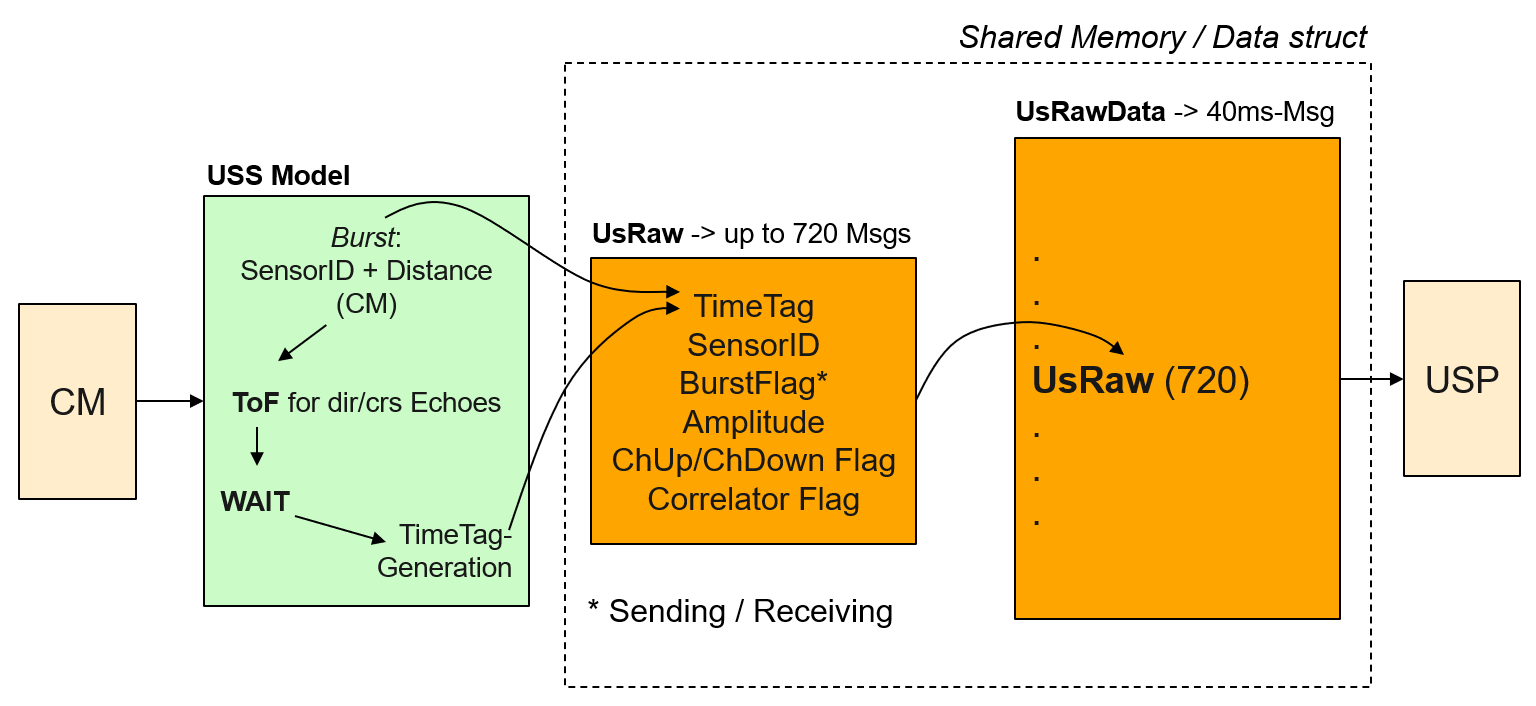
# Phen-003 Documentation: UDP Data Format

## General Approach

This feature shall provide the API/Interface between the USS model and the according USP function which is receiving the data. In general, the following picture is depicting the connection between CarMaker (CM) and Ultrasonic Signal Processing (USP) throughout the USS Model (vCUS-Phen-Mod) using shared memory (Struct: UsRawData.UsRaw).



