CU InSpace

Radio Packet Format

2021-10-10

Samuel Dewan

DRAFT

Contents

1 Introduction

To do.

2 Packet Layout

CU InSpace's radio packets consist of a packet header followed by one or more segments called blocks. Each block has its own header followed by a variable amount of data. All headers and data segments have a length that is a multiple of four bytes. The layout of a packet is shown in figure 2.1.

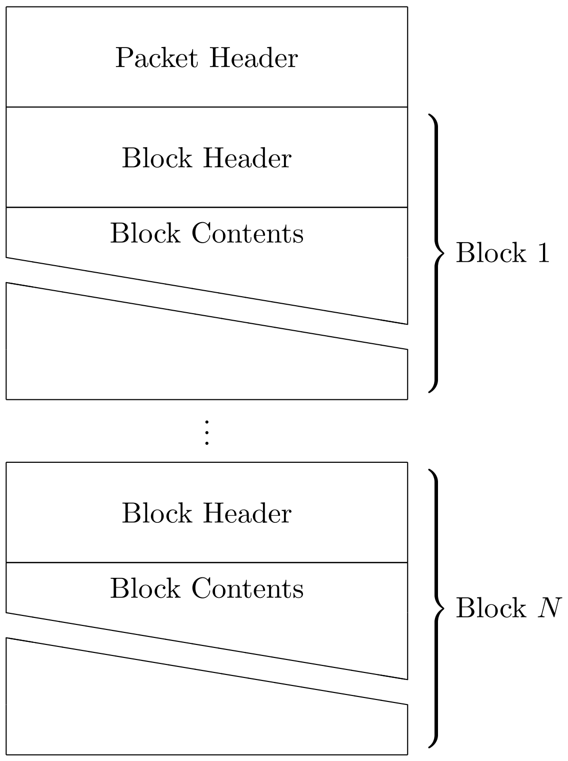

Figure 2.1: Layout of radio packet

2.1 Packet Header

Every packet is preceded by a 12 byte header. The header contains an amateur radio call sign and the packets total length as well as other information which describes the packet as a whole. The packet header follows the format described in figure 2.2.

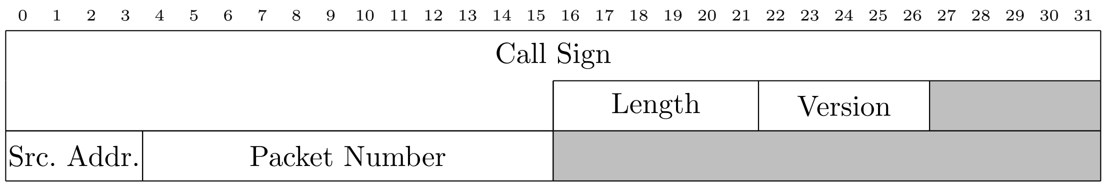

Figure 2.2: Packet header format

Address	Device
0x0	Ground Station
0x1	Rocket
0x2 through 0xE	Reserved for future use
0xF	Multicast

Table 2.1: Device addresses

Call Sign The first six bytes of the packet must contain an amateur radio callsign in ASCII. If the call sign being used is less than size bytes long, the call sign will be padded with NUL characters at the end.

Length The *Length* field indicates the total length of the packet in bytes. This includes the packet header.

The length of a packet must always be a multiple of four bytes. Because of this the *Length* field is encoded in units of four bytes per least significant bit. Because a packet with a length of zero is not possible, the value in the *Length* field is one less than the length of the packet divided by four. This results in a possible range of packet lengths of four to 256 bytes in multiples of four (though a valid packet must have a length of at least twelve bytes).

The folowing equations show how the length of a packet in bytes can be converted to the corrisponding *Length* value and vice versa.

$$\begin{aligned}\text{Length} &= \frac{\text{packet length in bytes}}{4} - 1 \\ \text{packet length in bytes} &= (\text{Length} + 1) \cdot 4\end{aligned}$$

Version The *Version* field indicates what version of the radio packet format the packet adheres to. The receiver should not attempt to parse a packet that has a version number which it does not recognize. The *Call Sign* and *Version* fields will remain for all future versions of this packet format, all other fields could change.

