

Objective

To explain the usage of the LINK State Machine 2 function block.

Equipment

DSD (or ConfigEd) software

Introduction

This application note focuses on the State Machine 2 function block.

A state machine uses input events to determine the machine's 'state' that causes an output(s) to change. It can simplify sequential functions in numerous situations. LINK implements a state machine using three basic function blocks: Logic-To-Event, State Machine 2, and Lookup. Additional LINK function blocks can be added as needed to meet the design.

Events

Any input condition change is an event. Typically, these events cause output(s) to change. State machine discriminates these events by their assigned "event number." A Logic-To-Event function block supplies the event number.

States

States are referred to as modes or conditions of a machine. For example, a rewind could be in a stop condition, speed mode or tension mode. See Figure1 - Spindle A States 1, 5, and 2. The State Machine 2 function block determines the state by testing that all the input events for a particular state are true.

Outputs

Outputs are the settings (setpoints, etc.) the machine must get when it enters a state. A Lookup function block produces the output values.

State Transitional Diagram (Mapping)

Before creating the software configuration, you need to map the logical state scheme (State Transitional diagram) and determine the number of different events and states.

To design a state machine control, start by drawing a State Transitional diagram that defines:

1. Every state in which this machine can exist.
2. All the input events required to transition the machine for each state.
3. The outputs which the machine controls in each state.

Assign unique, non-zero, ordinal numbers to each event and state (these numbers cannot be larger than 65535). For each state, specify the event(s) that acts on it. For each significant event, specify the following transitional information:

1. The new destination state
2. Output status.

If you have questions, please call the Product Support Group at (704) 588-3246.

Example

The following example shows a basic State Transitional diagram for a dual turret winder.

