**ARS 548** **ROS2 Driver**

Valid for Radar-SW 05.48.04/05.48.05/05.48.06/05.48.07

Driver version V0.2.3

06.2024

Contents

[1. Overview 1](#_Toc163405956)

[2. Installation 1](#_Toc163405957)

[2.1 Directory Structure 1](#_Toc163405958)

[2.2 Install ars548 package 2](#_Toc163405959)

[2.3 Install USB Camera Module (optional) 2](#_Toc163405960)

[3. Network configuration 3](#_Toc163405961)

[4. Environment Setup 3](#_Toc163405962)

[4.1 Step 1: Initialize ROS 2 Iron Environment 4](#_Toc163405963)

[4.2 Step 2: Initialize ARS548 Package Environment 4](#_Toc163405964)

[5. Launch 4](#_Toc163405965)

[5.1 Real-time Reception 4](#_Toc163405966)

[5.2 Recording Playback 4](#_Toc163405967)

[6. Nodes 5](#_Toc163405968)

[6.1 Radar Communication Node 5](#_Toc163405969)

[6.2 Additional Nodes 6](#_Toc163405970)

[7. Development Instructions 6](#_Toc163405971)

[7.1 Demo Programs 6](#_Toc163405972)

[8. Appendix 9](#_Toc163405973)

[8.1 Topics and Messages 9](#_Toc163405974)

[8.2 Message Structures 10](#_Toc163405975)

# Overview

The ARS548 ROS2 driver is a software component developed specifically for the ARS548 radar. It enables the communication between the sensor and the ROS2 framework, allowing users to easily integrate the sensor into their ROS2-based robotic systems.

The driver is built on top of the [ROS2 Iron](https://docs.ros.org/en/iron/index.html) distribution, which provides a robust and reliable foundation for developing robotic applications. It utilizes the capabilities of ROS2, such as the pub-sub messaging system, to receive data from the ARS548 radar and publish it for further processing and analysis.

With the ARS548 ROS2 driver, users can access various sensor data such as distance measurements, velocity, and angular position. This information can be utilized by other ROS2 components to perform tasks such as obstacle avoidance, localization, mapping, and more.

# Installation

Unzip the ars548-iron-v0.2.3.zip file to the desired installation directory, assuming it is ~/workcopy/ars548-iron-v0.2.3:

bash

mkdir ~/workcopy

cd ~/workcopy

unzip ars548-iron-v0.2.3.zip

## Directory Structure

| **Path** | **Description** |
| --- | --- |
| ars548-iron-v0.2.3 | setup files |
| ars548-iron-v0.2.3/ars548 | root directory of the ars548 package |
| ars548-iron-v0.2.3/ars548/include | header files |
| ars548-iron-v0.2.3/ars548/lib/ars548 | node executable files |
| ars548-iron-v0.2.3/ars548/share/ars548/cmake | CMake-related files |
| ars548-iron-v0.2.3/ars548/share/ars548/msg | message definition files |
| ars548-iron-v0.2.3/ars548/share/ars548/srv | service definition files |
| ars548-iron-v0.2.3/ars548\_ros2\_demo | C++ console demo |
| ars548-iron-v0.2.3/ars548\_ros2\_qt\_demo | C++ Qt demo |
| ars548-iron-v0.2.3/ars548\_ros2\_py\_demo | Python Qt demo |

## Install ars548 package

Go to the installation directory and run the installation script:

bash

cd ars548-iron-v0.2.3

./setup

The installation script will automatically modify several initialization scripts related to environment variables in the current directory.

If the execution is successful, it will prompt:

bash

Setup path: /home/dev/workcopy/ars548-iron-v0.2.3

OK

If the authorization file ars548.lic is distributed with the product, make sure it is in the same directory as libars548.so, which should be ars548/lib/.

## Install USB Camera Module (optional)

The USB Camera Module is not part of the ars548 package. However, during testing, a USB camera was used to display the images of the areas detected by the radar. Therefore, it is included in the default\_launch.py file to start together, and an image component is added in rviz for visualization. you can decide whether to install this module based on your own needs.

bash

sudo apt install ros-iron-camera-calibration-parsers

sudo apt install ros-iron-camera-info-manager

sudo apt install ros-iron-launch-testing-ament-cmake

sudo apt install ros-iron-usb-cam

After connecting the USB camera, you can start the usb\_cam node and use rviz2 to view it by adding the Image type display component and selecting /image\_raw as the topic.

