

Documentation for "nav-only"

Joshua Stubbs

April 5, 2016

1 General

Nav-only is a modification of the main Pyxis code that enables the user to run the receiver quicker by inputting an existing APT (Amplitude Phase Time) binary file. RNX binaries (used for positioning, and is the main file for analysing the results of a run) or KML files (for Google Earth) can be outputted.

1.1 System Requirements

The code has been tested on a 64 bit machine running LinuxMint 17.2, and this is recommended for operation. Ubuntu will likely work too.

Before running anything a few things need to be fixed;

- Run the following commands in a terminal window to install the required packages:
 - `sudo apt-get update; sudo apt-get upgrade -y; sudo apt-get dist-upgrade -y`
 - `sudo apt-get install build-essential doxygen graphviz texlive-latex-extra texlive-latex-base texlive-latex-recommended texlive-fonts-recommended latex-xcolor check`
- Run the script "remakePath.sh" by navigating to (in a terminal window) `stand_alone_nav/largeFiles` and then running it by typing `./remakePath.sh`

2 Important Files

The code should be setup to run everything, without having to change any file paths. However, if you do want to change how the code is run, the following files are important:

The first file of importance is "conf.h" (located in `/src/conf`). Make sure that **NAVDEBUG** is set to 1, and then that the appropriate outputs (such as RNX) are enabled. Note that **LOGAPT** is set to 0 regardless of selection.

The second file of importance is "conf.swc.h" (located in `/src/conf`). This is where you will give the input/output file paths. Even though **APTOUT** was disabled in "conf.h" the file path needs to be defined here too, as it will actually specify the *input* path to the existing APT file. **KMLOUT** and **RNXOUT** will specify the output paths of the KML file and RNX binaries. Note that in the code the input file name of **APTOUT** has to use the `_%d_%d` naming convention, or the code will bug (described in more detail in section 5.1 of this document). The actual file has to use the `_0_0` naming, though.

Included in the folder is also an APT binary needed to run the code.

3 Building/Running

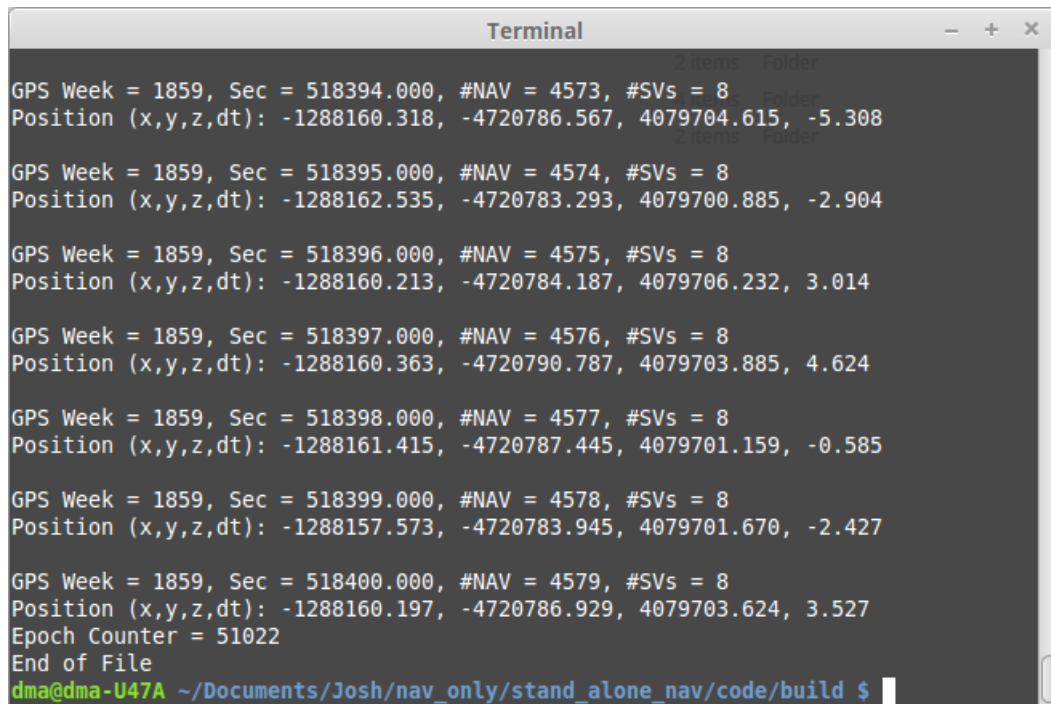
The default setup of the code will read the APT file located in "stand_alone_nav/input" and output a RNX binary to "stand_alone_nav/output".

Below the two main methods for building and running the code are described.

3.1 Terminal

Building and running the code through the terminal is simple, but offers little assistance in terms of debugging. Editing of files can be done through a text editor of your choice. The necessary steps are:

1. Open a terminal window
2. Navigate to `stand_alone_nav/code/build/` directory using the `"cd"` command
3. run the command `"make clean;make"` to build the code (needs to be done each time something is changed in any file)
4. run the nav-only code with the command: `"../bin/nav/navexe"`
5. messages can be seen in the terminal window, and will give a prompt when done. See figure 1 for what successful operation looks like.



```
Terminal
GPS Week = 1859, Sec = 518394.000, #NAV = 4573, #SVs = 8
Position (x,y,z,dt): -1288160.318, -4720786.567, 4079704.615, -5.308

GPS Week = 1859, Sec = 518395.000, #NAV = 4574, #SVs = 8
Position (x,y,z,dt): -1288162.535, -4720783.293, 4079700.885, -2.904

GPS Week = 1859, Sec = 518396.000, #NAV = 4575, #SVs = 8
Position (x,y,z,dt): -1288160.213, -4720784.187, 4079706.232, 3.014

GPS Week = 1859, Sec = 518397.000, #NAV = 4576, #SVs = 8
Position (x,y,z,dt): -1288160.363, -4720790.787, 4079703.885, 4.624

GPS Week = 1859, Sec = 518398.000, #NAV = 4577, #SVs = 8
Position (x,y,z,dt): -1288161.415, -4720787.445, 4079701.159, -0.585

GPS Week = 1859, Sec = 518399.000, #NAV = 4578, #SVs = 8
Position (x,y,z,dt): -1288157.573, -4720783.945, 4079701.670, -2.427

GPS Week = 1859, Sec = 518400.000, #NAV = 4579, #SVs = 8
Position (x,y,z,dt): -1288160.197, -4720786.929, 4079703.624, 3.527
Epoch Counter = 51022
End of File
dma@dma-U47A ~/Documents/Josh/nav_only/stand_alone_nav/code/build $
```

