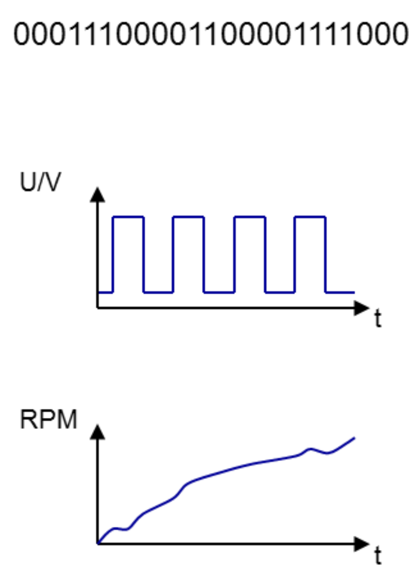
Pre-processing of revolution speed data in ArtemiS SUITE[1]

Introduction

Many of the analysis functions provided by ArtemiS SUITE are calculated and displayed against revolution speed (revolutions per minute, RPM) rather than time (e.g., ***FFT vs. RPM***, ***Order Spectrum vs. RPM***, etc.). In order to calculate these analyses, information about the current revolution speed is required. This information can be acquired and stored in three different ways:



1. Pulse channel
Revolution speed information is recorded as a pulse channel via a separate pulse input of the recording hardware and saved to a digital channel. Only two states are saved: *0* and *1*.

2. Trigger signal
As with a pulse channel, revolution speed information is recorded as a pulse signal. However, in this case, the signal is not saved as a logical bit sequence (*0*, *1*), but as a voltage curve in an analog channel.

3. Direct recording
The revolution speed curve is saved directly to an analog channel. For such a recording, the hardware must be equipped with a DC-coupled input.

Figure 1: Saving of RPM information

In case of direct recording (case 3), the measurement unit and the values of the channels are specified directly in the channel definition. Information about using such a revolution speed curve can be found in the Help System of ArtemiS SUITE and in the Application Note “Using different reference quantities in ArtemiS SUITE”.
This Application Note, on the other hand, explains the necessary pre-processing of revolution speed information recorded as a pulse signal (cases 1 and 2). These cases require the definition of a pulse channel or a trigger channel, respectively, which mainly defines the measurement unit and the relation between the pulse frequency and the revolution speed.
For the pulse channel (case 1), this relation is usually specified directly in the HEAD Recorder prior to the recording. However, it can also be defined or adapted afterwards in ArtemiS SUITE. As long as the pulse signal contains equidistant pulses and only one gap per revolution, which is not larger than two teeth, the conversion from pulses to RPM can take place automatically. This means that after the

[1] The descriptions in this Application Note refer to version 9.2. The general procedures also apply to other versions. However, the scope of functionality and the user interface may differ.

recording, you can add the file containing the pulse channel to the Source Pool of a Pool Project, for example, and directly perform an analysis against RPM, since the automatic conversion of pulse data to RPM information takes place within the analysis.
In order to use a trigger signal (case 2), you must first create a trigger channel in ArtemiS SUITE (see section “Processing trigger signals”). Afterwards, as long as the trigger signal contains equidistant pulses and only one gap not larger than two teeth, the current revolution speed is determined automatically during the analysis as in case 1.
The decoding of pulse signals based on more complex patterns (such as non-equidistant with several or longer gaps, or overlapping zebratape patterns) can be performed in a Decoder Project. Such a project allows you to decode pulse signals prior to further processing (e.g., an analysis) and to save the result as an analog channel (leading to case 3) in addition to the existing channels of the file. The exact procedure is described in the section “Revolution speed acquisition with complex pulse patterns, e.g., cogwheels with missing teeth”.
Furthermore, the following two chapters in this Application Note first explain the TTL logic and possible error sources in pulse data acquisition.
One more note to complete this introduction: for measurements with defective or missing revolution speed information, you can generate an artificial revolution speed curve from visible order curves using the RPM Generator of ArtemiS SUITE (ASM 24, Data Preparation, is required). See the Help System of ArtemiS SUITE for instructions on use of this tool.