# Network configuration

To successfully communicate with the radar, you need to configure your computer's network settings correctly:

* Identify a wired network card that communicates with the radar.
* Set a vlan for the network card with an id of 19 (this step is the key to successfully sending data to the radar, such as sending configuration to the radar).
* Set a static IP address for the network card, which needs to be in the same subnet as the radar's IP address (10.13.1.0/24).
* The radar's default IP address is usually 10.13.1.113, so don't conflict with that.

This is an example of configuring a network card using netplan in Ubuntu 22.04:

network:

version: 2

renderer: networkd

ethernets:

  enp0sX:

    dhcp4: no

vlans:

  vlan19:

    id: 19

    link: enp0sX

    addresses:

      - 10.13.1.200/24

# Environment Setup

Before launching any files or running nodes in a new terminal, ensure you source both the ROS2 Iron and ARS548 package environments as outlined below.

## Step 1: Initialize ROS 2 Iron Environment

bash

source /opt/ros/iron/setup.bash

This command prepares your terminal to use the ROS 2 Iron distribution by setting up the appropriate environment variables.

## Step 2: Initialize ARS548 Package Environment

bash

source ~/workcopy/ars548-iron-v0.2.3/setup.sh

This ensures that your current terminal session is configured with the correct environment variables for the ARS548 package.

# Launch

In ROS2, a launch file is used to start up and manage multiple nodes and configurations at once, simplifying the process of running complex systems. Along these lines, the ars548 package also provides some launch files, which are designed for two scenarios: radar data real-time reception and recording playback.

## Real-time Reception

Specify the IP address of the network card connected to the radar and the IP address of the radar:

bash

export ARS548\_BIND\_IP=10.13.1.200

export ARS548\_RADAR\_IP=10.13.1.113

These two IP addresses are used by default, so there is no need to set them if they match your situation.

Start all nodes at once:

bash

ros2 launch ars548 default\_launch.py

## Recording Playback

Playback of recordings in .rec format generated by our RT-Viewer software is supported.

Specify the recording file or directory to be played back:

bash

export ARS548\_REPLAY\_FILE=/home/dev/workcopy/records/2023-10-28\_17.13.00

launch replay:

bash

ros2 launch ars548 replay\_launch.py

# Nodes

The ARS548 ROS2 driver consists of several nodes, each designated for specific tasks within the ROS2 framework.

## Radar Communication Node

The radar\_node establishes a two-way communication channel with the ARS548 radar, facilitating not only the retrieval of raw data but also enabling the transmission of configuration commands back to the radar.

Since running the radar\_node requires administrator privileges, to execute it as a regular user, please perform the following two commands:

bash

export FASTRTPS\_DEFAULT\_PROFILES\_FILE=<install dir>/ars548/share/ars548/fastdds\_root\_workaround.xml

sudo -E env "PYTHONPATH=$PYTHONPATH" "LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH" "PATH=$PATH" "USER=$USER" bash -c "<install dir>/ars548/lib/ars548/radar\_node <bind\_ip> <radar\_ip>"

Make sure to replace <install dir> with the actual installation directory of the ars548 driver, <bind\_ip> with the IP address of the network card connected to the radar, and <radar\_ip> with the IP address of the radar.

Taking the installation directory /home/dev/workcopy/ars548-iron-v0.2.3 as an example, it would be:

bash

export FASTRTPS\_DEFAULT\_PROFILES\_FILE=/home/dev/workcopy/ars548-iron-v0.2.3/ars548/share/ars548/fastdds\_root\_workaround.xml

sudo -E env "PYTHONPATH=$PYTHONPATH" "LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH" "PATH=$PATH" "USER=$USER"  bash -c "/home/dev/workcopy/ars548-iron-v0.2.3/ars548/lib/ars548/radar\_node 10.13.1.200 10.13.1.113"

## Additional Nodes

Beyond the radar\_node, the ARS548 package further incorporates several nodes for additional functionalities, as listed below.

| **Node Name** | **Command** | **Description** |
| --- | --- | --- |
| axis | ros2 run ars548 axis | Coordinate axis drawing node |
| point\_cloud | ros2 run ars548 point\_cloud | Point cloud data publishing node |
| rec\_replay | ros2 run ars548 rec\_replay <.rec-file-or-directory> | Recording playback node |

Once again, ensure that both your ROS2 setup file and ars548 setup file have been sourced to update the environment variables. This step is crucial before attempting to run any nodes in a new terminal.

# Development Instructions

The radar\_node provided by the ars548 package communicates with the radar and republishes the data as ROS2 messages. As a developer, you can perform further processing and application of the radar data by subscribing to the topics that interest you.

