Wojciech Kaczmarski SP5WWP et al.

M17 Protocol Specification
Part I - Air Interface

August 13, 2025

Version 2.0

LaTeX version compiled by Steve Miller KC1AWV

Contents

List of Tables

List of Figures

Revision History

Rev	Date	Author(s)	Description
1.2	09 Sep 2024	N7TAE, SP5WWP	Removed Definitions and Control Packets sections, rewrote Callsign Encoding appendix using examples in C.
1.3	17 Oct 2024	N7TAE, SP5WWP	Introduced new LSD data type to clarify and correct discussion around LICH, and LSF.
1.4	01 Jan 2025	SP5WWP	Removed the KISS appendix and created a separate KISS specification document.
1.5	11 Feb 2025	N7TAE, VK7XT	Rearranged the Data Link and Application Layer chapters for better flow, removed IP Network chapter and File Type appendix, added more details to Packet Mode, 3 new IP packets defined, and added new clarifying bit tables.
1.6	08 Aug 2025	SP5WWP	Section 3.4 was moved to Part II.
2.0	12 Aug 2025	N7TAE, N7ADJ, SP5WWP	GNSS Meta data changed extensively. Values are now metric, and a new param related to HDOP was added.

Licenses

Introduction

M17 is an RF protocol that is:

- Completely open: open specification, open source code, open source hardware, open algorithms. Anyone must be able to build an M17 radio and interoperate with other M17 radios without having to pay anyone else for the right to do so.
- Optimized for amateur radio use.
- Simple to understand and implement.
- Capable of doing the things hams expect their digital protocols to do:
  - Voice (eg: DMR, D-Star, etc)
  - Point to point data (eg: Packet, D-Star, etc)
  - Broadcast telemetry (eg: APRS, etc)
  - Extensible, so more capabilities can be added over time.

To do this, the M17 protocol is broken down into three protocol layers, like a network:

1. Physical Layer: How to encode 1s and 0s into RF. Specifies RF modulation, symbol rates, bits per symbol, etc.
2. Data Link Layer: How to packetize those 1s and 0s into usable data. Packet vs Stream modes, headers, addressing, etc.
3. Application Layer: Accomplishing activities. Voice and data streams, control packets, beacons, etc.

This document will introduce, define and discuss these layers in detail.

Glossary

Common terms used in M17

BER Bit Error Rate

ECC Error Correcting Code

FEC Forward Error Correction

Frame The individual components of a stream, each of which contains payload data interleaved with frame signalling.

Link Setup Data (LSD) The SRC and DST callsign address fields, TYPE field and the META data.

Link Setup Frame (LSF) The first data frame of any transmission. It contains an LSD and a CRC.

LICH Link Information Channel. The LICH carries all information of an M17 link. The first frame of a transmission contains full link setup data, and subsequent frames each contain one sixth of this data, so that late-joiners can obtain the full link setup data information.

Packet A single burst of data transmitted in Packet Mode.

Superframe A set of six consecutive frames in the stream mode which collectively contain full LSD are grouped into a superframe.

Chapter 1

Physical Layer

This section describes the M17 standard radio physical layer suitable for use where a transmission bandwidth of 9 kHz is permitted.

1.1 4-level Frequency-shift Keying Modulation (4FSK)

The M17 standard uses 4FSK at 4800 symbols/s (9600 bits/s) with a deviation index h=1/3 for transmission in a 9 kHz channel bandwidth. Minimum channel spacing is 12.5 kHz.

1.2 Dibit, Symbol, and Frequency-shift

Each of the 4-level frequency-shifts can be represented by dibits (2-bit values) or symbols, as shown in Table 1 below.

In the case of dibits, the most significant bit is sent first. When four dibits are grouped into a byte, the most significant dibit of the byte is sent first. For example, the four dibits contained in the byte `0xB4` (0b 10 11 01 00) would be sent as the symbols (-1, -3, +3, +1).

Dibit		Symbol	Deviation
MSB	LSB		
0	1	+3	+2.4 kHz
0	0	+1	+0.8 kHz
1	0	-1	-0.8 kHz
1	1	-3	-2.4 kHz

Table 1.1: Dibit symbol mapping to 4FSK deviation

1.3 4FSK Generation

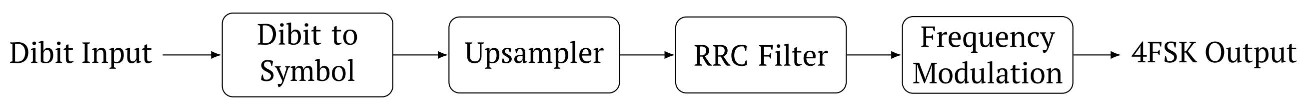


Figure 1.1: 4FSK Generation

Dibits are converted to symbols. The symbol stream is upsampled to a series of impulses which pass through a root-raised-cosine (alpha=0.5) shaping filter before frequency modulation at the transmitter and again after frequency demodulation at the receiver.

Upsampling by a factor of 10 is recommended (48000 samples/s).

The root-raised-cosine filter should span at least 8 symbols (81 taps at the recommended upsample rate).

1.4 Transmission

A complete transmission shall consist of a Preamble, a Synchronization Burst, Payload, and an End of Transmission marker.

PREAMBLE	SYNC BURST	PAYLOAD	EoT
40ms	16 bits	Multiples of 2 bits	40ms
(192 symbols)	(8 symbols)	(multiples of 1 symbol)	(192 symbols)

Table 1.2: Physical Layer Transmission

Transmissions may include more than one synchronization burst followed by a payload.

PREAMBLE	SYNC BURST	PAYLOAD	•••	SYNC BURST	PAYLOAD	EoT

Table 1.3: Physical Layer Transmission with Multiple Synchronization Bursts

1.4.1 Preamble

Every transmission shall start with a preamble, which shall consist of 40 ms (192 symbols) of alternating outer symbols (+3, -3) or (-3, +3), see section 2.4 for details. To ensure a zero crossing prior to a synchronization burst, the last symbol transmitted within the preamble shall be opposite the first symbol transmitted in the synchronization burst.

1.4.2 Synchronization Burst (Sync Burst)

A sync burst of 16 bits (8 symbols) shall be sent immediately after the preamble. The sync burst is constructed using only outer symbols, with codings based on Barker codes. Properly chosen sync burst coding assists in symbol clocking and alignment. Different sync burst codes may also be used by the Data Link Layer to identify the type of payload to follow.

1.4.3 Payload

Payload shall be transmitted in multiples of 2 bits (1 symbol).

1.4.4 Randomizer

To avoid transmitting long sequences of constant symbols (e.g. +3, +3, +3, . . .), a simple randomizing algorithm is used. At the transmitter, all payload bits shall be XORed with a pseudorandom predefined sequence before being converted to symbols. At the receiver, the randomized payload symbols are converted to bits and are again passed through the same XOR algorithm to obtain the original payload bits.

The pseudorandom sequence is composed of the 46 bytes (368 bits) found in the Randomizer appendix table B.

Before each bit of payload is converted to symbols for transmission, it is XORed with a bit from the pseudorandom sequence. The first payload bit is XORed with most significant bit (bit 7) of sequence byte 0 (0×D6), second payload bit with bit 6 of sequence byte 0, continuing to the eighth payload bit and bit 0 of sequence byte 0. The ninth payload bit is XORed with bit 7 of sequence byte 1 (0×B5), tenth payload bit with bit 6 of sequence byte 1, etc.

When payload bits have XORed through sequence byte 45 (0×C3), the pseudorandom sequence is restarted at sequence byte 0 (0×D6).

On the receive side, symbols are converted to randomized payload bits. Each randomized payload bit is converted back to a payload bit by once again XORing each randomized bit with the corresponding pseudorandom sequence bit.

1.4.5 End of Transmission marker (EoT)

Every transmission ends with a distinct symbol stream, which shall consist of 40 ms (192 symbols) of a repeating (0×55) (0×5D) (+3, +3, +3, +3, +3, +3, -3, +3) pattern.

1.4.6 Carrier-sense Multiple Access (CSMA)

CSMA may be used to minimize collisions on a shared radio frequency by having the sender ensure the frequency is clear before transmitting. Higher layers (Data Link and Application) may require the use of CSMA, and may specify parameters other than the defaults.

P-persistent access is used with a default probability of p = 0.25 and default slot time of 40 ms.

1.5 Physical Layer Flow Summary

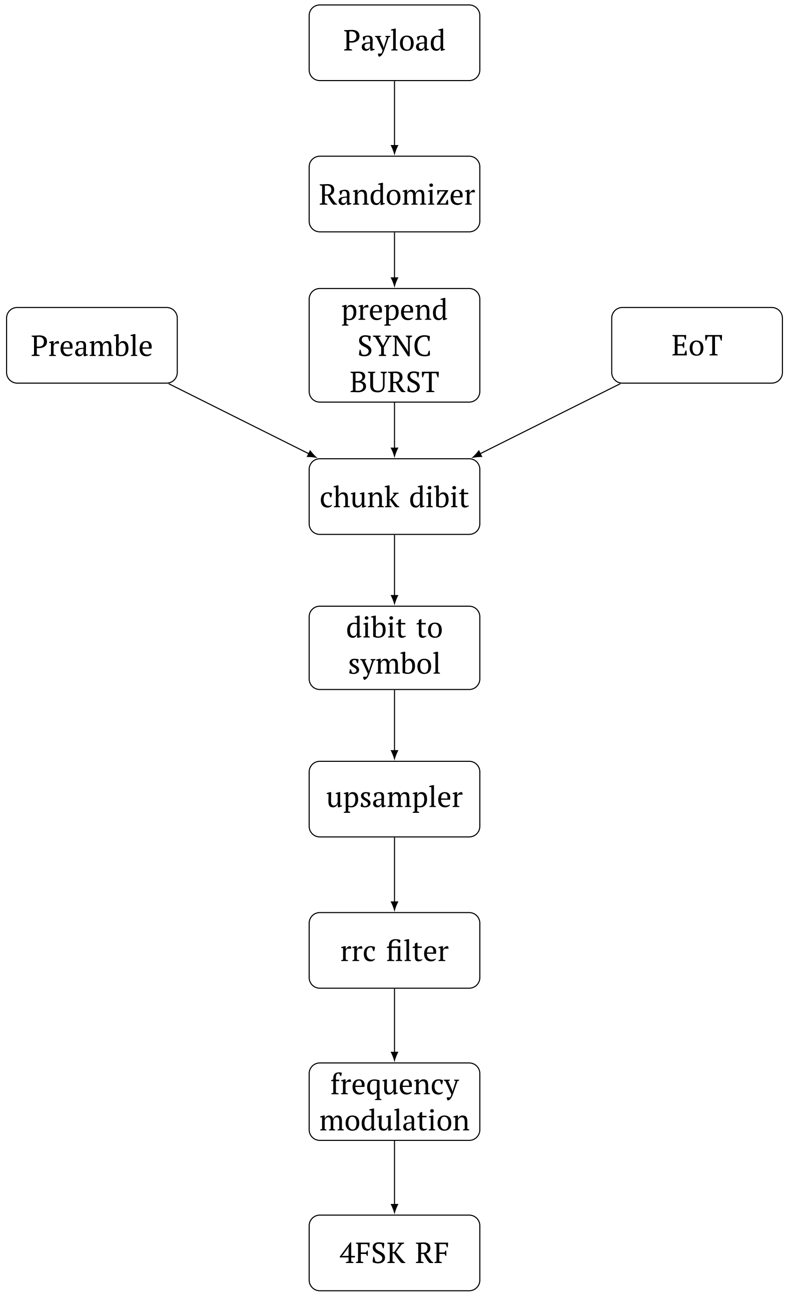


Figure 1.2: Physical Layer Flow

Chapter 2

Data Link Layer

2.1 Frame

A Frame shall be composed of a frame type specific Synchronization Burst (Sync Burst) followed by 368 bits (184 symbols) of Payload. The combination of Sync Burst plus Payload results in a constant 384 bit (192 symbol) Frame. At the M17 data rate of 4800 symbols/s (9600 bits/s), each Frame is exactly 40ms in duration.

