```
! ************************************************************
! MAIN PROGRAM
! ************************************************************
PROGRAM FCAT_TRIAS

    use testcycle
    use hvtransfc
    use vehicleparams
    use msubfc, only : calcfcAT, calcave
    use dataio
    use moddrv
    implicit none

    type ( sspec ) :: spec
    type ( seng ) :: eng
    type ( stm ) :: tm
    type ( stc ) :: tc
    type ( sctr ) :: ctrl
    type ( scycle ) :: vpat, vpat2

    type ( vehout ), pointer :: vhx(:), vhx2(:)
    type ( vehout ) :: vhx0

    real(8) :: avv, avfc, avv2, avfc2, avvt, avfct
    real(8) :: ece
    integer :: terr1, terr2, terr3
    integer :: i, iret, iret2
    integer :: modesw
    character (len=1024) :: specf='', outf=''
    character (len=1024) :: strversion = 'FCAT_Ver1.00'
    character (len=1024) :: com_version

    print *, "【FCAT_TRIAS Version", trim(strversion),"】"

!   <<< 初期設定 >>>
    call GetArgument ( specf, outf )
    call read_input_data ( specf, spec, tm, eng, tc, ctrl )

    vpat%ndat = nurban
    vpat%v => vurban
    vpat%g => gurban

    vpat2%ndat = nhighway
    vpat2%v => vhighway
    vpat2%g => ghighway

    allocate ( vhx(0:vpat%ndat), vhx2(0:vpat2%ndat) )

    call setparametersAT ( spec, tm, ctrl )
    call showinputdata ( spec, eng, tm )

!   <<<<<<<<<<<<<<<<<<<<< 都市内モード >>>>>>>>>>>>>>>>>>>>>>>>>>>>
    modesw = 1
    iret = 0
    vhx0%v = vpat%v(1)
    vhx0%s = tm%grs
    vhx0%lu = 0
    vhx0%tstop = 0
    vhx0%opn = 0.0_8
    vhx0%c = 1
    vhx0%tt = 0.0_8

    call gearpos ( modesw, 0, tm, ctrl, vpat%v(1), vhx0, vhx(0)%s, vhx(0)%lu, vhx(0)%tstop )
    call driveAT ( spec, eng, tm, tc, ctrl, vhx0%v, vpat%v(1), vpat%g(1), vhx(0), iret, 1 )

    vhx(0)%tstop = 0

    write(*,'(a)'),'Start urban testcycle ... '

    do i = 1, vpat%ndat

        write(*,'(a,i0,a,$)'),' T = ',i,char(13)
        call drivecycle ( modesw, i, spec, eng, tm, tc, ctrl, vhx, vpat, iret )

        if ( iret /= 0 ) then
            vpat%ndat = i-1
            exit
        end if

    end do

    if ( iret == 0 ) then
        write(*,*)
        write(*,'(a)'),'... Finished. '
    else
        write(*,'(a)'),'... Calculation stopped. '
    end if

!   <<< 燃料消費量計算 >>>
    call calcfcAT ( vhx, vpat%ndat, eng )
    call calcave ( vhx, 1, vpat%ndat, avv, avfc, terr1 )
    call calcave ( vhx, 645, 766, avvt, avfct, terr3 )


!   <<<<<<<<<<<<<<<<<<<<< 都市間モード >>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
    modesw = 2
    iret2 = 0
    vhx0%v = vpat2%v(1)
    vhx0%s = tm%grs
    vhx0%lu = 0
    vhx0%tstop = 0
    vhx0%opn = 0.0_8
    vhx0%c = 1

    call gearpos ( modesw, 0, tm, ctrl, vpat2%v(1), vhx0, vhx2(0)%s, vhx2(0)%lu, vhx2(0)%tstop )
    call driveAT ( spec, eng, tm, tc, ctrl, vhx0%v, vpat2%v(1), vpat2%g(1), vhx2(0), iret2, 1 )
    vhx2(0)%tstop = 0
    vhx2(0)%tt = 0.0_8

    write(*,'(a)'),'Start highway testcycle ... '

    do i = 1, vpat2%ndat

        write(*,'(a,i0,a,$)'),' T = ',i,char(13)
        call drivecycle ( modesw, i, spec, eng, tm, tc, ctrl, vhx2, vpat2, iret )

        if ( iret2 /= 0 ) then
            vpat2%ndat = i-1
            exit
        end if

    end do

    if ( iret2 == 0 ) then
        write(*,*)
        write(*,'(a)'),'... Finished. '
    else
        write(*,'(a)'),'... Calculation stopped. '
    end if

!   <<< 燃料消費量計算 >>>
    call calcfcAT ( vhx2, vpat2%ndat, eng )
    call calcave ( vhx2, 1, vpat2%ndat, avv2, avfc2, terr2 )

    print*
    print*, "------------------------------------------------------------"
!    ----- 燃費値 ---------
    ece = 1.0_8 / ( ( 1.0_8 - spec%ric ) / avfc + spec%ric / avfc2 )

    if ( iret == 0 ) then
        print '( 2A,F8.4,A,F0.1,A,i0 )', "【URBAN】,    ","FC(km/l) : ",avfc, ", Ave.Speed(km/h) : ",avv, ", 追従不能時間(S) : ", terr1
    else
        print '( a )', "【URBAN】エラーにより途中終了しました. "
    end if

    if ( iret2 == 0 ) then
        print '( 2A,F8.4,A,F0.1,A,i0 )', "【HIGHWAY】,  ","FC(km/l) : ",avfc2, ", Ave.Speed(km/h) : ",avv2, ", 追従不能時間(S) : ", terr2
    else
        print '( a )', "【HIGHWAY】エラーにより途中終了しました. "
    end if

    if ( iret == 0 .and. iret2 == 0 ) then
        print '( 2A,F8.4,A,F0.1,A,i0 )', "【MID-TOWN】, ","FC(km/l) : ",avfct, ", Ave.Speed(km/h) : ",avvt, ", 追従不能時間(S) : ", terr3
        print *
        print '( 2A,F8.4 )', "【AVERAGE】,  ","FC(km/l) : ",ece
    else
        print '( A )', "【MID-TOWN】エラーにより途中終了しました. "
        print *
        print '( A )', "【AVERAGE】エラーにより途中終了しました. "
   end if


!   <<< ファイル出力 >>>
    com_version = "【FCAT08 Version - " // trim(strversion) // "】"
    call outaveragefcAT ( 21, outf, specf, spec, avfc, avv, terr1, avfc2, avv2, terr2, ece, avfct, avvt, terr3, iret, iret2, com_version )
    call writeresultAT2 ( 21, outf, vpat, vhx, "URBAN" )
    call writeresultAT2 ( 21, outf, vpat2, vhx2, "HIGHWAY" )

    deallocate ( vhx, vhx2 )

end program FCAT_TRIAS
```

```
module moddrv

    interface

        real(8) function drvfrc ( dw, wt, mua, mur, v0, v1, grd )
            real(8) :: dw, wt, mua, mur, v0, v1, grd
        end function

        subroutine driveAT( spec, eng, tm, tc, ctrl, v0, v1, grd, vhx, iret, precon )
            use hvtransfc
            type ( sspec ) :: spec
            type ( seng ) :: eng
            type ( stm ) :: tm
            type ( stc ) :: tc
            type ( vehout ) :: vhx
            type ( sctr ) :: ctrl
            real(8) :: v0, v1, grd
            integer :: iret, precon
        end subroutine

        subroutine convergence ( spec, eng, tm, tc, v0, v1, grd, vhx, iret, precon )
            use hvtransfc
            type (sspec) :: spec
            type (seng) :: eng
            type (stm) :: tm
            type (stc) :: tc
            type ( vehout ) :: vhx
            real(8) :: v0, v1, grd
            integer :: iret, precon
        end subroutine

        real(8) function plosstq( ptq, np )
            use hvtransfc
            type (scrv) :: ptq
            real(8) :: np
        end function

        subroutine thrfrommap2( eng, ne, te, taout )
            use hvtransfc
            type (seng) :: eng
            real(8) :: ne, te, taout
        end subroutine

        subroutine egmapTE( gegmp, ne, opn, teout )
            use hvtransfc
            type ( sgridengmap ) :: gegmp
            real(8) :: ne, opn, teout
        end subroutine

        subroutine lucheck( bl, opn, v, iret )
            use hvtransfc
            type(scrv) :: bl
            real(8) :: opn, v
            integer :: iret
        end subroutine

        subroutine gpcheck ( bl, opn, v, iret, luout )
            use hvtransfc
            type(scrv) :: bl
            real(8) :: opn, v
            integer :: iret, luout
        end subroutine

        subroutine gearpos ( modesw, ii, tm, ctrl, vp, vhx, sout, lu, tstop )
            use hvtransfc
            type ( stm ) :: tm
            type (vehout) :: vhx
            type ( sctr ) :: ctrl
            integer :: sout, lu, tstop, ii, modesw
            real(8) :: vp
        end subroutine

        subroutine drivecycle ( modesw, i, spec, eng, tm, tc, ctrl, vhx, vpat, iret )
            use hvtransfc
            type ( sspec ) :: spec
            type ( seng ) :: eng
            type ( stm ) :: tm
            type ( stc ) :: tc
            type ( sctr ) :: ctrl
            type ( scycle ) :: vpat
            type ( vehout ), pointer :: vhx(:)
            integer :: i, iret, modesw
        end subroutine

    end interface

end module



! ***********************************************************
! FUNCTION drvfrc
!  dw       : 回転部分相当重量 (kg)
!  wt       : 試験時重量 (kg)
!  mua, mur : 空気抵抗係数 (N/km/h2), 転がり抵抗係数 (N/kg)
```

```
!  v0, v1   : 初速度, 目標速度 (km/h)
!  grd  : 勾配 (%)
! ************************************************************
real(8) function drvfrc ( dw, wt, mua, mur, v0, v1, grd )
    implicit none

    real(8) :: dw, wt, mua, mur, v0, v1, grd
    real(8) :: rr, mass, acc, fgrade

    rr =  mur * wt + mua * v1**2
    mass = wt + dw
    acc = ( v1 - v0 ) / 3.6_8
    fgrade = wt * dsin( datan( grd / 100.0_8 ) ) * 9.8_8

    drvfrc = rr + mass * acc  + fgrade

end function drvfrc




!************************************************************
! SUBROUTINE CONVERGENCE
!  spec     : 車両諸元データ
!  eng      : エンジン諸元データ
!  tm       : 変速機諸元データ
!  tc       : トルコン諸元データ(トルク比,容量係数,ポンプ損失トルク)
!  v0, v1, grd : 現在の速度と目標速度,勾配
!  VHX      : 運転状態出力
!  IRET     : 計算結果戻り値 (0 : 正常, 0以外 : 不具合あり)
!  PRECON   : 初期状態計算用オプション(1で初期値計算,0で通常計算)
!************************************************************
subroutine convergence ( spec, eng, tm, tc, v0, v1, grd, vhx, iret, precon )
    use hvtransfc
    use moddrv
    use msubfc
    implicit none

    type (sspec) :: spec
    type (seng) :: eng
    type (stm) :: tm
    type (stc) :: tc
    type (vehout) :: vhx
    real(8) :: v0, v1, grd
    integer :: iret, precon
    integer :: flg, flg2
    real(8) :: ttout, dtt
    real(8) :: f0, ds, dv, tqdif, maxt, dne

    vhx%v = v1
    dv = 1.0_8
    flg2 = 0

    do while ( flg2 == 0 )

        if ( precon == 1 ) v0 = vhx%v

        vhx%nt = vhx%v * tm%g(vhx%s)%gr * spec%fgr / (0.3768_8*spec%rt)
        f0 = drvfrc( tm%g(vhx%s)%dw, spec%wt, spec%mua, spec%mur, v0, vhx%v, grd )

        if ( f0 >= 0.0_8 ) then
            vhx%tt = f0 * spec%rt / ( tm%g( vhx%s )%gr * spec%fgr * tm%g( vhx%s )%egr * spec%efgr )
        else
            vhx%tt = f0 * spec%rt * tm%g( vhx%s )%egr * spec%efgr / ( tm%g( vhx%s )%gr * spec%fgr )
            dv = dv / 2.0_8
            vhx%v = vhx%v + dv
            cycle
        end if

!       ------------------------------------------------------------
        if ( vhx%lu == 2 ) then
            vhx%ne = vhx%nt
            vhx%sr = 1.0_8
            vhx%tp = vhx%tt
            vhx%tr = 1.0_8
            vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )
            vhx%te = vhx%tp + vhx%top

!       ------------------------------------------------------------
        else
            vhx%ne = vhx%nt
            if ( vhx%ne < eng%nidle ) vhx%ne = eng%nidle

            ds = 5.0_8
            flg = 0

            do while ( flg == 0 )

                vhx%sr = vhx%nt / vhx%ne
                call HERMITEIP( tc%trd%x, tc%trd%y, tc%trd%n, vhx%sr, vhx%tr )
                call HERMITEIP( tc%cfd%x, tc%cfd%y, tc%cfd%n, vhx%sr, vhx%cf )

                ttout = vhx%nt**2 * ( vhx%tr * vhx%cf / vhx%sr**2 )
                dtt = ttout - vhx%tt

                if ( ( dabs(dtt) < 1.0E-10 ) .and. ( dabs(dtt) >= 0.0_8 ) ) then
```

```
                flg = 1
            else if ( dtt < 0.0_8 ) then
                vhx%ne = vhx%ne + ds
            else
                ds = ds / 2.0_8
                vhx%ne = vhx%ne - ds
            end if

!           ---- 収束しない場合 ----------------
            if ( ds==0.0_8 ) then
                write (0,*)
                write (0,'(a)') "-----------------------------------------------"
                write (0,'(a)') "(subroutine convergence) トルコン運転状態が収束しません・・・・."
                write (0,'(2(a,f0.2),a,i0)') "Vtarget=",v1,"(km/h), Ne=",vhx%ne,"(rpm), Gear=",vhx%s
                write (0,'(2(a,f0.3),a,f0.4)') "速度比=", vhx%sr, ", トルク比=", vhx%tr, ", 容量=", vhx%cf
                write (0,'(a)') "計算を中止します....."
                iret = -2
                return
            end if

        end do

        vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )
        vhx%tp = vhx%tt / vhx%tr
        vhx%te = vhx%tp + vhx%top

    end if

!   ------------------------------------------------
    call egmapTE( eng%gegmp, vhx%ne, 100.0_8, maxt )
    tqdif = maxt - vhx%te
    dne = dabs( vhx%ne - eng%nex)

    if ( ( ( dabs(tqdif) < 1.0E-10 ) .and. ( dabs(tqdif) >= 0.0_8 ) ) .or. dne < 1.0e-10) then
        flg2 = 1
    else if ( tqdif < 0.0_8 .or. vhx%ne>eng%nex ) then
        vhx%v = vhx%v - dv
    else
        dv = dv / 2.0_8
        vhx%v = vhx%v + dv
    end if

end do

call thrfrommap2 ( eng, vhx%ne, vhx%te, vhx%opn )

end subroutine




! ************************************************************************
! FUNCTION plosstq
!  tq       : 対象ギヤ段のロストルク特性 (scrv)
!  np       : 入力軸回転数, rpm
!  plosstq  : 区分三次エルミート補間によるオイルポンプ駆動損失推定値, (Nm)
! ************************************************************************
real(8) function plosstq( tq, np )
    use hvtransfc
    use msubfc
    implicit none

    type (scrv) :: tq
    real(8) :: np

    call HERMITEIP( tq%x, tq%y, tq%n, np, plosstq )

end function plosstq




!*********************************************************
! SUBROUTINE egmapTE
!  gegmp    : エンジントルクマップ
!  ne       : エンジン回転数(rpm, in)
!  opn      : 開度(0-1, in)
!  teout    : エンジントルク出力
!*********************************************************
subroutine egmapTE( gegmp, ne, opn, teout )
    use hvtransfc
    use msubfc
    implicit none

    type ( sgridengmap ) :: gegmp
    real(8) :: ne, opn, teout
    real(8), allocatable :: x(:), y(:)
    real(8), allocatable :: thr(:), tq(:)
    integer :: i, j

!   ------------------------------------------------------
    allocate ( thr(1:gegmp%thrn), tq(1:gegmp%thrn) )

    do i = 1, gegmp%thrn
```

```
        thr(i) = gegmp%thr(i)
        allocate( x(1:gegmp%n(i)), y(1:gegmp%n(i)) )

        do j = 1, gegmp%n(i)
            x(j) = gegmp%r(i,j)
            y(j) = gegmp%t(i,j)
        end do

        call HERMITEIP( x, y, gegmp%n(i), ne, tq(i) )
        deallocate ( x, y )

    end do

    call HERMITEIP( thr, tq, gegmp%thrn, opn, teout )

    deallocate ( thr, tq )

end subroutine




!************************************************
! SUBROUTINE DRIVEAT
! SPEC      : 車両諸元
! ENG       : エンジン諸元
! TM        : 変速機諸元 (ギヤ比)
! TC        : トルコン諸元
! CTRL      : シフト制御データ
! VV0, VV1  : 初速, 目標速度
! grd       : 勾配
! VHX       : 車両状態のシミュレーション結果
! IRET      : 演算結果指標 (０：正常, 0以外：不具合あり)
! PRECON    : 初期状態計算スイッチ(1:初期状態,0:通常走行計算)
!************************************************
subroutine driveAT( spec, eng, tm, tc, ctrl, vv0, vv1, grd, vhx, iret, precon )
    use hvtransfc
    use moddrv
    use msubfc
    implicit none

    type (sspec) :: spec
    type (seng) :: eng
    type (stm) :: tm
    type (stc) :: tc
    type (vehout) :: vhx
    type (sctr) :: ctrl
    real(8) :: vv0, vv1, grd
    integer :: iret
    integer :: precon
    real(8) f0, ftq, ds
    real(8) fe1, fedif
    integer :: flg
    real(8) :: tpx, ttx

!   ------------------------------------------------------------
    if( vv1 == 0.0_8 ) then

        vhx%lu = 0
        vhx%v = 0
        vhx%nt = 0.0_8
        vhx%opn = 0.0_8
        vhx%npera = 0.0_8
        vhx%ne = eng%nidle

!       <<< アイドルストップ >>>
        if ( ctrl%swistop == 1 .and. vhx%s == -1 ) then
            vhx%s = tm%grs
            vhx%ne = 0
            vhx%te = 0
            vhx%maxt = 0.0_8
            vhx%nne = 0.0_8
            vhx%nte = 0.0_8
            vhx%tr = 0
            vhx%cf = 0
            vhx%sr = 0.0_8
            vhx%tp = 0.0_8
            vhx%tt = 0.0_8
            vhx%top = 0.0_8
            return
        end if

!       <<< ニュートラルアイドル >>>
        if ( ctrl%swnidl == 1 .and. vhx%s == 0 ) then
            vhx%s = tm%grs
            vhx%sr = ctrl%sr_nidl
            vhx%tr = 0.0_8

            call HERMITEIP( ctrl%cf_nidl%x, ctrl%cf_nidl%y, ctrl%cf_nidl%n, vhx%sr, vhx%cf )
            vhx%tp = vhx%cf * ( vhx%ne**2 )

!       <<< Dレンジアイドル >>>
        else
            vhx%s = tm%grs
            vhx%sr = 0.0_8
            call HERMITEIP( tc%trd%x, tc%trd%y, tc%trd%n, vhx%sr, vhx%tr )
```

```
            call HERMITEIP( tc%cfd%x, tc%cfd%y, tc%cfd%n, vhx%sr, vhx%cf )
            vhx%tp = vhx%cf * ( vhx%ne**2 )
        end if

        vhx%c = 1
        vhx%tt = vhx%tp * vhx%tr
        vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )
        vhx%te = vhx%tp + vhx%top
        call egmapTE( eng%gegmp, vhx%ne, 100.0_8, vhx%maxt )

        call thrfrommap2 ( eng, vhx%ne, vhx%te, vhx%opn )

        vhx%nte = 100.0_8 * ( vhx%te / vhx%maxt )
        vhx%nne = 0.0_8
        return
    end if

!   ------------------------------------------------------------
!   <<< タービン回転数 & 駆動力 >>>
    vhx%nt = vv1 * tm%g( vhx%s )%gr * spec%fgr / ( 0.3768_8 * spec%rt )
    f0 = drvfrc ( tm%g(vhx%s)%dw, spec%wt, spec%mua, spec%mur, vv0, vv1, grd )
    vhx%v = vv1

!   ------------------------------------------------------------
!    L/U ON時
    if ( vhx%lu == 2 ) then
        vhx%c = 1
        if ( f0 >= 0.0_8 ) then
            vhx%tt = f0 * spec%rt / ( tm%g( vhx%s )%gr * spec%fgr * tm%g( vhx%s )%egr * spec%efgr )
        else
            vhx%tt = f0 * spec%rt * tm%g( vhx%s )%egr * spec%efgr / ( tm%g( vhx%s )%gr * spec%fgr )
        end if

        vhx%ne = vhx%nt

        if ( vhx%ne <= eng%nidle ) then
            vhx%lu = 0
            vhx%c = 0
        else
            vhx%sr = 1.0_8
            vhx%tr = 1.0_8
            vhx%tp = vhx%tt
            vhx%top = plosstq( tc%ptq%g( vhx%s ), vhx%ne )
            vhx%te = vhx%tp + vhx%top

            call egmapTE( eng%gegmp, vhx%ne, 0.0_8, ftq )
            if( vhx%te <= ftq ) then
                vhx%te = ftq
                vhx%tp = vhx%te - vhx%top
                vhx%tt = vhx%tp
            end if
        end if
    end if

!   ------------------------------------------------------------
!    L/U OFF時
    if ( vhx%lu == 0 ) then

        if ( vhx%nt < eng%nidle ) then
            vhx%ne = eng%nidle
        else
            vhx%ne = vhx%nt
        end if

        if ( f0 >= 0.0_8 .or. vhx%nt < eng%nidle ) then
            ds = 5.0_8
        else
            ds = -5.0_8
        end if

        flg = 0
        do while ( flg == 0 )

            f0 = drvfrc ( tm%g(vhx%s)%dw, spec%wt, spec%mua, spec%mur, vv0, vv1, grd )
            vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )

            if ( vhx%ne >= vhx%nt ) then
                vhx%sr = vhx%nt / vhx%ne
            else
                vhx%sr = vhx%ne / vhx%nt
            end if

!           ------------------------------------------------------------
            if ( f0 >= 0.0_8 ) then
                vhx%c = 1
                call HERMITEIP( tc%trd%x, tc%trd%y, tc%trd%n, vhx%sr, vhx%tr )
                call HERMITEIP( tc%cfd%x, tc%cfd%y, tc%cfd%n, vhx%sr, vhx%cf )
                vhx%tt = f0 * spec%rt / ( tm%g(vhx%s)%gr * spec%fgr * tm%g(vhx%s)%egr * spec%efgr )

                if( vhx%ne <= eng%nidle ) then
                    vhx%tp = vhx%tt / vhx%tr
                    tpx = vhx%cf * vhx%ne * vhx%ne
                    ttx = vhx%tr * tpx

                    if( vhx%tt <= ttx ) then
                        vhx%tt = ttx
```

```
                    vhx%tp = tpx
                    flg = 1
                end if

                vhx%te = vhx%tp + vhx%top
                fe1 = vhx%nt**2 * ( vhx%tr * vhx%cf / vhx%sr**2 )
                fedif = fe1 - vhx%tt

            else
                fe1 = vhx%nt**2 * ( vhx%tr * vhx%cf / vhx%sr**2 )
                fedif = fe1 - vhx%tt
                vhx%tp = vhx%tt / vhx%tr
                vhx%te = vhx%tp + vhx%top
            end if

            if( flg == 0 ) then
                if ( dabs(fedif) < 1.0E-10 .and. dabs(fedif) >= 0.0_8 ) then
                    flg = 1
                else if ( fedif < 0.0_8 ) then
                    vhx%ne = vhx%ne + ds
                else
                    ds = ds / 2.0_8
                    vhx%ne = vhx%ne - ds
                end if
            end if

!           ------------------------------------------------------
            else
                vhx%tt = f0 * spec%rt * tm%g(vhx%s)%egr * spec%efgr / ( tm%g(vhx%s)%gr * spec%fgr )
                call HERMITEIP( tc%cfb%x, tc%cfb%y, tc%cfb%n, vhx%sr, vhx%cf )
                call HERMITEIP( tc%trb%x, tc%trb%y, tc%trb%n, vhx%sr, vhx%tr )

                if ( vhx%nt <= eng%nidle ) then
                    vhx%c = 1
                    call HERMITEIP( tc%cfd%x, tc%cfd%y, tc%cfd%n, vhx%sr, vhx%cf )
                    call HERMITEIP( tc%trd%x, tc%trd%y, tc%trd%n, vhx%sr, vhx%tr )
                    vhx%ne = eng%nidle
                    vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )
                    vhx%tp = vhx%cf * vhx%ne * vhx%ne
                    vhx%tt = vhx%tr * vhx%tp
                    vhx%te = vhx%tp + vhx%top
                    flg = 1

                else if( vhx%ne <= eng%nidle .or. vhx%c == 0 ) then
                    vhx%tp = 0.0_8
                    vhx%ne = eng%nidle
                    vhx%top = plosstq( tc%ptq%g(vhx%s), vhx%ne )
                    vhx%te = vhx%tp + vhx%top
                    flg = 1
                    vhx%c = 0
                    vhx%lu = 0

                else
                    vhx%c = 1
                    vhx%tp = vhx%tt * vhx%tr
                    vhx%te = vhx%tp + vhx%top

                    call egmapTE( eng%gegmp, vhx%ne, 0.0_8, ftq )
                    if( vhx%te < ftq ) then
                        vhx%te = ftq
                        vhx%tp = vhx%te - vhx%top
                        vhx%tt = vhx%tp / vhx%tr
                    end if

                    fe1 = vhx%cf * ( vhx%nt **2 )
                    fedif = fe1 + vhx%tt
                end if

                if( flg == 0 ) then
                    if ( dabs(fedif) < 1.0E-10 .and. dabs(fedif) >= 0.0_8 ) then
                        flg = 1
                    else if ( fedif < 0.0_8 ) then
                        vhx%ne = vhx%ne + ds
                    else
                        ds = ds / 2.0_8
                        vhx%ne = vhx%ne - ds
                    end if

                end if

            end if

!           <<< 収束しない場合は途中停止 >>>
            if ( ds==0.0_8 ) then
                write (0,*)
                write (0,'(a)') "-------------------------------------------------"
                write (0,'(a)') "(subroutine driveAT) トルコンの速度比が収束しません・・・・."
