8.9 The FlexLink Standard

The FlexLink standard is an open physical layer, PHY, and media access controller layer, MAC, specification of a packet based OFDM radio link used for applications where the transceiver stations may be in motion and/or stationery. The time division duplex, TDD, link will support point to point and point to multipoint communication. The specification originally took shape to provide the framework for a radio link that supports video transmission from fast moving terminals such as drones. However, in general, the specification provides for both high reliability and high throughput transmissions between multiple terminals that are in motion or stationery. Whereas modem specifications, such as 3GPP's 5G New Radio or IEEE 802.11 WLAN, already exist for similar applications, they have certain drawbacks that FlexLink attempts to sidestep. The following table provides a brief overview of the features of the different technologies.

Table 8-20: Comparison of Different Standards

	5G New Radio	Wi-Fi 6 / 7	FlexLink
Data Throughput	Low to Very High < 100KBPS to > 1GBPS	Medium to Very High 6MBPS to > 1GBPS	Low to High < 100KBPS to > 100MBPS
Mobility	Good	Poor	Excellent
Bandwidth	<= 400MHz	<= 160MHz	20MHz
Data Transfer	Cellular Network Based	Packet Based	Packet Based
Implementation	Very Difficult	Difficult	Reasonable
Documentation	Extensive but Awful	Reasonable	Good

Whereas the 4G LTE and 5G New Radio specifications do not call for RF packet-based transmissions, they are designed from the ground up to enable communication in a mobile environment. FlexLink borrows many concepts from these standards but enhances mobility by supporting terminals that move faster than road or rail vehicles covered by cellular technologies. Another advantage is that the FlexLink specification does not need to be as complicated as those of the aforementioned 3GPP, as it does not need to function in complex cellular network.

The IEEE Wi-Fi technologies make little effort to support moving terminals as they are geared to stationary indoor communication, and neither do they support high reliability / low throughput transmissions. The Wi-Fi standards are designed specifically to deliver high throughput to many different stationary users via RF packets. As the initial FlexLink specification is limited to a 20MHz bandwidth, its throughput will not reach the large numbers of the 5G and WLAN, and it doesn't need to. In all other respects, it will improve on previous technologies in terms of transmission robustness, flexibility of use, implementation simplicity and documentation.

The FlexLink specification is not simply a dry explanation of how to generate a transmit signal, but rather teaches the technology in a concise way such that a single lead engineer can understand it and a small team of engineers can implement it. In addition to the written documentation, FlexLink is supported by a full suite of Python scripts, which models its performance.

Table Of Contents

8.9.1 Features that Drive the FlexLink Specification

FlexLink is designed to act as a point-to-point link, where a service terminal communicates with a client terminal. It further supports a point-to-multipoint configuration where the service terminal controls and communicates with several clients at once. Any or all terminals may be in motion or stationery. Point-to-point should also be a **pear-to-pear** connection.

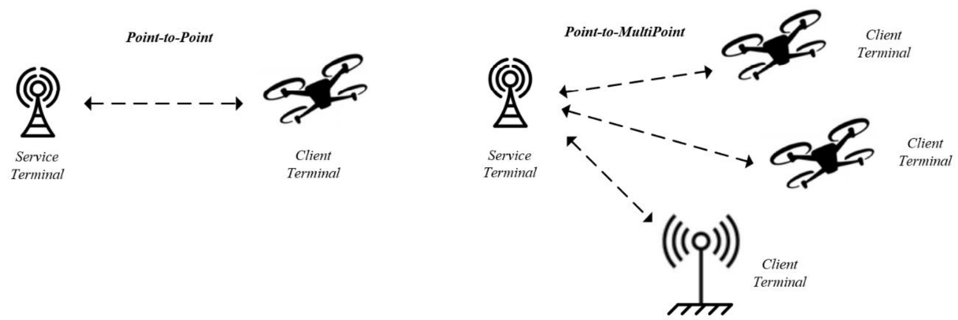


Figure 8-83: Point-to-Point and Point-to-Multipoint Configurations

Mobility

Given the fact that the terminals may be in motion dictates that the method of modulation should be either OFDM, used in 802.11 Wi-Fi Modems and in the 4G/5G downlink, or SC-FDMA, which is used in 4G/5G uplink. The ease of equalization and straight forward use of multiple antennas offered by these modulation techniques make them the only viable option for this application. Whereas SC-FDMA signals do feature better peak to average power ratio than OFDM and therefore facilitate more efficient power amplification, both link directions were chosen to use OFDM modulation to minimize the complexity of the standard and subsequent implementation effort. To ensure that the frequency response remains flat over a single subcarrier, the subcarrier spacing was chosen to be 20MHz/1024 = 19.53125KHz. FlexLink will support Doppler frequency that are significantly higher than those supported in LTE 4G.

Data Throughput

FlexLink technology does not require the massive data throughput offered by modern 5G and WLAN technologies. A maximum theoretical throughput of 110-120MBPS (1024 QAM) allows us to use a standard 20MHz channel and avoid the use of spatial multiplexing MIMO. This simplifies the standard as RF radio components with this channel size are readily available and the lack of spatial multiplexing MIMO significantly eases the implementation effort. Also note that spatial multiplexing MIMO is only a realistic option when both terminals are stationary and multipath conditions are well behaved. The lack of spatial multiplexing is not a significant hinderance as multiple antennas at the transmitter and receivers can equivalently be used to boost throughput via the use of transmit and receive diversity, which don't require stationary positioning. The specification will likely be extended in the future to support higher and lower bandwidths, thus raising the throughput ceiling.

Robust Transmission

FlexLink technology is specifically conceived to process low data rate information in poor signal to noise ratio conditions. Transmissions can include both high data rate and low data rate / high reliability components, allowing the link to fail gradually rather than abruptly. For example, as the link to a drone becomes less reliable due to noise, video transmission may cease but telemetry continues unabated as it can be set up with better error protection. At that point, the drone can switch to transmitting images rather than video as these are lower data rate messages. In tactical situations, low SNR transmissions from a control terminal to a drone have the added advantage that they are very hard to detect and thus jam. But in general, being able to receive information at low SNR extends the useful range of the link, thus making it more useful. In order to support both high and low throughput, the construction and layout of synchronization and reference information is quite flexible. The sequences embedded in the preamble allow synchronization down to approximately -8dB of signal to noise ratio. Once synchronization is fully established, transmission will be possible without the use of the preamble to signal to noise ratios below -10dB.

Multiple Antennas Layout

FlexLink technology stipulates the use of one or two transmit antennas. The use of two transmit antennas provides transmit diversity via space frequency block coding. The number of receive antennas is not specified. At a minimum, a single receive antenna is required and any further antenna / receiver port may be used to enable receive diversity via maximum ratio combining or any other combing scheme. The figure below illustrates the standard antenna configuration for a link featuring no more than two antennas at each terminal. Clearly, FlexLink supports the SISO (single input single output) configuration as this is the most basic setup needed to establish an RF link. Whereas technically, FlexLink does not mandate any additional antennas, a 2x2 MIMO configuration using transmit and receive diversity via SFBC (space frequency block coding) and MRC (maximum ratio combining) is recommended. Given that a second transmit antenna is an option in the specification, all FlexLink receivers must be able to decode the space frequency block coding at each receiver path.

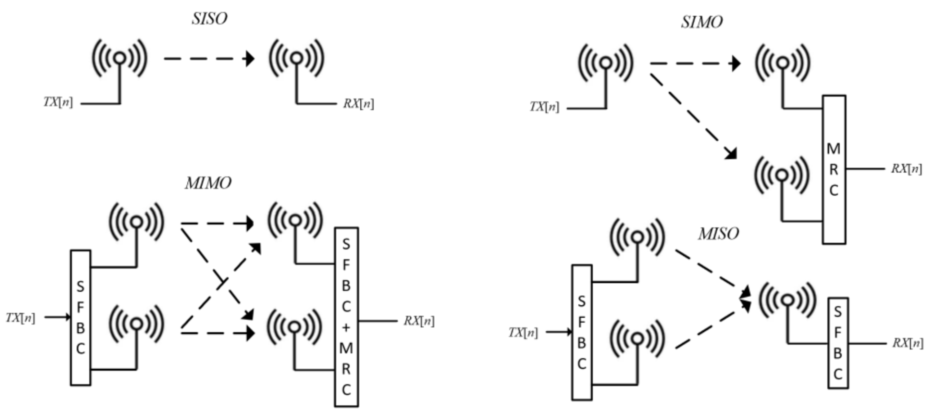


Figure 8-84: Basic Standard Antenna Configurations

Packet Transmission

The link is packet-based, meaning that all synchronization and data information is contained within a bound time period. However, there are time stamps included in the signal field that allow both timing and frequency synchronization by observing consecutive packets.

8.9.2 The FlexLink Protocol Stack

The FlexLink technology specifies a limited set of sublayers in the overall protocol stack. It is limited to portions of the *data link layer* and the *physical layer*, whereas all other components of the protocol stack such as application and networking layers may be customized by the implementer.

The Data Link Layer

The data link layer is traditionally divided into a *radio link layer*, or RLL, and *media access control* layer, or MAC. The radio link layer ensures that data provided from the upper layers is partitioned in a way that the MAC can easily place it into the physical layer. Whereas FlexLink does not define this layer, it nevertheless assumes that it exists, and it is referred to as such in this specification. The radio link layer shall be customized by the implementer.

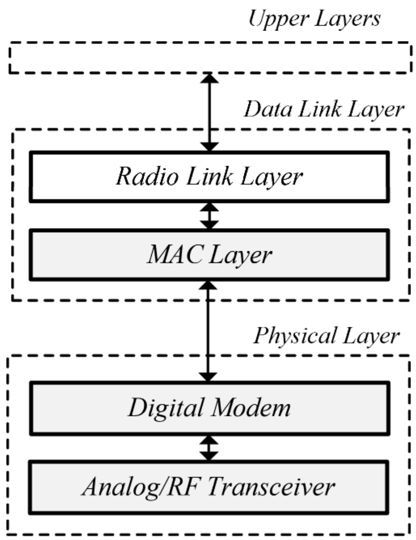


Figure 8-85: FlexLink Protocol Stack

The Media Access Controller

The primary purpose of the media access controller is to manage and provide access to the physical radio link. As such it has the following responsibilities.

→ Cooperation with the radio link layer to disassemble and reassemble upper layer data sequences for mapping into and demapping from the physical layer.

→ The organization of user data for all transport blocks into the physical layer's resource grid.

→ Continuous analysis of the RF link to facilitate link adaptation.

→ The proper management of control information and signal field content.

→ Management of the automatic request repeat mechanism, also called ARQ.

The Digital Modem

The primary purpose of the digital modem is to convert data provided by the MAC into an IQ stream that modifies the RF carrier on the transmit side and to convert that IQ stream back into data for MAC inside the receiver. The main responsibilities of the digital modem are as follows:

→ FlexLink uses OFDM to map and demap information onto the IQ waveform. The physical layer must implement the FFT and IFFT mechanisms to implement the OFDM process.

→ Implementation of all error correction and detection mechanisms.

→ Implementation of time and frequency synchronization mechanisms.

→ Computation of the frequency response and implementation of the equalization process.

→ Implementation of the rate matching, QAM mapping / demapping and interleaving processes.

→ Implementation of transmitter and receiver-diversity mechanisms.

The Analog / RF Transceiver

The FlexLink specification does not place many restrictions on the analog transceiver as the standard is meant to be used in a variety of applications that may require different levels of RF output power and carrier center frequency. There is one aspect of RF performance that must be met in all scenarios.

→ To avoid loss of orthogonality in the OFDM demodulator, the phase noise performance of the LO synthesizer shall be no worse than the following.

Table 8-21: Phase Noise Performance of FlexLink Compatible Transceiver

Frequency	Phase Noise in dBc/Hz
10 KHz	-95
100 KHz	-100
1 MHz	-121
10 MHz	-138

8.9.3 The Physical Layer

The following table provides a brief overview of some of the physical layer parameters of the FlexLink standard.

Table 8-22: Basic Physical Layer Parameters

Specification	Comments
Method of Modulation	Orthogonal Division Multiplexing (OFDM)
Base Sample Rate	20MSPS / 40MSPS (Oversampled)
I/FFT Size	N = 1024 / N = 2048 (Oversampled)
Subcarrier Spacing	20MHz/1024 = 19.53125KHz
OFDM Symbol Period (IFFT+CP)	1024 + 116 = 1140 samples at 20MHz
OFDM Symbol Period (IFFT+CP)	51.2e-6 sec + 5.8e-6 sec = 57e-6 seconds
Bandwidth WLAN Mode	841 Subcarriers ≈ 16.5MHz
Bandwidth LTE Mode	913 Subcarriers ≈ 17.9MHz
Reference Symbol Periodicity	Every 1st, 3rd, 6th, 12^{th} OFDM Symbol
Reference Signal Spacing	Every 3^{rd}, 6^{th}, 12^{th}, 24^{th} Subcarrier
FEC (Payloads A and B)	LDPC
FEC (Signal Field)	Binary Convolutional Coding / Polar Coding (optional)
QAM Constellations	BPSK, QPSK, 16QAM, 64QAM, (256QAM, 1024QAM optional)
Max Data Rate (1024 QAM)	118 MBPS (LTE BW) / 110 MBPS (WLAN BW)
Min Data Rate (BPSK)	200KBPS
AGC Burst Length	5e-6 seconds (100 samples at 20MHz)
Preamble A Length	50e-6 / 250e-6 (1000 / 5000 samples at 20MHz)
Preamble B Length	5.12e-6 (1024 samples at 20MHz)

Table 8-23: Basic Terminology

Term	Explanation
TDD	Time Division Duplexing
Downlink	Link Direction from the service to the client terminal
Uplink	Link Direction from the client to the service terminal
QAM	Quadrature Amplitude Modulation
SFBC	Space Frequency Block Coding

8.9.4 Packet Construction

The FlexLink specification dictates a TDD type link, in which transmit packets and receive packets do not overlay in time.

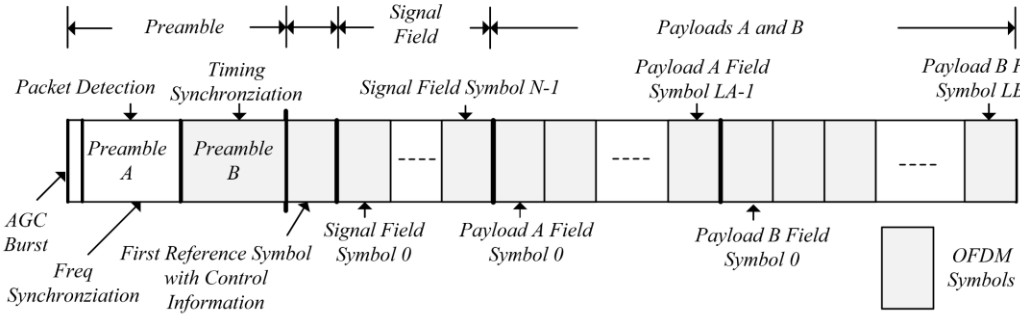


Figure 8-86: Basic Packet Structure for FlexLink

FlexLink provides a preamble, a signal field as well as two payload fields: Payload *A* and *B*. The payloads are separated such that each may have different modulation and coding schemes, and the packet can thus convey messages with different reliability. Both payloads *A* and *B* can carry MAC header information, user data and responses to automatic repeat requests (ARQ).

8.9.4.1 The Preamble

The preamble is composed of three sections each facilitating different synchronization tasks.

AGC Burst

The initial portion of the preamble is a five microsecond long wideband AGC (automatic gain control) burst that facilitates the convergence of all gain stages in the analog radio portion of the modem with the goal of providing IQ data that is within the proper ADC input range.

Preamble A

Preamble A is a custom sequence that facilitates the following synchronization tasks:

Packet detection – This algorithm uses the *preamble A* to discern and distinguish a FlexLink packet from any other that might legally be transmitted in the band.

Frequency offset acquisition – This algorithm uses the *preamble A* to compute and correct the frequency offset to avoid loss of orthogonality in the OFDM demodulator.

Preamble B

Preamble B is a custom OFDM symbol that facilitates the following synchronization tasks.

FFT Timing determination – *Preamble B* consists of a sequence of samples that will have excellent autocorrelation properties with a distinct peak that allows the receiver to determine the arrival time of the strongest RF path.

Channel Estimation - *Preamble B* may also be used to estimate the frequency response of the channel and subsequently help to equalize the control information in the first reference symbol.

8.9.4.2 The Resource Grid, Control Information and Reference Signal Locations

The resource grid is a convenient construct allowing us to organize information for subsequent OFDM modulation. The figure on the following page illustrates the resource grid specified by the FlexLink standard. FlexLink supports two bandwidths that may be considered as a 20MHz channel. The first bandwidth is composed of 913 subcarriers, which at a subcarrier spacing of 20MHz/1024, or 19.53125KHz, covers 17.84MHz. The second bandwidth is composed of 841 subcarriers and covers 16.43MHz. These layouts are chosen to be compatible with RF chip sets and filters used for LTE and WLAN. Note that the resource grid is further divided into resource blocks, consisting of 12 subcarriers, and triplets, consisting of 3 subcarriers. No information of any kind may be placed at the center DC subcarrier.

→ LTE Variant: 913 subcarriers, (913 – 1)/3 = 304 triplets, (913 -1)/12 = 76 resource blocks
→ WLAN Variant: 841 subcarriers, (841 – 1)/3 = 280 triplets, (841 -1)/12 = 70 resource blocks

The First Reference Symbol with Control Information

The first reference (OFDM) symbol constitutes the initial opportunity for the receiver to observe reference signals from both TX antenna ports. It allows the receiver to compute a channel estimate and SNR value for both transmit paths. The first reference symbol features a static subcarrier arrangement of reference signals and control information, but no user data.

Control information is embedded within the first reference symbol. The information is embedded this way so that the transmitting MAC can control some aspects of the link on the fly, without previously having to inform the receiver of the configuration change. This is also important so that the transmitter can communicate in an ad hoc manner with unknown receivers. Control information, which is mapped using BPSK, is transmitted on TX port 0 only and must be equalized using reference signals associated with TX port 0 before being decoded.

Table 8-24: Understanding Control Information

Information	Number of Bits	Description
Reference Symbol Periodicity Index	2 bits	[0, 1, 2, 3] →1, 3, 6, and 12 OFDM Symbols
Reference Signal Spacing Index	2 bits	[0, 1, 2, 3] → Every 3rd, 6th, 12th, 24th Subcarrier
Number Signal Field Symbols Index	2 bits	[0, 1, 2, 3] → The number of OFDM symbols for the signal field: 1, 2, 4, or 10
Signal Field Format Index	2 bits	0/1/2/3 – Format 1 / Format 2 / Format 3 / Format 4
Signal Field Modulation Flag	1 bit	[0, 1] → BPSK, QPSK.
Number Tx Antenna Ports Flag	1 bit	[0, 1] → 1 or 2 Transmit antennas
DC Subcarriers Flag	1 bit	[0, 1] → 1 Subcarrier / 13 subcarriers
Parity Check (even parity)	1 bit	Basic Error Detection method
	12 bits	

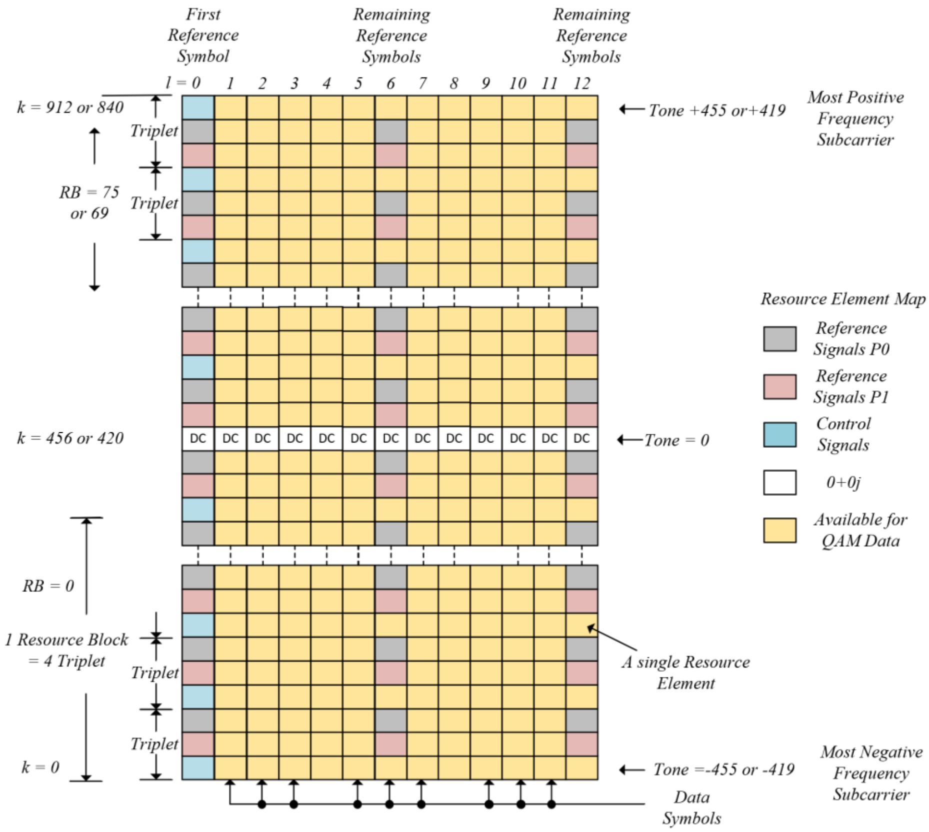


Figure 8-87: Resource Grid Layout for FlexLink Channel

Before explaining how the reference signals are arranged and what the meaning of the control information is, please note the following nomenclature related to the resource grid. Subcarriers and OFDM symbols are enumerated via the variables k and l, as done in LTE. The figure also provides the IEEE convention, which identifies the subcarriers as tones.