TTL logic

HEAD acoustics offers frontends, such as the SQuadriga II, which provide separate pulse inputs in addition to the inputs for connecting sensors (e.g., microphones or acceleration sensors). These pulse inputs allow the connection of revolution speed sensors delivering a TTL-compatible signal. “TTL” stands for “transistor–transistor logic” and describes a signal form as shown in figure 2.

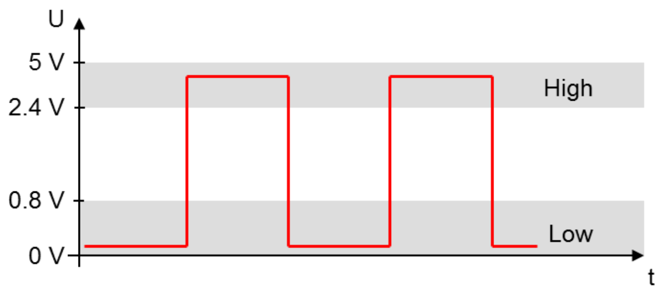


Figure 2: Example of a TTL signal

Such a TTL-compatible pulse signal recorded via the pulse input of a frontend is saved as a *pulse channel* (see case 1 above). Only two states can be recorded: *Low* (≙ *0*) and *High* (≙ *1*). A SQuadriga II interprets voltages between 0 and 0.8 V as *Low* and voltages between 2.5 and 5 V as *High*.
In ArtemiS SUITE, this pulse signal is then automatically[2] converted into revolution speed information. The parameters for the conversion are described in the “Pulse channels” section of the Help System.

[2] Provided that the pulse signal contains equidistant pulses with no more than one gap per revolution.

Sources of error in pulse data acquisition

If the revolution speed information is stored as a pulse signal, the current revolution speed can be calculated from it. For this purpose, the time period Δt between two rising[3] edges of the pulse signal is determined (see figure 3).

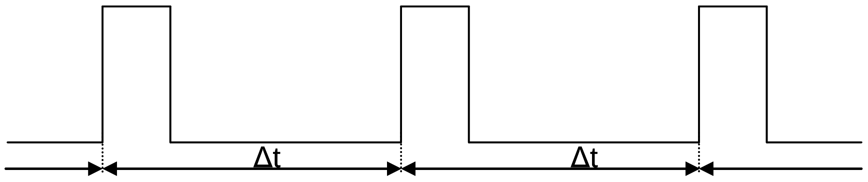


Figure 3: Analog pulse signal

The pulse frequency (i.e., number of pulses per second) is determined as the reciprocal value of the period Δt, and then multiplied by the pulse factor to determine the current revolution speed:

$$revolution\ speed = \frac{1}{\Delta t} \cdot pulse\ factor = pulse\ frequency \cdot pulse\ factor$$

The pulse factor is the quotient of two constants: mapping factor, and pulse rate. The pulse rate specifies how many pulses are generated during one revolution of the measurement object, whereas the mapping factor is used for adapting the measurement units. For revolution speed calculation, the mapping factor is: $60\,\frac{s}{min}$

That way, the result is the current revolution speed with the unit [revolutions/min]. The more precisely the period Δt can be determined, the more precisely the current revolution speed can be calculated. However, digital signal processing does not evaluate the analog pulse signal, but the discrete values of the signal sampled with a certain sampling rate f_s. Figure 4 shows an example of a pulse signal encoded in a digital pulse channel. The sampled values stored in a pulse channel are a 1-bit signal containing only the values *0* and *1*.

