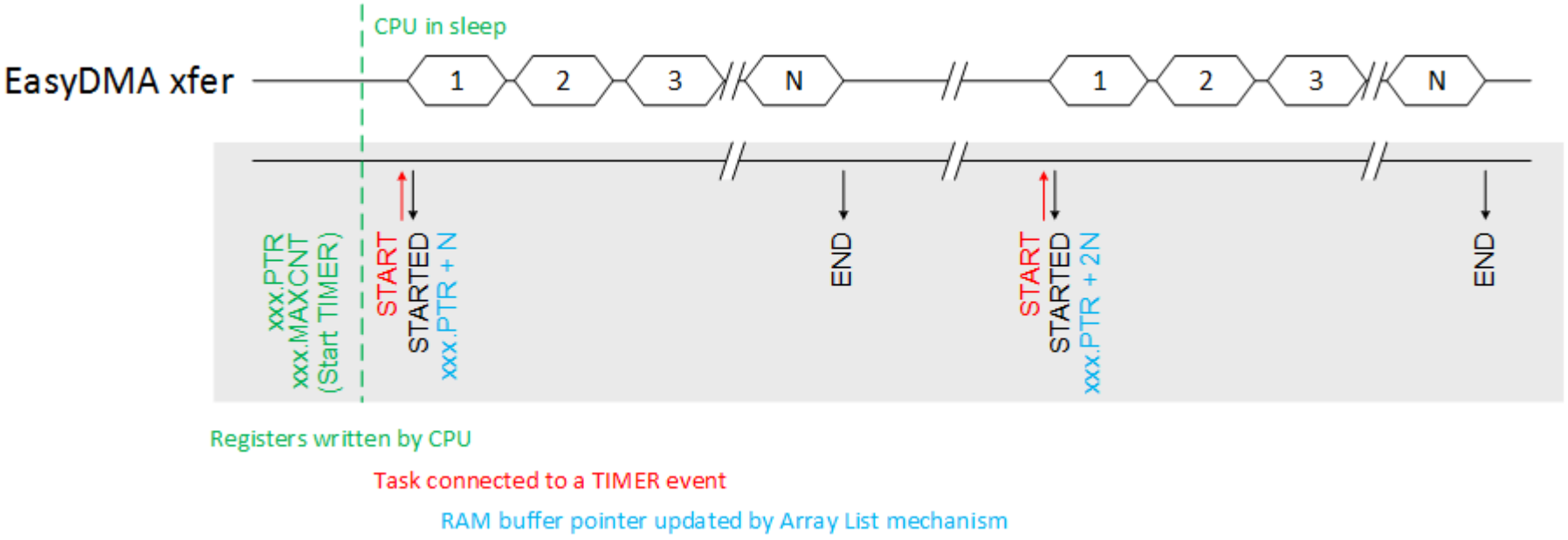
Array lists ("Autolog")

- Available for SPIM and TWIM
- Auto-increment of EasyDMA address pointer
- Allows fully automated sensor data collection, through use of PPI
  - assumes a known data packet size, not changing over time



Array lists (“Autolog”)

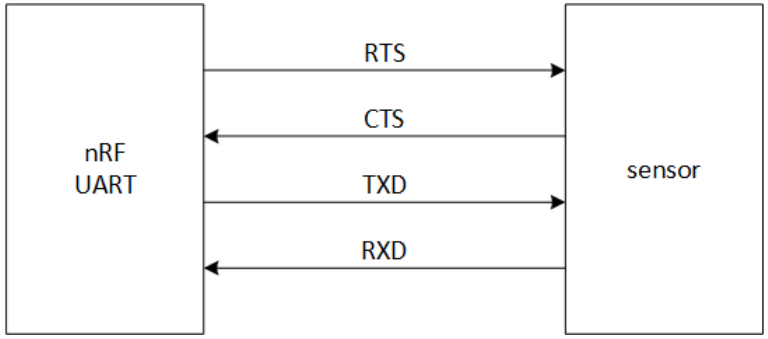
- Example with SPIM and TIMER
  - Collects N bytes for every transfer



Outline

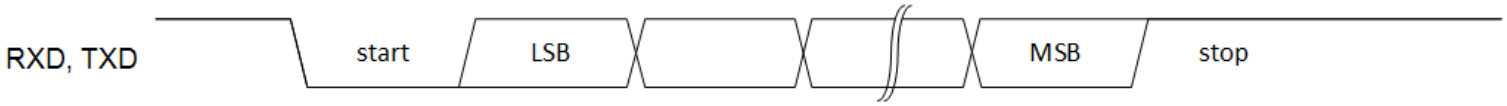
- Peripheral use
- Timers and RTC
- TWI
- **UART**
- SPI
- Comparators
- ADC
- Demo

UART, UARTE



Features:

- Up to 1Mbaud
- One instance in nRF52832
- UARTE with EasyDMA
- Optional CTS & RTS flow control by hardware
- UARTE: more accurate baudrates
  (14’400, 28’800, 38’400, 57600, 115’200, and 460’800)