Source Address This field indicates the source of the packet. Possible device addresses are listed in table 2.1. 0xF is not a valid source address.

Packet Number This field contains a count value that should be incremented with every packet transmition from a given device. The *Packet Number* field together with the *Source Address* field form the packet's *Deduplication Code*. See section 4.1 for more information on deduplication.

2.2 Block Header

Every block starts with a four byte block header. This header indicates the type of information contained in the block as well as the blocks's intended recipient. The block header is formated as described in figure 2.3.

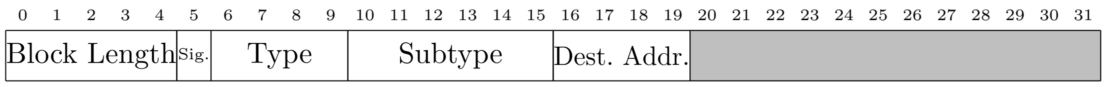

Figure 2.3: Block header format

Value	Block Type
0x0	Control
0x1	Command
0x2	Data
0x3 through 0xF	Reserved for future use

Table 2.2: Block types

Block Length The *Block Length* indicates how many bytes are in the block, including the block header. The offset of the the block after the current block is equal to the sum of the offset of the current block and the current blocks length.

The length of a block must always be a multiple of four, therefore the *Block Length* field is encoded in units of four bytes per least significant bit. Because a valid block will always contain at least 4 bytes, the value of the *Block Length* field is one less than the length of the block divided by four. This means that the field has a range from four bytes to 128 bytes in four byte increments.

The folowing equations show how the length of a block in bytes can be converted to the corrisponding *Length* value and vice versa.

$$\text{Block Length} = \frac{\text{block length in bytes}}{4} - 1$$
$$\text{block length in bytes} = (\text{Block Length} + 1) \cdot 4$$

If the length of a block is such that some or all of the block would exceed the total packet length, then the block is invalid and will be ignored.

Has Signature (Sig.) The *Has Signature* bit is used for cryptographic signing. See section 6 for more details.

Type The block *Type* field indicates the purpose of the block. Possible values are listed in table 2.2. Block types are discussed in more detail in section 3.

Subtype The *Subtype* indicates the type of information or request contained in the block. The possible block subtypes for each block type are listed and discussed in more detail in section 3.

Destination Address This field indicates what device the block is intended for. Received blocks which are intended for a different device should be ignored. Possible addresses are listed in table 2.1. A destination address of 0xF indicates that the block is intended for anyone who receives it.

3 Block Formats

Blocks have a type which indicates their purpose. The possible block types are listed in table 2.2. This section describes the possible block subtypes and their associated data formats.

3.1 Control Blocks

Command blocks contain instructions or requests for information. The possible subtypes for command blocks are listed in table 3.1.

Value	Description
0x0	Signal report
0x1	Command Acknowledgment
0x2	Command Nonce Request
0x3	Command Nonce
0x4	Beacon
0x5	Beacon Response
0x6 through 0x3F	Reserved for future use

Table 3.1: Control Block Subtypes

3.1.1 Signal Report

A signal report contains information designed to allow the recipient to assess the quality of the radio link. Signal reports may be sent at any time by any party and should contain information pertaining to the last packet received from the block's destination device.

Signal reports follow a query response structure. An initial signal report that is marked as a request is sent first. The recipient of this request then responds with a signal report that is not marked as a request.

The format of the payload data for a signal report is described in figure 3.1.

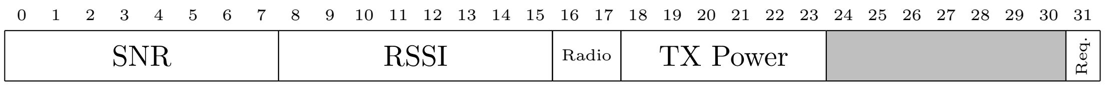

Figure 3.1: Signal Report Payload Format

SNR Signal to Noise Ratio for the last packet received from the destination device in units of 1 dB/LSB. This value is encoded as a one byte signed integer in twos compliment format.

In a response signal report this should always be the SNR for the packet that contained the request.

