6Lo Working Group   K. Lynn, Ed.
Internet-Draft   Verizon Labs
Intended status: Standards Track   J. Martocci
Expires: May 4, 2017   Johnson Controls
                                                        C. Neilson
                                                     Delta Controls
                                                      S. Donaldson
                                                         Honeywell
                                                  October 31, 2016

Transmission of IPv6 over MS/TP Networks
draft-ietf-6lo-6lobac-06

Abstract

   Master-Slave/Token-Passing (MS/TP) is a medium access control method
   for the RS-485 physical layer, which is used extensively in building
   automation networks. This specification defines the frame format for
   transmission of IPv6 packets and the method of forming link-local and
   statelessly autoconfigured IPv6 addresses on MS/TP networks.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on May 4, 2017.

Table of Contents

1. Introduction

Master-Slave/Token-Passing (MS/TP) is a medium access control (MAC)
protocol for the RS-485 [TIA-485-A] physical layer, which is used
extensively in building automation networks. This specification
defines the frame format for transmission of IPv6 [RFC2460] packets
and the method of forming link-local and statelessly autoconfigured
IPv6 addresses on MS/TP networks. The general approach is to adapt
elements of the 6LoWPAN specifications [RFC4944], [RFC6282], and
[RFC6775], where noted, to constrained wired networks.

An MS/TP device is typically based on a low-cost microcontroller with
limited processing power and memory. Together with low data rates
and a small MAC address space, these constraints are similar to those
faced in 6LoWPAN networks and suggest some elements of that solution
might be leveraged. MS/TP differs significantly from 6LoWPAN in at
least three respects: a) MS/TP devices typically have a continuous
source of power, b) all MS/TP devices on a segment can communicate
directly so there are no hidden node or mesh routing issues, and c)
recent changes to MS/TP provide support for larger payloads,

   eliminating the need for fragmentation and reassembly below IPv6.

   The following sections provide a brief overview of MS/TP, then
   describe how to form IPv6 addresses and encapsulate IPv6 packets in
   MS/TP frames. This document also specifies a header compression
   mechanism, based on [RFC6282], that is REQUIRED in order to reduce
   latency and make IPv6 practical on MS/TP networks.

1.1. Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

1.2. Abbreviations Used

   ASHRAE: American Society of Heating, Refrigerating, and Air-
            Conditioning Engineers (http://www.ashrae.org)

   BACnet: An ISO/ANSI/ASHRAE Standard Data Communication Protocol
            for Building Automation and Control Networks

   CRC: Cyclic Redundancy Check

   MAC: Medium Access Control

   MSDU: MAC Service Data Unit (MAC client data)

   MTU: Maximum Transmission Unit

   UART: Universal Asynchronous Transmitter/Receiver

1.3. MS/TP Overview

   This section provides a brief overview of MS/TP, which is specified
   in ANSI/ASHRAE 135-2012 (BACnet) Clause 9 [Clause9] and included
   herein by reference. BACnet [Clause9] also covers physical layer
   deployment options.

   MS/TP is designed to enable multidrop networks over shielded twisted
   pair wiring. It can support network segments up to 1000 meters in
   length at a data rate of 115,200 bit/s, or segments up to 1200 meters
   in length at lower bit rates. An MS/TP link requires only a UART, an
   RS-485 [TIA-485-A] transceiver with a driver that can be disabled,
   and a 5 ms resolution timer. These features make MS/TP a cost-
   effective field bus for the most numerous and least expensive devices
   in a building automation network.

The differential signaling used by [TIA-485-A] requires a contention-
free MAC. MS/TP uses a token to control access to a multidrop bus.
A master node may initiate the transmission of a data frame when it
holds the token. After sending at most a configured maximum number
of data frames, a master node passes the token to the next master
node (as determined by MAC address). If present on the link, legacy
MS/TP implementations (including all slave nodes) ignore the frame
format defined in this specification.

BACnet Addendum 135-2012an [Addendum_an] defines a range of Frame
Type values to designate frames that contain larger data and data CRC
fields, encoded using Consistent Overhead Byte Stuffing [COBS] (see
Appendix B). The purpose of COBS encoding is to eliminate preamble
sequences from the Encoded Data and Encoded CRC-32K fields. The
maximum length of an MSDU as defined by this specification is 1500
octets (before encoding). The Encoded Data is covered by a 32-bit
CRC [CRC32K] (see Appendix C). The CRC-32K is then COBS encoded.

MS/TP COBS-encoded frames have the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x55      |     0xFF      |  Frame Type   |      DA       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SA        |    Length (MS octet first)    |  Header CRC   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.               Encoded Data (2 - 1506 octets)                 .
.                                                               .
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |  Encoded CRC-32K (5 octets)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+
|                                               | optional 0xFF |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