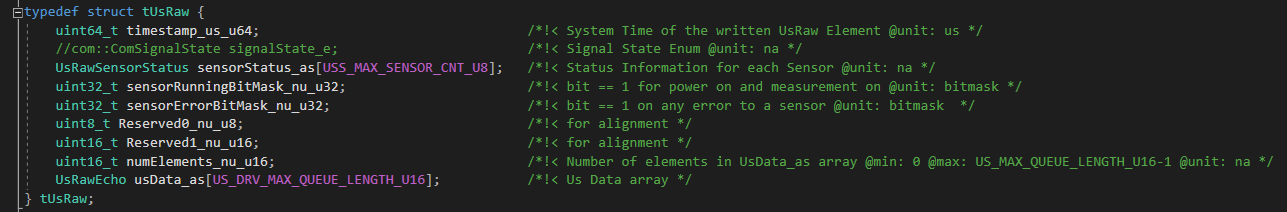
Step1: CM is generating positions and orientations for both the Ego Vehicle as well as the Traffic Objects (TOs). This data is the processed in the USS Model.

Step2: Based on the Stochastic fire Scheme Pattern (SFSP) the model is generating a UsRaw-Msg containing a Burst-Flag. Then the model is calculating (via Phen-001/Phen-002/Phen-004) reflection points (NearestPoints) delivering the distance to one TO’s reflection point. The TO’s reflection point is determined for direct echoes (Phen-004) as well as for cross echoes (Phen-005). The accordingly derived distances will be processed to the physical ToF for both direct and indirect echoes. This will be used as an input to an artificial WAIT-algorithm to determine the points of time when to send (again) UsRaw-Msg containing a Received-Flag.

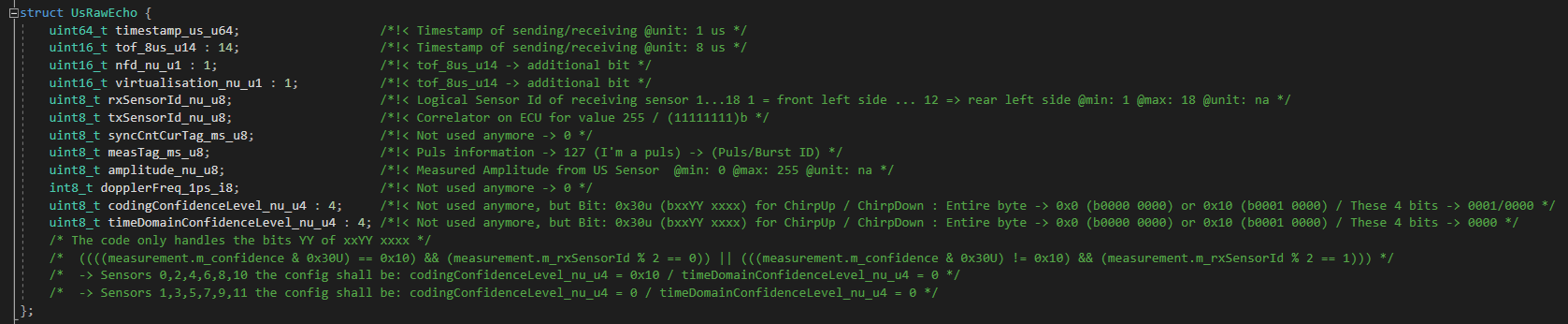
Step3: The generated UsRaw-Msgs will be consolidated over time in the struct UsRawData. After 40ms the UsRawData-Msg will be sent to USP. The maximum size of UsRawData for UsRaw-Msgs is 720. Additional Msgs will be deleted (but it’s most unlikely that this number will be exceeded).

