



Advanced Agricultural Electronics Networks Development

Lecture No. 8, Behavioral Modeling and Operating Modes

BAE 5030-352

Spring 2010

Dr. Marvin Stone

Biosystems and Agricultural Engineering
Oklahoma State University

Behavioral Modeling

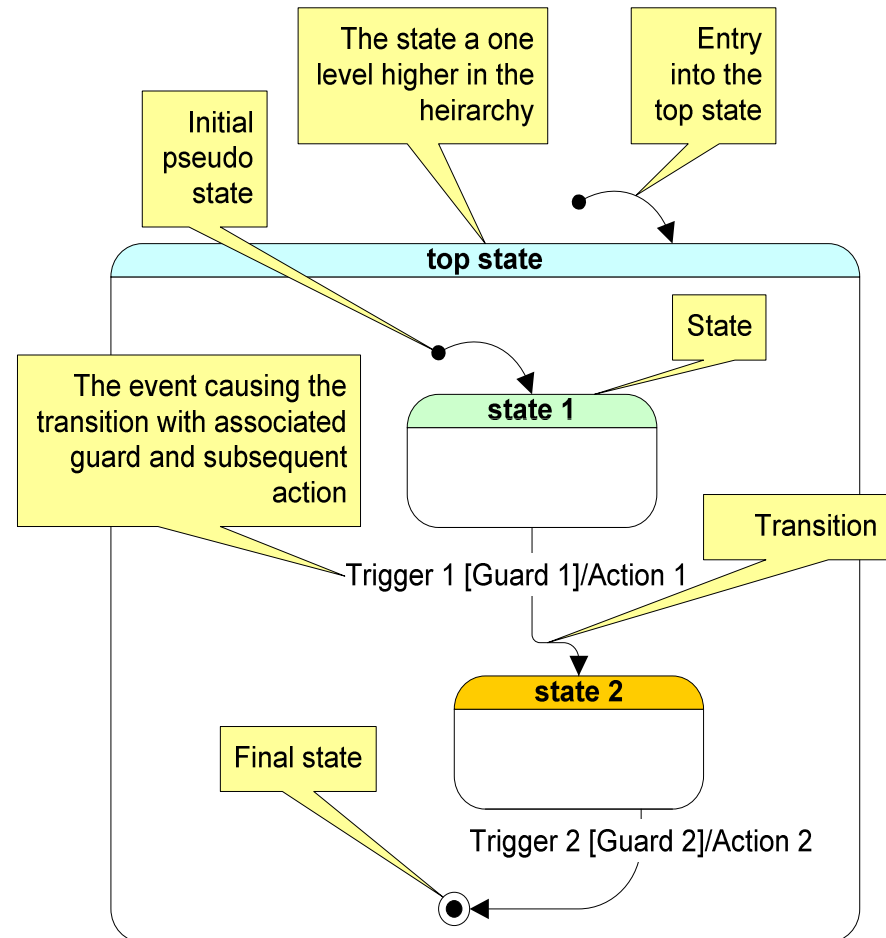
- Logic tables – a tabular relationship between all inputs and outputs. Inputs and outputs can include parameters in memory
- Statecharts – modeling technique suited to describing systems where past state or memory defines the transitions to the next state
 - Mealy automata – processes occur on transitions between states
 - Moore automata – processes occur within a state
 - UML - Unified Modeling Language – specifies a syntax for state machine modeling