There are four frame types each with their own specific Sync Burst: Link Setup Frames (LSF), Bit Error Rate Test (BERT) Frames, Stream Frames, and Packet Frames.

SYNC BURST	PAYLOAD
16 bits	368 bits
(8 symbols)	(184 symbols)

Table 2.1: Frame

2.2 Forward Error Correction (FEC)

The Data Link Layer Contents of a specific frame are modified using various Error Correction Code (ECC) methods. Applying these codes at the transmitter allows the receiver to correct some amount of induced errors in a Forward Error Correction (FEC) process. It is this ECC/FEC data that is inserted into the Payload portion of the Frame. The exact ECC/FEC techniques used vary by frame type.

Applying ECC/FEC may be a multi-step process. To distinguish data bits at the various stages of the process, Bit Types are defined as shown in the following table. It is important to note that not all ECC/FEC processes utilize both Type 2 and Type 3 bits. Prior to decoding Data Link Layer contents, a receiver would need to convert incoming bits from Type 4 back to Type 1 bits, which may also include conversion through Type 3 and/or Type 2 bits. The exact ECC/FEC methods and Bit Types utilized will be indicated for each frame type.

Type	Description
Type 1	Data link layer content bits
Type 2	Bits after appropriate encoding
Type 3	Bits after puncturing. described in Appendix E
Type 4	Interleaved (re-ordered) bits

Table 2.2: Bit Types

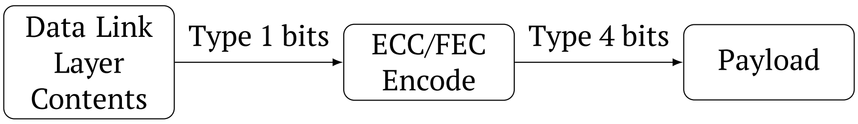


Figure 2.1: Transmit Contents to Payload

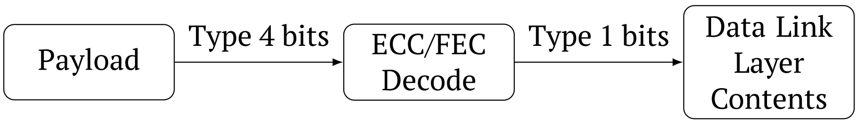


Figure 2.2: Receive Payload to Contents

2.3 Modes

The Data Link layer shall operate in one of three modes during a Transmission.

- Stream Mode Data are sent in a continuous stream for an indefinite amount of time, with no break in physical layer output, until the stream ends. e.g. voice data, bulk data transfers, etc. Stream Mode shall start with an LSF and is followed by one or more Stream Frames.
- Packet Mode Data are sent in small bursts, up to 823 bytes at a time, after which the physical layer stops sending data. e.g. messages, beacons, etc. Packet Mode shall start with an LSF and is followed by one to 33 Packet Frames.
- BERT Mode PRBS9 is used to fill frames with a deterministic bit sequence. Frames are sent in a continuous sequence. Bert Mode shall start with a BERT frame, and is followed by one or more BERT Frames.

  NOTE As is the convention with other networking protocols, all values and data structures are encoded in big endian byte order.

2.4 Synchronization Burst (Sync Burst)

All frames shall be preceded by 16 bits (8 symbols) of Sync Burst. The Sync Burst definition straddles both the Physical Layer and the Data Link Layer.

Only LSF and BERT Sync Bursts may immediately follow the Preamble, and each requires a different Preamble symbol pattern as shown in the table below.

During a Transmission, only one LSF Sync Burst may be present, and if present, it shall immediately follow the Preamble.

BERT Sync Bursts, if present, may only follow the Preamble or other BERT frames.

Multiple Stream or Packet Sync Bursts may be present during a Transmission, depending on the mode.

Frame Type	Preamble	Sync Burst Bytes	Sync Burst Symbols
LSF	+3, -3	0x55 0xF7	+3, +3, +3, +3, -3, -3, +3, -3
BERT	-3, +3	0xDF 0x55	-3, +3, -3, -3, +3, +3, +3, +3
Stream	None	0xFF 0x5D	-3, -3, -3, -3, +3, +3, -3, +3
Packet	None	0x75 0xFF	+3, -3, +3, +3, -3, -3, -3, -3

Table 2.3: Frame Specific Sync Bursts

2.5 Link Setup Data and Frame (LSD and LSF)

2.5.1 Link Setup Data

The Link Setup Data, LSD, is a data structure that is common to both Stream and Packet Mode and contains information needed to establish a link. This data is a fundamental part of both RF data streams and and any kind of internet protocol packets.

Field	Length	Description
DST	48 bits	An encoded destination
SRC	48 bits	An encoded source
TYPE	16 bits	Specifies all characteristic of the payload
META	112 bits	Meta data

Table 2.4: Link Setup Data Contents

Total: 28 bytes, 224 Type 1 bits

- DST and SRC and typically encoded radio amateur callsigns or special identifiers.
- TYPE specifies all the characteristics of the payload.
- META data is suitable for cryptographic metadata like IVs or single-use numbers, or non-crypto metadata like the sender's GNSS position.

2.5.2 Link Setup Frame

The link Setup Frame, LSF, is an LSD followed immediately by a 16 bit cyclic redundancy check, CRC. The LSF is the first data frame in a Stream Mode or Packet Mode transmission.

Field	Length	Description
LSD	224 bits	Shown in the previous table
CRC	16 bits	CRC for the link setup data

Table 2.5: Link Setup Frame contents

Total: 30 bytes, 240 Type 1 bits

The CRC can be used to validate the integrity of the contained LSD and is described below.

2.6 CRC

M17 uses a non-standard version of 16-bit CRC with polynomial $x^{16} + x^{14} + x^{12} + x^{11} + x^8 + x^5 + x^4 + x^2 + 1$ or `0×5935` and initial value of `0×FFFF`. This polynomial allows for detecting all errors up to hamming distance of 5 with payloads up to 241 bits, which is less than the amount of data in each frame.

As M17's native bit order is most significant bit first, neither the input nor the output of the CRC algorithm gets reflected.

The CRC field enables verification of the other 28 bytes forming the LSF: 6-byte DST, 6-byte SRC, 2-byte TYPE, and 14-byte META fields. Data integrity of an LSF frame is established by computing the CRC of the first 28 bytes and storing the resulting checksum in the trailing 2-byte CRC field, which can be compared by a recipient after repeating the same checksum process. Alternatively, a CRC computed over the entire 30-byte LSF frame, including a valid CRC field, will always equal zero.

The test vectors in the following table are calculated by feeding the given message to the CRC algorithm.

Message	CRC Output
(empty string)	0xFFFF
ASCII string “A”	0x206E
ASCII string “123456789”	0x772B
Bytes 0x00 to 0xFF	0x1C31

Table 2.6: CRC Test Vectors

2.7 LSF Contents ECC/FEC

The 240 Type 1 bits of the Link Setup Frame Contents along with 4 flush bits are convolutionally coded using a rate 1/2 coder with constraint K=5. 244 bits total are encoded resulting in 488 Type 2 bits.

Type 3 bits are computed by P_1 puncturing the Type 2 bits, resulting in 368 Type 3 bits.

Interleaving the Type 3 bits produces 368 Type 4 bits that are ready to be passed to the Physical Layer. Interleaving is described in Appendix F.

Within the Physical Layer, the 368 Type 4 bits are randomized and combined with the 16-bit LSF Sync Burst, which results in a complete frame of 384 bits (384 bits / 9600bps = 40 ms).

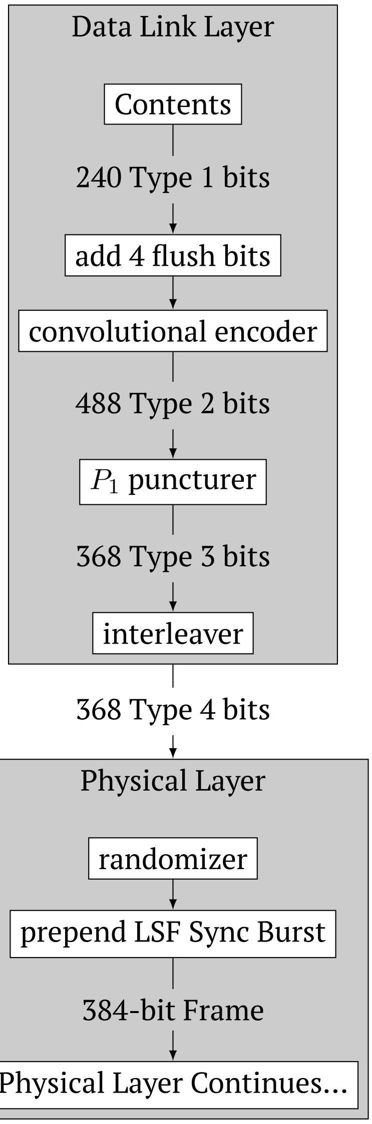


Figure 2.3: LSF Construction

Details of the convolutional encoder are in Appendix C

2.8 Stream Mode

In Stream Mode, an *indefinite* amount of data is sent continuously without breaks in the physical layer. Stream Mode shall always start with an LSF that has the LSF TYPE Packet/Stream indicator bit set to 1 (Stream Mode). Other valid LSF TYPE parameters are selected per application.

Following the LSF, one or more Stream Frames may be sent.

