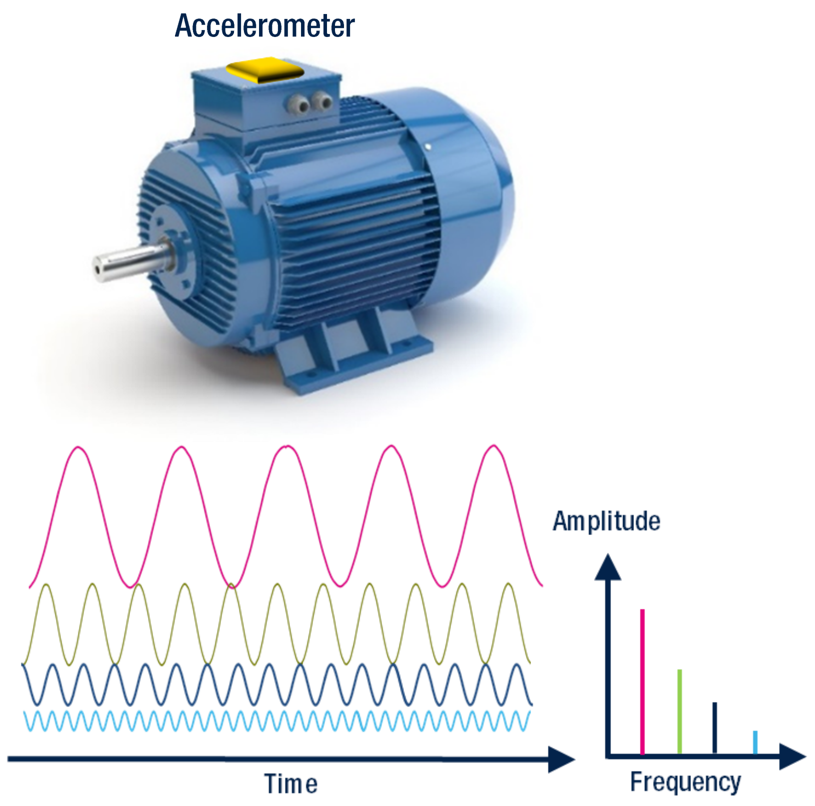
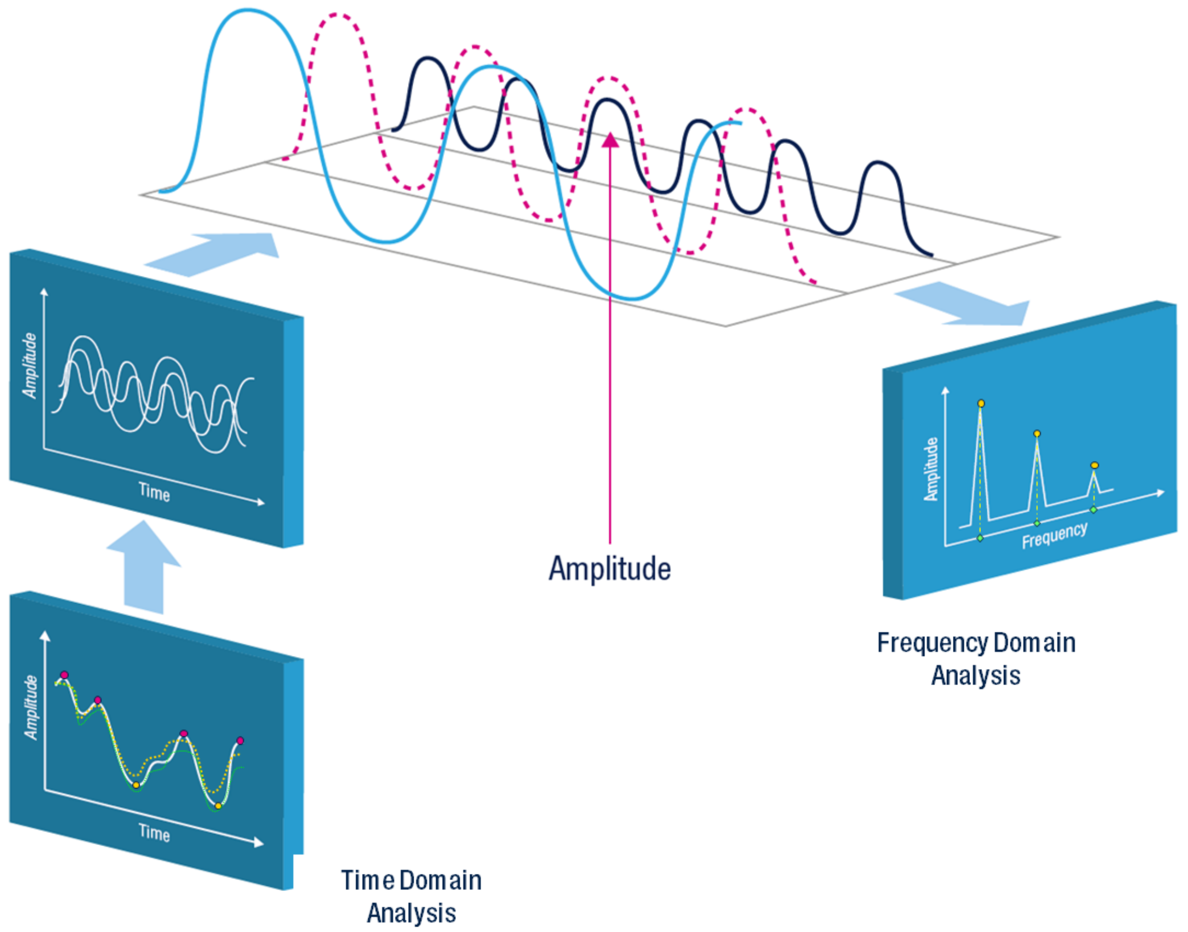
# MotionSP Middleware (Rel. 2.3.2)

## Implementation for FP-IND-PREDMNT1

# Vibration Monitoring using **MotionSP** Middleware





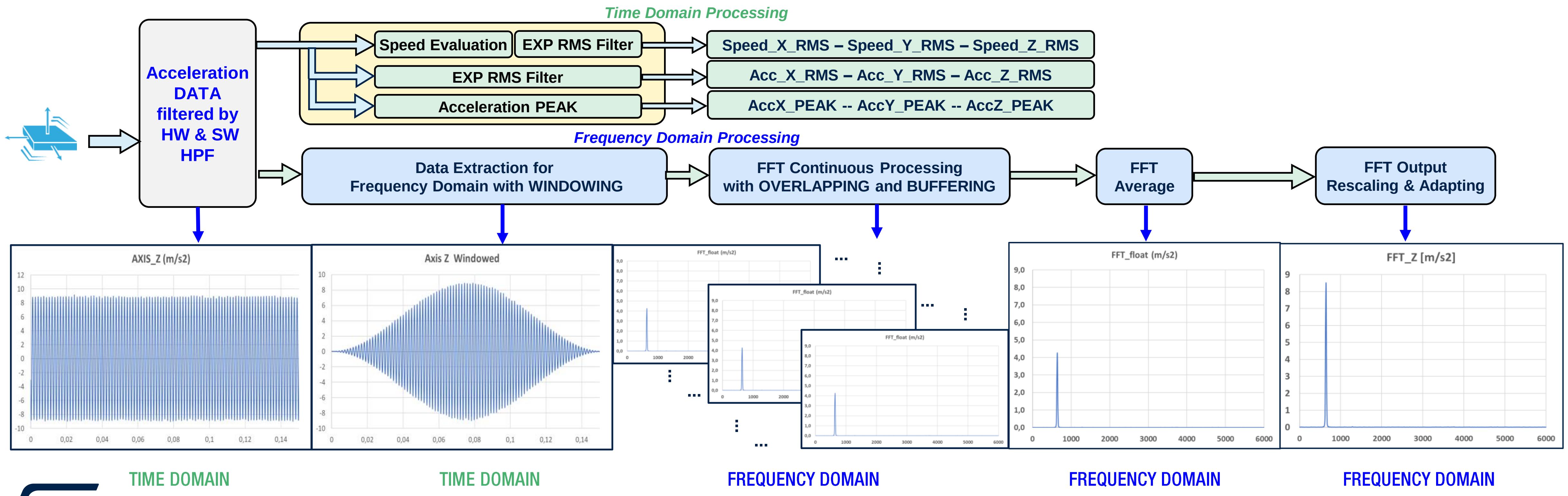
In rotating equipment, the vibrational data are often used to get indication of motor condition health, so, starting from the data incoming from an accelerometer mounted on the machinery and performing the proper analysis, is possible to control all the equipment functionality.
A standard Signal processing is usually based on Time & Frequency Domain Analysis as exposed above

life.augmented

# MotionSP Architecture Overview

## Middleware for advanced Time & Frequency domain Signal Processing

- ❑ **Time Domain Analysis:**
  - **High Pass Filtering -- Speed Estimation -- Exponential Filtering for RMS** moving average (Speed or Acceleration) -- **Max peak** Evaluation
- ❑ **Frequency Domain Analysis**:
  - Programmable **FFT Size** (256, 512, 1024, 2048 points)
  - Programmable Total Acquisition Time **(Tacq)** for analysis
  - Programmable **Windowing Methods**: Rectangular, Hanning, Hamming, Flat Top
  - Programmable **FFT overlapping (OVL)** percentage and continuous FFT outputs buffering to perform the **FFT averaging (AVG)** when Tacq is expired.



