Interface Manual for Radar sR13-12VHe and sR17-12VHe

Version 0.1

This document describes the Sentire™ Radar modules **sR13-12VHe** and **sR17-12VHe**. The module sR13-12VHe operates in the frequency range 12.5 GHz to 14.5 GHz and the module sR17-12VHe operates in the frequency range 16.5 GHz to 18.5 GHz. The main communication interface is Ethernet "**e**" but an USB connection can also be established. This manual is focusing on the communication interfaces and the associated protocol. Furthermore, the format and the necessary settings for the continuous transmission (streaming) of certain measurement data are described. To ensure that the desired data is transmitted in the correct format and at the correct speed, the module must first be configured or the current configuration must at least be checked. For this purpose, and also to start or stop the stream, various commands are used, which are explained in section 4.

Section 2 gives a brief overview of the system concept and functionality. Section 3 deals with the meaning and relevance of the parameter configurations used here. Finally, section 5 describes the format of the data stream.

It should be noted, that the backend board is also used for other frontend boards and so the firmware has some options and functions which are included but not applicable here. Those parts in this document are printed in grey. Parameters of commands which are printed in grey must also be sent and read.

IMST GmbH

Carl-Friedrich-Gauss-Str. 2-4

47475 Kamp-Lintfort

Germany

radar@imst.de

www.imst.com

www.radar-sensor.com

Table of Contents

1 Document History

Version	Release Date	Revisions
0.1	tbd	First release of this document

Table 1: Document history

2 Functionality

The radar module consists of two boards: backend and frontend. The frontend contains signal generation (synthesizer, VCO) and analogue signal processing (mixer, amplifier). Everything else follows on the backend. Here are the communication interfaces (Ethernet, USB), DC supply of the module as well as control and data processing as also shown in Figure 1.

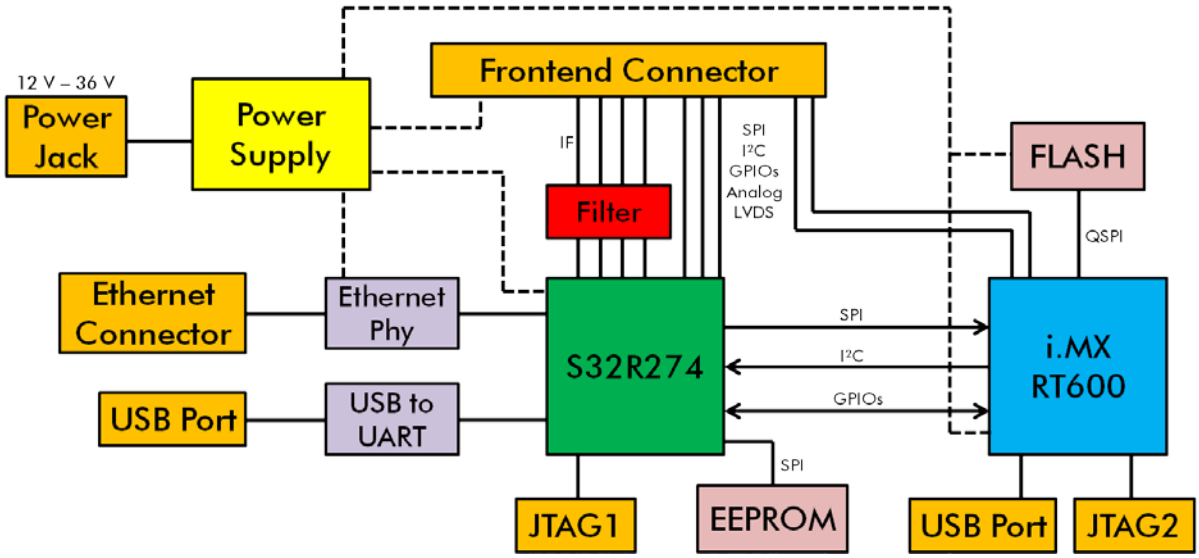

Figure 1: Simplified block-diagram of the core components of the backend

The most important component here is the microcontroller unit (MCU), which controls and provides all functions of the module. The S32R274 chip from NXP is used here. This has three processor cores and a computing unit for signal processing of radar data.

Each of the three processor cores takes on certain tasks. The first core initialises all important modules and monitors the system. The second core contains the radar measurement process. This consists of configuring the front end and the radar signal processing, triggering and sampling a measurement and signal evaluation. In the case of streaming, this core would evaluate measured data up to a configured point and then trigger the third processor core to send the data. The third core takes care of the communication interfaces to the outside.

Available are a USB interface, which is available through UART via a converter chip, and an Ethernet interface, through which TCP and UDP ports can be reached. Depending on the settings, all processes run asynchronously for the most part and only trigger each other.

In addition, another processor is implemented on the backend. This is a combination of ARM MCU and DSP for audio applications, also from NXP. This processor will later enable the classification of tracked targets by means of Micro-Doppler sequences and a neural network. The required data is exchanged between the processors via the interfaces (SPI, I²C).

3 Configuration

The backend of the radar module is multifunctional and designed for different frontends and applications. Therefore, there is also a variety of parameters, functions and commands. The parameter structures that can be configured externally are listed below.

3.1 Radar Parameters

The radar parameters in Table 2 define the measurement and signal processing, as well as the transmitted data. The column "Default" contains meaningful or recommended values. Table 3 shows the setting options of the parameter "RadarCube", which is essentially responsible for the measurement procedure.

Name	Data type	Description	Default
RadarCube	uint16	Enum: Measurement procedure and resulting data layout. See Table 3.	10
ContinuousMeas	uint8	Flag: Switches continuous measurement On (1) or Off (0). If switched on, the radar measures automatically and tries to keep the interval "MeasInterval".	1
MeasInterval	uint16	Measurement interval in milliseconds. Used for continuous measurement. If the data processing is faster than the interval, the radar waits before the next measurement. Value range: 1 - 10000	0
Processing	uint16	Enum: Data processing steps to be performed for continuous measurement. For measurements with only one chirp, the maximum step is the Range FFT. 0: No Processing (ADC Data), 1: Range FFT, 2: Doppler FFT, 3: Combining (Non Coherent Integration from 4 resp. 12 Channels and computation of the magnitudes), 4: Peak Detection, 5: CFAR, 6: Detection (Among other things, calculation of the object angles), 7: Tracking	7
RangeWinFunc	uint16	Enum: Window function for the Range FFT 0: No Window 1: Blackman, 2: Hamming, 3: Hann, 4: Nuttal	2
DopplerWinFunc	uint16	Enum: Window function for the Doppler FFT 0: No Window 1: Blackman, 2: Hamming, 3: Hann, 4: Nuttal	2
DopplerFftShift	uint8	Flag: If 1, negative Doppler bins are sent first during transmission.	1
MinRangeBin	uint16	Minimum range bin for data transmission and CFAR. Value range depends on "RadarCube". Must be smaller than "MaxRangeBin".	0
MaxRangeBin	uint16	Maximum range bin for data transmission and CFAR. Value range depends on "RadarCube". Must be greater than "MinRangeBin".	Max
MinDopplerBin	int16	Minimum Doppler Bin for data transmission and CFAR. Value range depends on "RadarCube". Must be smaller than "MaxDopplerBin".	0
MaxDopplerBin	int16	Maximum Doppler Bin for data transmission and CFAR. Value range depends on "RadarCube". Must be greater than "MinDopplerBin".	Max
CfarWindowSize	uint16	Window size for CFAR. (0 – 30)	10
CfarGuardInt	uint16	Safety interval for CFAR (0 – 10)	2
RangeCfarThresh	uint16	Threshold for CFAR in range direction (0 – 100)	10
TriggerThresh	int16	Threshold in [dB] for Pin Trigger of RadarCubes 0 – 3 (-100 – 100)	10
PeakSearchThresh	uint16	Threshold for Peak Search (1 – 30)	10
SuppressStaticTargets	uint16	Number of Doppler bins around the zero Doppler line for which the range profiles are deleted before Peak Search. Suppresses static targets. 0: no deletion, 1: bin corresponding to speed 0, 2: as for 1 and +-1 bin, 3: as for 1 and +-2 bins, 4: as for 1 and +-3 bins.	0
MaxTargets	uint16	Maximum number of possible detected targets (1-128)	20
MaxTracks	uint16	Maximum number of possible tracked targets (1-30)	20
MaxHorSpeed	uint16	Tracker: Maximum permitted speed in horizontal direction [m/s] (0 – 1000)	5
MaxVerSpeed	uint16	Tracker: Maximum permitted speed in vertical direction [m/s] (0 – 1000)	1
MaxAccel	uint16	Tracker: Maximum permitted acceleration [m/s²] (0 – 100)	10
MaxRangeError	uint16	Tracker: Maximum allowed error in the distance [m/10] (1 – 1000), should be at least 2x "TargetSize"	20
MinConfirm	uint16	Tracker: Minimum number of confirmations required to be considered a track.	2

		(0 – 100)	
TargetSize	uint16	Size of expected target [m/10] (0 – 1000), for humans ~0.5m	5
MergeLimit	uint16	Tracker: Limit for track/track merging (1 – 1000)	15
SectorFiltering	uint8	Flag: If 1, the Sector Filtering algorithm is active	0
SpeedEstimation	uint16	Estimation of radar system speed to filter static targets in a moving environment. 0: OFF 1: Speed estimation only 2: Speed estimation and filter out static detections and tracks 3: Speed estimation and filter out static tracks only	0
DspDopplerProc	uint8	Flag: If 1, Doppler spectra of found targets are transmitted to the DSP for classification.	0
RxChannels	uint16	Bitmask: Data of the Rx channels corresponding to a "1" bit are transmitted, others are not. With MIMO, the first 12 bits are used, otherwise the first 4 bits. For radar cubes with only one chirp, there is a special case in the range FFT, but this is not relevant here.	0xF
CfarSelect	uint16	Selection of the CFAR algorithm: 0 = no CFAR, 1 = Range CFAR, 2 = Doppler CFAR, 3 = Range & Doppler CFAR	1
DopplerCfarThresh	uint16	Threshold for CFAR in Doppler direction (0 – 100)	10

Table 2: Radar parameters

Number	Radar Cube	Comment
0	256 Range Bins, 1 Chirp, 2 complex Rx Channels	only range FFT possible
1	512 Range Bins, 1 Chirp, 2 complex Rx Channels	only range FFT possible
2	1024 Range Bins, 1 Chirp, 2 complex Rx Channels	only range FFT possible
3	2048 Range Bins, 1 Chirp, 2 complex Rx Channels	only range FFT possible
4	64 Range Bins, 64 Chirps, 4 Rx Channels	
5	64 Range Bins, 128 Chirps, 4 Rx Channels	
6	64 Range Bins, 256 Chirps, 4 Rx Channels	
7	128 Range Bins, 64 Chirps, 4 Rx Channels	
8	128 Range Bins, 128 Chirps, 4 Rx Channels	
9	128 Range Bins, 256 Chirps, 4 Rx Channels	
10	256 Range Bins, 64 Chirps, 4 Rx Channels	
11	256 Range Bins, 128 Chirps, 4 Rx Channels	
12	256 Range Bins, 256 Chirps, 4 Rx Channels	Compressed data
13	512 Range Bins, 64 Chirps, 4 Rx Channels	Compressed data
14	512 Range Bins, 128 Chirps, 4 Rx Channels	Compressed data
15	128 Range Bins, 64 Chirps, 3 Tx Channels 4 Rx Channels	MIMO
16	128 Range Bins, 128 Chirps, 3 Tx Channels 4 Rx Channels	MIMO
17	128 Range Bins, 256 Chirps, 3 Tx Channels 4 Rx Channels	MIMO, Compressed data
18	256 Range Bins, 64 Chirps, 3 Tx Channels 4 Rx Channels	MIMO
19	256 Range Bins, 128 Chirps, 3 Tx Channels 4 Rx Channels	MIMO, Compressed data
20	512 Range Bins, 64 Chirps, 3 Tx Channels 4 Rx Channels	MIMO, Compressed data

Table 3: Possible values for the parameter "RadarCube"

3.2 Frontend Parameters

The frontend parameters define the signal generation and are hardware-dependent. Here, frontends are used which transmit in the range 12.5 GHz to 14.5 GHz and 16.5 GHz to 18.5 GHz. The ramp timings are read only here and cannot be changed directly. They are dependent on the Radar Parameter *"RadarCube"*. The "Default" column contains sensible or recommended values.

