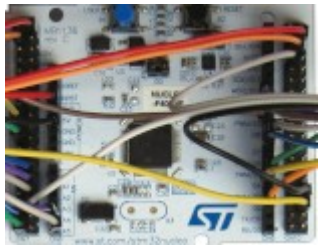
Mikroprozessortechnik

Prof. Dr. Michael Lipp

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

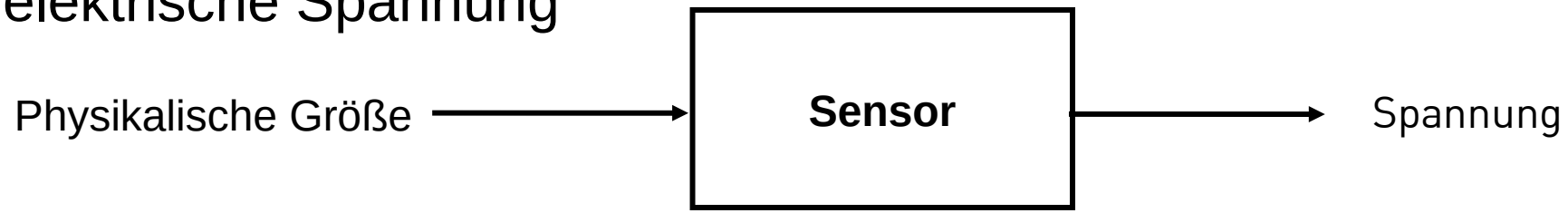
h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES



ADC – Analog Digital Converter

Verarbeitung analoger Signale

- **Analoge Eingangsgrößen**
  - Sensoren wandeln oft physikalische Größen in eine analoge, elektrische Spannung



  - Die Spannung ist in der Regel zeit- und wertkontinuierlich
- **Verarbeitung mit einem µC (digital)**
  - Umwandlung in eine zeit- und wertdiskrete, digitale Darstellung durch Analog/Digital-Wandlers (engl. Analog/Digital Converter, ADC)

Verarbeitung analoger Signale

NTC ist ein Temperatur abhängige Widerstand

- **Analoge Signale am µC – Beispiel NTC**
  - Heißleiter: NTC
  - Widerstand ändert sich mit der Temperatur T
  - µC Anschluss mittels Spannungsteiler

