ENG 4350 6.0 FW16/17- Space Hardware

Lab P3 – Software Development Planning/Coordination

Submitted by:

Keith Menezes & James Brook

Submitted to:

Prof. Hugh Chesser

Contents

Purpose

The purpose of the document is to synthesize the P1 and P2 modules to outline the development plan that our main program will call to perform the preliminary steps of the satellite tracking process.

Outline

User Input Parser

The user is required to have some input into the code. We have narrowed it down to 4 files that are needed:

- Celestrak Two-Line Element (TLE) of the ~32 GPS Satellites
- Algonquin Radio Observatory Station File
- Tracking Schedule
- Link Inputs

Two Line Element (TLE)

The TLE format has been created by the North American Aerospace Defense Command (NORAD) which tracks objects greater than 30cm in size. The TLE is formatted in 3 lines with the name of the satellite in the first, and orbital parameters and spacecraft characteristics in the second and third.

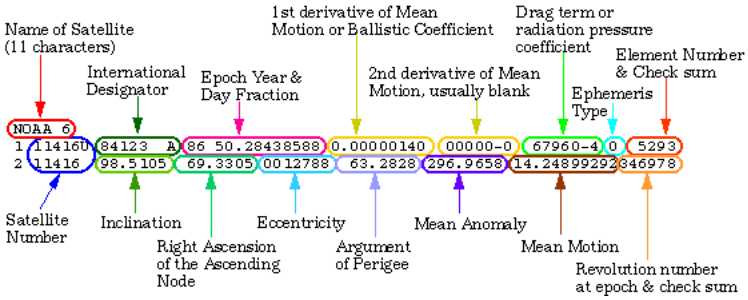

Figure 1: Two Line Element Set Format

1st line contains:

- Name of the satellite (up to 11 characters)

2nd line contains:

- Satellite number
- International designator
- Epoch year and day fraction
- 1st derivative of mean motion or ballistic coefficient
- 2nd derivative of mean motion (usually blank for this section)
- Drag term or radiation pressure coefficient
- Ephemeris type
- Element number and check sum

3rd line contains:

- Satellite number

- Inclination
- Right ascension of the ascending node
- Eccentricity
- Argument of perigee
- Mean anomaly
- Mean motion
- Revolution number at epoch and check sum

The user input parser will extract the TLE information by calling our int ReadNoradTLE() function.

```
int ReadNoradTLE(Satellite sats[], char *file) {
	FILE *fp = fopen(file,"r+");
	for (int i = 0; i < 32; i++) {
		ReadSingleNoradTLE(&sats[i], fp);
	}
	fclose(fp);
	return 0;
}
```

It can be seen that there at maximum going to be 32 satellites in the TLE file from Celetrack and that the function takes an array of Satellite structures. Therefore, the function uses a helper method ReadSingleNoradTLE which is called to read a single TLE within the file and populate each individual satellite with its respective orbital parameters.

```
int ReadSingleNoradTLE(Satellite *sat, FILE *fp){…}
```

For reference, the Satellite struct contains the following parameters.

```
typedef struct Satellite {
	char name[128]; //Satellite Name
	double refepoch; //Epoch year and day fraction
	double incl; //Inclination
	double raan; //Right Ascension of the Ascending Node
	double eccn; //Eccentricity
	double argper; //Argument of Perigee
	double meanan; //Mean Anomaly
	double meanmo; //Mean Motion
	double ndot; //1st derivative of mean motion or ballistic coefficient
	double nddot6; //2nd derivative of mean motion
	double bstar; //Drag term or radiation pressure coefficient
	int orbitnum; //Orbit Number
}Satellite;
```

Algonquin Radio Observatory (ARO) Station File

The station file consists of characteristics of the Algonquin Radio Observatory:

- Name of the ground station
- Latitude
- Longitude
- Elevation
- Diameter of the antenna
- Focal ratio

- Focal length of the antenna
- Surface accuracy
- Beamwidth of the antenna
- Maximum speed of the azimuth and elevation turning
- Maximum and minimum limit of the elevation range

A station struct has been created to hold reference to the station parameters:

```
typedef struct Station {
/* Station File parameters */
	char *name;
	double stnlat; //latitude
	double stnlong; //longitude
	double stnalt; //altitude
	double utc_offset; //UTC offset
	int az_el_nlim;
/* Azimuth Elevation Limit */
	struct az_el_lim {
		double az;	//azimuth
		double elmin; //min elevation limit
		double elmax; //max elevation limit
	} az_el_lim;
	double st_az_speed_max; //max azimuth speed
	double st_el_speed_max; //max elevation speed
} Station;
```

From the main program a call would be made to the int ReadStationFile() which will perform a check to see if the file is within the directory, if not it will throw an error.