The following demo program demonstrates how to subscribe to topics and call the associated services.

## Demo Programs

Before running demo programs, please make sure radar\_node is already running and communicating with the radar properly. Usually, the radar\_node is run in a separate console window. Therefore, when running the demo programs, you need to open a new console window and source the setup files of both ROS2 and ars548.

bash

source /opt/ros/iron/setup.bash

source ars548-iron-v0.2.3/setup.sh

### ars548\_ros2\_demo

This is a simplest C++ demo that demonstrates how to subscribe to the messages published by the radar\_node and output the number of targets detected by the radar.

Build:

bash

cd ars548-iron-v0.2.3/ars548\_ros2\_demo

colcon build

Execution:

bash

source install/local\_setup.sh

ros2 run ars548\_ros2\_demo ars548\_ros2\_demo

### ars548\_ros2\_qt\_demo

This is a C++ demo that demonstrates how to subscribe to messages to receive radar data and modify the configuration of the radar by calling the services provided by the radar\_node, using a Qt window interface.

Build:

bash

cd ars548-iron-v0.2.3/ars548\_ros2\_qt\_demo

colcon build

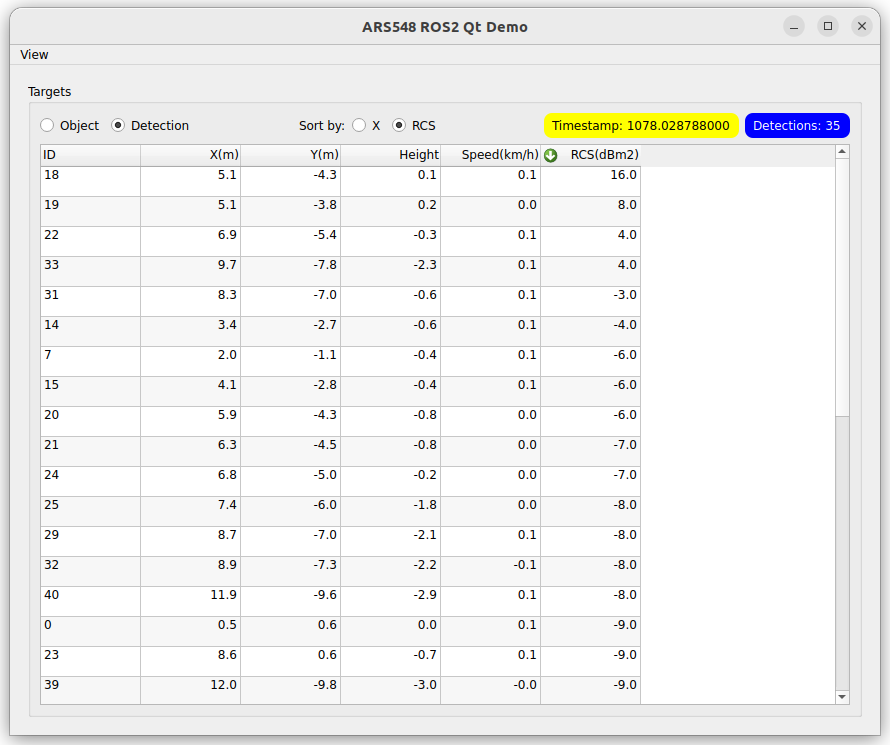
Execution:

bash

source install/local\_setup.sh

ros2 run ars548\_ros2\_qt\_demo ars548\_ros2\_qt\_demo

After running the program, you will see the figure below.



The table on the main window displays the targets data received from the radar. You can toggle between target displays using the Object/Detection radio buttons and sort your desired data by clicking the corresponding column header.

Clicking View in the upper left corner will expand the menu, and clicking on the corresponding menu item will open the corresponding window, providing options to view and set sensor and filter data for the radar.

### ars548\_ros2\_py\_demo

This is a Python demo that has the same functionality as the ars548\_ros2\_qt\_demo, but it is implemented using the Python language.

