Trio Motion Technology

Trio Packaging Series

First Edition • 2023

Revision: 1.1

Trio Programming Guides are designed to aid learning of the TrioBASIC language through description and examples. Each one will cover a particular topic and discuss which commands and parameters in the TrioBASIC are required to complete the task.

A general understanding of TrioBASIC is required and it is recommended to attend an introduction to TrioBASIC training course. The programming guides are not a replacement for the TrioBASIC help files which can be found in *Motion* Perfect as well as the manual which covers each command and parameter in more detail and should be referenced when required.

Any examples given in the programming guide will work and have been tested on an isolated controller. If you choose to use these examples on a machine please take care that it will not cause damage or injury and that they are correctly included in the project changing parameters and values where required.

UK | USA | CHINA | INDIA

www.triomotion.com

SAFETY WARNING

During the installation or use of a control system, users of Trio products must ensure there is no possibility of injury to any person, or damage to machinery.

Control systems, especially during installation, can malfunction or behave unexpectedly.

Bearing this in mind, users must ensure that even in the event of a malfunction or unexpected behaviour the safety of an operator or programmer is never compromised.

This document uses the following icons for your reference:

Information that relates to safety issues and critical software information.

Information to highlight key features or methods.

Useful tips and techniques.

Example programs.

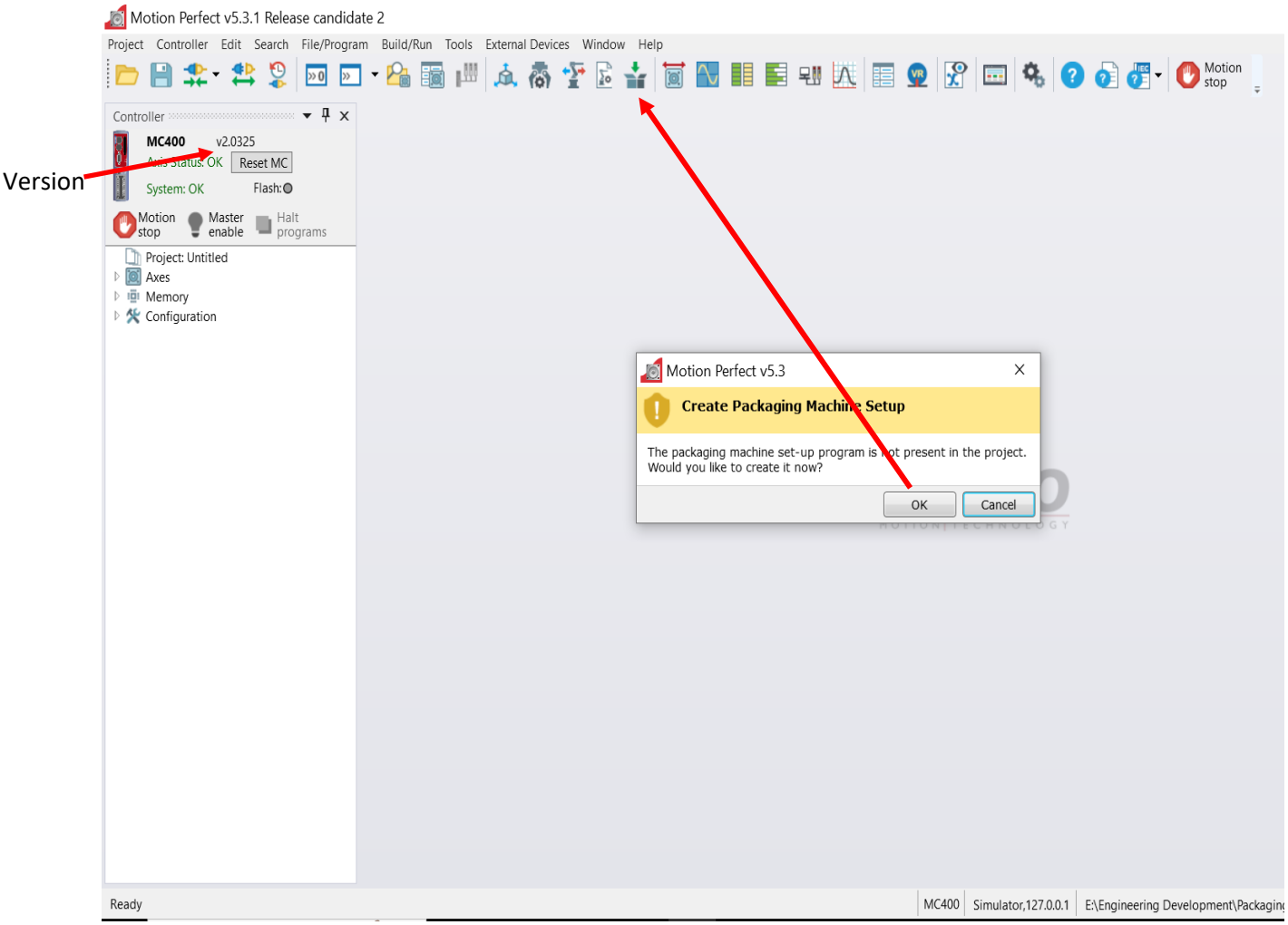
Contents

1. Packaging Machine Setup

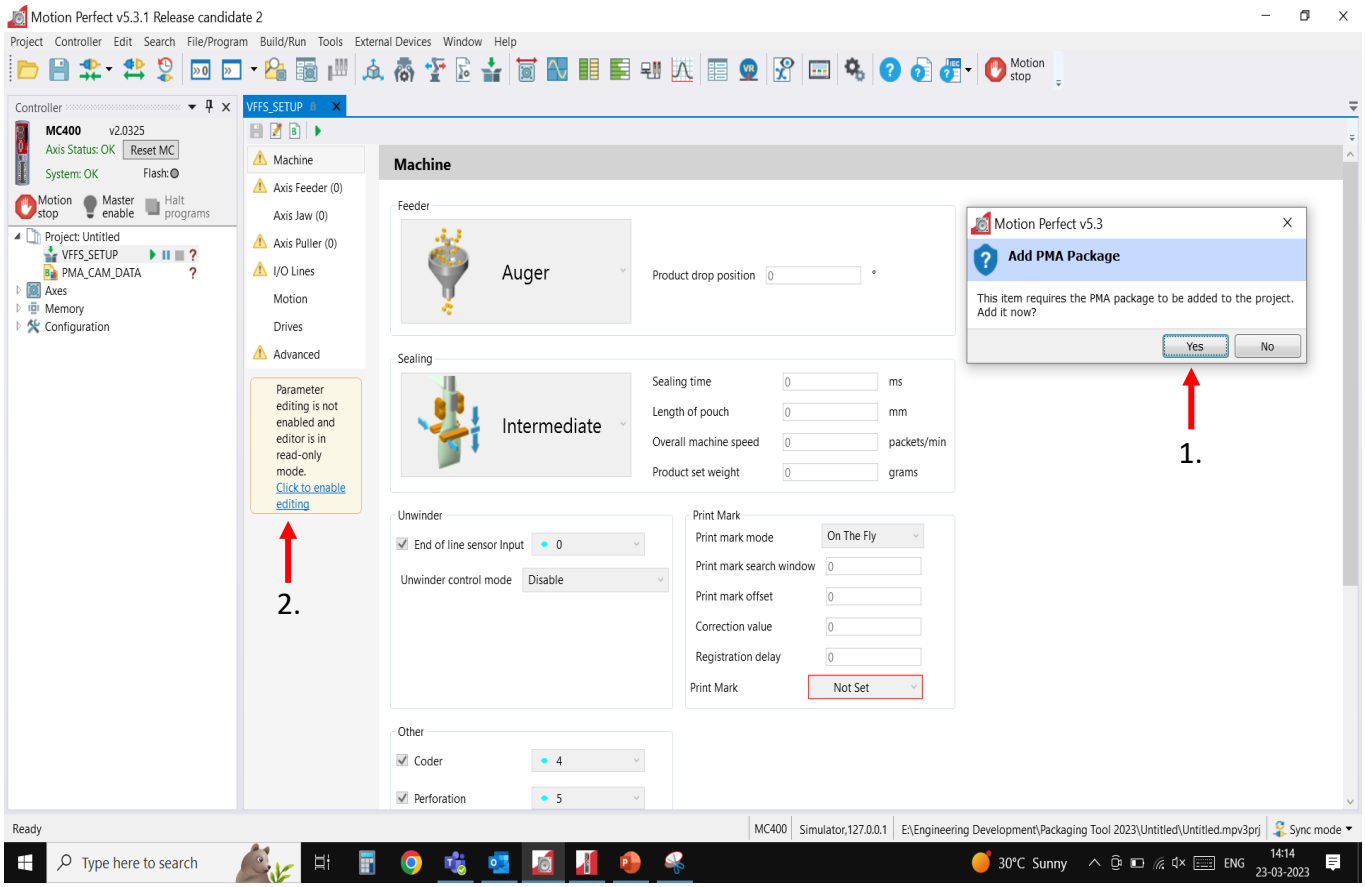
1.1. Definition

The Packaging Machine Setup is Trio software package that offers complete control and management of a Vertical Form Fill Seal machine. It is aimed at reducing programming time and optimizing productivity.

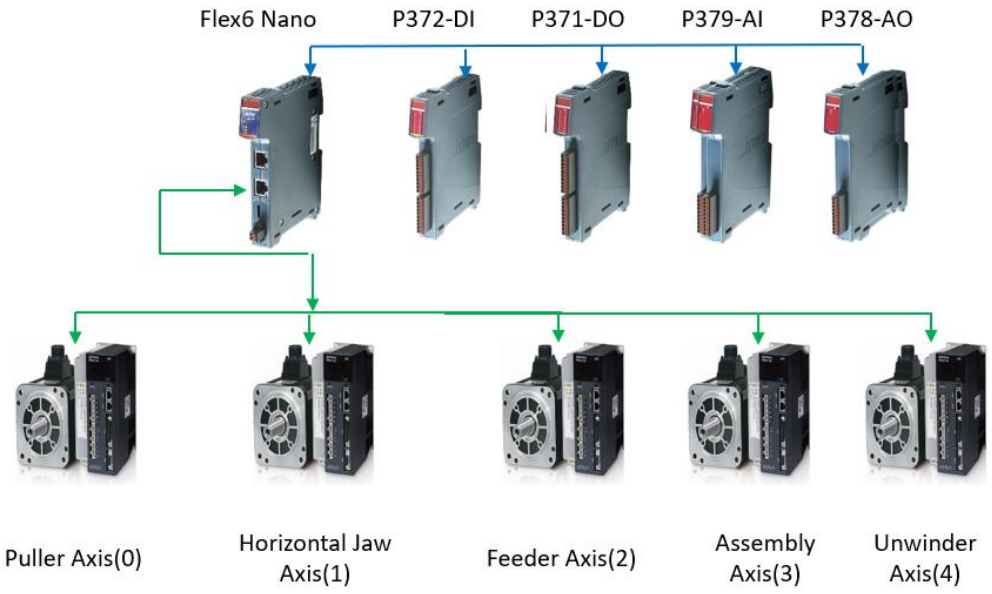
It requires Motion Perfect 5.3.2 or higher software, controller firmware version must be higher than 2.0325, an ETHERCAT based Controller, Inputs and Outputs as well as Servo Drives & motors.