Tracking Schedule

The tracking schedule is a file that will output the time interval during our allotted time to track the satellites. This can be in a file or a user input with the format consisting of something like:

- Tracking start date/time: 2017-05-01 07:00:00 //May 1st 7am
- Tracking stop date/time: 2017-05-01 08:00:00 //May 1st 8am
- Output time step (sec): 5.0 // a couple seconds step size

This information will be used for the visibility, where checks will be made to see if the satellite is within the output will be put into a table and be used in the visibility part of the code. If the satellite is within the visibility range that the user wants, then it will progress to computing the antenna pointing computations.

Link Inputs

The link inputs consist of the frequency band center, antenna efficiency, antenna diameter, maximum bandwidth RCV gain and noise temperature. The link inputs will be a text/data file and a helper method will be created to read the file. This link inputs will eventually be used in the link calculations to determine the signal strength and verified with a spectrum analyzer. In the code, we want to find the detailed information of a certain satellite and output onto the screen for the user to observe if they are looking at the correct satellite.

Error Checking

Exceptions or errors will be thrown for various problems that may occur throughout the main functions process. For example, if a file is not found or formatted correctly. Checks will be made to ensure that the ARO is not commanded to exceed its limits in the azimuthal or elevation directions. In P1 a function void ERRMSG() can be thrown when an exception or error is caught. This will display to the console with a designator to specify within which function or process the error occurred, which will make it easier for debugging.

Visibility

The purpose of the visibility is to check which satellite is within range of the ground station. When the satellite is visible, a function will calculate the Acquisition of Signal (AOS) and Loss of Signal (LOS) times, which are times when the satellite is in range of the ARO. These specific satellites will be printed into the AOS/LOS table with their times and names. This will help determine whether the ARO should proceed to point to the specific spacecraft.

If the program chooses not to proceed it will go back to the AOS/EOS table and select the next satellite and calculate its position and velocity. If the code chooses to proceed with this satellite, then it will go to the next step which is the pointing and it will combine with the calculation from the position and velocity calculator to point towards the satellite. The visibility depends on the function int range_ECF2topo() which will calculate the satellite position and velocity in the topocentric system.

Satellite Position and Velocity

The satellite position and velocity calculator is used to determine the location of the satellite using the azimuth, elevation and range and their rates of change with time. The satellite structure array of populated orbital parameters will be used to calculate the position of the spacecraft using the Earth Centered Inertial frame (ECI). Next the station geodetic coordinates which is taken from the station file will be used and converted into the Earth Centered Fixed frame (ECF). The conversions will be taken within the given tracking time interval. Then the satellite will then convert from ECI to ECF frame. A rotation is needed so we will take the current and epoch time from the TLE file and the longitude and latitude from the ground station file to do the derivation. The ECF calculated in the forms of vector from the ground station to the spacecraft will then be converted to topocentric coordinates. Finally, the azimuth, elevation, range rate and Doppler frequency shift of the GPS signal can be calculated on the vector from the . At the end of the transformations, the values calculated will moved to the link calculations part. Functions that will be used in this calculator are listed below.

```
int sat_ECI(Vector *eci_position, Vector *eci_velocity, double eccentricity, double
ecc_anomaly, double a_semi_major_axis, double omega_longitude_ascending_node, double
omega_argument_periapsis, double inclination, double nt_mean_motion);
```

Function sat_ECI() uses all of the orbital parameters. To print out the vector position of the satellite in X, Y and Z direction and the velocity vector in X, Y and Z direction in Earth Centered Inertial frame.

```
int sat_ECF(Vector *sat_ecf_position, Vector *sat_ecf_velocity, double theta_t,
Vector *eci_position, Vector *eci_velocity);
```

Function sat_ECF() is using the position and velocity vector from ECI to calculate the ECF and the theta_t value. This function also uses the matrix function to calculate the position and velocity coordinates in the ECF frame.

```
int station_ECF(Vector *stn_ECF_pos, double station_longitude, double
station_latitude, double station_elevation);
```

Function int station_ECF() takes in the station's latitude and longitude, which will be read from the station file using the ReadStationFile() function. The output of the station_ECF() function will be the X, Y and Z coordinates in ECF as well as the magnitude of the vector.

```
typedef struct LookAngles {
	double azimuth_velocity;
	double elevation_velocity;
	double azimuth;
	double elevation;
} LookAngles;
```

```
void range_topo2look_angles(LookAngles *LA, double azimuth, double elevation, double
azimuth_velocity, double elevation_velocity, Vector *range_topo_position, Vector
*range_topo_velocity);
```

The void range_topo2lookangles() function takes a pointer to the LookAngle struct, azimuth, elevation, with their respective rates of change with time, vectors of topocentric range and velocity X, Y, Z direction and the magnitude. Using an internal function called mycross() which performs the cross product, it will calculate and output the azimuth, elevation, azimuth and elevation velocity.

