1. **Data Collection**

The Applanix can collect data in multiple methods; however, only two methods were used in data collection during DEEPWAVE. The primary method of collection was Ethernet logging, with the use of a PC card being the secondary.

 The Applanix system had multiple issues with data collection. An issue with GPS satellite recognition was resolved prior within the first few flights, however the system was not recording data in a readable format. The data must be recorded in “PosPAC” format in order for the processing software to be able to read it. However, most flights were recorded as “Trimble Real Time”. It is not believed that the data from the flights with “Trimble Real Time” format will be able to be recovered. A summary of which flights had successful data collection can be found in Table 1.1.

To resolve the data formatting issue, a number of adjustments were made to the startup procedure. The most significant change was the addition of the PC card in RF19. Although initially the system failed to write to the PC card, if the PC card was present for the entire initialization process at some point the system would start automatically recording data to the PC card. Around the same time as the addition of the PC card, the amount of time given for the system to boot was increased from 30 minutes to almost two hours. Around RF23 it was discovered that if power was given to the system earlier and then the system was left to initialize the Applanix would start writing data automatically to the PC card. After the system had started to write data to the PC card, Ethernet logging could be started and would be successful in regards to the type of data file created. More information on the startup and data collection process can be found in appendix A[[1]](#footnote-1).

Table 1.1: Data logging success

|  |  |  |
| --- | --- | --- |
| **Flight Name** | **Ethernet Logging** | **PC Card Logging** |
| RF01 | No | No |
| RF02 | No | No |
| RF03 | No | No |
| RF04 | No | No |
| RF05 | No | No |
| RF06 | No | No |
| RF07 | Yes | No |
| RF08 | Yes | No |
| RF09 | No | No |
| RF10 | No | No |
| RF11 | No | No |
| RF12 | No | No |
| RF13 | Yes | No |
| RF14 | No | No |
| RF15 | Yes | No |
| RF16 | No | No |
| RF17 | No | No |
| RF18 | No | No |
| RF19 | No | Yes |
| RF20 | No | Yes |
| RF21 | No | Yes |
| RF22 | No | Yes |
| RF23 | Yes | Yes |
| RF24 | No | Yes |
| RF25 | Yes | Yes |
| RF26 | Yes | Yes |

Table 1.1 Summary of which method of data logging was successful for each flight

1. **Data Processing**

Processing involves two steps. The first is through the PosPAC software. This converts the raw data into the processed format. Currently, RAF only has access to the more limited processing software. However, one flight (RF15) was processed using the more advanced processing by the University of Wyoming. Preliminary analysis shows no significant difference in the data provided by either processing method. The second step in the data processing procedure is to convert the output data into a readable ASCII format[[2]](#footnote-2). During this step, the calculation for true heading is performed[[3]](#footnote-3). However, this calculation had difficulties when the platform heading crossed through 180 degrees.

Applanix True Heading RF15

Corrected, 1 Hz

 Fig 2.1 : Difference between corrected (right) and uncorrected (left) true heading data. It should be noted that the uncorrected scale is from -180 to +180 while the corrected scale is from 0 to 360.

There was no obvious pattern to the fluctuations in the heading. To correct for as much of the noise as possible, a fix was implemented during the merge into the NetCDF files. If the calculated true heading differed from the aircraft true heading by more than 5 degrees, 360 was added to the number[[4]](#footnote-4). This reduced the frequency of the fluctuations observed in the true heading data, however some still remain. It was hoped that the more sophisticated processing software could resolve this, but that had been shown not to be true.

During data processing, the time output by the Applanix data system needed to be converted and corrected to match the time given by the other systems on the aircraft. First the time had to be converted from UTC seconds after midnight Sunday to UTC seconds after midnight. Then, a correction of -16 seconds was applied to account for the drift between GPS time and UTC.

Not all of the flights were able to be processed due to the formatting issues mentioned under data collection. In addition, three flights of PC Card files were not able to be read in by the processing software due to problems that arose during the concatenation process. The PC card file for RF26 is incomplete, and the Ethernet logging file for the same flight has been misplaced. It is not believed to have been a data recording error, but a data transference error that lost the copies of the files. Data logging was successful on the return ferry flights; however this data has also been lost. Data from the initial ferry and test flights was also unable to be processed due to formatting errors.