Table 8-25: Nomenclature Related to the Resource Grid

Term	Explanation
Resource Grid	A convenient construct used to organize information into OFDM Symbols.
Resource Element	An element in the resource grid holding a single QAM value at a single subcarrier and OFDM symbol.
QAM Symbol	A single complex number representing a position in a QAM Constellation. Every resource element will be occupied by one QAM symbol.
Resource Block	A collection of 4 Triplets = 12 Resource Elements (76 LTE RB, 70 WLAN RB)
k	The subcarrier index (as used in the 3GPP specification)
l	The OFDM symbol index
Tone	The subcarrier index (as defined in the IEEE specifications)
P0 / P1	Port 0 / Port 1 refer to the first and second TX Antennas respectively
Reference Signal	A resource element with a value known to the receiver ahead of time. Reference signals are used for channel and SNR estimation.
Reference Symbol	An OFDM symbol that contains reference signals.
Data Symbol	An OFDM symbol that exclusively holds user data.

Reference Symbol Periodicity

The reference symbols may be spaced at a periodicity of 1, 3, 6 or 12 OFDM symbols. The previous figure shows a periodicity of 6 OFDM symbols. The faster the relative motion of the different terminals, the faster the multipath channel changes. This requires a tight spacing of reference symbols.

→ A spacing of 1 OFDM symbol is provided for two scenarios. For transmission in very high Doppler conditions, as well as in conjunction with coarse reference signal subcarrier spacing such as 12 or 24 subcarriers. This arrangement can be used for stationary terminals that only need to track phase walk and timing drift, but no actual change in multipath conditions. This method is similar to pilot tone construction in 802.11 OFDM modems.

→ A spacing of 3 OFDM symbols is used for fast moving terminals such as high speed drones.

→ A spacing of 6 or 12 OFDM symbols is used for terminals that move with progressively less velocity.

A larger spacing of reference symbols frees up more resources for user data. However, a larger spacing than 12 OFDM symbols provides a negligible increase in user data resources, and the control information therefore does not accommodate further options. Including the reference symbol periodicity in the control information is important as a terminal in motion can thus autonomously change the spacing as it changes velocity and thus Doppler conditions.

The reference symbol spacing needed to resolve Doppler, or the equivalent frequency offset, can be determined by first understanding the kind of Doppler we will face. For the drone application, we may assume that the maximum doppler will be related to a maximum speed, i.e. 300Kph, or 84 m/sec. The maximum Doppler frequency at a center frequency of 5.9GHz is computed as follows:

$$F_{Doppler} = F_{Center} \frac{V_{max} m/s}{300e6 m/s} Hz = 5.9e9 \ \frac{84}{300e6} Hz = 1652 Hz$$

The maximum absolute Doppler frequency that can be resolved by a reference signal spacing of T seconds is proportional to the Nyquist BW of $1/T$ Hz. Therefore, for a spacing of 3 OFDM symbols, where each OFDM symbol features a period of 57 microseconds, the maximum detectable Doppler is $\pm 1/(2\cdot T) = \pm 1/(2\cdot 3\cdot 57e{-6}) \approx \pm$ 3KHz. Note that these calculations assume that the terminals are moving toward or away from one another in a straight line. Any other motion produces less Doppler. See section 7.3.2 for a more detailed description of Doppler. Note further that Doppler appears as a frequency offset, which can be detected during preamble analysis and removed. Therefore, the actual problems faced by the terminal are the differences in Doppler frequency between the different paths, not absolute Doppler frequency.

Reference Signal Spacing along the Frequency Axis

Reference signals may be spaced every 3rd, 6th, 12th, or 24th subcarrier. Note that the first reference symbol is always set to every 3rd subcarrier. There are two aspects of the link that influence the reference signal spacing. The larger the delay spread of the multipath channel, the more quickly the frequency response changes across the bandwidth of the channel. Large delays between paths, which occur for outdoor communication and are typically worse in rural than urban environments, thus require a tighter reference signal spacing than indoor scenarios where the paths will arrive at the receiver with small delays due to the small distance between terminals. A second aspect that influences reference signal spacing is the need for very accurate estimation of the frequency response and signal to noise ratio. The larger the number of reference signals, to which the receiver has access, the more accurately the frequency response of the channel and the signal to noise ratio may be estimated.

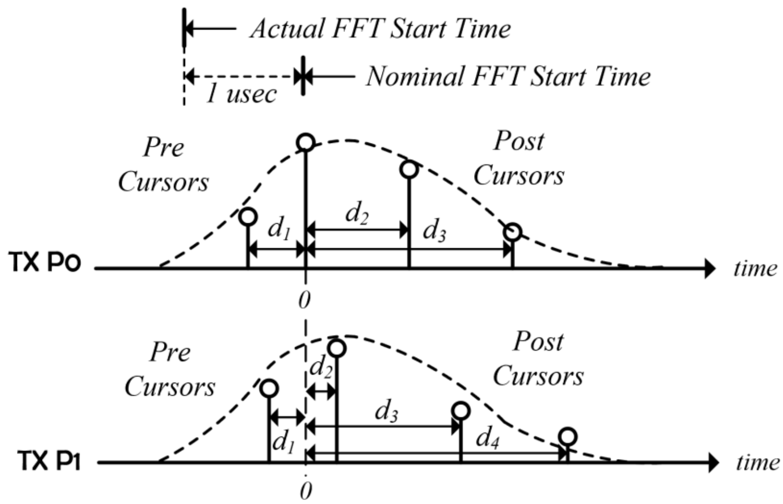


Figure 8-88: Example Channel Impulse Response Due to TX Ports P0 and P1

To better understand how tightly reference signals need to be placed along the frequency axis, let's consider the figure above, which shows the impulse response of the two channels that FlexLink supports. The first channel exists between transmit antenna port P0 and a single receive antenna, whereas the second channel exists between transmit antenna port P1 and that same receive antenna. Note that *preamble B* is used for timing synchronization, meaning that once we have seen the maximum correlation peak produced by the strongest path emanating from transmit port P0, then we will also know when the strongest copy of the IFFT portion of the first OFDM symbol from that port begins. In the figure above, we indicate that moment as the nominal FFT start time at *time* = 0. The frequency response changes at a rate that is proportional to the delay between this nominal FFT start time and the start time of the IFFT portion of the OFDM symbol of the various paths. From section 8.1.3.2, we know that the frequency response of a multipath channel at a single instance of time is defined as follows.

$$FreqResponse(f) = \sum_{p=0}^{P-1} C_p \cdot e^{-j2\pi d_p f}$$

The next figure shows the frequency responses for TX antenna port 0 given three paths. The topmost plot is the frequency response of the strongest path, to which we want to synchronize the FFT operation. The response should be a constant complex value, but let's assume that we can't estimate the arrival time of the strongest correlation peak perfectly and some very small amount of delay remains. The second plot represents the frequency response due to a precursor that arrives at d_1 = 0.5 microseconds ahead of the strongest path. The third plot represents the frequency response of a post cursor arriving with a delay of d_2 = 2 microseconds. The total frequency response is the summation of the constituent responses due to each path.

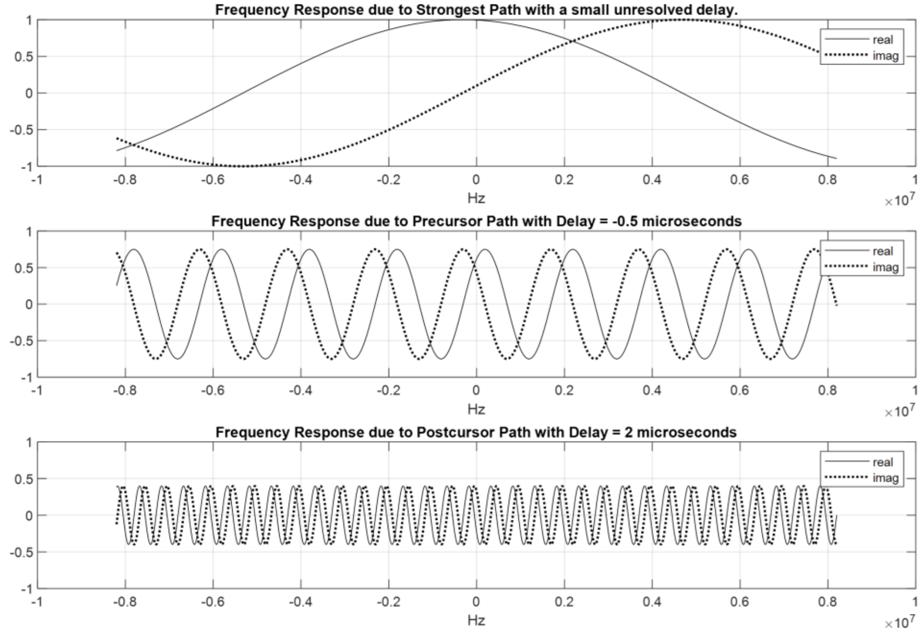


Figure 8-89: Frequency Response due to Three Separate Paths

The reference signals along the frequency axis must be spaced so that they can properly sample the frequency response component that oscillates at the highest rate. The Nyquist theorem tells us that we must sample a time domain signal at twice its highest frequency component. For the complex sinusoid, the minimum sampling rate would be $2f_o$ Hz, with the sample period being $1/(2f_o)$ seconds.

$$ComplexSinusoid(t) = C \cdot e^{-j2\pi d f_o t}$$

In our case, the complex sinusoid extends over frequency rather than time. Whereas this seems a bit awkward as we don't usually encounter signals that change as a function of frequency, the concept is exactly the same. Therefore, the sampling rate is at least $2d_p$, with a sample period no larger than $1/(2d_p)$. FlexLink is set up to handle an absolute maximum delay of 5.8 microseconds, which is the cyclic prefix length. A larger delay would cause intersymbol interference.

$$ComplexSinusoid(f) = C_p \cdot e^{-j2\pi d_p f}$$

The minimum sample period for the maximum delay of about 5.8 microseconds (the CP length) is therefore 1/(2·5.8e-6 seconds) = 86KHz. The tightest available reference signal spacing in FlexLink is 3 subcarrier · 19.53125KHz ≈ 58KHz, which is better than what we need.

Changing from the Nominal to the Actual FFT Start Time

Remember that an OFDM symbol is composed of the output of an IFFT operation and a cyclic prefix that is meant to avoid intersymbol interference. Once we have acquired timing via the correlation against the *preamble B* sequence, we know where in the received signal the IFFT portion of the first OFDM symbols starts. That moment would appear to be the right moment to start reading samples into the FFT buffer. For this reason, we call it the nominal FFT start. However, if a precursor exists, then that precursor overlays the end of the IFFT portion of the strongest path as shown in the figure below. We must therefore choose a different time, the actual FFT start, at which we begin reading samples into the FFT buffer.

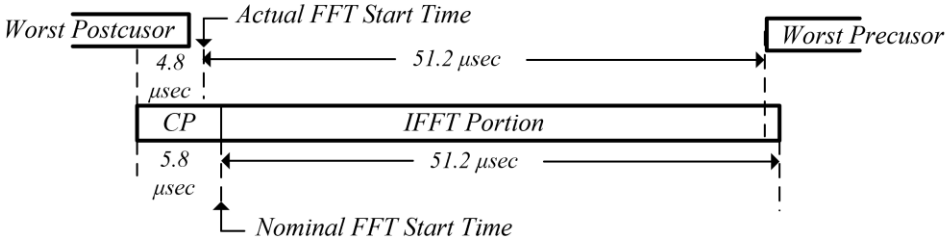


Figure 8-90: Desired vs Actual FFT Start Time

Common sense indicates that the strongest signal will arrive earlier than the weaker ones as those usually travel further and thus experience more attenuation. However, the strongest path only arrives first if there is a direct line of sight. As a rule of thumb, we move the FFT start time 1 microsecond into the cyclic prefix in order to avoid precursors of the next OFDM symbol and post cursors of previous OFDM symbol. Starting the FFT at an earlier time makes it appear as if we are processing a delayed copy of the OFDM symbol, and we should therefore correct the FFT outputs as follows.

$$FFT\_Output(f) = FFT\_Output_{1\mu sec}(f) \cdot e^{-j2\pi(-1e-6)f}$$

Now that the FFT outputs have been corrected, the delay range that we can compensate extends from -1 microsecond to 4.8 microseconds.

<u>Arrangement of Reference Signals</u>

In addition to the previously discussed factors that influence reference symbol periodicity and reference signal spacing, there is another component to the way the reference signals are spaced across the resource grid. They are evenly spaced in both the frequency and in the time domain. This is not the case for LTE, which heavily supports mobility. The reason for the even spacing is to enable the use of the discrete Fourier transform in both frequency and time direction to accurately estimate the frequency response of the communication channel and the signal to noise ratio at the reference signals positions. We can also use this approach to estimate the channel impulse response, which will enable us to maintain synchronization at low signal to noise ratios.

<u>Signal Field</u>

The signal field defines the amount of information transmitted in *payloads A* as well as the way this information is error-encoded and mapped into QAM constellations. The information in the signal field may be spread across 1, 2, 4, or 10 OFDM symbols and be mapped into BPSK or QPSK constellations. The signal field may be constructed in two different configurations, where configuration 1 is primarily used in high mobility or low SNR situations, whereas the other is used for stationary applications with reasonable SNR.

TX Antenna Ports

The FlexLink may operate with either one TX (port P0) or two TX antennas (ports P0 and P1). In the case of two TX antennas, space frequency block coding (section 9.2.2) shall be used to provide transmit diversity.

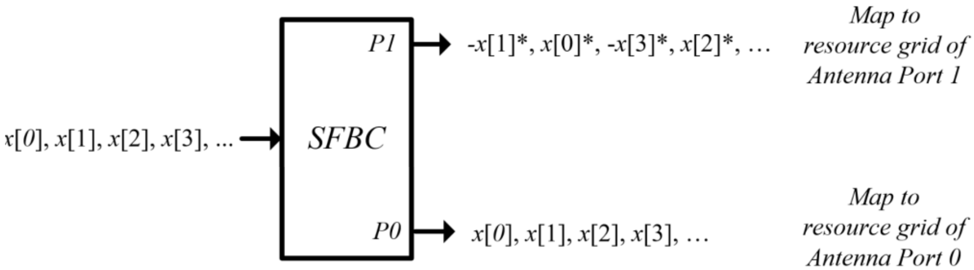


Figure 8-91: Space Frequency Block Coding Mechanism

See section 8.9.5 for more details on how the resource grid is constructed when SFBC is used in the two TX antenna configuration.

DC Subcarriers

Like the LTE specification, the FlexLink standard does not transmit any data at the DC subcarrier. This is done as most Zero-IF receivers must suppress DC offsets in their analog components and are thus unable to process information at the exact center frequency of the signal. Furthermore, in order to remain compatible with RF chipsets designed for WLAN applications, FlexLink may also forgo transmitting user data on the four inner triplets (12 subcarriers) closest to the DC subcarrier. This results in a total of 13 subcarriers, where no data information is placed. The increase in the number DC subcarriers from 1 to 13 may be necessary when using WLAN RF chipsets as the IEEE specifications call out for a subcarrier spacing of 312.5KHz rather than 19.53125KHz. WLAN RF chipsets can therefore assume a larger gap in signal content at DC, allowing them to utilize faster DC offset tracking. Note that both reference signals and control information mapping are prohibited only at the center DC subcarriers, but unlike resource elements carrying data, they should be placed at the remaining 12 DC subcarriers. The receive is at liberty to use or ignore reference signals and control information within these 12 DC subcarriers.

8.9.4.3 OFDM Modulation and Demodulation

Whereas the core sample rate of the FlexLink modem is 20MHz, the analog-to-digital and digital-to-analog converters need to operate at least twice as fast in order to make analog filtering at the ADC inputs and DAC outputs practical. Therefore, in the transmit direction, the resource grid may be fed to a 1024-point or 2048-point IFFT depending on the manner of upsampling desired. If an IFFT of size 2048 is used, the output sequence is automatically sampled at 40MHz, whereas the use of an IFFT of size 1024 will require an upsampling step (see section 3.4.1).

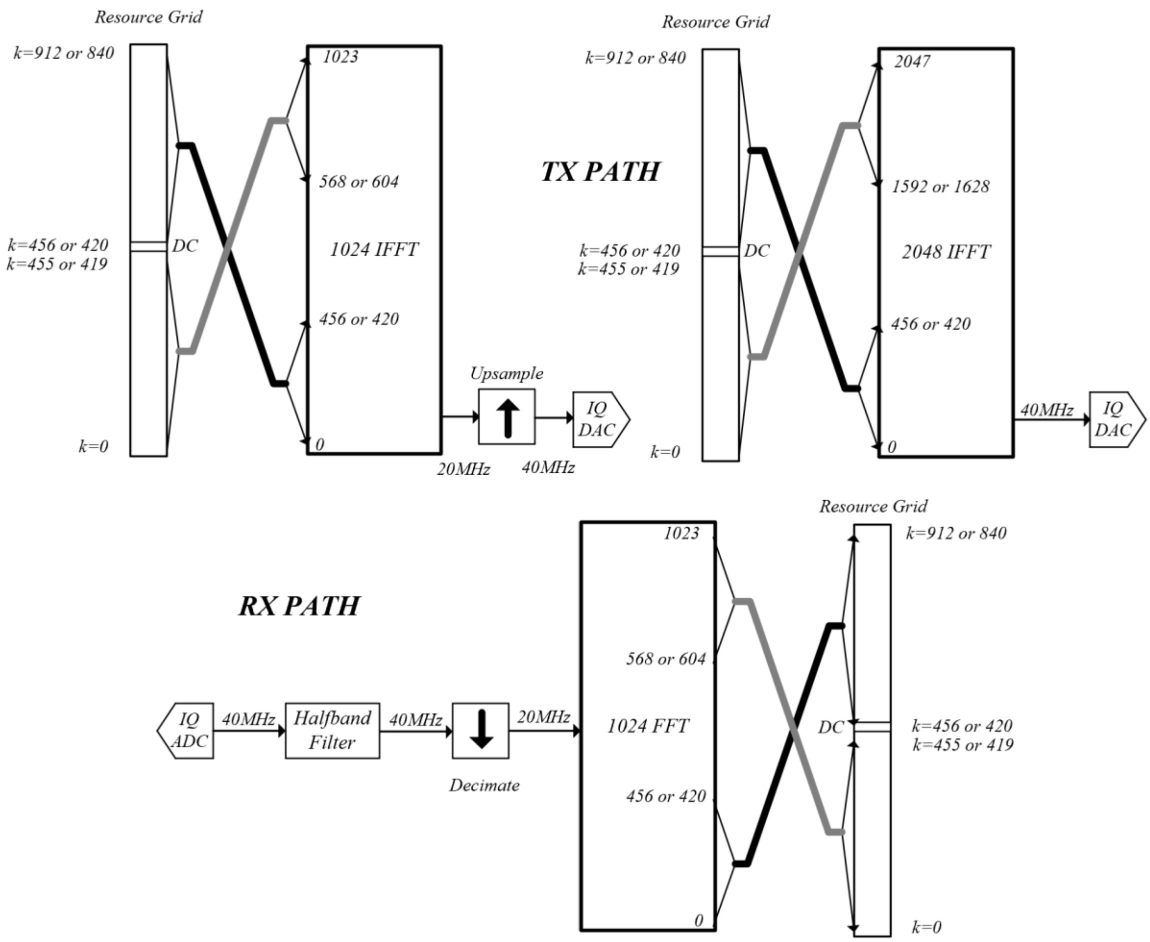


Figure 8-92: FFT/IFFT Mapping to Resource Grid

Similarly, at the receiver, the analog signal needs to be sampled at a rate of at least 40MHz, and then down sampled (see section 3.4.2) to 20MHz before being provided to the FFT input buffer.

Otherwise, the mapping itself is straightforward with the positive frequencies mapping to the lower I/FFT input ports, whereas the negative frequencies connect the upper I/FFT input ports, as dictated by Nyquist's aliasing rules.