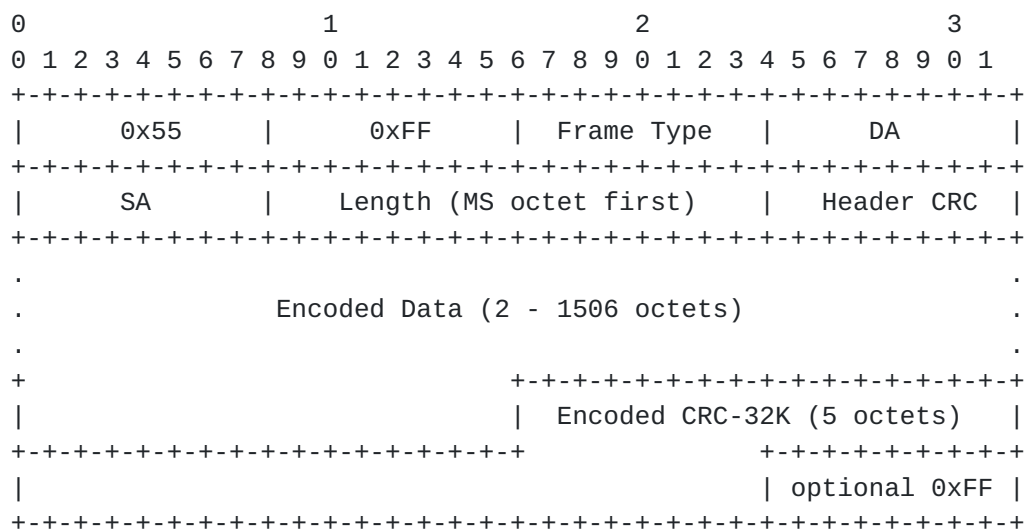   Figure 1: MS/TP COBS-Encoded Frame Format

MS/TP COBS-encoded frame fields have the following descriptions:

```
    Preamble              two octet preamble: 0x55, 0xFF
    Frame Type            one octet
    Destination Address   one octet address
    Source Address        one octet address
    Length                two octets, most significant octet first
    Header CRC            one octet
    Encoded Data          2 - 1506 octets (see Appendix B)
    Encoded CRC-32K       five octets (see Appendix C)
    (pad)                 (optional) at most one octet of trailer: 0xFF
```

The Frame Type is used to distinguish between different types of MAC frames. The types relevant to this specification (in decimal) are:

```
  0  Token
  1  Poll For Master
  2  Reply To Poll For Master
     ...
 34  IPv6 over MS/TP (LoBAC) Encapsulation
```

Frame Types 8 - 31 and 35 - 127 are reserved for assignment by ASHRAE. Frame Types 32 - 127 designate COBS-encoded frames and MUST convey Encoded Data and Encoded CRC-32K fields. All master nodes MUST understand Token, Poll For Master, and Reply to Poll For Master control frames. See Section 2 for additional details.

The Destination and Source Addresses are each one octet in length. See Section 3 for additional details.

For COBS-encoded frames, the Length field indicates the size of the [COBS] Encoded Data field in octets, plus three. (This adjustment is required in order for legacy MS/TP devices to ignore COBS-encoded frames.) See Section 4 and Appendices for additional details.

The Header CRC field covers the Frame Type, Destination Address, Source Address, and Length fields. The Header CRC generation and check procedures are specified in BACnet [Clause9].

Use of the optional 0xFF trailer octet is discussed in BACnet [Clause9].

1.4. Goals and Constraints

The primary goal of this specification is to enable IPv6 directly on wired end devices in building automation and control networks by leveraging existing standards to the greatest extent possible. A secondary goal is to co-exist with legacy MS/TP implementations. Only the minimum changes necessary to support IPv6 over MS/TP were

specified in BACnet [Addendum_an] (see Section 1.3).

In order to co-exist with legacy devices, no changes are permitted to the MS/TP addressing modes, frame header format, control frames, or Master Node state machine as specified in BACnet [Clause9].

2. MS/TP Mode for IPv6

ASHRAE has assigned an MS/TP Frame Type value of 34 to indicate IPv6 over MS/TP (LoBAC) Encapsulation. This falls within the range of values that designate COBS-encoded data frames.

All MS/TP master nodes (including those that support IPv6) MUST implement the Master Node state machine specified in BACnet [Clause9] and handle Token, Poll For Master, and Reply to Poll For Master control frames. MS/TP master nodes that support IPv6 MUST also implement the Receive Frame state machine specified in [Clause9] as extended by BACnet [Addendum_an].

All MS/TP nodes that support IPv6 MUST support a data rate of 115,200 bit/s and MAY optionally support lower data rates as defined in BACnet [Clause9].

3. Addressing Modes

MS/TP node (MAC) addresses are one octet in length. The method of assigning MAC addresses is outside the scope of this specification. However, each MS/TP node on the link MUST have a unique address in order to ensure correct MAC operation.

BACnet [Clause9] specifies that addresses 0 through 127 are valid for master nodes. The method specified in Section 6 for creating a MAC-layer-derived Interface Identifier (IID) ensures that an IID of all zeros can never result.

A Destination Address of 255 (all nodes) indicates a MAC-layer broadcast. MS/TP does not support multicast, therefore all IPv6 multicast packets MUST be broadcast at the MAC layer and filtered at the IPv6 layer. A Source Address of 255 MUST NOT be used.

Hosts learn IPv6 prefixes via router advertisements according to [RFC4861].

4. Maximum Transmission Unit (MTU)

BACnet [Addendum_an] supports MSDUs up to 2032 octets in length. This specification defines an MSDU length of at least 1280 octets and at most 1500 octets (before encoding). This is sufficient to convey

the minimum MTU required by IPv6 [RFC2460] without the need for link-
layer fragmentation and reassembly. Support for an MSDU length of
1500 octets is RECOMMENDED.

5. **LoBAC Adaptation Layer**

The relatively low data rates of MS/TP dictate header compression as
a means to reduce latency. This section specifies an adaptation
layer to support compressed IPv6 headers as specified in Section 10.
IPv6 header compression MUST be implemented on all nodes.

