FP ZERO

NOTE: This page is about the C3D Parameter FORCE_PLATFORM:ZERO. For Visual3D's ability to track different baseline frames for different force platforms see FP_ZEROS.

The FORCE_PLATFORM:ZERO C3D parameter defines the range of frames used to calculate an average baseline value for each ANALOG signal that is then subtracted from every frame. This does not affect the ANALOG signals (as seen in the data tree), this processing takes place when the Force Platform signals are computed.

Note that most of the manufacturers arbitrarily set the range to be the first 10 frames of the trial. If the force platform is loaded during these first 10 frames, the ground reaction force signals will be incorrect; in this case the user should either:

- set the frames to 0,0 so that no baseline is subtracted; or
- select frames in which the force platform is unloaded.

Modify_Force_Platform_Parameters

The FP_ZERO parameter can be updated using the Modify_Force_Platform_Parameters pipeline command and specifying the desired /FP_ZERO parameter. It can also be updated interactively using the associated dialog.

```
Modify_Force_Platform_Parameters
/FP_USED=2
/FP_ZERO=0+0
/FP_ZEROS=0+0+0+0
;
```

NOTE: The value specified for /FP_ZERO can be an expression.

General Examples

The following two examples are general examples for working with the ZERO parameter.

Example 1: Using the last 10 frames for non-treadmill data

For non-treadmill data we recommend that you choose the last 10 frames of the trial as almost every activity ends with the participant off of the force platform.

```
Modify_Force_Platform_Parameters
/FP_USED=2
/FP_ZERO=EOF-10+EOF
/FP_ZEROS=EOF-10+EOF+EOF-10+EOF
```

```
;
```

Example 2: Using COFP DATA_NOT_FOUND for Treadmill Trials

Working with treadmill data, we suggest finding a period of time for each force platform where the COFP signal is DATA_NOT_FOUND since this is a reliable indication that the force platform is unloaded and that these frames can be used as a baseline. This will likely have to be done manually per force platform and per file as your treadmill force platforms will likely have different loading and unloading patterns throughout the trials.

Examples Removing Baselines without the ZERO Parameter

In some cases it is not feasible to use the ZERO parameter to define a BASELINE for each signal. All force platform signals have some drift in the amplifiers, so it isn't a good idea to simply set the ZERO parameter to 0,0 so that no baseline is subtracted. The following three examples demonstrate alternatives to using the ZERO parameter to remove a baseline. In each example the ZERO parameter is set to 0,0 so that no inadvertent baseline is removed:

Example 1: Subtract the mean value for several frame ranges

This example demonstrates how to subtract the mean value from several ranges from each analog signal. Consider a subject running on an instrumented treadmill. The mean value of each ANALOG signal is determined when the subject is airborne (e.g. between LTO and RHS) and a collection of these airborne phases are averaged. This overall mean value is to be subtracted from each ANALOG signal. The option to use Processed Analog signals for the Force Platform calculations should be selected. The FORCE_PLATFORM:ZERO parameter is set to (0,0) because it is to be ignored.

Given 12 ANALOG signals associated with a type 6 Force Platform:
FX1,FY1,FZ1,FX2,FY2,FZ2,FX3,FY3,FZ3,FX4,FY4,FZ4

```
For_Each
/Iteration_Parameter_Name= SIGNAL_INDEX
/Items= FX+FY+FZ
;

For_Each
/Iteration_Parameter_Name= NUMBER
/Items= 1+2+3+4
;

! Compute the mean value of the airborne phases.
Metric_Mean
/Signal_types= ANALOG
/Signal_Names= ::SIGNAL_INDEX&::NUMBER
/Signal_Folder= ORIGINAL
```

```
/Signal_Components= ALL_COMPONENTS
/Event_Sequence= LTO+RHS
/Exclude_Events=
/Metric_Name= _
/Apply_As_Suffix_To_Signal_Name= TRUE
/Generate_Mean_And_STDEV= TRUE
/Append_To_Existing_Values= FALSE
;

Multiply_Signals_By_Constant
/Signal_types= METRIC
/Signal_Names= ::SIGNAL_INDEX&::NUMBER&%%__%%MEAN
/Signal_Folder= PROCESSED
/Result_Names= OFFSET
/Result_Suffix=
/Signal_Components=
/Constant= -1
;

Set_Pipeline_Parameter_To_Data
/Parameter_Name= BASELINE
/Signal_types= METRIC
/Signal_Names= OFFSET
/Signal_Folder= PROCESSED
;

! **NOTE** The following command creates an ANALOG PROCESSED signal, not a
DERIVED PROCESSED signal.

Add_Constant_To_Signals
/Signal_types= ANALOG
/Signal_Names= ::SIGNAL_INDEX&::NUMBER
/Signal_Folder= ORIGINAL
/Result_Names=
/Result_Suffix=
/Signal_Components=
/Constant= ::BASELINE
;

End_For_Each
/Iteration_Parameter_Name= NUMBER
;

End_For_Each
/Iteration_Parameter_Name= SIGNAL_INDEX
;
```

Example 2: Using Evaluate_Expression to subtract mean values from several

frame ranges

Again we will see how to subtract from each analog signal its mean value computed from several frame ranges; this time using Evaluate_Expression. Again we consider a subject running on an instrumented treadmill. The mean value of each ANALOG signal is determined when the subject is airborne (e.g. between LTO and RHS) and a collection of these airborne phases are averaged. This overall mean value is to be subtracted from each ANALOG signal. The option to use Processed Analog signals for the Force Platform calculations should be selected. The FORCE_PLATFORM:ZERO parameter is set to (0,0) because it is to be ignored.