Name	Data type	Description	Default
MinFrequency	uint32	Minimum frequency [kHz] for FMCW measurements Value range: 12500000/16500000 to "MaxFrequency	12500000/ 16500000
MaxFrequency	uint32	Maximum frequency [kHz] for FMCW measurement Value range: "MinFrequency" to 14500000/18500000	14500000/ 18500000
SignalType	uint16	Enum: Type of modulation of the transmitted signal 1: CW @ Min. Frequency (no modulation), 2: CW @ Max. Frequency (no modulation), 3: FMCW Up-Ramp, 4: FMCW Down-Ramp	3 or 4
TxChannelSelection	uint16	Bit mask for switching the Tx channels on and off. Do not use!	0x1
RxChannelSelection	uint16	Bit mask for switching the Tx channels on and off. Do not use!	0xF
TxPowerSetting	int16	Output power of Transmitter [dBm EIRP] Range: 0 dBm to 30 dBm (sR13-12VH), 0 dBm to 28 dBm (sR17-12VH). **Note**: If the value **0x7FFF** is sent, the internal temperature compensation is switched OFF and the maximum EIRP value is set	20
RxPowerSetting	int16	Unused	
RampInit	uint32	Time interval [ns] before the start of a ramp. (read only)	
RampTime	uint32	Sampling time of the ramp [ns]. Dependent on number of samples. (read only)	
RampReset	uint32	Time interval [ns] for resetting a ramp. Calculated by the radar and used here for information. (read only)	
RampDelay	uint32	Time interval [ns] between several ramps. Unused here.	0
Reserve1	uint16		
Reserve2	uint16		
Reserve3	uint16		
RangeOffset	uint16	Offset [mm] from Transceiver Chip to Radome	0

Table 4: Frontend Parameters

3.3 Ethernet Configuration

The Ethernet interface is defined by the following parameters. Theoretically, any number of TCP and UDP ports can be defined. Currently, two TCP and two UDP ports are active. In addition, there is a fixed UDP port which responds to broadcast requests to find out the IP address and the possible ports (see section 3.3.1). This would also be possible with the USB interface and the command 0x0020.

Up to 4 different multicast groups can be defined. If the first byte of the address is zero, the respective group is not used for filtering. The port for multicast messages is also fixed.

The radar can ask a NTP server periodically for the current time. The parameter "SNTP Mode" enables this feature. If set to 1 the given server is asked every 10 minutes for the current time. If set to 2 the radar waits for broadcast messages with the current time.

Name	Data type	Description	Default
DHCP	uint8	not yet implemented	0
AutoIP	uint8	not yet implemented	0
IPv4	4 x uint8	IP Address as single Bytes.	192.168.0.2
TCP Port #1	uint16	Port number of first TCP port.	1024
TCP Port #2	uint16	Port number of second TCP port.	1025
UDP Port #1	uint16	Port number of first UDP port.	4120
UDP Port #2	uint16	Port number of second UDP port.	4121
NetMask	4 x uint8	Subnet mask	255.255.0.0
GateWay	4 x uint8	Standard Gateway	192.168.0.1
Multicast Group #1	4 x uint8	Multicast group 1	227.115.82.100
Multicast Group #2	4 x uint8	Multicast group 2	0.115.82.101
Multicast Group #3	4 x uint8	Multicast group 3	0.115.82.102
Multicast Group #4	4 x uint8	Multicast group 4	0.115.82.103

SNTP Mode	uint8	0 = OFF, 1 = Poll, 2 = Listen	0
NTP Server Address	4 x uint8		
UDP Multicast Port	uint16	Port number for UDP multicast messages (read only)	4440
UDP Broadcast Port	uint16	Port number for UDP broadcasts (read only)	4444
MAC	6 x uint8	MAC Address of module (read only)	

Table 5: Ethernet configuration

3.3.1 UDP broadcast port

The radar module always opens an UDP broadcast port with port number **4444**. For messages received on this port, the module expects the magic word **0x494D5354** as first value, which corresponds to the letters "IMST".

If only the magic word is sent, the radar module responds with the values shown in Table 6. This response message can be used to identify radar modules in the local network. The radar saves its Ethernet configuration in the EEPROM. In case the current configuration is unknown, this message can be used to find out the current configuration. Another possibility is to use the USB interface and execute the command 0x0020 (read out Ethernet configuration).

If a command plus CRC follows the magic word, the radar responds to the sender as for any other port. Please note that the magic word must not be part of the CRC calculation.

Name	Data type	Description
Magic Word	uint32	Value 0x494D5354 (IMST in ASCII representation)
Device Number	uint32	Unique device number
NumTcpPorts	uint16	Number of available TCP ports
for (n=0; n< NumTcpPorts; n++)		
TcpPort[n]	uint16	Number of TCP port n
NumUdpPorts	uint16	Number of available UDP ports
for (n=0; n< NumUdpPorts; n++)		
UdpPort[n]	uint16	Number of UDP port n
NumMulticastGroups	uint16	Number of multicast groups for filtering
for (n=0; n< NumMulticastGroups; n++)		
MulticastGroup[n]	4 x uint8	Multicast group address (IP address)

Table 6: Reply to UDP Broadcast Message

3.3.2 UDP multicast groups

If configured, the radar module opens an UDP port with port number **4440** which filters for up to 4 different multicast groups. The default groups are listed in Table 7. In case the first octet is zero, the respective group is deactivated. To configure the multicast groups, currently only the radar modules website can be used (call the radars IP address as URL in a browser with Javascript enabled).

Number	Multicast Group
1	227.115.82.100
2	0.115.82.101
3	0.115.82.102
4	0.115.82.103

Table 7: Default Multicast Groups

With multicast messages multiple modules in a local network can be addressed. The multicast port works as all other ports for commands.

4 Commands

To read or change the parameters described above, commands are sent via one of the interfaces. There are many different commands, also for reading out data. First, however, the protocol with which the commands are exchanged will be explained.

4.1 Protocol

The command protocol consists of simple requests and responses, with the radar module acting as the server. The same protocol is used for all interfaces, which can be seen in Table 8. The following applies to all data: MSB first.

Step	Host to Radar	Radar to Host
1	Command ID (2 Bytes)	
2	Data (optional)	
3	CRC Checksum (2 Bytes)	
4		Command ID (2 Bytes)
5		Status word (2 Bytes)
6		Data (optional)
7		CRC Checksum (2 Bytes)

Table 8: Command protocol

The host sends a command ID followed by optional data (e.g. radar parameters) and CRC checksum to the radar module. The radar module responds with at least the same command ID (if the ID is not understood, 0xE0F0 is sent), a status word (see Table 9) and a CRC checksum. If parameters or data are requested, they are sent between the status word and the CRC checksum.

Bit	Name	Description
0x0000	OK	No events
0x0001	CRC Error	Received CRC checksum was not correct. Parameters are not accepted.
0x0002	Invalid Rx Data	One or more received parameters have been corrected.
0x0004	Measurement Timeout	Waiting for a measurement by the radar process took too long.
0x0008	Invalid Interface	Wrong interface for this command.
0x0010	FE Error	Problem with the frontend hardware.
0x0020	FE Temp Error	Temperature on the frontend is too high and the frontend has been switched off.
0x0100	Global Error Occurred	An error has occurred and has not yet been corrected.
0x0200	Global Error Logged	An error has occurred, perhaps already corrected and noted.

Table 9: Bits of Status Word (Commands)

The status word provides information about the current communication or possible errors in the radar module. For example, if parameters are to be set and the module corrects received values, this is noted in the status word. Table 9 shows the different bits of the status word.

The CRC checksum is calculated over the entire data packet and appended to the packet so that the receiver (host or radar) can simply calculate the sum over the packet and get a "0" as the result.

The checksum used here is calculated according to the CRC-16-CITT standard and is used with the parameters in Table 10.

Parameter	Value
Polynomial	0x1021
Initial value	0xFFFF
Finale XOR	0x0000
Input reflected	No
Output reflected	No

Table 10: CRC Parameters

4.2 USB Interface

The USB interface uses the same protocol as the Ethernet interface and works with the basic data from Table 11.

Parameter	Value
Baud rate	2 MBaud
Parameter size	8 Bits
Parity	Odd
Stop bits	1

Table 11: USB Parameters

4.3 Command: Read out error

With this command, the global error mask and all module error masks are read out. The global error mask is a bit mask in which each bit represents an error in a specific area or module of the radar. A module error mask is a bit mask that describes errors in a specific module (e.g. Ethernet). In this command, all possible error masks are transmitted, i.e. the global error mask plus 16 module error masks. If a bit in the global mask is not yet assigned, the value 0 is transmitted as the module error mask. Table 12 lists the bits of the global error mask that have been assigned so far. The meaning of the individual error bits of the module error masks is not explained here.

Bit	Module
0x0001	System in general
0x0002	EEPROM
0x0004	UART/USB
0x0008	Ethernet
0x0010	Radar Processing
0x0020	Signal Processing Toolbox
0x0040	Frontend
0x0080	Parameter Update
0x0100	SPI to DSP
0x0200	I2C to DSP
0x0400	Error in DSP
0x0800	Error Log is full
0x1000	Spare

0x2000	Spare
0x4000	Spare
0x8000	Spare

Table 12: Meaning of the bits in the global error mask

The following table shows the structure of the command for reading out the error masks.

Command ID	0xE000	
Description	Reading out the error masks	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Global error mask	uint16	Global error mask with bits corresponding to Table 12
Module error mask	16 x uint16	A 16-bit error mask for each bit of the global error mask, even if some bits are not yet assigned.
CRC	uint16	CRC Checksum

Table 13: Command 0xE000: Reading out the error masks

4.4 Command: Read out error logs

The error logs are the stored values of the error masks. For each error that occurs, a bit is set in the error mask and in the error log. Thus, this command works in the same way as the command for reading out the error mask. The difference to the error mask is that the bits of the log are not automatically reset. For this, the "Command: Clear error logs" must be used. In addition, the first 100 errors that occur are saved in a table.

Command ID	0xE001	
Description	Reading out the error logs	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Global error log	uint16	Global error log with bits corresponding to Table 12
Module error log	16 x uint16	A 16-bit error log for each bit of the global error log, even if some bits are not yet occupied.
CRC	uint16	CRC Checksum

Table 14: Command 0xE001: Read out error logs

4.5 Command: Clear error logs

With this command, individual entries in the global error log and the module error logs can be deleted. For this purpose, the host sends a mask that contains the bits to be deleted according to Table 12. If, for example, the value 0x0002 is transmitted, the bit for an EEPROM error in the global log is deleted, but also the module error log as a whole.

Command ID	0xE002	
Description	Deleting the error logs	
Name	**Data type**	**Description**
Host → Radar		
ID	uint16	Command ID
Deleting mask	uint16	Mask for deleting specific error logs. Each bit corresponds to one bit of the global error log and one module error log.
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 15: Command 0x0002: Deleting the error logs

4.6 Command: Read out error log table

The first 100 errors that occur are stored in a table in addition to the error log. This table contains the time of occurrence of the error, measured in system time, and the occurred error as a mask value in the format according to Table 12. This command first sends the number of errors that have occurred so far and then the time stamp and the corresponding bit for each error that has occurred.

