AVDS© - Matlab®

Toolbox

Reference Manual

May 2001

Table of Contents

Introduction

AVDS is a software tool that engineers can use on low-cost workstations, that provides a graphical environment for interactive simulation of user-defined vehicle models and animation of saved vehicle trajectory data. This enables engineers to implement their designs and simulations in a more realistic environment and give them a much clearer insight into the working of their control system design and vehicle models.

The AVDS-Matlab Toolbox provides Matlab functions, Simulink blocks and AVDS files that enable the user to make connections between simulations running in Matlab and AVDS. There are three connection types available between AVDS and Matlab. They are they are through:

- ***Interactive Simulation* –** a simulation of a user-defined vehicle and flight control system (FCS), that runs *near real-time*[*], while displaying the vehicle's motion graphically and accepts user inputs through an external device such as a joystick or mouse.
- ***Data Playback*** – a function that graphically animates one or more vehicles based on trajectory data that has been saved from any source, i.e. batch simulation, flight test or AVDS simulation.
- ***Network Connection* –** a form of interactive simulation or data playback where users can send AVDS data packets across the local Ethernet connection to AVDS. AVDS uses the positions, angular orientations and other information supplied in the packets to animate the vehicle(s).

All three of these connection types are implemented in the AVDS-Matlab Toolbox. The choice of the connection type depends on the type of simulation as shown in Table 1.

Table 1: Recommended Selection of AVDS-Matlab tools based on simulation requirement.

Simulation Requirement	Interactive Simulation	Data Playback	Network Connection
near real-time	●		●
non-real-time		●	
interactive	●		●**
multiple Vehicles		●	●
multiple computers			●

[*] ***Terminology Note***: The term "***near real-time***" is used to indicate that these simulations will run close to real-time, but the closeness to real-time will depend on the computer hardware and the complexity of the simulations. To check how close to real-time the simulations are running one can develop a metric by comparing the elapsed-time from the AVDS-Matlab Toolbox functions to the elapsed simulation time calculated during the simulation.

[**] AVDS is not currently configured to pass user inputs across the network. To run a simulation interactively through the Network Connection, it is necessary for the user to make provision for user inputs on the computer that the simulation is running on.

Installation

To install the AVDS-Matlab Toolbox follow these steps:

1) Start Matlab and change to the "toolbox" directory. For example:

```
cd 'C:\Program Files\AVDS\Utilities\MatlabToolbox\toolbox'
```

2) Run the install.m script. The command is:

```
install
```

NOTE: The next two steps are necessary only for versions of AVDS that were distributed without the AVDS-Matlab toolbox. For more information see the file:

"MATLABToolbox\Toolbox\RequiredAVDSFiles**ReadMe.RequiredAVDSFiles.txt**"

3) Copy the files in the "MatlabToolbox\toolbox\RequiredAVDSFiles**model**"

and "MatlabToolbox\toolbox\RequiredAVDSFiles**userfiles**" directories

into "AVDS**model**" and "AVDS**userfiles**" directories.

4) Edit the file "AVDS\model**modelcap.txt**" to add the names of the model files, i.e. add the lines:

```
Matlab_AC.dll
Matlab_FCS.dll
```

This will make the functions and blocks of the AVDS-Matlab Toolbox available in the Matlab path. The AVDS-Matlab Toolbox Simulink blocks will be available through the Simulink Library Browser.

Examples

In the directory "MatlabToolbox**Examples**" there are subdirectories that contain examples of the use of all of the functions and blocks in the AVDS-Matlab Toolbox.

Interactive Simulation Examples

The " MATLABToolbox\Examples**SimulationExample**" suddirectory contains four examples that demonstrate near real-time interactive simulation using AVDS and Matlab/Simulink. There are examples of implementing a flight control system in Matlab, "***AVDS_FCS_Matlab.m***", and in Simulink, "***ExampleFCS.mdl***", with the aircraft model running in AVDS. Also contained in the directory are examples of running the vehicle model in Matlab, "***AVDS_AC_Matlab.m***", and Simulink, "***ExampleAC.mdl***", with the FCS model in AVDS. For more information on these examples and steps for running them, see the file "MATLABToolbox\Examples\SimulationExample**ReadMe.SimulationExample.txt**", type `help filename` for help on a function or read the comments provided in the files.

Network Examples

The "MATLABToolbox\Examples**NetworkExample**" directory contains a Matlab function, "***ExampleSend.m***" that demonstrates sending packets for multiple vehicles from Matlab to AVDS. The function "***ExampleReceive.m***" demonstrates receiving packets in Matlab from AVDS. For Simulink, the directory contains the model "***NetworkSendExample.mdl***" That demonstrates sending AVDS packets for multiple vehicles. Both the Matlab and Simulink examples synchronize the vehicle models to near real-time using the timing functions from the toolbox. For more information on these examples and steps for running them, see the file "MATLABToolbox\Examples\NetworkExample**ReadMe.NetworkExample.txt**", type `help filename` for help on a function or read the comments provided in the files.

Playback Examples

The "MATLABToolbox\Examples**PlaybackExample**" directory contains a Matlab example, "**SampleAVDSPlayback.m**", and a Simulink model example, "**PlaybackExample.mdl**", that demonstrate saving data for AVDS Playback for multiple vehicles. For more information on these examples and steps for running them, see the file,
"MATLABToolbox\Examples\PlaybackExample**ReadMe.PlaybackExample.txt**", type `help filename` for help on a function or read the comments provided in the files.

Source Code

The directory "MatlabToolbox\toolbox**SourceCode**" contains subdirectories that contain Microsoft Visual C++ Version 6.0 projects that were used to build the Mex files and S-Function DLLs for the AVDS/Matlab toolbox. For those developers not using MSVC++, makefiles were exported for all of the projects to aid in re-building the DLLs. The software contained in the subdirectories is:

"MatlabToolbox\toolbox\SourceCode**SourceCodeSimulation**" - Contains the files necessary to build the interactive simulation DLLs for both the S-Functions and Mex function.

"MatlabToolbox\toolbox\SourceCode**SourceCodeNetwork**" - Contains the files necessary to build the networking function DLLs for both the S-Function and Mex function.

"MatlabToolbox\toolbox\SourceCode**SourceCodePlayback**" - Contains the files necessary to build the playback function DLLs for S-Functions.

"MatlabToolbox\toolbox\SourceCode**TimeFunctions**" - Contains the files necessary to build the timing function library, which is used to retrieve the elapsed time from the Computer.

Toolbox Functions and Blocks

Matlab M-Functions

SimulationConnection

Description:

This function provides a two-way communications connection between Matlab and AVDS through shared memory. Depending on the entry for the **ConnectionType** argument, this function can be configured to communicate to either the "*MATLAB_FCS*" or "*MATLAB_AC*" user defined simulation block in AVDS.

