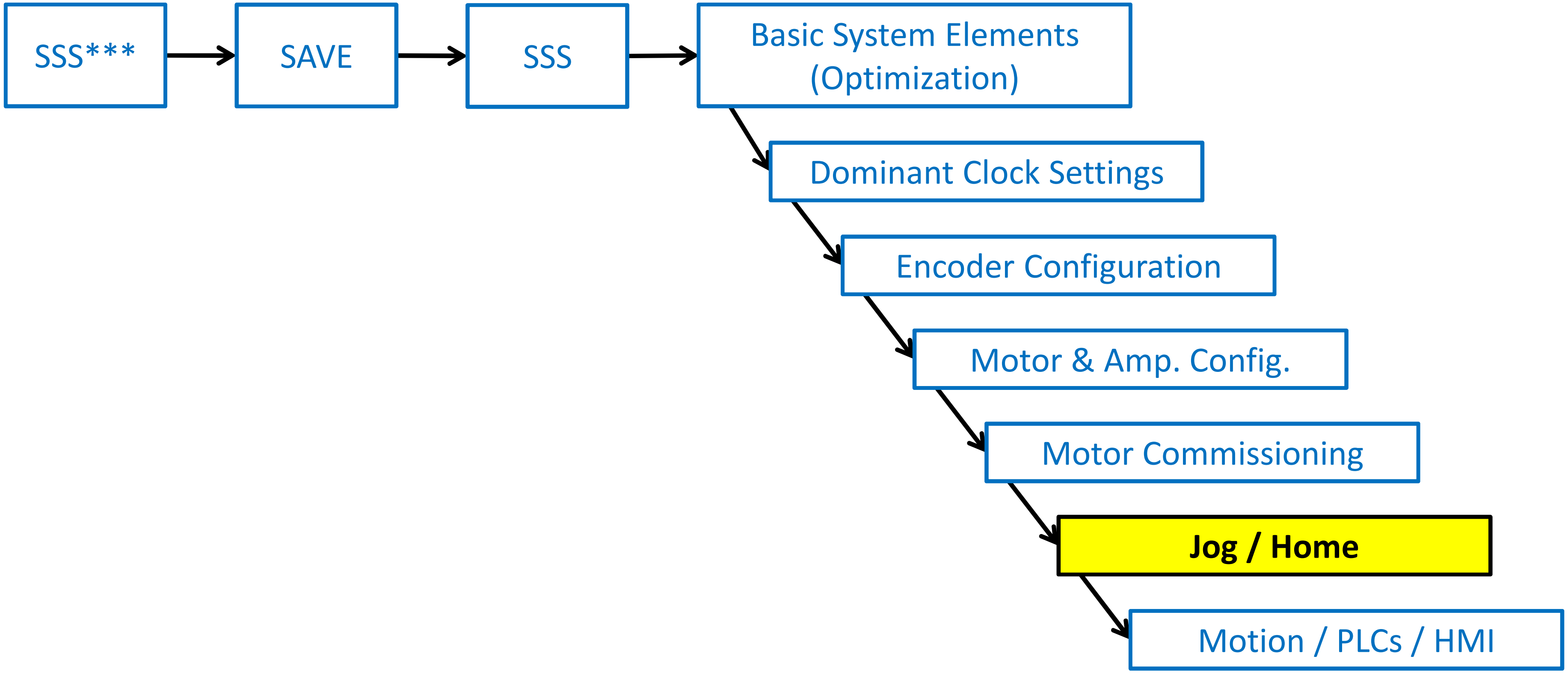
OMRON | DELTA TAU

Motor Jogging

System Configuration



SSS***

SAVE

SSS

Basic System Elements (Optimization)

Dominant Clock Settings

Encoder Configuration

Motor & Amp. Config.

