IEICE TRANS. COMMUN., VOL.E104–B, NO.12 DECEMBER 2021

PAPER

Backward-Compatible Forward Error Correction of Burst Errors and Erasures for 10BASE-T1S

Gergely HUSZAK[†a)], ***Student Member***, **Hiroyoshi MORITA**[†], ***Senior Member***, ***and*** **George ZIMMERMAN**[††], ***Nonmember***

SUMMARY IEEE P802.3cg established a new pair of Ethernet physical layer devices (PHY), one of which, the short-reach 10BASE-T1S, uses 4B/5B mapping over Differential Manchester Encoding to maintain a data rate of 10 Mb/s at MAC/PLS interface, while providing in-band signaling between transmitter and receivers. However, 10BASE-T1S does not have any error correcting capability built into it. As a response to emerging building, industrial, and transportation requirements, this paper outlines research that leads to the possibility of establishing low-complexity, backward-compatible Forward Error Correction with per-frame configurable guaranteed burst error and erasure correcting capabilities over any 10BASE-T1S Ethernet network segment. The proposed technique combines a specialized, systematic Reed-Solomon code and a novel, three-tier, technique to avoid the appearance of certain inadmissible codeword symbols at the output of the encoder. In this way, the proposed technique enables error and erasure correction, while maintaining backwards compatibility with the current version of the standard.

key words: *10BASE-T1S, backward-compatible FEC, burst error and erasure, Ethernet, forbidden symbols, wired IoT*

1. Introduction

The IEEE project 802.3cg (P802.3cg) [1], [2] concluded in 2019 [3], after about 3 years of research and standardization work involving several key individuals working in automotive, industrial, building automation, process control, and in-system networking technology. The project defined two 10 Mb/s baseband Ethernet Physical Layer (PHY) devices, each for use over a single balanced pair of conductors. One PHY, 10BASE-T1L, was for reaches up to 1 km, and the other PHY, 10BASE-T1S was specified for short reach applications such as automotive or in-system networks. 10BASE-T1S included a mode for shared-media, a.k.a. multidrop, operation. This project marked a return for Ethernet standards not only to 10 Mb/s speeds, but also to shared media communications, allowing more than two nodes to be attached to a single piece of wire. The shared media mode has garnered interest for extending the capabilities of 10BASE-T1S, which motivates the work in this paper.

IEEE Std 802.3 [4] uses certain nomenclature which we will use here as well. The standard refers to the "portion of the Physical Layer that contains the functions for transmission, reception, and – depending on the PHY – collision detection, clock recovery, and skew alignment" as the Physical Medium Attachment (PMA). Above the PMA, further from the medium, resides the Physical Coding Sublayer (PCS), which "contains the functions to encode data bits for transmission via the PMA and to decode the received conditioned signal from the PMA". This paper discusses constraints and design on a Forward Error Correction (FEC) approach which would reside in the PCS, as shown in Fig. 1, for reasons explained later.

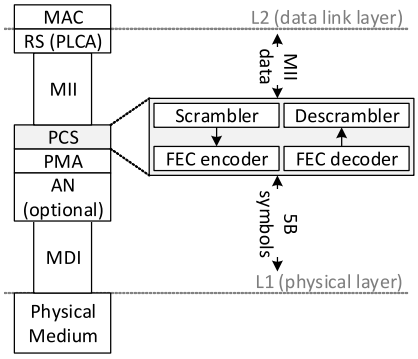

Fig. 1 Layering of functions in a typical PHY with autonegotiation (AN), forward error correction (FEC), scrambler, and reconciliation sublayer (RS).

The 10BASE-T1S PHY is a 10 Mb/s, short-reach, ultra-low complexity PHY, with an optional multidrop mode of operation. The main body of specification of 10BASE-T1S is covered by Clause 147 of IEEE Std 802.3cg-2019 [3], which deals with PMA and PCS. Additionally, a multidrop 10BASE-T1S PHY can provide packet fairness [5], bounded and calculable channel access delay, and effective throughput of near 10 Mb/s even at network saturation using the Physical Layer Collision Avoidance (PLCA) protocol specified in Clause 148 of IEEE Std 802.3cg-2019 [6], [7].

The rising interest in 10BASE-T1S multidrop operation resulted in the successful conclusion of 802.3cg being quickly followed by the formation of the IEEE 802.3 10 Mb/s Single-Pair Ethernet (10SPE) Multidrop Enhancements Study Group (SPMD), which initiated the standards project IEEE P802.3da [8] in June 2020. The new project is aimed at extending the performance and services offered while interoperating with the existing 10BASE-T1S PHYs in multidrop mode [9]. Among the areas of interest in the new project is the addition of Forward Error Correcting (FEC)

Manuscript received January 27, 2021.
Manuscript revised May 13, 2021.
Manuscript publicized June 23, 2021.

[†]The authors are with Graduate School of Information System, University of Electro-Communications (UEC), Chofu-shi, 182-8585 Japan.
[††]The author is with CME Consulting Inc., CA, USA.
a) E-mail: gergely@mail.uec.jp

DOI: 10.1587/transcom.2021EBP3016

coding. This especially benefits industrial use-cases [10] with external incidental and periodic impulse noise. An FEC would permit sending frames so that retransmission of the original packet could be avoided when errors (and possibly erasures) affect the data flow between a sender and any receivers on the multidrop network segment.

Traditionally, point-to-point Ethernet links, such as 10/100/1000BASE-T, have dealt with feature enhancements by using an Autonegotiation (AN) protocol to signal the capabilities of the PHY at the other end of the link and agree on the highest level of functionality common to the two PHYs. On a point-to-point Ethernet link AN affects only two PHYs making negotiation and single-ended upgrades simple. However, when the PHYs are on a shared media mixing segment, an extension of the AN approach[†] would limit all PHYs on the mixing segment to the capabilities of the least capable node on the mixing segment, making upgrades more difficult. Because impulse noise might be location-dependent in an operational environment, it is possible that only a subset of network nodes might be impacted by noise, creating a situation where some nodes might gain more benefit from upgrading to FEC transmission and reception than others.

An alternative to requiring upgrade of all the nodes on a shared segment would be to provide a method of coexistence so that messages intended for a node capable of decoding the new FEC could be transmitted without spreading errors throughout the network from PHYs incapable of decoding the FEC. While there are many well-known error-correcting codes [11] which might be used, the unique primary challenge in this case is backwards compatibility. Backwards compatibility requires that the encoding is formulated so that any new (FEC-enabled) nodes are able to coexist on the same shared medium as the nodes previously specified in IEEE 802.3cg without knowledge of the FEC. In short, any FEC must fit within the existing line coding scheme and not cause existing 802.3cg nodes to forward erroneous frames to their MACs. Achieving these goals while keeping PHY complexity (measured in gate count) and encoding/decoding latency low are the primary objectives of this paper.

To remain compatible with the present 802.3cg 10BASE-T1S PHY with PLCA (PPHY), while also implementing an FEC with known burst and erasure correcting capabilities, a new type of PHY (NPHY) must meet a specific set of conditions described in detail later in our paper. We show that these conditions can be met by the careful selection and combined application of a set of techniques. These techniques take advantage of the inherent and unused redundancy present in the PPHY's 4B/5B mapping while systematically avoiding certain inadmissible 5B symbols, subsequently referred to as Forbidden Symbols (FS), in order to control how the new frames are interpreted by the receiver in a PPHY.

It should be noted that while the problem is presented here in regards to the specific coding in 10BASE-T1S, the method described may be used to enhance any similar system where code groups are used to encode control information along with transmitted data in a network, allowing both enhanced and legacy nodes to be supported.

[†] At the time of writing this article, no extension of AN exists that would be capable of operating on a mixing segment.

Table 1 Terms and definitions.

Term	Definition	Reference
NPHY	The new, improved, 10BASE-T1S PHY proposed by this paper	Sect. 1
PCS	Physical Coding Sublayer	
PLCA	Physical Layer Collision Avoidance, specified by Clause 148 of IEEE Std 802.3cg-2019	
PMA	Physical Medium Attachment	
FS	Forbidden (inadmissible) 5B symbol	Sect. 1, Table 2
DME	Differential Manchester Encoding	Sect. 2
PPHY	The present 10BASE-T1S PHY specified by Clause 147 of IEEE Std 802.3cg-2019 [3]	
DS	5B data **symbol**, or sequence of these, carrying concatenated user data	Sect. 3.2.2, Fig. 5
MS	4B user data **symbol** (a.k.a. nibble), or sequence of these, conveyed via MII	
PS	Parity **symbol**, or sequence of these, of the RS codeword	
SB	Signaling **bit**, or sequence of these, in the RS codeword used for FS avoidance	
FTR	FS transcoding recipe	Sect. 4.2
LSB	Least Significant Bit	Sect. 4

1.1 Outline of the Paper

Section 2 introduces the inner workings of the 10BASE-T1S PHY, including the necessary conditions to remain backward-compatible with it, and what constitutes an FS at different parts of the FEC codeword. Section 3 explains how the unused redundancy present in PPHY's 4B/5B mapping may be used to implement a backward-compatible FEC for burst errors and erasures, including restrictions on the rate, field size and code parameters, and it shows that candidate codes exist. This is followed by Sect. 4, which proposes novel techniques and their unique combination, and various constraints related to avoiding the appearance of FS in the codeword. Section 5 discusses techniques to implement error-resilient framing to achieve arbitrary burst error correcting capabilities. Section 6 analyses the scheme's error burst error correcting capabilities and the encoding delay, and describes the method through which its correctness was verified. The last 2 sections state this paper's conclusions (Sect. 7), and highlight some promising future research directions rooted in the results presented herein (Sect. 8).

Additionally, several of the results are explained through the $\{19, 19\}$ coding scheme and an example based on that (Sect. 4.7), because, as we will show in Sect. 4.6, this is the simplest coding scheme that may exist.