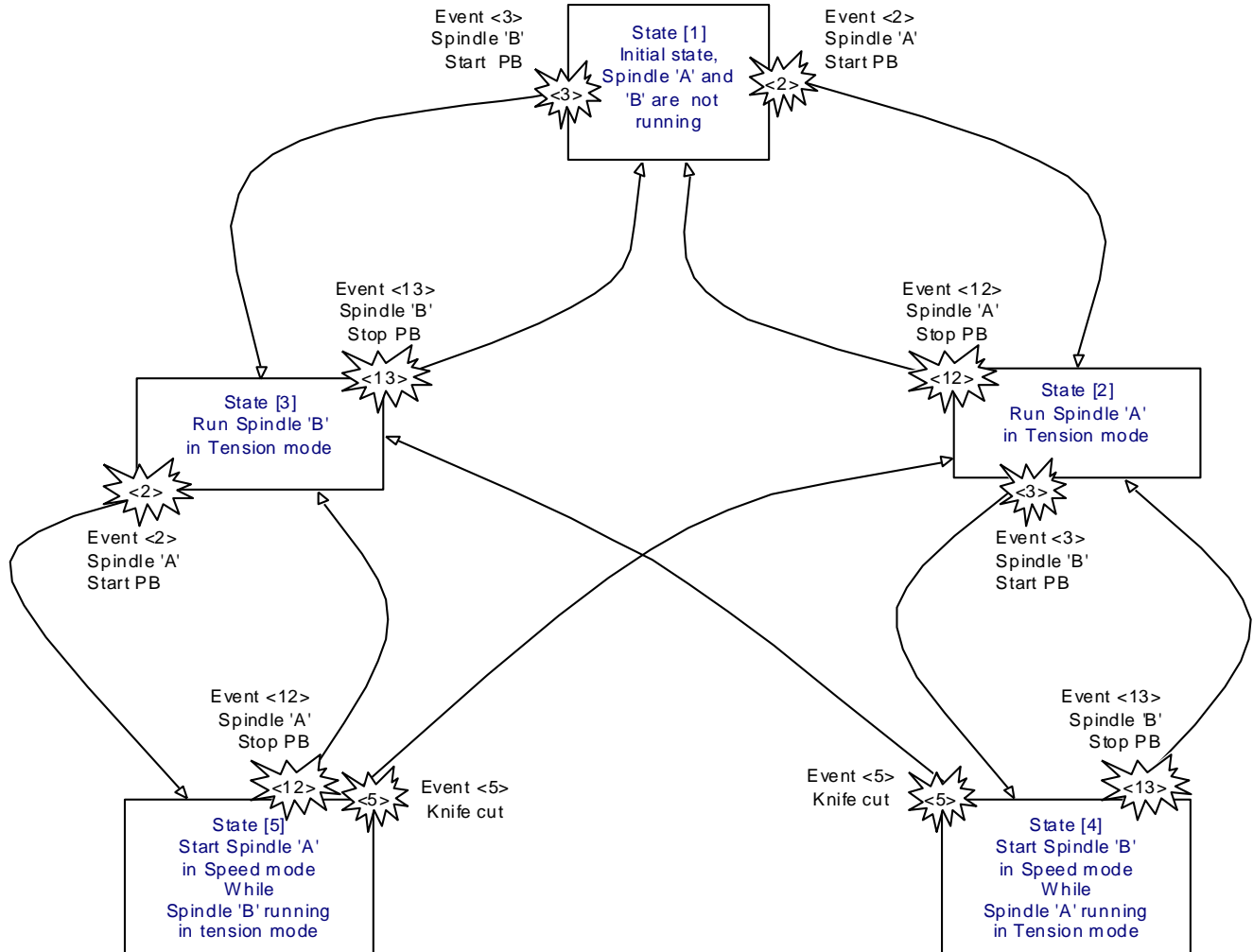


Figure 1- Typical State Transitional Diagram for Dual Turret Winder

If you have questions, please call the Product Support Group at (704) 588-3246.

The State Transitional diagram (Figure 1) reads as follows.

State [1]
Initial state,
Spindle 'A' and
'B' are not
running

State 1 - when the machine is in this state (the initial state) both spindles are at rest and not running.



Event 2 - pushing the "Spindle 'A' Start" pushbutton results in Event 2, which causes the machine to change to State 2.

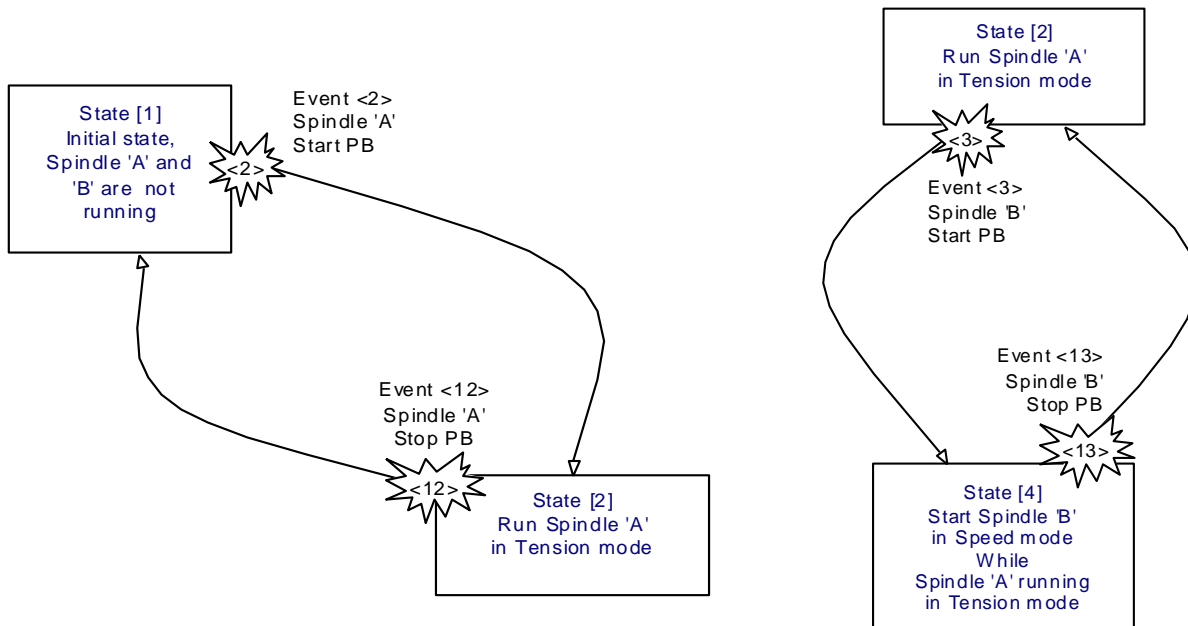
State [2]
Run Spindle 'A'
in Tension mode

State 2 - the Winder Spindle 'A' runs in tension mode.



