KEN-D PROTOCOL

By: Ken McCaughey
On: 2025-07-26
Ver 1.0.0

Ken's Electronic Network (KEN) Data Protocol

This protocol is derived from the original Ken's Electronic Network (KEN) Protocol. It is a very minimalist version that trades features for absolute simplicity. While this protocol borrows some features from KEN-C, is not compatible with it (KEN-A != KEN-B != KEN-C != KEN-D).

This serial communications protocol is intended for small micro-controller applications. This data link layer protocol specifies frame elements that define a minimalist features. This protocol does not attempt to define all the rules in how they should or could be used. The intent of this protocol is to have a set of tools and a structure to facilitate communications to meet a particular application.

Universal compatibility is not the goal of this protocol. Ease of use and implementation is the goal.

Contents

FEATURES

- Intended for small micro-controllers.
- Good for point-to-point messaging as well as small networks.
- Derived from the original Ken Electronic Network (KEN) Protocol.
- Features very compact header.
    - Trades features and flexibility for link overhead efficiency.
    - Header components are fixed.
- All control features in a header are position dependent.
    - Nibble based with limited capabilities.
- Message sizes scalable based on complexity and options used.
    - Can be as short as 3-bytes.
        - One data byte.
    - Maximum frame length limited to 127 bytes.
        - Fixed 2-byte overhead.
        - Maximum payload data of 125 bytes.
- Supports addressing (required).
    - Up to 15 nodes.

NOT INCLUDED

These are the things NOT included or specified in this protocol:

- Physical layer.
- Timing rules.
- Rules for managing connections and error management.
- Human readability.
- Byte (or bit) stuffing.
- Error control.
- Checksum.
- Mechanism to auto assign addresses.
- Rules to enforce compatibility between different applications.
- Feature for universal plug-and-play like USB or TCP/IP.
- Custom header features (see original KEN Protocol).

BASIC RULES

- Keep it simple.
- All header elements are position dependent.
- If this is not a good fit, take a look at KEN-A or KEN-B or KEN-C.

FRAME FORMAT

All message frames begin with a Frame Length (FL) byte with msb = 1. The only header control bytes are the to/from address. The FL is of the entire frame. Therefore the maximum frame size is 127 bytes. Payload data can range from 0 bytes to 125, with no checksum.

Frame highlights:

- `++` Frame length (FL)
- `**` Header control elements (hexadecimal)
- `==` Frame payload data

```
Frame Len [Addressing] [Data]
+++++++++ ************ ======

1  2  <-- Header Bytes
|  |
FL AB [Data]
++ ** ======
|  || |
|  || +--payload data
|  |+--to address
|  +---from address
+--frame length (FL)
```

Frame Length (FL), Header Byte 1

This required feature specifies the total number byte in the frame, including the required header and checksum, if used. Thus a minimum frame length is 3 bytes and up to 127.

Data payload size = 127 - 5 header bytes - checksum bytes (if used)

The msb of the FL is always set to 1. The remaining 7-bits define the length of the entire frame.

```
`0x80` || nn = 05 to 7F = Length is 5 to 127.
```

Examples:

- 5-byte data length = `0x85`
- 17-bytes data length = `0x91`
- 127-bytes data length = `0xFF`

Address, From (ie. from A to B), Header Byte 3, Upper Nibble

Required header element located in Header byte 3 in the upper nibble.

It is recommended to set this value to no-zero to avoid too many consecutive zeros in the frame header. Can be used as an ID code for a set of hardware.

```
n = 0 = Node does not have an assigned address.

n = 1 to F = address (15).
```

Address, To (ie. from A to B), Header Byte 3, Lower Nibble

Required header element located in Header byte 3 in the lower nibble.

It is recommended to set this value to no-zero to avoid too many consecutive zeros in the frame header. Can be used as a command code for for hardware.

```
n = 0 = All (Broadcast).

n = 1 to F = address (15).
```

DATA

The payload data can be binary or ASCII. It is up to the application to manage the meaning of any data being used. However, the size of the data payload is limited by the required data length byte in the header.

PROTOCOL TYPE

How do you compare different implementations? The protocol type generates a succinct uniform description. This is a form a configuration control. It can help to determine if two systems might be compatible. This is useful information to be included in source code comments.

There is only one protocol type as defined for this KEN protocol version. This version has a fixed header with all defined elements always present.

The type, or "flavor", of the KEN Data protocol is expressed with a two digit hexadecimal number. This is prefaced with the protocol version. See below for definitions.

Features Used (set bits to '1' if used)

```
msb           lsb
0 0 1 0   1 1 0 0
- - - -   - - - -
| | | |   | | | |
| | | |   | | | +-- not defined, always 0
| | | |   | | +---- not defined, always 0
| | | |   | +------ from address (An), always required
| | | |   +-------- to address (Bn), always required
| | | |
| | | +------------ not defined, always 0
| | +-------------- frame length, always required
| +---------------- not defined, always 0
+------------------ not defined, always 0
```

Type `KEN-D:2D`

- From/to addressing

USAGE TIPS

This is intended for point designs, not general compatibility with other KEN Data protocol designs (or the original KEN protocol for that matter). If more features are needed, then choose one of the different versions.

This protocol is designed for ASCII or binary data. Parsing the data is completely dependent on position in the frame. This saves overhead. The original KEN protocol utilized flags to locate frame components with an overhead penalty.

If the addressing is not needed, then just ignore the second byte. If the data is all ASCII and the from address is limited to <=7 (msb=0) then the only byte with the msb=1 would be the frame length. The receiver could then trigger off the msb of the first byte of a frame.

Since the header and data payload can all be binary characters, and there is no escape character, and no unique flags; any implementation must include idle breaks between frames.

All features are nibble based, and thus are limited in capabilities. As in KEN-C, the frame contents are position dependent. Since the header size is fixed, the start of the payload data is easily determined. The header can contain both binary or ASCII.

If some additional features are needed, the developer could define then as part of the data payload.

Any application should specify the Protocol Type prominently in the code comments. Be sure to update this when making any application revisions.

COMPLEXITY HIERARCHY

The list below defines a general hierarchy from simple to most complex features. This is a generalized progression in which features would most likely be added.

1. Basic flags (`FL`). Simplest.
2. Adding addressing.

EXAMPLE MESSAGES

Below are a number of example messages.

Frame highlights:

- `FL` Frame Length
- `**` Required header features
- `==` Frame payload data
  - `31` = Hex `0x31` `==`
  - `1` = ASCII "1" `=`
- `^^` Binary data

Examples

Minimum message with one byte of data and 2-bytes of overhead.

```
86 11 31
++ ** ==
```

Twenty nine ASCII data bytes with minimum overhead and data length with simple addressing (from node 0xA1 to 0xB2). 2-bytes of overhead.

```
9E 21 KEN-D Protocol - Hello World!
++ ** =============================
```

Older Ideas

Notes on older ideas that have been retired, but want to retain for possible reconsideration at a later date.

- Considered including some sort of a checksum. Having to specify the type and then adding the checksum made the frame pretty close to KEN-C. The point was to make this noticeable simpler.

END OF DOCUMENT