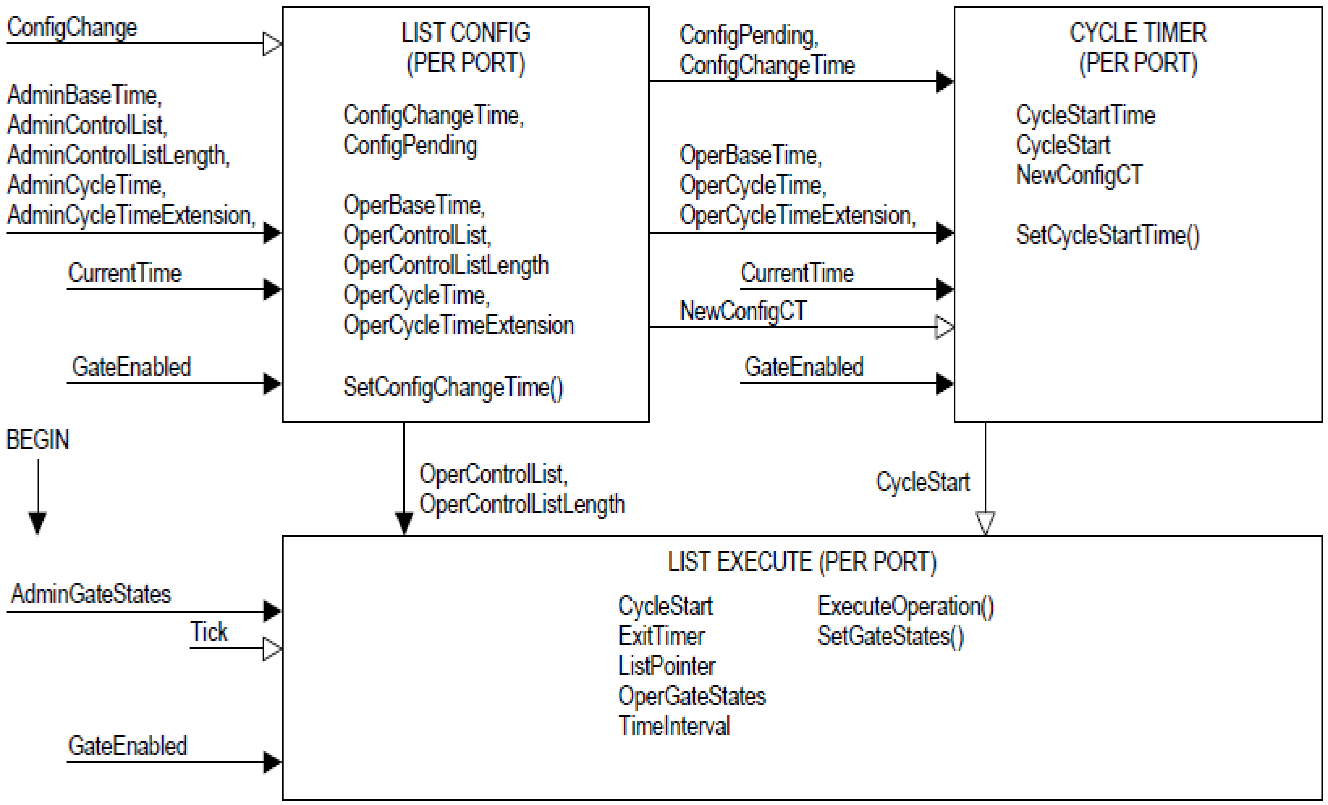
ListExecute Qbv StateMachine

Tick Variable. 60802 Sys Req Input

Thomas Enzinger, Marius Stanica, Rodney Cummings, Jordon Woods

Qbv state machines background

- Transmission gate: A gate that connects or disconnects the transmission selection function of the forwarding process from the queue, allowing or preventing it from selecting frames from that queue. The gate has two states, open and closed
- A gate control list associated with each Port contains an ordered list of gate operations
- Each gate operation changes the transmission gate state for the gate associated with each of the Port's traffic class queues
- The Cycle Timer state machine initiates the execution of the gate control list and ensures that the gating cycle time defined for the Port is maintained
- The List Execute state machine executes the gate operations in the gate control list, in sequence, and establishes the appropriate time delay between each operation
- The List Config state machine manages the process of updating the current active schedule, interrupting the operation of the other two state machines while the update process is performed, and re-starting them once the new schedule has been installed