UART usage

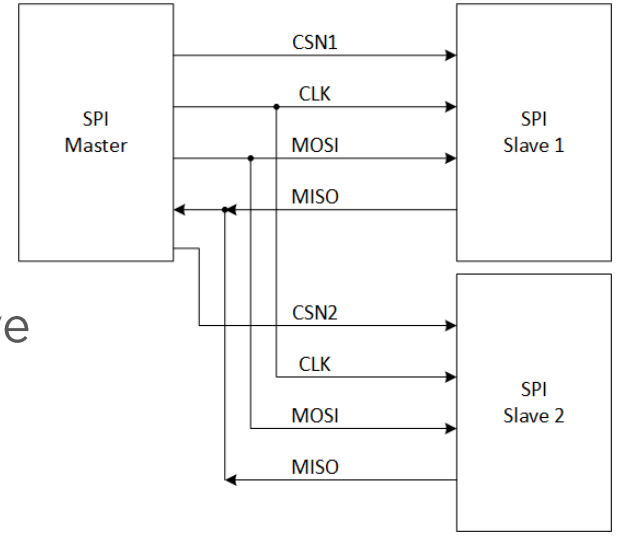
- Module can be started and stopped by HW through the PPI system or by SW
  - Easy to only start UART when needed
  - Can be done on Flow Control signal or RX pin
- Individual TX and RX buffers in Data RAM
  - Individually programmable buffer sizes
- Optional HW even parity generator

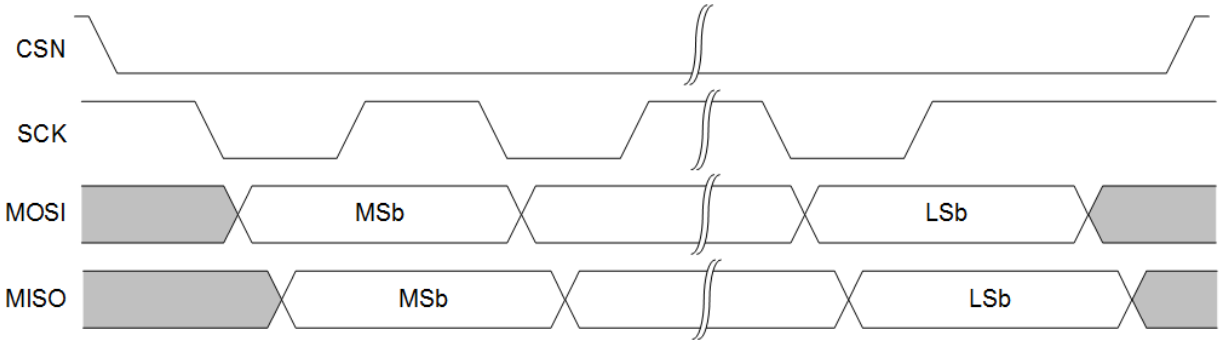
Outline

- Peripheral use
- Timers and RTC
- TWI
- UART
- **SPI**
- Comparators
- ADC
- Demo

SPIM, SPIS, SPI

- 3 wire full duplex synchronous interface
- Peer to peer
- SPIM (master) with EasyDMA & Autolog
- SPIS (slave) with EasyDMA
- HW Flow control through CSN for SPI Slave
- 3 instances in nRF52832





