Das Q-Nummernformat im Detail

Im Datenblock werden Zahlen im Q-Nummernformat angegeben. Dieses Format ist eine Möglichkeit Kommazahlen darzustellen, ohne dabei Kommas zu benutzen. Voraussetzung ist, dass Sender und Empfänger beide das Format einer Zahl kennen.

Man unterscheidet zwischen **signed** und **unsigned**. Eine **signed**-Zahl hat ein Vorzeichen und eine **unsigned**-Zahl nicht. Sie werden unterschiedlich beschrieben:

Notation	Bedeutung
Qm.n	Vorzeichenbehaftet (signed)
UQm.n	Vorzeichenfrei (unsigned)

Das **m** beschreibt bei der Aufstellung oben die Anzahl der Bits vor dem Komma, das **n** beschreibt die Anzahl der Bits nach dem Komma. Die Zahlen sind also binär angegeben. Ist eine Zahl **signed**, dann wird das erste Bit von **m** für das Vorzeichen reserviert:

Beispiel	Vorzeichenbit	Vorkommabits m	Nachkommabits n	Länge
Q8.8	1	7	8	2 Bytes
UQ8.8	0	8	8	2 Bytes
Q16	1	0	15	2 Bytes
UQ16	0	0	16	2 Bytes

Bei der Zusammensetzung der Zahl muss man beachten, dass m und n jeweils wie eigene binäre Zahlen behandelt werden. Bei **UQ8.8** steht vor dem Komma also eine binäre 8-Bit-Zahl und hinter dem Komma eine binäre 8-Bit-Zahl. Im Dezimalsystem würde vor dem Komma dann 1 * 8-Bit-Vorkommazahl und nach dem Komma 1 / 256 * 8-Bit-Nachkommazahl stehen.

Die Zahl hat nach dem Komma hat, ähnlich dem Dezimalsystem, eine Auflösung von $2^{\text{Anzahl Nachkommabits}}$. Das heißt bei 4 Nachkommabits kann man eine Einheit in $2^4 = 16$ Teile und bei 8 Nachkommabits in $2^8 = 256$ Teile aufteilen. Es gibt, auch wie im Dezimalsystem, als eine minimale Schrittweite, die von der Zahl der Nachkommastellen abhängt.

Notation	Beispiel	m	n	dezimal
UQ8.8	1100 1100 1010 1010	1100 1100	1010 1010	204.6640625
UQ1.15	1100 1100 1010 1010	1	(…)	1.59893798828125
UQ5.3	1100 1100	1100 1	100	25.5
UQ3.5	1100 1100	110	01100	6.375
UQ8	1100 1100	1100 1100	/	194

*bezieht sich nur auf **unsigned**

Bei einer Zahl die **signed** ist, sind für die Umformung zusätzliche Schritte notwendig. Bis jetzt bedeutet eine 8-Bit-Vorkommazahl, dass man von 0 bis 255 zählen konnte. Eine 8-Bit-Nachkommazahl teilt eine Einheit in $2^8 = 256$ Teile auf und gibt durch ihre Größe an, wie viele Teile von den 256 existieren.

Für **signed** geht das grundsätzlich gleich, nur kann man hier nicht einfach die Dezimalzahlen in Binärzahlen umrechnen und angeben. Denn negative Zahlen schreibt man im Zweierkomplement und auch das Vorzeichenbit gilt es zu beachten.

Das erste Bit steht für das Vorzeichen. Bei einer positiven Zahl ist es 0 und bei einer negativen Zahl ist es 1 (Dieser Zusammenhang macht die Umrechnung später einfacher). Der Wertebereich verschiebt sich bei Negativität jetzt von 0…255 zu -128…127, bei einer 8-Bit-Vorkommazahl. Da man nur bis 128 in eine Richtung zählen muss reichen verbleibenden 7 Bit aus, weil ja ein Bit für das Vorzeichen reserviert ist.

Bei positiven Zahlen steht im Vorzeichen eine 0. Mit den verbleibenden Bits zählt man dann bis maximal 127 im positiven (ohne Zweierkomplement, wie bei **unsigned**). Durch normales Weiterzählen würde das **Vorzeichenbit** jetzt 1. Hier beginnen die negativen Zahlen.

Nach der 127 (**0**111 1111) kommt die 128 (**1**000 0000). Da sie jetzt negativ sein soll, sagt man -128. Jetzt kommt die -127 (1000 0001) usw. Man zählt also im negativen rückwärts. In dieser Betrachtung wurde das Vorzeichenbit mit als Teil der Zahl gewertet. Es erklärt sich nun, warum die 1 Negativität anzeigt und nicht die 0.

Will man eine negative Vorkomma-Dezimalzahl ins Binäre umrechnen geht man so vor:

- Dezimalzahl binär aufschreiben (z.B. -16, 001 0000)
- 1 addieren, wenn der Nachkommateil 0 ist
- Bits invertieren (aus 0001 0000 wird 1110 1111)

Will man eine negative Vorkomma-Binärzahl ins Dezimale überführen:

- Binärzahl aufschreiben (z.B. 1110 1111, 239)
- 1 subtrahieren, wenn der Nachkommateil 0 ist
- Bits invertieren und umrechnen (aus 1110 1111 wird 0001 0000)

Da man negativ rückwärts zählt klappt das nur, wenn hinter dem Komma keine 0 steht. Falls doch würde 1110 1111 einer -17 entsprechen. Steht also im Nachkommateil eine 0, muss man vor der Umrechnung 1 addieren oder subtrahieren. Hinter dem Komma spielt das Vorzeichen keine Rolle, man muss aber trotzdem invertieren um das Rückwärtszählen auszugleichen.

Will man eine negative Nachkomma-Dezimalzahl ins Binäre umrechnen geht man so vor:

- Multipliziert sie mit 256 (für 8-Bit-Nachkommazahl, ansonsten $2^{\text{Anzahl Nachkommabits}}$),
- Rundet man das Ergebnis und subtrahiert es von 256,
- Und wandelt das Ergebnis in eine Binärzahl um.

In umgekehrter Richtung geht so auch die Umwandlung von Binär nach Dezimal.

Notation	Beispiel	m	n	dezimal
Q8.8	1100 1100 1010 1010	1100 1100	1010 1010	-51.3359375
Q8.8	0100 1100 1010 1010	0100 1100	1010 1010	76.6640625
Q1.15	1100 1100 1010 1010	1	(…)	-0.40106201171875
Q3.5	1010 1010	101	0 1010	-2.3125
Q8	1010 1010	/	1010 1010	0.33203125

*Gibt es keine Vorkommastelle gilt die Nachkommastelle als Solche also von -0.5…0.5