Click on this icon as shown in above figure. And click on **"OK"** button.



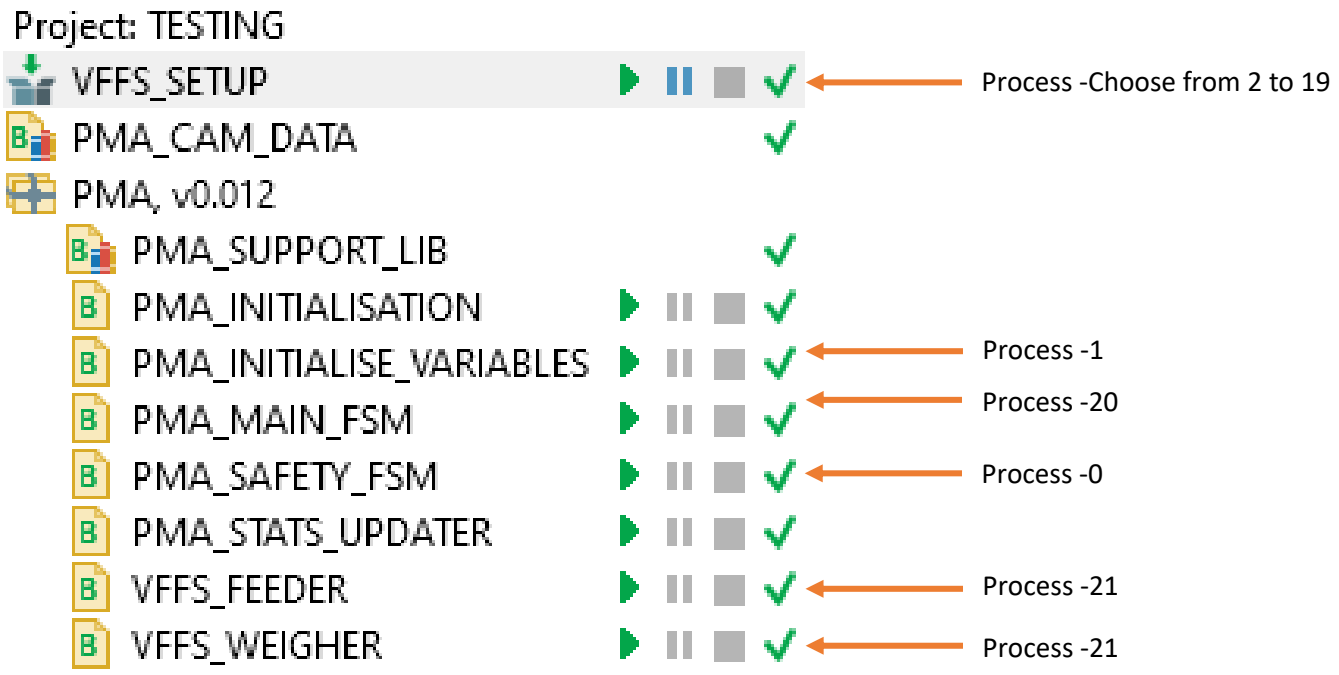
A pop-up message will appear asking user to confirm addition of PMA package, please click **"Yes"** first and then press **"click to enable editing**" to configure the machine setup for the packaging application.



Above architecture is shown as example.

1.2. Package Library

This package occupies 5 processes of the controller. VFFS_SETUP task can be run on the user's process either at power on or from any other program.



PMA_MAIN_FSM	Runs Continuously
PMA_SAFETY_FSM	
PMA_STATS_UPDATER	
VFFS_WEIGHER	If Weigher is selected as Feeder type
VFFS_FEEDER	If Auger or Cup filler selected as Feeder type
PMA_CAM_DATA	Basic Library
PM_SUPPORT LIB	
PMA_INITIALISATION	Basic Initialisation
PMA_INTIALISE_VARIABLES	

Currently, Process numbers related to this package are fixed.

1.3. VFFS Setup.Machine

Machine setting is for configuring machine mechanics. This is main page for setting up machine type, parts of machine, peripherals etc. This package scalable up to 5 axis depending upon the type of motion cycle chosen by the user namely,

1. Feeder (Auger /Cup)
2. Jaw (Horizontal Jaw)
3. Puller
4. Assembly (UP /Down Axis) –Only in Continuous Motion Cycle
5. Unwinder

Users can configure these axis as per the machine requirements.

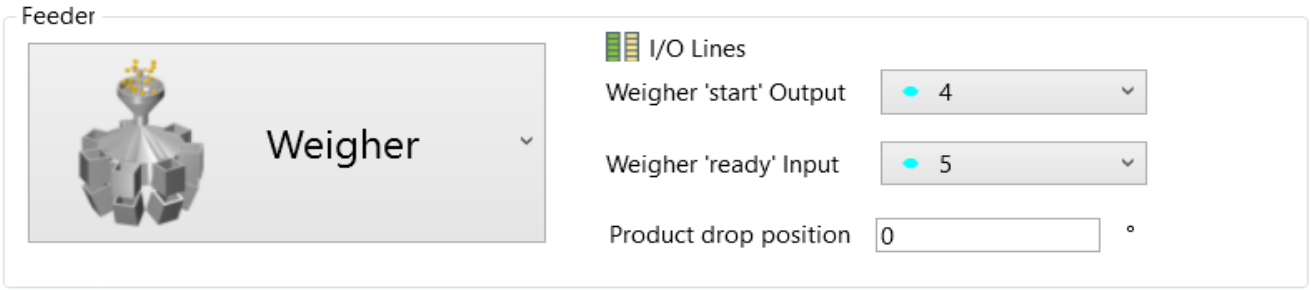
It supports three types of feeder systems as shown in the figure below.

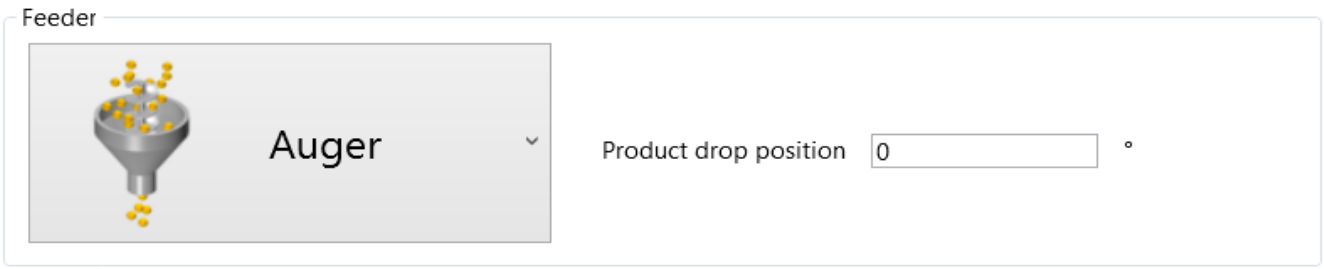
Multihead Weigher

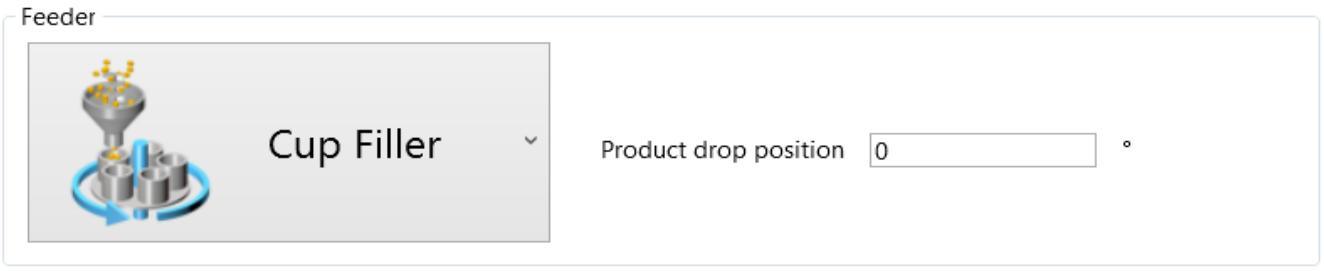
Auger

Cup Filler

If type of feeder is weigher system, configure the weigher ready input received from multihead weigher. Also for all feeder type either set Product drop position or product delay or both as required.

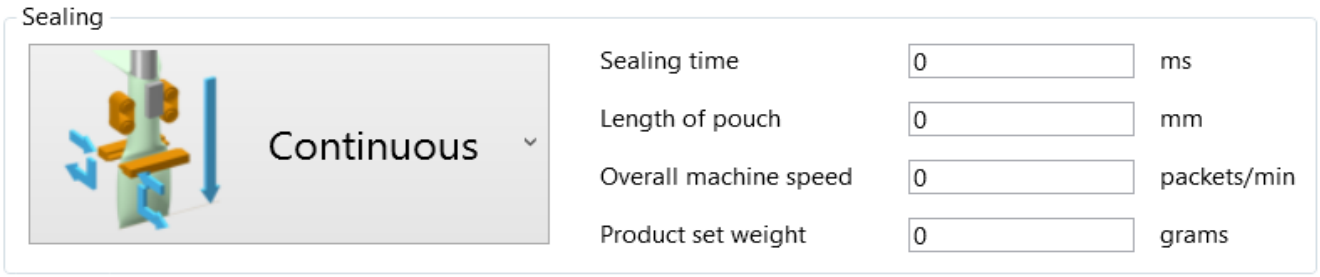


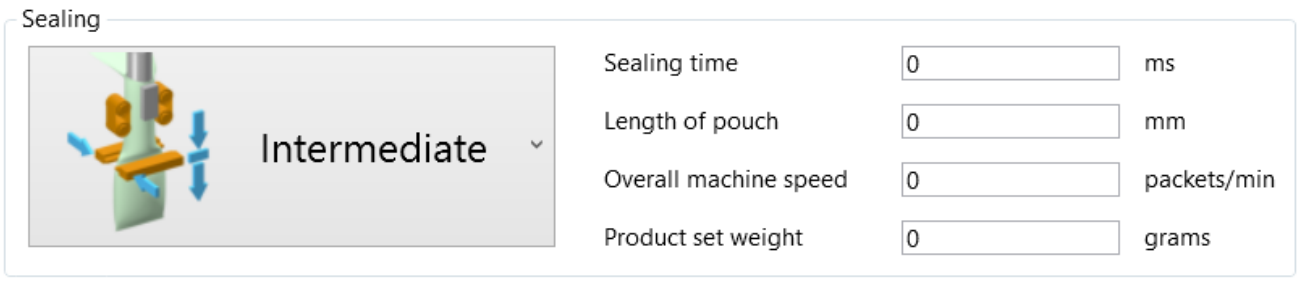




1.3.1 Sealing

User can select **“Continuous”** or **“Intermediate”** sealing type. Select the Machine motion cycle either continuous or intermittent.