Motor Commissioning

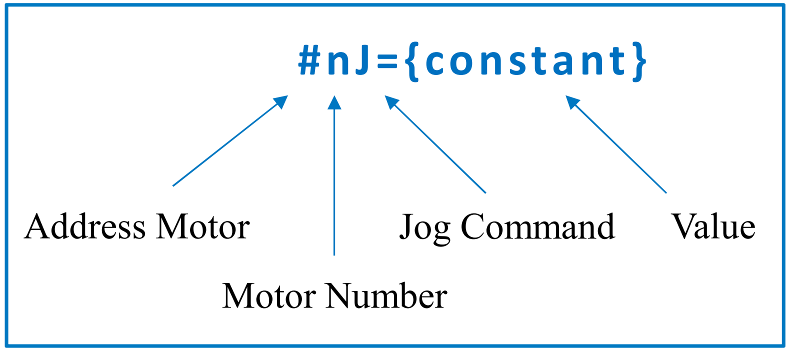
Jog / Home

Motion / PLCs / HMI

What is Jogging?

- Jogging is the simplest form of closed-loop motor move

- Typically, jogging acts on a single motor
  - Not associated with coordinate systems or motion programs
  - Can be performed for multiple motors at once



#nJ={constant}

- Jogging is ideal for point to point moves (positioning) or indefinite movement in one direction

- Jogging can be performed online or in-programs (with some exceptions)

Basic Online Jog Commands

COMMAND	DESCRIPTION	EXAMPLES
J/	Decelerate to a stop / put motor in closed loop	**#1J/**
J={constant}	Jog to a specified position	**#1J=10**
J^{constant}	Jog to a specified position relative to the motor’s actual position	**#1J^-15**
J+	Jog positive indefinitely	**#1J+**
J–	Jog negative indefinitely	**#1J-**
J=	Jog to “pre-jog” position (most recently programmed position)	**#1J=**
J=={constant}	Jog to a specified position, then set that position as the “pre-jog” position	**#1J==25**
J=*	Jog to **Motor[*x*].ProgJogPos**	**#1J=***

#1..4J/
#1,2,4J/
#1,4J+

Note

J is the short version with **Sys.NoShortCmds** at default of 0. If it is set to 1, must use the long version **Jog**

Note

Can change jog parameters at any time. New values do not take effect until the next jog command

Basic Program Jog Commands

COMMAND	DESCRIPTION	EXAMPLES
Jog/	Decelerate to a stop / put motor in closed loop	**Jog/1**
Jog={data}	Jog to a specified position	**Jog1=10**
Jog^{data}	Jog to a specified position relative to the motor's actual position	**Jog1^-15**
Jog+	Jog positive indefinitely	**Jog+1**
Jog–	Jog negative indefinitely	**Jog–1**
Jogret	Jog to "pre-jog" position (most recently programmed position)	**Jogret1**
Jogret={data}	Jog to a specified position, then set that position as the "pre-jog" position	**Jogret1=25**
Jogret=*	Jog to **Motor[*x*].ProgJogPos**	**Jogret1=***

Jog/1..4
Jog/1,2,4
Jog+1,4

Note

Short version **J** is not used in programs regardless of the **Sys.NoShortCmds** setting

Note

Can change jog parameters at any time. New values do not take effect until the next jog command

Jog Move Profile

- **A jog move profile is governed by 3 (saved) elements:**
  - Motor[].JogSpeed (always positive)
  - Motor[].JogTa
  - Motor[].JogTs

- **Acceleration/deceleration can be specified by either time OR rate**

- **Units**
  - JogSpeed [**motor units/msec**]
  - JogTa
    - [**msec**] if > 0, acceleration time
    - [**msec^2/motor unit**] if < 0, inverse acceleration rate
  - JogTs
    - [**msec**] if > 0, s-curve time
    - [**msec^3/motor unit**] if < 0, inverse jerk rate

