

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!//                                                                 //
!// Transformation algorithm into the Japanese new transient engine test cycle //
!// --- for Diesel engine, FORTRAN version 1.1 --- //
!//                                                                 //
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
! *****
! MODULE DEF1
! Specification parameters
! *****
```

```
MODULE DEF1
  IMPLICIT NONE
  REAL(8), PARAMETER :: PDWT      = 0.07_8      ! rotational mass : tire,propeller = W0 * 7%
  REAL(8), PARAMETER :: PDWE      = 0.03_8      ! rotational mass : engine = W0 * 3%
  REAL(8), PARAMETER :: PMEET      = 0.05_8      ! clutch meet engine speed = 5% normalized
  REAL(8), PARAMETER :: PRELEASE    = 0.04_8      ! clutch release engine speed = 4% normalized
  REAL(8), PARAMETER :: PSGW       = 55.00_8     ! passenger's weight (kg)
  REAL(8), PARAMETER :: EGR_DIRECT = 0.98_8      ! transmit efficiency : directly connected
  REAL(8), PARAMETER :: EGR_ND     = 0.95_8      ! transmit efficiency : other gears, final gear
  REAL(8) :: nmins(1:5)             ! normalized minimum engine speed
  REAL(8) :: tmup8t(1:4)            ! torque margin(max_torque / torque), GVW>=8t
  REAL(8) :: tmunder8t(1:4)        ! torque margin, GVW<8t
  DATA nmins      / 0.05_8, 0.05_8, 0.11_8, 0.19_8, 0.26_8 / ! over 6th-gear is fixed to 0.26
  DATA tmup8t     / 2.0_8, 2.0_8, 1.7_8, 1.3_8      / ! value for over 4th-gear is 1.3
  DATA tmunder8t  / 2.4_8, 2.4_8, 1.7_8, 1.6_8     / ! value for over 4th-gear is 1.6
END MODULE DEF1
```

```
! *****
! MODULE DEF2
! Basic parameters for calculation
! *****
```

```
MODULE DEF2
  REAL(8), PARAMETER :: VERNO = 1.1_8
  INTEGER, PARAMETER :: THOLD = 3          ! minimum gear holding time
```

```

INTEGER,PARAMETER :: GRB   = 2           ! bottom gear
INTEGER,PARAMETER :: GRS   = 2           ! start gear
INTEGER,PARAMETER :: SKIPLIMIT =4       ! upper limit of shift skip
END MODULE DEF2

```

```

! *****
! MAIN PROGRAM convertD
! *****

```

```

PROGRAM convD
  USE DEF2
  IMPLICIT NONE

```

```

INTERFACE

```

```

  SUBROUTINE readpat ( uid, fname, len, t, vdest )
    CHARACTER( LEN = * ) fname
    INTEGER len, uid
    REAL(8), POINTER :: vdest(:)
    INTEGER, POINTER :: t(:)
  END SUBROUTINE

```

```

  SUBROUTINE reads ( uid, fname, w0, wld, bw, bh, ngr, gr, fgr, rt, nex, nrate, nidle, crew )
    INTEGER ngr, uid, crew
    REAL(8) w0, wld, bw, bh, fgr, rt, nex, nrate, nidle
    REAL(8),POINTER :: gr(:)
    CHARACTER( LEN = * ) fname
  END SUBROUTINE

```

```

  SUBROUTINE readtq ( uid, fname, rev, tq, ndata )
    INTEGER uid, ndata
    REAL(8),POINTER :: rev(:), tq(:)
    CHARACTER( LEN = * ) fname
  END SUBROUTINE

```

```

  SUBROUTINE writeres ( uid, fname, t, vdest, vreal, ne, te, nnorm, tnorm, sp, len )
    REAL(8) vreal(:), ne(:), te(:), nnorm(:), tnorm(:)

```

```

REAL(8),POINTER :: vdest(:)
INTEGER,POINTER :: t(:)
INTEGER sp(:), uid, len
CHARACTER( LEN = * ) fname

```

```
END SUBROUTINE
```

```

SUBROUTINE setparam ( w0, wld, bw, bh, nrate, nidle, ngr, gr, rt, crew, wt, dw, muR, muA, &
                    nes, nec, nel, tm, egr, efgr, tcl )

```

```

REAL(8) w0, wld, wt, muR, muA, efgr, tcl
REAL(8) nes, nec, nidle, nrate, rt, bw, bh
INTEGER ngr, crew
REAL(8),POINTER :: gr(:), egr(:), tm(:), nel(:), dw(:)

```

```
END SUBROUTINE
```

```

RECURSIVE SUBROUTINE runmode ( i, vdest, len, vpast, shiftp, theld, nrecur, nidle, nel, &
                             nes, nec, nex, tm, mass, dw, muR, muA, tcl, gr, fgr, egr, &
                             efgr, ngr, rv, tq, ntq, evi, eti, spi, vi, verr )

```

```

REAL(8),POINTER :: dw(:), nel(:), tm(:), egr(:), gr(:), rv(:), tq(:), vdest(:)
REAL(8) nes, nec, nidle, nex, vpast, mass
REAL(8) muR, muA, tcl, vi, evi, eti, verr, fgr, efgr
INTEGER nrecur, ntq, i, ngr, shiftp, spi, theld, len

```

```
END SUBROUTINE
```

```
REAL(8) FUNCTION maxtq ( r, tq, ndata, rev )
```

```

INTEGER ndata
REAL(8),POINTER :: r(:), tq(:)
REAL(8) rev

```

```
END FUNCTION maxtq
```

```
END INTERFACE
```

```
! ----- VARIABLE DEFINITION -----
```

```

CHARACTER( LEN = 512 ) :: patname= ' '           ! filename of pattern
CHARACTER( LEN = 512 ) :: specname= ' '         ! spec filename
CHARACTER( LEN = 512 ) :: torqname= ' '        ! filename of max torque
CHARACTER( LEN = 512 ) :: outputname=' '       ! dataset & output filename
REAL(8), POINTER :: vdest(:)                   ! pattern array
INTEGER, POINTER :: t(:)                       ! time

```

```

INTEGER len                ! number of pattern data
REAL(8), POINTER :: dw(:)  ! rotational weight (kg)
REAL(8) w0                 ! empty vehicle weight
REAL(8) payload            ! payload (kg)
REAL(8) wtest              ! test weight (kg)
REAL(8) rt, tcl            ! tire radius & circumference(m)
REAL(8) bw, bh             ! overall width & height(m)
REAL(8) muR, muA           ! rolling & air drag coefficient
REAL(8) nex, nrate, nidle  ! engine speed : maximum, rated, idle
REAL(8) nes, nec           ! clutch meet&release engine speed (rpm)
REAL(8), POINTER :: gr(:), egr(:) ! gear ratio, efficiency
REAL(8), POINTER :: tm(:)  ! torque margin
REAL(8), POINTER :: mins(:) ! minimum engine speed
REAL(8) fgr, efgr          ! final ratio & efficiency
INTEGER ngr                ! number of gear
INTEGER crew               ! crew (persons)
INTEGER ntq                ! number of torque data
REAL(8), POINTER :: rv(:), tq(:) ! maximum torque & engine speed

! ----- VARIABLE FOR OUTPUT -----
REAL(8), ALLOCATABLE :: vreal(:), ne(:), te(:) ! speed(kph), engine speed(rpm) & torque(Nm)
REAL(8), ALLOCATABLE :: nnorm(:), tnorm(:)    ! normalized engine speed & torque (%)
INTEGER, ALLOCATABLE :: sp(:)                 ! shift position
INTEGER i, spi, shiftp, theld
REAL(8) vi, vp, eti, evi, verr, maxt