Table 8-26: FFT/IFFT Mapping to Resource Grid

Subcarrier Index k	IFFT Port Indices N =1024 / 2048	Frequencies	Bandwidth
0:455	568:1023 / 1592:2047	(-456:1:-1)·SC = -8.90625MHz : -SC	LTE
456:912	0:456	(0:1:456)·SC = 0Hz : 8.90625MHz	LTE
0:419	604:1023 / 1628:2047	(-420:1:-1)·SC = -8.203125MHz : -SC	WLAN
420:840	0:420	(0:1:420)·SC = 0Hz : 8.203125MHz	WLAN

The following MatLab code implements the OFDM modulator for the FlexLink standard.

```
function OutputSequence = OfdmModulator(ResourceGrid, OutputSampleRate)

% Determine the dimensions of the resource grid
[NumSubcarriers, NumOfdmSymbols] = size(ResourceGrid);

% Error checking (NumSubcarriers = 913 or 841 for the LTE or WLAN bandwidth)
assert(NumSubcarriers == 913    || NumSubcarriers == 841,    "Invalid number of subcarriers.");
assert(OutputSampleRate == 20e6 || OutputSampleRate == 40e6, "Invalid output sample rate.");

% Pick the IFFT size
if(OutputSampleRate == 20e6)
    IFFT_Size = 1024;   % IFFT length (1024/20MHz) = 51.2 microseconds
    CP_Length = 116;    % CP length   (116/20MHz)  =  5.8 microseconds
else
    IFFT_Size = 2048;   % IFFT length (2048/40MHz) = 51.2 microseconds
    CP_Length = 232;    % CP length   (232/40MHz)  =  5.8 microseconds
end

OfdmSymbolLength = CP_Length + IFFT_Size;

% Define the positive and negative frequency subcarriers in the resource grid
% k = 420 and 456 are the center DC carrier and thus belong to the positive frequencies
bIsLteBandwidth = NumSubcarriers == 913;
if(bIsLteBandwidth)
    PosSubcarrierIndices = 456:912; NegSubcarrierIndices = 0:455;
else
    PosSubcarrierIndices = 420:840; NegSubcarrierIndices = 0:419;
end

% Define the positive and negative frequency indices in the IFFT input buffer
if(bIsLteBandwidth)
    PosIfftIndices    = 0:456;
    if(IFFT_Size == 1024); NegIfftIndices = 568:1023;
    else;                  NegIfftIndices = 1592:2047; end
else
    PosIfftIndices    = 0:420;
    if(IFFT_Size == 1024); NegIfftIndices =  604:1023;
    else;                  NegIfftIndices =  1628:2047; end
end

% Start the OFDM Modulation Process (l is the 0 based OFDM symbol index)
OutputSequence      = zeros(1, NumOfdmSymbols * OfdmSymbolLength);
IfftInputBuffer     = zeros(IFFT_Size, 1);
OutputSequenceIndex = 1; % Due to MatLab indexing (otherwise = 0)
for l = 0:NumOfdmSymbols - 1
    OfdmSymbol   = zeros(1, OfdmSymbolLength);
    IfftInputBuffer(PosIfftIndices + 1, 1) = ResourceGrid(PosSubcarrierIndices + 1, l+1);
    IfftInputBuffer(NegIfftIndices + 1, 1) = ResourceGrid(NegSubcarrierIndices + 1, l+1);
    IfftOutputBuffer                       = ifft(IfftInputBuffer);
    % Fetch the Cyclic Prefix
    CyclicPrefix     = IfftOutputBuffer(IFFT_Size - CP_Length + 1:end, 1);
    % Place the cyclic prefix at the start of the OFDM symbol
    OfdmSymbol(1, 1:CP_Length)   = CyclicPrefix.';
    % Place the IFFT portion next to the cyclic prefix
    OfdmSymbol(1, CP_Length+1:end) = IfftOutputBuffer.';
    % Place the OfdmSymbol into the output buffer
    OutputSequence(1,OutputSequenceIndex:OutputSequenceIndex + OfdmSymbolLength - 1) = OfdmSymbol;
    % Update the index into the output sequence
    OutputSequenceIndex = OutputSequenceIndex + OfdmSymbolLength;
end
```

The following MatLab code implements the OFDM demodulator for the FlexLink standard.

```
function ResourceGrid = OfdmDemodulator(InputSequence, StartSample, bLteBw)

% Note that the sample rate is 20MHz (there is no choice as there was in the modulator)
% Therefore,
SampleRate        = 20e6;   % 20MHz
FFT_Size          = 1024;   % IFFT length (1024/20MHz) = 51.2 microseconds
CP_Length         = 116;    % CP length (116/20MHz)    =  5.8 microseconds
OfdmSymbolLength  = CP_Length + FFT_Size;
SubcarrierSpacing = SampleRate / FFT_Size;

% An OFDM symbol is the concatenation of the CP and the IFFT portion
% The StartSample input argument is the estimated position of the first sample
% of the Ifft portion of the first valid OFDM symbol in the FlexLink Packet.
% The start time is found via the correlation against PreambleB.
% We will start 1 microsecond inside the CP to avoid intersymbol
% interference due to channel precusors.
OneMicrosecondsInSamples = 20;
assert(StartSample > OneMicrosecondsInSamples, "Improper input sequence.");

% Determine the number of available OFDM symbols we can demodulator
NumAvailableOfdmSymbols = floor( (length(InputSequence))/OfdmSymbolLength);

if(bLteBw == true)          % LTE BW
    NumSubcarriers        = 913;
    PosSubcarrierIndices   = 456:912;    % Indices in resource grid
    NegSubcarrierIndices   = 0:455;      % Indices in resoruce grid
    PosFftIndices          = 0:456;      % Indices in FFT input buffer
    NegFftIndices          = 568:1023;   % Indices in FFT input buffer
else                        % WLAN BW
    NumSubcarriers        = 841;
    PosSubcarrierIndices   = 420:840;    % Indices in resource grid
    NegSubcarrierIndices   = 0:419;      % Indices in resoruce grid
    PosFftIndices          = 0:420;      % Indices in FFT input buffer
    NegFftIndices          = 604:1023;   % Indices in FFT input buffer
end

% Phase compensation for 1 microsecond advance
TonesIEEE      = -(NumSubcarriers - 1)/2 : 1 : (NumSubcarriers - 1)/2;
OneMicroSecond = OneMicrosecondsInSamples / SampleRate;   % = 1e-6
Compensation   = exp(1j*2*pi*OneMicroSecond*TonesIEEE*SubcarrierSpacing).';

% Start the OFDM Demodulation Process (l is the 0 based OFDM symbol index)
ResourceGrid   = zeros(NumSubcarriers, NumAvailableOfdmSymbols);
StartIndex     = StartSample - OneMicrosecondsInSamples;
Range          = StartIndex : StartIndex + FFT_Size - 1;
for l = 0:NumAvailableOfdmSymbols - 1
    FftInputBuffer   = InputSequence(1, Range + 1).';
    FftOutputBuffer  = fft(FftInputBuffer);

    % Place the compensated FftOutput into the resource grid
    ResourceGrid(PosSubcarrierIndices + 1, l + 1) = FftOutputBuffer(PosFftIndices + 1, 1);
    ResourceGrid(NegSubcarrierIndices + 1, l + 1) = FftOutputBuffer(NegFftIndices + 1, 1);
    ResourceGrid(:, l+1)                          = ResourceGrid(:, l+1) .* Compensation;

    Range                                         = Range + OfdmSymbolLength;
end
```

8.9.4.4 The Signal Field

The signal field defines the amount of information transmitted in *payload A* as well as the manner, in which this information is error-encoded and mapped into QAM constellations. In turn, *payload A* will carry user data as well as information that configures transmissions in *payload B*. Two payloads exist in order to better enable different use-cases. For the drone application, the ground station may not be able to receive a video stream anymore, but the telemetry information exchange should stay in takt. Therefore, *payload A* can feature far better error correction than *payload B* if so desired. The information in the signal field is provided by the media access controller, or *MAC*.

Format 1

Format 1 of the signal field is the simplest way, in which *payload A* can be configured. Each transport block is broken up into a certain number, *NDB A*, of data blocks, which become code blocks and code words once the FEC encoding and rate matching is added respectively. See the section on payload *A* and *B* for a definition of the various types of data units that are processed in the bit encoding chain.

Table 8-27: Signal Field Description Format 1

Information	Number of Bits	Description
Reserved	1	For future expansion
CBS_A_Flag	2	(0,1,2,3) Code block size (648, 1296, 1944 bits, 1944)
FEC_A_Flag	2	(0, 1, 2, 3) LDPC coding rates ½, 2/3, ¾, 5/6
Reserved	1	For future expansion
NDB_A	14	The number of data blocks to be sent in payload A.
RM_A_Flag	3	This flag indicates the rate matching factor C2. (0, 1, 2, 3, 4, 5, 6, 7) C2 = 0.0, 0.5, 0.75, 1, 3, 7, 15, 31
BPS_A_Flag	2	Bits Per Symbol (QAM) 0, 1, 2, 3 = BPSK, QPSK, 16QAM, 64QAM
Num OFDM Symbols	14	The number of OFDM symbols in the packet, where the first reference symbol is the first OFDM symbol in the packet.
TX Reference Clock Count	14	This value is the remainder of the integer division of the reference clock count and 2^{14}, at the moment the transmit state machine orders the read out of the preamble.
Client_Flag	1	0/1 – This transmission is from a service / client terminal
CRC	10	Cyclic Redundancy check bits that protect the 54 signal field bits
	64	

The format 1 signal field is primarily used in situations where the terminals are in rapid motion or when the overall signal to noise ratio of the link is poor. The bit stream of the format 1 signal field must either be encoded using a ½ rate binary convolutional code or a ¼ rate N = 256 Polar code and the encoded bits repeated, or rate matched, until they fill the number of OFDM symbols indicated in the control information. See section '*Error Detection and Correction*' for more details.

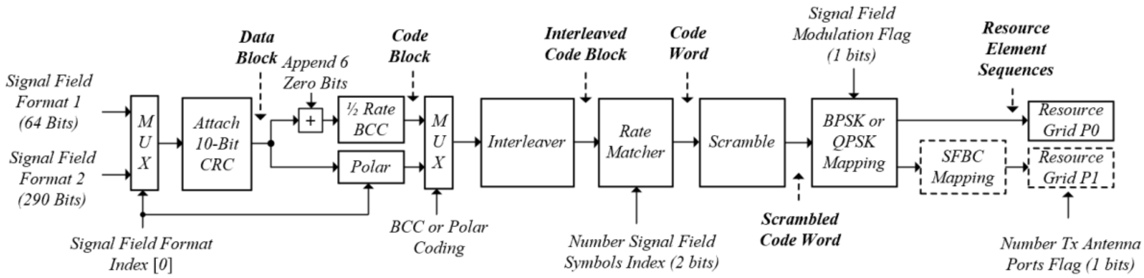


Figure 8-93: Encoding Process for Signal Fields Format 1 and 2

Format 2

Format 2 of the signal field follows the same pattern as format 1 except that it provides a custom QAM constellation for each resource block. This information is usually provided if the transmitter knows the frequency response of the channel, which must have been provided by the receiver in an earlier packet. Terminals need to be almost stationary for this use case.

Table 8-28: Signal Field Description Format 2

Information	Number of Bits	Description
Reserved	1	For future expansion
CBS_A_Flag	2	(0,1,2,3) Code block size (648, 1296, 1944 bits, 1944) if LDPC
FEC_A_Flag	2	(0, 1, 2, 3) LDCP ½, 2/3, ¾, 5/6
Reserved	1	For future expansion
NDB_A	14	The number of data blocks to be sent in the payload.
RM_A_Flag	3	The amount of rate matching (0, 1, 2, 3, 4, 5, 6, 7) C2 = 0.0, 0.5, 0.75, 1, 3, 7, 15, 31
BPS_A_Flag	76·3 = 228	The number of bits per QAM symbol for each resource blocks: 0, 1, 2, 3, 4, 5, 6, 7 = 0bits, 1bit, 2bits, 4bits, 6bits, 8bits, 10bits, 10bits
Num OFDM Symbols	14	The number of OFDM symbols in the packet, where the first reference symbol is the first OFDM symbol in the packet.
TX Reference Clock Count	14	This value is the remainder of the integer division of the reference clock count and 2^{14}, at the moment the transmit state machine orders the read out of the preamble.
Client_Flag	1	0/1 – This transmission is from a service / client terminal
CRC	10	Cyclic Redundancy check bits that protect the 280 signal field bits
	290	

The bit stream of the format 2 signal field must either be encoded using a ½ rate binary convolutional code, or a N = 1024 Polar code, and the encoded bits repeated, or rate matched, until they fill the number of OFDM symbols indicated in the control information. The vector BPS_A_Flag (of length 228 bits) indicates the QAM constellation for the data resource element values of each of the 70 (WLAN BW) or 76 (LTE BW) resource blocks. For example, if the first three bits of the vector are 011 = 3, then all data QAM values in resource block 0 will be 16QAM modulated. For the WLAN bandwidth case, the last 18 bits of the vector are undefined.

8.9.4.5 Synchronization Strategies

Synchronization is a collective term used to describe the activities in the receiver that facilitate the proper demodulation of the signal. Therefore, synchronization and demodulation are separate processes, with the former happening first. When developing synchronization sequences for the FlexLink standard, the overriding constraints were that FlexLink should achieve synchronization at very low signal to noise ratio (close to -8dB) and work using inexpensive reference oscillators with reasonable stability. When using less expensive reference crystal oscillators whose stability is larger than 1ppm, the receiver will see very significant frequency offsets embedded in the received signal as well as timing drift, which is generally less of a problem.

As an example, assume that reference crystal oscillators with a stability of 10ppm (part per million) at the transmitter and receiver are used to establish a link at 2.4HGz. The frequency error seen at the receiver may be as bad as 2.4e9Hz · 2 · 10e-6 = 48KHz. Not correcting this error would make our OFDM demodulator, which works with 19.53125KHz subcarrier spacing, completely deaf. We need to detect and correct this error even at poor SNR. The following activities are part of the synchronization process.

Automatic Gain Control (AGC)

AGC is the process of detecting the average magnitude of the incoming signal and readjusting its size such that clipping is minimized at the analog to digital converters. FlexLink provides an AGC burst at the beginning of the preamble to facilitate this process. The AGC burst is a Zadoff-Chu sequence that is spread across the entire bandwidth of the signal. The low peak to average power ratio of the Zadoff-Chu sequence, and the fact that it is a wide-band signal, makes it the perfect sequence for fast and accurate averaging of the magnitude.

Packet Detection

Some mechanism must exist to determine that a FlexLink packet has arrived as compared to some other type of signal radiating within the RF channel. The packet detector described in 8.9.4.2 can detect *Preamble A* at signal to noise ratios close to -10dB and at frequency offsets far exceeding those of the example above. *Preamble A* may feature a length of either 50 or 250 microseconds. The first 50 microseconds are sufficient for task of packet detection.

Frequency Offset Acquisition

Whereas the packet detector can indicate the arrival of the packet and do so at very large frequency offsets, we need a mechanism to compute the frequency offset and then correct for it. Latent offset after frequency offset correction will cause loss of orthogonality in the OFDM demodulator, which will make the signal to noise ratio at the FFT output worse. The frequency detector will be able to determine the frequency offset to enough accuracy to avoid the effects of loss of orthogonality. In order to compute the frequency offset, *Preamble A* must feature a length of 250 microseconds. During normal operation, the long version of *Preamble A* will be used when the two terminals make initial contact and occasionally for resynchronization.

Timing Acquisition

Timing acquisition is the process of determining the start of the OFDM symbols and therefore which sections of the received IQ sequence should be read into the FFT buffer of the OFDM demodulator. The receiver will use the *Preamble B*, which is a Zadoff-Chu sequence with very good auto-correlation performance. It is the sequences auto-correlation properties, not its low peak to average power ratio, that are used for this purpose.

Clocking Strategy

This specification requires a uniform clocking strategy, which features a single reference oscillator (usually a TXCO) that provides the same reference clock to the base band processor, the mixed signal converters (ADCs / DACs), and to the synthesizer, which generates the local oscillator, LO, signals that up and down convert the IQ information between baseband and RF frequencies. This is done in order to simplify synchronization as frequency offset and timing drift will now track one another (see 7.3.2).

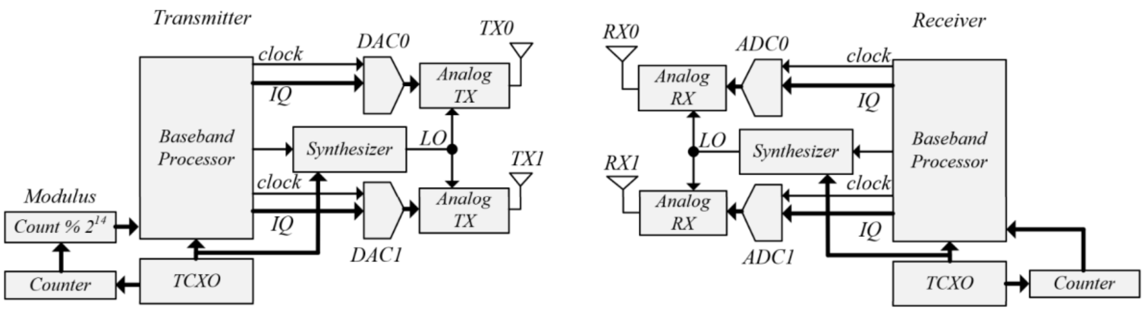


Figure 8-94: Unified Clocking Strategy

Timing Drift and Frequency Offset Tracking

As mentioned in section 7.3.2, the unified clocking strategy links the timing drift of the IQ signal to the frequency offset. We can deduce one by knowing the other. As the frequency of the 20MHz reference clock can drift with temperature, it is important to be able to reacquire the frequency offset from time to time. We could occasionally transmit the long version of *preamble A*, but we may also use a different scheme that is far more accurate.

Note that the signal field contains a ‘Reference Clock Count’. This reference clock count is equal to the value of the transmit modulo counter (see figure above) at the moment when the transmitter’s state machine begins to read out the preamble and passes it to the digital to analog converters. In the figure above, we can see the counter and modulo operation providing the baseband processor with a *mod*(Count, 2^{14}) value that changes with every clock. The transmitter provides this value in the signal field and once two packets have arrived at the receiver, the receiver can compute the difference in the TX clock counts for the two packets. The receiver keeps a count of its own reference clock as well and assigns a reference count at the time that peaks are detected during the *preamble B* processing of the two packets. The ratio of the transmit and receive count differences is equivalent to the difference in time base between the transmitter and receiver. With this information at hand, processing of additional long *preamble A* sequences is no longer necessary, as we may deduce frequency offset and timing drift from the time base difference between the TX and RX reference oscillators.

Synchronization using Reference Signal

In the case of very low signal to noise ratios, when normal synchronization mechanisms are no longer applicable, synchronization can also be achieved by analyzing the references signals over an extended period of time to produce a channel impulse response estimate. Using this method, synchronization and data transfer are possible at signal to noise ratios below -10dB. This technique only works after initial synchronization has already been established.

8.9.4.6 Payload A and B

FlexLink provides two different entities that will hold the bulk of the data to be transmitted. The entities, called payload *A* and *B*, are split for the following reasons:

→ Payload A is the most basic method of transmitting bulk data and is the preferred method to communicate with moving FlexLink terminals.

→ Payload B is used to optimize communication, for with multiple FlexLink terminals simultaneously.

The data sent in each payload consists of different bit segments, which, as they are encoded, assume different names as can be seen in the figure below. However, all payload data starts as a transport block which is constructed either in the MAC or passed to the MAC from a higher level in the protocol stack.

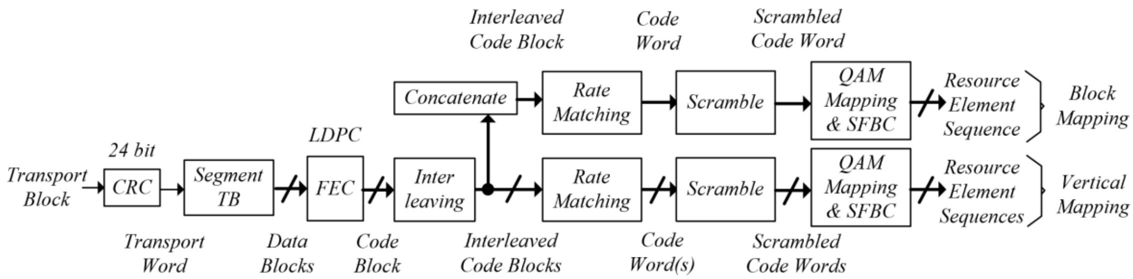


Figure 8-95: Encoding Chain for Payloads A and B

Transport Blocks and Words

A transport block is a set of MAC bits that is protected by a single 24-bit CRC, features a common error encoding method and is mapped in a sequential manner to a distinct area in the resource grid. A *transport word* is simply the concatenation of the transport block and its 24-bit CRC. The transport word size is thus 24 bits larger than the transport block size. Neither *payload A* nor *payload B* may carry more than 256 transport words,

$$TWS = TBS + 24$$

Data Blocks and Code Blocks

The transport word needs to be segmented into between 1 and 256 different data blocks, which have the correct size for compliant FEC encoding. Once the error correcting bits are added, a data block becomes a code block. In the table below, we can see that there are 12 different data block sizes, DBS, but only three possible code block sizes, CBS, which are 648, 1296, and 1944 bits.

$$CBS = DBS \cdot CodingRate$$

Table 8-29: Data Block Sizes vs Code Block Sizes for LDPC ½, 2/3, ¾, 5/6 Rate Encoding

DBS (data block size)	CBS (code block size)
324 / 648 / 972	LDPC ½ → [648 / 1296 / 1944]
432 / 864 / 1296	LDPC 2/3 → [648 / 1296 / 1944]
486 / 972 / 1458	LDPC ¾ → [648 / 1296 / 1944]
540 / 1080 / 1620	LDPC 5/6 → [648 / 1296 / 1944]

Interleaved and Rate Matching for Vertical Mapping

As shown in the previous figure, a code block is interleaved, as detailed in section 8.9.3.9, with no change in block size. This process is undertaken to avoid bursts of errors entering the FEC decoder. Furthermore, to decode signals with low signal to noise ratio, the MAC can request that the bits in the interleaved code blocks be repeated a certain number of times to form the *code word*. The receiver can then sum up the copies of the received soft bits before executing the FEC decoder. The code word size, CWS, is computed as follows.

$$CWS = CBS + NumRepeatedBits$$

$$NumRepeatedBits = CBS \cdot RateMatchingFactor + NumFillerBits$$

The rate matching factor is chosen by the MAC, which also makes it available in the signal field during transmission. This factor is a constant for all code blocks in a particular transport block. The number of filler bits is such that the code word will fill an integer number of resource blocks. In this way, each successive code word will always start at the beginning of a resource block.

There exists a second rate matching method, in which code blocks are first concatenated before repeating the bits that they contain. In that case, we have a single (scrambled) code word (see previous figure) for each transport block. In the expression below, we use the term filler bits as the number of repeated bits such that they fill the entire target area in the resource grid.

$$CWS = CBS \cdot NumDataBlocks + NumFillerBits$$

Scrambling

Scrambling is used in the FlexLink specification to ensure that no clear pattern exists when QAM symbols are placed into the reference grid. Any clearly defined QAM symbol pattern, due to long runs of 0 or 1 bits, will cause excessive peaking in the OFDM signal.

Resource Element Sequence

A resource element sequence is a sequence of QAM symbols holding the scrambled code word. The number of elements in a resource element sequence must be such that an integer number of resource blocks can be filled completely. The resource element sequence is then either encoded via SFBC, space frequency block coding, and then passed to the two resource grids for OFDM modulation and transmission via antenna port P0 and P1, or it is directly passed to the resource grid for P0 only.

