Fixing the List Config/Cycle Timer state machine race conditions

Re: Q-rev/D1.0 SA ballot comment I-22, 802.1 maintenance item 0319.
Following up Alon Regev's work.

Mick Seaman
mickseaman@gmail.com

Overview

- State machine descriptions—some best practice
  - Apply to the Net Mgmt/List Config/Cycle Timer/List Execute interaction
  - Try to avoid unnecessarily changing what is already documented
- Current state machine interactions
  - Race conditions
- A better Cycle Timer state machine
- Proposed List Config and Cycle Timer changes
  - List Execute unchanged

State machine best practice

If one machine, “the director”, prompts another “the actor”:

- Director sets a communication variable and the actor clears it **when the assigned task is complete** (not when actor notices the communication)
  - — Requires clarity on what the task is, e.g. is the task just adding an item to a list for future work, or is it performing that work
  - — The director can move to his next task (possibly a repeated request) when the variable has been cleared.
  - — The director “owns” the variable and is responsible for its initialization on BEGIN (and whenever otherwise necessary).
  - — Only one set/clear variable should be used per task (or task group)

State machine specs prompt the designer toward a minimal state solution with clear flow:

- — Inconsistent with burying changes to state variables inside procedures.
- — States should be ‘came from’ conditions

Net mgmt./List Config/Cycle Timer/List Execute configuration—current (1)

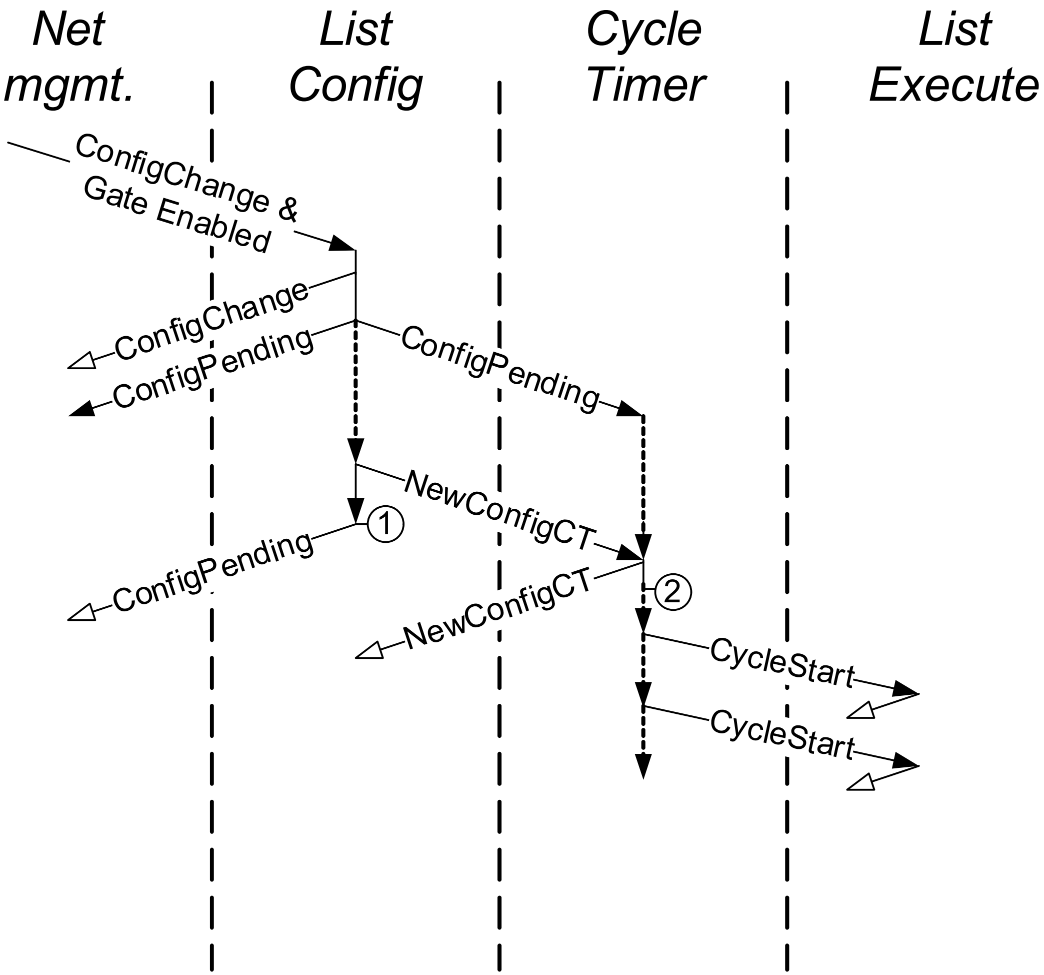
Configuration change race

State machines do not constrain the execution order of List Config:UPDATE_CONFIG and Cycle Timer:SET_CYCLE_START_TIME, (1) and (2) in the Figure.