Command ID	0xE003	
Description	Read out error log table	
Name	**Data type**	**Description**
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
N_Err	uint16	Number of errors that occurred
for (n = 0; n < N_Err; n++)		
System time [n]	uint64	System time in milliseconds at the time of the error occurrence
Error bit [n]	uint16	Error bit according to Table 12
CRC	uint16	CRC Checksum

Table 16: Command 0xE003: Read out error log table

4.7 Command: Deleting the error log table

This command deletes all entries in the error log table.

Command ID	0xE004	
Description	Deleting the error log table	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 17: Command 0xE004: Deleting the error log table

4.8 Command: Read module information

With this command various information about the radar module is read out, which can be helpful in case of queries.

Command ID	0x0001	
Description	Read module information	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Module Number	uint32	Unique number of the radar module
Frontend	uint32	Code for a radar frontend for which the firmware is designed
Firmware Version	uint32	Version of the firmware in the format: Bytes 1 (MSB) and 2: main version, Byte 3: sub-version, Byte 4: sub-Sub-version
Firmware Revision	uint32	Revision of firmware
Firmware Date	uint32	Date on which the firmware was published in the format: Byte 1 (MSB): Day, Byte 2: Month, Bytes 3 and 4: Year
CRC	uint16	CRC Checksum

Table 18: Command 0x0001: Read module information

4.9 Command: Read system time

This command returns the current system time in milliseconds in Unix time format.

Command ID	0x0003	
Description	Read system time	
Name	Data type	Description
Host → Radar		

ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
System time	uint64	Current system time [ms] in Unix-Format
CRC	uint16	CRC Checksum

Table 19: Command 0x0003: Read system time

4.10 Command: Set system time

With this command, the system time of the radar module can be set. The transmitted value should correspond to the Unix time format and be measured in milliseconds. Please note that the radar module always starts counting again from 0 after a power loss.

Command ID	0x0004	
Description	Set system time	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
System time	uint64	New system time [ms] in Unix-format
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 20: Command 0x0004: Set system time

4.11 Command: Reset module

After transmission of this command the microcontroller unit of the radar module performs a reset. Until all sections have been rebooted and communication is possible again, one or two seconds have to be waited.

Command ID	0x0005	
Description	Reset MCU of radar module	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 21: Command 0x0005: Reset module

4.12 Command: Read radar parameters

This is one of the most important commands, as it is used to read out the current radar configuration. Without knowledge of these parameters, it is difficult to interpret or work with the transmitted data. The individual parameters and their meaning have already been explained in Table 2.

Command ID	0x000A	
Description	Read radar parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Radar parameters according to Table 2		
CRC	uint16	CRC Checksum

Table 22: Command 0x000A: Read radar parameters

4.13 Command: Set radar parameters

With this command, the radar parameters can be changed. The values should be within the intended value range. Otherwise, the respective value is corrected before it is accepted. This is marked by the bit "0x0002: Invalid Rx Data" in the status word. The new parameters are only accepted after the command, as a measurement may still be taking place in the background. The new parameters are taken over by the radar process through a reconfiguration.

If the command ID "0x000B" is sent, the parameters are also stored in the EEPROM so that they are available again after a power failure. If this is not desired, the command ID "0x800B" can be sent.

Command ID	0x000B or 0x800B	
Description	Set radar parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Radar parameters according to Table 2		
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 23: Command 0x000B or 0x800B: Set radar parameters

4.14 Command: Reset radar parameters

This command resets the radar parameters to default values, configures the radar process with them and stores them in the EEPROM.

Command ID	0x000C	
Description	Reset radar parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 24: Command 0x000C: Reset radar parameters

4.15 Command: Read out frontend parameters

This command is used to read out the front-end parameters that configure the transmission signal and the data acquisition. The individual parameters and their meaning have already been explained in Table 4.

Command ID	0x0010	
Description	Read out frontend parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Frontend parameters according to Table 4		
CRC	uint16	CRC Checksum

Table 25: Command 0x0010: Read out frontend parameters

4.16 Command: Set frontend parameters

With this command, the frontend parameters can be changed. The values should be within the intended value range. Otherwise, the respective value is corrected before it is accepted. This is marked by the bit "0x0002: Invalid Rx Data" in the status word. The new parameters are only accepted after the command, as a measurement may still be taking place in the background. The new parameters are taken over by the radar process through a reconfiguration.

If the command ID "0x0011" is sent, the parameters are also stored in the EEPROM so that they are available again after a power failure. If this is not desired, the command ID "0x8011" can be sent.

Command ID	0x0011	
Description	Set frontend parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Frontend parameters according to Table 4		
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 26: Command 0x0011: Set frontend parameters

4.17 Command: Reset frontend parameters

This command resets the frontend parameters to default values, configures the radar process and the frontend with them and stores them in the EEPROM.

Command ID	0x0012	
Description	Reset Frontend Parameters	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 27: Command 0x0012: Reset Frontend Parameters

4.18 Command: Read out Ethernet configuration

With this command, the current Ethernet configuration can be read out. The structure and possible values have already been mentioned in Table 5.

Command ID	0x0020	
Description	Read out Ethernet configuration	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Ethernet configuration corresponding to Table 5 including read only parameters		
CRC	uint16	CRC Checksum

Table 28: Command 0x0020: Read out Ethernet configuration

4.19 Command: Change Ethernet configuration

With this command, the current Ethernet configuration can be changed. This command can also be executed with the USB interface, e.g. to replace a forgotten configuration. The new parameters are only accepted after the command.

If the command ID "0x0021" is sent, the parameters are also stored in the EEPROM so that they are available again after a power failure. If this is not desired, the command ID "0x8021" can be sent.

Note that **the read only parameters are not sent here**, as they are fixed.

Command ID	0x0021	
Description	Change Ethernet configuration	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Ethernet configuration corresponding to Table 5 without read only parameters		
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 29: Command 0x0021: Change Ethernet configuration

4.20 Command: Reset Ethernet configuration

With this command, the Ethernet configuration can be reset to default values according to Table 5. This command can also be executed with the USB interface, e.g. to reset a forgotten configuration.

Command ID	0x0022	
Description	Reset Ethernet configuration	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 30: Command 0x0022: Reset Ethernet configuration

4.21 Command: Request Stream

This command is used to request a stream. This means that the radar module remembers the incoming connection and then sends the data to this address. This works for both the Ethernet and the USB interface.

In order for the module to send data on its own, the Radar Parameter "Continuous Measurement" must be set to "1". In addition, the Radar Parameter "Processing" and "Radar Cube" as well as the parameter "Stream_Variable" sent here determine the number and type of data sent (more on this in the "Streaming" section). In addition, the parameter "Stream_Mask" adjusts the header of the data packets.

Command ID	0x0023	
Description	Request a data stream	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Stream_Mask	uint16	Mask with bits to adjust the sent header (see Table 53)
Stream_Variable	uint16	Value and meaning depend on "Processing" and "Radar Cube
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 31: Command 0x0023: Request stream

4.22 Command: Start Ethernet Stream

With this command, the stream of any Ethernet connection can be activated. The prerequisite is that the specified port also exists in the Ethernet configuration. Unlike in "Command: Request Stream" connection must be specified here. The parameters "IF_Type", "Radar Port", "Host IP" and "Host Port" are used for this.

"IF_Type" can take the values 0, 1 or 2 and indicates the type of Ethernet port used. Here, 0 stands for the connection with which the request was made, just as with "Command: Request Stream". The value 1 stands for one of the TCP ports and the value 2 for one of the UDP ports.

"Radar Port" must be specified for values of "IF_Type" > 0 and selects the port of the selected Ethernet type. For "IF_Type" = 0, the value of "Radar Port" does not matter.

"Host IP" and "Host Port" are only relevant if "IF_Type" = 2, i.e. UDP. Otherwise, their values do not matter, but must be transmitted. These parameters indicate the host to which the data is to be sent. For example, an Ethernet host can be specified via the USB interface and a stream can be activated, provided that the requirements in the network are met.

In order for the module to send data on its own, the "Continuous Measurement" parameter in the Radar Parameter must be set to "1". In addition, the Radar Parameter "Processing" and "Radar Cube" as well as the parameter "Stream_Variable" sent here determine the number and type of data sent (more on this in the "Streaming" section). In addition, the parameter "Stream_Mask" adjusts the header of the data packets.

Command ID	0x0024	
Description	Start data streams via Ethernet	
Name	Data type	Description
Host → Radar		

ID	uint16	Command ID
Stream_Mask	uint16	Mask with bits to adjust sent header (see Table 53)
Stream_Variable	uint16	Value and significance depend on "Processing" and "Radar Cube
IF_Type	uint16	Kind of port: 0 = current connection, 1 = TCP, 2 = UDP
Radar Port	uint16	Port of the radar which should send data
Host IP	4 x uint8	IP address of the host to be sent to
Host Port	uint16	Port of the host to be sent to
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 32: Command 0x0024: Start Ethernet stream

4.23 Command: Stop Ethernet Stream

With this command, a running Ethernet stream can be stopped. If the addressed port is not a stream, this command has no effect. As in "Command: Start Ethernet Stream", the port of the stream must be specified by the parameters "IF_Type" and "Radar Port". Here, too, the value "0" for "IF_Type" means the connection with which this command was sent. In addition, the value 3 can also be specified here for "IF_Type", which simply deactivates streaming for all possible ports (TCP and UDP).

Basically, the radar module continues to listen to the connection with which streaming is taking place. However, since the stream is controlled via interrupts, it can happen that commands on this port are overheard during short measurement intervals. However, any other possible connection can also be used to stop a stream.

Command ID	0x0025	
Description	Stopping a data stream via the Ethernet interface	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
IF_Type	uint16	Kind of Port: 0 = current connection, 1 = TCP, 2 = UDP, 3 = all ports
Radar Port	uint16	Port of the radar which sends data (only for IF_Type = 1 or 2 important)
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 33: Command 0x0025: Stop Ethernet stream

4.24 Command: Stop USB Stream

With this command, a running USB stream can be stopped. No further parameters are necessary, as there is only one USB interface.

Command ID	0x0026	
Description	Stopping a data stream via the USB interface	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 34: Command 0x0026: Stop USB stream

4.25 Command: Request Multi Data Stream

This command requests a data stream similar to command "Request Stream" (4.21) but the stream started here can send data of any processed step like in command "Read Data" (4.26).

In order for the module to send data on its own, the Radar Parameter "Continuous Measurement" must be set to "1". The Radar Parameter "Processing" determines the possible data types which can be sent by the stream. Only data resulting by processed steps can be sent.

To activate data of certain steps, the parameter "Stream_Data_Mask" is used. In this bitmask, the first 7 bits stand for a step of the Radar Parameter "Processing", starting with bit0 = Range FFT, bit1 = Doppler FFT … . If no bit is set, raw data will be sent.

To be able to send any type of processed data in the same manner like streams started by the other commands, additional parameters are needed to adjust some of the data types. These parameters are:

- *Chirp_No/Bin_Int*: For range FFT data,
  - if RadarCube > 3: number of the chirp from which the data origins or 0xFFFF to request the whole data cube,
  - else: interval of range bins.
- *Bin_No*: For Doppler FFT data: number of the range from which the data origins or 0xFFFF to request the whole data cube.
- *Doppler_Format*: For Tracking list: format of Doppler spectra for each found target. 0 for not spectra, 1 for complex Doppler spectrum for each enable Rx channel for each target, >1 for magnitude Doppler spectrum for each target.

Like in the "Request Stream" command, also the parameter "Stream_Mask" is used to adjust the packet header.