Time Domain Processing
Acceleration DATA filtered by HW & SW HPF
Speed Evaluation
EXP RMS Filter
EXP RMS Filter
Acceleration PEAK
Speed_X_RMS – Speed_Y_RMS – Speed_Z_RMS
Acc_X_RMS – Acc_Y_RMS – Acc_Z_RMS
AccX_PEAK -- AccY_PEAK -- AccZ_PEAK
Frequency Domain Processing
Data Extraction for Frequency Domain with WINDOWING
FFT Continuous Processing with OVERLAPPING and BUFFERING
FFT Average
FFT Output Rescaling & Adapting
AXIS_Z (m/s2)
Axis Z Windowed
FFT_float (m/s2)
FFT_float (m/s2)
FFT_float (m/s2)
FFT_float (m/s2)
FFT_Z [m/s2]
TIME DOMAIN
TIME DOMAIN
FREQUENCY DOMAIN
FREQUENCY DOMAIN
FREQUENCY DOMAIN

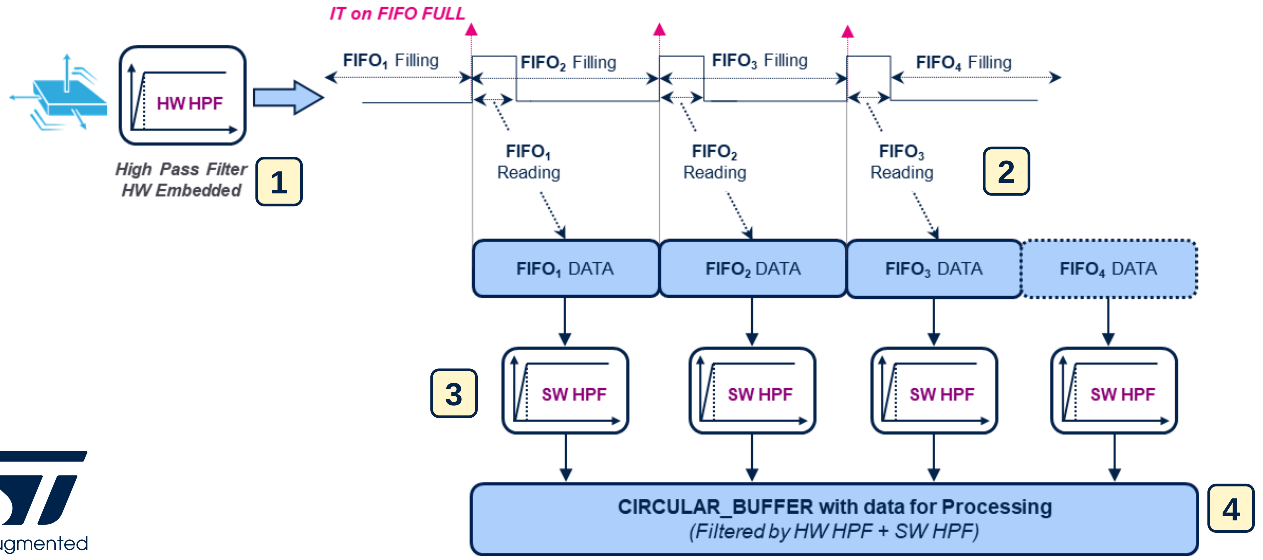
## Time Domain Processing

life.augmented

# Acceleration Data from FIFO to Circular Buffer

## Pre-Filtering the data to store inside the Circular Buffer for Processing

- The acceleration data incoming from the MEMS accelerometer are usually affected by an intrinsic offset which is easily recognizable as a DC bias in its waveform. Any numerical processing that uses this acceleration data will be affected by a drift respect to the theoretical expected value, causing an unpredictable error for all subsequent processing in the time or frequency domain.
- To avoid this drift, is better to enable the internal **HW H**igh **P**ass **F**ilter **(HW HPF)** available inside the MEMS,  in order to store the data already filtered inside the internal FIFO. In case of a large drift is possible to add a **SW H**igh **P**ass **F**ilter **(SW HPF)** , available inside the library, during the circular buffer creation, that will be the final data storage for Time and Frequency Domain processing.
- This processing chain can operate in continuous mode, providing all the data to the processing, but the user, at the application level, must pay attention to the FIFO management: FIFO reading time must be smaller than the FIFO filling time, such as detailed in the figure below

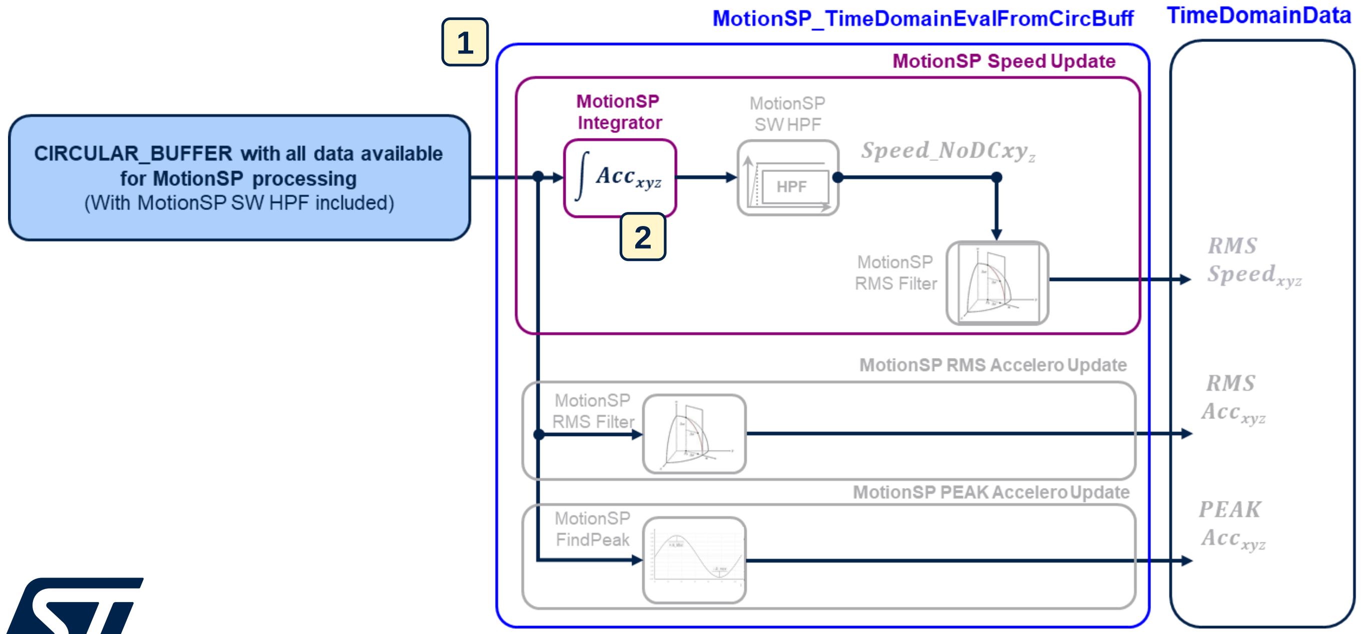


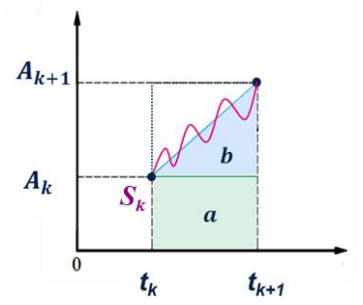
| | Application API used |
|---|---|
| | MotionSP_VibrationAnalysisVariableInit |
| 1 | EnableDisable_ACC_HP_Filter |
| | MotionSP_AcceleroFifoConfig |
| 2 | MOTION_SENSOR_FIFO_Read |
| 4 | FillAccCircBuffFromFifo |