(1) clears ConfigPending, which affects setCycleTimer() processing at (2).

Note

SetCycleStartTime() can be executed by toggling GateEnabled, so a change to clearing ConfigPending in the Cycle Timer machine needs to be conditional. Maintenance item 0319 proposes an extra state to clear the varaiable, but also clears it on one condition in the procedure.

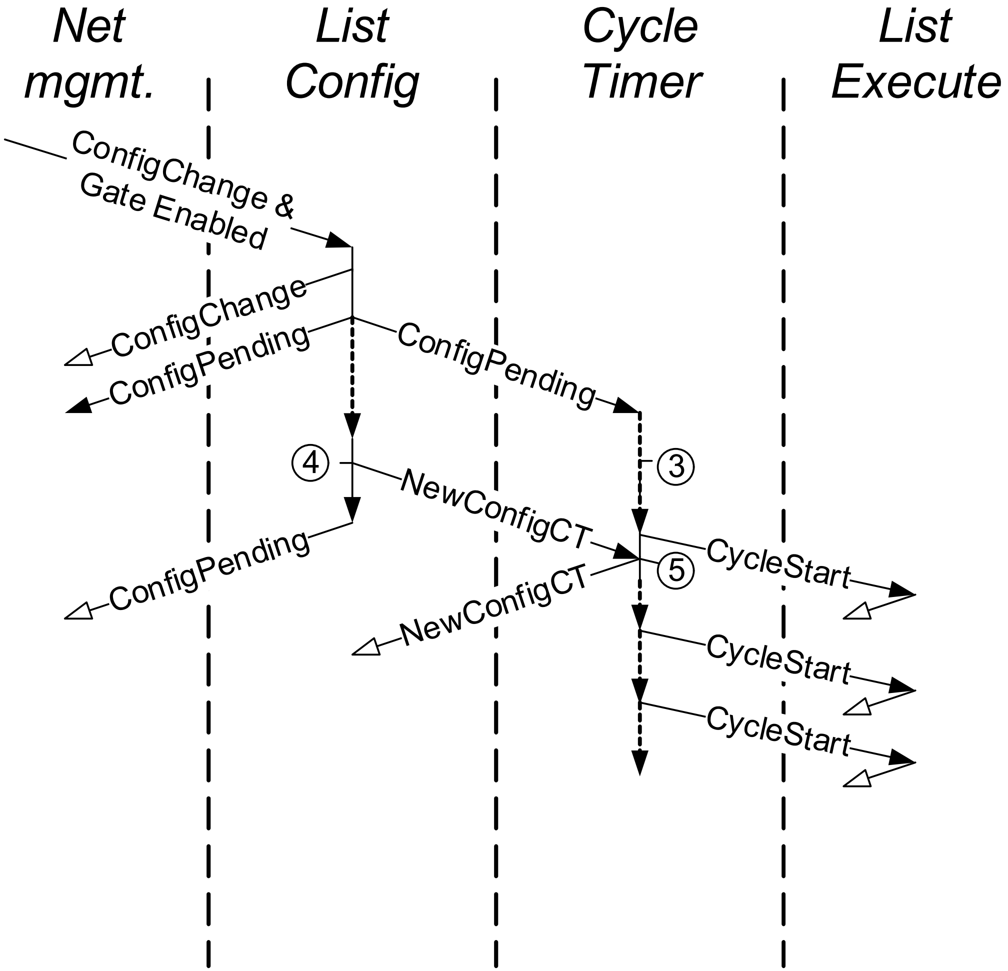


Net mgmt./List Config/Cycle Timer/List Execute configuration—current (2)

Another race?