RSSI Received Signal Strength Indication for the last packet received from the destination device in units of 1 dB/LSB. This value is encoded as a one byte signed integer in twos compliment format

In a response signal report this should always be the RSSI for the packet that contained the request.

Radio The index of the radio used by the device that send the requesting signal report. This field is not meaningful to the recipient of the request, the recipient simply echos it back to the sender of the request in the response signal report. This field is encoded as a two bit unsigned integer.

TX Power The transmit power with which the packet containing the signal report block was sent in units of 1 dB/LSB. This value is encoded as a six bit signed integer in twos compliment format

Request When set, this bit indicates that this is a request signal report and that the receiver should respond. If this bit is not set then this block is a response signal report.

3.1.2 Beacon

The beacon frame is used while searching for a peer device. See section 5 for more information on search mode operation. This block subtype does not have any payload data.

3.2 Command Blocks

Command blocks contain instructions or requests for information. The possible subtypes for command blocks are listed in table 3.2.

Value	Description
0x0	Reset rocket avionics
0x1	Request telemetry data
0x2	Deploy parachute
0x3	Tare sensors
0x4 through 0x3F	Reserved for future use

Table 3.2: Command Block Subtypes

3.2.1 Reset Rocket Avionics

When this block is received by the rocket avionics the rocket's microcontroller will be reset. If this block's ACK requested bit is set the acknowledgment will be sent before the avionics reset. The acknowledgments for any other blocks in the same packet as the reset command are not guaranteed to be sent. This block subtype does not have any payload data.

3.2.2 Request Telemetry Data

This command is used to explicitly request that the rocket avionics sent particular telemetry data back. Up to four different telemetry data blocks can be requested per command. The payload data for the request telemetry data command is made up of four bytes, each of which may indicate a requested data block subtype. The format of the request block payload is described in figure 3.2.

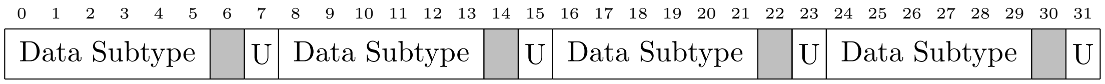

Figure 3.2: Telemetry Data Request Payload Format

Data Subtype The subtype of data block that is being requested.

Used (U) For each of the four bytes in the request telemetry packet's payload this bit indicates whether the byte contains a valid request. Any bytes for which this bit is not set should be ignored.

3.2.3 Deploy parachute

This command indicates to the rocket avionics that it should immediately deploy the drogue parachute. This block subtype does not have any payload data.

3.2.4 Tare sensors

This command tells the rocket to zero out all of its sensors. This command should only be send when the rocket is on the pad ready to launch. The rocket avionics can assume that when it receives this command it is stationary and in the correct orientation for launch.

3.3 Data Blocks

Data blocks contain telemetry data. The possible subtypes data blocks are listed in table 3.3.

3.3.1 Debug Message

Debug messages are string messages intended to provide human readable debugging output. They are encoded as UTF-8 strings and do not require a terminating NUL character because the string length can be extrapolated from the block length. Every debug message is preceded by a 32 bit unsigned mission time value.

Note that like all other data blocks debug messages must be a multiple of four bytes long. If the string to be logged is not a multiple of four bytes it must be padded with NUL characters at the end to make it fit the required alignment. The length recorded in the block header must be a multiple of four, and therefore must include any NUL padding bytes at the end of the string.

Value	Description
0x00	Debug message
0x01	Status
0x02	Startup message
0x03	Altitude
0x04	Acceleration
0x05	Angular velocity
0x06	GNSS location
0x07	GNSS metadata
0x08	Power information
0x09	Temperatures
0x0A	MPU-9250 IMU data
0x0B	KX134-1211 accelerometer data
0x0C through 0x3F	Reserved for future use

Table 3.3: Data Block Subtypes

3.3.2 Altitude

The altitude block is used to convey altimeter data. This block contains both the altitude value calculated by the rocket and the raw pressure and temperature values if they are available. The format of the altitude block payload is described in figure 3.3.