Given 12 ANALOG signals associated with a type 6 Force Platform:FX1,FY1,FZ1,FX2,FY2,FZ2,FX3,FY3,FZ3,FX4,FY4,FZ4

```
! Compute the mean values for all ANALOG signals.

Metric_Mean
/RESULT_METRIC_NAME=_BASE
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
! /RESULT_METRIC_FOLDER=PROCESSED
/SIGNAL_TYPES=ANALOG
! /SIGNAL_NAMES=
! /SIGNAL_FOLDER=ORIGINAL
! /SIGNAL_COMPONENTS=ALL_COMPONENTS
/EVENT_SEQUENCE=LOFF+RON
/EXCLUDE_EVENTS=
! /GENERATE_MEAN_AND_STDDEV=TRUE
! /APPEND_TO_EXISTING_VALUES=FALSE
;

! Setup a for each loop over the ANALOG signals

For_Each
/ITERATION_PARAMETER_NAME=SIGNAL_INDEX
/ITEMS=F1X1+F1Y1+F1Z1+F1X2+F1Y2+F1Z2+F1X3+F1Y3+F1Z3+F1X4+F1Y4+F1Z4
;

! Subtract the mean value from each signal. This step is a little quirky
because I chose to use the evaluate_expression to perform this task.
Unfortunately the (::) identified gets confused when four colons (e.g. ::::)
are used in sequence. This forced me to define Global Parameters. This isn't
actually a terrible thing because it was helpful when I was debugging the
pipeline.

Set_Pipeline_Parameter
/PARAMETER_NAME=TEMP
/PARAMETER_VALUE=ANALOG::ORIGINAL::&::SIGNAL_INDEX
;

Set_Pipeline_Parameter
```

```
/PARAMETER_NAME=BASE
/PARAMETER_VALUE=METRIC::PROCESSED::&::SIGNAL_INDEX&_BASE_MEAN
;

Set_Pipeline_Parameter
/PARAMETER_NAME=EXPRESSION
/PARAMETER_VALUE=::TEMP&-&::BASE
;

! Now we can evaluate the expression.

Evaluate_Expression
/EXPRESSION=::EXPRESSION
/RESULT_NAME=::SIGNAL_INDEX
/RESULT_TYPE=ANALOG
/RESULT_FOLDER=PROCESSED
;

End_For_Each
/ITERATION_PARAMETER_NAME=SIGNAL_INDEX
;
```

Example 3: Using mean values from another trial

This example demonstrates how to compute mean values for the force platform analog signals in one trial, and then use these values in other trials. For this example assume that:

- one of the files has TAG=walk.
- there are two AMTI force platforms
- the force platforms are unloaded for the first 50 frames

```
! Create events to identify Frame 1 and Frame 50 in the file with TAG=walk

Select_Active_File
/FILE_NAME=walk
! /QUERY=
;

Explicit
/EVENT_NAME=FRAME1
/FRAME=1
;

Explicit
/EVENT_NAME=FRAME50
/FRAME=50
;

! Compute the mean value of the force platform ANALOG signals
```

```
Metric_Mean
/RESULT_METRIC_NAME=_
/APPLY_AS_SUFFIX_TO_SIGNAL_NAME=TRUE
! /RESULT_METRIC_FOLDER=PROCESSED
/SIGNAL_TYPES=ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG+ANALOG
+ANALOG+ANALOG+ANALOG
/SIGNAL_NAMES=F1X+F1Y+F1Z+F2X+F2Y+F2Z+M1X+M1Y+M1Z+M2X+M2Y+M2Z
/SIGNAL_FOLDER=ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINA
L+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL+ORIGINAL
/SIGNAL_COMPONENTS=X
/EVENT_SEQUENCE=FRAME1+FRAME50
/EXCLUDE_EVENTS=
! /GENERATE_MEAN_AND_STDDEV=TRUE
! /APPEND_TO_EXISTING_VALUES=FALSE
;

! Construct a For_Each loop to process each ANALOG signal independently

For_Each
/ITERATION_PARAMETER_NAME=SIG
/ITEMS=F1X+F1Y+F1Z+M1X+M1Y+M1Z+F2X+F2Y+F2Z+M2X+M2Y+M2Z
;

! Set the active folder to GLOBAL and create a pipeline parameter for
appropriate MEAN signal

Select_Active_File
/FILE_NAME=GLOBAL
! /QUERY=
;

Set_Pipeline_Parameter_To_Data_Value
/PARAMETER_NAME=TEMP
/SIGNAL_TYPES=METRIC
/SIGNAL_NAMES=::SIG&%%__%%MEAN
/SIGNAL_FOLDER=PROCESSED
! /INDEX=
;

! Define a pipeline parameter to create the expression that we will evaluate
in the next step. This is a little awkward but necessary because if we tried
to use the evaluation expression with pipeline parameters it get's confused
(maybe someday we will find a solution to this problem).

Set_Pipeline_Parameter
/PARAMETER_NAME=EQUATION
/PARAMETER_VALUE=ANALOG::ORIGINAL::&::SIG&-&::TEMP
! /PARAMETER_VALUE_SEARCH_FOR=
! /PARAMETER_VALUE_REPLACE_WITH=
! /PARAMETER_VALUE_APPEND=
```

```
;

! Make all of the files in the Workspace active and subtract the GLOBAL mean
values from the signals.

Select_Active_File
/FILE_NAME=ALL_FILES
! /QUERY=
;

Evaluate_Expression
/EXPRESSION=::EQUATION
/RESULT_NAME=::SIG
/RESULT_TYPE=ANALOG
! /RESULT_FOLDER=PROCESSED
;

! For example, the first time through the loop /EXPRESSION=
ANALOG::ORIGINAL::F1X - ::SIG%%__%%MEAN

End_For_Each
/ITERATION_PARAMETER_NAME=SIG
;
```