Analoog-digitaalmuundur
ADC
Pisi-XBee5

TTÜ Robotiklubi

Tallinn University of Technology Robotics Club

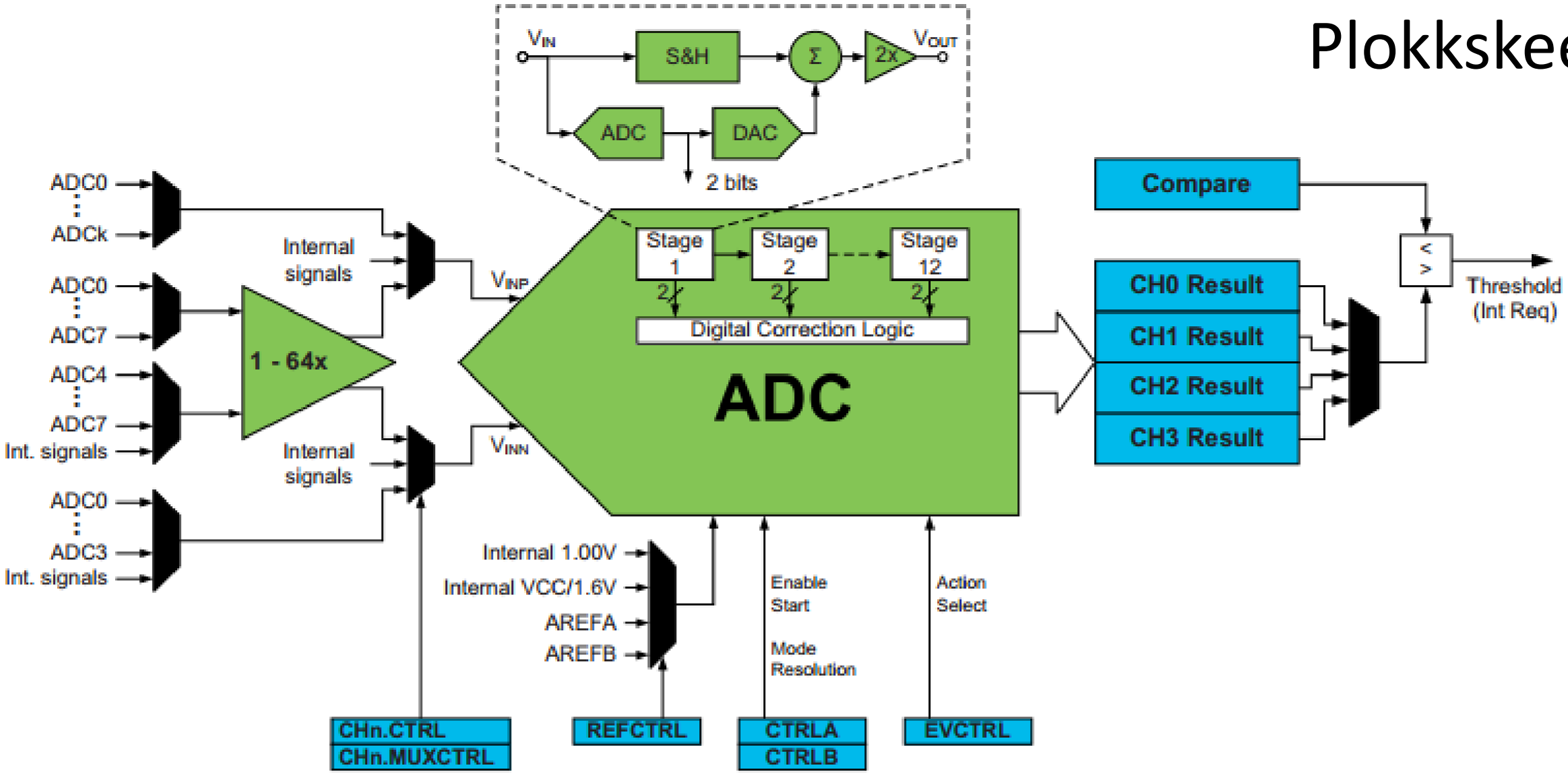
ADC

- muundab analoogpinge väärtuse digitaalseks väärtuseks
- Digitaalne väärtus on 12-bitine.
- Võrdluspinge määrab maksimaalse digitaalse väärtuse.
- Maksimaalne väärtus xmega kontrolleritel 2^{12} - 1 ehk 4095.
- AVR ADC töötab võrdlusmeetodil (inglise keeles *successive approximation*).