                write (0,'(2(a,f0.2),a,i0)') "Vtarget=",vv1,"(km/h), Ne=",vhx%ne,"(rpm), Gear=",vhx%s
                write (0,'(2(a,f0.3),a,f0.4)') "速度比=",vhx%sr,", トルク比=",vhx%tr,", 容量=",vhx%cf
                write (0,'(a)') "計算を中止します....."
                iret = -1
                return
            end if

        end do
```

```
    end if

    call thrfrommap2( eng, vhx%ne, vhx%te, vhx%opn )

!   ------------------------------------------------
!   追従できない場合, 全負荷走行として収束計算
    if( vhx%opn >= 100.0_8 .or. vhx%ne>=eng%nex ) then
        call convergence ( spec, eng, tm, tc, vv0, vv1, grd, vhx, iret, precon )
        if ( iret /= 0 ) return
    end if

    call egmapTE( eng%gegmp, vhx%ne, 100.0_8, vhx%maxt )
    vhx%nte = 100.0_8 * ( vhx%te / vhx%maxt )
    vhx%nne = 100.0_8 * ( vhx%ne-eng%nidle ) / ( eng%nrate-eng%nidle )
    vhx%npera = vhx%nt / tm%g( vhx%s )%gr

end subroutine




! ************************************************************
!  LUCHECK
!  [input]
!  bl       : ロックアップ線 (scrv)
!  opn      : 入力開度 %
!  v        : 入力速度 km/h
!  [output]
!  iret     : 判定結果 (戻り値)
! ************************************************************
subroutine lucheck( bl, opn, v, iret )
    use hvtransfc
    implicit none

    integer, parameter :: EQUAL = 0                                 ! 速度線に一致
    integer, parameter :: ABOVE = 1                                 ! 速度線より大きい
    integer, parameter :: BELOW = -1                                ! 速度線より小さい
    integer, parameter :: UNDEF = -2                                ! 速度線の定義なし
    type( scrv ) :: bl                                              ! 変速・L/U線
    integer :: i, iret
    real(8) :: opn, v, vout, a

!   <<< L/U線の定義なし -2 >>>
    if ( bl%n == 0 ) then
        iret = UNDEF
        return
    end if

!   <<< 変速点判定 (直線補間) >>>
    if ( opn < 0.0_8 ) then
        opn = 0.0_8
    else if ( opn > 100.0_8) then
        opn = 100.0_8
    end if

    if ( opn < bl%x(1) .or. opn > bl%x(bl%n) ) then
        iret = UNDEF
        return
    end if

    vout = -1.0_8
    do i = 2, bl%n
        if ( opn <= bl%x(i) ) then
            a = (bl%y(i) - bl%y(i-1)) / (bl%x(i) - bl%x(i-1))
            vout = bl%y(i-1) + a * ( opn - bl%x(i-1) )
            exit
        end if
    end do

!   <<< L/U線判定 >>>
    if ( v > vout ) then
        iret = ABOVE
    else if (v < vout) then
        iret = BELOW
    else
        iret = EQUAL
    end if

end subroutine




! ************************************************************
!  GPCHECK
!  [input]
!  bl       : 変速線 (scrv)
!  opn      : 入力開度 %
!  v        : 入力速度 km/h
!  [output]
!  iret     : 判定結果 (戻り値)
!  luout    : 変速線から取得した変速後のLU出力
! ************************************************************
subroutine gpcheck ( bl, opn, v, iret, luout )
    use hvtransfc
```

```
    implicit none

    integer, parameter :: EQUAL = 0                             ! 速度線に一致
    integer, parameter :: ABOVE = 1                             ! 速度線より大きい
    integer, parameter :: BELOW = -1                            ! 速度線より小さい
    integer, parameter :: UNDEF = -2                            ! 速度線の定義なし

    type( scrv ) :: bl
    integer :: i, iret, luout
    real(8) :: opn, v, vout, a

!   <<< 変速線の定義なし -2 >>>
    if ( bl%n == 0 ) then
        iret = UNDEF
        luout = 0
        return
    end if

!   <<< 変速点判定 (直線補間) >>>
    if ( opn < 0.0_8 ) then
        opn = 0.0_8
    else if ( opn > 100.0_8) then
        opn = 100.0_8
    end if

    if ( opn < bl%x(1) .or. opn > bl%x(bl%n) ) then
        iret = UNDEF
        luout = 0
        return
    end if

    vout = -1.0_8
    do i = 2, bl%n
        if ( opn <= bl%x(i) ) then
            a = (bl%y(i) - bl%y(i-1)) / (bl%x(i) - bl%x(i-1))
            vout = bl%y(i-1) + a * ( opn - bl%x(i-1) )

            if( opn == bl%x(i) ) then
                luout = bl%l(i)
            else
                luout = bl%l(i-1)
            end if
            exit
        end if
    end do

!   <<< 変速線判定 >>>
    if ( v > vout ) then
        iret = ABOVE
    else if (v < vout) then
        iret = BELOW
    else
        iret = EQUAL
    end if

end subroutine




! ************************************************************
!  SUBROUTINE GEARPOS
!  modesw   : 車速モード判別 1:都市内モード
!  ii       : タイムステップ
!  tm       : 変速機諸元
!  ctr      : 変速制御マップ
!  vp       : 時刻i-1 の車速
!  vhx      : 車両走行状態データ
!  sout     : ギヤ位置出力
!  lu       : ロックアップ状態出力
!  tstop    : 停止後経過時間カウンタ
! ************************************************************
subroutine gearpos ( modesw, ii, tm, ctr, vp, vhx, sout, lu, tstop )
    use hvtransfc
    implicit none

    integer, parameter :: UNDEF = -2                            ! 速度線なし
    integer, parameter :: EQUAL = 0                             ! 速度線に一致
    integer, parameter :: ABOVE = 1                             ! 速度線より大きい
    integer, parameter :: BELOW = -1                            ! 速度線より小さい

    type (stm) :: tm
    type (sctr) :: ctr
    type (vehout) :: vhx
    integer :: i, sout, iret, lu, lunew, luout, ii, tstop, modesw
    real(8) :: vp

    sout = vhx%s

!   ----------------------------------------------------------
!    発進, 停止, Nアイドル制御
    if ( vhx%v <= 0.0_8 ) then

        if ( vhx%v <= 0.0_8 ) then
            tstop = vhx%tstop + 1
        else
```

```
            tstop = 1
        end if

!       <<< ニュートラルアイドル処理 >>>
        if ( modesw == 1 .and. ctr%swnidl == 1 ) then

            if ( ii <= 25 ) then
                if ( ctr%sw_nidl_start == 1 .and. tstop > ctr%time_nidl_start ) then
                    sout = 0
                else
                    sout = sout
                end if
            else
                if ( tstop > ctr%time_nidl ) then
                    sout = 0
                else
                    sout = sout
                end if
            end if

        end if

!       <<< アイドルストップ処理 >>>
        if ( modesw == 1 .and. ctr%swistop == 1 ) then

            if ( ii <= 25 ) then
                if ( ctr%sw_iss_start == 1 .and. tstop > ctr%time_iss_start ) then
                    sout = -1
                else
                    sout = sout
                end if
            else
                if ( tstop > ctr%time_iss ) then
                    sout = -1
                else
                    sout = sout
                end if
            end if

        end if

        lu = 0
        return

    end if


!   ---------------------------------------------------------
!    時刻iに走行の場合
    tstop = 0
    if( vhx%v <= 0.0_8 ) then
        sout = tm%grs
        lu = 0
        return
    end if


!   ---------------------------------------------------------
!    自動変速　ギヤ位置決定
    lunew = vhx%lu

!   ---------------------------------------------------------
!    シフトアップ線
    do i = vhx%s, tm%ngr-1

        if ( ( vhx%v < vp ) .and. ( vhx%tp <= 0.0_8 .or. vhx%opn == 0.0_8 ) ) exit

        if ( vhx%v >= minval(ctr%up%g(i)%y) ) then
            call gpcheck( ctr%up%g(i), vhx%opn, vhx%v, iret, luout )
            if ( iret==ABOVE .or. iret==EQUAL ) then
                sout = i + 1
                lunew = luout
            end if
        else
            exit
        end if
    end do

!   ---------------------------------------------------------
!    シフトダウン線
    do i = vhx%s, 2, -1
        if( vhx%v < maxval(ctr%dwn%g(i)%y) )then
            call gpcheck( ctr%dwn%g(i), vhx%opn, vhx%v, iret, luout )
            if( iret == BELOW ) then
                sout = i-1
                lunew = luout
            end if
        else
            exit
        end if
    end do

    if ( sout /= vhx%s ) then

        if ( lunew == 2 ) then
            call lucheck( ctr%lon%g(sout), vhx%opn, vhx%v, iret )
```

```
            if( iret == UNDEF ) lunew = 0
        end if

    end if



!   ------------------------------------------------------
!    ロックアップ制御
    if ( lunew == 0 ) then

        call lucheck( ctr%lon%g(sout), vhx%opn, vhx%v, iret )
        if( iret == ABOVE .or. iret == EQUAL ) then
            lunew = 2
        end if

    else if ( lunew == 2 ) then

        call lucheck( ctr%loff%g(sout), vhx%opn, vhx%v, iret )
        if( iret == BELOW ) then
            lunew = 0
        end if

    end if

    lu = lunew

end subroutine




!************************************************************
! SUBROUTINE THRFROMMAP2
!  eng     : エンジン諸元データ
!  ne      : エンジン回転数 (rpm)
!  te      : エンジントルク (Nm)
!  taout   : アクセル開度出力 (%)
!************************************************************
subroutine thrfrommap2( eng, ne, te, taout )
    use hvtransfc
    use msubfc
    implicit none

    type ( seng ) :: eng
    real(8) :: ne, te, taout
    real(8), allocatable :: x(:), y(:)
    real(8), allocatable :: thr(:), tq(:)
    integer :: i, j, flg
    real(8) :: ds, tqh, tqdif

    if ( ne == eng%nidle ) then
        taout = 0.0_8
        return
    end if

    allocate ( thr( 1:eng%gegmp%thrn ), tq( 1:eng%gegmp%thrn ) )

    do i = 1, eng%gegmp%thrn

        thr(i) = eng%gegmp%thr(i)
        allocate( x( 1:eng%gegmp%n(i) ), y( 1:eng%gegmp%n(i) ) )

        do j = 1, eng%gegmp%n(i)
            x(j) = eng%gegmp%r( i, j )
            y(j) = eng%gegmp%t( i, j )
        end do

        call HERMITEIP( x, y, eng%gegmp%n(i), ne, tq(i) )
        deallocate ( x, y )

    end do

    if( te > maxval(tq) ) then
        taout = 100.0_8

    else if ( te < minval(tq) ) then
        taout = 0.0_8

    else
!       <<< 区分三次エルミート補間によるアクセル開度算出 >>>
        ds = 100.0_8
        flg = 0
        taout = 0.0_8
        do while ( flg == 0 )
            call HERMITEIP( thr, tq, eng%gegmp%thrn, taout, tqh )

            tqdif = tqh - te
            if ( dabs(tqdif) < 1.0E-10 .and. dabs(tqdif) >= 0.0_8 ) then
                flg = 1
            else if ( tqdif < 0.0_8 ) then
                taout = taout + ds
            else
                ds = ds / 2.0_8
                taout = taout - ds
            end if
```

```
            if ( taout < 0.0_8 ) then
                taout = 0.0_8
                flg = 1
            else if ( taout > 100.0_8 ) then
                taout = 100.0_8
                flg = 1
            end if
        end do

    end if

    deallocate ( thr, tq )

end subroutine




!************************************************************
!  SUBROUTINE DRIVECYCLE 1ステップ走行
!   modesw  : 走行モード識別, 1=都市内, 2=都市間
!   i       : タイムステップ
!   spec    : 車両諸元データ
!   eng     : エンジン諸元データ
!   tm      : 変速機諸元データ
!   tc      : トルコン諸元データ
!   ctrl    : 変速制御マップ
!   vhx     : 走行状態出力
!   vpat    : 車速パタン入力
!   iret    : 計算戻り値
!************************************************************
subroutine drivecycle ( modesw, i, spec, eng, tm, tc, ctrl, vhx, vpat, iret )
    use hvtransfc
    implicit none

    type ( sspec ) :: spec
    type ( seng ) :: eng
    type ( stm ) :: tm
    type ( stc ) :: tc
    type ( sctr ) :: ctrl
    type ( scycle ) :: vpat
    type ( vehout ), pointer :: vhx(:)
    integer :: gp, lu, tstop, i, iret, modesw

    vhx(i)%s = vhx(i-1)%s
    vhx(i)%lu = vhx(i-1)%lu
    vhx(i)%tstop = vhx(i-1)%tstop
    vhx(i)%c = vhx(i-1)%c
    tstop = vhx(i-1)%tstop

!   <<< ATトルク伝達の計算 >>>
    call driveAT( spec, eng, tm, tc, ctrl, vhx(i-1)%v, vpat%v(i), vpat%g(i), vhx(i), iret, 0 )

!   <<< 変速条件を満たしたか? >>>
    call gearpos ( modesw, i, tm, ctrl, vhx(i-1)%v, vhx(i), gp, lu, tstop )

    vhx(i)%tstop = tstop

!   <<< 変速した場合, 新しいギヤで状態再計算 >>>
    if ( vhx(i)%s /= gp .or. vhx(i)%lu /= lu ) then

        vhx(i)%s = gp
        vhx(i)%lu = lu
        if( vhx(i)%s ==tm%grs .and. vhx(i-1)%s > vhx(i)%s ) then
            vhx(i)%c = 0
        end if
!       --- トルク伝達再計算 ---
        call driveAT( spec, eng, tm, tc, ctrl, vhx(i-1)%v, vpat%v(i), vpat%g(i), vhx(i), iret, 0 )

    end if

    if ( vpat%v(i) > vhx(i)%v ) then
        vhx(i)%verrflg = 1
    else
        vhx(i)%verrflg = 0
    end if

end subroutine
```

```
! ************************************************************
! MODULE MSUBFC
!  definition of fuel consumption calculating subroutine
! ************************************************************
module msubfc

    interface

        subroutine fcgrid3( nidle, gfcm, fcmidl, fcm, gegmp )
            use hvtransfc
            real(8) :: nidle
            type (sgridmap) :: gfcm
            type (sfcmap) :: fcm, fcmidl
            type (sgridengmap) :: gegmp
        end subroutine fcgrid3

        subroutine calcfcAT( vhx, n, eng )
            use hvtransfc
            type (vehout), pointer :: vhx(:)
            type (seng) :: eng
            integer :: n
        end subroutine calcfcAT

        subroutine calcave( vhx, s, n, avv, avfc, terr )
            use hvtransfc
            type ( vehout ), pointer :: vhx(:)
            integer :: s, n, terr
            real(8) :: avv, avfc
        end subroutine calcave

        subroutine HERMITEIP( xx, yy, n, xin, yout )
            real(8) :: xx(*), yy(*)
            integer :: n
            real(8) :: xin, yout
        end subroutine

    end interface

end module msubfc


! ************************************************************
! SUBROUTINE FCGRID3 燃費マップよりNX×NY点の格子マップ作成
!  gfcm   : 加工済み燃費マップ
!  fcm    : 入力燃費マップ
!  fcmidl : アイドル回転での燃費曲線 (停止時計算用)
!  gegmp  : 加工済みエンジントルクマップ
! ************************************************************
subroutine fcgrid3( nidle, gfcm, fcm, fcmidl, gegmp )
    use hvtransfc
    implicit none

    integer, parameter :: nx = 20, ny = 20
    real(8) :: nidle
    type (sfcmap) :: fcm, fcmidl
    type (sgridmap) :: gfcm
    type (sgridengmap) :: gegmp

    integer :: i, j, ierr
    integer :: rn, tn, tnmax
    real(8), allocatable :: r(:), t(:,:), f(:,:), x(:)
    integer, allocatable :: tqn(:)
    real(8) :: nemin, nemax, tqmin, tqmax, ftq
    real(8) :: dne, dtq, rr
    logical :: skip, bidle
    real(8), allocatable :: d1(:,:), f1(:,:)
    real(8), pointer :: pchval(:)

    allocate ( gfcm%rev(1:nx), gfcm%tq(1:ny), gfcm%fc(nx, ny) )
    gfcm%nx = nx
    gfcm%ny = ny

    nemin = minval( fcm%rev )
    nemax = maxval( fcm%rev )
    tqmin = minval( fcm%tq )
    tqmax = maxval( fcm%tq )

    rn = 1
    tn = 1
    tnmax = 1
    rr = fcm%rev(1)
    do i = 1, fcm%ndata
        if ( rr /= fcm%rev(i) ) then
            rn = rn + 1
            rr = fcm%rev(i)
            if( tn > tnmax ) tnmax = tn
            tn = 1
        end if
        tn = tn + 1
    end do

    allocate ( r(1:rn), tqn(1:rn) )
    allocate ( t(rn, tnmax), f(rn, tnmax) )


    j = 1
```

```
    r(j) = fcm%rev(1)
    tqn(j) = 1
    call egmapTE( gegmp, r(j), 0.0_8, ftq )
    t( j, tqn(j) ) = ftq
    f( j, tqn(j) ) = 0.0_8
    do i = 1, fcm%ndata
        if ( r(j) /= fcm%rev(i) ) then
            j = j + 1
            r(j) = fcm%rev(i)
            tqn(j) = 1
            call egmapTE( gegmp, r(j), 0.0_8, ftq )
            t( j, tqn(j) ) = ftq
            f( j, tqn(j) ) = 0.0_8
            tqmin = min( tqmin, t(j,tqn(j)) )
        end if
        tqn(j) = tqn(j) + 1
        t( j, tqn(j) ) = fcm%tq(i)
        f( j, tqn(j) ) = fcm%fc(i)
    end do


    dne = ( nemax - nemin ) / real(nx-1, 8)
    dtq = ( tqmax - tqmin ) / real(ny-1, 8)

    do i = 1, nx
        gfcm%rev(i) = nemin + dne * real(i-1, 8)
    end do
    do i = 1, ny
        gfcm%tq(i) = tqmin + dtq * real(i-1, 8)
    end do


    allocate ( f1(rn, ny), d1(rn, tnmax) )
    do i = 1, rn
        allocate( x( tqn(i) ) )
        do j = 1, tqn(i)
            x(j) = t( i, j )
        end do

        call DPCHIM( tqn(i), x, f(i,1), d1(i,1), rn, ierr )
        if( ierr < 0 ) call errhermite( 1, ierr )

        skip = .true.
        allocate ( pchval( ny ) )
        call DPCHFE( tqn(i), x, f(i,1), d1(i,1), rn, skip, gfcm%ny, gfcm%tq, pchval, ierr )
        if( ierr < 0 ) call errhermite( 2, ierr )

        do j = 1, ny
            f1(i,j) = pchval(j)
        end do

        deallocate ( pchval, x )
    end do

    deallocate ( d1 )


    allocate ( d1(rn, ny) )
    do i = 1, ny
        call DPCHIM( rn, r, f1(1,i), d1(1,i), 1, ierr )
        if( ierr < 0 ) call errhermite ( 3, ierr )

        skip = .true.
        allocate ( pchval( 1:nx ) )
        call DPCHFE( rn, r, f1(1,i), d1(1,i), 1, skip, gfcm%nx, gfcm%rev, pchval, ierr )
        if( ierr < 0 ) call errhermite ( 4, ierr )

        do j = 1, nx
            gfcm%fc(j,i) = pchval(j)
        end do

        deallocate ( pchval )
    end do



    bidle = .false.
    do i = 1, rn
        if ( r(i) == nidle ) then
            fcmidl%ndata = tqn(i)
            allocate ( fcmidl%rev( 1:tqn(i) ) )
            allocate ( fcmidl%tq(1:tqn(i) ) )
            allocate ( fcmidl%fc(1:tqn(i) ) )

            do j = 1, fcmidl%ndata
                fcmidl%rev(j) = r(i)
                fcmidl%tq(j) = t( i, j )
                fcmidl%fc(j) = f( i, j )
            end do
            bidle = .true.
            exit
        end if
!
    end do
!
    if (bidle == .false. ) then
```

```
        write(0,*) "燃費マップにアイドル回転数での測定データが入力されていません. "
        write(0,*) "Nidle 入力値 = ", nidle, "rpm"
        pause
        stop
    end if

    deallocate ( d1, f1, r, tqn, t, f )

end subroutine




! ************************************************************
! SUBROUTINE CALCFCAT  1秒毎の燃料消費量計算
!  vhx : 入力走行データ
!  n     : データ点数
!  eng   : エンジン諸元データ
! ************************************************************
subroutine calcfcAT( vhx, n, eng )
    use hvtransfc
    implicit none

    type (vehout), pointer :: vhx(:)
    type (seng), target :: eng
    integer :: n

    type (sgridengmap), pointer :: gegmp
    type (sgridmap), pointer :: gfcm
    integer :: i, j, ierr
    real(8) :: ftq
    real(8), allocatable :: f(:), d2(:), d(:,:)
    logical :: skip

    gfcm => eng%gfcm
    gegmp=> eng%gegmp

    allocate( d( gfcm%nx, gfcm%ny ) )
    allocate( f(1:gfcm%ny), d2(1:gfcm%ny) )

!   ------------------------------------------------------
    do j = 1, n

        if ( vhx(j)%ne <= 0.0_8 ) then
            vhx(j)%fc = 0.0_8
            cycle
        end if

!       ----------------------------
!        アイドル回転での燃料消費量
        if ( vhx(j)%ne == eng%nidle ) then
            call HERMITEIP( eng%fcmidl%tq, eng%fcmidl%fc, eng%fcmidl%ndata, vhx(j)%te, vhx(j)%fc )
            cycle
        end if

!       --------------------------------
!        アイドル以外での燃料消費量計算
        call egmapTE( gegmp, vhx(j)%ne, 0.0_8, ftq )

        if ( vhx(j)%te <= ftq ) then
            vhx(j)%fc = 0.0_8

        else
            do i = 1, gfcm%ny
                call DPCHIM( gfcm%nx, gfcm%rev, gfcm%fc(1,i), d(1,i), 1, ierr )
                if( ierr < 0 ) then
                    call errhermite ( 5, ierr )
                end if

                skip = .true.
                call DPCHFE(gfcm%nx,gfcm%rev,gfcm%fc(1,i),d(1,i),1,skip,1,vhx(j)%ne,f(i),ierr)
                if( ierr < 0 ) then
                    call errhermite ( 6, ierr )
                end if
            end do

            skip = .true.
            call DPCHIM( gfcm%ny, gfcm%tq, f, d2, 1, ierr )
            if( ierr < 0 ) then
                call errhermite ( 7, ierr )
            end if

            call DPCHFE( gfcm%ny,gfcm%tq,f,d2,1,skip,1,vhx(j)%te,vhx(j)%fc,ierr )
            if( ierr < 0 ) then
                call errhermite ( 8, ierr )
            end if

            if( vhx(j)%fc < 0.0_8 ) then
                vhx(j)%fc = 0.0_8
            end if
        end if
    end do

    deallocate( d, d2,f )
    nullify ( gegmp, gfcm )
end subroutine
```

```
! ************************************************************
! SUBROUTINE CALCAVE 平均速度, 燃費の計算
!  vhx  : 入力走行データ
!  s    : 計算開始指標
!  n    : データ点数
!  avv  : 平均速度 (km/h)
!  avfc : 燃費 (km/L)
!  terr : 追従不能累積時間 (s)
! ************************************************************
subroutine calcave( vhx, s, n, avv, avfc, terr )
    use hvtransfc
    implicit none

    type (vehout), pointer :: vhx(:)
    integer :: n, i, s, terr
    real(8) :: avv, avfc, sumv, sumf

    sumv = 0.0_8
    sumf = 0.0_8
    terr = 0

    do i = s, s+n-1
        sumv = sumv + vhx(i)%v
        sumf = sumf + vhx(i)%fc
        if ( vhx(i)%verrflg == 1 ) then
            terr = terr + 1
        end if

    end do

    avv = sumv / (real(n,8) * 1.0_8 )
    avfc = sumv / sumf

end subroutine



! ************************************************
!  SUBROUTINE HERMITEIP 区分三次エルミート補間計算
!  xx     : x座標配列
!  yy     : y座標配列
!  n      : xx, yyの要素数
!  xin    : x 入力値
!  yout   : エルミート補間値 (出力値)
! ************************************************
subroutine HERMITEIP( xx, yy, n, xin, yout )
    implicit none

    real(8) :: xx(*), yy(*)
    integer :: n, ierr
    real(8) :: xin, yout
    logical :: skip
    real(8), allocatable :: d(:)

    allocate( d(n) )
    skip = .true.