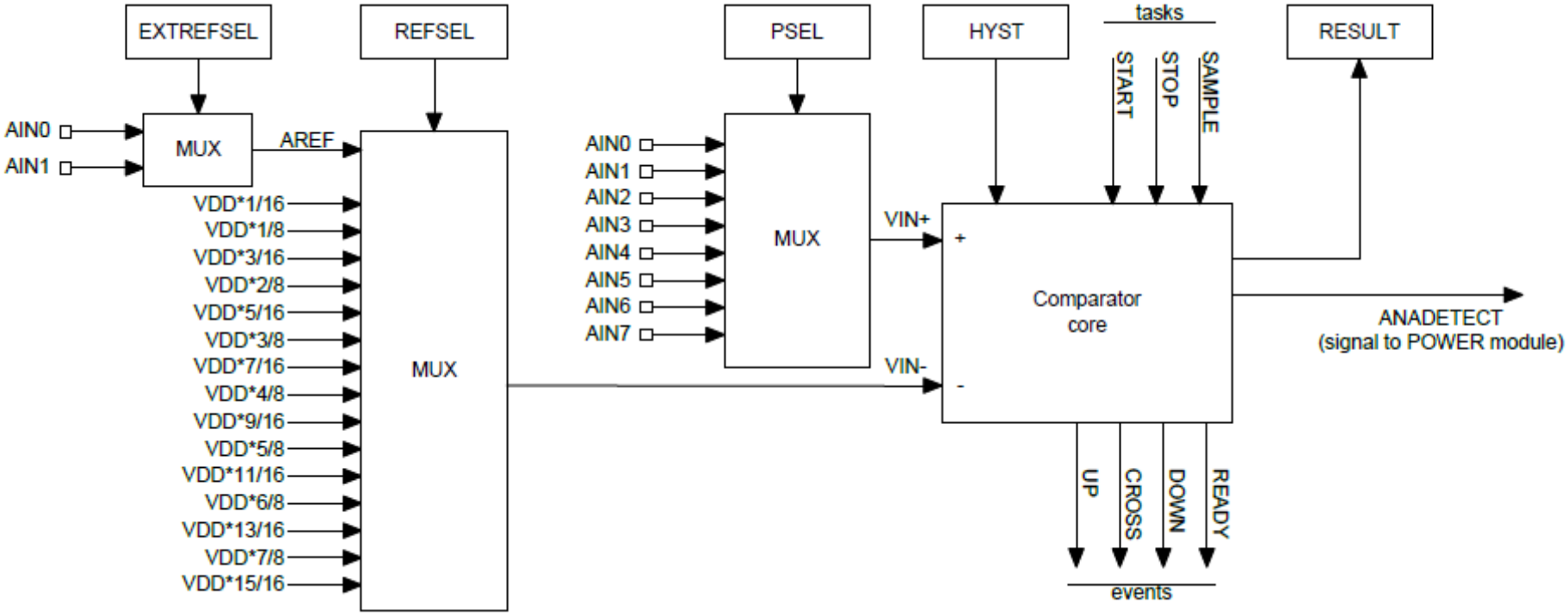
Outline

- Peripheral use
- Timers and RTC
- TWI
- UART
- SPI
- **Comparators**
- ADC
- Demo

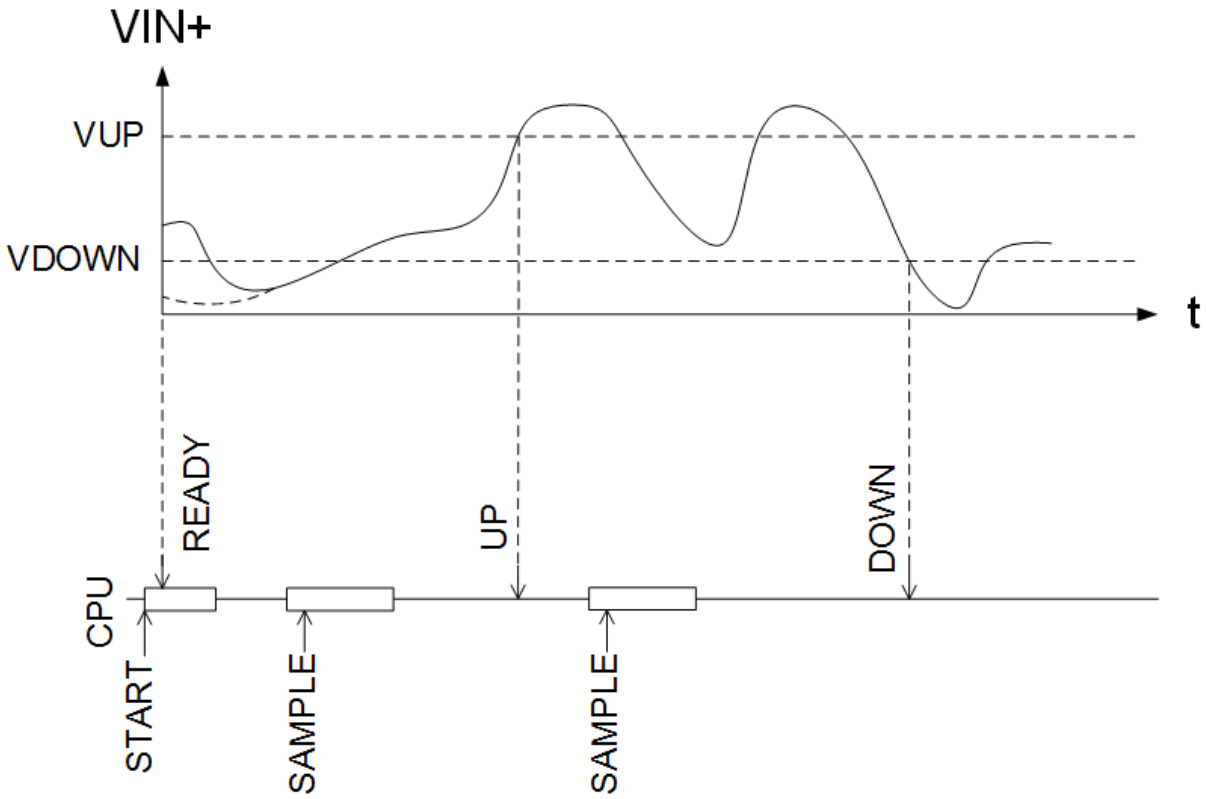
Low Power Comparator (LPCOMP)

- Eight inputs (AIN0 to AIN7)
- 0 - VDD input range
- Ultra low power
- Reference voltage options:
  - Two external analog reference inputs, or
  - 15-level internal reference ladder (VDD/16)
- Optional hysteresis enable on input
- Continuous monitoring
- Wake up source from system ON or OFF mode
- Sampling function