1.3.1.1. Parameters

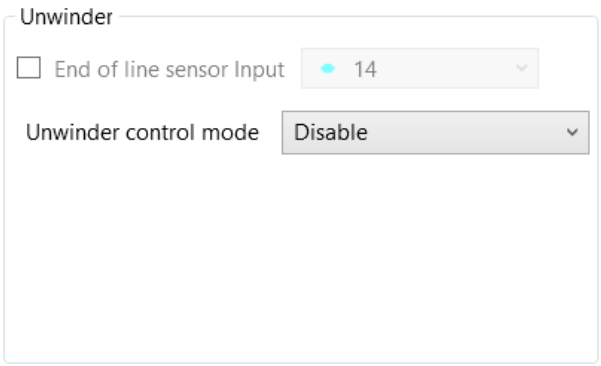
Sealing time	Sealing time in mSec required to seal the Pouch horizontally.
Length of Pouch	Pouch length in mm

Overall machine speed	Machine speed in terms of packets/min
Product Set weight	This parameter is applicable only in Auger Feeder. set required weight of the material in terms of grams.

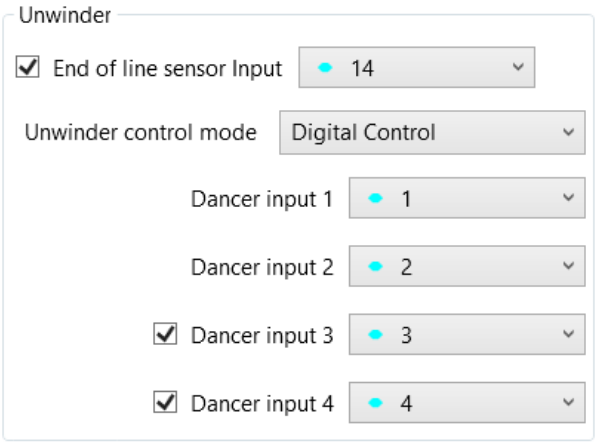
1.3.2. Unwinder

This is optional , where user can enable or disable end of line (film) input as well as Unwinder control mode. There are three different types of unwinder control available.

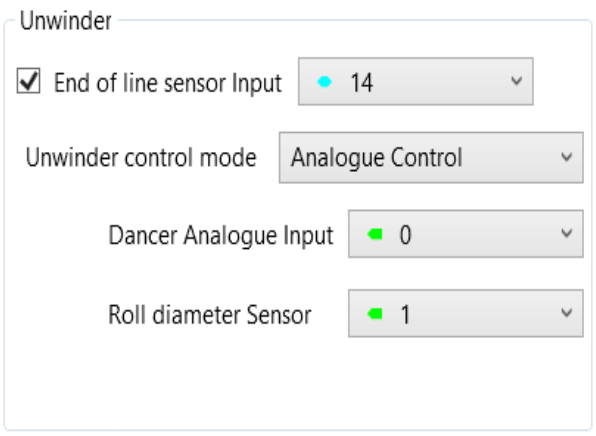
1.3.2.1. Modes



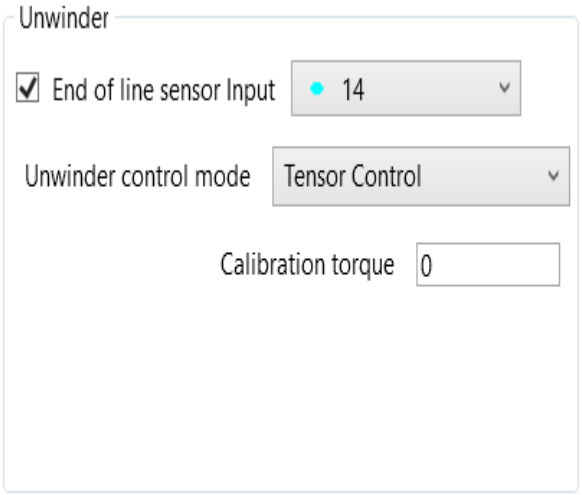
The following figure shows unwinder control mode is disable and in this case, there will not be an option to configure Unwinder as an Axis. End of line sensor input is configurable. If it is selected, in auto mode, this sensor must be ON else an alarm “End of line” will be generated and machine will be stopped.



This figure shows, End of line is enabled, and Digital control mode is chosen for Unwinder. It also gives selection of 2 to 4 Digital inputs for varying the unwinder speed based on the dancer position sensed.

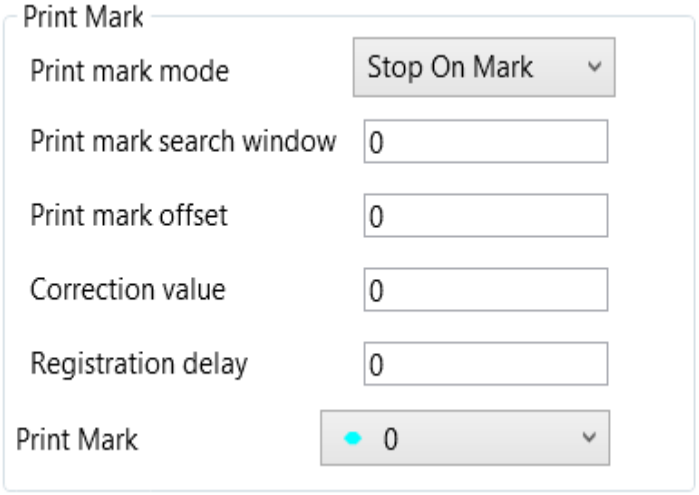


This is analogue control mode in which user has to connect dancer analogue and roll diameter inputs.



This shows tension control of Unwinder Servo in which user must set calibrated torque in terms of percentage.

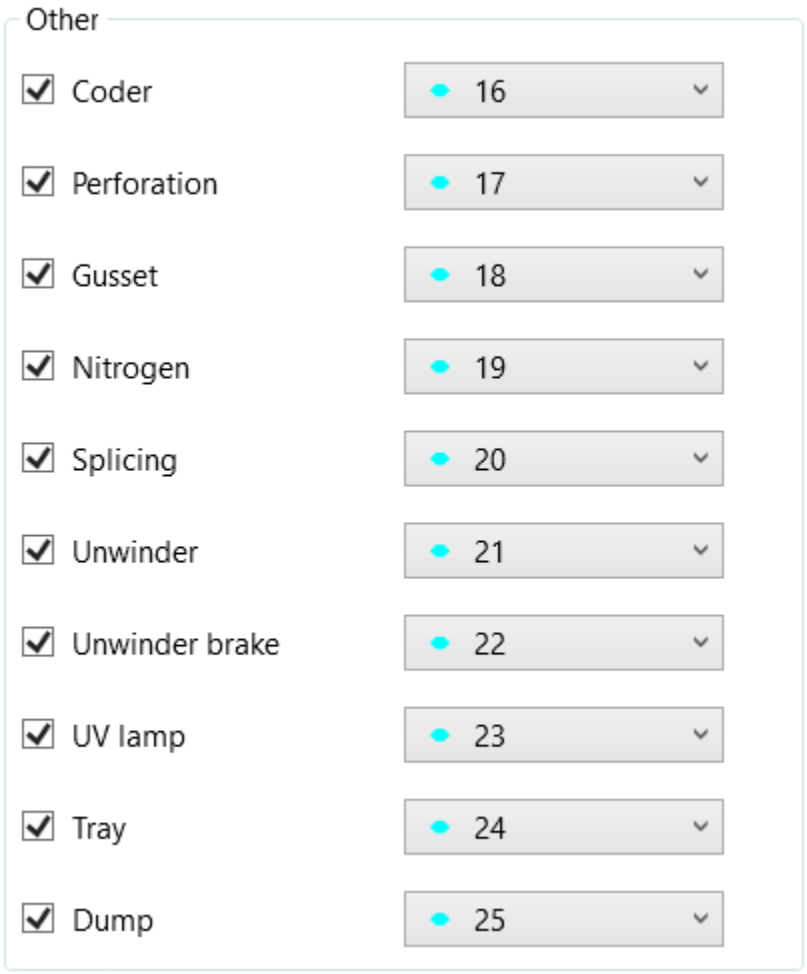
1.3.3. Print Mark



1.3.3.1. Parameters

Print mark mode	❖ Stop on Mark--- Puller stops when Print Mark event is registered. ❖ On the Fly ------- Pouch length adjusted based on mark actual position while pouch length puller servo is in motion.
Print mark search window	specifies the region in which controller must look for print mark event. It will be in mm.
Print mark offset	is related to homing of Puller servo. Its value must be greater than or equal to 0 mm. If set more than 0, the puller servo moves to specified mm after the print mark event is registered.
Correction Value	Specifies the minimum and maximum correction of pouch length in terms of mm.
Registration Delay	It determines registration delay in mSec.
Print Mark	Please assign appropriate digital input number for registration.

