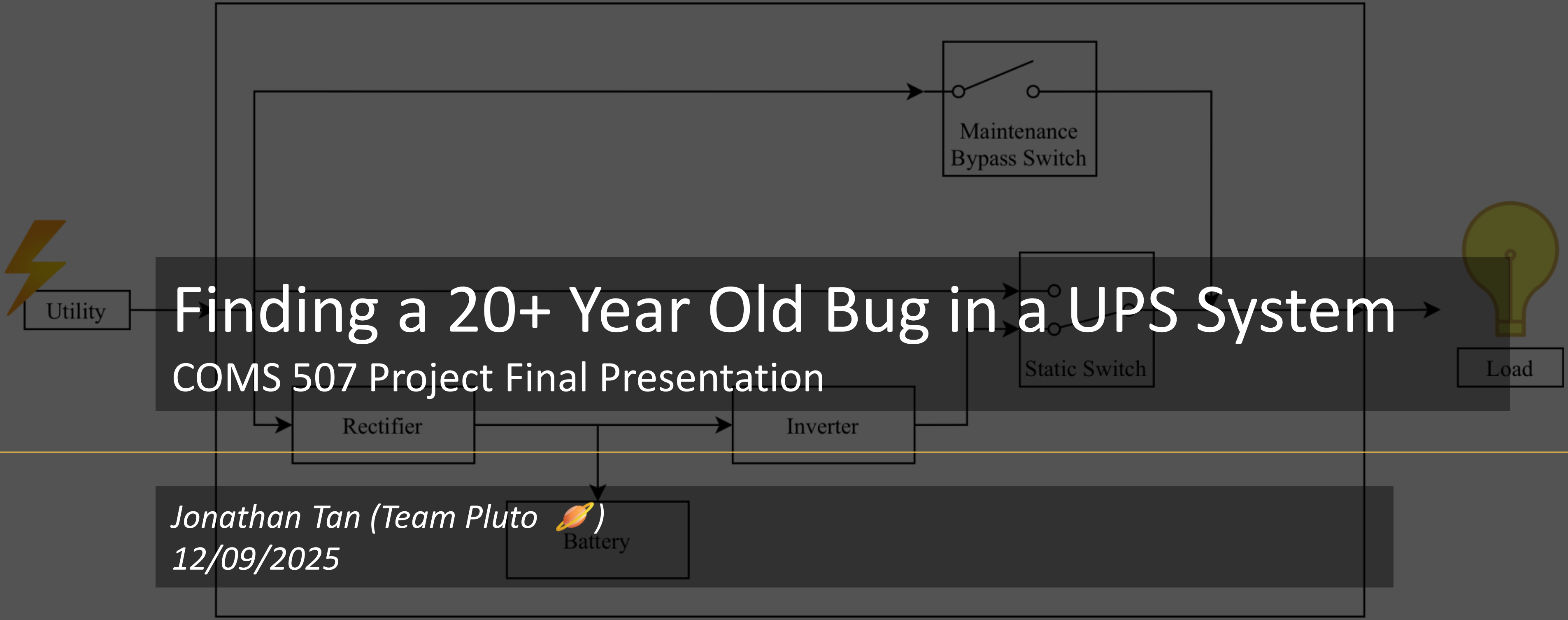


Finding a 20+ Year Old Bug in a UPS System

COMS 507 Project Final Presentation

Jonathan Tan (Team Pluto 🪐)

12/09/2025

IOWA STATE UNIVERSITY

Problem Statement

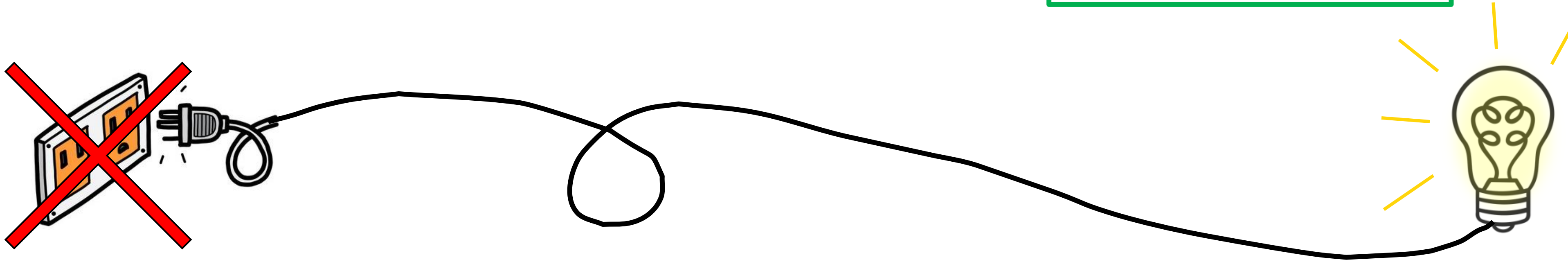
- There is this weird issue in this UPS system, where sometimes, normally after maintenance events, load drop (bad) will happen.
- What is the bug? And if 20+ years ago, the designers have model checked their specs, will they have caught the bug?

Project Goal

- Answer question: If the original developers had model checked the system, would they ever catch this bug at design time instead of having an intern spend 3 months fixing it 25 years later?
- Verify property: $\mathrm{ALWAYS}(\mathrm{good}) \equiv \mathrm{ALWAYS}(\neg \mathrm{load\_drop})$
- Tool: Spin
- Expected outcome: A counterexample trace from Spin telling me what the issue in the spec is