Features

- One ADC with 12-bit resolution
- 2 Msps sample rate
- Signed and Unsigned conversions
- 4 result registers with individual input channel control
- 12 single ended inputs
- 8x4 differential inputs
- 4 internal inputs:
  - Integrated Temperature Sensor
  - DAC Output
  - VCC voltage divided by 10
  - Bandgap voltage
- Software selectable gain of 2, 4, 8, 16, 32 or 64
- Selectable accuracy of 8- or 12-bit.
- Internal or External Reference selection
- Event triggered conversion for accurate timing
- DMA transfer of conversion results
- Interrupt/Event on compare result

Plokkskeem



Note: 1. k = 7 for XMEGA A1/A3 devices, k = 11 for XMEGA A4 devices.

Kalibreerimine (valikuline)

- **ADCA Calibration register 0 and 1**
  - **Sisaldavad ADC tehasekalibratsiooni**
  - Xmega manual lk 38
- **CALL and CALH – Calibration Value register**
  - **Siia registrisse kirjutatakse tehasekalibratsiooni andmed**
  - Xmega manual lk 301

```
adc->CALL = SP_ReadCalibrationByte( PROD_SIGNATURES_START + ADCACAL0_offset );
adc->CALH = SP_ReadCalibrationByte( PROD_SIGNATURES_START + ADCACAL1_offset );
```