In the Cycle Timer machine, if ConfigPending and ConfigChangeTime is no later than CurrentTime + OperCycleTime + OperCycleTimeExtension then SetCycleStartTime case (d) delays the next CycleStartTime to ConfigChangeTime.

State List Config:UPDATE.. is entered when ConfigChangeTime <= CurrentTime and Cycle Timer:START_CYCLE is entered when CycleStartTime <= CurrentTime, i.e. at the same time following case (d) above. Execution of the two states is mutually exclusive, but either can precede the other. List Execute:NEW_CYCLE can be entered as soon as START_CYCLE has executed, so can precede List Config:UPDATE.., and can take the first item from the old OperControlList before that list is updated.



A better Cycle Timer machine (1)

Don’t use SetCycleStartTime() to test Config Pending and hide the significant state difference between:

— continuing an established set of cycles (with one set of Oper .. parameters)

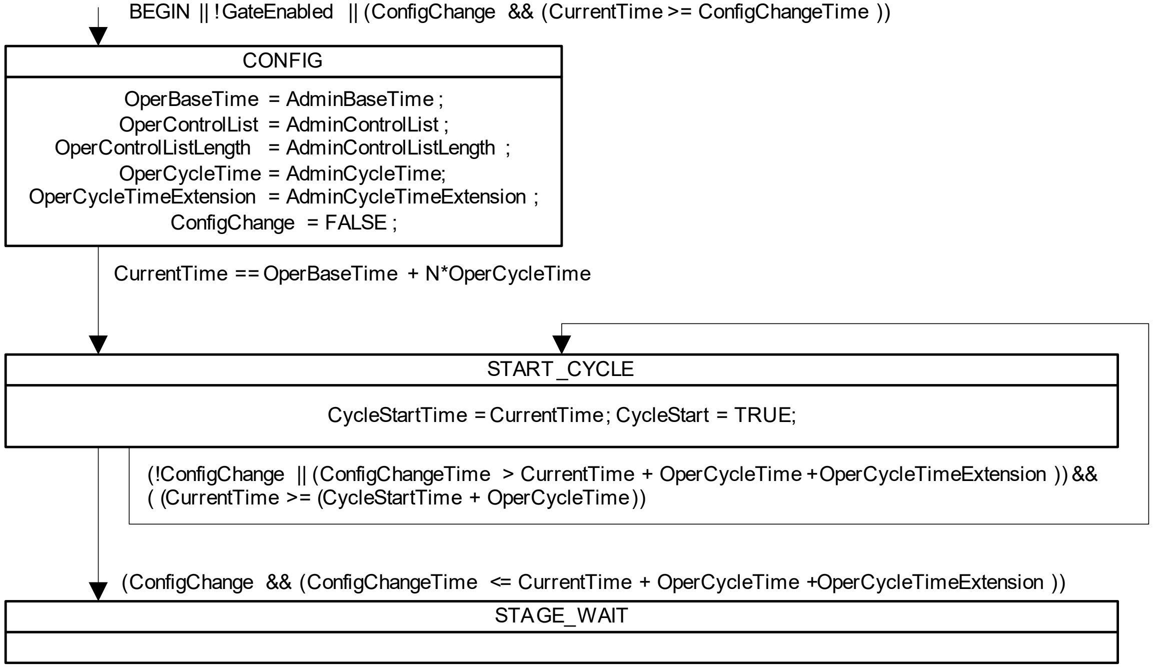
— setting up a new, possibly delayed set of cycles with a new set of Oper.. parameters

Don’t split the logic for a new set of cycles between machines. Add update functionality to the Cycle Timer, make the update conditions explicit [no need for SetCycleStartTime()]. All List Config (if kept) has to do is calculate Config Change Time. List Execute machine unchanged.

Note: Open transitions take precedence, times are ‘equal’ if within implementation granularity, all eligible transitions occur before time advances.

Current curiosities:
1. Oper.. variables only set when GateEnabled, race condition on start up [removed].

2.Cycle start skipped when one more cycle would fit before ConfigChangeTime.