Pointing

The pointing is a requirement by the ARO, which will be commanded by our code to follow the propagated path of the specific satellite chosen. The output on the screen from the pointing would be a table with the UTC, azimuth, elevation, azimuth/elevation velocity, range, range rate, Doppler and level. The Doppler is a satellite tracking technique used to determine the distance between the satellite and the receiver at the time of closet approach. The output on the screen of the AZ/EL table should be:

UTC	AZ Deg	EL deg	AZ-vel deg/sec	EL-vel deg/sec	Range km	Range Rate km/sec	Doppler kHz	Level dBm
YY-DOY-hh:mm:ss	XXX.XX	XXX.XX	XXX.XX	XXX.XX	XXXXX	XXX	XXX.XXX	XX

Figure 2: Tracking Data

Link Equation

The numerator of this equation is the signal strength we expect to read on the spectrum analyzer:

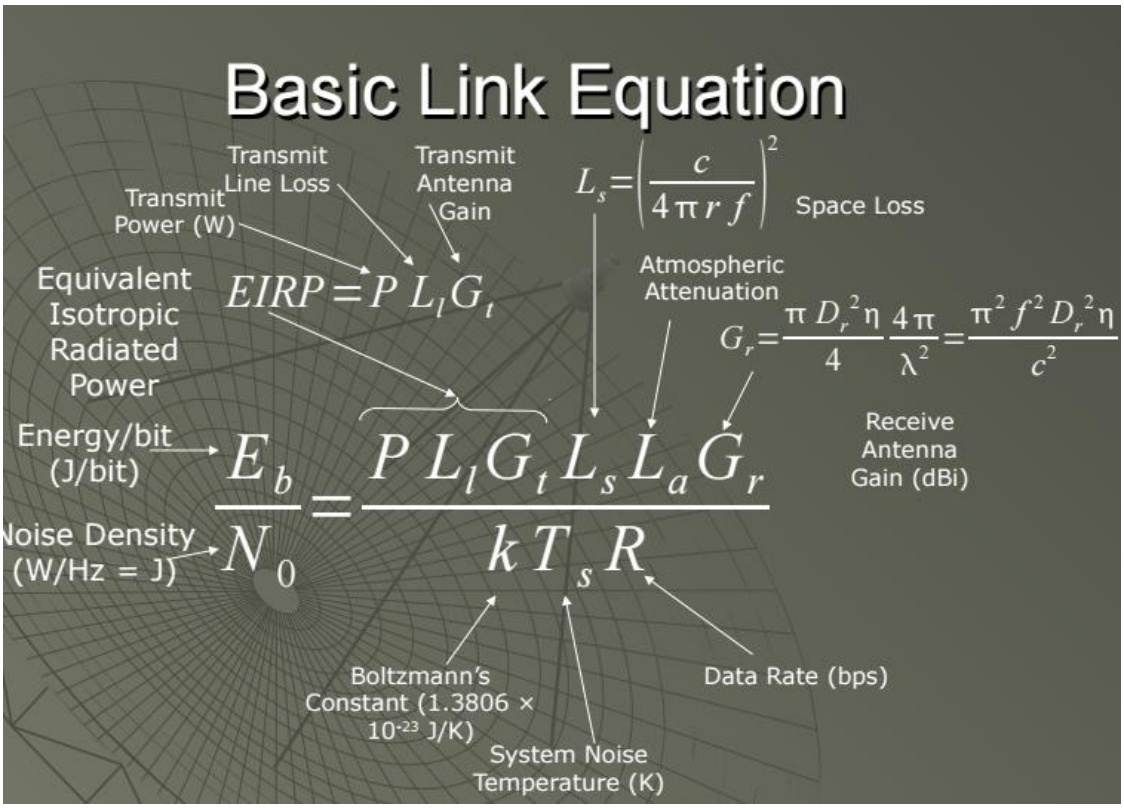

Figure 3: Basic Link Equation (Source Chesser CSDC PDR)

The RF characteristics of the GPS signal transmission and the station receiving equipment are required to determine the signal strength with the spectrum analyzer. The signal strength is what we need to calculate to check that we are within the 3dB bandwidth of the GPS satellite.