1.2 Channel Model

Following industrial and automotive requirements, our paper assumes a binary burst error channel with optional erasure detection, where the erasure detection may rely on side-channel information coming from a receiver that is capable of signaling erasure.

The burst error environment assumed here is rooted in IEC 61000–4–4 Electrical Fast Transient (EFT) [12] test common in industrial systems. This test exposes the communications link to a sequence of 50 ns disturbance pulses in the test setup shown by Fig. 2. This arrangement is analogous to the schematic diagram of Shannon's general communication system [13]. Given that the line-encoded bit time in

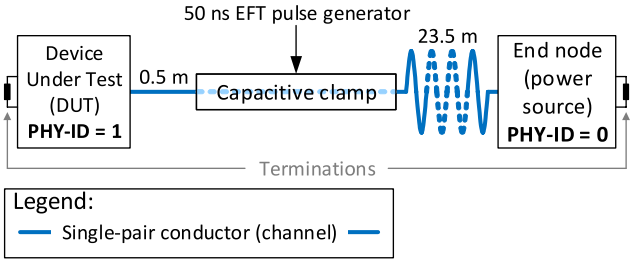

Fig. 2 IEC 61000–4–4 electrical fast transient (EFT) test setup.

10BASE-T1S is 80 ns, 50 ns stimuli would be expected to influence at most 2 bits per pulse. In practice, however, due to the common mode impedance the EFT pulse coupled to the differential pair spreads in time and can cause errors on the channel for up to 450 ns per pulse. This has been shown by analogue simulations [14] and results in up to 6 bits of burst errors.

We believe that these values represent a test setup that is specific to industrial environment, and thus our work is aimed at leaving burst error correcting capabilities as a free variable, possibly configured for each frame separately as required by the specific system and the actual, assumed, or predicted status of the segment.

2. The Present 10BASE-T1S PHY (PPHY)

2.1 The PPHY Frame

An Ethernet frame is encapsulated into a 10BASE-T1S frame, which starts off with a fixed sequence of 5B symbols with carefully crafted auto- and cross-correlation properties [15], followed by five 5B symbols (25 bits) encoding a per-frame lock sequence for the 17-bit scrambler of the PPHY. The lock sequence is followed by the actual data received from the MAC/PLS, and the frame is terminated by the appropriate End Sequence Delimiters (ESD), indicating success or failure of the frame transmission attempt.

2.2 FEC in the Abstract Layering

The PPHY incorporates a multiplicative scrambler [16], to eliminate unacceptable electromagnetic emissions[†] from so-called "killer packets" containing periodic data patterns. The multiplicative scrambler operates only on payload data and not on 4B/5B encoded control symbols. Because multiplicative scramblers propagate errors in the received data sequence, it is desirable to place the FEC to operate on the scrambled data sequence to minimize errors into the descrambler at the receiver as shown in Fig. 1.

2.3 4B/5B Encoding

Clause 147 defines a specific 4B/5B mapping[††] as follows:

[†]Subclause "147.5.4.4.2 PSD mask" in IEEE Std 802.3cg-2019 [3].

[††]Table "147–1—4B/5B Encoding" in IEEE Std 802.3cg-2019 [3].

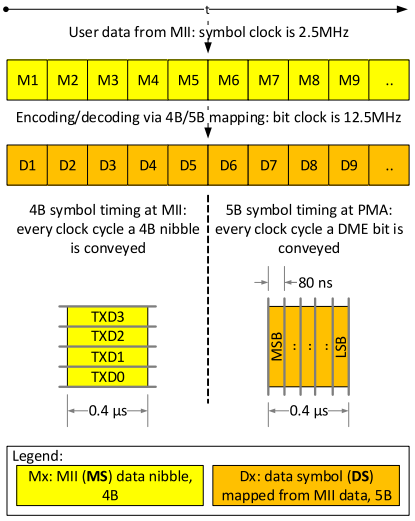

Fig. 3 Timing of 4B/5B mapping of PPHY.

1. 4-bit user data nibbles (4B symbols) at the Media-Independent Interface (MII) are mapped to 5-bit (5B) symbols of the PMA to be transmitted on the wire using Differential Manchester Encoding (DME);
2. 8 of the remaining 16 5B symbols are used to control functions of the PCS and the PLCA [17], [18];
3. The remaining 8 5B symbols are unassigned and unused.

At the MII, 4-bit nibbles are clocked at a rate of 2.5 Mb/s, while the PMA is handling DME bits at 12.5 Mb/s, the details of which are shown by Fig. 3. As visible in the figure, a DME encoded 5B symbol is transmitted by the PMA in the same amount of time (nominally 400 ns) as a 4B nibble is received via the MII.

2.4 Backward Compatibility

In order for 10BASE-T1S NPHYs and PPHYs to be backwards compatible and coexist on the same network segment, the following requirements must be met:

1. Any 10BASE-T1S PHY must be able to receive the bit stream predictably. This means that the PMA of both the PPHY and NPHY must transmit and receive DME bits at a rate of 12.5 Mb/s (i.e. 2.5 MHz for 5B symbols);
2. PPHY's PCS receive (PCS_RX) function must manifest predictable behavior when receiving a coded sequence from an NPHY;
3. Transmission of NPHY should not produce control sequences that would disrupt the PLCA cycle of PPHY;
4. The data rate at the MAC/PLS interface of 10 Mb/s must be maintained, preferably without the need for buffers within the PHY. This is desirable, because such buffers would have to scale with the size of the largest Ethernet frame transmitted.

Achieving the 1st criterion constrains the NPHY to

DME transmission of 5B symbols at a 12.5 Mb/s line rate. This, together with the 4th criterion of a 10 Mb/s rate at the MAC/PLS interface, implies that whatever encoding is used must have a rate of no less than $4/5 = 0.8$.

2.4.1 Backward Compatibility with PPHY's PCS

Detailed analysis and simulation (shown in Sect. 6.2.1) of the PPHY's PCS_RX state diagram, the essence of which is depicted in Fig. 4, showed that to prevent a PPHY from going through an unpredictable sequence of states in all cases when receiving a frame from NPHY, its PCS_RX has to be locked into the controllable iteration. The iteration is highlighted by the brown dotted area in Fig. 4 and is implemented by the DATA state itself. The details of the criteria necessary for this are summarized by Table 3, from which it is apparent that receiving certain 5B symbols ('T', 'R', and 'I' represented by the binary values shown by Table 2) cause the PCS_RX to exit DATA state.

It is worth noting that another cyclic sequence of states exists in the PCS_RX that may also allow locking the receiver. This sequence, the WAIT_SYNC→(SYNCING→(COMMIT→(WAIT_SSD→)))WAIT_SYNC→.., is highlighted by the light blue dotted area in Fig. 4, and is there to decode the preamble of the packet. However, this cycle is not useful because it interferes with the operation of the PLCA by deasserting RX_DV (see next subsection for more details).

As explained in Sect. 2.1 a PPHY frame is terminated by ESD. In a noisy environment special care must be taken to maintain error resilience not only for the payload, but also for this component of the packet. To be able to signal end of frame – and possibly additional side-information – with error resilience that is at least as good as that of the payload, an additional (4th) 5B symbol is reserved from the set of those 5B symbols withou a defined mapping in the PPHY. In this paper we will refer to this FECESD FS as 'X', forming the 4th element of the FS set we build on. For the reasons explained in Sect. 5.1, 'X' should be treated as an FS only among user data symbols (DS) of the codeword.

In short, an NPHY can remain compatible with the PPHY's PCS by using an encoding which excludes a specific set of 5B symbols from its alphabet while in the DATA. This locks the PPHY receiver's PCS_RX in the DATA state and prevents the receiver propagating the data to the MAC.

In addition to this, the received frame should be specifically marked as bad by the PPHY. This is achieved by ensuring a PPHY transitions from DATA to WAIT_SYNC through BAD_ESD, as that asserts the RX_ERR MII signal, and therefore rules out the possibility that the data may be erroneously received by the PPHY's MAC. This transition through BAD_ESD can be ensured by terminating the frame with a 'T' followed by a 'K'[†].

[†]'K' has the special function ESDERR and binary values 10001 assigned to by Table "147–1—4B/5B Encoding" in IEEE Std 802.3cg-2019 [3].

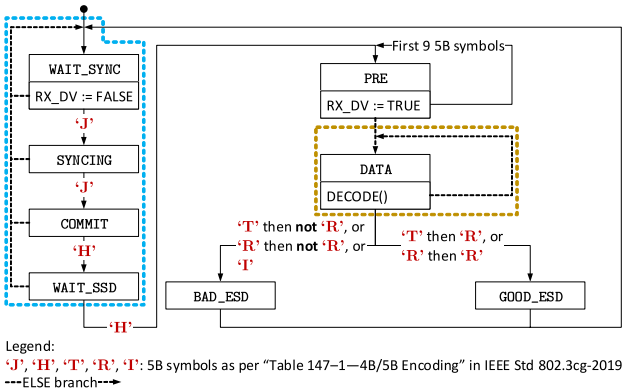

Fig. 4 Summary of the 2 cycles in the PCS_RX of PPHY.

Table 2 Forbidden symbols (FS) for the NPHY.

In Clause 147 (4B/5B)			
Symbol name	Special function	Binary value	Exponential and polynomial forms
'T'	ESD, HB	01101	$\alpha^8 = \alpha^3 + \alpha^2 + 1$
'R'	ESDOK, ESDBRS	00111	$\alpha^{11} = \alpha^2 + \alpha + 1$
'I'	SILENCE	11111	$\alpha^{15} = \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
Not in Clause 147			
'X'	FECESD	*(as applicable)*	

Note: the exponential forms are shown for the example case where the field generator polynomial $p(x) = x^5 + x^2 + 1$ over GF(2^5), where α is the primitive element (root) of the field

Table 3 Exit scenarios from state DATA of PCS_RX of the PPHY.

