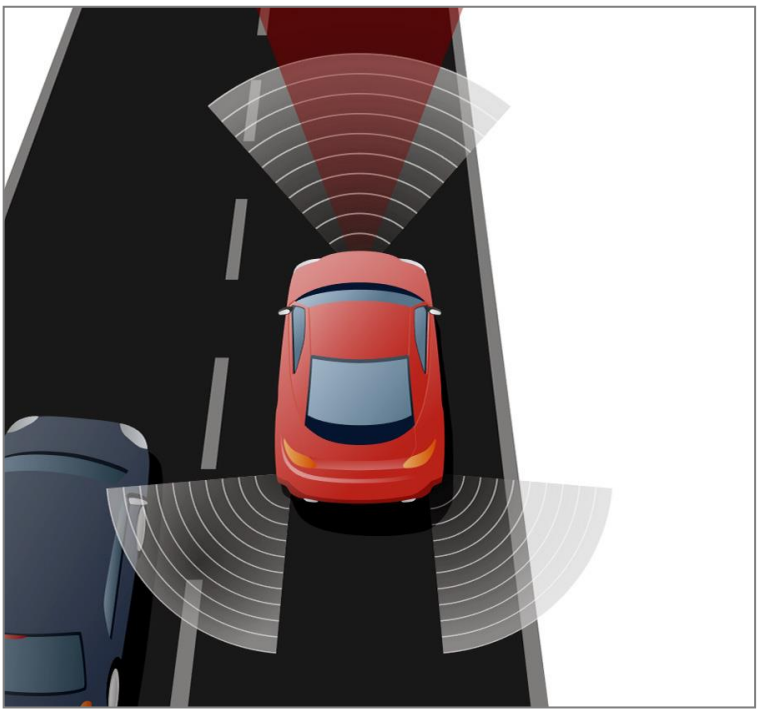
RCOM

Lane position and vehicle-to-vehicle measurement



RCOM Manual

Confidently. Accurately.

Legal notices

Information furnished is believed to be accurate and reliable. However, Oxford Technical Solutions Limited assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Oxford Technical Solutions Limited. Specifications mentioned in this publication are subject to change without notice and do not represent a commitment on the part of Oxford Technical Solutions Limited. This publication supersedes and replaces all information previously supplied. Oxford Technical Solutions Limited products are not authorised for use as critical components in life support devices or systems without express written approval of Oxford Technical Solutions Limited.

All brand names are trademarks of their respective holders.

The software is provided by the contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Copyright notice

Revision

Document revision: 181122 (see revision history for detailed information).

Contact details

Oxford Technical Solutions Limited
Park Farm Business Centre
Middleton Stoney
Oxfordshire
OX25 4AL
United Kingdom

Tel: +44 (0) 1869 814 251

Web: http://www.oxts.com
Email: support@oxts.com

Table of contents

Introduction

RCOM is a data format designed by OxTS for the efficient communication of RT-Range data. The format contains the data required to calculate range measurements between inertial navigation systems and information about lane positioning.

RCOM is transmitted over Ethernet by the RT-Range using a UDP broadcast. The port number is 3003.

It is possible to transmit the packets using other serial interfaces as it contains sufficient information for the software to synchronise the packet. Over Ethernet the packet is synchronised anyway, so the synchronisation information in the RCOM message is not needed.

This manual gives a description of the RCOM format so that custom software can be written for specific needs and applications. The main purpose of this document is for internal use by engineers at OxTS. We do not fully support all of the information in this document to external customers.

Definitions of the binary numbers used

Table 1. Word length definitions

Terminology	Data length
Byte (UByte)	8-bit integer (unsigned)
Short (UShort)	16-bit integer (unsigned)
Word (UWord)	24-bit integer (unsigned)
Long (ULong)	32-bit integer (unsigned)
Float	32-bit IEEE 754 floating-point
Double	64-bit IEEE 754 floating-point

Note: The U prefix indicates a value is unsigned; otherwise it is signed using 2's complement.

In this manual, hexadecimal numbers are written in the C-language convention with "0x" preceding the number. For example 50 decimal will be written as 0x32.

Note: all multi-byte values are sent in little-endian format. This means little-end first, or least significant byte first (LSB), which is compatible with Intel microprocessors.

RCOM packets

Packet type

There are five different types of packet within the RCOM format. Each one contains different information depending on its intended use. The *packet type* of each packet is identified in the header at the start of the packet. Table 2 shows the different types and their identifier.

Table 2. List of RCOM packets

Identifier	Packet type	Table
0x00	Range packet (obsolete)	
0x01	Lane packet	Table 5
0x02	Extended (Ext.) range packet	Table 16
0x03	Wrapped NCOM packet	Table 40
0x04	Trigger time packet	Table 41
0x05	Polygon packet	Table 42
0x06	Multiple sensor point packet	Table 43 and Table 44

Header

Each RCOM packet comprises a header and a data section. The header contains a sync character, a packet type and a length. The data section always has a checksum as the last byte. Table 3 describes the format of the header.

Table 3. RCOM header description

Byte	Description	Type
0	Sync byte (0x57)	UByte
1	Packet type (defined in Table 2)	UByte
2–3	Length of data section (excludes header, includes checksum)	UShort

Care should be taken when writing a decoder for RCOM. OxTS may extend the packets from time to time. When a packet is extended, the checksum will always remain at the

end of the data section. An old decoder can still decode all of the information that it is familiar with, but will have to ignore part of the data at the end of the packet that is new.

Checksum

The checksum for the RCOM packet is computed using modulo 256, and the sum of all characters excluding the Sync byte. Figure 1 shows an example function in C that computes the checksum. The variable `len` passed to this function is the length of the data section in the RCOM packet (bytes 2 to 3).