    call dpchim(n, xx, yy, d, 1, ierr)
    if( ierr < 0 ) call errhermite ( 1, ierr )

    call dpchfe(n, xx, yy, d, 1, skip, 1, xin, yout, ierr)
    if( ierr < 0 ) call errhermite ( 2, ierr )

    deallocate( d )

end subroutine



! ************************************************************
! SUBROUTINE ERRHERMITE
!  error handler for hermite interpolation
! ************************************************************
subroutine errhermite ( errid, ierr )
    implicit none

    integer :: errid, ierr
    character (len=1024) :: msg

    select case ( errid )
!   ------------------------------------------------------
!   at FCGRID
!   ------------------------------------------------------
    case( 1 )
        msg = "Error : DPCHIM ( called from FCGRID, 1st )"
    case( 2 )
        msg = "Error : DPCHFE ( called from FCGRID, 1st )"
    case( 3 )
        msg = "Error : DPCHIM ( called from FCGRID, 2nd )"
    case( 4 )
```

```
        msg = "Error : DPCHFE ( called from FCGRID, 2nd )"

!   ----------------------------------------------------------
!  at CALCFC
!   ----------------------------------------------------------
    case( 5 )
        msg = "Error : DPCHIM ( called from CALCFC, 1st )"
    case( 6 )
        msg = "Error : DPCHFE ( called from CALCFC, 1st )"
    case( 7 )
        msg = "Error : DPCHIM ( called from CALCFC, 2nd )"
    case( 8 )
        msg = "Error : DPCHFE ( called from CALCFC, 2nd )"
    end select

    write( 0, '( A )' ) msg
    write ( 0, '( A, I )' ) "ierr = ", ierr
    stop ' Terminated by error of Hermite function. Check input fuel map.'

end subroutine



! ************************************************************
! DPCHST:  SIGN TESTING OF 2 ARGUMENTS
!     Returns:
!        -1. if ARG1 and ARG2 are of opposite sign.
!         0. if either argument is zero.
!        +1. if ARG1 and ARG2 are of the same sign.
! ************************************************************
real(8) function DPCHST( arg1, arg2 )
    implicit none

    real(8) :: arg1, arg2

    if ( ( arg1 == 0.0_8 ) .or. ( arg2 == 0.0_8 ) ) then
        DPCHST = 0.0_8
    else
        DPCHST = dsign(1.0_8, arg1) * dsign(1.0_8, arg2)
    end if

end function



! ************************************************************
! DPCHIM: Piecewise Cubic Hermite Interpolation to Monotone data
! ************************************************************
subroutine DPCHIM ( n, x, f, d, incfd, ierr )
    implicit none
    interface
        real(8) function DPCHST( arg1, arg2 )
            real(8) :: arg1, arg2
        end function
    end interface

    integer :: n, incfd, ierr
    real(8) :: x(*), f(incfd,*), d(incfd,*)

    integer :: i, nsec
    real(8) :: dx1, dx2, del1, del2, dsave, dxsum, dxsumt3, dmax, dmin
    real(8) :: w1, w2, drat1, drat2, sgn

!   ----- check argument ---------------------------------------
    ierr = 0
    if( n < 2 ) then
        ierr = -1
        return
    end if

    if( incfd < 1 ) then
        ierr = -2
        return
    end if

    do i = 2, n
        if( x(i) > x(i-1) ) then
            continue
        else
            ierr = -3
            return
        end if
    end do

!   ----- Piecewise Cubic Hermite Interpolation ----------------
    nsec = n -1
    dx1 = x(2) - x(1)
    del1 = ( f(1,2) - f(1,1) ) / dx1
    dsave = del1

    if( n == 2 )then
        d(1,1) = del1
        d(1,2) = del1
        return
    end if
```

```
    dx2 = x(3) - x(2)
    del2 = ( f(1,3) - f(1,2) ) / dx2

    dxsum = dx1 + dx2
    w1 = ( dx1 + dxsum ) / dxsum
    w2 = -dx1 / dxsum
    d(1,1) = w1 * del1 + w2 * del2
    if( DPCHST( d(1,1), del1 ) <= 0.0_8 ) then
        d(1,1) = 0.0_8
    else if( DPCHST( del1, del2 ) < 0.0_8 ) then
        dmax = 3.0_8 * del1
        if( abs( d(1,1) ) > abs( dmax ) ) then
            d(1,1) = dmax
        end if
    end if

    do i = 2, nsec
        if( i /= 2 )then
            dx1 = dx2
            dx2 = x(i+1) - x(i)
            dxsum = dx1 + dx2
            del1 = del2
            del2 = ( f(1,i+1) - f(1,i) ) / dx2
        end if

        d( 1, i ) = 0.0_8
        sgn = DPCHST( del1, del2 )

        if( sgn == -1.0_8 ) then
            ierr = ierr + 1
            dsave = del2

        else if( sgn == 0.0_8 ) then
            if( del2 /= 0.0_8 ) then
                if( DPCHST( dsave, del2 ) < 0.0_8 ) then
                    ierr = ierr + 1
                end if
                dsave = del2
            end if

        else
            dxsumt3 = dxsum * 3.0_8
            w1 = ( dxsum + dx1 ) / dxsumt3
            w2 = ( dxsum + dx2 ) / dxsumt3
            dmax = max( abs(del1), abs(del2) )
            dmin = min( abs(del1), abs(del2) )
            drat1 = del1 / dmax
            drat2 = del2 / dmax
            d(1,i) = dmin / ( w1 * drat1 + w2 * drat2 )
        end if
    end do

    w1 = -dx2 / dxsum
    w2 = ( dx2 + dxsum ) / dxsum
    d( 1, n ) = w1 * del1 + w2 * del2
    if( DPCHST( d(1,n), del2 ) <= 0.0_8 ) then
        d(1,n) = 0.0_8
    else if( DPCHST( del1, del2 ) < 0.0_8 ) then
        dmax = 3.0_8 * del2
        if( abs( d(1,n) ) > abs( dmax ) ) then
            d( 1, n ) = dmax
        end if
    end if

end subroutine DPCHIM



! ************************************************************
! DPCHFE:  Piecewise Cubic Hermite Function Evaluator
! ************************************************************
subroutine DPCHFE( n, x, f, d, incfd, skip, ne, xe, fe, ierr )
    implicit none

    integer :: n, incfd, ne, ierr
    real(8) :: x(*), f(incfd,*), d(incfd,*), xe(*), fe(*)
    logical :: skip

    integer i, ierc, ir, j, j2,jfirst, next(2), nj

    ierr = 0
!   ----- check argument ----------------------------------------
    if( skip ) then
        if( n < 2 ) then
            ierr = -1
            return
        end if

        if( incfd < 1 ) then
            ierr = -2
            return
        end if

        do i = 2, n
            if( x(i) <= x(i-1) ) then
                ierr = -3
```

```
                return
            end if
        end do

        if( ne < 1 ) then
            ierr = -4
            return
        end if
    end if


!   ----- piecewise cubic hermite function evaluator ------------
    skip = .true.
    jfirst = 1
    ir = 2

    do while ( jfirst <= ne )
        j2 = ne + 1
        do j = jfirst, ne
            if( xe(j) >= x(ir) ) then
                if( ir == n ) then
                    j2 = ne + 1
                else
                    j2 = j
                end if
                exit
            end if
        end do
        nj = j2 - jfirst

        if( nj /= 0 ) then
            call DCHFEV (x(ir-1),x(ir),f(1,ir-1),f(1,ir),d(1,ir-1),d(1,ir),nj,xe(jfirst),fe(jfirst),next,ierc)
            if( ierc < 0 ) then
                ierr = -5
                return
            end if

            if( next(2) == 0 ) then

            else
                if( ir < n ) then
                    ierr = -5
                    return
                else
                    ierr = ierr + next(2)
                end if
            end if

            if( next(1) == 0 ) then

            else
                if( ir > 2 ) then
                    do i = jfirst, j2-1
                        if( xe(i) < x(ir-1) ) then
                            exit
                        end if
                    end do
                    if( i > j2-1 ) then
                        ierr = -5
                        return
                    end if
                    j2 = i
                    do i = 1, ir-1
                        if( xe(j2) < x(i) ) exit
                    end do
                    ir = max( 1, i-1 )
                else
                    ierr = ierr + next(1)
                end if
            end if
            jfirst = j
        end if

        ir = ir + 1
        if( ir > n ) exit
    end do

end subroutine



! ************************************************************
! DCHFEV:  Cubic Hermite Function EValuator
! ************************************************************
subroutine DCHFEV (x1, x2, f1, f2, d1, d2, ne, xe, fe, next, ierr)
    implicit none

    integer :: i, ne, ierr, next(2)
    real(8) :: x1, x2, f1, f2, d1, d2, xe(*), fe(*)
    real(8) :: c2, c3, del1, del2, delta,h, x, xmi, xma

    ierr = 0
!   ----- check arguments----------------------------------------
    if( ne < 1 ) then
        ierr = -1
        return
```

```
    end if

    h = x2 - x1
    if( h == 0.0_8 ) then
        ierr = -2
        return
    end if

!   ----- cubic hermite function evaluator ----------------------
    next(1) = 0
    next(2) = 0
    xmi = min( 0.0_8, h )
    xma = max( 0.0_8, h )

    delta = ( f2 - f1 ) / h
    del1 = ( d1 - delta ) / h
    del2 = ( d2 - delta ) / h
    c2 = -( del1 + del1 + del2 )
    c3 = ( del1 + del2 ) / h

    do i = 1, ne
        x = xe(i) - x1
        fe(i) = f1 + x * ( d1 + x * ( c2 + x * c3 ) )
        if( x < xmi ) then
            next(1) = next(1) + 1
        end if
        if( x > xma ) then
            next(2) = next(2) + 1
        end if
    end do

end subroutine
```

```
module vehicleparams

    interface
        subroutine setparametersAT( spec, tm, ctrl )
            use hvtransfc
            type (sspec) :: spec
            type (stm) :: tm
            type ( sctr ) :: ctrl
        end subroutine

        subroutine showinputdata( spec, eng, tm )
            use hvtransfc
            type (sspec) spec
            type (seng) eng
            type (stm) :: tm
        end subroutine
    end interface

end module




!*********************************************************************
! SUBROUTINE SETPARAMETERSAT
! spec  : 車両諸元
! tm    : 変速機諸元
! ctrl  : 変速制御データ
!*********************************************************************
subroutine setparametersAT( spec, tm, ctrl )
    use hvtransfc
    implicit none

    real(8), parameter :: PDWT       =  0.07_8
    real(8), parameter :: PDWE       =  0.03_8
    real(8), parameter :: PSGW       = 55.00_8
    real(8), parameter :: EGR_DIRECT =  0.98_8
    real(8), parameter :: EGR_ND     =  0.96_8
    real(8), parameter :: EGR_FINAL  =  0.95_8

    type (stm) :: tm
    type (sspec) :: spec
    type (sctr) :: ctrl
    integer :: i, j
    real(8) :: dwt, dwe, beta

!   ------------------
!    標準車両諸元設定
!   ------------------
    call setvehicletype( spec%tpname, spec )

    spec%gvw = spec%w0 + spec%wld + REAL(spec%crew, 8) * PSGW

    select case ( spec%tpname(1:1) )
    case ( 'T' )
        spec%wt = spec%w0 + spec%wld / 2.0_8 + PSGW
        spec%mur = ( 0.00513_8 + 17.6_8/spec%wt ) * 9.8_8
        spec%mua = ( 0.00299_8 * spec%bh*spec%bw - 0.000832_8 ) * 9.8_8
    case ( 'B' )
        spec%wt = spec%w0 + spec%crew * PSGW / 2.0_8
        spec%mur = ( 0.00513_8 + 17.6_8/spec%wt ) * 9.8_8
        spec%mua = ( 0.00299_8 * spec%bh*spec%bw - 0.000832_8 ) * 0.680_8 * 9.8_8
    case default
        write (0, '(3A)') &
          " Vehicle Type Error, [ ", trim(spec%tpname), " ] is not defined."
        stop
    end select

!   ----------------------
!    変速機パラメータ設定
!   ----------------------
    if ( spec%tpname == 'TT1' .or. spec%tpname == 'T10' .or. spec%tpname == 'T11' ) then
        if ( tm%ngr == 7 ) then
            if ( tm%g(7)%gr == 1.0_8 ) then
                beta = 0.6_8
            else if ( tm%g(6)%gr ==1.0_8 ) then
                beta = 0.81_8
            else
                beta = 1.0_8
            end if
        else if ( tm%ngr == 12 ) then
            if ( tm%g(12)%gr == 1.0_8 ) then
                beta = 0.6_8
            else
                beta = 1.0_8
            end if
        else
            beta = 1.0_8
        end if
    else
        beta = 1.0_8
    end if

!   --- 回転部分相当質量, 効率設定 ---
    dwt = spec%w0 * PDWT
    dwe = spec%w0 * PDWE
```

```
    spec%efgr = EGR_FINAL

    do i = 1, tm%ngr

        tm%g(i)%dw = dwt + dwe * ( tm%g(i)%gr * beta )**2

        if ( tm%g(i)%gr == 1.0_8 ) then
            tm%g(i)%egr = EGR_DIRECT
        else
            tm%g(i)%egr = EGR_ND
        end if
    end do


!   ----------------------------
!    L/Uマップの回転数－速度変換
!   ----------------------------
!   --- LOCK-UP ON ---
    do i = 1, tm%ngr
        do j = 1, ctrl%lon%g(i)%n
            ctrl%lon%g(i)%y(j) = ctrl%lon%g(i)%y(j) / spec%fgr * (0.3768_8*spec%rt)
        end do
    end do

!   --- LOCK-UP OFF ---
    do i = 1, tm%ngr
        do j = 1, ctrl%loff%g(i)%n
            ctrl%loff%g(i)%y(j) = ctrl%loff%g(i)%y(j) / spec%fgr * (0.3768_8*spec%rt)
        end do
    end do

!   ----------------------------
!    変速マップの回転数－速度変換
!   ----------------------------
!   --- up ---
    do i = 1, tm%ngr
        do j = 1, ctrl%up%g(i)%n
            ctrl%up%g(i)%y(j) = ctrl%up%g(i)%y(j) / spec%fgr * (0.3768_8*spec%rt)
        end do
    end do

!   --- down ---
    do i = 1, tm%ngr
        do j = 1, ctrl%dwn%g(i)%n
            ctrl%dwn%g(i)%y(j) = ctrl%dwn%g(i)%y(j) / spec%fgr * (0.3768_8*spec%rt)
        end do
    end do


end subroutine




!*************************************************************
!  SHOWINPUTDATA
!  spec   : 車両諸元
!  eng    : エンジン諸元
!  tm     : 変速機諸元
!*************************************************************
subroutine showinputdata( spec, eng, tm )
    use hvtransfc
    implicit none

    type (sspec) :: spec
    type (seng) :: eng
    type (stm) :: tm
    integer :: i

    print*, "-------------------------------------------------------"
    print '(2A)', " Vehicle Name    : ", trim(spec%vhname)
    select case ( spec%tpname(1:1) )
    case ( 'T' )
        print '(A)', " Vehicle type  : TRUCK"
    case ( 'B' )
        print '(A)', " Vehicle type  : BUS"
    end select
    print '(2A)',     " Category      : ", trim(spec%tpname)

    print '(A,F0.2,A)', " HIGHWAY RATIO : ", spec%ric * 100.0_8, " [%]"
    print*

    print*, "-------------------------------------------------------"
    print '(A,F8.2,A)', " GVW =", spec%gvw, "[kg]"
    print '(A,F8.2,A,F8.2,A)', " Wcurb =", spec%w0, "[kg], Wtest =", spec%wt, "[kg]"
    print '(A,F8.3,A,F8.3,A,F8.3,A)', " Width =", spec%bw,              &
          "[m],  Height=" , spec%bh, "[m], Tire radius=",spec%rt, "[m]"
    print '(A,I3)'," Crew =", spec%crew
    print '(A,F8.2,A,F8.2,A,F8.2,A)', " Nidle =", eng%nidle,            &
          "[rpm], Nrate =", eng%nrate, "[rpm], Nex  =", eng%nex, "[rpm]"
    print*
    print '(A,F10.6,A)', " MuAir =", spec%mua, " [N/(km/h)^2]"
    print '(A,F10.6,A)', " MuRoll =", spec%mur, " [N/kg]"
    print*
    print*
```

```
    print '(A,I0)', " START GEAR = ", tm%grs
    print '(A,I3)', " NUMBER OF GEAR =", tm%ngr
    print '(A)', " GEAR   RATIO  EFFICIENCY  DW[kg]"
    do i = 1, tm%ngr
        print '(I4,A,F8.3,F10.3,F15.5)', i, ": ", tm%g(i)%gr, tm%g(i)%egr, tm%g(i)%dw
    end do
    print '(A,F8.3,F10.3)', " FIN: ", spec%fgr, spec%efgr
    print*

end subroutine showinputdata




! ************************************************************
! SUBROUTINE SETVEHICLETYPE
!  categoryname : 燃費区分呼び名
!  spec : 車両諸元データ
! ************************************************************
subroutine setvehicletype( categoryname, spec )
    use hvtransfc
    implicit none
    type (sspec) :: spec
    character (len=*) :: categoryname

    call toUpper( categoryname )

    select case ( categoryname )
!   ----- TRUCK TRACTOR, "TTx" ---------------------------------
    case ("TT1")                                                    ! TT1 GVW <=20t
        spec%w0 = 10525.0_8
        spec%wld = 24000.0_8
        spec%crew = 2
        spec%bh = 2.927_8
        spec%bw = 2.490_8
        spec%ric = 0.2_8

    case ("TT2")                                                    ! TT2 GVW >20t
        spec%w0 = 19028.0_8
        spec%wld = 40000.0_8
        spec%crew = 2
        spec%bh = 2.890_8
        spec%bw = 2.490_8
        spec%ric = 0.1_8

!   ----- GENERAL TRUCK, "Tx" ----------------------------------
    case ("T1")                                                     ! T1 3.5t< GVW <= 7.5t
        spec%w0 = 1957.0_8                                          ! Payload <=1.5t
        spec%wld = 1490.0_8
        spec%crew = 3
        spec%bh = 1.982_8
        spec%bw = 1.695_8
        spec%ric = 0.1_8

    case ("T21")                                                    ! T1 3.5t< GVW <= 7.5t
        spec%w0 = 2500.0_8                                          ! Payload <=1.5t
        spec%wld = 3475.0_8
        spec%crew = 3
        spec%bh = 1.982_8
        spec%bw = 1.695_8
        spec%ric = 0.1_8

    case ("T2")                                                     ! T2 3.5t< GVW <= 7.5t
        spec%w0 = 2356.0_8                                          ! 1.5t < Payload <=2t
        spec%wld = 2000.0_8
        spec%crew = 3
        spec%bh = 2.099_8
        spec%bw = 1.751_8
        spec%ric = 0.1_8

    case ("T3")                                                     ! T3 3.5t< GVW <= 7.5t
        spec%w0 = 2652.0_8                                          ! 2t < Payload <=3t
        spec%wld = 2995.0_8
        spec%crew = 3
        spec%bh = 2.041_8
        spec%bw = 1.729_8
        spec%ric = 0.1_8

    case ("T4")                                                     ! T4 3.5t< GVW <= 7.5t
        spec%w0 = 2979.0_8                                          ! 3t < Payload
        spec%wld = 3749.0_8
        spec%crew = 3
        spec%bh = 2.363_8
        spec%bw = 2.161_8
        spec%ric = 0.1_8

    case ("T5")                                                     ! T5 7.5t< GVW <= 8t
        spec%w0 = 3543.0_8
        spec%wld = 4275.0_8
        spec%crew = 2
        spec%bh = 2.454_8
        spec%bw = 2.235_8
        spec%ric = 0.1_8

    case ("T6")                                                     ! T6 8t< GVW <= 10t
        spec%w0 = 3659.0_8
```

```
        spec%wld = 5789.0_8
        spec%crew = 2
        spec%bh = 2.625_8
        spec%bw = 2.239_8
        spec%ric = 0.1_8

    case ("T7")                                                 ! T7 10t< GVW <= 12t
        spec%w0 = 4048.0_8
        spec%wld = 7483.0_8
        spec%crew = 2
        spec%bh = 2.541_8
        spec%bw = 2.350_8
        spec%ric = 0.1_8

    case ("T8")                                                 ! T8 12t< GVW <= 14t
        spec%w0 = 4516.0_8
        spec%wld = 7992.0_8
        spec%crew = 2
        spec%bh = 2.572_8
        spec%bw = 2.379_8
        spec%ric = 0.1_8

    case ("T9")                                                 ! T9 14t< GVW <= 16t
        spec%w0 = 5533.0_8
        spec%wld = 8900.0_8
        spec%crew = 2
        spec%bh = 2.745_8
        spec%bw = 2.480_8
        spec%ric = 0.1_8

    case ("T10")                                                ! T10 16t< GVW <= 20t
        spec%w0 = 8688.0_8
        spec%wld = 11089.0_8
        spec%crew = 2
        spec%bh = 3.049_8
        spec%bw = 2.490_8
        spec%ric = 0.1_8

    case ("T11")                                                ! T11 20t< GVW
        spec%w0 = 8765.0_8
        spec%wld = 15530.0_8
        spec%crew = 2
        spec%bh = 2.934_8
        spec%bw = 2.490_8
        spec%ric = 0.3_8

!   ----- ROOT BUS, CREW>=11persons, "BRx" ----------------------
    case ("BR1")                                                ! BR1 6t< GVW <= 8t
        spec%w0 = 5186.0_8
        spec%wld = 0.0_8
        spec%crew = 39
        spec%bh = 2.880_8
        spec%bw = 2.072_8
        spec%ric = 0.0_8

    case ("BR2")                                                ! BR2 8t< GVW <= 10t
        spec%w0 = 6672.0_8
        spec%wld = 0.0_8
        spec%crew = 46
        spec%bh = 2.947_8
        spec%bw = 2.301_8
        spec%ric = 0.0_8

    case ("BR3")                                                ! BR3 10t< GVW <= 12t
        spec%w0 = 7324.0_8
        spec%wld = 0.0_8
        spec%crew = 62
        spec%bh = 2.949_8
        spec%bw = 2.304_8
        spec%ric = 0.0_8

    case ("BR4")                                                ! BR4 12t< GVW <= 14t
        spec%w0 = 8654.0_8
        spec%wld = 0.0_8
        spec%crew = 77
        spec%bh = 2.969_8
        spec%bw = 2.385_8
        spec%ric = 0.0_8

    case ("BR5")                                                ! BR5 14t< GVW
        spec%w0 = 9790.0_8
        spec%wld = 0.0_8
        spec%crew = 79
        spec%bh = 2.962_8
        spec%bw = 2.490_8
        spec%ric = 0.0_8

!   ----- GENERAL BUS, CREW>=11persons, "Bx" --------------------
    case ("B1")                                                 ! B1 3.5t< GVW <= 6t
        spec%w0 = 3543.0_8
        spec%wld = 0.0_8
        spec%crew = 29
        spec%bh = 2.593_8
        spec%bw = 2.027_8
        spec%ric = 0.1_8
```

```
    case ("B2")                                                         ! B2 6t< GVW <= 8t
        spec%w0 = 5622.0_8
        spec%wld = 0.0_8
        spec%crew = 29
        spec%bh = 3.019_8
        spec%bw = 2.197_8
        spec%ric = 0.1_8

    case ("B3")                                                         ! B3 8t< GVW <= 10t
        spec%w0 = 6608.0_8
        spec%wld = 0.0_8
        spec%crew = 49
        spec%bh = 3.105_8
        spec%bw = 2.314_8
        spec%ric = 0.1_8

    case ("B4")                                                         ! B4 10t< GVW <= 12t
        spec%w0 = 8022.0_8
        spec%wld = 0.0_8
        spec%crew = 58
        spec%bh = 3.160_8
        spec%bw = 2.399_8
        spec%ric = 0.1_8

    case ("B5")                                                         ! B5 12t< GVW <= 14t
        spec%w0 = 9774.0_8
        spec%wld = 0.0_8
        spec%crew = 60
        spec%bh = 3.168_8
        spec%bw = 2.490_8
        spec%ric = 0.1_8

    case ("B6")                                                         ! B6 14t< GVW <= 16t
        spec%w0 = 12110.0_8
        spec%wld = 0.0_8
        spec%crew = 62
        spec%bh = 3.320_8
        spec%bw = 2.490_8
        spec%ric = 0.35_8

    case ("B7")                                                         ! B7 16t< GVW
        spec%w0 = 14583.0_8
        spec%wld = 0.0_8
        spec%crew = 51
        spec%bh = 3.668_8
        spec%bw = 2.490_8
        spec%ric = 0.35_8

    case default
        write (0,'(3A)') " Category Error : [ ", trim(categoryname), " ] is not defined."
        stop
   end select

end subroutine




! ************************************************************
! SUBROUTINE TOUPPER
!  convert small letters to capital letters
!  buf : target & converted strings
! ************************************************************
subroutine toUpper( buf )
    implicit none
    integer strlength, i
    character ( len=* ) :: buf

    strlength = len( buf )
    do i = 1, strlength
        if( ichar(buf(i:i))>=97 .and. ichar(buf(i:i))<=122 ) then
            buf(i:i) = char(ichar(buf(i:i))-32)
        end if
    end do

end subroutine
```

```
module dataio

    interface

        subroutine read_input_data( specf, spec, tm, eng, tc, ctrl )
            use hvtransfc
            character (len=*) :: specf
            type ( sspec ) :: spec
            type ( seng ) :: eng
            type ( stm ) :: tm
            type ( stc ) :: tc
            type ( sctr ) :: ctrl
        end subroutine

        subroutine read_vehicle_spec_data ( uid, specf, spec )
            use hvtransfc
            integer :: uid
            type ( sspec ) :: spec
            character (len=*) :: specf
        end subroutine

        subroutine read_transmission_spec_data( uid, tmf, tm )
            use hvtransfc
            integer :: uid
            character ( len=* ) :: tmf
            type ( stm ) :: tm
        end subroutine

        subroutine read_engine_fuel_map( uid, mapf, fcm )
            use hvtransfc
            integer :: uid
            type ( sfcmap ) :: fcm
            character ( len=* ) :: mapf
        end subroutine read_engine_fuel_map

        subroutine read_engine_spec_data( uid, engf, eng )
            use hvtransfc
            integer uid
            character ( len=* ) :: engf
            type ( seng ) :: eng
        end subroutine

        subroutine read_engine_torque_map( uid, mapf, gegmp )
            use hvtransfc
            integer :: uid
            character ( len=* ) :: mapf
            type ( sgridengmap ) :: gegmp
        end subroutine

        subroutine read_shift_control_data21 ( uid, ctrlrname, ngr, ctrl )
            use hvtransfc
            integer :: uid, ngr
            character ( len=* ) :: ctrlrname
            type ( sctr ) :: ctrl
       end subroutine

        subroutine read_shift_diagram( uid, mapf, ngr, sftmap )
            use hvtransfc
            character ( len=* ) :: mapf
            type ( sshiftmap ) :: sftmap
            integer :: uid, ngr
        end subroutine

        subroutine read_lockup_diagram( uid, mapf, ngr, sftmap )
            use hvtransfc
            character ( len=* ) :: mapf
            type ( sshiftmap ) :: sftmap
            integer :: uid, ngr
        end subroutine

        subroutine read_torque_converter_data ( uid, tcf, ngr, tc )
            use hvtransfc
            type ( stc ) :: tc
            character ( len=* ) :: tcf
            integer :: uid, ngr
        end subroutine

        subroutine readcurve ( uid, fname, crv )
            use hvtransfc
            integer :: uid
            character ( len=* ) :: fname
            type ( scrv ) :: crv
        end subroutine

        subroutine readpump ( uid, pumpf, ngr, ptq )
            use hvtransfc
            integer :: uid, ngr
            type ( sptq ) :: ptq
            character ( len=* ) :: pumpf
        end subroutine

        subroutine writeresultAT2 (uid, fname, vpat, vhx, comment)
            use hvtransfc
            integer :: uid
            character ( len=* ) :: fname
            type ( scycle ) :: vpat
```

```
            type ( vehout ), pointer :: vhx(:)
            character ( len=* ) :: comment
        end subroutine

        subroutine outaveragefcAT ( uid, outf, specf, spec, eu, avvu, terr1, eh, avvh, terr2, ece, et, avvt, terr3, iret, iret2, comment )
            use hvtransfc
            type ( sspec ) :: spec
            integer :: uid, iret, iret2, terr1, terr2, terr3
            character ( len=* ) :: outf, specf
            real(8) :: eu, avvu, eh, avvh, et, avvt, ece
            character (len=* ) :: comment
        end subroutine
    end interface

end module dataio



!*************************************************************
! SUBROUTINE GETARGUMENT 入力ファイル名, 出力ファイル名を取得
!  specf : 諸元ファイル名
!  outf  : 出力ファイル名
!  入力が無かった場合には手入力を指示.