! ----- READ INPUT DATA, SPEC, MAXIMUM TORQUE, TEST CYCLE -----
OPEN ( 11, file = 'DATA', status = 'OLD', ACCESS='SEQUENTIAL' ) ! open 'DATA' file
READ ( 11, fmt = "(a512)" ) patname ! test cycle
READ ( 11, fmt = "(a512)" ) specname ! spec data
READ ( 11, fmt = "(a512)" ) torqname ! maximum torque data
CLOSE ( 11 )

CALL readpat ( 12, patname, len, t, vdest )
CALL reads ( 13, specname, w0, payload, bw, bh, ngr, gr, fgr, rt, nex, nrate, nidle, crew )
CALL readtq ( 14, torqname, rv, tq, ntq )

```

```

! ----- CALCULATE AND DISPLAY VEHICLE SPECIFICATION -----
PRINT '("[ VERSION ",F4.1," ])", Verno
PRINT*
CALL setparam ( w0, payload, bw, bh, nrate, nidle, ngr, gr, rt, crew, wtest, dw, muR, muA,      &
               nes, nec, mins, tm, egr, efgr, tcl )

PRINT '(" W0      =",F8.2,"[kg], Wtest =",F8.2,"[kg]")', w0, wtest
PRINT '(" Width =",F8.3,"[m], Height=",F8.3,"[m], Tire radius=",F8.3,"[m]")', bw, bh, rt
PRINT '(" Crew =",I3)', crew
PRINT*
PRINT '(" Nidle =",F8.2,"[rpm], Nrate =",F8.2,"[rpm], Nex  =",F8.2,"[rpm]")', nidle, nrate, nex
PRINT '(" Nes   =",F8.2,"[rpm], Nec   =",F8.2,"[rpm]")', nes, nec
PRINT '(" MuAir =",F10.6," [kgf/(km/h)^2], MuRoll =",F10.6," [kgf/kg]")', muA, muR
PRINT*
PRINT '(" NUMBER OF GEAR =",I3)', ngr
PRINT '(" GEAR  RATIO EFFICIENCY TORQ MARGIN   DW[kg]")'
DO i = 1, ngr
    PRINT '(I4,": ",F8.3,F10.3,F12.3,F15.5)', i, gr(i), egr(i), tm(i), dw(i)
END DO
PRINT '(" FIN: ",F8.3,F10.3)', fgr, efgr
PRINT*

! ----- ALLOCATE MEMORY FOR OUTPUT -----
ALLOCATE ( vreal(len) )           ! calculated speed
ALLOCATE ( sp(len) )             ! shift position
ALLOCATE ( ne(len) )             ! engine speed (rpm)
ALLOCATE ( te(len) )             ! engine torque (Nm)
ALLOCATE ( nnorm(len) )          ! normalized engine speed (%)
ALLOCATE ( tnorm(len) )          ! normalized engine torque (%)

! ----- MAIN LOOP -----
theld = THOLD
shiftp = 0                       ! initial shift position = neutral
vp = vdest(1)                    ! initial speed = vdest(1)

DO i = 1, len
    verr = 0.0_8

```

```

CALL runmode ( i, vdest, len, vp, shiftp, theld, THOLD, nidle, mins, nes, nec,nex,      &
              tm, wtest, dw, muR, muA, tcl, gr, fgr, egr, efgr, ngr, rv, tq,      &
              ntq, evi, eti, spi, vi, verr)

maxt = maxtq ( rv, tq, ntq, evi )          ! maximum torque
vreal(i) = vi                             ! store calculation result
ne(i) = evi
te(i) = eti
nnorm(i) = ( ne(i) - nidle ) / ( nrate -nidle ) * 100.0_8
tnorm(i) = te(i) / maxt * 100.0_8
sp(i) = spi
shiftp = spi                              ! for next input
vp = vi
END DO

! ===== OUTPUT CALCULATION RESULT =====
DO WHILE ( outputname == '' )
  WRITE( *, '(A,$)' ) 'TYPE FILENAME FOR OUTPUT : '
  READ ( *, * ) outputname
END DO

CALL writeres ( 15, outputname, t, vdest, vreal, ne, te, nnorm, tnorm, sp, len )

! ===== DEALLOCATE MEMORY =====
DEALLOCATE ( vdest, t )
DEALLOCATE ( gr, egr, tm, mins )
DEALLOCATE ( vreal, ne, te, nnorm, tnorm )
DEALLOCATE ( sp )

STOP
END PROGRAM convD

! *****
! *
! * SUBROUTINE setparam : Fix all parameters *

```

```

! *
! *****
SUBROUTINE setparam ( w0, wld, bw, bh, nrate, nidle, ngr, gr, rt, crew, wt, dw,      &
                    muR, muA, nes, nec, nel, tm, egr, efgr, tcl )

    USE DEF1
    IMPLICIT NONE

! ===== VARIABLE DEFINITION =====
! [IN]-----
REAL(8) w0, wld          ! empty vehicle weight, payload(kg)
REAL(8) bw, bh          ! overall width & height(m)
REAL(8) nrate, nidle    ! rated, idle, clutch meet, release speed(rpm)
INTEGER ngr             ! number of gear
REAL(8), POINTER :: gr(:) ! gear ratio
REAL(8) rt              ! tire radius(m)
INTEGER crew            ! crew (persons)
! [OUT]-----
REAL(8) wt              ! test weight (kg)
REAL(8), POINTER :: dw(:) ! rotational weight (kg)
REAL(8) muR, muA        ! rolling & air drag coefficient
REAL(8) nes, nec        ! clutch meet&release engine speed (rpm)
REAL(8), POINTER :: nel(:) ! minimum engine speed
REAL(8), POINTER :: tm(:) ! torque margin
REAL(8), POINTER :: egr(:) ! gear efficiency
REAL(8) efgr            ! efficiency of final gear
REAL(8) tcl             ! tire circumference(m)
! [LOCAL]-----
REAL(8) gvw, dwt, dwe   ! GVW (kg)
INTEGER i

! ===== ALLOCATE MEMORY =====
ALLOCATE ( egr(ngr) )
ALLOCATE ( tm(ngr) )
ALLOCATE ( nel(ngr) )
ALLOCATE ( dw(ngr) )

! ===== WEIGHT, DRAG, TIRE =====

```

```

dwt = w0 * PDWT          ! tire&shaft rotational weight (kg)
dwe = w0 * PDWE          ! engine rotational weight (kg)
dw = dwt + dwe * gr * gr ! ROTATIONAL MASS OF EACH GEAR (KG)
wt = w0 + wld / 2.0_8 + PSGW ! TEST WEIGHT (KG)
gvw = w0 + wld + REAL(crew, 8) * PSGW ! GVW (kg)
muA = 0.00299_8 * bw * bh - 0.000832_8 ! AIR DRAG COEFFICIENT
muR = 0.00513_8 + ( 17.6_8 / wt ) ! ROLLING RESISTANCE COEFFICIENT
tcl = 2.0_8 * rt * 3.14_8 ! TIRE CIRCUMFERENCE (M)

```

```

PRINT '( " GVW   =", F8.2, "[kg]" )', gvw

```

```

! ===== CLUTCH ON/OFF, MINIMUM ENGINE SPEED (RPM) =====

```

```

nes = PMEET * ( nrate - nidle ) + nidle ! CLUTCH MEET (RPM)
nec = PRELEASE * ( nrate - nidle ) + nidle ! CLUTCH RELEASE (RPM)
DO i = 1, ngr
  IF ( i >= 6 ) THEN
    nel(i) = nmins(5) * ( nrate - nidle ) + nidle ! over 6thgear
  ELSE
    nel(i) = nmins(i) * ( nrate - nidle ) + nidle ! 1st-5th gear
  END IF
END DO

```