LPCOMP block diagram



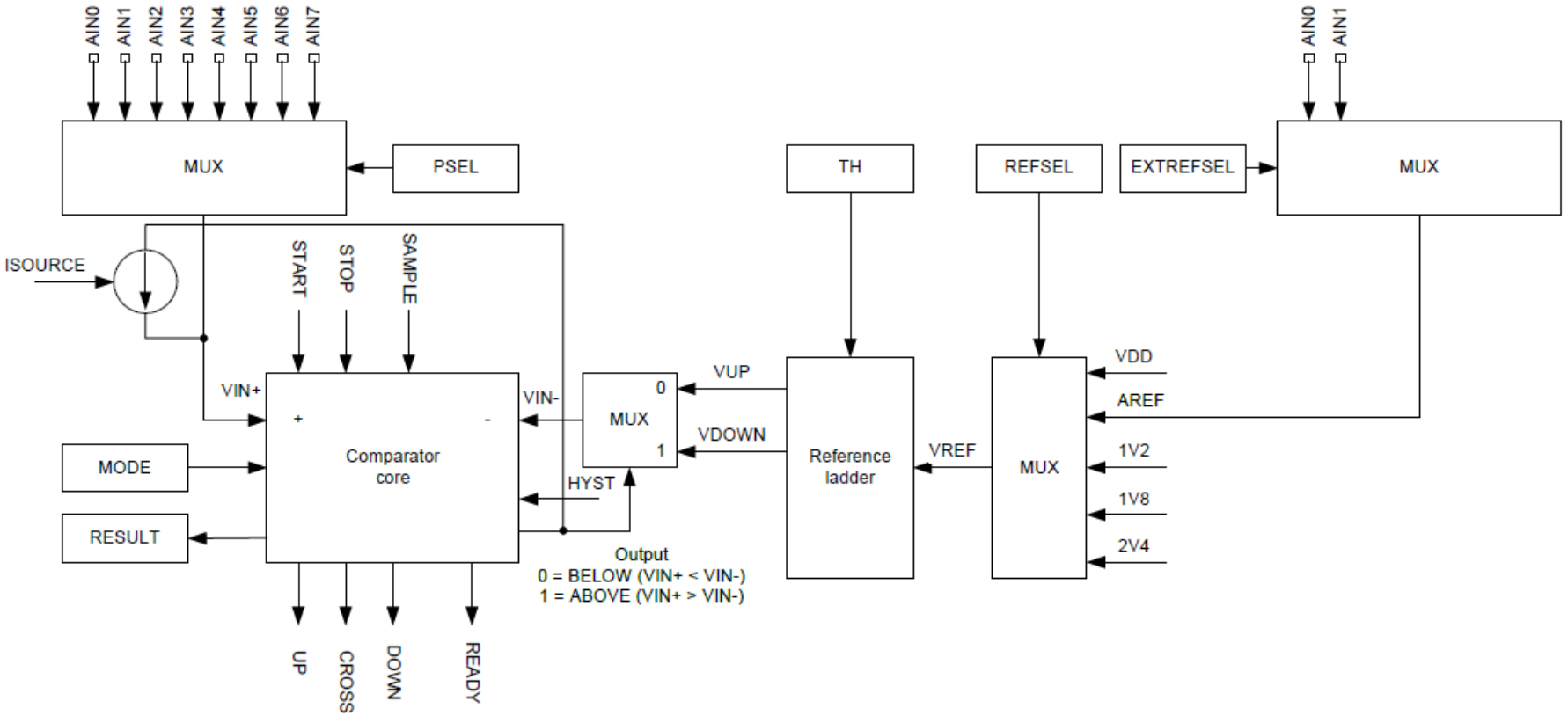
LPCOMP events



General Purpose comparator (COMP)

- New in nRF52832
- Similar to LPCOMP
- More accurate internal references (1.2 V, 1.8 V and 2.4 V)
- Single-ended mode: flexible hysteresis using a 64-level reference ladder
- Single-pin capacitive sensor support through optional current source
- Three operation modes: low power, normal and high-speed

COMP block diagram



Outline

- Peripheral use
- Timers and RTC
- TWI
- UART
- SPI
- Comparators
- **ADC**
- Demo