Table 2.1 Data processing success of each format

|  |  |  |
| --- | --- | --- |
| **Flight Name** | **Ethernet Logging** | **PC Card Logging** |
| RF07 | Yes[[5]](#footnote-5) | N/A |
| RF08 | Yes | N/A |
| RF13 | Yes | N/A |
| RF15 | Yes | N/A |
| RF19 | No | No |
| RF20 | No | Yes |
| RF21 | No | No |
| RF22 | No | Yes |
| RF23 | Yes | Yes |
| RF24 | No | No |
| RF25 | Yes | Yes |
| RF26 | Yes | Data missing after 20 minutes |

Table 2.1 Summary of data files that were successfully processed. N/A indicates the time period before data logging to the PC card was successful.

The Ethernet logging is the preferred method of recording data[[6]](#footnote-6), as data is recorded to the PC card in approximately ten minute increments resulting in over 40 files for each research flight. These files can be concatenated into a single file that encompasses the entire flight; however for three flights the concatenation was unsuccessful for indeterminate reasons. It is believed to be possible to process each segment individually, however, the time investment of that endeavor has not been deemed necessary. Only the files that were read into the processing software as “PosPac” were able to be processed as is summarized in Table 2.1.

1. **Data Merging**

The original merge of the data into the NetCDF files was done using MATLAB code. Recent efforts have been focused on constructing code in R to complete the merge of the sample rate data. Table 3.1 summarizes the current progress on the data merge.

To accomplish the merge of the data, the data had to be interpolated back to the whole second and have the frequency reduced from 10 Hz to 1 Hz. This was done using a linear interpolation of the two data points nearest to the desired whole second; one data point from before the desired whole second data point and one data point from after. This interpolation[[7]](#footnote-7) produces an array of data that is then called in MATLAB for the next step which entails the actual data merge process.

The actual data merge involves defining the variables, providing the data and the variable attributes before inserting the new variables into the appropriate NetCDF file. Figure 3.1 demonstrates the process that must be performed to each the variables to be added the NetCDF.

 

Figure 3.1: Screenshot of a segment of applanix\_varadd.m

Multiple obstacles were encountered during this process. As the Applanix was not started or stopped in unison with the ADS system, the data array had to be trimmed and added to in order to match the length of the data from the ADS system. Care had to be taken to ensure that the data was placed properly in time, resulting in an extra step of correlating array index to data file time stamp. This resulted the need for the merge code to be edited for each individual flight.

Due to issues with number formatting when switching between environments of NetCDF and MATLAB, a copy was made of the “ZERO” variable from the NETCDF file. This was then added to the array of Applanix data to ensure that the proper number format was created in the NetCDF environment while not altering the data value.

At this point in the data merge, the correction to the true heading calculation discussed in data processing was applied.

The next step is to merge the sample rate data. However, to do this a multi-dimensional variable must be defined and due to limitations in MATLAB NetCDF code library this does not appear to be possible under the current methodology. The decision was made to try the more adaptable R language. The shortfall of changing to R is that R cannot interpret the raw data file. R is successful in the interpretation of the ASCII file generated in MATLAB. While not ideal in terms of simplicity or continuity, using MATLAB to convert the Applanix data file into ASCII format before have R work the data has been successful. The code for the sample rate data merge is nearly complete[[8]](#footnote-8); however a successful merge has not yet been performed.

Table 3.1: Data merge progress

|  |  |  |
| --- | --- | --- |
| **Flight Name** | **Ethernet Logging** | **PC Card Logging** |
| 1 Hz | 10 Hz | 1Hz | 10 Hz |
| RF07 | Yes | No | N/A | N/A |
| RF08 | No | No | N/A | N/A |
| RF13 | No | No | N/A | N/A |
| RF15 | Yes | No | No | No |
| RF19 | N/A | N/A | N/A | N/A |
| RF20 | N/A | N/A | No | No |
| RF22 | N/A | N/A | N/A | N/A |
| RF23 | No | No | No | No |
| RF24 | N/A | N/A | N/A | N/A |
| RF25 | No | No | No | No |
| RF26 | No | No | N/A | N/A |

Table 3.1 Summary of data merge into netcdf status. N/A indicates that the data is unavailable to complete the merge, No indicates the merge has not been started yet or is as of yet unsuccessful.