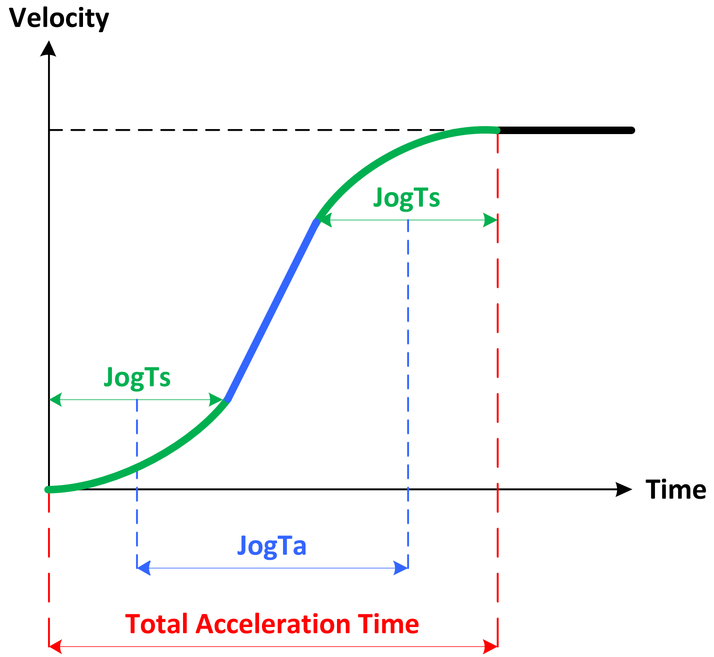
Note

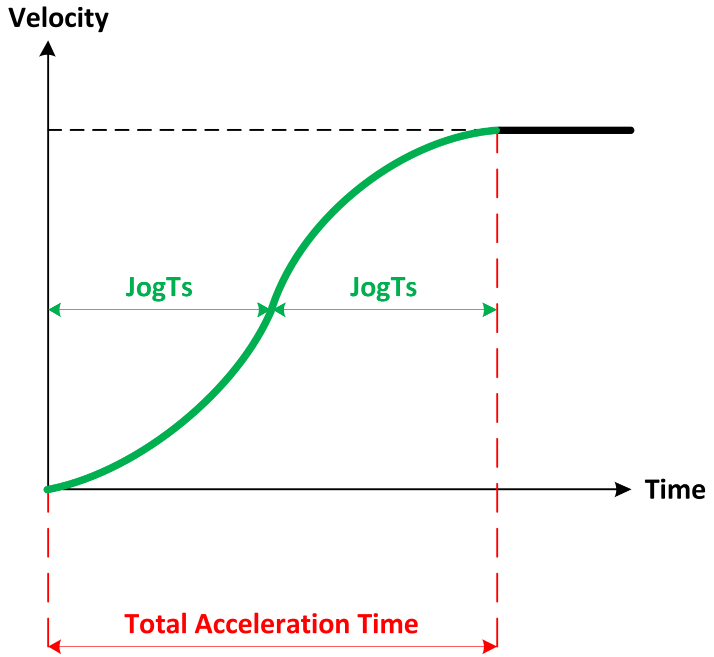
If **JogTa** is less than 0, **JogTs** must also be set less than 0, so both specify rate

Time Acceleration Settings

- JogTa and JogTs are > 0
- Rate of acceleration will be different for various speeds
- Can be optimized for pre-defined move sizes
- Accel and jerk limits are not known/critical
  - Set by “look and feel”

