# KEN PROTOCOL COMPARISON

By: Ken McCaughey
On: 2025-03-21

Ken's Electronic Network (KEN) Advanced Protocol

This serial communications protocol is intended for small micro-controller applications. This data link layer protocol specifies frame elements that define a variety of features. This protocol does not attempt to define all the rules in how they should or could be used. There are considerable permutations and it is up to the user to decide what fits a given application. Not all permutations that can be created are compatible with each other. The intent of this protocol is to have a set of tools and a structure to facilitate communications to meet a particular application. Only implement what is needed for a given application and tailor as needed.

There are three versions that are incompatible with each other. However, they share a basic architecture and concepts. This allows portions of code written for one flavor to be reused in another.

Why have three versions? In small systems size can matter. So can implementation complexity. Most applications will not use most of the available features. The additional variations scale down the overhead and some of the capabilities. They all share some basic principles making some code reuse possible. The variations offer choices for a best fit for a given application.

Below is a high level discussion comparing the variations. This is followed by a table to highlight some technical differences. For reference the basic frame structure for each variation is described.

Reference the individual protocol specification documents for complete details on each variation.

## KEN-A

KEN-A is an Advanced feature rich and has a lot of flexibility. It does this at the cost of frame overhead. It is designed for primarily for ASCII payload data with limited binary data capabilities. All headers and flags are binary (msb=1), making them easier to identify and parse. It also has features to avoid having to keep track of the length of the frame contents, thus permitting very long frames. It has features to allow for a lot of customization.

## KEN-B

KEN-B is a more Basic version of KEN-A. It uses many of the same principles as KEN-A with regards to the header. It does not rely on flags in the same way. Since there is no unique start or end of frame flags, frames must be separated by a break in time. End of frame is determined by knowing the frame length. KEN-B frames start with a one byte Frame Length (FL). The FL is always followed by the Header Config Byte (HCB), which defines the rest of the header contents and size. The header elements are all binary (msb=1) and use the same prefixes as in KEN-A, thus can be parsed same way. This also makes it easier to visually inspect with a hex viewer. Most header elements are all limited to one byte each, which in some cases limits a capability. Since the HCB defines a header portion of the frame, the start of the payload data is easily determined. There are no issues with binary in the data payload. The data contents can be ASCII, binary, or a mixture. However, since all the header elements are binary, if the payload is all ASCII it is easy to parse based on msb. The size of the data payload is a function of the maximum frame size minus the number of of header bytes and checksum method, if used.

## KEN-C

KEN-C is a more Compact version of KEN-B. This has a compressed header of fixed features and fixed size of 5-bytes. All features are nibble based, and thus are limited in capabilities. As in KEN-B, the frame contents are position dependent and rely on knowing the frame length. Since the header size is fixed, the start of the payload data is easily determined. There are no issues with binary in the data payload. The data contents can be ASCII, binary, or a mixture. Note that due to the nibble wise header compression, the header can contain both binary and ASCII. The header must be parsed based on frame position. The size of the data payload is a function of the maximum frame size minus the number of of header bytes and checksum method, if used.

Below is a summary comparison table of differences.

| Protocol Feature | KEN-A | KEN-B | KEN-C |
|---|---|---|---|
| Current version | 1.1.0 | 1.1.0 | 1.1.0 |
| Hex viewer human readable frames | Yes | Yes | Not really |
| Maximum frame size | Unlimited | 127-bytes | 127-bytes |
| Unique start/end frame flags | Yes | No | No |
| Uses a header flag | Start/end | Yes, HCB | No |
| Frame flags | Yes | Limited | No |
| Requires frame breaks | No | Yes | Yes |
| Header size | Variable | Variable | Fixed |
| Minimum header size | 2-bytes | 2-bytes | 5-bytes |
| Maximum header size (minus check) | ~20-bytes | 13-bytes | 5-bytes |
| Headers elements have an ID flag | Yes | Yes | None |
| Headers element flags msb=1 | Always | Always | N/A |
| Position dependent header elements | Some | All | All |
| Allows for binary payload data | Limited | Yes | Yes |
| Requires length information | No | Yes | Yes |
| Length parameter covers... | Data only | Frame | Frame |
| Headers reduce payload data size | No | Yes | Yes |
| Maximum ASCII payload data w/o check | Unlimited | 125-bytes | 122-bytes |
| Maximum ASCII payload data w check | Unlimited | 122-bytes | 120-bytes |
| Maximum binary payload data | 127-bytes | 125-bytes | 122-bytes |
| Maximum addresses | 127 | 15 | 15 |
| Maximum sub-frames | 127 | 15 | 15 |
| Allows custom features | Yes | No | No |

## KEN-A FRAME FORMAT

Note that most significant byte is abbreviated as MSB (upper case) and most significant bit is msb (lower case). Similar for least significant byte or bits, LSB and lsb respectively. All header elements have MSB=1 to distinguish from data with MSB=0.