Appendix A: Applanix Operations

**START-UP**

* Power on master PC – switch on front panel (or rear) – blue light comes on
* Enter password in PC: hiaper
* **If using PC card**, put PC card into Applanix
* Power on Applanix at least 60 minutes before flight by pressing and holding the green button in the upper right until the sys light flashes
* Use VNC to display desktop on aeros laptop:
	+ Double click tiger vnc
	+ When prompted for server name, enter applanix-pc
	+ Press OK; vnc connection should then open
* Click POS-AV icon (lower left)
* Wait to see address 128.46.118.136 in the window
* Click connect (green plug icon at top) - standby icon should turn red

**Standby Icon**



**Starting Data Logging**

* Data logging is started up as near to doors close as possible
* INT light on Applanix unit should be green before starting logging (See pg. 4 for diagram)
* If using PC card, start PC card logging before Ethernet logging
* To Start PC Card Logging:
* Pull-down menu Logging => PC card logging
	+ If the Applanix has started logging, stop logging then restart with the proper groups and file name
* Press POSPac and verify which groups are selected and change frequency to 10Hz
* Change File name to PROJRF##.000 and then press start logging
* start Ethernet logging AFTER logging is successfully started to PC card
* To Start Ethernet Logging:
* Pull-down menu Logging => Ethernet logging => Secondary tab (log file will show “Default2”) => Browse
* Click on 2014.152\_DEEPWAVE folder; type in file name: DWAVxxxx (the xxxx represents flight type and number)
* Verify that the append button is checked
* Press POSPac, verify appropriate groups are checked and change frequency to 10 Hz
* Press Start logging
* Minimize logging status window (shows that data is logging)
* IF NOT USING PC CARD: POSAV error : logging device error => OK



**Monitoring**

* PC Card: Writing (On main POS-AV display)
* Check installation parameters:

Settings => Installation => Lever arms

* 

EXAMPLE

* Settings => Installation => Tags

Check Time Tag 1 = GPSTime

And also all other time tags are GPS Time

* View => GPS data and verify we have greater than 8 satellites
* verify the SNR for all satellites are > 30 for L1 and > 18 for L2
* In normal operations, IMU status goes from failure to OK,

GPS Status goes from not available to C/A

**Applanix Indicator lights - After Applanix is properly initialized**

SYS – GREEN (flash or steady)

IMU – GREEN

GPS – YELLOW (steady)

PPS – GREEN (flash)

LAN – GREEN (flash)

PWR – GREEN (steady)

INT –**GREEN**

EXT **GREEN (if PC Card present & writing)**

**Applanix Controller Front Panel Switches and Indicators**

****

**SHUTDOWN**

* After plane has stopped, continue logging with POS-AV software on for at least 10 minutes.
* After 10 minutes, stop logging

Logging => Ethernet logging => stop logging and note time on log sheet

* Close POS-AV software

File => exit

Message log has changed, would you like to save it? => yes

 => TF/RFXX\_log.txt => okay

* Insert the flash drive in front of CPU
* Go to **D:**/gpsdata/2014.152\_DEEPWAVE
* Copy file DWAVxxxx and rfxx\_log.txt to USB drive
* Turn off computer
* Turn off Applanix => press and hold power button on the front until power button starts blinking.

Appendix B: read\_applanix\_vnav2.m, original by Jennifer Haas

function [C] = read\_applanix\_vnav2(filename,fileout)

%EDITED FOR RF23 Monday July14 for Stuart

% EDIT August 2014 Kelly Schick

% Read a binary applanix real time solution file

% Run POSPAC software to read in recorded applanix file

% It automatically creates a binary file for real time INS solution

% Look for file in project folder, ie if the project is named "current"

% PosPac MM5\current\Mission\Extract\vnav\_Mission.out

% call function with filename = 'vnav\_Mission.out'

% call the function assuming filename is in the current working directory

% or call with full pathname to file

% Format for vnav file and rms error file vrms file is in the

% POSPac MMS GNSS-Inertial Tools.pdf manual in section 10.