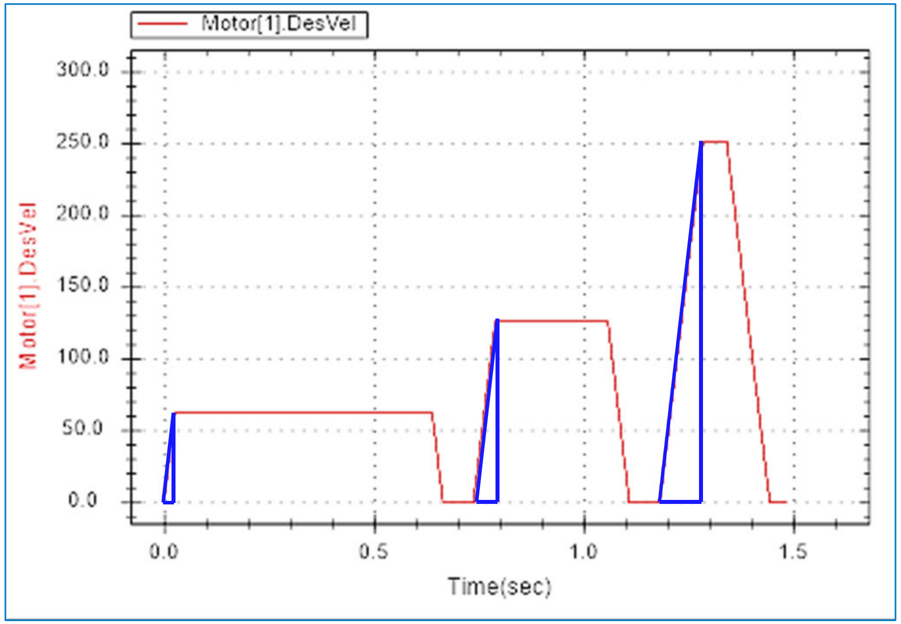
- Time accel. rules
  - If JogTa >= JogTs
    - Total Accel. Time = JogTa + (½ JogTs + ½ JogTs)
      = JogTa + JogTs
  - If JogTa < JogTs
    - Total Accel. Time (extended) to = 2 * JogTs





Rate Acceleration Settings

- JogTa and JogTs are < 0
- Provides more seamless blending into newly commanded moves
- Rate of acceleration <u>same</u> for various speeds
  - Optimized move times
- Known, or desired acceleration and jerk limits



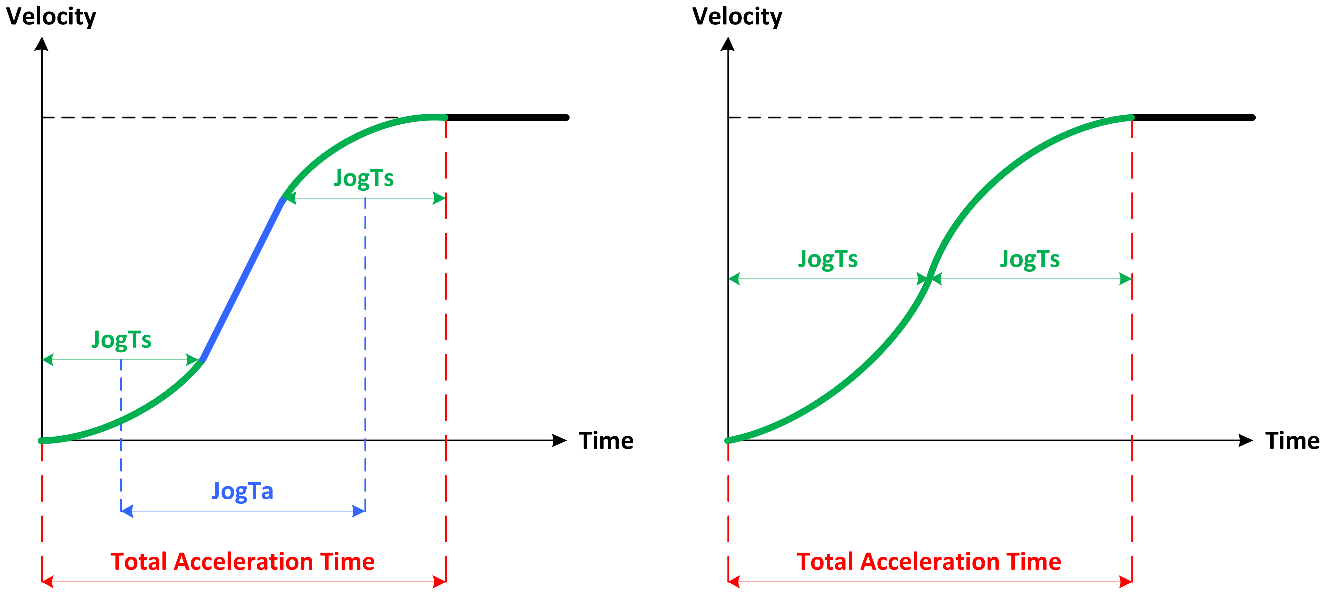
- E.g. For a linear motor set up in mm:
  - JogTa = -1000 / (**Gs** * 9.806)
  - JogTs = -1000000 / (**GsPerS** * 9.806)
  - Where:
    - **Gs** is desired acceleration in G gravity (or m/s^2)
    - **GsPerS** is desired jerk in Gs per second (or m/s^3)
- E.g. For a rotary motor set up in degrees:
  - JogTa = -1000 / (**DegPerS2**)
  - JogTs = -1000000 / (**DegPerS3**)
  - Where:
    - **DegPerS2** is desired acceleration rate in degrees per s^2
    - **DegPerS3** is desired jerk rate in degrees per s^3