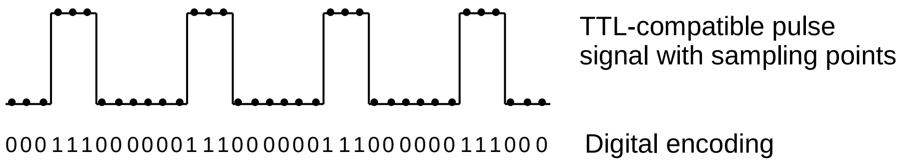


Figure 4: Sampling and encoding of revolution speed as a pulse channel

Figure 5 shows three examples of a pulse signal sampled with different sampling rates. The sampling positions are marked with dots. The sampling rate decreases from the first to the third example. To determine the period Δt, the sampling points between two consecutive *0* → *1* transitions are counted, i.e.:

$$\Delta t = n \cdot \frac{1}{f_s}$$

[3] It is also possible to use the falling edge for the calculation; see section “Pulse channels” in the Help System of ArtemiS SUITE.

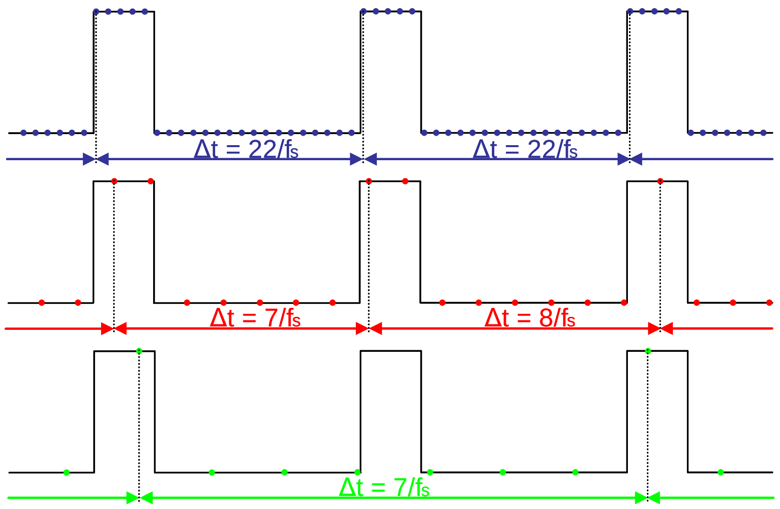


Figure 5: Pulse signal sampled with different sampling rates

It is obvious that the precision with which Δt is determined depends directly on the sampling rate. The lower the sampling rate compared to the pulse frequency, the less precise the determined value of Δt. If the sampling rate is not sufficiently high, the current revolution speed is not calculated correctly. In the center image of figure 5, Δt has the value $7/f_s$ in one case and $8/f_s$ in another, even though the distance between the rising edges of the analog pulse signal being sampled digitally is actually constant. The calculated current revolution speed therefore jumps between two or more values (this is called "jitter"). This error is caused by too low a sampling rate.
In the bottom example of figure 5, the sampling rate is so low that not even all pulses are detected and evaluated, so that the period Δt would be much too long and the resulting revolution speed calculation would deliver a completely incorrect value.
The sampling of the signal leads to a systematic error, because the time position of the rising edge in the pulse signal cannot be determined exactly.

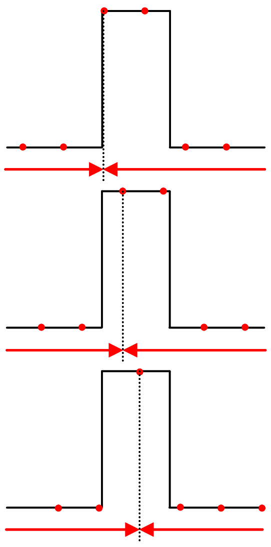


Figure 6 illustrates this systematic error. A pulse is sampled with the same sampling rate f_s in all three cases, but the time positions of the sampling points are shifted. In the top example, the sampling point representing the $0 \rightarrow 1$ transition for determining the period Δt is located almost exactly on the actual rising edge of the analog signals, whereas it is shifted to increasingly later times in the center and bottom examples. The maximum possible error in the localization of the rising edge depends on the sampling rate and can be calculated as $\frac{1}{f_s}$.
The lower the sampling rate compared to the pulse frequency, the higher the error, and the higher the sampling rate, the smaller the jitter effect.