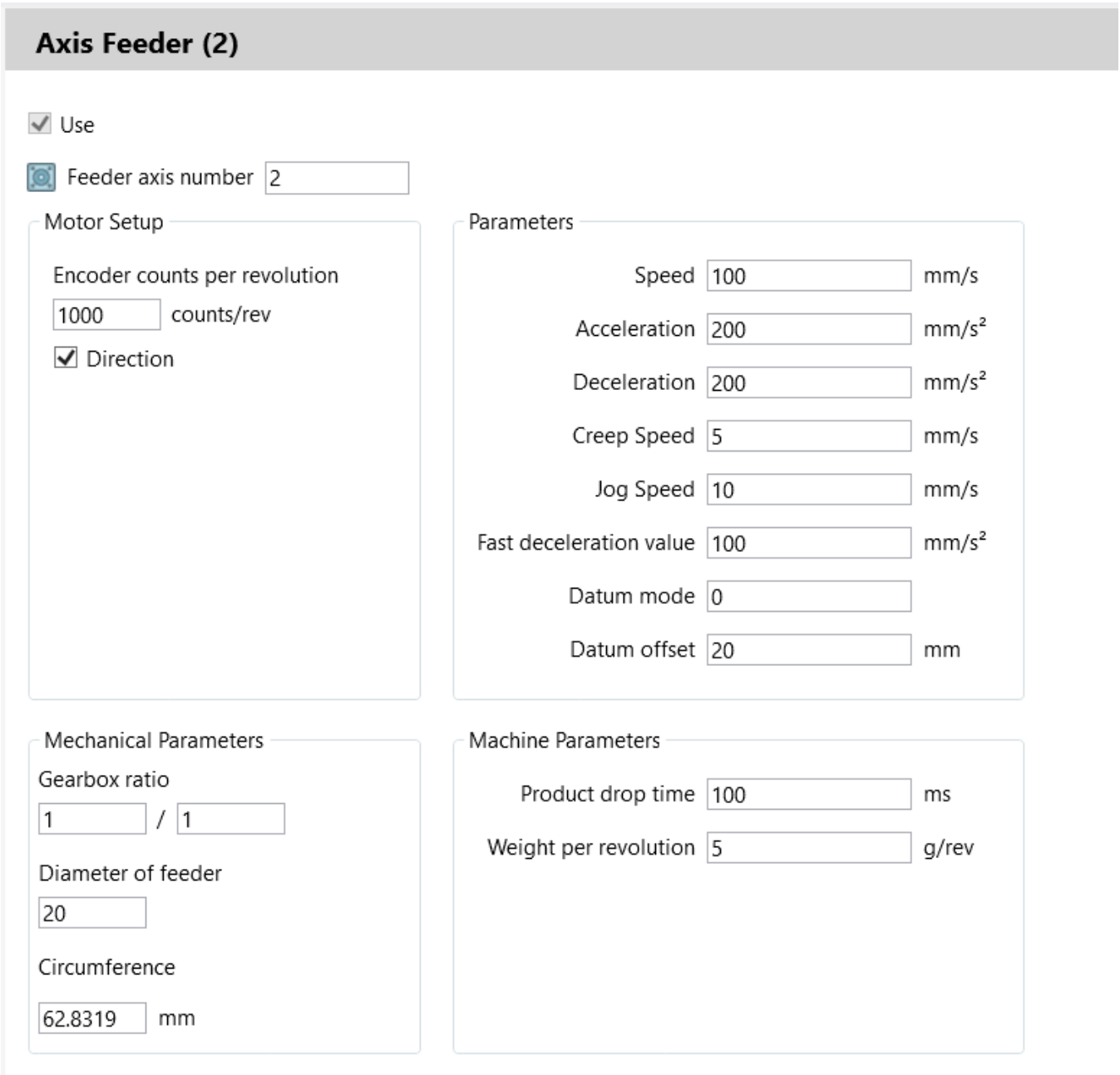
1.3.4. Other



These are optional outputs, if selected please assign unique output number, the template will not compile successfully if user is not assigning a number or assigning duplicate number.

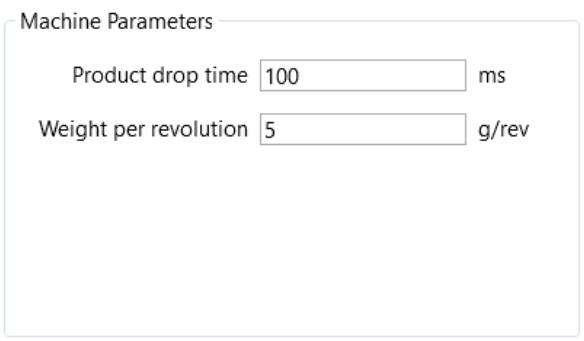
1.4. Axis Configuration

1.4.1. Axis Feeder



Use	If not selected, then this axis will not be active.
Feeder axis number 2	Please assign unique axis number.

Encoder counts per revolution 1000 counts/rev	Please enter servo encoder counts per revolution.
☐ Direction	This parameter is used to change the direction of the motor from the controller. If selected, then direction of the motor will be reversed in case of forward command is given from controller and vice versa. If not selected, then direction of the motor will be same as per the command given by the controller.
Parameters Speed 100 mm/s Acceleration 200 mm/s² Deceleration 200 mm/s² Creep Speed 5 mm/s Jog Speed 10 mm/s Fast deceleration value 100 mm/s² Datum mode 0 Datum offset 20 mm	These parameters are axis specific. And are useful during Manual and Homing operations related to specific axis.
Mechanical Parameters Gearbox ratio 1 / 1 Diameter of feeder 20 Circumference 62.8319 mm	Must enter exact gearbox ratio or pulley ratio which is connected between servo motor shaft and the load. Diameter and circumference are the optional parameters.



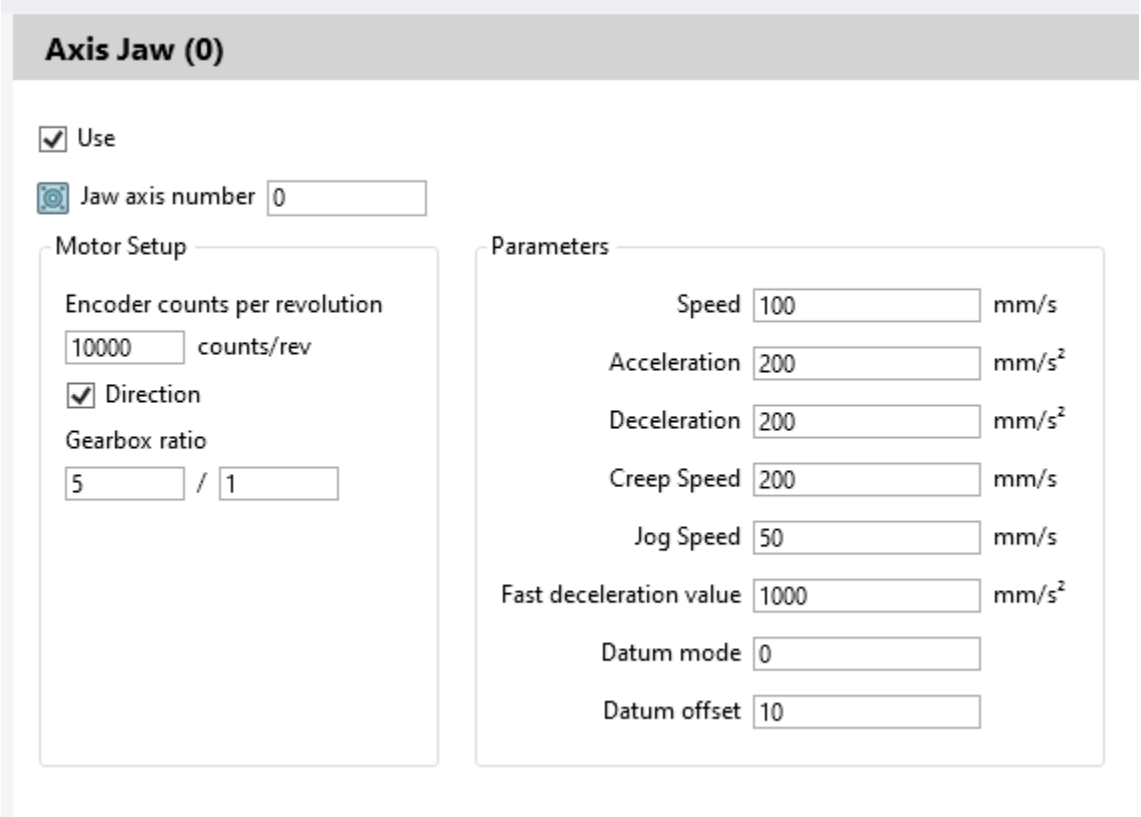
Product Drop Time: -

1) In case of Weigher machine, please add material travel delay to adjust sealing position.
2) In case of Auger feeder, please add Auger Start degree w.r.t master axis so that material does not comes in sealing position.
3) In case of cup filler, this will be the delay after detecting the "cup_home_input" to start the cycle.

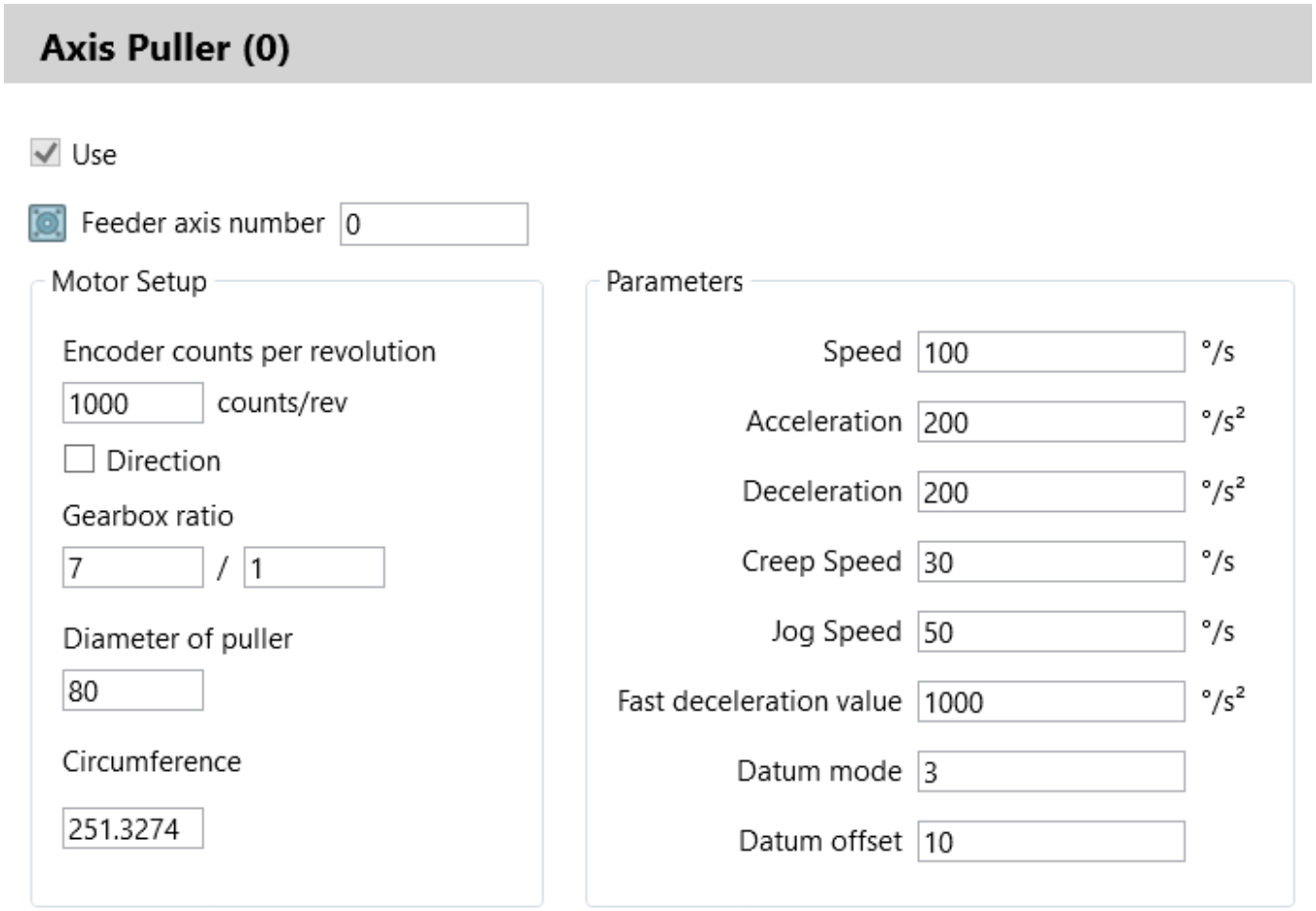
Weight per revolution: -

This parameter is useful in Auger Feeder machine only, where user has to enter average weight in terms of grams obtain after auger feeder is moved one revolution on the load side.