| | MotionSP API used |
|---|---|
| 3 | MotionSP_accDelOffset |

# Speed Estimation using numerical integrator block

- Once the static component of the acceleration has been filtered using HW & SW high pass filter, we can perform a speed estimation using a numerical integration method on the acceleration values.
- The numerical integration technique used is the **Trapezoidal Rule**, where works by approximating the area under unknown function calculating the trapezoidal underlying area; it belongs to the finite difference methods, starting from the discretization of waveform to integrate, using equal time steps **ΔT**, such as is already provided by the digital accelerometer such as described in the picture below.
- Fixing a speed value at the initial step, this method allows to evaluate the next speed value using two consecutive acceleration samples **($A_k$ , $A_{k+1}$)** the time sampling (**$\Delta T = t_{k+1} - t_k = 1/ODR$**), and the previous speed value **($S_k$ )**.

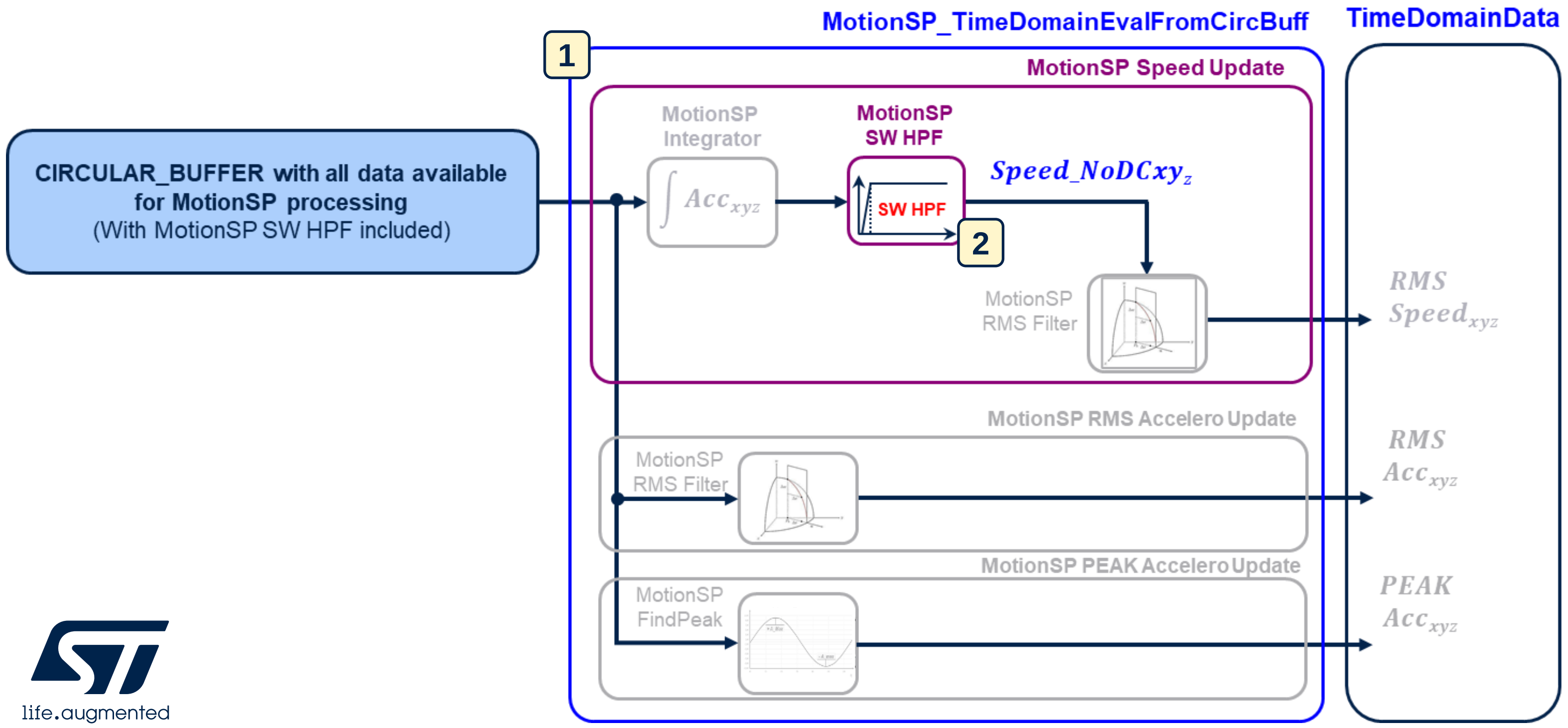




| | MotionSP API used |
|---|---|
| 1 | MotionSP_TimeDomainEvalFromCircBuff |
| 2 | MotionSP_TD_SpeedEvalFromCircBuff |

# Speed Filtering

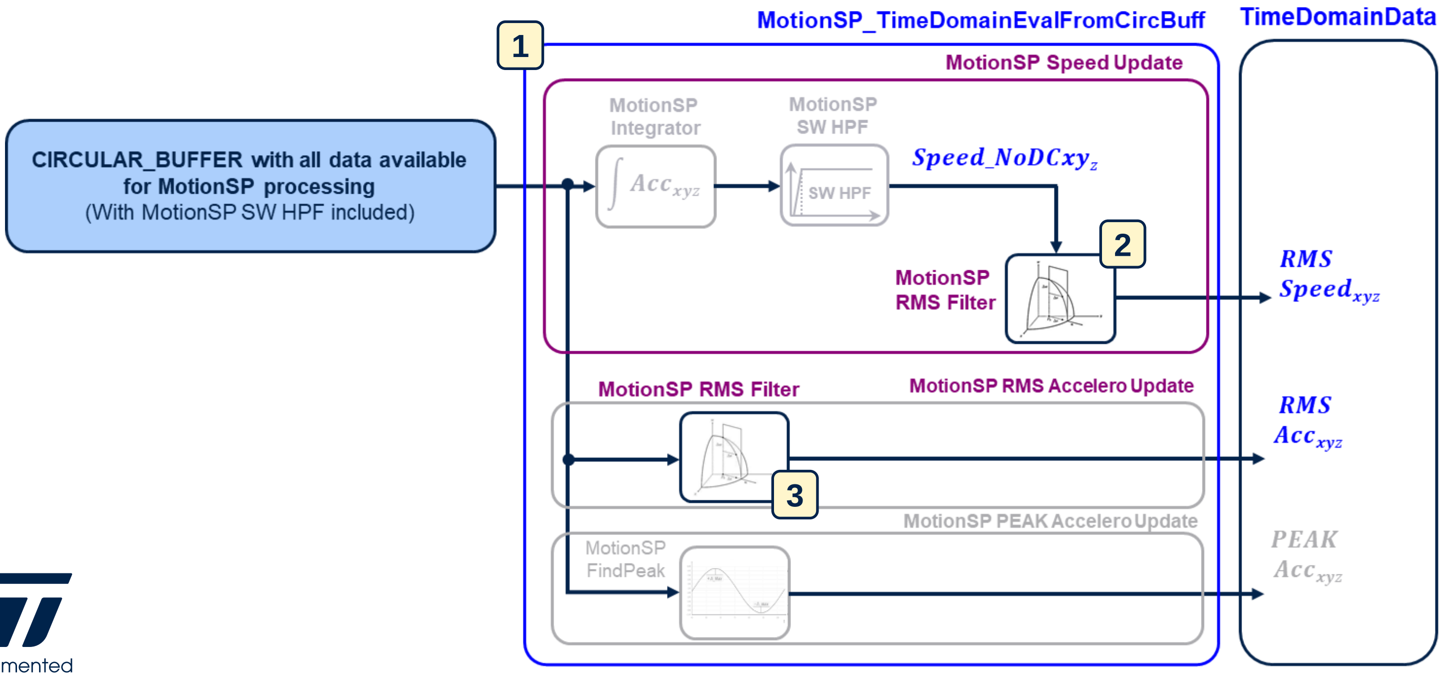
- Another problem that can interfere with the processing accuracy is that the initial speed value is not known with an acceptable degree of certainty.
- The estimation fixed for the initial speed value may inherit large linear drift errors because the signal will probably retain a small DC component after it has been integrated.
- To avoid this high dependence on the initial conditions, we can insert another **SW HPF** block more after the numerical integration run for speed estimation, so this method allows to reduce also the intrinsic offset produced by the integrator, avoiding an undesired drift on the vibrational speed values.