Frequency band center, F_cnt	MHz	1227
Antenna efficiency, AE		0.70
Antenna diameter, D	m	46
Bandwidth max, B	MHz	2
RCV gain, RG	dB	56
RCV noise temperature, RNT	deg K	200

Example function that still needs to be modified for practical use:

```
/*
 * Function to print the AOS/LOS link strength
 *Frequency band = 1227
 *Frequency Antenna efficiency = 0.70
 *Frequency Antenna diameter = 46
 *Frequency Bandwidth = 2
 *Frequency RCV Gain = 56
 *Frequency RCV noise temp = 200
 *
 * */
double linkstrength(double range_z){
```

```
// Constants
	double fre = 1227.0;
	double eff = 0.7;
	double diam = 46;//m
	double BW = 2;
	double RCV_gain = 56;
	double RCV_noise = 200;
	double light = 3*10^8; // m/s
// Calculations
	double EIRP = 8.3988; //3dBW
	double La = 0.1; //dBW
	double Ls = 10*log((light/(4*PI*fre*range_z*1000))^2);// dB
double Gr = 10*log((((PI*PI) * (fre*fre) * (diam*diam) *eff)/light*light));// dB
	double linksstren = EIRP - La + Gr + Ls +30;
	return linksstren;
}
```

Testing and Verification methods for Main

The verification will be with various online websites:

Julian Date Calculator

- http://aa.usno.navy.mil/data/docs/JulianDate.php

Satellite Look Angle Calculator

- http://www.groundcontrol.com/Satellite_Look_Angle_Calculator.htm

Latitude, Longitude, Height to/from ECEF (X,Y,Z)

- http://www.oc.nps.edu/oc2902w/coord/llhxyz.htm

Online Satellite and Flare Tracking

- http://www.satflare.com/track.asp#TOP

Satview.org – Taking Satellites

- http://www.satview.org/

Heavens-Above

- http://www.heavens-above.com/

Online Satellite Calculations

- http://www.satellite-calculations.com/

The most robust test to perform will be using the Systems Tool Kit (STK) by Analytical Graphics Incorporated. The TLE's can be imported and dynamic charts can be used to display the ECI/ECF/Topocentric coordinates of the satellite, as well as visually see if the satellite is in view of the ARO.

Here is glimpse of the testing scenario developed:

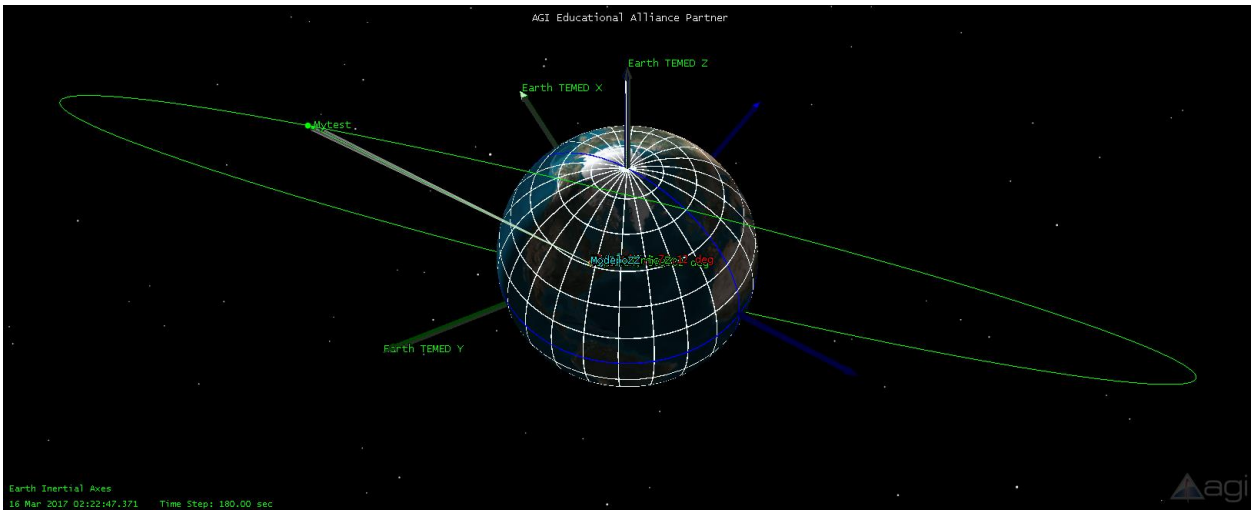

Figure 4: View of the ARO beam width within the scope of the satellite the user wants to track

