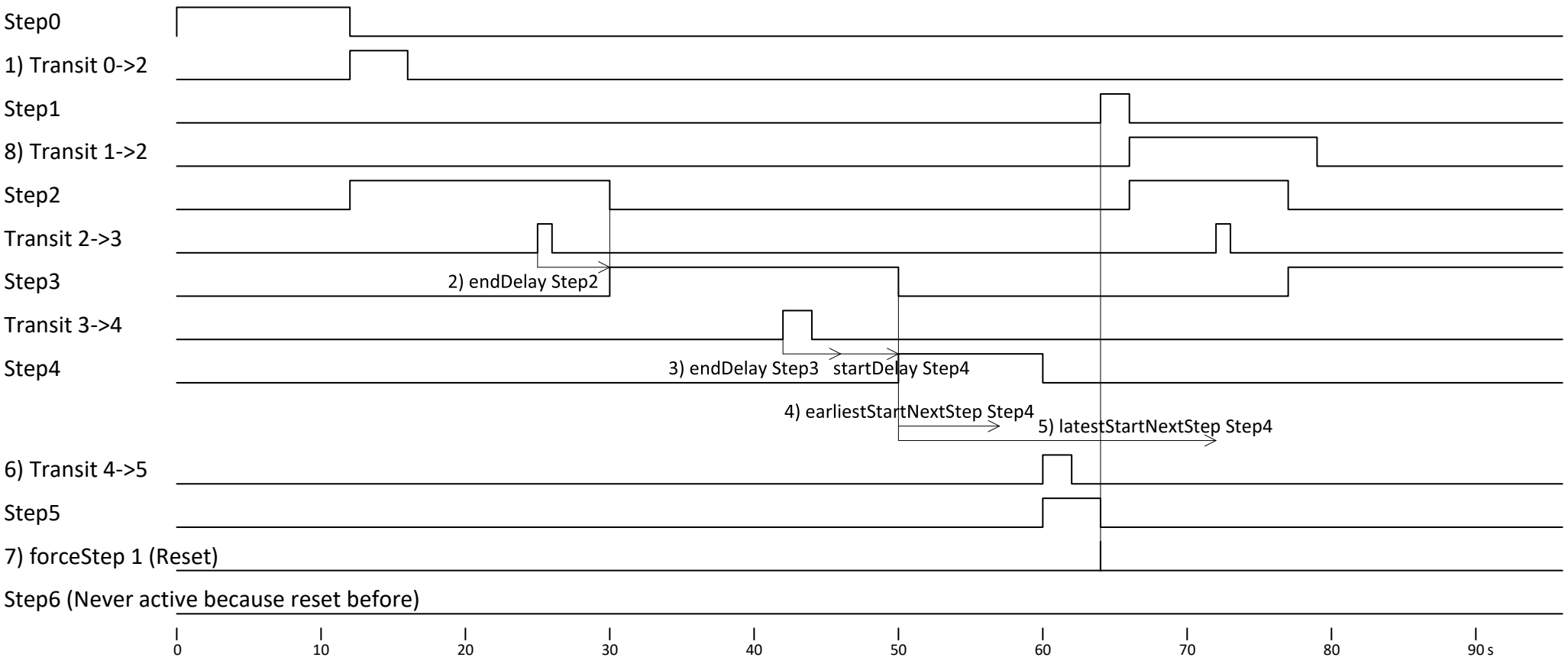
Step Sequence Timing 04.08.2025 Andreas1313

Example for a possible timing for the timing diagramm below

	Step0	Step1	Step2	Step3	Step4	Step5	Step6
a) startDelay_ms	0	0	0	0	**5000**	0	0
b) earliestStartNextStep_ms	5	5	5	5	**7000**	5	0
c) latestStartNextStep_ms	0	0	22000	300000	**22000**	22000	22000
endDelay_ms	0	0	**5000**	**5000**	0	0	0