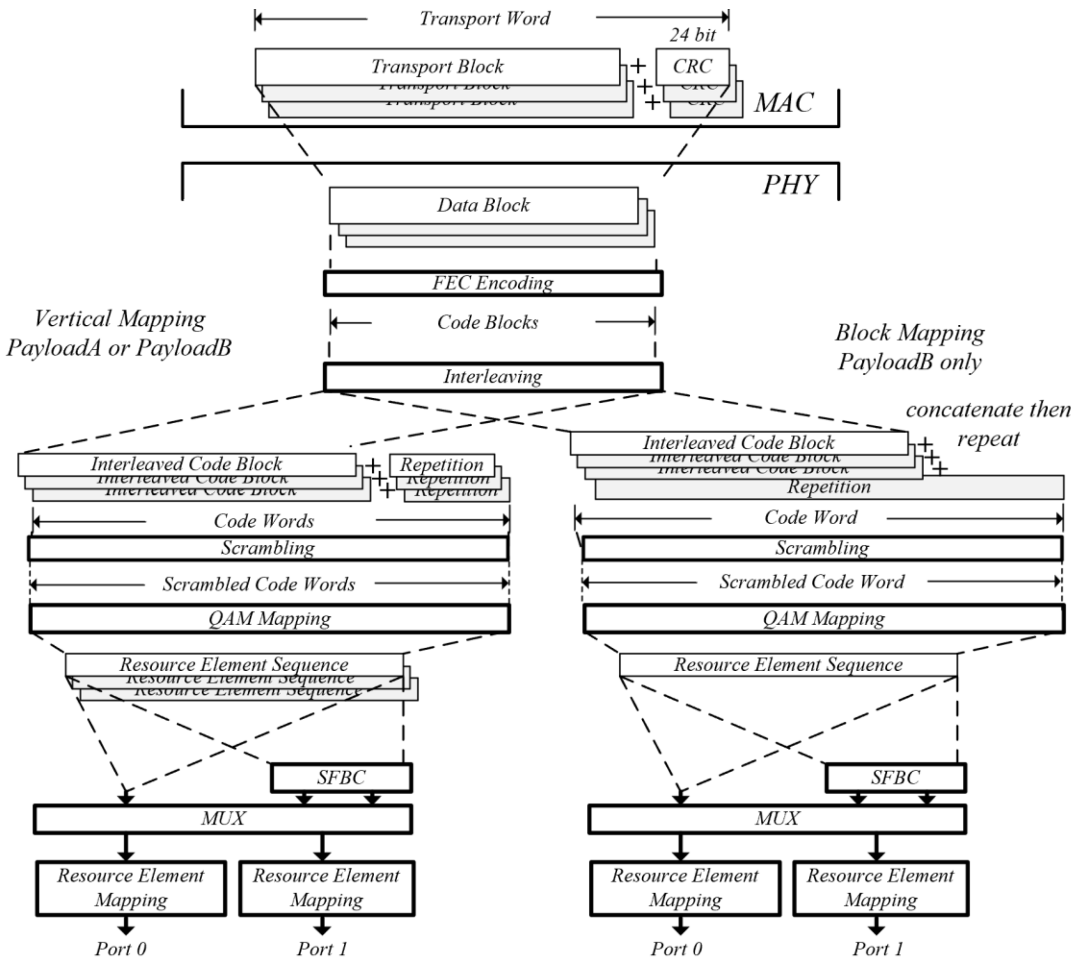


Figure 8-96: Processing Chain for Data Bits and Resource Elements

Vertical vs Block Mapping

There are two types of resource element mapping available. *Vertical mapping* will map all resource element sequences belonging to a transport block in a vertical fashion starting at subcarrier $k = 0$ to $k = 912$ or 840, depending on the channel bandwidth (LTE / WLAN). *Vertical mapping* is the only mapping method available for payload *A*. *Block mapping* will map all resource elements belonging to a single transport block into a rectangular region in the resource grid. Both vertical and block mapping are available in *payload B*. In the figure below, both mapping techniques are shown. The left image shows vertical mapping in *payload B*, whereas the right image shows block mapping in *payload B*. Note that the resource element sequences 0 and 1 for *payload A* both cover a full OFDM symbol plus one resource block. This is to be expected given that they represent two scrambled code words with identical number of bits.

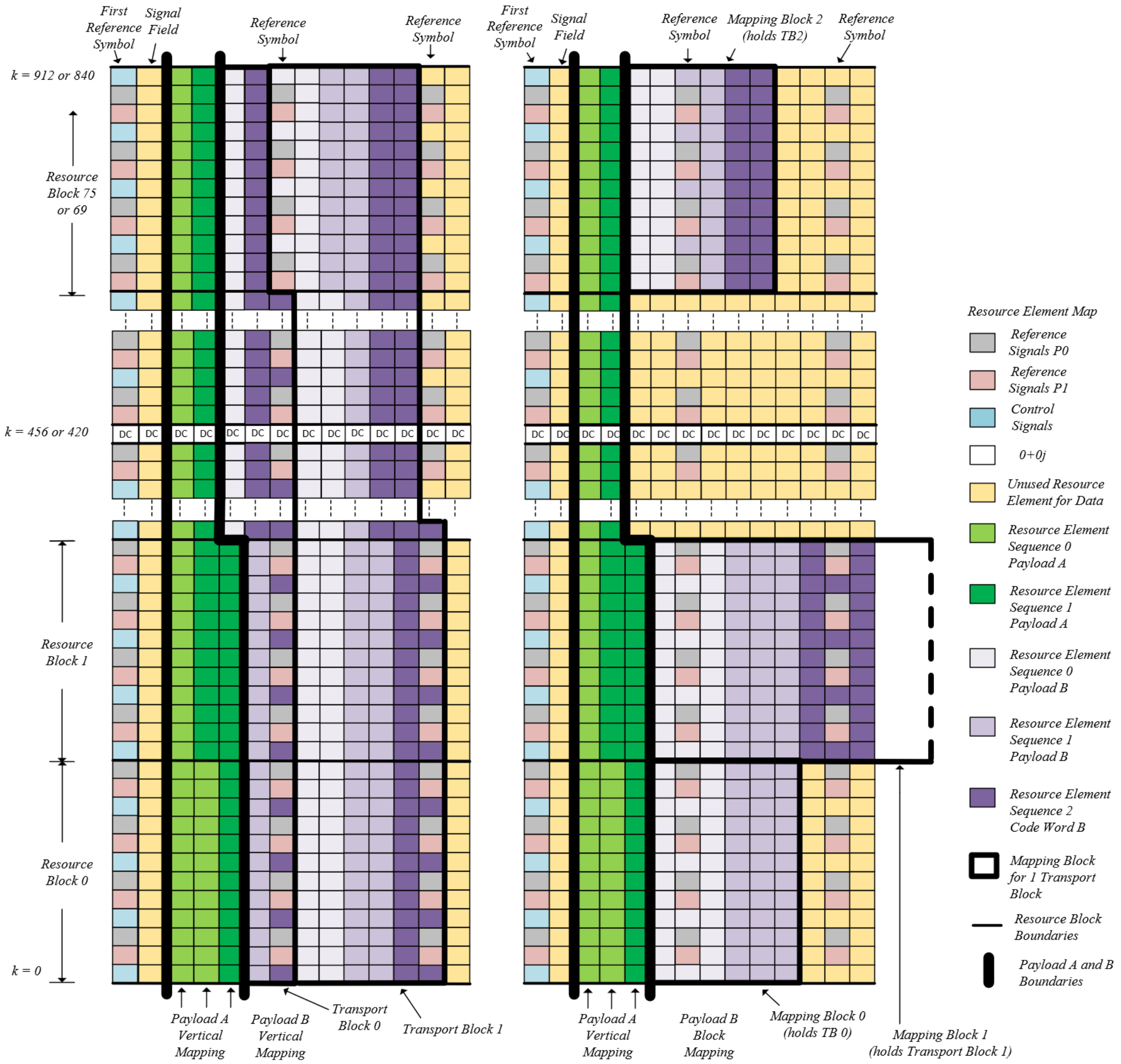


Figure 8-97: Vertical and Block Mapping in Payload A and B

Block mapping is used when the transmitter knows the frequency response that the receiver will see, a fact that is only possible if both stations are relatively stationary thus giving the receiver time to convey the frequency response back to the transmitter in a timely manner. With this information, the transmitter can properly encode and place information belonging to each transport block thus optimizing the overall throughput. The MAC may even boost the power of certain blocks if it chooses to forgo transmission in other portions of the resource grid.

Static and Adaptive Vertical Mapping in Payload A

Vertical mapping may be subdivided into *static* and *adaptive* variants, which are configured via the signal field formats 1 and format 2 respectively. The *static* variant is the default method discussed in the last paragraph, and all transport blocks will be processed with the same coding rate, rate matching and QAM constellations. As mentioned earlier, if the terminals are not moving quickly and the frequency responses remain stable, the receiver will estimate the CINR for every resource block and convey this information back to the transmitter. This will allow the transmitter to *adaptively* select the proper QAM constellation when mapping data into each resource block. Those resource blocks with poor CINR will use a more robust mapping such as BPSK or QPSK, whereas those with good CINR may use a higher constellation map. The MAC may even decide to forgo IQ mapping entirely in certain resource blocks, which allows it to boost power for the remaining resource block. The MAC may even change the bandwidth of the signal if IQ information is not mapped into resource blocks at the extremes of the band.

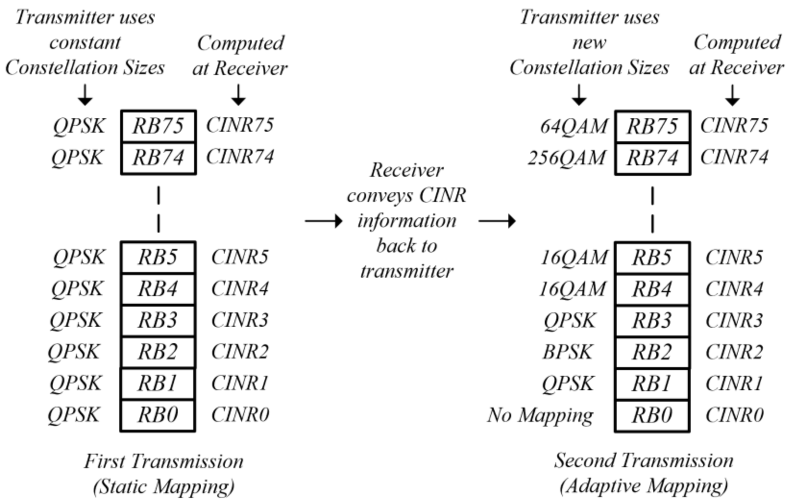


Figure 8-98: Static versus Adaptive Mapping

Mapping in Payload B

Payload B may use either vertical mapping (both static and adaptive) or block mapping. As compared to *payload A*, each transport block may be assigned a unique coding rate and rate matching configuration in vertical mapping. However, if the channel conditions are known, block mapping is superior as it provides more flexibility to optimally place, size (in terms of power), encode, rate match and QAM map the transport blocks. The mapping details for *payload B* will be specified by information element groups provided at the start of *payload A*.

8.9.4.7 Error Detection and Correction

Error Detection

There are two CRC generator polynomials that are used in FlexLink. They are of order 24, and 10 and obey the expressions listed below. See section 6.6.1 for a description of how to generate the cyclic redundancy check for these polynomials. The order 24 CRC will protect each transport block, whereas the order 10 CRC will be used to detect errors in the signal fields.

→ Generator 24 = [1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1]
→ Generator 10 = [1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1]

Error Correction

FlexLink defines three different methods of error correction as seen in the table below.

Table 8-30: Error Correction Techniques

Data Entity	Method	References
Signal Field (Required)	½ Rate BCC	IEEE 802.11-2012 Section 18.3.5.6
Signal Field (Optional)	Polar Coding	See Description in this Section
PayloadA and PayloadB (Required)	LDPC	IEEE 802.11-2012 Annex F

Required Method – ½ Rate BCC

The required error correction methods follow techniques that have been well established in the IEEE 802.11 technology family (see IEEE 802.11-2012 Sections 18.3.5.6 and Annex F). FlexLink will use ½ rate binary convolutional coding, or BCC, to protect the signal field, whereas the two payloads will use LDPC coding to correct errors. The reason for this choice is the readily available IP for these correction schemes.

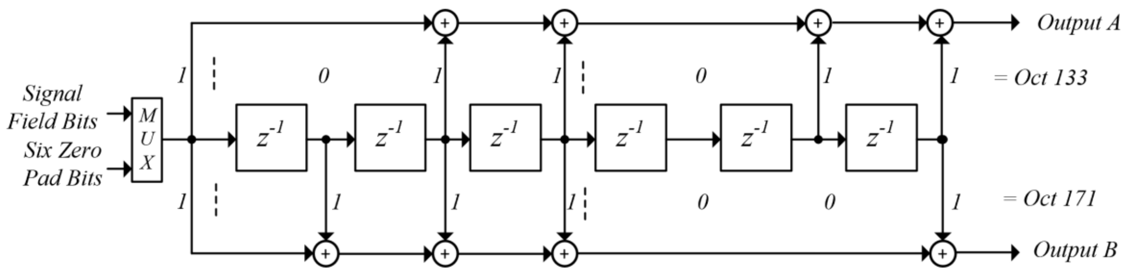


Figure 8-99: ½ Rate Convolutional Encoder with Constraint Length = 7

The ½ rate BCC uses generator polynomials *oct*133 and *oct*171 to produce output bits *A* and *B*, where bit *A* is inserted into the output stream before bit *B*. The signal field features either 64 or 290 bits. Six zero pad bits need to be appended at the end of the signal fields prior to the encoding process to return the encoder to the zero state. Therefore, a total of 64+6=70 and 290+6=296 bits are BCC encoded for signal field format 1 and 2 respective. This provides the Viterbi decoder with the information needed to start the trace back process (see 6.6.5).

<u>Required Method – LDPC</u>

Both *payload A* and *payload B* are protected using low density parity check, or LDPC, codes with coding rates of ½, 2/3, ¾, and 5/6. The LDPC matrix definitions can be found in IEEE 802.11-2012 Annex F, and an example of the matrix for an output length of n = 648 and a code rate of ½ is provided in 6.6.10.2. The specification allows for three different code block sizes. It should be noted that LDPC codes perform better the larger their input sizes are. It is therefore advisable to use the 1944 code block size whenever possible.

Table 8-31: Data Block Sizes vs Code Block Sizes for LDPC ½, 2/3, ¾, 5/6 Rate Encoding

Code Rate	Data Block Sizes	Code Block Sizes
½	**324 / 648 / 972**	[648 / 1296 / 1944]
2/3	**432 / 864 / 1296**	[648 / 1296 / 1944]
¾	**486 / 972 / 1458**	[648 / 1296 / 1944]
5/6	**540 / 1080 / 1620**	[648 / 1296 / 1944]

<u>Optional Method – Polar Code</u>

It is well known that binary convolutional codes are inferior to any of the more modern codes like LDPC, Turbo, and Polar codes. However, for short input bit lengths, the advantage of these newer techniques is not quite as evident as would be the case for large input bit lengths. Of the newer codes, the Polar codes have shown the best performance for shorter bit lengths and will clearly be superior to the BCC code when processing signal field format 2, which features 290 bits. For this reason, two polar codes have been introduced as an optional feature to protect the signal field. The two polar codes are defined as follows.

Table 8-32: Polar Code Configuration for Signal Field Formats 1 and 2

Signal Field Format	Input Size (Bits)	Output Size Bits	Erasure Probability
1	$K = 64$	$N = 256$	$p = 0.55$
2	$K = 290$	$N = 1024$	$p = 0.55$

In polar codes, erasure probabilities need to be calculated for each input bit position of the polar encoder (see section 6.6.11.1). Use the function *FindChannelProbabilities*(N, p) to determine the reliability of each input bit position and order them from lowest erasure probability (most reliable bits) to highest erasure probability (least reliable bits). Place the signal field bits starting with bit 0 at the position with the lowest erasure probability and bits 63 or 289 at the bit position with the 64^{th} or 290^{th} lowest erasure probabilities respectively. The remaining inputs are considered frozen at 0. Section 6.6.11 explains the process of polar coding used with FlexLink.

Important: A receiver that can use polar decoding, should run both BCC and polar decoding simultaneously to determine which encoding was actual used in the transmit signal. The type of encoding is not provided in the control information but can be provided in the MAC information elements.

8.9.4.8 Interleaving and Rate Matching

Interleaving

Most forward error correction algorithms have more difficulty correcting bit errors that appear in bursts (quick succession), than bit errors that are randomly distributed throughout the received bit sequence. During symbol mapping, a set of sequential transmit bits are mapped into single QAM symbols, which in turn are placed into resource elements at consecutive subcarrier positions. As the frequency response of the channel may feature nulls, it is likely that individual subcarriers or a small group of adjacent subcarriers will feature very poor SNR at these regions. Some or all the bits residing inside the QAM symbols at these subcarriers may be in error. Interleaving is a process by which the position of bits in the encoded bit sequence, the code word, is scrambled before being mapped into QAM symbols and then into consecutive resource elements. At the receiver, the recovered code word, which is demapped from QAM symbols that appear at sequential subcarriers, is first deinterleaved before being error decoded. This deinterleaving process scrambles the position of sequential, or burst like, errors into randomly distributed errors.

Given a vector of CWS error encoded data bits, our code word, we wish to map each bit into new positions of a second vector, which is the interleaved code word. Remember that the code block sizes are CBS = 648, 1296, and 1944 for LDPC coding. Consider the following vectors.

$$Code\ Block = [b_0, b_1, \dots, b_{CBS-1}]$$

The first step of the interleaving process is to place the bits of the code block into the columns of an index matrix as seen below. The index matrix shall feature R = 61 rows and C = $ceil(CBS/R)$ columns. The index matrix below is configured for a code block size of 648, yielding 11 columns.

$$IndexMatrix = \begin{bmatrix} 0 & 61 & 122 & \cdots & 549 & 610 \\ 1 & 62 & 123 & \cdots & 550 & 611 \\ 2 & 63 & 124 & \cdots & 551 & 612 \\ 3 & 64 & 125 & \cdots & 552 & 613 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 59 & 120 & 181 & \cdots & 608 & NA \\ 60 & 121 & 182 & \cdots & 609 & NA \end{bmatrix}$$

The second step of the interleaving process is to read out the bits along the rows of the matrix. The consecutive row numbers, out of which we will read the bits, are specified in the order vector. The first and last rows, out of which we will read the bits, are rows 0 and row 55.

$$Order = [\,[0,12,24,36,48,60], [6, 18, \dots, 54], [3,15, \dots, 51], [9, 21, \dots, 57], [1, 13, \dots, 49], [11, 23, \dots, 59\,],$$
$$[2, 14, \dots, 50], [10, 22, \dots, 58], [4, 16, \dots, 52], [8, 20, \dots, 56], [5, 17, \dots, 53], [7, 19, \dots, 55\,]]$$

As only 648 out of 61·11 = 671 positions in the matrix can be mapped, the remaining entries in the last column are marked *NA*. When reading the indices out of the matrix along the rows, these entries must be skipped.

As there is a limited number of code block lengths, there is no need to have the baseband processor go through this matrix reshuffling every time a code block must be interleaved to form an interleaved code block. We simply precompute the interleaved positions for the three available code block sizes. During the deinterleaving process, the same vector is used.

```
% ------------------------------------------------
% Index scrambling in the Interleaver
% ------------------------------------------------
% Constants
% ------------------------------------------------
CBS_LDPC  = [648, 1296, 1944, 1944]; % The possible code block sizes for LDPC  Coding

% ----------------
% Configuration of the FEC (In the Signal Field)
% ----------------
CBS_A_Flag = 1;
CBS        = CBS_LDPC(1, CBS_A_Flag + 1);

% Indices before Interleaving 0, 1, 2, 3, 4, ... CBS-1
NormalIndices = 0:1:CBS-1;
NumRows       = 61;
NumColumns    = ceil(CBS/NumRows);
NumElements   = NumRows * NumColumns;

% In the index matrix below, a -1 represents an NA value in the Interleaving matrix
IndexMatrix   = -ones(NumRows, NumColumns);

% Place the indices into the columns of the IndexMatrix
for Index = NormalIndices
    Row    = mod(Index, NumRows);
    Column = floor(Index / NumRows);
    IndexMatrix(Row + 1, Column + 1) = Index;
end

RowOrder = [0:12:60,  6:12:54, 3:12:51, 9:12:57, 1:12:49, 11:12:59, ...
            2:12:50, 10:12:58, 4:12:52, 8:12:56, 5:12:53,  7:12:55];

% Read out the indices given the RowOrder below
InterleavingIndexVector = zeros(1, CBS);
Count              = 0;
for RowIndex = RowOrder
    for ColumnIndex = 0:NumColumns - 1
        if(IndexMatrix(RowIndex + 1, ColumnIndex + 1) ~= -1)
            InterleavingIndexVector (1, Count + 1) = IndexMatrix(RowIndex + 1, ColumnIndex + 1);
            Count                                  = Count + 1;
        end
    end
end

disp([(0:CBS-1)', InterleavedIndices'])
```

Rate Matching

The rate matching step provides a different layer of diversity, which attempts to improve the signal to noise ratio of those bits that must be error corrected. By repeating bits at the transmitter, there will be several copies of the same bit available at the receiver. The log likelihood ratio soft bit values (the output of the QAM demapping step) associated with these copies must be summed prior to error correction. The summing process averages away some of the noise associated with each LLR soft bit yielding a superior LLR bit value. There are two types of rate matching available in FlexLink.

Rate Matching for Vertical Mapping

In vertical mapping, bits are converted to QAM symbols which are then mapped to subcarrier 0, 1, 2, 3, … 840 or 912, before moving back to subcarrier 0 of the following OFDM symbols. The number of rate matched bits to be mapped are determined in two steps.

$$Number\ Rate\ Matched\ Bits = CBS\ \cdot (1 + C_2)$$

The factor C_2 is fetched from the following vector given the index, the *RM_A_Flag*, found in the signal field.

$$Vector = [0, 0.5, 0.75, 1, 2, 7, 15, 31]$$

Values of *RM_A_Flag* = 0 and 5 indicate that none of the bits are repeated and that all bits are repeated seven times yielding a total of eight copies of each bit.

The second step requires the use of the capacity tables. Once the number of rate matched bits has been found, we determine how many resource blocks are required to hold these bits. As we will always use an integer number of resource blocks to hold the information, there will inevitably be additional bit positions (filler bits) available in the last resource block. Therefore, the number of rate matched bits is augmented to include these bits.

$$CWS = CBS\ \cdot (1 + C_2) +\ NumberOfFillerBits$$

Rate Matching for Block Mapping and the Signal Field

For the signal field and the block mapping technique employed in *payload B*, no rate matching factors are provided. For the case of the signal field, interleaved bits are repeated until all OFDM symbols specified for the signal field are filled. As for the block mapping case, interleaved code blocks are concatenated, before being repeated. If we assume that we have NDB interleaved code blocks each featuring 1944 bits, then rate matching is achieved by repeating the first bit of each code block, then the second bit of each code block and so on. If the rate matching instructions call out for a large number of repetitions, then the block mapping word is simply repeated.

We need some MatLab code.