```

! ===== TORQUE MARGIN =====

```

```

DO i = 1, ngr
  IF ( gvw >= 8000.0_8 ) THEN ! GVW>=8t
    IF ( i >= 5 ) THEN
      tm(i) = tmup8t(4)
    ELSE
      tm(i) = tmup8t(i)
    END IF
  ELSE ! GVW<8t
    IF ( i >= 5 ) THEN
      tm(i) = tmunder8t(4)
    ELSE
      tm(i) = tmunder8t(i)
    END IF
  END IF
END IF

```



```

END DO

! ===== TRANSMISSION EFFICIENCY =====
efgr = egr_nd                                ! FINAL EFFICIENCY
DO i = 1, ngr
  IF ( gr(i) == 1.0_8 ) THEN
    egr(i) = EGR_DIRECT
  ELSE
    egr(i) = EGR_ND
  END IF
END DO

END SUBROUTINE setparam

! *****
! SUBROUTINE runmode : follow velocity pattern & output engine state
! *****
RECURSIVE SUBROUTINE runmode ( i, vdest, len, vpast, shiftp, thold, nrecur, nidle, nel, nes, nec, &
                             nex, tm, mass, dw, muR, muA, tcl, gr, fgr, egr, efgr, ngr, rv, tq, &
                             ntq, evi, eti, spi, vi, verr )

USE DEF2
IMPLICIT NONE

INTERFACE
  SUBROUTINE sftdwn ( i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex,gr,fgr,      &
                   egr,efgr,ngr,rv,tq,ntq,tcl,dw,muR, muA,mass, cnt, spiout, verr )
    INTEGER i, len, shiftp, ngr, ntq, spiout, cnt, reacc
    REAL(8), POINTER :: vdest(:), rv(:), tq(:), gr(:), egr(:), nel(:), dw(:)
    REAL(8) vpast, nes, nec, nex, fgr, efgr, tcl, muR,muA, mass,nidle,verr
  END SUBROUTINE

  SUBROUTINE sftuptm ( reacc, i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex, gr, fgr,&
                    egr, efgr ,ngr, rv, tq, ntq, tcl, dw, tm, muR, muA, mass, cnt, spiout, verr )
    INTEGER i, len, shiftp, ngr, ntq, thold, spiout, cnt, reacc
    REAL(8), POINTER :: vdest(:), rv(:), tq(:), gr(:), egr(:), nel(:), dw(:), tm(:)
  END SUBROUTINE

```

```
REAL(8) vpast, nes, nec, nex, fgr, efgr, tcl, muR, muA, mass, nidle, verr
END SUBROUTINE
```

```
SUBROUTINE sftup ( reacc, i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex, gr, fgr, &
                 egr, efgr, ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, spiout, minerr )
  INTEGER i, len, shiftp, ngr, ntq, spiout, reacc
  REAL(8), POINTER :: vdest(:), rv(:), tq(:), gr(:), egr(:), nel(:), dw(:)
  REAL(8) vpast, nes, nec, nex, fgr, efgr, tcl, muR, muA, mass, nidle, minerr
END SUBROUTINE
```

```
SUBROUTINE engstat ( nex, nel, enginev, enginet, shiftp, ngr, gr, fgr, tcl, mass, dw, muR, &
                  muA, vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
  REAL(8) enginev, enginet, fgr, tcl, mass, muR, muA, vdest
  REAL(8) efgr, maxt, nidle, nes, nex, nec, vpast, vv
  REAL(8), POINTER :: dw(:), rv(:), tq(:), gr(:), egr(:), nel(:)
  INTEGER shiftp, sw, ntq, ngr
END SUBROUTINE engstat
```

```
REAL(8) FUNCTION drvfr ( w, dw, muR, muA, sp, v1, v2 )
  INTEGER sp
  REAL(8) w, muR, muA, v1, v2
  REAL(8), POINTER :: dw(:)
END FUNCTION drvfr
```

```
INTEGER FUNCTION startgear ( i, vdest, nex, nel, nidle, nes, nec, gr, ngr, fgr, tcl, &
                          mass, dw, muR, muA, egr, efgr, rv, tq, ntq, stime )
  INTEGER i, ngr, ntq, stime
  REAL(8) nex, fgr, tcl, mass, muR, muA, efgr, nes, nidle, nec
  REAL(8), POINTER :: dw(:), gr(:), egr(:), rv(:), tq(:), nel(:), vdest(:)
END FUNCTION startgear
```

```
END INTERFACE
```

```
! ===== VARIABLE DEFINITION =====
! [IN]-----
INTEGER i                ! time step i
REAL(8), POINTER :: vdest(:) ! velocity pattern
INTEGER len              ! pattern length
```

```

REAL(8) vpast          ! vehicle speed (time i-1)
INTEGER shiftp        ! last gear position (time i-1)
INTEGER theld         ! gear holding time(s)
INTEGER nrecur        ! number of recursion
REAL(8) nes, nec, nidle, nex ! clutch meet&release, idle&maximum engine speed
REAL(8),POINTER :: nel(:) ! minimum(each gear) & maximum engine speed
REAL(8),POINTER :: tm(:) ! torque margin array
REAL(8) mass          ! test weight(kg)
REAL(8),POINTER :: dw(:) ! rotational weight for each gear
REAL(8) muR, muA, tcl ! rolling&air drag coefficient,tire circumference
REAL(8),POINTER :: gr(:), egr(:) ! gear ratio, transmit efficiency
REAL(8) fgr, efgr     ! gear ratio, transmit efficiency (final)
REAL(8),POINTER :: rv(:), tq(:) ! maximum torque dataset
INTEGER ntq, ngr      ! number of torque data, number of gear
! [OUT]-----
REAL(8) vi, evi, eti  ! speed(km/h), engine speed(rpm), torque(Nm)
INTEGER spi           ! gear position (time i)
! [IN & OUT]-----
REAL(8) verr          ! velocity error
! [LOCAL]-----
REAL(8) errmin        ! minimum error
INTEGER sw            ! engine condition code
REAL(8) maxt          ! maximum engine torque
REAL(8) tm1           ! calculated torque margin
INTEGER s             ! shift position
REAL(8) dv            ! velocity difference
INTEGER reacc         ! reacceleration (0:OFF, 1:ON)
INTEGER spi2, theld2, ct, sw2,s2 ! for recursion
REAL(8) ev2, et2, vi2, verr2, verr3
INTEGER stime        ! start time
INTEGER j

! ===== DETECT END OF RECURSIVE CALCULATION =====
ct = nrecur - 1      ! recursion counter
IF( ( ct < 0 ) .OR. ( i > len ) ) THEN
  spi = shiftp
  verr = verr

```



```

! ===== DECELERATE WITH NEUTRAL POSITION =====
ELSE                                     ! decelerate
    vi = vdest(i)                        ! calculate speed = target
    evi = nidle                          ! neutral
    eti = 0.0_8                          ! engine torque = 0(Nm)
    theld = THOLD                        ! enable shift change
    verr = verr + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
    RETURN
END IF

! ===== PAST SHIFT POSITION WAS NOT NEUTRAL =====
ELSE
! ---- Check if Ne is in range with shift[i-1] ----
ev2 = vdest(i) * 60.0_8 * gr(shiftp) * fgr / ( 3.6_8 * tcl )
IF ( ( dv >= 0.0_8 ) .AND. ( ev2 < nel(shiftp) ) .AND. ( shiftp > GRB ) ) THEN
    spi = GRB                            ! shift to bottom gear
    reacc = 1                            ! reacceleration mode ON
ELSE
    spi = shiftp                          ! normal driving
END IF
END IF

! ===== CALCULATE ENGINE RUNNING CONDITION =====
! ===== GEAR POSITION SPI IS DETERMINED BASED ON THIS CALCULATION RESULT =====
CALL engstat ( nex, nel, evi, eti, spi, ngr, gr, fgr, tcl, mass, dw, muR, muA, vdest(i),    &
    vpast, vi, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )

! ===== CASE THAT CANNOT CHANGE GEAR POSITION, DECELERATION , THELD < THOLD =====
IF ( ( theld < THOLD ) .OR. ( dv < 0.0_8 ) ) THEN
    IF ( ( sw == 4 ) .OR. ( sw == 3 ) ) THEN
        spi = 0                          ! release clutch
        theld = THOLD                    ! reset hold time
        RETURN                            ! STOP RECURSION
    ELSE IF ( sw == 2 ) THEN              ! over run
        IF ( spi < ngr ) THEN             ! if spi is not top gear
            vi = 0.0_8                    ! ** CANNOT CONTINUE CALCULATION **
            RETURN                        ! OUTPUT 0km/h AS THE ABNORMAL RESULT
        END IF
    END IF
END IF

```

```

        END IF
    END IF
! ----- CALCULATE CUMULATIVE ERROR -----
verr2 = verr + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
theld2 = theld + 1
CALL runmode ( i+1, vdest, len, vi, spi, theld2, ct, nidle, nel, nes, nec, nex, tm, mass, &
              dw, muR, muA, tcl, gr, fgr, egr, efgr, ngr, rv, tq, ntq, ev2, et2, spi2, vi2, verr2)
verr = verr2                ! total error
theld = theld + 1          ! increment gear hold time
RETURN
END IF

! ===== CASE THAT CAN CHANGE GEAR POSITION, REACCELERATION, ( THELD>=THOLD & DV>=0 ) ==
! ===== REACCELERATION =====
IF ( reacc == 1 ) THEN
! ===== TRY SHIFT-UP WITH TORQUE MARGIN COMPARISON =====
call sftuptm ( reacc, i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex, gr, fgr, egr, &
              efgr, ngr, rv, tq, ntq, tcl, dw, tm, muR, muA, mass, ct, spi2, verr2 )
IF ( spi < spi2 ) THEN
    spi = spi2
    errmin = verr + verr2
ELSE
! ----- FIND OPTIMAL GEAR THAT CAN KEEP 3 SECONDS -----
call sftup( reacc, i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex,          &
           gr, fgr, egr, efgr, ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, s2, verr2 )
    spi = s2
    errmin = verr + verr2
END IF

! ===== OVER RUN. SHIFT-UP REQUIRED =====
ELSE IF ( sw == 2 ) THEN                ! over run
    IF ( spi < ngr )THEN
! ===== TRY SHIFT-UP WITH TORQUE MARGIN COMPARISON =====
call sftuptm ( reacc, i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex, gr, fgr, &
              egr, efgr, ngr, rv, tq, ntq, tcl, dw, tm, muR, muA, mass, ct, s2, verr2 )
IF ( spi == s2 ) THEN
! ----- FIND OPTIMAL GEAR THAT CAN KEEP 3 SECONDS -----

```

```

        call sftup( reacc, i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex,      &
                  gr, fgr, egr, efgr, ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, s2, verr2 )
    END IF
! ----- ENGINE RUNNING CONDITION WITH OPTIMAL GEAR POSITION S2 -----
    CALL engstat ( nex, nel, evi, eti, s2, ngr, gr, fgr, tcl, mass, dw, muR, muA, vdest(i), &
                 vpast, vi, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )

    spi = s2
    theld2 = 1                                ! update gear hold time
ELSE
    theld2 = theld + 1                        ! top gear
END IF
! ----- CALCULATE CUMULATIVE ERROR -----
    verr2 = verr + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
    CALL runmode ( i+1, vdest, len, vi, spi, theld2, ct, nidle, nel, nes, nec, nex, tm, mass, &
                 dw, muR, muA, tcl, gr, fgr, egr, efgr, ngr, rv, tq, ntq, ev2, et2, spi2, vi2, verr2 )
    errmin = verr2                            ! minimum error when gear was not changed

! ===== CASE THAT KEEPS SAME GEAR POSITION =====
ELSE
    theld2 = theld + 1
    verr2 = verr + ( vdest(i-1) + vdest(i) ) / 2.0_8 - ( vi + vpast ) / 2.0_8
! ----- CALCULATE CUMULATIVE ERROR -----
    CALL runmode ( i+1, vdest, len, vi, spi, theld2, ct, nidle, nel, nes, nec, nex, tm, mass, &
                 dw, muR, muA, tcl, gr, fgr, egr, efgr, ngr, rv, tq, ntq, ev2, et2, spi2, vi2, verr2 )
    errmin = verr2                            ! minimum error when gear is not changed
! ===== TRY SHIFT-UP WITH TORQUE MARGIN COMPARISON =====
    call sftuptm ( reacc, i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex, gr, fgr, egr, &
                 efgr, ngr, rv, tq, ntq, tcl, dw, tm, muR, muA, mass, ct, spi2, verr2 )
    IF ( ( spi2 > spi ) .AND. ( errmin >= ( verr + verr2 ) ) ) THEN
        spi = spi2
        errmin = verr + verr2
    END IF
! ===== TRY SHIFT-DOWN =====
    IF ( ( shiftp == spi ) .AND. ( sw == 1 ) ) THEN
        CALL sftdwn ( i, vdest, len, vpast, spi, nidle, nel, nes, nec, nex, gr, fgr, &
                    egr, efgr, ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, ct, s2, verr2 )
        IF ( ( spi > s2 ) .AND. ( errmin > ( verr + verr2 ) ) ) THEN

```

```

        spi = s2
        errmin = verr + verr2
    END IF
END IF
END IF

```

```

! ===== FINAL CONDITION, GEAR POSITION IS SPI =====
CALL engstat ( nex, nel, evi, eti, spi, ngr, gr, fgr, tcl, mass, dw, muR, muA, vdest(i), vpast, &
              vi, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
IF ( ( dv >= 0.0_8 ) .AND. ( evi < nes ) ) THEN
    evi = nes                ! This is for only the case : spi=GRB & reacc=1
ELSE IF ( evi < nec ) THEN ! clutch release
    evi = nidle             ! idling speed
    eti = 0.0_8
    spi = 0                 ! neutral
END IF

IF ( spi == 0 ) THEN
    theld = THOLD
ELSE IF ( shiftp /= spi ) THEN
    theld = 1                ! reset gear hold time.
ELSE
    theld = theld + 1        ! increment hold time
END IF
verr = errmin

```

```

END SUBROUTINE runmode

```

```

! *****
! *
! FUNCTION startgear : Chose start gear
! *
! *****
INTEGER FUNCTION startgear ( i, vdest, nex, nel, nidle, nes, nec, gr, ngr, fgr, tcl, mass, dw, &

```


muR, muA, egr, efgr, rv, tq, ntq, stime)

USE DEF2

IMPLICIT NONE

INTERFACE

SUBROUTINE engstat (nex, nel, enginev, enginet, sp, ngr, gr, fgr, tcl, mass, dw, muR, muA, &
vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw)

REAL(8) enginev, enginet, fgr, tcl, mass, muR, muA, vdest, vpast, vv

REAL(8) efgr, maxt, nidle, nes, nec, nex

REAL(8), POINTER :: dw(:), rv(:), tq(:), gr(:), egr(:), nel(:)

INTEGER sp, sw, ntq, ngr

END SUBROUTINE engstat

REAL(8) FUNCTION maxtq (rv, tq, ndata, rev)

INTEGER ndata

REAL(8), POINTER :: rv(:), tq(:)

REAL(8) rev

END FUNCTION maxtq

END INTERFACE

! ===== VARIABLE DEFINITION =====

! [IN]-----

INTEGER i ! current time step
REAL(8), POINTER :: vdest(:) ! pattern array
REAL(8) mass ! vehicle weight (kg)
REAL(8), POINTER :: dw(:) ! rotational weight (kg)
REAL(8) tcl ! tire circumference(m)
REAL(8) muR, muA ! rolling & air drag coefficient
REAL(8) nex, nidle ! engine speed : maximum, idle
REAL(8) nes, nec ! clutch meet&release engine speed (rpm)
REAL(8), POINTER :: nel(:) ! minimum engine speed (rpm)
REAL(8), POINTER :: gr(:), egr(:) ! gear ratio, efficiency
REAL(8) fgr, efgr ! final ratio & efficiency
INTEGER ngr ! number of gear
REAL(8), POINTER :: rv(:), tq(:) ! maximum torque data set
INTEGER ntq ! number of torque data

! [OUT]-----