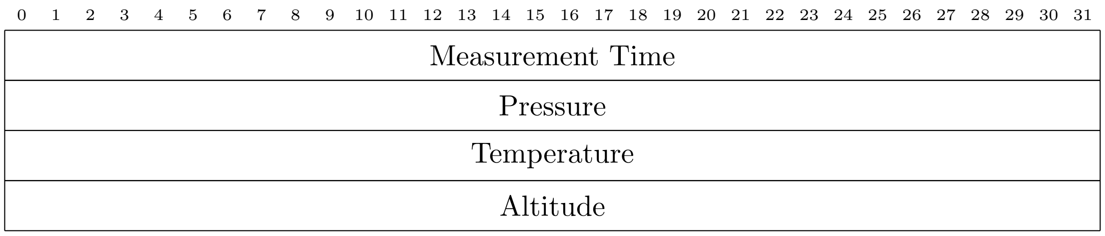

Figure 3.3: Altitude Data Payload Format

Measurement Time The mission time when the measurement was taken.

Pressure The measured pressure in units of Pascals. This field is a signed 32 bit integer in two's complement format.

Temperature The measured temperature in units of 1 millidegree Celsius/LSB. This field is a signed 32 bit integer in two's complement format.

Altitude The calculated altitude in units of 1 mm/LSB. This field is a signed 32 bit integer in two's complement format.

3.3.3 Acceleration

The acceleration block is used to convey generic 3-axis accelerometer data. This block is intended to abstract over the details of any individual accelerometer. The format of the acceleration block payload is described in figure 3.4.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Measurement Time																															
Full Scale Range								▨								X-Axis															
Y-Axis																Z-Axis															

Figure 3.4: Acceleration Data Payload Format

Measurement Time The mission time when the measurement was taken.

Full Scale Range The full scale range of the accelerometer in g. This value represents the maximum and minimum acceleration that can be measured, for example a value of 16 in this field represents a full scale range of $\pm 16\,\mathrm{g}$. If the resolution of the accelerometer is less than 16 bits the full scale range must be adjusted to compensate, for example a 10 bit accelerometer with a $\pm 2\,\mathrm{g}$ full scale range would use the value 64 for this field.

This value can be used to calculate the accelerometer sensitivity, which in turn can be used to convert the acceleration measurements for each axis into useful units. For a measurement m with a full scale range f the acceleration in g can be found like this:

$$\text{acceleration} = m \cdot \frac{f}{2^{15}}$$

***-Axis** These fields represent the acceleration measurements for each axis. These fields are signed integers in two's complement format. If the resolution of the accelerometer is less than 16 bits the values must be sign extended.

3.3.4 Angular Velocity

The angular velocity block is used to convey generic 3-axis gyroscope data. This block is intended to abstract over the details of any individual gyroscope. The format of the acceleration block payload is described in figure 3.5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Measurement Time																															
Full Scale Range																X-Axis															
Y-Axis																Z-Axis															

Figure 3.5: Angular Velocity Data Payload Format

Measurement Time The mission time when the measurement was taken.

Full Scale Range The full scale range of the gyroscope in degrees per second. This value represents the maximum and minimum angular velocity that can be measured, for example a value of 2000 in this field represents a full scale range of ±2000 degreespersecond. If the resolution of the gyroscope is less than 16 bits the full scale range must be adjusted to compensate, for example a 12 bit gyroscope with a ±500 degreepersecond full scale range would use the value 4000 for this field.

This value can be used to calculate the gyroscope sensitivity, which in turn can be used to convert the acceleration measurements for each axis into useful units. For a measurement m with a full scale range f the angular velocity in degrees per second can be found like this:

$$\text{angular velocity} = m \cdot \frac{f}{2^{15}}$$

***-Axis** These fields represent the angular velocity measurements for each axis. These fields are signed integers in two's complement format.

3.3.5 GNSS Location

The format of the GNSS location block is described in figure 3.6.