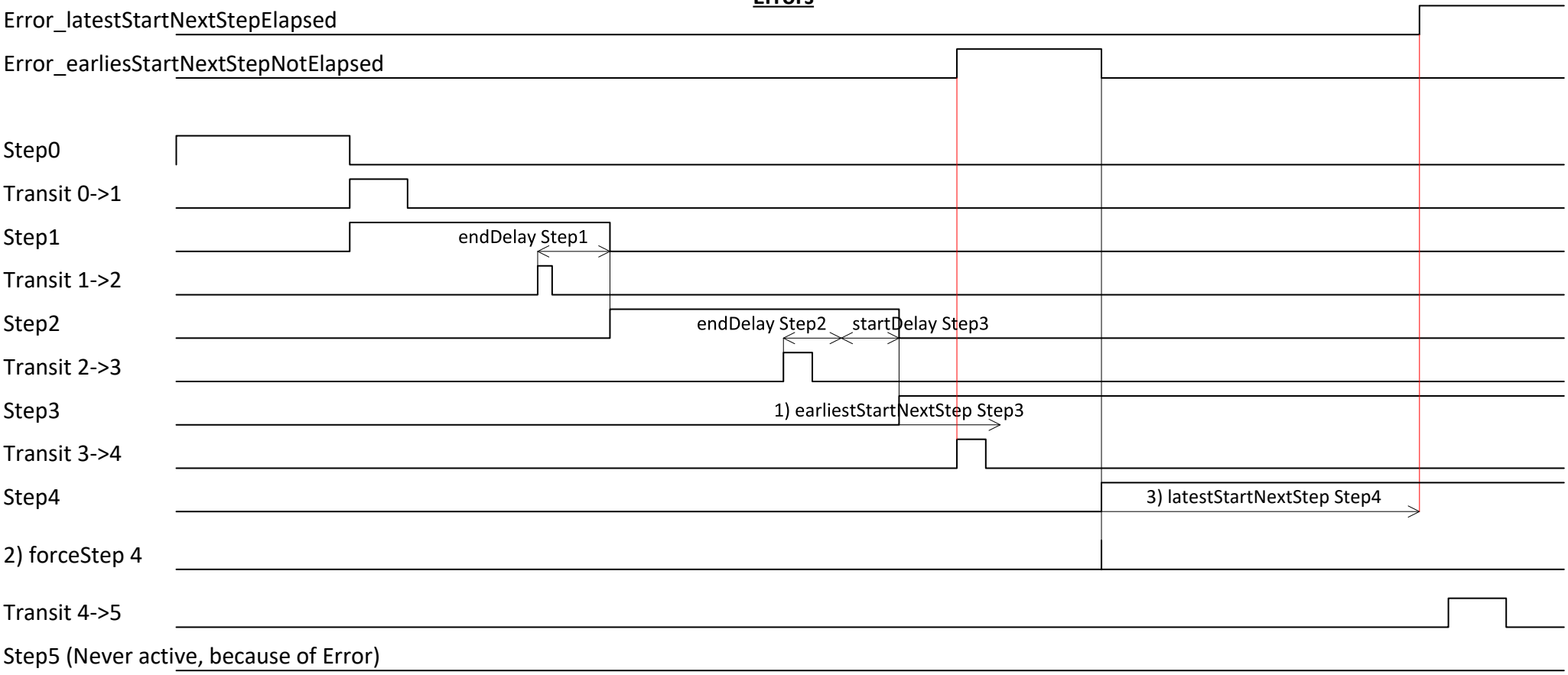
a) Because forceStep skips the startDelay, I would use startDelay only where it makes sense. endDelay is the normal delay.
b) earliestStartNextStep_ms: When this time is elapsed, it is allowed to transfer to the next step.
c) latestStartNextStep_ms: The next transit must be in this time. **0 is to switch the latestStartNextStep totally off – no error.**
Hint for earliestStartNextStep_ms and latestStartNextStep_ms. When you want a stable run, but also an error, when something is totally running wrong.
- Set all earliestStartNextStep_ms and latestStartNextStep_ms to 0.
- Measure the timing (with this library).
- Set for example: earliestStartNextStep_ms = yourMeasuredTime_ms / 10.
- Set for example: latestStartNextStep_ms = yourMeasuredTime_ms * 10.
- Set only critical timings nearer to yourMeasuredTime.



1) Transit 0->2.	You can transit between every step
2) endDelay Step2:	After the endDelay it switches to the next step.
3) endDelay Step3 startDelay Step4:	If you have both, they are cumulated
4) earliestStartNextStep Step4:	After this time is elapsed, it is allowed to transit to the next Step(5). See error page.
5) latestStartNextStep Step4:	Before this time is elapsed, the transit to Step(5) must be done. See error page.
6) Transit 4->5:	It is between end of earliestStartNextStep Step4 and latestStartNextStep Step4, so it is correct.
7) forceStep 1 (Reset):	Here we jump to Step1 and not to Step0. Perhaps you want to do something different from normal startup at a reset.
8) Transit 1->2:	A transit can be very long. But of course should be back when this step is called again.

Step Sequence Timing
Errors

04.08.2025 Andreas1313



1) earliestStartNextStep Step3: The error is immediately when the transit is to early.
The Step4 is not activated here, because of the error.
A startDelay Step4 would not be part of the earliestStartNextStep. So this timing graph would exactly look the same.

2) forceStep 4: In this example we force Step4. We imagine that the next position was reached manually after the error, and than we force the next step.
Errors are reset.
Forcing a step skips the startDelay (when Step4 would have a startDelay).
latestStartNextStep (and earliestStartNextStep) always start at the activation of a step.

3) latestStartNextStep Step4: The error is immediately when the latestStartNextStep is elapsed.

Because forceStep skips the startDelay, I would use startDelay only where it makes sense. endDelay is the normal delay.

Hints for Professional User 04.08.2025 Andreas1313

A force to the actual step make sense. Example: When you have an error, you can repeat the actual step.
The earliestStartNextStep of this step and the latestStartNextStep of this step also start from beginning.

You can use latestStartNextStep not only for critical errors. Example: When you want a warning, you can use it:
Show a warning when you have the error and transfer to next step with a longer latestStartNextStep time.
When you do not have an error, you can skip the next step (with the longer latestStartNextStep time).

earliestStartNextStep: Recommendation: When you do not want to allow that the next transfer is already active, when the actual step getting active, than set this to a value a little bit more than your processor loop time. It will throw an error when the next step would be immediately activated.