Problem statement. Scheduled Traffic State Machines. ListExecute state machine

- EXECUTE_CYCLE state start is driven by the synchronized clock distributed i.e. over 802.1AS mechanisms (UCT) but the Tick driving various events (gate operations) within the cycle, it is driven by an implementation-specific system clock
- **So**: there is no explicit mechanism specified, ensuring that events (gate operations) happening within the gating cycle are not driven by the local clock of a device, which could be imprecise/depending on the quartz quality

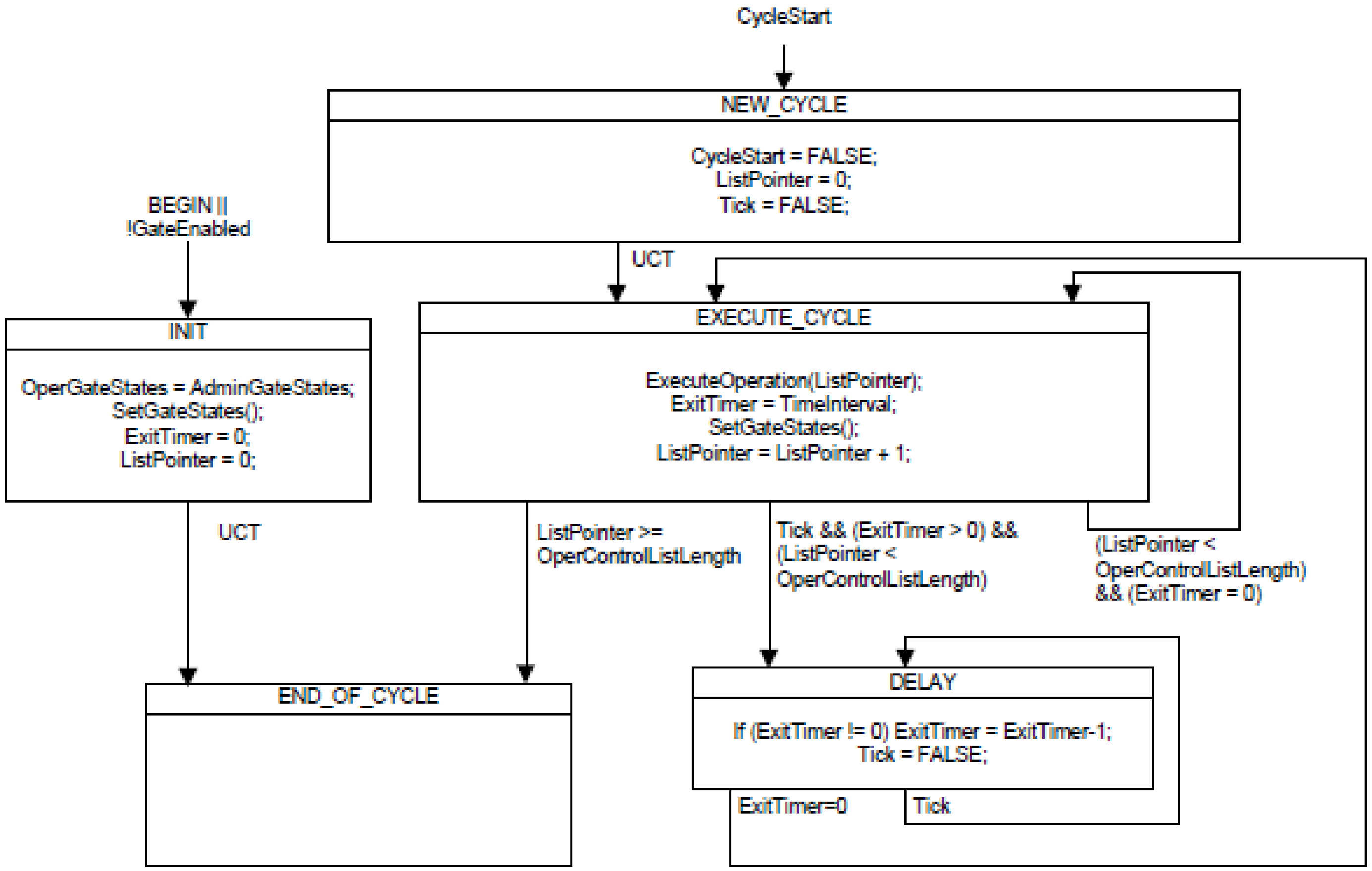
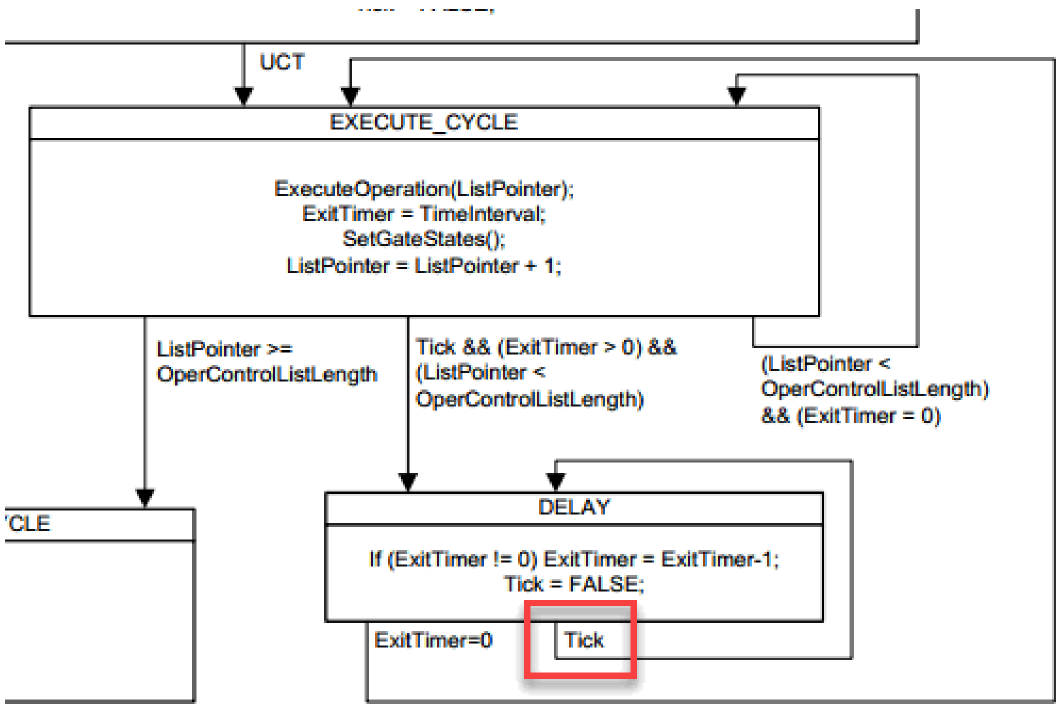
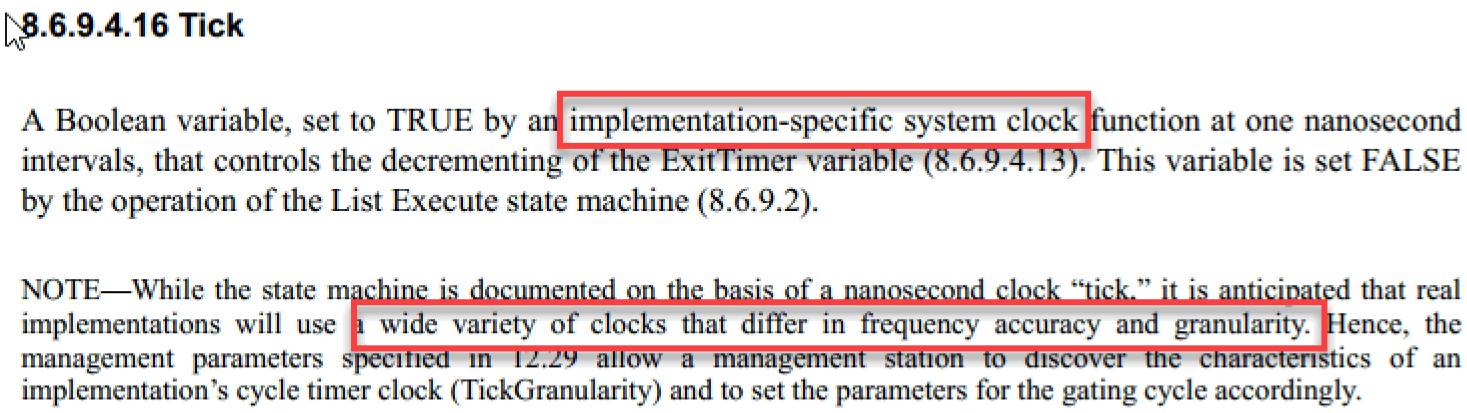