Figure 1: Successful execution as seen when run in the terminal

3.2 Eclipse

1. Open Eclipse
2. Create a new workspace. This will create a folder of your project name that will be your working directory
3. Click file → new → project and create a new "C makefile project with existing code".
4. Browse to the "code" folder, and make sure that "C" is selected. Select to use "Linux GCC" as toolchain. Give your project an appropriate name too
5. Click the "Workbench" arrow in the top right corner of the window to open the workspace
6. Click "Open perspective" in the top right corner and select "C/C++"
7. You can now browse your project using the left pane, and edit files as wanted. Set up file paths as described in section 2 of this document
8. Right click your project name in the project explorer and select properties → C/C++ Build
9. Under "Build Location" click workspace and select the "build" folder in your code folder. Click "Apply" and "OK" to return to the workspace

10. clean project by selecting the menu option: Project → clean → clean all
11. click the hammer to build the project
12. select "always run in background" and then click "run in background". Select the "Console" view at the bottom to see what's going on
13. set up how to run the code by going to the "Run" menu and then selecting "Run Configurations"
14. select "C/C++ Application", then click "New launch configuration"
15. (you might have to select your project folder under "Project" before proceeding to the next step)
16. click Browse, and then navigate to the location of "project_name"/bin/nav/ and select "navexe" (where "project_name" is the name of your project)
17. click "Apply" and then run to run your code
18. you can now run nav-only by pressing the green play button at top of screen
19. save, clean, build and run the code whenever a file is changed

4 Post Processing

To analyse the results of running the above mentioned code some post processing tools are provided in the "stand_alone_nav/postProcessing" folder. Their function and operation is described in this section.

MATLAB is required to run all of the post processing tools. Both scripts have options to change file paths and file names for the input file. As the scripts are currently set up, however, they should work straight away with the generated RNX file.

4.1 Analyse RNX binary

The folder "RNX_Analyser" contains the necessary script to analyse the RNX binary file without converting it.

Running "AnalysisLSSScript.m" should show a variety of plots and save them in "stand_alone_nav/output/RNX_Analysis"

4.2 Convert RNX binary to Rinex

The folder "RNX2Rinex" contains the necessary script to convert your newly created RNX binary to a RINEX file. RINEX is a common file type to use for GPS applications.

Running "lssBin2RunexScript.m" should create a RINEX file in the "stand_alone_nav/output/RINEX_conversion" folder.

5 Known Errors

5.1 Problem when running "nav-only" with anything else than 0_0 file

This issue is a three-part problem;

First, only one APT file can be run at the time. When the end of a file is reached, the next one will not be automatically read in as when running the code for the full receiver.

Secondly, only the 0_0 APT file can be read. If any other APT file is input, an error will occur when the ephemeris is updated (see figure 2 below).

Thirdly, in order to even read the 0_0 APT file the input file path has to use the `_%d_%d` convention. I.e, your input file is called "aptBinaries_0_0.bin", but you have to give the input file name as `aptBinaries_%d_%d.bin`.

```
<terminated> nav [C/C++ Application] /home/dma/workspace/dma_stable/bin/nav/navexe (3/16/16 2:35 PM)

GPS Week = 1884, Sec = 86400.000, #NAV = 86364, #SVs = 7
Position (x,y,z,dt): -1288159.949, -4720786.908, 4079689.880, 1.575
Epoch Counter = 950443
Ephemeris update: TOW = 60, PRN = 9
Ephemeris update: TOW = 72, PRN = 16
Ephemeris update: TOW = 132, PRN = 30
Ephemeris update: TOW = 156, PRN = 23
Ephemeris update: TOW = 180, PRN = 7
Ephemeris update: TOW = 306, PRN = 27
Processed 860.047 seconds
Wrong subframe ID(7), IF DATA IS NOT RELIABLE!
Ephemeris update: TOW = 930, PRN = 8
Wrong subframe ID(0), IF DATA IS NOT RELIABLE!
Processed 870.048 seconds
Wrong subframe ID(6), IF DATA IS NOT RELIABLE!
Wrong subframe ID(7), IF DATA IS NOT RELIABLE!
Wrong subframe ID(7), IF DATA IS NOT RELIABLE!
Wrong subframe ID(0), IF DATA IS NOT RELIABLE!
Wrong subframe ID(0), IF DATA IS NOT RELIABLE!
Wrong subframe ID(0), IF DATA IS NOT RELIABLE!
Wrong subframe ID(0), IF DATA IS NOT RELIABLE!
Processed 880.048 seconds
Ephemeris update: TOW = 7400, PRN = 28
```

Figure 2: Error at ephemeris update when not using `_%d_%d` system

5.2 Eclipse Errors

Eclipse will sometimes complain about errors on line 0 of the code, but these do not seem to impair operation of the code.

There is also the possibility of errors occurring when a file is opened and edited in eclipse that contains true or false statements. The code can still be built and run however.