Statechart definitions

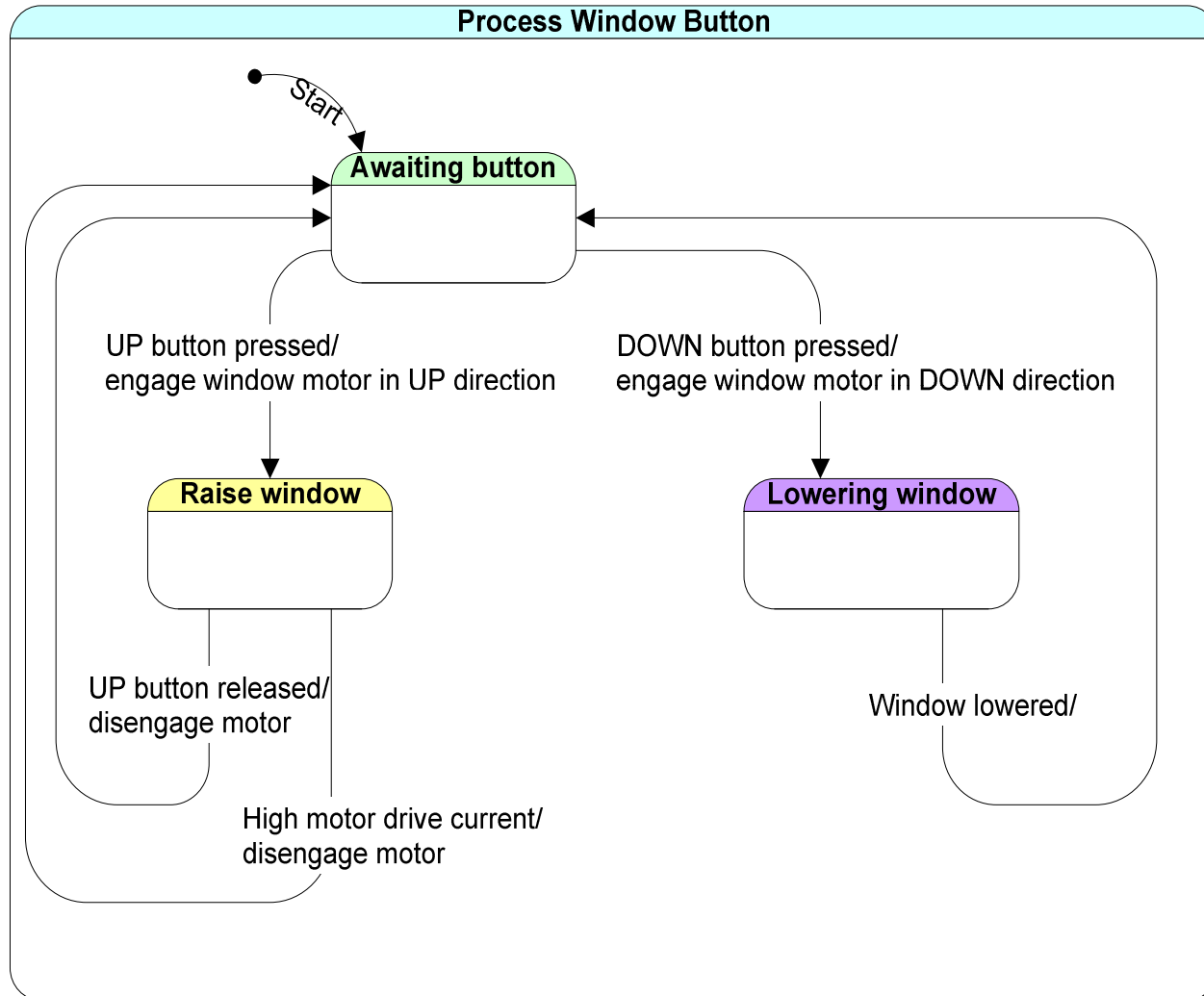
- *State* - a distinguishable consistent characteristic behavior of a system that persists for a significant period of time
- *Event* – a change in stimulus to a system that occurs in an infinitesimal period of time
- *Transitions* - responses to events that move the system from state to state
- *Finite State Machine* – A model of a system and its behavior that captures states, events, and transitions between states
- Finite State Machine characteristics
 - Characteristic behavior within a *state* is distinct and unchanging
 - Finite number of *states*
 - Residence time in *states* is a significant period
 - *Transitions* are the response of the system to *events*
 - *Transitions* between *states* are distinct and unchanging
 - *Transitions* between *states* are finite in number
 - *Transition time* between *states* is infinitesimal