Jogging Quiz

- **For these time acceleration settings**
  - JogTa = 150, JogTs = 0, JogSpeed = 0.100
  - JogTa = 50, JogTs = 75, JogSpeed = 0.200
  - JogTa = 125, JogTs = 25, JogSpeed = 0.300

    - What is the total acceleration time in each case?
    - Does JogSpeed matter?

- If JogTa >= JogTs
  - Total Accel. Time = JogTa + (½ JogTs + ½ JogTs)
    = JogTa + JogTs
- If JogTa < JogTs
  - Total Accel. Time (extended) to = 2 * JogTs



Jogging Quiz

- **For a linear actuator set up in motor units of millimeters**
  - If JogTa = - 67.985588 [ms^2/mm]
    - What is the maximum acceleration in Gs?
  - If JogTs =-463.53809 [ms^3/mm]
    - What is the maximum jerk in Gs/s?
  - Note:
    - 1G = 9.806 m/s^2

JogTa = -1000 / (Gs * 9.806)

JogTs = -1000000 / (GsPerS * 9.806)

Note

In a track test, the Ferrari LaFerrari launches off the line with a jerk rate of about 700 Gs/s

Plot Tool

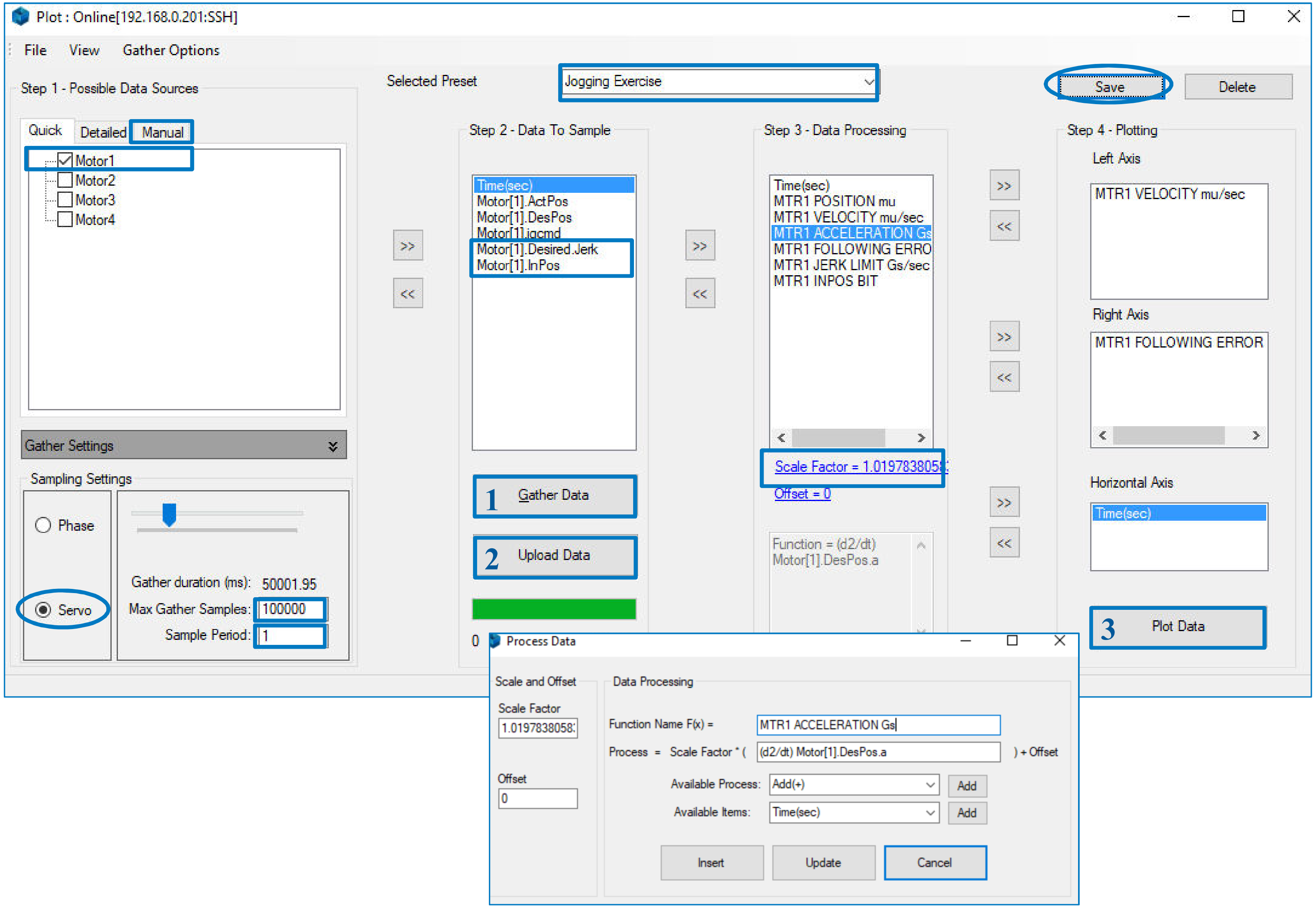
Rename (optional)