## Structure of UsRaw / UsRawData

The following part represents the current struct USRaw



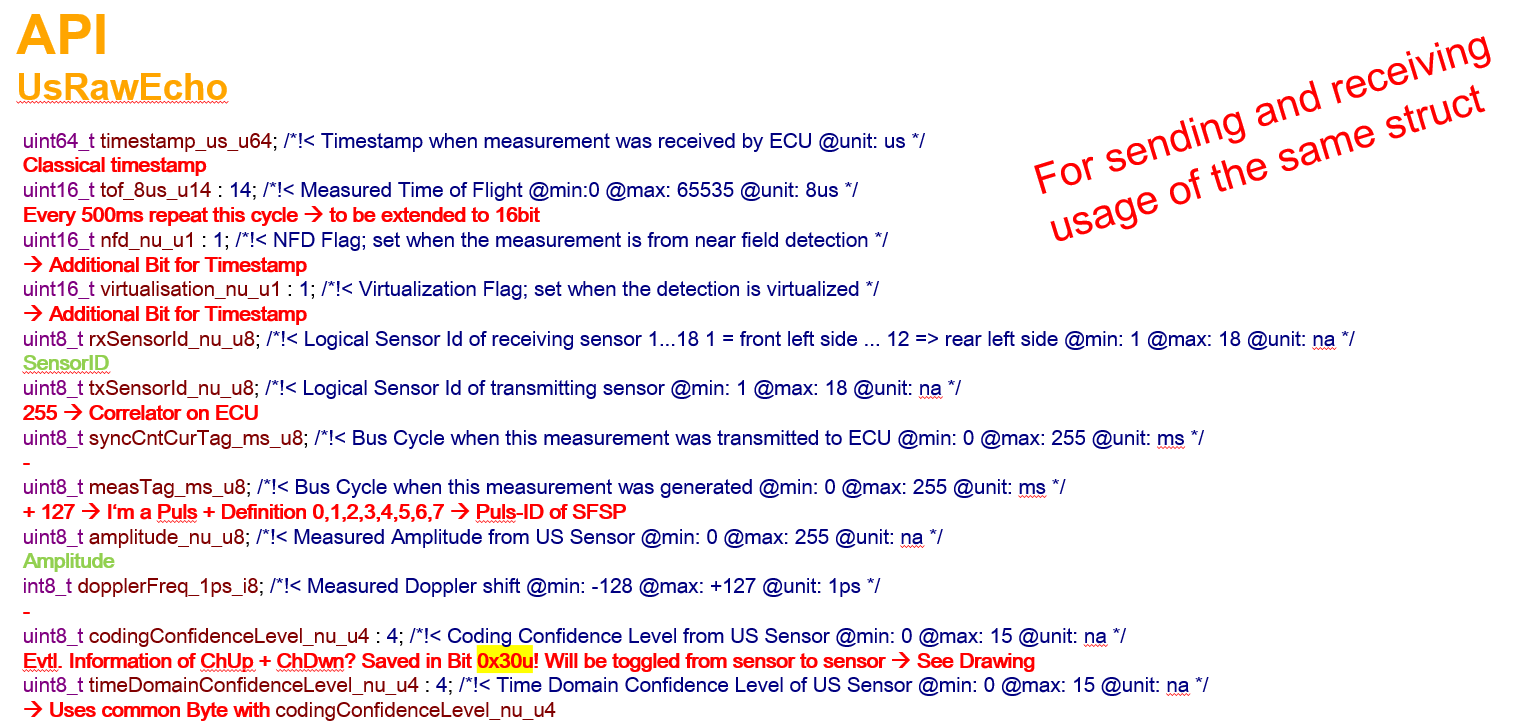
The measurements per echo are stored, like mentioned in the first section, in the struct usData\_as[720] with data type UsRawEcho. The description is



## Usage of UsRawEcho

In order to keep interfaces stable and to reduce the effort of reallocating bit-positions to the current sensor data sent via DSI3 protocol, the payload of the existing UsRawEcho struct will be reinterpreted. Accordingly, the meanings of the naming of the variables might differ for some of the variables in the struct.

The following list shows the new interpretation:



**tof\_8us\_u14 + nfd\_nu\_u1 + virtualisation\_nu\_u1**

The 64bit time stamp of timestamp\_us\_u64 will be casted to a 16bit timestamp and divided into the variables tof\_8us\_u14 (14bit) + nfd\_nu-u1 (1bit) + virtualization\_nu\_u1 (1bit).

The according resolution / counting steps in the variable will be in 8us steps.

**measTag\_ms\_u8**

The meas tag will be reinterpreted as burst flag indicating which stochastic code burst is used. This is realized by the positions of stochastic code in the parameter set. The content of the variable is then represented by

0..127 -> Received Msg as a counter between 0 and 126

and

127 -> Stochastic code burst number 1

128 -> Stochastic code burst number 2

...

134 -> Stochastic code burst number 8

135..255 -> Not defined

**codingConfidenceLevel\_nu\_u4 + timeDomainConfidenceLevel\_nu\_u4**

The code only handles the bits YY of its bit content xxYY xxxx. The code in USP is checking for:

((((measurement.m\_confidence & 0x30U) == 0x10) && (measurement.m\_rxSensorId % 2 == 0)) || (((measurement.m\_confidence & 0x30U) != 0x10) && (measurement.m\_rxSensorId % 2 == 1)))

Which means that for sensors 0,2,4,6,8,10 the config shall be: codingConfidenceLevel\_nu\_u4 = 0x10 / timeDomainConfidenceLevel\_nu\_u4 = 0 and for sensors 1,3,5,7,9,11 the config shall be: codingConfidenceLevel\_nu\_u4 = 0 / timeDomainConfidenceLevel\_nu\_u4 = 0