«File»
Stubs

- odoBackup_Node_Value:uint32_T=0
- read_counter:uint16_T=0
- write_counter:uint16_T=0

- stub_ee_write_request(value:uint32_T):...
- stub_ee_read_request():uint32_T

«Interface»
I_EEPROM_Manager

- ee_write_request(key:EEPROM_DATA_KEY,value:uint32_T):void
- ee_read_request(storedValue:EEPROM_DATA_KEY):uint32_T

«Usage»

«Interface»
I_OdometerBackup

- set_IgnitionStatus(ignition:ignition_status_T):void
- set_OdometerMasterValue(new_odoMasterValue:uint32_T):void
- set_HardReset():void
- get_Odometer_Node():uint32_T

itsI_EEPROM_Manager   1

I_OdometerBackup

«Realization»

OdometerBackup

- tamper_flg:RiCBoolean=FALSE
- odoMasterValue:uint32_T=0
- odoBackupValue:uint32_T=0

- set_IgnitionStatus(ignition:ignition_status_T):void
- set_OdometerMasterValue(new_odoMasterVal...
- evIgnitionStatus(ignitionStatus:ignition_status_T)
- checkInitialMileage():void
- checkForTamper():void
- evOdoMasterValue(new_odo_value:uint32_T)
- set_HardReset():void
- evHardReset()
- get_Odometer_Node():uint32_T
- Operation_21():void

odoBackupValue:uint32_T

OdometerBackUpNODE

{D}
NODE = {ABS || BCM || DDM || PCM || PDM || RCM}

VERIFY: Missing msg strategy.

IF (MasterOdo missing > 5s)
    PAUSE Backup logic

IF (MasterOdo return from missing > odoBackup + 1000)
    Flag Tamper Event: Stop Backup Logic
ELSE
    Re-init odoBackup => odoBackup = MasteOdo ?

initialization

Wait until WAIT_FOR_MILEAGE before performing backup logic.

evOdoMasterValue/checkInitialMileage(me); /* internal transition */

Is an EE write on a HardReset neccessary?

Yes - this protects against the following False Tamper Event: an evHardReset in RUN || START after the MasterOdo has reached a value greater than odoBackupValue + MASTER_ODO_GT_THRESH. Note: on entry of the initialization state, odoBackupValue is set to the value in EE.

evHardReset/
I_EEPROM_Manager_ee_write_request(me->itsI_EEPROM_Manager, EE_ODO_VALUE_NODE,me->odoBackupValue);

[me->odoBackupValue > WAIT_FOR_MILEAGE]

ignition_status

pause

evOdoMasterValue[me->tamper_flg == FALSE]/
me->odoMasterValue = params->new_odo_value;
me->odoBackupValue =
        I_EEPROM_Manager_ee_read_request(me->itsI_EEPROM_Manager,EE_ODO_VALUE_NODE);

[params->ignitionStatus == OFF]/
I_EEPROM_Manager_ee_write_request
(me->itsI_EEPROM_Manager,EE_ODO_VALUE_NODE,me->odoBackupValue);

C

evIgnitionStatus

backup_enabled

[params->ignitionStatus == UNKNOWN ||
params->ignitionStatus == INVALID]

If igitionStatus is UNKNOWN or INVALID default into state that reacts to a new Master Odo value. This reduces the risk of setting a false evTamer. It may be possible to receive a new Master Odo value with an unknown or invalid igition status signal.

[me->tamper_flg == TRUE]/
I_EEPROM_Manager_ee_write_request(me->itsI_EEPROM_Manager,EE_ODO_VALUE_NODE,me->odoBackupValue);

backup_enabled

backup_odometer

evOdoMasterValue/checkForTamper(me);

evIgnitionStatus[params->ignitionStatus==ACCESSORY]

evIgnitionStatus[params->ignitionStatus == RUN ||
params->ignitionStatus == START]

accessory

checkInitialMilage_flowchart   «FlowChart»

#define WAIT_FOR_MILEAGE 1000U /* 1000 km */

Tamper logic does not start until the odoMasterVlaue > WAIT_FOR_MILEAGE. This does not protect against continuing to re-programing the Master Odometer before WAIT_FOR_MILEAGE. Therefore, other logic contained in the MasterOdometer Source must protect against this.

[me->odoBackupValue > WAIT_FOR_MILEAGE]

[else]

[me->odoMasterValue > WAIT_FOR_MILEAGE]

[else]

me->odoBackupValue = me->odoMasterValue;

I_EEPROM_Manager_ee_write_request
  (me->itsI_EEPROM_Manager,EE_ODO_VALUE_NODE,me->odoBackupValue);

«FlowChart»

#define DO_NOTHING_VALUE 0U
#define MASTER_ODO_LT_THRESH 1U
#define MASTER_ODO_GT_THRESH 1000U

[me->odoMasterValue == DO_NOTHING_VALUE]

[me->odoMasterValue < (me->odoBackupValue - MASTER_ODO_LT_THRESH) ||
me->odoMasterValue > (me->odoBackupValue + MASTER_ODO_GT_THRESH)]

me->odoBackupValue = me->odoMasterValue;

me->tamper_flg = TRUE;