```
┌──────────────────── MODULE op_counter ────────────────────┐
EXTENDS Integers, TLC, Sequences
CONSTANTS
  N
  P ≜ 1 . . N

--algorithm op_counter
variables
  msg = [m ∈ P ↦ 0] ;   simulate broadcast

macro Broadcast(v)begin   simulate broadcast
  msg := [m ∈ P ↦ v] ;
end macro ;

fair process Counter ∈ P
variables
  count = 0 ;   local counter
begin Update:
  either Increment:
    count := count + 1 ;
    Broadcast(count) ;
  or Decrement:
    count := count − 1 ;
    Broadcast(count) ;
  end either ;
end process ;

end algorithm  ;
 BEGIN TRANSLATION
VARIABLES msg, pc, count

vars ≜ ⟨msg, pc, count⟩

ProcSet ≜ (P)

Init ≜  Global variables
         ∧ msg = [m ∈ P ↦ 0]
          Process Counter
         ∧ count = [self ∈ P ↦ 0]
         ∧ pc = [self ∈ ProcSet ↦ "Update"]

Update(self) ≜ ∧ pc[self] = "Update"
               ∧ ∨ ∧ pc′ = [pc EXCEPT ![self] = "Increment"]
                 ∨ ∧ pc′ = [pc EXCEPT ![self] = "Decrement"]
               ∧ UNCHANGED ⟨msg, count⟩

Increment(self) ≜ ∧ pc[self] = "Increment"
                  ∧ count′ = [count EXCEPT ![self] = count[self] + 1]
```

```
                              $\land\ msg' = [m \in P \mapsto count'[self]]$
                              $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"Done"}]$

$Decrement(self) \triangleq$  $\land\ pc[self] = \text{"Decrement"}$
                              $\land\ count' = [count\ \text{EXCEPT}\ ![self] = count[self] - 1]$
                              $\land\ msg' = [m \in P \mapsto count'[self]]$
                              $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"Done"}]$

$Counter(self) \triangleq Update(self) \lor Increment(self) \lor Decrement(self)$

$Next \triangleq (\exists\, self \in P : Counter(self))$
           $\lor$  Disjunct to prevent deadlock on termination
             $((\forall\, self \in ProcSet : pc[self] = \text{"Done"}) \land \text{UNCHANGED}\ vars)$

$Spec \triangleq$ $\land\ Init \land \Box[Next]_{vars}$
         $\land\ \forall\, self \in P : \mathrm{WF}_{vars}(Counter(self))$

$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{"Done"})$

 END TRANSLATION

 Eventual Convergence:
 Safety: i, j : $C(xi) = C(xj)$ implies that the abstract states of i and j are equivalent.
$Safety \triangleq (\forall\, k, l \in P : msg[k] = msg[l])$
 Liveness: i, j : f  $C(xi)$ implies that, eventually, f  $C(xj)$.
$Liveness \triangleq \Diamond(\forall\, k \in P : count[k] = msg[k])$

└──────────────────────────────────────────────────────────────────┘
\* Modification History
\* Last modified Sun $Dec$ 09 19:15:24 $PST$ 2018 by $ocosta$
\* Created Sat $Dec$ 01 16:58:15 $PST$ 2018 by $ocosta$
```