- Cmd position [mu]
- Following error [mu]
- Cmd velocity [mu/s]
- Cmd Acceleration [Gs]

- Motor[1].Desired.Jerk
  - Jerk Limit [Gs/s]
- Motor[].InPos

Scale Factors for #1

- Accel in Gs:
  - = 1 / (9.806 * 1000)
- Jerk in Gs/s:
  - = 6 * 1000000 / 9.806



Plot : Online[192.168.0.201:SSH]

File View Gather Options

Selected Preset: Jogging Exercise

Step 1 - Possible Data Sources (Quick / Detailed / Manual)

☑ Motor1
□ Motor2
□ Motor3
□ Motor4

Gather Settings

Sampling Settings: ○ Phase ● Servo

Gather duration (ms): 50001.95
Max Gather Samples: 100000
Sample Period: 1

Step 2 - Data To Sample

Time(sec)
Motor[1].ActPos
Motor[1].DesPos
Motor[1].jogcmd
Motor[1].Desired.Jerk
Motor[1].InPos

1 Gather Data
2 Upload Data

0

Step 3 - Data Processing

Time(sec)
MTR1 POSITION mu
MTR1 VELOCITY mu/sec
MTR1 ACCELERATION Gs
MTR1 FOLLOWING ERRO
MTR1 JERK LIMIT Gs/sec
MTR1 INPOS BIT

Scale Factor = 1.0197838058
Offset = 0

Function = (d2/dt) Motor[1].DesPos.a

Step 4 - Plotting

Left Axis: MTR1 VELOCITY mu/sec
Right Axis: MTR1 FOLLOWING ERROR
Horizontal Axis: Time(sec)

