Generic UDP Software Interface Manual

For all CKAS 6DOF Motion Systems

Published by CKAS Mechatronics, Melb, VIC Australia
info@ckas.com.au
1 Table of Contents

1 Table of Contents ..................................................................................... 2
2 General Warnings .................................................................................... 3
3 Background information ........................................................................... 4
4 Mathematical Conventions ....................................................................... 5
  4.1 Axes ................................................................................................... 5
  4.2 Reference ........................................................................................... 5
    4.2.1 World Co-ordinate Axis (W) ...................................................... 6
    4.2.2 Body Co-ordinate Axis (B) ......................................................... 6
    4.2.3 Simulator Platform Co-ordinate Axis (S) ..................................... 6
    4.2.4 Graphical Representation ........................................................... 7
  4.3 Important Parameters .......................................................................... 8
    4.3.1 Vehicle Velocity ........................................................................ 8
    4.3.2 Vehicle Acceleration ................................................................. 8
    4.3.3 Rotation Matrix ......................................................................... 8
    4.3.4 Vehicle Angular Velocity ........................................................... 9
    4.3.5 Vehicle Angular Acceleration ..................................................... 9
    4.3.6 Simulator Platform Manual Position ......................................... 9
    4.3.7 Simulator Platform Manual Orientation ..................................... 10
5 UDP Generic Data Receiver Emulator ..................................................... 12
  5.1 Data Packet ...................................................................................... 14
    5.1.1 Format ....................................................................................... 14
    5.1.2 Address ..................................................................................... 14
    5.1.3 Data rate .................................................................................. 14
    5.1.4 Precision .................................................................................. 14
    5.1.5 Parameters .............................................................................. 14
    5.1.6 Implementation ........................................................................ 15
6 Appendix I – Installation of LabVIEW Runtime Engine ....................... 16
# General Warnings

CKAS Motion Systems are in general very dangerous pieces of machinery if used incorrectly. Take genuine warning of the hazard signs placed on the machines. They pose **Electrical Hazards, Crushing Hazards, Dropping Hazards and Lifting Hazards.**

**Electrical Hazards:** CKAS Motion systems are powered by 240V AC and the Motors are powered by up to 240V AC. Any visible cables MAY CARRY 240V AC. Stay clear of any cables on the machines.

**Crushing Hazard:** CKAS Motion systems are extremely dynamic pieces of equipment. They move in highly unpredictable ways, and with extreme crushing force. **STAND CLEAR THEY OPERATE! DO NOT** “ride” the motion systems without a certified simulator cabin attached and **DO NOT** work on them or perform maintenance on them when operating. They have the capability of producing in excess of **1500kg** of force, which is more than enough to **INJURE** or **KILL**.

**Dropping Hazard:** CKAS Motion systems are very heavy pieces of equipment (up to **550kg**), and can pose a threat when being moved. **STAND CLEAR** of motion systems when they are being moved at all times.

**Lifting Hazard:** **DO NOT** attempt to lift any motion systems without the use of lifting equipment. Always use a forklift or other lifting device to move the motion systems. Using persons to lift motion systems poses a serious threat to workers, and very quickly become a dropping hazard.

This document does not describe how to use a motion system of any kind. Please refer to specific documentation for each motion system for how to operate it safely.
3 Background information

This document describes how to establish a link to the UDP Generic Data Receiver Emulator, to confirm that the client software to be used has the capability to connect to a CKAS 6DOF Motion System of some kind, through the Generic 6DOF Platform Driver.

The purpose of this document is to give the mechanical, electrical and software engineers a very good understanding of the conventions and mathematics used in the driving of the motion system, such that any vehicle interface can be achieved quickly and efficiently.

The basic premise of operation is that the client software (which typically will be connected to the vehicle or flight model in some way) will provide inertial data to the CKAS Platform Driver software over UDP, which in turn will drive the Motion System.
4 Mathematical Conventions

4.1 Axes

All data to be passed to either the UDP Generic Data Receiver Emulator or the UDP 6DOF Generic Platform driver must obey the following axis conventions for correct simulation.

ALL Axes systems used are RIGHT HANDED

The following is the axis system used for all CKAS software (which is most common in vehicle and flight simulation)

- Y +ve (Towards Right)
- X +ve (Towards Front)
- Z +ve (Towards Down)

This system must always be adhered to. If the client axis convention is different, then it MUST be converted before data is sent to the CKAS software (either emulator or platform driver).

4.2 Reference

There are three frames of reference which are used, they are:

1. World Co-ordinate Axis
2. Body Co-ordinate Axis
3. Simulator Platform Co-ordinate Axis
4.2.1 World Co-ordinate Axis (W)

The World Axis (W) is used to identify the **FIXED** co-ordinate system in the simulation software world. This axis exists in the **client** flight or vehicle model world, and cannot move. This axis is always denoted with the subscript W.

For example, the expression \( x''_w \) describes an acceleration in the X (forward-back) direction which is given in the World Co-ordinate System, within the client software.

Note: in the UDP Emulator, this might be written as \( x''(W) \).