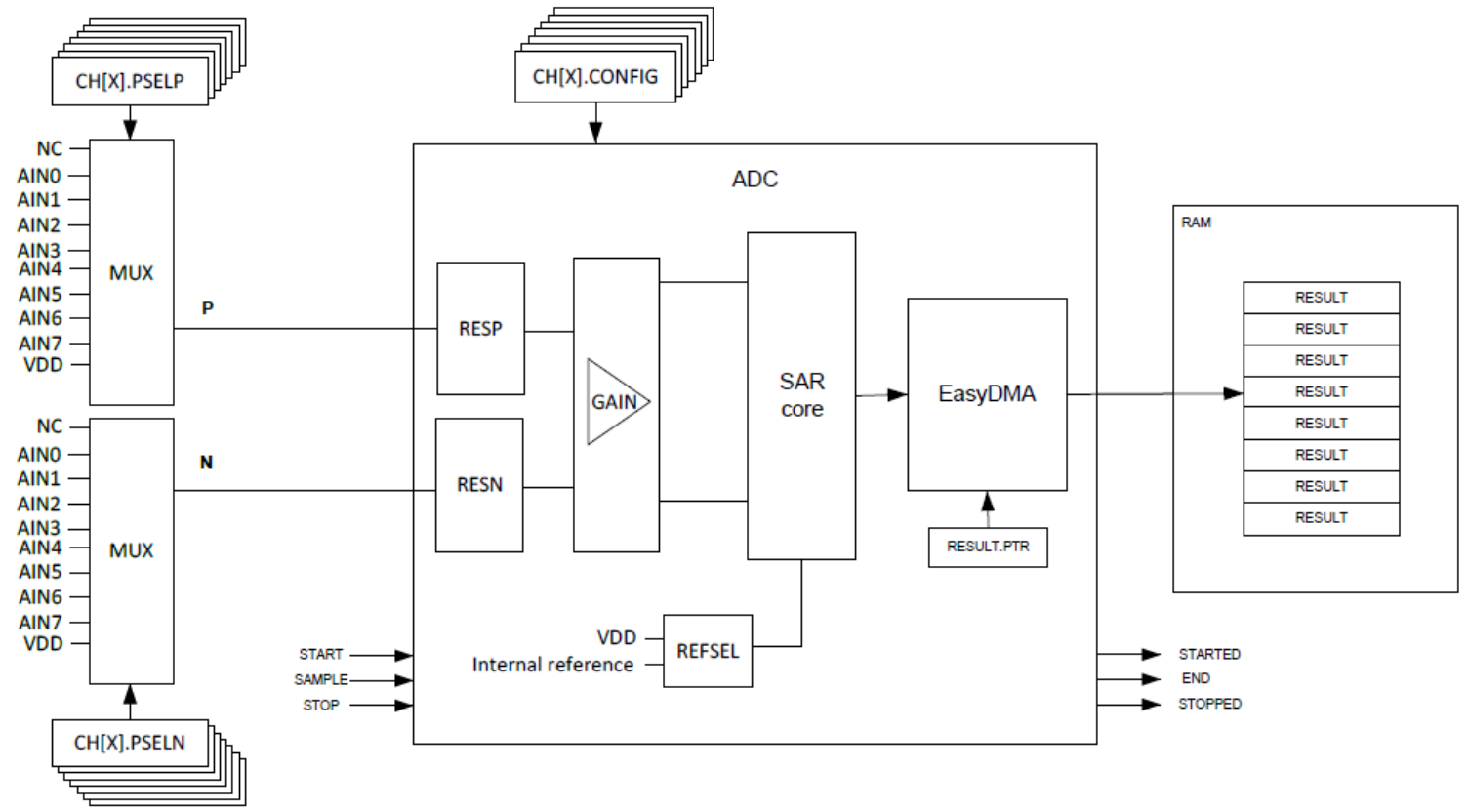
Successive Approximation ADC

- New in nRF52832 compared to nRF51

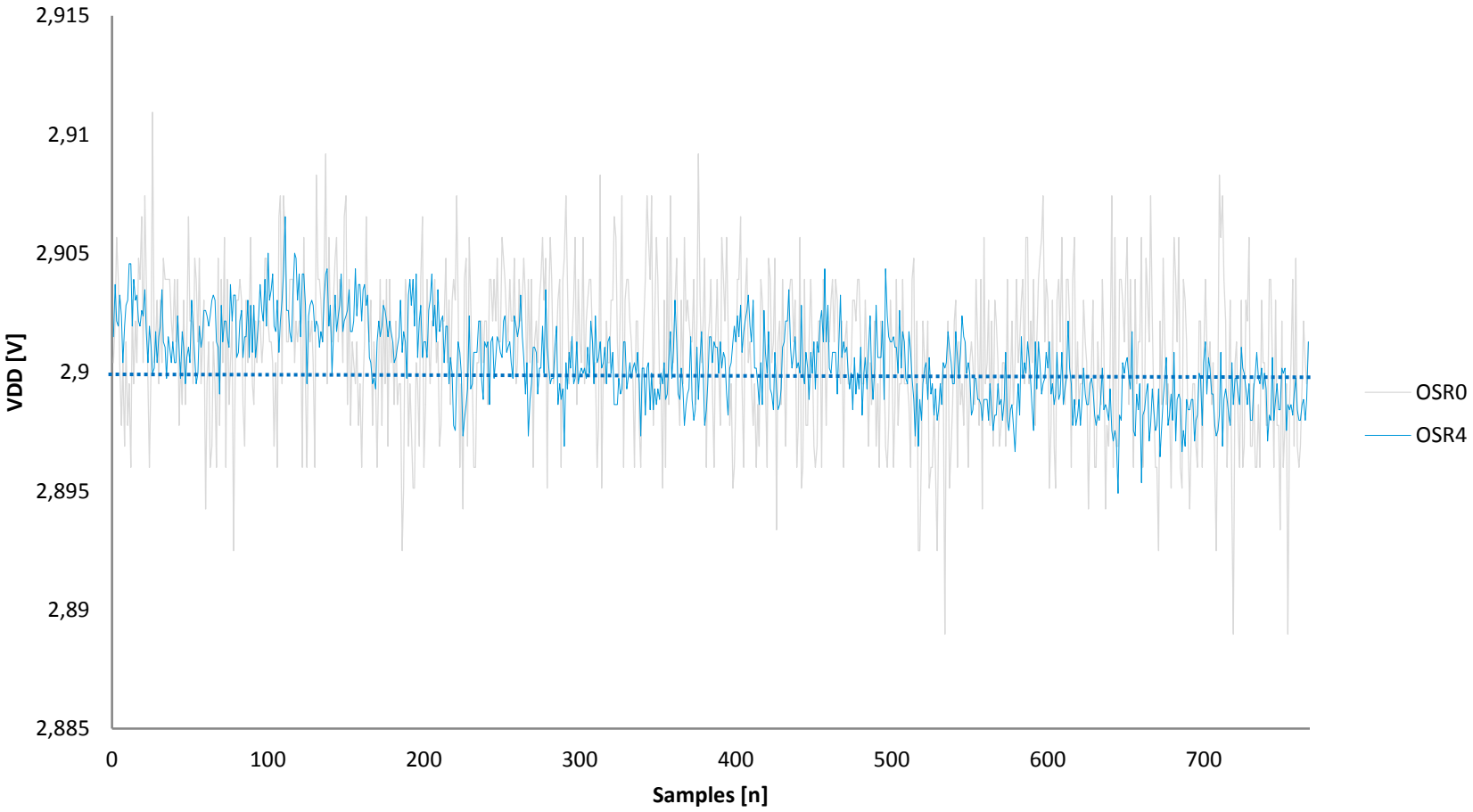
- 8/10/12-bit resolution
- Full scale input range (0 to VDD)
- Single-ended or differential mode
- Up to eight input channels
- Internal resistor string