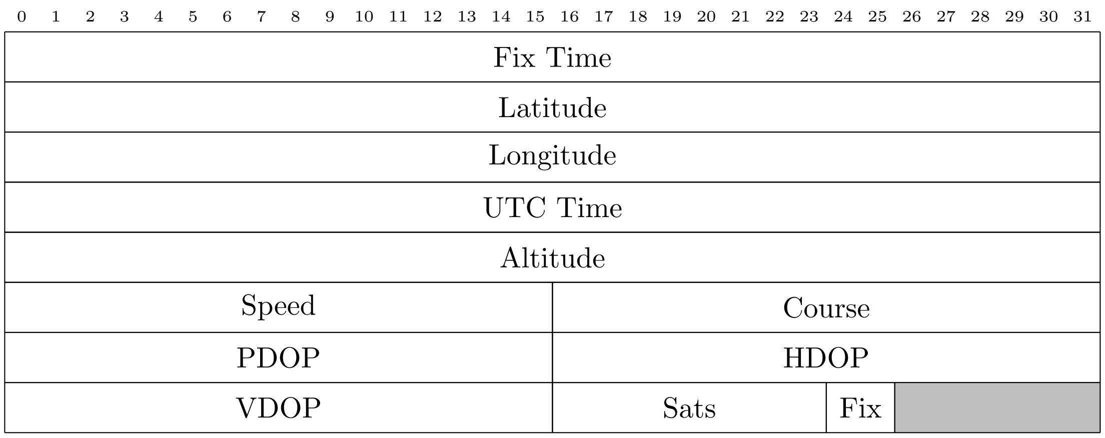

Figure 3.6: GNSS Location Data Payload Format

Fix Time The mission time when the fix was received.

Latitude The latitude of the rocket in units of $100\,\mu'$/LSB (micro-arcminutes per least significant bit). This field is a signed 32 bit integer in two's complement format.

Longitude The longitude of the rocket in units of $100\,\mu'$/LSB (micro-arcminutes per least significant bit). This field is a signed 32 bit integer in two's complement format.

UTC Time The UTC time when the fix was received in seconds since the Unix epoch.

Altitude The altitude as calculated by the GNSS module in units of millimetres above sea level. This field is a signed 32 bit integer in two's complement format.

Speed The speed over ground of the rocket in hundredths of a knot. This field is a signed 16 bit integer in two's complement format.

Course The course over ground of the rocket in hundredths of a degree. This field is a signed 16 bit integer in two's complement format.

PDOP Position Dilution of Precision * 100.

HDOP Horizontal Dilution of Precision * 100.

VDOP Vertical Dilution of Precision * 100.

Sats The number of GNSS satellites used in the fix.

Fix The fix type as shown in table 3.4.

Fix Type Value	Description
0x0	Unknown
0x1	Not available
0x2	2D fix
0x3	3D fix

Table 3.4: GNSS Fix Type Values

3.3.6 GNSS Metadata

The format of the GNSS location block is described in figure 3.7. The length of this block varies, after the GLONASS satellites in use bitfield 4 bytes of information is encoded for each satellite in view of the GNSS module. Note that not all satellites which are in view will also be in use.

Mission Time The mission time when the fix was received.

GPS Satellites in Use This bitfield indicates which GPS satellites are used in the current fix. Each bit position represents a GPS pseudo-random noise sequence.

GLONASS Satellites in Use This bitfield indicates which GLONASS satellites are used in the current fix. Each bit position represents a slot number. Bit zero corresponds to slot 65, bit 1 to slot 66 and so on.

Elevation This field represents the elevation for a GNSS satellite in degrees.