Statechart elements and syntax

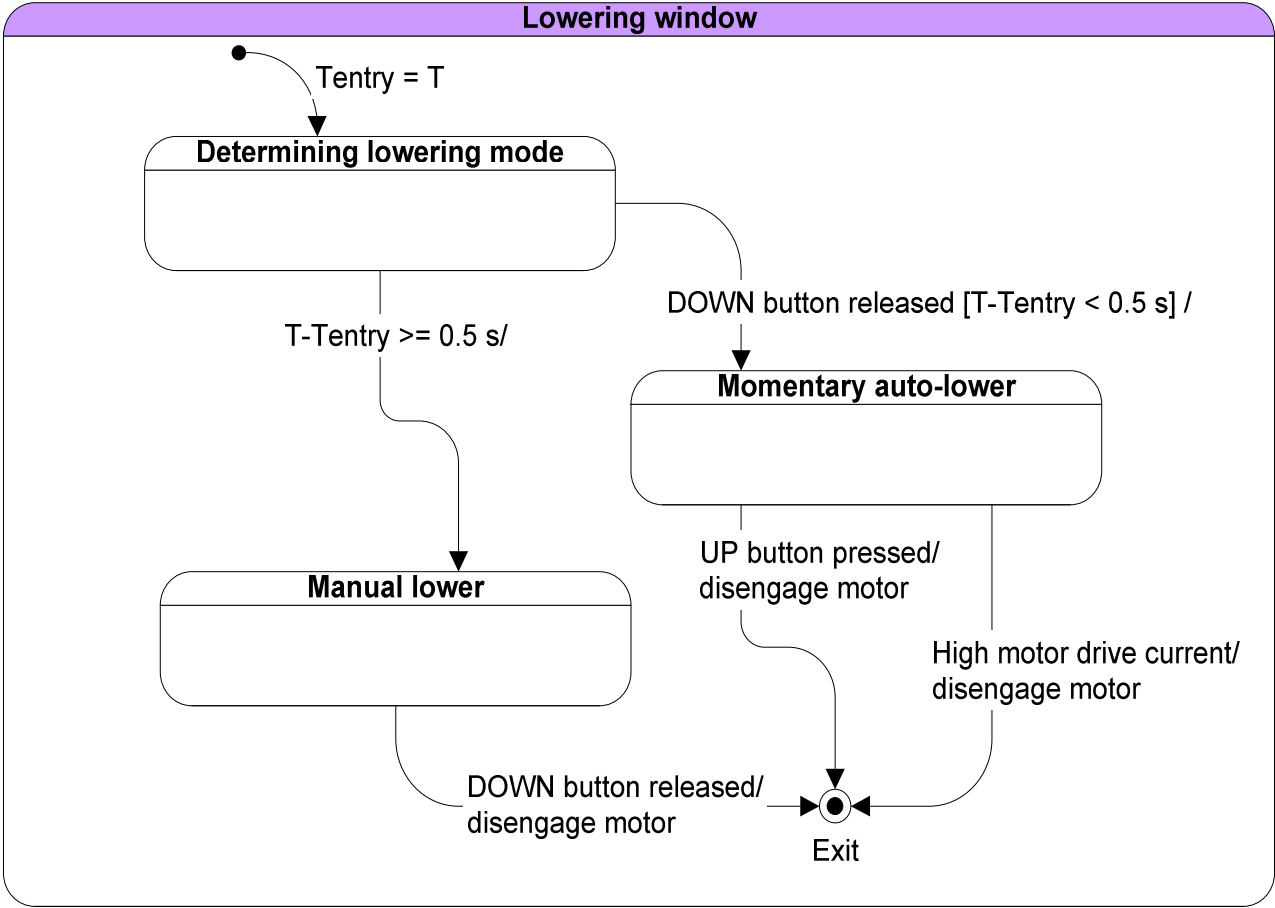
- Start occurs at the initial pseudo state
- The trigger event causes the change of state if the guard condition is true
- System may exit the top state when the final state is reached
- Time between states is infinitesimal
- Elements (Simple statecharts)
 - Initial state
 - State
 - Transition
 - Event
 - Trigger
 - » Guard
 - Action
 - Final state