3 Plot Data

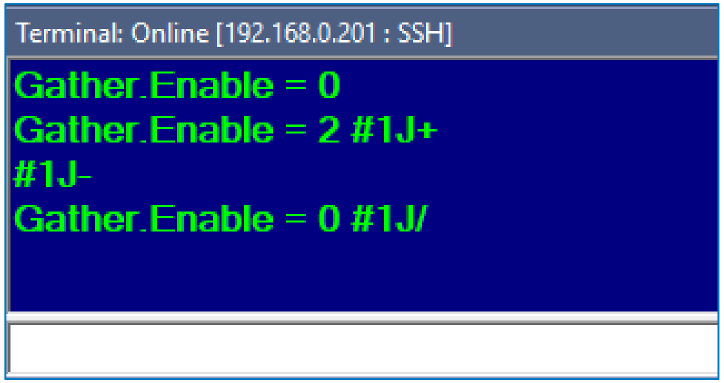
Process Data

Scale and Offset
Scale Factor: 1.0197838058
Offset: 0

Data Processing
Function Name F(x) = MTR1 ACCELERATION Gs
Process = Scale Factor * ((d2/dt) Motor[1].DesPos.a) + Offset
Available Process: Add(+)
Available Items: Time(sec)

Jogging Exercise

- **Motor #1 is set up in motor units of millimeters**
  - What should JogSpeed be set for a setting of 200 mm/s?
  - Find the shortest acceleration time using JogTa and JogTs so that for any jog move the following error does not exceed 50 µm (0.050 mm)
    - Following error limit is set to 500 µm (Motor[1].FatalFeLimit = 0.500)
    - Use the plot tool to display the following error during move(s)
  - Test for jog direction reversal at top speed!
  - What is the observed maximum commanded acceleration in Gs? – plot



```
Gather.Enable = 0
Gather.Enable = 2 #1J+
#1J-
Gather.Enable = 0 #1J/
```

Abort Deceleration

Abort Deceleration

- **Uses Motor[].AbortTa, and Motor[].AbortTs**

- **Automatically triggered**
  - Exceeds a hardware over-travel limit
  - Exceeds a software over-travel limit
  - Coordinate system ABORT commanded
  - Coordinate system run-time error

- **Same "deceleration" profile and rules as jogging**

- **Axis/machine specific**
  - Typically, configured to allow stopping from top speed without:
    - Violating the hardware limits
    - Excessive jerk or vibration
    - Exceeding the following error limit

- **Units (same rules as JogTa, and JogTs)**
  - AbortTa
    - [**msec**] if > 0, acceleration time
    - [**msec^2/motor unit**] if < 0, inverse acceleration rate
  - AbortTs
    - [**msec**] if > 0, s-curve time
    - [**msec^3/motor unit**] if < 0, inverse jerk rate

Note

If **AbortTs** is less than 0, **AbortTa** must also be set less than 0, so both specify rate

Note

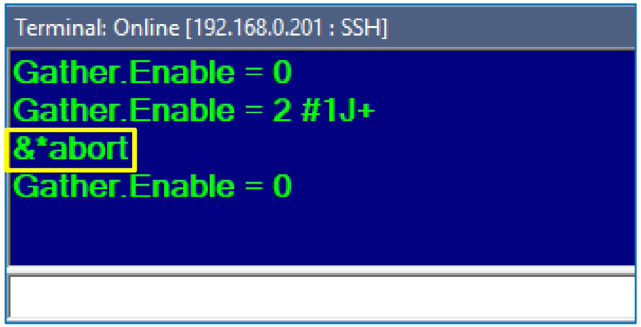
Certain axes may be oversized and physically incapable of stopping within the hardware limits at top speeds. In these cases, software limits (MinPos and MaxPos) are configured with an extended abort deceleration time (or lower rate)

Abort Deceleration Exercise

- **Your linear motor (#1) has a maximum speed of 250 mm/s**
  - Find the shortest decel. time using AbortTa or AbortTs so that aborting from top speed the following error does not exceed 100 µm (0.100 mm)
    - Following error limit is set to 500 µm (Motor[1].FatalFeLimit = 0.500)
    - Abort using one of the hardware over-travel limits
    - Use the plot tool to display the following error during deceleration
  - Test in both directions!
  - What is the observed maximum commanded acceleration in Gs? – plot

Note

Can you use the ABORT command instead of the hardware over-travel limit to abort



Terminal: Online [192.168.0.201 : SSH]

```
Gather.Enable = 0
Gather.Enable = 2 #1J+
&*abort
Gather.Enable = 0
```