	RX_{n-3}='I'	RX_{n-3}='T'	RX_{n-3}='R'	ELSE	
				RX_{n-1}='R'	$RX_{n-1}\neq$'R'
RX_{n-2}='T'	*BAD*	–			*BAD*
RX_{n-2}='R'		*GOOD*		–	
ELSE		–			

Legend:
GOOD: The transition DATA→GOOD_ESD
BAD: The transition DATA→BAD_ESD
–: No transition from state DATA

2.4.2 Backward Compatibility with PLCA

Analysis and simulation (shown in Sect. 6.2.1) of the PLCA Control function[††] (PLCA_CTRL) reveals that the correct counting of PLCA Transmit Opportunities requires PLCA_CTRL to be locked in the RECEIVE state for the duration of the frame reception. This is guaranteed only if PCS_RX of PPHY is first locked in the DATA and leaves it via BAD_ESD as described in the previous subsection. This is sufficient to guarantee compatibility at the Reconciliation Sublayer, as from the perspective of the signaling, the new type of frame from an NPHY is indistinguishable from a normal frame from a PPHY, and will not affect PLCA_CTRL.

[††]Subclause "148.4.4 PLCA Control" in IEEE Std 802.3cg-2019 [3].

2.4.3 Summary of Backward Compatibility

To summarize the criteria for an NPHY's FEC-encoded transmission to be backward compatible with a PPHY, it has to use the nothing more than than redundancy present in PPHY's 4B/5B mapping and must avoid the appearance of FSs in the codewords. Assuming that $|\mathrm{FS}| = 4$, where the operator $|S|$ denotes the cardinality of set S, the quantity of this unused information is $\log_2(32-4)-4 \approx 0.8$ bit per 5B symbol.

3. The FEC of Choice

3.1 Considerations for Choosing the FEC Scheme

The sensor and IoT applications desired for 10BASE-T1S require low-complexity implementations, and hence our research is driven towards known low-complexity coding techniques. Research within 802.3 showed that linear block codes have considerable history in Ethernet [4], [19], [20], which oriented us in this direction. We therefore focus on ways to use coding structures familiar in industry that have existing, proven implementations.

A linear block code is characterized by an $(n, k, d_{\min})$ triplet, where n denotes the number of codeword symbols, k stands for that of message symbols, and $d_{\min}$ represents the minimum distance of the code [22].

As shown later in Sect. 3.2.3, short block-length is not only preferred to minimize encoding delay at the relatively slow line rate of 10 Mb/s, but is required by existence constraints for the coding scheme.

In order to enable the FS to be escaped before encoding, as described under Sect. 4, and to take advantage of code shortening, our work further focused on systematic codes.

Maximum Distance Separable (MDS) codes are linear block codes which guarantee that $d_{\min} = n - k + 1$ [21]. In other words MDS codes are $(n, k, n-k+1)$, often referred to as (n, k), and $t = \lfloor (n-k)/2 \rfloor$, where t denotes the maximum number of correctable errors. As described in Sect. 2.4.3 the inherent and reusable redundancy present in the 4B/5B mapping used by the PPHY is relatively small, and therefore MDS codes are the preferred choice for establishing error resilience with low complexity.

Our research within IEEE 802.3 and that involving vendors of FEC IP blocks showed that while there is a multitude of proven FEC implementations using linear codes and the industry has ample experience with them, these all work over binary (extended) fields. Practical implementations of FEC schemes using ternary or larger base fields were not found by the researchers,and hence our results focus on codes from binary (extended) fields.

However, it is known [22] that the only MDS binary codes that exist are the trivial (n, n), the repetition $(n, 1)$, and the Single Parity Check $(n, n-1)$ codes. Therefore our research has focused on Reed-Solomon (RS) codes [23] over extended binary fields.

3.2 Reed-Solomon (RS) Error Correcting Codes

RS codes [23] are a group of linear cyclic MDS codes over a finite field (GF) that can be used in systematic mode. They belong to the family of Bose-Chaudhuri-Hocquenghem (BCH) codes [24], [25], and satisfy all the criteria listed in Sect. 3.1.

Moreover RS codeword shortening allows fine-grained adjustment of code performance. Additionally, encoders and decoders for known code parameters can be considerably optimized, and a multitude of efficient and proven silicon implementations exist for extended fields where the base prime is 2 [26]. For these reasons RS codes are a perfect fit for the problem at hand, subject to the constraint that a method can be found to avoid FS.

3.2.1 Field Size vs. Interleaving, and RS Code Parameters

A key attribute of an RS code is the finite field (GF) over which it is defined. FEC IP blocks are commonly optimized [27] to use pre-calculated lookup tables to speed up arithmetic operations between field elements (scalars), and polynomials over these. As the sizes of these lookup tables scale quadratically with the cardinality of the field [27], it is essential to keep the field size small to manage complexity. The reduction in burst-error correction capability may be offset by the well-known technique of interleaving. A lower bound on the field size is present if interleaving is used: if the number of bits necessary to represent all field elements is not divisible by $\log_2(32) = 5$, interleaving does not achieve the increase in these capabilities. Therefore the smallest extended field for our FEC is GF(2^5), also referred to as GF(32).

Let $[\alpha, \beta]$ denote the closed integer interval with the minimum and maximum values of α and β (respectively). It is known [26] that over GF(32) an (n, k) RS code exists, such that $n \leq 32 - 1 = 31$, and code shortening makes it possible to choose any integer in $[1, n - d_{\min} + 1]$ as k. This is because the unused input symbols are replaced by a pre-agreed constant symbol pattern, and these can be omitted during the transmission of the codeword due to the fact that the code is systematic. In this paper we will refer to this RS code through its parameters $(n, n-2t)$.

3.2.2 Encoding Process

To summarize previous subsections, we can state that an $(n, n-2t)$ RS code over GF(32) is applicable to the problem at hand. An encoding process would first receive 4B user data nibbles from the MII, concatenate them, apply the necessary technique to avoid the appearance of FS anywhere in the codeword, and finally feed the resulting block to the RS encoder. The encoded block is passed to the PMA for DME-encoded serial transmission over the wire.

Before going further with our observations, let us introduce 2 new parameters as follows:

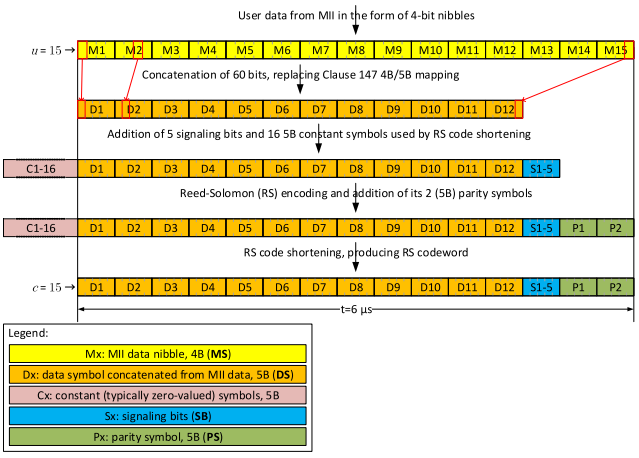

Fig. 5 A possible encoding process of a $\{15, 15\}$ coding scheme, using a (15, 13) RS FEC over GF(32).

- c denotes the total number of 5B symbols in the RS codeword, including the parity check symbols, thus $c = n = k + 2t$;
- u represents the total number of 4B user data symbols, thus $u \leq \lfloor 5k/4 \rfloor$.

There are at least 3 things worth noting here:

- This process created an FEC codeword consisting of some 5B symbols concatenated from u 4B user data symbols (referred to as DS), some signaling bits (SB), and $2t$ parity symbols (PS);
- The FEC codeword created has the exact same length in time domain as the user data, which consisted of 4B symbols (denoted as MS);
- The signaling bits, SB, are free bits that may be used to avoid the appearance FS anywhere in the codeword, the details of which is described under Sect. 4.

From hereon we will refer to such a construct as a $\{c, u\}$ coding scheme. A summary of all the parameters and the encoding process described above is depicted in Fig. 5 for the case $c = 15$ and $u = 15$.

Figure 5 also shows the additional step necessary for code shortening, during which transmitter and receiver use a previously agreed constant pattern at $|\mathrm{GF}| - 1 - c$ 5B symbol positions, referred to as Cx. In the example shown by Fig. 5 this is $32 - 1 - 15 = 16$, and those 16 5B symbols are referred to as C1-C16. Note that implementations may omit the step of filling these locations without penalty if constant symbols represented by all zero bits are used [27]. In the later part of our paper we will not show these steps either.

3.2.3 Existence of Code Parameters

Section 3.2.2 showed an encoding process that relies on an example selection of values for the coding parameters c and u, however it is apparent that these are not free variables. In this subsection we list and explain the 3 basic criteria (necessary conditions) that need to be met for a coding scheme candidate to exist:

1. The rate r of the code, which is defined as $r = 4u/5c$, must satisfy $r \geq 0.8$ to meet the backward-compatibility criterion described in Sect. 2.4, thus:

$$u \geq c \tag{1}$$

2. The number of bits represented by MS must fit into the DS that are present in the codeword, thus if t denotes the number of correctable 5B symbols errors, and ℓ denotes the number of bits embodying SB, then:

$$5(c - 2t) \geq 4u + \ell \tag{2}$$

3. The number of SB shall be sufficient to be able to point to the first FS in the linked list, as described under Sect. 4.3, thus:

$$\ell \geq \log_2(\lfloor 4u/5 \rfloor + 1) \tag{3}$$

From (2) it follows that $\ell \leq 5(c - 2t) - 4u$ and if we maximize for ℓ, which we will subsequently do, then $\ell = 5(c - 2t) - 4u$.

Figure 6 shows the relationship between all possible values of c and u, and how each of necessary conditions gets satisfied throughout the parameter space, under the assumption that $t = 1$ using the following color scheme:

1. Black border (encircling the lower triangle and the diagonal it forms) shows positions where (1) is satisfied;
2. Yellow border (surrounding approximately the upper triangle) shows positions where (2) is met;
3. White border (surrounding a slightly different part of the upper triangle) shows positions where (3) is satisfied.