Syntax:

```
[] = SimulationConnection(Mode,…)
```

1. **AC/FCSInitialization**: `SimulationConnection('OpenConnection', ConnectionID, ConnectionType)`
2. **Send To AVDS**: `SimulationConnection('SendData',ConnectionID,VectorToAVDS)`
3. **From AVDS**: `[VectorFromAVDS] =SimulationConnection('ReceiveData', ConnectionID)`
4. **Close Connections**: `[] = SimulationConnection('CloseConnections')`

Arguments:

- ***Mode*** - defines the mode of operation. Valid values are:

  `'OpenConnection'` – establishes the shared memory connection between AVDS and Matlab.

  `'SendData'` – sends a data vector to AVDS through the shared memory connection.

  `'ReceiveData'` - receives a data vector from AVDS through the shared memory connection.

  `'CloseConnections'` – close all open shared memory connections.

- ***ConnectionID* –** unique numerical ID for this connection.
- ***ConnectionType* –** Type of connection to be establish with AVDS. Valid values for this parameter are:

  `'AC'` – connect to the "*MATLAB_AC*" user defined simulation block in AVDS.

  `'FCS'` - connect to the "*MATLAB_FCS*" user defined simulation block in AVDS.

- ***VectorToAVDS* –** sent to AVDS during `'SendData'` call to the function. The use of this argument changes based on the value of **ConnectionType** during the `'OpenConnection'` call:

  '`FCS`': **VectorToAVDS:**

Element	Name	Definition
1. – 3.	**ElevatorCmd, AileronCmd, RudderCmd**	The elevator, aileron and rudder command from the Matlab FCS to AVDS. The nominal values for these signals in AVDS are in the range, ***[-1.0,1.0].***
4.	**ThrottleCmd**	The throttle command from the Matlab FCS. The nominal values for the throttle in AVDS are in the range, ***[0.0,1.0].***
5. - 14.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

'AC': VectorToAVDS:

Element	Name	Definition
1. – 3.	**xpos, ypos, zpos**	The position of the vehicle along the North and East axes, and the altitude, respectively. These values are in feet.
4. – 6.	**xrot, yrot, zrot**	The angular orientation of the vehicle given as Eurler angles in radians. These angles are Phi, Theta and Psi, respectively.
7.	**alpha**	The angle of attack in radians.
8.	**beta**	The sideslip angle in radians.
9.	**G**	Number of G's.
10.	**Velocity**	Total velocity in feet/second.
11.	**M**	Mach number.
12.	**DeltaRudder**	Rudder deflection in radians.
13.	**DeltaElevator**	Elevator deflection in radians.
14.	**DeltaAileron**	Aileron deflection in radians.
15.	**eng**	The engine level. Must in the range of ***[0,1]***.
16. - 25.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

- ***VectorFromAVDS* –** returned from the function during a `'ReceiveData'` call. The use of this vector changes based on the value of **ConnectionType** during the `'OpenConnection'` call. Note: the definitions in these tables are the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.

'FCS': VectorFromAVDS:

Element	Name	Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2. - 4.	**StickPitch, StickRoll. StickYaw**	User inputs from the input device, i.e. joystick, mouse, keyboard. These inputs are in the range ***[-1.0,1.0]***.
5.	**StickThrottle**	User inputs from the throttle. This input are in the range ***[0.0,1.0]***.
6. – 8.	**P, Q, R**	The vehicle's angular rates about the x, y and z body axes, respectively. These values are in radians.
9.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
9. - 19.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

'AC': VectorFromAVDS:

Element	Name	Definition
1. – 3.	**xpos, ypos, zpos**	The position of the vehicle along the North and East axes, and the altitude, respectively. These values are in feet.
4. – 6.	**xrot, yrot, zrot**	The angular orientation of the vehicle given as Eurler angles in radians. These angles are Phi, Theta and Psi, respectively.
7.	**alpha**	The angle of attack in radians.
8.	**beta**	The sideslip angle in radians.
9.	**G**	Number of induced G's.
10.	**Velocity**	Total velocity in feet/second.
11.	**M**	Mach number.
12.	**DeltaRudder**	Rudder deflection in radians.
13.	**DeltaElevator**	Elevator deflection in radians.
14.	**DeltaAileron**	Aileron deflection in radians.
15.	**eng**	The engine level. Must in the range of ***[0,1]***.
16.. - 25.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

Usage:

To communicate between MATLAB and AVDS first open the communications connection with a call to **SimulationConnection** using the syntax of #1 above. Next use #2 and #3 to accomplish the communications. Finally use #4 to close all of the connections.

Examples:

The SimulationConnection function is used by the examples "**AVDS_AC_Matlab.m**" and "**AVDS_FCS_Matlab.m**".

SaveAVDSData

Description:

Used to manage data for output to an AVDS playback file.

Syntax:

```
[OutputData] = SaveAVDSData(Action,InputData,Entity)
```

Arguments:

- ***Action*** - defines the mode of operation. Valid values are:
  1. `'Clear'` - removes any stored data, entities, and latitude/longitude data.
  2. `'AddEntity'` - adds a new entity to the stored data. (Optional, defaults to one entity)
  3. `'AddLatLong'` - adds latitude and longitude starting position to the file. This is an Optional entry in the playback files. The default is not to user initial latitude and longitude. Without the initial latitude and longitude AVDS uses selected airport as the starting position.
  4. `'Store'` - saves one or more rows of data for the given entity. New rows of data must have the same number of columns as existing data for any given entity.
  5. `'WriteEntity'` - write the stored data to a file with the given file name, see below.
- ***InputData*** - the use of this argument changes based on the value of 'Action':

  `'Clear'`: InputData is not used.

  `'AddEntity'`: InputData is not used.

  `'AddLatLong'`: InputData contains a two element row/column vector. The first element contains the starting latitude (degrees decimal in the range ***[-90.0, 90.0]***) and the second element contains the starting longitude (degrees decimal in the range ***[-180.0, 180.0]***).

  `'Store'`: InputData contains a row vector or matrix of data for AVDS to playback.

  `'WriteEntity'`: InputData contains the name of the file for the data.
- ***Entity*** - the ID number of the entity to perform the 'Action' on. (Defaults to Entity=1)
- ***OutputData*** - the value of this changes based on the value of 'Action

  `'Clear'`: [].

  `'AddEntity'`: The ID number of the entity just added.

  `'AddLatLong'`: [].

  `'Store'`: [].

  `'WriteEntity'`: The name of the file saved.

Usage -

SaveAVDSData is used in conjunction with WritePlaybackInit to produce the files necessary for AVDS to load and configure dynamics data for playback. SaveAVDSData is used to save the data files and WritePlaybackInit is used to save the playback configuration file. AVDS is capable of playing back data for multiple vehicles (entities), so SaveAVDSData contains the functionality necessary to save multiple playback data files.

AVDS playback data files are ASCII (text) files where the data variables are in columns with the rows representing slices of time. The columns can be in any order and the only requirement on the content/number of columns is that there must be at least one column that contain time information, elapsed or delta, in seconds. If the column order is to be changed, the **WritePlaybackInit** function

should be modified to reflect the changes. Alternatively, AVDS's Playback Configuration dialog window can be used to create a new playback initialization file.