BEGIN || !GateEnabled || (ConfigChange && (CurrentTime >= ConfigChangeTime))

CONFIG

OperBaseTime = AdminBaseTime ;
OperControlList = AdminControlList ;
OperControlListLength = AdminControlListLength ;
OperCycleTime = AdminCycleTime;
OperCycleTimeExtension = AdminCycleTimeExtension ;
ConfigChange = FALSE ;

CurrentTime == OperBaseTime + N*OperCycleTime

START _CYCLE

CycleStartTime = CurrentTime; CycleStart = TRUE;

(!ConfigChange || (ConfigChangeTime > CurrentTime + OperCycleTime +OperCycleTimeExtension)) &&
((CurrentTime >= (CycleStartTime + OperCycleTime))

(ConfigChange && (ConfigChangeTime <= CurrentTime + OperCycleTime +OperCycleTimeExtension))

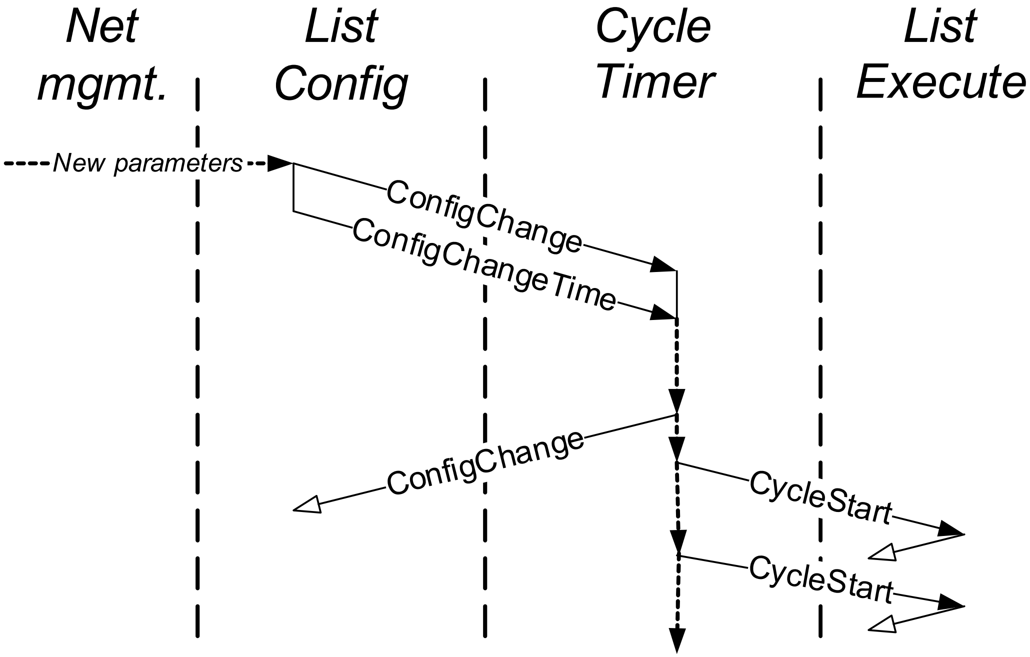
STAGE_WAIT

A better Cycle Timer machine (2)

ConfigChange Time calculation, and ConfigChange set as a direct procedural result of a network management update of list parameters (no List Config machine required).

CycleTimer machine clears (resets) ConfigChange when the Oper.. variables have been updated with the new Admin variables [job done].

No need for ConfigPending or NewConfigCT.



But there are a lot of references to ConfigPending in Q-Rev/D1.0, including throughout the management clauses. Need to be sensitive to current implementations and users, *sometimes fixing a mistake is a mistake*.

Following slides suggest a softer change, retaining ConfigPending.