For discussion on the cases when $t > 1$, see Sect. 5.4.

4. Avoiding Forbidden Symbols (FS)

Section 3 described a set of basic existence criteria and construction techniques that allow creating different $\{c, u\}$ coding schemes that meet the backward-compatibility requirements described in Sect. 2.4 with the exception of avoiding the FS. We therefore refer to these only as candidates, since they are not yet complete solutions. In this section we offer a solution for avoiding the FS.

Referring to Fig. 5, there are 3 separate parts of a codeword where FS avoidance for the 5B domain has to be implemented. Each of these are addressed separately by the subsections herein, as follows:

1. FS among the DS: the concatenated 4B user data nibbles received from the MII (MS);
2. FS among the SB: the ℓ signaling bits that became available as a result of the DS concatenation process described under Sect. 3.2;
3. FS among the PS: parity symbols (PS) created by the RS encoder through polynomial division.

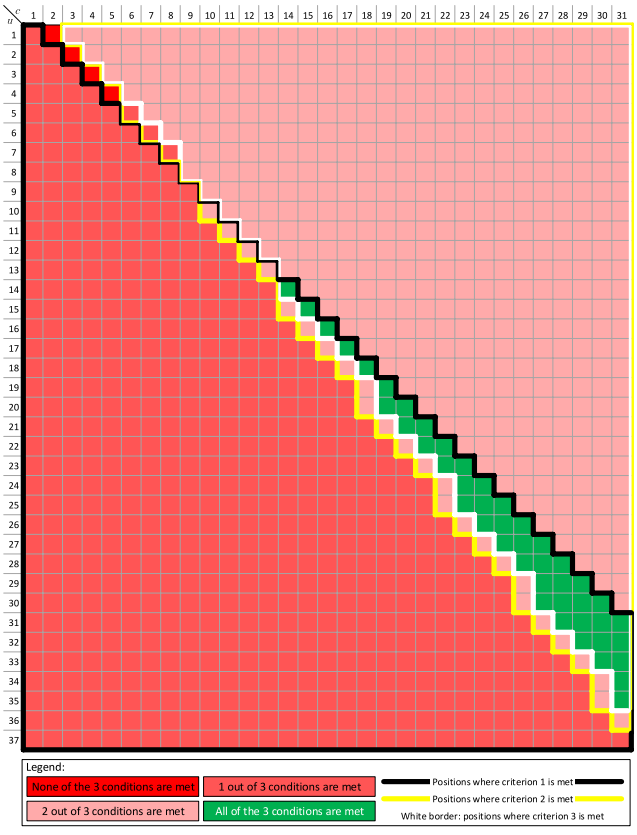

Fig. 6 Existence of all $\{c, u\}$ coding scheme candidates for $t = 1$.

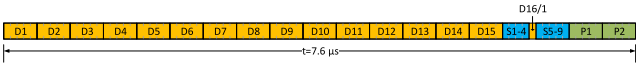

Fig. 7 Layout of a $\{19, 19\}$ coding scheme in a (19, 17) RS codeword.

Our paper proposes a unique combination of 3 novel techniques, each handling one of the 3 areas, as described by the following subsections. To the best of the authors' knowledge, these techniques have been not considered before either alone or in conjunction.

4.1 Layout of the RS Codeword

An implementation may choose an arbitrary ordering of DS, SB, and PS. However, we will subsequently use the arrangement shown by Fig. 7, as it maintains the "LSB first" bit order defined by Clause 147, it also allows pre-coding, and it is compatible with the future extensions proposed in Sect. 8.

4.2 FS among the DS

The 5B symbols comprising the user data are formed by the concatenation of the unconstrained (free) user input data from the MAC, as shown by Fig. 5. Therefore no general assumption can be made with regards to their values. In this subsection we are discussing only DS that are completely made up by MS. The mechanisms for MS and SB appearing in a mixed manner are described in Sect. 4.3.

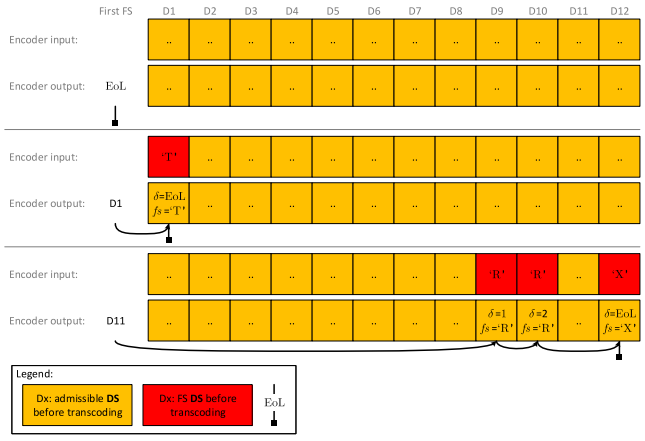

Fig. 8 Some simple example linked lists over 12 DS.

Because the encoding of the location of the FS must meet the rate requirement, something more efficient than a simple bitmap encoding is needed. We propose the combined application of the following 2 methods to eliminate FS among the DS:

1. A linked list that walks through all the FS present in a forward-only manner while transcoding each FS to an admissible 5B symbol;
2. A special constellation-ID that allows forming the linked list in those cases when the distance between any 2 neighboring FS is too large to be directly represented.

A naïve approach to form the linked list would be as follows:

1. Encode in the SB the index of the first FS among the DS, or encode End of List (EoL) if the DS contain no FS;
2. Transcode the FS pointed to by the SB to an admissible 5B symbol value. The transcoding encodes the value of the FS that was replaced (subsequently referred to as fs) and the distance to the next FS in the array of DS (or EoL) (subsequently referred to as δ);
3. This process is repeated until no FS appears among the DS.

Because all FS are transcoded by the linked list, and all the transcoded fs-δ pairs must be represented only by admissible symbols, no FS appear in the output of this iterative process. Figure 8 shows some simple examples of how this works under a $\{15, 15\}$ coding scheme.

However, this naïve approach is not yet complete. Let us make the following observations:

1. An unconstrained 5B symbol can represent $2^5 = 32$ values, which must encode both δ and fs at the same time;
2. we have 4 FS that we have to consider among DS, decreasing the number of admissible 5B symbol values, which may be used to encode δ, from 32 to 28;
3. δ has to also be able to encode a value representing EoL.

From these observations it follows that the

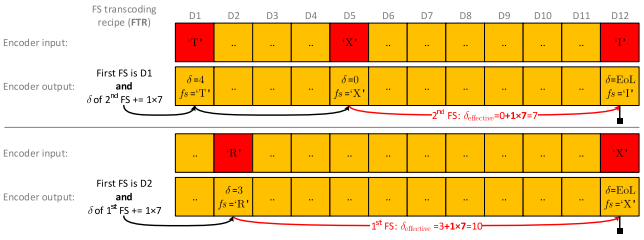

Fig. 9 Some complex example linked lists over 12 DS.

naïve method does not allow δ to be larger than $\delta_{\max} = \lfloor (32-4)/4 \rfloor - 1 = 6$. For example if D5 and D12 were FS, the δ part of the 5B symbol value transcoding D5 would be unable to represent the distance of $\delta = 7$ between these.

This problem is resolved by the application of the special constellation-ID mentioned above, which carries additional information with respect to the position of all FS before transcoding is carried out. The special constellation-ID indicates either:

1. None of the distances between any 2 neighboring FS is larger than the $\delta_{\max}$;
   or
2. There exist one or more neighboring FS, the raw distance ($\delta_{\text{effective}}$) between which is larger than $\delta_{\max}$. For these do the following:
   a. Let δ for those be $\delta = (\delta_{\text{effective}} \bmod (\delta_{\max} + 1))$;
   b. Remember both the sequence number and $\lfloor \delta_{\text{effective}} / (\delta_{\max} + 1) \rfloor$ for those.

Now, if we make SB encode the index of the first FS among the DS as well as this special constellation-ID, this method provides a complete solution capable of eliminating all FS among the DS, irrespective of their actual values and locations. In the later part of our paper, we refer to this construct encoded by the SB as an FS Transcoding Recipe (FTR). Figure 9 shows how FTRs work in some example cases under a $\{15, 15\}$ coding scheme.

4.3 FS among the SB

In this subsection we discuss how many signaling bits (referred to as SB) are needed to represent all necessary FTRs, while also making sure no new FS are created by the SB during the process.

As explained SB are ℓ bits in the codeword, from among which (ℓ mod 5) bits form a mixed DS, in which user data from the MS and SB coexist, while the remaining $\lfloor \ell/5 \rfloor$ 5B symbols are formed by purely SB, as shown by Fig. 7.

In Table 4 we show how many signaling values can be encoded by a 5B symbol with any SB in it. Table 4 also explains what these values are rooted in. From this, for example, it is visible, that a $\{19, 19\}$ coding scheme, where $\ell = 4 + 5 = 9$, can encode $13 \times 29 = 377$ FTRs, so that 9 SB (bits) can carry $\log_2(377) \approx 8.6$ bits of information, while avoiding the appearance of FS among the 5B symbols.

Table 4 Relationship between number of SB in a 5B symbol and the number of signaling values it can encode.

