



FORTTRAN Code for the PSCAD Telecommunication Model Presented in the Paper: “A Realistic Telecommunication Model for Electromagnetic Transient Simulations and Control Assessment of Multi-Terminal VSC-HVDC Networks in PSCAD/EMTDC”

Document Version
Final published version

[Link to publication record in Manchester Research Explorer](#)

Citation for published version (APA):

Carmona Sanchez, J., Green, P., Barnes, M., & Marjanovic, O. (2018, Sep 5). FORTRAN Code for the PSCAD Telecommunication Model Presented in the Paper: “A Realistic Telecommunication Model for Electromagnetic Transient Simulations and Control Assessment of Multi-Terminal VSC-HVDC Networks in PSCAD/EMTDC”.

Citing this paper

Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights

Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy

If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [<http://man.ac.uk/04Y6Bo>] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.



FORTRAN Code for the PSCAD Telecommunication Model Presented in the Paper:

“A Realistic Telecommunication Model for Electromagnetic Transient Simulations and Control Assessment of Multi-Terminal VSC-HVDC Networks in PSCAD/EMTDC”

Carmona-Sanchez, J, Green, P R, Barnes, M, Marjanovic, O

The University of Manchester, School of EEE

ACDC Conference 2019

```
! Define variables
! Ssamf = Signal sampling frequency
  #Local real Ssamf
! Ssamf = Signal sampling time
  #Local real Ssamf
! Ts = Time sampling (for communication between interactions)
  #Local real Ts
  #Local real Ts0
! ft = Future time (i.e. current time plus time delay specified at sampling time)
  #Local real ft
  #Local real ft0
! Time delay
  #Local real Td
! Pointers
  #Local integer Pointer
  #Local integer Pointer0
  #Local integer Pointer2
  #Local integer Pointer20
! Time - Read Buffer
  #Local real TrB
  #Local real TrB0
! Data - Read from Buffer
  #Local real Tsend
  #Local real DAT
  #Local real DAT0
  #Local real Tread

! Initial Conditions
  Ssamf = $Sf ! In Hertz
  Ssamf = 1/Ssamf
  Td=$Tdelay

! Retrieve values from the previous time-step
  Ts0 = STORF(NSTORF)
  ft0 = STORF(NSTORF+1)
  TrB0 = STORF(NSTORF+2)
  DAT0 = STORF(NSTORF+3)
  Pointer0 = STORI(NSTORI)
  Pointer20 = STORI(NSTORI+1)
! End retrieving values

!*****
!Writing to the buffer
!*****
If (TIME .EQ. 0) then ! Only for initialization purposes if time = 0s
  $Out=$In ! The signal is sampled and passed to the output (for display purposes only)
  Ts0 = TIME ! Time at which the signal was sampled
  ft0 = TIME + Td ! Future time at which the signal will be the output for the channel
  Pointer0 = 1 ! Place in the buffer
  open (unit = 1, file = 'Buffer.txt', STATUS='old', action='write')
  write (1,*) TIME, $In, ft0 ! Writing the actual time, the sampled signal and the future
time at which it will be read from the buffer
  Close (1)
end if

If ((TIME-Ts0) .GT. Ssamf) then ! Sampling and writing code for times greater than 0s
  $Out=$In ! The signal is sampled and passed to the output (for display purposes only)
  Ts = TIME ! Time at which the signal was sampled
  ft = TIME + Td ! Future time at which the signal will be the output for the channel
  Pointer = Pointer0+1 ! Place in the buffer
  open (unit = 1, file = 'Buffer.txt', STATUS='old', action='write', position='append')
```

```

        write (1,*) TIME, $In, ft ! Writing the actual time, the sampled signal and the future
                                time at which it will be read from the buffer
    Close (1)
else
Ts = Ts0
ft = ft0
Pointer = Pointer0
end if

!*****
! Reading from the buffer
!*****
If (TIME .EQ. 0) then ! Only for initialization purposes if time = 0s
    TrB0 = ft
    Pointer20 = Pointer0
    DAT0=$In
end if

If (TrB0 .GT. ft) then ! This part of the code actualizes the time to read the buffer (TrB) if
the actual data entering the buffer has a smaller future time (ft). Also actualizes the
pointer
    TrB = ft
    Pointer2 = Pointer
else
    TrB = TrB0
    Pointer2 = Pointer20
end if

If ((TIME-TrB0) .GT. 0) then ! Reading from buffer
    open (unit = 1, file = 'Buffer.txt', STATUS='old', action='read')
    do i=1, Pointer2-1
        read (1,*) !Reads the buffer from line 1 to the previous line where the data is
    end do
    read (1,*) Tsend, DAT, Tread ! Actual data to be read
    $Outd=DAT ! Outputs the data (this is the delayed data)
    read (1,*) Tsend, DAT, Tread ! Reads the next line to actualize the future time to read
                                the buffer (TrB)
    TrB = Tread ! Next time to read the buffer
    Pointer2 = Pointer2+1 ! Pointer that specifies the position of the data for the next
                            time to read the buffer
    do i=1, Pointer-Pointer2 ! Reads the rest of the lines in the buffer to check if there
                            is a newest value for (TrB)
        read (1,*) Tsend, DAT, Tread
        If (TrB .GT. Tread) then ! In case there is a newest value then actualize TrB
            and pointer
                TrB = Tread
                Pointer2 = Pointer2+i
            end if
    end do
    Close (1)
else
DAT=DAT0
end if

! Store the output values
STORF(NSTORF) = Ts
STORF(NSTORF+1) = ft
STORF(NSTORF+2) = TrB
STORF(NSTORF+3) = DAT
STORI(NSTORI) = Pointer
STORI(NSTORI+1) = Pointer2

! Increment Memory
NSTORF = NSTORF+4
NSTORI = NSTORI+2

```