Implementations MAY also support Generic Header Compression (GHC)
[RFC7400] for transport layer headers. A node implementing [RFC7400]
MUST probe its peers for GHC support before applying GHC.

The encapsulation format defined in this section (subsequently
referred to as the "LoBAC" encapsulation) comprises the MSDU of an
IPv6 over MS/TP frame. The LoBAC payload (i.e., an IPv6 packet)
follows an encapsulation header stack. LoBAC is a subset of the
LoWPAN encapsulation defined in [RFC4944] and extended by [RFC6282],
therefore the use of "LOWPAN" in literals below is intentional. The
primary difference between LoWPAN and LoBAC is omission of the Mesh,
Broadcast, Fragmentation, and LOWPAN_HC1 headers.

All LoBAC encapsulated datagrams transmitted over MS/TP are prefixed
by an encapsulation header stack consisting of a Dispatch value
followed by zero or more header fields. The only sequence currently
defined for LoBAC is the LOWPAN_IPHC header followed by payload, as
shown below:

```
         +---------------+---------------+------..-----+
         | IPHC Dispatch |  IPHC Header  |   Payload   |
         +---------------+---------------+------..-----+
```

 Figure 2: A LoBAC Encapsulated LOWPAN_IPHC Compressed IPv6 Datagram

The Dispatch value may be treated as an unstructured namespace. Only
a single pattern is used to represent current LoBAC functionality.

```
   Pattern      Header Type
  +------------+------------------------------------------------------+
  | 01  1xxxxx | LOWPAN_IPHC - LOWPAN_IPHC compressed IPv6 [RFC6282] |
  +------------+------------------------------------------------------+
```

             Figure 3: LoBAC Dispatch Value Bit Pattern

Other IANA-assigned 6LoWPAN Dispatch values do not apply to 6LoBAC
unless otherwise specified.

6. Stateless Address Autoconfiguration

   This section defines how to obtain an IPv6 Interface Identifier. The
   general procedure for creating a MAC-address-derived IID is described
   in [RFC4291] Appendix A, "Creating Modified EUI-64 Format Interface
   Identifiers", as updated by [RFC7136].

   The IID SHOULD NOT embed an [EUI-64] or any other globally unique
   hardware identifier assigned to a device (see Section 12).

   The Interface Identifier for link-local addresses SHOULD be formed by
   concatenating a node's' 8-bit MS/TP MAC address to the seven octets
   0x00, 0x00, 0x00, 0xFF, 0xFE, 0x00, 0x00. For example, an MS/TP MAC
   address of hexadecimal value 0x4F results in the following IID:

```
|0              1|1              3|3              4|4              6|
|0              5|6              1|2              7|8              3|
+---------------+---------------+---------------+---------------+
|0000000000000000|0000000011111111|1111111000000000|0000000001001111|
+---------------+---------------+---------------+---------------+
```

   This is the RECOMMENDED method of forming an IID for use in link-
   local addresses, as it affords the most efficient header compression
   provided by the LOWPAN_IPHC [RFC6282] format specified in Section 10.

   A 64-bit random IID is RECOMMENDED for each globally scoped address
   and SHOULD be locally generated according to one of the methods cited
   in Section 12. A node that generates a 64-bit random IID MUST
   register it with its local router(s) by sending a Neighbor
   Solicitation (NS) message with the Address Registration Option (ARO)
   and process Neighbor Advertisements (NA) according to [RFC6775].

   An IPv6 address prefix used for stateless autoconfiguration [RFC4862]
   of an MS/TP interface MUST have a length of 64 bits.

7. IPv6 Link Local Address

   The IPv6 link-local address [RFC4291] for an MS/TP interface is
   formed by appending the Interface Identifier, as defined above, to
   the prefix FE80::/64.

```
  10 bits          54 bits                  64 bits
+----------+-----------------------+----------------------------+
|1111111010|        (zeros)        |     Interface Identifier   |
+----------+-----------------------+----------------------------+
```

8. Unicast Address Mapping

The address resolution procedure for mapping IPv6 non-multicast
addresses into MS/TP MAC-layer addresses follows the general
description in Section 7.2 of [RFC4861], unless otherwise specified.

The Source/Target Link-layer Address option has the following form
when the addresses are 8-bit MS/TP MAC-layer (node) addresses.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type       |   Length=1    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00       | MS/TP Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+      Padding (all zeros)      +
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option fields:

Type:

   1: for Source Link-layer address.

   2: for Target Link-layer address.

Length: This is the length of this option (including the type and
   length fields) in units of 8 octets. The value of this field is 1
   for 8-bit MS/TP MAC addresses.

MS/TP Address: The 8-bit address in canonical bit order [RFC2469].
   This is the unicast address the interface currently responds to.

9. Multicast Address Mapping

All IPv6 multicast packets MUST be sent to MS/TP Destination Address
255 (broadcast) and filtered at the IPv6 layer. When represented as
a 16-bit address in a compressed header (see Section 10), it MUST be
formed by padding on the left with a zero:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x00       |     0xFF      |
+-+-+-+-+-+-+-+-+---------------+
```

10. Header Compression

   LoBAC uses LOWPAN_IPHC IPv6 compression, which is specified in
   [RFC6282] and included herein by reference. This section will simply
   identify substitutions that should be made when interpreting the text
   of [RFC6282].

   In general the following substitutions should be made:

    - Replace instances of "6LoWPAN" with "MS/TP network"

    - Replace instances of "IEEE 802.15.4 address" with "MS/TP address"

   When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short
   address") it MUST be formed by padding the MS/TP address to the left
   with a zero:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x00     | MS/TP address |
+-+-+-+-+-+-+-+-+-+---------------+
```

   If LOWPAN_IPHC compression [RFC6282] is used with context, the
   router(s) directly attached to the MS/TP segment MUST disseminate the
   6LoWPAN Context Option (6CO) according to [RFC6775], Section 7.2.