Figure 1. Checksum C code

```
static unsigned char rcomtx_compute_checksum(unsigned char *pkt,
int len)
{
   int i;
   unsigned char csum = 0x00;

   // The checksum ignores the initial sync byte
   // and the last (checksum) byte
   for (i = 1; i < (len + 3); i++)
      csum += pkt[i];

   return csum;
}
```

Status channels

Many RCOM packets contain status information that is updated at a lower rate than the normal data. To accommodate the lower data rate information, only one status message is contained within each RCOM packet. This means several RCOM packets of the same type need to be received in order to have a complete picture of the status.

While the status channels are exclusive to each RCOM packet type, some of the information contained within them is common across different RCOM packets. For example, the RT-Range Software Dev ID is transmitted in both the lane packet, and extended range packet. The information will be the same in both.

Range packet (obsolete)

The range packet is now obsolete as it only supported one target. This format was retired when multiple targets were introduced.

Lane packet

The lane packet contains measurements from the vehicle to the lane markings. The lane markings are defined in the map file currently loaded into the RT-Range.

The lane packet is described in Table 5, and the status messages are outlined in Table 4.

Table 4. Status channel definitions for lane packets (byte 49)

Value	Status channel information	See
0	GPS coarse time	Table 6
1	RT-Range software development ID	Table 7
2	Map number	Table 8
6	OS and script version	Table 9
7	UTC offset and CPU load	Table 10
8	Point A lever-arm	Table 11
9	Point B lever-arm	Table 12
10	Point C lever-arm	Table 13
15	Command communication status	Table 14

Note: missing channels are not defined and are not currently used.

Table 5. Lane packet

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type	UByte		
2–3	Length of data section[1]	UShort		
4–5	GPS time into minute	UShort	0.001 s	0xFFFF
6	Line number to left of A	UByte		0xFF
7	Line number to right of A	UByte		0xFF
8 ⋮ 11	Distance along lane 1	Long	0.001 m	0x80000000
12–13	Lateral distance to left of A	Short	0.001 m	0x8000
14–15	Lateral velocity to left of A	Short	0.01 m/s	0x8000
16–17	Lateral acceleration to left of A	Short	0.01 m/s²	0x8000
18–19	Lateral distance to right of A	Short	0.001 m	0x8000
20–21	Lateral velocity to right of A	Short	0.01 m/s²	0x8000
22–23	Lateral acceleration to right of A	Short	0.01 m/s²	0x8000
24–25	Lateral distance from Point A to Line 1	Short	0.001 m	0x8000
26–27	Lateral distance from Point A to Line 2	Short	0.001 m	0x8000
28–29	Lateral distance from Point A to Line 3	Short	0.001 m	0x8000
30–31	Lateral distance from Point A to Line 4	Short	0.001 m	0x8000
32–33	Lateral distance from Point A to Line 5	Short	0.001 m	0x8000
34–34	Lateral distance from Point A to Line 6	Short	0.001 m	0x8000
36–37	Lateral distance from Point A to Line 7	Short	0.001 m	0x8000
38–39	Lateral distance from Point A to Line 8	Short	0.001 m	0x8000
40–41	Lateral distance from Point B to line on left of A	Short	0.001 m	0x8000
42–43	Lateral distance from Point C to line on right of A	Short	0.001 m	0x8000
44	Line on left of Point B	UByte		0xFF
45	Line on right of Point B	UByte		0xFF
46	Line on left of Point C	UByte		0xFF
47	Line on right of Point C	UByte		0xFF
48	Reserved			
49	Status channel	UByte		
50 ⋮ 57	*These bytes are specific to each lane status channel message*			
58–59	Lateral velocity from Point A to Line 1	Short	0.01 m/s	0x8000

60–61	Lateral velocity from Point A to Line 2	Short	0.01 m/s	0x8000
62–63	Lateral velocity from Point A to Line 3	Short	0.01 m/s	0x8000
64–65	Lateral velocity from Point A to Line 4	Short	0.01 m/s	0x8000
66–67	Lateral velocity from Point A to Line 5	Short	0.01 m/s	0x8000
68–69	Lateral velocity from Point A to Line 6	Short	0.01 m/s	0x8000
70–71	Lateral velocity from Point A to Line 7	Short	0.01 m/s	0x8000
72–73	Lateral velocity from Point A to Line 8	Short	0.01 m/s	0x8000
74–75	Lateral distance from Point B to Line 1	Short	0.001 m	0x8000
76–77	Lateral distance from Point B to Line 2	Short	0.001 m	0x8000
78–79	Lateral distance from Point B to Line 3	Short	0.001 m	0x8000
80–81	Lateral distance from Point B to Line 4	Short	0.001 m	0x8000
82–83	Lateral distance from Point B to Line 5	Short	0.001 m	0x8000
84–85	Lateral distance from Point B to Line 6	Short	0.001 m	0x8000
86–87	Lateral distance from Point B to Line 7	Short	0.001 m	0x8000
88–89	Lateral distance from Point B to Line 8	Short	0.001 m	0x8000
90–91	Lateral distance from Point C to Line 1	Short	0.001 m	0x8000
92–93	Lateral distance from Point C to Line 2	Short	0.001 m	0x8000
94–95	Lateral distance from Point C to Line 3	Short	0.001 m	0x8000
96 –97	Lateral distance from Point C to Line 4	Short	0.001 m	0x8000
98–99	Lateral distance from Point C to Line 5	Short	0.001 m	0x8000
100–101	Lateral distance from Point C to Line 6	Short	0.001 m	0x8000
102–103	Lateral distance from Point C to Line 7	Short	0.001 m	0x8000
104–105	Lateral distance from Point C to Line 8	Short	0.001 m	0x8000
106–107	Curvature of Line 1	Short	10^{-4} 1 / m	0x8000
108–109	Curvature of Line 2	Short	10^{-4} 1 / m	0x8000
110–111	Curvature of Line 3	Short	10^{-4} 1 / m	0x8000
112–113	Curvature of Line 4	Short	10^{-4} 1 / m	0x8000
114–115	Curvature of Line 5	Short	10^{-4} 1 / m	0x8000
116–117	Curvature of Line 6	Short	10^{-4} 1 / m	0x8000
118–119	Curvature of Line 7	Short	10^{-4} 1 / m	0x8000
120–121	Curvature of Line 8	Short	10^{-4} 1 / m	0x8000
122–123	Curvature of Point A	Short	10^{-4} 1 / m	0x8000
124–125	Curvature of Point B	Short	10^{-4} 1 / m	0x8000
126–127	Curvature of Point C	Short	10^{-4} 1 / m	0x8000
128–129	Heading with respect to line on left of A	Short	0.01°	0x8000
130–131	Heading with respect to line on right of A	Short	0.01°	0x8000