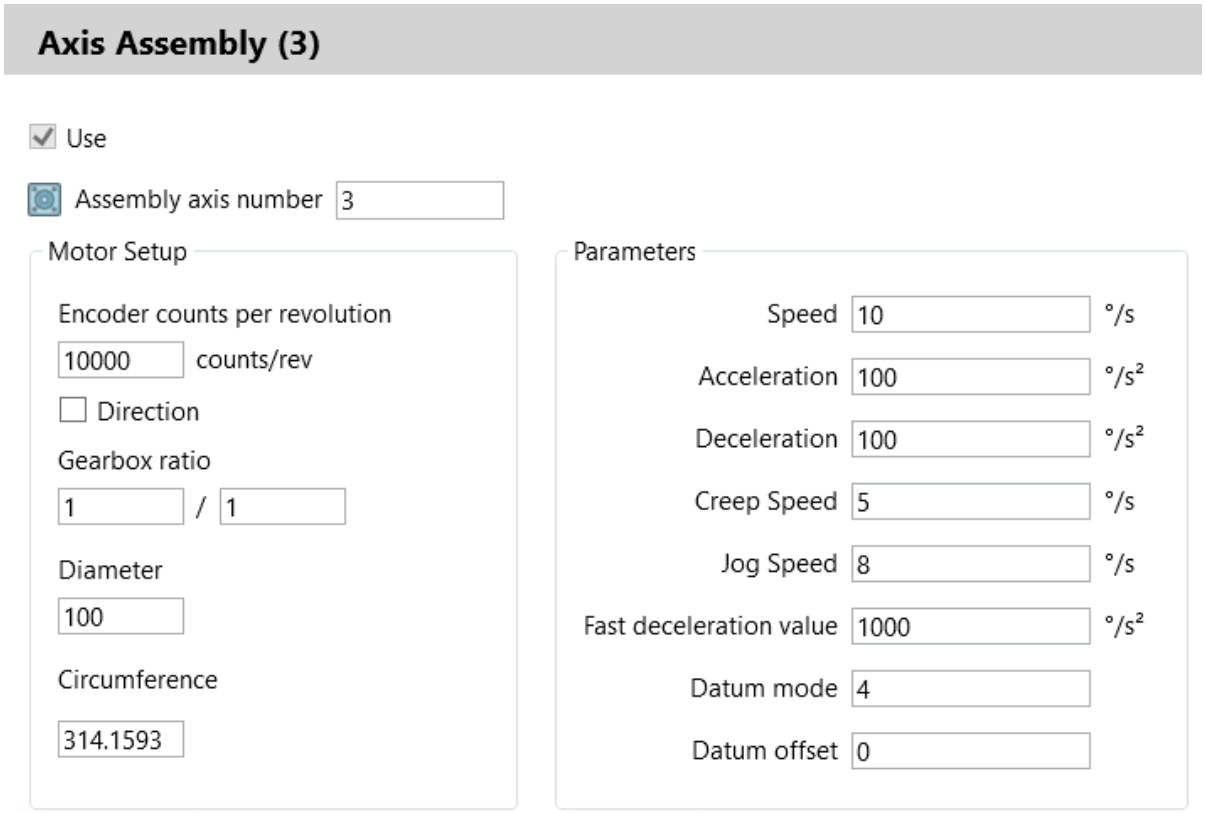
1.4.1. Axis Jaw



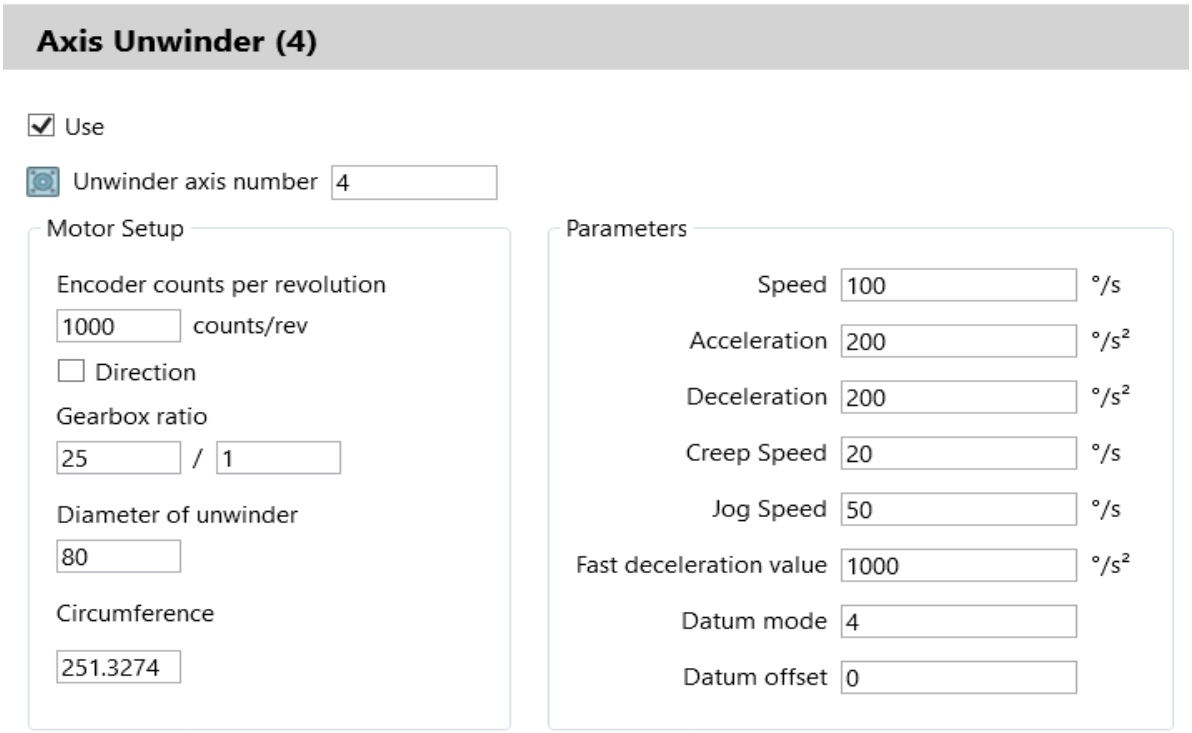
1.4.2. Axis Puller



1.4.3. Axis Assembly



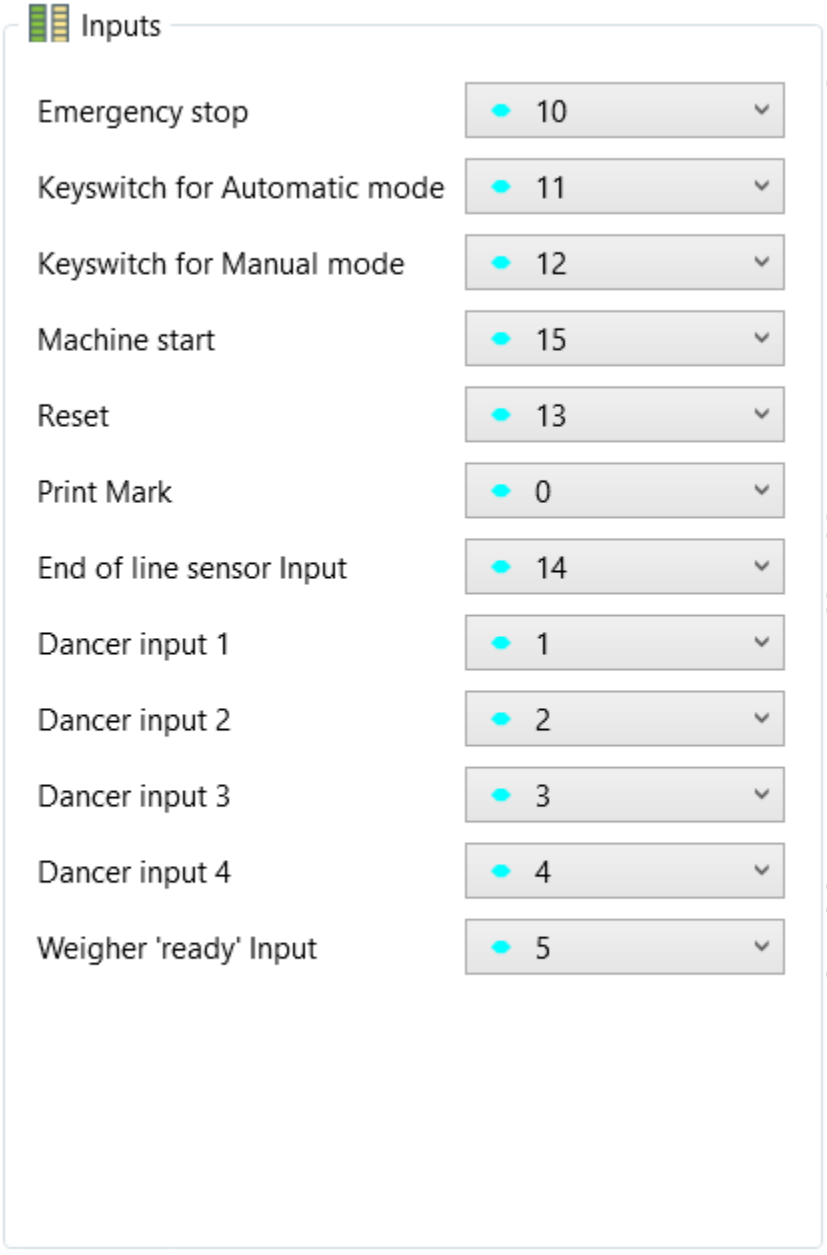
1.4.4. Axis Unwinder



1.5. IO Lines

1.5.1. Inputs

This section refers to all machine I/O summary. Based on previous selection some of the I/O's would be visible. Also there are common I/O for machine operation which are setable here.