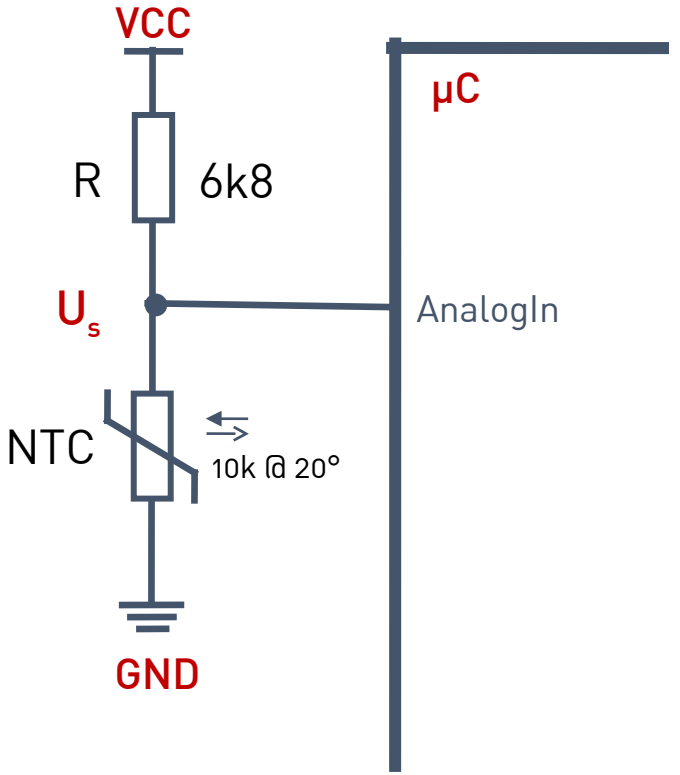
Änderung der Temperatur

↓

Änderung des Widerstandswertes

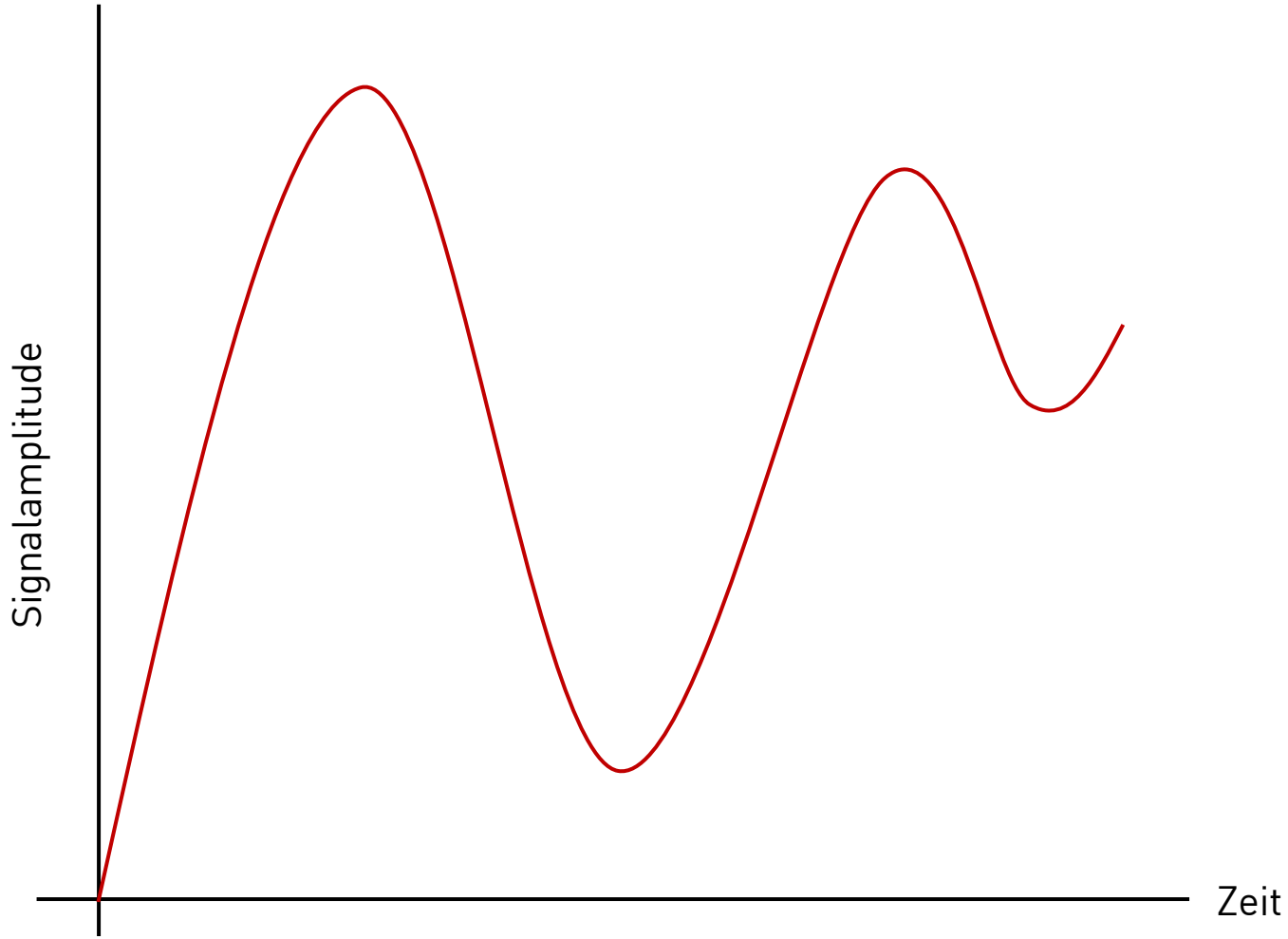
↓

Änderung der Spannung U_s am Eingang des ADCs



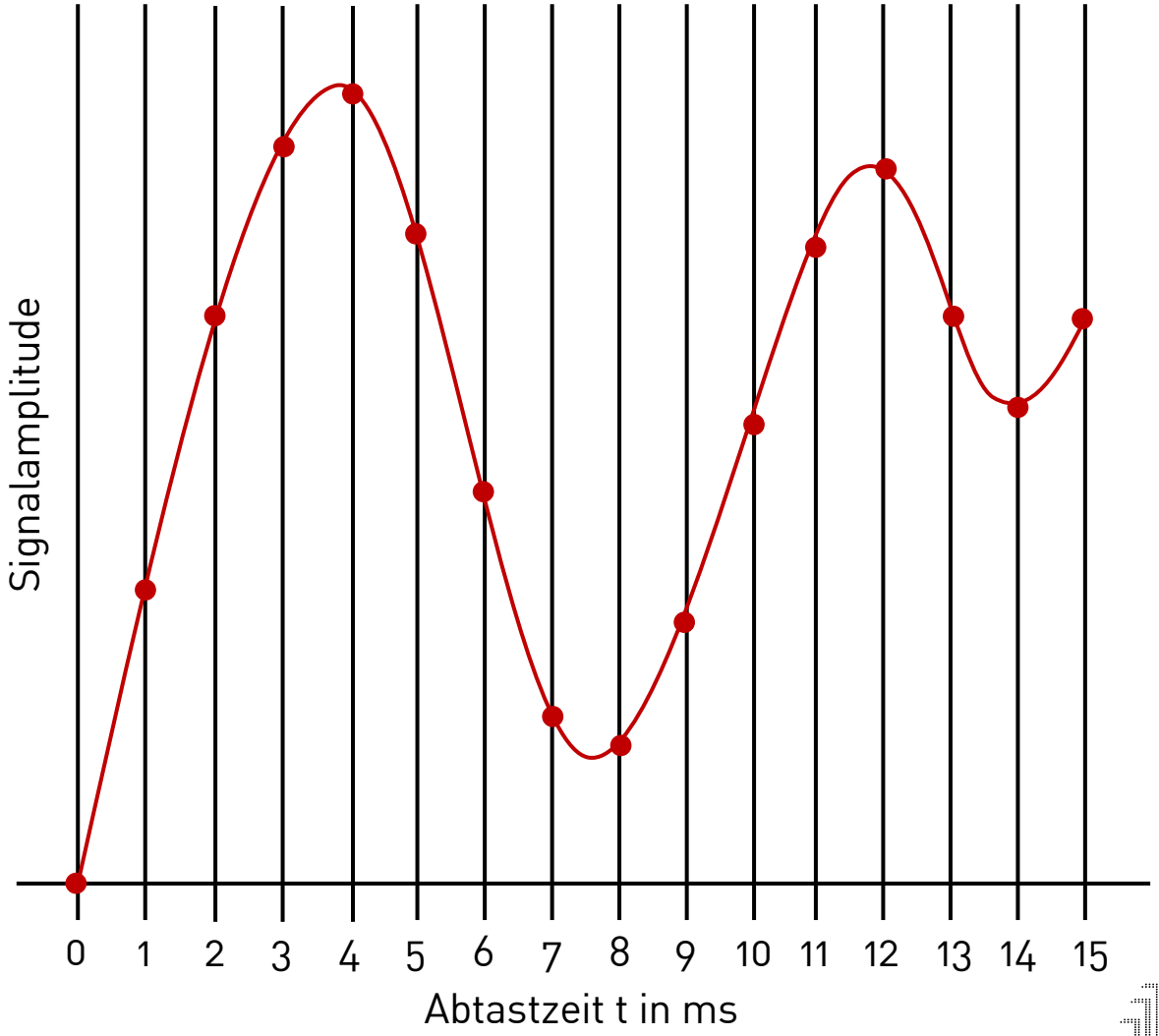
Verarbeitung analoger Signale

- **Analoges Signal**
  - Zeit- und wert-kontinuierlich



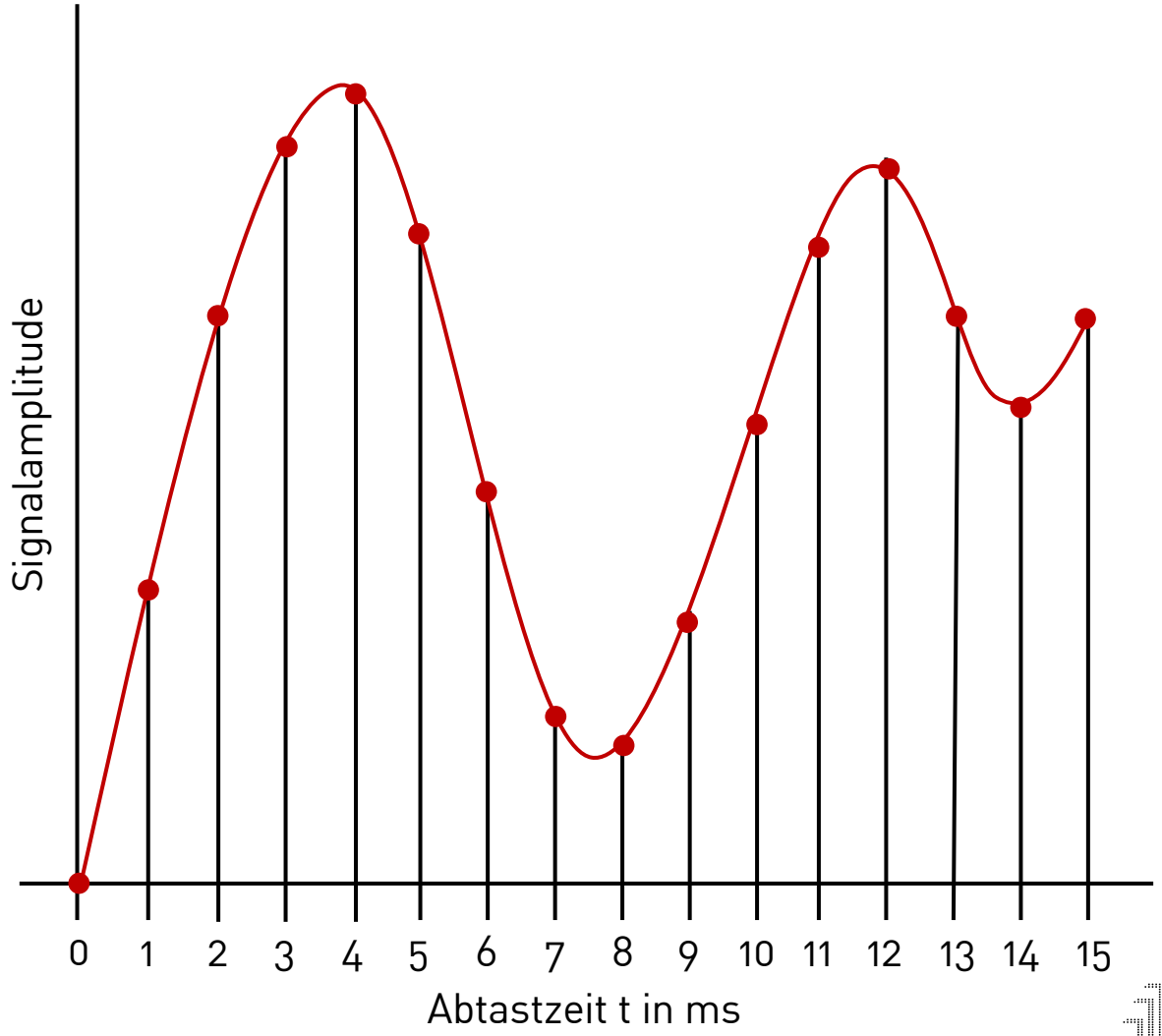
Verarbeitung analoger Signale

- **Abtastung...**



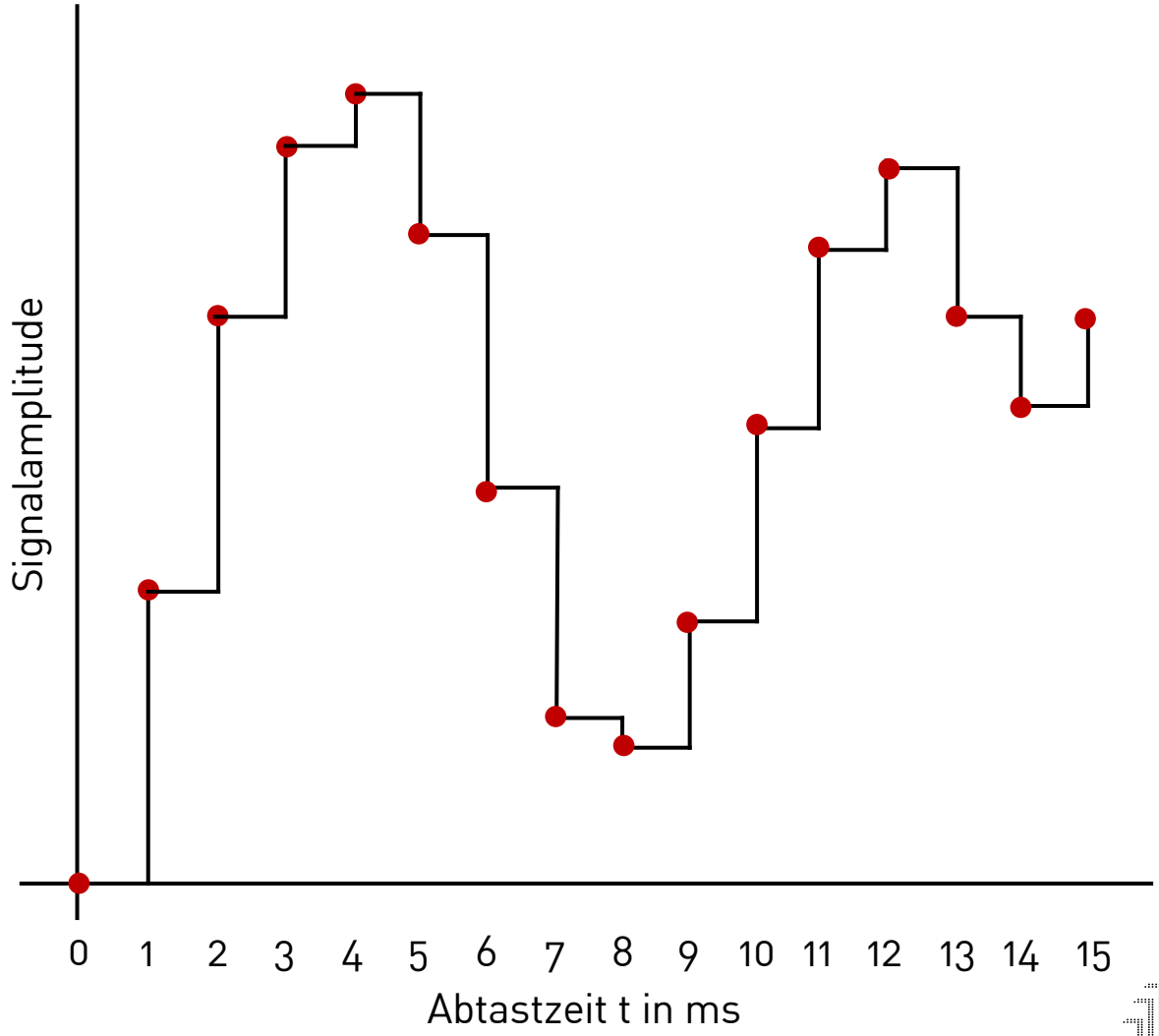
Verarbeitung analoger Signale

- **Abgetastetes Signal...**



Verarbeitung analoger Signale

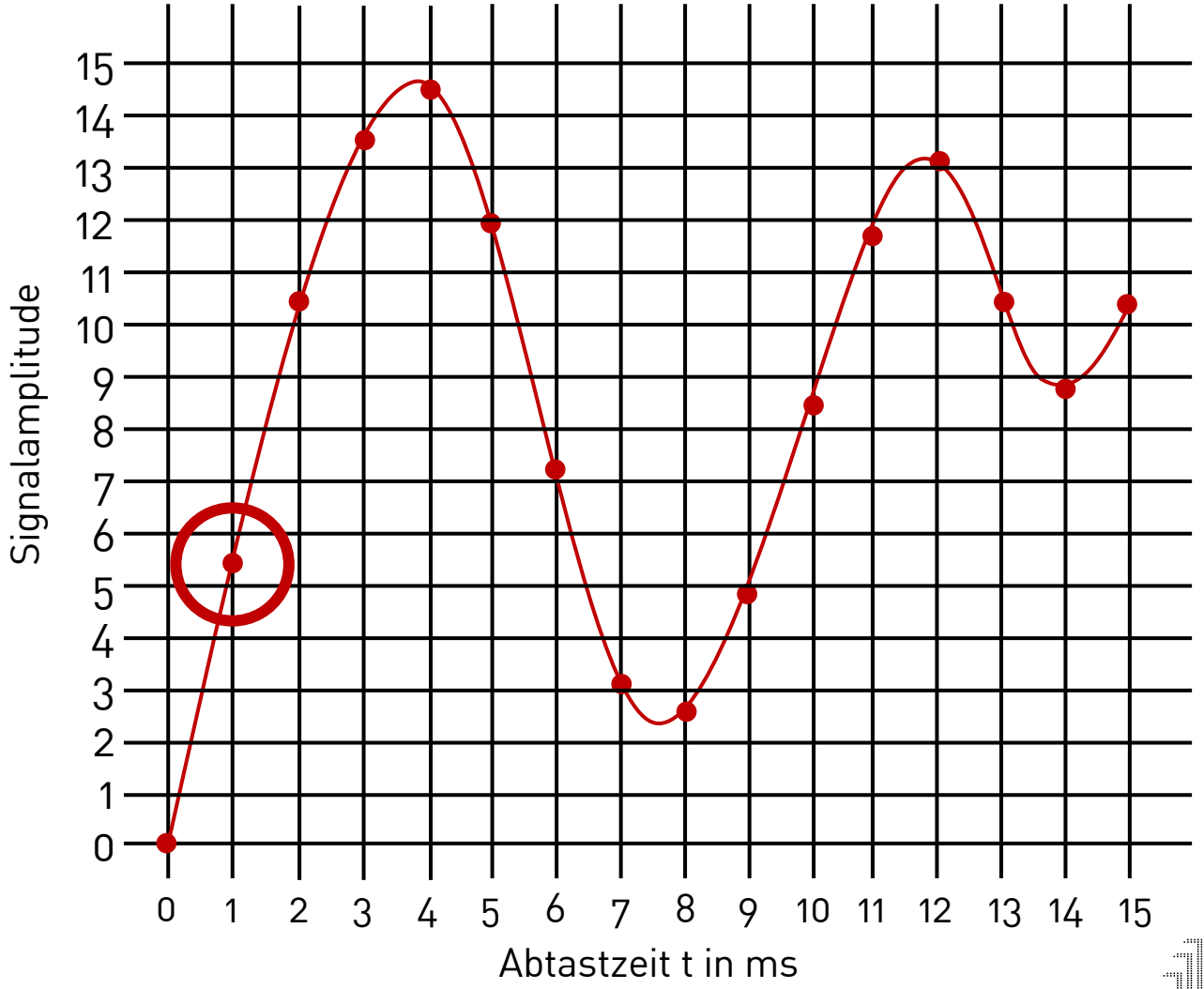