Event 12- pushing the "Spindle 'B' Stop" pushbutton results in Event 12, which causes the machine to switch back to State 1.

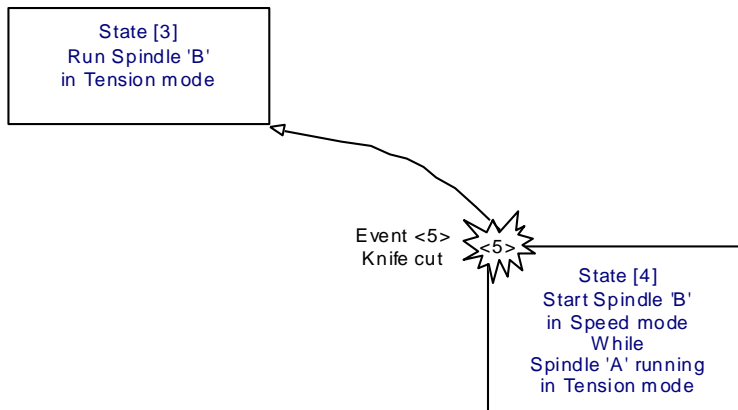
So far this state machine acts like a complicated "set/reset" or "latch" switch. As the logic gets more complex, the simplicity of State Machine becomes more apparent.



The Spindle 'B' logic is similar to Spindle 'A.' Only the event and state numbers are different.

If you have questions, please call the Product Support Group at (704) 588-3246.

Speed Match - When in State 2, pushing the “Spindle ‘B’ Start” pushbutton (Event 3) causes state machine to move to State 4, which sets Spindle ‘B’ to run in speed mode. Pressing the “Spindle ‘B’ Stop” pushbutton (Event 13) causes state machine to go back to State 2.



Web Transfer - “Knife cut” produces Event 5, which moves the machine to State 3. In this state, Spindle ‘A’ is turned off and Spindle ‘B’ is switched to tension mode.

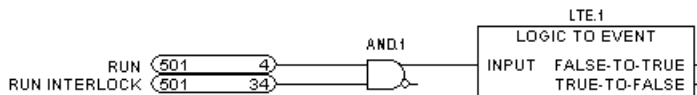
Creating LINK Configuration

After designing a State Transitional diagram, it must be converted to LINK. The following LINK function blocks are used. The state machine function block will be referred to as SM2 in this note.

Logic to Event (Block | FSM | Logic to Event)

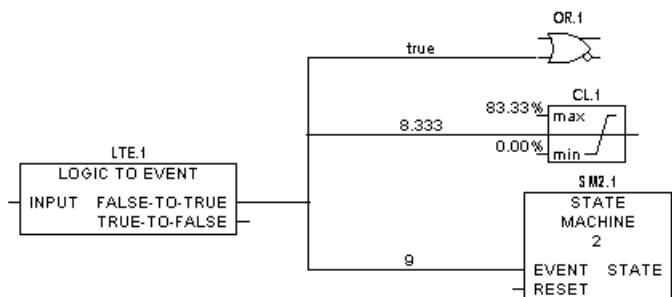
This function block converts the logic signal transition into Logic, Ordinal or Value events.

Input - expects a True or False logic. An input transition from True to False or vice versa causes an event output.

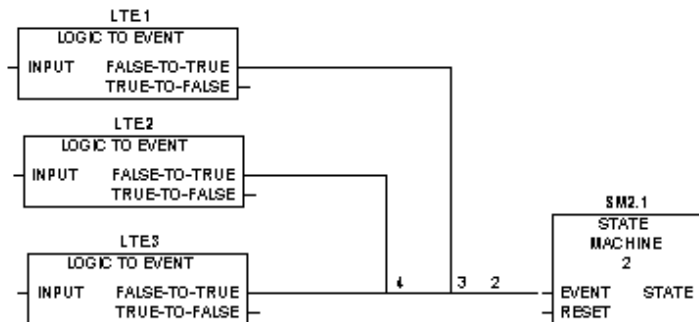


Output - this function block does not supply any data of its own. You must specify the output data when you set up the function block.

When connecting to SM2, this output data is the actual event number (Ordinal).