132	Checksum	UByte

Note 1: in earlier versions this packet was shorter in length. Take care in the decoder to only decode values that exist in the data section of the transmitted packet.

Table 6. Lane packet, status channel 0, GPS coarse time

Byte	Description	Type	Unit	Invalid
50 ⋮ 53	GPS time in minutes. Minutes since 6th January 1980	Long	Minute	0x8000000
54 ⋮ 57	Reserved			

Table 7. Lane packet, status channel 1, RT-Range software Dev ID

Byte	Description	Type	Format	Invalid
50 ⋮ 57	RT-Range Software Dev ID 8 chars represent the ASCII text of the software development ID or version	Byte	ASCII	

Table 8. Lane packet, status channel 2, map number

Byte	Description	Type	Unit	Invalid
50	Map number	UByte		
51 ⋮ 57	Reserved			

Table 9. Lane packet, status channel 6, OS and script version

Byte	Description	Type	Unit	Invalid
50	Major OS version	UByte		0xFF
51	Minor OS version	UByte		0xFF
52	OS revision version	UByte		0xFF
53 ⋮ 55	Script version	UWord		0xFFFFFF
56–57	Reserved			

Table 10. Lane packet, status channel 7, UTC offset and CPU load

Byte	Description	Type	Unit	Invalid
50–51	UTC Offset	Short	1 s	0x8000
52–56	Reserved			
57	CPU load	UByte	0.4%	0xFF

Table 11. Lane packet, status channel 8, Point A lever-arm

Byte	Description	Type	Unit	Invalid
50 ⋮ 52	Point A lever-arm (displacement arm) in the *x*-axis of the vehicle frame	Word	0.001 m	0x800000
53 ⋮ 55	Point A lever-arm (displacement arm) in the *y*-axis of the vehicle frame	Word	0.001 m	0x800000
56–57	Point A lever-arm (displacement arm) in the *z*-axis of the vehicle frame	Short	0.001 m	0x8000

Table 12. Lane packet, status channel 9, Point B lever-arm

Byte	Description	Type	Unit	Invalid
50 ⋮ 52	Point B lever-arm (displacement arm) in the *x*-axis of the vehicle frame	Word	0.001 m	0x800000
53 ⋮ 55	Point B lever-arm (displacement arm) in the *y*-axis of the vehicle frame	Word	0.001 m	0x800000
56–57	Point B lever-arm (displacement arm) in the *z*-axis of the vehicle frame	Short	0.001 m	0x8000

Table 13. Lane packet, status channel 10, Point C lever-arm

Byte	Description	Type	Unit	Invalid
50 ⋮ 52	Point C lever-arm (displacement arm) in the *x*-axis of the vehicle frame	Word	0.001 m	0x800000
53 ⋮ 55	Point C lever-arm (displacement arm) in the *y*-axis of the vehicle frame	Word	0.001 m	0x800000
56–57	Point C lever-arm (displacement arm) in the *z*-axis of the vehicle frame	Short	0.001 m	0x8000

Table 14. Lane packet, status channel 15, command communication status

Byte	Description	Type	Unit	Invalid
50–51	UDP command characters received. This value will wrap when it overflows	UShort		
52–53	UDP command packets received. This value will wrap when it overflows	UShort		
54–55	UDP command characters skipped. This value will wrap when it overflows	UShort		
56–57	UDP command errors. This value will wrap when it overflows	UShort		

Extended range packet

The extended range packet contains range measurements of the targets relative to the hunter. One packet is used for each target. The extended range packet is described in Table 16 and its status channels are summarised in Table 15.

Table 15. Status channel definitions for extended range packets (byte 41)

Value	Status channel information	See
0	Latency measurement	Table 17
1	RT-Range software development ID	Table 18
2	Target serial communications status	Table 19
3	Target wireless LAN communications status	Table 20
4	Ext. range packet, status channel 4, hunter Ethernet communication status	Table 21
5	Ext. range packet, status channel 5, range output latency and offset measurement	Table 22
6	OS and script version	Table 23
7	UTC offset, range configuration and CPU load	Table 24
8	Fixed point position	Table 25
9	Hunter and target IP addresses	Table 26
10	Fixed point altitude	Table 27
11	RT-Range local co-ordinates origin, latitude and longitude	Table 28
12	RT-Range local co-ordinates origin, altitude and heading	Table 29
13	Range hunter lever arm	Table 30
14	Range-target lever arm	Table 31
15	Command communication status	Table 32
16	Range accuracy	Table 33
17	Target vehicle geometry	Table 34
18	Acceleration filter settings	Table 35
19	Extrapolation filter settings	Table 36
20	Feature point position	Table 37
21	Feature point altitude and heading	Table 38
22	Hunter vehicle geometry	Table 39