- One-shot or scan sampling mode
- Optional oversampling mode
- Continuous sampling (~200kHz) without the need of an external timer
- On-the-fly limits checking
- EasyDMA

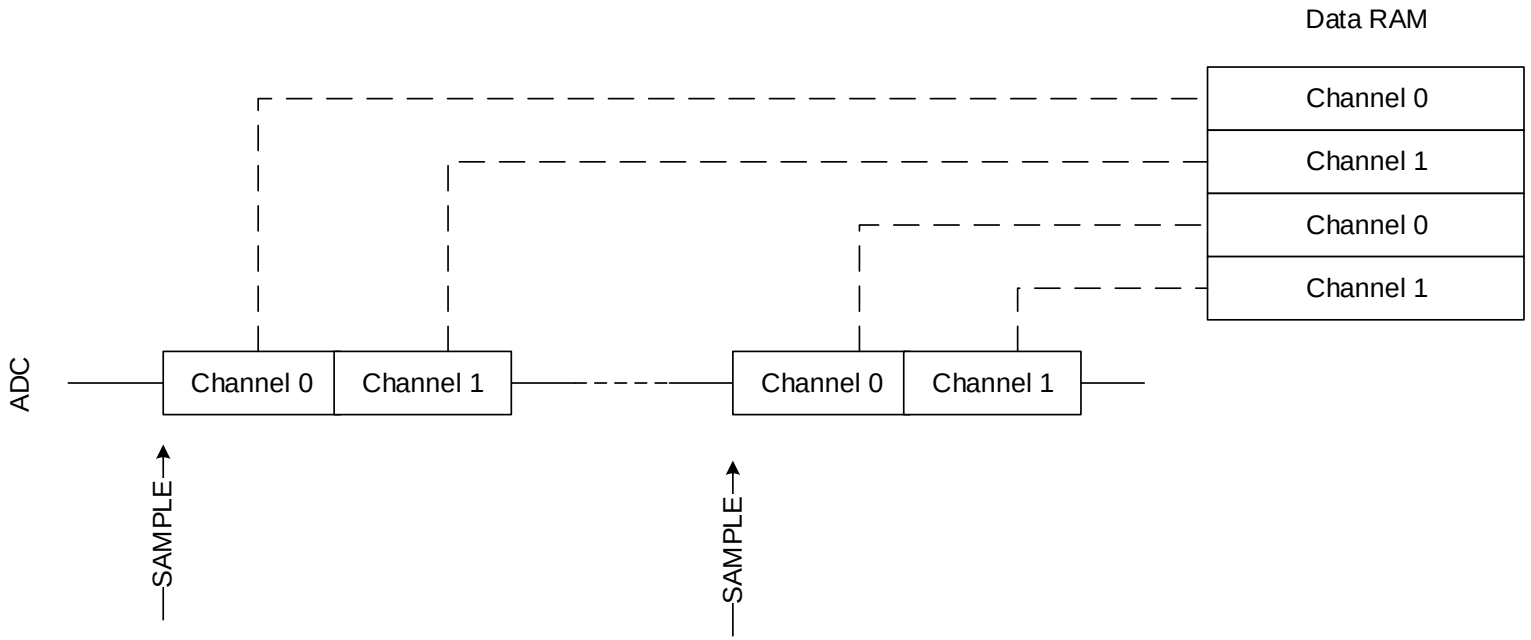
SAADC block diagram



Oversample

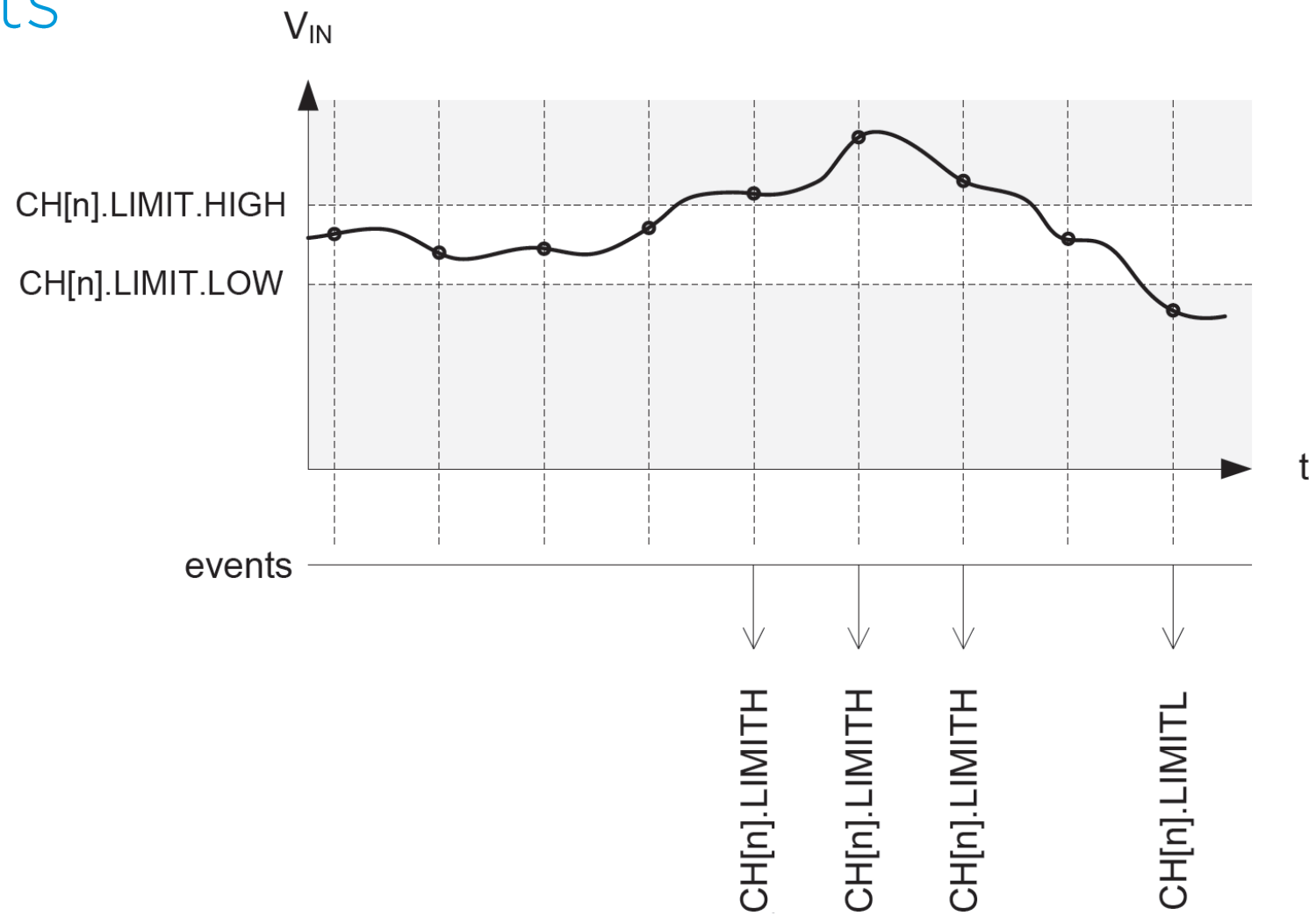

2,915
2,91
2,905
2,9
2,895
2,89
2,885
VDD [V]
0
100
200
300
400
500
600
700
Samples [n]
OSR0
OSR4

Scan



- One SAMPLE task will sample up to 8 channels
- Each channel can be configured independently
  - PSELP – Connection to positive input
  - PSELN – Connection to negative input
  - CONFIG (RESP, RESN, GAIN, REFSEL, TACQ, MODE)

