**LancingHSM State Pseudo Code**

**Key:**

Capital Case used for variables and function names (Ed’s Convention)

/\* comments \*/

// place holder for code

Only code needed for lowest level state (During func)

Write Durings for everything, but comment out the function call if not needed

**Start LanchingHSMS (CurrentEvent)**

// Handle history or irregular entry here

If not history entry

CurrentState = RetractingLance;

End

/\* Call ENTRY function for LanchingHSMS \*/

Run LanchingHSMS(CurrentEvent)

End Start

**LanchingHSMS\_t Run LanchingHSMS (CurrentEvent)**

Init MakeTransition, NextState, EntryEventKind, ReturnEvent

/\* React differently based on the state you’re in \*/

Switch (CurrentState)

Case RetractingLance:

/\* Execute During to respond to event \*/

CurrentEvent = DuringRetractingLance (CurrentEvent)

/\* Handle all events relevant to RetractingLance \*/

If ES\_NO\_EVENT does not == CurrentEvent

Switch (CurrentEvent)

Case Retract Timeout

NextState = DeployingLance;

MakeTransition = True;

EntryEventKind = EV\_ENTRY;

ReturnEvent = CurrentEvent;

Break;

End switch

End if

Break;

Case DeployingLance:

/\* Execute During to respond to event \*/

CurrentEvent = DuringDeployingLance (CurrentEvent)

/\* Handle all events relevant to DeployingLance \*/

If ES\_NO\_EVENT does not == CurrentEvent

Switch (CurrentEvent)

Case Deploy Timeout

NextState = RetractingLance;

MakeTransition = True;

EntryEventKind = EV\_ENTRY;

ReturnEvent = CurrentEvent;

Break;

End switch

End if

Break;

End switch

/\* Handle transitions between states \*/

If (MakeTransition)

/\* Exit this state \*/

CurrentEvent.EventType = EV\_EXIT;

Run LancingHSMS(CurrentEvent)

/\* Enter next state \*/

CurrentState = NextState; // Change to next state

CurrentEvent.EventType = EventEntryKind;

Run LancingHSMS(CurrentEvent)

End if

Return ReturnEvent

End Run

**LanchingHSMS\_t Query LanchingHSMS (CurrentEvent)**

Return CurrentState;

End Query

**During RetractingLance (CurrentEvent)**

Init ReturnEvent

If (EV\_ENTRY == CurrentEvent)

// Execute Entry actions here

Start Retract Timer

/\* No lower level SM(s) \*/

Else if (EV\_ExIt == CurrentEvent)

/\* No lower level SM(s) \*/

// Execute Exit actions here

Lower Lance

Else

/\* No lower level SM(s) \*/

// Execute any internal event response here

End if

return ReturnEvent

End During

**During DeployingLance (CurrentEvent)**

Init ReturnEvent

If (EV\_ENTRY == CurrentEvent)

// Execute Entry actions here

Start Deploy Timer

/\* No lower level SM(s) \*/

Else if (EV\_ExIt == CurrentEvent)

/\* No lower level SM(s) \*/

// Execute Exit actions here

Raise Lance

Else

/\* No lower level SM(s) \*/

// Execute any internal event response here

End if

return ReturnEvent

End During