Table 16. Extended range packet

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type	UByte		
2–3	Length of data section[1]	UShort		
4–5	GPS time into minute	UShort	0.001 s	0xFFFF
6	Target number (1–4)	UByte		
7	Total number of targets (1–4)	UByte		
8 ⋮ 11	Lateral range	Long	0.001 m	0x80000000
12 ⋮ 15	Longitudinal range	Long	0.001 m	0x80000000
16–17	Lateral range rate	Short	0.01 m/s	0x8000
18–19	Longitudinal range rate	Short	0.01 m/s	0x8000
20 ⋮ 23	Hunter measurement point position *x* for this target	Long	0.001 m	0x80000000
24 ⋮ 27	Hunter measurement point position *y* for this target	Long	0.001 m	0x80000000
28 ⋮ 31	Target measurement point position *x*	Long	0.001 m	0x80000000
32 ⋮ 35	Target measurement point position *y*	Long	0.001 m	0x80000000
36–37	Heading angle of hunter	UShort	0.01 deg	0xFFFF
38–39	Heading angle of target	UShort	0.01 deg	0xFFFF
40	Range status	UByte		
41	Status channel	UByte		
42 ⋮ 49	*These bytes are specific to each range status channel message*			
50–51	Forward velocity of hunter	Short	0.01 m/s	
52–53	Lateral velocity of hunter	Short	0.01 m/s	0x8000
54–55	Lateral range acceleration	Short	0.01 m/s²	0x8000
56–57	Longitudinal range acceleration	Short	0.01 m/s²	0x8000
58	Nearest target polygon vertex to hunter point	UByte		0xFF

	left			
59	Nearest target polygon vertex to hunter point right	UByte		0xFF
60	Target visibility 0 = Not visible 100 = Visible	UByte		0xFF
61	Target feature point type 0 = Disabled 0xFE = Unknown	UByte		0xFF
62–63	Target feature point index 0 = Disabled 0xFFFE = Out of range	UShort		0xFFFF
64	Nearest hunter polygon vertex to target point left	UByte		0xFF
65	Nearest hunter polygon vertex to target point right	UByte		0xFF
66	Nearest target polygon vertex to hunter polygon left	UByte		0xFF
67	Nearest target polygon vertex to hunter polygon right	UByte		0xFF
68	Nearest hunter polygon vertex to target polygon left	UByte		0xFF
69	Nearest hunter polygon vertex to target polygon right	UByte		0xFF
70	Nearest target polygon vertex to hunter point scale	UByte	0.004	0xFF
71	Nearest hunter polygon vertex to target point scale	UByte	0.004	0xFF
72	Nearest target polygon vertex to hunter polygon scale	UByte	0.004	0xFF
73	Nearest hunter polygon vertex to target polygon scale	UByte	0.004	0xFF
74 ⋮ 77	Hunter polygon origin position *x*	Long		0x80000000
78 ⋮ 81	Hunter polygon origin position *y*	Long		0x80000000
82 ⋮ 85	Target polygon origin position *x*	Long		0x80000000
86 ⋮ 89	Target polygon origin position *y*	Long		0x80000000

90 ⋮ 93	Hunter unit position *x*	Long		0x80000000
94 ⋮ 97	Hunter unit position *y*	Long		0x80000000
98 ⋮ 101	Target unit position *x*	Long		0x80000000
102 ⋮ 105	Target unit position *y*	Long		0x80000000
106-107	Pitch angle of hunter	Short	0.01 deg	0x8000
108-109	Roll angle of hunter	Short	0.01 deg	0x8000
110-111	Pitch angle of target	Short	0.01 deg	0x8000
112-113	Roll angle of target	Short	0.01 deg	0x8000
114 ⋮ 117	Resultant range (magnitude in measurement plane) from multiple sensor point 1 to target	ULong	0.001 m	0xFFFFFFFF
118	Percentage of target visible in field of view of multiple sensor 1 0 = None of target visible in field of view 100 = Target fully visible inside field of view	UByte		0xFF
119	Percentage of field of view of multiple sensor 1 occupied by visible part of target 0 = Target occupies none of field of view 100 = Target occupies all of field of view	UByte		0xFF
120 ⋮ 185	11 further blocks of 6 bytes of multiple sensor point measurements for sensor points 2 to 12, using the same structure as bytes 114-119 above	(6-byte blocks) * 11		As above
186	Checksum	UByte		

Note 1: in earlier versions this packet was shorter in length. Take care in the decoder to only decode values that exist in the data section of the transmitted packet.

Table 17. Ext. range packet, status channel 0, latency measurement

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	GPS time in minutes. Minutes since 6th January 1980	Long	1 minute	0x80000000
46	Position mode of hunter, see position mode description in NCOM manual	UByte		> 127
47	Position mode of target, see position mode description in NCOM manual	UByte		> 127
48–49	Target latency	UShort	0.001 s	0xFFFF

Note that target latency can only be used as snapshot indicator at the time this status message is transmitted. The OxTS software will hold this value constant until it is updated, which will not be with the next range packet. Using target latency as an exact value can be misleading, it should be thought of as an indicator.

Table 18. Ext. range packet, status channel 1, RT-Range software Dev ID

Byte	Description	Type	Unit	Invalid
42 ⋮ 49	RT-Range software development ID. 8 chars represent the ASCII text of the software development ID or version.	Byte[8]	ASCII	

Table 19. Ext. range packet, status channel 2, target serial communications status

Byte	Description	Type	Unit	Invalid
42–43	Characters received from target radio. This value will wrap when it overflows	UShort		
44–45	Packets received from target radio. This value will wrap when it overflows	UShort		
46–47	Characters skipped from target radio because of errors. This value will wrap when it overflows	UShort		
48–49	Reserved			

Table 20. Ext. range packet, status channel 3, target wireless LAN communications status

Byte	Description	Type	Unit	Invalid
42–43	Characters received from target via wireless LAN. This value will wrap when it overflows	UShort		
44–45	Packets received from target via wireless LAN. This value will wrap when it overflows	UShort		
46–47	Characters skipped from target via wireless LAN. This value will wrap when it overflows	UShort		
48–49	Reserved			

The number of the target that these values apply to is in the extended range packet.

