```
typedef
    cruiseStts_Type      = enum (csOn, csMaintain, csOff)
    throttleStts_Type    = enum (tsOff, tsMan, tsAuto)
    engineStts_Type      = enum (esOn, esOff)

agent vehicle {
    const
        mass = 1500    /* total mass of car & contents [kg] */
        windK = 10     /* wind resistance factor [N m-2 s2] */
        rollK = 100    /* rolling resistance factor [N m-1 s] */
        gravK = 9.81   /* acceleration due to gravity [N m2 s2] */
        brakK = 150    /* braking constant [N m-1 s] */
    state
        actSpeed     : analog      /* actual speed */
        accel        : analog      /* acceleration */
        windF        : analog      /* wind resistance force */
        rollF        : analog      /* rolling resistance force */
        gradF        : analog      /* gradient force */
        tracF        : analog      /* engine traction force */
        brakF        : analog      /* braking resistance force */
        brakePos     : analog (0.0, 1.0)      /* normalised */
        accelPos     : analog (0.0, 1.0)      /*   position   */
    oracle brakePos
    derivate
        windF    = windK * actSpeed2
        rollF    = rollK * actSpeed
        gradF    = gravK * mass * sin ( gradient * π / 200 )
        brakF    = brakK * actSpeed * brakePos
        tracF    = enginePower / actSpeed
        accel    = (tracF–brakF–gradF–rollF–windF) / mass
        actSpeed = integ_wrt_time (accel, 0)
}

agent engine {
    const  maxEnginePower = 74500        /* watts (i.e. approx 100 hp) */
    state
        engineStts       : engineStts_Type
        enginePower      : analog
    oracle       throttlePos   : analog
    derivate     enginePower = maxEnginePower * throttlePos
}
```

```
agent environment {
    state       gradient : analog (-25.0, 25.0)          /* realistic gradient [%] */
    derivate    gradient = rand ( analog (-25.0, 25.0) )
}

agent throttle_manager {

    agent autoThrottle_manager {
        const     initAutoThrottle = |accelPos|
        state
            deltaAutoThrottle       : analog
            autoThrottle            : analog
            speedErr                : analog
        derivate
            LIVE = (throttleStts == tsAuto)
            speedErr = cruiseSpeed - measSpeed
            deltaAutoThrottle = ((GainK ∗ speedErr) – autoThrottle) / TimeK
            autoThrottle = integ_wrt_time (deltaAutoThrottle, initAutoThrottle)
    }

    const
        GainK = 0.5    /* auto throttle controller gain */
        TimeK = 2.0    /* auto throttle controller time const. */
    state
        throttleStts     : throttleStts_Type
        throttlePos      : analog (0.0, 1.0)      /* normalised position */
    oracle
        measSpeed, cruiseSpeed,
        cruiseStts, engineStts, accelPos
    derivate
        throttlePos =      (throttleStts == tsOff) ? 0.0 :
                           (throttleStts == tsMan) ? accelPos :
                           (throttleStts == tsAuto) ?
                           min(max(autoThrottle, accelPos), 1)
    handle throttleStts
    protocol
        (engineStts == esOff) → throttleStts = tsOff
        (cruiseStts != csMaintain) ∧ (engineStts == esOn) → throttleStts = tsMan
        (cruiseStts == csMaintain) ∧ (engineStts == esOn) → throttleStts = tsAuto
}
```

```
agent speed_transducer {
    const
        wheelDiam       = 0.45      /* wheel diameter [m] */
        wheelCirc       = π ∗ wheelDiam   /* wheel circumference [m] */
        countPeriod     = 0.2       /* counter/timer period */
        maxCountVal     = 65535     /* 16-bit counter */
    state
        measSpeed     : analog
        pulseRate     : analog      /* wheel revs / sec [s-1] */
        countVal      : analog      /* counter/timer value [s-1] */
    derivate
        pulseRate     = actSpeed div wheelCirc      /* integer division */
        countVal      = (pulseRate ∗ countPeriod) mod maxCountVal
        measSpeed     = (countVal ∗ wheelCirc) / countPeriod
}

agent cruise_cutout {
    const     minCruiseSpeed = 20.0        /* miles/hour */
    state     cruiseStts       = csOff      : cruiseStts_Type
              cruiseSpeed      = minCruiseSpeed     : analog
    handle cruiseStts
    oracle    cruiseStts, brakePos
    derivate  braking = brakePos != 0.0
    protocol
        braking ∧ cruiseStts == csMaintain → cruiseStts = csOn      /* brake */
}

agent driver {
    const     maxCruiseSpeed      = 70.0      /* miles/hour */
              minCruiseSpeed      = 20.0      /* miles/hour */
    handle engineStts, cruiseStts, cruiseSpeed
    oracle
        engineStts, cruiseStts,
        cruiseSpeed, measSpeed, brakePos
    derivate
        brakePos = user_input (brakePos_Type)
        accelPos = user_input (accelPos_Type)
    protocol
        engineStts == esOff → engineStts = esOn     /* on */
        engineStts == esOn → engineStts = esOff     /* off */
        cruiseStts != csOff → cruiseStts = csOff       /* switch off cruise controller */
        cruiseStts == csOff → cruiseStts = csOn        /* switch on cruise controller */
        cruiseStts == csMaintain →  cruiseStts = csOn  /* return to manual control */
        cruiseStts == csOn →  cruiseStts = csMaintain  /* resume to cruiseSpeed */
        cruiseStts == csOn →  cruiseSpeed = | measSpeed |;  cruiseStts = csMaintain
                                            /* maintain the current speed */
        cruiseStts != csOff ∧ cruiseSpeed < maxCruiseSpeed
            → cruiseSpeed = | cruiseSpeed | + 1        /* increase cruiseSpeed */
        cruiseStts != csOff ∧ cruiseSpeed > minCruiseSpeed
```

```
        → cruiseSpeed = | cruiseSpeed | – 1        /* decrease cruiseSpeed */
}
```