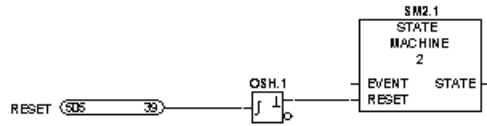
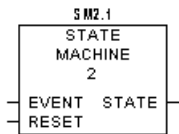
All “Logic to Event” blocks connect to a same “EVENT” input of SM2.



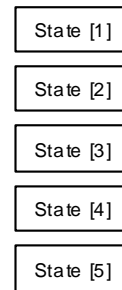
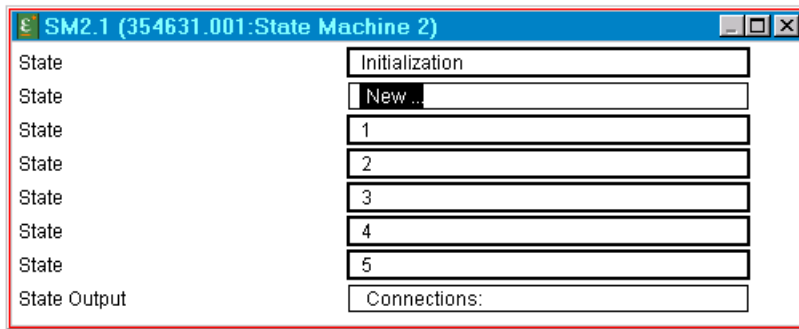
If you have questions, please call the Product Support Group at (704) 588-3246.

State Machine 2 (Block | FSM | State machine 2)

Reset - this overriding reset forces SM2 to its original state; using a “Logic One Shot” is recommended.



States – Set up the states as shown in the following sections. Double-click on SM2 to begin entering states.



Initialization: You can specify the initial state by double-clicking on “Initialization”. The default state is one. SM2 defaults to this state during module power up or reset.

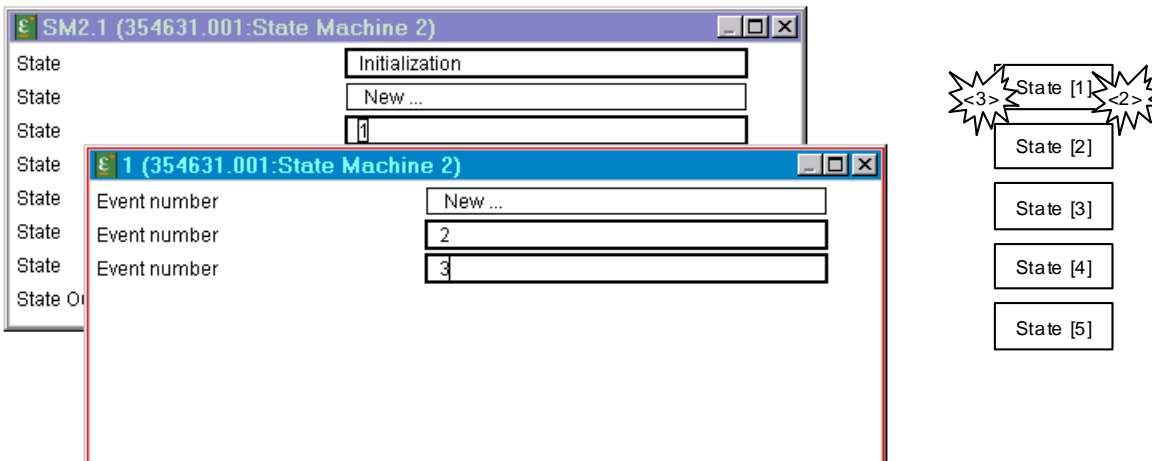
State: Create a new state double-clicking on “New ...”. The program creates states and predefines the state numbers; for example, States 1, 2, 3, etc. If different state numbers are desired, simply change states numbers. Make sure the state numbers are not duplicated.

Important! Do not use State or Event number Zero (0). SM2 will accept zero (0) but it can cause errors.

Deleting a State: Click on the desired state and then, while pressing the <CTRL> key, push the <BACKSPACE> key. Select REMOVE from the pop-up window.