Table 21. Ext. range packet, status channel 4, hunter Ethernet communication status

Byte	Description	Type	Unit	Invalid
42–43	Characters received from hunter via Ethernet. This value will wrap when it overflows	UShort		
44–45	Packets received from hunter via Ethernet. This value will wrap when it overflows	UShort		
46–47	Characters skipped from hunter via Ethernet. This value will wrap when it overflows	UShort		
48–49	Reserved			

Table 22. Ext. range packet, status channel 5, range output latency and offset measurements

Byte	Description	Type	Unit	Invalid
42–43	Hunter output latency	UShort	0.001 s	0xFFFF
44–45	Range longitudinal offset	Short	0.001 m	0x8000
46–47	Range lateral offset	Short	0.001 m	0x8000
48–49	Reserved			

Table 23. Ext. range packet, status channel 6, OS and script version

Byte	Description	Type	Unit	
42	Major OS version	UByte		0xFF
43	Minor OS version	UByte		0xFF
44	OS revision version	UByte		0xFF
45 ⋮ 47	Script version	UWord		0xFFFFFF
48–49	Reserved			

Table 24. Ext. range packet, status channel 7, UTC offset, range configuration and CPU load

Byte	Description	Type	Unit	Invalid
42–43	UTC offset	Short	1 s	0x8000
44	Range reference plane configuration 0 = Level plane 1 = Hunter plane	UByte		0xFF
45	Target feature set number 0 = No feature set	UByte		0xFF
46–47	Number of feature points in feature set 0 = No feature set	UShort		0xFFFF
48	Maximum number of feature points per feature cell 0 = No feature set 0xFE = (>= 254)	UByte		0xFF
49	CPU load	UByte	0.4%	0xFF

Table 25. Ext. range packet, status channel 8, fixed point position

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Fixed point latitude	Long	$1° \times 10^{-7}$	0x80000000
46 ⋮ 49	Fixed point longitude	Long	$1° \times 10^{-7}$	0x80000000

The number of the target that these values apply to is in the extended range packet.

Table 26. Ext. range packet, status channel 9, hunter and target IP addresses

Byte	Description	Type	Unit	Invalid
42	Hunter IP byte 1	UByte		
43	Hunter IP byte 2	UByte		
44	Hunter IP byte 3	UByte		
45	Hunter IP byte 4	UByte		
46	Target IP byte 1	UByte		
47	Target IP byte 2	UByte		
48	Target IP byte 3	UByte		
49	Target IP byte 4	UByte		

An IP address is invalid if all four bytes are equal to zero.

Table 27. Ext. range packet, status channel 10, fixed point altitude and heading

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Fixed point altitude	Long	0.001 m	0x80000000
46 ⋮ 49	Fixed point heading	ULong	$1° \times 10^{-7}$	0xFFFFFFFF

The number of the target that these values apply to is in the extended range packet.

Table 28. Ext. range packet, status channel 11, RT-Range local co-ordinates origin, latitude and longitude

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Local co-ordinates origin latitude	Long	$1° \times 10^{-7}$	0x80000000
46 ⋮ 49	Local co-ordinates origin longitude	Long	$1° \times 10^{-7}$	0x80000000

Table 29. Ext. range packet, status channel 12, RT-Range local co-ordinates origin, altitude and heading

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Local co-ordinates origin altitude	Long	0.001 m	0x80000000
46 ⋮ 49	Local co-ordinates *x*-axis heading	ULong	$1° \times 10^{-7}$	0xFFFFFFFF

Table 30. Ext. range packet, status channel 13, range hunter lever-arm

Byte	Description	Type	Unit	Invalid
42 ⋮ 44	Hunter lever-arm (displacement arm) in the *x*-axis of the vehicle frame	Word	0.001 m	0x800000
45 ⋮ 47	Hunter lever-arm (displacement arm) in the *y*-axis of the vehicle frame	Word	0.001 m	0x800000
48–49	Hunter lever-arm (displacement arm) in the *z*-axis of the vehicle frame	Short	0.001 m	0x8000

Table 31. Ext. range packet, status channel 14, range target lever-arm

Byte	Description	Type	Unit	Invalid
42 ⋮ 44	Target lever-arm (displacement arm) in the *x*-axis of the vehicle frame	Word	0.001 m	0x800000
45 ⋮ 47	Target lever-arm (displacement arm) in the *y*-axis of the vehicle frame	Word	0.001 m	0x800000
48–49	Target lever-arm (displacement arm) in the *z*-axis of the vehicle frame	Short	0.001 m	0x8000

Table 32. Ext. range packet, status channel 15, command communication status

Byte	Description	Type	Unit	Invalid
42–43	UDP command characters received. This value will wrap when it overflows	UShort		
44–45	UDP command packets received. This value will wrap when it overflows	UShort		
46–47	UDP command characters skipped. This value will wrap when it overflows	UShort		
48–49	UDP command errors. This value will wrap when it overflows	UShort		

Table 33. Ext. range packet, status channel 16, range accuracy

Byte	Description	Type	Unit	Invalid
42–43	Range Longitudinal Accuracy	UShort	0.001 m	0xFFFF
44–45	Range Lateral Accuracy	UShort	0.001 m	0xFFFF
46–47	Range Vertical Accuracy	UShort	0.001 m	0xFFFF
48–49	Range Magnitude Accuracy	UShort	0.001 m	0xFFFF

The vertical accuracy will be the component in the direction perpendicular to the hunter plane when the range measurements are configured in “hunter plane” mode (instead of the default “level plane” mode).