Uninterruptible Power Supply (UPS) System Overview

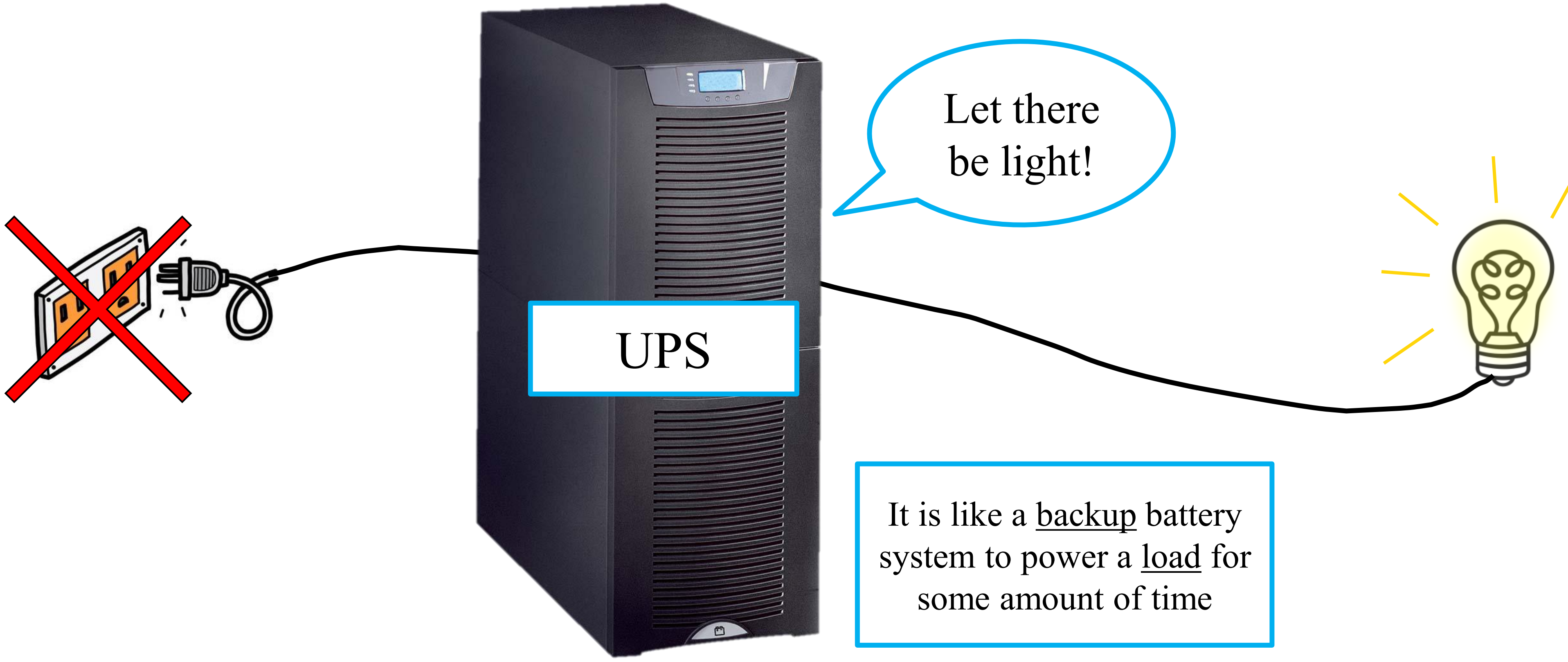
We want our favorite light on



Power outage

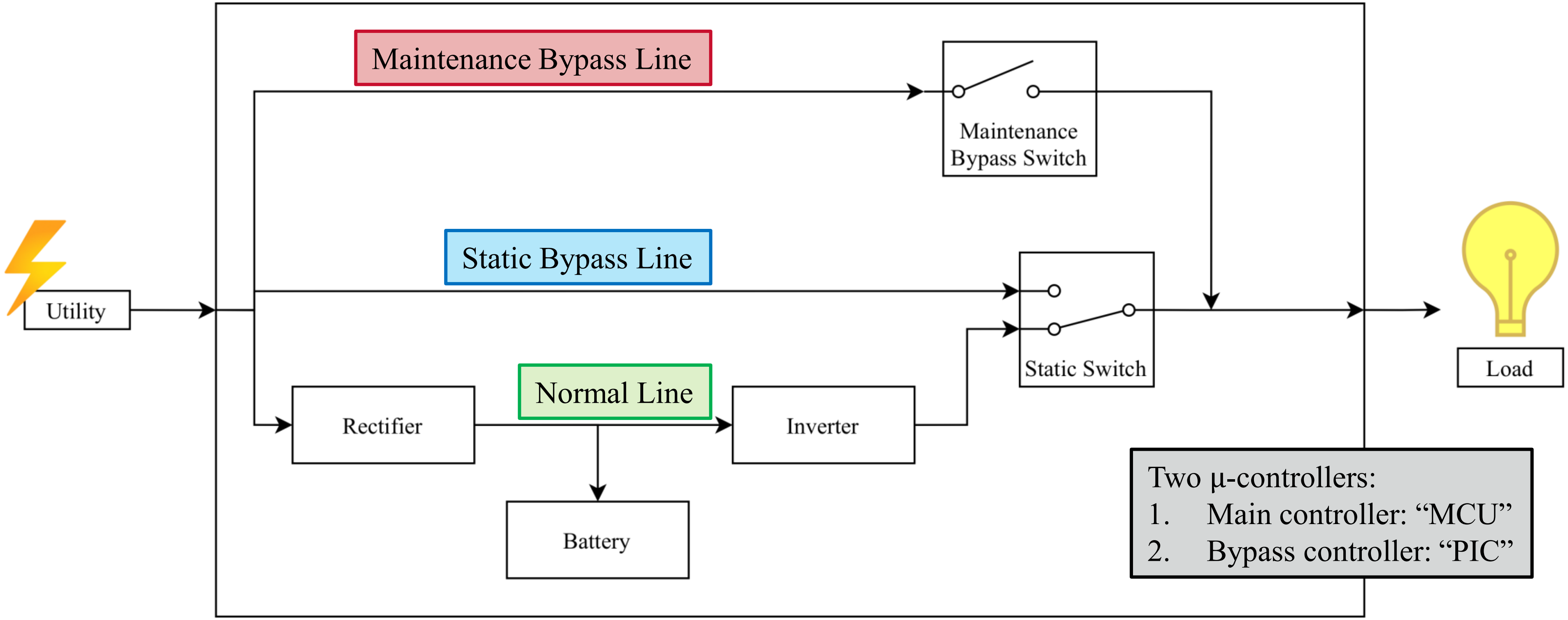
No light ☹

Uninterruptible Power Supply (UPS) System Overview



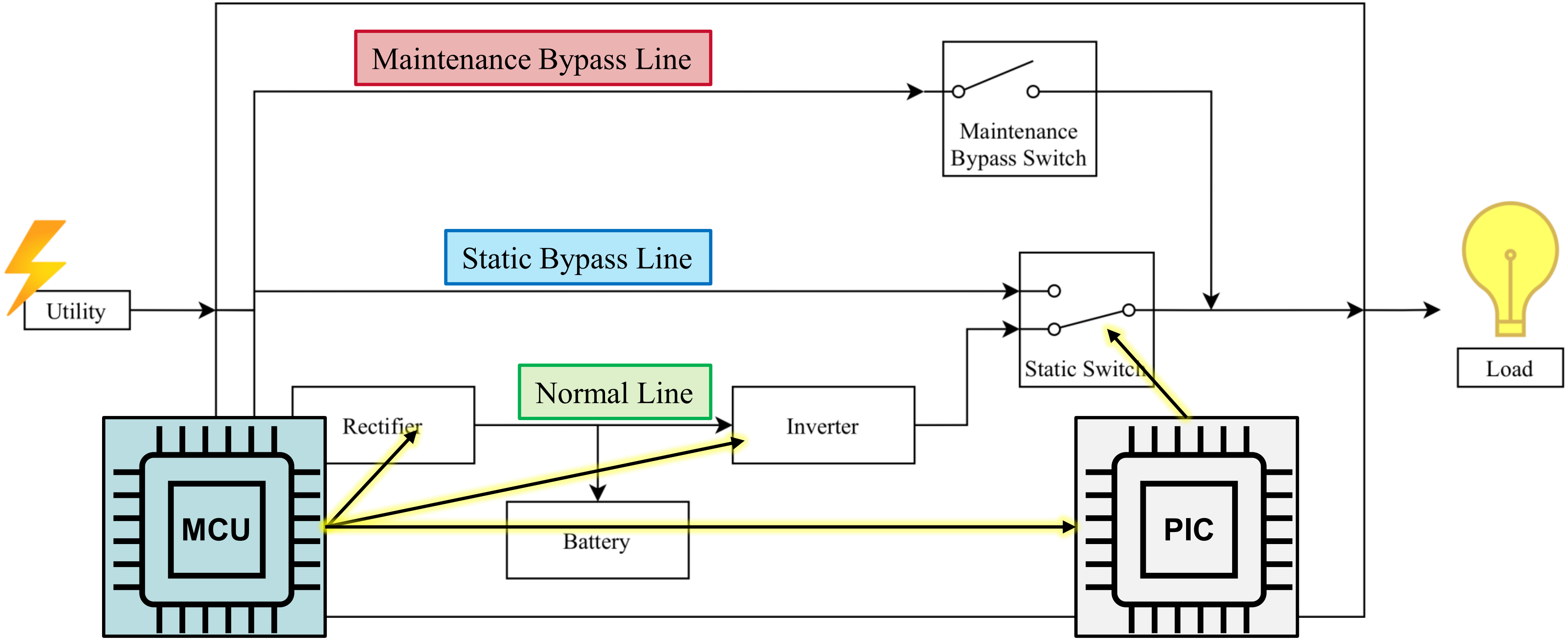
It is like a backup battery system to power a load for some amount of time

Uninterruptible Power Supply (UPS) System Overview



Two µ-controllers:
1. Main controller: "MCU"
2. Bypass controller: "PIC"

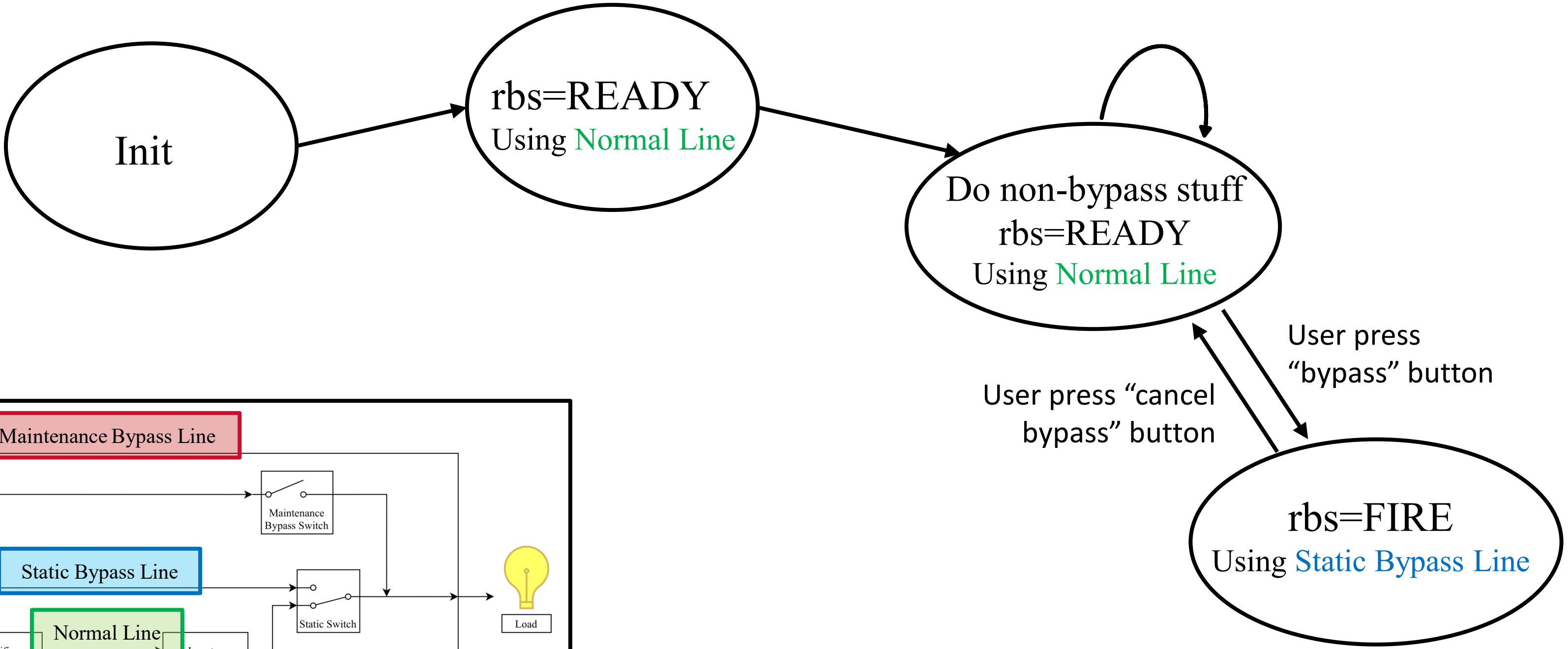
Uninterruptible Power Supply (UPS) System Overview