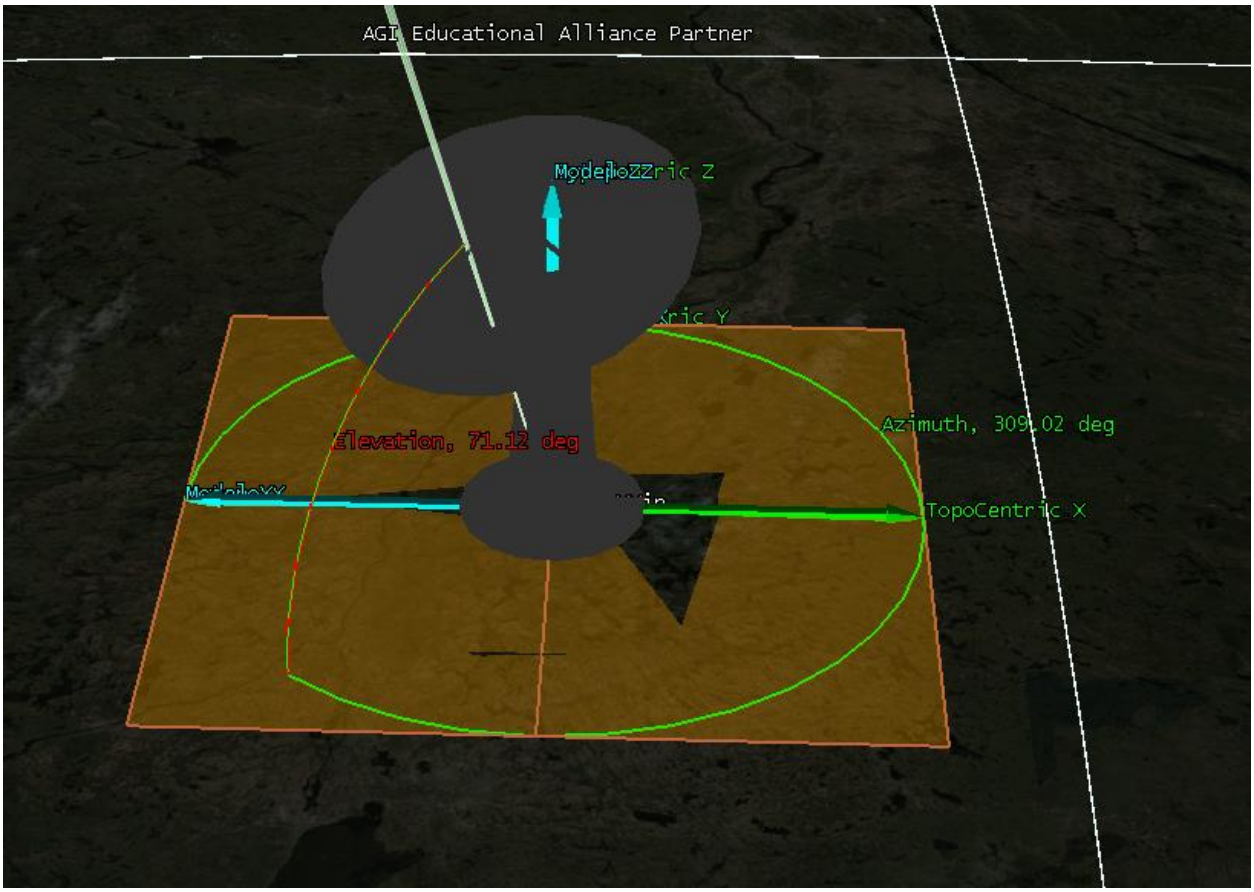

Figure 5: View of the ARO displaying the Azimuth and Elevation Angles

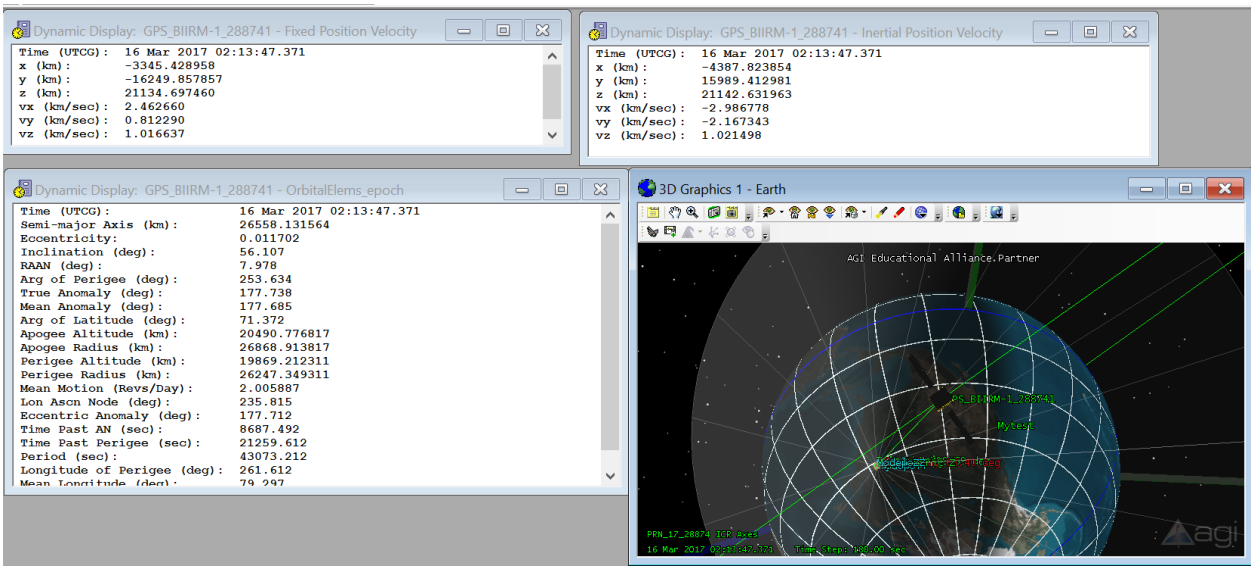

Figure 6: Dynamic Graphs of the Keplerian Elements at Epoch, ECI, ECF

Example Access Report to confirm time the satellite is in view

```
19 Mar 2017 17:17:07

 AGI Educational Alliance Partner
Facility-Algonquin-To-Satellite-GPS_2F-1_36585, Satellite-PRN_01_37753,
Satellite-PRN_02_28474, Satellite-PRN_03_40294, Satellite-PRN_05_35752,
Satellite-PRN_07_32711, Satellite-PRN_08_40730, Satellite-PRN_09_40105,
Satellite-PRN_11_25933, Satellite-PRN_12_29601, Satellite-PRN_13_24876,
Satellite-PRN_14_26605, Satellite-PRN_15_32260, Satellite-PRN_16_27663,
Satellite-PRN_17_28874, Satellite-PRN_18_26690, Satellite-PRN_19_28190,
Satellite-PRN_20_26360, Satellite-PRN_21_27704, Satellite-PRN_22_28129,
Satellite-PRN_23_28361, Satellite-PRN_24_38833, Satellite-PRN_25_36585,
Satellite-PRN_26_40534, Satellite-PRN_27_39166, Satellite-PRN_28_26407,
Satellite-PRN_29_32384, Satellite-PRN_30_39533, Satellite-PRN_31_29486:
Access Summary Report


Algonquin-To-GPS_2F-1_36585
---------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ------------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000
```

```
Algonquin-To-PRN_01_37753
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_02_28474
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_03_40294
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000
```

```
Algonquin-To-PRN_05_35752
-------------------------
No Access Found


Algonquin-To-PRN_07_32711
-------------------------
No Access Found


Algonquin-To-PRN_08_40730
-------------------------
No Access Found


Algonquin-To-PRN_09_40105
-------------------------
No Access Found


Algonquin-To-PRN_11_25933
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:00:10.061            10.061

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:00:10.061            10.061
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:00:10.061            10.061
Mean Duration
10.061
Total Duration
10.061


Algonquin-To-PRN_12_29601
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000
```

```
Algonquin-To-PRN_13_24876
-------------------------
No Access Found


Algonquin-To-PRN_14_26605