% Output:

% call function with fileout = the output ascii file name

%

% Output array B has dimension [ndata,17] where the 17 columns are:

% time\_(seconds\_of\_week)

% latitude\_(deg)

% longitude\_(deg)

% altitude\_(meters)

% x\_velocity\_(meters/second)

% y\_velocity\_(meters/second)

% z\_velocity\_(meters/second)

% roll\_(deg)

% pitch\_(deg)

% platform\_heading\_(deg)

% wander\_angle\_(deg)

% x\_body\_acceleration\_(meters/second^2)

% y\_body\_acceleration\_(meters/second^2)

% z\_body\_acceleration\_(meters/second^2)

% x\_body\_angular\_rate\_(deg/second)

% y\_body\_angular\_rate\_(deg/second)

% z\_body\_angular\_rate\_(deg/second)

fid = fopen(filename,'r');

% Read all the values into 1D array A assuming double floating point

A=fread(fid,'double','n');

B = reshape(A,17,[])';

fclose(fid);

[a,b]=size(B);

C=zeros(a,b+1);

C(1:a,1:b)=B; %create a new matrix for column manipulation %%EDIT 8 AUG 2014 KES

% output the ascii data

fid2 = fopen(fileout,'w');

%fprintf(fid2,['time\_(seconds\_of\_week) lat\_(deg) long\_(deg) ' ...

 %'ht\_ellipsoid\_(meters\_WGS84) x\_vel\_(m/s) y\_vel\_(m/s) z\_vel\_(m/s) ' ...

 %'roll\_(deg) pitch\_(deg) heading\_(deg) wander\_angle\_(deg) ' ...

 %'x\_body\_accel\_(m/s^2) y\_body\_accel\_(m/s^2) z\_body\_accel\_(m/s^2) ' ...

 %'x\_body\_angular\_rate\_(deg/s) y\_body\_angular\_rate\_(deg/s) ' ...

 %'z\_body\_angular\_rate\_(deg/s)\n']); %EDIT 7 AUG 2014 KES

fprintf(fid2,['time\_(UTC\_seconds\_of\_day) lat\_(deg) long\_(deg) ' ... %EDIT 7 AUG 2014 KES

 'ht\_ellipsoid\_(meters\_WGS84) x\_vel\_(m/s) y\_vel\_(m/s) z\_vel\_(m/s) ' ...

 'roll\_(deg) pitch\_(deg) heading\_(deg) wander\_angle\_(deg) true\_heading\_(deg) ' ... %EDIT 8 AUG 2014 KES

 'x\_body\_accel\_(m/s^2) y\_body\_accel\_(m/s^2) z\_body\_accel\_(m/s^2) ' ...

 'x\_body\_angular\_rate\_(deg/s) y\_body\_angular\_rate\_(deg/s) ' ...

 'z\_body\_angular\_rate\_(deg/s)\n']);

% Change from radians to degrees everywhere

C(:,2) = B(:,2)\*180/pi();

C(:,3) = B(:,3)\*180/pi();

%B(:,8) = B(:,4)\*180/pi();

C(:,8) = B(:,8)\*180/pi(); %EDIT 5 AUG 2014 KES

C(:,9) = B(:,9)\*180/pi();

C(:,10) = B(:,10)\*180/pi(); %platform heading

C(:,11) = B(:,11)\*180/pi(); %wander angle

%add a column to calculate true heading by subtracting wander angle from

%platform heading %%EDIT 8 AUG 2014

C(:,12) = C(:,10)-C(:,11);

% for j=1:a

% if 0<= C(j,10) && C(j,10) < 180

% C(j,12) = (360+C(j,10))-C(j,11);

% elseif 180 <= C(j,10) && C(j,10) <=360

% C(j,12)=C(j,10)-C(j,11);

% end

% if C(j,12) > 360

% C(j,12)=C(j,12)-360;

% elseif C(j,12)<0

% C(j,12)=C(j,12)+360;

% end

% end

C(:,13) = B(:,12);

C(:,14) = B(:,13);

C(:,15) = B(:,14);

C(:,16) = B(:,15)\*180/pi();

C(:,17) = B(:,16)\*180/pi();

C(:,18) = B(:,17)\*180/pi();

%convert time to seconds UTC, not seconds of week %%EDIT 7 AUG 2014 KES

days = 4 \* 86400\*ones(a,1); %Sunday = 0, Monday =1, Tuesday = 2, Wednesday =3, Thursday = 4, Friday =5, Saturday =6

C(:,1) = B(:,1) - days -16; %GPS time is 16 seconds ahead of UTC

fprintf(fid2,'%10.4f %14.9f %14.9f %12.5f %10.5f %10.5f %10.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f %11.5f\n',C'); %EDIT 8 AUG 2014 KES

%changed to matrix C and added a %11.5f to incoroporate new variable.

fclose(fid2);