8.9.4.9 Scrambler

The FlexLink transmitter uses scrambling in order to randomize the bit stream prior to being QAM mapped. OFDM modems prefer to see complex values in their IFFT input buffers that appear random. This is especially true when using BPSK or QPSK values as is the case for the control information in the first reference symbol, the signal field information and the reference signals. Having runs of 0 or 1 bits before QAM mapping can make the QAM values in the IFFT input buffers appear too regular causing large peaks to appear in the OFDM modulated output waveform. FlexLink will use a linear feedback shift register producing a pseudorandom bit stream that repeats every 255 bits. The polynomial expression and construction of the shift register is shown below.

$$f(x) = x^8 + x^6 + x^5 + x^4 + 1$$

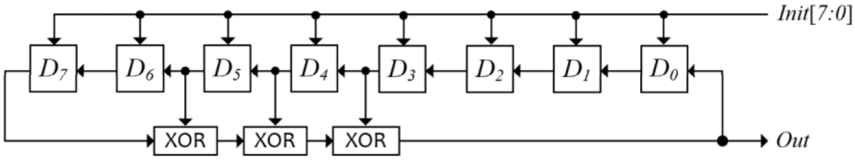


Figure 8-100: Length Eight Linear Feedback Shift Register

Unless otherwise indicated in the specification, the default initializer for the shift register is Init[7:0] = b1000 0000 producing a 255 bit scrambling sequence [1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, …1, 0, 0, 0, 0, 0, 0, 0], which will repeat indefinitely.

Using the Scrambling Sequence for Reference Signal Orientation

The reference signals for both antenna port 0 and 1 get their orientation from the basic scrambling sequence mentioned above. As is explained in more detail in section 8.9.5.1, we may compile the resource elements associated with either P0 or P1 reference signals into a list as the subcarrier index, *k*, increases from 0 to 840 / 912. For P0, the resource elements on this list, starting with the resource element with the lowest *k*, shall be BPSK modulated with the above scrambling sequence = [1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, …]. The orientation of reference elements for antenna ports 0 and 1 are identical.

Control Information Scrambling

Section 8.9.5.2 explains how control information is generated and ordered into a vector that is mapped into the control information opportunities that are reserved in the first OFDM symbol of the resource grid. Before BPSK modulating the control information and mapping it into the first reference symbol, it must first be scrambled according to the following simple expression.

$$ScrambledControlInfo[n] = mod(ControlInfo[n] + ScamblingSeqence, 2)$$

8.9.4.10 QAM Mapping

The QAM mapping technique used in FlexLink is identical to that defined in the 802.11 Wifi specifications.

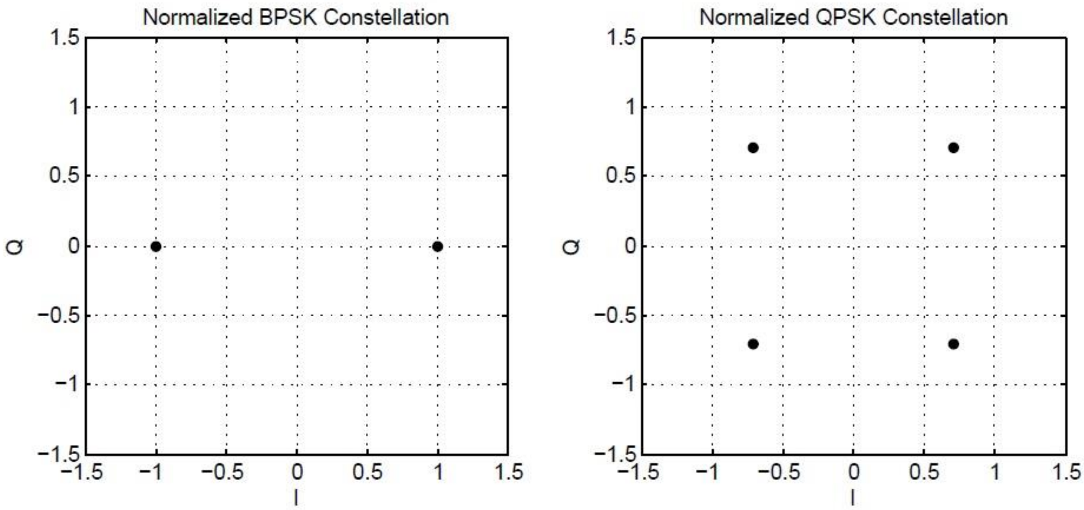


Figure 8-101: BPSK and QPSK Constellations

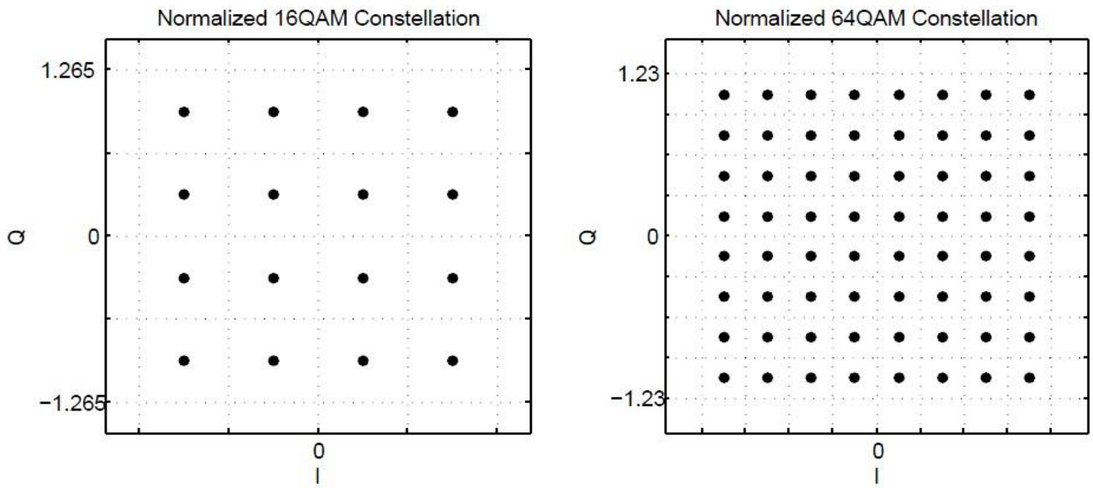


Figure 8-102: 16-QAM and 64-QAM Constellation

The following encoding tables map bits into the QAM constellations provided in the figures above. Note that the I-Out and Q-Out values for the QPSK, 16-QAM, and 64-QAM encoding tables need to be multiplied by the scalars $1/\sqrt{2}$, $1/\sqrt{10}$, and $1/\sqrt{42}$ respectively to normalize the variance of the constellations to unity.

Table 8-33: BPSK and QPSK Encoding Tables

BPSK		
Input Bit b_0	I-Out	Q-Out
0	-1	0
1	1	0

QPSK			
Input Bit b_0	I-Out	Input Bit b_1	Q-Out
0	-1	0	-1
1	1	1	1

Table 8-34: 16-QAM and 64-QAM Encoding Tables

16-QAM			
Input Bit B_0b_1	I-Out	Input Bit b_2b_3	Q-Out
00	-3	00	-3
01	-1	01	-1
11	1	11	1
10	3	10	3

64-QAM			
Input Bit $b_0b_1b_2$	I-Out	Input Bit $b_3b_4b_5$	Q-Out
000	-7	000	-7
001	-5	001	-5
011	-3	011	-3
010	-1	010	-1
110	1	110	1
111	3	111	3
101	5	101	5
100	7	100	7

8.9.5 Preamble Construction

The following section will discuss the requirements and design of the different preamble components. The preamble is only transmitted from antenna port 0, as FlexLink supports a single TX antenna configuration. In addition, when both transmit antenna ports are active and have a clear line of sight to the RX terminal, then it is possible that sending identical signals from both antenna ports would destructively interfere at the receiver antennas. To avoid this situation, port 1 remains inactive during preamble transmission. During OFDM symbol transmission, the space frequency block coding ensures that the two transmit antennas don't send identical information.

8.9.5.1 Construction and Processing of the AGC Burst

The purpose of the AGC burst is to present the receiver with a waveform that allows for fast detection of the signal magnitude. As such, the waveform should be wideband and feature a low peak to average power ratio. The AgcBurst shall occupy 5 microseconds, which at 20MHz yields 100 samples. The AgcBurst shall be a Zadoff-Chu sequence defined as a time domain sequence that is generated via the mathematical steps shown below. The parameters of the Zadoff-Chu sequence were chosen for the following reasons.

→ The length N_{zc} = 887 should be provide about as much bandwidth as the WLAN BW case.

→ The root index u = 54 is not critical for the AgcBurst. However, this Zadoff-Chu sequence is also used in the definition of the *preamble B*, which is detected using sliding correlation. The root index of 54 features particularly small secondary autocorrelation peaks.

The derivation of the AgcBurst begins with the expression of a Zadoff-Chu sequence. The length of the sequence shall be N_{zc} = 887, the root index u = 54 and the discrete time index spans n = 0, 1, … N_{zc}-1.

$$\boxed{zc[n] = \exp\left(-j\frac{\pi u n(n+1)}{N_{ZC}}\right)}$$

We now take the N_{zc} = 887-point discrete Fourier transform.

$$FFT\_Output[m] = \frac{1}{N_{ZC}}\sum_{n=0}^{N_{ZC}-1} zc[n]\cdot e^{-j2\pi nm/N_{ZC}}$$

By definition, we will have a certain number of positive frequency subcarriers, m = 0, 1, …*ScPositive*-1 and a certain number of negative frequency subcarriers, m = N_{zc} - *ScNegative*, … N_{zc}-1.

$$ScPositive = ceil\left(\frac{N_{zc}}{2}\right) = 444$$

$$ScNegative = floor\left(\frac{N_{zc}}{2}\right) = 443$$

The FFT output shall be mapped into a length 1024 or 2048 IFFT input buffer depending on the desired output sample rate.

$$IFFT\_Input[1{:}ScPositive] = FFT\_Output[1{:}ScPositive]$$
$$IFFT\_Input[1024 - ScNegative{:}1023] = FFT\_Output[N_{zc} - ScNegative{:}N_{zc} - 1]$$

or

$$IFFT\_Input[2048 - ScNegative: 2048] = FFT\_Output[N_{zc} - ScNegative: N_{zc} - 1]$$

The DC tone, *IFFT_Input*[0], remains unmapped as it will not survive the zero-IF down conversion process. Execute the IFFT and retain the first 100 (20MHz / 1024 IFFT scenario) or 200 (40MHz / 2048 IFFT scenario) samples. The following MatLab code computes the AGC burst for 20MHz and 40MHz sample rates. The peak to average power ratio is a low 3dB, and the waveform features a bandwidth of 887·SubcarrierSpacing = 887 ·20MHz/1024 = 17.325MHz.

$$AgcBurst_{1024} = \frac{1024}{887}\ IFFT\ (IFFT\_INPUT)\ [0: 99]$$

or

$$AgcBurst_{2048} = \frac{2048}{887}\ IFFT\ (IFFT\_INPUT)\ [0: 199]$$

```
function AgcBurst = GenerateAgcBurst(SampleRate)

% Error checking
assert(SampleRate == 20e6 || SampleRate == 40e6, "Invalid Sample Rate");

% We want the AGC burst to last 5 microseconds. Thus compute the number of
% samples that we need to retain at the end of the calculation
AgcBurstLengthInSec     = 5e-6;      % = either 100 or 200 samples
NumberOfSamplesToRetain = floor(AgcBurstLengthInSec * SampleRate);

% Define the Zadoff-Chu Sequence
Nzc        = 887;           % Force broadband waveform
ScPositive = ceil(Nzc/2);   % The number of positive subcarriers including 0Hz
ScNegative = floor(Nzc/2);  % The number of negative subcarriers
u1         = 54;            % This could be a different number
n          = 0:Nzc-1;       % Discrete time indices

% Generate the Zadoff-Chu seqeuence
zc         = exp(-1j*pi*u1*n.*(n+1)/Nzc);

% We now take the DFT and IFFT in succession. I use the name 'DFT' as the
% sequence is 887 samples long rather than a convenient length required
% for FFT operation. The final waveform is a resampled version of zc
% and has the bandwidth (887 * Subcarrier Sapcing) we want.
AgcBurstDft = fft(zc);

% We now map the DFT output into the IFFT
if(SampleRate == 20e6); IFFT_InputBuffer = zeros(1, 1024);
else;                   IFFT_InputBuffer = zeros(1, 2048); end

IFFT_InputBuffer(1, 1:ScPositive)          = AgcBurstDft(1, 1:ScPositive);
IFFT_InputBuffer(1, end-ScNegative+1:end)  = AgcBurstDft(1, end-ScNegative+1:end);

% Null out the DC term as it won't survive the DC offset removal in the RF
% receiver. This step is not that important. It is just done for completeness.
IFFT_InputBuffer(1,1)                   = 0; % Null out the DC term

AgcBurstFull                            = ifft(IFFT_InputBuffer);

% Retain only those samples that we need
AgcBurst = (length(AgcBurstFull)/Nzc)*AgcBurstFull(1, 1:NumberOfSamplesToRetain);
```

8.9.5.2 Construction and Processing of Preamble A

Preamble A enables the following synchronization processes at the receiver.

→ *Preamble A* is used to detect the presence of a FlexLink packet.
→ *Preamble A* is used to determine the frequency offset in the packet.

<u>*Defining and Detecting Preamble A*</u>

To define the equation for *Preamble A*, let's restate the base sample rate of 20MHz and the FFT size of 1024 used during the OFDM modulation process which together yield a subcarrier spacing of 20MHz/1024 = 19.53125KHz. Subsequently, the equation of the *Preamble A* is as follows.

$$\begin{aligned} PreambleA(t) &= \cos\left(2\pi\cdot 32\frac{20e6}{1024}\cdot t+\frac{\pi}{4}\right)+\cos\left(2\pi\cdot 96\frac{20e6}{1024}\cdot t+\frac{3\pi}{4}\right) \\ &= \cos\left(2\pi\cdot 625000\cdot t+\frac{\pi}{4}\right)+\cos\left(2\pi\cdot 1875000\cdot t+\frac{3\pi}{4}\right) \end{aligned}$$

Preamble *A* is defined as the super position of two cosine waveforms at 32 and 96 times the subcarrier spacing yielding 625KHz and 1.875MHz respectively for the sample rate of 20MHz. The waveform is periodic with a period of 1/625KHz = 1.6 microseconds, or 32 samples. Compared to the single sided bandwidth of the overall waveform, which is just shy of 10MHz, these tones feature relatively low frequencies. This is done to apply low pass filtering and significantly decrease noise present in the channel. The noise reduction and thus increase of signal to noise ratio is approximately equal to 10e6Hz/2.5e6Hz = 4, or 6dB.

The structure of the packet detector seen below is almost identical to what was used in the 802.11A modem discussed in section 8.3.2. The autocorrelation $x[n]\cdot conj(x[n\text{-}32])$, or $x[n]\cdot x[n\text{-}32]^*$, will average to a constant, sizable real value, as $x[n]$ and $x[n\text{-}32]$ are equal during the preamble. When *Preamble A* is not present, $x[n]$ and $x[n\text{-}32]$ are unrelated and $x[n]\cdot x[n\text{-}32]^*$ will average toward zero. The variance is computed in the lower branch of the structure in order to normalize the auto correlation to a maximum value of 1.0. The main differences between this version and the one used in the 802.11A example are the addition of the low pass filtering process and the potentially much larger averaging length *N* of 512, which enables us to detect packets with signal to noise ratios down to -10dB. When SNR conditions are known to be reasonably good, then *N* may be reduced to increase the speed of the packet acquisition.

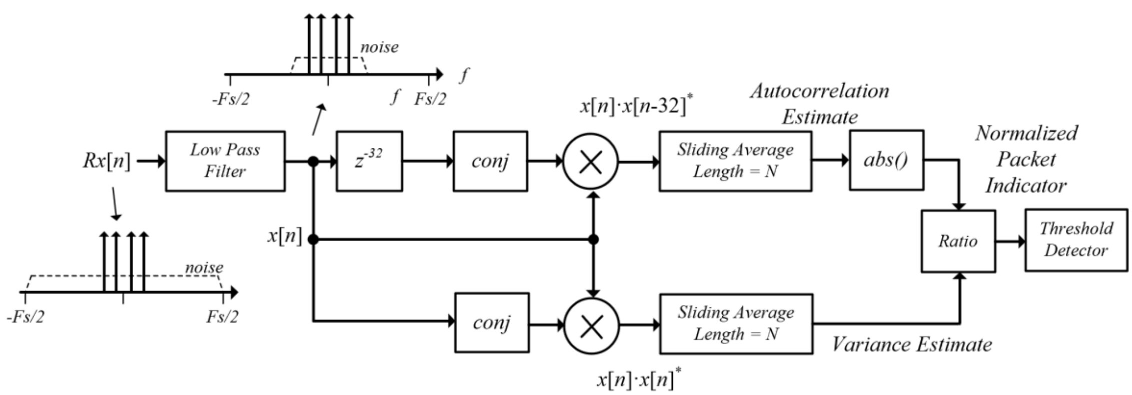


Figure 8-103: FlexLink Packet Detector

The figure below illustrates the situation where the signal to noise ratio is 0dB and the averaging length is N = 512. The sequence being passed through the detector consists of noise until sample 500, when *Preamble A* arrives. N = 512 samples, 25.6 microseconds, later, the ratio value has reached its peak. For a low signal to noise ratio of -10dB, detection can be determined by observing 200 consecutive ratio values larger 0.18.

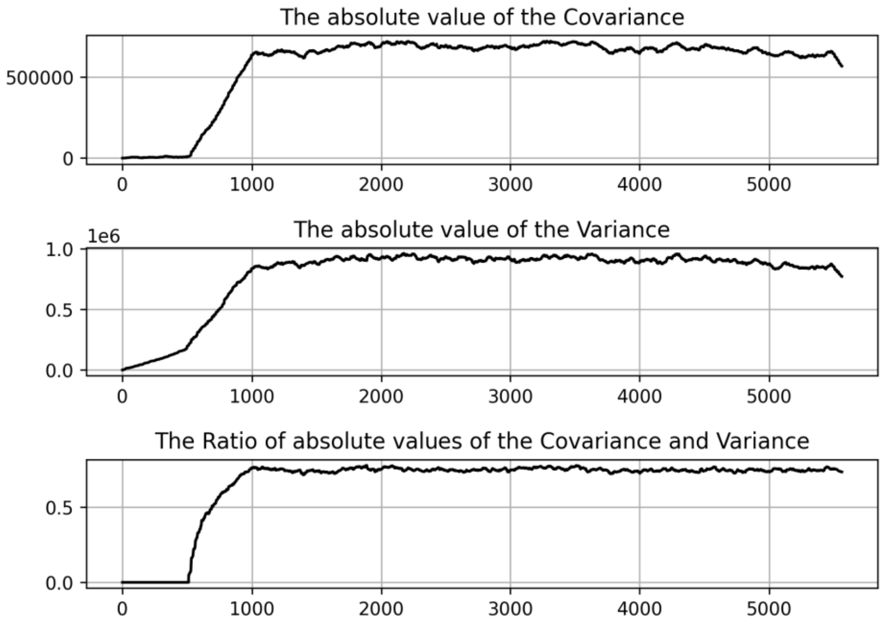


Figure 8-104: Preamble A Detector Performance for an SNR = 0dB and N = 512

The Length of Preamble A

The FlexLink transmitter can select between *Preamble A* lengths of 50 or 250 microseconds. The 50-microsecond length allows only for packet detection, whereas the 250-microsecond length allows for both packet detection and frequency offset estimation. The transmitter therefore has to make a choice as to the length of Preamble *A*, and the natural question beckons as to how the receiver will know the transmitter's selection ahead of time. In the absence of any information exchanged previously, the receiver needs to run the *Preamble B* detector at the same time as packet and frequency offset detectors. If the algorithm detects Preamble *B* within 100 microseconds of the packet detection, then *Preamble A* was set to 50 microseconds.

Frequency Offset Estimation

The difference between the center frequency of the transmitted signal and the local oscillator frequency, which is used to down convert the arriving signal from the RF to the base band domain in the receiver, consists of two factors. First, there is the unavoidable mismatch in the temperature compensated crystal oscillators (TCXO) that provide the reference clock to both the synthesizer, base band processor and mixed signal converters (ADCs/DACs). Each oscillator will produce a reference frequency that will have a range of some number of parts per million, or ppm. For example, two 20MHz TCXO with a deviation of ±1ppm, may each produce reference frequencies errors of 20e6 ·±1e-6 Hz = ±20Hz, which does not sound like a lot. However, once these references are used to generate local oscillator frequencies at 5GHz, the frequency error in these LO signals ballons to 5e9 ·±1e-6 Hz = ±5KHz. In the worst case, the frequency error can compound at the receiver to a total of 10KHz, which would completely invalidate the orthogonality of the signal when we take the FFT during OFDM demodulation. After all, the subcarrier spacing is only 19.53125KHz. See section 8.1.3.3 for a thorough explanation of loss of orthogonality in OFDM receivers. The frequency offset that our OFDM demodulator can easily tolerate when receiving 64QAM is on the order of ±250Hz. We therefore need a frequency offset detector that can resolve the error to within 250Hz in good SNR conditions, which are required when receiving 64QAM.

The second factor influencing the frequency offset at the receiver is Doppler. The frequency error due to Doppler is explained extensively in section 7.3.2 and reduces to the following expression, where *C*, F_{TX}, and *Vel* represent the speed of light, the signal's center frequency, and the terminals relative velocity toward one another respectively.

$$F_{Doppler} \cong F_{TX}\left(\frac{Vel}{C}\right)$$

For a center frequency of 5GHz and a velocity of 200 Kph, or 55 meters per second, the Doppler shift yields a value of ±925Hz, depending on whether the terminals are moving directly toward or away from one another.

$$\pm 5GHz\left(\frac{55\ m/sec}{300e6\ m/sec}\right) \cong \pm 925Hz$$

The frequency detector will resolve both the natural frequency offset due to the TCXO mismatch and Doppler simultaneously.

When to Estimate the Frequency Offset

Estimating the frequency offset in low signal to noise ratio conditions is much more sophisticated than detecting the presence of the FlexLink packet. When frequency offset detection is needed, the *Preamble A* length must always be 250 microseconds in length. The long *Preamble A* should be used to enable frequency detection for the following condition.

→ When the terminals make contact for the first time.
→ When the TXCOs drift due to changes in temperature. This is a slow process.
→ When a terminal executes a significant change in velocity and/or direction.
→ When a client terminal acts as a beacon for other terminals that want to access the serving terminal.