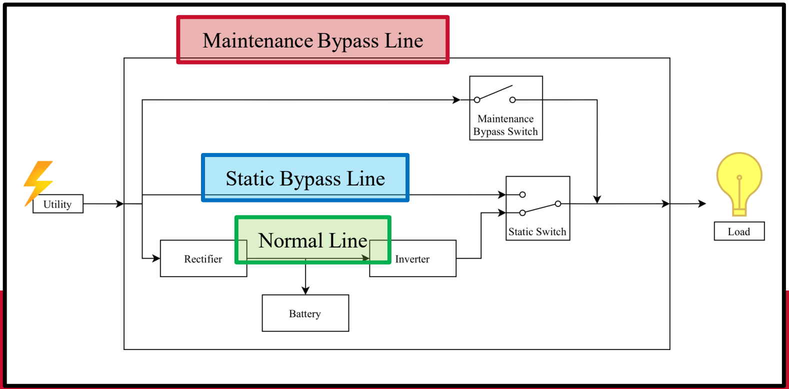


FSMs, many FSMs…

Environment (Simplified Model)

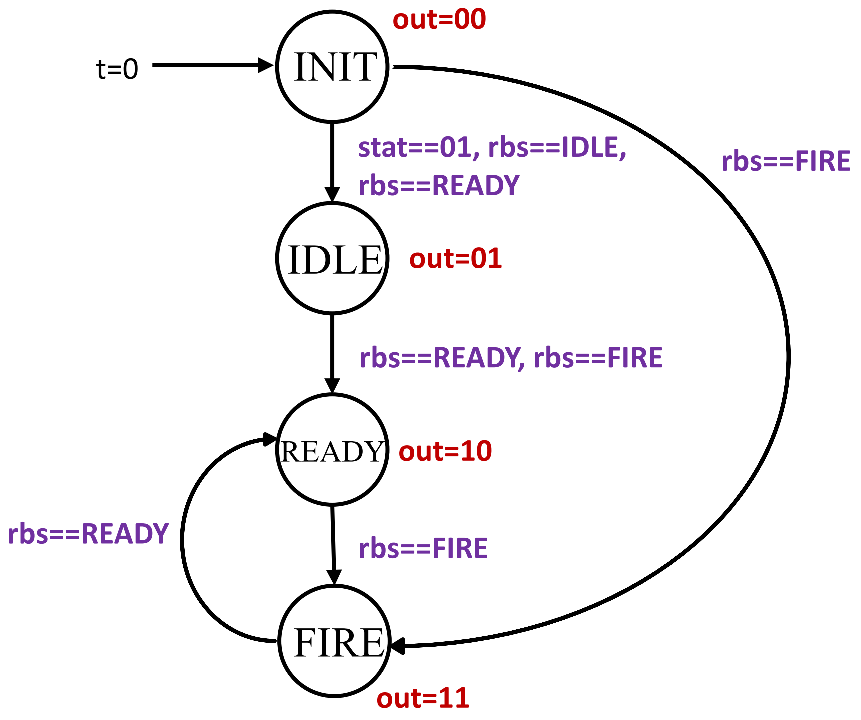
- Huge FSM, lives in the MCU, controls every aspect of the UPS system