```

! * sw=1 less torque, engine speed is in range *
! * sw=2 over rev. torque is inrange *
! * sw=3 less rev. This case occurs when vehicle has to re-accelerate *
! * sw=4 clutch release *
! * sw=5 start mode. Ne <= Nes, shift = 1st or 2nd *
! * *
! *****

```

```

SUBROUTINE engstat ( nex, nel, enginev, enginet, sp, ngr, gr, fgr ,tcl, mass, dw, muR, muA, &
                   vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )

```

```

USE DEF2
IMPLICIT NONE

```

```

INTERFACE

```

```

REAL(8) FUNCTION maxtq ( rv, tq, ndata, rev )
  INTEGER ndata
  REAL(8),POINTER :: rv(:), tq(:)
  REAL(8) rev
END FUNCTION maxtq

```

```

REAL(8) FUNCTION drvfr ( w, dw, muR, muA, sp, v1, v2 )
  IMPLICIT NONE
  INTEGER sp
  REAL(8) w, muR, muA, v1, v2
  REAL(8),POINTER :: dw(:)
END FUNCTION drvfr

```

```

END INTERFACE

```

```

! ===== VARIABLE DEFINITION =====

```

```

! [IN]-----
REAL(8) mass           ! vehicle weight (kg)
REAL(8), POINTER :: dw(:) ! rotational weight (kg)
REAL(8) tcl           ! tire circumference(m)
REAL(8) muR, muA      ! rolling & air drag coefficient
REAL(8) nex, nidle    ! maximum, idle speed (rpm)
REAL(8) nes, nec      ! clutch meet&release speed (rpm)
REAL(8), POINTER :: nel(:) ! minimum engine speed (rpm)
REAL(8), POINTER :: gr(:), egr(:) ! gear ratio, efficiency

```

```

REAL(8) fgr, efgr          ! final ratio & efficiency
INTEGER ngr                ! number of gear
REAL(8) vpast, vdest      ! current & target speed (km/h)
REAL(8),POINTER :: rv(:), tq(:) ! maximum torque data set
INTEGER ntq                ! number of torque data
INTEGER sp                 ! current shift position
! [OUT]-----
REAL(8) enginev, engine   ! engine speed(rpm), torque(Nm)
REAL(8) maxt              ! maximum torque(Nm)
REAL(8) vv                ! calculated vehicle speed (km/h)
INTEGER sw                ! RETURN CODE
! [LOCAL]-----
REAL(8) dv, tqdif, df, ds
INTEGER flg

! ===== CALCULATE INITIAL CONDITION =====
vv = vdest
df = drvfr ( mass, dw, muR, muA, sp, vpast, vv )
enginev = df * tcl / ( 2.0_8 * 3.14_8 * gr(sp) * fgr * egr(sp) * efgr )
enginev = ( vv * 60.0_8 ) * ( gr(sp) * fgr ) / ( tcl * 3.6_8 )
dv = vv - vpast

! ===== DECELERATION =====
IF ( dv < 0.0_8 ) THEN
  IF ( enginev < nec ) THEN          ! release clutch, idling
    enginev = nidle
    enginev = 0.0_8
    sw = 4                          ! [[ RETURN CODE = 4 ]]
  ELSE
    sw = 0                          ! [[ RETURN CODE = 0 ]] normal
  END IF
  maxt = maxtq ( rv, tq, ntq, enginev )
  RETURN
END IF

! ===== ACCELERATION OR CONSTANT SPEED =====
! ===== ENGINE SPEED IS LOWER THAN MINIMUM SPEED =====

```

```

IF ( enginev < nel(sp) ) THEN                ! enginev < minimum engine speed
  IF ( sp > GRB) THEN                          ! above 3rd gear
    vv = 0.0_8                                ! ** CANNOT DRIVE WITH THIS CONDITION **
    maxt = maxtq ( rv, tq, ntq, nel(sp) )      ! maximum torque at nel(shiftp)
    sw = 3                                     ! [[ RETURN CODE = 3 ]]
    RETURN
  ELSE                                         ! 1st or 2nd gear
    enginev = nes                              ! clutch meet
    maxt = maxtq ( rv, tq, ntq, enginev )
    IF ( enginet > maxt ) THEN                 ! poor torque
      sw = 1                                   ! [[ RETURN CODE = 1 ]], GOTO CONVERGENT PROCESS
    ELSE
      sw = 5                                   ! [[ RETURN CODE = 5 ]] starting
    RETURN
  END IF
END IF
END IF
! ===== ENGINE SPEED IS HIGHER THAN MAXIMUM SPEED =====
ELSE IF ( enginev >= nex ) THEN
  IF ( sp == ngr )THEN
    enginev = nex                             ! case of top gear
  END IF
  vv = enginev * tcl * 3.6_8 / ( 60.0_8 * gr(sp) * fgr )
  df = drvfrc ( mass, dw, muR, muA, sp, vpast, vv )
  enginet = df * tcl / ( 2.0_8 * 3.14_8 * gr(sp) * fgr * egr(sp) * efgr )
  maxt = maxtq ( rv, tq, ntq, enginev )
  IF ( enginet > maxt ) THEN
    sw = 1                                     ! [[ RETURN CODE = 1 ]], GOTO CONVERGENT PROCESS
  ELSE
    sw = 2                                     ! [[ RETURN CODE = 2 ]], OVER RUN
  RETURN
END IF
! ===== MINIMUM <= ENGINE SPEED < MAXIMUM =====
ELSE                                         ! mins(s) <= enginev <= maxs
  maxt = maxtq ( rv, tq, ntq, enginev )
  IF ( enginet > maxt ) THEN
    sw = 1                                     ! [[ RETURN CODE = 1 ]], GOTO CONVERGENT PROCESS
  ELSE

```

```

        sw = 0                                ! [[ RETURN CODE = 0 ]] normal
        RETURN
    END IF
END IF

```

```

! ===== CONVERGENT CALCULATION OF VEHICLE SPEED =====

```

```

IF ( engine_t > max_t ) THEN                    ! poor torque, sw=1
    ds = 1.0_8
    flg = 0
    DO WHILE ( flg == 0 )
        df = drvfrct ( mass, dw, muR, muA, sp, vpast, vv )
        engine_t = df * tcl / ( 2.0_8 * 3.14_8 * gr(sp) * fgr * egr(sp) * efgr )
        engine_v = vv * 60.0_8 * gr(sp) * fgr / ( tcl * 3.6_8 )

        IF ( ( sp > GRB ) .AND. ( engine_v < nel(sp) ) ) THEN
            max_t = maxtq ( rv, tq, ntq, nel(sp) )
        ELSE IF ( engine_v < nes ) THEN
            max_t = maxtq ( rv, tq, ntq, nes )
            engine_v = nes
        ELSE
            max_t = maxtq ( rv, tq, ntq, engine_v )
        END IF

        tqdif = max_t - engine_t                ! difference between max_t and engine_t

        IF ( ( tqdif < 1.0E-6 ) .AND. ( tqdif >= 0.0_8 ) ) THEN
            flg = 1
        ELSE IF ( tqdif < 0.0_8 ) THEN
            vv = vv - ds
        ELSE
            ds = ds / 2.0_8
            vv = vv + ds
        END IF
    END DO
END IF