All message frames begin with `0xFB` and end with `0xFE` flags. All other used header elements (`0x8n` to `0xEn`) are generally in numerical order before the data flag. The checksum, if used, is at the end prior to the end of message flag. Some header parameters may be followed by ASCII variables.

Frame highlights:

- `++` Frame flags, most optional (hexadecimal)
- `**` Header control elements, optional (hexadecimal)
- `--` Flag or header ASCII component
- `==` Frame payload data
- `~~` Data covered by checksum

```
Flag [Header/Control Elements] Flag [Data] Flag [Check] Flag
++++ ************************ ++++ ====== ++++ ------- ++++
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

FB 8n 9n An Bn Cn Dn En FD [Data] FC [Check] FE
++ ** ** ** ** ** ** ** ++ ====== ++ ------- ++
|  |  |  |  |  |  |  |  |  |      |  |       |
|  |  |  |  |  |  |  |  |  |      |  |       +--end of frame flag *
|  |  |  |  |  |  |  |  |  |      |  +--checksum
|  |  |  |  |  |  |  |  |  |      +-----start check flag
|  |  |  |  |  |  |  |  |  +--ASCII data
|  |  |  |  |  |  |  |  +-----start of data flag #
|  |  |  |  |  |  |  +--error control
|  |  |  |  |  |  +-----data length
|  |  |  |  |  +--------connection control
|  |  |  |  +--to address
|  |  |  +-----from address
|  |  +--sequence number
|  +-----checksum type (start of checksummed characters)
+--beginning of frame flag *

 * required frame element
 # there are a different flags for different data types
```

## KEN-B FRAME FORMAT

All message frames begin with a Frame Length (FL) byte with the msb = 1. This is followed by the Header Config Byte (HCB). The HCB defines which protocol features being implemented and thus scales the size of the header. Features used are defined by bit-wise flags in the HCB. All used header elements (`8n` to `Fn`) are in a fixed order followed by the data and optional checksum values. Some frame flags may be followed by variables.

Frame highlights:

- `++` Header, required (hexadecimal)
- `**` Header control elements, optional (hexadecimal)
- `--` Flag or header ASCII component
- `==` Frame payload data
- `~~` Data covered by checksum

```
Frame Len HCB [Header/Control Elements] [Data] [Check]
+++++++++ +++ ************************* ====== -------
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

FL HC 8n 9n An Bn Cn En Ff [Data] [Check]
++ ++ ** ** ** ** ** ** ** ====== -------
|  |  |  |  |  |  |  |  |  |      |
|  |  |  |  |  |  |  |  |  |      +--checksum (up to 2-bytes) #
|  |  |  |  |  |  |  |  |  +--payload data
|  |  |  |  |  |  |  |  +--frame flags #
|  |  |  |  |  |  |  +--error control
|  |  |  |  |  |  +-----connection control
|  |  |  |  |  +--to address
|  |  |  |  +-----from address
|  |  |  +--sequence number
|  |  +-----checksum type
|  +--Header Config Byte (HCB) *
+-----Frame Length (FL) *

 * required frame element
 # optional, size depends on checksum type
```

## KEN-C FRAME FORMAT

All message frames begin with a Frame Length (FL) byte with msb = 1. The remaining four header control bytes use are required and the order is fixed. The FL is of the entire frame, including any checksum data. Therefore the maximum frame size is 127 bytes. Payload data can range from 0 bytes to 122, with no checksum, and 120 with a CRC-16 checksum.

Frame highlights:

- `++` Frame length (FL)
- `**` Header control elements (hexadecimal)
- `--` Flag or header ASCII component
- `==` Frame payload data
- `~~` Data covered by checksum

```
Framea Len [Header/Control Elements]  [Data] [Check]
++++++++++ ************************  ====== ------
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1  2  3  4  5 <-- Header Bytes
|  |  |  |  |
FL 89 AB CE Ff [Data] [Check]
++ ** ** ** ** ====== -------
|  || || || |  |      |
|  || || || |  |      +--checksum (up to 2-bytes) #
|  || || || |  +--payload data
|  || || || +--frame numbers
|  || || |+--error control
|  || || +---connection control
|  || |+--to address
|  || +---from address
|  |+--sequence number
|  +---checksum type
+--frame length (FL)

 # optional, size depends on checksum type
```

# END OF DOCUMENT