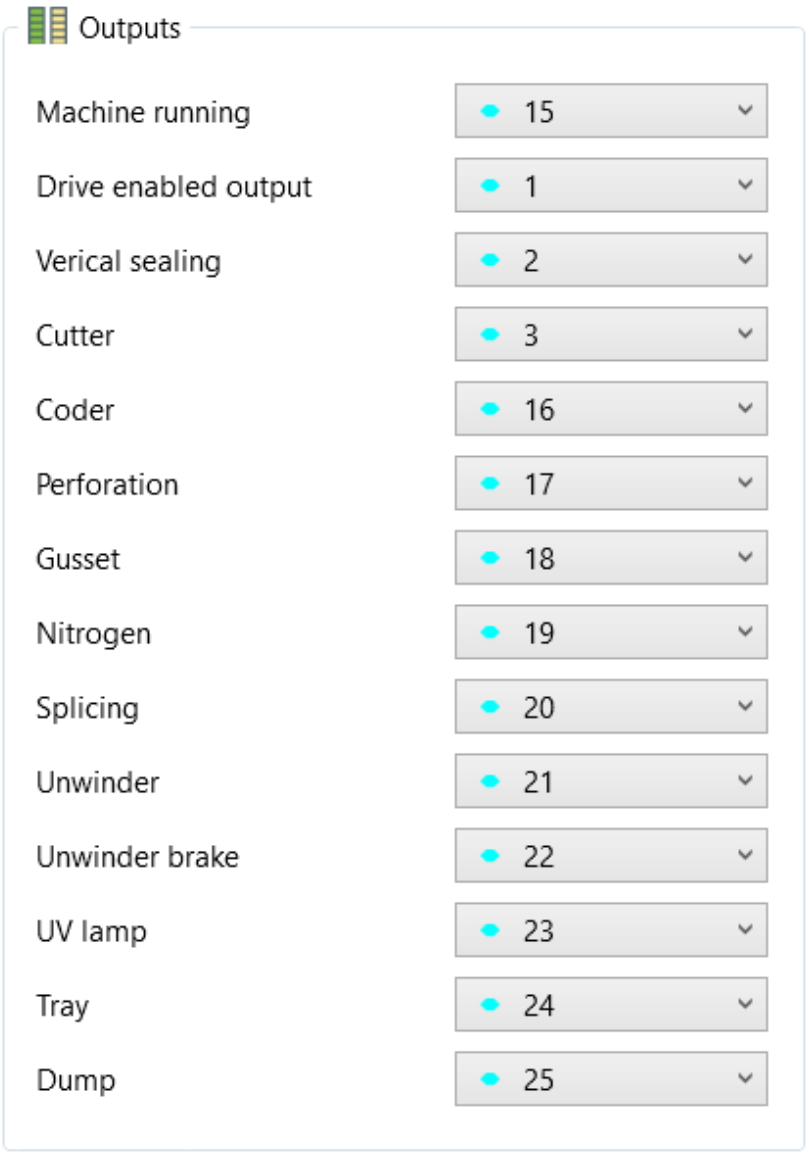
These inputs are for general machin operation so available in all feeder types and are configurable.

If **End of line sensor** is selected in "Unwinder" Group, then this input will be visible else it will not.

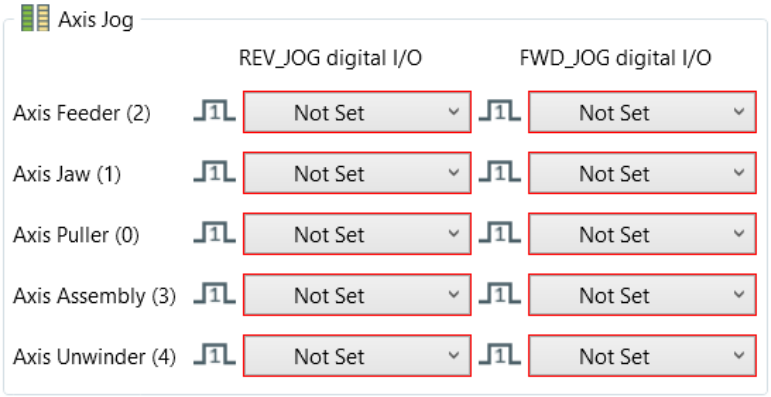
If **Digital control** mode selected in "Unwinder" Group, then these inputs will be visible else these will not appear.

If **Weigher Feeder** selected in "Machine" Group, then this input will be visible.

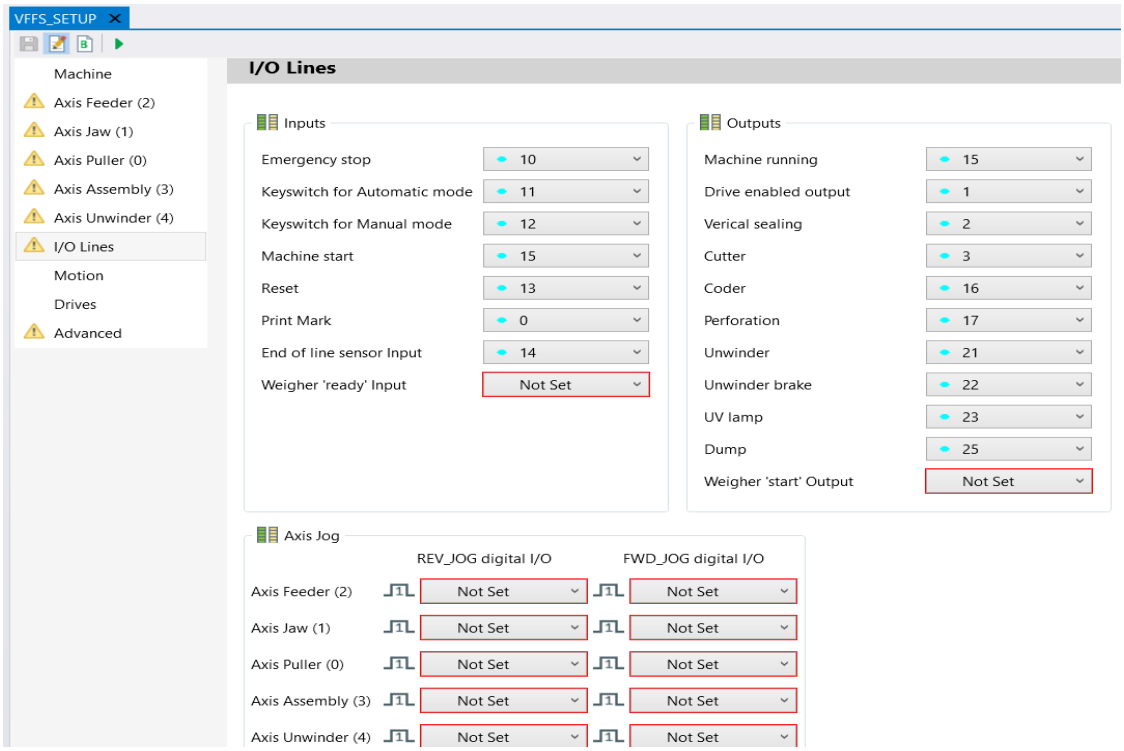
1.5.2. Outputs


Outputs
Machine running
15
Drive enabled output
1
Verical sealing
2
Cutter
3
Coder
16
Perforation
17
Gusset
18
Nitrogen
19
Splicing
20
Unwinder
21
Unwinder brake
22
UV lamp
23
Tray
24
Dump
25
These outputs are available in all feeder types and are configurable as per the requirement.
These outputs are optional outputs which can be configured according to requirement.

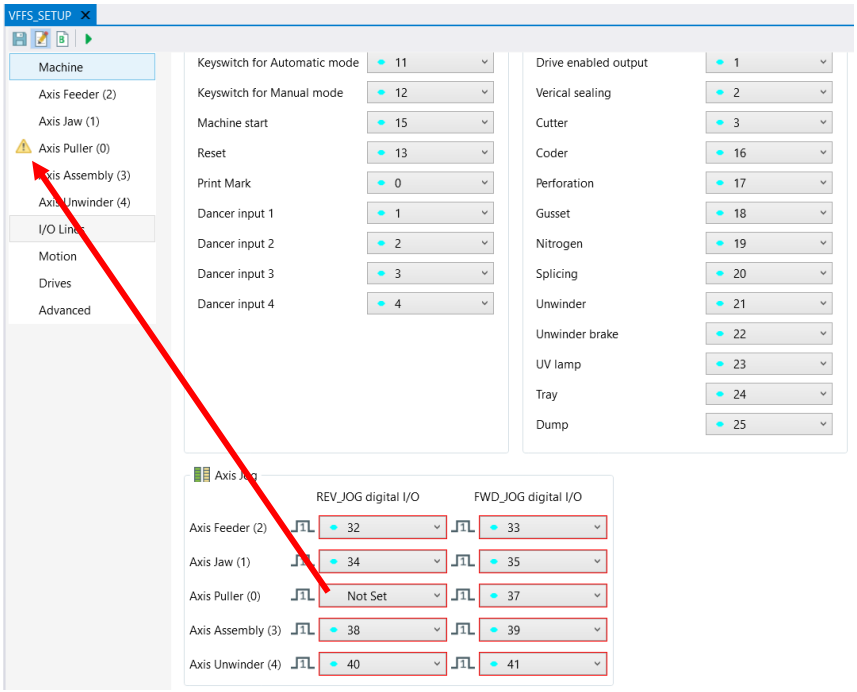
1.5.3. Axis Jog

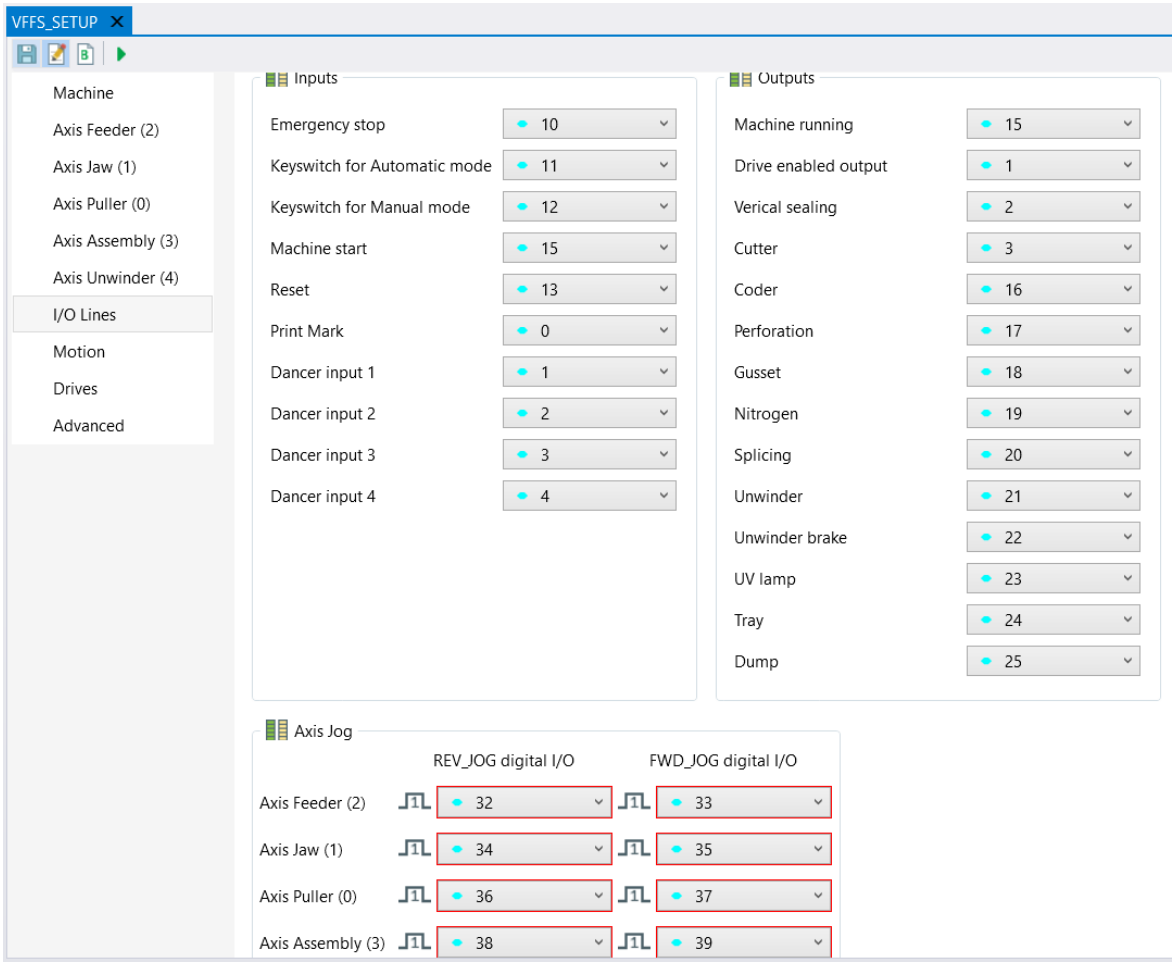


These inputs are used for joggin the axis in manual mode and can be either real or virtual inputs.



