ListExecute Qbv StateMachine

Tick and Clock Topics. 60802 Sys Req Input

Thomas Enzinger, Marius Stanica, Rodney Cummings, Jordon Woods

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 but the Tick driving various events within the cycle, it is driven by an implementation-specific system clock

- **So**: there is no explicit mechanism specified, ensuring that events happening within the gating cycle are not driven by the local clock of a device, which could be imprecise/depending on the quartz quality
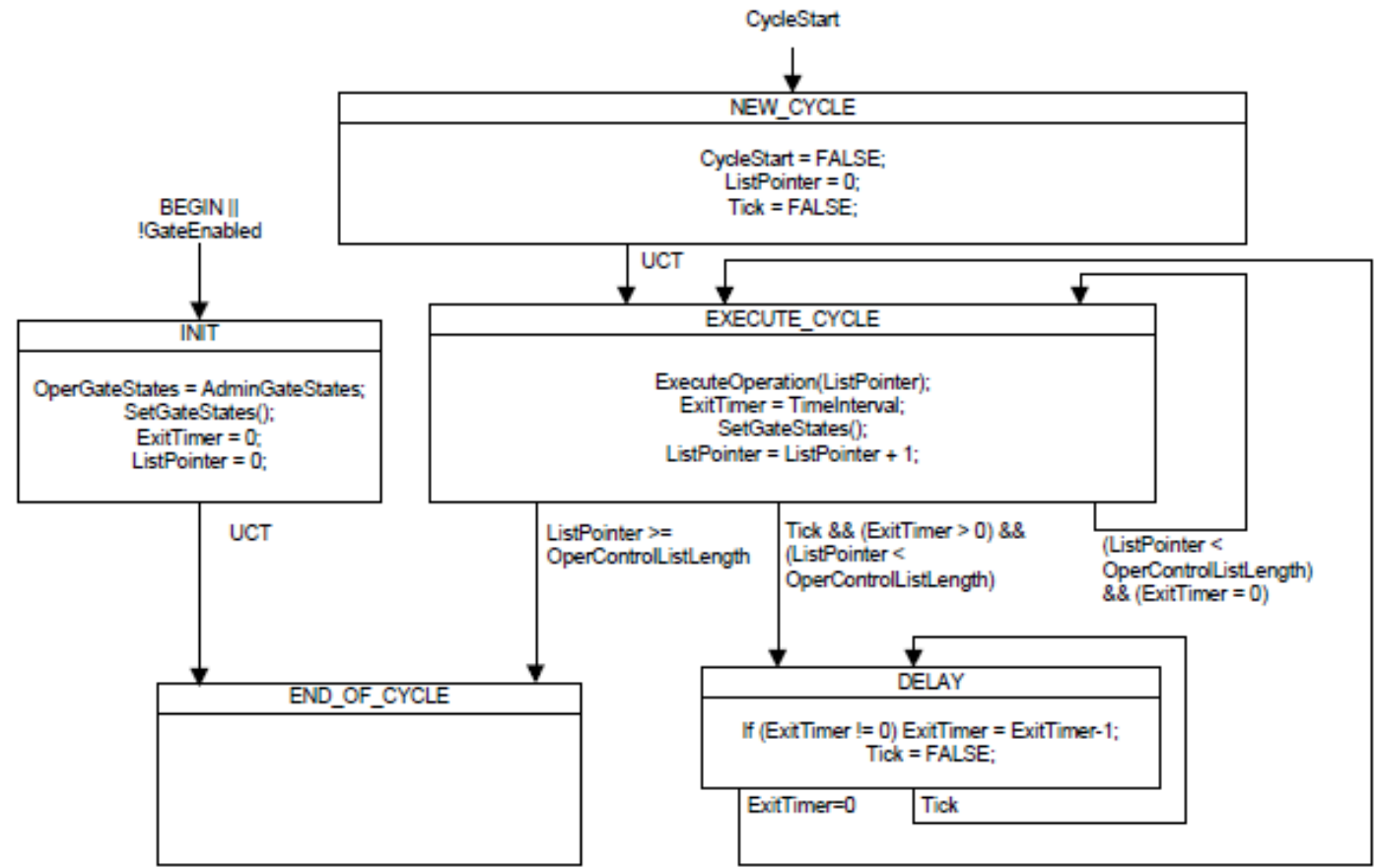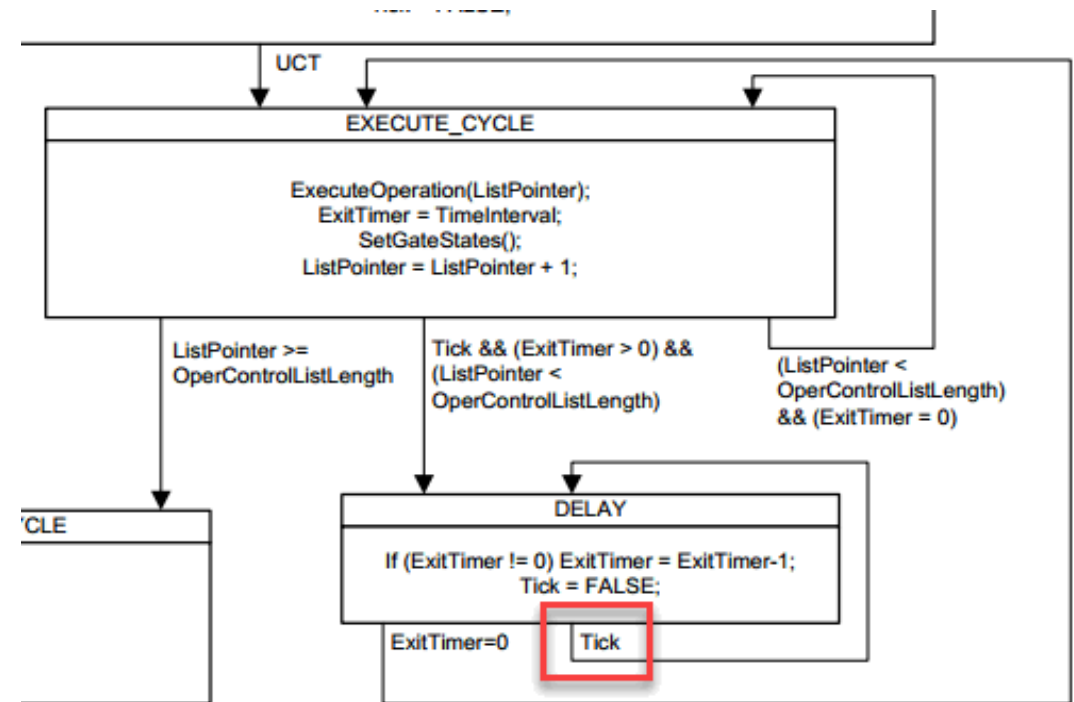
CycleStart

NEW_CYCLE

CycleStart = FALSE;
ListPointer = 0;
Tick = FALSE;

BEGIN ||
!GateEnabled

UCT

INIT

OperGateStates = AdminGateStates;
SetGateStates();
ExitTimer = 0;
ListPointer = 0;

EXECUTE_CYCLE

ExecuteOperation(ListPointer);
ExitTimer = TimeInterval;
SetGateStates();
ListPointer = ListPointer + 1;

UCT

ListPointer >=
OperControlListLength

Tick && (ExitTimer > 0) &&
(ListPointer <
OperControlListLength)

(ListPointer <
OperControlListLength)
&& (ExitTimer = 0)

END_OF_CYCLE

DELAY

If (ExitTimer != 0) ExitTimer = ExitTimer-1;
Tick = FALSE;

ExitTimer=0   Tick

Figure 8-15—List Execute state machine

Example

- SetGateStates() operations cannot surely be driven by the 802.1AS clock, whilst 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 >=        Tick && (ExitTimer > 0) &&      (ListPointer <
OperControlListLength  (ListPointer <                  OperControlListLength)
                       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:
  - Let us have a system requirement in the JP 60802 System Requirements list so that we mandate the Tick to be driven by the 802.1AS synchronized clock