The default column order for AVDS is:

TimeSeconds, NorthPositionFeet, EastPositionFeet, DownPositionFeet, XRotationPsiDeg, YRotationThetaDeg, ZRotationPsiDeg, craftmask, crafttype, EngineLevel, AlphaDeg, BetaDeg, G, VFtSec, MachNumber, DeltaSurface01Rad, DeltaSurface02Rad, DeltaSurface03Rad, DeltaSurface04Rad, DeltaSurface05Rad, DeltaSurface06Rad, DeltaSurface07Rad, DeltaSurface08Rad, DeltaSurface09Rad, DeltaSurface10Rad, DeltaSurface11Rad, DeltaSurface12Rad, DeltaSurface13Rad, DeltaSurface15Rad, DeltaSurface15Rad, WeaponsSelect, WeaponsLaunch, Explode

Note: For the definitions of these variables see the AVDS User's Manual, Table 5-1 Dynamic Items.

Examples -

For one entity the user calls SaveAVDSData in the following sequence:

```
SaveAVDSData('Clear'); %this clears any old data and resets number of entities
SaveAVDSData('AddLatLong',[37.793625,-122.32856]); %this is optional
       ...
SaveAVDSData('Store',DataRow); %this is repeated as many times as required.
       ...
SaveAVDSData('WriteEntity','MyFile.save.dat');
                                    %writes the latitude/longitude and stored data to the file
```

For more that one entity the user calls SaveAVDSData in the following sequence:

```
SaveAVDSData('Clear'); %this clears any old data and resets number of entities
for (i = 1:NumberEntities),
       SaveAVDSData('AddEntity');
end;
for (i = 1:NumberEntities),
       SaveAVDSData('AddLatLong',[37.793625,-122.32856]); %this is optional
end;
. . .
for (iEntities = 1:NumberEntities),   %this is repeated as many times as required.
       SaveAVDSData('Store',DataRow{iEntities},iEntities);
end;
       ...
for (iEntities = 1:NumberEntities),
       SaveAVDSData('WriteEntity',EntityFileNames(iEntities),iEntities);
                    %writes the latitude/longitude
                    %and stored data to the file
end;
```

WritePlaybackInit

Description:

This function writes an initialization file that configures AVDS playback

Syntax:

```
[] = WritePlaybackInit(InitFileName, Section, DataFileName)
```

Arguments:

- ***InitFileName*** - The name of the initialization file to which the information is written.
- ***Section*** - Section of the initialization file to write. Valid choices are:

  `'Default'` - The Default section opens the file erases the contents, if any, and writes the general initialization information to the file.

  `'Vehicle01'` - The Vehicle01 section opens the file and appends the initialization information for one entity based on the default AVDS playback data file format

  `'VehicleSimple'` - The VehicleSimple section opens the file and appends the initialization information for one entity based on the default AVDS playback data file format. Only write entries for time, linear positions and angular orientations.

  `'UserDefined'` - The user can define custom section definitions based on one's own data file formats. New Sections can be defined by adding new cases to the switch statement in the function.
- ***DataFileName*** - This is the name of the file that AVDS will use to load the data for this entity

Usage:

USAGE - To use this function first call it with the desired initialization file name, e.g.

```
WritePlaybackInit('Configuration1.ply.ini');
```

NOTE: the extension '.ply.ini' is the defult used by AVDS.

Next call the function for each entity that there is data for, e.g. if there is data for 3 vehicles:

```
WritePlaybackInit('Configuration1.ply.ini','Vehicle01','Data01.save.dat');
WritePlaybackInit('Configuration1.ply.ini','Vehicle01','Data01.save.dat');
WritePlaybackInit('Configuration1.ply.ini','Vehicle01','Data03.save.dat');
```

AVDS Default Data Format:

AVDS playback data files are ASCII (text) files where the data is in columns. The columns can be in any order and the only requirement on the content/number of columns is that there must be at least one column that contains time information, elapsed or delta, in seconds. For the default column see the function ”**SaveAVDSData**”

AVDSNetwork

Description:

Sends/receives vehicle packets between AVDS and MATLAB using multicast network functions. There are four types of calls for this function: ***Vehicle Initialization***, ***Send Data***, ***Receive Data*** and ***Close Connections***.

Syntax:

```
[] = AVDSNetwork(action, …)
```

1. ***Vehicle Initialization****:* `[] = AVDSNetwork('InitializeVehicle',VehicleID,Data2Network)`
2. ***Send Data:*** `[] = AVDSNetwork('SendData',VehicleID,Data2Network)`
3. ***Receive Data:*** `[ReceivedMatrix] = AVDSNetwork('ReceiveData')`
4. ***Close Connections:*** `[] = AVDSNetwork('CloseConnections')`

Arguments:

- ***Action*** - defines the mode of operation. Valid values are:
  1. `'InitializeVehicle'` – opens the connection and initializes the packet structure.
  2. `'SendData'` – send one packet of data to the local network.
  3. `'ReceiveData'` – receives all the packets that have been sent since the last call to this function.
  4. `'CloseConnections'` – closes all open connections.
- ***VehicleID*** - identification number for the vehicle. If packets are being sent out for more than one vehicle then a unique VehicleID is required for each vehicle.
- ***Data2Network*** - vector containing the vehicle data to send out. During an `'InitializeVehicle'` call to the function this Data2Network is optional, if it is not present then default initial values will be used for the vehicle. The elements in the Data2Network vector change based on the type of call:

  During an `'InitializeVehicle'` the values for Data2Network are:

  ***Data2Network*=[CraftType, InitLongitudeDegDecimal, InitLatitudeDegDecimal, InitAltitudeFeet]**

  - **CraftType** – the zero-based index number from AVDS's craft menu to use for the vehicle image display.
  - **InitLongitudeDegDecimal** – starting longitude position, the rest of the vehicle positions will based on this value. Longitude is in degrees decimal and must be in the range ***[-180.0,180.0]***.
  - **InitLatitudeDegDecimal** – starting latitude position, the rest of the vehicle positions will based on this value. Latitude is in degrees decimal and must be in the range ***[-90.0,90.0]***.
  - **InitAltitudeFeet** – Starting altitude in feet above mean-sea-level (MSL).

  During a `'SendData'` the values for Data2Network are:

  ***Data2Network* = [PositionEastFeet, PositionNorthFeet, AltitudeFeet, PhiDeg, ThetaDeg, PsiDeg, DeltaElevatorRad, DeltaAileronRad, DeltaRudderRad, DeltaEngine0_255, uFeetPerSec, vFeetPerSec, wFeetPerSec, pDegPerSec, qDegPerSec, rDegPerSec]**

  - **PositionEastFeet** – East position, relative to the initial position, in feet.
  - **PositionNorthFeet** – North position, relative to the initial position, in feet.
  - **AltitudeFeet** – MSL altitude in feet.