11. IANA Considerations

   This document uses values previously reserved by [RFC4944] and
   [RFC6282] and makes no further requests of IANA.

   Note to RFC Editor: this section may be removed upon publication.

12. Security Considerations

   Globally scoped addresses that contain IIDs generated using MS/TP
   node addresses may expose a network to address scanning attacks. For
   this reason, it is RECOMMENDED that a different (but stable) IID be
   generated for each globally scoped address in use according to, for
   example, [RFC3315], [RFC3972], [RFC4941], [RFC5535], or [RFC7217].

   MS/TP networks are by definition wired and not susceptible to casual
   eavesdropping. By the same token, MS/TP nodes are stationary and
   correlation of activities or location tracking of individuals is
   unlikely. See [I-D.ietf-6lo-privacy-considerations] for a full
   discussion of mitigation of the threats posed to constrained nodes.

13. Acknowledgments

We are grateful to the authors of [RFC4944] and members of the IETF
6LoWPAN working group; this document borrows liberally from their
work. Ralph Droms and Brian Haberman provided indispensable guidance
and support from the outset. Peter van der Stok, James Woodyatt, and
Carsten Bormann provided detailed reviews. Stuart Cheshire invented
the very clever COBS encoding. Michael Osborne made the critical
observation that separately encoding the data and CRC32K fields would
allow the CRC to be calculated on-the-fly. Alexandru Petrescu, Brian
Frank, Geoff Mulligan, and Don Sturek offered valuable comments.

14. References

14.1. Normative References

[Addendum_an]
          ASHRAE, "ANSI/ASHRAE Addenda an, at, au, av, aw, ax, and
          az to ANSI/ASHRAE Standard 135-2012, BACnet - A Data
          Communication Protocol for Building Automation and Control
          Networks", July 2014,
          <https://www.ashrae.org/File%20Library/docLib/StdsAddenda/
          07-31-2014_135_2012_an_at_au_av_aw_ax_az_Final.pdf>.

[Clause9] American Society of Heating, Refrigerating, and Air-
          Conditioning Engineers, "BACnet - A Data Communication
          Protocol for Building Automation and Control Networks",
          ANSI/ASHRAE 135-2012 (Clause 9), March 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
          (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460,
          December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
          C., and M. Carney, "Dynamic Host Configuration Protocol
          for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
          2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)",
          RFC 3972, DOI 10.17487/RFC3972, March 2005,
          <http://www.rfc-editor.org/info/rfc3972>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <http://www.rfc-editor.org/info/rfc4291>.

   [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
              "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
              DOI 10.17487/RFC4861, September 2007,
              <http://www.rfc-editor.org/info/rfc4861>.

   [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless
              Address Autoconfiguration", RFC 4862,
              DOI 10.17487/RFC4862, September 2007,
              <http://www.rfc-editor.org/info/rfc4862>.

   [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy
              Extensions for Stateless Address Autoconfiguration in
              IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007,
              <http://www.rfc-editor.org/info/rfc4941>.

   [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
              "Transmission of IPv6 Packets over IEEE 802.15.4
              Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007,
              <http://www.rfc-editor.org/info/rfc4944>.

   [RFC5535] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535,
              DOI 10.17487/RFC5535, June 2009,
              <http://www.rfc-editor.org/info/rfc5535>.

   [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6
              Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,
              DOI 10.17487/RFC6282, September 2011,
              <http://www.rfc-editor.org/info/rfc6282>.

   [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C.
              Bormann, "Neighbor Discovery Optimization for IPv6 over
              Low-Power Wireless Personal Area Networks (6LoWPANs)",
              RFC 6775, DOI 10.17487/RFC6775, November 2012,
              <http://www.rfc-editor.org/info/rfc6775>.

   [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6
              Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136,
              February 2014, <http://www.rfc-editor.org/info/rfc7136>.

   [RFC7217] Gont, F., "A Method for Generating Semantically Opaque
              Interface Identifiers with IPv6 Stateless Address
              Autoconfiguration (SLAAC)", RFC 7217,
              DOI 10.17487/RFC7217, April 2014,
              <http://www.rfc-editor.org/info/rfc7217>.

   [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for
              IPv6 over Low-Power Wireless Personal Area Networks
              (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November
              2014, <http://www.rfc-editor.org/info/rfc7400>.

14.2. Informative References

   [COBS]   Cheshire, S. and M. Baker, "Consistent Overhead Byte
              Stuffing", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL.7,
              NO.2 , April 1999,
              <http://www.stuartcheshire.org/papers/COBSforToN.pdf>.

   [CRC32K]   Koopman, P., "32-Bit Cyclic Redundancy Codes for Internet
              Applications", IEEE/IFIP International Conference on
              Dependable Systems and Networks (DSN 2002) , June 2002,
              <http://www.ece.cmu.edu/~koopman/networks/dsn02/
              dsn02_koopman.pdf>.