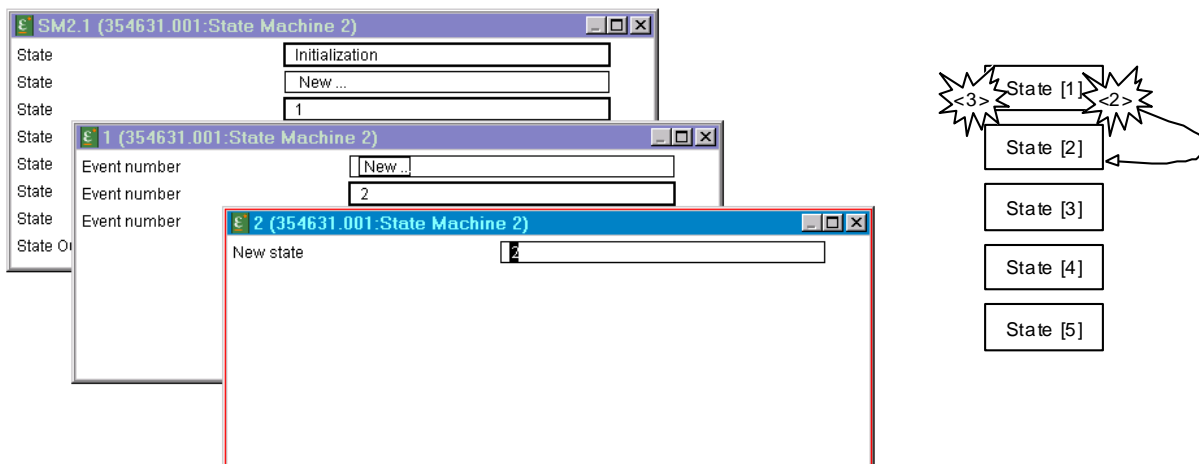
NOTE. The first state cannot be removed.

If you have questions, please call the Product Support Group at (704) 588-3246.



Events: Double-click on a specific state number and an Event pop-up window appears. Double-click on “New...”, to create the events related to this specific state. Editing the event numbers is similar to state numbers.

Destination State: “Double click” on any event to create its destination state.



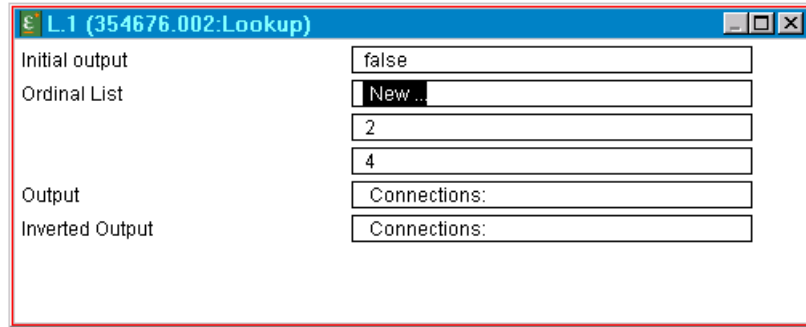
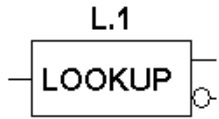
Output – the State Machine 2 function block has a single, ordinal output, which contains the most recent state number. Typically Lookup function blocks are used to ‘decode’ the output.

NOTE. The original State Machine has number of event driven outputs. See “A Quick Look At the ‘Original’ State Machine” for more information.

If you have questions, please call the Product Support Group at (704) 588-3246.

Lookup (Block | Logic | Lookup)

The Lookup function block is a simple comparator. It compares the ordinal input to its pre-configured ordinal number list. When the two numbers are equal, Lookup generates a “True” output.



Initial State - assigns the state of output before any input is triggered. The default Initial output is “False.” You can change it based on your application requirements.

Ordinal List - Double-click on “New ...” to add Ordinal numbers to the list. The zero (0) that is automatically generated is changed to the state numbers for which a TRUE output is required. Editing is similar to the State Machine 2.

NOTE. There is no output interruption between TRUE states.

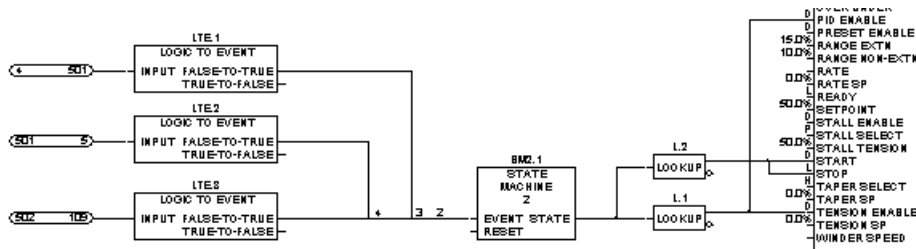


Figure 2 - Typical State Machine 2 Configuration with Inputs and Outputs.