- **PhiDeg** – Phi Euler angle in degrees.
- **ThetaDeg** – Theta Euler angle in degrees.
- **PsiDeg** – Psi Euler angle in degrees.
- **DeltaElevatorRad** – Elevator deflection angle in degrees.
- **DeltaAileronRad** – Aileron deflection angle in degrees.
- **DeltaRudderRad** – Rudder deflection angle in degrees.
- **DeltaEngine0_255** – Engine level 0 to 255.
- **UFeetPerSec** – Velocity along the X body axis in feet per second.
- **VFeetPerSec** – Velocity along the Y body axis in feet per second.
- **WFeetPerSec** – Velocity along the Z body axis in feet per second.
- **PDegPerSec** – Angular velocity about the X body axis in degrees per second.
- **QDegPerSec** – Angular velocity about the Y body axis in degrees per second.
- **RDegPerSec** – Angular velocity about the Z body axis in degrees per second.

- ***ReceivedMatrix*** - is a matrix that contains all of the vehicle packets received since the last `'ReceiveData'` call. Received packets are appended as new rows to ReceivedMatrix. All vehicle packets available on the local network are appended to ReceivedMatrix, therefore one must extract the CraftID from the packets to differentiate between the packets from various vehicles. For example, the structure of a vehicle packet (row=1) vector is:

  ReceivedMatrix(1,:)=[VehicleID, PositionEastFeet, PositionNorthFeet, AltitudeFeet, PhiDeg, ThetaDeg, PsiDeg, DeltaElevatorRad, DeltaAileronRad, DeltaRudderRad, DeltaEngine0_255, UFeetPerSec, VFeetPerSec, WFeetPerSec, PDegPerSec, QDegPerSec, RDegPerSec]

  - **VehicleID** – identification number for the vehicle. If packets are being sent out for more than one vehicle then a unique VehicleID is required for each vehicle.
  - **PositionEastFeet** – East position, relative to the initial position, in feet.
  - **PositionNorthFeet** - North position, relative to the initial position, in feet.
  - **AltitudeFeet** - MSL altitude in feet.
  - **PhiDeg** – Phi Euler angle in degrees.
  - **ThetaDeg** – Theta Euler angle in degrees.
  - **PsiDeg** – Psi Euler angle in degrees.
  - **DeltaElevatorRad** - Elevator deflection angle in degrees.
  - **DeltaAileronRad** - Aileron deflection angle in degrees.
  - **DeltaRudderRad** - Rudder deflection angle in degrees.
  - **DeltaEngine0_255** - Engine level 0 to 255.
  - **UFeetPerSec** - Velocity along the X body axis in feet per second.
  - **VFeetPerSec** - Velocity along the Y body axis in feet per second.
  - **WFeetPerSec** - Velocity along the Z body axis in feet per second.
  - **PDegPerSec** - Angular velocity about the X body axis in degrees per second.
  - **QDegPerSec** - Angular velocity about the Y body axis in degrees per second.

- **RDegPerSec** - – Angular velocity about the Z body axis in degrees per second.

Usage:

To send packets from MATLAB to AVDS:

- initialize the vehicle using the syntax of ***Vehicle Initialization***, above.
- send the vehicle packets using ***Send Data***.
- when the simulation is finished, close the connection using ***Close Connections***.

To receive vehicle packets in MATLAB:

- receive the vehicle packets using ***Receive Data*** above.
- when the simulation is finished, close the connection using ***Close Connections***.

Examples:

Send Packets:

```
VehicleID = 123;
Data2Network = [1,-122.063400,37.981523,5000.0];
AVDSNetwork('InitializeVehicle',VehicleID,Data2Network)
Data2Network = [0,0,5000.0,0.0,0.0,45.0, …
                    0.0873,0.0,0.0,53, …
                    300.0,0.0,0.0,0.0,0.0,0.0];
AVDSNetwork('SendData',VehicleID,[1,-122.063400,37.981523,5000.0])
. . .
Data2Network = [345533,-234747,12367.0,5.8,13.4,135.6, …
                    0.163,0.028,0.00023,203, …
                    523.0,50.0,23.0,18.5,-23.6,5.34];
AVDSNetwork('SendData',VehicleID,Data2Network)
AVDSNetwork('CloseConnections')
```

Receive Packets:

```
[ReceivedMatrix] = AVDSNetwork('ReceiveData');
                % (process ReceivedMatrix vehicle packets)
   . . .
[ReceivedMatrix] = AVDSNetwork('ReceiveData');
                % (process ReceivedMatrix vehicle packets)
AVDSNetwork('CloseConnections');
```

Matlab Mex Functions

AVDSSimulation

Description:

This function provides a two-way communications connection between Matlab and AVDS through shared memory. Depending on the entries for the ***IDToAVDS*** and ***IDToMatlab*** arguments, this function can be configured to communicate to either the "*MATLAB_FCS*" or "*MATLAB_AC*" user defined simulation block in AVDS.

Syntax:

```
[] = AVDSSimulation(Mode, …)
```

```
AVDSSimulation(1,ConnectionID, ToAVDS_ID, SizeToAVDS, ToMATLAB_ID, SizeToMatlab);
AVDSSimulation(2,ConnectionID,VectorToAVDS);
VectorToMATLAB = AVDSSimulation(3,ConnectionID);
AVDSSimulation(4);
```

Arguments:

- ***Mode*** – Type of **AVDSSimulation** function call. Valid choices are:

  `1` – Open Connection, establishes the shared memory connection between AVDS and Matlab.

  `2` – Send Data, sends a data vector to AVDS through the shared memory connection.

  `3` – Receive Data, receives a data vector from AVDS through the shared memory connection.

  `4` – Close Connections, close all open shared memory connections.

- ***ConnectionID –*** unique numerical ID for this connection.
- ***ToAVDS_ID*** - numerical ID for the Matlab to AVDS connection. Used to select the connection type, 'AC' or 'FCS'. Valid values for this parameters are:

  `1` – 'FCS' ID, connect to the "*MATLAB_FCS*" user defined simulation block in AVDS.

  `3` – 'AC' ID, connect to the "*MATLAB_AC*" user defined simulation block in AVDS.

- ***SizeToAVDS*** – the size of the data vector that is sent from Matlab to AVDS. Valid entries for this parameter are:

  `14` – for the 'FCS' connection.

  `25` – for the 'AC' connection.

- ***ToMATLAB_ID*** - numerical ID for the AVDS to Matlab connection. Used to select the connection type, 'AC' or 'FCS'. Valid values for this parameters are:

  `2` – 'FCS' ID, connect from the "*MATLAB_FCS*" user defined simulation block in AVDS.

  `4` – 'AC' ID, connect from the "*MATLAB_AC*" user defined simulation block in AVDS.

- ***SizeToMatlab*** – the size of the data vector that is sent from AVDS to Matlab. Valid entries for this parameter are:

  `19` – for the 'FCS' connection.

  `16` – for the 'AC' connection.