!*************************************************************
subroutine GetArgument ( specf, outf )
    implicit none

    character ( len=* ) :: specf, outf
    call GETARG( 1, specf )
    call GETARG( 2, outf )

!   --- 指定が無い場合は手入力とする ---
    if ( specf == "" ) then
        do while ( specf == '' )
            write( *, '(A,$)' ) 'Input Spec filename? : '
            read ( *, * ) specf
        end do
    end if

    if ( outf == "" ) then
        do while ( outf == '' )
            write( *, '(A,$)' ) 'Output filename? : '
            read ( *, * ) outf
        end do
    end if

end subroutine



!*************************************************************
! SUBROUTINE READ_INPUT_DATA 入力データの読み込み制御
!  specf : 諸元ファイル名
!  spec  : 車両データ
!  tm    : 変速機データ
!  eng   : エンジンデータ
!  tc    : トルコンデータ
!  ctrl  : シフト制御データ
!*************************************************************
subroutine read_input_data( specf, spec, tm, eng, tc, ctrl )
    use hvtransfc
    use msubfc
    implicit none

    character ( len=* ) :: specf
    type ( sspec ) :: spec
    type ( seng ) :: eng
    type ( stm ) :: tm
    type ( stc ) :: tc
    type ( sctr ) :: ctrl

    character (len=1024) :: basedir, tmdir, engdir, tcdir, ctldir
    character (len=1024) :: basename, tmname, engname, tcname, ctlname
    character (len=1024) :: fname

!   --- 車両諸元定義ファイル ---------------------
    call read_vehicle_spec_data ( 11, specf, spec )
    call getdirectory( specf, basedir, basename )

!   --- 変速機諸元定義ファイル -------------------
    fname = trim(basedir) // trim( spec%tmf )
    call read_transmission_spec_data( 12, trim(fname), tm )
    call getdirectory( fname, tmdir, tmname )

!   --- エンジン諸元定義ファイル -----------------
    fname = trim(basedir) // trim( spec%engf )
    call getdirectory( fname, engdir, engname )
    call read_engine_spec_data( 13, trim(fname), eng )

!   --- 燃費マップ -------------------------------
    fname = trim(engdir) // trim( eng%fcmapf )
    call read_engine_fuel_map( 17, trim(fname), eng%fcm )

!   --- トルクマップ -----------------------------
    fname = trim(engdir) // trim( eng%egmapf )
```

```
    call read_engine_torque_map( 18, trim(fname), eng%gegmp )

!   --- 燃費マップの整形処理 ---------------------
    call fcgrid3( eng%nidle, eng%gfcm, eng%fcm, eng%fcmidl, eng%gegmp )

!   --- 変速制御定義ファイル ---------------------
    fname = trim(basedir) // trim( spec%ctrlf )
    call getdirectory( fname, ctldir, ctlname )
    call read_shift_control_data21 ( 19, trim(fname), tm%ngr, ctrl )

!   --- トルクコンバータ諸元データの読み込み -----
    fname = trim(basedir) // trim( spec%tcf )
    call getdirectory( fname, tcdir, tcname )
    call read_torque_converter_data ( 10, trim(fname), tm%ngr, tc )

end subroutine read_input_data




!*************************************************************
! SUBROUTINE READ_VEHICLE_SPEC_DATA 車両諸元データの読み込み
!  uid  : 装置番号
!  spcf : 車両諸元の入力ファイル名
!  spec : 車両諸元データ
!*************************************************************
subroutine read_vehicle_spec_data ( uid, specf, spec )
    use hvtransfc
    implicit none

    integer :: uid
    type ( sspec ) :: spec
    character (len=*) :: specf

    open ( uid, file=trim(specf), status='OLD', err=100 )

    read ( uid, *, err=110 ) spec%vhname                                ! 車両識別名
    read ( uid, *, err=110 ) spec%tpname                                ! 燃費区分

    read ( uid, *, err=110 ) spec%fgr                                   ! 終減速比
    read ( uid, *, err=110 ) spec%rt                                    ! タイヤ有効径

    read ( uid, *, err=110 ) spec%engf                                  ! エンジン諸元ファイル名
    read ( uid, *, err=110 ) spec%tmf                                   ! 変速機諸元ファイル名
    read ( uid, *, err=110 ) spec%tcf                                   ! トルコン諸元ファイル名
    read ( uid, *, err=110 ) spec%ctrlf                                 ! シフト制御設定ファイル名

    close( uid )
    return

!   -------------------------------------------------------
!   error handling
!   -------------------------------------------------------
100 write (0,'(3A)') " Cannot open SPEC file [ ", trim(specf), " ]"
    pause
    stop
110 write (0,'(3A)') " Failed to read SPEC file [ ", trim(specf), " ]"
    close ( uid )
    pause
    stop

end subroutine read_vehicle_spec_data




!*************************************************************
! SUBROUTINE READ_TRANSMISSION_SPEC_DATA 変速機データの読み込み
!  uid  : 装置番号
!  tmf  : 変速機諸元ファイル名
!  tm   : 変速機諸元格納変数
!*************************************************************
subroutine read_transmission_spec_data ( uid, tmf, tm )
    use hvtransfc
    implicit none

    integer :: uid, i
    character ( len=* ) :: tmf
    type ( stm ) :: tm

    open ( uid, file = trim(tmf), status = 'old', err=200 )

    read ( uid, *, err=210 ) tm%grs
    read ( uid, *, err=210 ) tm%ngr

    allocate ( tm%g(1:tm%ngr) )
    do i = 1, tm%ngr
        read ( uid, *, err=210 ) tm%g(i)%gr
    end do

    close( uid )
    return

!   -------------------------------------------------------
!   error handling
!   -------------------------------------------------------
```

```
200 write (0,'(3A)') " Cannot open TRANSMISSION file [ ", trim(tmf), " ]"
    pause
    stop
210 write (0,'(3A)') " Failed to read TRANSMISSION file [ ", trim(tmf), " ]"
    close( uid )
    pause
    stop

end subroutine read_transmission_spec_data



!*************************************************************
! SUBROUTINE READ_ENGINE_SPEC_DATA エンジン諸元の読み込み
!  uid  : 装置番号
!  engf : エンジン諸元の入力ファイル名
!  eng  : エンジン諸元データ
!*************************************************************
subroutine read_engine_spec_data( uid, engf, eng )
    use hvtransfc
    implicit none

    integer uid
    character ( len=* ) :: engf
    type ( seng ) :: eng

    open ( uid, file=trim( engf ), status='old', err=300 )

    read ( uid, *, err=310 ) eng%fcmapf
    read ( uid, *, err=310 ) eng%egmapf
    read ( uid, *, err=310 ) eng%nidle
    read ( uid, *, err=310 ) eng%nrate
    read ( uid, *, err=310 ) eng%nex

    close ( uid )
    return

!   -------------------------------------------------------
!   error handling
!   -------------------------------------------------------
300 write (0,'(3A)') " Cannot open ENGINE file [ ", trim(engf), " ]"
    pause
    stop
310 write (0,'(3A)') " Failed to read ENGINE file [ ", trim(engf), " ]"
    close ( uid )
    pause
    stop
end subroutine



!*************************************************************
! SUBROUTINE READ_ENGINE_FUEL_MAP 燃費マップの読み込み
!  uid  : 装置番号
!  mapf : 燃費マップファイル名
!  fcm  : 燃費マップ
!*************************************************************
subroutine read_engine_fuel_map( uid, mapf, fcm )
    use hvtransfc
    implicit none

    type ( sfcmap ) :: fcm
    character ( len=* ) :: mapf
    character ( len=1024 ) :: tmp
    real(8) :: r, t, f
    integer :: uid, gap, i, j, k, ios, head, ct

!   -----------------
!    データ数カウント
!   -----------------
    fcm%ndata = 0
    open( uid, file = mapf, status = 'old', err=400 )

    read ( uid, *, err=410 ) tmp
    read ( uid, *, err=410 ) tmp

    ios = 0
    do while ( ios == 0 )
        read( uid, *, iostat=ios, err=410 ) r, t, f
        if( ios == 0 ) fcm%ndata = fcm%ndata + 1
    end do

    rewind( uid )
    allocate( fcm%rev(1:fcm%ndata) )
    allocate( fcm%tq(1:fcm%ndata) )
    allocate( fcm%fc(1:fcm%ndata) )

!   -------------------
!    燃費マップ読み込み
!   -------------------
    read ( uid, *, err=410 ) tmp
    read ( uid, *, err=410 ) tmp

    do i = 1, fcm%ndata
```

```
        read ( uid, *, err=410 ) fcm%rev(i), fcm%tq(i), fcm%fc(i)
    end do

    close( uid )

!   ------------------------------------
!    エンジン回転数を基準とした昇順ソート
!   ------------------------------------
    gap = ( fcm%ndata + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap+1, fcm%ndata
            do j = i-gap, 1, -gap
                if ( fcm%rev(j) <= fcm%rev(j+gap) ) exit
                r              = fcm%rev(j)
                fcm%rev(j)     = fcm%rev(j+gap)
                fcm%rev(j+gap) = r
                t              = fcm%tq(j)
                fcm%tq(j)      = fcm%tq(j+gap)
                fcm%tq(j+gap)  = t
                f              = fcm%fc(j)
                fcm%fc(j)      = fcm%fc(j+gap)
                fcm%fc(j+gap)  = f
            end do
        end do
        gap = gap / 2
    end do

!   ---------------------------------------
!    エンジン回転数ごとにトルクで昇順ソート
!   ---------------------------------------
    head = 1
    r = fcm%rev( head )
    ct = 0
    do i = 1, fcm%ndata
        ct = ct + 1
        if ( r==fcm%rev(i) .and. i<fcm%ndata ) cycle

        if ( r /= fcm%rev(i) ) ct = ct -1
        gap = ( ct + 1 ) / 2
        do while ( gap >= 1 )
            do k = head + gap - 1, head + ct - 1
                do j = k - gap, head, -gap
                    if ( fcm%tq(j) <= fcm%tq(j+gap) ) exit
                    r              = fcm%rev(j)
                    fcm%rev(j)     = fcm%rev(j+gap)
                    fcm%rev(j+gap) = r
                    t              = fcm%tq(j)
                    fcm%tq(j)      = fcm%tq(j+gap)
                    fcm%tq(j+gap)  = t
                    f              = fcm%fc(j)
                    fcm%fc(j)      = fcm%fc(j+gap)
                    fcm%fc(j+gap)  = f
                end do
            end do
            gap = gap / 2
        end do

!       -----------------
!        次の回転数へ移行
!       -----------------
        r = fcm%rev(i)
        ct = 1
        head = i

    end do
    return

!   ------------------------------------------------------------
!   error handling
!   ------------------------------------------------------------
400 write (0,'(3A)') " Cannot open FUEL MAP file [ ", trim(mapf), " ]"
    pause
    stop
410 write (0,'(3A)') " Failed to read FUEL MAP [ ", trim(mapf), " ]"
    close ( uid )
    pause
    stop
end subroutine read_engine_fuel_map




!*************************************************************
! SUBROUTINE READ_ENGINE_TORQUE_MAP トルクマップ読み込み
!  uid     : 装置番号
!  mapf    : エンジントルクマップのファイル名
!  gegmp   : エンジントルクマップ格納用変数
!*************************************************************
subroutine read_engine_torque_map( uid, mapf, gegmp )
    use hvtransfc
    implicit none

    integer :: uid
    character (len=*) :: mapf
    type ( sgridengmap ) :: gegmp
    character (len=1024) :: tmp
```

```
    real(8), allocatable :: r(:), t(:), ta(:)
    real(8) :: rr, tt, ttaa, thr
    integer :: gap, i, j, k, n, ios, head, ct

!   -----------------
!    データ数カウント
!   -----------------
    n = 0
    open ( uid, file = mapf, status = 'old', err=500 )
    read ( uid, *, err=510 ) tmp

    ios = 0
    do while ( ios == 0 )
        read ( uid, *, iostat = ios, err=510 ) ttaa, rr, tt
        if ( ios == 0 ) n = n + 1
    end do

    rewind( uid )
    allocate( r(1:n), t(1:n), ta(1:n) )

!   -------------
!    本番読み込み
!   -------------
    read ( uid, *, err=510 ) tmp

    do i = 1, n
        read ( uid, *, err=510 ) ta(i), r(i), t(i)
    end do

    close ( uid )

!   ------------------------------------
!    アクセル開度を基準とした昇順ソート
!   ------------------------------------
    gap = ( n + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap + 1, n
            do j = i - gap, 1, -gap
                if ( ta(j) <= ta(j+gap) ) exit
                rr        = r(j)
                r(j)      = r(j+gap)
                r(j+gap)  = rr
                tt        = t(j)
                t(j)      = t(j+gap)
                t(j+gap)  = tt
                ttaa      = ta(j)
                ta(j)     = ta(j+gap)
                ta(j+gap) = ttaa
            end do
        end do
        gap = gap / 2
    end do

!   ------------------------------------
!    アクセル開度ごとに回転数で昇順ソート
!   ------------------------------------
    gegmp%tqn = 0
    gegmp%thrn = 0
    head = 1
    thr = ta( head )
    ct = 0

    do i = 1, n
        ct = ct + 1
        if ( i < n .and. thr == ta(i) ) cycle

!       ---------------------------------------------
!       同一開度を抽出したら回転数を基準に並べなおす
!       ---------------------------------------------
        if ( thr /= ta(i) ) ct = ct -1
        gap = ( ct + 1 ) / 2
        do while ( gap >= 1 )
            do k = head+gap-1, head+ct-1
                do j = k-gap, head, -gap
                    if ( r(j) <= r(j+gap) ) exit
                    rr        = r(j)
                    r(j)      = r(j+gap)
                    r(j+gap)  = rr
                    tt        = t(j)
                    t(j)      = t(j+gap)
                    t(j+gap)  = tt
                    ttaa      = ta(j)
                    ta(j)     = ta(j+gap)
                    ta(j+gap) = ttaa
                end do
            end do
            gap = gap / 2
        end do
        gegmp%thrn = gegmp%thrn + 1
        gegmp%tqn = max ( gegmp%tqn, ct )

!       ---------------
!        次の開度へ移行
!       ---------------
        thr = ta(i)
        ct = 1
```

```
        head = i

    end do

!   ---------------------------------------
!    エンジントルクマップ配列へのデータ格納
!   ---------------------------------------
    allocate ( gegmp%thr(1:gegmp%thrn), gegmp%n(1:gegmp%thrn) )
    allocate ( gegmp%r(1:gegmp%thrn, 1:gegmp%tqn) )
    allocate ( gegmp%t(1:gegmp%thrn, 1:gegmp%tqn) )

    j = 1
    gegmp%thr(j) = ta(1)
    gegmp%n(j) = 0
    do i = 1, n
        if ( gegmp%thr(j) /= ta(i) ) then
            j = j + 1
            gegmp%thr(j) = ta(i)
            gegmp%n(j) = 0
        end if
        gegmp%n(j) = gegmp%n(j) + 1
        gegmp%r( j, gegmp%n(j) ) = r(i)
        gegmp%t( j, gegmp%n(j) ) = t(i)
    end do

    deallocate ( r, t, ta )
    return

!   ---------------------------------------------------
!   error handling
!   ---------------------------------------------------
500 write (0,'(3A)') " Cannot open ENGINE TORQUE MAP file [ ", trim(mapf), " ]"
    pause
    stop
510 write (0,'(3A)') " Failed to read ENGINE TORQUE MAP [ ", trim(mapf), " ]"
    close ( uid )
    pause
    stop

end subroutine read_engine_torque_map







!*************************************************************
! SUBROUTINE READ_SHIFT_DIAGRAM 変速マップの読み込み
!  uid     : 装置番号
!  mapf    : シフトマップデータのファイル名
!  ngr     : ギヤ段数
!  sftmap  : シフトマップデータの格納用変数
!   %g(i)%x  : アクセル開度 (%)
!   %g(i)%y  : 速度 (km/h)
!   %lu(i)   : 変速後のロックアップ初期値(0:OFF,2:ON)
!*************************************************************
subroutine read_shift_diagram ( uid, mapf, ngr, sftmap )
    use hvtransfc
    use dataio
    implicit none

    character (len=*) :: mapf
    type (sshiftmap) :: sftmap
    integer :: uid, ngr
    character (len=1024) :: tmp
    real(8) :: thrt, v
    integer :: g, gap, i, j, k, ios, head, ct, n, l
    integer :: err0
    real(8), pointer :: thr(:), vv(:)
    integer, pointer :: gg(:), lu(:)

!   --------------
!    変数の初期化
!   --------------
    sftmap%ngr = ngr
    allocate( sftmap%g(1:ngr) )

    do i = 1, ngr
        sftmap%g(i)%n = 0
    end do

    if ( mapf == "" ) return

!   ------------------
!    データ数カウント
!   ------------------
    open ( uid, file=mapf, status='old', err=700 )

    n = 0
    read ( uid, *, err=710 ) tmp

    ios = 0
    do while ( ios == 0 )
        read( uid, *, iostat=ios, err=710 ) g, thrt, v, l
        if( ios==0 ) n = n + 1
```

```
    end do

    if ( n == 0 ) then
        close ( uid )
        return
    else
        allocate( gg(1:n), thr(1:n), vv(1:n), lu(1:n) )
    end if

    rewind( uid )

!   ---------------------------------------
!    ギヤ位置, 開度, 出力軸回転数の取り込み
!   ---------------------------------------
    read ( uid, *, err=710 ) tmp

    do i = 1, n
        read ( uid, *, err=710 ) gg(i), thr(i), vv(i), lu(i)
    end do

    close( uid )

!   ---------------------------
!    ギヤ位置を基準に昇順ソート
!   ---------------------------
    gap = ( n + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap + 1, n
            do j = i - gap, 1, -gap
                if ( gg(j) <= gg(j+gap) ) exit
                g         = gg(j)
                gg(j)     = gg(j+gap)
                gg(j+gap) = g
                thrt      = thr(j)
                thr(j)    = thr(j+gap)
                thr(j+gap) = thrt
                v         = vv(j)
                vv(j)     = vv(j+gap)
                vv(j+gap) = v
                l         = lu(j)
                lu(j)     = lu(j+gap)
                lu(j+gap) = l
            end do
        end do
        gap = gap / 2
    end do

!   -----------------------------
!    ギヤ段毎に開度での昇順ソート
!   -----------------------------
    head = 1
    g = gg( head )
    ct = 0
    do i = 1, n

        if( gg(i) <= 0 .or. gg(i) > ngr ) then
            write (0,'(A)')  "【入力データエラー】"
            write (0,'(A)')  "変速マップデータのギヤ段設定が正しくありません... "
            write (0,'(2A)')  "File = ", trim(mapf)
            write (0,'(A,i0)') "データ行 = ", i
            write (0,'(A,i0)') "ギヤ入力値 = ",gg(i)
            stop
        end if

        sftmap%g(gg(i))%n = sftmap%g(gg(i))%n + 1
        ct = ct + 1
        if ( g==gg(i) .and. i < n ) cycle

!       ------------------------------------------------
!        同一ギヤ位置のデータ範囲を開度基準に昇順ソート
!       ------------------------------------------------
        if ( g /= gg(i) ) ct = ct -1
        gap = ( ct + 1 ) / 2
        do while ( gap >= 1 )
            do k = head + gap -1, head + ct -1
                do j = k - gap, head, -gap
                    if ( thr(j) <= thr(j+gap) ) exit
                    thrt       = thr(j)
                    thr(j)     = thr(j+gap)
                    thr(j+gap) = thrt
                    v          = vv(j)
                    vv(j)      = vv(j+gap)
                    vv(j+gap)  = v
                    g          = gg(j)
                    gg(j)      = gg(j+gap)
                    gg(j+gap)  = g
                    l          = lu(j)
                    lu(j)      = lu(j+gap)
                    lu(j+gap)  = l
                end do
            end do
            gap = gap / 2
        end do
        g = gg(i)
        ct = 1
        head = i
```

```
    end do

!   ---------------------
!    ソート結果を格納する
!   ---------------------
    err0 = 0
    i = 1
    do j = 1, ngr
        if ( sftmap%g(j)%n <= 0 ) cycle

!        変速線, LU制御線の格納配列
        allocate ( sftmap%g(j)%x( 1:sftmap%g(j)%n ), sftmap%g(j)%y( 1:sftmap%g(j)%n ) )
        allocate ( sftmap%g(j)%l( 1:sftmap%g(j)%n ) )

        do k = 1, sftmap%g(j)%n
            sftmap%g( j )%x( k ) = thr(i)
            sftmap%g( j )%y( k ) = vv(i)
            sftmap%g( j )%l( k ) = lu(i)
            i = i + 1
        end do

        if ( sftmap%g(j)%n == 1 ) then
            write (0,'(A)')  "【入力データエラー】"
            write (0,'(A)')  "変速マップにはギヤ毎2点以上のデータが必要です. "
            write(0,'(2A)')   "データ : ", trim(mapf)
            write(0,'(A,i0)') "ギヤ段 : ", j
            err0 = 1
        end if
    end do

!   ---------------------------
!    変速線にエラーがあった場合
!   ---------------------------
    if ( err0 == 1 ) then
        write(0,'(a)') "=== DATA ERROR ==="
        do i = 1, n
            write(0,'(i0,f10.3,f10.3)') gg(i), thr(i), vv(i), lu(i)
        end do
        deallocate ( gg, thr, vv, lu )
        stop
    end if

    deallocate ( gg, thr, vv, lu )
    return

!   ------------------------------------------------------------
!   error handling
!   ------------------------------------------------------------
700 write (0,'(3A)') " Cannot open gear-shift diagram file [ ", trim(mapf), " ]"
    pause
    stop
710 write (0,'(3A)') " Failed to read gear-shift diagram file [ ", trim(mapf), " ]"
    close ( uid )
    pause
    stop

end subroutine read_shift_diagram