   [EUI-64]   IEEE, "Guidelines for 64-bit Global Identifier (EUI-64)
              Registration Authority", March 1997,
              <http://standards.ieee.org/regauth/oui/tutorials/
              EUI64.html>.

   [I-D.ietf-6lo-privacy-considerations]
              Thaler, D., "Privacy Considerations for IPv6 Adaptation
              Layer Mechanisms", draft-ietf-6lo-privacy-
              considerations-04 (work in progress), October 2016.

   [IEEE.802.3]
              "Information technology - Telecommunications and
              information exchange between systems - Local and
              metropolitan area networks - Specific requirements - Part
              3: Carrier Sense Multiple Access with Collision Detection
              (CMSA/CD) Access Method and Physical Layer
              Specifications", IEEE Std 802.3-2012, December 2012,
              <http://standards.ieee.org/getieee802/802.3.html>.

   [RFC2469] Narten, T. and C. Burton, "A Caution On The Canonical
              Ordering Of Link-Layer Addresses", RFC 2469,
              DOI 10.17487/RFC2469, December 1998,
              <http://www.rfc-editor.org/info/rfc2469>.

   [TIA-485-A]
              Telecommunications Industry Association, "TIA-485-A,
              Electrical Characteristics of Generators and Receivers for
              Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-
              485-A-98) (R2003)", March 2003.

[Appendix A](#). **Abstract MAC Interface**

   This Appendix is informative and not part of the standard.

   BACnet [[Clause9](#)] provides support for MAC-layer clients through its
   SendFrame and ReceivedDataNoReply procedures. However, it does not
   define a network-protocol independent abstract interface for the MAC.
   This is provided below as an aid to implementation.

[A.1](#). **MA-DATA.request**

[A.1.1](#). **Function**

   This primitive defines the transfer of data from a MAC client entity
   to a single peer entity or multiple peer entities in the case of a
   broadcast address.

[A.1.2](#). **Semantics of the Service Primitive**

   The semantics of the primitive are as follows:

```
   MA-DATA.request (
                    destination_address,
                    source_address,
                    data,
                    type
                   )
```

   The 'destination_address' parameter may specify either an individual
   or a broadcast MAC entity address. It must contain sufficient
   information to create the Destination Address field (see [Section 1.3](#))
   that is prepended to the frame by the local MAC sublayer entity. The
   'source_address' parameter, if present, must specify an individual
   MAC address. If the source_address parameter is omitted, the local
   MAC sublayer entity will insert a value associated with that entity.

   The 'data' parameter specifies the MAC service data unit (MSDU) to be
   transferred by the MAC sublayer entity. There is sufficient
   information associated with the MSDU for the MAC sublayer entity to
   determine the length of the data unit.

   The 'type' parameter specifies the value of the MS/TP Frame Type
   field that is prepended to the frame by the local MAC sublayer
   entity.

A.1.3. **When Generated**

   This primitive is generated by the MAC client entity whenever data
   shall be transferred to a peer entity or entities. This can be in
   response to a request from higher protocol layers or from data
   generated internally to the MAC client, such as a Token frame.

A.1.4. **Effect on Receipt**

   Receipt of this primitive will cause the MAC entity to insert all MAC
   specific fields, including Destination Address, Source Address, Frame
   Type, and any fields that are unique to the particular media access
   method, and pass the properly formed frame to the lower protocol
   layers for transfer to the peer MAC sublayer entity or entities.

A.2. **MA-DATA.indication**

A.2.1. **Function**

   This primitive defines the transfer of data from the MAC sublayer
   entity to the MAC client entity or entities in the case of a
   broadcast address.

A.2.2. **Semantics of the Service Primitive**

   The semantics of the primitive are as follows:

      MA-DATA.indication (
                        destination_address,
                        source_address,
                        data,
                        type
)

   The 'destination_address' parameter may be either an individual or a
   broadcast address as specified by the Destination Address field of
   the incoming frame. The 'source_address' parameter is an individual
   address as specified by the Source Address field of the incoming
   frame.

   The 'data' parameter specifies the MAC service data unit (MSDU) as
   received by the local MAC entity. There is sufficient information
   associated with the MSDU for the MAC sublayer client to determine the
   length of the data unit.

   The 'type' parameter is the value of the MS/TP Frame Type field of
   the incoming frame.

A.2.3. **When Generated**

   The MA_DATA.indication is passed from the MAC sublayer entity to the
   MAC client entity or entities to indicate the arrival of a frame to
   the local MAC sublayer entity that is destined for the MAC client.
   Such frames are reported only if they are validly formed, received
   without error, and their destination address designates the local MAC
   entity. Frames destined for the MAC Control sublayer are not passed
   to the MAC client.

A.2.4. **Effect on Receipt**

   The effect of receipt of this primitive by the MAC client is
   unspecified.

Appendix B. **Consistent Overhead Byte Stuffing [COBS]**

   This Appendix is informative and not part of the standard.

   BACnet [Addendum_an] corrects a long-standing issue with the MS/TP
   specification; namely that preamble sequences were not escaped
   whenever they appeared in the Data or Data CRC fields. In rare
   cases, this resulted in dropped frames due to loss of frame
   synchronization. The solution is to encode the Data and 32-bit Data
   CRC fields before transmission using Consistent Overhead Byte
   Stuffing [COBS] and decode these fields upon reception.

   COBS is a run-length encoding method that nominally removes '0x00'
   octets from its input. Any selected octet value may be removed by
   XOR'ing that value with each octet of the COBS output. BACnet
   [Addendum_an] specifies the preamble octet '0x55' for removal.