4.2.2 Body Co-ordinate Axis (B)

The Body Axis (B) is used to identify the **MOVING** co-ordinate system which is attached to the cockpit of the vehicle being simulated in the client simulation world. This axis translates and rotates following the simulated vehicle (preferably centred at the pilot’s or driver’s head).

For example, the expression \( x''_b \) describes an acceleration in the X (forward-back) direction which is given in the Body Co-ordinate System, within the client software.

Note: in the UDP Emulator, this might be written as \( x''(B) \).

4.2.3 Simulator Platform Co-ordinate Axis (S)

The Simulator Platform Axis (S) is used to identify the **MOVING** co-ordinate system which is attached to the physical flying platform of the motion system. This axis translates and rotates following the real motion system hardware.

For example, the expression \( x''_s \) describes an acceleration in the X (forward-back) direction which is given in the Simulator Platform Co-ordinate System, within the real world.

Note: in the UDP Emulator, this might be written as \( x''(S) \).
4.2.4 Graphical Representation

Below is a graphical representation of the axes described above.

**Client Simulated World**

**Real World**

These axis conventions must be fully adhered to by the programmers of the client software to ensure safe and correct use of the motion system.
4.3 Important Parameters

The washout algorithm used by all of the CKAS Platform Driver software is based heavily on the Classical Washout Algorithm Approach, with some major modifications to make the algorithm more suited to small scale motion systems. Essentially, the main premise is that the motion system constantly tries to recreate the real vehicle’s “inertial force” given by $a - g$ where $a$ is the acceleration of the vehicle and $g$ is gravity.

This in turn means that certain parameters must be passed to the washout algorithm from the client simulation software in order to make the simulation experience correct. The parameters required are described in more detail below.

4.3.1 Vehicle Velocity

The velocity vector components of the cockpit of the vehicle being simulated, expressed in the body co-ordinates of the simulated vehicle (in metres per second – m/s).

Mathematical Expression: $X'_B (x'(t), y'(t), z'(t))$

UDP Emulator Expression: $x'(B), y'(B), z'(B)$

4.3.2 Vehicle Acceleration

The acceleration vector components of the cockpit of the vehicle being simulated, expressed in the body co-ordinates of the simulated vehicle (in metres per second squared – m/s$^2$). **NOTE:** This is NOT the measured inertial acceleration – i.e. it does NOT include the gravity component.

Mathematical Expression: $X''_B (x''(t), y''(t), z''(t))$

UDP Emulator Expression: $x''(B), y''(B), z''(B)$

4.3.3 Rotation Matrix

The rotation matrix or matrix of directional cosines used to transform a vector from World co-ordinates to a vector in Body co-ordinates by vector multiplication. The rotation matrix is used because it is axes convention independent. The client software can use any system of re-orientation (eg. Euler or otherwise) and still the rotation matrix will be the same. The only condition is that the following expressions must hold true:

$$X_B = R_{W\rightarrow B} \cdot X_W$$

and

$$R_{W\rightarrow B}^T = R_{W\rightarrow B}^{-1} = R_{B\rightarrow W}$$
4.3.4 Vehicle Angular Velocity

The instantaneous angular velocity vector components of the cockpit of the vehicle being simulated, expressed in the body co-ordinates of the simulated vehicle (in radians per second – rad/s).

Mathematical Expression: \( \Omega_B(R(t), Q(t), P(t)) \)
UDP Emulator Expression: \( R(B), Q(B), P(B) \)

Where:
- \( R(t) \) is the instantaneous “yaw” component, i.e. rotation around \( z_B \).
- \( Q(t) \) is the instantaneous “pitch” component, i.e. rotation around \( y_B \).
- \( P(t) \) is the instantaneous “roll” component, i.e. rotation around \( x_B \).

4.3.5 Vehicle Angular Acceleration

The instantaneous angular acceleration vector components of the cockpit of the vehicle being simulated, expressed in the body co-ordinates of the simulated vehicle (in radians per second squared – rad/s²).

Mathematical Expression: \( \Omega_B'(R'(t), Q'(t), P'(t)) \)
UDP Emulator Expression: \( R'(B), Q'(B), P'(B) \)

Where:
- \( R'(t) \) is the instantaneous “yaw” component, i.e. rotation around \( z_B \).
- \( Q'(t) \) is the instantaneous “pitch” component, i.e. rotation around \( y_B \).
- \( P'(t) \) is the instantaneous “roll” component, i.e. rotation around \( x_B \).

4.3.6 Simulator Platform Manual Position

This parameter allows manual positioning of the motion system if the client requires the facility. The position of the centre of the flying platform is given in millimetres (mm) from the centroid, and this value is ADDED to the washout algorithm computations before being sent to the motion system.

The use of this parameter is optional (must be included as zeros in the data transfer if not used), but can be very useful to developers who wish to “inject” special effects, such as “bumps” or “knocks” when driving over objects etc.
Often, the washout filter parameters are tuned to cut-off frequencies which “miss” certain effects, so it is important to “inject” inject them manually at the client side by commanding the motion system to “add” an extra component of movement.