Clearly, the long *preamble A* needs to be used by the transmitter to make initial contact with another station as the initial frequency offset may be significant even for accurate TXCOs. Furthermore, temperature changes can cause the reference frequency to drift to a point where the resulting change in LO frequency causes offset that must be detected anew. Also, the media access controller at the respective terminals needs to keep track of the motion of both terminals and reissue the long *preamble A* each time a significant change in velocity or direction is executed. This is especially necessary when larger QAM constellations are transmitted, and a loss of orthogonality can't be tolerated.

Lastly, in a point to multipoint situation, where a new client terminal wants access to a distant serving terminal, it is helpful for other closer terminals to occasionally transmit the long *Preamble A* when communicating with the serving terminal.

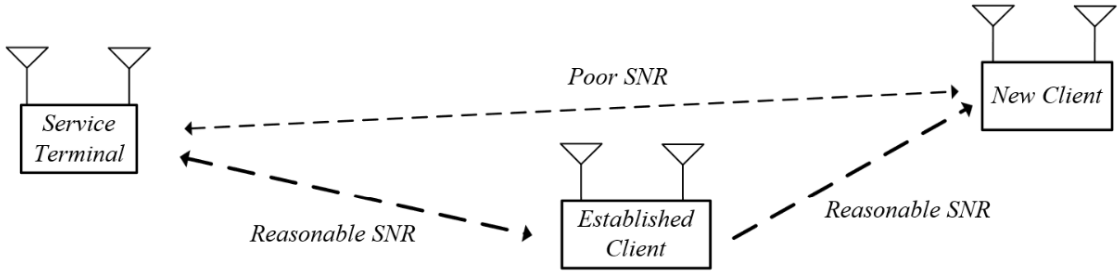


Figure 8-105: New Client Accessing the Serving Terminal

In the figure above, a new terminal with a poor SNR link to the serving terminal can estimate the frequency offset to a transmission of an established client rather than to the serving terminal. To make this possible, the established client must occasionally transmit the long *preamble A*, as well as information (in *payload A*) regarding its own relative frequency offset to the serving terminal that it has previously computed. With these two pieces of information the new client computes a better frequency offset estimate to the serving terminal and establishes communication.

<u>How to Detect Frequency Offset in Low SNR Conditions</u>

Once the Packet has been detected, the receiver needs to read the next 4096 samples, which span approximately 200 microseconds, into an FFT input buffer. The samples should be premultiplied by a Hanning window as they enter the buffer. The cosine waveforms of the Preamble A, with frequencies of 625000Hz and 1875000Hz, will appear at tones ±128 and ±384 in the FFT output. The Hanning overlay will spread content from those tones to the neighboring tones but suppress the DFT leakage into all other tones. The content of the Hanning overlaid cosine waveforms is therefore nicely contained within the center tones and the direct neighboring tones. This now allows us to zero out all other tones and take the inverse FFT, which recreates a hugely noise reduced version of the waveform that was in the FFT input buffer.

The procedure described above can be implemented in a variety of clever ways. In fact, there is no need to take the full FFTs or IFFTs to achieve the same results and estimate the frequency offset when we implement the detector in hardware.

8.9.5.3 Construction and Processing of Preamble B

The purpose of *Preamble B* is to present the receiver with a waveform that allows it to acquire timing. Remember from our discussion in 8.1.4 that the Zadoff-Chu sequence features low PAP and excellent auto-correlation behavior. This time, we are interested in the auto-correlation behavior, which will yield a very distinct peak when we correlate the received waveform against a copy of *Preamble B*. Finding the position of the peak means that we have acquired timing, or, in other words, we now have the ability to determine which sample in the received waveform represents the start of *Preamble B*. As the receiver is aware of the layout of the packet, it will thus know the start samples of all OFDM symbols and can begin the OFDM demodulation task. The process of generating Preamble B is identical to that of the AGC burst, with the exception that we retain a different number of samples from the Zadoff Chu output sequence.

From section 8.9.4.1, recall that the AGC burst is the initial portion of the following sequence that stretches of *N* (1024 or 2048) samples.

$$AgcBurstFull_N = \frac{N}{887}\, IFFT\ (IFFT\_INPUT)$$

Preamble B is an OFDM symbol, whose IFFT section is equal to $AgcBurstFull_N$, and whose cyclic prefix is made up of the last 116 or 232 samples of $AgcBurstFull_N$.

$$PreambleB_{1024} = [AgcBurstFull_{1024}[0{,}115], AgcBurstFull_{1024}]$$

or

$$PreambleB_{2048} = [AgcBurstFull_{2048}[0{,}231], AgcBurstFull_{2048}]$$

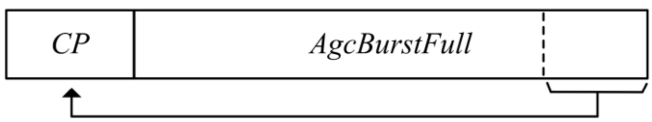


Figure 8-106: Preamble B Construction as an OFDM Symbol

<u>Processing of Preamble B</u>

Finding the position of the preamble B in the received packet requires the use of sliding correlation, which is explained in section 2.2.2 and 2.5.2.

8.9.6 Resource Grid Construction and Procedures

This section will define how data QAM values, reference signals, control information and DC (0+j0) content are mapped into the resource grid, whose columns are subsequently loaded into the IFFT input buffer as part of the OFDM modulation process. Whereas the mapping of reference signals and control data may not be completely obvious, data QAM values are mapped sequentially (k = 0, 1, 2, …) to every subcarrier not reserved for reference signals or DC content. The following figure illustrate how data QAM values are mapped into a column of the resource grid that contains only data and a column that contains both data and reference signals. For a two TX antenna configuration, resource grid P0 must map 0 + j0 (no content) to positions where resource grid P1 would apply its **reference** signals and visa-versa.

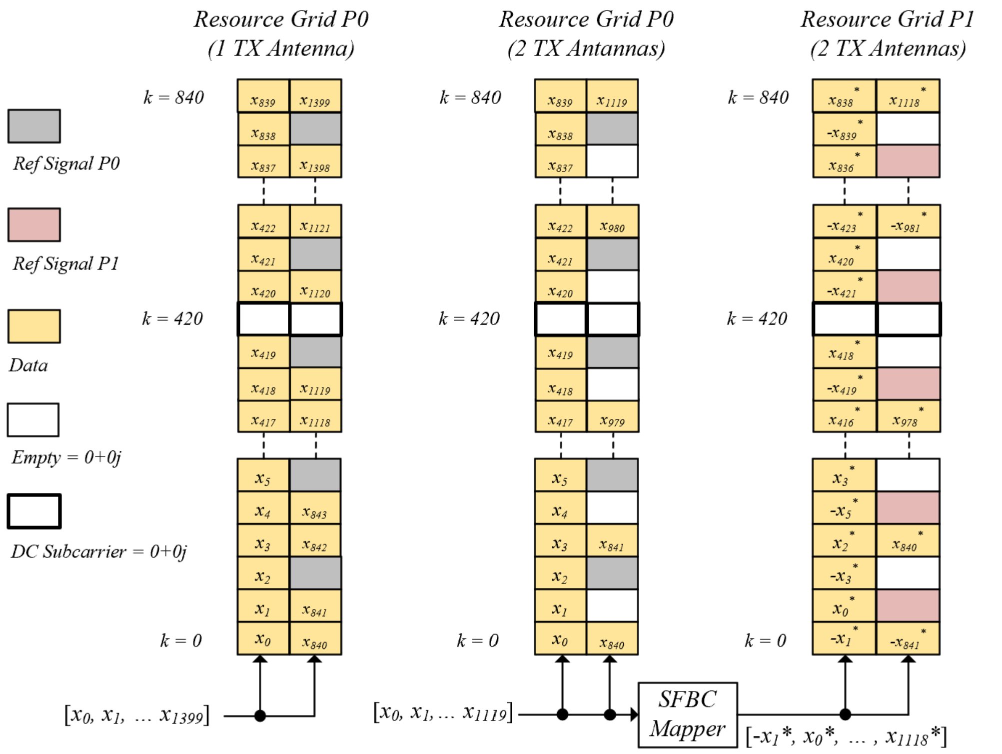


Figure 8-107: Data QAM Symbol Mapping for 1 and 2 TX Antennas (WLAN BW)

<u>*Mapping at DC Subcarriers*</u>

A value of 0 + j0 must be placed at the center DC subcarrier (k = 420 or 456) of both resource grids. For the case that there are 13 DC subcarriers, the 12 subcarriers not at the center may carry control information, reference signals but **no** data QAM values.

8.9.6.1 Reference Signals Generation and Mapping

There are two bits in the control information, which is embedded into the first reference symbol, that indicate the way reference signals should be placed into a column of the resource grid. These two bits represent the reference signal spacing, which is the distance between reference signals of a single port, as every 3rd, 6th, 12th and 24th subcarrier. The table below indicates, at which subcarrier indices, *k*, the reference signals should be placed.

Table 8-35: Location of Reference Signals

Spacing	Expression P0	Expression P1
3	$k = 2 + n \cdot 3$	$k = 1 + n \cdot 3$
6	$k = 2 + n \cdot 6$	$k = 1 + n \cdot 6$
12	$k = 5 + n \cdot 12$	$k = 4 + n \cdot 12$
24	$k = 11 + n \cdot 24$	$k = 10 + n \cdot 24$

Note that the variable *n* increments from 0, 1, 2, 3, … to whatever integer is required to fill the entire OFDM symbol (resource grid column).

Reference Signal Orientation for Transmit Port 0 and Port 1

The reference signal orientation for transmit port 0 is determined by BPSK mapping the default scrambling sequence specified in section 8.9.3.8. Given the reference signal index, *n*, as seen in the table above, and the associated subcarrier index, *k*[*n*], the reference signal orientations are as follows.

$$ReferenceSignal[k[n]] = BPSK(ScramblingSequence[n]$$

The reference signal index, *n* = 0, 1, 2, …, extends until all reference signal opportunities in a reference symbol are exhausted.

Precomputing Basic Bit Capacity Tables

Mapping QAM symbols into the resource grid takes a bit of planning, and constructing the following capacity tables ahead of time will save the computation unit, doing the mapping, some work. Therefore, given the resource block index, the bits per QAM symbol for that resource block, and the reference signal layout, compute the capacity as bits per resource blocks and write them into the following table.

Table 8-36: Capacity Table (See Code Example)

Resource Block Index	Bits Per Symbol (BPS)	Capacity (Bits) Data Symbol	Capacity (Bits) Reference Symbol
0	4	48	32
1	4	48	32
2	4	48	32
:	:	:	:
69 / 75	4	48	32

For *Payload A*, the bits per symbol for each resource block are provided in the signal field, and the reference signal spacing is provided in the control information. The bits per second may be constant for all resource blocks or they may vary. For *Payload B*, the bits per symbol are indicated in MAC subheaders (Id2 / Id3), which appear in *Payload A*. We revisit this table when mapping the payloads. The MatLab code below generates the following tables that the MAC should be using during the mapping process.

→ The Capacity Tables, indicating the number of data bits that each resource block can hold for a pure OFDM data symbol, which contains no reference signals, and a reference symbol, which contains both data and reference signals.

→ The Data Mapping Tables, which indicate whether a subcarrier holds 0 + 0j (0), data (1), a reference signal for port0 (2) or a reference signal for port 1 (3).

→ The Data Index Table for a Data Symbol, which is an array of monotonically increasing subcarriers indices k, which hold data in an OFDM data symbol. The table makes it easy for the baseband processor to map to only data carrying subcarriers.

→ The Data Index Table for a Reference Symbol, which is an array of monotonically increasing subcarriers indices k, which hold data in an OFDM reference symbol.
The code starts by defining some basic constants that we have seen in the specifications. This is followed by parameters that the MAC must set including the bandwidth, some of the control information and the BPS_A_Flag in the signal field. These parameters are shown in **bold** blue letters and may be changed by the user to see their effects.

```
% -------------------------------------------------
% Constants
% -------------------------------------------------
NumOfREPerRB                 = 12; % The number of resource elements per resource block
PossibleNumberResourceBlocks = [70, 76];       % WLAN / LTE bandwidths
PossibleRefSymbolPeriodicity = [1, 3, 6, 12];  % OFDM symbols
PossibleRefSignalSpacings    = [3, 6, 12, 24]; % Subcarriers
PossibleDCSubcarriers        = [1, 13];        % Subcarriers
BPS_Static                   = [1, 2, 4, 6];   % Bits/QAM symbol signal field format 1
                                               % Bits/QAM symbol signal field format 2
BPS_Dynamic                  = [0, 1, 2, 4, 6, 8, 10, 10];


% -------------------------------------------------
% Configuration (Global): The modem must be set up for either the WLAN or LTE BW
% -------------------------------------------------
BwFlag                       = 0;  % 0 - WLAN BW / 1 - LTE BW

% Which results in the following parameters
NumberResourceBlocks         = PossibleNumberResourceBlocks(1, BwFlag + 1);
NumberSubcarriers            = NumOfREPerRB * NumberResourceBlocks + 1;
Center_Subcarrier_Index      = (NumberSubcarriers - 1) / 2;

if(BwFlag == 0)
    disp(' ------------------    WLAN Bandwidth Configuration  ---------------------');
else
    disp(' ------------------    LTE  Bandwidth Configuration  ---------------------');
end

disp(['Total number of resource blocks:                ', num2str(NumberResourceBlocks)]);
disp(['Total number of subcarriers:                    ', num2str(NumberSubcarriers)]);
disp(['The center subcarrier carrier index k:          ', num2str(Center_Subcarrier_Index)]);

% -------------------------------------------------
% Configuration (Control Info): MAC must configure the following control information
% -------------------------------------------------
RefSymbolPeriodicityIndex    = 1;
ReferenceSignalSpacingIndex  = 1;
SignalFieldFormatIndex       = 0;
```

```
NumTxAntennaPortsFlag        = 1;
DcSubcarriersFlag            = 1;

% Which results in the following parameters and masks
RefSymbolPeriodicity  = PossibleRefSymbolPeriodicity(1, RefSymbolPeriodicityIndex + 1);
RefSignalSpacing      = PossibleRefSignalSpacings(1, ReferenceSignalSpacingIndex + 1);
SignalFieldFormat     = SignalFieldFormatIndex + 1;
NumTxAntennaPorts     = NumTxAntennaPortsFlag  + 1;
NumDcSubcarriers      = PossibleDCSubcarriers(1, DcSubcarriersFlag + 1);

% We build a DC mask indicating at which subcarrier we may not map data QAM Symbols
DC_Mask               = zeros(1, NumberSubcarriers);
C                     = floor(NumDcSubcarriers/2);
DC_Mask(1, 1 + Center_Subcarrier_Index - C: 1 + Center_Subcarrier_Index + C) = ones(1,
NumDcSubcarriers);

% We build a reference signal mask indicating at which subcarriers we may
% not map data QAM symbols due to the presence of a reference signal
RefSignal_Mask        = -1*ones(1, NumberSubcarriers);
n0 = 0;
n1 = 0;
for k = 0:NumberSubcarriers-1
    switch(RefSignalSpacing)
        case 3
            if (k == 2 + n0*3)    % Antenna port 0
                RefSignal_Mask(1, k+1) = 0; n0 = n0 + 1;
            end
            if (k == 1 + n1*3 && NumTxAntennaPorts == 2) % Antenna port 1
                RefSignal_Mask(1, k+1) = 1; n1 = n1 + 1;
            end
        case 6
            if (k == 3 + n0*6)    % Antenna port 0
                RefSignal_Mask(1, k+1) = 0; n0 = n0 + 1;
            end
            if (k == 2 + n1*6 && NumTxAntennaPorts == 2) % Antenna port 1
                RefSignal_Mask(1, k+1) = 1; n1 = n1 + 1;
            end
        case 12
            if (k == 6 + n0*12)    % Antenna port 0
                RefSignal_Mask(1, k+1) = 0; n0 = n0 + 1;
            end
            if (k == 5 + n1*12 && NumTxAntennaPorts == 2) % Antenna port 1
                RefSignal_Mask(1, k+1) = 1; n1 = n1 + 1;
            end
        case 24
             if (k == 12 + n0*24)    % Antenna port 0
                 RefSignal_Mask(1, k+1) = 0; n0 = n0 + 1;
             end
             if (k == 11 + n1*24 && NumTxAntennaPorts == 2) % Antenna port 1
                 RefSignal_Mask(1, k+1) = 1; n1 = n1 + 1;
             end
    end
end

disp(' ')
disp(' ----------------  Parameters resulting from control information -------------');
disp(['Reference symbol periodicity (in OFDM symbols): ', num2str(RefSymbolPeriodicity)]);
disp(['Reference signal spacing (in subcarriers):      ', num2str(RefSignalSpacing)]);
disp(['Signal field format:                            ', num2str(SignalFieldFormat)]);
disp(['Number of Tx Antennas:                          ', num2str(NumTxAntennaPorts)]);
disp(['Number of DC subcarriers:                       ', num2str(NumDcSubcarriers)]);


% ------------------------------------------------
% Configuration (Signal Field): MAC must configure the BPS_A_Flag signal field element
```

```
% ------------------------------------------------
% The MAC configures BPS_A_Flag_SF1 or BPS_A_Flag_SF2
BPS_A_Flag_SF1 = 2;                                  % 0,1,2,3  (For signal field format 1)
BPS_A_Flag_SF2 = randi([0, 7], 1, NumberResourceBlocks); % An array of values
                                                         % 0,1,2,3,4,5,6,7
                                                         % For signal field format 2

% Let's assign a Bit per QAM symbol value for each subcarrier based on BPS_A_FLAG
% This is help us create the capacity tables, which happens next
BPS_RE_Array   = zeros(1, NumberSubcarriers);    % An array of bits per QAM symbol

if(SignalFieldFormat == 1)       % Static mapping
    BPS_RE_Array = BPS_Static(1, BPS_A_Flag_SF1 + 1) * ones(1, NumberSubcarriers);
elseif (SignalFieldFormat == 2)  % Dynamic Mapping
    for k = 0:1:NumberSubcarriers-1
        % Careful, the center subcarrier does not belong to any resource block
        if(k == Center_Subcarrier_Index)
            continue
        elseif(k < Center_Subcarrier_Index)
            CurrentResourceBlock    = floor(k/NumOfREPerRB);
        else
            CurrentResourceBlock    = floor((k-1)/NumOfREPerRB);
        end
        CurrentBitsPerQamSymbol = BPS_A_Flag_SF2(1, CurrentResourceBlock + 1);
        BPS_RE_Array(1, k+1)    = CurrentBitsPerQamSymbol;
    end
else
    assert (false, "Invalid Signal Filed Format.");
end

% ----------------------------------------------
% The MAC builds the capacity tables, data mapping tables and data index tables
% ----------------------------------------------
% Row 1 holds the resource block capacities for an OFDM data symbol (no ref signals)
% Row 2 holds the resource block capacities for an OFDM reference symbol (both data and reference
signals)
CapacityTables    = zeros(2, NumberResourceBlocks);

% The DataMappingTable is of length NumberSubcarriers and contains the following mapping values:
% 0 -> Meaning that 0 + 0j is to be mapped here
% 1 -> Meaning that Data can be mapped at this index.
% 2 -> Meaning that reference rignal for Antenna Port0 are to be mapped here
% 3 -> Meaning that reference rignal for Antenna Port1 are to be mapped here
% Row 1, of the table, uses the mapping values for a OFDM data symbol.
% Row 2, of the table, uses the mapping values for a Reference symbol.
% The table features NumberSubcarriers columns with index k = 0, 1, ... NumberSubcarriers-1.
% The first row of this table is all ones.
DataMappingTables = zeros(2, NumberSubcarriers);

% The data index table is consists of those indices k that hold data REs.
% The baseband processor uses the table to quickly figure out at which
% k it is allows to map data.
DataIndexTableDataSymbol = [];
DataIndexTableRefSymbol  = [];


% Let's build all the tables
for k = 0:NumberSubcarriers-1
    % Determine the CurrentResourceBlock
    % Careful, the center subcarrier does not belong to any resource block
    if(k == Center_Subcarrier_Index)
        continue
    elseif(k < Center_Subcarrier_Index)
        CurrentResourceBlockIndex = floor(k/NumOfREPerRB);
    else
```

```
        CurrentResourceBlockIndex = floor((k-1)/NumOfREPerRB);
    end


    % ------------------------------------------------------------
    % This portion deals with the OFDM data symbol (No ref signals present)
    % ------------------------------------------------------------
    % 1. Update row 1 of CapacityTables and DataMappingTables
    % 2. Fill out DataIndexTableDataSymbol

    % If this is not a DC subcarrier, then append the k value
    IsDcSubcarrier            = DC_Mask(1, k+1)        == 1;

    % Update the first row of the Capacity table (OFDM data symbol)
    BPS = BPS_RE_Array(1, k + 1);
    if(IsDcSubcarrier == false)
        CapacityTables(1, CurrentResourceBlockIndex + 1) = ...
            CapacityTables(1, CurrentResourceBlockIndex + 1) + BPS;
    end

    % Update the first row of the DataIndexTableDataSymbol
    if(IsDcSubcarrier == false)
        DataMappingTables(1, k + 1) = 1;
        DataIndexTableDataSymbol    = [DataIndexTableDataSymbol, k];
    end


    % ------------------------------------------------------------
    % This portion deals with the OFDM ref symbol (Ref signals and data are present)
    % ------------------------------------------------------------
    IsRefSignalP0             = RefSignal_Mask(1, k+1) == 0;
    IsRefSignalP1             = RefSignal_Mask(1, k+1) == 1;

    % Can we map a data QAM Symbol to the resource element at this subcarrier?
    IsDataSubcarrier          = IsRefSignalP0  == false && ...
                                IsRefSignalP1  == false && ...
                                IsDcSubcarrier == false;

    if(IsDataSubcarrier == true)
        DataMappingTables(2, k + 1) = 1;
        DataIndexTableRefSymbol     = [DataIndexTableRefSymbol, k];
    end

    % This table is simply a formal mapping information table
    % Fill in row two of the DataMappingTables
    if(IsRefSignalP0 == true);  DataMappingTables(2, k+1) = 2; end
    if(IsRefSignalP1 == true);  DataMappingTables(2, k+1) = 3; end
    if(IsDcSubcarrier == true); DataMappingTables(2, k+1) = 0; end


    % If this is not a data carrying resource element then skip as there
    % is no sense in adding anything to the bit capacity.
    if(IsDataSubcarrier == false)
        continue
    end

    % Update the second row of the Capacity table (OFDM reference symbol)
    BPS = BPS_RE_Array(1, k + 1);
    CapacityTables(2, CurrentResourceBlockIndex + 1) = ...
        CapacityTables(2, CurrentResourceBlockIndex + 1) + BPS;
end

figure(1)
stem(0:NumberSubcarriers-1, DC_Mask, 'k'); grid on;
title('The DC Mask')
```

```
xlabel('Sucarrier Index k')

figure(2)
stem(0:NumberSubcarriers-1, RefSignal_Mask, 'k'); grid on;
title('The Reference Signal Mask')
xlabel('Sucarrier Index k')

figure(3)
stem(DataIndexTableDataSymbol, 'k'); grid on;
title('The Subcarriers Indices at which to Map Data QAM Symbols (OFDM Data Symbol)')

figure(4)
stem(DataIndexTableRefSymbol, 'k'); grid on;
title('The Subcarriers Indices at which to Map Data QAM Symbols (OFDM Rerence Symbol)')

figure(5)
subplot(2,1,1)
stem(0:NumberSubcarriers-1, DataMappingTables(1,:), 'k'); grid on; hold on;
title('Data Mapping Table OFDM Data Symbol')
xlabel('Subcarrier Index k')
subplot(2,1,2)
stem(0:NumberSubcarriers-1, DataMappingTables(2,:), 'k'); grid on; hold on;
title('Data Mapping Table OFDM Reference Symbol')
xlabel('Subcarrier Index k')

figure(6)
subplot(2,1,1)
bar(0:NumberResourceBlocks-1, CapacityTables(1,:), 'k'); grid on; hold on;
title('Capacity Table Data Symbol')
xlabel('Resource Block Index')
ylabel('Number of Data Bits')
subplot(2,1,2)
bar(0:NumberResourceBlocks-1, CapacityTables(2,:), 'k'); grid on; hold on;
title('Capacity Table Reference Symbol')
xlabel('Resource Block Index')
ylabel('Number of Data Bits')
```

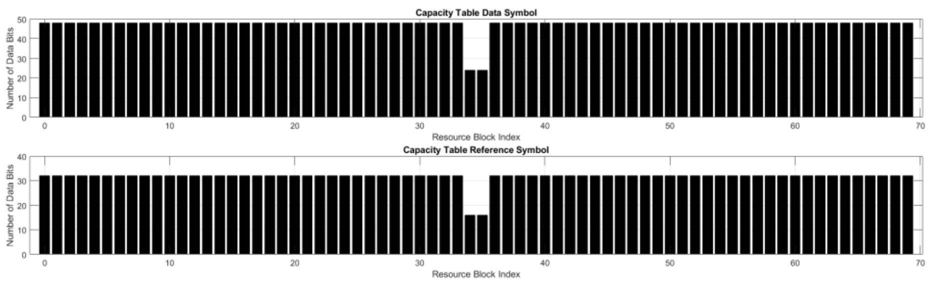


Figure 8-108: The Capacity Table Shown as a Bar Plot

The configuration in the code indicated the presence of 13 DC subcarriers, which reduces the capacity for the two resource blocks in the center as we can't map data QAM symbols there.