- **Abgetastetes Signal**
  - Durch die Abtastung erhält man ein zeit-diskretes (immer noch wertkontinuierliches) Signal



Verarbeitung analoger Signale

- **Abgetastetes Signal**
  - Durch die Abtastung erhält man ein zeit-diskretes (immer noch wertkontinuierliches) Signal
  - „wertkontinuierlich“ nicht darstellbar

Sample @ t1 5 oder 6 ???

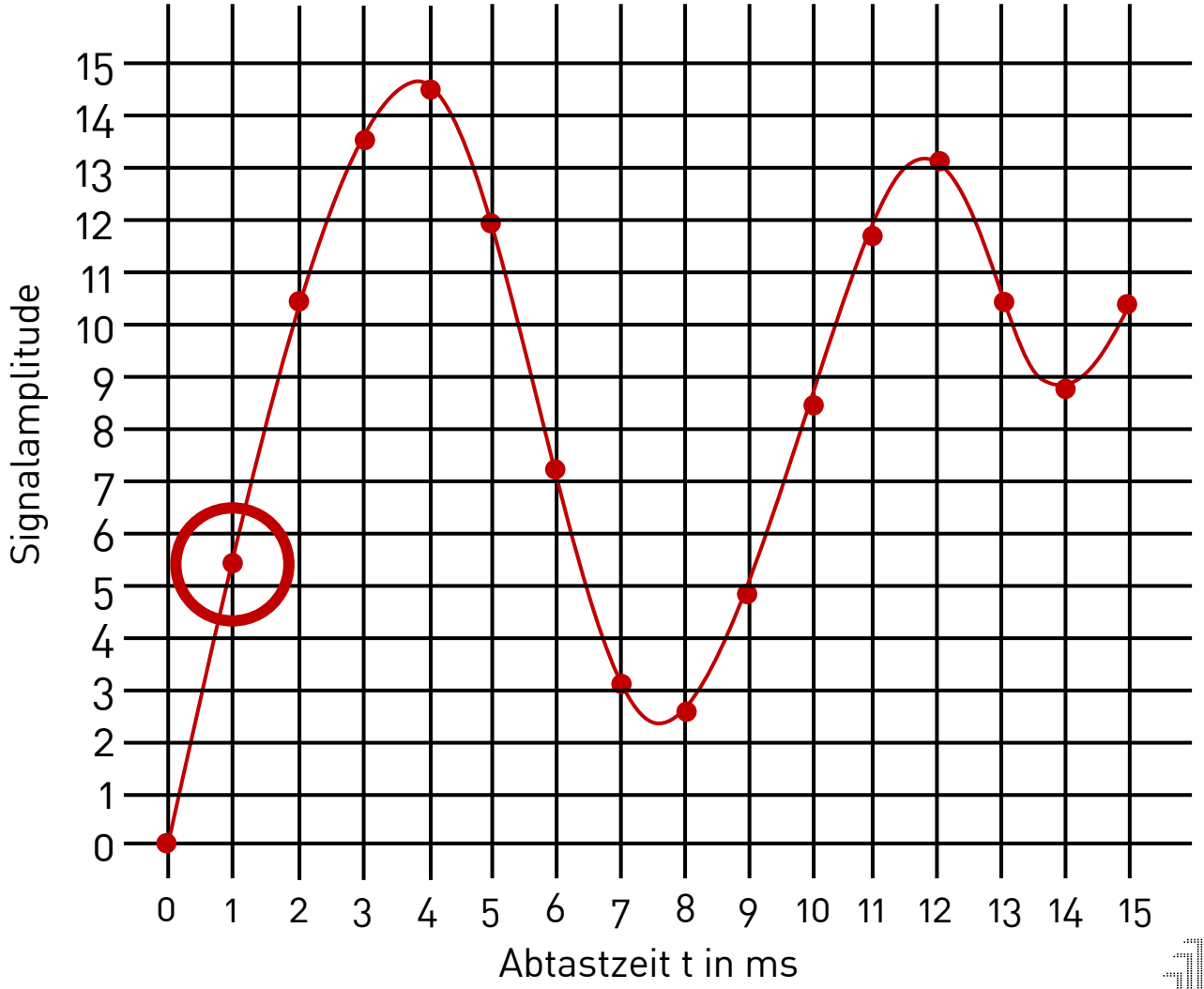


Verarbeitung analoger Signale

- **Abgetastetes Signal**
  - Durch die Abtastung erhält man ein zeit-diskretes (immer noch wertkontinuierliches) Signal
  - „wertkontinuierlich“ nicht darstellbar

Sample @ t1 5 oder 6 ???