Limits

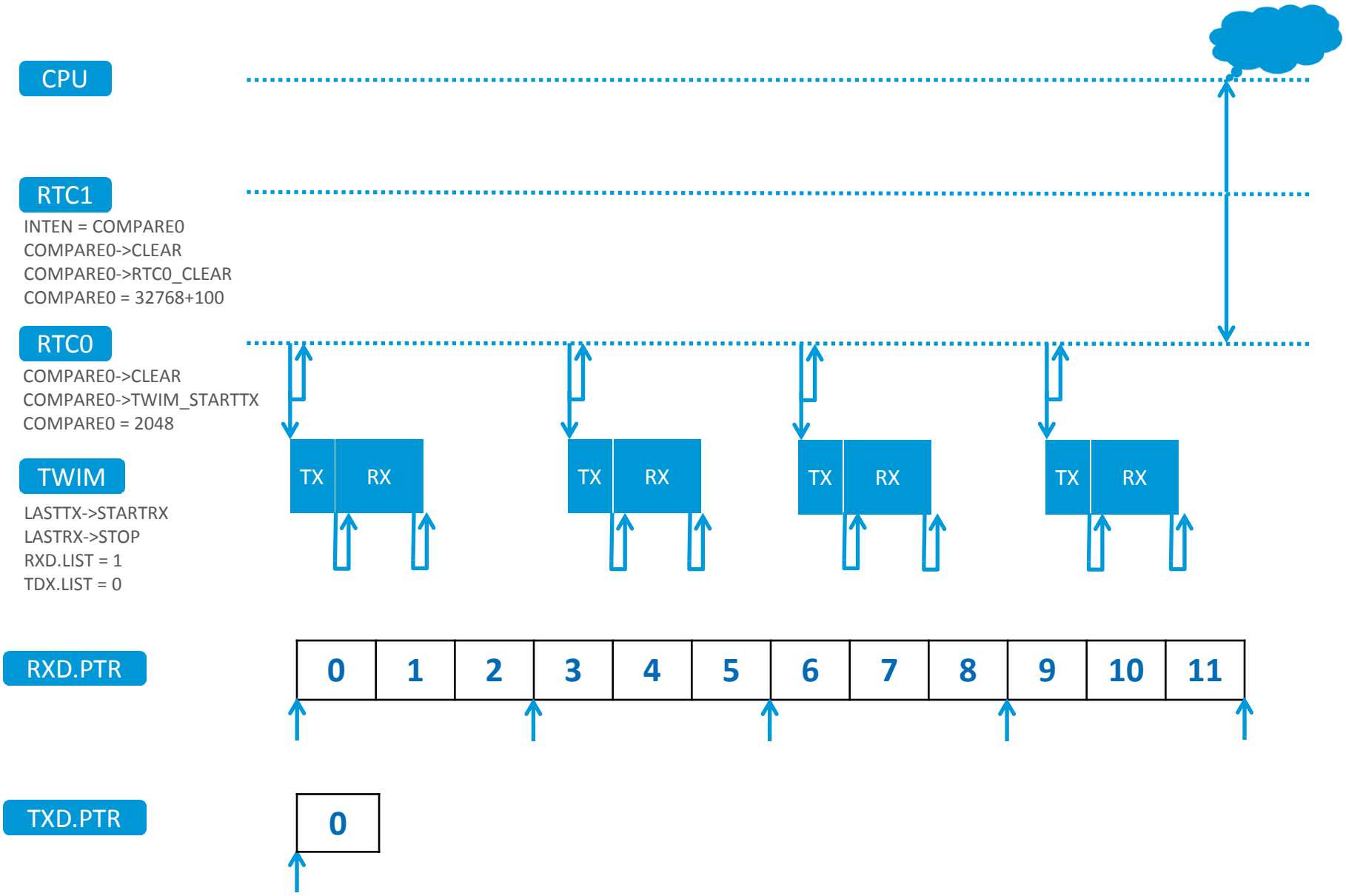


Outline

- Peripheral use
- Timers and RTC
- TWI
- UART
- SPI
- Comparators
- ADC
- **Demo**

TWIM with AUTOLOG

- Collect accelerometer samples
  - 1 byte of TX (accelerometer register address 0x00)
  - 3 bytes of RX (accelerometer data)
- RTC0 triggers the TXRX transfer
- TWIM shorts for seamless TX to RX transition
- AUTOLOG increments RX buffer address once every transfer is performed
- RTC1 wakes up CPU to process the data after 16 aquisiotions



Setting up RTC0

```
nrf_drv_rtc_init(&rtc0, NULL, rtc_event_handler);
nrf_drv_rtc_cc_set(&rtc0, 0, CC_VALUE, false);
nrf_drv_rtc_enable(&rtc0);

nrf_drv_ppi_channel_alloc(&ppi_channel);
nrf_drv_ppi_channel_assign(ppi_channel, (uint32_t)&NRF_RTC0->EVENTS_COMPARE[0], (uint32_t)&NRF_TWIM0->TASKS_STARTTX);
nrf_drv_ppi_channel_enable(ppi_channel);

nrf_drv_ppi_channel_alloc(&ppi_channel);
nrf_drv_ppi_channel_assign(ppi_channel, (uint32_t)&NRF_RTC0->EVENTS_COMPARE[0], (uint32_t)&NRF_RTC0->TASKS_CLEAR);
nrf_drv_ppi_channel_enable(ppi_channel);
```

Setting up RTC1

```
nrf_drv_rtc_init(&rtc1, NULL, rtc_event_handler);
nrf_drv_rtc_cc_set(&rtc1, 0, NUMBER_OF_XFERS*CC_VALUE+100, true);
nrf_drv_rtc_enable(&rtc1);

nrf_drv_ppi_channel_alloc(&ppi_channel);
nrf_drv_ppi_channel_assign(ppi_channel, (uint32_t)&NRF_RTC1->EVENTS_COMPARE[0], (uint32_t)&NRF_RTC1->TASKS_CLEAR);
nrf_drv_ppi_channel_enable(ppi_channel);

nrf_drv_ppi_channel_alloc(&ppi_channel);
nrf_drv_ppi_channel_assign(ppi_channel, (uint32_t)&NRF_RTC1->EVENTS_COMPARE[0], (uint32_t)&NRF_RTC0->TASKS_CLEAR);
nrf_drv_ppi_channel_enable(ppi_channel);
```

Setting up TWI TXRX

```
nrf_drv_twi_xfer_desc_t xfer_desc;
xfer_desc.address = MMA7660_DEFAULT_ADDRESS;
xfer_desc.type = NRF_DRV_TWI_XFER_TXRX;
xfer_desc.primary_length = 1;
xfer_desc.secondary_length = 3;
xfer_desc.p_primary_buf = m_txbuf;
xfer_desc.p_secondary_buf = m_rxbuf;

uint32_t flags = NRF_DRV_TWI_FLAGS_HOLD_XFER                    |
                 NRF_DRV_TWI_FLAGS_REPEATED_XFER                |
                 NRF_DRV_TWI_FLAGS_NO_XFER_EVT_HANDLER          |
                 NRF_DRV_TWI_FLAGS_RX_POSTINC;

nrf_drv_twi_xfer(&m_twi_master, &xfer_desc, flags)
```