PREAMBLE	LSF SYNC BURST	LSF FRAME	STREAM SYNC BURST	STREAM FRAME	•••	STREAM SYNC BURST	STREAM FRAME	EoT

Table 2.7: Stream Mode

2.8.1 Stream Frames

Stream Frames are composed of frame signalling information contained within the Link Information Channel (LICH) combined with Stream Contents. Both the LICH and Stream Contents utilize different ECC/FEC mechanisms, and are combined at the bit level in a Frame Combiner.

Link Information Channel (LICH) The LICH allows for late listening and independent decoding to check destination address if the LSF for the current transmission was missed.

Each Stream Frame contains a 48-bit Link Information Channel (LICH). Each LICH within a Stream Frame includes a 40-bit chunk of the 240-bit LSF frame that was used to establish the stream. A 3-bit modulo 6 counter (LICH_CNT) is used to indicate which chunk of the LSF is present in the current Stream Frame. LICH_CNT starts at 0, increments to 5, then wraps back to 0.

<table>
<tr><th>Bit / Byte</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="8" rowspan="3">40-bit chunk of full LSF Contents (Type 1 bits)</td></tr>
<tr><td>...</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td><td colspan="3">LICH_CNT</td><td colspan="5">Reserved</td></tr>
</table>

Table 2.8: Link Information Channel Contents

Total: 48 bits

The 40-bit chunks start from the beginning of the LSF.

LICH_CNT	LSF bits
0	0:39
1	40:79
2	80:119
3	120:159
4	160:199
5	200:239

Table 2.9: LICH_CNT and LSF bits

LICH Contents ECC/FEC The 48-bit LICH Contents is partitioned into 4 12-bit parts and encoded using Golay (24, 12) code. This produces 96 encoded Type 2 bits that are fed into the Frame Combiner.

Stream Contents The LSD META field can change during a transmission and this will affect bits 112:239 of the LSF, resulting in changes in the LICH channel with LICH_CNT 2 through 5. In addition, the stream frames also contain the stream contents that has two fields that will change with every frame.

Field	Length	Description
FN	16 bits	Frame Number
STREAM	128 bits	Stream data, can contain arbitrary data

Table 2.10: Stream Contents

Total: 144 Type 1 bits

The Frame Number (FN) starts from 0 and increments every frame to a maximum of `0×7fff` where it will then wrap back to 0. The most significant bit in the FN is used for transmission end signaling. When transmitting the last frame, it shall be set to 1 (one), and 0 (zero) in all other frames.

Stream data (STREAM) is obtained by extracting 128 bits at a time from the continuous stream of application layer data. If the last frame will contain less than 128 bits of valid data, the remaining bits should be set to zero. The stream may end at the frame boundary.

Mode	Codec 2 rate	Frame t + 0	Frame t + 1...
Voice	3200	128 bits encoded speech	128 bits encoded speech
Voice + Data	1600	64 bits encoded speech + 64 bits arbitrary data	64 bits encoded speech + 64 bits arbitrary data

Table 2.11: STREAM Payload Examples

Stream Contents ECC/FEC The 144 Type 1 bits of Stream Contents along with 4 flush bits are convolutionally coded using a rate 1/2 coder with constraint K=5. 148 bits total are encoded resulting in 296 Type 2 bits.

These bits are P_2 punctured to generate 272 Type 3 bits that are fed into the Frame Combiner.

Frame Combiner The 96 Type 2 bits of the ECC/FEC LICH Contents are concatenated with 272 Type 3 bits of the ECC/FEC Stream Contents resulting in 368 of combined Type 2/3 bits.

Field	Length	Description
LICH	96 bits	ECC/FEC LICH Contents Type 2 bits
STREAM	272 bits	ECC/FEC STREAM Contents Type 3 bits

Table 2.12: LICH and Stream Combined

Total: 368 Type 2/3 bits

Interleaving the Combined Type 2/3 bits produces 368 Type 4 bits that are ready to be passed to the Physical Layer.

Within the Physical Layer, the 368 Type 4 bits are randomized and combined with the 16-bit Stream Sync Burst, which results in a complete frame of 384 bits (384 bits / 9600bps = 40 ms).

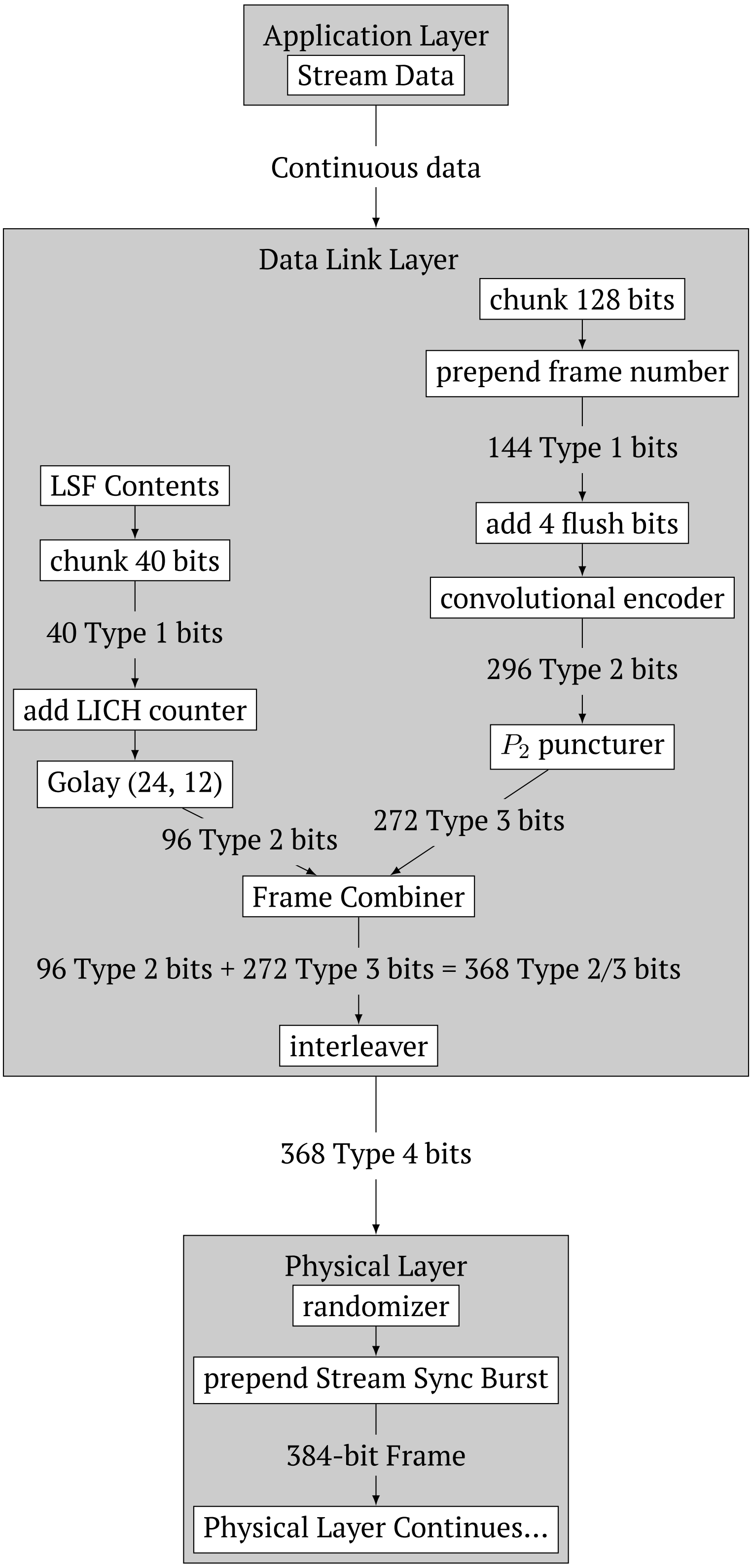


Figure 2.4: Stream Frame Construction

2.8.2 Stream Superframes

Stream Frames are grouped into Stream Superframes, which is the group of 6 frames that contain everything needed to rebuild the original LSF packet, so that the user who starts listening in the middle of a stream (late-joiner) is eventually able to reconstruct the LSF message and understand how to receive the in-progress stream.

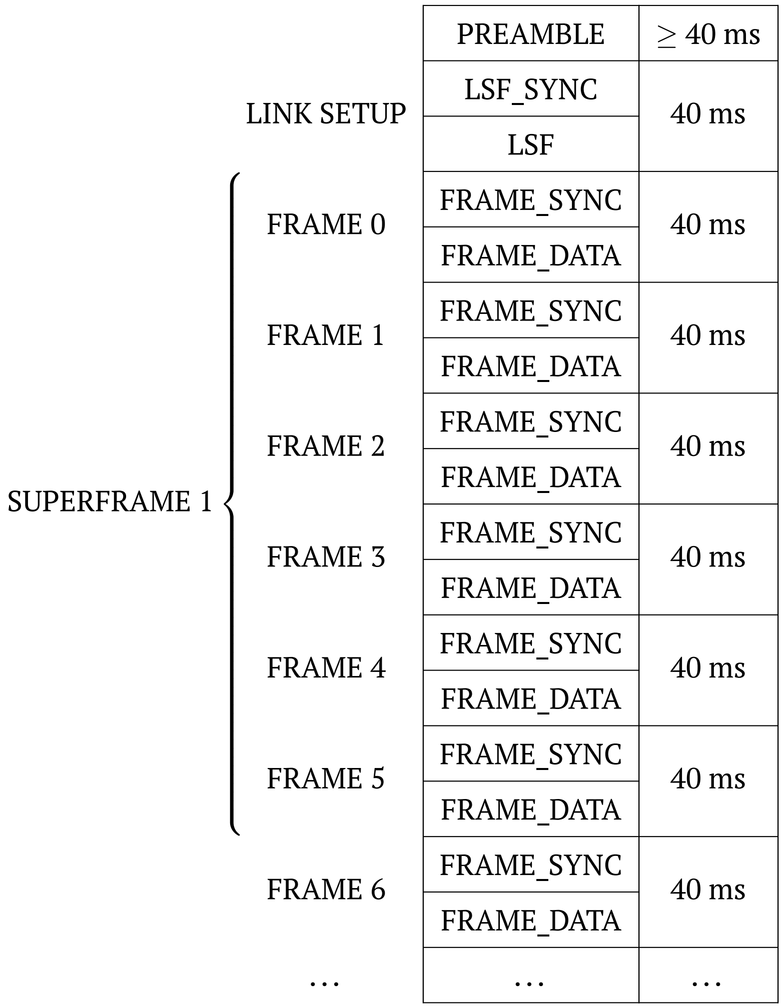


Figure 2.5: Stream Superframes

2.9 Packet Mode

In Packet Mode, a Single Packet with up to 823 bytes of Application Packet Data along with an appended two byte CRC may be sent over the physical layer during one Transmission. The total number of bytes ranges from 25 to 825 (33*25) bytes in 25 byte increments.