A quick look at the “Original” State Machine

State Machine (Block | FSM | State machine)

This State Machine is more powerful and complicated than State Machine 2. The following example is a State Machine similar to the sample State Machine 2 configuration. (Figure 2)

As Figure 3 shows, State Machine has event outputs that are triggered during state changes. A state change includes exiting and re-entering a same state. In order to demonstrate this powerful feature, a JOG FORWARD / JOG REVERSE function was added to Spindle ‘A’.

If you have questions, please call the Product Support Group at (704) 588-3246.

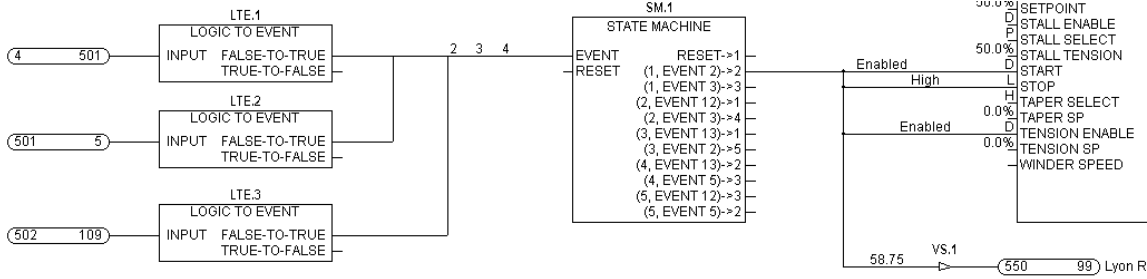


Figure 3 - Typical State Machine with Inputs and Outputs.

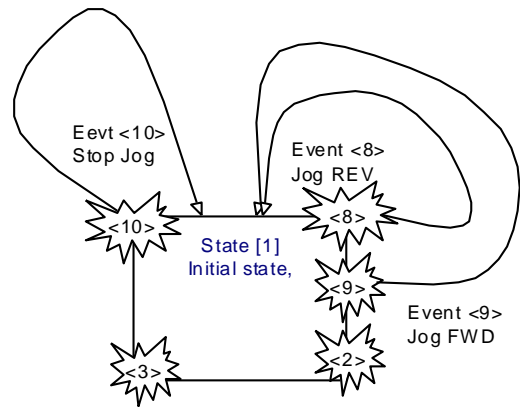
New Functionality

The new pushbuttons operate as follows:

- JOG FWD pushbutton ON generates Event 9
- JOG FWD pushbutton OFF generates Event 10
- JOG REV pushbutton ON generates Event 8
- JOG REV pushbutton OFF generates Event 10

Events 8 and 9 cause State Machine to exit State 1, which creates an output event, and Event 10 causes it to return to State 1.

The illustration to the right shows the new functionality.



The outputs are connected as shown below.

- (1, EVENT 8)->1 assigns a value of -5.0 to value sender VS.1, which is the winder jog speed
- (1, EVENT 8)->1 sets sender LS.1 TRUE, which causes winder to jog at -5.0% speed.
- (1, EVENT 10)->1 resets sender LS.1 FALSE, which causes winder to stop.
- (1, EVENT 9)->1 assigns a value of 5.0 to value sender VS.1, which is the winder jog speed
- (1, EVENT 9)->1 sets sender LS.1 TRUE, which causes winder to jog at 5.0% speed.
- (1, EVENT 10)->1 resets sender LS.1 FALSE, which causes winder to stop.

For More information on this State Machine refer to DSD Help.

Hints. An average State Machine configuration can have as many as 200 interrupts and easily become overwhelming. Carefully plan line-breaks and use meaningful descriptions ahead of time. This will make design and troubleshooting much simpler.

Adding an event or a state changes the display length of this block and may overlap your existing connections.

There is a maximum of 512 interrupts per LINK module. As a result, for a large projects either use a State Machine 2 exclusively or use State Machine 2 with State Machine for sub-states.

If you have questions, please call the Product Support Group at (704) 588-3246.