

A01 — Conveyor Belt

FMU model and guide for PLC programming

Version 1.0

Student exercise — Homa Virtual Plant (Model-in-the-Loop)

Homa inženjering

www.homa.hr · info@homa.hr

© 2026 Homa inženjering. All rights reserved.

Contents

1. Exercise objective.....	3
2. Process description (what the model represents).....	3
3. FMU interface — signals.....	3
3.1. FMU inputs.....	3
3.2. FMU outputs (measurements the PLC reads).....	3
4. What the PLC must achieve (control objectives).....	4
5. Suggested control structure (guideline).....	4
6. Recommended sample times.....	4
7. Test scenarios and acceptance criteria.....	4
8. Integration into Homa Virtual Plant.....	5
9. Notes on the model.....	5

1. Exercise objective

You are given an **FMU model** representing a conveyor belt with a hopper. Your task is, through **Homa Virtual Plant** to connect a PLC to this model and write a **PLC program that controls the process** according to the given objectives.

This is a **Model-in-the-Loop (MIL)** setup: the model mimics a real plant, your PLC drives it, and you develop and test your program on a virtual system. The model is “open” — it contains no controller of its own; **all control is performed by your PLC** through the model inputs.

2. Process description (what the model represents)

The belt conveys material from a hopper. Material inflow (`u_material_in`) fills the hopper; the belt speed is commanded. The focus is on **sequences, interlocks and jam detection**.

Available / in the process:

- **Belt drive:** speed is commanded via `u_speed_cmd`.
- **Hopper:** `y_hopper_level` rises with inflow; too much material or excessive motor current causes a **jam**.
- **Protection:** the PLC detects a jam and safely resets it (`u_reset_jam`).

3. FMU interface — signals

This is the most important part. Pay attention to the signal direction:

Signal direction: what is an **INPUT** for the FMU is an **OUTPUT** (command) for your PLC, and vice versa. The actuators your PLC **writes**; the measurements your PLC **reads**.

3.1. FMU inputs

a) Actuators — controlled by your PLC (PLC outputs):

Variable (FMU)	Type	Unit	Range	Description
<code>u_speed_cmd</code>	Real	–	0.0 – 1.0	Commanded belt speed
<code>u_enable</code>	Real (0/1)	–	0 or 1	Line enable
<code>u_reset_jam</code>	Real (0/1)	–	0 or 1	Jam reset

b) Disturbances / scenario — the environment, NOT controlled by your PLC:

Variable (FMU)	Type	Unit	Range	Description
<code>u_material_in</code>	Real	kg/s	0 – ~10	Material inflow (upstream)

You set the disturbances on the **PLC HMI** (as a parameter / simulated sensor), and the PLC forwards them to the FMU. They are not an output of your controller.

3.2. FMU outputs (measurements the PLC reads)

Variable (FMU)	Type	Unit	Range	Description
<code>y_speed</code>	Real	m/s	0 – 1.5	Belt speed
<code>y_material_out</code>	Real	kg/s	0 – ~8	Material discharge

Variable (FMU)	Type	Unit	Range	Description
y_hopper_level	Real	–	0 – 1.2	Hopper level (1 = full)
y_motor_current	Real	A	~5 – 60	Motor current
y_jam	Real (0/1)	–	0 or 1	Jam (1 = blockage)

4. What the PLC must achieve (control objectives)

Your PLC program must:

1. Start/stop sequence with interlocks (downstream readiness, high hopper level).
2. Command the belt speed (u_speed_cmd) as required.
3. Jam detection (high y_motor_current / stall) and safe stop (trip).
4. Controlled reset and restart after a jam.
5. Prevention of an unexpected start.

The objectives are given; **you design the control method yourself** (a suggested structure is provided below as an aid).

5. Suggested control structure (guideline)

Sequence (states):

- Stop → Start-permissive → Ramp-up → Run; transitions on conditions (Start, downstream readiness, hopper level).

Interlocks:

- Do not start if the level is too high or downstream is not ready.

Jam detection and reset:

- If y_motor_current is too high or the belt stalls under a run command for longer than e.g. 3 s → alarm and trip. Reset only via u_reset_jam after the level drops.

6. Recommended sample times

Layer	Recommended cycle
Sequences and interlocks	50 – 200 ms
Motor / interlock signals	10 – 50 ms
FMU co-simulation communication step	~0.05 – 0.2 s

7. Test scenarios and acceptance criteria

Scenario	How to trigger	Expected PLC behaviour	Pass criterion
Normal start	u_enable=1, u_speed_cmd>0	orderly ramp-up and run	no unexpected start
High hopper level	u_material_in high	stop filling / the line	y_hopper_level does not exceed the limit
Jam	overload (y_jam=1)	alarm and trip	jam detected, line stops
Reset after a jam	u_reset_jam after the level drops	controlled restart	safe restart
Sensor fault	freeze	safe response	no unsafe operation

Scenario	How to trigger	Expected PLC behaviour	Pass criterion
	y_motor_current		

General criteria: controlled variables within limits, no instability / excessive cycling, correct response to faults.

8. Integration into Homa Virtual Plant

6. Load `Plant.fmu` (the compiled model) into Homa VP.
7. Map the FMU variables: **actuators** (`u_*`) as PLC outputs, **measurements** (`y_*`) as inputs.
8. Disturbances (`u_material_in`) are set on the HMI as operating conditions and tests; the PLC forwards them to the FMU.
9. Select a communication channel (e.g. OPC UA / S7) and run the closed loop.

The model is configured for **StopTime 600 s** (Interval 1 s). Real-time pacing is driven by Homa VP via a speed factor.

9. Notes on the model

Important: the model is a simplified starter plant for developing and testing PLC logic — **it is not a validated industrial model**. The parameters are illustrative; calibrate them before real use.

- The dynamics are approximate; the purpose is a control exercise, not exact process analysis.
- `y_jam` is a simulated fault (jam) that the PLC must detect and safely handle.