If any inputs are not defined, then there will be a warning sign indicated as shown in above figure. If these inputs are not set, then there will be a warning sign in front of respective axis. For example, if REV_JOG digital I/O and FWD_JOG digital I/O related to Axis Puller is not Set then please see the image below,

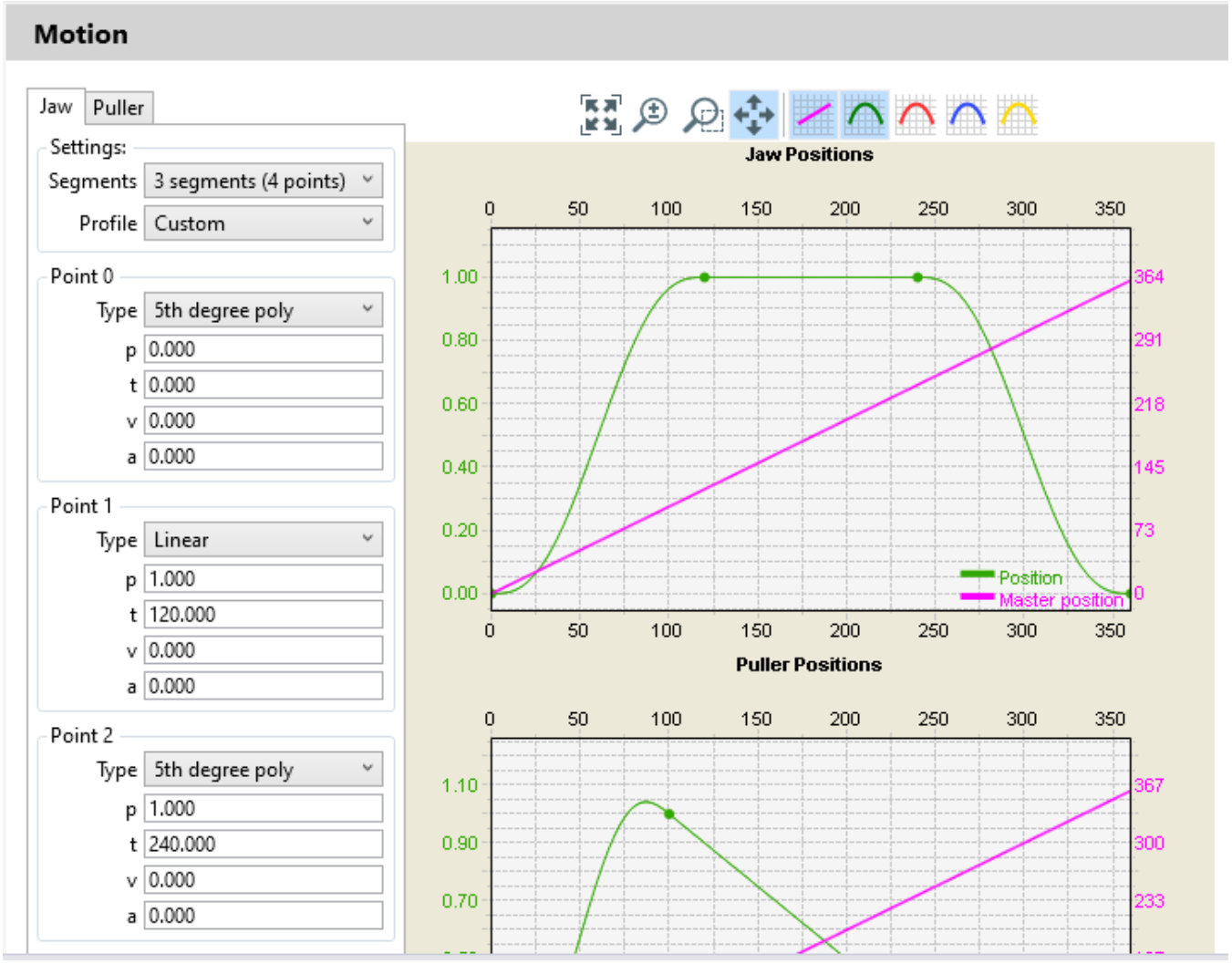




There will not be any warning if we assign proper inputs / outputs /axis numbers.

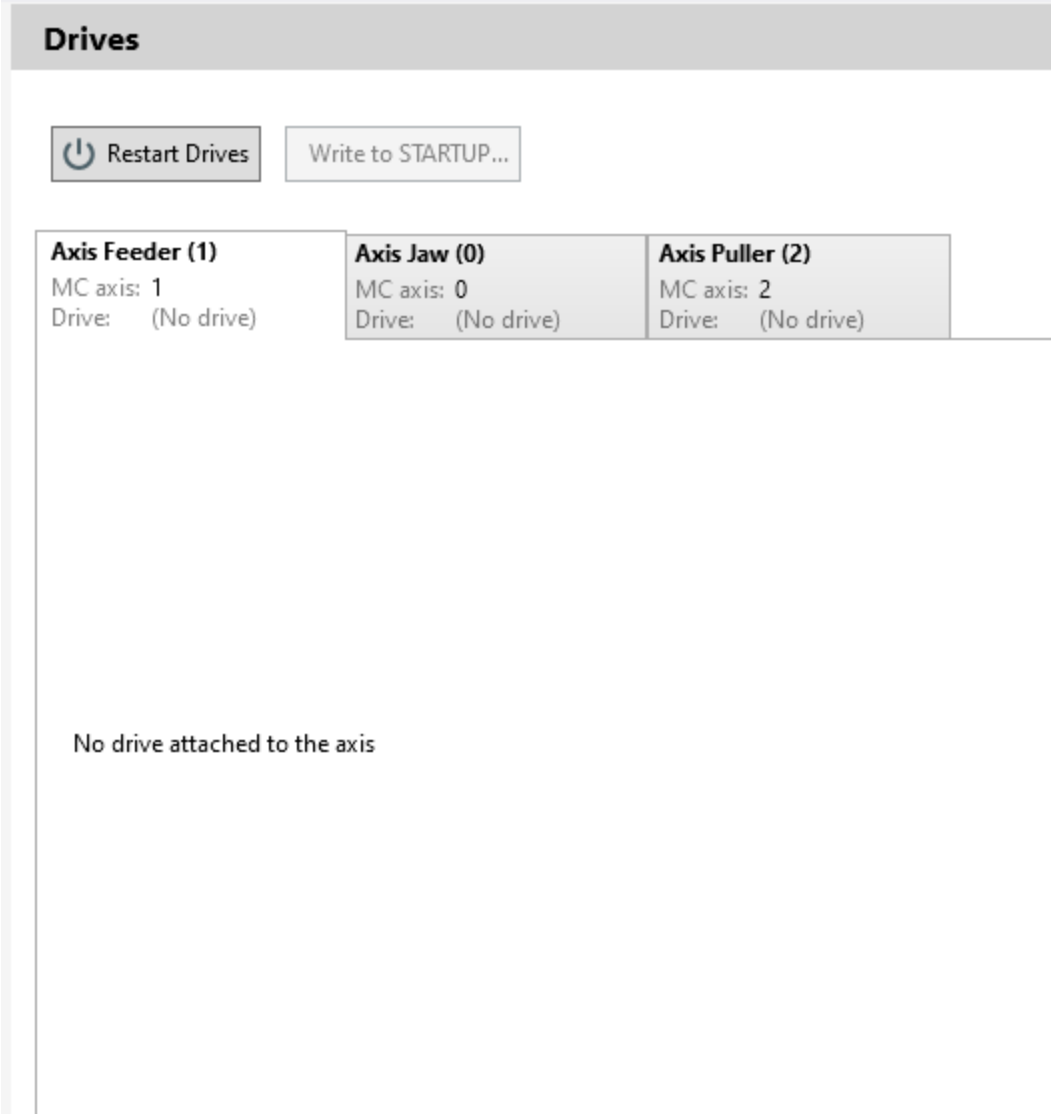
1.6. Motion

This section allows user to create own motion profile for individual axis which are configured.



1.7. Drives

This refers to servo drives when connected with real drives. Drive parameters are directly accessible to for setting.


Drives
Restart Drives
Write to STARTUP...
Axis Feeder (1)
MC axis: 1
Drive: (No drive)
Axis Jaw (0)
MC axis: 0
Drive: (No drive)
Axis Puller (2)
MC axis: 2
Drive: (No drive)
No drive attached to the axis