Build:

bash

cd ars548-iron-v0.2.3/ars548\_ros2\_py\_demo

colcon build

Execution:

bash

source install/local\_setup.sh

python3 ars548\_ros2\_py\_demo/main.py

# Appendix

## Topics and Messages

The topics published by the ars548 package are listed in the following table. You can subscribe to the topics that interest you for further processing or display.

| **Node Name** | **Published Topic** | **Message Type** | **Description** |
| --- | --- | --- | --- |
| radar\_node | /ars548/detection\_list | Ars548DetectionList | Raw radar detection data |
|  | /ars548/object\_list | Ars548ObjectList | Recognized radar object data |
|  | /ars548/sensor\_status | Ars548SensorStatus | Radar sensor status data |
|  | /ars548/filter\_status | Ars548FilterStatus | Radar filter status data |
|  | /ars548/target\_list/detection | Ars548TargetList | The target's horizontal/vertical distance and velocity relative to the radar are provided to facilitate further processing by the user. |
|  | /ars548/target\_list/object | Ars548TargetList | The target's horizontal/vertical distance and velocity relative to the radar are provided to facilitate further processing by the user. |
| point\_cloud | /ars548/viz/detection | sensor\_msgs/msg/PointCloud2 | Point cloud data for rviz2 display |
|  | /ars548/viz/object | sensor\_msgs/msg/PointCloud2 | Point cloud data for rviz2 display |

Please refer to the message type definitions for the specific fields and descriptions.

## Message Structures

### Ars548Detection

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| float32 | f\_azimuth\_angle | Detection Azimuth Angle |
| float32 | f\_azimuth\_angle\_std | Azimuth Angle Std |
| uint8 | u\_invalid\_flags | Detection Invalid Flags |
| float32 | f\_elevation\_angle | Detection Elevation Angle |
| float32 | f\_elevation\_angle\_std | Elevation Angle Std |
| float32 | f\_range | Detection Radial Distance |
| float32 | f\_range\_std | Radial Distance Std |
| float32 | f\_range\_rate | Detection Radial Velocity |
| float32 | f\_range\_rate\_std | Radial Velocity Std |
| int8 | s\_rcs | Detecion RCS |
| uint16 | u\_measurement\_id | Detection ID |
| uint8 | u\_positive\_predictive\_value | Existence Probability |
| uint8 | u\_classification | Detection Classification |
| uint8 | u\_multi\_target\_probability | Multi-Target Probability |
| uint16 | u\_object\_id | Associated Object |
| uint8 | u\_ambiguity\_flag | tbd |
| uint16 | u\_sort\_index | tbd |

### Ars548DetectionList

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint64 | crc | checksum (E2E Profile 7) |
| uint32 | length | len (E2E Profile 7) |
| uint32 | sqc | sqc (E2E Profile 7) |
| uint32 | data\_id | data id (E2E Profile 7) |
| uint32 | timestamp\_nanoseconds | timestamp nanoseconds |
| uint32 | timestamp\_seconds | timestamp seconds |
| uint8 | timestamp\_sync\_status | timestamp sync status |
| uint32 | event\_data\_qualifier | event data qualifier |
| uint8 | extended\_qualifier | extended qualifier |
| uint16 | origin\_invalid\_flags | sensor position invalid flags |
| float32 | origin\_xpos | sensor x position |
| float32 | origin\_xstd | sensor x position std |
| float32 | origin\_ypos | sensor y position |
| float32 | origin\_ystd | sensor y position std |
| float32 | origin\_zpos | sensor z position |
| float32 | origin\_zstd | sensor z position std |
| float32 | origin\_roll | sensor roll angle |
| float32 | origin\_rollstd | sensor roll angle std |
| float32 | origin\_pitch | sensor pitch angle |
| float32 | origin\_pitchstd | sensor pitch angle std |
| float32 | origin\_yaw | sensor yaw angle |
| float32 | origin\_yawstd | sensor yaw angle std |
| uint8 | list\_invalid\_flags | invalid flags |
| Ars548Detection[] | detections | array of ars548detection messages |
| float32 | list\_radvel\_domain\_min | radial velocity domain min |
| float32 | list\_radvel\_domain\_max | radial velocity domain max |
| uint32 | list\_num\_of\_detections | number of detections |
| float32 | aln\_azimuth\_correction | azimuth correction |
| float32 | aln\_elevation\_correction | elevation correction |
| uint8 | aln\_status | status of alignment |

### Ars548FilterStatus

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint32 | timestamp\_nanoseconds | Timestamp Nanoseconds |
| uint32 | timestamp\_seconds | Timestamp Seconds |
| uint8 | timestamp\_sync\_status | Timestamp Sync Status |
| uint8 | filter\_configuration\_counter | Counter that counts up if new filter configuration has been received and accepted |
| uint8 | detection\_sort\_index | Detection list sorting index |
| uint8 | object\_sort\_index | Object list sorting index |
| Ars548FilterStatusEntry[] | detection\_filter | Detection filter entries (index 1 to 7) |
| Ars548FilterStatusEntry[] | object\_filter | Object filter entries (index 1 to 24) |

