```
┌──────────────────── MODULE VectorClocks ────────────────────┐
EXTENDS Integers
CONSTANTS Procs, MAX

--algorithm VectorClocks
variables
  msgs = [p ∈ Procs ↦ [q ∈ Procs ↦ 0]] ;   defined as Vector Clock

define
    returns the maximum value for each element of two vectors
  Max(v1, v2) ≜ [p ∈ Procs ↦ IF v1[p] > v2[p] THEN v1[p] ELSE v2[p]]
    increments by 1 the 'e' element of the vector 'v'
  Increment(e, v) ≜ [p ∈ Procs ↦ IF p = e THEN v[p] + 1 ELSE v[p]]
end define ;

fair process VectorClock ∈ Procs
variables
  vc = [p ∈ Procs ↦ 0]  Initially all clocks are zero
begin Main:
  while vc[self] < MAX do
    either Receive:  increments local clock and calcs maximum of two clocks
      vc := Increment(self, Max(vc, msgs[self])) ;
    or Send:  increments local clock and sends it to another process
      vc[self] := vc[self] + 1 ;
      with p ∈ Procs \ {self} do   send vc to 'p' via msgs[p]
        msgs[p] := vc ;
      end with ;
    end either ;
  end while ;
end process ;

end algorithm  ;
 BEGIN TRANSLATION
VARIABLES msgs, pc

 define statement
Max(v1, v2) ≜ [p ∈ Procs ↦ IF v1[p] > v2[p] THEN v1[p] ELSE v2[p]]

Increment(e, v) ≜ [p ∈ Procs ↦ IF p = e THEN v[p] + 1 ELSE v[p]]

VARIABLE vc

vars ≜ ⟨msgs, pc, vc⟩

ProcSet ≜ (Procs)

Init ≜  Global variables
          ∧ msgs = [p ∈ Procs ↦ [q ∈ Procs ↦ 0]]
          Process VectorClock
        ∧ vc = [self ∈ Procs ↦ [p ∈ Procs ↦ 0]]
        ∧ pc = [self ∈ ProcSet ↦ "Main"]

Main(self) ≜ ∧ pc[self] = "Main"
             ∧ IF vc[self][self] < MAX
                   THEN ∧ ∨ ∧ pc′ = [pc EXCEPT ![self] = "Receive"]
                          ∨ ∧ pc′ = [pc EXCEPT ![self] = "Send"]
                   ELSE ∧ pc′ = [pc EXCEPT ![self] = "Done"]
             ∧ UNCHANGED ⟨msgs, vc⟩

Receive(self) ≜ ∧ pc[self] = "Receive"
                ∧ vc′ = [vc EXCEPT ![self] = Increment(self, Max(vc[self], msgs[self]))]
                ∧ pc′ = [pc EXCEPT ![self] = "Main"]
                ∧ msgs′ = msgs

Send(self) ≜ ∧ pc[self]  = "Send"
             ∧ vc′ = [vc EXCEPT ![self][self] = vc[self][self] + 1]
             ∧ ∃ p ∈ Procs \ {self} :
                  msgs′ = [msgs EXCEPT ![p] = vc′[self]]
             ∧ pc′ = [pc EXCEPT ![self] = "Main"]

VectorClock(self) ≜ Main(self) ∨ Receive(self) ∨ Send(self)

Next ≜ (∃ self ∈ Procs : VectorClock(self))
           ∨  Disjunct to prevent deadlock on termination
             ((∀ self ∈ ProcSet : pc[self] = "Done") ∧ UNCHANGED vars)

Spec ≜  ∧ Init ∧ □[Next]_vars
        ∧ ∀ self ∈ Procs : WF_vars(VectorClock(self))

Termination ≜ ◇(∀ self ∈ ProcSet : pc[self] = "Done")

 END TRANSLATION

 Boundedness
VectorClockOK ≜ (∀ k, l ∈ Procs : vc[k][k] ≥ vc[l][k])
```

* Modification History
* Last modified Sun *Nov* 25 18:51:00 *PST* 2018 by *ocosta*
* Created Sat *Nov* 17 15:07:39 *PST* 2018 by *ocosta*