Appendix C: applanix\_frequencyconvert.m

%Kelly Schick 13 August 2014

%Goal: write code that converts 10 Hz data into 1 Hz data

function [outmatrix]=applanix\_frequncyconvert(infile)

%extract 10 data point from matrix/file. these will become the "old points"

inmatrix=infile;%dlmread(infile,'',1,0);

[rows,columns]=size(inmatrix);

endpoint=floor(rows/10)-1;

slope=zeros(1,columns-1);

outmatrix=zeros(endpoint,columns);

old\_group=inmatrix(1:10,:);

%calculate averages of all variables -put into array (old\_average)

old\_averages=mean(old\_group);

%enter iterative process

for i=1:endpoint

 %extract set of new points

 new\_group =inmatrix((i\*10)+1:(i+1)\*10,:);

 %calculate averages of all variables in new set of 10

 new\_averages = mean(new\_group);

 % ??FIND TIME DESIRED (WHICH WHOLE SECOND?) ??

 time\_desired = floor((new\_averages(1)+old\_averages(1))/2);

 %find slope of linear equation created by old & new averages

 time\_run = new\_averages(1)-old\_averages(1);

 var\_rise = new\_averages(2:columns)-old\_averages(2:columns);

 for j=1:(columns-1)

 slope(j)=var\_rise(j)/time\_run;

 end

 %solve for value at desired time

 %(y-y\_avg)=slope\*(time\_desired-time\_avg)

 outmatrix(i,1)=time\_desired;

 outmatrix(i,2:columns)=(slope'\*(old\_averages(1)-time\_desired)+new\_averages(2:columns)')';

 %set new points to old

 old\_averages=new\_averages;

 %write values out at whole seconds to new file

 %repeat until end of file ??POSSIBLY limit number of iterations based on

 %divisions by ten so as to avoid out of bounds errors?

end

Appendix D: applanix\_varadd\_rf##.m[[9]](#footnote-9)

%Writes variables created by read\_applanix\_vnv.m to netcdf files

%read in data from matrix created by applanix\_frequency convert

%[outmatrix]=applanix\_frequncyconvert(infile)dlmread or fread ascii file, then run frequency convert after making it

%onehertz=applanix\_frequncyconvert('RF07\_ASCII.txt');

%work as a function

A=onehertztest;

[a,b]=size(A);

stime=7688;%15030; %UTC start time of netcdf file

i1=A(1,1)-stime; %index of applanix start time

i2=i1+a-1; %index of applanix end time

ncid = netcdf.open('/h/eol/schick/DEEPWAVE/netcdffiles/DEEPWAVErf15.nc','NC\_WRITE')

%extract time dimension from netcdf file

time\_dim\_ID=netcdf.inqDimID(ncid,'Time');

[dummy, time\_dim\_len] = netcdf.inqDim(ncid, time\_dim\_ID)

%Define new variables (lat, lon, height, x\_vel,y\_vel,z\_vel,roll, pitch,

%heading, x\_accel, y\_accel, z\_accel, x\_angular, y\_angular, z\_angular

% Read in ZERO var (used to preserve data format

varid\_zero=netcdf.inqVarID(ncid,'ZERO');

ZERO=netcdf.getVar(ncid,varid\_zero);

if time\_dim\_len < a

 a = time\_dim\_len

 i2=i1+a;%-1;

end

%write new variables to file

%Latitude

netcdf.reDef(ncid);

LAT\_APP\_ID=netcdf.defVar(ncid,'LAT\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,LAT\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,LAT\_APP\_ID,'long\_name','Latitude - Applanix');

netcdf.putAtt(ncid, LAT\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

LAT\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

LAT\_APP(i1:i2,1)=ZERO(1:a,:)+A(1:a,2);%write new variable preserving format

netcdf.putVar(ncid,LAT\_APP\_ID,LAT\_APP);

%longitude

netcdf.reDef(ncid);

LON\_APP\_ID=netcdf.defVar(ncid,'LON\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,LON\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,LON\_APP\_ID,'long\_name','Longitude - Applanix');

netcdf.putAtt(ncid, LON\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

LON\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

LON\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,3);%write new variable preserving format

netcdf.putVar(ncid,LON\_APP\_ID,LON\_APP);

%ellipsoid height

netcdf.reDef(ncid);