Figure 8-15—List Execute state machine

Example

- SetGateStates() operations cannot surely be driven by the 802.1AS clock, while the Tick is not driven by the 802.1AS clock (see definition 8.6.9.4.16)
- Thus: 2 devices with oscillator drifts of +-100ppm and 1 event per each, scheduled at the end of a 10ms gating cycle, the execution of events would potentially be 2us apart from each other (scaling with the gating-cycle length)

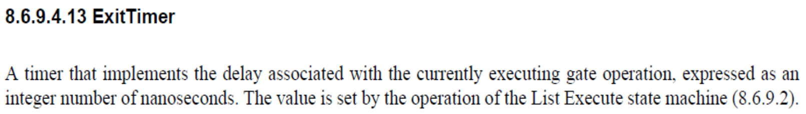

UCT
EXECUTE_CYCLE
ExecuteOperation(ListPointer);
ExitTimer = TimeInterval;
SetGateStates();
ListPointer = ListPointer + 1;
ListPointer >= OperControlListLength
Tick && (ExitTimer > 0) && (ListPointer < OperControlListLength)
(ListPointer < OperControlListLength) && (ExitTimer = 0)
'CLE
DELAY
If (ExitTimer != 0) ExitTimer = ExitTimer-1;
Tick = FALSE;
ExitTimer=0
Tick



8.6.9.4.16 Tick

A Boolean variable, set to TRUE by an implementation-specific system clock function at one nanosecond intervals, that controls the decrementing of the ExitTimer variable (8.6.9.4.13). This variable is set FALSE by the operation of the List Execute state machine (8.6.9.2).

NOTE—While the state machine is documented on the basis of a nanosecond clock “tick,” it is anticipated that real implementations will use a wide variety of clocks that differ in frequency accuracy and granularity. Hence, the management parameters specified in 12.29 allow a management station to discover the characteristics of an implementation’s cycle timer clock (TickGranularity) and to set the parameters for the gating cycle accordingly.



8.6.9.4.13 ExitTimer

A timer that implements the delay associated with the currently executing gate operation, expressed as an integer number of nanoseconds. The value is set by the operation of the List Execute state machine (8.6.9.2).

Handling the issue at the scheduler implementation

- Considering the point presented to be a "fairly static" systematic error, that can be considered in a scheduler, there is still the issue of oscillator drift being shock/temperature-dependent, so a machine’s timing behavior might change with change in environmental conditions (something that would be normally taken care of, by using the synchronized clock)
    - scheduler implementations need to be aware of the implementation: scheduling events at the end of the scheduling list might not be executed (e.g. If the device is @-100ppm) => a certain amount of entries at the end of the table need to remain unused

Other points. Proposal

- Interoperability issues may arise
- Proposal for system requirements:
  - ~~O1: Let us have a system requirement in the JP 60802 System Requirements list so that we mandate the Tick to be regularly (every cycle) synchronized to the 802.1AS synchronized working clock~~
  - R1: Implementation-specific system clock shall be synchronized by default to the 802.1AS working clock
  - ~~O3: ExecuteOperation() shall be synchronized with the working clock~~
  - R2: Implementation specific system clock shall have a given accuracy and granularity (with an upper bound)
- We may also need to create a maintenance item request to IEEE 802.1Q
  - When we dive in the drafting, we need to review the problem and then we need to decide who resolves this (in case we cannot solve it in the JP 60802)
  - We need to have a mechanism in place for signalling/feed back to the 802.1 TSN TG such issues