-------------------------
No Access Found


Algonquin-To-PRN_15_32260
-------------------------
No Access Found


Algonquin-To-PRN_16_27663
-------------------------
No Access Found


Algonquin-To-PRN_17_28874
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ------------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_18_26690
-------------------------
No Access Found


Algonquin-To-PRN_19_28190
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ------------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
```

```
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_20_26360
-------------------------
No Access Found


Algonquin-To-PRN_21_27704
-------------------------
No Access Found


Algonquin-To-PRN_22_28129
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    -----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_23_28361
-------------------------
No Access Found


Algonquin-To-PRN_24_38833
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    -----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
```

```
Total Duration
1800.000


Algonquin-To-PRN_25_36585
-------------------------
No Access Found


Algonquin-To-PRN_26_40534
-------------------------
No Access Found


Algonquin-To-PRN_27_39166
-------------------------
No Access Found


Algonquin-To-PRN_28_26407
-------------------------
                  Access       Start Time (UTCG)           Stop Time (UTCG)
Duration (sec)
                  ------    ----------------------    ---------------------
--    --------------
                       1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000

Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Max Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1800.000
Total Duration
1800.000


Algonquin-To-PRN_29_32384
-------------------------
No Access Found


Algonquin-To-PRN_30_39533
-------------------------
No Access Found


Algonquin-To-PRN_31_29486
-------------------------
No Access Found

Global Statistics
-----------------
Min Duration           1    1 May 2017 00:00:00.000    1 May 2017
00:00:10.061           10.061
```

```
Max Duration            1    1 May 2017 00:00:00.000    1 May 2017
00:30:00.000          1800.000
Mean Duration
1637.278
Total Duration
18010.061
```