Number of		A possible method of optimal construction
SB in the 5B symbol	signaling values	
1	1	LSB or the central bit must be picked and it needs to be 0 to avoid FS: $2^1 - 1 = 1$
2	3	Assign LSB and the central bit, and avoid using 1-1: $2^2 - 1 = 3$
3	6	Same as above, then pick any 3rd bit: $2(2^2 - 1) = 6$
4	13	Pick any 4 bits and avoid the 3 patterns FS have their: $2^4 - 3 = 13$
5	29	Simply avoid using the 3 FS: $2^5 - 3 = 29$

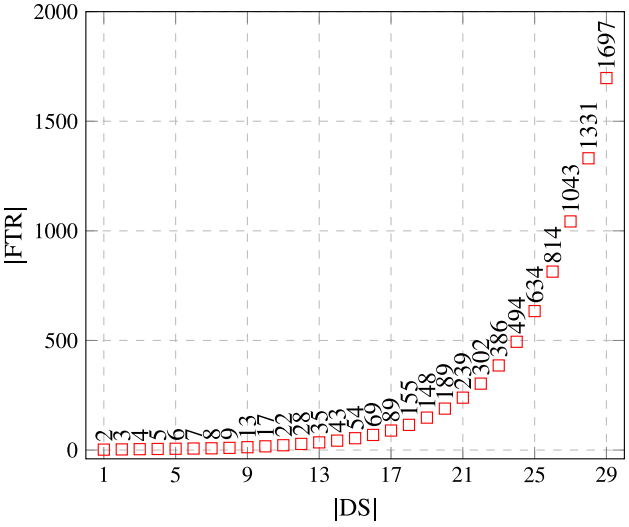

Fig. 10 The relationship between number of complete DS (horizontal axis) and the number of FTRs each requires (vertical axis).

As discussed in Sect. 4.2, one of the main roles of SB is to encode the FTR required by the coding scheme to avoid FS among the DS. It is obvious however that the number of FTRs needed by the scheme to operate scales with the number of complete DS in it. This relationship is depicted in Fig. 10, the values in which were determined by an exhaustive search. It can be seen, for example, that a $\{19, 19\}$ coding scheme with 15 complete DS requires 54 FTRs. We believe that this graph can also be expressed by a closed, recursive, combinatorial formula, but the details of this are outside of the scope of our work presented herein.

Note that some of the information available in the SB will have to be used to avoid FS appearing among the PS. This role is assigned to the terminal 5B symbol that comprises of SB. In short, non-terminal 5B symbol encoding SB represent only the FTR, while the terminal one supports this, and contributes to avoiding FS appearing among the PS, the details of which is explained in Sect. 4.4.

4.4 FS among the PS

These $2t$ 5B symbols are the direct product of the RS encoding of the data bits. Because the data bits are unconstrained

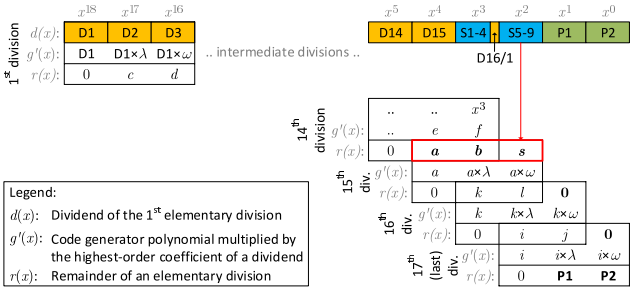

Fig. 11 The (optimized) polynomial long division of a {19, 19} coding scheme that produces PS P1 and P2.

and the RS code is MDS, there is no mapping that would avoid the appearance of FS among the PS in all cases. As a result, some additional – yet unused – free bits in the encoder's input must be reserved to influence its output.

Figure 11 shows how PS are produced according to known techniques for RS encoding by the iterative process of polynomial long division. It can be noted that for a given code generator polynomial $g(x) = x^2 + \lambda x + \omega$, the PS are fully determined by 3 components $n - k$ before the last sub-division (highlighted by the thick red caret), namely:

- a and b: the cumulative remainders of all the previous sub-divisions;
- s: the last 5 bits of SB, in this case S5-9.

This allows the normal RS encoding process to be modified so that when a and b become available during the execution of the polynomial division, s would be selected from the set of admissible 5B symbol values, so that neither P1 nor P2 would end up being FS. The direct application of this method uses the space in s efficiently, but it requires a large lookup table that selects s based on exact values of both a and b. While this process may be implementable, it may not be feasible for low-complexity systems. Therefore we propose to make this lookup considerably simpler by relying on the following 2 observations:

- All 3 FS to be avoided among the PS[†] are identical in their LSB and the central bits, as shown by Table 2;
- Galois field elements are added by modulo-2 addition of the coefficients, which in binary form means bit-by-bit Exclusive OR (XOR, denoted by $\oplus$) of the 2 scalars.

Given these observations, Fig. 11 shows that the 2 SB are a result of the following set of elementary equations over GF(32), where $\mathrm{P1}, \mathrm{P2}, a, b, i, j, k, l, s, \lambda, \omega \in \mathrm{GF}(32)$:

$$\begin{aligned} \mathrm{P1} &= j + i\lambda \\ \mathrm{P2} &= i\omega \\ i &= l + k\lambda \\ j &= k\omega \\ k &= b + a\lambda \\ l &= s + a\omega \end{aligned} \tag{4}$$

The straightforward simplification of (4) leads to the following results:

[†] As explained in Sect. 5.1 these are 'T', 'R', and 'I'.

$$\mathrm{P1} = \overbrace{a\lambda^3 + b(\lambda^2 + \omega)}^{\mathrm{P1_{left}}} + s\lambda \tag{5}$$

$$\mathrm{P2} = \overbrace{a(\lambda^2\omega + \omega^2) + b\lambda\omega}^{\mathrm{P2_{left}}} + s\omega \tag{6}$$

What is worth noticing here is that P1 is fully determined by $s\lambda$, and the same holds for P2's relationship with $s\omega$. Therefore to avoid FS among the PS, the $\{c, u\}$ coding scheme may proceed as follows:

1. Choose any bit location in the 5B symbol, where all FS have matching values: from hereon we will use LSB, as all 3 FS have the value 1 at that position;
2. Given $\mathcal{A} = \mathrm{GF}(32)\backslash\{\mathrm{'T'}, \mathrm{'R'}, \mathrm{'I'}\}$, and $v \in \{0, 1\}$, let us define the following 2 parametric partitionings of the admissible 5B symbol values, where the function LSB_of(n) returns the LSB of n, as follows:

$$\mathcal{P}_1(v) = \{x \in \mathcal{A} \mid \mathrm{LSB\_of}(x\lambda) = v\} \tag{7}$$

$$\mathcal{P}_2(v) = \{x \in \mathcal{A} \mid \mathrm{LSB\_of}(x\omega) = v\} \tag{8}$$

3. During the polynomial division shown by Fig. 11, when a and b become available, calculate $\mathrm{P1_{left}}$ and $\mathrm{P2_{left}}$, as per (5) and (6), respectively;
4. Finally, pick a single value for s such, that:

$$s \in \mathcal{P}_1(\mathrm{LSB\_of}(\mathrm{P1_{left}})) \cap \mathcal{P}_2(\mathrm{LSB\_of}(\mathrm{P2_{left}})). \tag{9}$$

As this method guarantees that both $\mathrm{LSB\_of}(\mathrm{P1_{left}} + s\lambda)$ and $\mathrm{LSB\_of}(\mathrm{P2_{left}} + s\omega)$ will always be 0, it follows that neither P1, nor P2 may turn out to be any of the 3 applicable FS.

It is worth mentioning however that the cardinalities of the 4 sets ($|\mathcal{P}_1(0)|$, $|\mathcal{P}_1(1)|$, $|\mathcal{P}_2(0)|$, and $|\mathcal{P}_2(1)|$) depend on the actual values for λ and ω. The cardinalities of these sets may each be anywhere in the interval $[5, 7]$, depending on where the 3 FS fall in these 4 sets for the given λ and ω. As per Fig. 11, it is the terminal 5B symbol that comprises of SB that needs not only to contribute to encoding the FTR, but also implements the avoidance of FS in the PS. Because of this, the number of values available to encode FTR by this symbol is in the same interval, thus it is at most 7.

4.5 Information-Theoretic Observations

It is well worth mentioning that the relationship shown by Fig. 10 may also be approached from the perspective of information theory. If the vertical axis of Fig. 10 representing the FTR required for a given number of DS is made logarithmic, as in Fig. 12, then it reflects the amount of information (in bits) needed by the DS to avoid FS. From the correlation curve, it can be observed that for every DS added ≈0.35 bits of information is needed to avoid the appearance of FS using the technique proposed in this paper. This observation will also be useful in Sect. 5.4 when we show why no $\{c, u\}$

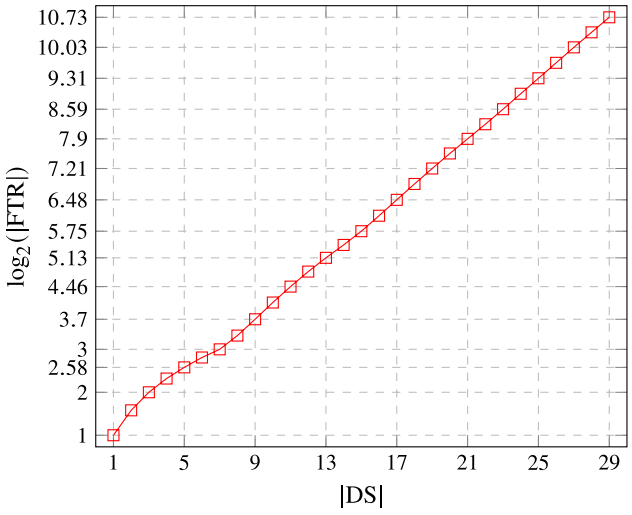

Fig. 12 Correlation between number of complete DS and the quantity of information (in bits) needed to encode all FTRs.

Table 5 Existence of the simplest $\{c, u\}$ coding schemes with FTR.

		FTRs		
$\{c, u\}$	ℓ	possible	required	May exist
$\{15, 15\}$	5	7	28	No
$\{16, 16\}$	6			
$\{17, 17\}$	7	3×7 = 21	35	
$\{18, 18\}$	8	6×7 = 42	43	
$\{19, 19\}$	9	13×7 = 91	54	**Yes**

coding schemes may exist for $t > 1$.