- ***VectorToAVDS –***sent to AVDS during a *Send Data* call to the function. The use of this argument changes based on the value of **ToAVDS_ID** during the *Open Connection* call:

'FCS': **`VectorToAVDS`:**

Element	Name	Definition
1. – 3.	**ElevatorCmd, AileronCmd, RudderCmd**	The elevator, aileron and rudder command from the Matlab FCS to AVDS. The nominal values for these signals in AVDS are in the range, ***[-1.0,1.0].***
4.	**ThrottleCmd**	The throttle command from the Matlab FCS. The nominal values for the throttle in AVDS are in the range, ***[0.0,1.0].***
5. - 14.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

'AC': **`VectorToAVDS`:**

Element	Name	Definition
1. – 3.	**xpos, ypos, zpos**	The position of the vehicle along the North and East axes, and the altitude, respectively. These values are in feet.
4. – 6.	**xrot, yrot, zrot**	The angular orientation of the vehicle given in Eurler angle in radians. These angles are Phi, Theta and Psi, respectively.
7.	**alpha**	The angle of attack in radians.
8.	**beta**	The sideslip angle in radians.
9.	**G**	Number of induced G's.
10.	**Velocity**	Total velocity in feet/second.
11.	**M**	Mach number.
12.	**DeltaRudder**	Rudder deflection in radians.
13.	**DeltaElevator**	Elevator deflection in radians.
14.	**DeltaAileron**	Aileron deflection in radians.
15.	**eng**	The engine level. Must in the range ***[0,1].***
16.. - 25.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

- ***VectorToMatlab –*** returned from the function during a *Receive Data* call. The use of this vector changes based on the value of **ToMATLAB_ID** during the *Open Connection* call. Note: the definitions in this table are the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.

'FCS': **`VectorToMatlab` :**

Element	Name	Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2. - 4.	**StickPitch, StickRoll. StickYaw**	User inputs from the input device, i.e. joystick, mouse, keyboard. These inputs are in the range ***[-1.0,1.0]***.
5.	**StickThrottle**	User inputs from throttle. This input is in

		the range ***[0.0,1.0]***.
6. – 8.	**P, Q, R**	The vehicle's angular rates about the x, y and z body axes, respectively. These values are in radians.
9.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
9. - 19.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

'AC': **`VectorToMatlab`:**

Element	Name	Definition Note: the definitions in this table are for the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2.	**dElevator**	The elevator command from the FCS.
3.	**dAileron**	The aileron command from the FCS.
4.	**dRudder**	The rudder command from the FCS.
5.	**dThrottle**	The throttle command. This signal either originates in the FCS or comes directly from the joystick.
6.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
7. - 16.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

Usage:

To communicate between MATLAB and AVDS first open the communications connection with a call to **AVDSSimulation** using the syntax of #1 above. Next use #2 and #3 to accomplish the communications. Finally use #4 to close all of the connections.

NOTE: The sizes and ID numbers of the vectors must be the same as those in the AVDS shared simulation model. The definitions for this model are in the file

".\SharedMemoryLibrary\AVDS2MATLAB.h"

Examples:

The AVDSSimulation MEX function is used by the Matlab function "**SimulationConnection.m**".

NetworkAVDS

Description:

Sends/receives vehicle packets between AVDS and MATLAB using multicast network functions. There are four types of calls for this function: ***Vehicle Initialization (1)***, ***Send Data (2)***, ***Receive Data (3)*** and ***Close Connections (4)***.

Syntax:

```
NetworkAVDS(mode, …);
```

1. ***Vehicle Initialization****:* `= NetworkAVDS(1, IDConnection, CraftType, InitLongitude, InitLatitude, InitAltitude)`
2. ***Send Data:*** `[] = NetworkAVDS(2,IDConnection,SendData)`
3. ***Receive Data:*** `[ReceiveMatrix] = NetworkAVDS(3)`
4. ***Close Connections:*** `[] = NetworkAVDS(4)`

Arguments:

- ***Mode*** - Type of NetworkAVDS function call. Valid choices are:

  `0` - Initialize Vehicle -

  `1` - Send Data -

  `2` - Receive Data -

  `3` - Close Connection –

- ***IDConnection*** – is an identification number for the vehicle. If packets are being sent out for more than one vehicle then a unique VehicleID is required for each vehicle.
- ***CraftType*** – the zero-based index number from AVDS's craft menu to use for the vehicle image display.
- ***InitLongitudeDegDecimal*** – starting longitude position, the rest of the vehicle positions will based on this value. Longitude is in degrees decimal and must be in the range [-180.0,180.0].
- ***InitLatitudeDegDecimal*** – starting latitude position, the rest of the vehicle positions will based on this value. Latitude is in degrees decimal and must be in the range [-90.0,90.0].
- ***InitAltitudeFeet*** – Starting altitude in feet above mean-sea-level (MSL).
- ***SendData*** – This is an array of data that is sent out on the local network. The elements in the array are:

  `SendData` **= [PositionEastFeet, PositionNorthFeet, AltitudeFeet, PhiDeg, ThetaDeg, PsiDeg, DeltaElevatorRad, DeltaAileronRad, DeltaRudderRad, DeltaEngine0_255, uFeetPerSec, vFeetPerSec, wFeetPerSec, pDegPerSec, qDegPerSec, rDegPerSec]**

  - **PositionEastFeet** – East position, relative to the initial position, in feet.
  - **PositionNorthFeet** – North position, relative to the initial position, in feet.
  - **AltitudeFeet** – MSL altitude in feet.
  - **PhiDeg** – Phi Euler angle in degrees.
  - **ThetaDeg** – Theta Euler angle in degrees.
  - **PsiDeg** – Psi Euler angle in degrees.
  - **DeltaElevatorRad** – Elevator deflection angle in degrees.
  - **DeltaAileronRad** – Aileron deflection angle in degrees.

- **DeltaRudderRad** – Rudder deflection angle in degrees.
- **DeltaEngine0_255** – Engine level 0 to 255.
- **UFeetPerSec** – Velocity along the X body axis in feet per second.
- **VFeetPerSec** – Velocity along the Y body axis in feet per second.
- **WFeetPerSec** – Velocity along the Z body axis in feet per second.
- **PDegPerSec** – Angular velocity about the X body axis in degrees per second.
- **QDegPerSec** – Angular velocity about the Y body axis in degrees per second.
- **RDegPerSec** – Angular velocity about the Z body axis in degrees per second.