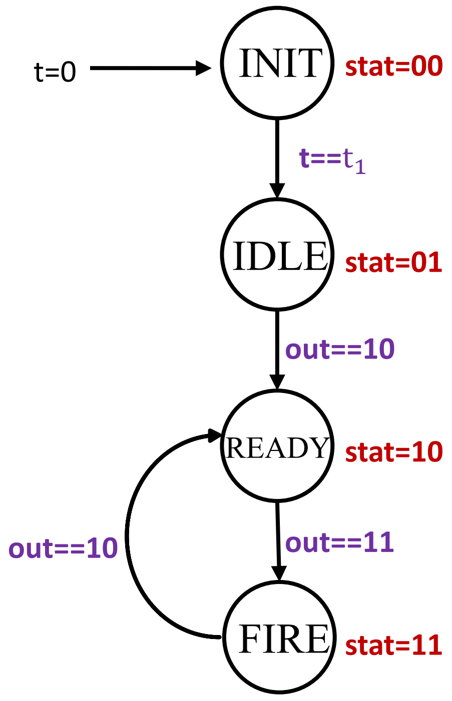


Bypass State Machine Subsystem Overview

MCU_byp_State

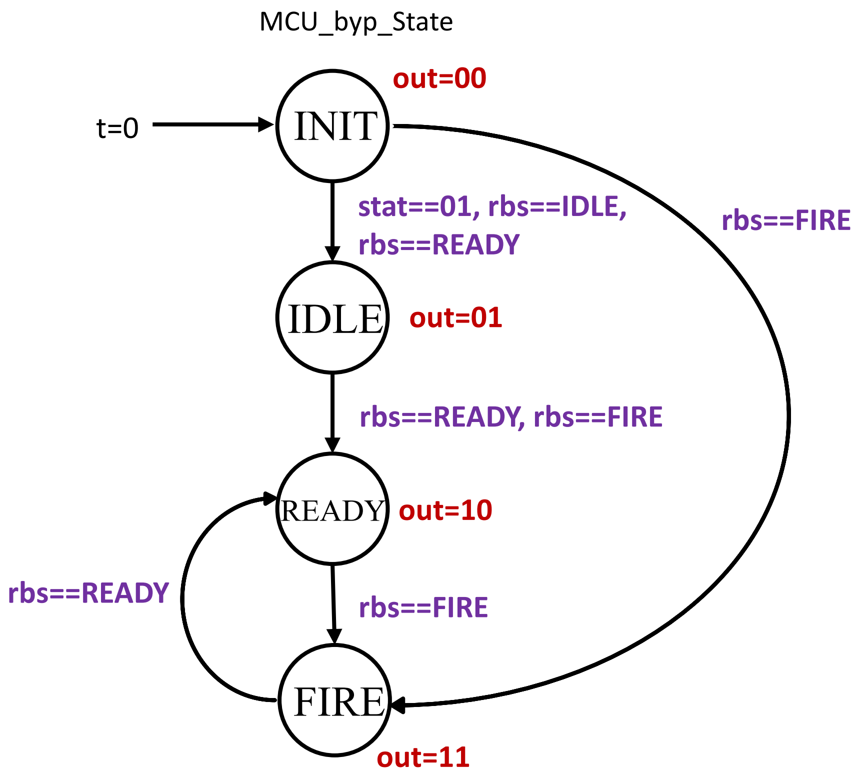


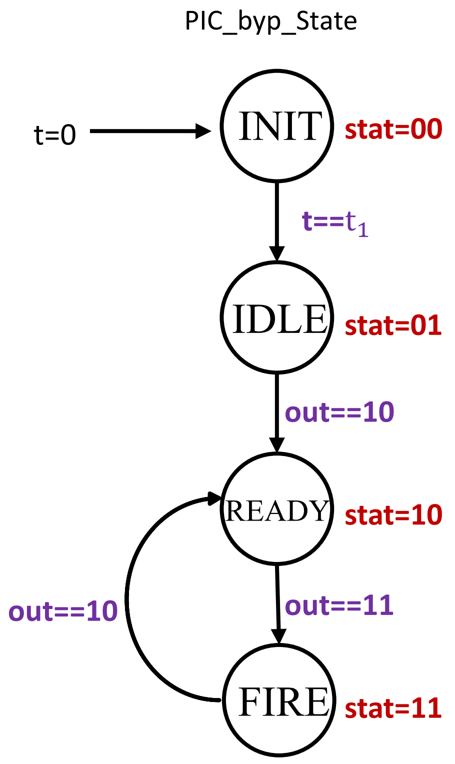
PIC_byp_State



rbs: Request Bypass State

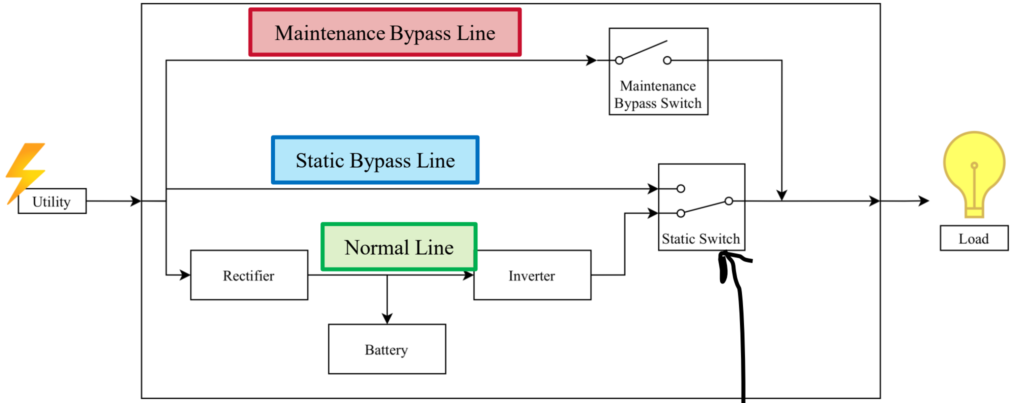
Bypass State Machine Transition Specification

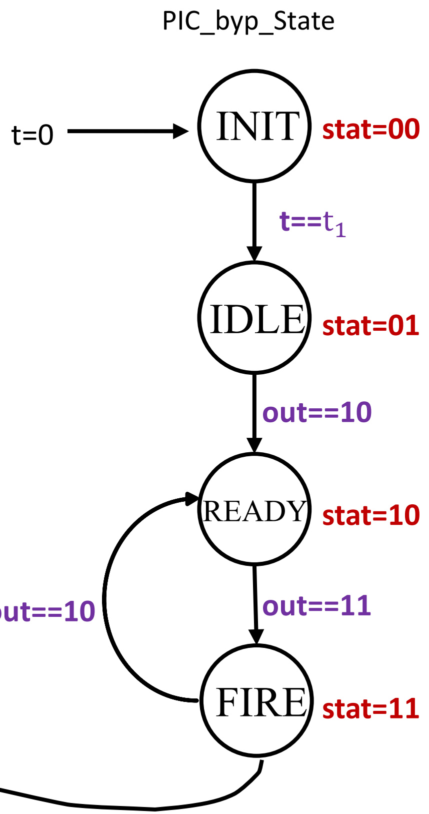




rbs: Request Bypass State

Bypass State Machine Transition Specification

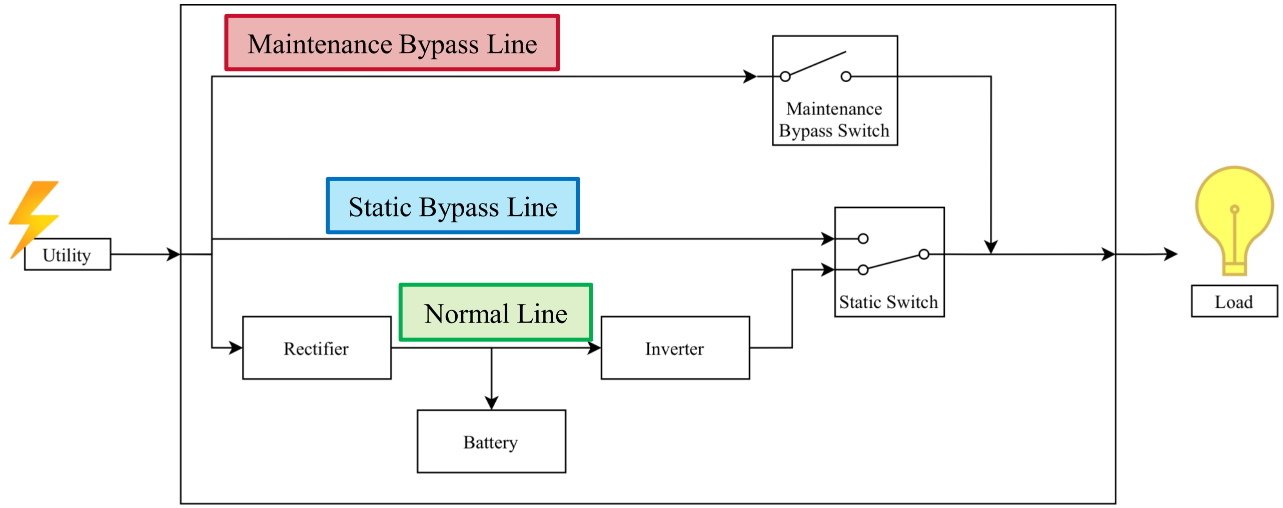




Reminder, the PIC's FIRE state controls the static switch

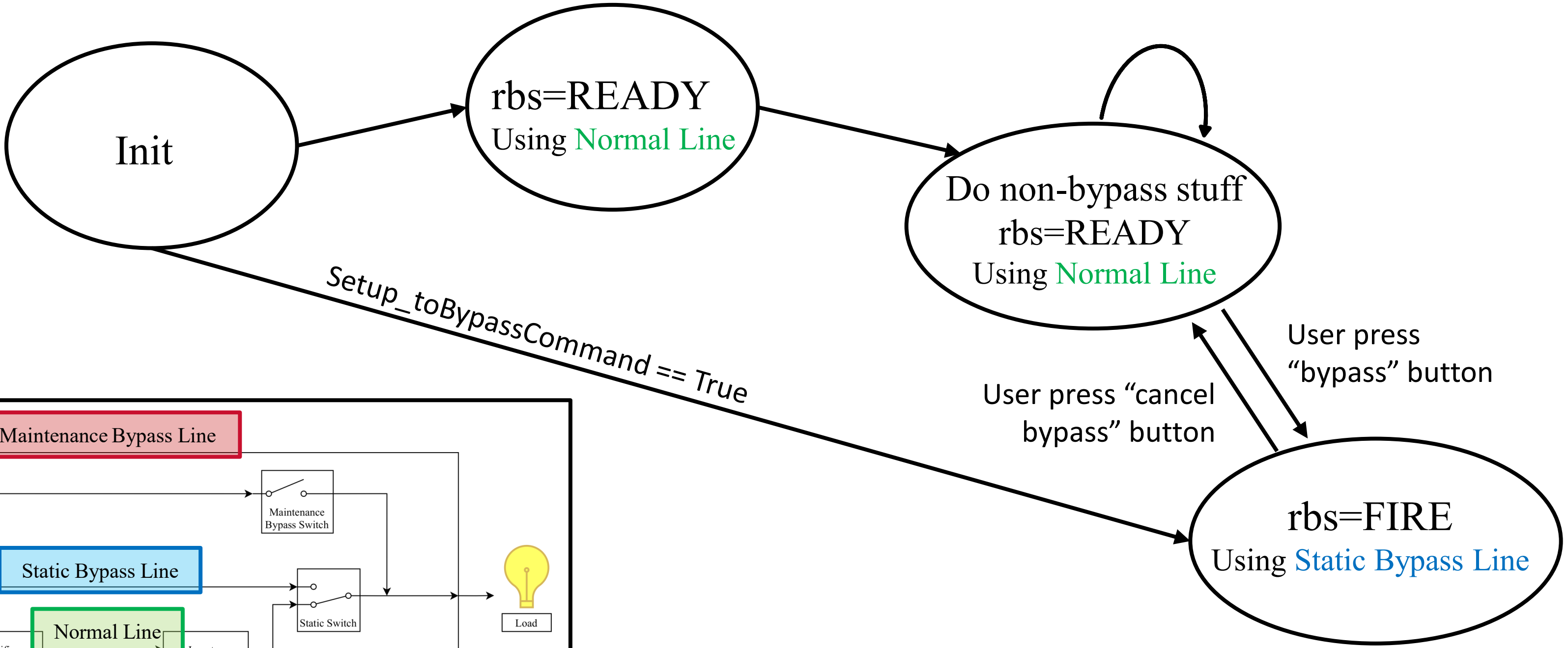
Maintenance Event

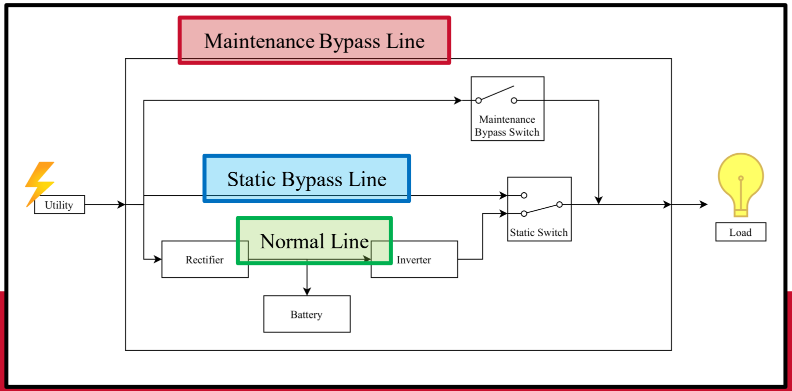
- Sometimes, maintenance requires load to be transferred onto Maintenance Bypass Line
- Requires Normal Line and Static Bypass Line to be powered off
- Meaning, PIC and MCU will also be powered off
- Before powering off the MCU, a flag, Setup_toBypassCommand, is set to True in the MCU