Table 34. Ext. range packet, status channel 17, target vehicle geometry

Byte	Description	Type	Unit	Invalid
42–43	Target vehicle length	UShort	0.001 m	0xFFFF
44–45	Target vehicle width	UShort	0.001 m	0xFFFF
46–47	Target polygon number	UShort		0xFFFF
48–49	Target vehicle height	UShort	0.001 m	0xFFFF

Table 35. Ext. range packet, status channel 18, acceleration filter settings

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Acceleration filter cut-off frequency 0 = Disabled	Float	Hz	< 0
46 ⋮ 49	Acceleration filter damping ratio 0 = Disabled	Float		< 0

Table 36. Ext. range packet, status channel 19, extrapolation filter settings

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Extrapolation filter cut-off frequency 0 = Disabled	Float	Hz	< 0
46 ⋮ 49	Extrapolation filter damping ratio 0 = Disabled	Float		< 0

Table 37. Ext. range packet, status channel 20, feature point position

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Feature point latitude	Long	$1° \times 10^{-7}$	0x80000000
46 ⋮ 49	Feature point longitude	Long	$1° \times 10^{-7}$	0x80000000

The number of the target that these values apply to is in the extended range packet.

Table 38. Ext. range packet, status channel 21, feature point altitude and heading

Byte	Description	Type	Unit	Invalid
42 ⋮ 45	Feature Point Altitude	Long	0.001 m	0x80000000
46 ⋮ 49	Feature Point Heading	ULong	$1° \times 10^{-7}$	0xFFFFFFFF

The number of the target that these values apply to is in the extended range packet.

Table 39. Ext. range packet, status channel 22, hunter vehicle geometry

Byte	Description	Type	Unit	Invalid
42–43	Hunter vehicle length	UShort	0.001 m	0xFFFF
44–45	Hunter vehicle width	UShort	0.001 m	0xFFFF
46–47	Hunter polygon number	UShort		0xFFFF
48–49	Hunter vehicle height	UShort	0.001 m	0xFFFF

Wrapped NCOM packet

The wrapped NCOM packet is used to encapsulate data from the RTs into RCOM. This allows RT-Range Post-process to export data from the RTs in the same file as the RT-Range data. It also allows NAVgraph and NAVdisplay to display RT data with RT-Range data. RT-Range Post-process encodes NCOM data into RCOM messages when post-processing is used to compute the RT-Range data, rather than real-time data. Note that more than one RT is wrapped into this packet, so it is important to identify which RT is being received while unwrapping the NCOM. To decode the NCOM data, see the NCOM Manual. Table 40 describes how NCOM is wrapped into an RCOM message.

Table 40. Wrapped NCOM packet

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type	UByte		
2–3	Length of data section	UShort		
4	IP address byte 1 of RT sending the NCOM packet	UByte		
5	IP address byte 2 of RT sending the NCOM packet	UByte		
6	IP address byte 3 of RT sending the NCOM packet	UByte		
7	IP address byte 4 of RT sending the NCOM packet	UByte		
8	NCOM provenance 0x00 = UDP 0x01 = COM1 ⋮ 0x0F = COM15 0x10 = FILE	UByte		0xFF
9 ⋮ *x*	Wrapped NCOM packet *x* = NCOM packet length + 8			
x + 1	Checksum	UByte		

For NCOM data from a file (NCOM provenance = FILE), the IP address are filled with the characters "hunt" for hunter NCOM data or "tgt" followed by the target number for target NCOM.

An IP address is invalid if all four bytes are equal to zero; this will be the case if the NCOM data has been received on a serial port (NCOM provenance = COM*n*).

Trigger time packet

The trigger time packet is used in RCOM files (not over Ethernet) to identify the time of the trigger that started logging the file. It allows software to have "pre-trigger" data in the file. RT-Range Post-process will have the "TimeFromStart" field as zero at the time of the trigger. Data before the trigger will have a negative "TimeFromStart".

There is no status field in this packet.

Table 41. Trigger time packet

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type	UByte		
2–3	Length of data section	UShort		
4–5	GPS time into minute of trigger	UShort	0.001 s	0xFFFF
6	GPS time offset from millisecond of trigger	Char	0.004 ms	0x80
7 ⋮ 10	GPS time in minutes of trigger. Minutes since 6th January 1980	Long	1 minute	0x80000000
11	Checksum	UByte		

Note that the "GPS time offset from millisecond of trigger" can be negative. The field "GPS time into minute of trigger" is rounded to the closest millisecond and the "GPS time offset from millisecond of trigger" is negative when the "GPS time into minute of trigger" is rounded up.

Polygon packet

These packets contain information about the polygon configuration used for the hunter and all targets. The data given in polygon packets can be rotated and translated by an RCOM decoder to generate polygons in the local reference frame allowing a real-time display of the hunter and target polygons. These packets contain static configuration data and are transmitted at a low-rate, it is likely that several packets will need to be received before a full description of the polygon configuration is available.

The amount of data required to describe a polygon is significant. To be able to maintain a steady stream of transmitted data, the data required for a single polygon can be spread over several polygon packets. Each packet will contain a continuous run of polygon vertex information beginning at the specified starting polygon vertex index encoded in the packet. Every polygon packet also encodes the total number of vertices of the polygon, and whether the polygon is associated with the hunter or a given target.

The polygon packet can encode two- or three-dimensional vertices, and can describe the vertex position information at a low or high resolution. Currently, only two-dimensional low resolution vertices are used. Here, each vertex is stored as a pair of shorts—where the first short is the forward co-ordinate, and the second short is the right co-ordinate.