Example Main Function

```
/*
 * P3_Keith_James.c
 *
 *  Created on: Mar 19, 2017
 *      Author: james
 */

#include <stdio.h>
#include <math.h>
#include "Propagate.h"
#include "Basic.h"
#include "FileIO.h"
#include "Vector.h"
#include "STKout.h"
#include "DateAndTimeCalculations.h"
#include "Matrix.h"
#include "Vector.h"
#include <time.h>
#include <stdlib.h>
#include <string.h>
#define CUBE_ROOT(X) (exp(log(X)/ 3.))
#define PI
3.14159265358979323846264338327950288419716939937510582097494459230781640628

int main(){

	Banner();
	printf("\nImporting station data...\n\n");
	Station *stn = (Station*) malloc(sizeof(Station));
	ReadStationFile(stn, '0');
	printf("Complete\n\n");

	printf("Enter the number next to the corresponding option:\n");
	printf("1   view station file data\n");
	printf("2   continue\n\n");
	int input1;
	printf("Entry: ");
	fflush(stdout);
	scanf("%d", &input1);
	if (input1 == 1){
	printf("\nStation File Contents:\n");
	printf("1  Name: %s\n", stn->name);
	printf("2  Station Latitude: %f\n", stn->stnlat);
	printf("3  Station Longitude: %f\n", stn->stnlong);
```

```
	printf("4  Station Altitude: %f\n", stn->stnalt);
	printf("5  UTC Offset: %f\n", stn->utc_offset);
	printf("6  Azimuth Elevation nlim: %d\n", stn->az_el_nlim);
	printf("7  Azimuth Elevation Limit Azimuth: %f\n", stn->az_el_lim.az);
	printf("8  Azimuth Elevation Limit Elevation Min: %f\n", stn-
>az_el_lim.elmin);
	printf("9  Azimuth Elevation Limit Elevation Max: %f\n", stn-
>az_el_lim.elmax);
	printf("10 Station Azimuth Speed Max: %f\n", stn->st_az_speed_max);
	printf("11 Station Elevation Speed Max: %f\n", stn->st_el_speed_max);
	}
	printf("\nImporting TLE file sats...\n\n");
	char *file = "TLE.txt";
	Satellite sats[32];
	ReadNoradTLE(sats, file);
	printf("Complete\n");
	int x;
	for(x = 0;x < 1;){
	printf("\n\nEnter the number next to the corresponding option:\n");
		printf("1   view TLE data\n");
		printf("2   continue\n\n");
		int input2;
		printf("Entry: ");
		fflush(stdout);
		scanf("%d", &input2);
		if(input2 == 1){
			printf("\nEnter the satellite number you would like to view: ");
			int num;
			fflush(stdout);
			scanf("%d", &num);

			printf("\nInformation for sat number %d\n", num);
					printf("\n    name is %s", sats[num].name);
					printf("    refepoch is %f\n", sats[num].refepoch);
					printf("    incl is %f\n", sats[num].incl);
					printf("    raan is %f\n", sats[num].raan);
					printf("    eccn is %f\n", sats[num].eccn);
					printf("    argper is %f\n", sats[num].argper);
					printf("    meanan is %f\n", sats[num].meanan);
					printf("    meanmo is %f\n", sats[num].meanmo);
					printf("    ndot is %f\n", sats[num].ndot);
					printf("    nddot6 is %f\n", sats[num].nddot6);
					printf("    bstar is %f\n", sats[num].bstar);
					printf("    orbitnum is %f\n", sats[num].orbitnum);
			}
		if(input2 == 2){x++;}
		}
	// Visibility
	printf("\nEnter a Julian Date start time for the visibility check (span is 30
minutes from start time): ");
	double input3;
	fflush(stdout);
	scanf("%lf", &input3);
	printf("\n\nThe following are in view for a 30 minute range starting from
%f:\n", input3);
```

```
	int j = 1;
	for (j = 1;j < 32;j++){
	int i;
	int view = 0;
		for (i = 0; i <30;i++){

	double mA;
	double mM;
	double t = input3 + i*0.000694;
	double satEpoch = sats[j].refepoch;
	double mA0 = sats[j].meanan;
	double nMM = sats[j].meanmo;
	double ndMM = sats[j].ndot;
	double n2dMM = sats[j].nddot6;
	mean_anomaly_motion(&mA, &mM, t, satEpoch, mA0, nMM, ndMM, n2dMM);
	double eccAnom = KeplerEqn(mA, sats[j].eccn);
	Vector *eciP;
	eciP = (Vector*)malloc(sizeof(Vector));
	Vector *eciV;
	eciV = (Vector*)malloc(sizeof(Vector));
	double sMA = CUBE_ROOT(398600.4418/(4*PI*PI*mM*mM));
	sat_ECI(eciP, eciV, sats[j].eccn, eccAnom, sMA, sats[j].raan,sats[j].argper,
sats[j].incl, mM);
	Vector *ecfP;
	ecfP = (Vector*)malloc(sizeof(Vector));
	Vector *ecfV;
	ecfV = (Vector*)malloc(sizeof(Vector));
	double thetat = THETAN(sats[j].refepoch);
	sat_ECF(ecfP, ecfV, thetat, eciP, eciV);
	Vector *stnP;
	stnP = (Vector*)malloc(sizeof(Vector));
	station_ECF(stnP, stn->stnlong, stn->stnlat, stn->stnalt);
	Vector *rtP;
	rtP = (Vector*)malloc(sizeof(Vector));
	Vector *rtV;
	rtV = (Vector*)malloc(sizeof(Vector));
	range_ECF2topo(rtP, rtV, *stnP, ecfP, ecfV, stn->stnlong, stn->stnlat);
	double az;
	double el;
	double azV;
	double elV;
	LookAngles *LA =(LookAngles*) malloc(sizeof(LookAngles));
	range_topo2look_angles(LA, az, el, azV, elV, rtP, rtV);

	if (LA->elevation <= stn->az_el_lim.elmax){
		if (LA->elevation >= stn->az_el_lim.elmin){
			view++;
		}
	}
	}
		if (view > 0){
			printf("%s", sats[j].name);
		}
	}
	return 0;
```

```
}
```