4.6 Existence of Optimal $\{c, u\}$ Coding Schemes with FTR

The number of FTRs required by a $\{c, u\}$ coding scheme directly influences its complexity, and as shown in Fig. 10, |FTR| scales with |DS| exponentially. Moreover the larger parameters c and u get, the smaller the relative error correcting capabilities, expressed by t/u, the coding scheme has. For these 2 reasons alone it is apparent that c and u should be chosen to be as small as possible for a given implementation.

Earlier Fig. 6 showed all $\{c, u\}$ coding schemes that satisfy the 3 basic existence criteria introduced in Sect. 3.2.3. According to that, the simplest candidate is the $\{14, 14\}$ scheme. However this would leave at most $\ell = 5(c-2) - 4u = 4$ bits for the SB, which would allow at most 13 FTRs (as shown by Table 4). However Fig. 10 indicates that for the 11 complete DS in the $\{14, 14\}$ code, 22 FTRs are required, and therefore, the $\{14, 14\}$ coding scheme is insufficient.

Similar analysis can be performed on candidates of increasing complexity, and as shown in Table 5 the simplest $\{c, u\}$ coding scheme that exists is $\{19, 19\}$. It is worth noting that the calculations under column "FTR possible" of this table follow the explanation in, and the values listed under, column "Number of signaling values" of Table 4, as well as the reasoning in Sect. 4.4. For example the number of possible FTRs for a $\{19, 19\}$ coding scheme is listed as 13×7 = 91. In this calculation the value of 13 stems from the fact that the 4 bits referred to as S1-4 in Fig. 7 can encode this many distinct values, while avoiding an FS being formed, as explained in Table 4. The value of 7 in this calculation follows the explanation in Sect. 4.4, which shows that the 5 bits referred to as S5-9 in Fig. 7 can encode at most this many distinct values, while avoiding FS being formed by themselves and by any of the 2 PS.

From these it follows, that under the assumptions and using the techniques presented in this paper a $\{19, 19\}$ coding scheme is optimal both from practical (engineering) and theoretical perspectives.

For other reasons one might choose to use a more complex implementable coding scheme. While these clearly diverge from the above-mentioned optimality, in return a more complex scheme would allow additional information to be encoded into the SB. For example, a $\{20, 20\}$ coding scheme, would allow up to 29×7 = 203 (as per Table 4) possible FTRs, while only 69 of those would be required to avoid FS anywhere in the codeword. This allows $\lfloor \log_2(203/69) \rfloor = 1$ free extra bit to be available to represent arbitrary information for every RS codeword, at a cost of decreasing the relative error correcting capability to one 5B symbol per twenty 5B symbols.

Following this thought pattern, it is apparent that using a $\{21, 21\}$ coding scheme would make little sense, as the number of possible FTRs would still be 1×29×7 = 203 with an inferior relative error correcting capability compared to that of a $\{20, 20\}$ coding scheme. Using the method described here, the relevant parameters of any arbitrary $\{c, u\}$ coding scheme may be analyzed similarly.

4.7 An Example $\{19, 19\}$ Coding Scheme

Previous sections showed the theoretical constructs behind the $\{c, u\}$ coding scheme. In this subsection we present the full details of an example $\{19, 19\}$ coding scheme based on those, covering both encoding and decoding, by the direct application of all the methods described in Sect. 4.

For this example the field and code generator polynomials have been chosen according to Sect. 6.2.2, thus $\lambda = 3$ and $\omega = 2$. The FS 'X' is selected to be represented by the decimal value 0, LSB is used to avoid FS among the PS, and the bits in the codeword are laid out according to Fig. 7. It is important to emphasize that all these choices are only to show a working example in our paper and an implementation is free to make different selections, for example for product-specific considerations. Due to the underlying construct any such scheme will correctly operate as long as all transmitters and receivers on a segment use the same constants.

A $\{19, 19\}$ coding scheme incorporates 15 complete DS, which requires being able to encode 54 FTRs, as per Fig. 10. As shown by Table 4 a single 5B symbol can encode at most 29 values while avoiding FS, and therefore two 5B symbols formed by the SB will be needed to encode the 54 FTRs. This is done by breaking down each FTR into FTR_{high} (encoded by S5-9) and FTR_{low} (repre-

Table 6 An example assignment of FTRs and meaning in a {19, 19} coding scheme.

FTR_{high}	FTR_{low}	FTR	*A*	*B*	FTR_{high}	FTR_{low}	FTR	*A*	*B*
1	1	**1**	–		5	1	**37**	D3	δ of 6th += 1×7
1	2	**2**	D1	–	5	2	**38**	D4	δ of 1st += 1×7
1	3	**3**	D2		..				
..					5	6	**42**	D4	δ of 5th += 1×7
1	9	**9**	D8		5	7	**43**	D5	δ of 1st += 1×7
2	1	**10**	D9	–	..				
2	7	**16**	D15		6	1	**46**	D5	δ of 4th += 1×7
2	8	**17**	D1	δ of 1st += 1×7	6	2	**47**	D6	δ of 1st += 1×7
..					..				
3	6	**24**	D1	δ of 8th += 1×7	6	4	**49**	D6	δ of 3rd += 1×7
3	7	**25**	D2	δ of 1st += 1×7	6	5	**50**	D7	δ of 1st += 1×7
..					6	6	**51**	D7	δ of 2nd += 1×7
4	4	**31**	D2	δ of 7th += 1×7	6	7	**52**	D8	δ of 1st += 1×7
4	5	**32**	D3	δ of 1st += 1×7	6	8	**53**	D1	δ of 1st *and* 2nd += 1×7
..					6	9	**54**	D1	δ of 1st += 2×7

Legend:
A: The first FS in the DS array
B: Special treatment of some FS: adding multiples of 7 to their δ

Table 7 An example assignment of 5B symbol values and transcoding functions usable by the DS of any $\{c, u\}$ coding scheme, if 'X' is 00000.

Binary, **decimal** values	fs	δ	Binary, **decimal** values	fs	δ	Binary, **decimal** values	fs	δ	Binary, **decimal** values	fs	δ
00001, **1**	'T'	EoL	01010, **10**	'T'	2	10011, **19**	'T'	4	11011, **27**	'T'	6
00010, **2**	'R'		01011, **11**	'R'		10100, **20**	'R'		11100, **28**	'R'	
00011, **3**	'I'		01100, **12**	'I'		10101, **21**	'I'		11101, **29**	'I'	
00100, **4**	'X'		01110, **14**	'X'		10110, **22**	'X'		11110, **30**	'X'	
00101, **5**	'T'	1	01111, **15**	'T'	3	10111, **23**	'T'	5			
00110, **6**	'R'		10000, **16**	'R'		11000, **24**	'R'				
01000, **8**	'I'		10001, **17**	'I'		11001, **25**	'I'				
01001, **9**	'X'		10010, **18**	'X'		11010, **26**	'X'				

sented by S1-4) using, for example, the one-to-one mapping FTR = 9(FTR_{high} − 1) + FTR_{low}.

Under these conditions, it is clear that (5) and (6) simplify to P1 = 15*a* + 7*b* + 3*s* and P2 = 14*a* + 6*b* + 2*s* (respectively), from which it follows that:

$$P1_{left} = 15a + 7b$$
$$P2_{left} = 14a + 6b$$
$$\mathcal{P}_1(0) = \{2, 4, 6, 8, 10, 12, 14, 17, 19, 21, 23, 25, 27, 29\}$$
$$\mathcal{P}_1(1) = \{1, 3, 5, 9, 11, 15, 16, 18, 20, 22, 24, 26, 28, 30\}$$
$$\mathcal{P}_2(0) = \{1 - 6, 8 - 12, 14, 15\}$$
$$\mathcal{P}_2(1) = \{16 - 30\}$$

Now we have everything to lay out the lookup tables that map the admissible symbols the way we like. In this example we do it using the most natural – incremental – approach, as follows:

- Table 6: The 54 FTRs are assigned values in increasing order of FTR position and complexity;
- Table 7: The 28 values used for transcoding FS among complete DS are assigned to $fs - \delta$ pairs so that fs keeps appearing in the order shown in Table 2, while δ increases sequentially;
- Table 8: The 9 values for FTR_{low} are assigned in increasing order;
- Table 9: Finally, The 6 values for FTR_{high} are assigned in increasing order, while maintaining the presence of values from $\mathcal{P}_1(0)$, $\mathcal{P}_1(1)$, $\mathcal{P}_2(0)$, and $\mathcal{P}_2(1)$ so that (5) can always be satisfied: for example in the group where FTR_{high} = 1, 1∈$\mathcal{P}_1(1)$ and $\mathcal{P}_2(0)$, 2∈$\mathcal{P}_1(0)$ and $\mathcal{P}_2(0)$, 16∈$\mathcal{P}_1(1)$ and $\mathcal{P}_2(1)$, and 17∈$\mathcal{P}_1(0)$ and $\mathcal{P}_2(1)$.

Table 8 An example assignment of 5B symbol values and FTR_{low} usable by S1-4 of a {19, 19} coding scheme.

Binary, **decimal** values	FTR_{low}	Binary, **decimal** values	FTR_{low}	Binary, **decimal** values	FTR_{low}
0001, **1**	1	0101, **5**	4	1001, **9**	7
0010, **2**	2	0111, **7**	5	1010, **10**	8
0100, **4**	3	1000, **8**	6	1011, **11**	9

Table 9 An example assignment of 5B symbol values and FTR_{high} usable by S5-9 of a {19, 19} coding scheme.