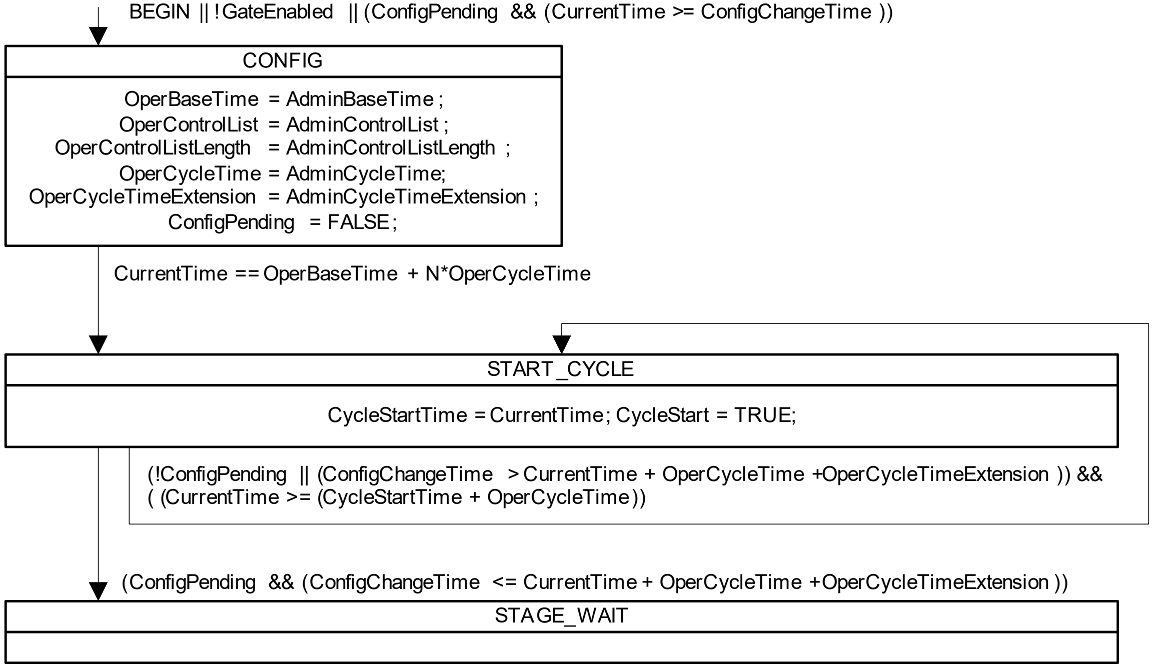
Proposed Cycle Timer machine

As on Slide 6, but substituting ConfigPending for ConfigChange. SetCycleStartTime() procedure (8.6.9.1.1) no longer used, move Notes to 8.6.9.1. Trivial changes to 8.6.9.4.8, 8.6.9.4.12, and 8.6.9.4.21—refer to the state machine, not the procedure.

Note: Open transitions take precedence, times are ‘equal’ if within implementation granularity, all eligible transitions occur before time advances.

Curiosity left unchanged:

1. Cycle start skipped when one more cycle would fit before ConfigChangeTime.



BEGIN || !GateEnabled || (ConfigPending && (CurrentTime >= ConfigChangeTime))

CONFIG

OperBaseTime = AdminBaseTime ;
OperControlList = AdminControlList ;
OperControlListLength = AdminControlListLength ;
OperCycleTime = AdminCycleTime;
OperCycleTimeExtension = AdminCycleTimeExtension ;
ConfigPending = FALSE;

CurrentTime == OperBaseTime + N*OperCycleTime

START _CYCLE

CycleStartTime = CurrentTime; CycleStart = TRUE;

(!ConfigPending || (ConfigChangeTime > CurrentTime + OperCycleTime +OperCycleTimeExtension)) &&
((CurrentTime >= (CycleStartTime + OperCycleTime))

(ConfigPending && (ConfigChangeTime <= CurrentTime + OperCycleTime +OperCycleTimeExtension))

STAGE_WAIT

Net mgmt./List Config/Cycle Timer/List Execute configuration—proposed

States/config progress:

0: !ConfigChange && !ConfigPending && (CurrentTime >= ConfigChangeTime)

List is executing, or will start executing at net possible Cycle Start time (or !GateEnabled)