Figure 6: Sampling of a pulse signal

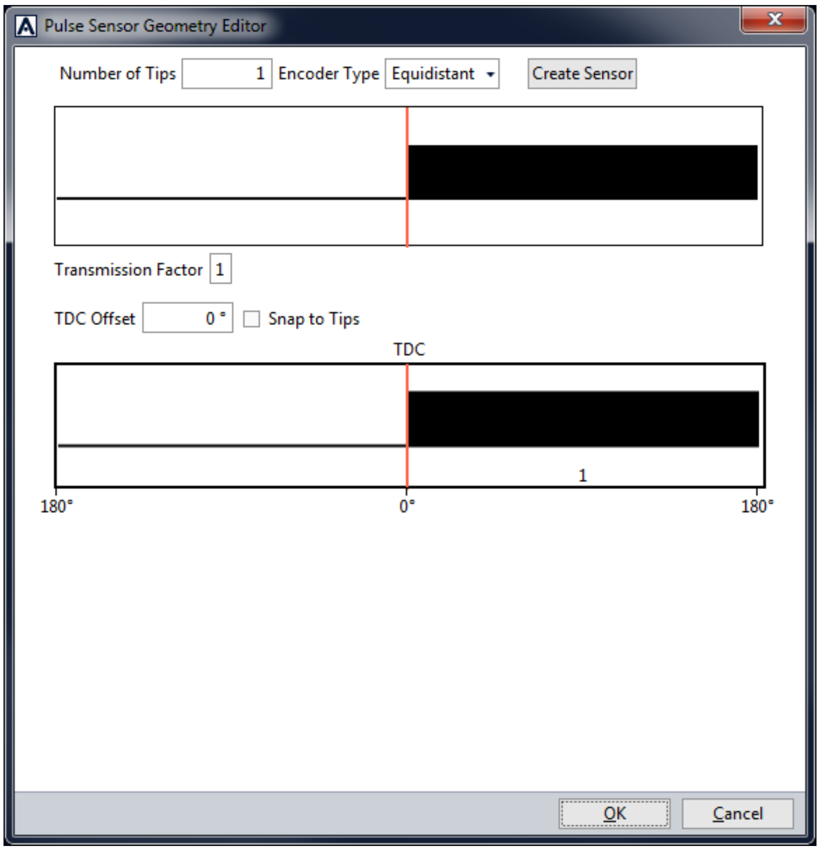
In addition, it is important that the distance between the sampling points is considerably smaller than the pulse width of the signal. This is necessary to ensure that all pulses are reliably found rather than slipping through the sampling grid as did the one in the bottom image of figure 5.
Under some circumstances it can be advisable to sample the pulse channel with a higher rate than the audio channels. For example, if only acceleration channels are recorded, for which a sampling rate of 8

kHz is sufficient, the pulse channel should be sampled at a higher rate (“oversampling”). The pulse inputs of a SQuadriga II are automatically sampled 32 times faster than the audio channels. The pulse inputs of a HEAD*lab* system are sampled by 24 times the main sampling rate (48 kHz).
Some measurement setups, however, deliver a very high pulse frequency. This is the case, for example, if the pulse signal for calculating the revolution speed of an engine running at 6000 revolutions per minute is sampled on a cogwheel with 360 teeth, and the measurement setup is designed so that each tooth delivers a pulse. Such a configuration would result in a pulse frequency of 36,000 pulses per second. When selecting your recording front end, please pay attention to the fact that the sampling rate for the pulse inputs is in line with the pulse frequency of your test setup. If a suitable sampling rate cannot be provided, the test setup has to be modified in a way that a lower pulse frequency is generated.
Another possible source of error exists if the pulse frequency delivered by the measurement setup is too low. A low pulse frequency means a long time interval between the individual pulses. This would cause the revolution speed measurement to be sluggish and thus imprecise due to the long waiting period between the pulses. This delayed availability of the revolution speed values would also have a negative effect on the real-time display of RPM-dependent analyses, which would then display their results with a delay. Most software applications for recording and analysis therefore have a lower frequency limit specified in the program code (e.g., 1 Hz in the HEAD Recorder). As soon as the pulse frequency drops below this limit, i.e., the distance between the pulses becomes too large, the software displays a revolution speed of 0 RPM.