Table 42. Polygon packet

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type (0x05)	UByte		
2–3	Length of data section	UShort		
4–5	Reserved	UShort		0xFFFF
6	Polygon Id 0 = Hunter 1–4 = Target #	UByte		
7	Total number of polygon vertices	UByte		
8	Starting vertex index for the vertex data of this packet	UByte		
9	Polygon vertex encoding of this packet: Bit 7 = Byte resolution 2 or 3, (bit 7 value + 2) Bit 6 = Dimensions 2 or 3, (bit 6 value + 2) Bit 5–0 = Vertices count 1 to 64 vertices, (bits 5–0 + 1) The number of bytes of polygon vertex data is: v_n = resolution × dimension × count.	UByte		
10 ⋮ v_n + 9	Polygon vertex data: Data encoded as 2 or 3-dimensional vertex, with 2 or 3 byte resolution. 2 bytes resolution decode as Short with units of 0.001 m. 3 byte resolution decode as Word with units of 0.0001 m	Short or Word	0.001 m or 0.0001 m	0xFFFF or 0xFFFFFF
v_n +10	Checksum	UByte		

The polygon vertex data consists of successive hunter-frame pairs or triplets offsets for each vertex: sequence of "count" x, y pairs in 2 dimensions; sequence of "count" x, y, z triplets in 3 dimensions.

Multiple sensor points packet

These packets contain information about the configuration of multiple sensor points attached to the hunter. The data given in multiple sensor packets can be rotated and translated by an RCOM decoder to calculate the position and field of view of each multiple sensor point in the local reference frame (for example so these can be displayed in real-time alongside hunter and target positions). These packets contain static configuration data and are transmitted at a low-rate. The packets are designed so information about different sensor points can be split across a number of packets; should this be the case, it will be necessary to receive several packets before the configuration of all multiple sensor points is known.

The multiple sensor point packet can encode two- or three-dimensional information and can describe this information at a low or high resolution. In two-dimensions the configuration of each multiple sensor point consists of 7 parameters (x offset, y offset, heading, half horizontal field of view, half vertical field of view, minimum distance, maximum distance) followed by an 8-byte sensor name; in three-dimensions it consists of 10 parameters (x offset, y offset, z offset, heading, pitch, roll, half horizontal field of view, half vertical field of view, minimum distance, maximum distance) followed by an 8-byte sensor name. In low-resolution mode each parameter is stored as a two-byte short; in high resolution mode, each parameter is stored as a 3-byte word. Currently, only two-dimensional low-resolution information is transmitted.

There are four different packet structures, depending on the number of dimensions and the resolution of the packet. The two low-resolution packet structures are described in separate tables directly below.

Table 43. Multiple sensor point packet (two dimensions, low resolution)

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type (0x06)	UByte		
2–3	Length of data section	UShort		
4–5	Reserved	UShort		0xFFFF
6	Multiple sensor point object ID (0 = Hunter)	UByte		
7	Total number of multiple sensor points	UByte		
8	Index of first multiple sensor point in this packet: index of 0 means first sensor point in the packet is multiple sensor point 1 (`first` = index + 1)	UByte		
9	Multiple sensor point encoding of this packet: Bit 7 = byte resolution (bit 7 = 0 for low resolution, 1 for high resolution) Bit 6 = dimensions (bit 6 = 0 for two dimensions) Bit 5–4 = reserved Bit 3-0 = `count` of 1 to 16 sensor points, (bits 3–0 + 1) The number of bytes of multiple sensor point data to follow in packet is: $v_n = 22 \times$ `count`	UByte		
10-11	X offset of multiple sensor point `first`	Short	0.001 m	0x8000
12-13	Y offset of multiple sensor point `first`	Short	0.001 m	0x8000
14-15	Heading offset of multiple sensor point `first`	UShort	0.01 °	0xFFFF
16-17	Half of horizontal field of view of multiple sensor point `first`	UShort	0.01 °	0xFFFF
18-19	Half of vertical field of view of multiple sensor point `first`	UShort	0.01 °	0xFFFF
20-21	Minimum distance of multiple sensor point `first`	UShort	0.001 m	0xFFFF
22-23	Maximum distance of multiple sensor point `first`	UShort	0.001 m	0xFFFF
24-31	Name of multiple sensor point `first`	8 Chars		
32 ⋮ v_n +9	(`count` – 1) further blocks of 22 bytes of multiple sensor point configuration information for sensor points (`first` + 1) to (`first` + `count` - 1), using the same structure as bytes 10-31 above.	(22-byte blocks) * (`count`-1)		
v_n +10	Checksum	UByte		

In two dimensions, each multiple sensor point is described by 7 parameters and a sensor name; in low resolution mode each parameter is encoded by two bytes; if `count` is 1, the checksum is at index 32, and the block (32 … v_n +9) is not present.

Table 44. Multiple sensor point packet (three dimensions, low resolution)

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type (0x06)	UByte		
2–3	Length of data section	UShort		
4–5	Reserved	UShort		0xFFFF
6	Multiple sensor point object ID (0 = Hunter)	UByte		
7	Total number of multiple sensor points	UByte		
8	Index of first multiple sensor point in this packet: index of 0 means first sensor point in the packet is multiple sensor point 1 (`first` = index + 1)	UByte		
9	Multiple sensor point encoding of this packet: Bit 7 = byte resolution (bit 7 = 0 for low resolution, 1 for high resolution) Bit 6 = dimensions (bit 6 = 1 for three dimensions) Bit 5–4 = reserved Bit 3-0 = `count` of 1 to 16 sensor points, (bits 3–0 + 1) The number of bytes of multiple sensor point data to follow in packet is: $v_n = 28 \times$ `count`	UByte		
10-11	X offset of multiple sensor point `first`	Short	0.001 m	0x8000
12-13	Y offset of multiple sensor point `first`	Short	0.001 m	0x8000
14-15	Z offset of multiple sensor point `first`	Short	0.001 m	0x8000
16-17	Heading offset of multiple sensor point `first`	UShort	0.01 °	0xFFFF
18-19	Pitch offset of multiple sensor point `first`	Short	0.01 °	0x8000
20-21	Roll offset of multiple sensor point `first`	Short	0.01 °	0x8000
22-23	Half of horizontal field of view of multiple sensor point `first`	UShort	0.01 °	0xFFFF
24-25	Half of vertical field of view of multiple sensor point `first`	UShort	0.01 °	0xFFFF
26-27	Minimum distance of multiple sensor point `first`	UShort	0.001 m	0xFFFF
28-29	Maximum distance of multiple sensor point `first`	UShort	0.001 m	0xFFFF
30-37	Name of multiple sensor point `first`	8 Chars		
38 ⋮ v_n +9	(`count` – 1) further blocks of 28 bytes of multiple sensor point configuration information for sensor points (`first` + 1) to (`first` + `count` - 1), using the same structure as bytes 10-37 above.	(28-byte blocks) * (`count`-1)		
v_n +10	Checksum	UByte		