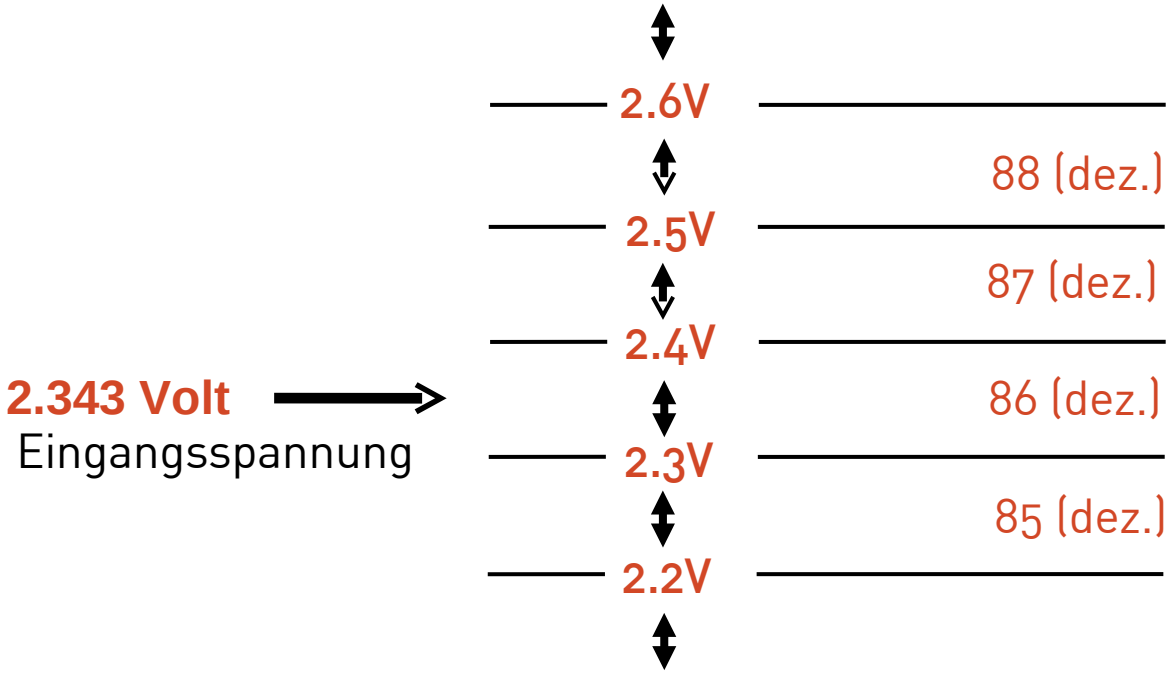
→ Amplitude eben-falls diskretisieren



Verarbeitung analoger Signale

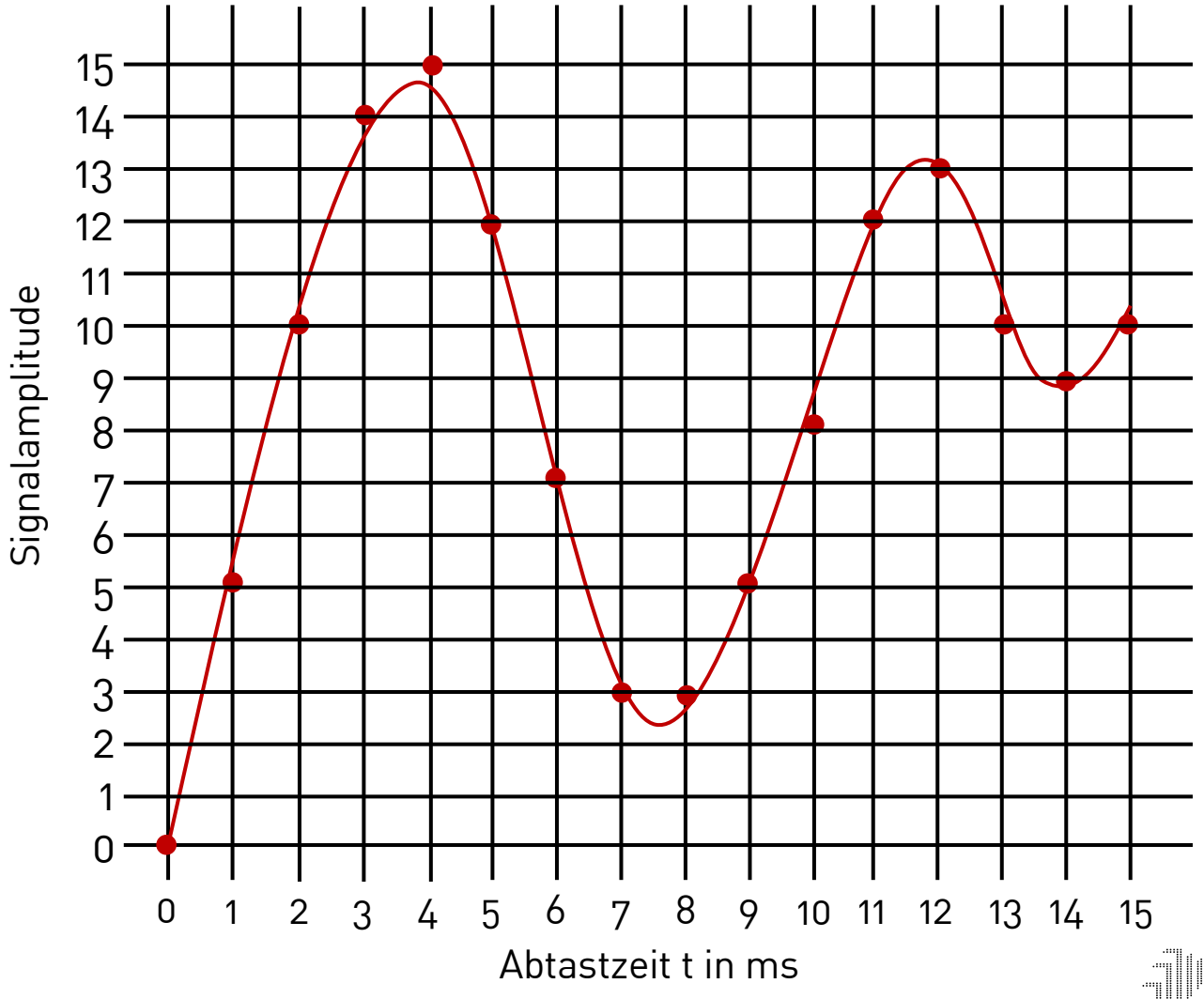
- **Quantisierung der Signalamplitude**
  - Signalamplitude wird diskretisiert

Ausschnitt: Quantisierungsstufen 8-bit Wandler



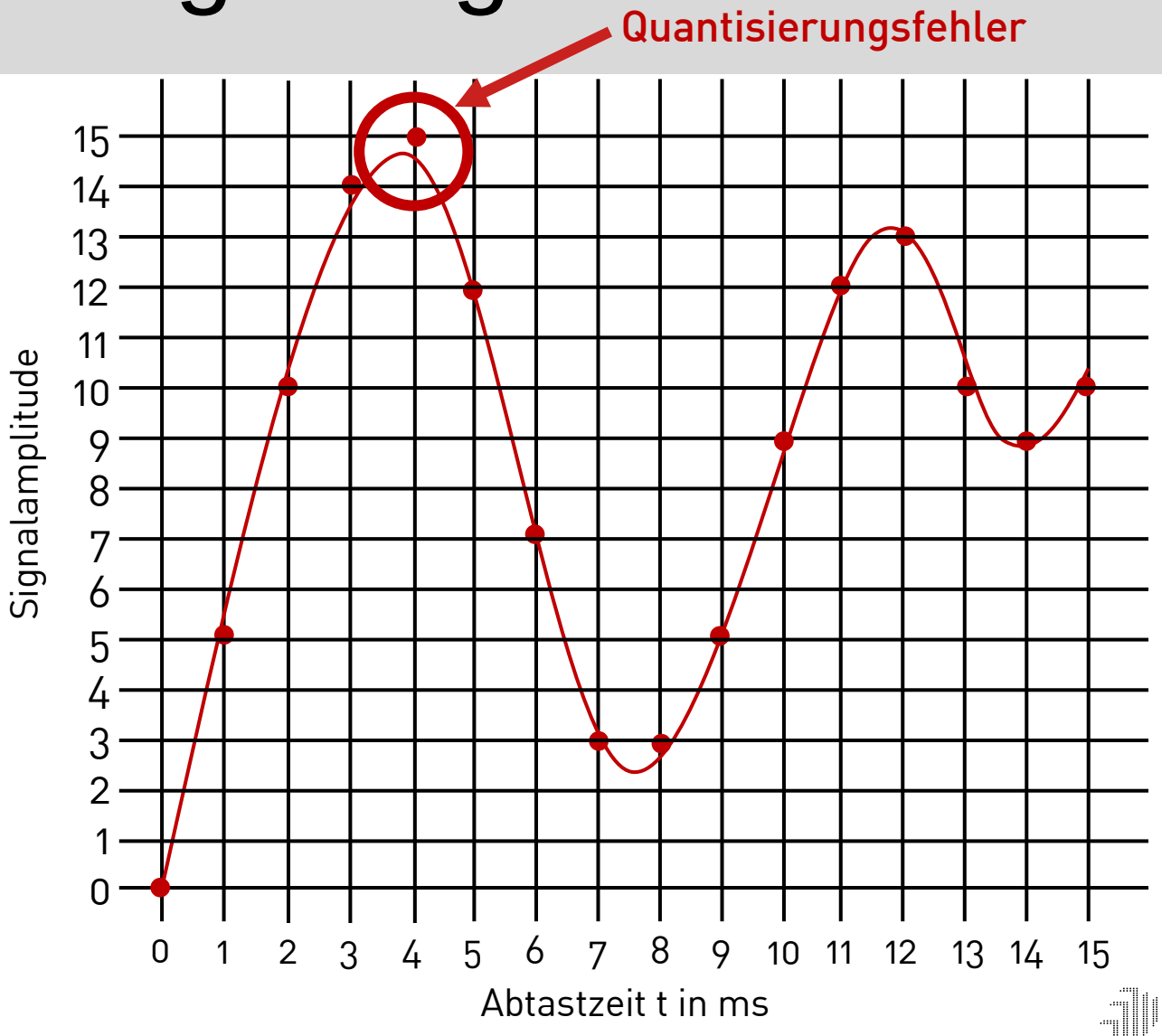
Verarbeitung analoger Signale

- **Wandlungsergebnis ist zeit- und wert-diskret**



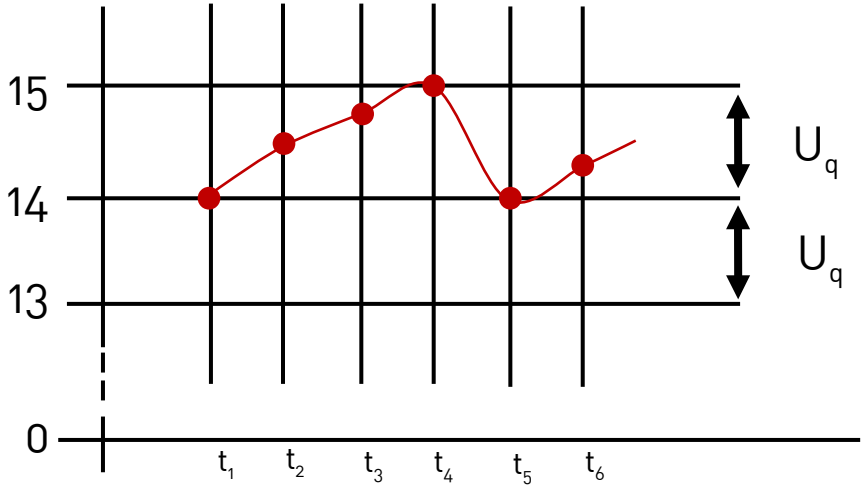
Verarbeitung analoger Signale

Sample zum Zeitpunkt t	ADC Code
15	1010
14	1001
13	1010
12	1101
11	1100
10	1001
9	0101
8	0011
7	0011
6	0111
5	1100
4	1111
3	1110
2	1010
1	0101
0	0000