The resulting data stream sends the data of the process steps requested in the same format as described in for the single data formats in chapter Streaming.

Command ID	0x0027	
Description	Start a data stream sending data from multiple processing steps	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Stream_Mask	uint16	Mask with bits to adjust sent header (see Table 53)
Stream_Data_Mask	uint16	Mask determining data to be sent
Chirp_Num/Bin_Int	uint16	Value adjusting sent range FFT data
Bin_Num	uint16	Value adjusting sent Doppler FFT data
Doppler_Format	uint16	Format of Doppler spectra send for targets
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 35: Command 0x0027: Start Multi Data Stream

4.26 Command: Configure Stream

With this command a stream can be requested like in the commands "Request Stream" (4.21) and "Request Multi Data Stream" (4.25). Furthermore some special stream settings can be adjusted here.

If multiple radar sensors are running at the same time in the same area, there is a chance of disturbing each other. With help of the parameters "Stream_Meas_Mode" and the "Start_Delays" plus multi- or broadcast messages, radar modules can be configured to measure in certain time slots (see section 5.6).

"Stream_Meas_Mode" can have the following values:

- 0: Continuous, the default mode. Data is sent in the given "MeasInterval" like the streams started by the other commands.
- 1: Triggered Start: A stream with the current interface is activated but waits for a "Trigger Stream" (4.27) command. If triggered and after waiting for "Start_Delay" milliseconds, the default continuous mode is joined. The delay value used is addressed by the trigger commands "Delay_Index" value.
- 2: Triggered Measurement: A stream with the current interface is activated but waits for a "Trigger Stream" command for each measurement. When triggered the radar waits for "Start_Delay" milliseconds before the measurement. The delay value used is addressed by the trigger commands "Delay_Index" value.

There are four possible "Start_Delay" values. The value of "Delay_Index" sent by the "Trigger Stream" (4.27) command decides which of them is used for triggering the next measurement.

The parameter "Stream_Data_Mode" decides if the data sent is determined by the Radar Parameter "ProcessStep" like in command "Request Stream (4.21) or if the parameter "Stream_Data_Mask" is used like in "Request Multi Data Stream" (4.25).

Command ID	0x0028	
Description	Configure and start/activate a possibly delayed data stream maybe waiting on a trigger command	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Stream_Data_Mode	uint16	Single (0) or multiple data mode (1)
Stream_Meas_Mode	uint16	Continuous (0), Triggered Start (1) or Triggered Measurement (2) mode as described above.
Start_Delays	4 x uint32	4 Delays in milliseconds before a triggered measurement starts. "Delay_Index" in command Command: Trigger Stream decides which of them is used.
Stream_Mask	uint16	Mask with bits to adjust sent header (see Table 53)
Stream_Data_Mask	uint16	Mask determining data to be sent
Chirp_Num/Bin_Int	uint16	Value adjusting sent range FFT data
Bin_Num	uint16	Value adjusting sent Doppler FFT data
Doppler_Format	uint16	Format of Doppler spectra send for targets
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 36: Command 0x0028: Configure Stream

4.27 Command: Trigger Stream

This command is used to trigger a stream when configured in the "Stream_Meas_Modes" triggered start or triggered measurement (see sections 4.26, 5.6). If this command is used for synchronization purposes, it should be sent as broadcast message or to a configured multicast group.

This command also sends a time value to update the radar system time. The transmitted value should correspond to the Unix time format and be measured in milliseconds.

The value of "Use_Time" determines how the sent time value should be interpreted. If a "1" is sent, the time value is set as new system time. If a "2" is sent, the time value is interpreted as a future timestamp used to trigger the next measurement where the value of the addressed "Start_Delay" is also recognized.

"Delay_Index" addresses one of the four delays adjusted by command "Configure Stream" (4.26).

Command ID	0x0029	
Description	Trigger a waiting stream	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Time	uint64	New system time [ms] in Unix-format
Use_Time	uint16	How the time value should be used: • 1: Time is set as new system time • 2: Time is a future time used to trigger the next measurement

		later • Else: Time is not used
Delay_Index	uint16	Index (0-3) to select one of the four possible delays set by command Command: Configure Stream
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 37: Command 0x0029: Trigger Stream

4.28 Command: Read Data

This is a basic command to read out any kind of data that has been processed by the triggered measurement like the multi data stream.

To determine the data to be sent, the bitmask "Data_Mask" is used. In this mask, the first 7 bits stand for a step of the Radar Parameter "Processing", starting with bit0 = Range FFT, bit1 = Doppler FFT … . If no bit is set (Data_Mask = 0), raw data will be sent.

In case of continuous measurement (Radar Parameter "CountinuousMeas" = 1), the parameter "Processing" defines the possible available data to be sent back. This means, even if the tracking list is requested for example, but Processing is set to 5 (CFAR), the requested "Data_Mask" would be cut like Data_Mask &= 0x1F, because the data of later processing steps is not available. In this case, the command would set the bit 2 (Invalid Rx Data) in the status word (see Table 9).

In case "ContinuousMeas" = 0 the highest bit in the "Data_Mask" determines the steps to be processed by the next measurement, so each requested data type will be sent back.

The data format of the different data types sent in this command is the same as in the commands described in the following sections. Because of this, some additional parameters are needed for some data types. These parameters have always to be sent, even if the respective bit for the data type is not set. The parameters are:

- *Chirp_No*: For raw and range FFT data: number of the chirp from which the data origins or 0xFFFF to request the whole data cube.
- *Bin_No*: For Doppler FFT data: number of the range from which the data origins or 0xFFFF to request the whole data cube.
- *Doppler_Format*: For Tracking list: format of Doppler spectra for each found target. 0 for not spectra, 1 for complex Doppler spectrum for each enable Rx channel for each target, >1 for magnitude Doppler spectrum for each target.

Since the amount of data sent by this command may vary because of detection and track lists, the radar sends the number of data bytes following in the answer right after the time stamp.

Note that the range FFT data is overwritten internally by Doppler FFT data when the radar processes further than the range FFT.

Note that the CFAR map is only valid if the radar processes only to the step CFAR. If the radar processes further (Detections, Tracking), the map is not computed/updated.

For additional information about the data sent in this command, see also the following command descriptions.

Command ID	0x0030	
Description	Read data	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Data_Mask	uint16	Mask determining data to be sent
Chirp_No	uint16	Number of the chirp of which the data should be transferred
Bin_No	uint16	Number of the range bin of which the data should be transferred
Doppler_Format	uint16	Format of Doppler spectra send for targets
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
Data Size	uint32	Number of data bytes following
if (Data_Mask == 0)		
Raw data as in Table 39 (only data loop)		
else {		
if (Data_Mask & 0x1) (Range FFT)		
Range FFT data as in Table 40 (only data loop)		
if (Data_Mask & 0x2) (Doppler FFT)		
Doppler FFT data as in Table 41 (only data loop)		
if (Data_Mask & 0x4) (Magnitude Map)		
Magnitude map as in Table 42 (only data loop)		
if (Data_Mask & 0x8) (Peak Map)		
Peak map as in Table 43 (only data loop)		
if (Data_Mask & 0x10) (CFAR Map)		
CFAR map as in Table 44 (only data loop)		
if (Data_Mask & 0x20) (Detection List)		
Detection list as in Table 46 (SysDopplerBin, SysSpeed, NumDetections and following loop)		
if (Data_Mask & 0x40) (Track List)		
Track list with Doppler spectra as in Table 48 (NumTracks and following loop)		
}		
CRC	uint16	CRC Checksum

Table 38: Command 0x0030: Read Data

4.29 Command: Read Raw Data

With this command, the unprocessed data can be read out. For this purpose, the radar signal processing is switched off and the ADC data are written unchanged into the RAM.

This command sends the data of a chirp, which is defined with the parameter "Chirp_No". If a variant with only one chirp was selected for the “Radar Parameter” “Radar Cube”, "Chirp_No" is irrelevant.

If the value "0xFFFF" is entered for "Chirp_No" and a "Radar Cube" with several chirps is set, the data for all chirps are sent, i.e. the entire data cube. For "Radar Cubes" with compressed data (see Table 3) this data wouldn't fit into RAM, so only zeros are sent in this case.

If the "Radar Parameter" "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

In addition, the number of transmitted data is influenced by the "Radar Parameter" "RxChannels". Data is only transmitted for activated channels. If a MIMO "Radar Cube" is activated, data can be sent for up to 12 channels, otherwise for up to 4 channels.

In Table 39, Max_Chan means the maximum possible number of channels. If a MIMO Radar Cube is active, Max_Chan = 12, otherwise Max_Chan = 4. Num_Smpl is the number of samples of the Radar Cube. For Radar Cubes with multiple chirps and 4 or 12 channels, the number of samples is twice the number of range bins (e.g. 256 range bins = 512 samples). For Radar Cubes with 2 complex channels, the number of range bins is equal to the number of samples. The value 1 must be entered for Num_Chirps, **unless** the value "0xFFFF" is sent for "Chirp_No". Then the number of chirps in the radar cube must be entered for Num_Chirps.

Command ID	0x0031	
Description	Read raw data	
Name	**Data type**	**Description**
	Host → Radar	
ID	uint16	Command ID
Chirp_No	uint16	Number of the chirp of which data should be transferred
CRC	uint16	CRC Checksum
	Radar → Host	
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
for (chirp = 0; chirp < *Num_Chirps*; chirp++)		
for (channel = 0; channel < *Max_Chan*; channel++)		
if (channel is enabled)		
ADC value	*Num_Smpl* x uint16	Digitized values of the received signal
CRC	uint16	CRC Checksum

Table 39: Command 0x0031: Read raw data

4.30 Command: Read Range FFT Data

With this command, the results of the range FFT can be read out. The data of the chirp defined with the parameter "Chirp_No" is sent, unless the value "0xFFFF" is specified for "Chirp_No". In this case, the data of all chirps, i.e. the entire data cube, is sent. For "Radar Cubes" with compressed data (see Table 3) only zeros are sent.

The number of transmitted data is defined via the "Radar Parameter" "RxChannels", "MinRangeBin" and "MaxRangeBin". A distinction is made here between two cases: "Radar Cubes" with several chirps and 4 or 12 channels and "Radar Cubes" with only one chirp and 2 complex channels.

If the "Radar Parameter" "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

In the case of several chirps and 4 or 12 channels, a complex value is sent for each activated channel for each permitted range bin of the defined chirp.

In the case of only one chirp with 2 complex channels, the parameter "Chirp_No" is ignored. The "Radar Parameter" "RxChannels" has a slightly different meaning here. Since only 2 channels are available, the first two bits (0x1, 0x2) are used to address the complex data and the bits 0x4, and 0x8 to address the magnitude data, which are also calculated in this case. First, for each activated channel (0x1 and/or 0x2), a complex value is transmitted for each allowed range bin. This is followed by a magnitude value for each activated channel (0x4 and/or 0x8) for each permitted range bin. Finally, the results of a threshold comparison (channel, range bin, magnitude value) follow here.

In the case of a radar cube with multiple chirps, a complex value is sent for each active channel and each activated range bin. The possible number of channels Max_Chan is 12 for MIMO Radar Cubes, otherwise 4. The number of transmitted chirps Num_Chirps is 1, unless the value "0xFFFF" is specified for "Chirp_No". Then Num_Chirps corresponds to the number of chirps of the Radar Cube.