Processing of trigger signals

If revolution speed information for a recording is provided in the form of pulses, but the frontend does not have a digital pulse input for recording them, the pulses can instead be recorded to a normal analog channel and later used for determining the revolution speed in ArtemiS SUITE. Pulse information in an analog channel can be used as a reference quantity by declaring an additional digital channel of the type ***trigger channel***. To do so, proceed as follows.
First, open the file in the Channel Editor (right-click on the file and select ***Edit HDF with*** -> ***Channel Editor***). Then create a digital trigger channel based on the analog channel containing the pulse signal (***Signal Channels*** → right-click on the analog channel with the revolution speed signal → ***Add new Trigger Channel***). This opens a window where you can specify the channel name and the physical quantity and unit for the new channel. For revolution speed, you can leave the default settings in place (***Speed of Rotation*** and ***RPM***). Now click on the button ***Edit Pulse Sensor Geometry*** to open an editor (see figure 7), where you can specify the required sensor information (such as the number of pulses per revolution etc.).
For a simple revolution speed sensor delivering one pulse per revolution, enter ***1*** as the ***Number of Tips*** and set the ***Encoder Type*** to ***Equidistant***[4]. Then click on **Create Sensor** and confirm your entries with the ***OK*** button. Finally, click on **Save Changes** in the Channel Editor to save the trigger channel to your file. Note that this will overwrite the original file.[5]

[4] The procedure for specifying more complex pulse patterns is described in the next chapter.
[5] If you want to keep your original file unchanged, create a copy of it in advance.



Number of Tips: 1
Encoder Type: Equidistant
Transmission Factor: 1
TDC Offset: 0 °
☐ Snap to Tips

Figure 7: The Pulse Sensor Geometry Editor

Of course, in order to reduce your work, it is also possible to apply this procedure for creating a new trigger channel and editing the sensor information to multiple files at once in the Channel Editor. To do so, first highlight all files[6] to which you want to add a trigger channel, e.g., by clicking on them with the left mouse button while keeping the CTRL key pressed. Then open the context menu with a right-click and load all files into the Channel Editor by selecting ***Edit HDF with*** -> ***Channel Editor***.
In the Channel Editor, select all analog signal channels containing pulse information, and perform the steps described above for all selected channels at once.

[6] All selected files must contain pulse information with the same pulse sensor geometry.

Revolution speed acquisition with complex pulse patterns

In the automotive industry, it is not uncommon that several teeth are missing on a cogwheel, e.g., for marking the top dead center. It is also common to use pulse sensors that generate a pulse pattern with non-equidistant pulses or what is called a zebratape pattern.
Acquiring revolution speed information with such pulse sensors exceeds the limits of automatic revolution speed calculation in ArtemiS SUITE because, for example, the missing teeth in case of a cogwheel gap would not be recognized as such, but would be misinterpreted as a sudden drop of the revolution speed (see figure 8).