   The minimum overhead of COBS is one octet per encoded field. The
   worst-case overhead in long fields is bounded to one octet per 254,
   or less than 0.4%, as described in [COBS].

   Frame encoding proceeds logically in two passes. The Encoded Data
   field is prepared by passing the MSDU through the COBS encoder and
   XOR'ing the preamble octet '0x55' with each octet of the output. The
   Encoded CRC-32K field is then prepared by calculating a CRC-32K over
   the Encoded Data field and formatting it for transmission as
   described in Appendix C. The combined length of these fields, minus
   two octets for compatibility with existing MS/TP devices, is placed
   in the MS/TP header Length field before transmission.

   Example COBS encoder and decoder functions are shown below for
   illustration. Complete examples of use and test vectors are provided
   in BACnet [Addendum_an].

```
   #include <stddef.h>
   #include <stdint.h>

  /*
   * Encodes 'length' octets of data located at 'from' and
   * writes one or more COBS code blocks at 'to', removing any
   * 'mask' octets that may present be in the encoded data.
   * Returns the length of the encoded data.
   */

  size_t
  cobs_encode (uint8_t *to, const uint8_t *from, size_t length,
               uint8_t mask)
  {
    size_t code_index = 0;
    size_t read_index = 0;
    size_t write_index = 1;
    uint8_t code = 1;
    uint8_t data, last_code;

    while (read_index < length) {
      data = from[read_index++];
      /*
       * In the case of encountering a non-zero octet in the data,
       * simply copy input to output and increment the code octet.
       */
      if (data != 0) {
        to[write_index++] = data ^ mask;
        code++;
        if (code != 255)
          continue;
      }
      /*
       * In the case of encountering a zero in the data or having
       * copied the maximum number (254) of non-zero octets, store
       * the code octet and reset the encoder state variables.
       */
      last_code = code;
      to[code_index] = code ^ mask;
      code_index = write_index++;
      code = 1;
    }
    /*
     * If the last chunk contains exactly 254 non-zero octets, then
     * this exception is handled above (and returned length must be
     * adjusted). Otherwise, encode the last chunk normally, as if
     * a "phantom zero" is appended to the data.
     */
```

```
   if ((last_code == 255) && (code == 1))
     write_index--;
   else
     to[code_index] = code ^ mask;

   return write_index;
 }
```

```
   #include <stddef.h>
   #include <stdint.h>

   /*
    * Decodes 'length' octets of data located at 'from' and
    * writes the original client data at 'to', restoring any
    * 'mask' octets that may present in the encoded data.
    * Returns the length of the encoded data or zero if error.
    */
   size_t
   cobs_decode (uint8_t *to, const uint8_t *from, size_t length,
                uint8_t mask)
   {
     size_t read_index = 0;
     size_t write_index = 0;
     uint8_t code, last_code;

     while (read_index < length) {
       code = from[read_index] ^ mask;
       last_code = code;
       /*
        * Sanity check the encoding to prevent the while() loop below
        * from overrunning the output buffer.
        */
       if (read_index + code > length)
         return 0;

       read_index++;
       while (--code > 0)
         to[write_index++] = from[read_index++] ^ mask;
       /*
        * Restore the implicit zero at the end of each decoded block
        * except when it contains exactly 254 non-zero octets or the
        * end of data has been reached.
        */
       if ((last_code != 255) && (read_index < length))
         to[write_index++] = 0;
     }
     return write_index;
   }
```

Appendix C. Encoded CRC-32K [CRC32K]

   This Appendix is informative and not part of the standard.

   Extending the payload of MS/TP to 1500 octets required upgrading the
   Data CRC from 16 bits to 32 bits. P.Koopman has authored several
   papers on evaluating CRC polynomials for network applications. In

[CRC32K], he surveyed the entire 32-bit polynomial space and noted
some that exceed the [IEEE.802.3] polynomial in performance. BACnet
[Addendum_an] specifies the CRC-32K (Koopman) polynomial.

The specified use of the calc_crc32K() function is as follows.
Before a frame is transmitted, 'crc_value' is initialized to all
ones. After passing each octet of the [COBS] Encoded Data through
the function, the ones complement of the resulting 'crc_value' is
arranged in LSB-first order and is itself [COBS] encoded. The length
of the resulting Encoded CRC-32K field is always five octets.

Upon reception of a frame, 'crc_value' is initialized to all ones.
The octets of the Encoded Data field are accumulated by the
calc_crc32K() function before decoding. The Encoded CRC-32K field is
then decoded and the resulting four octets are accumulated by the
calc_crc32K() function. If the result is the expected residue value
'CRC32K_RESIDUE', then the frame was received correctly.

An example CRC-32K function in shown below for illustration.
Complete examples of use and test vectors are provided in BACnet
[Addendum_an].

```c
#include <stdint.h>

/* See BACnet Addendum 135-2012an, section G.3.2 */
#define CRC32K_INITIAL_VALUE (0xFFFFFFFF)
#define CRC32K_RESIDUE (0x0843323B)

/* CRC-32K polynomial, 1 + x**1 + ... + x**30 (+ x**32) */
#define CRC32K_POLY (0xEB31D82E)

/*
 * Accumulate 'data_value' into the CRC in 'crc_value'.
 * Return updated CRC.
 *
 * Note: crc_value must be set to CRC32K_INITIAL_VALUE
 * before initial call.
 */
uint32_t
calc_crc32K (uint8_t data_value, uint32_t crc_value)
{
  int b;

  for (b = 0; b < 8; b++) {
    if ((data_value & 1) ^ (crc_value & 1)) {
      crc_value >>= 1;
      crc_value ^= CRC32K_POLY;
    } else {
      crc_value >>= 1;
    }
    data_value >>= 1;
  }
  return crc_value;
}
```

Appendix D. Example 6LoBAC Packet Decode

   This Appendix is informative and not part of the standard.