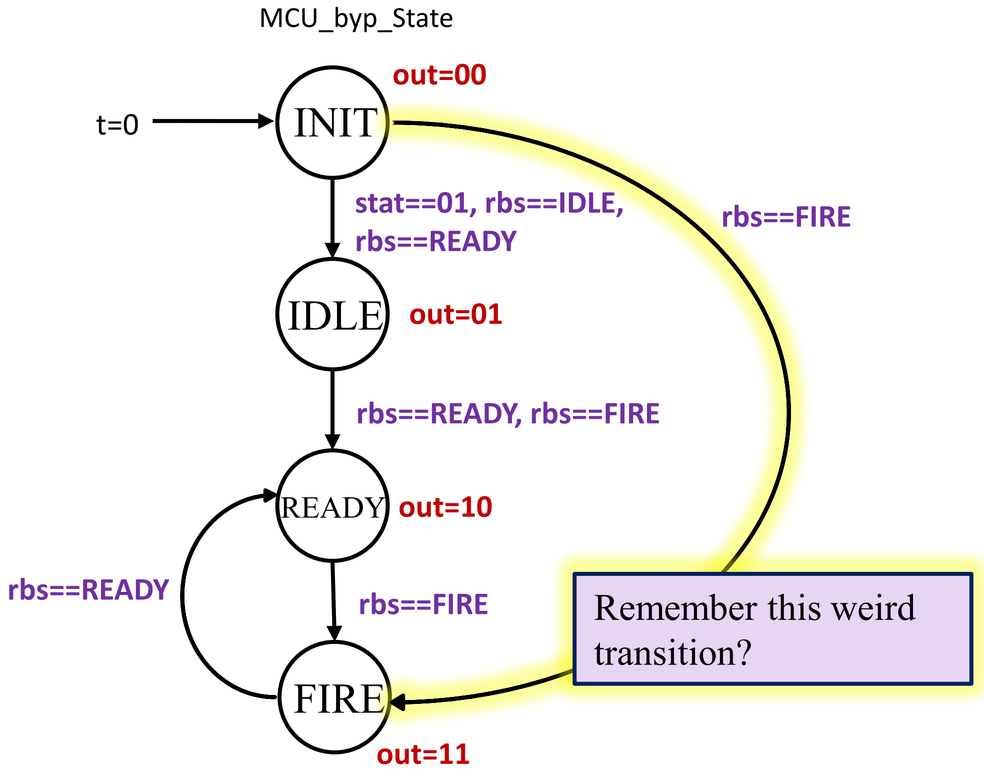
Environment (Simplified Model with Maintenance Support)

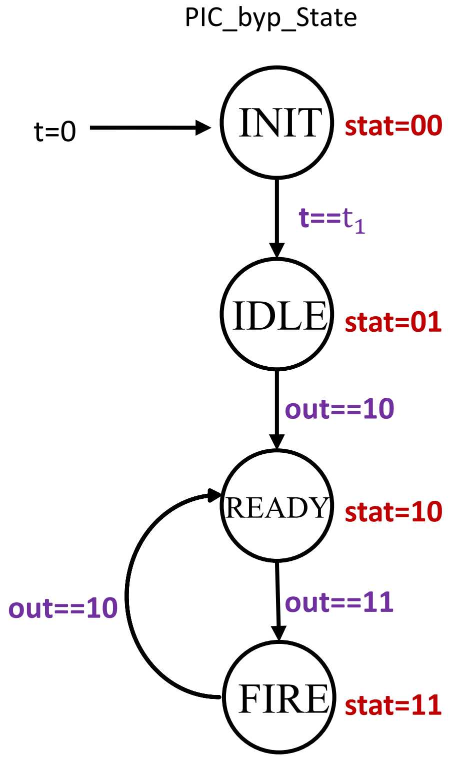
- “Before powering off the MCU, a flag, Setup_toBypassCommand, is set to True in the MCU”





Bypass State Machine Subsystem Overview





rbs: Request Bypass State

Model Checking with Spin

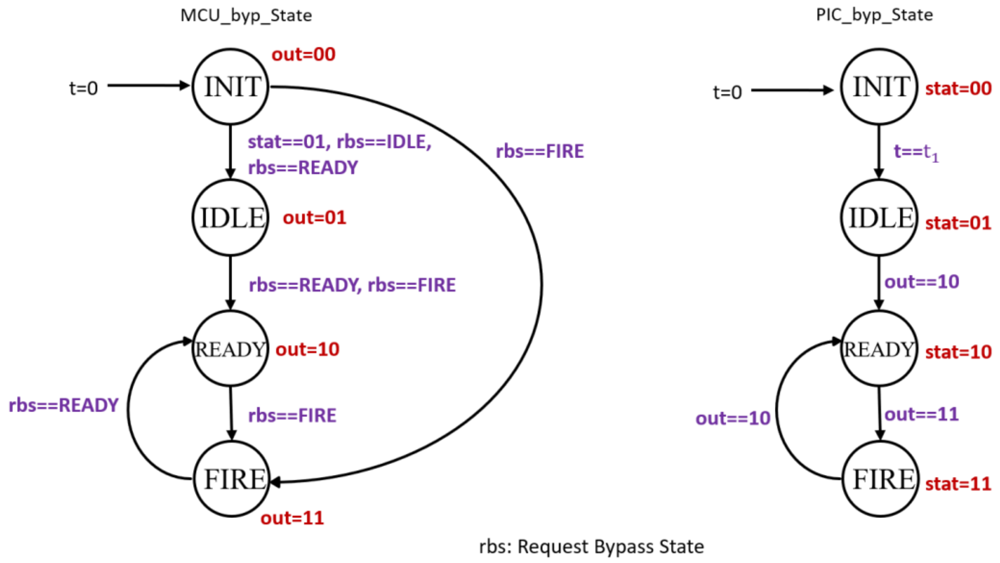
Model Checking with Spin

- Modeled the simplified environment FSM
- Modeled the MCU Bypass FSM
- Modeled the PIC Bypass FSM