Bytes	Meaning
0..n-1	Application Packet Data
n..n+1	CRC

Table 2.13: Single Packet

n is the number of bytes of the Application Packet Data. The CRC calculation used here is described in Section 2.6.

Packet Mode shall always start with an LSF that has the LSF TYPE Packet/Stream indicator bit set to 0 (Packet Mode). Following the LSF, 1 to 33 Packet Frames may be sent.

PREAMBLE	LSF Sync Burst	LSF Frame	Packet Sync Burst	Packet Frame	•••	Packet Sync Burst	Packet Frame	EoT

Table 2.14: Packet Mode

2.9.1 Packet Frames

Packet Frames contain Packet Contents after ECC/FEC is applied.

Byte \ Bit	7	6	5	4	3	2	1	0
0	200-bit chunk of Single Packet							
...								
24								
25	End of Frame	Packet Frame/Byte Counter						

Table 2.15: Packet Contents

Packet Contents Total: 206 Type 1 bits

The packet metadata field contains the 1-bit End of Frame (EOF) indicator, and the 5-bit Packet Frame/Byte Counter. This is *NOT* to be confused with the LSF's 112-bit metadata field.

Data starting with the first byte of the Packet Data, and ending with 2 computed and appended CRC bytes (big-endian) is split in groups of 25 bytes (chunks). The CRC value is calculated over the whole Packet Data, including the terminating null-byte in the case of text. Each Packet Frame payload contains up to a 25-byte chunk of the Data. If fewer than 25 bytes can be extracted from the Data (i.e. for the last Packet Frame), the Data chunk is padded with null bytes (after the terminating CRC) to reach 25 bytes total.

The Packet Frame Counter is reset to zero at the start of Packet Mode. For each Packet Frame where there is at least 1 byte remaining in the Packet Data after removing a 25-byte chunk, the EOF metadata bit is set to zero, the Packet Frame Counter value is inserted into the Packet Frame/Byte Counter metadata field, and the Packet Frame Counter is incremented afterwards.

When there are no bytes remaining in the Packet Data after removing a 25-byte (or less) chunk, the EOF bit is set to one, the Packet Byte Counter is set to the number of valid bytes present in the current frame (1 to 25) and both fields are concatenated into the Packet Frame/Byte Counter metadata field. This results in a minimum of 1 to a maximum of 33 Packet Frames per transmission. Packet Mode is ended with an End of Transmission frame.

Byte \ Bit	7	6	5	4	3	2	1	0
0	0	Frame number, 0..31						

Table 2.16: Packet Metadata Field with EOF = 0

Byte \ Bit	7	6	5	4	3	2	1	0
0	1	Number of bytes in frame, 1..25						

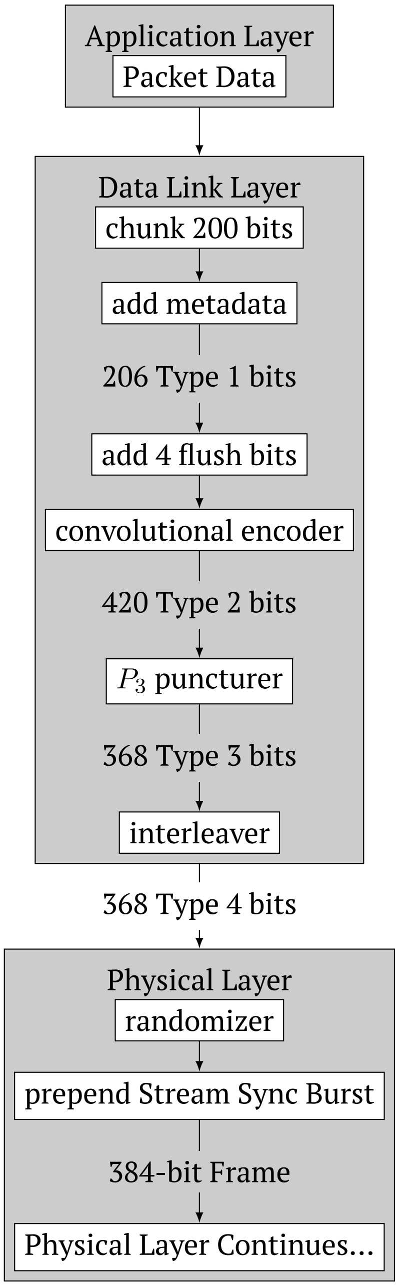
Table 2.17: Packet Metadata Field with EOF = 1

Packet Contents ECC/FEC The 206 Type 1 bits of the Packet Contents along with 4 flush bits are convolutionally coded using a rate 1/2 coder with constraint K=5. 210 bits total are encoded resulting in 420 Type 2 bits.

These bits are P_3 punctured to generate 368 Type 3 bits.

Interleaving the Type 3 bits produces 368 Type 4 bits that are ready to be passed to the Physical Layer.

Within the Physical Layer, the 368 Type 4 bits are randomized and combined with the 16-bit Packet Sync Burst, which results in a complete frame of 384 bits (384 bits / 9600bps = 40 ms).



;

Figure 2.6: Packet Frame Construction

2.9.2 Packet Superframes

A Packet Superframe consists of at least 1 and up to the 33 Packet Frames to reconstruct the original Single Packet.

2.9.3 Net Throughput

Packet Mode achieves a base throughput of 5 kbps, and a net throughput of over 4.5 kbps can be achieved for large payloads.

Below is a graph of the net throughput in bits/second *vs.* payload size in bytes.

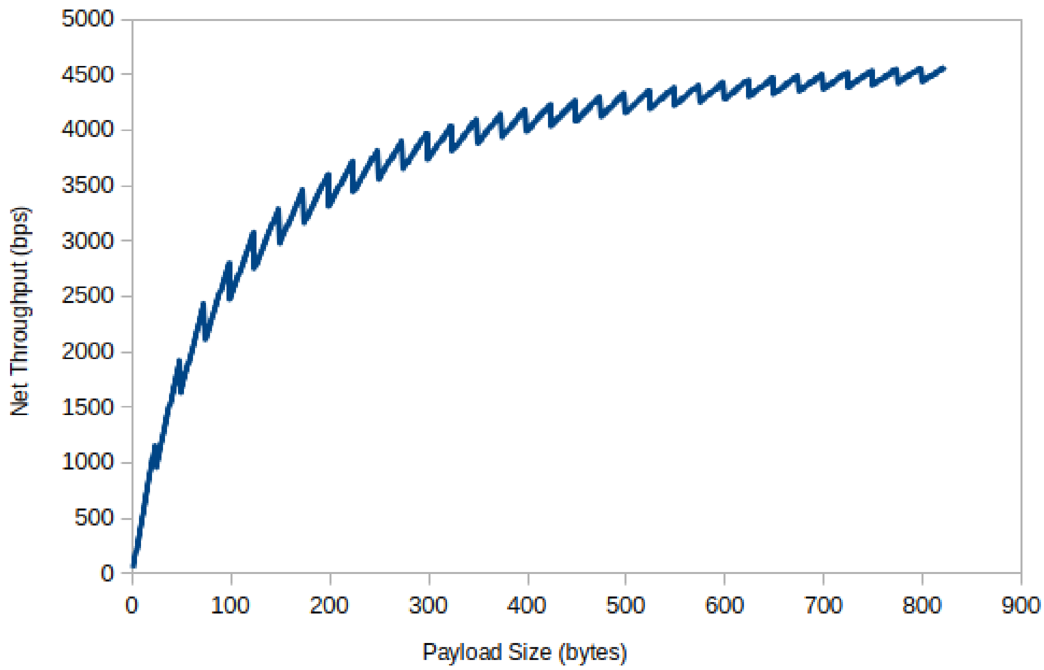


Figure 2.7: Packet Mode Net Throughput

2.10 BERT Mode

BERT mode is a standardized, interoperable mode for bit error rate testing. The preamble is sent, followed by an indefinite sequence of BERT frames. Notably, an LSF is not sent in BERT mode.

The primary purpose of defining a bit error rate testing standard for M17 is to enhance interoperability testing across M17 hardware and software implementations, and to aid in the configuration and tuning of ad hoc communications equipment common in amateur radio.

PREAMBLE	BERT Sync Burst	BERT Frame	•••	BERT Sync Burst	BERT Frame	EoT

Table 2.18: Packet Mode

2.10.1 BERT Frames

BERT Frames contain BERT Contents after ECC/FEC is applied.

BERT Contents The BERT Contents consists of 197 bits from a PRBS9 generator. This is 24 bytes and 5 bits of data. The next BERT Contents starts with the 198th bit from the PRBS9 generator. The same generator is used for each subsequent BERT Contents without being reset. The number of bits pulled from the generator, 197, is a prime number. This will produce a reasonably large number of unique frames even with a PRBS generator with a relatively short period.

See Appendix G for BERT generation and reception details.

Bits	Meaning
0-196	BERT PRBS9 Payload

Table 2.19: BERT Contents

Total: 197 Type 1 bits

BERT Contents ECC/FEC The 197 Type 1 bits of the Packet Contents along with 4 flush bits are convolutionally coded using a rate 1/2 coder with constraint K=5. 201 bits total are encoded resulting in 402 Type 2 bits.

These bits are P_2 punctured to generate 368 Type 3 bits.

Interleaving the Type 3 bits produces 368 Type 4 bits that are ready to be passed to the Physical Layer.

This provides the same error ECC/FEC used for Stream Frames.

Within the Physical Layer, the 368 Type 4 bits are randomized and combined with the 16-bit BERT Sync Burst, which results in a complete frame of 384 bits (384 bits / 9600bps = 40 ms).

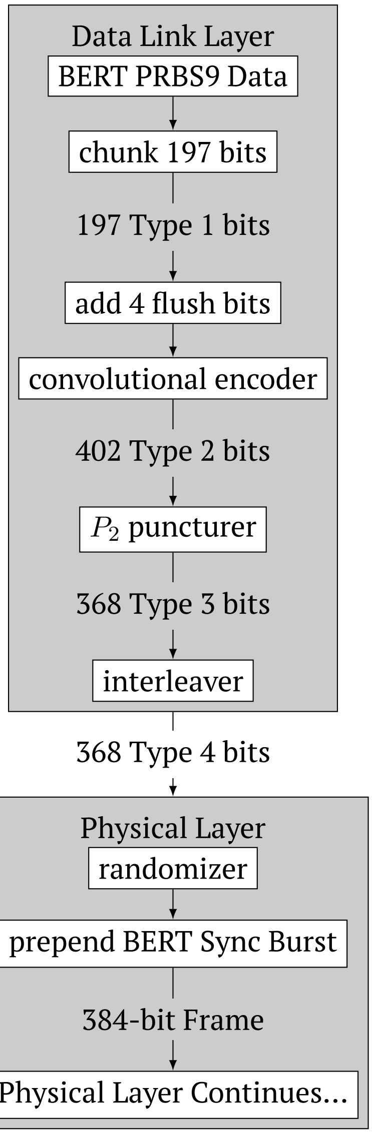


Figure 2.8: BERT Frame Construction

Chapter 3

Application Layer

Stream mode is primarily for an audio stream containing low bit rate speech encoded using the open source Codec 2 codec. One of two different Codec 2 rates can be used in M17, and so Stream mode can be used in a voice-only mode using the higher bit rate codec, and voice+data mode using a lower bit rate codec which leave room for a parallel data stream, and a data only stream mode where arbitrarily large data objects can be streamed. Stream mode is intended to be used over the air by amateur radio operators worldwide. Implementation details for M17 clients, repeaters, and gateways ensure that an M17 Amateur Radio Voice Application is legal under all licensing regimes.