The most typical way to use this is to inject a “step” function for a short time in the $z_S$ axis. This comes through to the cabin as a “bump” or “jolt” and can be very effective in adding to effects such as rough terrain, etc. The steps should be kept small (i.e. approx 5 – 15mm up) to be effective.

Another useful aspect of this facility is that it allows the client to completely manually drive the motion system if needed. If all the velocity and acceleration parameters mentioned earlier are set to zeros, then only the manual move component will have any effect.

Mathematical Expression: $X_S(x(t), y(t), z(t))$
UDP Emulator Expression: $x(S), y(S), z(S)$

**WARNING:** This parameter is passed directly to the motion system, so special care should be taken when using this parameter. It is possible to cause damage to the motion system or shorten its life by over driving it constantly.

### 4.3.7 Simulator Platform Manual Orientation

This parameter allows manual orientation of the motion system if the client requires the facility. The orientation of the flying platform is given in radians (rad) from the neutral position, and this value is ADDED to the washout algorithm computations before being sent to the motion system.

The use of this parameter is optional (must be included as zeros in the data transfer if not used), but can be very useful to developers who wish to “inject” special effects, such as “bumps” or “knocks” when driving over objects etc.

This parameter is similar to the previous, except it applies to rotations rather than translations.

Mathematical Expression: $\Psi_S(\psi(t), \theta(t), \phi(t))$
UDP Emulator Expression: yaw(S), pitch(S), roll(S)

Where:
- $\psi(t)$ is the “yaw” component (rotation around $z_S$), and is applied first resulting in a newly rotated platform axis $S_1$
- $\theta(t)$ is the “pitch” component (rotation around $y_{S_1}$) and is applied second to the recently rotated axis $S_1$, resulting in a newly rotated axis $S_2$.
- $\phi(t)$ is the “roll” component (rotation around $x_{S_2}$) and is applied last to the recently rotated axis $S_2$, resulting in the final rotated axis $S_3$. 
This can be shown graphically below:

The order of rotations is yaw, pitch then roll.
5  UDP Generic Data Receiver Emulator

STOP – Before proceeding, the LabVIEW Runtime Engine must be loaded on to the computer to be used with the UDP Generic Data Receiver Emulator. If this step has not been taken, then refer to Appendix I before proceeding.

Extract the UDP Generic Data Receiver Emulator “rar” package to any directory (eg C:\CKAS Application Software).

Start the Executable “UDP Generic Data Receiver Emulator.exe” and the following screen will appear (assuming Appendix I – Installation of LabVIEW Runtime Engine has been followed correctly).

Certain key aspects of this software are described next.
This button will stop and exit the software.

The PAUSE light indicates that no UDP data is being received.

The UDP port being listened to.

The format of the data stream required.

The actual UDP data packet being currently received.

Graphs of various data being received.
5.1 Data Packet

5.1.1 Format

The format of the UDP data Packet is a Plain Ascii Text Packet with the data stream to contain the parameters required in a comma separated format.
(Note: ensure the client software is NOT sending binary coded data – it must be plain ascii)

5.1.2 Address

The IP address of the UDP Data Receiver host computer must be specified by the client sender (can be on the same computer), and also the port number (1254 by default)

5.1.3 Data rate

The UDP packets must be sent to the host at MINIMUM 20Hz (maximum 50ms between packets). The sample rate of the CKAS Platform Driver software is 20Hz. A send rate which is slower than this will cause the software to intermittently pause. The MAXIMUM SIMULATED FREQUENCY is 10Hz at the motion system.

5.1.4 Precision

The format of the packet data floating points should be 0.6f (or 6 digits after the decimal point). The UDP Emulator will only read 6 decimal places, so there is no benefit in including more on the host side.

5.1.5 Parameters

The earlier mentioned parameters (see 4.3 ) must ALL be sent to the UDP Emulator. If a particular parameter is not needed, then it should be sent as zero. The UDP packet should have the following format:

x'(B), y'(B), z'(B), x''(B), y''(B), z''(B), R11(W->B), R12(W->B), R13(W->B), R21(W->B), R22(W->B), R23(W->B), R31(W->B), R32(W->B), R33(W->B), R(B), Q(B), P(B), R'(B), Q'(B), P'(B), x(S), y(S), z(S), yaw(S), pitch(S), roll(S)

As an example, assume there is a body which moves at 9m/s forwards, is accelerating forwards at 3m/s² and is aligned with the world axis and does not rotate at all. In this case, the UDP packet would look like the following:
5.1.6 Implementation

A possible implementation of the client side packet sender might look something like that depicted below.

- Fetch data from simulation engine in floating point format (includes velocities, acceleration, etc)
- Convert data to text and comma separate, eg `sprintf( "%.6f, %.6f,...", x, y...)`
- Send string over UDP to correct IP address and Port, Then repeat at least every 50ms

(Note: There should be no carriage returns; they have been inserted in this file to make the data more readable)
6 Appendix I – Installation of LabVIEW Runtime Engine

Any computer which will run the UDP Emulator software must have the LabVIEW Runtime Engine software installed first. The LabVIEW Runtime Engine should be downloaded from the CKAS Mechatronics website before proceeding from:


1. Execute this file and follow the installation procedure.

End of Document.