Verarbeitung analoger Signale

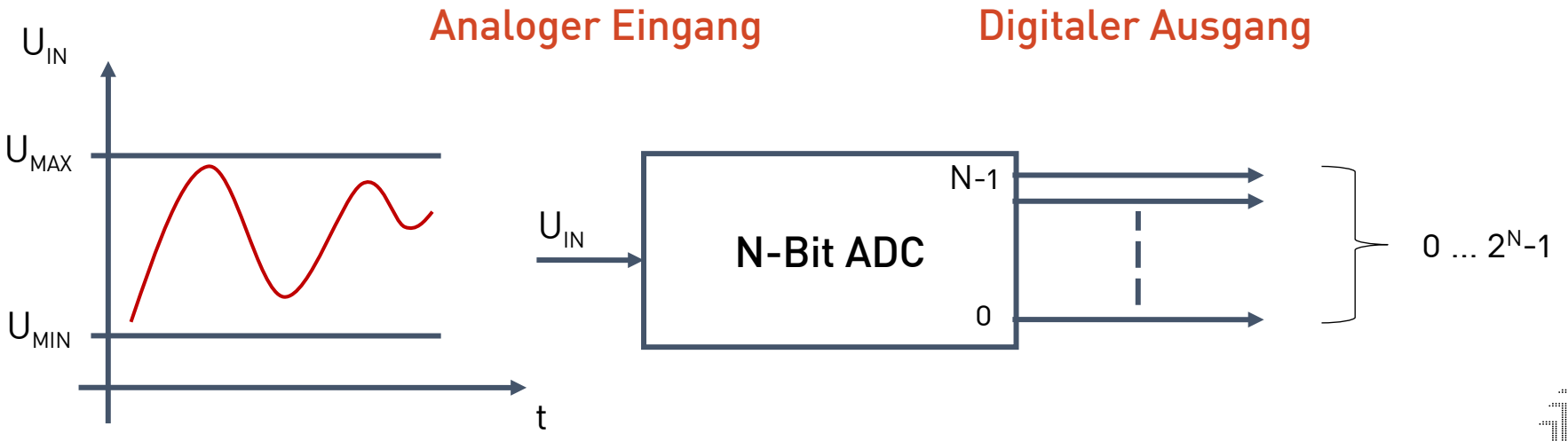
- **Quantisierungsfehler**
  - Der Quantisierungsfehler entsteht durch Runden auf eine ganze Zahl
  - Der maximale Quantisierungsfehler eines idealen AD-Wandlers liegt bei einer halben Quantisierungsstufe
  - Dieser Fall wird erreicht, wenn sich die wahre Eingangsspannung in der Mitte eines Quantisierungsintervalls (Spannungsbereich) befindet.



@t_1 → Quantisierungsfehler: **MIN** 0 U_q
@t_2 → Quantisierungsfehler: **MAX** 0.5 U_q
@t_1 → Quantisierungsfehler: 0.25 U_q
@t_1 → Quantisierungsfehler: ...
@t_1 → Quantisierungsfehler: ...
@t_1 → Quantisierungsfehler: ...

Digital-Analog Wandler

- **Zusammenfassung**
  - Ein ADC besitzt einen analogen Eingang (sogar eventuell mehrere)
  - Ein ADC besitzt N digitale Ausgänge
  - Der ADC hat somit eine Auflösung von N Bit
  - Der ADC wandelt ein analoges Signal in einen digitalen Wert zwischen 0 und 2^N-1



Digital-Analog Wandler

- **Zusammenfassung**
  - Auflösung in Bit definiert die Anzahl möglicher Ausgangszustände D
  - B – Bit → 2^N mögliche Ausgangszustände
  - N ist die Anzahl der Bits des Analog-Digital Wandlers

8-bit ADC → 256 mögliche Ausgangszustände

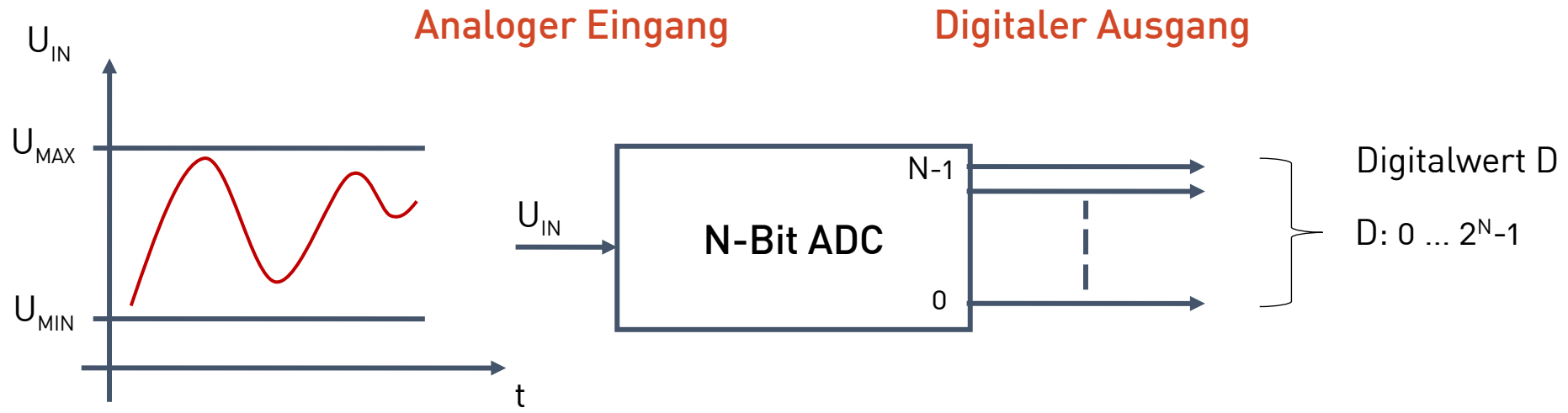
10-bit ADC → 1024 mögliche Ausgangszustände

12-bit ADC → 4096 mögliche Ausgangszustände

Digital-Analog Wandler

- **Zusammenfassung**
  - Die Umsetzung eines analogen Wertes $\mathbf{U_{IN}}$ in einen Digitalwert **D** ist eine lineare Abbildung:

$$D = \frac{U_{IN} - U_{MIN}}{U_{MAX} - U_{MIN}} (2^N - 1)$$



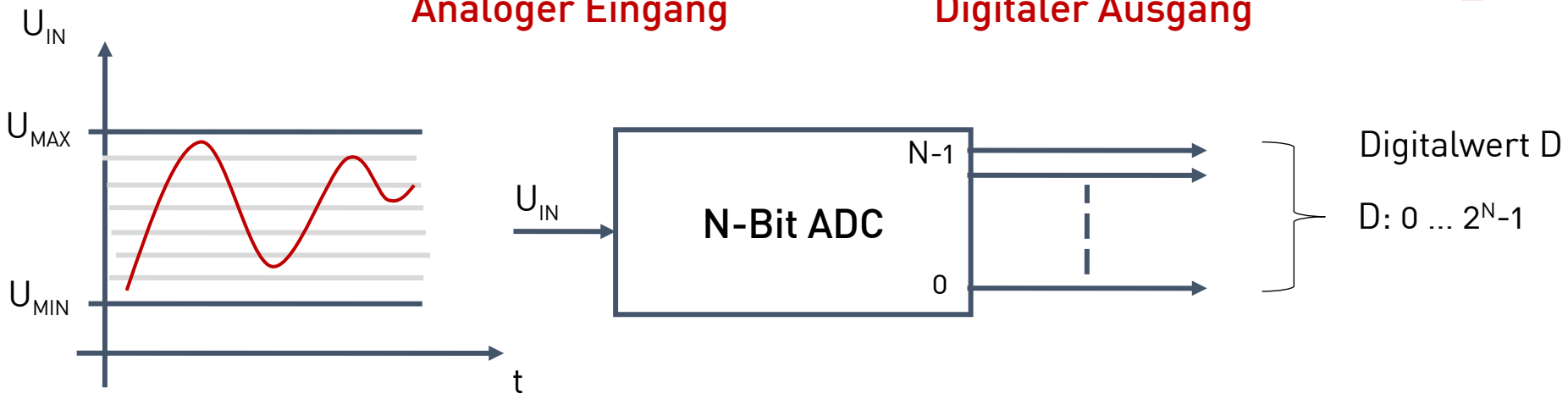
Digital-Analog Wandler

- **Zusammenfassung**
  - Die Breite einer Quantisierungsstufe U_{LSB} wird durch die Auflösung N des ADCs sowie durch den Eingangsspannungsbereich bestimmt:

$$U_q = \frac{U_{MAX} - U_{MIN}}{2^N}$$

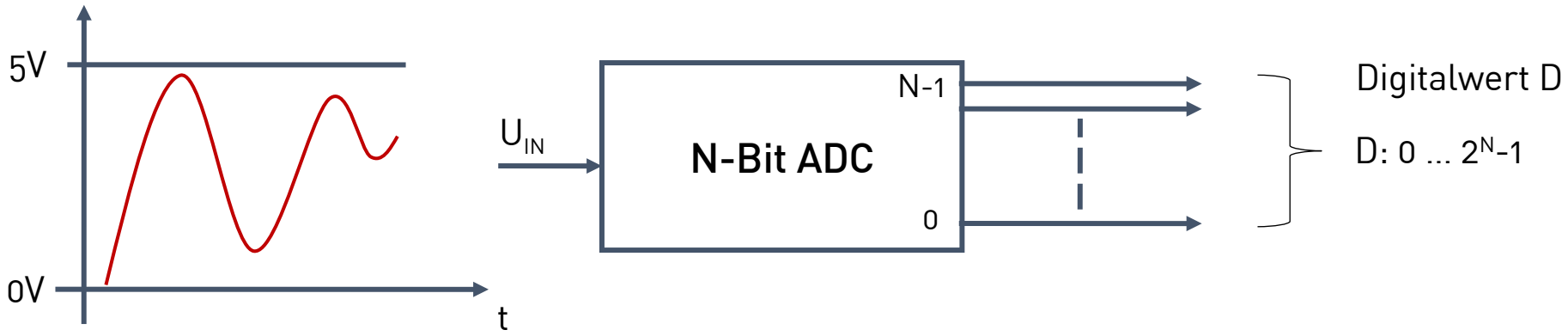
!