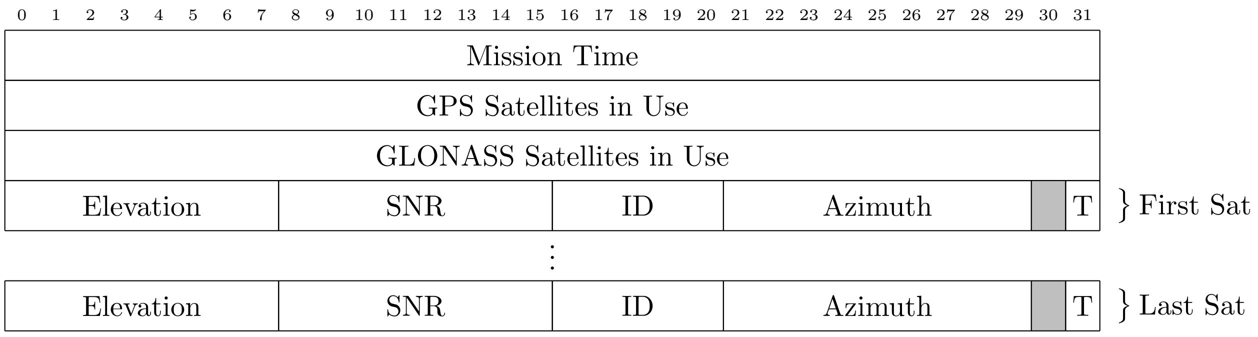

Figure 3.7: GNSS Metadata Payload Format

SNR This field represents the signal to noise ratio for a signal from a satellite in dB Hz.

ID This field contains the satellite pseudo-ransom noise sequence for GPS satellites or the satellite ID for GLONASS satellites.

Azimuth This field contains the satellite azimuth in degrees.

T This field encodes the satellite type, it is 0 for GPS satellites or 1 for GLONASS satellites.

3.3.7 MPU9250 IMU Data

The format of the MPU9250 IMU data block is described in figure 3.8. This block can contain a variable amount of acceleration, angular velocity and magnetic flux density data. The format of each individual data entry is described in figure 3.9. Note that since each entry is 21 bytes long entries after the first one may not be properly aligned.

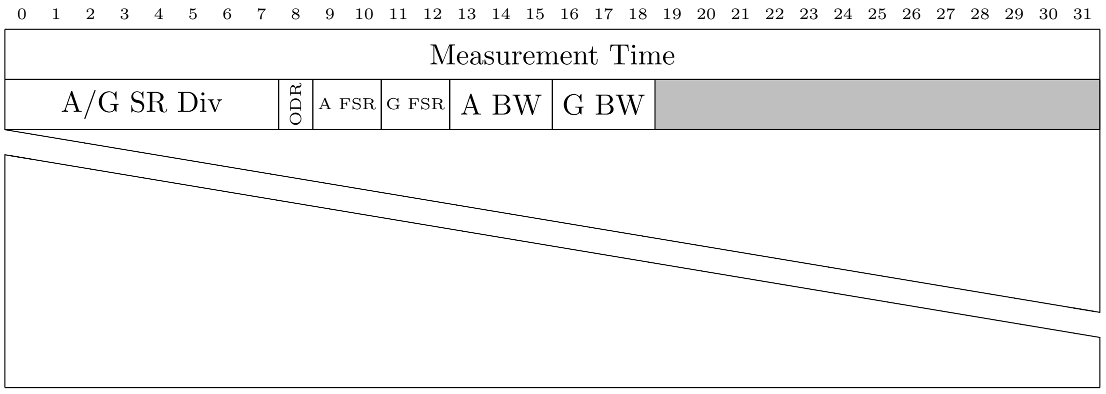

Figure 3.8: MPU9250 Inertial Measurement Unit Data Payload Format

Measurement Time The time of the last measurement in the block.

A/G SR Div The accelerometer/gyroscope sample rate divisor field can be used to calculate the sample rate for the accelerometer and gyrosope. The folowing equation can be used to find the sample rate:

$$\text{sample rate} = \frac{1000}{\text{A/G SR Div} + 1}$$

MS This field encodes the sample rate for the magnetometer. The possible values for this field are shown in table 3.5.

MS Value	Sample Rate
0x0	8 Hz
0x1	100 Hz

Table 3.5: MPU9250 Magnetometer Sample Rate Values

A FSR This field encodes the full scale range for the accelerometer. The possible values for this field are shown in table 3.6.

Accel. FSR Value	Full Scale Range
0x0	±2 g
0x1	±4 g
0x2	±8 g
0x3	±16 g

Table 3.6: MPU9250 Accelerometer Full Scale Range Values

G FSR This field encodes the full scale range for the gyroscope. The possible values for this field are shown in table 3.7.

Gyro. FSR Value	Full Scale Range
0x0	$\pm 250\,^{\circ}\,\mathrm{s}^{-1}$
0x1	$\pm 500\,^{\circ}\,\mathrm{s}^{-1}$
0x2	$\pm 1000\,^{\circ}\,\mathrm{s}^{-1}$
0x3	$\pm 2000\,^{\circ}\,\mathrm{s}^{-1}$

Table 3.7: MPU9250 Gyroscope Full Scale Range Values

A BW This field encodes the accelerometer low pass filter bandwidth. The possible values for this field are shown in table 3.8.