CTRLB

- `ADCA.CTRLB = 0x01 << 4;`

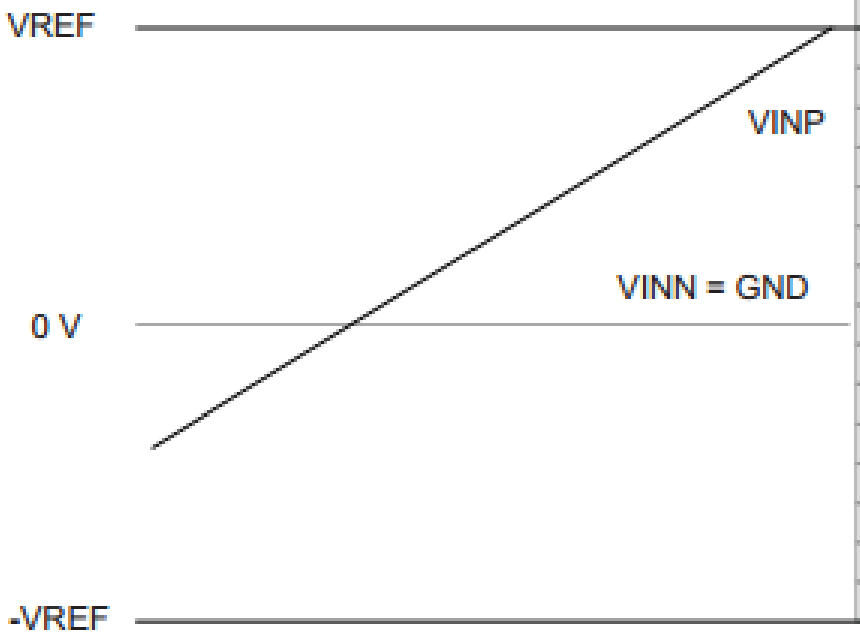
või

- `ADCA.CTRLB = (ADC_CONMODE_bm|ADC_RESOLUTION_12BIT_gc);`

- Conversion mode: signed
- Resolution: 12bit

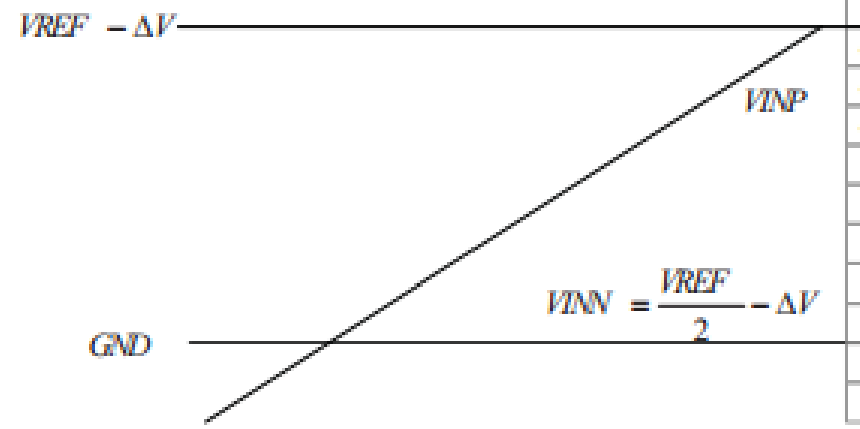
Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	CONVMODE	FREERUN	RESOLUTION[1:0]		–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial Value	0	0	0	0	0	0	0	0

Figure 25-10.Signed single-ended and internal input, input range, and result representation.



Dec	Hex	Binary	16-bit result register
2047	7FF	0111 1111 1111	0000 0111 1111 1111
2046	7FE	0111 1111 1110	0000 0111 1111 1110
2045	7FD	0111 1111 1101	0000 0111 1111 1101
...	...	...	...
3	3	0000 0000 0011	0000 0000 0000 0011
2	2	0000 0000 0010	0000 0000 0000 0010
1	1	0000 0000 0001	0000 0000 0000 0001
0	0	0000 0000 0000	0000 0000 0000 0000
-1	FFF	1111 1111 1111	1111 1111 1111 1111
-2	FFE	1111 1111 1110	1111 1111 1111 1110
...	...	...	...
-2045	803	1000 0000 0011	1111 1000 0000 0011
-2046	802	1000 0000 0010	1111 1000 0000 0010
-2047	801	1000 0000 0001	1111 1000 0000 0001
-2048	800	1000 0000 0000	1111 1000 0000 0000

Figure 25-11.Unsigned single-ended and internal input, input range, and result representation.



Dec	Hex	Binary	16-bit result register
4095	FFF	1111 1111 1111	0000 1111 1111 1111
4094	FFE	1111 1111 1110	0000 1111 1111 1110
4093	FFD	1111 1111 1101	0000 1111 1111 1101
...	...	...	...
203	0CB	0000 1100 1011	0000 0000 1100 1011
202	0CA	0000 1100 1010	0000 0000 1100 1010
201	0C9	0000 1100 1001	0000 0000 1100 1001
200	0C8	0000 1100 1000	0000 0000 1100 1000
...			
0	0	0000 0000 0000	0000 0000 0000 0000