Command ID	0x0032	
Description	Read Range FFT Data	
Name	**Data type**	**Description**
Host → Radar		
ID	uint16	Command ID
Chirp_No	uint16	Number of the chirp of which data should be transferred
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
if (Radar Cube > 3)		
for (chirp = 0; chirp < *Num_Chirps*; chirp++)		
for (channel = 0; channel < *Max_Chan*; channel++)		
if (channel enabled)		
for (n = MinRangeBin; n <= MaxRangeBin; n++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
else		
for (channel = 0; channel < 2; channel++)		
if (channel enabled)		
for (n = MinRangeBin; n <= MaxRangeBin; n++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
for (channel = 2; channel < 4; channel++)		
if (channel enabled)		
for (n = MinRangeBin; n <= MaxRangeBin; n++)		
Mag	uint16	Magnitude value of FFT result
Channel	uint16	Channel (1 or 2) in which the first peak was found that exceeded the threshold ("DetectionThresh"). 0 if no peak was found.

Bin	uint16	Range Bin of the channel where the first peak was found that exceeded the threshold ("DetectionThresh"). 0 if no peak was found.
Peak Mag	uint16	Magnitude value of the peak if one was found.
endif		
CRC	uint16	CRC Checksum

Table 40: Command 0x0032: Read Range FFT Data

4.31 Command: Read out Doppler FFT data

With this command, the results of the Doppler FFT can be read out. The data of the range bin defined with the parameter "Bin_No" is sent, unless the value "0xFFFF" is specified for "Bin_No". In this case, the data of all range bins, i.e. the entire data cube, is sent. For "Radar Cubes" with compressed data (see Table 3) only zeros are sent.

The number of transmitted data is defined via the "Radar Parameter" "RxChannels", "MinDopplerBin" and "MaxDopplerBin". If "Bin_No" = 0xFFFF, the number of transmitted data is additionally limited by "MinRangeBin" and "MaxRangeBin".

If the "Radar Parameter" "ContinuousMeas" has the value 0, this command starts a measurement in the correct process step.

Please note that the "Radar Parameter" "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin, which corresponds to the velocity 0, is in the middle.

This command only sends meaningful data if a setting with multiple chirps has been selected for the "Radar Parameter" "Radar Cube".

In Table 41, Max_Chan has the value 12 for MIMO Radar Cubes and otherwise the value 4.

Command ID	0x0033	
Description	Read out Doppler FFT data	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Bin_No	uint16	Number of the range bin of which data should be transferred
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Zeitstempel	uint64	Radar system time at the time of measurement in [ms]
if (Bin_No == 0xFFFF)		
for (rbin = MinRangeBin; rbin <= Num_Range_Bins; rbin++)		
for (channel = 0; channel < Max_Chan; channel++)		
if (channel enabled)		
for (dbin = MinDopplerBin; dbin <= MaxDopplerBin; dbin++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
else		
for (channel = 0; channel < Max_Chan; channel++)		

if (channel enabled)		
for (dbin = MinDopplerBin; dbin <= MaxDopplerBin; dbin++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
endif		
CRC	uint16	CRC Checksum

Table 41: Command 0x0033: Read out Doppler FFT data

4.32 Command: Read magnitudes of the Range Doppler Map

With this command, the magnitudes of the Range Doppler map can be read out. These are the result of a non-coherent integration of the results after the Doppler FFT of the four or twelve channels and the calculation of the magnitudes using internal (Mag2 and Log2) functions.

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

The number of data sent is configured by the Radar Parameter "RxChannels", "MinRangeBin", "MaxRangeBin", "MinDopplerBin" and "MaxDopplerBin".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right sides of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin, which corresponds to the speed 0, is in the middle. This command only sends meaningful data if a setting with multiple chirps has been selected for the Radar Parameter "Radar Cube".

Command ID	0x0034	
Description	Read magnitudes of the Range Doppler map	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = MinRangeBin; r <= MaxRangeBin; r++)		
Mag	uint16	Magnitude value of Range-Doppler-map
CRC	uint16	CRC Checksum

Table 42: Command 0x0034: Read magnitudes of the range Doppler map

4.33 Command: Read Peak Map

With this command the peak map can be read out, which is the result of a peak search within the magnitudes of the range Doppler map from command 0x0034 (section 4.32). To decide whether a peak is present or not, the Radar Parameter "PeakSearchThresh" is used in conjunction with an

adaptive threshold. The resulting peak map is sent as a bitmap. Each bin of the range Doppler map corresponds to a bit in the peak map. The bit is "1" if the threshold was exceeded, or "0" if not.

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

The number of data sent is configured by the Radar Parameter "MinDopplerBin" and "MaxDopplerBin". "MinRangeBin" and "MaxRangeBin" have no influence on the number of bytes in this case, since 32 range bins are combined into one word. Thus, the number of 32-bit words for the range bins results with Num_RWords = MaxRangeBins/32, where MaxRangeBins corresponds to the number of possible range bins depending on the current "Radar Cube".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin corresponding to the velocity 0 is in the centre.

The peak map is used to check the settings of the signal processing, as it will be used in further steps. In addition, only meaningful data is sent if a setting with several chirps has been selected for the radar parameter "Radar Cube".

Command ID	0x0035	
Description	Read peak map	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = 0; r < Num_RWords; r++)		
MapValue	uint32	Part of Peak-Map with 32 Bits for 32 Range Bins (big endian)
CRC	uint16	CRC Checksum

Table 43: Command 0x0035: Read peak map

4.34 Command: Read CFAR Map

This command can be used to read out the CFAR map, which is the result of applying a CFAR algorithm to the peak map from command 0x0035 in Section 4.33. CFAR is configured via the Radar Parameter "CfarThresh", "CfarGuardInt" and "CfarWindowSize". The resulting CFAR map is sent as a bitmap. Each bin of the range Doppler map corresponds to a bit in the CFAR map. The bit is "1" if a peak has been detected, or "0" if not.

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

The number of data sent is configured by the Radar Parameter "MinDopplerBin" and "MaxDopplerBin". "MinRangeBin" and "MaxRangeBin" have no influence on the number of bytes in

this case, since 32 range bins are combined into one word. Thus, the number of 32-bit words for the range bins results with Num_RWords = MaxRangeBins/32, where MaxRangeBins corresponds to the number of possible range bins depending on the current "Radar Cube".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin corresponding to the velocity 0 is in the center.

The CFAR map is used to check the settings of the signal processing, as it will be used in further steps. In addition, only meaningful data is sent if a setting with multiple chirps has been selected for the Radar Parameter "Radar Cube".

Command ID	0x0036	
Description	Read CFAR map	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = 0; r < Num_RWords; r++)		
MapValue	uint32	Part of Peak-map with 32 Bits for 32 Range Bins (big endian)
CRC	uint16	CRC Checksum

Table 44: Command 0x0036: Read CFAR map

4.35 Command: Read all Maps

With this command, the range Doppler map from section 4.32, the peak map from section 4.33 and the CFAR map from section 4.34 can be read out at once. This command is primarily used to visually observe the effects of various parameter settings, provided the data can be displayed. The data is processed up to the process step CFAR (see parameter "Processing" of the Radar Parameter).

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

The different maps are simply sent one after the other as described in the corresponding sections. The influence of different Radar Parameter on the amount of data is also retained.

Command ID	0x0037	
Description	Read all maps	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID

Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
Data of Range-Doppler-map like in Table 42		
Data of Peak-map like in Table 43		
Data of CFAR-map like in Table 44		
CRC	uint16	CRC Checksum

Table 45: Command 0x0037: Read all maps

4.36 Command: Read Detections

With this command, the detections found can be read out. To determine the detections, the object angles in azimuth direction are determined for all positions remaining after CFAR. In the case of MIMO radar cubes, the elevation angle is also estimated. Afterwards, candidates at the same distance and almost the same angle are grouped together. The maximum number of transmitted detections is set by the Radar Parameter "NumTargets". In case the Radar Parameter "SpeedEstimation" is set to a value greater zero, the values "SysDopplerBin" and "SysSpeed" are sent before "NumDetections".

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

Command ID	0x0038	
Description	Read detections	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
if ("SpeedEstimation" > 0)		
SysDopplerBin	int16	Estimated Doppler bin representing radar system speed
SysSpeed	int16	Interpolated radar system speed [m/s * 100]
NumDetections	uint16	Number of detections found
for (n = 0; n < NumDetections; n++)		
Range Bin	uint16	Range Bin in which the detection was found
Doppler Bin	int16	Doppler Bin in which the detection was found
Magnitude	uint16	Magnitude value of the detection (20*log10(Amplitude)
Azimuth	int16	Azimuth angle [°]
Elevation	int16	Elevation angle [°]
CRC	uint16	CRC Checksum

Table 46: Command 0x0038: Read detections

4.37 Command: Read Tracks

With this command, the results of the last processing step, tracking, can be read out. The detections, which can be read out with the command from section 4.36, are processed further by means of a tracking algorithm. This can be adjusted with the Radar Parameter "MaxTracks", "MaxHorSpeed", "MaxVerSpeed", "MaxAccel", "MaxRangeError", "MinConfirm", "AssignLimit" and "MergeLimit". The result is a target list, in which each target gets a fixed number, which it keeps over time, if it is found again. Targets, which are not found again in the meantime, can be predicted for a certain time.

Furthermore, the magnitudes of the Doppler spectra of the targets found can be sent to the second processor (MCU/DSP) for classification. For this, the second processor must be available and the Radar Parameter "DspDopplerProc" must be set to 1. If this is the case, the target list receives additional entries for the classification result. The correct result becomes available only after a certain number of measurements, because Doppler sequences are evaluated.

In case the Radar Parameter "SpeedEstimation" is set to a value greater zero, the values "SysDopplerBin" and "SysSpeed" are sent before "NumTracks".

If the Radar Parameter "ContinuousMeas" has the value 0, this command first starts a measurement in the correct process step.

Command ID	0x0039	
Description	Reading out the tracking results	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
if ("SpeedEstimation" > 0)		
SysDopplerBin	int16	Estimated Doppler bin representing radar system speed
SysSpeed	int16	Interpolated radar system speed [m/s * 100]
NumTracks	uint16	Number of tracks sent
for (n = 0; n < NumTracks; n++)		
ID	uint16	ID of the target (0 to MaxTracks-1)
Range	float32	Distance of target in [m]
Speed	float32	Speed of target in [m/s]
Magnitude	uint16	Magnitude value (20*log10(Amplitude)
Azimuth	float32	Azimuth angle [°]
Elevation	float32	Elevation angle [°]
LifeTime	uint32	Value corresponding to the confirmations (retrievals) of the target
if („DspDopplerProc" == 1)		
InferenceResult	8 x uint16	Result vector of the neural network inference. 8 probabilities [% * 100]
CRC	uint16	CRC Checksum

Table 47: Command 0x0039: Read tracks

4.38 Command: Reading Doppler Spectra from Tracks

With this command, in addition to the tracks from section 4.37, the Doppler spectra, which can probably be assigned to the tracks, can be read out. The Doppler spectra can be requested for all activated channels in complex values or as magnitudes of the range Doppler map.

Since the distance of a target is calculated in meters after tracking, it is subsequently calculated to which range bin the target best fits. The Doppler spectrum is then selected with the range bin determined in this way. If the Radar Parameter "ContinuousMeas" has the value 0, this command starts a measurement in the correct process step.

In case the Radar Parameter "SpeedEstimation" is set to a value greater zero, the values "SysDopplerBin" and "SysSpeed" are sent before "NumTracks".

Please note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin, which corresponds to the velocity 0, is in the center.

The parameter "Spec_Format" determines the data format in which the Doppler spectra are sent. It can take the values 0 =no Doppler spectra, 1 = complex values for each activated channel and 2 = magnitudes. The number of values is limited by the Radar Parameter "MinDopplerBin" and "MaxDopplerBin".

In Table 48 the parameter Max_Chan stands for the maximum possible number of channels. This is 12 for MIMO radar cubes, otherwise 4.