Accel. BW Value	Low Pass Filter 3 dB Bandwidth
0x0	5.05 Hz
0x1	10.2 Hz
0x2	21.2 Hz
0x3	44.8 Hz
0x4	99 Hz
0x5	218.1 Hz
0x6	420 Hz

Table 3.8: MPU9250 Accelerometer Low Pass Filter Bandwidth Values

G BW This field encodes the gyroscope low pass filter bandwidth. The possible values for this field are shown in table 3.9.

Gyro. BW Value	Low Pass Filter 3 dB Bandwidth
0x0	5 Hz
0x1	10 Hz
0x2	20 Hz
0x3	41 Hz
0x4	92 Hz
0x5	184 Hz
0x6	250 Hz

Table 3.9: MPU9250 Gyroscope Low Pass Filter Bandwidth Values

Accel. * The acceleration data fields are big endian values encoded in two's complement.

Gyro. * The angular velocity data fields are big endian values encoded in two's complement.

Mag. * The magnetic flux denisity data fields are little endian values encoded in two's complement.

O This field indicates whether there was an overflow while measuring the magnetic flux denisity. If this bit is set the magnetometer data for the sample is not valid.

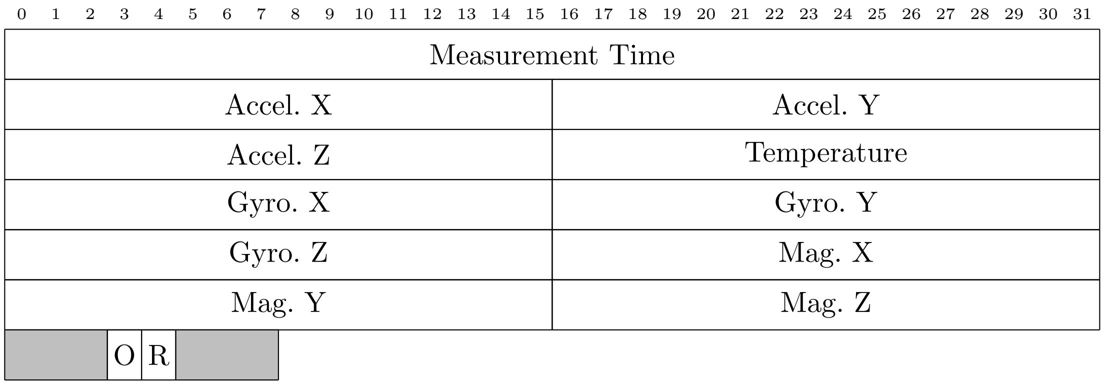

Figure 3.9: MPU9250 IMU Measurement Entry Foramt

R This field indicates the resolution of the magnetometer data. If this bit is set resolution is 16 bits, otherwise it is 14 bits.

3.3.8 KX134-1211 Accelerometer Data

The format of the KX134-1211 accelerometer data block is described in figure 3.10. This block can contain a variable amount of acceleration data. This data will be in sets of three measurements (x, y, z), each measurement is either 8 or 16 bits long and is encoded in two's complement format.