| | MotionSP API used |
|---|---|
| 1 | MotionSP_TimeDomainEvalFromCircBuff |
| 2 | MotionSP_speedDelOffset |

# Acceleration & Speed RMS estimation

- ❑ Many RMS methods are based on standard formulas that work on a well-defined data array, which is filled in and processed from time to time.
- ❑ This approach is not very suitable for vibration analysis where it is preferred to monitor the RMS value in continuous mode, observing its trend, so, for this reason it is preferred to focus the computational effort towards a moving average method that requires less use of data buffering and continuous availability of the estimated RMS.
- ❑ In this library the Acceleration & Speed RMS are estimated using a recursive 1st order filter based on "**Exponential Weighting Method Average**" which allows to perform a sort of Moving Root Mean Square processing, such as already included in the Matlab® DSP Toolbox for signal processing as ***dsp.Moving RMS function.***

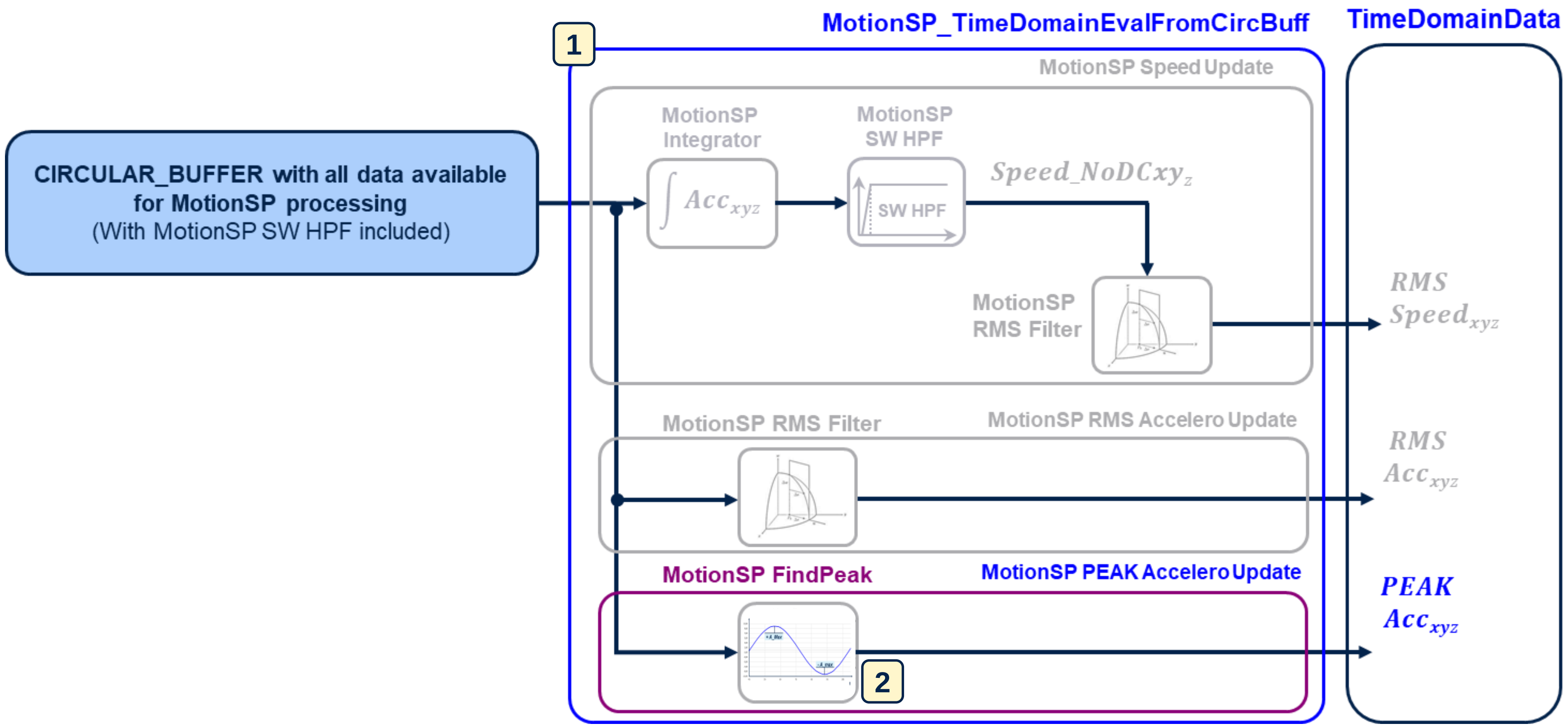


| | MotionSP API used |
|---|---|
| 1 | MotionSP_TimeDomainEvalFromCircBuff |
| 2 | MotionSP_SwSpeedRmsFilter |
| 3 | MotionSP_TD_AccRmsEvalFromCircBuff |

life.augmented

# Acceleration Peak

- The peak processing method is simply based on continuous comparison between all the values incoming from the FIFO reading, during the circular buffer filling, taking care to their absolute value

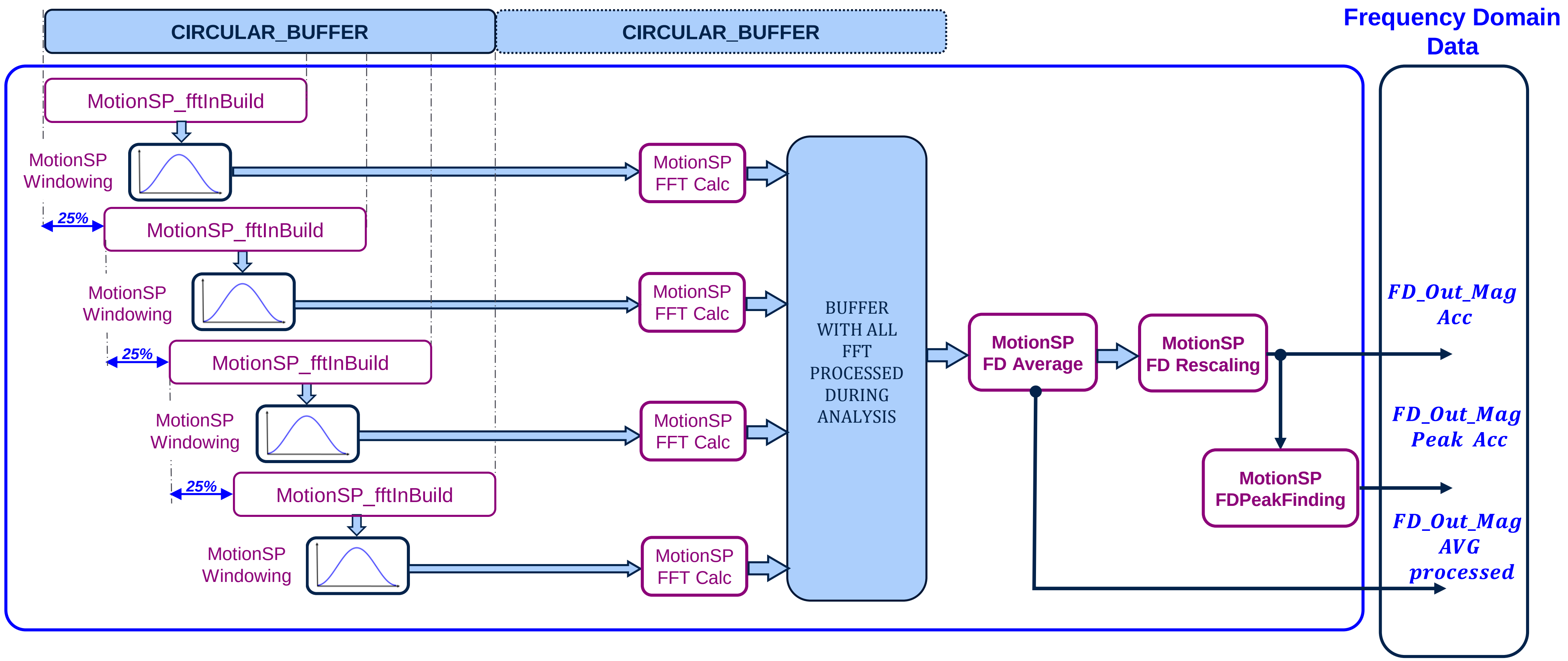


| MotionSP API used | |
|---|---|
| 1 | MotionSP_TimeDomainEvalFromCircBuff |
| 2 | MotionSP_TD_PeakEvalFromCircBuff |