Binary, **decimal** values	FTR_{high}	Binary, **decimal** values	FTR_{high}	Binary, **decimal** values	FTR_{high}
00001, **1**	1	00101, **5**	3	01010, **10**	5
00010, **2**		00110, **6**		01011, **11**	
10000, **16**		10100, **20**		11000, **24**	
10001, **17**		10101, **21**		11001, **25**	
00011, **3**	2	01000, **8**	4	01100, **12**	6
00100, **4**		01001, **9**		01111, **15**	
10010, **18**		10110, **22**		11010, **26**	
10011, **19**		10111, **23**		11011, **27**	

4.7.1 Encoding Process

The example encoding shown by Fig. 13 consists of the following main steps:

1. First, u = 19 user data 4B nibbles (MS) from the MAC/-PLS interface are conveyed through the MII and collected by the encoder;
2. These are concatenated into 5B symbols (DS), FS are located and identified, determining the values for FTRs and providing FTR_{high} and FTR_{low};
3. All FS are transcoded using the linked list, S1-4 are filled in based on FTR_{low}, then a and b are calculated, which – in conjunction with FTR_{high} – is used to determine s, to be used directly by S5-9 in the codeword;
4. All these are fed to the (19, 17) RS encoder, which provides the systematic RS codeword with $2t$ = 2 PS;

The output codeword of this process can either be fed directly to the channel, or to the L-interleaver, to produce a superblock, which is then conveyed to the channel.

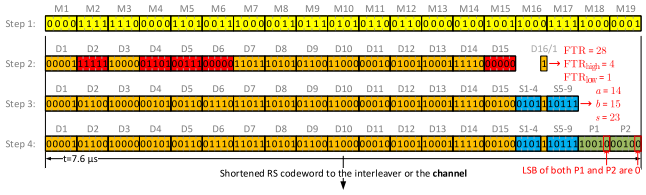

Fig. 13 A complete encoding in an example {19, 19} coding scheme.

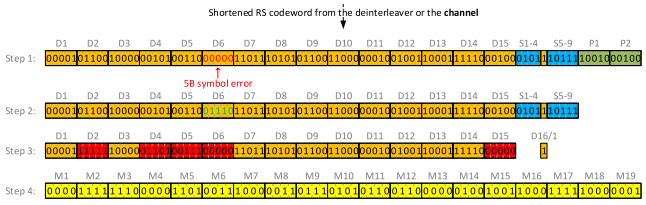

Fig. 14 A complete decoding in an example {19, 19} coding scheme.

4.7.2 Decoding Process

The example decoding shown by Fig. 14 is just an inversion of the encoding process shown in Sect. 4.7.1, thus we present it here only for the sake of completeness:

1. The codeword, with up to $t = 1$ symbol error or $2t = 2$ symbol erasures, arrives from the channel or the L-deinterleaver;
2. It is fed to the RS decoder, which either corrects the error/erasures, or signals non-correctable errors/erasures (which can be used to provide higher overall error resilience), or if the quantity of these is beyond the known error correcting capability of the code it may do a miss-correction;
3. The FTR is decoded and the linked-list is walked to undo the FS transcoding applied by the encoder;
4. Finally, the user data is separated into 4B nibbles (considering the framing in Sect. 5.1) and conveyed to the MAC/PLS interface via the MII.

This process is able to correct burst errors consisting of any combination of up to L consecutive 5B symbols.

5. Extensions to the Encoding Process

5.1 End of Sequence Delimiter under FEC (FECESD)

As presented in Sect. 2.4.1, an additional 5B symbol value is reserved to allow reliable signaling of the end of frame. In the scope of this research, when the PCS Transmit function (PCS_TX) detects end of frame, it inserts this reserved symbol followed by an additional symbol indicating success or failure of frame transmission, analogous to the way a PPHY makes either `BAD_ESD` or `GOOD_ESD` follow `DATA` state (see Fig. 4).

As the proposed FEC relies on per-frame fixed size superblocks, codeword padding is applied after the symbol that follows FECESD. Given that the symbol following FECESD may have 28 different values, this technique allows the transmission of $\lfloor \log_2(28/2) \rfloor = 3$ additional bits of information, out of which 1 bit is used to indicate presence of symbol padding in the last 5B symbol, and every frame is terminated as follows:

- If the bits embodying the received MS for the last codeword is not divisible by 5, then constant symbol padding is applied until this criterion is met;
- A FECESD 5B symbol is inserted, followed by an admissible symbol indicating a good or bad ESD and the presence of symbol padding;
- Codeword and superblock padding is/are applied as necessary;
- Finally, the fixed 5B symbol sequence of 'T' followed by a 'K' is inserted to force all MACs above PPHYs to discard the frame, as described under Sect. 2.4.1.

For the reasons explained above, FECESD needs to be treated as an FS only when it appears among 5B symbols that form a complete DS. It may indeed appear among 5B symbols formed by SB, or those encoding PS.

5.2 Configurable Burst Length via Interleaving

With the channel model introduced in Sect. 1.2 the well-known technique of interleaving can be used to improve the burst error correction capabilities of the code according to known results [21], [28]. When used with interleaving of depth L, the RS codeword of length n combines with the other interleaved RS codewords to form a superblock of length nL.

5.3 Erasure Correction

Erasure detection can rely on optional side-channel information provided by the receiver, whenever certain parameters of the received digital or analog signals are outside of the valid range. For example erasure may be assumed when the signal's swing, or DME timing is/are beyond a specified range[†], including tunable tolerances and margins.

5.4 $\{c, u\}$ Coding Schemes For $T > 1$

Until this subsection, we have discussed applicability of $\{c, u\}$ coding schemes over GF(32) under the assumption that $t = 1$, despite the fact that should any $t > 1$ schemes exist, those may provide improvement to the relative error correcting capabilities. In this subsection we focus on this undiscussed area of the solution candidate space, and show that no $\{c, u\}$ coding schemes exist beyond $t = 1$.

To achieve this, we have solved the system of 3 inequalities presented in Sect. 3.2.3 for $t \geq 1$, the output of which is visualized by Fig. 15. Following the color-coding of Fig. 6,

[†] Figure "147–13—DME encoding scheme" and Table "147–2—DME timings" in IEEE Std 802.3cg-2019 [3].

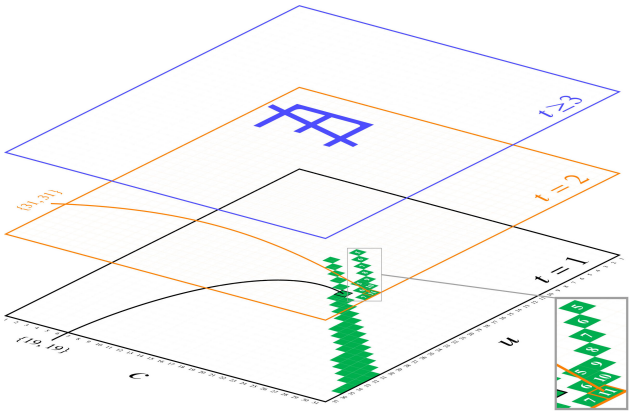

Fig. 15 Existence of all $\{c, u\}$ coding scheme candidates for $t \geq 1$ (white numbers in the green carets under $t = 2$ represent the values for ℓ).

the green squares represent the $\{c, u\}$ coding scheme candidates that meet the necessary conditions, while we omitted the red coloring for clarity. What is immediately apparent here is that solution candidates exist for $t = 2$, but the space for $t \geq 3$ is void of solutions.

As explained in Sect. 3.2.3, $\ell = 5(c - 2t) - 4u$ represents the number of bits available among the SB to encode FTR and to avoid FS among the PS. This is the factor that essentially decides whether a solution candidate may lead to an actual solution. Figure 15 shows that $t = 2$ offers 10 solution candidates, the values of ℓ for which are shown by the white numbers in the green carets. The value for ℓ is largest for $\{31, 31\}$. Therefore if we show that these $\ell = 11$ bits are insufficient for a $\{31, 31\}$ coding scheme to exist, it follows that none of the remaining 9 candidates may lead to a solution either. This is because increasing u decreases ℓ, while decreasing c does not free up even a single complete bit, as stated in Sect. 4.5.

In each $\{c, u\}$ coding scheme, the number of PS is $2t$, thus for $t = 2$ the scheme has to avoid FS for 4 PS. This will require 4 bits, and therefore a complete 5B symbol, as shown in Table 4, to encode 4 parametric partitionings $(\mathcal{P}_1 - \mathcal{P}_4)$. This is an extension of the 2 parametric partitionings for $t = 2$ discussed in Sect. 4.4. Moreover, an additional bit is needed to avoid D25 from being an FS. This leaves $11 - 5 - 1 = 5$ bits among the SB available to encode the 494 FTRs necessitated by the 24 complete DS in this scheme, as shown in Fig. 10. As 5 bits can encode at most 29 values without using FS, encoding 494 FTRs is not possible, which leads to the conclusion that no $\{c, u\}$ coding schemes exist for $t > 1$. □

6. Analysis

In this section we analyze the applicability, some of the characteristics, and the performance of the proposed scheme.

6.1 Encoding Delay

In general, systematic error correcting codes have the advantage of being encodable "on-the-fly": when the data arrives, the encoder can already start forming the codewords, producing output without delay. However in the case proposed herein, this does not stand. The reason for this is three-fold:

- The user input data concatenation process depicted in Fig. 5 causes a short delay in the output stream: e.g. to be able to create D12, M15 must be received;
- The escaped elements of the FS linked list and the values of SB may be formed only when the last 4B symbol arrives from MII;
- If interleaving is applied, it imposes an additional (fixed) delay, as the channel input can be formed only after the RS codewords at the interleaver's inputs are available.

An upper bound on the total delay caused by these factors is $5cL + \epsilon$ bit times, where ϵ denotes the implementation-dependent encoding and decoding delays in the silicon, and $5cL$ is the size of the superblock in time domain.

In contrast, for a PPHY that works without FEC and relies on retransmissions, the lower bound for these delays is the total transmission time for the packet. In the case of Ethernet this is 512 bit times, in addition to the delays imposed by the higher layers carrying out the retransmissions. The packet retransmission delays are – typically several magnitudes – larger than the delays attributed to our proposed scheme with any reasonable value for L, irrespective of whether the higher layers use a positive or a negative acknowledgement scheme for triggering a retransmission.

Additionally, actual implementations hiding [29] or incorporating [30] the MII interface may partially or completely eliminate encoding and decoding delay, benefiting the proposed scheme.