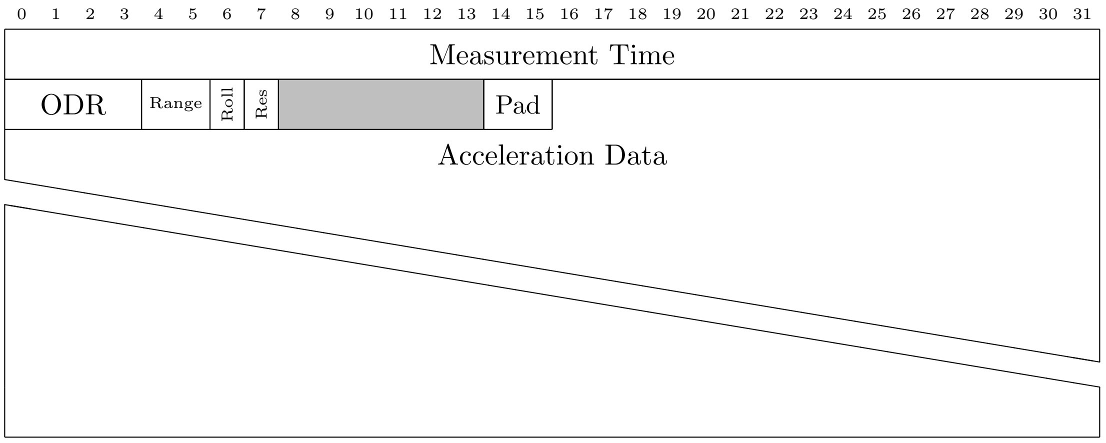

Figure 3.10: KX134-1211 Accelerometer Data Payload Format

Measurement Time The time of the last measurement in the block.

ODR This field encodes the output data rate at which the accelerometer is configured. The possible values for this field are shown in table 3.10.

Range This field encodes the configured full scale range of the accelerometer. The possible values for this field are shown in table 3.11.

ODR Value	Output Data Rate
0x0	0.781 Hz
0x1	1.563 Hz
0x2	3.125 Hz
0x3	6.25 Hz
0x4	12.5 Hz
0x5	25 Hz
0x6	50 Hz
0x7	100 Hz
0x8	200 Hz
0x9	400 Hz
0xa	800 Hz
0xb	1600 Hz
0xc	3200 Hz
0xd	6400 Hz
0xe	12 800 Hz
0xf	25 600 Hz

Table 3.10: KX134-1211 Output Data Rate Values

Range Value	Full Scale Range
0x0	±8 g
0x1	±16 g
0x2	±32 g
0x3	±64 g

Table 3.11: KX134-1211 Full Scale Range Values

Roll This field encode the accelerometer's configured low pass filter rolloff. The possible values for this field are shown in table 3.12.

Res This field encodes the output resolution for the accelerometer. The possible values for this field are shown in table 3.13.

Pad This field indicates the number of padding bytes which present after the end of the acceleration data. This field is important when parsing 8 bit data because it could appear that there is an extra sample due to padding.

Roll Value	Low Pass Filter Corner Frequency
0x0	ODR/9
0x1	ODR/2

Table 3.12: KX134-1211 Low Pass Filter Rolloff Values

Res Value	Resolution
0x0	8 bit
0x1	16 bit

Table 3.13: KX134-1211 Resolution Values

4 Other Considerations

4.1 Deduplication

4.2 Endianness

All fields are little endian unless otherwise indicated.

4.3 Alignment

The start of all data blocks must be alligned to a four byte boundry within the packet. To facilitate this data blocks must be padded to ensure that their lengths are always a multiple of four bytes. Padding bytes should be set to zero when transmiting and their content should be ignored by the receiver.

4.4 Timing

4.4.1 Backoff Period

4.4.2 Timeouts

4.5 Validation and Sanity Checks

4.6 Reserved Fields

Reserved fields should be set to zero when transmitting and ignored when receiving.

5 Dynamic Selection of Radio Settings

5.1 Search Mode

5.2 Hopping

6 Cryptographic Signature Support

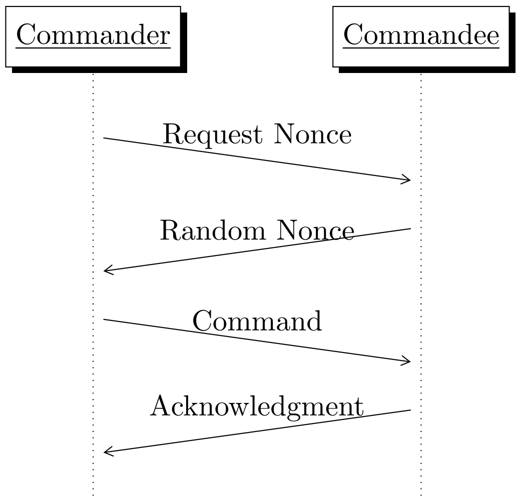

Figure 6.1: Signed command