## Frequency Domain Processing

life.augmented

# Frequency Domain Process Architecture



*Example with Overlapping = 75%*

# Frequency Domain Windowing

## Windowing functionality to avoid spectral leackage

- To perform signal frequency analysis, all the vibrational time domain data must be discretized and gathered over a time interval and then broken down into component frequency waves through an optimized Discrete Fourier Transform (DFT) algorithm known as Fast Fourier Transform (FFT).
- The parameters used in an FFT are:
  - N = discrete time domain samples number
  - fs = sampling frequency
  - fmax = fs/2 = maximum frequency spectrum
  - fs/N = frequency resolution; space between frequency lines (bins frequency step)
- FFT is applied successfully to a periodic signal and when an integer number of periods is contained in the acquisition time (**Fig.a**).
- Since the FFT is applied to a truncated signal, the number of signal periods could not be an integer and the acquired signal endpoints are discontinuous (**Fig.b**); these discontinuities causes other frequencies and the FFT is not the right spectrum of the original signal, but a spread version, with leakage components.

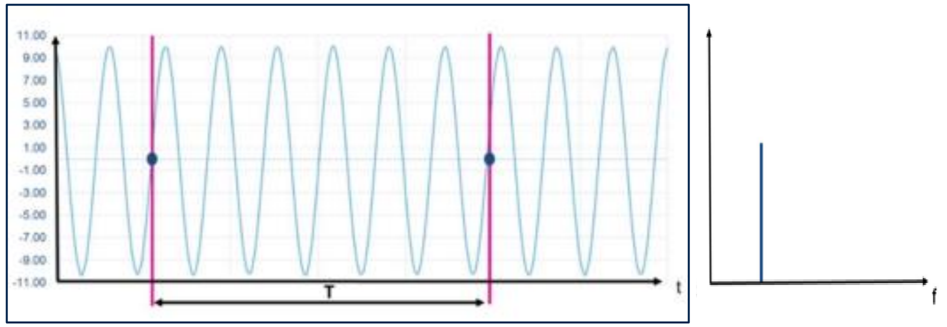


**Fig.a**

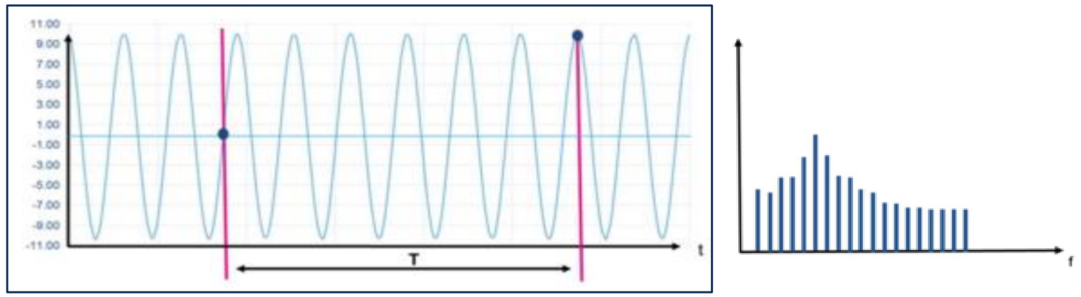
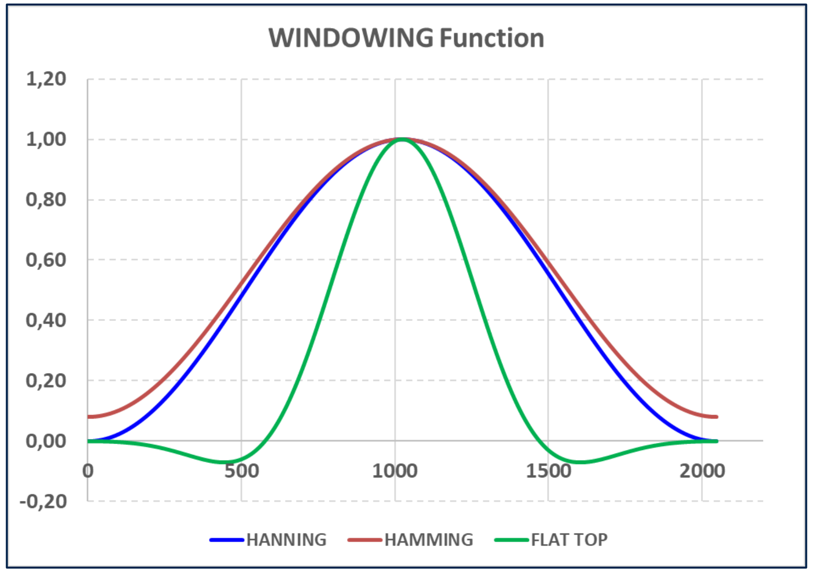


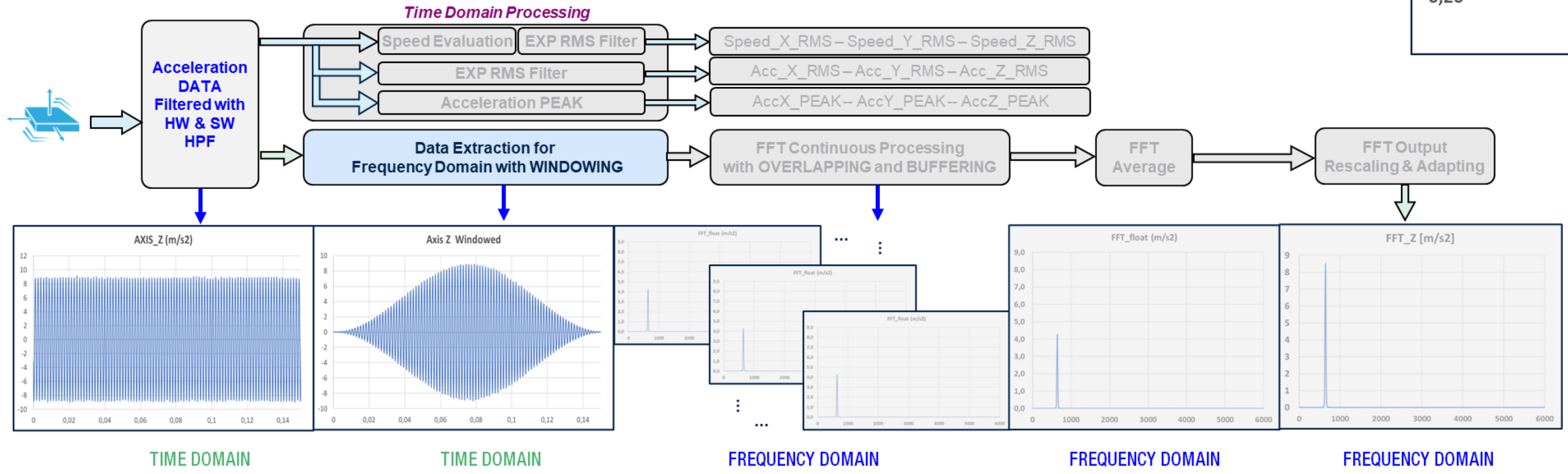
**Fig.b**

# Windowing Function

- ❑ To obtain reliable frequency domain data, certain data sampling techniques must be applied.
- ❑ To avoid the spectral leakage, a window function is multiplied to the acquired time domain data set before the FFT processing.
- ❑ The correct window function to apply depends on a trade-off between the width of the main lobe, side lobe attenuation, process loss and spectral leakage.
- ❑ The **HANNING**, **HAMMING**, **FLAT TOP** window represents a good solution for composite signals typical of rotating equipment