Analoger Eingang Digitaler Ausgang



Digital-Analog Wandler

- **Übungsaufgabe**
  - Ein Analog-Digital-Wandler besitzt eine Auflösung von 10-Bit und deckt einen Eingangsspannungsbereich von 0V – 5V ab.



  - Wie breit ist eine Quantisierungsstufe?
  - Welcher Digitalwert liefert der ADC für die Eingangsspannungen 0V, 5V, 1V, 2.5V?

Digital-Analog Wandler

- **Übungsaufgabe**
  - Mit einem Analog-digital-Wandler soll ein Eingangsspannungsbereich von 0V-3.3V abgedeckt werden.
    - Wie groß ist eine Quantisierungsstufe für den Fall eines 8-bit ADCs?
    - Wie groß ist eine Quantisierungsstufe für den Fall eines 10-bit ADCs?
    - Wie groß ist eine Quantisierungsstufe für den Fall eines 12-bit ADCs?

→ Fazit?

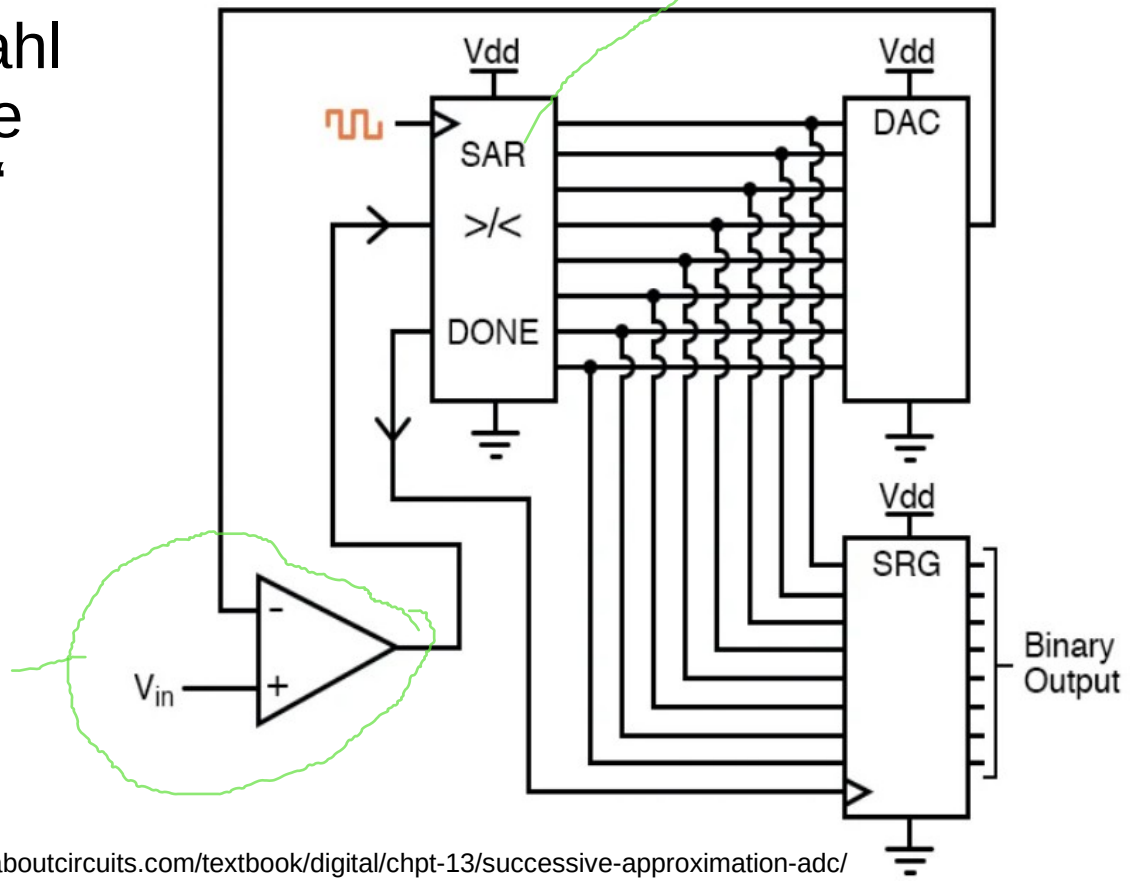
Je höher die Auflösung, desto kleiner die Quantisierungsstufe

ADC Funktionsweise

Es ist einfacher ein DAC zum bauen als vergleich zum ein ADC:

- **STM32F401RE: Successive Approximation**
  - Kinderspiel: „Rate meine Zahl zwischen 1 und 10, ich sage kleiner, größer oder Treffer“
  - Wie viele Versuche sind maximal erforderlich?

Successive Aprroximation Register und ist als 8 Bit Register gezeichnet



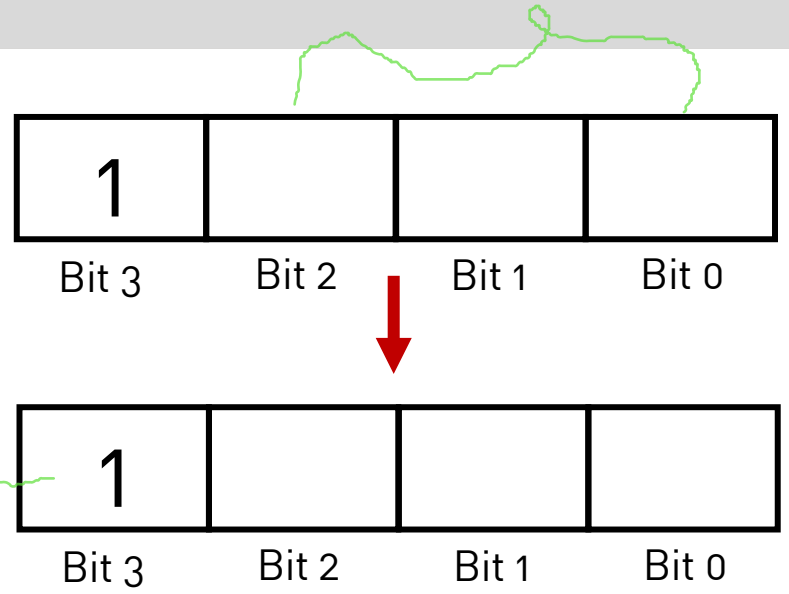
Das ist ein Komparator der vergleicht dem geratene Wert mit dem Eingangswert . Es sagt dann hinten größer oder kleiner und sagt nie ein Treffer. Je nachdem es größer oder kleiner sagt wird der Wert in SAR geändert genauso wie in der Zahlenrat Spiel

Quelle: https://www.allaboutcircuits.com/textbook/digital/chpt-13/successive-approximation-adc/