- ***ReceiveMatrix*** – This is a matrix of data that is received from the local network. Each row of the matrix contains the data received from one packet. The matrix contains data for all of the packets received sent the last call to this function. The elements in the array are:

  `ReceiveMatrix`**(1,:)=[VehicleID, PositionEastFeet, PositionNorthFeet, AltitudeFeet, PhiDeg, ThetaDeg, PsiDeg, DeltaElevatorRad, DeltaAileronRad, DeltaRudderRad, DeltaEngine0_255, uFeetPerSec, vFeetPerSec, wFeetPerSec, pDegPerSec, qDegPerSec, rDegPerSec]**

  - **VehicleID** – identification number for the vehicle. If packets are being sent out for more than one vehicle then a unique VehicleID is required for each vehicle.
  - **PositionEastFeet** – East position, relative to the initial position, in feet.
  - **PositionNorthFeet** - North position, relative to the initial position, in feet.
  - **AltitudeFeet** - MSL altitude in feet.
  - **PhiDeg** – Phi Euler angle in degrees.
  - **ThetaDeg** – Theta Euler angle in degrees.
  - **PsiDeg** – Psi Euler angle in degrees.
  - **DeltaElevatorRad** - Elevator deflection angle in degrees.
  - **DeltaAileronRad** - Aileron deflection angle in degrees.
  - **DeltaRudderRad** - Rudder deflection angle in degrees.
  - **DeltaEngine0_255** - Engine level 0 to 255.
  - **uFeetPerSec** - Velocity along the X body axis in feet per second.
  - **vFeetPerSec** - Velocity along the Y body axis in feet per second.
  - **wFeetPerSec** - Velocity along the Z body axis in feet per second.
  - **pDegPerSec** - Angular velocity about the X body axis in degrees per second.
  - **qDegPerSec** - Angular velocity about the Y body axis in degrees per second.
  - **rDegPerSec** - – Angular velocity about the Z body axis in degrees per second.

Usage:

To send packets from MATLAB to AVDS:

1. initialize the vehicle using the syntax of #1, above.
2. send the vehicle packets using #2.
3. when the simulation is finished, close the connection using #4.

To receive vehicle packets in MATLAB:

1. receive the vehicle packets using #3 above.
2. when the simulation is finished, close the connection using #5.

Examples:

The NetworkAVDS MEX function is used by the Matlab function "**NetworkConnection.m**".

Simulink Blocks

AC Connection Block

Description:

This block connects a Simulink vehicle simulation to AVDS through shared memory. User inputs and elapsed time are collected from AVDS and sent through shared memory to this block. Simulink calculates the vehicle linear and angular positions and sends them back through shared memory to AVDS for display. This block uses the S-function "**AVDSSimulation**" to connect to AVDS.

Inputs:

The signals that are input to this block are sent to the "*MATLAB_AC*" user defined simulation block in AVDS. There is one input port that contains 25 signals.

Input Signal	Input Name	Input Definition
1. – 3.	**xpos, ypos, zpos**	The position of the vehicle along the North and East axes, and the altitude, respectively. These values are in feet.
4. – 6.	**xrot, yrot, zrot**	The angular orientation of the vehicle given in Eurler angle in radians. These angles are Phi, Theta and Psi, respectively.
7.	**alpha**	The angle of attack in radians.
8.	**beta**	The sideslip angle in radians.
9.	**G**	Number of induced G's.
10.	**Velocity**	Total velocity in feet/second.
11.	**M**	Mach number.
12.	**DeltaRudder**	Rudder deflection in radians.
13.	**DeltaElevator**	Elevator deflection in radians.
14.	**DeltaAileron**	Aileron deflection in radians.
15.	**eng**	The engine level. Must in the range of [0,1].
16.. - 25.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

Outputs:

The signals that are output from this block are sent from the "*MATLAB_AC*" user defined simulation block in AVDS. There is one input port that contains 16 signals. Note: the definitions in this table are for the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.

Output Signal	Output Name	Output Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.

2.	**dElevator**	The elevator command from the FCS.
3.	**dAileron**	The aileron command from the FCS.
4.	**dRudder**	The rudder command from the FCS.
5.	**dThrottle**	The throttle command. This signal either originates in the FCS or comes directly from the joystick.
6.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
7. - 16.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

Parameters:
None.

Examples:
The **AC Connection** block is used in the example:

"MATLABToolbox\Examples\SimulationExample**ExampleAC.mdl**".

FCS Connection Block

Description:

This block connects a Simulink flight control simulation (FCS) simulation to AVDS through shared memory. User inputs, vehicle outputs and elapsed time are collected from AVDS and sent through shared memory to this block. Simulink calculates the FCS outputs based on the inputs and sends them back through shared memory to AVDS for display. This block uses the S-function "**AVDSSimulation**" to connect to AVDS.

Inputs:

The signals that are input to this block are sent to the "*MATLAB_FCS*" user defined simulation block in AVDS. There is one input port that contains 14 signals.

Input Signal	Input Name	Input Definition
1.	**ElevatorCmd**	The elevator command from the Simulink FCS.
2.	**AileronCmd**	The aileron command from the Simulink FCS.
3.	**RudderCmd**	The rudder command from the Simulink FCS.
5.	**ThrottleCmd**	The throttle command from the Simulink FCS.
4.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
5. - 14.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

Outputs:

The signals that are output from this block are sent from the "*MATLAB_FCS*" user defined simulation block in AVDS. There is one input port that contains 19 signals. Note: the definitions in this table are for the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.

Output Signal	Output Name	Output Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2. - 5.	**StickPitch, StickRoll. StickYaw, StickThrottle**	User inputs from the input device, i.e. joystick, mouse, keyboard. These inputs are in the range [-1.0,1.0].
6. – 8.	**P, Q, R**	The vehicle's angular rates about the x, y and z body axes, respectively. These values are in radians.
9.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
9. - 19.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

Parameters:
None.

Examples:
The **FCS Connection** block is used in the example:

"MATLABToolbox\Examples\SimulationExample**ExampleFCS.mdl**".

Playback Configuration File Block

Description:

This function writes an initialization file that configures AVDS playback. Based on the parameter values, this block constructs and saves an AVDS playback initialization file (*.ply.ini) that contains entries for the number of vehicles given in the parameters. This file is saved with the file name that is given. Each of the entries for playback entities includes a filename that is constructed by appending the entity number to the base file name given in the parameters. See the list of inputs to the **Playback Write Data Block** for the order of playback data columns that are configured using this block. This block uses the S-function, "**AVDSPlaybackSetup**" to perform its functions.

Inputs:

None.

Outputs:

None.

Parameters:

ConfigurationFileName – File name for the playback initialization file. The default AVDS extension for playback initialization files is ".ply.txt". Note: no extension is added to the given file name.

BaseDataFileName – This name is used along with the internally calculated entity number to construct data file name for all of the vehicles/entities.

NumberVehicles – The number of vehicles/entities to include in the playback initialization file.

Examples:

The **Playback Configuration File** block is used in the example:

"MATLABToolbox\Examples\PlaybackExample**PlaybackExample.mdl**".

Playback Write Data Block

Description:

Used to manage data for output to an AVDS playback file. This block opens/clears an ASCII (text) file at the beginning of the simulation and uses it to save the input vector in rows, based on simulation time. The file name is derived by appending the vehicle ID number to the end of the base data file name. AVDS playback data files are ASCII files with the data in columns. The columns can be in any order and the only requirement on the content/number of columns is that there must be at least one column that contains time information, elapsed or delta, in seconds. The order of the columns must be configured either by using the source code and changing the **AVDSPlaybackSetup** function or by using the Playback Configuration dialog window in AVDS. The default inputs for this block are the inputs that are configured with the **Playback Configuration File Block**. This block uses the S-function, "**AVDSPlaybackSave**" to perform its functions.