### Ars548FilterStatusEntry

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint8 | active | Flag to activate/deactivate filter |
| uint8 | filter\_id | Filter data index |
| float32 | minimum\_value | Minimum data value to pass the filter |
| float32 | maximum\_value | Maximum data value to pass the filter |

### Ars548Object

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint16 | u\_status\_sensor | tbd |
| uint32 | u\_id | ID of object |
| uint16 | u\_age | Age of object |
| uint8 | u\_status\_measurement | Object Status |
| uint8 | u\_status\_movement | Object Movement Status |
| uint16 | u\_position\_invalidflags | tbd |
| uint8 | u\_position\_reference | Reference point position |
| float32 | f\_position\_x | X Position |
| float32 | f\_position\_x\_std | X Position Std |
| float32 | f\_position\_y | Y Position |
| float32 | f\_position\_y\_std | Y Position Std |
| float32 | f\_position\_z | Z Position |
| float32 | f\_position\_z\_std | Z Position Std |
| float32 | f\_position\_covariancexy | Covariance X Y |
| float32 | f\_position\_orientation | Object Orientation |
| float32 | f\_position\_orientation\_std | Orientation Std |
| uint8 | u\_existence\_invalidflags | tbd |
| float32 | f\_existence\_probability | Probability of Existence |
| float32 | f\_rcs | RCS |
| uint8 | u\_classification\_car | Car Classification |
| uint8 | u\_classification\_truck | Truck Classification |
| uint8 | u\_classification\_motorcycle | Motorcycle Classification |
| uint8 | u\_classification\_bicycle | Bicycle Classification |
| uint8 | u\_classification\_pedestrian | Pedestrian Classification |
| uint8 | u\_classification\_animal | Animal Classification |
| uint8 | u\_classification\_hazard | Hazard Classification |
| uint8 | u\_classification\_unknown | Unknown Classification |
| uint8 | u\_classification\_overdrivable | Overdrivable Classification |
| uint8 | u\_classification\_underdrivable | Underdrivable Classification |
| uint8 | u\_dynamics\_absvel\_invalidflags | Invalid Flags AbsVel |
| float32 | f\_dynamics\_absvel\_x | X Abs Vel |
| float32 | f\_dynamics\_absvel\_x\_std | X Abs Vel Std |
| float32 | f\_dynamics\_absvel\_y | Y Abs Vel |
| float32 | f\_dynamics\_absvel\_y\_std | Y Abs Vel Std |
| float32 | f\_dynamics\_absvel\_covariancexy | Covariance Abs Vel X Y |
| uint8 | u\_dynamics\_relvel\_invalidflags | Invalid Flags RelVel |
| float32 | f\_dynamics\_relvel\_x | X Rel Vel |
| float32 | f\_dynamics\_relvel\_x\_std | X Rel Vel Std |
| float32 | f\_dynamics\_relvel\_y | Y Rel Vel |
| float32 | f\_dynamics\_relvel\_y\_std | Y Rel Vel Std |
| float32 | f\_dynamics\_relvel\_covariancexy | Covariance Rel Vel X Y |
| uint8 | u\_dynamics\_absaccel\_invalidflags | Invalid Flags AbsAccel |
| float32 | f\_dynamics\_absaccel\_x | X Abs Accel |
| float32 | f\_dynamics\_absaccel\_x\_std | X Abs Accel Std |
| float32 | f\_dynamics\_absaccel\_y | Y Abs Accel |
| float32 | f\_dynamics\_absaccel\_y\_std | Y Abs Accel Std |
| float32 | f\_dynamics\_absaccel\_covariancexy | Covariance Abs Accel X Y |
| uint8 | u\_dynamics\_relaccel\_invalidflags | Invalid Flags RelAccel |
| float32 | f\_dynamics\_relaccel\_x | X Rel Accel |
| float32 | f\_dynamics\_relaccel\_x\_std | X Rel Accel Std |
| float32 | f\_dynamics\_relaccel\_y | Y Rel Accel |
| float32 | f\_dynamics\_relaccel\_y\_std | Y Rel Accel Std |
| float32 | f\_dynamics\_relaccel\_covariancexy | Covariance Rel Accel X Y |
| uint8 | u\_dynamics\_orientation\_invalidflags | Invalid Flags Orientation |
| float32 | u\_dynamics\_orientation\_rate\_mean | Object Orientation Rate |
| float32 | u\_dynamics\_orientation\_rate\_std | Orientation Rate Std |
| uint32 | u\_shape\_length\_status | Shape Length Status |
| uint8 | u\_shape\_length\_edge\_invalidflags | Invalid Flags Shape Length |
| float32 | u\_shape\_length\_edge\_mean | Mean Shape Length |
| float32 | u\_shape\_length\_edge\_std | Shape Length Std |
| uint32 | u\_shape\_width\_status | Shape Width Status |
| uint8 | u\_shape\_width\_edge\_invalidflags | Invalid Flags Shape Width |
| float32 | u\_shape\_width\_edge\_mean | Mean Shape Width |
| float32 | u\_shape\_width\_edge\_std | Shape Width Std |