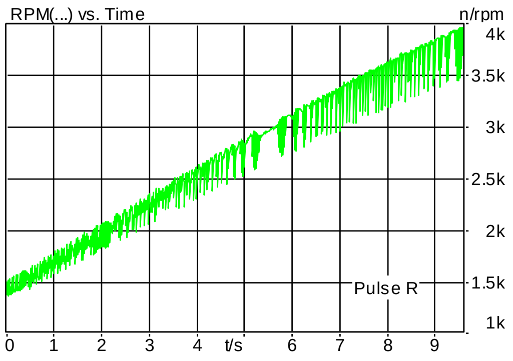
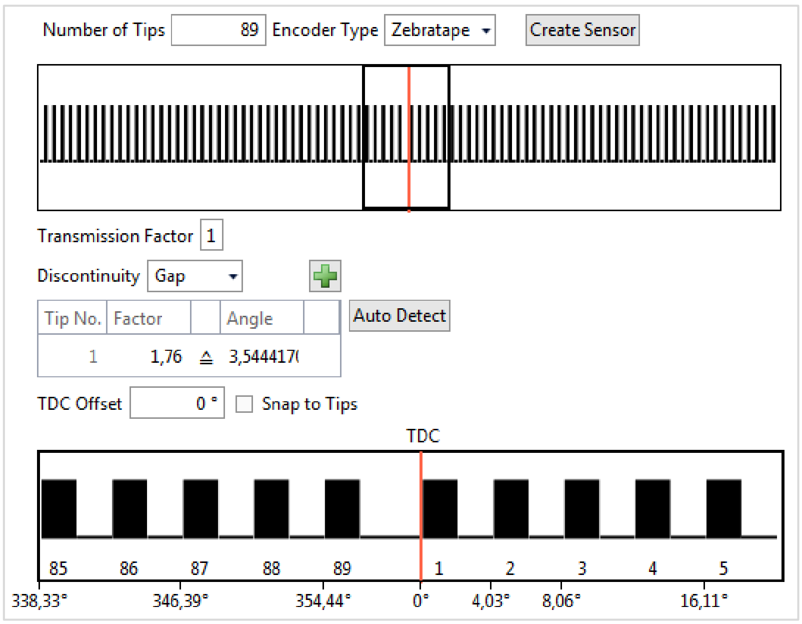


Figure 8: Revolution speed calculation with missing pulses without correction

Using the Pulse Sensor Geometry Editor and a Decoder Project, you can define such a complex pulse pattern for a convenient calculation of the correct revolution speed.
To do so, first open the file in the Channel Editor (right-click on the file -> ***Edit HDF with*** -> ***Channel Editor***).
If your revolution speed information is stored as a pulse signal in an analog channel, first create a digital trigger channel based on the analog channel. As described in the previous section, this will take you to the Pulse Sensor Geometry Editor.
If your revolution speed information is already stored in a digital channel, open the editor via the Properties window of the digital channel (***Pulse Channels*** tab -> right-click on the pulse channel with revolution speed data -> ***Properties*** -> ***Edit Pulse Sensor Geometry***).
In the Pulse Sensor Geometry Editor, you can now define the desired pulse pattern. To do so, first enter the ***Number of Tips*** (teeth, cogs) and select the ***Encoder Type***. The following types are available:

- ***Equidistant***: The markings (e.g., teeth) of the pulse encoder have a constant distance between each other. The width of all elements are the same. After selecting this setting and clicking on ***Create Sensor***, you can specify any pulse gaps and the position of the top dead center (TDC).
- ***Zebratape***: In this case, almost all of the markings (teeth) have the same distance, too. However, unlike the ***Equidistant*** case, there are discontinuities, e.g., where the zebratape overlaps, i.e., the width of a marking or gap is different from that of the other elements. After clicking on ***Create Sensor***, you can exactly specify up to four discontinuities (see figure 9).
- ***Non-Equidistant***: This setting allows you – after clicking on ***Create Sensor*** – to specify the ***Start [°]*** and ***End [°]*** angles individually for each individual marking of your encoder.