Packet mode is a one-shot method to send a small data packet over the air. It is intended primarily for text messaging, but other small binary objects can also be sent.

Both Stream and Packet mode begins with an Link Setup Frame, LSF.

3.1 LSF

Field	Length	Description
DST	48 bits	Destination address
SRC	48 bits	Source address
TYPE	16 bits	Information about the incoming data stream
META	112 bits	Metadata field
CRC	16 bits	For a description, see Section 2.6

Table 3.1: Link Setup Frame Contents

3.1.1 Address fields

Destination (DST) and source (SRC) addresses may be encoded amateur radio callsigns, or special identifiers. See the Address Encoding Appendix A for details on how up to 9 characters of text can be encoded into the 6-byte address value.

The source address is always the callsign of the station transmitting, be it a client, repeater, or gateway. This is not a problem for a client, but for a repeater/gateway this raises issues about identifying the original source of a transmission. Having a repeater/gateway always use its own callsign for the source field does ensure that there are no issues with licensing authorities. To retain identification of the original source for a voice stream, an extended callsign data field will be encoded in the LSF META field.

The destination address used by a client may simply be a callsign or reflector designation for a point to point contact, or may be a special identifier. Special identifiers are 6-byte addresses than can't be encoded in the standard way. For an explanation, see the Address Encoding Appendix.

3.1.2 TYPE

The 2-byte TYPE field contains information about the frames to follow LSF. The Packet/Stream indicator bit determines which mode (Packet or Stream) will be used during the transmission. The remaining field meanings are defined by the specific mode and application.

3.1.3 META

The 14-byte META field can and will contain a variety of data. In Stream mode, the META data will change as the stream evolves. The first meta field is available in the LSF frame but subsequent data/voice frames will potentially carry different META data in each superframe that follows. In Packet mode, one META field is available in the LSF frame. Different META data is described later in this chapter.

3.1.4 CRC

The last 2 bytes of the 30-byte LSF is a 16-bit CRC as described in Section 2.6.

3.2 Stream Mode

3.2.1 TYPE Field

The TYPE field contains all information need to properly interpret the stream frames.

<table>
<tr><th>Byte \ Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="4">Reserved</td><td>Signed Stream</td><td colspan="3">Channel Access Number…</td></tr>
<tr><td>1</td><td>…</td><td colspan="2">Encryption Subtype</td><td colspan="2">Encryption Type</td><td colspan="2">Data Type</td><td>Packet/ Stream</td></tr>
</table>

Table 3.2: LSF TYPE layout

Value	Mode
0	Packet mode
1	Stream mode

Table 3.3: Packet/Stream indicator

Value	Content
00_2	Reserved
01_2	Data
10_2	Voice
11_2	Voice+Data

Table 3.4: Data type

Value	Encryption
00_2	None
01_2	Scrambler
10_2	AES
11_2	Other/reserved

Table 3.5: Encryption type

For the encryption subtype, meaning of values depends on encryption type.

Value	Scrambler	AES
00_2	8-bit	128-bit
01_2	16-bit	192-bit
10_2	24-bit	256-bit
11_2	reserved	reserved

Table 3.6: Key lengths for encryption subtypes

Packet/Stream	1 = Stream Mode
Data Type	10_2 = Voice only (3200 bps)
Encryption Type	00_2 = None 01_2 = Scrambling 10_2 = AES
Encryption Subtype	Depends on Encryption Type
Channel Access Number (CAN)	0..15

Table 3.7: M17 Voice LSF TYPE definition

This application requires Stream Mode.

The Voice only Data type indicator specifies voice data encoded at 3200 bps using Codec 2.

3.2.2 Encryption Types

Encryption is **optional**. The use of it may be restricted within some radio services and countries, and should only be used if legally permissible.

Null Encryption Encryption type = 00_2

When no encryption is used, the 14-byte (112-bit) META field of the LSF and corresponding LICH of the stream can be used for transmitting relatively small amounts of extended data without affecting the bandwidth available for the audio. The full 14 bytes of META extended data is potentially decodable every six stream frames, at a 240 ms update rate. The extended data is transmitted in a simple round robin manner, with the only exception being GPS data which should be transmitted as soon as possible after the GPS data is received from its source.

The "Encryption subtype" bits in the Stream Type field indicate what extended data is stored in the META field.

Encryption subtype bits	LSF META data contents
00_2	Text Data
01_2	GNSS Position Data
10_2	Extended Callsign Data
11_2	Reserved

Table 3.8: Null encryption subtype bits

Text Data Encryption subtype = 00_2

The first byte of the Text Data is a Control Byte. To maintain backward compatibility, a Control Byte of 0x00 indicates that no Text Data is included.

Up to four Text Data blocks compose a complete message with a maximum length of 52 bytes. Each block may contain up to 13 bytes of UTF-8 encoded text, and is padded with space characters to fill any unused space at the end of the last used Text Data block.

The Control Byte is split into two 4-bit fields. The most significant four bits are a bit map of the message length indicating how many Text Data blocks are required for a complete message. There is one bit per used Text Data block, with 0001_2 used for one block, 0011_2 for the two, 0111_2 for three, and 1111_2 for four.

The least significant four bits indicate which of the Text Data blocks this text corresponds to. It is 0001_2 for the first, 0010_2 for the second, 0100_2 for the third, and 1000_2 for the fourth. Any received Control Byte is OR-ed together by the receiving station, and once the most significant and least significant four bits are the same, a complete message has been received.

It is up to the receiver to decide how to display this message. It may choose to wait for all of the Text Data to be received, or display the parts as they are received. It is not expected that the data in the text field changes during the course of a transmission.

If there is no data to be transferred, the Control Byte should be set to zero.

GNSS Data Encryption subtype = 01_2

Unlike Text and Extended Callsign Data, GNSS data is expected to be dynamic during the course of a transmission and to be transmitted quickly after the GNSS data becomes available. To stop the LSF/LICH data stream from being overrun with GNSS data relative to other data types, a throttle on the amount of GNSS data transmitted is needed. It is recommended that GNSS data be sent at an update rate no faster than once every five seconds.

The GNSS data fits within one 14-byte META field, which equates to six audio frames, and takes 240ms to transmit. This is a simple format of the GNSS data which does not require too much work to convert into, and provides enough flexibility for most cases. This has been tested on-air and successfully gated to APRS-IS, showing a location very close to the position reported by the GPS receiver.

The GNSS data includes eight numeric values using from 3 bits to 23 bits. All numeric fields are in order from most significant to least significant bit. There is also one 4 bit validity field that used to indicate which numeric fields are valid and two sign bits that indicate in which hemisphere a latitude or longitude is. North and East are 0 and South and West are 1.

GNSS Position Data uses the 112 bit (14 byte) META field as follows:

Byte \ Bit	7	6	5	4	3	2	1	0
0	Data Source				Station Type			
1	Validity				Radius			Bearing…
2	…							
3	Sign	Latitude…						
4	…							
5	…							
6	Sign	Longitude…						
7	…							
8	…							
9	Altitude…							
10	…							
11	Speed…							
12	…				*Reserved…*			
13	…							

Table 3.9: GNSS Data encoding

The first byte contains two 4 bit numeric fields. The first is the **Data Source** where: 0 is an M17 client, 1 is OpenRTX and 15 is "other" while values 2..14 are reserved. The second 4 field is **Station Type**, where 0 is a fixed station, 1 is a mobile station, 2 is a handheld and 15 is "other". Values 3..14 are reserved.

The second byte starts with a 4 bit **Validity** field. Bit 1000_2 is set if the latitude/longitude is valid. Bit 0100_2 is set if the **Altitude** data is valid. Bit 0010_2 is set if the velocity data is valid. Velocity data includes both **Bearing** and **Speed** data. Finally, bit 0001_2 is set if the **Radius** data is valid. If any of these validity bits are set to zero, all the corresponding GNSS data fields should be zeroed-out by the transmitter and regarded as invalid and ignored by the receiver.

The next three bits of the second byte is the numeric **Radius** field. This radius is an estimate of lateral position uncertainty and is based on the horizontal dilution of precision (HDOP) value provided by the GNSS module. HDOP measure is based on the number of received satellites and their geometric position relative to the receiver. Note: the HDOP value (and therefore radius) is only a coarse estimate and in some cases might not reflect the actual uncertainty metric.

The last bit of the second byte is the most significant bit of the 9 bit **Bearing** numeric field. This bit is combined with the third byte and contains the bearing value. This is the heading direction for the velocity data in degrees and should never contain a value greater than 359. Zero is due north and 90 is due east, *etc*.

The 24 bit **Latitude** is specifed in the next three bytes (fourth through sixth). After the sign bit, the remaining 23 bit numeric field is the binary fraction of 90 degrees, where the value 0 represents 0 degrees latitude, *i.e.*, the equator, and 8388607, $2^{23} - 1$, represents 90 degrees, *i.e.*, the poles. With 23 bits, this is a resolution of approximately 39 milliseconds of arc (1.2m).

The 24 bit **Longitude** follows in the seventh through ninth byte. After the sign bit, the remaining 23 bits is the binary fraction of 180 degrees, where the value 0 represents 0 degrees longitude, *i.e.*, the prime meridian, and 8388607 represents 180 degrees. With 23 bits, this is a resolution of approximately 77 milliseconds of arc (2.4m at the equator).

A 16 bit numeric **Alitude** field is in the tenth and eleventh bytes and decodes in 0.5 meter steps, offset by 500 meters. A value of 0 is an altitude of -500.0 meters, while the largest value of 65535 is an altitude of 31767.5 meters.

A 12 bit **Speed** numeric field is in the twelfth byte and the 4 most significant bits of the thirteen byte. The decoded values range from 0.0 to 2047.5 km/h in 0.5 km/h steps.

Finally, the remaining 4 bits of the thirteenth byte and the fourteenth byte are reserved and should be set to zero.