8.9.6.2 Mapping and Construction of the First Reference Symbol

The first reference signal is provided to enable the receiver to determine the frequency response and signal to noise ratio associated with transmit ports P0 and P1, if two transmit antenna ports are being used. Note that one port must assign a 0 + j0 value to those resource elements used by the other port. There may be no overlap as is the case for data resource elements formed via SFBC. The first reference signal also contains the crucial control information bits.

Control Information Bits

Table 8-37: Control Bit Ordering

Bit	Description	Bit	Description
c_0	MSB – Reference Symbol Periodicity	c_6	MSB – Signal Field Format
c_1	LSB – Reference Symbol Periodicity	c_7	LSB – Signal Field Format
c_2	MSB – Reference Signal Spacing	c_8	Signal Field Format is BPSK/QPSK
c_3	LSB – Reference Signal Spacing	c_9	Number of TX Antennas Ports
c_4	MSB – Number Signal Field Symbols	c_{10}	DC Subcarrier
c_5	LSB – Number Signal Field Symbols	c_{11}	Parity Check Bit (even parity)

Whereas the table above only assigns control bit positions, the meaning of these bits is explained in table 8-2. Note that the control information is only transmitted from antenna port P0. We currently define 12 control bits, which shall be mapped as BPSK symbols as follows.

→ Starting at $k = 0$ and proceeding to $k = 840/912$, identify those resource elements reserved for control information using the index b. There is one control element opportunity, O_b, for every subcarrier triplet, which works out to 280 / 304 total opportunities for the WLAN / LTE bandwidths. As there are 12 control information bits, they will be assigned to opportunities, O_b, as follows.

$$O_b = c_{b\%12} \quad b = 0, 1 \ldots 279 \; or \; 303$$

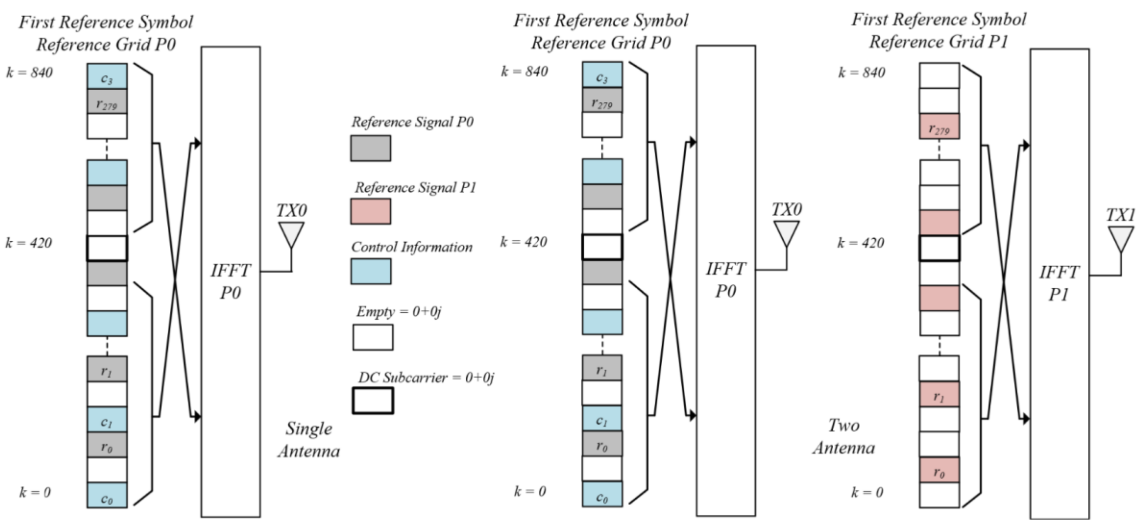


Figure 8-109: The First Reference Signal For Resource Grids P0 and P1 (WLAN BW)

In the figure above, showing the WLAN bandwidth scenario, the first control element opportunity, O_0, resides at subcarrier k = 0, whereas the last, O_{279}, resides at subcarrier k = 840. The control information bits mapped to these positions are $c_{0\%12} = c_0$ and $c_{279\%12} = c_3$. The control information is scrambled as shown in section 8.9.3.8.

8.9.6.3 Mapping and Construction of the Signal Field

Four control field quantities influence the construction and mapping of the signal field.

Table 8-38: Needed Control Information Parameters

Parameter	Size (bits)	Description
Signal Field Format Index	2	0/1/2/3 = Format 1/2/3/4
Number of Signal Field Symbols Index	2	0/1/2/3 = 1/2/4/8 OFDM Symbols
Signal Field Modulation Flag	1	0/1 = BPSK / QPSK
Number of TX Antenna Flag	1	0/1 = 1 or 2 TX Antenna Ports

Currently we define only two signal field formats, where the first format consists of 64 bits, and the second format consists of 290 bits as seen in section 8.9.3.4. The signal field using format 1 will be encoded using a 256 bit polar encoder resulting in a rate equal to ¼. The signal field using format 2 will be encoded using a 1024 bit polar encoder resulting in a rate close to ¼.

These 256 and 1024 bits are first interleaved and then repeated to completely fill between 1 2, 4, or 8 OFDM symbols (columns in the resource grid). The receiver will sum the repeated bits back into a single set of 256 and 1024 encoded bits. This summing process raises the signal to noise ratio of each bit as $10{\cdot}\log_{10}$(Number of Copies) dB. Thus, if the set of 256 or 1024 encoded bits are repeated once, then two copies of the bits exist and the signal to noise ratio increases by approximately $10{\cdot}\log_{10}(2)$ or 3dB prior to the polar decoding process in the receiver.

The media access controller continually observes the signal to noise ratio of the signal of the station(s), with which he is in contact, and will switch between BPSK and QPSK to optimize the number of OFDM symbols in order to provide the appropriate protection.

Like the rest of the resource grid, the signal field data resource elements are mapped using SFBC in case two antennas are indicated by the control information.

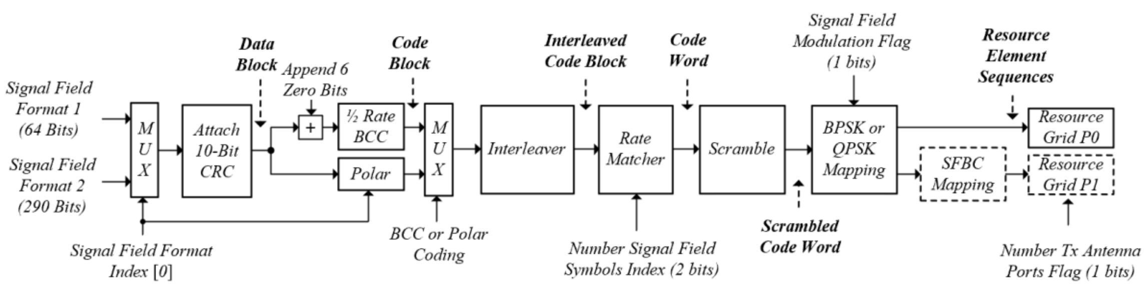


Figure 8-110: Encoding Chain for Signal Field Processing

8.9.6.4 Mapping and Construction of Payload A

Payload A contains the MAC header and may contain user information. In order to construct and map *payload A*, we use the following parameters, which are known to the MAC and are recorded in the signal field. Further, to construct *Payload A*, the MAC must have organized the encoding of all data blocks into code blocks and filled the Code Block Ram (see section 8.9.6).

Table 8-39: Required Parameters for PayloadA Generation

Parameter	Size (bits)	Description	Possible Values
CBS_A_Flag	2	Code Block Size LDPC	[0,1, 2, 3] → 648, 1296, 1944, 1944
NDB A	16	Number of Data Blocks	0 through 2^{16} -1
FEC_A_Flag	3	FEC Encoder Rate LDPC	[0,1, …,7] → ½,2/3,¾,5/6, ½, ½, ½, ½
RM_A_Flag	3	The rate matching amount	[0, 1, … 7] → 0.0, 0.5, 0.75, 1, 3, 7, 15, 31
BPS_A_Flag F1	2	Bits per QAM Symbol Signal Field Format 1	[0, 1, 2, 3] → 1, 2, 4, 6 bits per QAM Symbol (Signal Field Format 1)
BPS_A_Flag F2	3	Bits per QAM Symbol Signal Field Format 2	[0, 1, …, 7] → 0, 1, 2, 4, 6, (8, 10, 10 optional) Bits per QAM Symbol

The data blocks provided by the media access controller will have a 12 bit CRC attached to become data words. This process may happen in the MAC itself or be done in the physical layer. This secification does not define the location of the CRC attachment.

Step 1

The first parameter tells us whether the modem will use low density parity check (LDPC) or polar encoding to protect *Payload A* against errors. Together with the associated code block size and FEC encoding rate, we can determine the required size of the data word and data block.

$$data\ word\ size \rightarrow DWS\ = CBS\ \cdot Encoding\ Rate$$

Remember that the data word is composed of the data block and the appended CRC of the data block.

$$data\ block\ size \rightarrow DBS\ = DWS - CRC\ bit\ size$$

The total number of data bits that the MAC intends to map into *Payload A* now becomes known by simply multiplying the data block size by the number of data blocks in *payload A* (*NDB A*).

$$TotalNumUserBitPayloadA = DBS \cdot NDB$$

Step 2

It is now time to build the capacity table which we mentioned at the start of this section. If signal field format 1 is being transmitted, then the capacity table is a single number that represents the number of encoded bits that can be placed into a single resource block of a single OFDM symbol. Given that the resource block is composed of 12 subcarriers, the bit capacity of each resource block is as follows.

$$BitCapacityResourceBlock \leq 12 \cdot BPS\_A\_F1$$

The *BPS_A_F1* parameter equals 1 (BPSK), 2 (QPSK), 4 (16QAM), and 6 (64QAM).

If signal field format 2 is being used, then each resource block, starting with resource block 0 and ending in resource block 69 (WLAN bandwidth) or resource block 75 (LTE bandwidth) is enumerated as a 3-bit quantity. Remember that this type of information can only be available if the receiver has previously informed the transmitter about the channel conditions, making it possible for the transmitter to customize the bits per symbol for each resource block.

$$BitCapacityResourceBlock \leq 12 \cdot BPS\_A\_F2$$

Note further that we allow additional values for the BPS parameters including 0 bits, meaning that no information will be mapped into the associated resource block due to poor signal to noise ratio at this resource block in the channel. Optionally, 256 QAM and 1024 QAM may be used in those modems that support the feature.

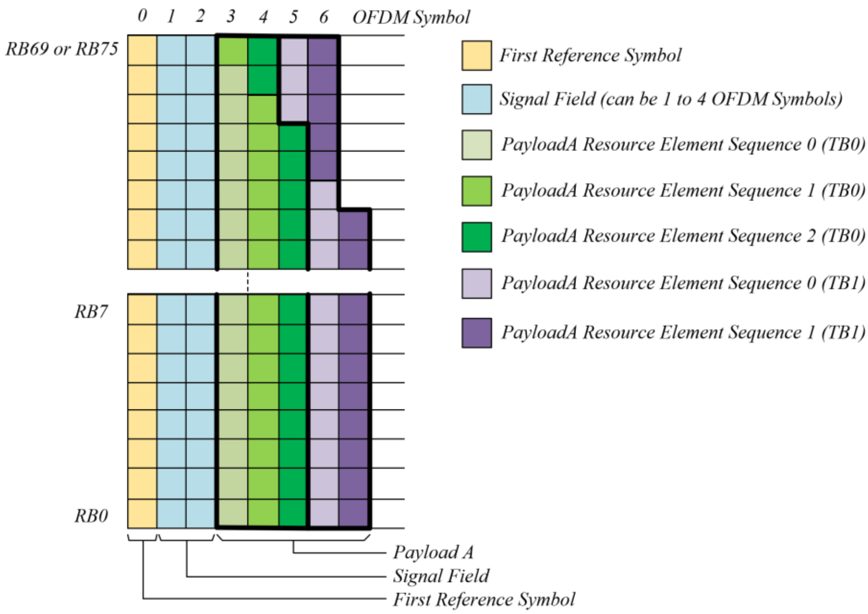


Figure 8-111: Mapping of Payload A

Step 3

There are a certain number of data blocks, NDB A, that need to be mapped into *Payload A*. Each data block will have a CRC appended to it to form the data word. The data word is error-encoded and interleaved to for the code block. At this point, the code blocks should be written into code block memory as we do not need to and may not want to execute the next steps immediately. Writing the code blocks into memory allows the transmitter to decouple the basic error encoding and interleaving steps from the rest of the packet creation. After all, it is likely that the payload transmit data is known well ahead of time of the actual packet transmission. Using code block RAM is an implementation suggestion, not a requirement, meant to ease the construction of the packet in the transmitter.

Step 4

The number of data blocks, NDB A, that are reserved for *payload A* is indicated by the signal field. Once the code words are available in RAM, the transmitter extracts each one, rate matches them to fill an integer number of resource blocks and QAM maps the bits into a resource element sequence. These resource element sequences can be seen in the next figure as the green and purple regions, which are always mapped from the lowest to the highest possible resource block index in each OFDM symbol. In the figure, NDB A is equal to five, representing two transport blocks. When the receiver demaps the different resource element sequences, it does not know which belong to which transport block. The receiver must decode the content of *payload A*, which reveals the MAC header. It is the information in the MAC subheaders that reveals which resource element sequences belong to which transport block.

8.9.6.5 Mapping and Construction of Payload B

Payload B can be mapped using vertical or block mapping. If vertical mapping is chosen, then the resource element sequences continue to be mapped (starting at RB64 / symbol 7 in figure below) as they had been for *Payload A*. The only difference between the data processing of *Payload A* and *Payload B* data is the fact that each transport block can have its own custom QAM constellation, rate matching and error encoding rate (see MAC subheaders ID2 and ID3).

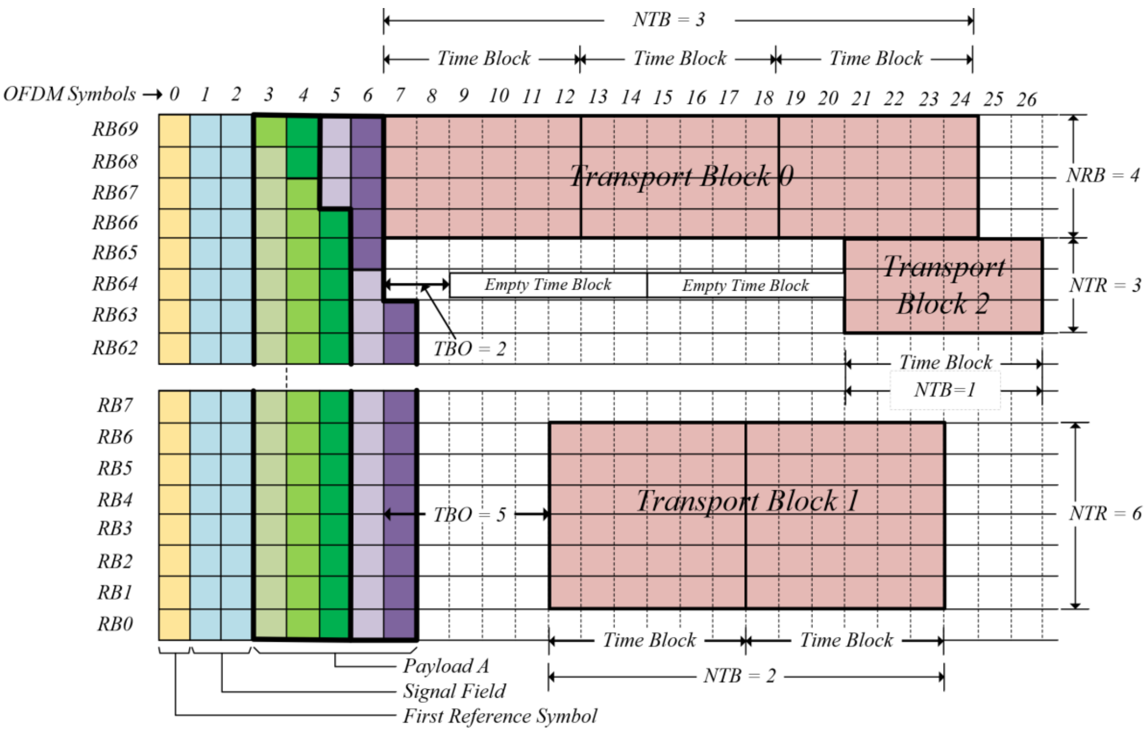


Figure 8-112: Block Mapping of Transport Blocks in Payload B

Block mapping is similar to vertical mapping with the exception that the resource block and OFDM symbol ranges are limited to a specific mapping area in the resource grid. Mapping QAM symbols into resource elements still happens from the lowest to the highest subcarrier index, *k*, and the lowest to the highest OFDM symbol, *l*. The following parameters define the location of a mapping block in the resource grid. The rightmost column indicates what these values should be for the mapping blocks shown in the last figure. Also remember that a single mapping block always holds a single transport block of original data bits provided by the MAC to the physical layer.

Table 8-40: Parameters Defining a Mapping Block in Payload B

Name	Bit Size	Description	Example Values
RBSI	7	The resource block start index.	[66, 1, 63]
NRB	6	The number of resource blocks.	[4, 6, 3]
TBO	3	The time block offset in OFDM symbols.	[0, 5, 2]
TBSI	10	The time block start index.	[0, 0, 2]
NTB	8	The number of time blocks.	[3, 2, 1]

Whereas resource block parameters are easy to understand from the last figure, the range of OFDM symbols covered by the mapping block is not. The following expression governs the start OFDM symbol index, *l*, of a mapping block.

$$\begin{aligned} \mathit{Start\ Ofdm\ Symbol\ Index} \qquad l = 1 &+ \mathit{Number\ of\ Signal\ Field\ Symbols} \\ &+ (\mathit{Number\ of\ PayloadA\ Symbols} - 1) \\ &+ TBO \\ &+ 6 \cdot TBSI \end{aligned}$$

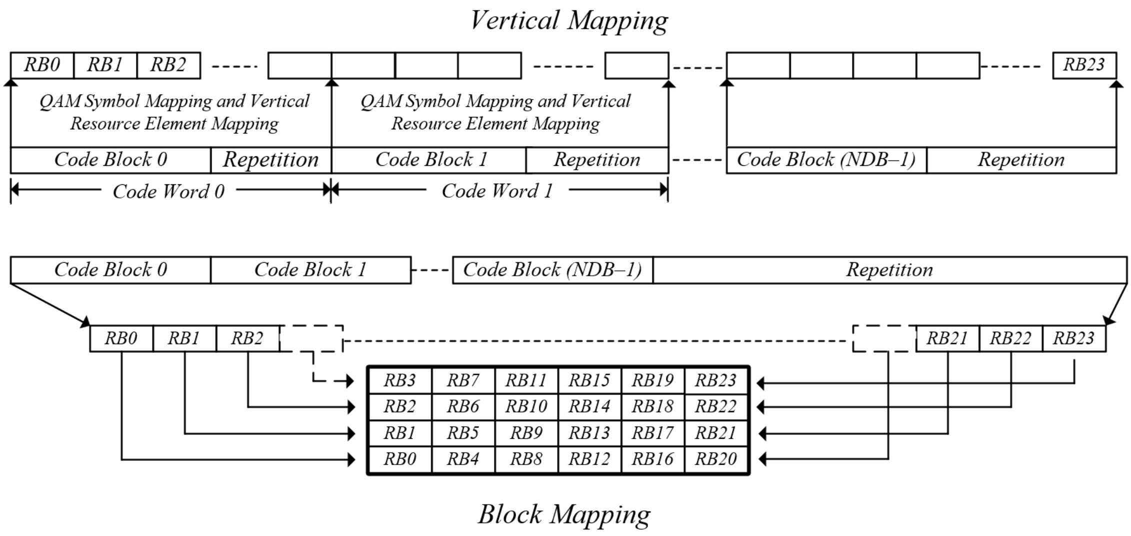
As an example, the mapping block for transport block 2 begins at OFDM symbol 21, which is the sum of 1 (first reference symbol), 2 (signal field symbols), 5 (*payload A* symbols) - 1, 2 (time block offset symbols), and 6 times the number of OFDM symbols in one time block.

$$l = 1 + 2 + 5 - 1 + 2 + 6 \cdot 2 = 21$$

To avoid using OFDM symbols to express the time extent of the mapping block, we introduce a time block consisting of 6 OFDM symbols. The time block start index simply refers to the number of empty blocks shown in the last figure for transport block 2, or any other transport block residing in *payload B.*

Rate Matching for Block Mapping in Payload B

One additional difference to vertical mapping is the way that rate matching is done for block mapping. In the vertical mapping case, a code block was rate matched to yield a code word. The code word size was then slightly increased so that after QAM mapping, an integer number of resource blocks could be fully occupied. In block mapping, this concept cannot be applied as easily since we must fill the entire mapping area, not just make a minor increase to fill the last resource block. For this reason, code blocks are concatenated first before being rate matched to fill the entire mapping block, and the expression *code word* is not used in block mapping.