Number of Tips: 89
Encoder Type: Zebratape
Transmission Factor: 1
Discontinuity: Gap

Tip No.	Factor	Angle
1	1,76 ≙	3,544170

TDC Offset: 0 °
Snap to Tips: ☐

Figure 9: Definition of a sensor with zebratape geometry

Once you have precisely specified your pulse encoder in the editor, confirm your entries with the OK button; then click on ***Save Changes*** to store the sensor information for all selected channels. Note that this will overwrite your original file.[7] Via multiple selection, you can also perform the sensor definition for multiple files or channels at once. Furthermore, it is possible to store the sensor definition in a sensor library and to load it via the button ***Import from Sensor Library***. This allows you to access your definition of a complex pulse pattern at any time without having to specify it anew each time.
After entering and saving your sensor information, open a Decoder Project in order to decode the revolution speed information from you data based on the sensor information you just specified.
To open a new Decoder Project[8], select ***START*** -> ***New*** -> ***Decoder Project***. A Decoder Project contains three pools similar to those of a normal Pool Project (see figure 10).

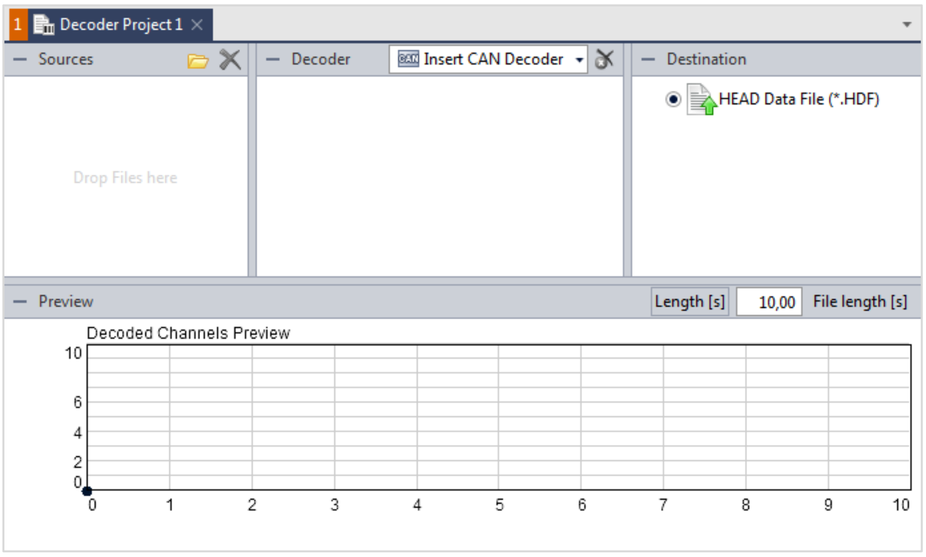
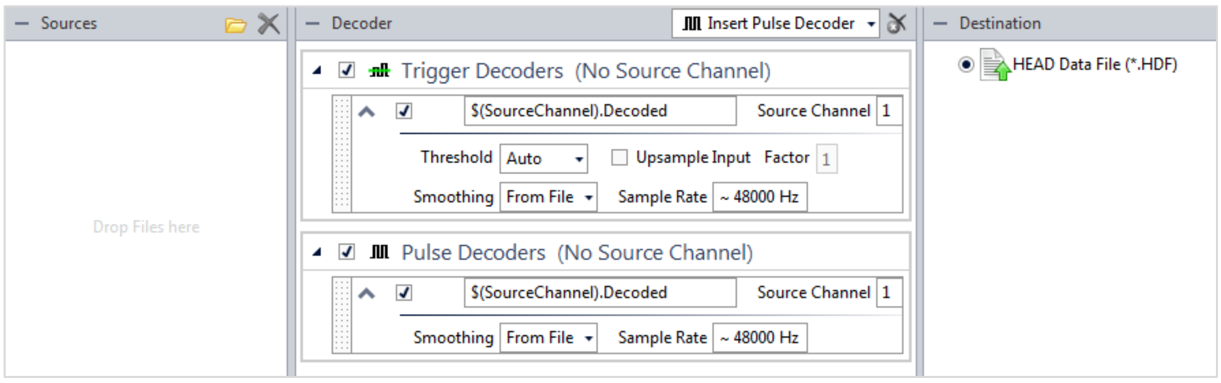

Length [s]: 10,00

Figure 10: Newly created Decoder Project

[7] If you want to keep your original file unchanged, create a copy of it in advance.
[8] A Decoder Project can only be opened if your ArtemiS SUITE license includes the Data Preparation Module (ASM 24).