Extended Callsign Data Encryption subtype = 10_2

This is only transmitted from repeaters/gateways and not from clients, who only receive and display this data. These fields should not appear over M17 Internet links as they should only be used over the air from a repeater/gateway.

The META field is split into two callsign fields. The first is always present, and the second is optional. The callsign data is encoded using the standard M17 callsign Address Encoding which takes six bytes to encode a nine character callsign. Any unused space in the META field contains 0x00 bytes. The first callsign field starts at offset zero in the META field, and the second callsign if present starts immediately after the first. There are two unused bytes at the end of the META field.

The use of these two callsign fields is as follows:

Source	Callsign Field 1	Callsign Field 2
Locally Repeated RF	Originator	Unused
ECHO Reply	Originator	Unused
Reflector Traffic	Originator	Reflector Name

Table 3.10: Extended Callsign Data encoding

The extended callsign data is not used under any other circumstances than the above currently.

It is not expected that the data in the extra callsign fields change during the course of a transmission.

Scrambling Encryption type = 01_2

Scrambling is an encryption by bit inversion using a bitwise exclusive-or (XOR) operation between the bit sequence of data and a pseudorandom bit sequence.

Pseudorandom bit sequence is generated using a Fibonacci-topology Linear- Feedback Shift Register (LFSR). Three different LFSR sizes are available: 8, 16 and 24-bit. Each shift register has an associated polynomial. The polynomials are listed in Table 7. The LFSR is initialized with a seed value of the same length as the shift register. The seed value acts as an encryption key for the scrambler algorithm. Figures 16 to 18 show block diagrams of the algorithm.

Encryption subtype	LFSR polynomial	Seed length	Sequence period
00_2	$x^8 + x^6 + x^5 + x^4 + 1$	8 bits	255
01_2	$x^{16} + x^{15} + x^{13} + x^4 + 1$	16 bits	65,535
10_2	$x^{24} + x^{23} + x^{22} + x^{17} + 1$	24 bits	16,777,215

Table 3.11: Scrambling

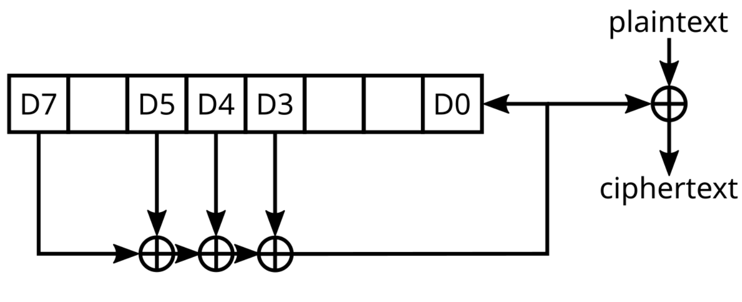


Figure 3.1: 8-bit LFSR taps

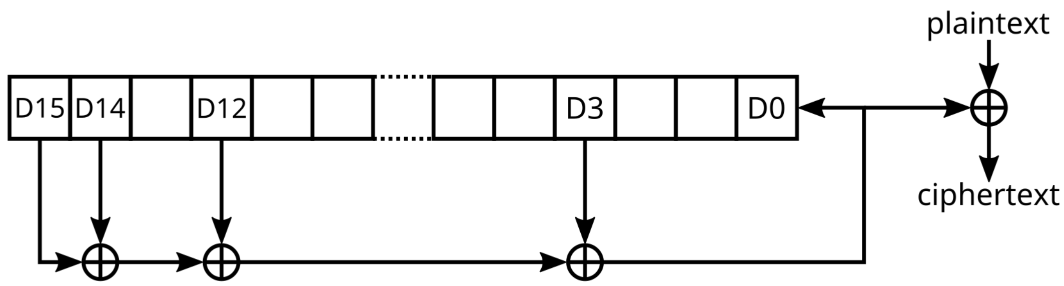


Figure 3.2: 16-bit LFSR taps

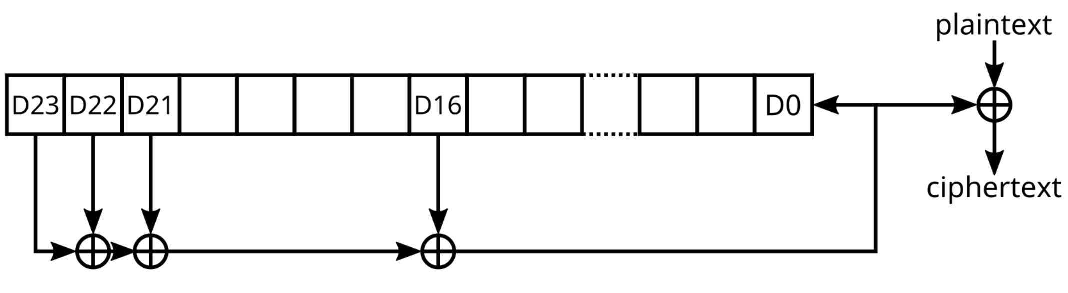


Figure 3.3: 24-bit LFSR taps

Advanced Encryption Standard (AES) Encryption type = 10_2

This method uses AES block cipher in counter mode (AES-CTR), with a 112-bit nonce that should never be used for more than one stream (transmission) and a 16-bit counter.

Key length is defined by the encryption subtype field.

Encryption subtype	Key length
00_2	128 bits
01_2	192 bits
10_2	256 bits
11_2	reserved

Table 3.12: AES key lengths

The 112-bit nonce value is stored in the META field. The FN (Frame Number) value is then used to fill out the remaining 16 bits of the counter, totalling to 128 bits, and always starts from 0 (zero) in a new voice stream.

> **NOTE** The effective capacity of the frame counter is 15 bits, as its most significant bit is used for transmission end signalling. At 25 frames per second and 2^{15} frames, the transmission can last up to 2^{15} frames / 25 frames per second = 1310 seconds, or almost 22 minutes, without rolling over the counter.

The random part of the nonce value should be generated with a hardware random number generator or any other cryptographicaly secure method of generating random values.

To combat replay attacks, a 32-bit timestamp shall be embedded into the cryptographic nonce field. The field structure of the 128 bit counter is shown in Table 9. Timestamp is the number of seconds that elapsed since January 1, 2020, 00:00:00 UTC, minus leap seconds.

128 bit counter structure FN field sets the most significant 16 bits of the counter, with the 32-bit least significant part holding the timestamp. The remaining 80-bit portion is filled with random data, re-generated per transmission.

Timestamp	Random Data	FN
32	80	16

Table 3.13: AES counter

> **WARNING** In CTR mode, AES encryption is malleable. That is, an attacker can change the contents of the encrypted message without decrypting it. This means that recipients of AES-encrypted data must not trust that the data is authentic. Users who require that received messages are proven to be exactly as-sent by the sender should use an appropriate digital signature algorithm, as described below.

3.2.3 Channel Access Number (CAN)

The Channel Access Number (CAN) is a four bit code that may be used to filter received audio, text, and GNSS data. A receiver may optionally allow reception from sources only if their transmitted CAN value matches the receiver's own specified CAN value.

3.2.4 Stream Frames

Stream Frames will contain chunked LSF contents (in the LICH field). The Stream Contents will include the incrementing 16-bit Frame Number, and 128 bits of data (unencrypted or encrypted).

3.2.5 Digital Signatures

M17 protocol provides a stream authentication method through Elliptic Curve Digital Signature Algorithm (ECDSA). The curve used is $secp256r1$. Signature availability is signalled with a specific bit in the TYPE field. Signature use reduces the maximum length of the stream by 4 frames.

Message Digest Algorithm for Voice Streams

At the beginning of the transmission, a $digest$ byte array of size 16 is initialized with zeros. After every stream frame (starting at frame 0) an exclusive or (XOR) operation is performed over the contents of the $digest$ array and the frame's payload. The $digest$ array is then rotated left by 1 byte. The result shall be retained in the array.

$$\begin{aligned} digest &:= digest \oplus payload \\ digest &:= rol(digest, 8) \end{aligned}$$

This process is repeated until there is no more data to transmit. In case there is any encryption enabled, the $payload$ input shall be the encrypted stream. This ensures the possibility of verification, even if the encryption details are not known to the receiving parties. Frame Numbers of the frames carrying the signature should follow a succession of $\{7FFC_{16}, 7FFD_{16}, 7FFE_{16}, FFFF_{16}\}$.

> **NOTE** The Frame Number's most significant bit of the last speech payload stream shall not be set, since it is not the last frame to be transmitted.

Signature Generation and Transmission

At the transmitter-side, the stream digest is signed with a 256-bit private key. The resulting 512-bit signature is split into 4 chunks and sent as additional payload at the end of the transmission. To keep the reassembled LSF data consistent, the LICH counter shall advance normally. The most significant bit of the Frame Number (signalling end of transmission) shall be set only in the last frame carrying the signature.

Signature Verification

At the receiver-side, the 512-bit signature is retrieved from the last 4 frames' contents, if the appropriate TYPE bit is set. The signature is then checked using a 512-bit public key.

> **NOTE** The verification process will work if and only if all the data is received successfully (without transmission errors or dropped frames).

3.3 Packet Mode

3.3.1 Packet Mode LSF TYPE

The TYPE field only defines the stream/packet bit, called "P/S" in the table below and the CAN bits. All other bits are reserved.

Byte \ Bit	7	6	5	4	3	2	1	0
0	*Reserved*					Channel Access Number…		
1	…	*Reserved*						P/S

Table 3.14: LSF TYPE layout

3.3.2 Packet Data

A single packet of up to 823 bytes of data may be sent in one transmission.

Packets are sent using Packet Mode.

Packets are composed of a 1..n byte data type specifier and up to $823 - n$ bytes of payload data. The data type specifier is a variable-length encoding using the same format as UTF-8. The data type specifier must be between 0 and $2^{21} - 1$ which will occupy between 1 and 4 bytes when encoded. Values from 0 to 127 are identical to their encoded form.

The data type specifier can also be used as a protocol specifier. For example, the following protocol identifiers are reserved in the M17 packet spec:

Identifier	Protocol
0x00	RAW
0x01	AX.25
0x02	APRS
0x03	6LoWPAN
0x04	IPv4
0x05	SMS (null-terminated, UTF-8 encoded string)
0x06	Winlink

Table 3.15: Packet protocol identifiers

The data type specifier is used to compute the CRC, along with the payload.

Appendix A

Address Encoding

A.1 The M17 aphabet

M17 uses a 48-bit (6-byte) address to represent the characters that define a source and destination. M17 uses a 40-character alphabet. Encoded, up to nine characters can be used to encode a source or destination address that will still fit in a 48-bit address field. These nine characters will usually, but not necessarily be an amateur radio callsign.

In nearly all circumstances, the source address will decode to an amateur callsign. But frequently, the destination address will not decode to an amateur radio callsign. Typically it will be a unit command, like `ECHO`, or `UNLINK`, or the module of a reflector, like `M17-M17 C`.