*Example with 2048 Samples*



Time Domain Processing
Acceleration DATA Filtered with HW & SW HPF
Speed Evaluation
EXP RMS Filter
EXP RMS Filter
Acceleration PEAK
Speed_X_RMS – Speed_Y_RMS – Speed_Z_RMS
Acc_X_RMS – Acc_Y_RMS – Acc_Z_RMS
AccX_PEAK-- AccY_PEAK-- AccZ_PEAK
Data Extraction for Frequency Domain with WINDOWING
FFT Continuous Processing with OVERLAPPING and BUFFERING
FFT Average
FFT Output Rescaling & Adapting
AXIS_Z (m/s2)
Axis Z Windowed
FFT_float (m/s2)
FFT_Z [m/s2]
TIME DOMAIN
TIME DOMAIN
FREQUENCY DOMAIN
FREQUENCY DOMAIN
FREQUENCY DOMAIN

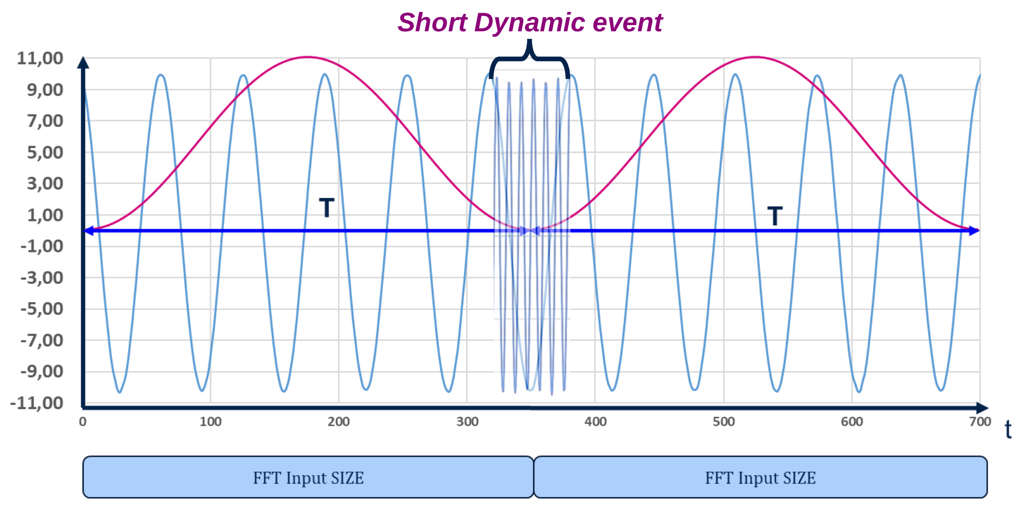
| Application API used |
|---|
| MotionSP_VibrationAnalysisVariableInit |

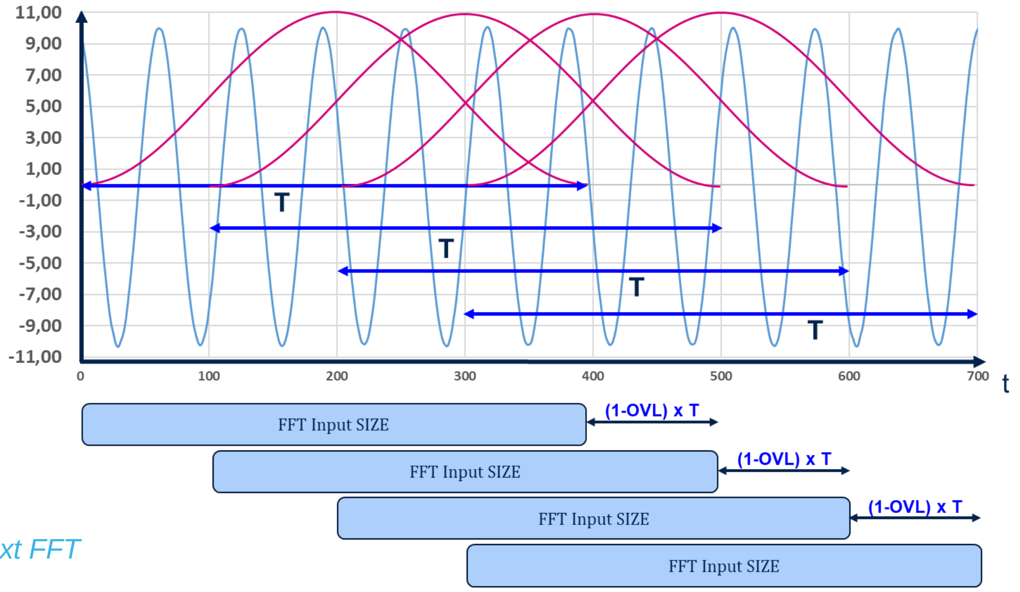
| MotionSP API used |
|---|
| MotionSP_SetWindFiltArray |

# Frequency Domain Overlapping

## Method to speed up the analysis timing covering the dynamic events

- Spectrum analyzers usually acquire the data sequentially, processing an FFT whenever all the expected samples are available
- This method has limits, even after inserting the window, for those events that occur during the transition between two FFTs, or that are due to events with short and fast dynamics, this because the shortest detectable event is related to the buffering time of all the samples in input.
- The visibility of events occurring faster than this buffering time can be reached by overlaying multiple Time-to-Frequency Transformations.
- Multiple FFTs will be performed, based on the same samples available in the sequential frame standard, but will be processed using multiple overlapping frames, so to detect all the different dynamic events.
- After the 1st FFT, the next FFTs will be executed using **old data** (FFT SIZE x OVL) plus the **new data** (FFT SIZE x (1-OVL)).





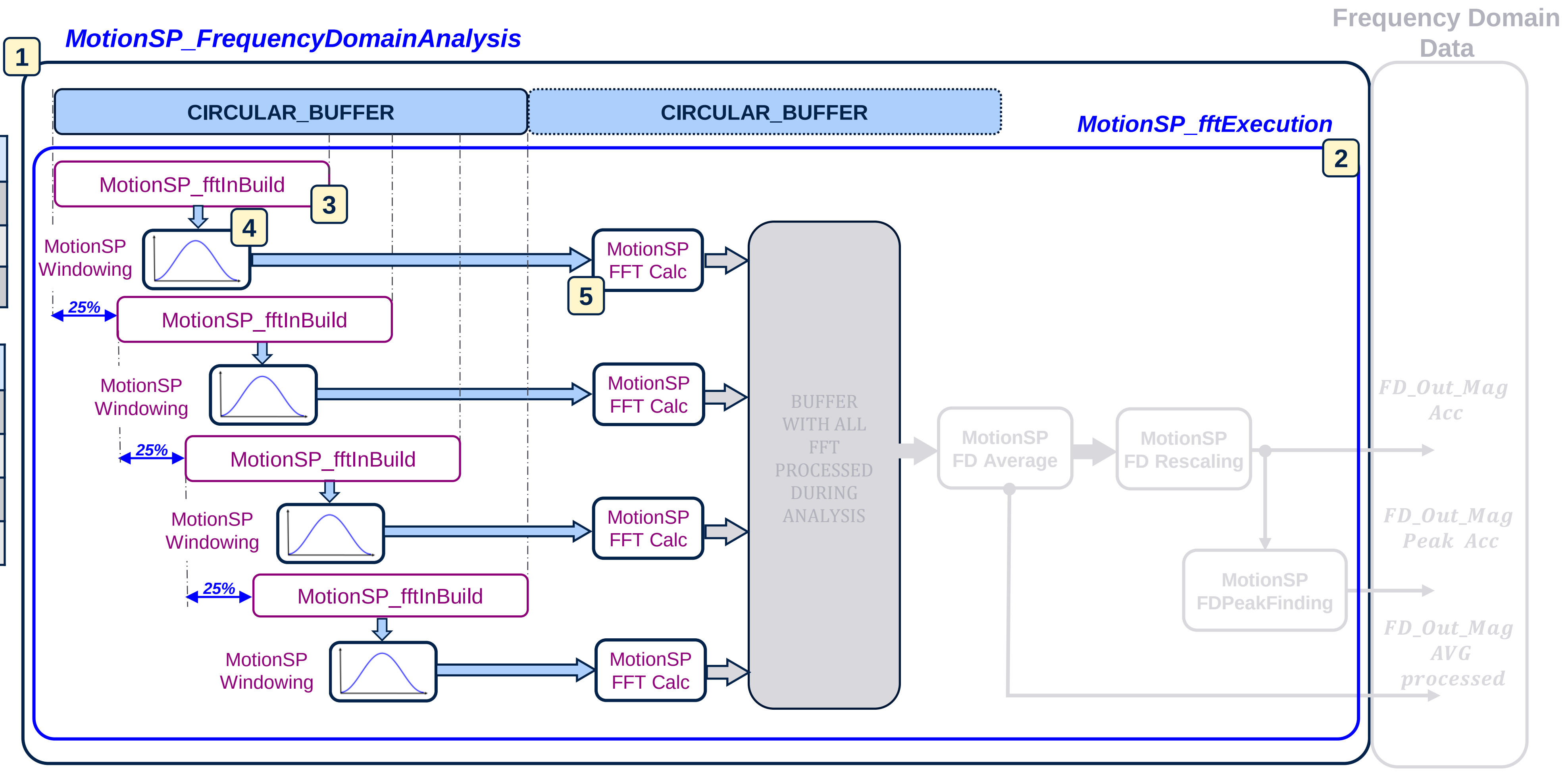
**T** = *Buffering Time to acquire sample up to FFT Input Size*
**(1-OVL) x T** = *Buffering Time to acquire new data to process the next FFT*