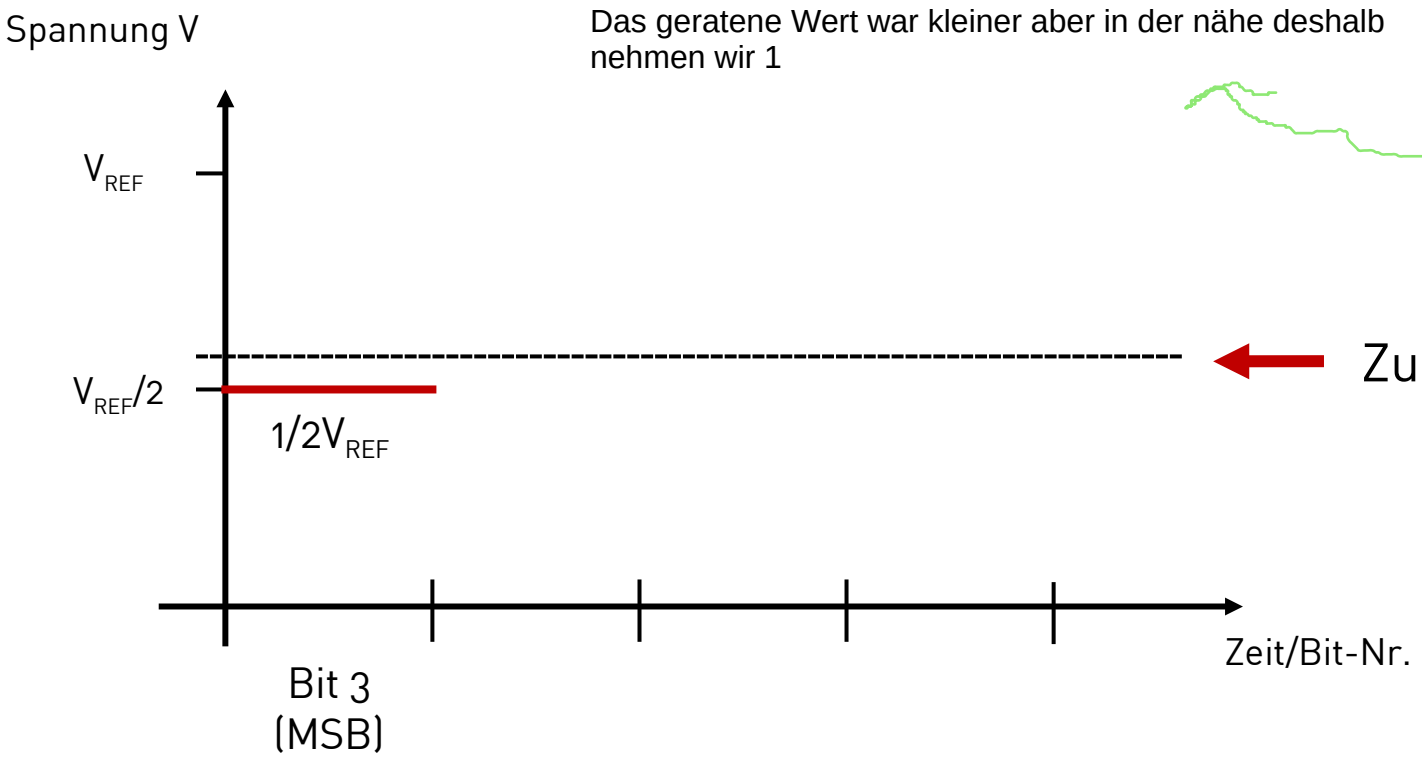
ADC Funktionsweise

Alles was hier steht ist unbestimmt. z.B 0 0 0 nehmen

- Beispiel 4-bit SAR ADC



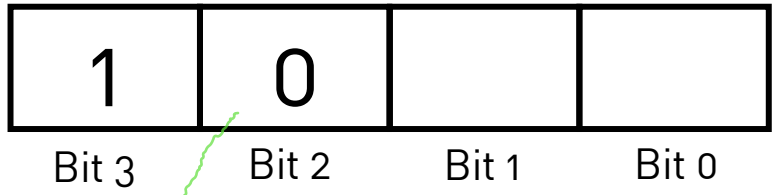
Das geratene Wert war kleiner aber in der nähe deshalb nehmen wir 1



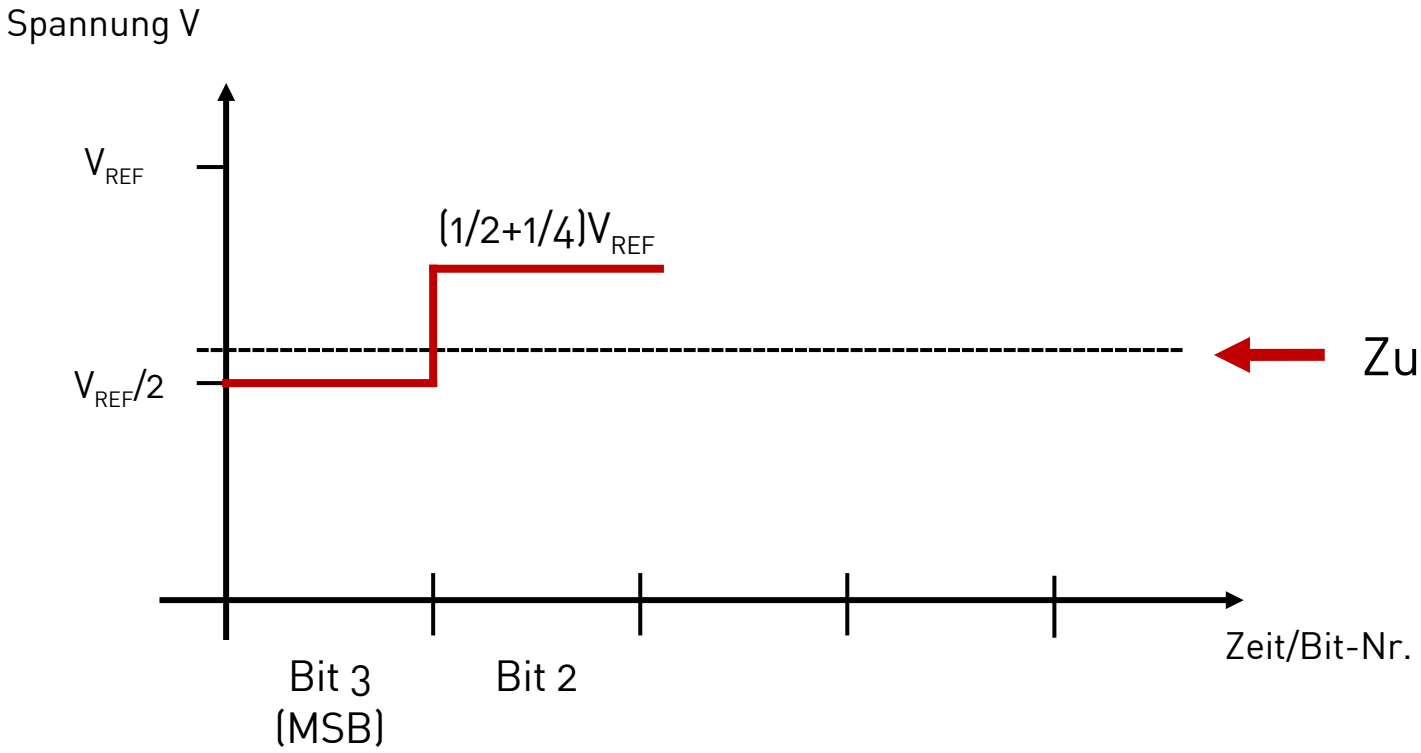
ADC Funktionsweise

- Beispiel 4-bit SAR ADC

Weil unsere Wert war kleiner deshalb müssen wir die Spannung erhöhen .
Deshalb nehmen wir wieder 1