In order to define how encoding and decoding are done, here are 40 characters used in M17 ordered by their value:

Value	Character	Name	ASCII	Note
0	`' '`	Space	0x20	Also, any invalid character
1 - 26	`'A'` - `'Z'`	Letter	0x41 - 0x5A	Uppercase
27 - 36	`'0'` - `'9'`	Digit	0x30 - 0x39	Decimal
37	`'-'`	Hyphen	0x2D	Dash
38	`'/'`	Slash	0x3F	Forward slash
39	`'.'`	Dot	0x3E	Period

Table A.1: M17 Callsign Alphabet

A.2 Callsign Encoding

Here are some facts and rules about the encoding an address from a callsign:

- A callsign is encoded backwards, from the last character to the first character. This means that the first character of the callsign is in the least significant bits of the address, while the last character is encode into the most significant bits of the address.
- Since the space character has a value of zero, trailing spaces will not affect the encoded value. For example the calcuated address of `'ABC'` is the same as `'ABC '`, or `'ABC      '`.
- If an uncoded address represents an amateur radio callsign it should be left-justified. That means that the first character will always be a digit or letter.

- Over 262 trillion address can be encoded from `0x1` (`A`) to `0xEE6B27FFFFFF` (`.........`) and only a fraction of these callsign actually look like an amateur radio callsign. Those encodable base-40 text strings that don't look like an amateur radio callsign can be used by applications for triggering events and features that their programs offer.
- A callsign consisting of only spaces is invalid, because it would have a corresponding address of zero. That address is defined to be invalid.
- Using this scheme, there are over 19 trillion 48-bit addresses that can't be encoded by nine characters. Only one of these non-encodable addresses ($2^{48} - 1$) has a specified use.
- After the base-40 value is calculated, the final 6-byte address is the big endian encoded representation of the base-40 value. This is also called network byte order.

As an example, the address of `AB1CD` would be calculated as:

('`A`': 1) + ('`B`': 2×40) + ('`1`': 28×40^2) + ('`C`': 3×40^3) + ('`D`': 4×40^4)

or, after refactoring and reordering:

$((((4) \times 40 + 3) \times 40 + 28) \times 40 + 2) \times 40 + 1$

producing the resulting address:

`0x9fdd51` (base-16), `10476881` (base-10).

A.3 Encoded Addresses

Because 40^9 is less than 2^{48}, there are some 48-bit addresses that can't be accessed. Here is a map of the address space:

Address Range	Category	Number of Addresses	Remarks
`0x000000000000`	INVALID	1	Forbidden
`0x000000000001` `0xEE6B27FFFFFF`	Codable	~262 trillion	"`A`" to "`.........`"
`0xEE6B28000000` `0xFFFFFFFFFFFE`	Uncodable	~19 trillion	for application use
`0xFFFFFFFFFFFF`	BROADCAST	1	valid only for a destination

Table A.2: M17 Addresses

The BROADCAST address should only be used as a destination address. It means that the M17 stream or packet is intended for any capable M17 receivers.

The Uncodable addresses can be used by applications for their own purposes and encoding/decoding algorithms for these addresses are left to the developer.

For Codable addresses, the following encoding and decoding examples written in C will not treat the BROADCAST address. This is an implementation detail left to the developers.

A.4 Encoder Example

```
void Encode(const char *callsign, uint8_t *pUChar)
{
  uint64_t address = 0; // the calculate address in host byte order

  if (pUChar && callsign && *callsign) // make sure we can return a non-zero
    address
  {
    const char *p = callsign;

    // find the last char, but don't select more than 9 characters
    while (*p++ && (p-callsign < 9)) ;

    // process each char from the end to the beginning
    for (p--; p>=callsign; p--)
    {
      unsigned val = 0; // the default value of the character
      if      ('A' <= *p && *p <= 'Z') val = *p - 'A' + 1;
      else if ('0' <= *p && *p <= '9') val = *p - '0' + 27;
      else if ('-' == *p)              val = 37;
      else if ('/' == *p)              val = 38;
      else if ('.' == *p)              val = 39;
      else if ('a' <= *p && *p <= 'z') val = *p - 'a' + 1;

      address = 40u * address + val; // increment and add
    }
  }

  for (int i=5; i>=0; i--) // put it in network byte order
  {
    pUChar[i] = address & 0xffu;
    address /= 0x100u;
  }
}
```

A.5 Decoder Example

```
char *Decode(const uint8_t* pUChar)
{
  static char cs[10];    // this is the return value
  memset(cs, NULL, 10); // initialize it to nothing

  if (NULL == pUChar) // nothing in, nothing out
    return cs;

  // calculate the address in host byte order
  uint64_t address = 0;
  for (int i=0; i<6; i++)
    address = address * 0x100u + pUChar[i];

  if (address >= 0xee6b28000000u) // is it in the undecodable range?
    return cs; // practical applications will do something here

  // the M17 alphabet, ordered by value
  const char *m17chars = " ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-/.";

  unsigned i = 0; // index for the current character

  while (address)
  {
    // the current character is the address modulus 40
    cs[i++] = m17chars[address % 40u];
    address /= 40u; // keep dividing the address until there's nothing left
  }

  return cs;
}
```

For an example of how to encode and decode BROADCAST, or how to use part of the Uncodable address space, see https://github.com/M17-Project/libm17.

Appendix B

Randomizer Sequence

Seq. number	Value	Seq. number	Value
00	0xD6	23	0x6E
01	0xB5	24	0x68
02	0xE2	25	0x2F
03	0x30	26	0x35
04	0x82	27	0xDA
05	0xFF	28	0x14
06	0x84	29	0xEA
07	0x62	30	0xCD
08	0xBA	31	0x76
09	0x4E	32	0x19
10	0x96	33	0x8D
11	0x90	34	0xD5
12	0xD8	35	0x80
13	0x98	36	0xD1
14	0xDD	37	0x33
15	0x5D	38	0x87
16	0x0C	39	0x13
17	0xC8	40	0x57
18	0x52	41	0x18
19	0x43	42	0x2D
20	0x91	43	0x29
21	0x1D	44	0x78
22	0xF8	45	0xC3

Table B.1: Randomizer values

Appendix C

Convolutional Encoder

The convolutional code shall encode the input bit sequence after appending 4 tail bits at the end of the sequence. Rate of the coder is R=½ with constraint length K=5. The encoder diagram and generating polynomials are shown below.

$$
\begin{aligned}
G_1(D) = & 1 + D^3 + D^4 \\
G_2(D) = & 1 + D + D^2 + D^4
\end{aligned}
$$

The output from the encoder must be read alternately.

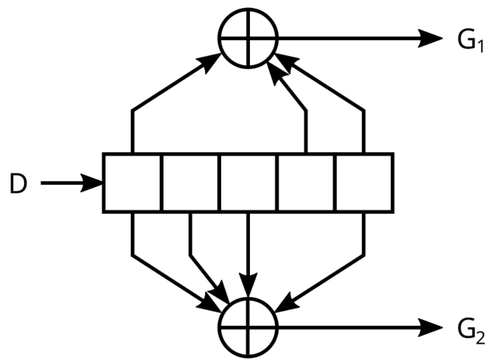


Figure C.1: Convolutional encoder

Appendix D

Golay Encoder

The extended Golay(24,12) encoder uses generating polynomial *g(x)* given below to generate the 11 check bits. The check bits and an additional parity bit are appended to the 12 bit data, resulting in a 24 bit codeword. The resulting code is systematic, meaning that the input data (message) is embedded in the codeword.

$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$

This is equivalent to 0xC75 in hexadecimal notation. Both the generating matrix G and parity check matrix H are shown below.

$$G = [I_{12}|P] = \left[I_{12} \begin{matrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{matrix} \right] \tag{D.1}$$

$$H = [P^T | I_{12}] = \left[\begin{matrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{matrix} I_{12}\right] \tag{D.2}$$

The output of the Golay encoder is shown in the table below.

Field	Data	Check bits	Parity
Position	23..12	11..1	0 (LSB)
Length	12	11	1

Table D.1: Golay encoder details

Four of these 24-bit blocks are used to reconstruct the LSF.

Sample MATLAB/Octave code snippet for generating G and H matrices is shown below.

```
P = hex2poly('0xC75');
[H,G] = cyclgen(23, P);

G_P = G(1:12, 1:11);
I_K = eye(12);
G = [I_K G_P P.'];
H = [transpose([G_P P.']) I_K];
```

Appendix E

Code Puncturing

Removing some of the bits from the convolutional coder's output is called code puncturing. The nominal coding rate of the encoder used in M17 is ½. This means the encoder outputs two bits for every bit of the input data stream. To get other (higher) coding rates, a puncturing scheme has to be used.

Two different puncturing schemes are used in M17 stream mode:

1. P_1 leaving 46 from 61 encoded bits
2. P_2 leaving 11 from 12 encoded bits

Scheme P_1 is used for the *link setup frame*, taking 488 bits of encoded data and selecting 368 bits. The $gcd(368, 488)$ is 8 which, when used to divide, leaves 46 and 61 bits. However, a full puncture pattern requires the puncturing matrix entries count to be divisible by the number of encoding polynomials. For this case a partial puncture matrix is used. It has 61 entries with 46 of them being ones and shall be used 8 times, repeatedly. The construction of the partial puncturing pattern P_1 is as follows:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \tag{E.1}$$

$$P_1 = \begin{bmatrix} 1 & M_1 & \cdots & M_{15} \end{bmatrix} \tag{E.2}$$

In which M is a standard 2/3 rate puncture matrix and is used 15 times, along with a leading 1 to form P_1, an array of length 61.

The first pass of the partial puncturer discards G_1 bits only, second pass discards G_2, third - G_1 again, and so on. This ensures that both bits are punctured out evenly.

Scheme P_2 is for frames (excluding LICH chunks, which are coded differently). This takes 296 encoded bits and selects 272 of them. Every 12th bit is being punctured out, leaving 272 bits. The full matrix shall have 12 entries with 11 being ones.

