Control System Training

MODULE 7 – Finite State Machines

Copyright Notice

Finite State Machine - Definition

- **A logic machine that can be in only one of a finite number of states at any one time.**

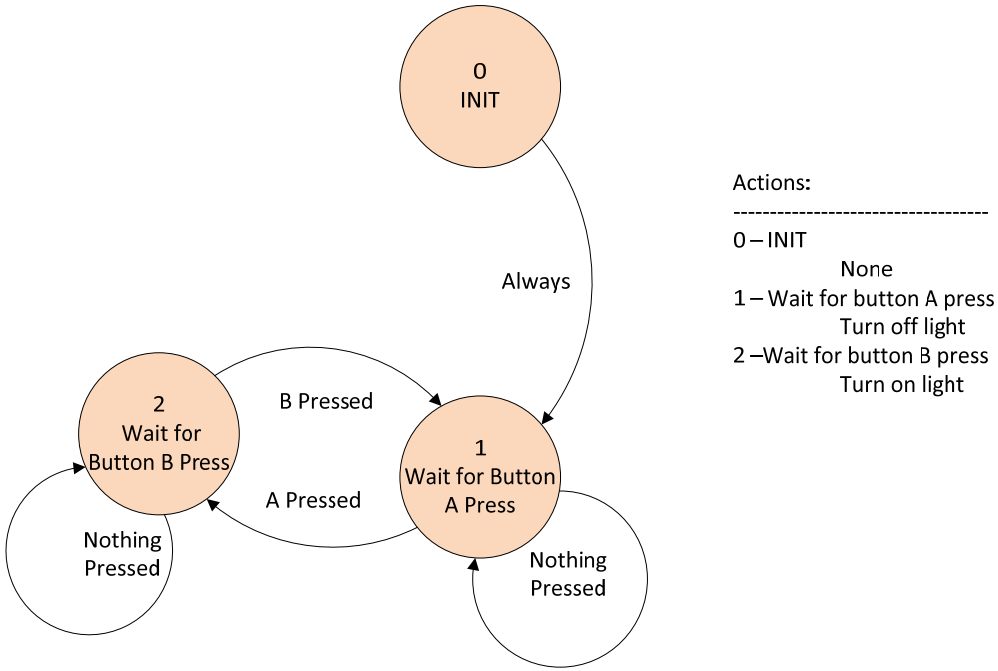
How the machine works

- **The machine loops forever, or until a state requests it to stop.**
- **During each loop the code for the "current" state is executed.**
- **For ROBOTS:**
  - Each robot control system "scan" cycle, 20 milliseconds, executes a single "state".
  - In some cases, this will mean that a 20 millisecond delay will be performed to get to the next state. Generally, 20 milliseconds is too small to be concerned with. If it is, then the state design should be reconsidered to avoid this delay.

What each “State” Does

- **Each state:**
  - Optionally does some work, performs, or initiates some action. The action could be:
    - Starting or stopping a motor
    - Opening or closing a solenoid
    - Setting a variable for another part of the code to use
    - A combination of things
  - Decides what the next state should be, and set the next state value.
    - Check various things to determine what next state should be.
    - Depending on the design, some states will not need any checks. They will know what state to set as next.
    - The next state could be the same state.

Drawing State Machines



Actions:
---
0 – INIT
None
1 – Wait for button A press
Turn off light
2 –Wait for button B press
Turn on light

- **Each State is a circle. Define them by unique name and/or state number.**
- **Lines to other states are transitions. Document what causes each transition to another state.**
- **Document actions in each state**

Selecting States

- **Start with an initialization State – State 0**
  - If somehow there is an error, and the state is set to zero, then this will cause initialization.
- **Have an error recovery state**
  - Recycle system, wait, and allow system to try again.
- **Keep states simple**
- **Keep states simple**
- **If possible join common sequences of actions together to reduce the number of states**