life.augmented

# Frequency Domain Overlapping

| Application API used |
|---|
| MotionSP_MainManager |
| MotionSP_VibrationAnalysisManager |
| 1 MotionSP_FrequencyDomainAnalysis |

| | MotionSP API used |
|---|---|
| 2 | MotionSP_fftExecution |
| 3 | MotionSP_fftInBuild |
| 4 | motionSP_fftUseWindow |
| 5 | MotionSP_fftCalc |



Example with Overlapping = 75%

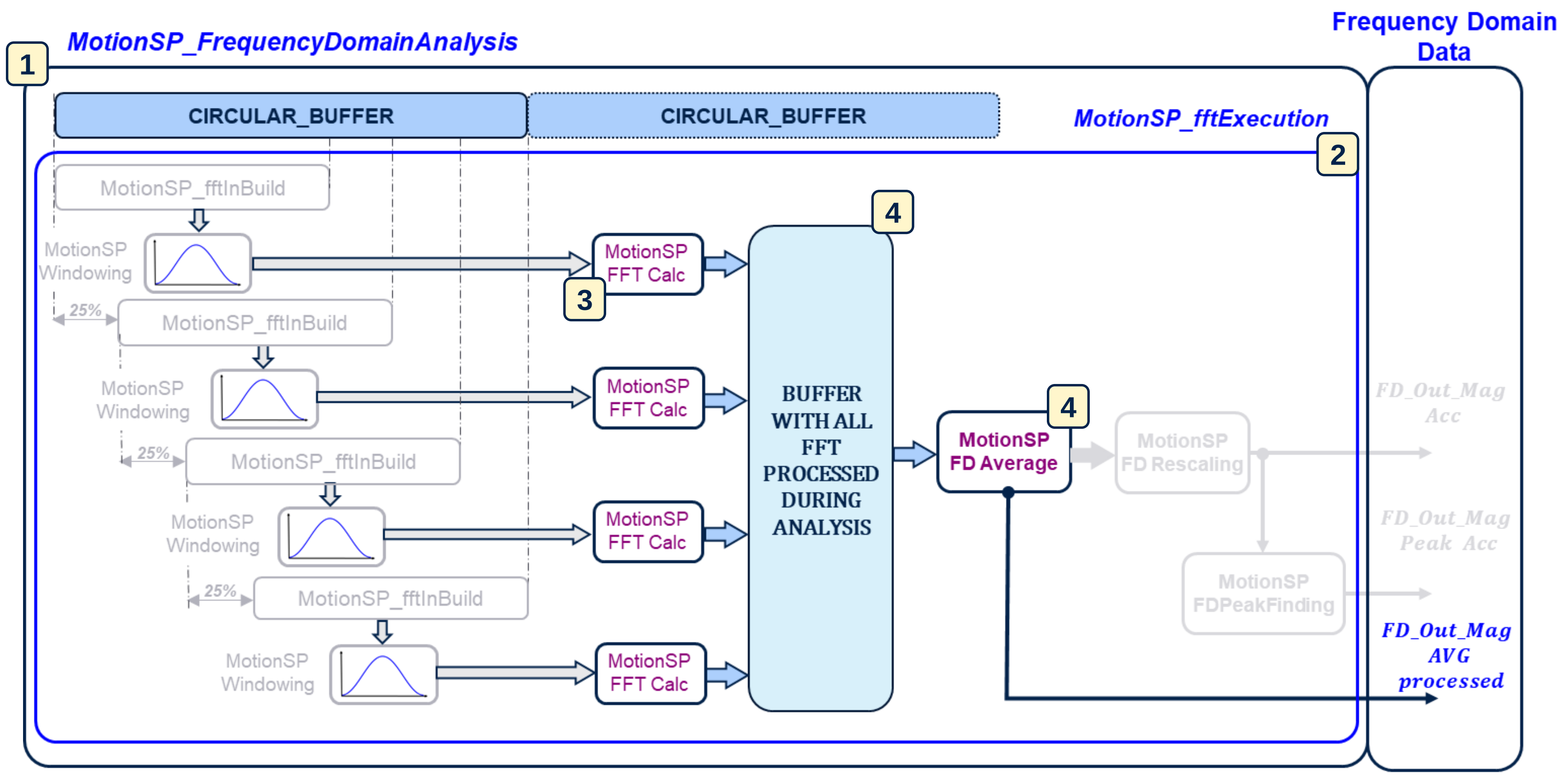
# Frequency Domain Averaging

## Methods for noise reduction

- All the signal processing mentioned before do not address the probable noise due to discrete processing and accelerometer sensitivity.
- The solution proposed for noise is to sum all the multiple FFTs computed during a predefined acquisition time and, when it is elapsed, processing the average amplitude values for each harmonic component.

|   | Application API used |
|---|---|
|   | MotionSP_MainManager |
|   | MotionSP_VibrationAnalysisManager |
| 1 | MotionSP_FrequencyDomainAnalysis |

|   | MotionSP API used |
|---|---|
| 2 | MotionSP_fftExecution |
| 3 | MotionSP_fftCalc |
| 4 | MotionSP_fftAverageCalcTime |