Inputs:

TimeSeconds, NorthPositionFeet, EastPositionFeet, DownPositionFeet, PsiRotationDeg, ThetaRotationDeg, PsiRotationDeg, craftmask, crafttype, EngineLevel, AlphaDeg, BetaDeg, G, VFtSec, MachNumber, DeltaSurface01Rad, DeltaSurface02Rad, DeltaSurface03Rad, DeltaSurface04Rad, DeltaSurface05Rad, DeltaSurface06Rad, DeltaSurface07Rad, DeltaSurface08Rad, DeltaSurface09Rad, DeltaSurface10Rad, DeltaSurface11Rad, DeltaSurface12Rad, DeltaSurface13Rad, DeltaSurface15Rad, DeltaSurface15Rad, WeaponsSelect, WeaponsLaunch, Explode

Note: For the definitions of these variables see the AVDS User's Manual, Table 5-1 Dynamic Items.

Outputs:

None.

Parameters:

BaseDataFileName – This name is used as part of the playback data file name. Note: When using the **Playback Configuration File Block**, this name must be the same as the BaseDataFileName for that block.

VehicleID – This is a unique numerical identification number for the vehicle/entity. This number is used, along with the BaseDataFileName to build the file name. Note: When using the **Playback Configuration File Block**, this number must be in the range of [1, NumberVehicles].

NumberInputs – This parameter determines the number of inputs to this block. Note: if it is desired to use a subset of the inputs listed , starting with the time, then the configuration file will not have to be altered, i.e. if the simulation outputs are time, X position, Y position and Z position, then use 4 inputs.

Examples:

The **Playback Write Data** block is used in the example:

"MATLABToolbox\Examples\PlaybackExample**PlaybackExample.mdl**".

Network Send Connection Block

Description:

Sends AVDS vehicle packets to the local network from Matlab using multicast network functions.

Inputs:

There is one input port that has 16 signals:

Input Element	Input Name	Input Definition
1.	***PositionEastFeet***	Vehicle position along the east-west axis in feet. This position is relative to the initial starting longitude.
2.	***PositionNorthFeet***	Vehicle position along the north-south axis in feet. This position is relative to the initial starting latitude
3.	***AltitudeFeet***	Altitude of the vehicle in feet above mean-sea-level.
4.	***PhiDeg***	Phi Euler angle in degrees.
5.	***ThetaDeg***	Theta Euler angle in degrees.
6.	***PsiDeg***	Psi Euler angle in degrees.
7.	***DeltaElevatorRad***	Elevator deflection in radians.
8.	***DeltaAileronRad***	Aileron deflection in radians.
9.	***DeltaRudderRad***	Rudder deflection in radians.
10.	***DeltaEngine0_255***	Engine level. Must be in the range [0 255].
11.	***UFeetPerSec***	Velocity along the X body axis in feet per second.
12.	***VFeetPerSec***	Velocity along the Y body axis in feet per second.
13.	***WFeetPerSec***	Velocity along the Z body axis in feet per second.
14.	***PDegPerSec***	Angular velocity about the X body axis in degrees per second.
15.	***QDegPerSec***	Angular velocity about the Y body axis in degrees per second.
16.	***RDegPerSec***	Angular velocity about the Z body axis in degrees per second.

Outputs:

None

Parameters:

VehicleHandle – This is a unique name to identify this vehicle on the network. Must be 28 characters or less.

TransmitRateHz – This is the rate in cycles per second (Hz) to transmit vehicle packets on the local network.

CraftType – This is the zero-based index number from AVDS's craft menu to use for the vehicle image display.

InitLatitudeDeg – This is the initial position of the vehicle in degrees latitude. Initial Latitude is in degrees decimal and must be in the range ***[-90.0,90.0]***.

InitLongitudeDeg – This is the initial position of the vehicle in degrees longitude. Initial Longitude is in degrees decimal and must be in the range ***[-180.0,180.0]***.

InitAltitudeFeet – This is the initial altitude of the vehicle in feet above mean-sea-level.

Examples:

The **Network Send Connection** block is used in the example:

"MATLABToolbox\Examples\PlaybackExample**NetworkSendExample.mdl**".

Simulink S-Functions

AVDSSimulation

Description:

This function provides a two-way communications connection between Simulink and AVDS through shared memory. Depending on the settings for the ***IDToAVDS*** and ***IDToSimulink*** parameters, this function can be configured to communicate to either the "*MATLAB_FCS*" or "*MATLAB_AC*" user defined simulation block in AVDS.

Inputs:

The signals that are input to this block are sent to AVDS. The number and definition of the signals in the input port depends on the value of the ***IDToAVDS*** parameter:

***IDToAVDS* = 1 (FCS)**:

Input Signal	Input Name	Input Definition
1.	**ElevatorCmd**	The elevator command from the Simulink FCS.
2.	**AileronCmd**	The aileron command from the Simulink FCS.
3.	**RudderCmd**	The rudder command from the Simulink FCS.
5.	**ThrottleCmd**	The throttle command from the Simulink FCS.
4.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
5. - 14.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

***IDToAVDS* = 3 (AC)**:

Input Signal	Input Name	Input Definition
1. – 3.	**xpos, ypos, zpos**	The position of the vehicle along the North and East axes, and the altitude, respectively. These values are in feet.
4. – 6.	**xrot, yrot, zrot**	The angular orientation of the vehicle given in Eurler angle in radians. These angles are Phi, Theta and Psi, respectively.
7.	**alpha**	The angle of attack in radians.
8.	**beta**	The sideslip angle in radians.
9.	**G**	Number of induced G's.
10.	**Velocity**	Total velocity in feet/second.
11.	**M**	Mach number.
12.	**DeltaRudder**	Rudder deflection in radians.
13.	**DeltaElevator**	Elevator deflection in radians.
14.	**DeltaAileron**	Aileron deflection in radians.
15.	**eng**	The engine level. Must in the range of [0,1].
16.. - 25.	**ToAVDS01 - ToAVDS10**	These are generic signals to AVDS for user definition.

Outputs:

The signals that are output from this block are sent from AVDS. The number and definition of the signals in the output port depends on the value of the ***IDToSimulink*** parameter. Note: the definitions in these tables are the default use of these signals in AVDS. The user can change the meaning of these signals by changing the simulation configuration in AVDS.

***IDToSimulink* = 2**:

Output Signal	Output Name	Output Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2. - 5.	**StickPitch, StickRoll. StickYaw, StickThrottle**	User inputs from the input device, i.e. joystick, mouse, keyboard. These inputs are in the range [-1.0,1.0].
6. – 8.	**P, Q, R**	The vehicle's angular rates about the x, y and z body axes, respectively. These values are in radians.
9.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
9. - 19.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

***IDToSimulink* = 4 (AC)**:

Output Element	Output Name	Output Definition
1.	**Alive**	A flag sent from AVDS to indicate if AVDS is running the simulation. When the simulation is running this flag is set to 1. Otherwise it is set to 0.
2.	**dElevator**	The elevator command from the FCS.
3.	**dAileron**	The aileron command from the FCS.
4.	**dRudder**	The rudder command from the FCS.
5.	**dThrottle**	The throttle command. This signal either originates in the FCS or comes directly from the joystick.
6.	**ElapsedTime**	The elapsed simulation time based on the computer's clock. This should be very close to real-time.
7. - 16.	**ToMATLAB01 - ToMATLAB10**	These are generic signals for user definition.