The puncturing scheme P_2 is defined by its partial puncturing matrix:

$$P_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{E.3}$$

The linearized representations are:

```
P1 = [1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
      1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
      0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1]
```

```
P2 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
```

One additional puncturing scheme P_3 is used in the Packet Mode. The puncturing scheme is defined by its puncturing matrix:

$$P_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{E.4}$$

The linearized representation is:

```
P3 = [1, 1, 1, 1, 1, 1, 1, 0]
```

Appendix F

Interleaving

For interleaving a Quadratic Permutation Polynomial (QPP) is used. The polynomial

$\pi(x) = (45x + 92x^2) \mod 368$

is used for a 368 bit interleaving pattern QPP.

input index	output index	input index	output index	input index	output index	input index	output index
0	0	92	92	184	184	276	276
1	137	93	229	185	321	277	45
2	90	94	182	186	274	278	366
3	227	95	319	187	43	279	135
4	180	96	272	188	364	280	88
5	317	97	41	189	133	281	225
6	270	98	362	190	86	282	178
7	39	99	131	191	223	283	315
8	360	100	84	192	176	284	268
9	129	101	221	193	313	285	37
10	82	102	174	194	266	286	358
11	219	103	311	195	35	287	127
12	172	104	264	196	356	288	80
13	309	105	33	197	125	289	217
14	262	106	354	198	78	290	170
15	31	107	123	199	215	291	307
16	352	108	76	200	168	292	260
17	121	109	213	201	305	293	29
18	74	110	166	202	258	294	350

input index	output index	input index	output index	input index	output index	input index	output index
19	211	111	303	203	27	295	119
20	164	112	256	204	348	296	72
21	301	113	25	205	117	297	209
22	254	114	346	206	70	298	162
23	23	115	115	207	207	299	299
24	344	116	68	208	160	300	252
25	113	117	205	209	297	301	21
26	66	118	158	210	250	302	342
27	203	119	295	211	19	303	111
28	156	120	248	212	340	304	64
29	293	121	17	213	109	305	201
30	246	122	338	214	62	306	154
31	15	123	107	215	199	307	291
32	336	124	60	216	152	308	244
33	105	125	197	217	289	309	13
34	58	126	150	218	242	310	334
35	195	127	287	219	11	311	103
36	148	128	240	220	332	312	56
37	285	129	9	221	101	313	193
38	238	130	330	222	54	314	146
39	7	131	99	223	191	315	283
40	328	132	52	224	144	316	236
41	97	133	189	225	281	317	5
42	50	134	142	226	234	318	326
43	187	135	279	227	3	319	95
44	140	136	232	228	324	320	48
45	277	137	1	229	93	321	185
46	230	138	322	230	46	322	138
47	367	139	91	231	183	323	275
48	320	140	44	232	136	324	228
49	89	141	181	233	273	325	365
50	42	142	134	234	226	326	318

input index	output index	input index	output index	input index	output index	input index	output index
51	179	143	271	235	363	327	87
52	132	144	224	236	316	328	40
53	269	145	361	237	85	329	177
54	222	146	314	238	38	330	130
55	359	147	83	239	175	331	267
56	312	148	36	240	128	332	220
57	81	149	173	241	265	333	357
58	34	150	126	242	218	334	310
59	171	151	263	243	355	335	79
60	124	152	216	244	308	336	32
61	261	153	353	245	77	337	169
62	214	154	306	246	30	338	122
63	351	155	75	247	167	339	259
64	304	156	28	248	120	340	212
65	73	157	165	249	257	341	349
66	26	158	118	250	210	342	302
67	163	159	255	251	347	343	71
68	116	160	208	252	300	344	24
69	253	161	345	253	69	345	161
70	206	162	298	254	22	346	114
71	343	163	67	255	159	347	251
72	296	164	20	256	112	348	204
73	65	165	157	257	249	349	341
74	18	166	110	258	202	350	294
75	155	167	247	259	339	351	63
76	108	168	200	260	292	352	16
77	245	169	337	261	61	353	153
78	198	170	290	262	14	354	106
79	335	171	59	263	151	355	243
80	288	172	12	264	104	356	196
81	57	173	149	265	241	357	333
82	10	174	102	266	194	358	286

input index	output index	input index	output index	input index	output index	input index	output index
83	147	175	239	267	331	359	55
84	100	176	192	268	284	360	8
85	237	177	329	269	53	361	145
86	190	178	282	270	6	362	98
87	327	179	51	271	143	363	235
88	280	180	4	272	96	364	188
89	49	181	141	273	233	365	325
90	2	182	94	274	186	366	278
91	139	183	231	275	323	367	47

F.1 References

- Trifina Lucian, Tarniceriu Daniela, Munteanu Valeriu. “Improved QPP Interleavers for LTE Standard.” ISSCS 2011 - International Symposium on Signals, Circuits and Systems (2011)

Appendix G

BERT Details

G.1 PRBS Generation

The PRBS uses the ITU standard PRBS9 polynomial: $x^9 + x^5 + 1$

This is the traditional form for a linear feedback shift register (LFSR) used to generate a pseudorandom binary sequence.

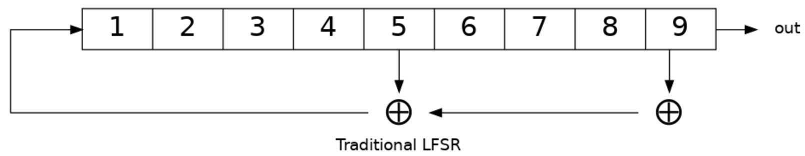


Figure G.1: Traditional form LFSR

However, the M17 LFSR is a slightly different. The M17 PRBS9 uses the generated bit as the output bit rather than the high-bit before the shift.

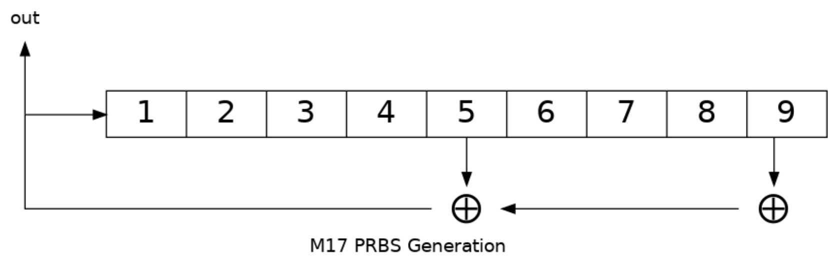


Figure G.2: M17 LFSR

This will result in the same sequence, just shifted by nine bits.

$M17\_PRBS_n = PRBS9_{n+8}$

The reason for this is that it allows for easier synchronization. This is equivalent to a multiplicative scrambler (a self-synchronizing scrambler) fed with a stream of 0s.

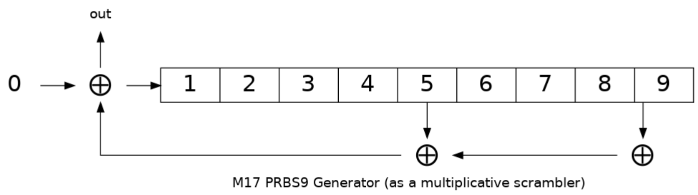


Figure G.3: M17 PRBS9 Generator

```
class PRBS9 {
  static constexpr uint16_t MASK = 0x1FF;
  static constexpr uint8_t TAP_1 = 8;       // Bit 9
  static constexpr uint8_t TAP_2 = 4;       // Bit 5

  uint16_t state = 1;

  public:
  bool generate()
  {
    bool result = ((state >> TAP_1) ^ (state >> TAP_2)) & 1;
    state = ((state << 1) | result) & MASK;
    return result;
  }
  ...
};
```

The PRBS9 SHOULD be initialized with a state of 1.

G.2 PRBS Receiver

The receiver detects the frame is a BERT Frame based on the Sync Burst received. If the PRBS9 generator is reset at this point, the sender and receiver should be synchronized at the start. This, however, is not common nor is it required. PRBS generators can be self-synchronizing.

G.2.1 Synchronization

The receiver will synchronize the PRBS by first XORing the received bit with the LFSR taps. If the result of the XOR is a 1, it is an error (the expected feedback bit and the input do not match) and the sync count is reset. The received bit is then also shifted into the LFSR state register. Once a sequence of eighteen (18) consecutive good bits are recovered (twice the length of the LFSR), the stream is considered synchronized.

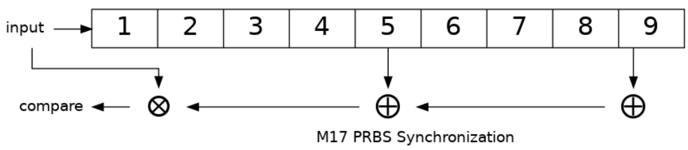


Figure G.4: M17 PRBS9 Synchronization

During synchronization, bits received and bit errors are not counted towards the overall bit error rate.

```
class PRBS9 {
  ...
  static constexpr uint8_t LOCK_COUNT = 18;   // 18 consecutive good bits.
```

```
    ...
    // PRBS Synchronizer. Returns 0 if the bit matches the PRBS, otherwise 1.
    // When synchronizing the LFSR used in the PRBS, a single bad input bit
    // will result in 3 error bits being emitted, one for each tap in the LFSR.
    bool synchronize(bool bit)
    {
      bool result = (bit ^ (state >> TAP_1) ^ (state >> TAP_2)) & 1;
      state = ((state << 1) | bit) & MASK;
      if (result) {
        sync_count = 0; // error
      } else {
        if (++sync_count == LOCK_COUNT) {
          synced = true;
          ...
        }
      }
      return result;
    }
    ...
};
```

G.2.2 Counting Bit Errors

After synchronization, BERT mode switches to error-counting mode, where the received bits are compared to a free-running PRBS9 generator. Each bit that does not match the output of the free-running LFSR is counted as a bit error.

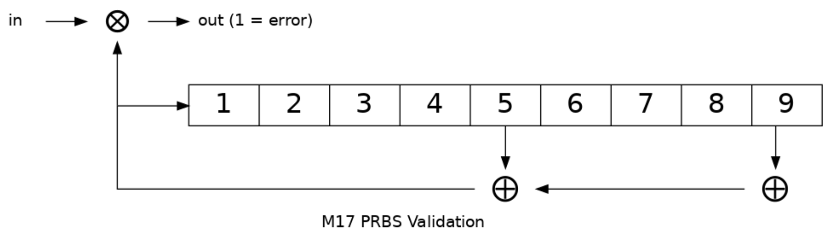


Figure G.5: M17 PRBS9 Validation

```
class PRBS9 {
    ...
    // PRBS validator.  Returns 0 if the bit matches the PRBS, otherwise 1.
    // The results are only valid when sync() returns true;
    bool validate(bool bit)
    {
      bool result;
      if (!synced) {
        result = synchronize(bit);
      } else {
        // PRBS is now free-running.
        result = bit ^ generate();
        count_errors(result);
      }
      return result;
    }
    ...
};
```

G.2.3 Resynchronization

The receiver must keep track of the number of bit errors over a period of 128 bits. If more than 18 bit errors occur, the synchronization process starts anew. This is necessary in the case of missed frames or other serious synchronization issues.

Bits received and errors which occur during resynchronization are not counted towards the bit error rate.

G.3 References

- ITU O.150 : Digital test patterns for performance measurements on digital transmission equipment
- PRBS (according ITU-T O.150) and Bit-Sequence Tester : VHDL-Modules

Appendix H

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the

History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix I

GNU General Public License, version 2

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify

the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to

the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show` `w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.