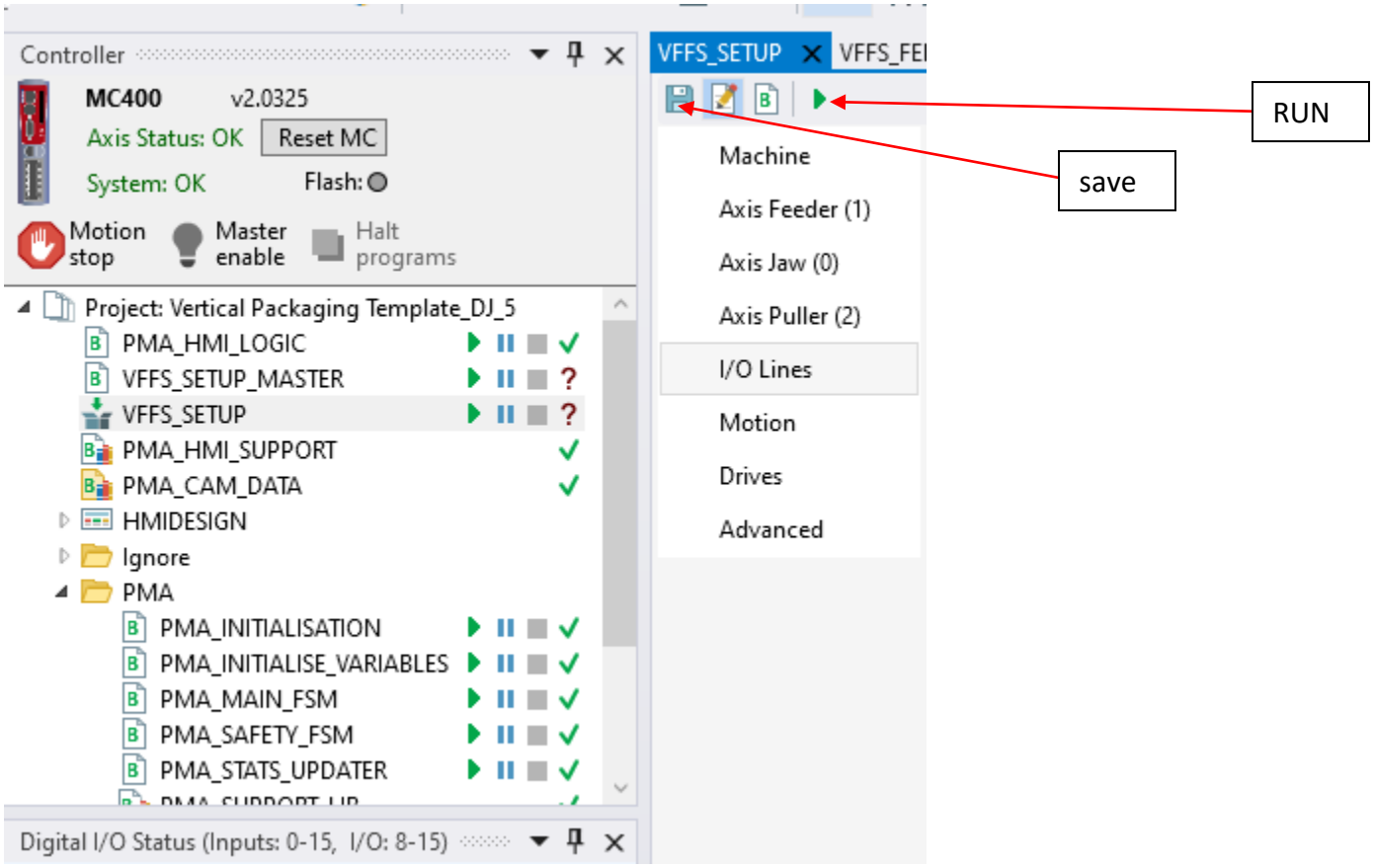
1.8. Advance

Advanced

Name	Value	Description
pkg_ver_maj	-1	
pkg_ver_min	-1	
feeder_type	auger	
weighter_output	31	Output for weigter start
weigher_ready_input	16	Input for wighter 'ready' state
feeder_number	1	Feeder axis number
enc_cnt_feeder	2	Encoder counts per revolution
dia_feeder	50	Diameter of feeder
gb_feeder	50000 / 8000	Gearbox ratio
speed_feeder	100	Speed
accel_feeder	1000	Acceleration
decel_feeder	1000	Deceleration
creep_feeder	5	Creep Speed
fastdec_feeder	10000	Fast deceleration value
jog_speed_feeder	50	Jog Speed
dir_feeder	0	Direction
dat_mode_feeder	0	Datum mode
dat_offset_feeder	0	Datum offset
fact_feeder	0	Factor / Multiplier
circum_feeder	157.0796	Circumference
weight_per_rev	100	Weight per revolution
feeder_drop_time	0	Product drop time
machine_motion	intermediate	
axis_type	virtual	
jaw_number	0	Jaw axis number
enc_cnt_jaw	2	Encoder counts per revolution
dia_jaw	10	Diameter of feeder

2. Running the Template

Once all parameters are set Save the Setup file with Save button as mentioned below and then RUN button to start running the template.



3. Package support

The package includes a support library 'PMA_SUPPORT_LIB' that includes functions that will help users develop their HMI design and debug issues. It is recommended to use these functions to get VR addresses instead of using static VR numbers. Newer versions of the package might have different VR numbers.

3.1. PMA_SUPPORT_LIB functions

The library includes the following functions:

3.1.1.1. vffs_get_vr(vr_name AS STRING)

This function returns VR number for a given name.

Example:

```
IF VR(vffs_get_vr ("initialisation_done")) = 1 THEN
' Do something
ENDIF
```

3.1.1.2. vffs_get_io_number(io_name AS STRING)

This function returns I/O number for a given name.

Example:

```
vffs_get_io_number("ESTOP")
```

Valid I/O strings are listed below:

IN_ESTOP	Emergency Stop Input		IN_UNWIND3	Unwinder input 3
IN_KEYSWITCH_AUTO	Automatic Key switch Input		IN_UNWIND4	Unwinder input 4
IN_KEYSWITCH_MANUAL	Manual Key switch Input		AIN_UNWIND_DIA	Unwinder diameter analogue input
IN_RESET	Reset switch input		AIN_UNWIND	Unwinder analogue input
IN_WEIGHER_READY	Weigher ready input		IN_EOL	End of line input
IN_MACHINE_START	Machine start input		IN_CUP_HOME	Cup home input
IN_MARK	Print mark input		IN_JAW_HOME	Jaw home input
IN_UNWIND1	Unwinder input 1		IN_ASM_HOME	Assembly home input
IN_UNWIND2	Unwinder input 2		IN_UNWIND_ERR	Unwinder error input
IN_AIR_OK	Air okay input		IN_WEB_LEFT	Web assembly left input

IN_DOOR	Door input		IN_CODER_READY	Coder ready input
IN_TAKEUP_CONV_READY	Take-up conveyor ready input		OP_CUTTER	Cutter output
IN_HOPPER_LEVEL	Hopper level input		OP_CODER	Coder output
IN_MACHINE_STOP	Machine stop input		OP_PERFORATION	Perforation output
IN_METAL_DETECT	Metal detect input		OP_GUSSET	Gusset output
IN_PRINTER_READY	Printer ready input		OP_N2PURGE	Nitrogen purge output
IN_N2_AIR_PRESSAURE	Nitrogen air pressure input		OP_SPLICING	Splicing output
IN_EARTH_FAULT	Earth fault input		OP_UNWINDER	Unwinder output
IN_HOPPER_DOOR	Hopper door input		OP_UNWINDER_BRAKE	Unwinder brake input
IN_FEEDER_EN	Feeder enable input		OP_UV_LAMP	UV lamp output
OP_ERROR	Error output		OP_TRAY	Tray output
OP_MACHINE_RUNNING	Machine running output		OP_DUMP	Dump output
OP_DRIVE_ENABLED	Drive enabled output		OP_HSEAL	Horizontal seal output
OP_VERT_SEAL	Vertical seal output		OP_WEB_LEFT	Web assembly left output
OP_TAKEUP_CONV	Take up conveyor output		OP_WEB_RIGHT	Web assembly right output
OP_STIRRER	Stirrer output		OP_EOL_HOLD	End of line hold output
OP_TLAMP_RED	Tower lamp red output		OP_PULLER_ENGAGE	Puller engage output
OP_TLAMP_YELLOW	Tower lamp yellow output		OP_POKER	Poker output
OP_TLAMP_GREEN	Tower lamp green output			
OP_BUZZER	Buzzer output			

Alternative way to get I/O numbers is to read the VR using vffs_get_vr.

Example :

```
VR(vffs_get_vr("estop_in"))
```

3.1.1.3. prog_get_state_process(program AS STRING)

Returns the status of a program. It can be used to check if a program is already running on a process.

Example:

```
IF prog_get_state_process("PMA_SAFETY_FSM") = -1 THEN
    PRINT  "PMA_SAFETY_FSM not running”
ELSE
    PRINT "PMA_SAFETY_FSM running on process "; prog_get_state_process("PMA_SAFETY_FSM")
ENDIF
```

3.1.1.4. parse_run_error(error_number AS INTEGER)

Returns a string of RUN_ERROR.

Example:

```
PRINT "The error ";RUN_ERROR;" : '";parse_run_error(RUN_ERROR);"'";" occurred in line "; ERROR_LINE ;"
```

3.1.1.5. parse_axisstatus(status_value AS INTEGER)

Returns a string of AXISSTATUS.

Example:

```
PRINT "Axis 0 : ";AXISSTATUS AXIS(0);" '"; parse_axisstatus(AXISSTATUS AXIS(0));"'"
```

3.1.1.6. vffs_get_error_int()

Returns error in the package as an INTERGER.

3.1.1.7. vffs_get_error_str()

Returns error in the package as a string.

3.1.1.8. enable_jaw()

Enables Jaw movements in VFFS.

3.1.1.9. enable_puller()

Enables Puller movements in VFFS.

3.1.1.10. parse_drive_error()

Parses EtherCat drive errors for Trio Drives.

Example:

```
CO_READ_AXIS(VR(puller_number), $603F, 0, 6, ecat_puller_drive_error)
VR(ecat_puller_drive_error) = VR(ecat_puller_drive_error) AND $ff
PRINT "Drive Error :" ; parse_drive_error(VR(ecat_puller_drive_error))
```

3.1.1.11. vffs_get_io_string(num AS INTEGER, io AS INTEGER)

Returns a string with the name of the I/O it is used for.

Example:

```
PRINT "Input 6 is "; vffs_get_io_string(6, 0)
PRINT "Output 6 is "; vffs_get_io_string(6, 1)
```

On channel 0:

Input 6 is MACHINE_START

Output 6 is ERROR

3.2. Package Errors

All package errors are listed below:

Error Bit	Description	Solution	Example
0	Emergency Pressed	Release Emergency	VR(error_code). ec_emergency_pressed
1	Motion Error	Turn on reset input	VR(error_code). ec_motion_error
2	System Error	Turn on reset input	VR(error_code). ec_system_error
3	MAIN_FSM Stopped	Check channel #0 for more info	VR(error_code). ec_main_fsm_stopped
4	SAFETY_FSM Stopped	Check channel #0 for more info	VR(error_code). ec_safety_fsm_stopped
5	Initialisation failed	Check channel #0 for more info	VR(error_code). ec_initialisation_failed
6	Auger delay invalid		VR(error_code). ec_auger_delay_invalid
7	Auger speed invalid	VR(speed_feeder) should not be 0	VR(error_code). ec_auger_speed_invalid
8	Auger set weight invalid	VR(set_weight) should not be 0	VR(error_code). ec_auger_setweight_invalid
9	Auger weight per revolution invalid	VR(weight_per_rev) should not be 0	VR(error_code). ec_auger_weightperunit_invalid
10	Auger drop delay invalid	VR(feeder_drop_time) should not be 0	VR(error_code). ec_auger_vopdelay_invalid
11	Auger pulses per revolution invalid	Check if VR(enc_cnt_feeder) is not a low value.	VR(error_code). ec_auger_ppr_invalid
12	Auger gear ratio invalid	VR(gb_feeder) should be valid	VR(error_code). ec_augergb_invalid
13	Auger accel time	VR(accel_feeder) or	VR(error_code).

	invalid	VR(decel_feeder) should be valid	ec_auger_acceltime_invalid

3.3. Debugging the package

To understand what state the package is in, or what it is doing, users can set "**VR(debug) = 1**" then open channel 5 and 7. The package will start printing debug strings on each channel. These strings will help understand what state the package is in and why.

✪ Please stop all program and set VR(debug) = 1 in VFFS_SETUP. Run VFFS_SETUP again to start seeing debug logs.

✪ Check channel 0 if information. The package prints errors or tips on channel 0 to help users get the machine running.