Command ID	0x003A	
Description	Reading out the tracking results and their Doppler spectra	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
Time stamp	uint64	Radar system time at the time of measurement in [ms]
if ("SpeedEstimation" > 0)		
SysDopplerBin	int16	Estimated Doppler bin representing radar system speed
SysSpeed	int16	Interpolated radar system speed [m/s * 100]
NumTracks	uint16	Number of tracks sent
for (n = 0; n < NumTracks; n++)		
ID	uint16	ID of the target (0 to MaxTracks-1)
Range	float32	Distance of target in [m]
Speed	float32	Speed of target in [m/s]
Magnitude	uint16	Magnitude value of the target (20*log10(Amplitude)
Azimut	float32	Azimuth angle [°]
Elevation	float32	Elevation angle [°]
LifeTime	uint32	Value corresponding to the confirmations (retrievals) of the target

if („DspDopplerProc" == 1)		
InferenceResult	8 x uint16	Result vector of the neural network inference. 8 probabilities [% * 100]
if (Spec_Format == 1)		
for (channel = 0; channel < *Max_Chan*; channel++)		
if (channel enabled)		
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
Re	int16	Real part of the Doppler FFT result
Im	int16	Imaginary part of the Doppler FFT result
else if (Spec_Format > 1)		
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
Magn.	uint16	Magnitude of the Doppler spectrum
endif (Spec_Format)		
CRC	uint16	CRC Checksum

Table 48: Command 0x003A: Reading Doppler spectra from tracks

4.39 Command: Configure Sector Filtering

A typical area monitored by a radar system contains different kinds of objects as streets, crossings, buildings or vegetation areas. It some applications, it makes sense to monitor only a certain part of an area, e.g. a lane of the road or a critical crossing. To enable discrimination between "active" and "passive" areas, an algorithm called Sector Filtering was implemented as part of the radar firmware. The main idea is to divide the measurement area into small sectors which can be switched on/off by the user or some automatic process.

The command 0x0040 allows different actions depending on the parameter *Code*. Please note that the Sector Filtering algorithm is only active if the radar parameter "SectorFiltering" is set (Table 2). Otherwise, the modification of the sector map will have no effect. A customized programming of the Sector Filtering algorithm can be done in the following order:

- install the radar sensor to its final position,
- start measurement process (periodic commands or continuous mode),
- clear sector map (command 0x0040 + code 3),
- start Teaching Mode (command 0x0040 + code 4),
- activate sectors by walking or driving across the "interesting" area,
- stop Teaching Mode (command 0x0040 + code 5),
- write sector map to EEPROM for permanent storage (command 0x0040 + code 2).

Command ID	0x0040	
Description	Configure sector filtering algorithm	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
Code	uint16	Additional configuration code (refer to Table 50)
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID

Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 49: Command 0x0040: Configure sector filtering algorithm

Code	Description
1	Read sector map from EEPROM
2	Write current sector map to EEPROM
3	Clear current sector map
4	Start Teaching Mode
5	Stop Teaching Mode
other	no action

Table 50: Functionality of the additional code of command 0x0040

4.40 Command: Read Sector Map

The sector map consists of 20 cells in range direction (0 to 100m in steps of 5m) and 18 cells in view angle direction (-90° to 90° in steps of 10°). In radar memory, each sector is represented by a byte, so the total size of the map is 20×18=360 bytes. Active sectors have the value 1 otherwise 0. The command 0x0041 can be used to read the complete sector map from radar for visualization reasons or for manual modification.

Command ID	0x0041	
Description	Read current sector map from radar	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
for (n = 0; n < 360; n++)		
Sector	uint8	Value of a particular sector (outer loop: range)
CRC	uint16	CRC Checksum

Table 51: Command 0x0041: Read sector map from radar

4.41 Command: Write Sector Map

After manual modification, the sector map can be written back to the radar sensor. Therefore, the same data order as in the command 0x0041 is used. Please note, that the new sector map is directly written to the radar's memory and can be instantly used for target filtering. For permanent storage, the map must be written to EEPROM by using the command 0x0040 with code 2 (refer to Section 4.39).

Command ID	0x0042	
Description	Write sector map to radar	
Name	Data type	Description
Host → Radar		
ID	uint16	Command ID
for (n = 0; n < 360; n++)		
Sector	uint8	Value of a particular sector (outer loop: range)
CRC	uint16	CRC Checksum
Radar → Host		
ID	uint16	Command ID
Status	uint16	Status word
CRC	uint16	CRC Checksum

Table 52: Command 0x0042: Write sector map to radar

5 Streaming

Streaming the results of the radar signal processing is probably the most effective way to read out the data. To be able to use this function, various settings must be made.

5.1 Settings

The continuous measurement must be switched on, i.e. the Radar Parameter "ContinuousMeas" must have the value 1. In addition, a measuring interval should be specified (parameter "MeasInterval"). The processing operation will try to keep the given interval. However, if the signal processing or the copying of the data for dispatch takes longer, the specified interval cannot be adhered to. For example, the calculation of the "Combining" process step takes >16ms for 128 chirps including measurement. In contrast, measurement and evaluation of 1024 range samples from two complex channels and one chirp only take approx. 330us. Furthermore, the parameter "Processing" must be selected according to the desired data. In addition, the value of the parameter "Radar Cube" plays a role for the number of data and also for the measurement.

5.2 Internal Processes

The data are always sent after the end of the set signal evaluation. The MCU in which the program runs has several processor cores. In one of them the measurement and evaluation is done, in another one the communication with the host is done. The sending of the Doppler spectra to the second processor takes place in the third core.

When the measured data has been processed, the core for communication with the host is triggered by an interrupt. The host immediately copies the corresponding data and starts to send them via the set Ethernet ports. After copying and before the actual sending, the data are released, so that the other processor core can measure again, if the set measuring interval allows it. So it can happen that a dataset cannot be sent for every measurement, if the sending takes too long. This can be recognized if the Measurement counter is appended. Nevertheless, a current measurement will always be sent. If the amount of data to be sent is larger than the internal buffer limit, then the data to be copied must be sent in between, which can lead to delays until the next measurement, since these are blocked until then.

5.3 Starting a Stream

To start a stream, the settings from section 5.1 should be made first. A stream can now be started via one of the three commands "Command: Request Stream", "Command: Start Ethernet Stream" and “Command: Request Multi Data Stream”. The easiest way to do this is with the command "Command: Request Stream", because the radar module simply remembers the current connection, activates the stream for it and sends the results of process step specified by the Radar Parameter “Processing”. A stream via the USB interface can only be started with the “Command: Request Stream”. In a similar easy way “Command: Request Multi Data Stream” starts a stream with the difference, that here data from multiple processing steps can be read. In all commands starting a stream, additional parameters must be sent to the radar.

The parameter **Stream_Mask** is a bit mask with bits according to Table 53. If bit 0x0001 is active, first of all the pattern 0xAA55CC33 is always sent, which is used to find the beginning of the data set, which can be helpful if data sets are simply stored seamlessly. If bit 0x0002 is active, the current value of the measurement counter is appended. The Measurement counter starts at 0 when

the radar is restarted and is incremented by one after each measurement. If bit 0x0004 is active, a CRC checksum is calculated over the entire data package and appended at the end. However, this takes some time, despite hardware acceleration.

Bit	Description
0x0001	Synchronization pattern. The first four bytes of a data set are always 0xAA55CC33
0x0002	The current measurement counter (four bytes) is sent along.
0x0004	Activates the CRC calculation over all sent data and appends the result to the end.
0x0100	Especially for Range FFT Data and "Radar Cube" <= 3. Limits the number of range bins by +- the value around the found target, which was passed as "Stream Variable".
0x0200	If the bit 0x0100 is active, a fixed number of range bins is always sent, regardless of the position of the target found. The number corresponds to "Stream Variable" * 2 + 1

Table 53: Meaning of the bits of the Stream_Mask

The bits 0x0100 and 0x0200 are special for measurements with one chirp and 2 complex channels. If neither of them is set, the number of transmitted range bins is limited by the Radar Parameter "MinRangeBin" and "MaxRangeBin". If bit 0x0100 is set, a maximum number of range bins, given by the value of "Stream_Variable", are sent before and after a found target. The search range is limited by "MinRangeBin" and "MaxRangeBin". If no target is found, the middle of the search range is taken as the target bin. If a target is found near the upper or lower bin limit, the number of range bins sent is reduced. This is prevented by bit 0x0200. This ensures that a constant number of range bins is always sent. For example, if a target is found near the lower bin limit (MinRangeBin), the center of the target range shifts upwards. Therefore the maximum value for "Stream Variable" in this case is also (MaxRangeBin - MinRangeBin)/2 - 1.

The parameter **Stream_Variable** can also have a meaning for other values of "RadarCube" and "Processing". Table 54 shows the possible meanings so far. Here for "Chirp Number" and "Range Bin No.", as in the commands, the value "0xFFFF" can also be set, whereby the entire data cube is sent.

Radar Cube \ Processing	0 (No Processing)	1 (Range FFT)	2 (Doppler FFT)	7 (Tracking)
0 (256 x 1 x 2 cmplx)	None	Bin interval (+-)	not possible	not possible
1 (512 x 1 x 2 cmplx)	None	Bin interval (+-)	not possible	not possible
2 (1024 x 1 x 2 cmplx)	None	Bin interval (+-)	not possible	not possible
3 (2048 x 1 x 2 cmplx)	None	Bin interval (+-)	not possible	not possible
4 (64 x 64 x 4)	Chirp number	Chirp number	Range Bin No.	Doppler Fmt.[1)]
5 (64 x 128 x 4)	Chirp number	Chirp number	Range Bin No.	Doppler Fmt.[1)]
…	Chirp number	Chirp number	Range Bin No.	Doppler Fmt.[1)]
19 (256 x128 x 3 x 4)	Chirp number	Chirp number	Range Bin No.	Doppler Fmt.[1)]
20 (512 x 64 x 3 x 4)	Chirp number	Chirp number	Range Bin No.	Doppler Fmt.[1)]

Table 54: Meaning of the value of "Stream_Variable"

1) "Doppler Fmt." is the format of the Doppler data sent for tracks. 0 = none, 1 = complex spectra for each active Rx channel, 2 = magnitude spectrum, see also "Spec_Format" in Table 48.

5.4 Multi Data Stream

A special case to be mentioned is the multi data stream. The stream discussed so far, sends the final result of the process step adjusted. The multi data stream can send the results of all steps processed until the one adjusted, with some limitations:

- If raw data is requested, no other data can be sent
- Once the Doppler FFT has been processed, the range FFT data has been overwritten
- A valid CFAR map can only be sent for Processing = CFAR

For starting a multi data stream, multiple parameters are needed. **Stream_Mask** has been already discussed in section 5.3. To determine the data to be sent, the parameter **Stream_Data_Mask** is used. Table 55 shows the possible bits of this bitmask and also a reference to the section, with the description how the data is transmitted for the respective data type. In these sections, only the data loops are important.

Bit	Description	Reference
no bit set	Raw data	5.5.1
0x0001	Range FFT data	5.5.2 & 5.5.3
0x0002	Doppler FFT data	5.5.4
0x0004	Magnitude Map	5.5.5
0x0008	Peak Map	5.5.6
0x0010	CFAR Map	5.5.7
0x0020	Detections	5.5.8
0x0040	Tracks + Doppler	5.5.9

Table 55: Possible values of Stream_Data_Mask

Also a parameter for each processing step, where the "*Stream_Variable*" is used, must be given. These parameters are here **Chirp_Num/Bin_Int** for raw and range FFT data, **Bin_Num** for Doppler FFT data and **Doppler_Format** for Tracks. The function of these parameters is the same as described in section 5.3 since the data is also sent in the same format, just in a row.