The Source Pool on the left side allows you to insert the time domain signal(s) containing a complex pulse pattern that is to be decoded for the further analysis. In the center pool, similar to the Analysis Pool of a normal Pool Project, you can select the decoder suitable for your data.
If your revolution speed information is encoded in a digital pulse channel, you must use a pulse decoder for decoding it, whereas revolution speed information stored in a trigger channel requires a trigger decoder. Insert the appropriate decoder type into the Decoder Pool using the button ***Insert Pulse Decoder*** or ***Trigger Decoder***, respectively.
By means of the decoder, ArtemiS SUITE calculates an additional channel, where a time-domain signal representing the current revolution speed is stored. In the preview window of the Decoder Project displayed below the pools (see figure 10), you can inspect the calculated revolution speed curve in advance.
The additional channel can be saved to a new file along with the original channels. In the right pool of the Decoder Project, you can specify the format for this new file. The available export formats are HDF, ATFX, and UFF[9]. If you want to further analyze the file in ArtemiS SUITE, select the HDF format, which allows the file to be inserted directly into a Pool Project for your analysis without any additional conversion.
In a Decoder Project, all input signals are always active. You can also activate multiple decoders at once for the calculation.
In the next step, configure the decoder according to your requirements. The following settings (see figure 11) are available (settings marked with + are only available for the trigger decoder):



Trigger Decoders (No Source Channel)
- ✓ $(SourceChannel).Decoded — Source Channel: 1
- Threshold: Auto — ☐ Upsample Input — Factor: 1
- Smoothing: From File — Sample Rate: ~ 48000 Hz

Pulse Decoders (No Source Channel)
- ✓ $(SourceChannel).Decoded — Source Channel: 1
- Smoothing: From File — Sample Rate: ~ 48000 Hz

● HEAD Data File (*.HDF)

Figure 11: Trigger decoder (top) and pulse decoder (bottom) in the Decoder Project

- ***Source Channel***: Specify the trigger channel created in the Channel Editor here. You can enter either the channel name or the channel number (wildcards * are supported).
- ***Threshold***$^{+}$**:** Here you can choose how the threshold value for pulse detection is set. Three modes are available: ***Auto***, ***Relative***, and ***Manual***. A detailed explanation of these settings can be found in the Help System of ArtemiS SUITE.
- ***Upsample Input***, ***Factor***$^{+}$**:** If this option is enabled, you can enter a factor in the entry field, by which the sampling rate for sampling the analog signal is to be increased prior to threshold detection. This can considerably reduce the number of artifacts in cases where the sampling rate is low compared to the measured revolution speed, leading to more reliable decoding due to the more precise interpolation of the signal curve.
- ***Smoothing***: Here you can specify to what degree the data are to be interpolated (averaged) when being sampled.
- ***Sample Rate***: Here you can specify the desired sampling rate for the decoded channel. The actual value is adapted (based on the respective source file) for optimal storage in the HEAD Data Format and is displayed in the legend of the preview window.

[9] UFF export can only be used if your ArtemiS SUITE license includes the Advanced Import & Export Module (ASM 23).

Once the decoder is configured and the preview window shows the correct revolution speed curve, you can create the new file(s) by clicking on the abacus icon and insert them into a Pool Project for further analysis. A particularly convenient way to do this is using the ***Recent Results*** list of the HEAD Navigator. The results calculated most recently can be found at the top of this list and can simply be dragged and dropped into a Pool Project. The new file contains an additional revolution speed channel with the newly decoded revolution speed curve.

Do you have questions or suggestions?
Please do not hesitate to contact us at imke.hauswirth@head-acoustics.de.
We look forward to your feedback!