Multi-state Systems and Function Sequences

Multi-State System
Definition: A system that can exist in multiple states (one state at a time) and transition from one state to another

Characteristics
- A series of system states
- Each state requires one or more functions
- Rules exist to determine when to transition from one state to another
Typical Solution

- Every time tick, the system should check if it is time to transition to the next state
- When it is time to transition, appropriate control variables are updated to reflect the new state

Categories

- Timed Multi-state systems:
  - Transitions depend only on time
- Input-based multi-state system:
  - Transitions depend only on external input
  - Not commonly used due to danger of indefinite wait
- Input-based/Timed Multi-state systems:
  - Transitions depend both on external input and time

Example Timed System: Traffic Light

- States:
  - Red
  - Red-Amber
  - Green
  - Amber
Time Constants

#define RED_DURATION 20
#define RED_AND_AMBER_DURATION 5
#define GREEN_DURATION 30
#define AMBER_DURATION 5

State Update Code

switch (Light_state_G)
{
    case RED:
    {
        Red_light = ON;
        Amber_light = OFF;
        Green_light = OFF;
        if (++Time_in_state == RED_DURATION)
        {
            Light_state_G = RED_AND_AMBER;
            Time_in_state = 0;
        }
        break;
    }
}

Example Timed System: Robotic Dinosaur

- States:
**Input/Timed Systems**

- Two or more states
- Each state associated with one or more function calls
- Transition between states controlled by a combination of time and user input

**Solution Characteristics**

- System keeps track of time
- If a certain user input is detected, a state transition occurs
- If no input occurs for a pre-determined period, a state transition occurs

**Example Input/Timed System: Washing Machine**
Functional Description

1. The user selects a wash program (e.g., 'Normal', 'Delicate') on the selector dial.
2. The user presses the 'Start' button.
3. The door lock is engaged.
4. The water valve is opened to allow water into the wash drum.
5. If the wash program involves detergent, the detergent hatch is opened. When the detergent has been added, the detergent hatch is closed.
6. When the 'fill water level' is sensed, the water valve is closed.
7. If the wash program involves warm water, the water heater is switched on. When the water reaches the correct temperature, the water heater is switched off.
8. The wash cycle is started to rotate the drums. The motor then goes through a series of movements, both forward and reverse, at various speeds to wash the clothes. (The precise set of movements carried out depends on the wash program that the user has selected.) At the end of the wash cycle, the motor is stopped.
9. The pump is switched on to drain the drum. When the drum is empty, the pump is switched off.

Sample System Update Code

case HEAT WATER:
{
    // For demo purposes only
    DEBUG_PORT = (BYTE) System state 0;
    // Remain in this state until water is hot
    if (Time_in_state.G >= MAX_WATER_HEAT_DURATION)
    {
        // Should have warmed the water by now...
        System state G = ERROR;
    }
    // Check the water temperature
    if (WATER_In_Water_Temperature() == 1)
    {
        // Water is at required temperature
        // Ready to go to next state
        System state G = WASH_01;
        Time_in_state G = 0;
    }
    break;
}