- Modeled normal operation
- Modeled maintenance event

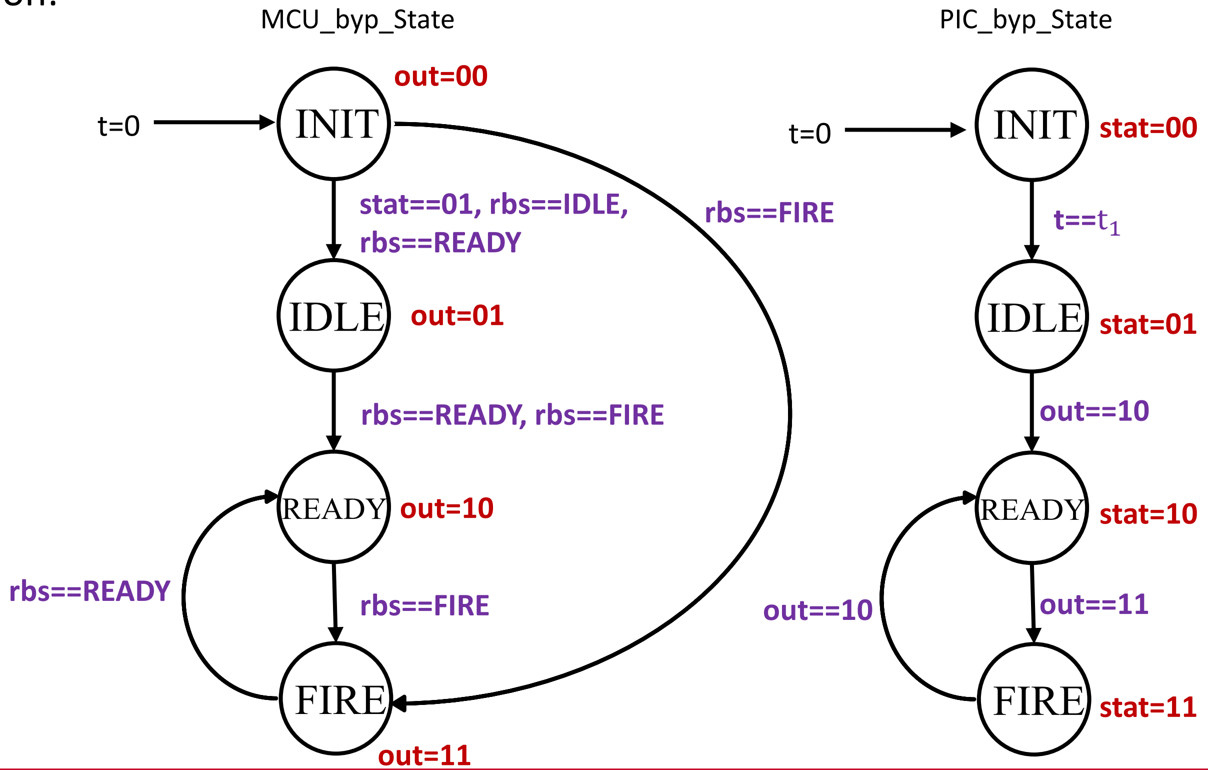
Startup in Normal Operation

- Performed as per specification:
  - Transition normal
  - Bypass activated/deactivated properly



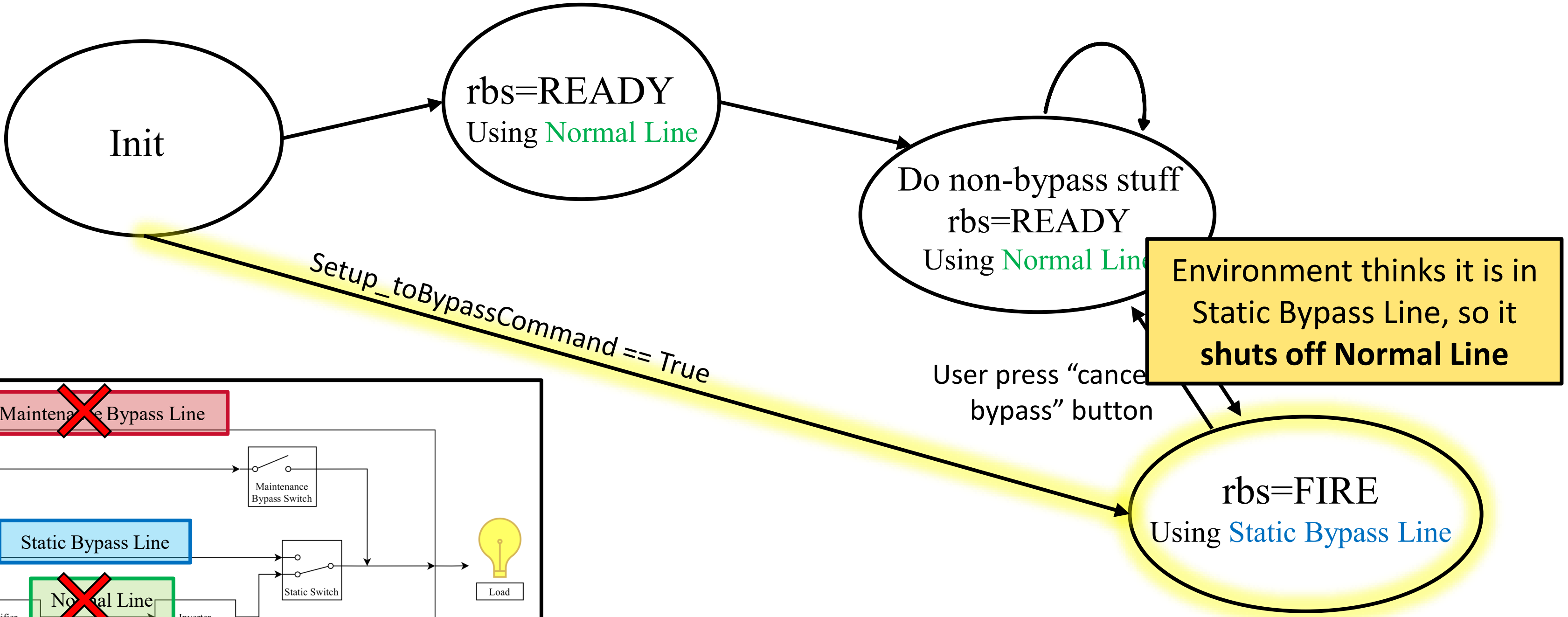
Startup after Maintenance Event

- Performed as per specification:

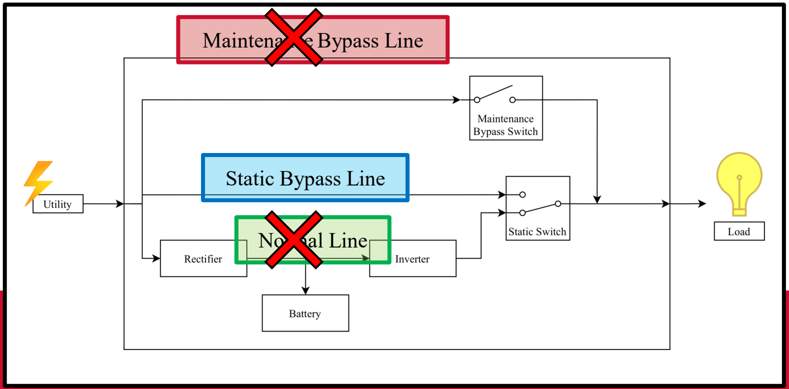


Startup after Maintenance Event

- Performed as per specification:

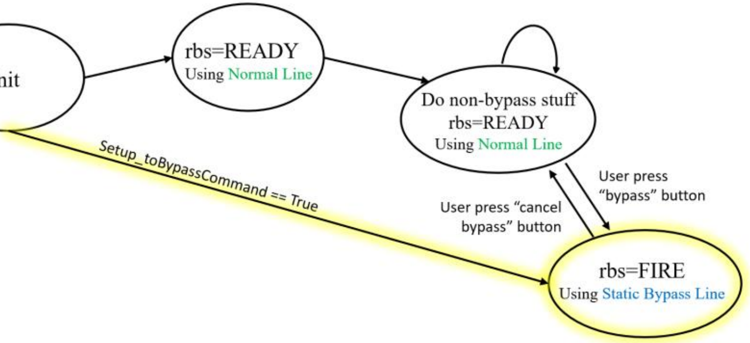


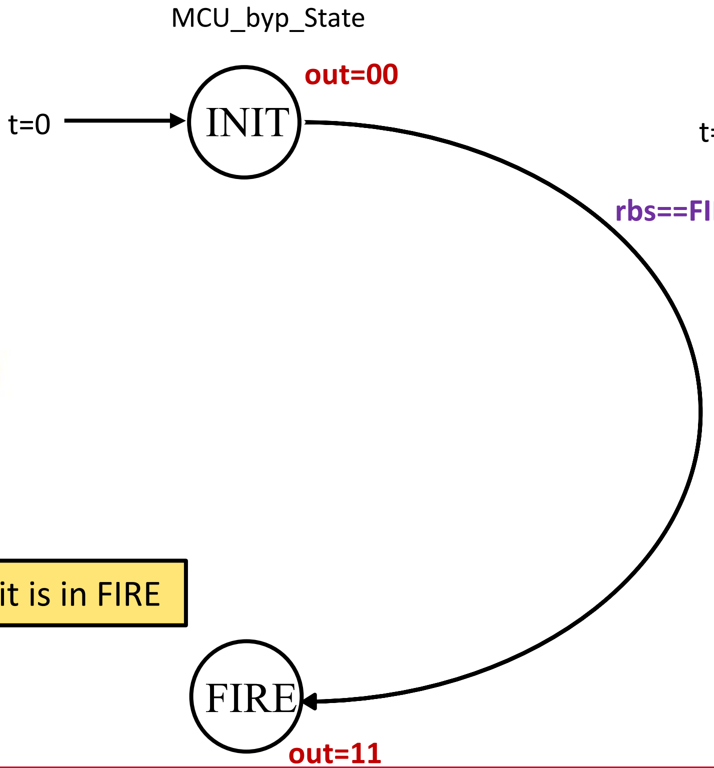
Environment thinks it is in Static Bypass Line, so it **shuts off Normal Line**

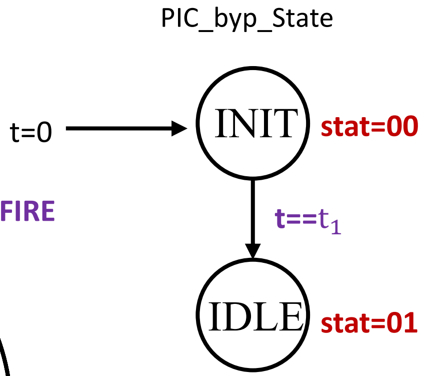


Startup after Maintenance Event

- Performed as per specification:



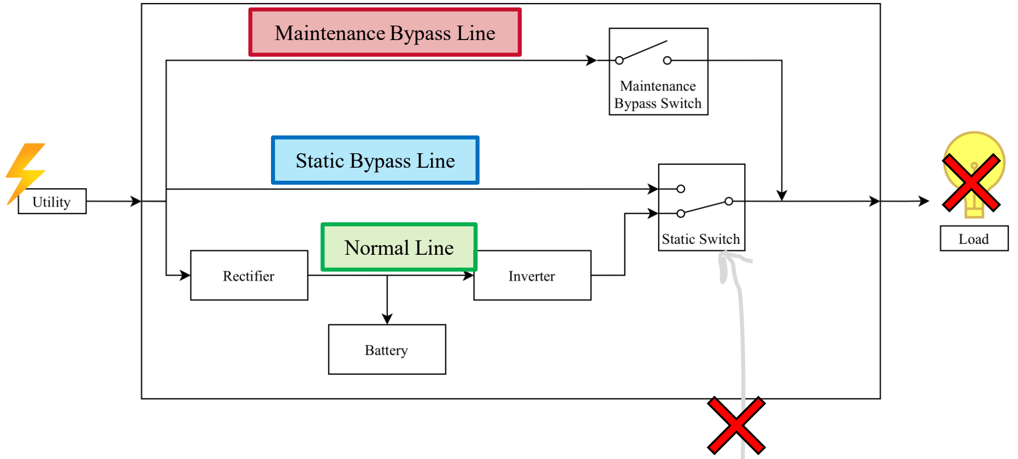


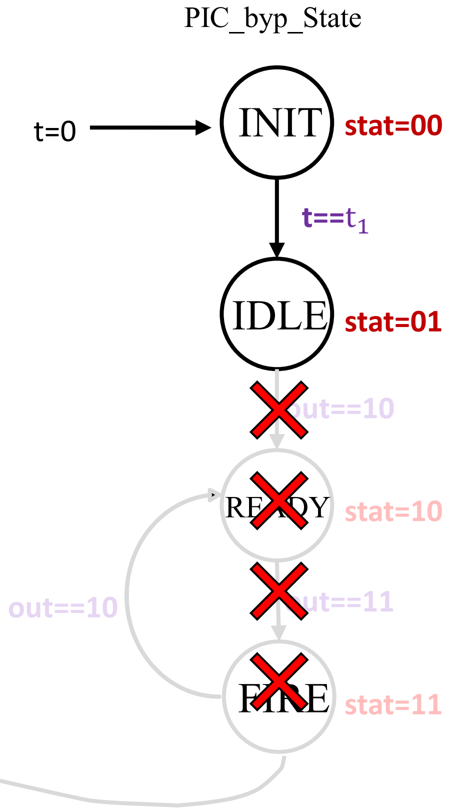


PIC most definitely is **not** in FIRE

MCU thinks it is in FIRE

Startup after Maintenance Event



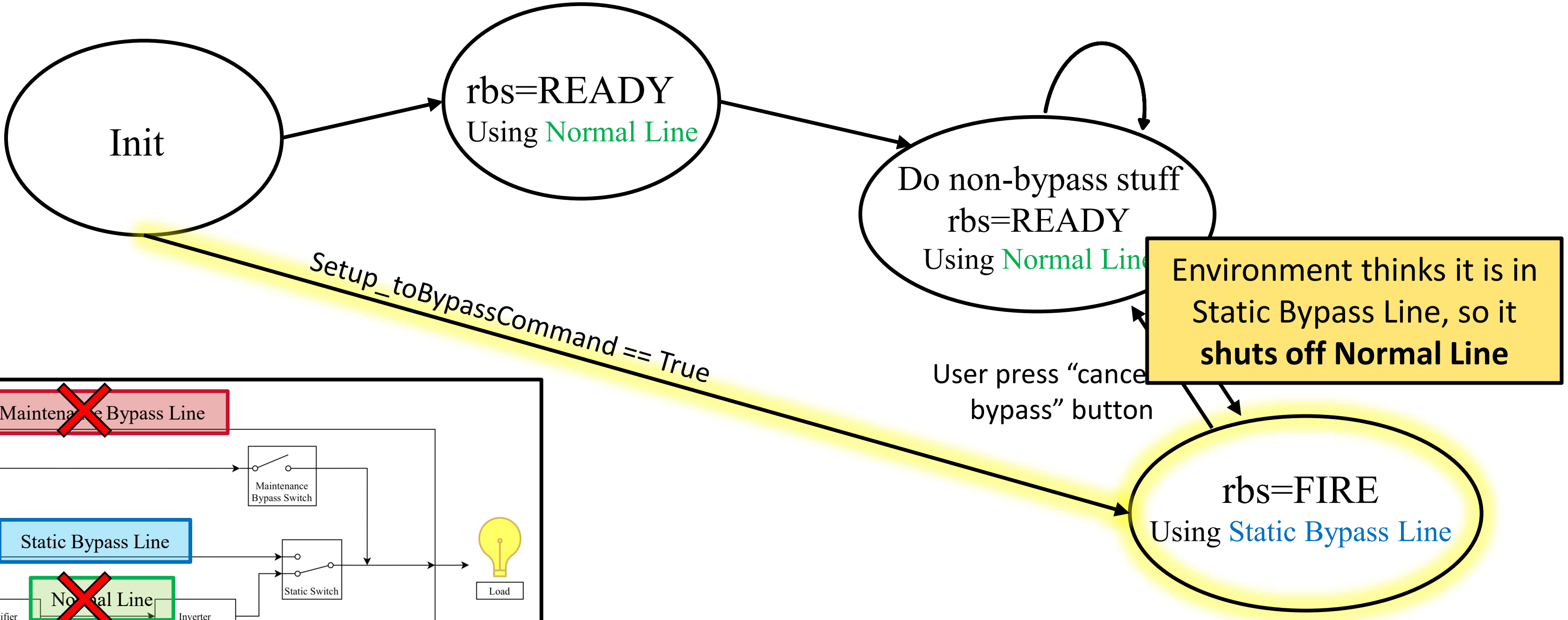


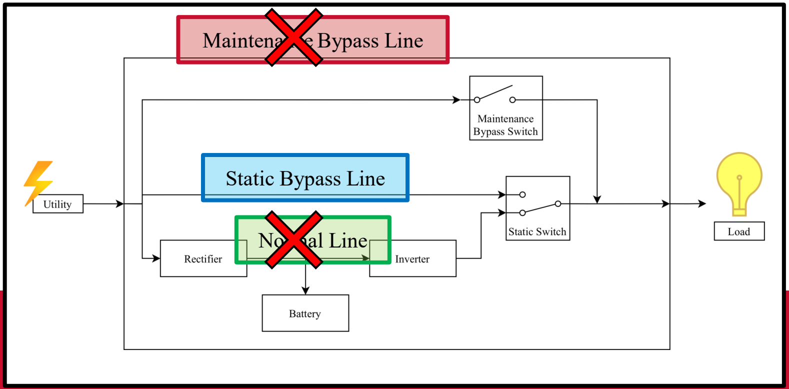
Reminder, the PIC's FIRE state controls the static switch

Spin Counterexample

```
6:    proc  0 (:init::1) bypass_state_machine.pml:399 (state 6)      [go_to_FIRE_asap = 1]
                go_to_FIRE_asap = 1
```

- Performed as per specification:


Init
rbs=READY
Using Normal Line
Do non-bypass stuff
rbs=READY
Using Normal Line
Setup_toBypassCommand == True
User press "cancel bypass" button
rbs=FIRE
Using Static Bypass Line
Environment thinks it is in Static Bypass Line, so it **shuts off Normal Line**