!*************************************************************
! SUBROUTINE READ_LOCKUP_DIAGRAM ロックアップマップの読み込み
!  uid     : 装置番号
!  mapf    : シフトマップデータのファイル名
!  ngr     : ギヤ段数
!  sftmap  : シフトマップデータの格納用変数
!   %g(i)%x  : アクセル開度 (%)
!   %g(i)%y  : 速度 (km/h)
!*************************************************************
subroutine read_lockup_diagram ( uid, mapf, ngr, sftmap )
    use hvtransfc
    use dataio
    implicit none

    character (len=*) :: mapf
    type (sshiftmap) :: sftmap
    integer :: uid, ngr
    character (len=1024) :: tmp
    real(8) :: thrt, v
    integer :: g, gap, i, j, k, ios, head, ct, n, err0
    real(8), pointer :: thr(:), vv(:)
    integer, pointer :: gg(:)

!   --------------
!    変数の初期化
!   --------------
    sftmap%ngr = ngr
    allocate( sftmap%g(1:ngr) )

    do i = 1, ngr
        sftmap%g(i)%n = 0
    end do

    if ( mapf == "" ) return
```

```
!   ---------------
!    データ読み込み
!   ---------------
    open ( uid, file=mapf, status='old', err=750 )

    n = 0
    read ( uid, *, err=760 ) tmp

    ios = 0
    do while ( ios == 0 )
        read( uid, *, iostat=ios, err=760 ) g, thrt, v
        if( ios==0 ) n = n + 1
    end do

    if ( n == 0 ) then
        close ( uid )
        return
    else
        allocate( gg(1:n), thr(1:n), vv(1:n) )
    end if

    rewind( uid )

!   ---------------------------------------
!    ギヤ位置, 開度, 出力軸回転数の読み込み
!   ---------------------------------------
    read ( uid, *, err=760 ) tmp

    do i = 1, n
        read ( uid, *, err=760 ) gg(i), thr(i), vv(i)
    end do

    close( uid )

!   ----------------------------
!    ギヤ位置を基準に昇順ソート
!   ----------------------------
    gap = ( n + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap + 1, n
            do j = i - gap, 1, -gap
                if ( gg(j) <= gg(j+gap) ) exit
                g         = gg(j)
                gg(j)     = gg(j+gap)
                gg(j+gap) = g
                thrt      = thr(j)
                thr(j)    = thr(j+gap)
                thr(j+gap) = thrt
                v         = vv(j)
                vv(j)     = vv(j+gap)
                vv(j+gap) = v
            end do
        end do
        gap = gap / 2
    end do

!   -------------------------------
!    ギヤ段ごとに開度での昇順ソート
!   -------------------------------
    head = 1
    g = gg( head )
    ct = 0
    do i = 1, n

        if( gg(i) <= 0 .or. gg(i) > ngr ) then
            write (0,'(A)')  "【入力データエラー】"
            write (0,'(A)')  "L/Uマップデータのギヤ段設定が正しくありません... "
            write (0,'(2A)')  "File = ", trim(mapf)
            write (0,'(A,i0)') "データ行 = ", i
            write (0,'(A,i0)') "ギヤ入力値 = ",gg(i)
            stop
        end if

        sftmap%g(gg(i))%n = sftmap%g(gg(i))%n + 1
        ct = ct + 1
        if ( g==gg(i) .and. i < n ) cycle

        if ( g /= gg(i) ) ct = ct -1
        gap = ( ct + 1 ) / 2
        do while ( gap >= 1 )
            do k = head + gap -1, head + ct -1
                do j = k - gap, head, -gap
                    if ( thr(j) <= thr(j+gap) ) exit
                    thrt       = thr(j)
                    thr(j)     = thr(j+gap)
                    thr(j+gap) = thrt
                    v          = vv(j)
                    vv(j)      = vv(j+gap)
                    vv(j+gap)  = v
                    g          = gg(j)
                    gg(j)      = gg(j+gap)
                    gg(j+gap)  = g
                end do
            end do
            gap = gap / 2
```

```
        end do
        g = gg(i)
        ct = 1
        head = i
    end do

!   -----------------
!    ソート結果を格納
!   -----------------
    err0 = 0
    i = 1
    do j = 1, ngr
        if ( sftmap%g(j)%n <= 0 ) cycle

        allocate ( sftmap%g(j)%x( 1:sftmap%g(j)%n ) )
        allocate ( sftmap%g(j)%y( 1:sftmap%g(j)%n ) )

        do k = 1, sftmap%g(j)%n
            sftmap%g( j )%x( k ) = thr(i)
            sftmap%g( j )%y( k ) = vv(i)
            i = i + 1
        end do

        if ( sftmap%g(j)%n == 1 ) then
            write (0,'(A)')  "【入力データエラー】"
            write (0,'(A)')  "ロックアップマップにはギヤ毎2点以上のデータが必要です. "
            write(0,'(2A)')   "データ : ", trim(mapf)
            write(0,'(A,i0)') "ギヤ段 : ", j
            err0 = 1
        end if
    end do

!   --------------------------
!    L/U線にエラーがあった場合
!   --------------------------
    if ( err0 == 1 ) then
        write(0,'(a)') "=== DATA ERROR ==="
        do i = 1, n
            write(0,'(i0,f10.3,f10.3)') gg(i), thr(i), vv(i)
        end do
        deallocate ( gg, thr, vv )
        stop
    end if

    deallocate ( gg, thr, vv )
    return

!   ------------------------------------------------------------
!   error handling
!   ------------------------------------------------------------
750 write (0,'(3A)') " Cannot open gear-shift diagram file [ ", trim(mapf), " ]"
    pause
    stop
760 write (0,'(3A)') " Failed to read gear-shift diagram file [ ", trim(mapf), " ]"
    close ( uid )
    pause
    stop

end subroutine read_lockup_diagram



!*************************************************************
! SUBROUTINE READ_TORQUE_CONVERTER_DATA トルコン諸元の読み込み
!  uid     : 装置番号
!  tcf     : トルコンデータのファイル名
!  ngr     : ギヤ段数
!  tc      : トルコンデータの格納用変数
!*************************************************************
subroutine read_torque_converter_data ( uid, tcf, ngr, tc )
    use hvtransfc
    implicit none

    type (stc) :: tc
    character (len=*) :: tcf
    integer :: uid, ngr
    character (len=1024) :: basedir, basename, fname

!   ----------------------------------
!    トルコン諸元データのファイル名生成
!   ----------------------------------
    call getdirectory( tcf, basedir, basename )
    tc%tcf = trim(tcf)

!   ------------------
!    データの読み込み
!   ------------------
    open ( uid, file=trim(tcf), status='old', err=800 )

    read ( uid, *, err=810 ) tc%cfdname                                  ! 容量係数 (駆動) ファイル名
    read ( uid, *, err=810 ) tc%cfbname                                  ! 容量係数 (逆駆動) ファイル名
    read ( uid, *, err=810 ) tc%trdname                                  ! トルク比 (駆動) ファイル名
    read ( uid, *, err=810 ) tc%trbname                                  ! トルク比 (逆駆動) ファイル名
    read ( uid, *, err=810 ) tc%ptqname                                  ! オイルポンプ特性ファイル名
```

```
    close ( uid )


!   -- トルク比(駆動)データの読み込み
    fname = trim(basedir) // trim(tc%trdname)
    call readcurve ( uid, trim(fname), tc%trd )

!   -- トルク比(逆駆動)データの読み込み
    fname = trim(basedir) // trim(tc%trbname)
    call readcurve ( uid, trim(fname), tc%trb )

!   -- 容量係数(駆動)データの読み込み --
    fname = trim(basedir) // trim(tc%cfdname)
    call readcurve ( uid, trim(fname), tc%cfd )

!   -- 容量係数(逆駆動)データの読み込み --
    fname = trim(basedir) // trim(tc%cfbname)
    call readcurve ( uid, trim(fname), tc%cfb )

!   -- オイルポンプ特性データの読み込み --
    fname = trim(basedir) // trim(tc%ptqname)
    call readpump ( uid, trim(fname), ngr, tc%ptq )
    return

! <<< error handling >>>
800 write (0,'(3A)') " Cannot open TORQUE CONVERTER file [ ", trim(tcf), " ]"
    pause
    stop
810 write (0,'(3A)') " Failed to read TORQUE CONVERTER file [ ", trim(tcf), " ]"
    close ( uid )
    pause
    stop

end subroutine read_torque_converter_data




!*************************************************************
! SUBROUTINE READCURVE X-Y曲線データの読み込み
!  uid    : 装置番号
!  fname  : ファイル名
!  crv    : 2D曲線格納変数
!*************************************************************
subroutine readcurve ( uid, fname, crv )
    use hvtransfc
    implicit none

    type ( scrv ) :: crv
    character (len=*) :: fname
    integer :: uid, i, j, gap, ios
    character (len=1024) :: tmp
    real(8) :: xx, yy

!   ------------------------------
!    データ点数カウント & 読み込み
!   ------------------------------
    crv%n = 0
    open ( uid, file = fname, status = 'old', err=900 )
    read ( uid,'(A)' ) tmp

    ios = 0
    do while ( ios == 0 )
        read ( uid, *, iostat = ios, err=910 ) xx, yy
        if( ios == 0 ) crv%n = crv%n + 1
    end do
    allocate ( crv%x(1:crv%n), crv%y(1:crv%n) )

    rewind ( uid )
    read ( uid,'(A)' ) tmp
    do i = 1, crv%n
        read ( uid, *, err=910 ) crv%x(i), crv%y(i)
    end do

    close ( uid )

!   ---------------------------
!    X軸データを基準に昇順ソート
!   ---------------------------
    gap = ( crv%n + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap +1, crv%n
            do j = i - gap, 1, -gap
                if ( crv%x(j) <= crv%x(j+gap) ) exit
                xx           = crv%x(j)
                crv%x(j)     = crv%x(j+gap)
                crv%x(j+gap) = xx
                yy           = crv%y(j)
                crv%y(j)     = crv%y(j+gap)
                crv%y(j+gap) = yy
            end do
        end do
        gap = gap / 2
    end do

    return
```

```
! <<< error handling >>>
900 write (0,'(3A)') " Cannot open curve data file [ ", trim(fname), " ]"
    pause
    stop
910 write (0,'(A,i0,3A)') " Failed to read curve data. Data No.= ",    &
                          crv%n+1, " in [ ", trim( fname ), " ]"
    close ( uid )
    pause
    stop

end subroutine readcurve




!*************************************************************
! SUBROUTINE READPUMP オイルポンプ損失トルクのデータ読み込み
!  uid     : 装置番号
!  pumpf   : オイルポンプロストルクのファイル名
!  ngr     : ギヤ段数
!  ptq     : オイルポンプ特性格納変数
!*************************************************************
subroutine readpump ( uid, pumpf, ngr, ptq )
    use hvtransfc
    implicit none

    character (len=*) :: pumpf
    type (sptq) :: ptq
    integer :: uid, ngr

    character (len=1024) :: tmp
    real(8) :: r, t
    integer :: g
    real(8), pointer :: rr(:), tt(:)
    integer, pointer :: gg(:)
    integer :: gap, i, j, k, ios, head, n, ct

!   ------------------
!    データ数カウント
!   ------------------
    open ( uid, file=pumpf, status='old', err=1000 )
    read ( uid, *, err=1010 ) tmp

    n = 0
    ios = 0
    do while ( ios==0 )
        read( uid, *, iostat=ios, err=1010 ) g, r, t
        if( ios==0 ) n = n + 1
    end do

    allocate( gg(1:n), rr(1:n), tt(1:n) )

    rewind( uid )

!   ----------------
!    データ読み込み
!   ----------------
    read ( uid, *, err=1010 ) tmp
    do i = 1, n
        read ( uid, *, err=1010 ) gg(i), rr(i), tt(i)
    end do
    close( uid )

!   ------------
!    データ整形
!   ------------
    ptq%ngr = ngr
    allocate( ptq%g(1:ngr) )

    do i = 1, ngr
        ptq%g(i)%n = 0
    end do

!   -------------------------
!    ギヤ段を基準に昇順ソート
!   -------------------------
    gap = ( n + 1 ) / 2
    do while ( gap >= 1 )
        do i = gap + 1, n
            do j = i - gap, 1, -gap
                if ( gg(j) <= gg(j+gap) ) exit
                g         = gg(j)
                gg(j)     = gg(j+gap)
                gg(j+gap) = g
                r         = rr(j)
                rr(j)     = rr(j+gap)
                rr(j+gap) = r
                t         = tt(j)
                tt(j)     = tt(j+gap)
                tt(j+gap) = t
            end do
        end do
        gap = gap / 2
    end do
```

```
!   ------------------------------
!    ギヤ毎にエンジン回転数でソート
!   ------------------------------
    head = 1
    g = gg( head )
    ct = 0

    do i = 1, n

        if( gg(i) <= 0 .or. gg(i) > ngr ) then
            print*, "【入力データエラー】"
            print*, "オイルポンプデータのギヤ段設定が正しくありません... "
            print*, "File = ", trim(pumpf)
            print*, "データ行 = ", i
            print*, "ギヤ入力値 = ",gg(i)
            stop
        end if

        ptq%g( gg(i) )%n = ptq%g( gg(i) )%n + 1
        ct = ct + 1
        if ( g==gg(i) .and. i<n ) cycle

        if ( g /= gg(i) ) ct = ct - 1
        gap = ( ct + 1 ) / 2
        do while ( gap >= 1 )
            do k = head + gap -1, head + ct -1
                do j = k -gap, head, -gap
                    if ( rr(j) <= rr(j+gap) ) exit
                    r         = rr(j)
                    rr(j)     = rr(j+gap)
                    rr(j+gap) = r
                    t         = tt(j)
                    tt(j)     = tt(j+gap)
                    tt(j+gap) = t
                    g         = gg(j)
                    gg(j)     = gg(j+gap)
                    gg(j+gap) = g
                end do
            end do
            gap = gap / 2
        end do
        g = gg(i)
        ct = 1
        head = i

    end do

!   ------------------------
!    オイルポンプ特性の記憶
!   ------------------------
    i = 1
    do j = 1, ngr

        allocate ( ptq%g(j)%x(1:ptq%g(j)%n), ptq%g(j)%y(1:ptq%g(j)%n) )

        do k = 1, ptq%g(j)%n
            ptq%g( j )%x( k ) = rr(i)
            ptq%g( j )%y( k ) = tt(i)
            i = i + 1
        end do

    end do

    deallocate ( gg, tt, rr )
    return

! <<< error handling >>>
1000 write (0,'(3A)') " オイルポンプデータがオープンできません. [ ", trim(pumpf), " ]"
    pause
    stop
1010 write (0,'(3A)') " オイルポンプデータの読み込みに失敗しました. [ ", trim(pumpf), " ]"
    close ( uid )
    pause
    stop

end subroutine




! ***************************************************************
!  SUBROUTINE GETDIRECTORY
! ***************************************************************
subroutine getdirectory( pathname, retpath, retname )
    implicit none
    character(len=*) :: pathname, retpath, retname
    integer :: cpos1, cpos2, cpos

    cpos1 = scan(pathname, "/", back = .true.)
    cpos2 = scan(pathname, "¥", back = .true.)
    cpos = max(cpos1, cpos2)

    if((cpos1/=0) .or. (cpos2/=0))then
        retpath = pathname(1:cpos)
        retname = pathname(cpos+1:len_trim(pathname))
    else
```

```
        retpath = ""
        retname = pathname
    end if

end subroutine



!************************************************************
!  SUBROUTINE OUTAVERAGEFCAT 燃費結果のファイル出力
! uid     : 装置番号
! outf    : 出力ファイル名
! specf   : 車両諸元ファイル名
! spec    : 車両諸元データ
! eu      : 都市内モード燃費計算値(km/L)
! avvu    : 都市内モード平均速度(km/h)
! eh      : 高速モード燃費計算値(km/L)
! ece     : 都市内＋都市間の燃費値(km/L)
! et      ; 市街地燃費 (km/L)
! avvt    : 市街地平均速度  (km/h)
! avvh    : 高速モード平均速度(km/h)
! iret, iret2  : 燃費計算結果判定 (市街地&高速モード, 正常終了時0, エラー発生時0以外)
! terr1, terr2, terr3 : 都市内, 都市間, 市街地の追従不能秒数
! comment : コメント文字列
!************************************************************
subroutine outaveragefcAT ( uid, outf, specf, spec, eu, avvu, terr1, eh, avvh, terr2, ece, et, avvt, terr3, iret, iret2, comment )
    use hvtransfc
    implicit none

    character, parameter :: ht = char(9)
    type ( sspec ) :: spec
    integer :: uid, iret, iret2, terr1, terr2, terr3
    character ( len=* ) :: outf, specf, comment
    real(8) :: eu, avvu, eh, avvh, et, avvt, ece

    open ( uid, file = outf, status = 'unknown' )

    write ( uid, '(A)' ) trim( comment )
    write ( uid, '(3A)' ) "VEHICLENAME", ht, trim(spec%vhname)
    write ( uid, '(3A)' ) "TYPE", ht, trim(spec%tpname)
    write ( uid, * )
    write ( uid, '(3A)' ) "SPEC FILE", ht, trim(specf)
    write ( uid, '(3A)' ) "ENGINE FILE", ht, trim(spec%engf)
    write ( uid, '(3A)' ) "TRANSMISSION FILE", ht, trim(spec%tmf)
    write ( uid, '(3A)' ) "TORQUE_CONVERTER FILE", ht, trim(spec%tcf)
    write ( uid, '(3A)' ) "SHIFT_CONTROL FILE", ht, trim(spec%ctrlf)
    write ( uid, * )
    write ( uid, '(2A,F8.5)' ) "FINAL GEAR RATIO", ht, spec%fgr
    write ( uid, '(2A,F8.5)' ) "TIRE RADIUS(m)",     ht, spec%rt
    write ( uid, * )

    if ( iret == 0 ) then
        write ( uid, '(4A,F8.4,3A,F5.1,3A,i0)' ) &
                "【URBAN】     ",ht,"FC(km/l)", ht, eu, ht, "Ave.Speed(km/h)", ht, avvu, ht, &
                '追従不能時間(S)', ht, terr1
    else
        write ( uid, '(A)' ) "【URBAN】エラーにより途中終了しました. "
    end if

    if ( iret2 == 0 ) then
        write ( uid, '(4A,F8.4,3A,F5.1,3A,i0)' )  &
                "【HIGHWAY】   ", ht, "FC(km/l)", ht, eh, ht, "Ave.Speed(km/h)", ht, avvh, ht, &
                '追従不能時間(S)', ht, terr2
    else
        write ( uid, '(A)' ) "【HIGHWAY】エラーにより途中終了しました. "
    end if

    if ( iret == 0 .and. iret2 == 0 ) then
        write ( uid, '(4A,F8.4,3A,F5.2)' )  &
                "【AVERAGE】   ", ht, "FC(km/l)", ht, ece, ht, "HIGHWAY RATIO", ht, spec%ric
        write ( uid, '(4A,F8.4,3A,F5.1,3A,i0)' ) &
                "【MID-TOWN】  ", ht, "FC(km/l)", ht, et, ht, "Ave.Speed(km/h)", ht, avvt, ht, &
                '追従不能時間(S)', ht, terr3
    else
        write ( uid, '(A)' ) "【AVERAGE】エラーにより途中終了しました. "
        write ( uid, '(A)' ) "【MID-TOWN】エラーにより途中終了しました. "
   end if

    close ( uid )

end subroutine outaveragefcAT



!************************************************
! SUBROUTINE WRITERESULTAT 燃費計算結果の書き出し
!  UID 装置番号
!  FNAME    : 出力ファイル名
!  VPAT     : 速度パタン
!  VHX      : 車両運転状態データ
!  COMMENT  : コメント
!************************************************
subroutine writeresultAT2 ( uid, fname, vpat, vhx, comment )
    use hvtransfc
    implicit none
```

```
    character, parameter :: ht = char(9)
    integer :: i, uid
    character ( len=* ) :: fname, comment
    type ( scycle ) :: vpat
    type ( vehout ), pointer :: vhx(:)

    open ( uid, file = fname, status = 'unknown', position = 'APPEND' )

    write ( uid, * )
    write ( uid, '(A)' ) trim(comment)

    write ( uid, '(35A)' ) &
            '時間(s)', ht, '目標速度(km/h)', ht, '計算速度(km/h)', ht, 'エンジン回転数(rpm)', ht,           &
            'エンジントルク(Nm)', ht, '正規化回転数(%)', ht, '正規化トルク(%)', ht,'出力軸回転数(rpm)', ht, &
            'オイルポンプトルク(Nm)', ht, 'トルコン入力軸トルク(Nm)', ht, 'トルコン出力軸トルク(Nm)', ht,   &
            '速度比', ht, 'トルク比', ht, 'ギヤ', ht,'ロックアップ(0:Off,2:On)', ht, '開度(%)', ht,         &
            '燃料消費量(L/h)', ht, '追従不能(1)'

    do i = 1, vpat%ndat
        write ( uid, '( i0,2(a,f0.2),5(a,f0.1),3(a,f0.2),2(a,f0.4),2(a,i0),(a,f0.3),(a,f0.6),(a,i0) )' )    &
            i, ht, vpat%v(i), ht, vhx(i)%v, ht, vhx(i)%ne, ht, vhx(i)%te, ht, vhx(i)%nne, ht,               &
            vhx(i)%nte, ht, vhx(i)%npera, ht, vhx(i)%top, ht, vhx(i)%tp, ht, vhx(i)%tt, ht, vhx(i)%sr,     &
            ht, vhx(i)%tr, ht, vhx(i)%s, ht, vhx(i)%lu, ht, vhx(i)%opn, ht, vhx(i)%fc, ht, vhx(i)%verrflg
    end do

    close ( uid )

end subroutine




!*************************************************************
! SUBROUTINE READ_SHIFT_CONTROL_DATA シフト制御データの読み込み
!  uid       : 装置番号
!  ctrlname  : シフト制御データセットの記述ファイル名
!  ngr       : ギヤ段数
!  ctrl      : シフト制御マップのデータセット
!*************************************************************
subroutine read_shift_control_data21 ( uid, ctrlname, ngr, ctrl )
    use hvtransfc
    implicit none

    integer :: uid, ngr
    type ( sctr ) :: ctrl
    character ( len=* ) :: ctrlname
    character (len=1024) :: basedir, basename, fname

    call getdirectory( ctrlname, basedir, basename )
    open ( uid, file = trim(ctrlname), status = 'old', err=600 )

!   ---------------------------
!    変速線, LU線の設定読み込み
!   ---------------------------
    read ( uid, *, err=610 ) ctrl%upnm
    read ( uid, *, err=610 ) ctrl%dwnnm
    read ( uid, *, err=610 ) ctrl%lonnm
    read ( uid, *, err=610 ) ctrl%loffnm

!   ---------------------------------------
!    ニュートラルアイドル制御の設定読み込み
!   ---------------------------------------
    read (uid, *, err=610) ctrl%swnidl

    if ( ctrl%swnidl == 1 ) then
        read ( uid, *, err=610 ) ctrl%time_nidl
        read ( uid, *, err=610 ) ctrl%sw_nidl_start
        read ( uid, *, err=610 ) ctrl%time_nidl_start
        read ( uid, *, err=610 ) ctrl%sr_nidl
        read ( uid, *, err=610 ) ctrl%cfnm_nidl
    end if

!   ---------------------------------------
!    アイドルストップ制御の設定読み込み
!   ---------------------------------------
    read (uid, *, err=610) ctrl%swistop

    if ( ctrl%swistop == 1 ) then
        read ( uid, *, err=610 ) ctrl%time_iss
        read ( uid, *, err=610 ) ctrl%sw_iss_start