Another difference to the other stream is that each data packet header contains the number of data bytes in the message, because the amount of data bytes may vary due to the detection and track lists and can also be very large, dependent on chosen data formats, so that the message is separated into many single packets.

If a line in Table 56 is highlighted in green, this value is optional and depends on the set bits in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
Data Mask	uint16	Current adjusted **Stream_Data_Mask** according to Table 55
Data Size	uint32	Number following data bytes
if (Stream_Data_Mask == 0) (Raw Data)		
Raw data like in Table 58: Data package raw data (only data loop)		
else {		
if (Stream_Data_Mask & 0x1) (Range FFT)		

Range FFT data like in Table 59 or Table 60 (only data loops) dependent on *RadarCube*		
if (Stream_Data_Mask & 0x2) (Doppler FFT)		
Doppler FFT data like in Table 61 (only data loop)		
if (Stream_Data_Mask & 0x4) (Magnitude Map)		
Magnitude map data like in Table 62 (only data loop)		
if (Stream_Data_Mask & 0x8) (Peak Map)		
Peak map data like in Table 63 (only data loop)		
if (Stream_Data_Mask & 0x10) (CFAR Map)		
CFAR map data like in Table 64 (only data loop)		
if (Stream_Data_Mask & 0x20) (Detections)		
Detection list like in Table 65 (NumDetections and following loop)		
if (Stream_Data_Mask & 0x40) (Tracks)		
Track list like in Table 66 (NumTracks and following loop)		
}		
CRC	uint16	CRC Checksum (optional)

Table 56: Format of a multi data stream message

5.5 Single Data Formats

As already mentioned, the sent data depend on the Radar Parameter "RadarCube" and "Processing". In the following the structure of the implemented data formats follows.

In each dataset always timestamp and status word are sent. Here the timestamp corresponds to the internal time in milliseconds at which the measurement was started. The status word is a bit mask and gives information about internal states at the time of the measurement. It can take combinations of the values from Table 57, which also corresponds to the values from Table 9.

Bit	Name	Description
0x0000	OK	No events
0x0010	FE Error	Problem with the frontend hardware.
0x0020	FE Temp Error	Temperature on the frontend is too high and the frontend has been switched off.
0x0100	Global Error Occurred	An error has occurred and has not yet been corrected.
0x0200	Global Error Logged	An error has occurred, perhaps already corrected, and noted.

Table 57: Bits of status words (streaming)

5.5.1 Processing = 0 (No Processing)

Similar to “Command: Read Raw Data” the raw data of a measurement is transmitted. These are limited only by the radar parameter "RxChannels". "Stream Variable" selects the chirp whose data is to be sent. However, if "Stream Variable" is set to "0xFFFF", data for all chirps will be sent. In case a “Radar Cube” with compressed data is active, only zeros will be sent.

In Table 58 Num_Smpl means the number of samples, which depends on the Radar Parameter "Radar Cube". For Radar Cubes with multiple chirps and 4 channels, the number of samples is

twice the number of range bins (e.g. 256 range bins = 512 samples). For Radar Cubes with 2 complex channels the number of range bins is equal to the number of samples.

Num_Chirps has the value 1, unless the value "0xFFFF" has been set for "Stream Variable". Then Num_Chirps corresponds to the number of chirps of the Radar Cube.

Max_Chan has the value 12 for MIMO Radar Cubes, otherwise 4.

If a row in Table 58 is highlighted in green, this value is optional and depends on the bits set in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (chirp = 0; chirp < *Num_Chirps*; chirp++)		
for (channel = 0; channel < *Max_Chan*; channel++)		
if (channel enabled)		
ADC Value	*Num_Smpl* x uint16	Digitized value of the received signal
CRC	uint16	CRC Checksum (optional)

Table 58: Data package raw data

5.5.2 Processing = 1 (Range FFT), RadarCube > 3

In this configuration a Range FFT is performed over several chirps and the data set of one chirp is sent. The chirp whose data is sent is selected with "Stream Variable". However, if "Stream Variable" is set to "0xFFFF", data will be sent for all chirps. In case a "Radar Cube" with compressed data is active, only zeros will be sent. The number of sent data is limited by the Radar Parameter "RxChannels", "MinRangeBin" and "MaxRangeBin".

Num_Chirps has the value 1 here, unless the value "0xFFFF" was set for "Stream Variable". Then Num_Chirps corresponds to the number of chirps of the Radar Cube.

Max_Chan has the value 12 for MIMO Radar Cubes, otherwise 4.

If a row in Table 59 is highlighted in green, this value is optional and depends on the bits set in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (chirp = 0; chirp < *Num_Chirps*; chirp++)		
for (channel = 0; channel < *Max_Chan*; channel++)		
if (channel enabled)		
for (n = *MinRangeBin*; n <= *MaxRangeBin*; n++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result

CRC	uint16	CRC Checksum (optional)

Table 59: Data package range FFT for a couple of chirps

5.5.3 Processing = 1 (Range FFT), RadarCube <= 3

In this configuration only one chirp is measured and a complex FFT is calculated. After the FFT, the magnitudes of the results are additionally calculated and a peak search is performed using the threshold "DetectionThresh" of the Radar Parameter. This search only takes place within the specified range from "MinRangeBin" to "MaxRangeBin" and is aborted as soon as a peak is found that lies above the threshold. As a result, channel, range bin number and magnitude value are appended. If no peak is found, the bin where the last peak was found is used up to three times to determine the range whose data will be sent if bit 0x0100 of the stream mask is set. Otherwise the middle of the range, indicated by "MinRangeBin" and "MaxRangeBin", is used. In any case, if no peak was found, a zero is sent for channel and magnitude value.

As already mentioned in section 5.3, the number of sent Range Bins is determined here by two bits "Stream Mask".

If no bit is set, the number of range bins sent is: Num_Bins = MaxRangeBin - MinRangeBin + 1.

If bit 0x0100 is set, the number depends on the position of the peak found and the value of "Stream Variable" (see Table 54). As an example, MinRangeBin = 100, MaxRangeBin = 700 and StreamVariable = 250. If a target is found at Range Bin Number 420, the range Bin 170 to Bin 670 would be Num_Bins = 501 Bins. But if a target is found at Range Bin Number 260, the range would be Bin 100 to Bin 510 and only Num_Bins = 411 Bins would be transmitted.

But if additionally the bit 0x0200 is set, the number of transmitted bits is fixed. This means that the area around a peak found remains the same size (2 * stream variable + 1) and only shifts. In the example just given, a target is again found at Bin Number 260. Here the range would now be Bin 100 to Bin 600 and thus Num_Bins = 501.

To enable the highest possible data rate, i.e. the smallest measurement interval (1ms), this data should be read out with a UDP port. In addition, the number of bytes should be limited to an Ethernet MTU (Maximum Transmission Unit) (approx. 1500 bytes), so that only one packet is sent. In concrete terms, this means a maximum value for "Stream Variable" of 90 for the UDP packet.

If a line in Table 60 is highlighted in green, this value is optional and depends on the bits set in "Stream Mask".

Please note that the imaginary part is always sent first here!

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (channel = 0; channel < 2; channel++)		
if (channel enabled)		
for (n = 0; n < *Bin_Num*; n++)		
Im[n]	int16	Imaginary part of FFT result

Re[n]	int16	Real part of FFT result
CRC	uint16	CRC Checksum (optional)

Table 60: Data package Range FFT Data for 2 complex Channels

5.5.4 Processing = 2 (Doppler FFT)

The data of this process step can only be transferred for radar cubes with multiple chirps.

With this command the results of the Doppler FFT are transmitted. StreamVariable" can be used to select a range bin whose Doppler spectrum is to be transmitted. If the value "0xFFFF" is set for "StreamVariable", the Doppler spectra of all active range bins are transmitted. In case a "Radar Cube" with compressed data is active, only zeros will be sent. The amount of data can be defined via the Radar Parameter "RxChannels", "MinDopplerBin" and "MaxDopplerBin", as well as "MinRangeBin" and "MaxRangeBin" if necessary.

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin corresponding to the velocity 0 is in the center.

Max_Chan has the value 12 for MIMO Radar Cubes, otherwise 4.

If a line in Table 61 is highlighted in green, this value is optional and depends on the bits set in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
if (StreamVariable == 0xFFFF)		
for (r = MinRangeBin; r <= MaxRangeBin; r++)		
for (channel = 0; channel < Max_Chan; channel++)		
if (channel enabled)		
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
else		
for (channel = 0; channel < Max_Chan; channel++)		
if (channel enabled)		
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
Re	int16	Real part of FFT result
Im	int16	Imaginary part of FFT result
endif		
CRC	uint16	CRC Checksum (optional)

Table 61: Data package Doppler FFT Data

5.5.5 Processing = 3 (Combining)

The data of this process step can only be transferred for radar cubes with multiple chirps.

The magnitudes of the range Doppler map are transmitted here. These are the result of a non-coherent integration of the results after the Doppler FFT of the four channels and the calculation of the magnitude using internal (Mag2 and Log2) functions.

The number of data sent is configured by the Radar Parameter "RxChannels", "MinRangeBin", "MaxRangeBin", "MinDopplerBin" and "MaxDopplerBin".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin, which corresponds to the velocity 0, is in the center.

If a line in Table 62 is highlighted in green, this value is optional and depends on the set bits in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = MinRangeBin; r <= MaxRangeBin; r++)		
Mag[d][r]	uint16	Magnitude value of the Range-Doppler-Map
CRC	uint16	CRC Checksum (optional)

Table 62: Data package Range-Doppler-Map Magnitude

5.5.6 Processing = 4 (Peak-Map)

The data of this process step can only be transmitted for radar cubes with multiple chirps.

The peak map is sent, which is the result of a peak search within the magnitudes of the range Doppler map from section 5.5.5. To decide whether a peak is present or not, a threshold is calculated using the Radar Parameter "PeakSearchThresh" and the history of previous measurements. The resulting peak map is sent as a bitmap. Each bin of the range Doppler map corresponds to a bit in the peak map. The bit is "1" if the threshold has been exceeded, or "0" if not.

The number of sent data is configured by the Radar Parameter "MinDopplerBin" and "MaxDopplerBin". "MinRangeBin" and "MaxRangeBin" have no influence on the number of bytes in this case, since 32 range bins are combined into one word. Thus the number of 32-bit words for the range bins results with Num_RWords = MaxRangeBins/32, where MaxRangeBins corresponds to the number of possible range bins depending on the current "Radar Cube".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin corresponding to the velocity 0 is in the center.

The peak map is used to check the settings of the signal processing, because it will be used in further steps.

If a line in Table 63 is highlighted in green, this value is optional and depends on the set bits in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)

Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = 0; r < Num_RWords; r++)		
Map[d][r]	uint32	Part of the peak map with 32 bits for 32 range bins (big endian)
CRC	uint16	CRC Checksum (optional)

Table 63: Data package Peak-Map

5.5.7 Processing = 5 (CFAR-Map)

The data of this process step can only be transmitted for radar cubes with multiple chirps.

The CFAR map is sent, which is the result of applying a CFAR algorithm to the peak map from section 5.5.6. CFAR is configured using the Radar Parameter "CfarThresh", "CfarGuardInt", and "CfarWindowSize". The resulting CFAR map is sent as a bitmap. Each bin of the range Doppler map corresponds to a bit in the CFAR map. The bit is "1" if a peak has been detected, or "0" if not.

The number of sent data is configured by the radar parameters "MinDopplerBin" and "MaxDopplerBin". "MinRangeBin" and "MaxRangeBin" have no influence on the number of bytes in this case, since 32 range bins are combined into one word. Thus the number of 32-bit words for the range bins results with Num_RWords = MaxRangeBins/32, where MaxRangeBins corresponds to the number of possible range bins depending on the current "Radar Cube".

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin corresponding to the velocity 0 is in the center.