Parameters:

- ***IDToAVDS*** - numerical ID for the Simulink to AVDS connection. Used to select the connection type, 'AC' or 'FCS'. Valid values for this parameters are:

  1 – 'FCS' ID, connect to the "*MATLAB_FCS*" user defined simulation block in AVDS.

  3 – 'AC' ID, connect to the "*MATLAB_AC*" user defined simulation block in AVDS.

- ***NumberSignalsToAVDS*** – the size of the data vector that is sent to AVDS. Valid entries for this parameter are:

  14 – for the 'FCS' connection.

`25` – for the ‘AC’ connection.

- ***IDToSimulink*** - numerical ID for the AVDS to Simulink connection. Used to select the connection type, ‘AC’ or ‘FCS’. Valid values for this parameters are:

  `2` – ‘FCS’ ID, connect from the “*MATLAB_FCS*” user defined simulation block in AVDS.

  `4` – ‘AC’ ID, connect from the the “*MATLAB_AC*” user defined simulation block in AVDS.

- ***NumberSignalsToSimulink*** – the size of the data vector that is sent from AVDS to Simulink. Valid entries for this parameter are:

  `19` – for the ‘FCS’ connection.

  `16` – for the ‘AC’ connection.

AVDSPlaybackSetup

Description:

This function writes an initialization file that configures AVDS playback. Based on the parameter values, this block constructs and saves an AVDS playback initialization file (*.ply.ini) that contains entries for the number of vehicles given in the parameters. This file is saved with the file name that is given. Each of the entries for playback entities includes a filename that is constructed by appending the entity number to the base file name given in the parameters.

Inputs:

None.

Outputs:

None.

Parameters:

ConfigurationFileName – File name for the playback initialization file. The default AVDS extension for playback initialization files is ".ply.txt". Note: no extension is added to the given file name.

BaseDataFileName – This name is used along with the internally calculated entity number to construct data file name for all of the vehicles/entities.

NumberVehicles – The number of vehicles/entities to include in the playback initialization file.

AVDSPlaybackSave

Description:

Used to manage data for output to an AVDS playback file. This block opens/clears an ASCII (text) file at the beginning of the simulation and uses it to save the input vector in rows, based on simulation time. The file name is derived by appending the vehicle ID number to the end of the base data file name. AVDS playback data files are ASCII files with the data in columns. The columns can be in any order and the only requirement on the content/number of columns is that there must be at least one column that contains time information, elapsed or delta, in seconds. The order of the columns must be configured either by using the source code and changing the **AVDSPlaybackSetup** function or by using the Playback Configuration dialog window in AVDS. The default inputs for this block are the inputs that are configured with the **AVDSPlaybackSetup**.

Inputs:

TimeSeconds, NorthPositionFeet, EastPositionFeet, DownPositionFeet, PsiRotationDeg, ThetaRotationDeg, PsiRotationDeg, craftmask, crafttype, EngineLevel, AlphaDeg, BetaDeg, G, VFtSec, MachNumber, DeltaSurface01Rad, DeltaSurface02Rad, DeltaSurface03Rad, DeltaSurface04Rad, DeltaSurface05Rad, DeltaSurface06Rad, DeltaSurface07Rad, DeltaSurface08Rad, DeltaSurface09Rad, DeltaSurface10Rad, DeltaSurface11Rad, DeltaSurface12Rad, DeltaSurface13Rad, DeltaSurface15Rad, DeltaSurface15Rad, WeaponsSelect, WeaponsLaunch, Explode

Note: For the definitions of these variables see the AVDS User's Manual, Table 5-1 Dynamic Items.

Outputs:

None.

Parameters:

BaseDataFileName – This name is used as part of the playback data file name. Note: When using the **AVDSPlaybackSetup** function, this name must be the same as the BaseDataFileName for that block.

VehicleID – This is a unique numerical identification number for the vehicle/entity. This number is used, along with the BaseDataFileName to build the file name. Note: When using the **AVDSPlaybackSetup** function, this number must be in the range of [1, NumberVehicles].

NumberInputs – This parameter determines the number of inputs to this block. Note: if it is desired to use a subset of the inputs listed , starting with the time, then the configuration file will not have to be altered, i.e. if the simulation outputs are time, X position, Y position and Z position, then use 4 inputs.

AVDSNetworkSend

Description:

Sends AVDS vehicle packets to the local network from Matlab using multicast network functions.

Inputs:

There is one input port that has 16 signals:

Input Signal	Input Name	Input Definition
1.	***PositionEastFeet***	Vehicle position along the east-west axis in feet. This position is relative to the initial starting longitude.
2.	***PositionNorthFeet***	Vehicle position along the north-south axis in feet. This position is relative to the initial starting latitude
3.	***AltitudeFeet***	Altitude of the vehicle in feet above mean-sea-level.
4.	***PhiDeg***	Phi Euler angle in degrees.
5.	***ThetaDeg***	Theta Euler angle in degrees.
6.	***PsiDeg***	Psi Euler angle in degrees.
7.	***DeltaElevatorRad***	Elevator deflection in radians.
8.	***DeltaAileronRad***	Aileron deflection in radians.
9.	***DeltaRudderRad***	Rudder deflection in radians.
10.	***DeltaEngine0_255***	Engine level. Must be in the range [0 255].
11.	***UFeetPerSec***	Velocity along the X body axis in feet per second.
12.	***VFeetPerSec***	Velocity along the Y body axis in feet per second.
13.	***WFeetPerSec***	Velocity along the Z body axis in feet per second.
14.	***PDegPerSec***	Angular velocity about the X body axis in degrees per second.
15.	***QDegPerSec***	Angular velocity about the Y body axis in degrees per second.
16.	***RDegPerSec***	Angular velocity about the Z body axis in degrees per second.

Outputs:

None.

Parameters:

VehicleHandle – This is a unique name to identify this vehicle on the network. Must be 28 characters or less.

TransmitRateHz – This is the rate in cycles per second (Hz) to transmit vehicle packets on the local network.

CraftType – This is the zero-based index number, numbering starts at zero, from AVDS's craft menu to use for the vehicle image display.

InitLatitudeDeg – This is the initial position of the vehicle in degrees latitude. Initial Latitude is in degrees decimal and must be in the range ***[-90.0,90.0]***.

InitLongitudeDeg – This is the initial position of the vehicle in degrees longitude. Initial Longitude is in degrees decimal and must be in the range ***[-180.0,180.0]***.

InitAltitudeFeet – This is the initial altitude of the vehicle in feet above mean-sea-level.