IF ( engine_v < nel(sp) ) THEN                  ! engine_v < minimum speed

```



```

ELSE
  DO x = 1, ndata                                ! interpolation
    IF ( ( rev >= r(x) ) .AND. ( rev < r(x+1) ) ) EXIT
  END DO
END IF

! ===== OUTPUT =====
maxtq = tq(x) + ( tq(x+1) - tq(x) ) / ( r(x+1) - r(x) ) * ( rev - r(x) )

END FUNCTION maxtq

! *****
! *
! * FUNCTION drvfrc
! * Calculate driving force when vehicle accelerates from v1 to v2
! *
! *****

REAL(8) FUNCTION drvfrc ( w, dw, muR, muA, sp, v1, v2 )
  IMPLICIT NONE

! ===== VARIABLE DEFINITION =====
! [IN]-----
  INTEGER sp                                ! shift position
  REAL(8) w                                  ! vehicle weight (kg)
  REAL(8),POINTER :: dw(:)                 ! rotational weight (kg)
  REAL(8) muR, muA                          ! rolling&air drag coefficient
  REAL(8) v1, v2                             ! current & target speed (km/h)
! [LOCAL]-----
  REAL(8) rr, tm, acc

  rr = 9.8_8 * ( muR * w + muA * ( v2 * v2 ) ) ! running resistance(N)
  tm = w + dw(sp)                             ! total mass (kg)
  acc = ( v2 - v1 ) / ( 3.6_8 * 1.0_8 )       ! acceleration (m/s2)

! ===== OUTPUT =====

```



```
drvfrc = rr + tm * acc          ! driving force (N)
```

```
END FUNCTION drvfrc
```

```
! *****  
! * * * * *  
! * SUBROUTINE readpat : readout test cycle *  
! * * * * *  
! *****  
SUBROUTINE readpat ( uid, fname, len, t, vdest )  
  IMPLICIT NONE  
  
! ===== VARIABLE DEFINITION =====  
! [IN]-----  
  INTEGER uid          ! unit ID  
  CHARACTER( LEN = * ) fname ! file name  
! [OUT]-----  
  INTEGER len          ! number of data  
  INTEGER, POINTER :: t(:) ! time  
  REAL(8), POINTER :: vdest(:) ! test cycle  
! [LOCAL]-----  
  INTEGER flg, i  
  REAL(8) timeT, vdestT ! dummy for counting number of data  
  CHARACTER(len=512) :: tmp  
  
! ===== COUNT NUMBER OF DATA =====  
  OPEN ( uid, FILE = fname, STATUS = 'OLD', ACCESS='SEQUENTIAL' )  
    READ( uid, '(A512)') tmp ! skip header  
    len = 0 ! initialize mode length  
    flg = 0 ! I/O status  
    DO WHILE ( flg == 0 )  
      READ ( uid,*, IOSTAT = flg ) timeT,vdestT ! read 1 line  
      SELECT CASE ( flg ) ! error check  
      CASE ( 0 )  
        len = len + 1 ! increment number of data
```

```

CASE ( 1: )                                ! if error occurred
WRITE ( 0 ) 'failed to read the pattern file. Check value at Line(',len+1,')'
CLOSE ( uid )
STOP
END SELECT
END DO
CLOSE ( uid )

```

```

! ===== ALLOCATE MEMORY =====
ALLOCATE ( vdest(0:len) )                  ! speed
ALLOCATE ( t(1:len) )                      ! time

```

```

! ===== READ TEST CYCLE =====
OPEN ( uid, FILE = fname, STATUS = 'OLD', ACCESS='SEQUENTIAL' )
READ( uid ,'(A512)') tmp                   ! skip header
DO i = 1, len
READ ( uid,* ) t(i), vdest(i)
END DO
CLOSE ( uid )

```

```

vdest(0) = vdest(1)

```

```

END SUBROUTINE readpat

```

```

! *****
! *
! * SUBROUTINE reads : Readout specification data
! *
! *****

```

```

SUBROUTINE reads ( uid, fname, w0, wld, bw, bh, ngr, gr, fgr, rt, nex, nrate, nidle, crew )
IMPLICIT NONE

```

```

! ===== VARIABLE DEFINITION =====
! [IN]-----
INTEGER uid                                ! unit ID

```

```

CHARACTER ( LEN = * ) fname                ! file name
! [OUT]-----
REAL(8) w0, wld                            ! empty vehicle weight, payload(kg)
INTEGER crew                               ! crew (persons)
REAL(8) bw, bh                             ! overall width & height(m)
REAL(8) rt                                 ! tire radius(m)
INTEGER ngr                                ! number of gear
REAL(8), POINTER :: gr(:)                 ! gear ratio
REAL(8) fgr                                ! final ratio
REAL(8) nex, nrate, nidle                 ! max, rate, idle engine speed(rpm)
! [LOCAL]-----
INTEGER i

! ===== READ SPECIFICATION DATA =====
OPEN ( uid, FILE = fname, STATUS = 'OLD', ACCESS='SEQUENTIAL', FORM='FORMATTED' )
  READ ( uid, * ) w0                        ! empty vehicle weight (kg)
  READ ( uid, * ) wld                      ! payload (kg)
  READ ( uid, * ) crew                    ! crew (persons)
  READ ( uid, * ) bh                      ! overall height (m)
  READ ( uid, * ) bw                      ! overall width (m)
  READ ( uid, * ) rt                      ! tire radius (m)
  READ ( uid, * ) ngr                    ! number of gear

  ALLOCATE( gr(ngr) )                    ! gear ratio array
  DO i = 1, ngr
    READ ( uid, * ) gr(i)                ! read gear ratio
  END DO

  READ ( uid, * ) fgr                    ! final ratio
  READ ( uid, * ) nidle                  ! idling speed(rpm)
  READ ( uid, * ) nrate                  ! rated speed (rpm)
  READ ( uid, * ) nex                    ! maximum speed (rpm)
CLOSE ( uid )

END SUBROUTINE reads

```

```

! *****
! *
! *SUBROUTINE readtq : readout maximum torque data
! *
! *****
SUBROUTINE readtq ( uid, fname, rev, tq, ndata )
  IMPLICIT NONE

! ===== VARIABLE DEFINITION =====
! [IN]-----
  INTEGER uid                ! unit ID
  CHARACTER( LEN = * ) fname ! file name
! [OUT]-----
  REAL(8),POINTER :: rev(:), tq(:) ! engine speed - maximum torque
  INTEGER ndata                ! number of data
! [LOCAL]-----
  INTEGER i, j, flg, gap
  REAL(8) revT, tmaxT
  CHARACTER(len=512) :: tmp

! ===== COUNT NUMBER OF DATA =====
OPEN ( uid, FILE = fname, STATUS = 'OLD', ACCESS='SEQUENTIAL' )
  READ ( uid,'(A512)' ) tmp ! skip header
  ndata = 0
  flg = 0 ! I/O status
  DO WHILE ( flg == 0 )
    READ ( uid, *, IOSTAT = flg ) revT, tmaxT ! read 1 line
    SELECT CASE ( flg ) ! error check
      CASE ( 0 )
        ndata = ndata + 1 ! increment number of data
      CASE ( 1: ) ! if error occurred
        WRITE ( 0, * ) 'Cannot read torque file. Check value at Line(',ndata+1,')'
        CLOSE ( uid )
        STOP
    END SELECT
  END DO
END DO

```

```

CLOSE ( uid )