The CFAR map is used to check the settings of the signal processing, since this is used in further steps.

If a line in Table 64 is highlighted in green, this value is optional and depends on the set bits in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
for (r = 0; r < Num_RWords; r++)		
Map[d][r]	uint32	Part of the CFAR map with 32 bits for 32 range bins (big endian)
CRC	uint16	CRC Checksum (optional)

Table 64: Data package CFAR-Map

5.5.8 Processing = 6 (Detection)

The data of this process step can only be transmitted for radar cubes with multiple chirps.

The results of the target detection are sent in the form of a list. To determine the detections, the object angles in azimuth direction are determined for all positions remaining after CFAR. In the case of MIMO radar cubes, the elevation angle is also estimated. After that, candidates with the

same distance and almost the same angle are combined. The maximum number of transmitted detections is given by the Radar Parameter "NumTargets". In case the Radar Parameter “SpeedEstimation” is set to a value greater zero, the values “SysDopplerBin” and “SysSpeed” are sent before “NumDetections”.

If a line in Table 65 is highlighted in green, this value is optional and depends on the set bits in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
if (“SpeedEstimation” > 0)		
SysDopplerBin	int16	Estimated Doppler bin representing radar system speed
SysSpeed	int16	Interpolated radar system speed [m/s * 100]
NumDetections	uint16	Number of detections found
for (n = 0; n < NumDetections; n++)		
Range Bin	uint16	Range Bin in which the detection was found
Doppler Bin	int16	Doppler Bin in which the detection was found
Magnitude	uint16	Magnitude value of the detection (20*log10(Amplitude)
Azimuth	int16	Azimuth angle [°]
Elevation	int16	Elevation angle [°]
CRC	uint16	CRC Checksum (optional)

Table 65: Data package Detection

5.5.9 Processing = 7 (Tracking)

The data of this process step can only be transmitted for radar cubes with multiple chirps.

The result of the tracking algorithm is sent in the form of a list. In case the Radar Parameter “SpeedEstimation” is set to a value greater zero, the values “SysDopplerBin” and “SysSpeed” are sent before “NumTracks”. Additionally, if the value of "StreamVariable" is set greater than zero, Doppler information is transmitted to each target.

The tracking algorithm can be adjusted with the Radar Parameter "MaxTracks", "MaxHorSpeed", "MaxVerSpeed", "MaxAccel", "MaxRangeError", "MinConfirm", "AssignLimit" and "MergeLimit". The result is a target list, in which each target gets a fixed number, which it keeps over time, if it is found again. Targets, which are not found again in the meantime, can be predicted for a certain time.

Furthermore, the magnitudes of the Doppler spectra of the targets found can be sent to the second processor (MCU/DSP) for classification. For this the second processor must be available and the Radar Parameter "DspDopplerProc" must be set to 1. If this is the case, the target list receives additional entries for the classification result. The correct result becomes available only after a certain number of measurements, since Doppler sequences are evaluated.

If the value of "StreamVariable" is set to 1 when starting the stream, the complex values of the Doppler spectra of all active channels are appended for each target. If the value is set to 2, the magnitudes of the Doppler spectrum of each target will be appended. The number of Doppler

values can be limited by "RxChannels", "MinDopplerBin" and "MaxDopplerBin" of the Radar Parameters.

Since the range of a target is calculated in meters after tracking, it is calculated afterwards to which range bin the target fits best. The Doppler spectrum is then selected with the range bin determined in this way.

Note that the Radar Parameter "DopplerFftShift" shifts the left and right side of the Doppler spectrum. If "DopplerFftShift" = 1, the Doppler bin, which corresponds to the velocity 0, is in the center.

In Table 66 the parameter Max_Chan stands for the maximum possible number of channels. This is 12 for MIMO Radar Cubes, otherwise 4.

If a line in Table 66 is highlighted in green, this value is optional and depends on the bits set in "Stream Mask".

Name	Data type	Description
Sync Word	uint32	Synchronization word (0xAA55CC33) (optional)
Meas Count	uint32	Measurement counter (optional)
Timestamp	uint64	Current system time in [ms] in Unix format
Status	uint16	Status word with bits according to Table 57
if ("SpeedEstimation" > 0)		
SysDopplerBin	int16	Estimated Doppler bin representing radar system speed
SysSpeed	int16	Interpolated radar system speed [m/s * 100]
NumTracks	uint16	Number of following tracks
for (n = 0; n < NumTracks; n++)		
ID	uint16	ID of the destination (0 to MaxTracks-1)
Range	float32	Distance of the target in [m]
Speed	float32	Speed of the target in [m/s]
Magnitude	uint16	Magnitude value of the target (20*log10(Amplitude)
Azimut	float32	Azimuth angle [°]
Elevation	float32	Elevation angle [°]
LifeTime	uint32	Value corresponding to the confirmations (retrievals) of the target
if („DspDopplerProc" == 1)		
InferenceResult	8 x uint16	Result vector of the neural network inference. 8 probabilities [% * 100]
if (StreamVariable == 1)		
for (channel = 0; channel < *MaxChan*; channel++)		
if (channel enabled)		
for (d = *MinDopplerBin*; d <= *MaxDopplerBin*; d++)		
Re	int16	Real part of the Doppler FFT result
Im	int16	Imaginary part of the Doppler FFT result
else if (StreamVarible > 1)		
for (d = MinDopplerBin; d <= MaxDopplerBin; d++)		
Mag	uint16	Magnitude of the Doppler spectrum
endif (StreamVariable)		
CRC	uint16	CRC Checksum (optional)

Table 66: Data package: Tracking

5.6 Stream Triggering

A special case is the observation of a scene with multiple radar modules at the same time. If streaming is used, the modules will perform measurements in given intervals on their own. Here the problem of interference may occur in case one module starts a measurement when another one is already performing.

To overcome this problem, the start of a stream can be triggered and delayed. The idea is to configure multiple radar modules with similar settings but different delays. After configuration each module waits for a start signal (command) which is sent ideally with a multi- or broadcast message, to address all of the respective modules at once. After receiving of the start signal, each module waits for the end its personal delay before performing a measurement. Two different modes have been implemented which are explained below.

To configure such a stream, the command "Configure Stream" (4.26) is used. The connection which configures the stream will be the target for the sent data. The parameter "Stream_Meas_Mode" determines the mode, how the measurement for the stream is triggered. It can have the following values:

- 0: Continuous mode: This is the default mode. Measurements are continuously performed in the interval defined by the Radar Parameter "MeasInterval".
- 1: Triggered Start: The module waits for a trigger command and then continuously performs measurements in the interval defined by the Radar Parameter "MeasInterval". See section 5.6.1 for more information.
- 2: Triggered Measurement: The module always waits for a trigger command to perform measurements but sends data on the stream interface. Radar Parameter "ContinuousMeas" must be 0. See section 5.6.2 for more information.

Which data is sent by the configured stream is determined by the parameter "Stream_Data_Mode" and can be set to "Single Data Mode" (see section 5.5) or "Multi Data Mode" (see section 5.4).

Four different "Start_Delay" values can be configured, which are addressed via index by the "Trigger Stream" command (4.27). These delays can be used to realize different intervals after each trigger. Another use case is to use different delays for different multicast groups.

5.6.1 Triggered Start

After configuring this mode, the radar module activates the stream for the current connection and waits for a trigger command before it performs the first measurement. If the trigger is received, the radar module waits the time defined by "Start_Delay" in milliseconds and then starts to perform measurements on its own in the interval defined by the Radar Parameter "MeasInterval" like in the default continuous mode. Therefore the Radar Parameter "ContinuousMeas" must be enabled. If multiple modules are configured with different delays, the measurements times of the single modules are cascaded as shown in the example in Figure 2.

A problem here might be that the internal clock is not perfect and so the measurement times of the modules might converge over time.

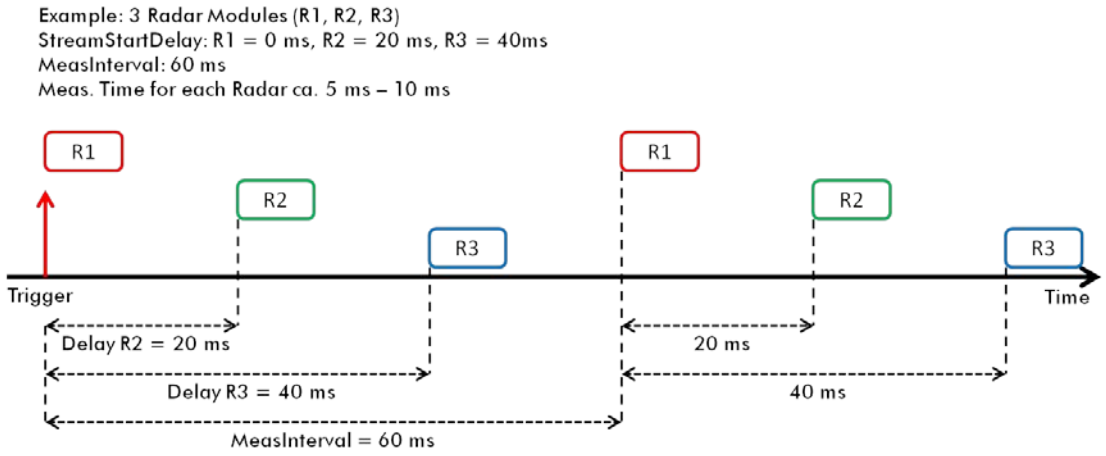

Figure 2: Example of a stream with triggered start

5.6.2 Triggered Measurement

When this mode is configured, the radar module also activates a stream for the current connection and waits for a trigger command before it performs a measurement. If the trigger is received, the radar module waits the time defined by "Start_Delay" in milliseconds and then performs one measurement. The difference to the "Triggered Start" mode is that each measurement must be triggered here. Therefore the Radar Parameter "ContinuousMeas" must be disabled. Figure 3 shows an example of this mode.

Because each measurement is triggered, there is no drift in the measurements interval of each module due to the inaccuracy of the clocks. Also with help of multicast different groups of modules could be addressed to measure at a certain time. But the trigger itself may also be a disadvantage in high latency networks.

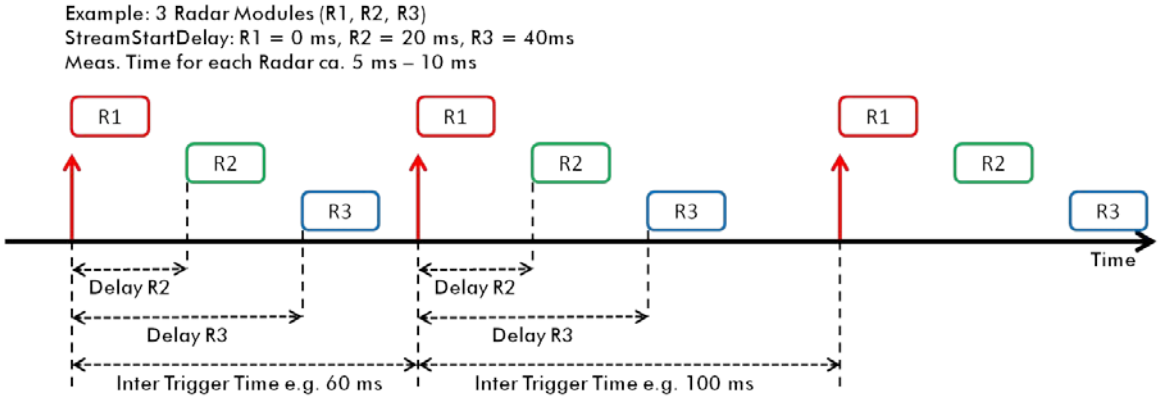

Figure 3: Example of a stream with triggered measurements

6 List of Tables