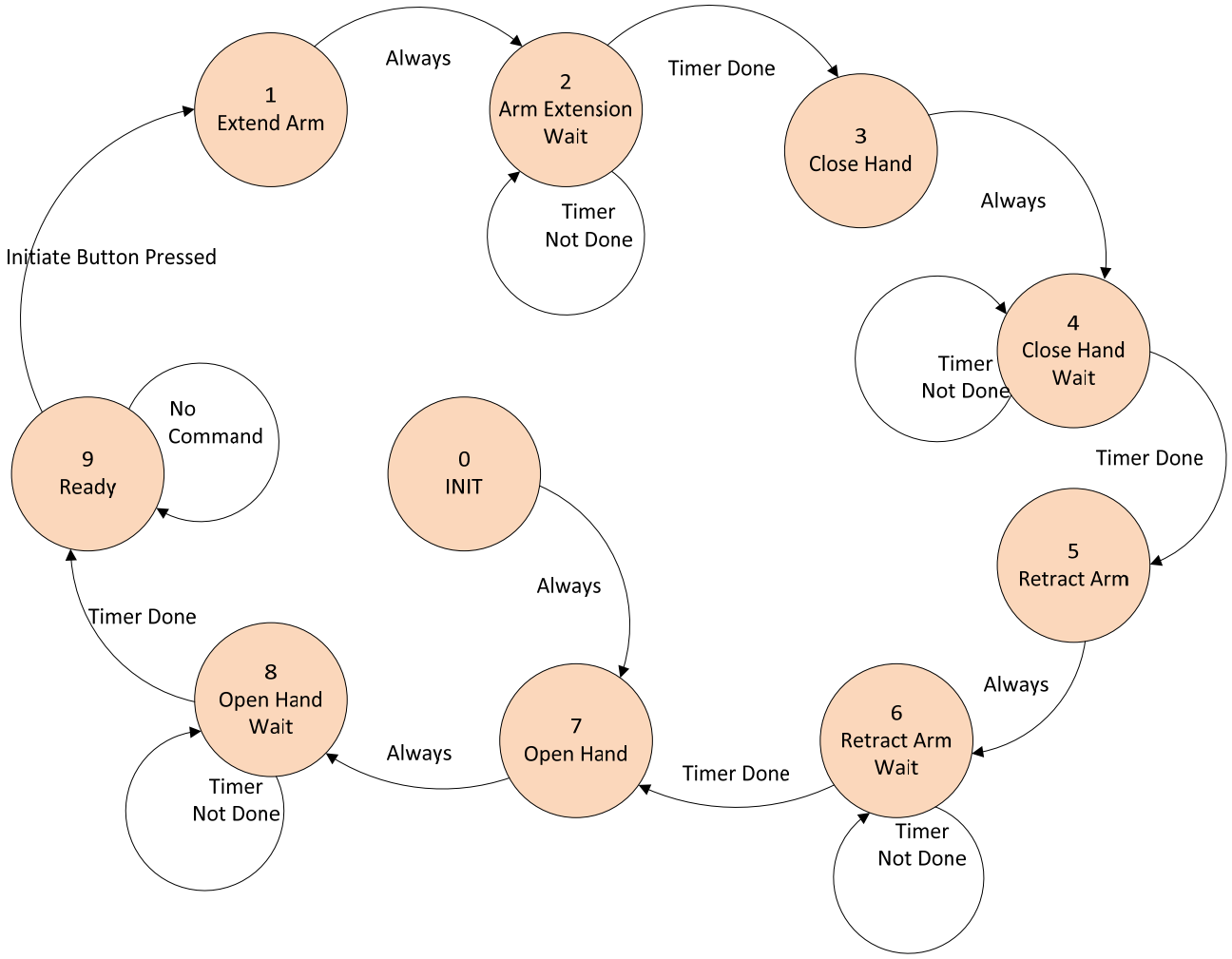
Designing Finite State Machine 1/3

- **Sample Problem – Cube Grab**
  - Control system uses a 20 msec loop time
  - System is ready when “hand” is opened and “arm” is retracted
  - Users pushes button to initiate “cube grab”. Auto repeat of cube grab is not allowed.
  - Ensure user pushed button for 60 msec
  - Close “arm” extension solenoid. Wait 1 second for arm to extend.
  - Close “hand” solenoid. Wait 2.0 seconds for “hand” to settle.
  - Open “arm” extension solenoid. Wait 1.3 seconds for arm to retract.
  - Open “hand” solenoid to release potential cube into bin. Wait 1.5 seconds for cube to drop before allowing next “grab” action.
- **Enhancement – Add a cancel button**

Designing Finite State Machine 2/3

- **Start with inputs**
- **Optionally draw a Logic Diagram Graph similar to sequential logic.**
- **Determine relationships**
  - What inputs does an output relate to
- **Draw State Diagram**
- **Define for Each State:**
  - Actions, if any
  - Tests to set next state
- **Repeat last two steps if needed to Finite State Machine**

Designing Finite State Machine 3/3



Actions:

---

0 – INIT
- Open Hand
- Retract Arm

1 – Extend Arm
- Extend Arm
- Initiate 1.0 Second Timer

2 – Arm Extension Wait
- None

3 – Close Hand
- Close Hand
- Initiate 2.0 second timer

4 – Close Hand Wait
- None

5 – Retract Arm
- Retract Arm
- Initiate 1.3 second timer

6 – Retract Arm Wait
- None

7 – Open Hand
- Open Hand
- Initiate 1.5 second timer

8 – Open Hand Wait
- None

9 – Ready
- None

Notes:

---

1) Initiate button on-delay, permissive checking, and edge triggering done outside of finite state machine. (This could be done either way.)
2) Cancel button would transition to 0-INIT state.

Sequential Logic or State Machine

- **Many problems can be solved either by sequential logic or state machines. How to choose between them:**
  - Simple machines might be implemented more easily with sequential logic.
  - More complex machines should use a finite state machine.
  - For machines that have to run continuously, use a finite state machine.

Exercise 7.1 - List Programming Objects

- **Create a list of potential objects from this chapter to program.**
  - List the inputs and outputs for each object.
- **HINT – This may be one of the very few places, that abstract and / or inherited classes are useful so that each uniquely defined state or state machine, follows the same pattern.**

Exercise 7.2 – Shoot frisbee

- **User pushes a button to shoot frisbee.**
- **Ensure user meant to push button. Button must be pressed for three cycles before initiating action. (Cycle time is 0.020 seconds).**
- **Can only shoot a frisbee if we have one. A limit switch indicates this. Also battery voltage must be > 11.5 volts. Can only shoot one frisbee at a time.**
- **Motors take 3 seconds to spin up to speed.**
- **Engage solenoid for 2 second to push frisbee into shooting wheel.**
- **Allow 2 more seconds for shooting to occur.**
- **After shooting is done, stop motor. (For now, don't allow continuous shooting.)**
- **Allow user to press a Cancel button. The cancel button must be pressed for at least 3 cycles before becoming active. After the Cancel, force a 5 second reset before allowing a new shot.**
- **It takes 5 seconds after shooting for a new frisbee to be in place ready to shoot.**
- **Design shooting logic. Also provide "ready to shoot" digital for dashboard display. Use ONLY a finite state machine, combined with combinatorial logic from Module 4 if needed. (It is possible to include Sequential Logic inside a Finite State Machine. For this exercise, don't do this.**