! ===== ALLOCATE MEMORY =====
ALLOCATE ( tq(ndata) )
ALLOCATE ( rev(ndata) )

! ===== READ MAXIMUM TORQUE DATA =====
OPEN ( uid, FILE = fname, STATUS = 'OLD', ACCESS='SEQUENTIAL' )
  READ (uid,'(A512)') tmp                ! skip header
  DO i = 1, ndata
    READ ( uid, * ) rev(i), tq(i)        ! read out torque data
  END DO
CLOSE (uid )

! ===== SORT TORQUE DATA IN ASCENDING ORDER =====
gap = ( ndata + 1 ) / 2
DO WHILE ( gap /= 0 )
  i = gap
  DO WHILE ( i <= ndata )
    j = i - gap
    DO WHILE ( ( j >= 1 ) .AND. ( rev(j) > rev(j+gap) ) )
      revT      = rev(j)
      rev(j)    = rev(j+gap)
      rev(j+gap) = revT
      tmaxT     = tq(j)
      tq(j)     = tq(j+gap)
      tq(j+gap) = tmaxT
      j = j - gap
    END DO
    i = i + 1
  END DO
  gap = gap / 2
END DO

END SUBROUTINE readtq

```

```

! *****
! *
! * SUBROUTINE writeres : output result
! *
! *****
SUBROUTINE writeres ( uid, fname, t, vdest, vreal, ne, te, nnorm, tnorm, sp, len )
  IMPLICIT NONE

! ===== VARIABLE DEFINITION =====
! [IN]-----
  INTEGER uid, len, sp(:)           ! unit, length, shift
  CHARACTER ( LEN = * ) fname      ! filename
  INTEGER, POINTER :: t(:)         ! time label
  REAL(8), POINTER :: vdest(:)     ! target speed(kph)
  REAL(8) vreal(:), ne(:), te(:), nnorm(:), tnorm(:) ! result
! [LOCAL]-----
  INTEGER i
  CHARACTER, PARAMETER :: ht = CHAR(9) ! horizontal tab

! ===== OUTPUT TO FILE =====
OPEN ( uid, FILE = fname, STATUS = 'UNKNOWN' )

  WRITE ( uid, '(7A,$)' ) 'time(s)', ht, 'Vtarget(km/h)', ht, 'Vreal(km/h)', ht, 'Ne(rpm)'
  WRITE ( uid, '(8A)' ) ht, 'Te(N-m)', ht, 'N_norm(%)', ht, 'T_norm(%)', ht, 'Shift'

  DO i = 1, len
    IF ( te(i) < 0.0_8 ) THEN           ! motoring
      WRITE ( uid, FMT = '(I4,2(1A,F6.2),1A,F8.1,1A,1A,1A,F6.2,1A,1A,1A,I2)' ) t(i), ht, &
        vdest(i), ht, vreal(i), ht, ne(i), ht, 'M', ht, nnorm(i), ht, 'M', ht, sp(i)
    ELSE
      WRITE ( uid, FMT = '(I4,2(1A,F6.2),2(1A,F8.1),2(1A,F6.2),1A,I2)' ) t(i), ht, vdest(i)&
        , ht, vreal(i), ht, ne(i), ht, te(i), ht, nnorm(i), ht, tnorm(i), ht, sp(i)

    END IF
  END DO

```

```
CLOSE ( uid )
```

```
END SUBROUTINE writeres
```

```
! *****
! *
! * SUBROUTINE sftuptm : calculation of shift up considering torque margin *
! * vehicle can shift-up when *
! * 1.calculated torque margin >= defined value *
! * 2.AND can drive at least 3 seconds with same position *
! *
! *****
SUBROUTINE sftuptm(reacc,i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex, gr, fgr, egr, &
                 efgr, ngr, rv, tq, ntq, tcl, dw, tm, muR, muA, mass, cnt, spiout, minerr )
  USE DEF2
  IMPLICIT NONE

  INTERFACE
    SUBROUTINE engstat ( nex, nel, enginev, enginev, shiftp, ngr, gr, fgr, tcl, mass, dw, muR, &
                      muA, vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
      REAL(8) enginev, enginev, fgr, tcl, mass, muR, muA, vdest
      REAL(8) efgr, maxt, nidle, nes, nex, nec, vpast, vv
      REAL(8), POINTER :: dw(:), rv(:), tq(:), gr(:), egr(:), nel(:)
      INTEGER shiftp, sw, ntq, ngr
    END SUBROUTINE engstat
  END INTERFACE

! ===== VARIABLE DEFINITION =====
! [IN]-----
INTEGER reacc           ! reacceleration mode
INTEGER i               ! current time step
REAL(8), POINTER :: vdest(:) ! test cycle
```

```

INTEGER len                ! number of data
REAL(8) vpast              ! vehicle speed (time i-1)
INTEGER shiftp             ! last gear position (time i-1)
REAL(8) nes, nec, nidle, nex ! clutch meet,release,idle,maximum speed
REAL(8),POINTER :: nel(:)  ! minimum(each gear) engine speed
REAL(8),POINTER :: gr(:), egr(:) ! gear ratio, transmit efficiency
REAL(8) fgr, efgr          ! gear ratio, transmit efficiency (final)
REAL(8),POINTER :: rv(:), tq(:) ! maximum torque dataset
INTEGER ntq, ngr           ! number of torque data, number of gear
REAL(8),POINTER :: dw(:)   ! rotational weight for each gear
REAL(8),POINTER :: tm(:)   ! torque margin
REAL(8) muR, muA, tcl      ! rolling&air drag coefficient, circumference
REAL(8) mass               ! test weight(kg)
INTEGER cnt                ! time length of prediction
! [OUT]-----
INTEGER spiout              ! output shift position
REAL(8) minerr              ! cumulative error with spiout
! [LOCAL]-----
INTEGER s, sw, x
REAL(8) verr, vold, vv, maxt, enginev, enginet, tm1

spiout = shiftp            ! initialize return value
minerr = 0.0_8             ! initialize minimum error
DO s = shiftp + 1, ngr     ! examine the possibility of shift up
!   ----- CANNOT UP SHIFT OVER SKIPLIMIT -----
!   IF ( ( s == shiftp + SKIPLIMIT ) .AND. ( reacc == 0 ) ) EXIT
!   ----- ENGINE RUNNING CONDITION -----
!   call engstat( nex, nel, enginev, enginet, s, ngr, gr, fgr, tcl,mass, dw, muR, muA,      &
!             vdest(i), vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw)
!   ----- ENGINE SPEED IS OUT OF RANGE, START MODE -----
!   IF ( ( sw == 2 ) .OR. ( sw == 3 ) .OR. ( sw == 5 ) ) CYCLE
!   ----- IN RANGE -----
!   IF ( enginet > 0.0_8 ) THEN
!       tm1 = maxt / enginet          ! calculate torque margin
!   ELSE
!       tm1 = 100.0_8                ! avoid /0 error, influence of minus
!   END IF

```



```

IF ( ( tm1 >= tm(s) ) .AND. ( sw == 0 ) ) THEN      ! try shift up
  vold = vpast
  verr = 0.0_8
  DO x=0, cnt                                     ! check drivability for cnt seconds
    call engstat ( nex, nel, enginev, enginet, s, ngr, gr, fgr, tcl,mass, dw, muR, muA, &
                  vdest(i+x), vold, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
    IF( ( sw == 2 ) .OR. ( sw == 3 ) )EXIT        ! OUT OF RANGE
    verr = verr + ( vdest(i+x) + vdest(i+x-1) ) /2.0_8 - ( vv + vold ) / 2.0_8
    vold = vv
  END DO

  IF( ( sw == 2 ) .OR. ( sw == 3 ) )THEN
    CYCLE                                       ! try shift up, again
  ELSE IF ( spiout == shiftp ) THEN
    spiout = s                                ! update output gear position
    minerr = verr
  ELSE IF(minerr >= verr)THEN
    spiout = s
    minerr = verr
  END IF
END IF
END DO