Vertical Mapping
RB0
RB1
RB2
RB23
QAM Symbol Mapping and Vertical Resource Element Mapping
QAM Symbol Mapping and Vertical Resource Element Mapping
Code Block 0
Repetition
Code Block 1
Repetition
Code Block (NDB–1)
Repetition
Code Word 0
Code Word 1
Code Block 0
Code Block 1
Code Block (NDB–1)
Repetition
RB0
RB1
RB2
RB21
RB22
RB23
RB3 RB7 RB11 RB15 RB19 RB23
RB2 RB6 RB10 RB14 RB18 RB22
RB1 RB5 RB9 RB13 RB17 RB21
RB0 RB4 RB8 RB12 RB16 RB20
Block Mapping

8.9.7 Transmitter Architecture and Procedures

A lot of processes need to act in concert in order to produce the transmit waveform at the antenna port(s). To reduce the amount of coordination, the transmitter decouples certain processes that may be executed and finish independently of one another.

The Preamble ROM

The preamble (AGC burst, *Preamble A*, and *Preamble B*) is a sequence of complex samples that never changes. It should therefore be prerendered and stored in memory to be read out via transmit antenna port 0 on demand.

FEC Encoding

In virtually all applications, the transmitter knows what it wants to convey to the receiver well ahead of the actual RF transmission. At our leisure, we can therefore build and store some or all of the code blocks that are destined for the signal field and the two payloads. These code blocks can remain in the code block RAM until they are needed by the resource element mapper.

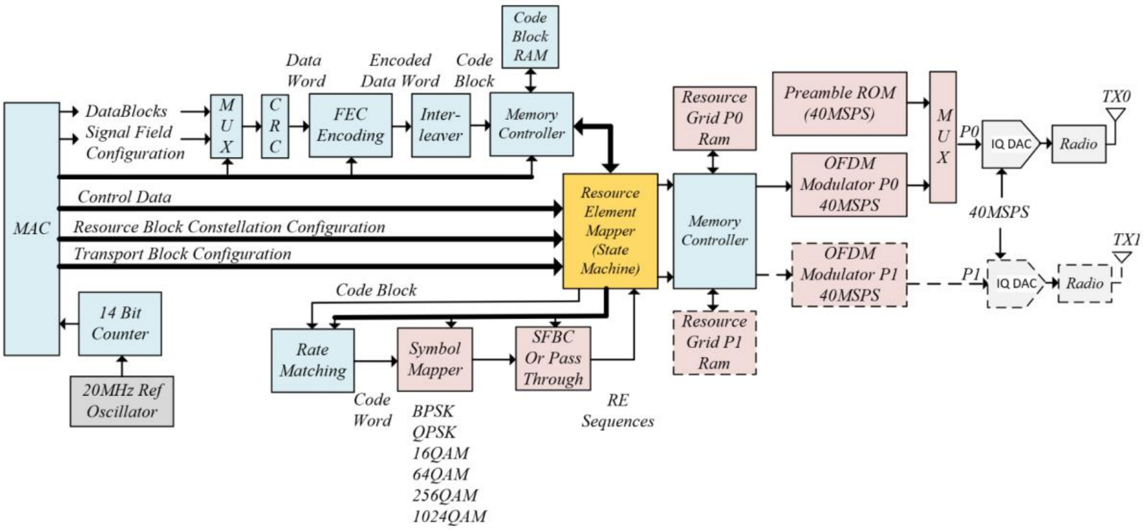


Figure 8-113: Conceptual Diagram of FlexLink Transmitter Implementation

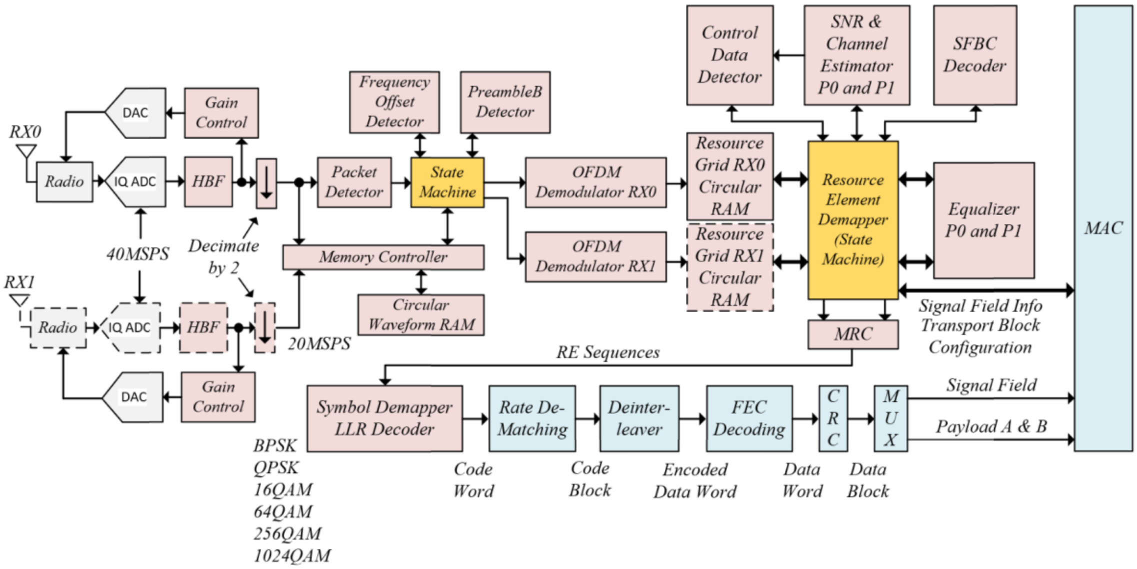
Resource Element Mapping

The resource element mapper is a state machine that is likely implemented as a combination of one or more CPUs and several finite state machines. It will have to extract the different code blocks from the code block RAM, apply rate matching, interleaving, symbol mapping and potentially SFBC encoding prior to placing them into the resource grid RAM modules. The overall encoding chain is broken into two sections as the resource element mapper needs additional configuration information from the MAC in order to properly rate match the information.

To be continued

8.9.8 Receiver Architecture and Procedures

Under construction.



8.9.9 The Media Access Control Layer

The responsibility of the media access controller, or MAC, is to manage access to the physical layer. The list below enumerates the main tasks of the MAC.

→ Cooperation with the radio link layer to disassemble upper layer data sequences into transport blocks on the TX side, and later the reassembly of transport blocks into upper layer sequences at the RX side.

→ Continuous analysis of the RF link including the frequency response, SNR and Doppler conditions of the communication channel being maintained with each client.

→ The proper management of control information and signal field content. After analyzing the communication channel, the overall link is optimized by properly setting reference symbol/signal spacing, the signal field format index and modulation flag as well as configuring parameters in the signal field to ensure proper reception of *payload A.*

→ The organization of user data of each transport block. FEC encoding, rate matching, QAM mapping and the physical placement of the transport blocks in the resource grid are all managed by the MAC. All these parameters must adaptively change with the channel conditions that the MAC continually analyzes.

→ The automatic request repeat mechanism, also called ARQ. Given the cyclic redundancy check associated with each transport block, the MAC determines which transport blocks contain errors and organizes their retransmission.

8.9.9.1 Information Element Groups

Some of the information that the MAC conveys via the physical layer is in the control information and signal field. However, much more information must be made available in the packet such as the following:

→ The resource grid organization or scheduling of transport blocks in both payload *A* and *B*.
→ Information regarding the link quality of the last packet received.
→ Broadcast messages.
→ Service request messages.
→ Information regarding automatic request and repeat processes.
→ MAC addresses.

This information is organized into information element groups, which are positioned at the start of *payload A*. Note that the receiver's physical layer can use information from the control information and signal field to completely decode and reproduce the bits belonging to payload *A*. The MAC could wait until this process is completed. In practice, the MAC should begin processing the IEGs as soon as they are available from the physical layer to reduce latency. Note further that in the case that the packet uses *Payload B.* the information in *Payload A* exclusively consists of information element groups.

By default, the information element groups are identified by an eight-bit IE Group ID, providing 256 information elements. The following organization should be followed.

IEG Indices	Explanation
0 - 128	The following indices are reserved for information element groups defined by this and future specifications. All implementation of FlexLink must support these information element groups.
128-254	The following indices are available for user defined information element groups that customize the FlexLink implementation.
255	Reserved ID for future extensions

Among the different information element groups, there are some that are mandatory and must appear in all packets, whereas others are optional. The following table provides information elements that must be present.

Table 8-41: Table of Required Information Element Groups

IEG Index	Information Element Group Name	Status	Explanation
0	MAC Source Header	Required	Basic information about the transmitting terminal.
1	Payload A Layout	Required	Specifies the layout of transport and data blocks in payload A.
2	Broadcast Message	Optional/R	Indicates intent to connect and defines terminal capabilities. Either IEG2 or IEG3 must be present. Both may be present.
3	MAC Destination Header	Optional/R	Enumerate terminal information, for which this packet carries transport blocks. Either IEG2 or IEG3 must be present. Both may be present.

Table 8-42: Table of Optional Information Element Groups

IEG Index	Information Element Group Name	Status	Explanation
4	Payload B Configuration	Optional	Specifies the layout and configuration of each transport block mapped to *payload B.*
5	SNR Report	Optional	Reports the signal to noise ratio of every resource block for a particular received packet.
6	CIR Report	Optional	Reports the channel impulse response of for a received packet.
7	Service Request	Optional	Requests certain services from the receiving terminal, such as SNR or CINR reports.

<u>Required vs Optional Information Element Groups</u>

The *MAC Source Header*, IEG 0, and *Payload A Layout*, IEG 1, must always be present in *payload A.*

→ If the terminal wishes to connect to another FlexLink transceiver, it will need to provide the Broadcast Message, IEG 2, in addition to the first two IEGs.

→ If the transmitted packet contains user data for a receiving terminal, it will have to provide the *MAC Destination Header* IEG.

→

This page will fill up all on its own as we continue to define the IEGs below.

8.9.9.2 Required Information Element Groups

The MAC Source Header (IEG 0)

This information element group must be the first sequence of bits appearing in *payload A*. All other information element groups feature an ID and may therefore appear out of order. The primary pieces of information of the MAC Source Header are the encryption key, MAC source address, the TX Packet ID and the number of information octets occupied by the information element groups.

Table 8-43: IEG 0 - MAC Header (Required)

Information Element	Bit Width	Explanation
EncryptionKey	[15:0]	A 16-bit sequence that may be used to obscure all or some of the information in this packet.
MacSourceAddress	[23:0]	The MAC address of the terminal transmitting this packet
TxPacketId	[11:0]	An ID used by the transmit MAC to coordinate the packet stream
ManufacturerId	[11:0]	Identify the Manufacturer
VersionNumber	[11:0]	Software Versioning Information
NumberOfInformation Octets	[11:0]	The total number of bytes taken up by MAC information element groups prior to the start of user data inside the first transport block.
	11	Total number of bytes in IEG

The *EncryptionKey* can be used by the transmitting terminal to indicate how information in *payloads A* and *B* are obscured. If this key has a value of 0, no encryption is used. Encryption is left to each implementation and is not discussed in this specification.

The *TxPacketId* is an overflowing count that increments with every packet sent by this terminal. The ID allows the receiver to detect whether it has missed any packets.

The *ManufacturerId* and *VersionNumber* are not yet defined.

The *NumberOfInformationOctets* represents the number of bytes taken up by all information element groups in *payload A*. The total number of bytes in the first transport block must be equal to or larger than this number.

Payload A Layout (IEG 1)

This IEG indicates the number of transport blocks located in *Payload A* as well as the number of data blocks in each of these transport blocks. The number of bits in each data block is determined by parameters CBS_A_FLAG, FEC_A_FLAG, and NDB_A in the signal field. The number of data blocks listed in the information element group below must be equal to NDB_A.

$$NDB\_A = \sum_{n=0}^{NTB-1} NDB[n]$$

Table 8-44: IEG 1 - The Payload A Layout (Required)

Information Element	Bit Width	Explanation
IE Group ID	[7:0] = 1	The ID of this Information Element Group
NTB	[7:0]	The number of transport blocks in payload A
NDB[0]	[7:0]	The number of data blocks in Transport Block 0
:	:	:
NDB[NTB-1]	[7:0]	The number of data blocks in Transport Block NTB-1
	NTB + 2	Total number of bytes in IEG

The code below determines the data block size based on the coding rate of the LDPC mechanism as well as its valid output sizes. It then ensures that the number of code blocks agree with what was provided in the signal field and then goes on to compute the total number of data bits in Payload A.

```
CBS_A_FLAG = 0     # Code Block Sizes:  0 -> 648, 1 -> 1296, 2 -> 1944, 3 -> 1944
FEC_A_FLAG = 0     # LDPC coding rates: 0 -> 1/2, 1 -> 2/3,  2 -> 3/4, 3 -> 5/6
NDB_A      = 10    # The number of data blocks to be sent in the payload A

# The data block size given the code block size and the LDPC coding rate
DataBlockSizes = np.array([[324,  648,  972,  972],
                           [432,  864, 1296, 1296],
                           [486,  972, 1458, 1458],
                           [540, 1080, 1620, 1620]], np.int32)

# The number of data blocks in each transport block of payload A
NDB              = np.array([1, 2, 3, 4], np.int32)
assert np.sum(NDB) == NDB_A, 'Invalid number of data blocks'

# The number of transport blocks to be sent in the payload A
NTB              = len(NDB)

# Iterate over all data blocks
TotalNumberDataBits = 0
for DataBlockIndex in range(0, NDB_A):
    TotalNumberDataBits += DataBlockSizes[FEC_A_FLAG, CBS_A_FLAG]
```

<u>*Broadcast Message (IEG 2):*</u>

To allow FlexLink transceivers with different capabilities to establish a communication link, this *Broadcast Message* IEG is provided. It informs a potential peer of its request to connect and its capabilities. This IEG is also used to respond to the request to connect.

Table 8-45: IEG 2 - The Broadcast Message (Optional)

Information Element	Bit Width	Explanation
IE Group ID	[7:0] = 2	The ID of this Information Element Group
Available Reference Symbol Spacing	[2:0]	A spacing of 3 OFDM symbols is always supported Bit 0 (LSB) → Supports spacing of 1 OFDM Symbols Bit 1 → Supports spacing of 6 OFDM Symbols Bit 2 → Supports spacing of 12 OFDM Symbols
Available Reference Signal Spacing	[2:0]	A spacing of 3 subcarriers is always supported Bit 0 (LSB) → Supports spacing of 6 Subcarriers Bit 1 → Supports spacing of 12 Subcarriers Bit 2 → Supports spacing of 24 Subcarriers
Support for Service Field Formats	[2:0]	Bit 0 → 0/1 Supports signal field format 2 Bit 1 → 0/1 Supports signal field format 3 Bit 2 → 0/1 Supports signal field format 4
Vertical Mapping Support Payload A	[0]	0 → Only static mapping is supported 1 → Both static and adaptive mapping is supported
Supports Payload B	[0]	0 → Supports processing of payload A only 1 → Supports processing of payload A and B
QAM Support	[1:0]	Bit 0 → 0/1 Does not / Does support 256 QAM Bit 1 → 0/1 Does not / Does support 1024 QAM
Preferred Terminal Mode	[0]	0 → This is a client or peer terminal 1 → This is a service terminal
Access Token	[23:0]	A 24-bit access token
Direction	[0]	Forward (0) – This is the original request for connection Reverse (1) – This is the response to the request
Reserved	[0]	Reserved for future use
	6	Total number of bytes in IEG

Not all FlexLink implementation may be fully featured, and this IEG enumerates the capabilities of the transceiver. However, there is a baseline set of abilities that all FlexLink implementation must support.

→ Reference symbol spacing of 3 OFDM symbols.

→ Reference signal spacing of 3 subcarriers.

→ It must be able to process payload A via static vertical mapping.

→ It must be able to process signal field format 1.

→ It must support the BPSK, QPSK, 16-QAM, and 64-QAM constellations.

Note 1: If *payload B* processing is supported, then all vertical and block mapping mechanisms must be supported for *payload A* and *B*. This flag will override the content of the *Vertical Mapping Support Payload A* information element.

Note 2: The access token is provided as a security feature. In the forward direction, the token will indicate to the receiving terminal whether the transmitting terminal is an accessible peer. The receiving terminal must respond with its own *Broadcast Message* and an appropriate *Access Token* for the connection to proceed. Each manufacturer may use their own mechanism to process the 24-bit access tokens. If the token included during the original request for connection is equal to zero, then the connection is public, and any peer may respond to the request for access.

<u>The MAC Destination Address Header (IEG 3)</u>

The MAC destination header enumerates the terminals, for which information is provided in this packet.

→ If more than one MAC destination address is provided, then this packet is sent by a service terminal and *payload B* is being used to talk to all clients. In that case, the *Payload B Configuration* IEG must be provided as well.

→ If only one MAC destination address is provided, then the transport blocks for the single peer may be located in either *payload A* or *payload B*. If the *Payload B Configuration* IEG is provided, then the transport blocks of the single peer are located in *payload B*.

→ The time stamp indicates the reference clock count when the maximum peak of preamble B was detected for the packet with the given RX packet ID.

→ **PeerID** – Each peer, for which we provide a MAC Destination Address also features a PeerID. This PeerID starts at 0 and increments for every successive MAC Destination Address provided.

Table 8-46: IEG 3 – MAC Destination Address Header (Optional)

Information Element	Bit Width	Explanation
IE Group ID	[7:0] = 3	The ID of this Information Element Group
NumPeersIndex	[7:0]	0 → Indicates a single peer with data in Payload A 1 → Indicates a single peer with data in Payload B 2– 255 → Indicates 2-255 peers with data in Payload B
MAC Destination Address	[23:0]	MAC Address of the receiving terminal (PeerId = 0)
RX Packet ID	[11:0]	The Packet ID of the last packet received from the terminal with the above MAC address.
Time Stamp	[13:0]	The local time stamp when it received the above packet
Average SNR (SNR_A)	[5:0]	A signed 6-bit integer indicating the average SNR for the above packet as Average SNR (dB) = -10 + SNR_A·0.75.
MAC Destination Address	[23:0]	The MAC Address of receiving terminal (PeerId = 1)
RX Packet ID	[11:0]	The Packet ID of the last packet received from the terminal with the above MAC address
Time Stamp	[13:0]	The local time stamp when it received the above packet
Average SNR (SNR_A)	[5:0]	A signed 6-bit integer indicating the average SNR for the above packet as Average SNR (dB) = -10 + SNR_A·0.75.
:	:	:
7· NumPeers + 2		Total number of bytes in IEG

8.9.9.3 Optional Information Element Groups

The following information element group provides the organizational information regarding the transport blocks' positioning and encoding in *payload B*. The number of terminals, for which information is provided in *payload B*, can be seen in the NumPeersIndex information element of IEG3.

<u>*Payload B Configuration (IEG 4)*</u>

Information Element	Bit Width	Explanation
IE Group ID	[7:0] = 4	The ID of this Information Element Group
RBSI	[7:0]	The resource block start index.
NRB	[7:0]	The number of resource blocks.
TBO	[2:0]	The time block offset in OFDM symbols.
TBSI	[10:0]	The time block start index.
NTB	[7:0]	The number of time blocks
CBS_A_Flag	[1:0]	[0,1,2,3] Code block size (648, 1296, 1944 bits, 1944)
FEC_A_Flag	[1:0]	(0, 1, 2, 3) LDPC coding rates ½, 2/3, ¾, 5/6
RM_A_Flag	[2:0]	This flag indicates the rate matching factor C2. [0, 1, 2, 3, 4, 5, 6, 7] C2 = 0.0, 0.5, 0.75, 1, 3, 7, 15, 31
BPS_A_Flag	[2:0]	[0, 1, 2, 3, 4, 5, 6, 7] = BPSK, QPSK, 16QAM, 64QAM, 256QAM, 1024QAM, 1024QAM, 1024QAM
RBSI	[7:0]	The resource block start index.
NRB	[7:0]	The number of resource blocks.
TBO	[2:0]	The time block offset in OFDM symbols.
TBSI	[10:0]	The time block start index.
:	:	:

Signal to Noise Ratio Reporting (IEG 5)

Information Element	Bit Width	Explanation
IE Group ID	[7:0] = 5	The ID of this Information Element Group
RX Packet ID	[11:0]	The Packet ID for which this SINR Report is valid.
Number Resource Blocks **NRB**	[7:0]	The number of resource blocks for which we will report SNR deltas below. This can be 0, in which case only the average SNR is reported.
SNR_RB 0	[4:0]	The SNR for resource block 0 = -10 + 1.5*SNR_RB Min = -10dB, Max = 36.5dB
:	:	:
SNR_RB NRB-1	[4:0]	The SNR for resource block NRB-1