### Ars548ObjectList

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint64 | crc | Checksum (E2E Profile 7) |
| uint32 | length | Len (E2E Profile 7) |
| uint32 | sqc | SQC (E2E Profile 7) |
| uint32 | dataid | Data ID (E2E Profile 7) |
| uint32 | timestamp\_nanoseconds | Timestamp Nanoseconds |
| uint32 | timestamp\_seconds | Timestamp Seconds |
| uint8 | timestamp\_syncstatus | Timestamp Sync Status |
| uint32 | eventdataqualifier | Event Data Qualifier |
| uint8 | extendedqualifier | Extended Qualifier |
| uint8 | objectlist\_numofobjects | Number of Objects |
| Ars548Object[] | objects | Array of Objects |

### Ars548SensorStatus

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint32 | timestamp\_nanoseconds | Timestamp Nanoseconds |
| uint32 | timestamp\_seconds | Timestamp Seconds |
| uint8 | timestamp\_sync\_status | Timestamp Sync Status |
| uint8 | swversion\_major | Software version (major) |
| uint8 | swversion\_minor | Software version (minor) |
| uint8 | swversion\_patch | Software version (patch) |
| float32 | longitudinal | Longitudinal sensor position (AUTOSAR) |
| float32 | lateral | Lateral sensor position (AUTOSAR) |
| float32 | vertical | Vertical sensor position (AUTOSAR) |
| float32 | yaw | Sensor yaw angle (AUTOSAR) |
| float32 | pitch | Sensor pitch angle (AUTOSAR) |
| uint8 | plug\_orientation | Orientation of plug |
| float32 | length | Vehicle length |
| float32 | width | Vehicle width |
| float32 | height | Vehicle height |
| float32 | wheelbase | Vehicle wheelbase |
| uint16 | maximum\_distance | Maximum detection distance |
| uint8 | frequency\_slot | Center frequency |
| uint8 | cycle\_time | Cycle time |
| uint8 | time\_slot | Cycle offset |
| uint8 | hcc | Country code |
| uint8 | powersave\_standstill | Power saving in standstill |
| uint32 | sensor\_ip\_address\_0 | Sensor IP address |
| uint32 | sensor\_ip\_address\_1 | Reserved |
| uint8 | configuration\_counter | Counter that counts up if new configuration has been received and accepted |
| uint8 | status\_longitudinalvelocity | Signals if current VDY is OK or timed out |
| uint8 | status\_longitudinalacceleration | Signals if current VDY is OK or timed out |
| uint8 | status\_lateralacceleration | Signals if current VDY is OK or timed out |
| uint8 | status\_yawrate | Signals if current VDY is OK or timed out |
| uint8 | status\_steeringangle | Signals if current VDY is OK or timed out |
| uint8 | status\_drivingdirection | Signals if current VDY is OK or timed out |
| uint8 | status\_characteristicspeed | Signals if current VDY is OK or timed out |
| uint8 | status\_radar | Signals if Radar Status is OK |
| uint8 | status\_voltagestatus | Bitfield to report under- and overvoltage errors |
| uint8 | status\_temperaturestatus | Bitfield to report under- and overtemperature errors |
| uint8 | status\_blockagestatus | Current blockage state and blockage self test state |

### Ars548Target

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| uint32 | id | unique id |
| float32 | distance | vertical distance (m) |
| float32 | position | horizontal distance (m) |
| float32 | height | height (m) |
| float32 | degrees | angle, reserved, currently 0 |
| float32 | kmph | speed (km/h) |
| float32 | rcs | radar cross section (dbm²) |

### Ars548TargetList

| **Field Type** | **Field Name** | **Description** |
| --- | --- | --- |
| Ars548Target[] | targets | Array of targets |