        read ( uid, *, err=610 ) ctrl%time_iss_start
    end if

    close( uid )

!   ---------------------------------------
!    指定ファイルからデータ取り込み
!   ---------------------------------------
!   -- シフトアップ線 --
    fname = trim(basedir) // trim(ctrl%upnm)
    call read_shift_diagram( uid+1, trim(fname), ngr, ctrl%up )

!   -- シフトダウン線 --
    fname = trim(basedir) // trim(ctrl%dwnnm)
    call read_shift_diagram( uid+1, trim(fname), ngr, ctrl%dwn )
```

```
!   -- L/U ON 制御線 --
    if( ctrl%lonnm /= "" ) then
        fname = trim(basedir) // trim(ctrl%lonnm)
    else
        fname = ""
    end if
    call read_lockup_diagram( uid+1, trim(fname), ngr, ctrl%lon )

!   -- L/U OFF 制御線 --
    if( ctrl%loffnm /= "" ) then
        fname = trim(basedir) // trim(ctrl%loffnm)
    else
        fname = ""
    end if
    call read_lockup_diagram( uid+1, trim(fname), ngr, ctrl%loff )

!   -- Nアイドル時の容量係数データ --
    if ( ctrl%swnidl == 1 ) then
        fname = trim(basedir) // trim(ctrl%cfnm_nidl)
        call readcurve ( uid, trim(fname), ctrl%cf_nidl )
    end if

    return

! <<< error handling >>>
600 write (0,'(3A)') " Cannot open SHIFT-CONTROL input file [ ", trim(ctrlname), " ]"
    pause
    stop
610 write (0,'(3A)') " Failed to read SHIFT-CONTROL input file [ ", trim(ctrlname), " ]"
    close ( uid )
    pause
    stop

end subroutine
```

```fortran
module TESTCYCLE
    integer, parameter :: nurban = 1830
    integer, parameter :: nhighway = 3120
    real(8), target :: vurban( 0:nurban+1 )
    real(8), target :: gurban( 0:nurban+1 )
    real(8), target :: vhighway( 0:nhighway+1 )
    real(8), target :: ghighway( 0:nhighway+1 )

    DATA vurban /                                                                                          &
        0.00_8,                                                                                            &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  4.19_8,  8.32_8,  12.33_8, 16.05_8, 18.74_8,   &
        20.28_8, 21.48_8, 23.13_8, 25.17_8, 27.19_8, 28.97_8, 30.43_8, 31.46_8, 32.24_8, 33.16_8,   &
        34.29_8, 35.40_8, 36.57_8, 38.08_8, 39.65_8, 40.59_8, 40.87_8, 41.03_8, 41.23_8, 41.24_8,   &
        41.15_8, 41.11_8, 41.02_8, 40.97_8, 41.25_8, 41.78_8, 42.20_8, 42.54_8, 42.96_8, 43.37_8,   &
        43.84_8, 44.73_8, 46.10_8, 47.57_8, 48.85_8, 49.89_8, 50.56_8, 50.81_8, 50.84_8, 50.87_8,   &
        50.88_8, 50.71_8, 50.31_8, 49.79_8, 49.16_8, 48.09_8, 46.37_8, 44.14_8, 41.46_8, 38.22_8,   &
        34.76_8, 31.55_8, 28.16_8, 23.82_8, 18.88_8, 14.51_8, 11.13_8, 8.59_8,  7.36_8,  8.01_8,    &
        9.99_8,  12.29_8, 14.48_8, 16.35_8, 17.11_8, 15.78_8, 12.39_8, 7.15_8,  1.80_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  3.70_8,  8.97_8,  10.99_8, 11.48_8, 15.12_8, 20.34_8, 23.32_8,   &
        25.11_8, 27.74_8, 30.38_8, 32.93_8, 36.44_8, 39.59_8, 40.72_8, 41.41_8, 43.50_8, 44.40_8,   &
        45.24_8, 45.41_8, 45.17_8, 44.76_8, 44.36_8, 44.01_8, 43.54_8, 42.85_8, 42.35_8, 42.47_8,   &
        42.94_8, 43.20_8, 43.31_8, 43.57_8, 43.96_8, 44.49_8, 45.41_8, 46.55_8, 47.53_8, 48.52_8,   &
        49.86_8, 51.32_8, 52.56_8, 53.69_8, 54.81_8, 55.85_8, 56.88_8, 57.88_8, 58.67_8, 59.31_8,   &
        59.92_8, 60.14_8, 59.88_8, 59.70_8, 59.85_8, 59.86_8, 59.62_8, 59.59_8, 59.81_8, 59.79_8,   &
        59.49_8, 59.24_8, 59.05_8, 58.78_8, 58.53_8, 58.37_8, 58.22_8, 58.08_8, 58.06_8, 58.09_8,   &
        58.05_8, 57.89_8, 57.72_8, 57.61_8, 57.52_8, 57.37_8, 57.14_8, 56.80_8, 56.53_8, 56.71_8,   &
        57.39_8, 57.96_8, 57.98_8, 57.78_8, 57.82_8, 58.01_8, 58.06_8, 57.80_8, 56.98_8, 55.49_8,   &
        53.69_8, 51.95_8, 50.25_8, 48.70_8, 47.64_8, 47.06_8, 46.64_8, 46.30_8, 46.39_8, 47.18_8,   &
        48.55_8, 49.91_8, 50.85_8, 51.65_8, 52.81_8, 54.13_8, 55.10_8, 55.75_8, 56.29_8, 56.14_8,   &
        54.54_8, 51.61_8, 48.27_8, 45.40_8, 43.49_8, 42.66_8, 42.71_8, 43.29_8, 44.16_8, 45.28_8,   &
        46.64_8, 48.05_8, 49.42_8, 51.05_8, 52.97_8, 54.57_8, 55.57_8, 56.53_8, 57.67_8, 58.42_8,   &
        58.81_8, 59.56_8, 60.52_8, 60.89_8, 60.87_8, 61.27_8, 61.88_8, 62.11_8, 62.23_8, 62.39_8,   &
        61.87_8, 60.48_8, 59.06_8, 58.16_8, 57.46_8, 56.79_8, 56.36_8, 56.16_8, 56.09_8, 56.15_8,   &
        56.18_8, 56.00_8, 55.71_8, 55.60_8, 55.76_8, 56.26_8, 57.22_8, 58.37_8, 59.12_8, 59.37_8,   &
        59.53_8, 59.73_8, 59.74_8, 59.59_8, 59.56_8, 59.65_8, 59.86_8, 60.40_8, 61.23_8, 61.99_8,   &
        62.64_8, 63.32_8, 63.74_8, 63.61_8, 63.25_8, 62.88_8, 62.25_8, 61.48_8, 61.06_8, 60.78_8,   &
        60.00_8, 58.97_8, 58.32_8, 58.01_8, 57.65_8, 57.20_8, 56.65_8, 55.92_8, 55.27_8, 54.77_8,   &
        54.16_8, 53.49_8, 53.06_8, 52.74_8, 52.38_8, 52.25_8, 52.33_8, 52.21_8, 52.05_8, 52.32_8,   &
        52.64_8, 52.38_8, 51.61_8, 50.48_8, 48.76_8, 46.68_8, 44.77_8, 42.88_8, 40.60_8, 38.17_8,   &
        35.70_8, 32.76_8, 28.21_8, 23.82_8, 20.17_8, 16.37_8, 10.92_8, 4.99_8,  1.06_8,  0.00_8,    &
        0.00_8,  0.00_8,  1.78_8,  4.02_8,  7.51_8,  12.17_8, 16.29_8, 18.22_8, 19.22_8, 21.99_8,   &
        24.70_8, 26.87_8, 27.96_8, 28.32_8, 28.05_8, 27.45_8, 27.05_8, 26.82_8, 26.53_8, 26.69_8,   &
        27.80_8, 29.17_8, 29.87_8, 30.11_8, 30.63_8, 31.59_8, 32.84_8, 34.17_8, 35.18_8, 35.58_8,   &
        35.67_8, 36.07_8, 37.08_8, 38.37_8, 39.26_8, 39.60_8, 39.96_8, 40.58_8, 40.91_8, 40.73_8,   &
        40.53_8, 40.51_8, 40.37_8, 40.06_8, 39.76_8, 39.46_8, 39.41_8, 39.81_8, 39.89_8, 38.96_8,   &
        37.88_8, 37.95_8, 39.17_8, 40.68_8, 41.98_8, 43.09_8, 44.24_8, 45.66_8, 47.17_8, 48.25_8,   &
        48.61_8, 48.39_8, 47.83_8, 47.28_8, 46.95_8, 46.61_8, 46.14_8, 45.86_8, 45.89_8, 45.76_8,   &
        45.18_8, 44.31_8, 43.27_8, 41.85_8, 39.69_8, 36.81_8, 33.66_8, 30.55_8, 27.25_8, 23.77_8,   &
        20.85_8, 18.65_8, 16.41_8, 13.89_8, 11.80_8, 10.42_8, 9.38_8,  8.61_8,  8.14_8,  7.47_8,    &
        6.43_8,  4.35_8,  2.49_8,  1.27_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  3.37_8,  9.10_8,  14.02_8, 17.20_8, 20.22_8, 23.49_8, 26.43_8, 28.90_8, 30.55_8,   &
        31.17_8, 31.42_8, 31.48_8, 30.84_8, 29.90_8, 29.66_8, 29.20_8, 28.45_8, 27.40_8, 26.21_8,   &
        25.27_8, 24.81_8, 24.97_8, 26.03_8, 27.81_8, 29.48_8, 30.48_8, 30.85_8, 30.59_8, 29.84_8,   &
        28.92_8, 27.47_8, 24.78_8, 21.41_8, 18.66_8, 16.85_8, 15.79_8, 16.08_8, 18.06_8, 21.01_8,   &
        24.26_8, 27.72_8, 31.07_8, 33.82_8, 35.90_8, 37.26_8, 37.71_8, 37.50_8, 37.07_8, 36.47_8,   &
        35.57_8, 34.41_8, 33.12_8, 31.87_8, 30.79_8, 29.85_8, 28.93_8, 28.08_8, 27.60_8, 28.02_8,   &
        29.68_8, 31.96_8, 33.94_8, 35.57_8, 37.21_8, 38.51_8, 39.39_8, 40.58_8, 42.20_8, 43.44_8,   &
        44.19_8, 44.96_8, 45.73_8, 46.29_8, 46.87_8, 47.51_8, 48.07_8, 48.82_8, 49.85_8, 50.68_8,   &
        51.26_8, 52.04_8, 52.82_8, 53.22_8, 53.53_8, 54.00_8, 54.31_8, 54.35_8, 54.37_8, 54.28_8,   &
        53.91_8, 53.18_8, 51.82_8, 49.83_8, 47.71_8, 45.39_8, 41.80_8, 37.47_8, 33.19_8, 30.27_8,   &
        26.16_8, 19.57_8, 13.81_8, 11.04_8, 9.11_8,  6.17_8,  3.13_8,  1.17_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        3.83_8,  9.38_8,  13.85_8, 14.91_8, 15.68_8, 19.52_8, 24.58_8, 27.20_8, 27.48_8, 27.85_8,   &
        29.15_8, 31.13_8, 33.52_8, 35.89_8, 37.09_8, 37.33_8, 37.10_8, 36.30_8, 35.03_8, 34.21_8,   &
        34.23_8, 34.31_8, 33.99_8, 33.82_8, 34.34_8, 35.49_8, 37.22_8, 39.53_8, 41.98_8, 44.08_8,   &
        45.69_8, 46.78_8, 47.45_8, 47.84_8, 47.82_8, 47.14_8, 46.06_8, 45.13_8, 44.55_8, 44.41_8,   &
        44.84_8, 45.56_8, 45.84_8, 45.28_8, 43.79_8, 41.57_8, 39.00_8, 36.35_8, 33.60_8, 30.97_8,   &
        28.86_8, 27.00_8, 24.95_8, 23.05_8, 21.71_8, 20.52_8, 19.39_8, 19.06_8, 19.70_8, 20.50_8,   &
        20.95_8, 21.18_8, 21.19_8, 20.66_8, 19.26_8, 16.67_8, 13.34_8, 10.48_8, 8.59_8,  6.93_8,    &
        4.36_8,  2.09_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
        1.05_8,  5.67_8,  9.44_8,  13.24_8, 16.38_8, 18.36_8, 19.93_8, 22.25_8, 25.25_8, 28.34_8,   &
        31.32_8, 33.95_8, 35.96_8, 37.89_8, 40.21_8, 42.12_8, 42.93_8, 43.53_8, 44.80_8, 46.02_8,   &
        46.29_8, 46.15_8, 46.42_8, 47.03_8, 47.57_8, 48.10_8, 48.68_8, 49.16_8, 49.56_8, 50.16_8,   &
        50.97_8, 51.75_8, 52.42_8, 53.00_8, 53.38_8, 53.57_8, 53.70_8, 53.61_8, 53.06_8, 52.29_8,   &
        51.78_8, 51.48_8, 50.93_8, 49.93_8, 48.45_8, 46.42_8, 43.97_8, 41.48_8, 39.39_8, 38.18_8,   &
        38.09_8, 38.70_8, 39.19_8, 39.06_8, 38.27_8, 37.02_8, 35.67_8, 34.61_8, 33.89_8, 33.32_8,   &
        32.62_8, 31.41_8, 29.63_8, 27.83_8, 26.44_8, 25.40_8, 24.84_8, 25.24_8, 26.34_8, 27.09_8,   &
        27.12_8, 27.01_8, 27.21_8, 27.70_8, 28.48_8, 29.54_8, 30.60_8, 31.61_8, 32.80_8, 34.11_8,   &
        35.20_8, 36.10_8, 37.13_8, 38.13_8, 38.62_8, 38.60_8, 38.48_8, 38.23_8, 37.40_8, 35.99_8,   &
        34.45_8, 33.07_8, 31.81_8, 30.59_8, 29.40_8, 28.40_8, 27.63_8, 26.57_8, 24.25_8, 20.69_8,   &
        14.60_8, 8.99_8,  4.76_8,  1.64_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,    &
```

```
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  3.57_8,  8.28_8,  11.75_8, 13.06_8, 15.07_8, 18.64_8, 21.16_8, 22.19_8, 22.89_8,    &
23.73_8, 23.37_8, 22.87_8, 22.73_8, 22.51_8, 22.01_8, 21.45_8, 21.23_8, 22.02_8, 23.88_8,    &
25.74_8, 26.82_8, 27.78_8, 29.33_8, 31.26_8, 33.32_8, 35.53_8, 37.60_8, 39.26_8, 40.64_8,    &
41.70_8, 42.23_8, 42.50_8, 42.75_8, 42.61_8, 41.89_8, 40.86_8, 39.56_8, 37.87_8, 36.03_8,    &
34.13_8, 31.63_8, 27.79_8, 22.97_8, 18.01_8, 13.36_8, 9.31_8,  6.70_8,  5.31_8,  3.98_8,     &
2.54_8,  1.40_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  2.62_8,  3.82_8,  4.08_8,  6.12_8,     &
8.81_8,  9.73_8,  9.59_8,  9.44_8,  9.45_8,  9.35_8,  9.30_8,  9.75_8,  10.70_8, 11.61_8,    &
12.02_8, 12.02_8, 11.71_8, 10.78_8, 9.34_8,  6.66_8,  4.63_8,  3.28_8,  1.70_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  2.43_8,  4.63_8,  7.93_8,  9.13_8,  10.21_8, 11.28_8,    &
12.87_8, 14.44_8, 15.28_8, 15.41_8, 15.33_8, 15.28_8, 14.97_8, 14.23_8, 13.70_8, 14.26_8,    &
15.77_8, 17.25_8, 18.21_8, 18.82_8, 19.00_8, 18.44_8, 17.29_8, 16.12_8, 15.00_8, 13.52_8,    &
11.83_8, 10.76_8, 10.49_8, 10.04_8, 8.94_8,  8.11_8,  8.15_8,  8.24_8,  7.77_8,  7.65_8,     &
8.64_8,  10.04_8, 10.94_8, 11.29_8, 11.36_8, 11.01_8, 10.01_8, 8.54_8,  7.13_8,  6.41_8,     &
6.79_8,  8.38_8,  10.73_8, 12.83_8, 14.04_8, 14.97_8, 16.40_8, 18.03_8, 19.52_8, 21.53_8,    &
24.25_8, 26.42_8, 27.30_8, 27.75_8, 28.38_8, 28.62_8, 28.01_8, 26.91_8, 25.46_8, 23.49_8,    &
20.45_8, 17.47_8, 14.80_8, 12.03_8, 9.34_8,  7.27_8,  5.43_8,  3.23_8,  1.22_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  1.92_8,  3.93_8,  6.80_8,  9.57_8,  12.26_8, 13.88_8, 14.61_8, 15.12_8, 15.52_8,    &
15.14_8, 13.51_8, 11.06_8, 8.82_8,  7.51_8,  7.24_8,  7.54_8,  7.69_8,  7.12_8,  5.85_8,     &
3.90_8,  2.23_8,  1.49_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  1.08_8,  1.34_8,  3.04_8,  3.84_8,  4.07_8,  5.12_8,  7.12_8,  9.07_8,  10.25_8,    &
10.65_8, 10.61_8, 10.78_8, 11.61_8, 12.65_8, 13.20_8, 13.16_8, 12.95_8, 12.77_8, 12.50_8,    &
12.07_8, 11.66_8, 11.35_8, 10.77_8, 9.56_8,  8.03_8,  6.72_8,  5.73_8,  4.94_8,  4.46_8,     &
4.29_8,  4.15_8,  3.85_8,  3.31_8,  2.49_8,  1.33_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
1.28_8,  1.60_8,  2.63_8,  5.02_8,  8.68_8,  12.57_8, 15.07_8, 16.22_8, 17.46_8, 19.65_8,    &
20.82_8, 21.47_8, 22.09_8, 22.09_8, 20.95_8, 18.99_8, 16.56_8, 14.08_8, 12.39_8, 11.84_8,    &
11.86_8, 12.11_8, 13.01_8, 14.67_8, 16.56_8, 18.29_8, 20.07_8, 22.45_8, 25.37_8, 27.84_8,    &
29.36_8, 30.76_8, 32.49_8, 33.61_8, 33.67_8, 33.55_8, 33.29_8, 32.04_8, 30.09_8, 28.23_8,    &
26.18_8, 23.77_8, 22.06_8, 21.48_8, 21.25_8, 21.09_8, 21.08_8, 20.47_8, 18.82_8, 16.86_8,    &
14.85_8, 11.76_8, 8.45_8,  5.33_8,  3.78_8,  2.45_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  1.86_8,     &
6.31_8,  9.90_8,  12.02_8, 13.52_8, 15.04_8, 14.83_8, 13.43_8, 12.27_8, 12.79_8, 14.79_8,    &
16.84_8, 18.64_8, 20.87_8, 23.02_8, 24.13_8, 24.60_8, 24.92_8, 24.67_8, 23.86_8, 22.97_8,    &
21.50_8, 19.10_8, 16.70_8, 15.04_8, 13.91_8, 13.35_8, 13.40_8, 13.35_8, 12.77_8, 11.82_8,    &
9.99_8,  7.19_8,  5.07_8,  4.85_8,  5.29_8,  4.82_8,  3.66_8,  1.87_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,     &
0.00_8,  0.00_8,  0.00_8,  0.00_8,  0.00_8,  3.50_8,  5.08_8,  5.97_8,  9.46_8,  13.96_8,    &
15.88_8, 16.84_8, 19.06_8, 21.53_8, 23.63_8, 25.88_8, 28.25_8, 30.55_8, 32.83_8, 34.81_8,    &
36.22_8, 37.19_8, 38.01_8, 38.69_8, 39.31_8, 40.16_8, 41.24_8, 42.33_8, 43.38_8, 44.56_8,    &
45.85_8, 47.02_8, 47.93_8, 48.80_8, 49.73_8, 50.57_8, 51.32_8, 52.19_8, 53.16_8, 53.98_8,    &
54.72_8, 55.55_8, 56.47_8, 57.48_8, 58.69_8, 60.00_8, 61.20_8, 62.42_8, 63.75_8, 65.05_8,    &
66.16_8, 67.12_8, 67.89_8, 68.54_8, 69.22_8, 69.98_8, 70.71_8, 71.47_8, 72.36_8, 73.35_8,    &
74.41_8, 75.52_8, 76.52_8, 77.39_8, 78.29_8, 79.22_8, 79.95_8, 80.45_8, 80.88_8, 81.25_8,    &
81.56_8, 81.81_8, 81.86_8, 81.66_8, 81.19_8, 80.68_8, 80.44_8, 80.39_8, 80.29_8, 80.21_8,    &
80.19_8, 80.03_8, 79.63_8, 79.25_8, 79.09_8, 79.08_8, 79.01_8, 78.84_8, 78.61_8, 78.44_8,    &
78.34_8, 78.23_8, 78.15_8, 78.19_8, 78.28_8, 78.34_8, 78.46_8, 78.72_8, 79.03_8, 79.30_8,    &
79.61_8, 79.99_8, 80.39_8, 80.75_8, 81.08_8, 81.39_8, 81.73_8, 82.05_8, 82.37_8, 82.74_8,    &
83.10_8, 83.34_8, 83.46_8, 83.51_8, 83.42_8, 83.22_8, 83.08_8, 82.97_8, 82.72_8, 82.41_8,    &
82.17_8, 81.84_8, 81.34_8, 80.89_8, 80.63_8, 80.42_8, 80.17_8, 79.93_8, 79.67_8, 79.45_8,    &
79.42_8, 79.50_8, 79.50_8, 79.53_8, 79.72_8, 79.88_8, 79.81_8, 79.69_8, 79.75_8, 79.95_8,    &
80.24_8, 80.68_8, 81.25_8, 81.84_8, 82.39_8, 82.90_8, 83.42_8, 83.92_8, 84.34_8, 84.67_8,    &
84.94_8, 85.12_8, 85.09_8, 84.86_8, 84.51_8, 84.09_8, 83.66_8, 83.30_8, 82.94_8, 82.54_8,    &
82.18_8, 81.96_8, 81.86_8, 81.85_8, 81.82_8, 81.64_8, 81.37_8, 81.15_8, 80.89_8, 80.50_8,    &
80.25_8, 80.39_8, 80.83_8, 81.44_8, 82.31_8, 83.38_8, 84.39_8, 85.24_8, 86.00_8, 86.67_8,    &
87.20_8, 87.55_8, 87.60_8, 87.39_8, 87.10_8, 86.87_8, 86.62_8, 86.35_8, 86.17_8, 85.99_8,    &
85.77_8, 85.59_8, 85.51_8, 85.45_8, 85.43_8, 85.61_8, 85.99_8, 86.30_8, 86.45_8, 86.50_8,    &
86.57_8, 86.66_8, 86.79_8, 86.98_8, 87.08_8, 86.85_8, 86.16_8, 85.28_8, 84.52_8, 83.98_8,    &
83.51_8, 83.10_8, 82.77_8, 82.60_8, 81.91_8, 80.94_8, 79.82_8, 78.50_8, 77.00_8, 75.57_8,    &
74.34_8, 73.14_8, 71.88_8, 70.73_8, 69.59_8, 67.81_8, 64.91_8, 60.93_8, 56.12_8, 50.87_8,    &
45.70_8, 40.78_8, 35.82_8, 30.85_8, 26.48_8, 23.12_8, 20.59_8, 18.47_8, 16.69_8, 15.82_8,    &
15.57_8, 15.98_8, 17.14_8, 18.68_8, 20.11_8, 21.30_8, 22.22_8, 22.50_8, 22.13_8, 21.85_8,    &
22.02_8, 22.17_8, 21.86_8, 21.08_8, 19.50_8, 16.78_8, 13.55_8, 11.03_8, 9.72_8,  9.38_8,     &
9.55_8,  9.71_8,  9.65_8,  9.80_8,  10.85_8, 12.80_8, 15.13_8, 17.67_8, 20.63_8, 23.74_8,    &
26.17_8, 27.49_8, 28.09_8, 28.36_8, 28.16_8, 27.31_8, 26.07_8, 24.71_8, 23.29_8, 22.00_8,    &
20.98_8, 19.93_8, 18.57_8, 17.29_8, 16.67_8, 16.69_8, 17.09_8, 17.92_8, 19.14_8, 20.34_8,    &
21.10_8, 21.30_8, 21.06_8, 20.63_8, 20.33_8, 20.44_8, 20.97_8, 21.60_8, 21.76_8, 21.39_8,    &
21.23_8, 21.63_8, 21.90_8, 21.45_8, 20.74_8, 20.26_8, 19.76_8, 19.11_8, 18.79_8, 18.97_8,    &
19.31_8, 19.90_8, 21.06_8, 22.54_8, 23.80_8, 24.79_8, 25.59_8, 26.01_8, 25.83_8, 25.26_8,    &
24.73_8, 24.39_8, 23.94_8, 23.30_8, 23.10_8, 23.72_8, 24.49_8, 24.77_8, 25.01_8, 25.58_8,    &
25.92_8, 25.88_8, 26.08_8, 26.44_8, 26.17_8, 25.39_8, 24.87_8, 24.61_8, 24.22_8, 23.93_8,    &
24.01_8, 24.00_8, 23.27_8, 22.03_8, 21.23_8, 21.51_8, 22.53_8, 23.61_8, 24.63_8, 25.66_8,    &
26.14_8, 25.76_8, 25.08_8, 24.34_8, 22.99_8, 21.14_8, 19.79_8, 19.14_8, 18.49_8, 17.60_8,    &
16.83_8, 16.34_8, 16.15_8, 16.24_8, 16.37_8, 16.26_8, 15.85_8, 15.12_8, 14.32_8, 13.93_8,    &
13.94_8, 13.75_8, 13.41_8, 13.58_8, 14.32_8, 15.23_8, 16.18_8, 16.91_8, 16.85_8, 16.20_8,    &
15.78_8, 15.84_8, 16.19_8, 16.95_8, 17.97_8, 18.49_8, 18.03_8, 16.97_8, 16.16_8, 16.41_8,    &
17.91_8, 19.70_8, 20.54_8, 20.40_8, 20.22_8, 20.55_8, 21.16_8, 21.53_8, 21.28_8, 20.29_8,    &
```

```
      18.95_8,  17.79_8,  16.89_8,  15.98_8,  15.16_8,  13.24_8,  10.27_8,  5.06_8,   0.00_8,   0.00_8,     &
      0.00_8                                                                                                 &
  /

  DATA gurban / 1832*0.0_8 /

  DATA vhighway / 3122*80.0_8 /

  DATA ghighway                                                                                              &
  /   0.000_8,                                                                                              &
      0.000_8,   0.000_8,   0.000_8,   0.000_8,  -0.228_8, -2.280_8, -2.280_8, -2.280_8, -2.280_8, -2.280_8, &
     -2.280_8, -2.280_8, -0.669_8, -0.490_8, -0.490_8, -0.490_8, -0.490_8, -0.490_8, 0.300_8,  0.494_8,      &
      1.270_8,   1.270_8,   3.135_8,   5.000_8,   5.000_8,   1.400_8,  -4.000_8, -4.000_8, -4.000_8, -1.780_8, &
      0.712_8,  -1.080_8, -1.080_8, -1.080_8,   2.200_8,   2.200_8,   2.200_8,   2.200_8,   2.200_8,   2.200_8, &
      2.200_8,   2.200_8,   2.200_8,   1.162_8,  -2.990_8, -2.990_8, -2.990_8, -2.990_8, -2.990_8, -2.639_8, &
      0.520_8,   0.520_8,   0.520_8,   0.520_8,   0.520_8,   0.768_8,   0.830_8,   0.830_8,   0.830_8,   0.830_8,   &
     -2.910_8, -2.910_8, -2.910_8, -2.910_8, -2.910_8,   1.786_8,   2.960_8,   2.960_8,   2.960_8,   2.960_8,   &
      2.960_8,   0.576_8,  -3.000_8,  -3.000_8,  -3.000_8,  -3.000_8,  -3.000_8,  -3.000_8,  2.247_8,   2.830_8,   &
      2.830_8,   2.830_8,   2.830_8,   2.830_8,   2.830_8,   2.830_8,   2.830_8,   1.122_8,   0.996_8,   2.410_8,   &
      2.410_8,   2.410_8,   2.410_8,   2.410_8,  -0.586_8, -1.870_8, -1.600_8, -0.520_8, -0.520_8, -0.520_8, &
     -0.288_8, -0.230_8, -0.230_8,   0.292_8,   2.380_8,   2.380_8,   2.380_8,  -0.221_8, -0.510_8, -0.510_8, &
     -0.510_8, -1.998_8, -2.370_8, -2.370_8, -2.370_8, -2.370_8, -2.370_8, -2.370_8, -2.370_8, -1.582_8, &
     -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8,  0.520_8,   0.520_8,   0.520_8,   &
      0.520_8,   0.520_8,   0.520_8,  -0.472_8, -1.960_8, -1.960_8, -1.006_8,  1.220_8,   1.220_8,   1.131_8,   &
      0.100_8,  -1.970_8, -1.970_8, -1.970_8, -1.970_8, -1.970_8, -1.970_8, -1.970_8, -0.745_8, -0.220_8, &
     -0.220_8, -0.036_8,   0.240_8,   0.240_8,   0.240_8,   0.240_8,   0.240_8,   0.090_8,  -0.060_8, -0.060_8, &
     -0.060_8, -0.060_8, -0.060_8,   0.868_8,   1.100_8,   1.100_8,   0.456_8,  -0.510_8, -0.510_8, -0.510_8, &
     -0.510_8, -0.249_8,   0.360_8,   0.360_8,   0.360_8,   0.360_8,   0.360_8,  -0.540_8, -0.640_8, -0.122_8, &
      1.950_8,   1.950_8,   1.680_8,   1.410_8,   1.410_8,   1.410_8,   1.410_8,   1.455_8,   1.860_8,   1.860_8,   &
      1.860_8,   1.860_8,   1.860_8,  -0.316_8, -0.860_8, -0.860_8, -0.860_8, -0.860_8, -1.135_8, -1.410_8, &
     -1.410_8, -1.410_8, -1.410_8, -1.239_8,   0.300_8,   0.300_8,   0.126_8,   0.010_8,   0.010_8,   0.010_8,   &
      0.010_8,   0.155_8,   0.300_8,   0.300_8,   0.300_8,   0.760_8,   2.600_8,   2.600_8,   1.300_8,   1.300_8,   &
      1.300_8,   1.300_8,   1.300_8,   1.492_8,   3.220_8,   3.220_8,   3.220_8,   3.220_8,   3.220_8,   1.596_8,   &
      0.900_8,   0.900_8,   0.900_8,   0.900_8,   0.778_8,  -0.320_8,  -0.320_8, -2.092_8, -4.750_8, -4.750_8, &
     -4.750_8,   2.585_8,   3.400_8,   2.334_8,  -1.930_8, -1.930_8, -1.930_8, -1.930_8, -1.930_8, -1.004_8, &
      2.700_8,   2.700_8,   2.700_8,   2.700_8,   2.970_8,   3.000_8,   3.000_8,   3.000_8,  -4.011_8, -4.790_8, &