1: ConfigChange && !ConfigPending

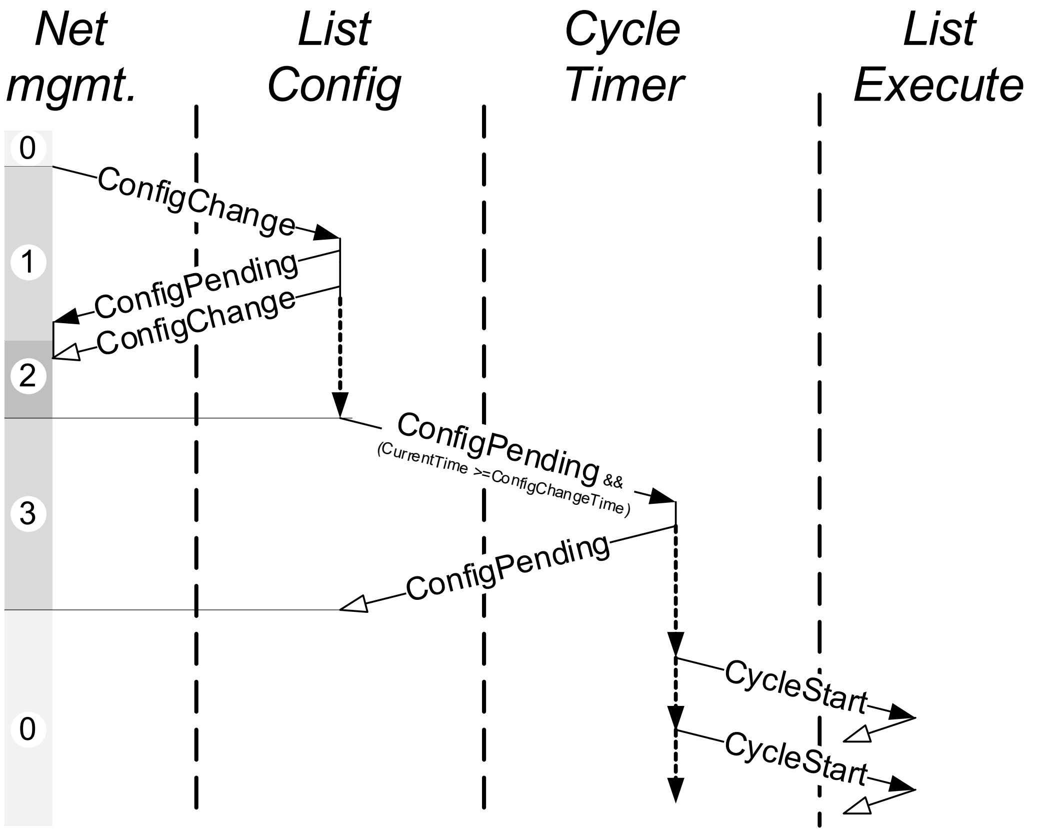
Probably not observable. List Config machine will set ConfigPending if !BEGIN,

2: ConfigPending && (CurrentTime < ConfigChangeTime)

Update of Oper variables is pending.

3: ConfigPending && (CurrentTime >= ConfigChangeTime) but Oper.. variables not updated

Probably not observable.



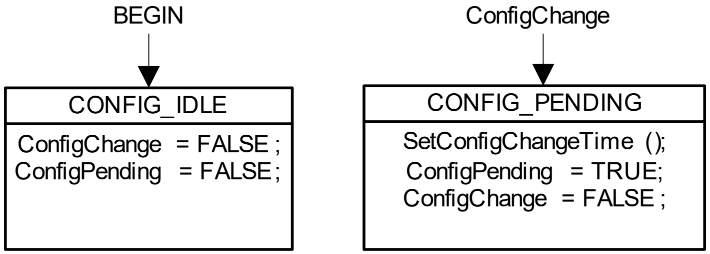
Proposed List Config machine

Trivial accompanying changes to 8.6.9. pg 229 line 19, and through the subclauses of 8.6.9.4 (substitute Cycle Timer state machine for references to List Config state machine).

Curiosities:

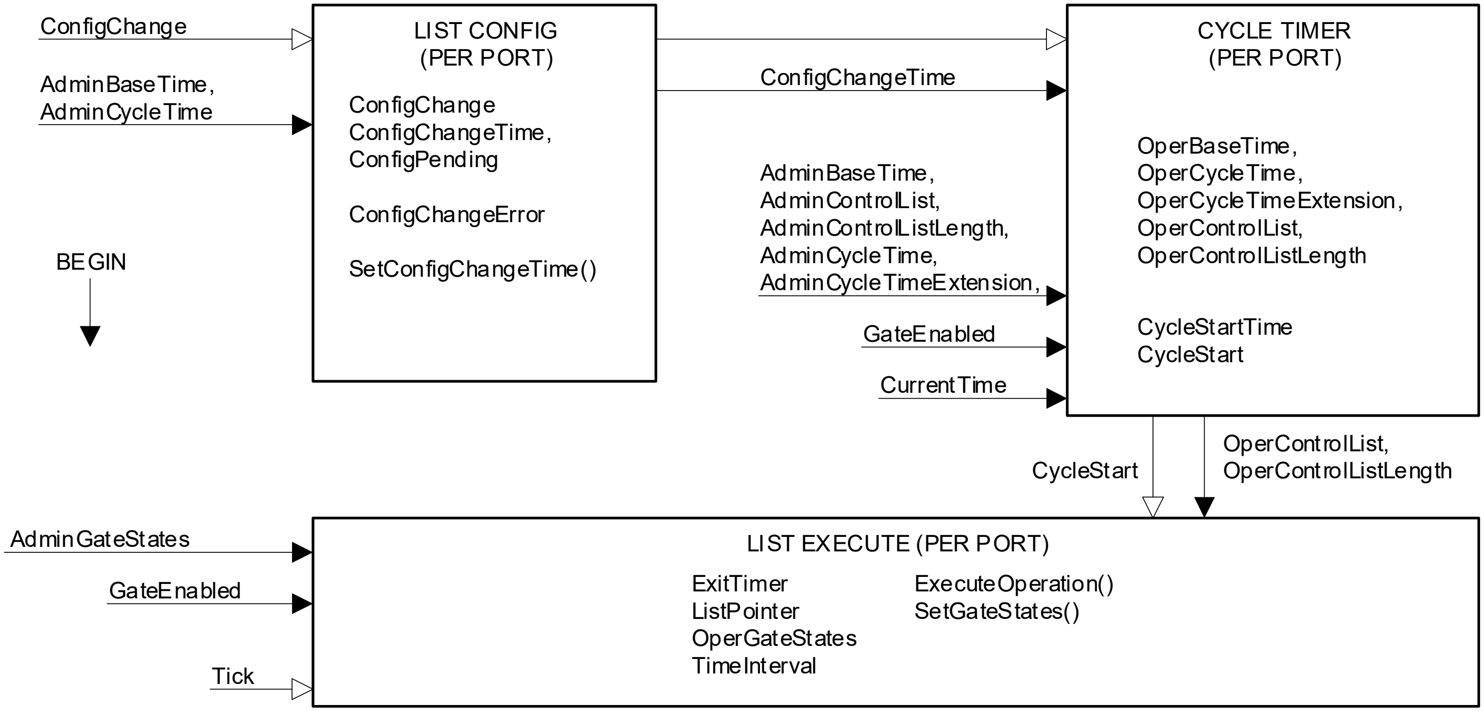
1. Was not possible to configure the Oper.. variables before enabling Gates [removed]. This change has no effect on the use of gates when !GateEnabled since the List Execute machine will hold them in the AdminGatesState anyway.

2. ConfigChange not initialized by any machine [now shown as initialized by this machine].



Proposed state machine overview (Figure 8-18)

Remove NewConfigCT. Variables that are shown as communicating between machines are only shown inside their ‘home’ machine, i.e. the machine that is responsible for their initialisation on BEGIN.



ConfigChange

AdminBaseTime,
AdminCycleTime

BEGIN

LIST CONFIG
(PER PORT)

ConfigChange
ConfigChangeTime,
ConfigPending

ConfigChangeError

SetConfigChangeTime()

ConfigChangeTime

AdminBaseTime,
AdminControlList,
AdminControlListLength,
AdminCycleTime,
AdminCycleTimeExtension,

GateEnabled

CurrentTime

CYCLE TIMER
(PER PORT)

OperBaseTime,
OperCycleTime,
OperCycleTimeExtension,
OperControlList,
OperControlListLength

CycleStartTime
CycleStart

CycleStart

OperControlList,
OperControlListLength

AdminGateStates

GateEnabled

Tick

LIST EXECUTE (PER PORT)

ExitTimer
ListPointer
OperGateStates
TimeInterval

ExecuteOperation()
SetGateStates()

NOTATION:
Variables are shown both within the machine where they are initialized and between machines where they are used to communicate information. In the latter case the arrow styles, running from one machine to another, provide an overview of how the variables are used:

——▶ Not changed by the target machine, this variable communicates between state machines for the same port.

——▷ Set (or cleared) by the originating machine, cleared (or set) by the target machine, communicates between machines for the same port.