*Example with Overlapping = 75%*

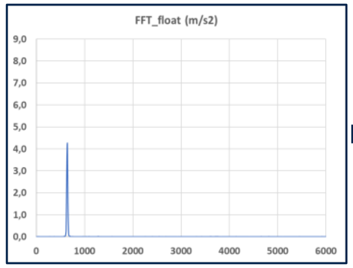
# Frequency Domain Output Rescaling

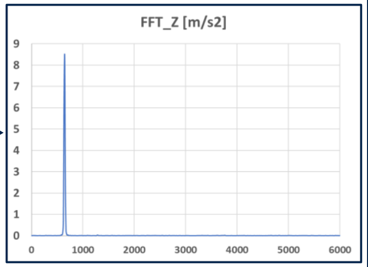
## FFT Output Normalization

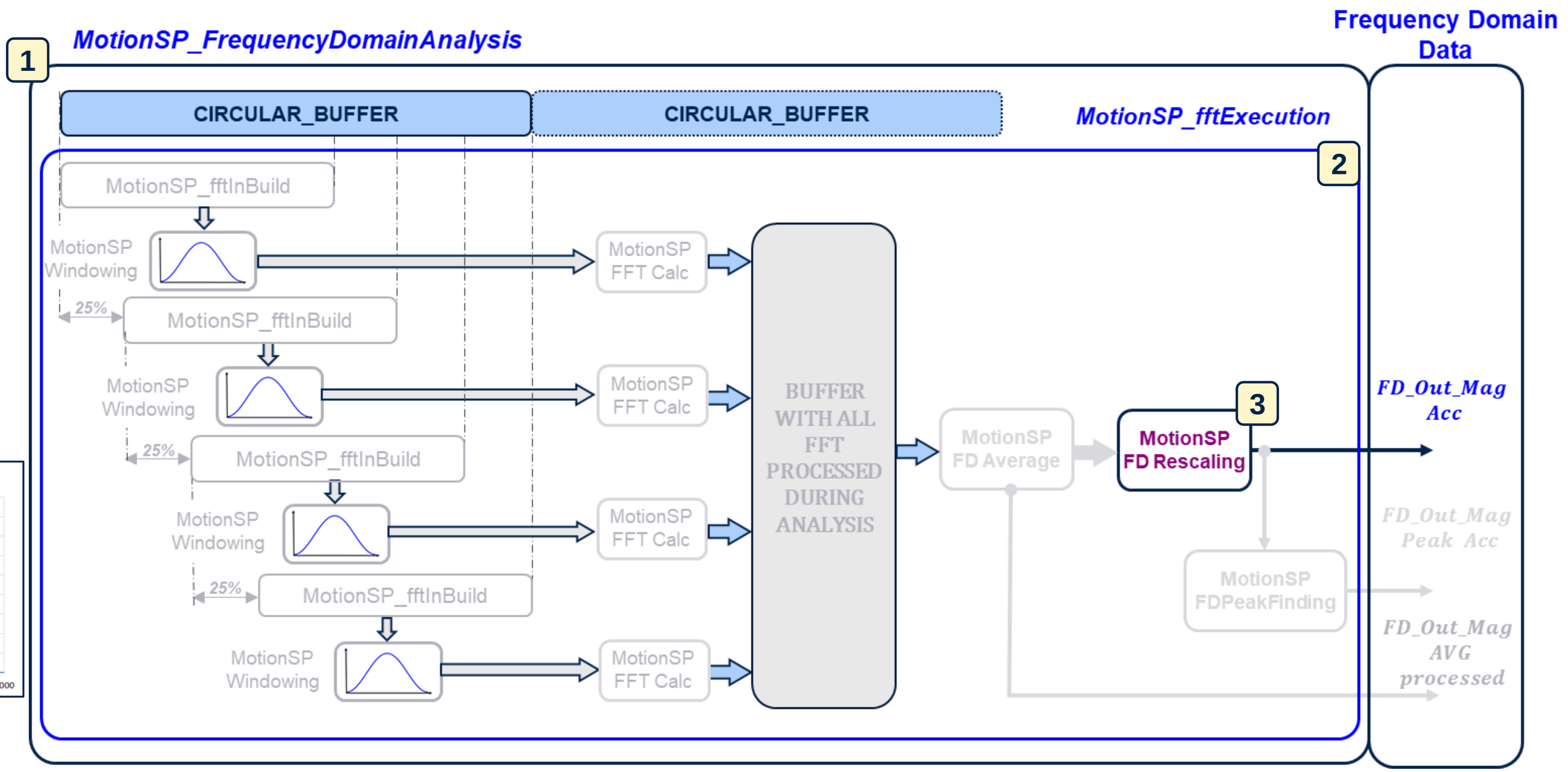
- The frequency domain outputs must be normalized in order to adapt what is processed with the real input data amplitude in time domain.
- This normalization is executed considering a specific rescaling factor related to the windowing filter used in input.

| | Application API used |
|---|---|
| | MotionSP_MainManager |
| | MotionSP_VibrationAnalysisManager |
| 1 | MotionSP_FrequencyDomainAnalysis |

| | MotionSP API used |
|---|---|
| 2 | MotionSP_fftExecution |
| 3 | MotionSP_fftAdapting |







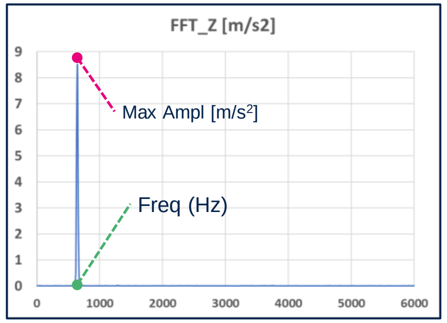
# Frequency Domain Output Peak Finding

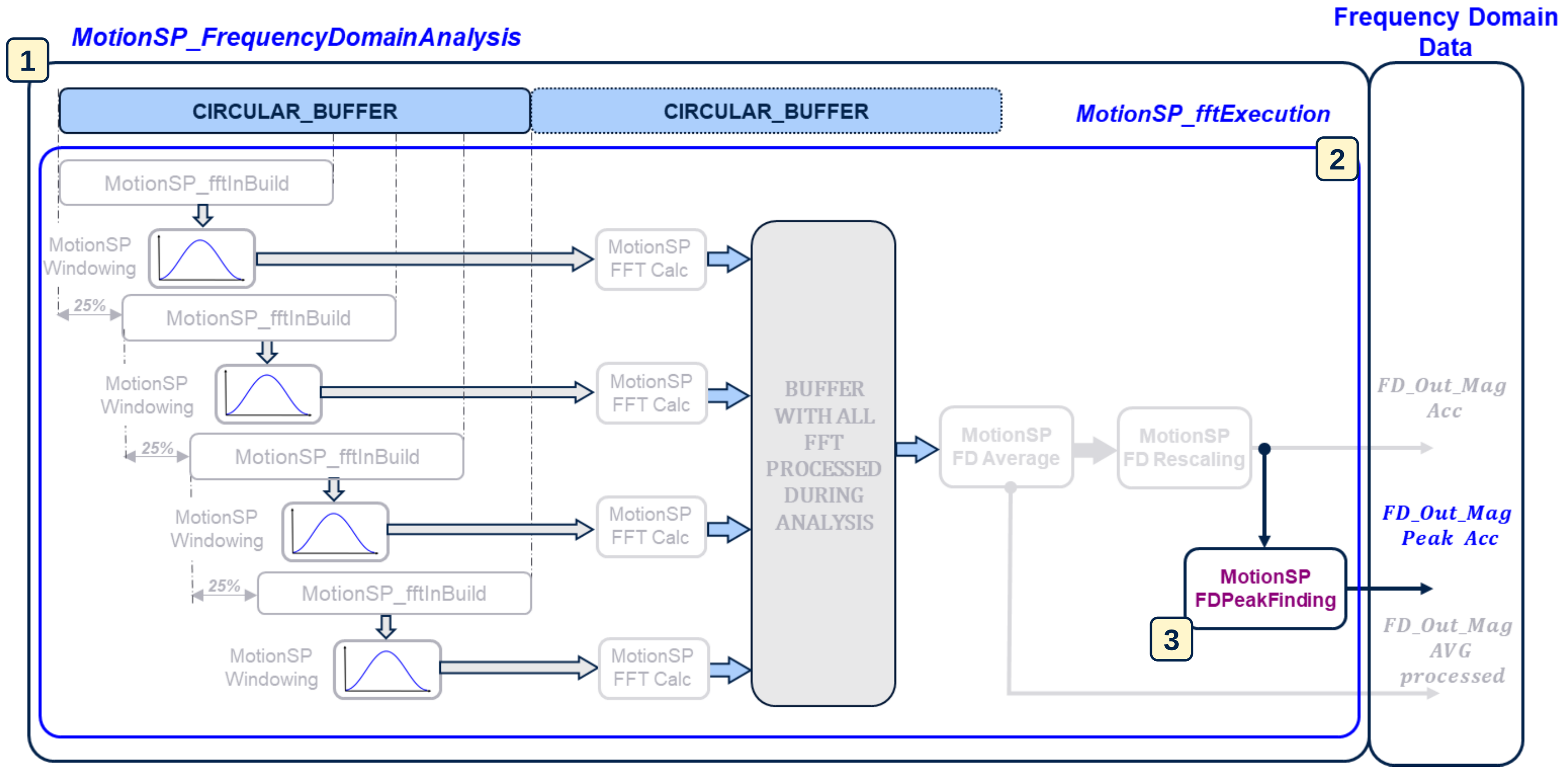
## Max Amplitude Evaluation

- ❑ The spectrum in the frequency domain is usually analyzed considering all the amplitudes calculated for each frequencies in the bandwidth but, in many cases is just important to detect the output the maximum absolute amplitude with the relative frequency.

| | Application API used |
|---|---|
| | MotionSP_MainManager |
| | MotionSP_VibrationAnalysisManager |
| 1 | MotionSP_FrequencyDomainAnalysis |

| | MotionSP API used |
|---|---|
| 2 | MotionSP_fftExecution |
| 3 | MotionSP_fftPeakFinding |





## MotionSP Configuration

- All the MotionSP implementation is included in the middleware but also at application level, where are handled the data incoming from the accelerometer, the circular buffer and some structure to support the processing and data sending via BLE or WiFi connectivity.
- At application level the mandatory files are ***MotionSP_Manager.c / MotionSP_Manager.h***
- The middleware configuration parameters are available in ***MotionSP_Config.h:***
  - **SIZE**: FD input data size for FFT processing
  - **ODR**: Real data input sampling time
  - **FS**: Full Scale
  - **OVL**: FFT overlapping
  - **TACQ**: Total acquisition time for analysis

Our technology starts with You

Find out more at www.st.com

life.augmented