HT\_APP\_ID=netcdf.defVar(ncid,'HT\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,HT\_APP\_ID,'units','meters\_WGS84');

netcdf.putAtt(ncid,HT\_APP\_ID,'long\_name','Ellipsoid Height - Applanix');

netcdf.putAtt(ncid, HT\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

HT\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

HT\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,4);%write new variable preserving format

netcdf.putVar(ncid,HT\_APP\_ID,HT\_APP);

%x velocity

netcdf.reDef(ncid);

XVEL\_APP\_ID=netcdf.defVar(ncid,'XVEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,XVEL\_APP\_ID,'units','m/sec');

netcdf.putAtt(ncid,XVEL\_APP\_ID,'long\_name','X velocity - Applanix');

netcdf.putAtt(ncid, XVEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

XVEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

XVEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,5);%write new variable preserving format

netcdf.putVar(ncid,XVEL\_APP\_ID,XVEL\_APP);

%y velocity

netcdf.reDef(ncid);

YVEL\_APP\_ID=netcdf.defVar(ncid,'YVEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,YVEL\_APP\_ID,'units','m/sec');

netcdf.putAtt(ncid,YVEL\_APP\_ID,'long\_name','Y Velocity - Applanix');

netcdf.putAtt(ncid, YVEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

YVEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

YVEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,6);%write new variable preserving format

netcdf.putVar(ncid,YVEL\_APP\_ID,YVEL\_APP);

%z velocity

netcdf.reDef(ncid);

ZVEL\_APP\_ID=netcdf.defVar(ncid,'ZVEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,ZVEL\_APP\_ID,'units','m/sec');

netcdf.putAtt(ncid,ZVEL\_APP\_ID,'long\_name','Z Velocity - Applanix');

netcdf.putAtt(ncid, ZVEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

ZVEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

ZVEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,7);%write new variable preserving format

netcdf.putVar(ncid,ZVEL\_APP\_ID,ZVEL\_APP);

%roll

netcdf.reDef(ncid);

ROLL\_APP\_ID=netcdf.defVar(ncid,'ROLL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,ROLL\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,ROLL\_APP\_ID,'long\_name','Roll - Applanix');

netcdf.putAtt(ncid, ROLL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

ROLL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

ROLL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,8);%write new variable preserving format

netcdf.putVar(ncid,ROLL\_APP\_ID,ROLL\_APP);

%pitch

netcdf.reDef(ncid);

PITCH\_APP\_ID=netcdf.defVar(ncid,'PITCH\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,PITCH\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,PITCH\_APP\_ID,'long\_name','Pitch - Applanix');

netcdf.putAtt(ncid, PITCH\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

PITCH\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

PITCH\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,9);%write new variable preserving format

netcdf.putVar(ncid,PITCH\_APP\_ID,PITCH\_APP);

%platform heading

netcdf.reDef(ncid);

PHDG\_APP\_ID=netcdf.defVar(ncid,'PHDG\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,PHDG\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,PHDG\_APP\_ID,'long\_name','Platform Heading - Applanix');

netcdf.putAtt(ncid, PHDG\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

PHDG\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

PHDG\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,10);%write new variable preserving format

netcdf.putVar(ncid,PHDG\_APP\_ID,PHDG\_APP);

%wander angle

netcdf.reDef(ncid);

WNDR\_ANG\_APP\_ID=netcdf.defVar(ncid,'WNDR\_ANG\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,WNDR\_ANG\_APP\_ID,'units','deg');

netcdf.putAtt(ncid,WNDR\_ANG\_APP\_ID,'long\_name','Wander Angle - Applanix');

netcdf.putAtt(ncid, WNDR\_ANG\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

WNDR\_ANG\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

WNDR\_ANG\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,11);%write new variable preserving format

netcdf.putVar(ncid,WNDR\_ANG\_APP\_ID,WNDR\_ANG\_APP);

%x acceleration

netcdf.reDef(ncid);

XACCEL\_APP\_ID=netcdf.defVar(ncid,'XACCEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,XACCEL\_APP\_ID,'units','m/sec^2');

netcdf.putAtt(ncid,XACCEL\_APP\_ID,'long\_name','X Acceleration - Applanix');

netcdf.putAtt(ncid, XACCEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

XACCEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

XACCEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,13);%write new variable preserving format

netcdf.putVar(ncid,XACCEL\_APP\_ID,XACCEL\_APP);

%y acceleration

netcdf.reDef(ncid);