     -3.546_8, -1.680_8, -1.680_8, -1.680_8, -1.680_8, -1.680_8, -1.680_8, -1.680_8, -1.515_8, -1.130_8, &
     -0.757_8,   2.600_8,   2.600_8,  -2.144_8, -2.797_8,   2.108_8,   2.540_8,   3.070_8,   3.600_8,   2.652_8,   &
      0.440_8,   0.440_8,  -1.926_8, -2.940_8,   0.012_8,   1.530_8,   1.080_8,   1.233_8,   1.250_8,   1.250_8,   &
      1.724_8,   2.040_8,   2.040_8,   2.040_8,   4.740_8,   5.040_8,   5.040_8,   5.040_8,   5.040_8,   3.164_8,   &
      2.360_8,   2.360_8,   2.360_8,   2.360_8,   1.754_8,   1.350_8,   1.350_8,   1.350_8,   1.350_8,   1.350_8,   &
      1.548_8,   1.680_8,   1.680_8,   1.244_8,  -0.500_8, -0.500_8, -0.500_8, -0.563_8, -0.710_8, -0.710_8, &
     -0.048_8,   2.600_8,   2.600_8,   2.600_8,   2.600_8,   2.600_8,   1.370_8,   0.550_8,   0.550_8,   1.040_8,   &
      3.000_8,   3.000_8,   3.000_8,   3.000_8,   3.000_8,   1.120_8,   1.120_8,   1.120_8,   1.120_8,   1.120_8,   &
      1.120_8,   2.880_8,   2.880_8,   2.880_8,   2.880_8,   2.880_8,   2.515_8,  -0.770_8, -0.770_8, -0.770_8, &
     -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8,   4.000_8,   4.000_8,   4.000_8,   4.000_8,   &
      4.000_8,   4.000_8,   4.000_8,   2.696_8,   2.370_8,   2.370_8,   2.370_8,  -0.126_8, -0.750_8, -0.750_8, &
     -0.750_8, -0.750_8, -0.750_8, -0.510_8,   1.650_8,   1.650_8,   1.347_8,   0.640_8,   0.640_8,   0.640_8,   &
      0.640_8,   0.640_8,  -0.180_8, -1.000_8, -1.000_8, -1.000_8, -0.325_8, -0.250_8, -0.250_8, -0.250_8, &
     -1.218_8, -1.460_8, -1.460_8, -1.460_8, -1.460_8, -1.873_8, -2.050_8, -2.050_8, -2.050_8, -2.050_8, &
     -2.050_8, -2.090_8, -2.450_8, -2.450_8, -2.450_8, -1.855_8, -1.260_8, -1.260_8, -1.260_8, -1.260_8, &
     -1.260_8, -1.260_8, -1.260_8, -1.260_8, -1.260_8, -1.260_8, -1.260_8, -2.644_8, -2.990_8, -2.990_8, &
     -2.990_8, -2.990_8, -2.917_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, &
     -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.260_8, -2.630_8, &
     -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8,  0.770_8,   0.770_8,   0.770_8,   0.770_8,   &
      0.770_8,   0.770_8,   0.770_8,   0.770_8,   0.770_8,   0.770_8,   0.770_8,   0.149_8,  -1.300_8, -1.300_8, &
     -1.300_8, -1.300_8, -1.300_8, -1.300_8, -0.180_8,   0.300_8,   0.300_8,   0.300_8,   0.300_8,   0.300_8,   &
      0.300_8,   0.300_8,   0.300_8,   0.300_8,   0.300_8,   0.300_8,   0.300_8,   0.012_8,  -0.660_8, -0.660_8, &
     -0.660_8, -0.660_8, -0.866_8, -2.720_8, -2.720_8, -2.720_8, -2.720_8, -2.720_8, -2.250_8, -0.370_8, -0.370_8, &
     -0.370_8, -0.370_8, -0.370_8, -0.370_8, -0.370_8, -0.370_8, -0.632_8, -1.680_8, -1.680_8, -1.680_8, &
     -1.680_8, -1.680_8, -1.680_8, -1.680_8, -1.680_8, -0.154_8,  0.500_8,   0.500_8,   0.500_8,   0.500_8,   &
      0.500_8,   0.500_8,  -0.217_8, -1.890_8, -1.890_8, -1.890_8, -1.890_8,  1.251_8,   1.600_8,   1.600_8,   &
      1.600_8,   1.600_8,   1.600_8,   1.222_8,  -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8,  1.020_8,   &
      2.330_8,   2.330_8,   2.330_8,   2.330_8,  -0.030_8, -0.620_8, -0.620_8, -0.620_8, -0.620_8,  0.148_8,   1.300_8,   &
      1.300_8,   1.300_8,   0.600_8,  -0.450_8, -0.450_8, -0.450_8, -0.208_8,  0.760_8,   0.760_8,   0.760_8,   &
      0.760_8,   0.760_8,   0.760_8,   0.760_8,   0.760_8,   0.760_8,  -2.760_8, -2.760_8, -2.760_8, -2.760_8, &
     -2.760_8, -2.760_8, -2.760_8, -2.760_8, -2.760_8, -2.760_8, -2.760_8, -2.760_8, -2.760_8, -0.192_8, &
      0.450_8,   0.450_8,   0.450_8,   0.880_8,   2.600_8,   2.600_8,   2.600_8,   2.600_8,   2.600_8,   2.600_8,   &
      2.600_8,   2.600_8,   2.600_8,   0.500_8,  -2.650_8, -2.650_8, -2.650_8, -2.650_8, -2.650_8, -2.650_8, &
     -2.650_8, -2.650_8, -2.650_8, -2.335_8,   0.500_8,   0.500_8,   0.500_8,   0.500_8,   0.500_8,   0.500_8,   &
      0.500_8,   0.050_8,  -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, -1.040_8, &
     -2.000_8, -2.000_8, -1.800_8,   0.000_8,   0.000_8,   0.000_8,   0.150_8,   0.500_8,   0.500_8,   0.500_8,   &
      0.248_8,  -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.228_8, &
      0.000_8,   0.000_8,   0.000_8,   0.000_8,   2.232_8,   2.790_8,   2.790_8,   2.790_8,   2.790_8,   1.772_8,   &
     -2.300_8, -2.300_8, -2.300_8, -2.300_8, -2.300_8, -2.300_8, -2.300_8, -2.300_8,  0.962_8,   2.360_8,   &
      2.360_8,   2.360_8,   2.360_8,   2.360_8,   2.360_8,   2.360_8,   2.360_8,  -0.692_8, -2.000_8, -2.000_8, &
     -2.000_8,   0.403_8,   0.670_8,   0.670_8,   0.670_8,  -0.257_8,  -0.360_8,  -0.360_8,  -0.360_8,  -0.360_8,   &
     -0.297_8,  -0.290_8,  -0.290_8,  -0.290_8,  -0.290_8,  -0.346_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,   &
     -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,  -0.370_8,   &
      0.080_8,   0.130_8,   0.130_8,   0.130_8,   0.130_8,   0.130_8,   0.130_8,   0.130_8,   1.230_8,   2.330_8,   2.330_8,   &
      2.330_8,   2.330_8,   2.330_8,   2.330_8,   2.330_8,   2.330_8,   2.330_8,   2.330_8,  -0.541_8, -0.860_8, &
     -0.860_8, -0.860_8, -0.860_8, -1.105_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, &
     -1.350_8, -1.350_8, -1.350_8, -1.350_8, -1.350_8, -0.618_8, -0.130_8, -0.130_8,  0.293_8,   0.340_8,   &
      0.340_8,   0.340_8,   0.340_8,   0.340_8,   0.340_8,   0.340_8,   0.340_8,  -0.045_8, -0.430_8, -0.430_8, &
     -0.430_8, -0.430_8, -0.430_8,   0.102_8,   0.900_8,   0.900_8,   0.900_8,   0.366_8,  -0.880_8, -0.880_8, &
     -0.880_8, -0.880_8, -0.880_8, -0.880_8, -0.880_8, -0.880_8,  0.208_8,   1.840_8,   1.840_8,   1.840_8,   &
      1.840_8,   1.840_8,   1.840_8,   1.840_8,   1.840_8,   1.406_8,  -2.500_8, -2.500_8, -2.500_8, -2.500_8, &
     -2.500_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, -0.100_8, &
     -0.100_8,   0.140_8,   0.200_8,   0.200_8,   0.200_8,   0.200_8,   0.220_8,   0.300_8,   0.300_8,   0.300_8,   &
      0.250_8,  -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8,  1.240_8,   1.600_8,   1.600_8,   1.240_8,   &
     -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -0.020_8, &
      0.200_8,   0.200_8,   0.200_8,   0.200_8,   0.200_8,   0.200_8,   0.083_8,   0.070_8,   0.070_8,   0.070_8,   &
      0.070_8,   0.070_8,   0.070_8,   0.070_8,   0.222_8,   0.450_8,   0.450_8,   0.450_8,   0.450_8,   0.450_8,   &
      0.450_8,   0.450_8,   0.450_8,   0.450_8,   0.385_8,  -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, &
     -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.398_8, -2.180_8, -2.180_8, -2.180_8, -2.180_8,  1.048_8,   &
      3.200_8,   3.200_8,   3.200_8,   3.200_8,   1.555_8,   0.850_8,   0.850_8,   0.850_8,  -1.090_8, -4.000_8, &
```

```
-4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, -4.000_8, 2.800_8, &
4.500_8, 4.500_8, 4.500_8, 4.500_8, 4.500_8, 4.500_8, 2.500_8, 2.000_8, 2.000_8, 2.000_8, &
2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 1.686_8, 0.430_8, 0.430_8, 0.430_8, 0.430_8, &
0.430_8, 0.430_8, 0.430_8, -1.714_8, -4.930_8, -4.930_8, -4.930_8, -4.930_8, -4.930_8, -2.059_8, &
-1.740_8, -1.740_8, -1.740_8, -1.740_8, -2.370_8, -3.000_8, -3.000_8, -2.719_8, -0.190_8, -0.190_8, &
-0.190_8, -0.190_8, -0.190_8, -0.190_8, -0.190_8, -0.190_8, 1.840_8, 1.840_8, 1.840_8, 1.840_8, &
1.732_8, 0.760_8, 0.760_8, 0.760_8, -0.423_8, -0.930_8, -0.930_8, -0.604_8, 0.700_8, 0.700_8, &
0.700_8, 0.700_8, -0.920_8, -1.100_8, -1.100_8, -1.100_8, -1.100_8, -0.828_8, -0.420_8, -0.420_8, &
-0.420_8, -0.420_8, -0.420_8, -0.420_8, -0.420_8, -0.078_8, 0.720_8, 0.720_8, 0.720_8, -0.370_8, &
-0.370_8, -0.370_8, -0.370_8, -0.812_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, &
-2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -2.580_8, -0.683_8, 0.130_8, 0.130_8, 0.130_8, &
0.130_8, 0.130_8, 0.130_8, 0.130_8, -0.493_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, &
-0.760_8, -0.760_8, -0.394_8, 0.460_8, 0.460_8, 0.460_8, 0.460_8, -0.317_8, -0.650_8, -0.650_8, &
-0.650_8, -0.650_8, -0.650_8, -0.650_8, -0.650_8, 0.136_8, 0.660_8, 0.660_8, 0.660_8, 0.660_8, &
0.660_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 1.068_8, -1.830_8, -1.830_8, -1.830_8, &
-1.830_8, -1.830_8, 0.126_8, 1.430_8, 1.430_8, 1.430_8, 1.430_8, 0.576_8, -2.840_8, -2.840_8, &
-2.840_8, -2.840_8, -2.840_8, -2.840_8, -1.270_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, &
0.300_8, -0.510_8, -0.510_8, -0.510_8, -0.510_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, 0.200_8, &
0.200_8, -0.087_8, -0.210_8, -0.210_8, -0.210_8, 0.036_8, 0.200_8, 0.200_8, 0.200_8, 0.060_8, &
0.000_8, 0.000_8, 0.000_8, -0.120_8, -0.400_8, -0.400_8, -0.400_8, -0.400_8, 0.800_8, 2.600_8, &
2.600_8, 2.600_8, 2.600_8, 2.600_8, 2.600_8, 2.600_8, 1.420_8, 0.240_8, 0.240_8, 1.235_8, &
2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, 2.230_8, &
2.230_8, 2.230_8, 0.529_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.760_8, -3.000_8, &
-3.000_8, -3.000_8, 0.210_8, 2.350_8, 2.350_8, 2.350_8, 2.350_8, 0.541_8, 0.340_8, 0.340_8, &
0.340_8, -0.170_8, -0.510_8, -0.510_8, -0.510_8, -1.302_8, -1.830_8, -1.830_8, -1.830_8, -1.830_8, &
-1.830_8, -1.830_8, -1.830_8, -1.830_8, -0.006_8, -0.302_8, -1.430_8, -1.430_8, -1.430_8, -1.100_8, &
-1.100_8, -1.100_8, -1.100_8, 0.300_8, 1.700_8, 1.700_8, 1.700_8, 1.700_8, 1.700_8, 1.228_8, &
-0.660_8, -0.660_8, -0.660_8, -0.660_8, 0.339_8, 2.670_8, 2.670_8, 0.674_8, -2.320_8, -2.320_8, &
-1.356_8, 2.500_8, 2.500_8, 2.500_8, 2.500_8, -1.000_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, &
-2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.500_8, -2.090_8, 1.600_8, 1.600_8, &
1.600_8, 1.600_8, 1.600_8, 2.230_8, 2.500_8, 2.500_8, 1.573_8, -0.590_8, -0.533_8, -0.400_8, &
-0.400_8, -0.400_8, 0.240_8, 0.400_8, 0.400_8, 0.220_8, -0.500_8, -0.500_8, -0.968_8, -1.670_8, &
-1.670_8, -1.670_8, -1.670_8, -0.985_8, -0.300_8, -0.960_8, -2.500_8, -2.220_8, -1.800_8, -1.760_8, &
-1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -1.600_8, -0.820_8, -0.300_8, &
-1.920_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.282_8, 0.020_8, 0.020_8, &
1.155_8, 2.110_8, 2.110_8, 1.805_8, -0.940_8, -0.772_8, 0.180_8, 0.250_8, 0.250_8, 0.250_8, &
0.400_8, 0.580_8, 1.300_8, 0.800_8, 0.300_8, 1.200_8, 1.800_8, 1.640_8, 1.000_8, 1.000_8, &
-0.400_8, -1.000_8, -0.280_8, -0.200_8, -0.200_8, -0.200_8, -0.080_8, 1.000_8, 1.000_8, 1.000_8, &
1.000_8, -1.240_8, -2.200_8, -2.200_8, -2.200_8, 1.800_8, 1.800_8, 1.800_8, 1.800_8, 1.800_8, &
2.790_8, 2.900_8, 2.900_8, 2.900_8, 1.720_8, -3.000_8, -3.000_8, -3.000_8, 1.000_8, 1.000_8, &
1.800_8, 3.000_8, 3.000_8, 1.500_8, 0.500_8, 0.920_8, 1.200_8, 1.280_8, 1.600_8, 1.600_8, &
2.720_8, 3.000_8, 1.945_8, 0.890_8, 0.861_8, 0.600_8, 0.600_8, 0.600_8, 1.288_8, 1.460_8, &
-0.670_8, -2.800_8, -2.800_8, -2.800_8, -2.800_8, -2.060_8, -1.320_8, -1.101_8, 0.870_8, 0.870_8, &
-0.544_8, -1.150_8, -1.150_8, -1.420_8, -1.450_8, -1.765_8, -1.900_8, -1.900_8, -1.900_8, -1.900_8, &
-0.600_8, -0.600_8, -0.744_8, -0.760_8, -0.760_8, -1.230_8, -1.230_8, -1.230_8, -1.230_8, -1.230_8, &
-1.230_8, -1.230_8, -1.230_8, 0.246_8, 0.410_8, 0.978_8, 1.120_8, 1.120_8, 1.120_8, 1.120_8, &
-0.608_8, -0.800_8, -0.020_8, 0.600_8, 1.000_8, 1.000_8, 1.900_8, 2.000_8, 2.000_8, 2.000_8, &
0.950_8, -1.500_8, -1.350_8, -0.750_8, -0.750_8, -0.750_8, -0.750_8, -0.750_8, -0.675_8, -0.500_8, &
-0.500_8, -0.300_8, 1.500_8, 1.500_8, 1.500_8, 1.500_8, 1.500_8, 1.950_8, 2.000_8, 0.600_8, &
-1.500_8, -1.500_8, 1.200_8, 3.000_8, 1.100_8, -0.800_8, -0.980_8, -1.100_8, -2.000_8, -2.000_8, &
-2.000_8, -2.000_8, -2.000_8, -0.926_8, -0.210_8, -0.210_8, -0.210_8, -0.912_8, -0.990_8, -0.621_8, &
0.240_8, 0.240_8, 0.966_8, 1.450_8, 1.450_8, 0.890_8, 0.050_8, 0.050_8, 0.050_8, 0.050_8, &
0.050_8, 0.995_8, 1.100_8, 1.100_8, -1.412_8, -2.040_8, -2.040_8, -2.040_8, -2.040_8, -0.870_8, &
0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, 0.300_8, -0.180_8, -0.300_8, -0.295_8, -0.290_8, &
-0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, -0.290_8, 2.671_8, 3.000_8, &
-0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, -0.970_8, &
-0.970_8, -0.970_8, -0.182_8, 1.000_8, 1.000_8, 1.000_8, 1.600_8, 2.000_8, 2.000_8, 1.040_8, &
0.400_8, 0.128_8, -0.280_8, -0.280_8, -0.280_8, -0.772_8, -1.100_8, -1.100_8, -1.100_8, -1.100_8, &
0.350_8, 0.350_8, 0.350_8, 0.350_8, -1.630_8, -1.630_8, -1.630_8, -1.365_8, -1.100_8, -1.100_8, &
-1.100_8, 0.850_8, 2.800_8, 2.800_8, 2.745_8, 2.250_8, 2.250_8, 2.250_8, 1.340_8, -2.300_8, &
-2.300_8, -2.180_8, -2.000_8, -2.000_8, -0.536_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, &
0.600_8, 0.600_8, 0.600_8, 0.600_8, 0.600_8, -0.840_8, -0.840_8, -0.144_8, 0.900_8, 0.900_8, &
0.900_8, 0.900_8, 0.900_8, 0.900_8, -0.189_8, -0.310_8, -0.310_8, -0.310_8, -0.310_8, -0.310_8, &
0.683_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, 3.000_8, &
3.000_8, 3.000_8, 3.000_8, 2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, 0.800_8, &
2.000_8, 2.000_8, 2.000_8, 0.144_8, -0.320_8, -0.320_8, -0.320_8, 0.608_8, 2.000_8, 0.512_8, &
0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 0.140_8, 1.814_8, &
2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, &
0.215_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -1.570_8, -0.331_8, 0.200_8, &
0.200_8, -0.250_8, -0.700_8, -0.630_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, -0.414_8, &
0.414_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, 0.000_8, 0.630_8, 0.700_8, 0.250_8, -0.200_8, &
-0.200_8, 0.331_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, 1.570_8, -0.215_8, &
-2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, &
-1.814_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, -0.140_8, &
-0.512_8, -2.000_8, -0.608_8, 0.320_8, 0.320_8, 0.320_8, -0.144_8, -2.000_8, -2.000_8, -2.000_8, &
-0.800_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, 2.000_8, -2.000_8, -3.000_8, -3.000_8, &
-3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.683_8, &
0.310_8, 0.310_8, 0.310_8, 0.310_8, 0.310_8, 0.189_8, -0.900_8, -0.900_8, -0.900_8, -0.900_8, &
-0.900_8, -0.900_8, 0.144_8, 0.840_8, 0.840_8, -0.600_8, -0.600_8, -0.600_8, -0.600_8, -0.600_8, &
-0.600_8, -0.600_8, -0.600_8, -0.600_8, -0.600_8, 0.536_8, 2.000_8, 2.000_8, 2.000_8, 2.180_8, 2.300_8, &
2.300_8, -1.340_8, -2.250_8, -2.250_8, -2.250_8, -2.745_8, -2.800_8, -2.800_8, -0.850_8, 1.100_8, &
1.100_8, 1.100_8, 1.365_8, 1.630_8, 1.630_8, 1.630_8, -0.350_8, -0.350_8, -0.350_8, -0.350_8, &
1.100_8, 1.100_8, 1.100_8, 1.100_8, 0.772_8, 0.280_8, 0.280_8, 0.280_8, -0.128_8, -0.400_8, &
-1.040_8, -2.000_8, -2.000_8, -1.600_8, -1.000_8, -1.000_8, -1.000_8, 0.182_8, 0.970_8, 0.970_8, &
0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, 0.970_8, &
-3.000_8, -2.671_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, 0.290_8, &
0.290_8, 0.295_8, 0.300_8, 0.180_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, &
0.870_8, 2.040_8, 2.040_8, 2.040_8, 2.040_8, 1.412_8, -1.100_8, -1.100_8, -0.995_8, -0.050_8, &
-0.050_8, -0.050_8, -0.050_8, -0.050_8, -0.890_8, -1.450_8, -1.450_8, -0.966_8, -0.240_8, -0.240_8, &
0.621_8, 0.990_8, 0.912_8, 0.210_8, 0.210_8, 0.210_8, 0.926_8, 2.000_8, 2.000_8, 2.000_8, &
2.000_8, 2.000_8, 1.100_8, 0.980_8, 0.800_8, -1.100_8, -3.000_8, -1.200_8, 1.500_8, 1.500_8, &
-0.600_8, -2.000_8, -1.950_8, -1.500_8, -1.500_8, -1.500_8, -1.500_8, -1.500_8, 0.300_8, 0.500_8, &
0.500_8, 0.675_8, 0.750_8, 0.750_8, 0.750_8, 0.750_8, 0.750_8, 0.750_8, 1.350_8, 1.500_8, -0.950_8, &
-2.000_8, -2.000_8, -2.000_8, -1.900_8, -1.000_8, -1.000_8, -0.600_8, 0.020_8, 0.800_8, 0.608_8, &
-1.120_8, -1.120_8, -1.120_8, -1.120_8, -0.978_8, -0.410_8, -0.246_8, 1.230_8, 1.230_8, 1.230_8, &
1.230_8, 1.230_8, 1.230_8, 1.230_8, 1.230_8, 0.760_8, 0.760_8, 0.744_8, 0.600_8, 0.600_8, &
```

```
1.900_8,  1.900_8,  1.900_8,  1.900_8,  1.765_8,  1.450_8,  1.420_8,  1.150_8,  1.150_8,  0.544_8,  &
-0.870_8, -0.870_8, 1.101_8,  1.320_8,  2.060_8,  2.800_8,  2.800_8,  2.800_8,  2.800_8,  0.670_8,  &
-1.460_8, -1.288_8, -0.600_8, -0.600_8, -0.600_8, -0.861_8, -0.890_8, -1.945_8, -3.000_8, -2.720_8, &
-1.600_8, -1.600_8, -1.280_8, -1.200_8, -0.920_8, -0.500_8, -1.500_8, -3.000_8, -3.000_8, -1.800_8, &
-1.000_8, -1.000_8, 3.000_8,  3.000_8,  3.000_8,  -1.720_8, -2.900_8, -2.900_8, -2.900_8, -2.790_8, &
-1.800_8, -1.800_8, -1.800_8, -1.800_8, -1.800_8, 2.200_8,  2.200_8,  2.200_8,  1.240_8,  -1.000_8, &
-1.000_8, -1.000_8, -1.000_8, 0.080_8,  0.200_8,  0.200_8,  0.200_8,  0.280_8,  1.000_8,  0.400_8,  &
-1.000_8, -1.000_8, -1.640_8, -1.800_8, -1.200_8, -0.300_8, -0.800_8, -1.300_8, -0.580_8, -0.400_8, &
-0.250_8, -0.250_8, -0.250_8, -0.180_8, 0.772_8,  0.940_8,  -1.805_8, -2.110_8, -2.110_8, -1.155_8, &
-0.020_8, -0.020_8, 0.282_8,  3.000_8,  3.000_8,  3.000_8,  3.000_8,  3.000_8,  3.000_8,  1.920_8,  &
0.300_8,  0.820_8,  1.600_8,  1.600_8,  1.600_8,  1.600_8,  1.600_8,  1.600_8,  1.600_8,  1.600_8,  &
1.760_8,  1.800_8,  2.220_8,  2.500_8,  0.960_8,  0.300_8,  0.985_8,  1.670_8,  1.670_8,  1.670_8,  &
1.670_8,  0.968_8,  0.500_8,  0.500_8,  -0.220_8, -0.400_8, -0.400_8, -0.240_8, 0.400_8,  0.400_8,  &
0.400_8,  0.533_8,  0.590_8,  -1.573_8, -2.500_8, -2.500_8, -2.230_8, -1.600_8, -1.600_8, -1.600_8, &
-1.600_8, -1.600_8, 2.090_8,  2.500_8,  2.500_8,  2.500_8,  2.500_8,  2.500_8,  2.500_8,  2.500_8,  &
2.500_8,  2.500_8,  2.500_8,  2.500_8,  1.000_8,  -2.500_8, -2.500_8, -2.500_8, -2.500_8, 1.356_8,  &
2.320_8,  2.320_8,  -0.674_8, -2.670_8, -2.670_8, -0.339_8, 0.660_8,  0.660_8,  0.660_8,  0.660_8,  &
-1.228_8, -1.700_8, -1.700_8, -1.700_8, -1.700_8, -1.700_8, -0.300_8, 1.100_8,  1.100_8,  1.100_8,  &
1.100_8,  1.430_8,  1.430_8,  1.430_8,  0.302_8,  0.006_8,  1.830_8,  1.830_8,  1.830_8,  1.830_8,  &
1.830_8,  1.830_8,  1.830_8,  1.830_8,  1.302_8,  0.510_8,  0.510_8,  0.510_8,  0.170_8,  -0.340_8, &
-0.340_8, -0.340_8, -0.541_8, -2.350_8, -2.350_8, -2.350_8, -2.350_8, -0.210_8, 3.000_8,  3.000_8,  &
3.000_8,  0.760_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  -0.529_8, -2.230_8, &
-2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, -2.230_8, &
-1.235_8, -0.240_8, -0.240_8, -1.420_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, &
-2.600_8, -0.800_8, 0.400_8,  0.400_8,  0.400_8,  0.400_8,  0.120_8,  0.000_8,  0.000_8,  0.000_8,  &
-0.060_8, -0.200_8, -0.200_8, -0.200_8, -0.036_8, 0.210_8,  0.210_8,  0.210_8,  0.087_8,  -0.200_8, &
-0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, 0.510_8,  0.510_8,  0.510_8,  0.510_8,  -0.300_8, &
-0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, 1.270_8,  2.840_8,  2.840_8,  2.840_8,  2.840_8,  &
2.840_8,  2.840_8,  -0.576_8, -1.430_8, -1.430_8, -1.430_8, -1.430_8, -0.126_8, 1.830_8,  1.830_8,  &
1.830_8,  1.830_8,  1.830_8,  -1.068_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -3.000_8, -0.660_8, &
-0.660_8, -0.660_8, -0.660_8, -0.660_8, -0.136_8, 0.650_8,  0.650_8,  0.650_8,  0.650_8,  0.650_8,  &
0.650_8,  0.650_8,  0.317_8,  -0.460_8, -0.460_8, -0.460_8, -0.460_8, 0.394_8,  0.760_8,  0.760_8,  &
0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.493_8,  -0.130_8, -0.130_8, -0.130_8, -0.130_8, &
-0.130_8, -0.130_8, -0.130_8, 0.683_8,  2.580_8,  2.580_8,  2.580_8,  2.580_8,  2.580_8,  2.580_8,  &
2.580_8,  2.580_8,  2.580_8,  2.580_8,  2.580_8,  2.580_8,  0.812_8,  0.370_8,  0.370_8,  0.370_8,  &
0.370_8,  -0.720_8, -0.720_8, -0.720_8, 0.078_8,  0.420_8,  0.420_8,  0.420_8,  0.420_8,  0.420_8,  &
0.420_8,  0.420_8,  0.828_8,  1.100_8,  1.100_8,  1.100_8,  1.100_8,  0.920_8,  -0.700_8, -0.700_8, &