   BACnet MS/TP, Src (2), Dst (1), IPv6 Encapsulation
       Preamble 55: 0x55
       Preamble FF: 0xff
       Frame Type: IPv6 Encapsulation (34)
       Destination Address: 1
       Source Address: 2
       Length: 537
       Header CRC: 0x1c [correct]
       Extended Data CRC: 0x9e7259e2 [correct]
   6LoWPAN
       IPHC Header
           011. .... = Pattern: IP header compression (0x03)
           ...1 1... .... = Traffic class and flow label:
                                 Version, traffic class, and flow label
                                 compressed (0x0003)
           .... .0.. .... = Next header: Inline
           .... ..00 .... = Hop limit: Inline (0x0000)
           .... 1... .... = Context identifier extension: True
           .... .1.. .... = Source address compression: Stateful
           ....01 .... = Source address mode:
                                 64-bits inline (0x0001)
           .... 0... = Multicast address compression: False
           .... .1.. = Destination address compression:
                                 Stateful
           ....10 = Destination address mode:
                                 16-bits inline (0x0002)
           0000 .... = Source context identifier: 0x00
           .... 0000 = Destination context identifier: 0x00
           [Source context: aaaa:: (aaaa::)]
           [Destination context: aaaa:: (aaaa::)]
       Next header: ICMPv6 (0x3a)
       Hop limit: 63
       Source: aaaa::1 (aaaa::1)
       Destination: aaaa::ff:fe00:1 (aaaa::ff:fe00:1)
   Internet Protocol Version 6, Src: aaaa::1 (aaaa::1),
                                Dst: aaaa::ff:fe00:1 (aaaa::ff:fe00:1)
       0110 .... = Version: 6
       .... 0000 0000 .... = Traffic class:
                                                 0x00000000
       .... 0000 00.. .... = Differentiated
                                                 Services Field:
                                                 Default (0x00000000)
       ....0. .... = ECN-Capable Transport

```
                                             (ECT): Not set
     .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
     .... .... ... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
     Payload length: 518
     Next header: ICMPv6 (58)
     Hop limit: 63
     Source: aaaa::1 (aaaa::1)
     Destination: aaaa::ff:fe00:1 (aaaa::ff:fe00:1)
  Internet Control Message Protocol v6
     Type: Echo (ping) request (128)
     Code: 0
     Checksum: 0x783f [correct]
     Identifier: 0x2ee5
     Sequence: 2
     [Response In: 5165]
     Data (510 bytes)
         Data: e4dbe8553ba0040008090a0b0c0d0e0f1011121314151617...
         [Length: 510]
```

```
Frame (547 bytes):
55 ff 22 01 02 02 19 1c 56 2d 83 56 6f 6a 54 54   U.".....V-.VojTT
54 54 54 54 57 54 56 54 d5 50 2d 6a 7b b0 5c 57   TTTTWTVT.P-j{.\W
b1 8e bd 00 6e f5 51 ac 5d 5c 5f 5e 59 58 5b 5a   ....n.Q.]\_^YX[Z
45 44 47 46 41 40 43 42 4d 4c 4f 4e 49 48 4b 4a   EDGFA@CBMLONIHKJ
75 74 77 76 71 70 73 72 7d 7c 7f 7e 79 78 7b 7a   utwvqpsr}|.~yx{z
65 64 67 66 61 60 63 62 6d 6c 6f 6e 69 68 6b 6a   edgfa`cbmlonihkj
15 14 17 16 11 10 13 12 1d 1c 1f 1e 19 18 1b 1a   ................
05 04 07 06 01 00 03 02 0d 0c 0f 0e 09 08 0b 0a   ................
35 34 37 36 31 30 33 32 3d 3c 3f 3e 39 38 3b 3a   54761032=<?>98;:
25 24 27 26 21 20 23 22 2d 2c 2f 2e 29 28 2b 2a   %$'&! #"-,/.)(+*
d5 d4 d7 d6 d1 d0 d3 d2 dd dc df de d9 d8 db da   ................
c5 c4 c7 c6 c1 c0 c3 c2 cd cc cf ce c9 c8 cb ca   ................
f5 f4 f7 f6 f1 f0 f3 f2 fd fc ff fe f9 f8 fb fa   ................
e5 e4 e7 e6 e1 e0 e3 e2 ed ec ef ee e9 e8 eb ea   ................
95 94 97 96 91 90 93 92 9d 9c 9f 9e 99 98 9b 9a   ................
85 84 87 86 81 80 83 82 8d 8c 8f 8e 89 88 8b 8a   ................
b5 b4 b7 b6 b1 b0 b3 b2 bd bc bf be b9 b8 bb ba   ................
a5 a4 a7 a6 a1 a0 a3 a2 ad ac af ae a9 a8 ab aa   ................
ab 54 57 56 51 50 53 52 5d 5c 5f 5e 59 58 5b 5a   .TWVQPSR]\_^YX[Z
45 44 47 46 41 40 43 42 4d 4c 4f 4e 49 48 4b 4a   EDGFA@CBMLONIHKJ
75 74 77 76 71 70 73 72 7d 7c 7f 7e 79 78 7b 7a   utwvqpsr}|.~yx{z
65 64 67 66 61 60 63 62 6d 6c 6f 6e 69 68 6b 6a   edgfa`cbmlonihkj
15 14 17 16 11 10 13 12 1d 1c 1f 1e 19 18 1b 1a   ................
05 04 07 06 01 00 03 02 0d 0c 0f 0e 09 08 0b 0a   ................
35 34 37 36 31 30 33 32 3d 3c 3f 3e 39 38 3b 3a   54761032=<?>98;:
25 24 27 26 21 20 23 22 2d 2c 2f 2e 29 28 2b 2a   %$'&! #"-,/.)(+*
d5 d4 d7 d6 d1 d0 d3 d2 dd dc df de d9 d8 db da   ................
c5 c4 c7 c6 c1 c0 c3 c2 cd cc cf ce c9 c8 cb ca   ................
f5 f4 f7 f6 f1 f0 f3 f2 fd fc ff fe f9 f8 fb fa   ................
e5 e4 e7 e6 e1 e0 e3 e2 ed ec ef ee e9 e8 eb ea   ................
95 94 97 96 91 90 93 92 9d 9c 9f 9e 99 98 9b 9a   ................
85 84 87 86 81 80 83 82 8d 8c 8f 8e 89 88 8b 8a   ................
b5 b4 b7 b6 b1 b0 b3 b2 bd bc bf be b9 b8 bb ba   ................
a5 a4 a7 a6 a1 a0 a3 a2 ad ac af ae a9 a8 50 cb   .............P.
27 0c b7                                          '..
```