RETURN
END SUBROUTINE sftuptm

```

```

! *****
! *
! * SUBROUTINE sftup : calculation of shift up
! * This subroutine is executed when
! * 1.Engine over-run without shift-change
! * 2.There is no choise of shiftup with torque margin
! * If reacc=0, returns the lowest gear that can drive for 3second with same position.
! * If reacc=1, returns the highest gear that can keep error minimum for 3seconds.
! *

```

```

! * If it is impossible to keep 3 seconds, decreases THOLD to 1 second step by step. *
! *                                                                                       *
! *****
SUBROUTINE sftup ( reacc, i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex, gr, fgr, egr, &
                 efgr, ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, spiout, minerr )

  USE DEF2
  IMPLICIT NONE

  INTERFACE
    SUBROUTINE engstat ( nex, nel, enginev, enginev, shiftp, ngr, gr, fgr, tcl, mass, dw, muR, &
                      muA, vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
      REAL(8) enginev, enginev, fgr, tcl, mass, muR, muA, vdest
      REAL(8) efgr, maxt, nidle, nes, nex, nec, vpast, vv
      REAL(8), POINTER :: dw(:), rv(:), tq(:), gr(:), egr(:), nel(:)
      INTEGER shiftp, sw, ntq, ngr
    END SUBROUTINE engstat
  END INTERFACE

  ! ===== VARIABLE DEFINITION =====
  ! [IN]-----
  INTEGER reacc                ! reacceleration mode
  INTEGER i                    ! current time step
  REAL(8), POINTER :: vdest(:) ! test cycle
  INTEGER len                  ! number of data
  REAL(8) vpast                ! vehicle speed (time i-1)
  INTEGER shiftp               ! last gear position (time i-1)
  REAL(8) nes, nec, nidle, nex ! clutch meet&release, idle&maximum engine speed
  REAL(8),POINTER :: nel(:)    ! minimum(each gear) & maximum engine speed
  REAL(8),POINTER :: gr(:), egr(:) ! gear ratio, transmit efficiency
  REAL(8) fgr, efgr            ! gear ratio, transmit efficiency (final)
  REAL(8),POINTER :: rv(:), tq(:) ! maximum torque dataset
  INTEGER ntq, ngr              ! number of torque data, number of gear
  REAL(8),POINTER :: dw(:)     ! rotational weight for each gear
  REAL(8) muR, muA, tcl        ! rolling&air drag coefficient, circumference
  REAL(8) mass                  ! test weight(kg)
  ! [OUT]-----
  INTEGER spiout                ! output shift position

```

```

! [LOCAL]-----
INTEGER x, y, z,sw
REAL(8) verr, minerr, vold, vv, maxt, enginev, enginet

spiout = shiftp          ! initial return value = current position
DO x = THOLD, 1, -1      ! search optimal gear from THOLD to 1sec.
  minerr = 0.0_8         ! initialize minimum trace error
  DO y = shiftp + 1, ngr ! from shiftp+1 to top gear
    vold = vpast
    verr = 0.0_8
    DO z = 0, x-1        ! calculate error for x seconds
      call engstat( nex, nel, enginev, enginet, y, ngr, gr, fgr, tcl, mass, dw, muR, muA, &
        vdest(i+z), vold, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw)

      IF( ( sw == 2 ) .OR. ( sw == 3 ) ) EXIT ! engine speed is out of range, exit loop

      verr = verr + ( vdest(i+z) + vdest(i+z-1) ) /2.0_8 - ( vv + vold ) / 2.0_8
      vold = vv
    END DO

    IF( ( sw == 2 ) .OR. ( sw == 3 ) )THEN
      CYCLE ! try next gear
    ELSE IF ( ( minerr > verr ).OR.(spiout == shiftp) ) THEN
!      --- executed only when reacc=1(reacceleration=ON) ---
      spiout = y ! update return value
      minerr = verr
    END IF

  END DO

  IF ( spiout /= shiftp ) EXIT
END DO

RETURN
END SUBROUTINE sftup

```

```

! *****
! *
! * SUBROUTINE sftdwn : calculation of shift down
! * This routine is executed when required engine torque is larger than maximum.
! *
! *****
SUBROUTINE sftdwn ( i, vdest, len, vpast, shiftp, nidle, nel, nes, nec, nex, gr, fgr, egr, efgr, &
                  ngr, rv, tq, ntq, tcl, dw, muR, muA, mass, cnt, spiout, minerr )

  USE DEF2
  IMPLICIT NONE

  INTERFACE
    SUBROUTINE engstat ( nex, nel, enginev, enginet, shiftp, ngr, gr, fgr, tcl, mass, dw, muR, &
                      muA, vdest, vpast, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )
      REAL(8) enginev, enginet, fgr, tcl, mass, muR, muA, vdest
      REAL(8) efgr, maxt, nidle, nes, nex, nec, vpast, vv
      REAL(8), POINTER :: dw(:), rv(:), tq(:), gr(:), egr(:), nel(:)
      INTEGER shiftp, sw, ntq, ngr
    END SUBROUTINE engstat
  END INTERFACE

! ===== VARIABLE DEFINITION =====
! [IN]-----
INTEGER i                ! current time step
REAL(8), POINTER :: vdest(:) ! test cycle
INTEGER len              ! number of data
REAL(8) vpast            ! vehicle speed (time i-1)
INTEGER shiftp           ! last gear position (time i-1)
REAL(8) nes, nec, nidle, nex ! clutch meet,release,idle,maximum speed
REAL(8),POINTER :: nel(:) ! minimum(each gear) engine speed
REAL(8),POINTER :: gr(:), egr(:) ! gear ratio, transmit efficiency
REAL(8) fgr, efgr        ! gear ratio, transmit efficiency (final)
REAL(8),POINTER :: rv(:), tq(:) ! maximum torque dataset
INTEGER ntq, ngr         ! number of torque data, number of gear
REAL(8),POINTER :: dw(:) ! rotational weight for each gear
REAL(8) muR, muA, tcl    ! rolling&air drag coefficient, circumference
REAL(8) mass             ! test weight(kg)

```

```

INTEGER cnt                                ! time length of prediction
! [OUT]-----
INTEGER spiout                             ! output shift position
! [INOUT]-----
REAL(8) minerr                             ! cumulative error with spiout
! [LOCAL]-----
INTEGER s, sw, x,z
REAL(8) verr, vold, vv, maxt, enginev, enginev

spiout = shiftp                            ! initialize output value
DO x = THOLD, 1, -1                         ! search optimal gear from THOLD to 1sec.
  minerr = 0.0_8                            ! initialize minimum trace error
  DO s = GRB, shiftp - 1
    vold = vpast
    verr = 0.0_8
    DO z=0, cnt
      CALL engstat ( nex, nel, enginev, enginev, s, ngr, gr, fgr, tcl, mass, dw, muR, muA, &
                    vdest(i+z), vold, vv, egr, efgr, maxt, nidle, nes, nec, rv, tq, ntq, sw )

      IF( ( sw == 2 ) .OR. ( sw == 3 ) )EXIT
      verr = verr + ( vdest(i+z) + vdest(i+z-1) ) / 2.0_8 - ( vv + vold ) / 2.0_8
      vold = vv
    END DO

    IF( ( sw == 2 ) .OR. ( sw == 3 ) )THEN
      CYCLE
    ELSE IF (( spiout == shiftp ) .or. (minerr >= verr))THEN
      spiout = s
      minerr = verr
    END IF
  END DO
  IF ( spiout /= shiftp ) EXIT
END DO

RETURN
END SUBROUTINE sftdwn

```