YACCEL\_APP\_ID=netcdf.defVar(ncid,'YACCEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,YACCEL\_APP\_ID,'units','m/sec^2');

netcdf.putAtt(ncid,YACCEL\_APP\_ID,'long\_name','Y Acceleration - Applanix');

netcdf.putAtt(ncid, YACCEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

YACCEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

YACCEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,14);%write new variable preserving format

netcdf.putVar(ncid,YACCEL\_APP\_ID,YACCEL\_APP);

%Z acceleration

netcdf.reDef(ncid);

ZACCEL\_APP\_ID=netcdf.defVar(ncid,'ZACCEL\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,ZACCEL\_APP\_ID,'units','m/sec^2');

netcdf.putAtt(ncid,ZACCEL\_APP\_ID,'long\_name','Z Acceleration - Applanix');

netcdf.putAtt(ncid, ZACCEL\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

ZACCEL\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

ZACCEL\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,15);%write new variable preserving format

netcdf.putVar(ncid,ZACCEL\_APP\_ID,ZACCEL\_APP);

%X ANGULAR RATE

netcdf.reDef(ncid);

X\_ANGRATE\_APP\_ID=netcdf.defVar(ncid,'X\_ANGRATE\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,X\_ANGRATE\_APP\_ID,'units','deg/sec');

netcdf.putAtt(ncid,X\_ANGRATE\_APP\_ID,'long\_name','X Angular Rate- Applanix');

netcdf.putAtt(ncid, X\_ANGRATE\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

X\_ANGRATE\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

X\_ANGRATE\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,16);%write new variable preserving format

netcdf.putVar(ncid,X\_ANGRATE\_APP\_ID,X\_ANGRATE\_APP);

%Y ANGULAR RATE

netcdf.reDef(ncid);

Y\_ANGRATE\_APP\_ID=netcdf.defVar(ncid,'Y\_ANGRATE\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,Y\_ANGRATE\_APP\_ID,'units','deg/sec');

netcdf.putAtt(ncid,Y\_ANGRATE\_APP\_ID,'long\_name','Y Angular Rate- Applanix');

netcdf.putAtt(ncid, Y\_ANGRATE\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

Y\_ANGRATE\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

Y\_ANGRATE\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,17);%write new variable preserving format

netcdf.putVar(ncid,Y\_ANGRATE\_APP\_ID,Y\_ANGRATE\_APP);

%Z ANGULAR RATE

netcdf.reDef(ncid);

Z\_ANGRATE\_APP\_ID=netcdf.defVar(ncid,'Z\_ANGRATE\_APP','float',time\_dim\_ID);

%netcdf.putAtt(ncid,RSTBC\_ID,'\_FillValue',-32767.0);

netcdf.putAtt(ncid,Z\_ANGRATE\_APP\_ID,'units','deg/sec');

netcdf.putAtt(ncid,Z\_ANGRATE\_APP\_ID,'long\_name','Z Angular Rate- Applanix');

netcdf.putAtt(ncid, Z\_ANGRATE\_APP\_ID,'SampledRate',10);

netcdf.endDef(ncid);

Z\_ANGRATE\_APP=ZERO-32767\*ones(time\_dim\_len,1); %set variable to NaN, preserving format

Z\_ANGRATE\_APP(i1:i2,1)=ZERO(1:a,:)+A(:,18);%write new variable preserving format

netcdf.putVar(ncid,Z\_ANGRATE\_APP\_ID,Z\_ANGRATE\_APP);

netcdf.close(ncid);

1. All referenced manuals and code can be located in the accompanying appendices or at ~schick/DEEPWAVE/Applanix/ref/ [↑](#footnote-ref-1)
2. See appendix B [↑](#footnote-ref-2)
3. To calculate the true heading, the wander angle is subtracted from the platform heading. [↑](#footnote-ref-3)
4. See appendix D [↑](#footnote-ref-4)
5. Although data logging and processing was successful for RF07, the data appears to have issues that make it unreliable. [↑](#footnote-ref-5)
6. The cause of the formatting issue remains unclear, however it appears to be resolved with the new startup procedure illustrated in Appendix A. [↑](#footnote-ref-6)
7. See appendix C [↑](#footnote-ref-7)
8. Due to the incomplete nature of the R code, it is not included in the first version of this document. [↑](#footnote-ref-8)
9. This code must be manipulated for every flight. The versions shown here is edited specifically for RF15 [↑](#footnote-ref-9)