# Radio over Ethernet (RoE) base protocol

## Overview

## RoE Ethernet Type

## Bit and octet ordering, and numerical presentation

## RoE common frame format

### ver (version) field

### pkt\_type (packet type) field

### flow\_id (flow identifier) field

### ordering indicator field

#### timestamp

The **timestamp** is 32 bits in size and in units of nanoseconds. The timestamp is the presentation time at the RoE packet receiving endpoint and calculated by the RoE packet sending endpoint. Both endpoints shall share the same understanding of the Time of Day (ToD).

The timestamp field is encoded as a 32 bit sliding window capable of representing ~2 seconds worth of time. This implies the timestamp field is capable of encoding a presentation time maximum ~1 second in the future. See Annex B for an example algorithm. The timestamp sliding window size is controlled by the following variables:

1. **tstampWindowSize** = “size of the sliding window”; the value shall be a power of 2
2. **tstampWindowMask** = **tstampWindowSize**-1
3. **tstampTstampMask** = (**tstampWindowSize**\*2)-1

Refer to subclause 4.11 for more details on the timestamp and the presentation time.

#### sequence number

[///Editor’s note: NEW TEXT HERE]

~~The sequence number field is 31 bits in size and wraps to seqNumMinimum after exceeding its maximum value seqNumMaximum-1. The highest value for the seqNumMaximum is 2^31-1. The following shall hold: 0≤seqNumMinimum<seqNumMaximum-1. The sequence number is increased by a constant value seqNumIncrement known by both RoE packet sending and receiving endpoint. The seqNumIncrement shall comply with: seqNumIncrement<(seqNumMaximum- seqNumMinimum-1).~~

~~The sequence number is initialized to an implementation specific value seqNumStart between seqNumMinimum and seqNumMaximum-1 at the endpoint reset. The internal structure of the sequence number is known and interpreted by RoE endpoints~~.

The sequence number constitutes of three independently sized fields, whose sizes depend on two variables p and q. Figure 5 illustrates the composition of different fields that together form the sequence number. The following rules shall apply:

0 <= p < 32 and 0 <= q <= p and (32-p)+(p-q) <= 32

The field from p to 32 is the p-counter field and the field from q to p is the q-counter field. At minimum the p-counter field shall exist. The other two fields may exist based on the p and q values. Each of the fields is initialized to a known value on the first use of the sequence number. Note that the seqNumPMax, seqNumPIncProp, seqNumQMax, seqNumQInc, seqNumQIncProp, and seqNumRsvd values shall remain unchanged during the lifetime of the RoE flow.

****Figure 5 – Sequence number composition

The following variables listed in Table 4 are used to describe the initial values, behavior and number space wrap properties of the sequence number.

Table 4 - Sequnce number related variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Bits | Default value | Description |
| seqNumPMax | 32 | 0xFFFFFFFF | Maximum numerical value of the p-counter field so that **seqNumPMax** < 1 << 32-p. |
| seqNumPVal | 32 | 0x00000000 | The counter value of the p-counter field. When the seqNumPVal > seqNumPMax then the counter field wraps to a value seqNumPVal modulo (seqNumPMax+1). The counter wrap also causes the seqNumQVal to be incremented by seqNumQInc based on the seqNumQIncProp setting. |
| seqNumPInc | 32 | Undefined | The increment value that is dependent on the seqNumPIncProp setting. |
| seqNumPIncProp | 3 | 0x1 | * 0x0 no increment.
* 0x1 increment by a constant on every sent packet.
* 0x2 Increment by a packet payload size on every sent packet.
* 0x3 to 0x7 reserved.
 |
| seqNumQMax | 32 | 0x00000000 | Maximum numerical value of the q-counter field so that **seqNumQMax** < 1 << p-q. |
| seqNumQVal | 32 | 0x00000000 | The counter value of the q-counter field. When the seqNumQVal > seqNumQMax then the counter field wraps to a value seqNumQVal modulo (seqNumQMax+1). |
| seqNumQInc | 32 | Undefined | The increment value that is dependent on the seqNumQIncProp setting. |
| seqNumQIncProp | 3 | 0x1 | * 0x0 no increment.
* 0x1 increment by a constant on seqNumPVal wrap.
* 0x2 to 0x7 reserved.
 |
| seqNumRsvd | 32 | 0x00000000 | Static value for reserved field, if available. Shall remain unchanged for the lifetime of the flow. |

**Annex C. Sequence number pseudo code**

typedef struct {

 uint32\_t pval;

 uint32\_t qval;

 uint32\_t pmax;

 uint32\_t qmax;

 uint32\_t rsvd;

 uint8\_t q, p;

} seqnum\_t;

// Initiaze the sequence number

int initSeqNum( seqnum\_t\* sn,

 uint32\_t pmax, uint32\_t qmax,

 uint32\_t pval, uint32\_t qval,

 uint32\_t rsvd ) {

 // sanity checks

 if (pval >= pmax) return -1;

 if (qmax == 0 && qval != 0) return -1;

 // initialize sequence number..

 sn->pmax = pmax;

 sn->qmax = qmax;

 sn->pval = pval;

 sn->qval = qval;

 sn->p = pmax > 0 ? 1 : 0;

 sn->q = qmax > 0 ? 1 : 0;

 // calculate q and p

 while (pmax >>= 1) sn->p++;

 while (qmax >>= 1) sn->q++;

 // more sanity checks

 if (sn->p == 0) return -1;

 if (sn->p + sn->q > 32) return -1;

 // remaining initialization

 sn->rsvd = sn->q + sn->p >= 32 ? 0 : rsvd << sn->q + sn->p;

 return 0;

}

// Construct 32 bit sequence number from counter fields

uint32\_t getSeqNum( const seqnum\_t\* sn ) {

 return sn->rsvd | sn->qval << sn->p | sn->pval;

}

// Increase sequence number counter fields(sender side operation)

uint32\_t incSeqNum( seqnum\_t\* sn, uint32\_t pinc, uint32\_t qinc ) {

 uint32\_t oldpval = sn->pval;

 sn->pval += pinc;

 if (sn->pval > sn->pmax || sn->pval < oldpval) {

 sn->pval = sn->pval - sn->pmax - 1;

 sn->qval += qinc;

 }

 if (sn->qval > sn->qmax) {

 sn->qval = sn->qval - sn->qmax - 1;

 }

 return getSeqNum( sn );

}