Statechart Example

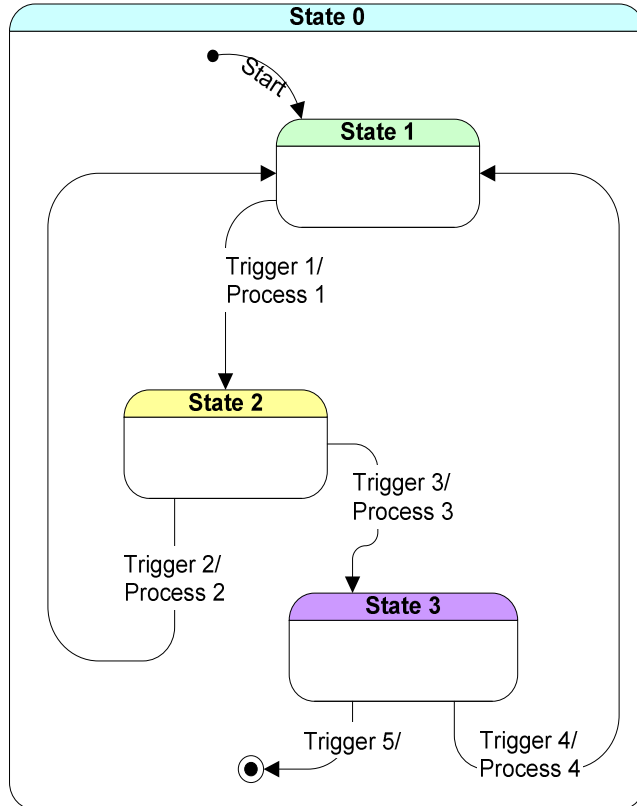


Hierarchical State Example



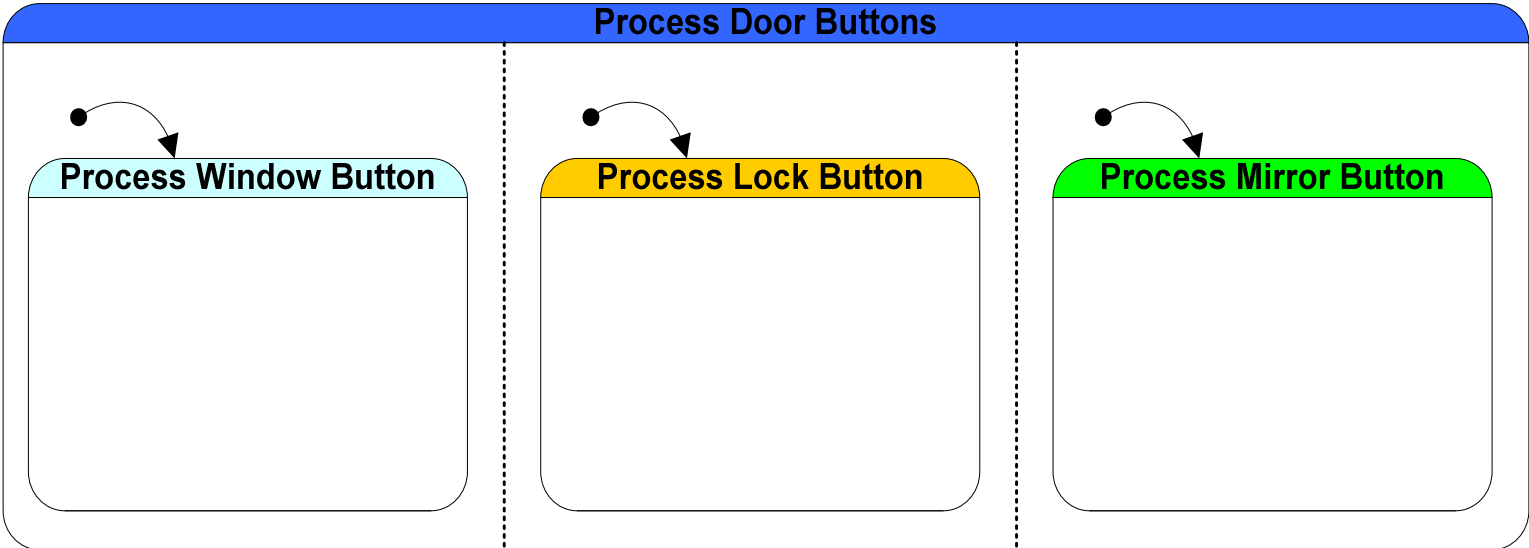
Software structure to realize a state machine

- Many possibilities but typical is infinite loop encompassing a case structure. (see Samek)
- Consider the following example:



```
void Process_State_0( void )
{
  state = STATE_1
  while(!Trigger_5 & (state == STATE_3)){
    switch (state) {
      case STATE_1:
        if (Trigger_1) {
          process_1();
          state = STATE_2
        }
        break;
      case STATE_2:
        if (Trigger_2) {
          process_2();
          state = STATE_2
        }
        if (Trigger_3) {
          process_3();
          state = STATE_2
        }
        break;
      case STATE_3:
        if (Trigger_4) {
          process_4();
          state = STATE_1
        }
    }
  }
}
```