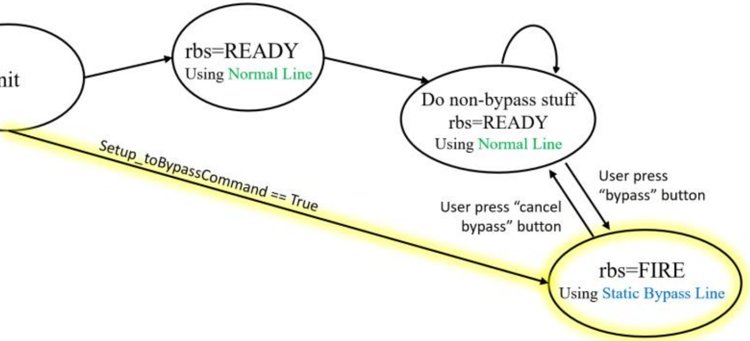


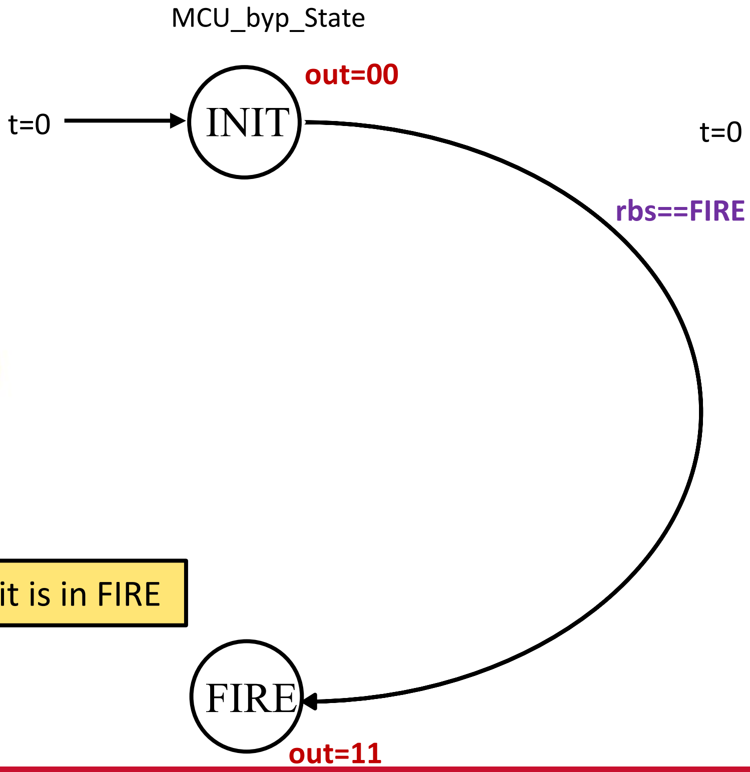
Spin Counterexample

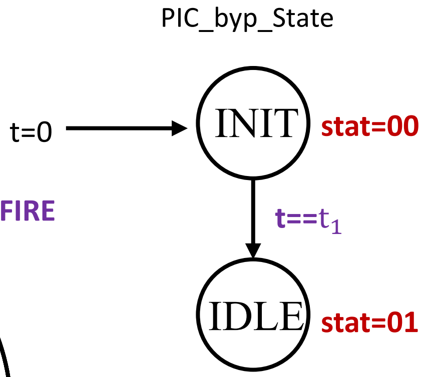
```
31:     proc  3 (PIC_byp_state_FSM:1) bypass_state_machine.pml:183 (state 8)    [PIC_byp_state = IDLE]
            PIC_byp_state = IDLE
```

```
24:     proc  2 (MCU_byp_State_FSM:1) bypass_state_machine.pml:91 (state 9)     [MCU_byp_State = FIRE]
            MCU_byp_State = FIRE
```

- Performed as per specification:





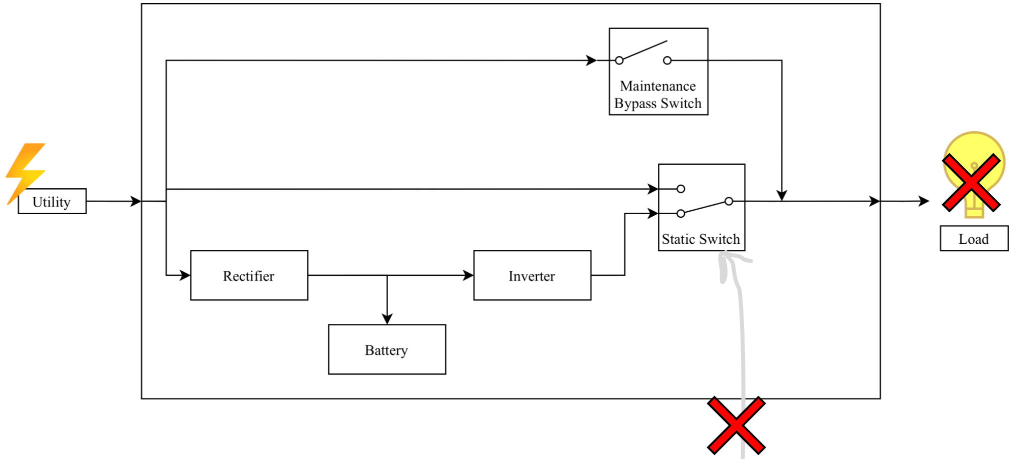


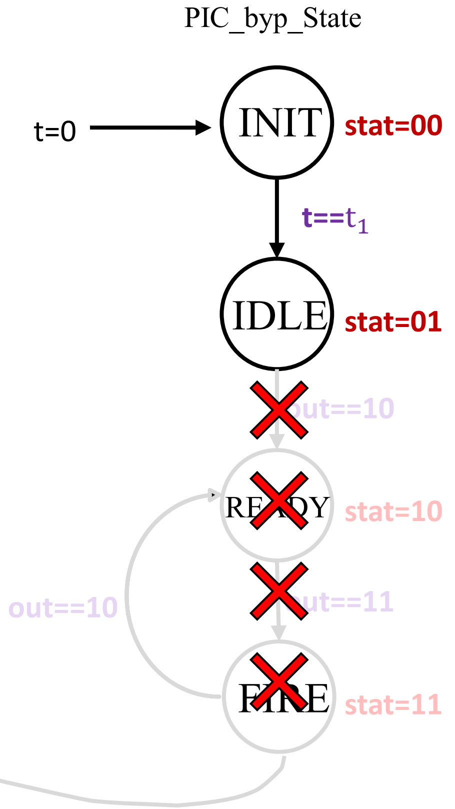
PIC most definitely is **not** in FIRE

MCU thinks it is in FIRE

Spin Counterexample

```
spin: bypass_state_machine.pml:360, Error: assertion violated
spin: text of failed assertion: assert(load_powered)
```





Reminder, the PIC's FIRE state controls the static switch

Model Checking with Spin
Is it correct?

Is it correct?

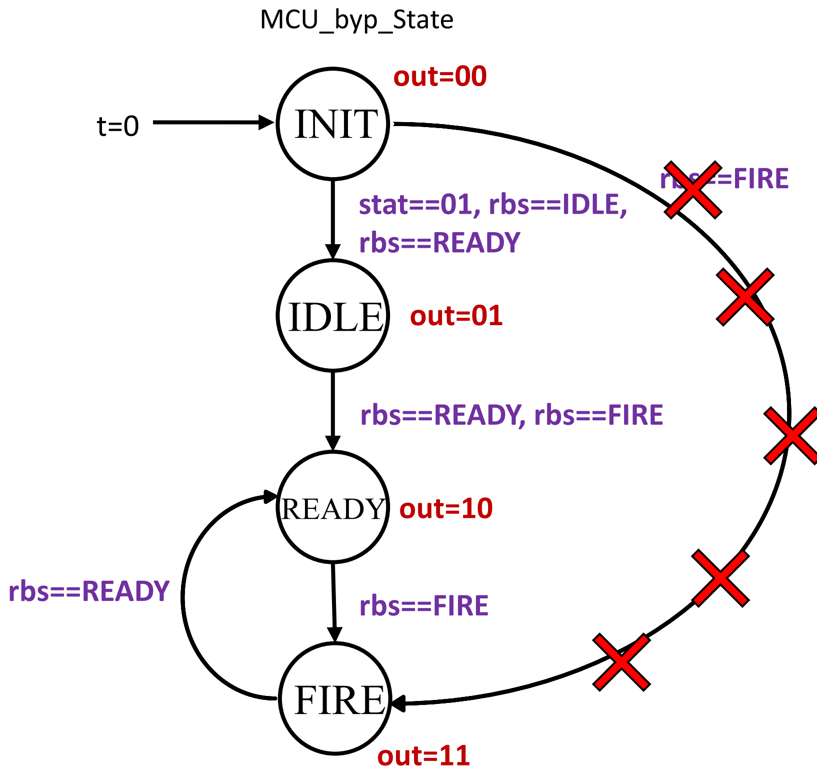
- As far as I can tell, the model accurately models the bypass subsystem
- Very lucky to get it first try… maybe because the model is simple enough

Interesting Facts about My Spin Experience

- Modeling capacitor discharge
  - In a real electrical system, capacitor creates a buffer between state change
  - Modeled using a counter (a state change has to persist for some amount of time before its effect is registered)
- Modeling random user control
  - Modeled using Promela's non-deterministic if statements

What was the fix?

1. Remove the **rbs==FIRE** skip
2. Use other mechanism in the environment to gracefully push MCU_byp_State to FIRE



Conclusion

If 20+ years ago, the designers have model checked their specs, will they have caught the bug?

Yes!