    Decoded Data and CRC32K (537 bytes):
    78 d6 00 3a 3f 00 00 00 00 00 00 00 01 00 01 80   x..:?...........
    00 78 3f 2e e5 00 02 e4 db e8 55 3b a0 04 00 08   .x?.......U;....
    09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18   ....
    19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28   .... !"#$%&'(
    29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38   )*+,-./012345678
    39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48   9:;<=>?@ABCDEFGH
    49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58   IJKLMNOPQRSTUVWX
    59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68   YZ[\]^_`abcdefgh
    69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78   ijklmnopqrstuvwx
    79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88   yz{|}~..........
    89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98   ....
    99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8   ....
    a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8   ....
    b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5 c6 c7 c8   ....
    c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 d6 d7 d8   ....
    d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8   ....
    e9 ea eb ec ed ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8   ....
    f9 fa fb fc fd fe ff 00 01 02 03 04 05 06 07 08   ....
    09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18   ....
    19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28   .... !"#$%&'(
    29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38   )*+,-./012345678
    39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48   9:;<=>?@ABCDEFGH
    49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58   IJKLMNOPQRSTUVWX
    59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 66 67 68   YZ[\]^_`abcdefgh
    69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78   ijklmnopqrstuvwx
    79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88   yz{|}~..........
    89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98   ....
    99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8   ....
    a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8   ....
    b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5 c6 c7 c8   ....
    c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 d6 d7 d8   ....
    d9 da db dc dd de df e0 e1 e2 e3 e4 e5 e6 e7 e8   ....
    e9 ea eb ec ed ee ef f0 f1 f2 f3 f4 f5 f6 f7 f8   ....
    f9 fa fb fc fd 9e 72 59 e2   ......rY.

```
Decompressed 6LoWPAN IPHC (558 bytes):
60 00 00 00 02 06 3a 3f aa aa 00 00 00 00 00 00    `.....:?........
00 00 00 00 00 00 00 01 aa aa 00 00 00 00 00 00    ................
00 00 00 ff fe 00 00 01 80 00 78 3f 2e e5 00 02    ..........x?....
e4 db e8 55 3b a0 04 00 08 09 0a 0b 0c 0d 0e 0f    ...U;..........
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f    ................
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f    0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f    @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f    PQRSTUVWXYZ[\]^_
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f    `abcdefghijklmno
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f    pqrstuvwxyz{|}~.
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f    ................
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f    ................
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af    ................
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf    ................
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf    ................
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df    ................
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef    ................
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff    ................
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    ................
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f    ................
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f     !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f    0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f    @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f    PQRSTUVWXYZ[\]^_
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f    `abcdefghijklmno
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f    pqrstuvwxyz{|}~.
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f    ................
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f    ................
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af    ................
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf    ................
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf    ................
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df    ................
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef    ................
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd          .............
```

Authors' Addresses

    Kerry Lynn (editor)
    Verizon Labs
    50 Sylvan Rd
    Waltham , MA   02451
    USA

    Phone: +1 781 296 9722
    Email: kerlyn@ieee.org

    Jerry Martocci
    Johnson Controls, Inc.
    507 E. Michigan St
    Milwaukee , WI   53202
    USA

    Email: jpmartocci@sbcglobal.net

    Carl Neilson
    Delta Controls, Inc.
    17850 56th Ave
    Surrey , BC   V3S 1C7
    Canada

    Phone: +1 604 575 5913
    Email: cneilson@deltacontrols.com

    Stuart Donaldson
    Honeywell Automation & Control Solutions
    6670 185th Ave NE
    Redmond , WA   98052
    USA

    Email: stuart.donaldson@honeywell.com