Concurrent tasks

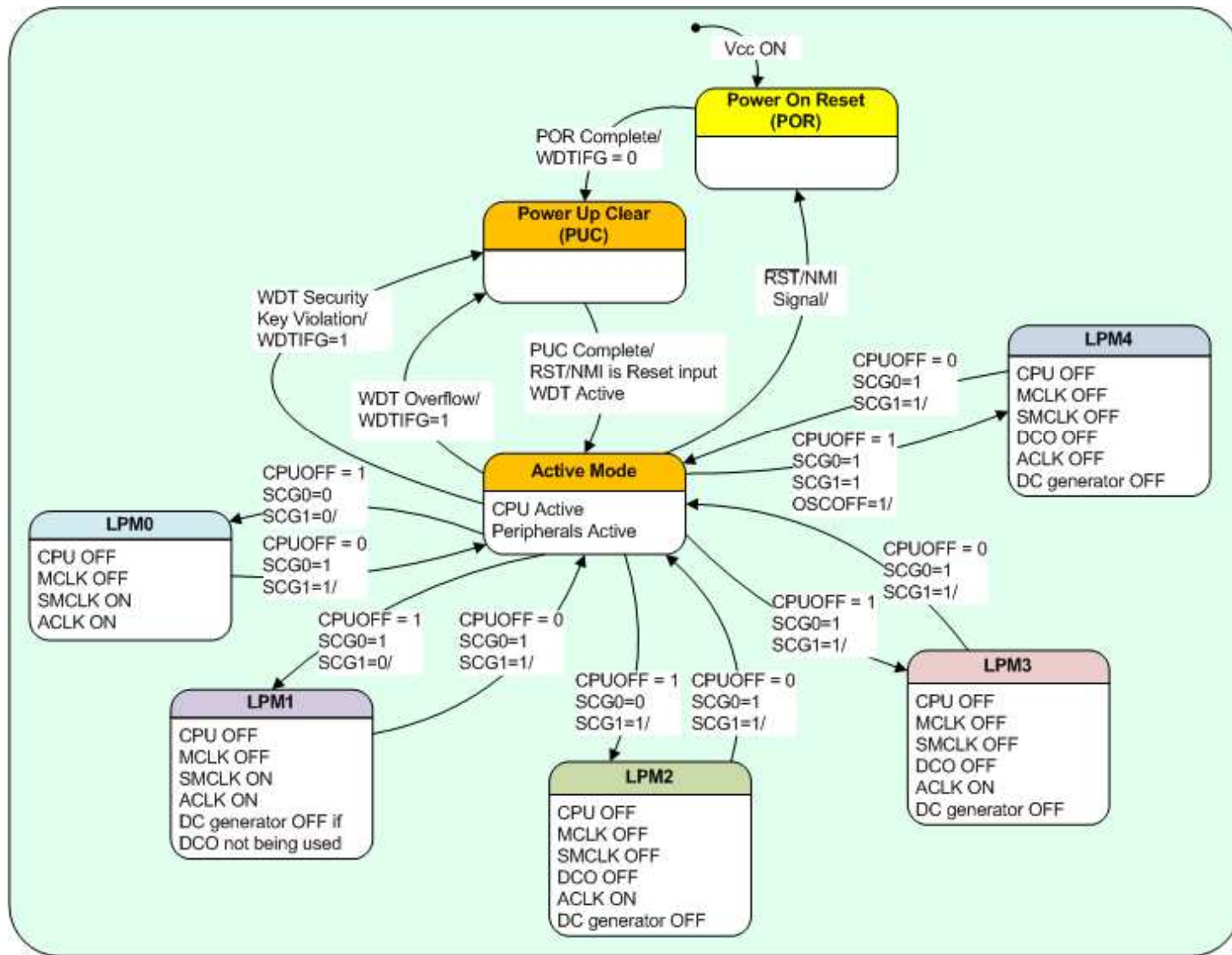


CPU and Clock Control Logic Table

Status Register Bits

SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled SMCLK, ACLK are active
0	1	0	1	LPM1	CPU, MCLK, DCO osc. are disabled DC generator is disabled if the DCO is not used for MCLK or SMCLK in active mode SMCLK, ACLK are active
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator remains enabled ACLK is active
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator disabled ACLK is active
1	1	1	1	LPM4	CPU and all clocks disabled

Power Modes State Chart



Setting SR bits in the C compiler

```
#include <msp430.h>

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    BCSCCTL2 |= DIVS_3;                 // SMCLK = DCO div 8
    TACTL = TASSEL_2+MC_2+TAIE+ID0;     // SMCLK selected, continuous timer mode,
                                        // Timer interrupt enabled, SMCLK div by 2
                                        // P1.0 output

    P1DIR |= 0x01;

    _BIS_SR(LPM0_bits + GIE);          // Enter LPM0 with general interrupt enable
}

#pragma vector=TIMER_A1_VECTOR        // Timer_A2 Interrupt Vector (TAIV) handler
__interrupt void Timer_A(void)        // Select to react to the timer A interrupt
{
    switch( TAIV )                    //Test the Timer A return value, TAIV
    {
        case TAIV_TACCR1: break;      // CCR1 case not handled
        case TAIV_TAIFG: P1OUT ^= 0x01; // Timer A overflow case
        break;
    }
}
```

Outline to explore the SR Modification process

- Review
 - MSP430x2xx User Guide
 - 2.3.1 Entering and exiting LP modes
 - Test8 code
 - #Pragma Vector (pragmatic or implementation-specific information)
 - Compiler reference manual “pragma directives”
 - __BIS_SR() in:
 - in430.h
 - Compiler reference manual “intrinsics”
 - LPM0_bits and GIE in:
 - MSP430F2013.h