MODULE *init_once*

EXTENDS *TLC*, *FiniteSets*, *Integers*

CONSTANTS *Processes*

```
--algorithm init_once
variables
    lock = FALSE ;
    is_init = FALSE ;

    pids = {} ;

define
    just_once ≜ ◇□(Cardinality(pids) = 1) ∧ □(Cardinality(pids) ≤ 1)
end define

 initializer
fair + process pid ∈ Processes
begin
    BeginInitOnce:
        while ¬is_init do
            LoadLockRelaxed:
                if ¬lock then
                    CompareExchange:
                        if ¬lock then
                            lock := TRUE ;
                            pids := pids ∪ {self} ;

                            Initialize:
                                skip ;

                            StoreIsInit:
                                is_init := TRUE ;
                        end if ;
                end if ;
        end while ;
end process ;

end algorithm ;
```

BEGIN TRANSLATION ($chksum(pcal) =$ "171ee2b4" $\wedge\ chksum(tla) =$ "1cd14bb5")

VARIABLES *lock*, *is_init*, *pids*, *pc*

define statement

$just\_once \triangleq \Diamond\Box(Cardinality(pids) = 1) \wedge \Box(Cardinality(pids) \leq 1)$

$vars \triangleq \langle lock,\ is\_init,\ pids,\ pc\rangle$

$ProcSet \triangleq (Processes)$

$Init \triangleq$ Global variables
$\land\ lock = \text{FALSE}$
$\land\ is\_init = \text{FALSE}$
$\land\ pids = \{\}$
$\land\ pc = [self \in ProcSet \mapsto \text{“BeginInitOnce”}]$

$BeginInitOnce(self) \triangleq \land\ pc[self] = \text{“BeginInitOnce”}$
$\land\ \text{IF } \neg is\_init$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“LoadLockRelaxed”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$LoadLockRelaxed(self) \triangleq \land\ pc[self] = \text{“LoadLockRelaxed”}$
$\land\ \text{IF } \neg lock$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“CompareExchange”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“BeginInitOnce”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$CompareExchange(self) \triangleq \land\ pc[self] = \text{“CompareExchange”}$
$\land\ \text{IF } \neg lock$
$\text{THEN } \land\ lock' = \text{TRUE}$
$\land\ pids' = (pids \cup \{self\})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Initialize”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“BeginInitOnce”}]$
$\land\ \text{UNCHANGED } \langle lock, pids \rangle$
$\land\ \text{UNCHANGED } is\_init$

$Initialize(self) \triangleq \land\ pc[self] = \text{“Initialize”}$
$\land\ \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“StoreIsInit”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$StoreIsInit(self) \triangleq \land\ pc[self] = \text{“StoreIsInit”}$
$\land\ is\_init' = \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“BeginInitOnce”}]$
$\land\ \text{UNCHANGED } \langle lock, pids \rangle$

$pid(self) \triangleq BeginInitOnce(self) \lor LoadLockRelaxed(self)$
$\lor\ CompareExchange(self) \lor Initialize(self)$
$\lor\ StoreIsInit(self)$

Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq \land\ \forall\, self \in ProcSet : pc[self] = \text{“Done”}$
$\land\ \text{UNCHANGED } vars$

$Next \triangleq (\exists\, self \in Processes : pid(self))$
$\lor\ Terminating$

$$Spec \triangleq \land Init \land \Box[Next]_{vars} \\ \land \forall self \in Processes : \mathrm{SF}_{vars}(pid(self))$$

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

END TRANSLATION