Jetzt sagt der Komparator das ist zu viel und gibt 0

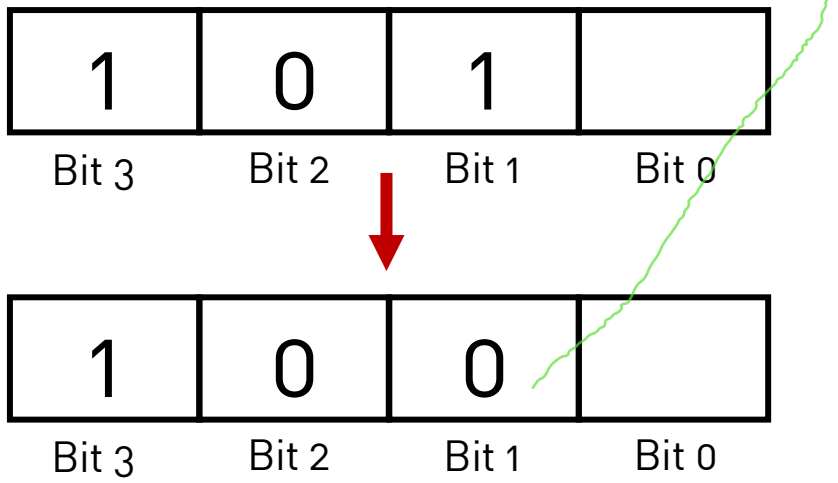


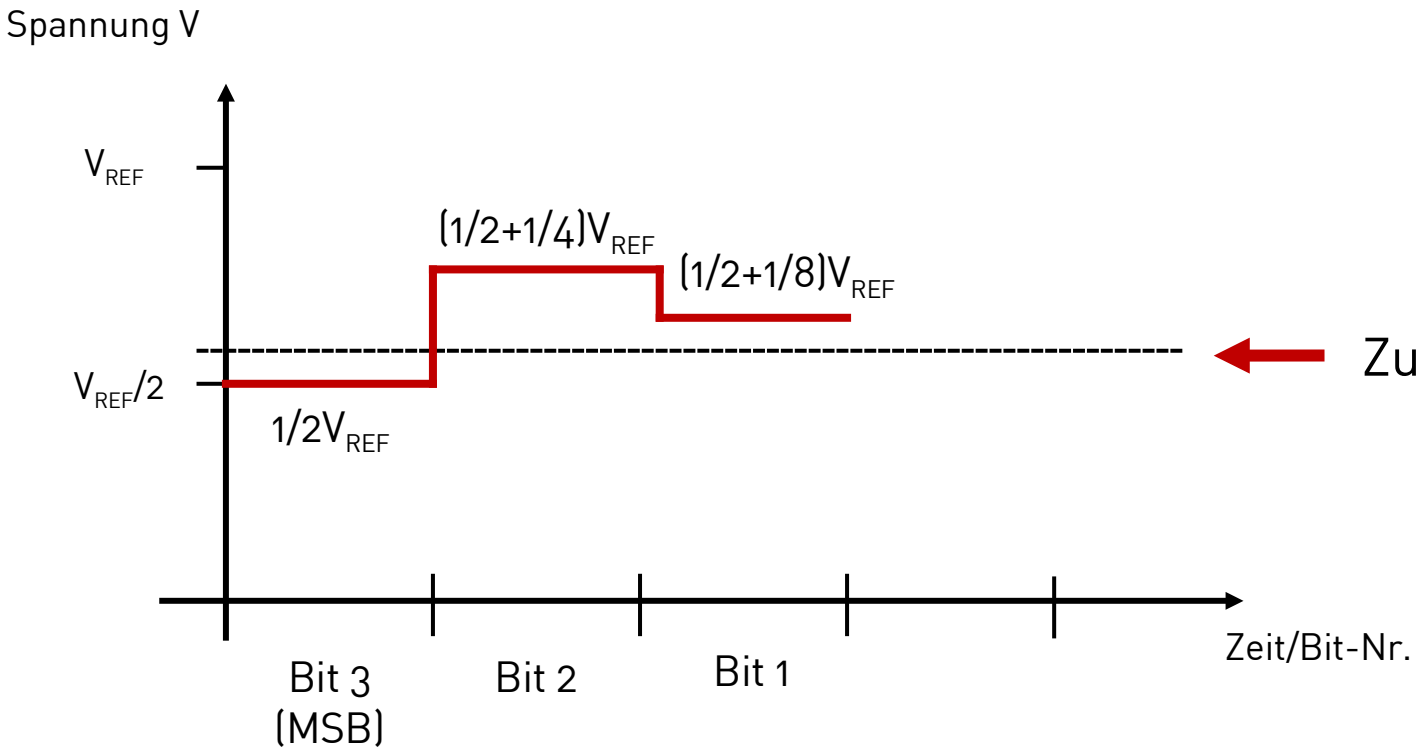
ADC Funktionsweise

Jetzt setzen wir die Wert 1 wieder und gucken ob es hoch oder klein ist

Das ist immer noch groß deshalb sagt der Komparator 0

- Beispiel 4-bit SAR ADC

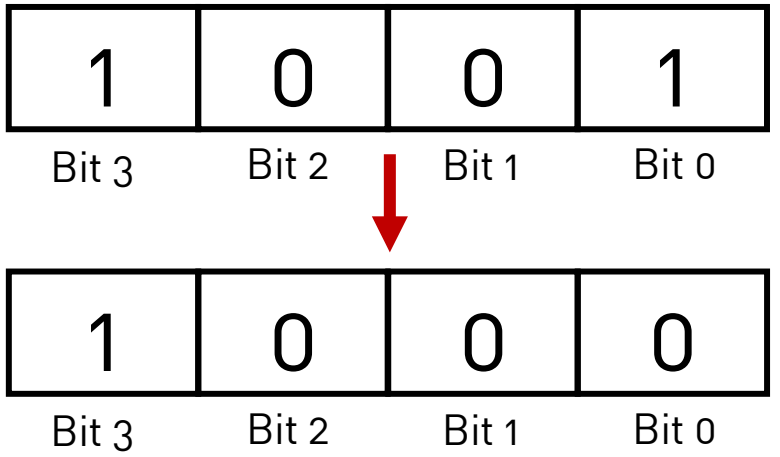


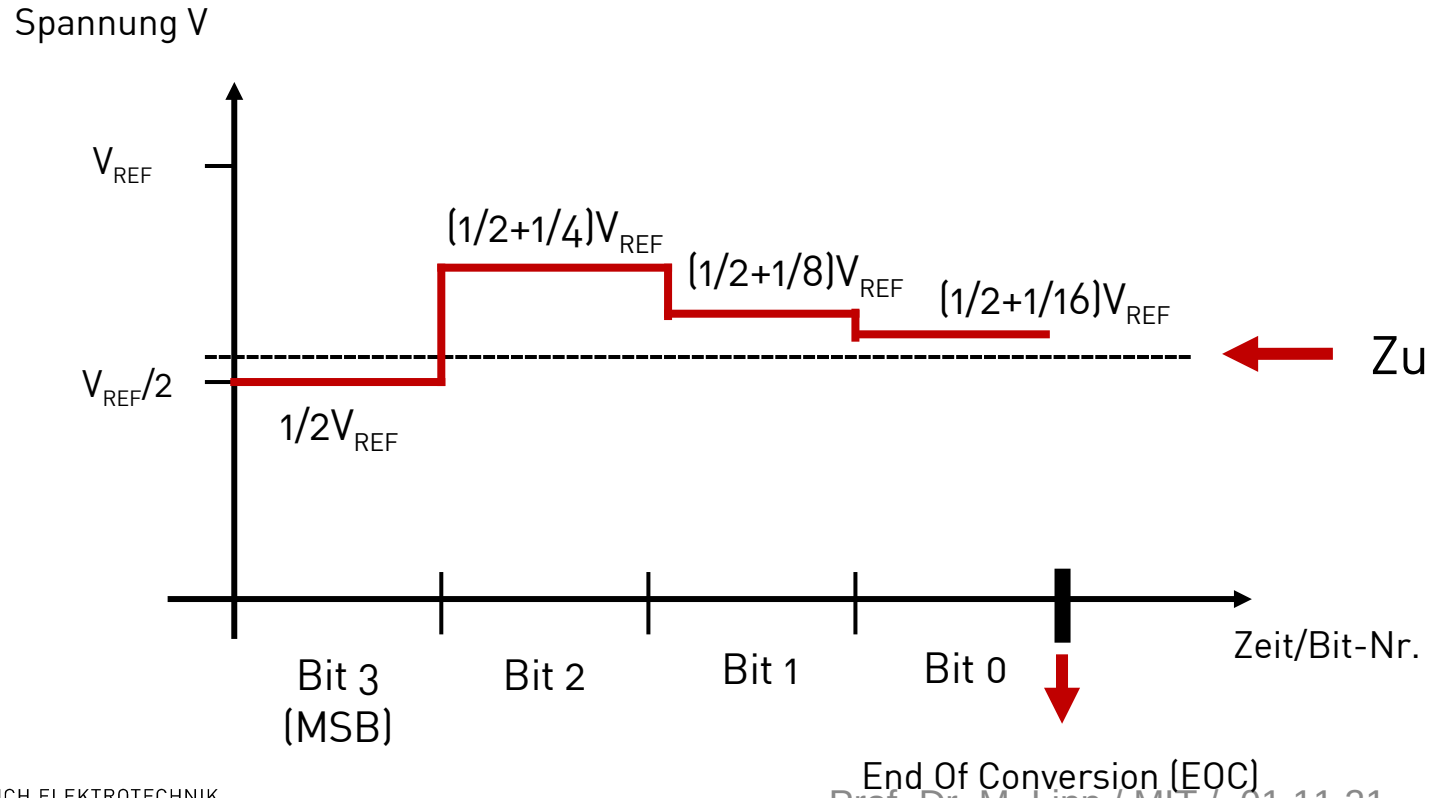


ADC Funktionsweise

Und dann raten wir wieder für die letzte Stelle als 1. Der Komparator gibt wieder 0 weil es trotzdem ein bissechen hoch ist von der bestimmende Spannung.

- Beispiel 4-bit SAR ADC





Damit ist 1000 unsere Wandlungsergebnis

Wir haben 12 Bit Ergebnis nach 12 Schritten aber hier 4 Bit deshalb ergebnis in 4 Schritte

Digital-Analog Wandler als Klasse

- **Klasse AnalogIn**
  - Konstruktor mit Pin-Name und optional `float`-Wert für Maximalwert
  - `float read()` → Wert lesen [0,0;1,0]
    - Kurzform: `operator float ()`
  - `float read_voltage()` → Wert lesen [0,0;Max]
    - Skaliert mit Maximalwert aus Konstruktoraufruf
  - `unsigned short read_u16()` → Wert lesen [0;0xffff]
- **Modelliert nur die Basisfunktion**

Digital-Analog Wandler als Klasse

- **Die Mbed-Klasse modelliert nur die Basisfunktionalität**
- **Der ADC ist (wie die GPIO-Pins) in vielerlei Hinsicht darüber hinaus konfigurierbar**
  - 12-bit, 10-bit, 8-bit oder 6-bit Auflösung
  - „Single-Shot“ oder kontinuierliche Konvertierung
  - Ausrichtung des Ergebnisses (Alignment)
  - Automatischer Wechsel des Eingangs
  - High/Low Threshold-Überwachung
  - …

Direkter Zugriff auf den ADC

- **Direkter Zugriff auf Register (analog GPIO)**
  - Beispiel ADC Control-Register 1

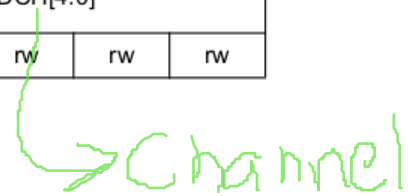
11.12.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES		AWDEN	JAWDEN	Reserved					
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw



  - Komplexer und komplizierter als bei GPIO

Hilfsmittel CMSIS

- **Cortex Microcontroller Software Interface Standard (CMSIS)**
  - Sammlung von
    - Typen (struct-Definitionen) und
    - Makros
  - Ermöglicht einfachen Zugriff auf die Register der ARM-Peripherie
    - Kenntnis des Reference Manual immer noch erforderlich!

Beispiel ADC Thresholds

Diese Werte liegen in Speicher auf einen bestimmte Adresse

- **ADC1**
  - Vordefinierter Zeiger auf `ADC_TypeDef`

```
typedef struct
{
  __IO uint32_t SR;     /*!< ADC status register, Address offset: 0x00 */
  __IO uint32_t CR1;    /*!< ADC control register 1, Address offset: 0x04 */
  __IO uint32_t CR2;    /*!< ADC control register 2, Address offset: 0x08 */
  // ...
  __IO uint32_t HTR;    /*!< ADC watchdog higher threshold register, 0x24 */
  __IO uint32_t LTR;    /*!< ADC watchdog lower threshold register,  0x28 */
  // …
} ADC_TypeDef;
```

  - Zugriff auf Register mit: `ADC1->`*`Feldname`*

Beispiel ADC Thresholds

- **Vordefinierte Bit-Masken für alle Register-Flags/-Bereiche**

```
#define ADC_SR_AWD_Pos          (0U)
#define ADC_SR_AWD_Msk          (0x1UL << ADC_SR_AWD_Pos)  /*!< 0x00000001 */
#define ADC_SR_AWD              ADC_SR_AWD_Msk             /*!<Analog watchdog flag */
// …
#define ADC_CR1_AWDCH_Pos       (0U)
#define ADC_CR1_AWDCH_Msk       (0x1FUL << ADC_CR1_AWDCH_Pos)     /*!< 0x0000001F */
#define ADC_CR1_AWDCH           ADC_CR1_AWDCH_Msk   /*!<AWDCH[4:0] bits (Analog watchdog
channel select bits) */
```

Ohne Mask ist immer den Mask

- **LED soll leuchten, wenn Limit überschritten wird**
  - Programmierung entsprechend [F401-RM]
    - Funktion: 11.3.7
    - Register: 11.12.1 (AWD), 11.12.2 (AWDEN)

Live Coding