In three dimensions, each multiple sensor point is described by 10 parameters and a sensor name; in low resolution mode each parameter is encoded by two bytes; if `count` is 1, the checksum is at index 38, and the block (38 … v_n +9) is not present.

Table 45. Multiple sensor point packet (two dimensions, high resolution)

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type (0x06)	UByte		
2–3	Length of data section	UShort		
4–5	Reserved	UShort		0xFFFF
6	Multiple sensor point object ID (0 = Hunter)	UByte		
7	Total number of multiple sensor points	UByte		
8	Index of first multiple sensor point in this packet: index of 0 means first sensor point in the packet is multiple sensor point 1 (`first` = index + 1)	UByte		
9	Multiple sensor point encoding of this packet: Bit 7 = byte resolution (bit 7 = 1 for high resolution) Bit 6 = dimensions (bit 6 = 0 for two dimensions) Bit 5–4 = reserved Bit 3-0 = `count` of 1 to 16 sensor points, (bits 3–0 + 1) The number of bytes of multiple sensor point data to follow in packet is: $v_n = 29 \times$ `count`	UByte		
10-12	X offset of multiple sensor point `first`	Word	0.0001 m	0x800000
13-15	Y offset of multiple sensor point `first`	Word	0.0001 m	0x800000
16-18	Heading offset of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
19-21	Half of horizontal field of view of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
22-24	Half of vertical field of view of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
25-27	Minimum distance of multiple sensor point `first`	UWord	0.0001 m	0xFFFFFF
28-30	Maximum distance of multiple sensor point `first`	UWord	0.0001 m	0xFFFFFF
31-38	Name of multiple sensor point `first`	8 Chars		
39 ⋮ v_n +9	(`count` – 1) further blocks of 29 bytes of multiple sensor point configuration information for sensor points (`first` + 1) to (`first` + `count` - 1), using the same structure as bytes 10-38 above.	(29-byte blocks) * (`count`-1)		
v_n +10	Checksum	UByte		

Table 46. Multiple sensor point packet (three dimensions, high resolution)

Byte	Description	Type	Unit	Invalid
0	Sync byte (0x57)	UByte		
1	Packet type (0x06)	UByte		
2–3	Length of data section	UShort		
4–5	Reserved	UShort		0xFFFF
6	Multiple sensor point object ID (0 = Hunter)	UByte		
7	Total number of multiple sensor points	UByte		
8	Index of first multiple sensor point in this packet: index of 0 means first sensor point in the packet is multiple sensor point 1 (`first` = index + 1)	UByte		
9	Multiple sensor point encoding of this packet: Bit 7 = byte resolution (bit 7 = 1 for high resolution) Bit 6 = dimensions (bit 6 = 1 for three dimensions) Bit 5–4 = reserved Bit 3-0 = `count` of 1 to 16 sensor points, (bits 3–0 + 1) The number of bytes of multiple sensor point data to follow in packet is: $v_n = 38 \times$ `count`	UByte		
10-12	X offset of multiple sensor point `first`	Word	0.0001 m	0x800000
13-15	Y offset of multiple sensor point `first`	Word	0.0001 m	0x800000
16-18	Z offset of multiple sensor point `first`	Word	0.0001 m	0x800000
19-21	Heading offset of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
22-24	Pitch offset of multiple sensor point `first`	Word	0.0001 °	0x800000
25-27	Roll offset of multiple sensor point `first`	Word	0.0001 °	0x800000
28-30	Half of horizontal field of view of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
31-33	Half of vertical field of view of multiple sensor point `first`	UWord	0.0001 °	0xFFFFFF
34-36	Minimum distance of multiple sensor point `first`	UWord	0.0001 m	0xFFFFFF
37-39	Maximum distance of multiple sensor point `first`	UWord	0.0001 m	0xFFFFFF
40-47	Name of multiple sensor point `first`	8 Chars		
48 ⋮ v_n +9	(`count` – 1) further blocks of 38 bytes of multiple sensor point configuration information for sensor points (`first` + 1) to (`first` + `count` - 1), using the same structure as bytes 10-47 above.	(38-byte blocks) * (`count`-1)		
v_n +10	Checksum	UByte		

Revision History

Table 47. Revision History

Revision	Comments
100316	Initial version based on the internal description “100311 RCOM Format.doc”.
100408	Added trigger time packet.
100419	Corrections.
100528	Correction to the time in the trigger time packet.
101129	Corrections. Extension to lane packet (curvature, distance to all lines from points B and C, heading relative to lines). Extension to extend range packet (time to collision based on acceleration, vehicle dimensions, support for polygon-shaped targets, target visibility, range accuracy information, support for filtering accelerations).
120607	Extension of extended range packet to support feature points (feature point index, feature point type, feature point position and heading, feature set parameters) and heading direction for fixed points.
150917	Corrections. Remove descriptions of obsolete range packets. Added polygon packet and polygon related material.
180723	Corrections. Extension of extended range packet with hunter and target pitch and roll, and measurements from 12 multiple sensor points. Addition of multiple sensor point configuration packet.
180828	Add CPU load to status channel 7 of lane and extended range packets.
181122	Added high resolution multiple sensor point packet definition for both 2d and 3d.