Example Main Output using May 1st 00:00:00 to 00:30:00

```
ENG4350 Team Info

Program Name

2017-03-19

Version 1

Welcome

Importing station data...

Complete

Enter the number next to the corresponding option:

1   view station file data

2   continue

Entry: 1

Station File Contents:

1  Name:  ARO

2  Station Latitude: 45.955503

3  Station Longitude: 281.926960

4  Station Altitude: 260.420000

5  UTC Offset: -4.000000

6  Azimuth Elevation nlim: 1

7  Azimuth Elevation Limit Azimuth: 0.000000

8  Azimuth Elevation Limit Elevation Min: 9.000000

9  Azimuth Elevation Limit Elevation Max: 89.000000

10 Station Azimuth Speed Max: 3.000000

11 Station Elevation Speed Max: 3.000000

Importing TLE file sats...

Complete

Enter the number next to the corresponding option:

1   view TLE data
```

2 continue

Entry: 1

Enter the satellite number you would like to view: 3

Information for sat number 3

name is GPS BIIR-5 (PRN 28)

refepoch is 17077.774306

incl is 56.656900

raan is 348.673600

eccn is 0.020101

argper is 269.945100

meanan is 6.180200

meanmo is 2.005676

ndot is -0.000001

nddot6 is 0.000000

bstar is 0.000000

orbitnum is 0.000000

Enter the number next to the corresponding option:

1 view TLE data

2 continue

Entry: 1

Enter the satellite number you would like to view: 30

Information for sat number 30

name is GPS BIIF-12 (PRN 32)

refepoch is 17077.486759

incl is 54.908700

raan is 223.671000

eccn is 0.001009

argper is 214.784000

meanan is 145.117100

meanmo is 2.005597

ndot is 0.000000

nddot6 is 0.000000

bstar is 0.000000

orbitnum is 0.000000

Enter the number next to the corresponding option:

1 view TLE data

2 continue

Entry: 2

Enter a Julian Date start time for the visibility check (span is 30 minutes from start time): **2457874.5**

The following are in view for a 30 minute range starting from **2457874.500000**:

GPS BIIR-5 (PRN 28)

GPS BIIR-6 (PRN 14)

GPS BIIR-9 (PRN 21)

GPS BIIR-11 (PRN 19)

GPS BIIR-12 (PRN 23)

GPS BIIR-13 (PRN 02)

GPS BIIRM-3 (PRN 12)

GPS BIIRM-4 (PRN 15)

GPS BIIRM-5 (PRN 29)

GPS BIIRM-6 (PRN 07)

GPS BIIF-2 (PRN 01)

GPS BIIF-3 (PRN 24)

GPS BIIF-6 (PRN 06)

GPS BIIF-12 (PRN 32)

Conclusion

It is clear that 7 satellites output from the main, confirm with the visibility STK Access Report, with only 4 misidentified. Future changes will be made to make sure it is as robust as possible as well making sure the names are exactly the same for easier cross-referencing.