-0.700_8, -0.700_8, 0.604_8,  0.930_8,  0.930_8,  0.423_8,  -0.760_8, -0.760_8, -0.760_8, -1.732_8, &
-1.840_8, -1.840_8, -1.840_8, -1.840_8, 0.190_8,  0.190_8,  0.190_8,  0.190_8,  0.190_8,  0.190_8,  &
0.190_8,  0.190_8,  2.719_8,  3.000_8,  3.000_8,  2.370_8,  1.740_8,  1.740_8,  1.740_8,  1.740_8,  &
2.059_8,  4.930_8,  4.930_8,  4.930_8,  4.930_8,  4.930_8,  1.714_8,  -0.430_8, -0.430_8, -0.430_8, &
-0.430_8, -0.430_8, -0.430_8, -0.430_8, -1.686_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, -2.000_8, &
-2.000_8, -2.000_8, -2.000_8, -2.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, -4.500_8, &
-2.800_8, 4.000_8,  4.000_8,  4.000_8,  4.000_8,  4.000_8,  4.000_8,  4.000_8,  4.000_8,  4.000_8,  &
4.000_8,  1.090_8,  -0.850_8, -0.850_8, -0.850_8, -1.555_8, -3.200_8, -3.200_8, -3.200_8, -3.200_8, &
-1.048_8, 2.180_8,  2.180_8,  2.180_8,  2.180_8,  0.398_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  &
0.200_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  -0.385_8, -0.450_8, -0.450_8, -0.450_8, -0.450_8, &
-0.450_8, -0.450_8, -0.450_8, -0.450_8, -0.450_8, -0.222_8, -0.070_8, -0.070_8, -0.070_8, -0.070_8, &
-0.070_8, -0.070_8, -0.070_8, -0.083_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, &
0.020_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  2.000_8,  &
-1.240_8, -1.600_8, -1.600_8, -1.240_8, 0.200_8,  0.200_8,  0.200_8,  0.200_8,  0.200_8,  -0.250_8, &
-0.300_8, -0.300_8, -0.300_8, -0.220_8, -0.200_8, -0.200_8, -0.200_8, -0.200_8, -0.140_8, 0.100_8,  &
0.100_8,  0.100_8,  0.100_8,  0.100_8,  0.100_8,  0.100_8,  0.100_8,  0.100_8,  0.100_8,  2.500_8,  &
2.500_8,  2.500_8,  2.500_8,  2.500_8,  -1.406_8, -1.840_8, -1.840_8, -1.840_8, -1.840_8, -1.840_8, &
-1.840_8, -1.840_8, -1.840_8, -0.208_8, 0.880_8,  0.880_8,  0.880_8,  0.880_8,  0.880_8,  0.880_8,  &
0.880_8,  0.880_8,  -0.366_8, -0.900_8, -0.900_8, -0.900_8, -0.102_8, 0.430_8,  0.430_8,  0.430_8,  &
0.430_8,  0.430_8,  0.045_8,  -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, -0.340_8, &
-0.340_8, -0.293_8, 0.130_8,  0.130_8,  0.618_8,  1.350_8,  1.350_8,  1.350_8,  1.350_8,  1.350_8,  &
1.350_8,  1.350_8,  1.350_8,  1.350_8,  1.350_8,  1.350_8,  1.105_8,  0.860_8,  0.860_8,  0.860_8,  &
0.860_8,  0.541_8,  -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, -2.330_8, &
-2.330_8, -2.330_8, -1.230_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.130_8, -0.080_8, &
0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  &
0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.346_8,  0.290_8,  0.290_8,  0.290_8,  0.290_8,  0.297_8,  &
0.360_8,  0.360_8,  0.360_8,  0.360_8,  0.257_8,  -0.670_8, -0.670_8, -0.670_8, -0.403_8, 2.000_8,  &
2.000_8,  2.000_8,  0.692_8,  -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, -2.360_8, &
-2.360_8, -0.962_8, 2.300_8,  2.300_8,  2.300_8,  2.300_8,  2.300_8,  2.300_8,  2.300_8,  2.300_8,  &
-1.772_8, -2.790_8, -2.790_8, -2.790_8, -2.790_8, -2.232_8, 0.000_8,  0.000_8,  0.000_8,  0.000_8,  &
0.228_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  0.760_8,  -0.248_8, &
-0.500_8, -0.500_8, -0.500_8, -0.150_8, 0.000_8,  0.000_8,  0.000_8,  1.800_8,  2.000_8,  2.000_8,  &
1.040_8,  0.400_8,  0.400_8,  0.400_8,  0.400_8,  0.400_8,  0.400_8,  0.400_8,  -0.050_8, -0.500_8, &
-0.500_8, -0.500_8, -0.500_8, -0.500_8, -0.500_8, -0.500_8, 2.335_8,  2.650_8,  2.650_8,  2.650_8,  &
2.650_8,  2.650_8,  2.650_8,  2.650_8,  2.650_8,  2.650_8,  2.650_8,  -0.500_8, -2.600_8, -2.600_8, &
-2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -2.600_8, -0.880_8, -0.450_8, -0.450_8, &
0.192_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  2.760_8,  &
2.760_8,  2.760_8,  2.760_8,  2.760_8,  -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, -0.760_8, &
-0.760_8, -0.760_8, -0.760_8, 0.208_8,  0.450_8,  0.450_8,  0.450_8,  0.450_8,  -0.600_8, -1.300_8, &
-1.300_8, -0.148_8, 0.620_8,  0.620_8,  0.620_8,  0.030_8,  -2.330_8, -2.330_8, -2.330_8, -2.330_8, &
-1.020_8, 0.290_8,  0.290_8,  0.290_8,  0.290_8,  0.290_8,  -1.222_8, -1.600_8, -1.600_8, -1.600_8, &
-1.600_8, -1.600_8, -1.251_8, 1.890_8,  1.890_8,  1.890_8,  1.890_8,  0.217_8,  -0.500_8, -0.500_8, &
-0.500_8, -0.500_8, -0.500_8, -0.500_8, 0.154_8,  1.680_8,  1.680_8,  1.680_8,  1.680_8,  1.680_8,  &
1.680_8,  1.680_8,  1.680_8,  0.632_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  0.370_8,  &
0.370_8,  0.370_8,  2.250_8,  2.720_8,  2.720_8,  2.720_8,  2.720_8,  0.866_8,  0.660_8,  0.660_8,  &
0.660_8,  0.660_8,  -0.012_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, &
-0.300_8, -0.300_8, -0.300_8, -0.300_8, -0.300_8, 0.180_8,  1.300_8,  1.300_8,  1.300_8,  1.300_8,  &
1.300_8,  -0.149_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, -0.770_8, &
-0.770_8, -0.770_8, -0.770_8, -0.770_8, 3.000_8,  3.000_8,  3.000_8,  3.000_8,  3.000_8,  3.000_8,  &
2.630_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  &
2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.260_8,  2.917_8,  2.990_8,  2.990_8,  &
2.990_8,  2.990_8,  2.644_8,  1.260_8,  1.260_8,  1.260_8,  1.260_8,  1.260_8,  1.260_8,  1.260_8,  &
1.260_8,  1.260_8,  1.260_8,  1.260_8,  1.855_8,  2.450_8,  2.450_8,  2.450_8,  2.090_8,  2.050_8,  &
2.050_8,  2.050_8,  2.050_8,  2.050_8,  1.873_8,  1.460_8,  1.460_8,  1.460_8,  1.460_8,  1.218_8,  &
0.250_8,  0.250_8,  0.250_8,  0.325_8,  1.000_8,  1.000_8,  1.000_8,  0.180_8,  -0.640_8, -0.640_8, &
-0.640_8, -0.640_8, -0.640_8, -0.640_8, -1.347_8, -1.650_8, -1.650_8, 0.510_8,  0.750_8,  0.750_8,  &
0.750_8,  0.750_8,  0.126_8,  -2.370_8, -2.370_8, -2.370_8, -2.696_8, -4.000_8, -4.000_8, -4.000_8, &
-4.000_8, -4.000_8, -4.000_8, -4.000_8, 0.770_8,  0.770_8,  0.770_8,  0.770_8,  0.770_8,  0.770_8,  &
0.770_8,  0.770_8,  0.770_8,  -2.515_8, -2.880_8, -2.880_8, -2.880_8, -2.880_8, -2.880_8, -1.120_8, &
```

```
        -1.120_8,  -1.120_8,  -1.120_8,  -1.120_8,  -1.120_8,  -3.000_8,  -3.000_8,  -3.000_8,  -3.000_8,  -3.000_8,  &
        -1.040_8,  -0.550_8,  -0.550_8,  -1.370_8,  -2.600_8,  -2.600_8,  -2.600_8,  -2.600_8,  -2.600_8,  0.048_8,   &
        0.710_8,   0.710_8,   0.563_8,   0.500_8,   0.500_8,   0.500_8,   -1.244_8,  -1.680_8,  -1.680_8,  -1.548_8,  &
        -1.350_8,  -1.350_8,  -1.350_8,  -1.350_8,  -1.350_8,  -1.754_8,  -2.360_8,  -2.360_8,  -2.360_8,  -2.360_8,  &
        -3.164_8,  -5.040_8,  -5.040_8,  -5.040_8,  -5.040_8,  -4.740_8,  -2.040_8,  -2.040_8,  -2.040_8,  -1.724_8,  &
        -1.250_8,  -1.250_8,  -1.233_8,  -1.080_8,  -1.530_8,  -0.012_8,  2.940_8,   1.926_8,   -0.440_8,  -0.440_8,  &
        -2.652_8,  -3.600_8,  -3.070_8,  -2.540_8,  -2.108_8,  2.797_8,   2.144_8,   -2.600_8,  -2.600_8,  0.757_8,   &
        1.130_8,   1.515_8,   1.680_8,   1.680_8,   1.680_8,   1.680_8,   1.680_8,   1.680_8,   1.680_8,   3.546_8,   &
        4.790_8,   4.011_8,   -3.000_8,  -3.000_8,  -3.000_8,  -2.970_8,  -2.700_8,  -2.700_8,  -2.700_8,  -2.700_8,  &
        1.004_8,   1.930_8,   1.930_8,   1.930_8,   1.930_8,   1.930_8,   -2.334_8,  -3.400_8,  -2.585_8,  4.750_8,   &
        4.750_8,   4.750_8,   2.092_8,   0.320_8,   0.320_8,   -0.778_8,  -0.900_8,  -0.900_8,  -0.900_8,  -0.900_8,  &
        -1.596_8,  -3.220_8,  -3.220_8,  -3.220_8,  -3.220_8,  -3.220_8,  -1.492_8,  -1.300_8,  -1.300_8,  -1.300_8,  &
        -1.300_8,  -1.300_8,  -2.600_8,  -2.600_8,  -0.760_8,  -0.300_8,  -0.300_8,  -0.300_8,  -0.155_8,  -0.010_8,  &
        -0.010_8,  -0.010_8,  -0.010_8,  -0.126_8,  -0.300_8,  -0.300_8,  1.239_8,   1.410_8,   1.410_8,   1.410_8,   &
        1.410_8,   1.135_8,   0.860_8,   0.860_8,   0.860_8,   0.860_8,   0.316_8,   -1.860_8,  -1.860_8,  -1.860_8,  &
        -1.860_8,  -1.860_8,  -1.455_8,  -1.410_8,  -1.410_8,  -1.410_8,  -1.410_8,  -1.680_8,  -1.950_8,  -1.950_8,  &
        0.122_8,   0.640_8,   0.540_8,   -0.360_8,  -0.360_8,  -0.360_8,  -0.360_8,  -0.360_8,  0.249_8,   0.510_8,   &
        0.510_8,   0.510_8,   0.510_8,   -0.456_8,  -1.100_8,  -1.100_8,  -0.868_8,  0.060_8,   0.060_8,   0.060_8,   &
        0.060_8,   0.060_8,   -0.090_8,  -0.240_8,  -0.240_8,  -0.240_8,  -0.240_8,  -0.240_8,  0.036_8,   0.220_8,   &
        0.220_8,   0.745_8,   1.970_8,   1.970_8,   1.970_8,   1.970_8,   1.970_8,   1.970_8,   1.970_8,   -0.100_8,  &
        -1.131_8,  -1.220_8,  -1.220_8,  1.006_8,   1.960_8,   1.960_8,   0.472_8,   -0.520_8,  -0.520_8,  -0.520_8,  &
        -0.520_8,  -0.520_8,  -0.520_8,  0.400_8,   0.400_8,   0.400_8,   0.400_8,   0.400_8,   0.400_8,   0.400_8,   &
        1.582_8,   2.370_8,   2.370_8,   2.370_8,   2.370_8,   2.370_8,   2.370_8,   2.370_8,   1.998_8,   0.510_8,   &
        0.510_8,   0.510_8,   0.221_8,   -2.380_8,  -2.380_8,  -2.380_8,  -0.292_8,  0.230_8,   0.230_8,   0.288_8,   &
        0.520_8,   0.520_8,   0.520_8,   1.600_8,   1.870_8,   0.586_8,   -2.410_8,  -2.410_8,  -2.410_8,  -2.410_8,  &
        -2.410_8,  -0.996_8,  -1.122_8,  -2.830_8,  -2.830_8,  -2.830_8,  -2.830_8,  -2.830_8,  -2.830_8,  -2.830_8,  &
        -2.830_8,  -2.247_8,  3.000_8,   3.000_8,   3.000_8,   3.000_8,   3.000_8,   3.000_8,   -0.576_8,  -2.960_8,  &
        -2.960_8,  -2.960_8,  -2.960_8,  -2.960_8,  -1.786_8,  2.910_8,   2.910_8,   2.910_8,   2.910_8,   2.910_8,   &
        -0.830_8,  -0.830_8,  -0.830_8,  -0.830_8,  -0.768_8,  -0.520_8,  -0.520_8,  -0.520_8,  -0.520_8,  -0.520_8,  &
        2.639_8,   2.990_8,   2.990_8,   2.990_8,   2.990_8,   2.990_8,   -1.162_8,  -2.200_8,  -2.200_8,  -2.200_8,  &
        -2.200_8,  -2.200_8,  -2.200_8,  -2.200_8,  -2.200_8,  -2.200_8,  1.080_8,   1.080_8,   1.080_8,   -0.712_8,  &
        1.780_8,   4.000_8,   4.000_8,   4.000_8,   -1.400_8,  -5.000_8,  -5.000_8,  -3.135_8,  -1.270_8,  -1.270_8,  &
        -0.494_8,  -0.300_8,  0.490_8,   0.490_8,   0.490_8,   0.490_8,   0.490_8,   0.669_8,   2.280_8,   2.280_8,   &
        2.280_8,   2.280_8,   2.280_8,   2.280_8,   2.280_8,   0.228_8,   0.000_8,   0.000_8,   0.000_8,   0.000_8,   &
        0.000_8                                                                                                        &
    /

end module TESTCYCLE
```

```
! ************************************************************
!   MODULE HVTRANSFC  データ構造定義
! ************************************************************

module hvtransfc

!   ------------------------------------------------------------
!   TYPE SFCMAP
!   ------------------------------------------------------------
    type sfcmap
        integer :: ndata
        real(8), pointer :: rev(:)
        real(8), pointer :: tq(:)
        real(8), pointer :: fc(:)
    end type sfcmap

!   ------------------------------------------------------------
!   TYPE SGRIDMAP
!   ------------------------------------------------------------
    type sgridmap
        integer :: nx
        integer :: ny
        real(8), pointer :: rev(:)
        real(8), pointer :: tq(:)
        real(8), pointer :: fc(:,:)
    end type sgridmap

!   ------------------------------------------------------------
!   TYPE SGRIDENGMAP エンジントルクマップ
!   ------------------------------------------------------------
    type sgridengmap
        integer :: tqn
        integer :: thrn
        integer, pointer :: n(:)
        real(8), pointer :: thr(:)
        real(8), pointer :: r(:,:)
        real(8), pointer :: t(:,:)
    end type sgridengmap

!   ------------------------------------------------------------
!   TYPE SSPEC  車両諸元
!   ------------------------------------------------------------
    type sspec
        real(8) :: mua                                  ! 空気抵抗係数, N/(km/h2)
        real(8) :: mur                                  ! 転がり抵抗係数, N/kg
        real(8) :: rt                                   ! タイヤ有効径, m
        real(8) :: bw, bh                               ! 全幅, 全高, m
        real(8) :: w0                                   ! 空車質量, kg
        real(8) :: wld                                  ! 最大積載量, kg
        real(8) :: wt                                   ! 試験時車両質量 kg
        real(8) :: gvw                                  ! GVW, kg
        real(8) :: fgr                                  ! デフ比
        real(8) :: efgr                                 ! デフ伝達効率
        real(8) :: ric                                  ! 高速モード走行割合 (0 - 1)
        real(8) :: pdwt                                 ! 回転部分相当質量（被駆動ギヤ～タイヤ, 空車質量の7%）
        real(8) :: pdwe                                 ! 回転部分相当質量（エンジン～駆動側ギヤ, 空車質量の3%）
        integer :: crew                                 ! 定員, 人
        character (len=512) :: vhname                   ! 車両名コメント
        character (len=512) :: tpname                   ! 標準車両諸元の分類名
        character (len=512) :: specf                    ! 車両諸元データのファイル名
        character (len=512) :: engf                     ! エンジン諸元データのファイル名
        character (len=512) :: tmf                      ! 変速機諸元データのファイル名
        character (len=512) :: ctrlf                    ! 変速制御データのファイル名
        character (len=512) :: tcf                      ! トルクコンバータ諸元のファイル名
    end type sspec

!   ------------------------------------------------------------
!   TYPE SENG  エンジン諸元
!   ------------------------------------------------------------
    type seng
        real(8) :: nex                                  ! 有負荷最高回転数, rpm
        real(8) :: nrate                                ! 定格回転数, rpm
        real(8) :: nidle                                ! アイドル回転数, rpm
        character (len=1024) :: engf, egmapf, fcmapf
        type (sfcmap) :: fcm                            ! 燃費マップ 生データ
        type (sfcmap) :: fcmidl                         ! 燃費マップ アイドル回転
        type (sgridmap) :: gfcm                         ! 燃費マップ 格子データ
        type (sgridengmap) :: gegmp                     ! エンジントルクマップ
    end type seng

!   ------------------------------------------------------------
!   TYPE SGDAT  ギヤデータ
!   ------------------------------------------------------------
    type sgdat
        real(8) :: gr                                   ! 変速比
        real(8) :: egr                                  ! 伝達効率
        real(8) :: dw                                   ! 回転部分相当質量, kg
    end type sgdat

!   ------------------------------------------------------------
!   TYPE stm  変速機諸元
!   ------------------------------------------------------------
    type stm
        integer :: grs                                  ! 発進ギヤ
        integer :: ngr                                  ! ギヤ段数
```

```
        type (sgdat), pointer :: g(:)                          ! ギヤデータ
    end type stm

!   ------------------------------------------------------------
!   TYPE SCYCLE  テストサイクル
!   ------------------------------------------------------------
    type scycle
        integer :: ndat                                        ! テストサイクルのデータ点数
        real(8), pointer :: v(:)                               ! 車速, km/h
        real(8), pointer :: g(:)                               ! 勾配, %
    end type scycle


!   ------------------------------------------------------------
!   TYPE SCRV
!   点(X,Y)の集合として表される曲線データを定義
!   トルク曲線, シフトマップなどの入力として使用
!   ------------------------------------------------------------
    type scrv
        integer :: n                                           ! データ点数
        real(8), pointer :: x(:)                               ! X軸データ
        real(8), pointer :: y(:)                               ! Y軸データ
        integer, pointer :: l(:)                               ! Y2軸データ
    end type scrv


!   ------------------------------------------------------------
!   TYPE SSHIFTMAP  変速点/LUマップ
!   ------------------------------------------------------------
    type sshiftmap
        integer :: ngr                                         ! ギヤ段数
        type(scrv), pointer :: g(:)                            ! ギヤ毎の開度-速度線
    end type  sshiftmap


!   ------------------------------------------------------------
!   TYPE sptq  オイルポンプ損失トルク
!   ------------------------------------------------------------
    type sptq
        integer :: ngr                                         ! ギヤ段数
        type(scrv), pointer :: g(:)                            ! ギヤ毎の回転数-損失トルク曲線
    end type sptq


!   ------------------------------------------------------------
!   TYPE stc  トルクコンバータ諸元
!   ------------------------------------------------------------
    type stc
        character (len=1024) :: tcf                            ! トルコン諸元 入力ファイル名
        character (len=1024) :: cfdname                        ! 容量係数(駆動) データ名
        character (len=1024) :: cfbname                        ! 容量係数(逆駆動) データ名
        character (len=1024) :: trdname                        ! トルク比(駆動) データ名
        character (len=1024) :: trbname                        ! トルク比(逆駆動) データ名
        character (len=1024) :: ptqname                        ! オイルポンプ損失トルクデータ名
        type (sptq) :: ptq                                     ! オイルポンプ損失トルク
        type (scrv) :: cfd                                     ! 容量係数(駆動) 格納変数
        type (scrv) :: cfb                                     ! 容量係数(逆駆動) 格納変数
        type (scrv) :: trd                                     ! トルク比(駆動) 格納変数
        type (scrv) :: trb                                     ! トルク比(逆駆動) 格納変数
    end type



!   ------------------------------------------------------------
!   TYPE VEHOUT   車両状態量格納変数
!   ------------------------------------------------------------
    type vehout
        real(8) :: v, ne, te, fc                               ! 速度,回転,トルク,燃料消費
        real(8) :: maxt, nne, nte                              ! Maxトルク, 正規化回転数&トルク
        real(8) :: opn                                         ! アクセル開度 %
        integer :: s                                           ! ギヤ位置
        integer :: lu                                          ! ロックアップ 0:off, 2:on
        integer :: tstop                                       ! 停止時間カウンタ
        real(8) :: npera                                       ! プロペラシャフト回転数
        integer :: c
        real(8) :: tp, tt, top                                 ! 入力軸, 出力軸トルク, オイルポンプ損失トルク
        real(8) :: sr, tr, cf, nt                              ! 速度比, トルク比, 容量係数, タービン回転数
        integer :: verrflg                                     ! 速度追従不能表示 (0:追従,1:追従不可)
    end type


!   ------------------------------------------------------------
!   TYPE sctr シフト制御特性データセット
!   ------------------------------------------------------------
    type sctr
        type ( sshiftmap ) :: lon, loff                        ! ロックアップマップ
        type ( sshiftmap ) :: up, dwn                          ! 変速マップ
        integer :: swnidl                                      ! ニュートラルアイドルオプション0:なし,1:あり
        integer :: sw_nidl_start                               ! 始動時25秒間のニュートラルアイドルの有効(1)/無効(0)
        integer :: time_nidl, time_nidl_start                  ! ニュートラルアイドル作動まで待ち時間, 始動時の待ち時間(sec)
        real(8) :: sr_nidl                                     ! ニュートラルアイドル時の速度比 (0～1)
        character (len=1024) :: cfnm_nidl                      ! 停止時容量係数のファイル名
        type (scrv) :: cf_nidl                                 ! 停止時容量係数 格納変数
        integer :: swistop                                     ! アイドルストップ有無 0:なし,1:あり
        integer :: time_iss, time_iss_start                    ! アイドルストップ作動まで待ち時間, 始動時の待ち時間(sec)
        integer :: sw_iss_start                                ! 始動時アイドルストップの有効(1)/無効(0)
```

```
        character (len=1024) :: lonnm, loffnm                   ! ロックアップマップのファイル名
        character (len=1024) :: upnm, dwnnm                     ! 変速マップのファイル名
    end type sctr


end module hvtransfc
```