6.2 Verification of the Results

To verify the results of this research, our work included the complete implementation of two independent programs in C, consisting of a total of over 7000 lines of code.

6.2.1 Verification of Conclusions on PPHY

A verbatim, unoptimized implementation of a PPHY, including complete Clause 147 and Clause 148 functionality has been carried out to confirm PPHY behavior, and to verify results. This simulator runs all Finite State Machines (FSM) of these 2 clauses in a synchronous manner to execute a configurable number of nodes over a mixing segment with and without PLCA, including the management of physical and logical collisions. This implementation has been used to verify the observations made in this paper with respect to PLCA, the compatibility criteria, and the PCS behavior. The source code in C of this system is available via GitHub [31].

6.2.2 End-to-End Verification of the Proposed FEC Scheme

To verify the performance of the proposed FEC scheme claimed in this paper, this research has implemented the complete encoding and decoding scheme over configurable field- and code-generator polynomials. An exhaustive test over the extended field GF(32) defined by the field generator polynomial of $p(x) = x^5 + x^2 + 1$, and the code generator polynomial of $g(x) = (x + \alpha^c)(x + \alpha^{c+1}) = x^2 + 3x + 2$ (for $c = 0$) has been performed using the $\{19, 19\}$ coding scheme over a $(19, 17)$ systematic RS code, based on the algorithm presented in this paper, while the choice of actual 5B symbol values was done according to Sect. 4.7.

These exhaustive simulations have run without errors, showing that scheme proposed in this paper is both implementable and working as theory suggests. Complete source code in C of these simulations is available via GitHub [32].

7. Conclusion

We have shown that it is possible to design and implement low-complexity, backward compatible FEC using the novel combination of an RS FEC scheme, a linked-list-based technique to skip forbidden symbols in the MII data part of the codeword, and a lightweight linear coding technique that guarantees the same for the signaling and parity symbols.

8. Future Work

A promising research direction is to better understand how to utilize the fact that 10BASE-T1S is DME-based, including extended detection of erasures and the end of frame. The latter would allow further shortening of the last codeword and subsequent superblocks right after the signaling implemented by the FECESD, reducing the bandwidth utilization when the channel is dominated by short frames.

Acknowledgments

The authors wish to acknowledge the contribution of Piergiorgio Beruto to the writing of this paper, as well as the anonymous reviewers' insightful comments and suggestions along the way.

References

[1] IEEE P802.3cg, "10 Mb/s Single Pair Ethernet Task Force (formerly the 10 Mb/s Single Twisted Pair Ethernet Task Force)"
[2] G.A. Zimmerman, P. Jones, J. Lewis, P. Beruto, S. Graber, and H. Stewart, "IEEE P802.3cg 10 Mb/s Single Pair Ethernet: A guide," Jan. 2019.
[3] IEEE Standard for Ethernet, Amendment 5: Physical Layers Specifications and Management Parameters for 10 Mb/s Operation and Associated Power Delivery over a Single Balanced Pair of Conductors, IEEE Std 802.3cg-2019.
[4] IEEE Standard for Ethernet, IEEE Std 802.3-2018.
[5] S. Pandey, P. Axer, and D. Pannell, "PLCA data-rate fairness," March 2018.
[6] P. Beruto and A. Orzelli, "Proposal for short-reach multi-drop 10M SPE (formerly PLCA)," Sept. 2017.
[7] P. Beruto and A. Orzelli, "802.3cg draft 2.0 PLCA (clause 148) overview," July 2018.
[8] IEEE P802.3da, "10 Mb/s Single Pair Multidrop Segments Enhancement Task Force".
[9] IEEE P802.da, "IEEE P802.3da Objectives".
[10] W. Koczwara and G. Zimmerman, "Impulse Immunity and FEC Objective," Jan. 2020.
[11] W. Cary Huffman, Fundamentals of Error-Correcting Codes, Cambridge University Press, Feb. 2010.
[12] IEC 61000–4–4:2012, Electromagnetic compatibility (EMC) – Part 4–4: Testing and measurement techniques – Electrical fast transient/burst immunity test.
[13] C.E. Shannon, "A mathematical theory of communication," Bell Syst. Tech. J., vol.27, no.4, pp.379–423, 623–656, July, Oct. 1948.
[14] C. Jones, "Noise immunity for single pair cabling and applicability to industrial applications," Jan. 2020.
[15] G. Huszak and H. Morita, "On the 10BASE-T1S preamble for multidrop," presented at the GIIS 2019, Paris, France, Dec. 2019.
[16] B.G. Lee and S.C. Kim, Scrambling Techniques for Digital Transmission, Telecommunication Networks and Computer Systems, Springer, May 2000.
[17] P. Beruto and G. Huszak, "PLCA burst mode," Sept. 2018.
[18] P. Beruto and A. Orzelli, "PLCA Burst mode," Nov. 2018.
[19] IEEE Standard for Ethernet, Amendment 8: Physical Layer Specifications and Management Parameters for 2.5 Gb/s, 5 Gb/s, and 10 Gb/s Automotive Electrical Ethernet, IEEE Std 802.3ch-2020.
[20] C. Liu, "100+Gb/s Ethernet forward error correction (FEC) analysis," Signal Integrity Journal, July 2019.
[21] R. Roth, Introduction to Coding Theory, Cambridge University Press, Feb. 2006.
[22] L.R. Vermani, Elements of Algebraic Coding Theory, Chapman Hall/CRC Mathematics Series, Jan. 1996.
[23] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J Soc. Ind. Appl. Math., vol.8, no.2, pp.300–304, 1960.
[24] R.C. Bose and D.K.R. Chaudhuri, "On a class of error correcting binary group codes," Information and Control, vol.3, no.1, pp.68–79, March 1960.
[25] A. Hocquenghem, "Codes correcteurs d'erreurs," Chiffres, vol.2, 1952.
[26] A.J. Han Vinck, Coding Concepts and Reed-Solomon Codes, Inst. For Experimental Mathematics, Essen, Germany.
[27] C.K.P. Clarke, "Reed-Solomon error correction," Research and Development BBC, July 2002.
[28] P. Anslow, "RS(544,514) FEC performance with 4:1 interleaving," Aug. 2018.
[29] CanovaTech, "CT25207: Simplified multidrop 10BASE-T1S Ethernet PHY with MAC controller for intra-system application," At https://www.canovatech.com/ct25207.html
[30] CanovaTech, "CT25206: Multidrop 10BASE-T1S Ethernet PHY with MAC controller," At https://www.canovatech.com/ct25206.html
[31] "Verbatim 10BASE-T1S PHY simulator," At https://github.com/ghuszak/public/tree/master/simu/
[32] "Verification of the proposed backward-compatible forward error correcting scheme for 10BASE-T1S," At https://github.com/ghuszak/public/tree/master/FEC_data

Gergely Huszak received BSc. in Engineering from the University of Pannonia, Veszprem, Hungary in 2009, followed by MSc. in Engineering from the University of Electro-Communications (UEC), Tokyo, Japan in 2012, currently working towards earning his PhD. in Engineering at his Alma Mater in Tokyo, Japan. He has also been working for the industry for over 20 years, designing and implementing wired and wireless M2M communication systems in Europe. Mr. Huszak was as an editor for IEEE Std 802.3cg-2019 and received Best Paper Award for his previous article [15] written with Hiroyoshi Morita.

Hiroyoshi Morita received the BE, ME, and DE degrees from Osaka University, Osaka, Japan, in 1978, 1980 and 1983, respectively. In 1983, he joined Toyohashi University of Technology, Aichi, Japan as a research associate in the School of Production System Engineering. In 1990, he joined the University of Electro-Communications (UEC), Tokyo, Japan, first as an assistant professor in the Department of Computer Science and Information Mathematics, where from 1992, he was an associate professor. He was then with the Graduate School of Information Systems, UEC since 1995, where from 2005, he is a professor. Since 2016, he has been with Graduate School of Informatics and Engineering, UEC. He was a visiting fellow in the Institute of Experimental Mathematics, University of Essen, Essen, Germany during 1993–1994. Prof. Morita's research interests include the combinatorial theory, information theory, and coding theory, with applications to the digital communication systems.

George Zimmerman was born in New York, NY USA received the B.S.E.E degree from Stanford University in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology in 1988 and 1990, respectively. Since 1985 he was with the Communications Systems Research Section of the Jet Propulsion Laboratory in Pasadena, CA USA. From 1995 through 2000 he was the Chief Scientist of PairGain Technologies working on standards and early technology for DSL wireline communications including ADSL, HDSL, and HDSL2. From 2000 to 2011 he was the Chief Technology Officer of Solarflare Communications, and from 2002 through 2006 participated in the IEEE 802.3 standardization of 10Gb/s Ethernet on twisted pair copper (10GBASE-T). Since 2011 he has been President of CME Consulting, Inc, an independent consultant for physical layer communications and standards. His current research interests include high performance, energy-efficient wireline transmission systems, powered wireline communications, and reuse of installed infrastructure. Dr. Zimmerman has been an active participant since 2002 in the IEEE 802.3 Ethernet Working Group, including contributions to wireline Ethernet transceiver standards for 1000BASE-T1, 2.5GBASE-T, 5GBASE-T, 25GBASE-T, 40GBASE-T, 10BASE-T1S, and 10BASE-T1L, as well as Energy Efficient Ethernet and Power over Ethernet (PoE). He most recently chaired the IEEE 802.3cg 10 Mb/s single pair Ethernet (10SPE) Task force, standardizing single-pair Ethernet transmission and powering for industrial and building automation, as well as intrasystem and automotive applications, and is a current member of the executive committee for the IEEE 802 Standards Committee. Dr. Zimmerman received the IEEE Standards Medallion in 2017. In addition to IEEE, he is also an active member of TIA TR42 and a Codemaking Panel member for the (US) National Electrical Code in the National Fire Protection Association (NFPA).