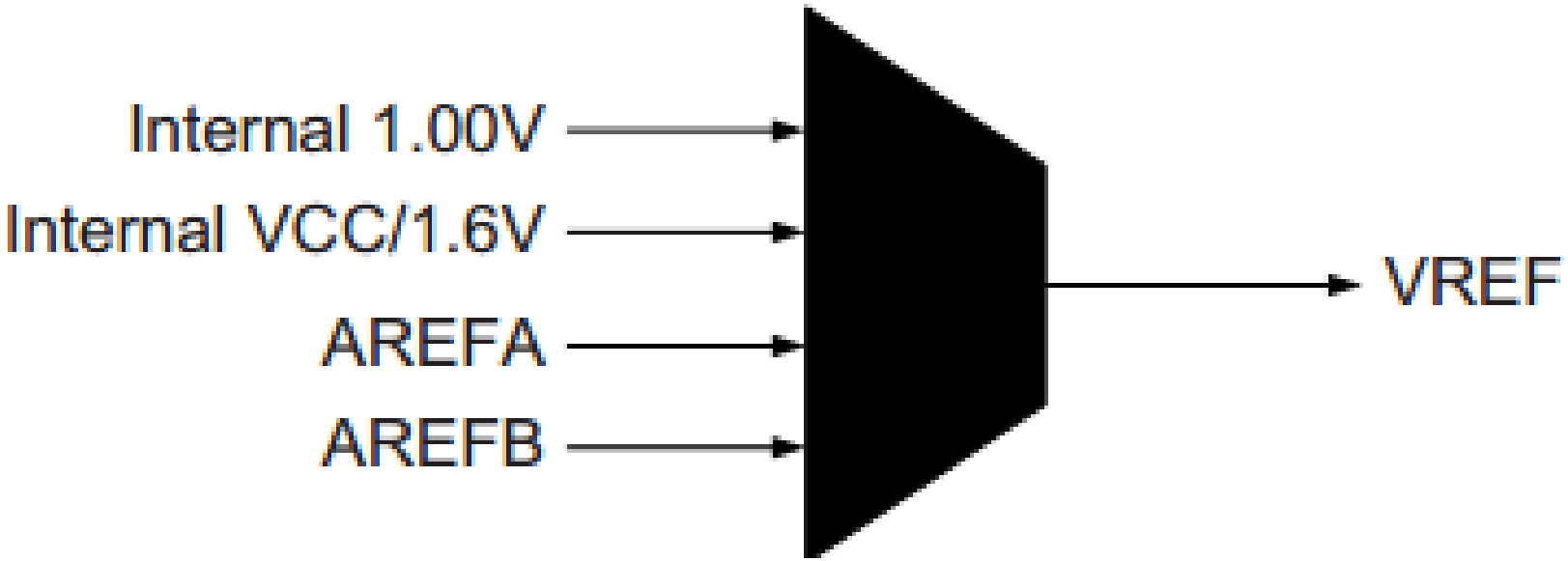
Signed vs unsigned

Referents pinge

```
ADCA.REFCTRL = 0x01 << 4;

või

ADC_Reference_Config(&ADCA, ADC_REFSEL_INTVCC_gc);
IntVCC = Vcc / 1.6 = 2,0625 V
```



Prescaler

PRESCALER[2:0]	Group configuration	Peripheral clock division factor
000	DIV4	4
001	DIV8	8
010	DIV16	16
011	DIV32	32
100	DIV64	64
101	DIV128	128
110	DIV256	256
111	DIV512	512

```
ADC_Prescaler_Config(&ADCA, ADC_PRESCALER_DIV64_gc);
```

Või:

```
ADCA.PRESCALER = 0x01 << 2;
```

Bit	7	6	5	4	3	2	1	0
+0x04	-	-	-	-	-	PRESCALER[2:0]		
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

```
ADCA.CH0.CTRL = 0x01;
```

Seadistame CH0 gain 1x ja single ended mode.

CTRL

Channel Control Register

Table 25-9. Channel input modes, CONVMODE=0 (unsigned mode).

INPUTMODE[1:0]	Group configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10		Reserved
11		Reserved

Bit	7	6	5	4	3	2	1	0
+0x00	**START**	–	–	**GAIN[2:0]**			**INPUTMODE[1:0]**	
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Setting up the which pins to convert.**

- ```
  ADCA.CH0.MUXCTRL = 0x01 << 3;
  ```

MUXCTRL

Bit	7	6	5	4	3	2	1	0
+0x01	–	MUXPOS[3:0]				–	MUXNEG[1:0]	
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
  ```

CTRLA Enable ADC

- Enable ADC

```
ADCA.CTRLA = 0x01;
```

- Oota et ADC oleks tööks valmis

```
//ADC_Wait_8MHz(&ADCA);
_delay_ms(100);
```

Bit	7	6	5	4	3	2	1	0
+0x00	DMASEL[1:0]		CHSTART[3:0]				FLUSH	ENABLE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ADC Lugemine

```
ADCA.CH0.MUXCTRL = 0x01 << 2; //loeme ADC0-i

// Start conversion (jätame enable ka tööle)
ADCA.CTRLA |= 0x01 << 2;

//ootame et ADC konverteerimise lõpetaks
while(!ADCA.CH0.INTFLAGS);

//Prindime väljundi
sprintf(buff,"%5d\n\r",ADCA.CH0RES);
radio_puts(buff);
_delay_ms(100);
```

```
#include <avr/io.h>
#include "drivers/board.h"
#include "drivers/motor.h"
#include <util/delay.h>

int main(void)
{
char buff[30];
clock_init();// Seadista süsteemi kell 32MHz peale
adc_init();// Seadista ADC kanal 0
radio_init(57600);// Seadista raadiomooduli UART

    while(1)
    {
        sprintf(buff,"%5d\n\r",adc_read(0));
        radio_puts(buff);
        _delay_ms(100);
    }
}
```

Kasutades teeki

Täname kuulamast!

Lauri Leemet
robotiklubi.ee
facebook.com/TTYRobotiklubi
robotiklubi@robotiklubi.ee