Project ReSeDA

Remote Sensing Data Assimilation

Contribution of Cranfield University, College of Aeronautics

March 1997 - February 2000

Report Index

1. Introduction

This report describes the contribution of Cranfield University to the EU-funded ReSeDA project.

ReSeDA Analysis Steps

The initial raw data collection programs were written by Alan Stevenson. All the later stages have been developed by Stephen Hobbs.

The data analysis programs are kept in the directory c:\myprogs\visualc\reseda or \dgps, with the data currently in directory e:\reseda (various subdirectories).

2. Raw Data Collection

Carried out by Alan Stevenson.

32 frames / s transmitted from aircraft to ground, with interleaved sensor and GPS data frames, i.e. 16 frames per second each for sensors and GPS. The information content is 16 samples per second per sensor with the samples evenly spaced in time, and 1 GPS position, time and velocity measurement per second.

Data are supplied as comma-separated-variable files (ASCII "csv" format, suitable for importing into Excel). The initial versions required use of Alan's program Splitall.exe to create the necessary files. Later versions are supplied already in .csv format.

The channels used for the aircraft payload data are:

Channel	**Sensor**	**Parameter**
ADC0	Vaisala barometer	pressure
ADC1	Vaisala humitter	humidity
ADC2	Vaisala humitter	temperature ("T_{slow}")
ADC3	PRT (Pt wire)	temperature ("T_{fast}")
ADC4	radiometer channel 1	
ADC5	radiometer channel 2	
ADC6	radiometer channel 3	
ADC7	radiometer channel 4	
ADC15	PRT (Pt wire)	temperature ("T_{fast}") (coarser resolution than ADC3 by factor of 2, hence larger range)

Table: Aircraft payload data channels and corresponding measurand.

Calibrations for these parameters are discussed below (section 5.2).

3. Data Pre-Processing

The principal steps carried out as data pre-processing are:

1. Read data from the .csv files into a proprietary binary format.

2. Carry out some initial data quality checks (based mainly on checking GPS time values and that counters increment as intended).

A binary file format is used because (1) it is more compact than ASCII and allows the whole experiment dataset to be contained in one file of about 1 Mbyte, and (2) data can be read and accessed more easily from a binary file. The difficulty is that binary files are less easy to interpret without specially written programs.

The main program used is res_expt.c (ReSeDA experiment). It contains several options, and uses a common binary file format for sensor and GPS data. This is wasteful of file capacity since initially a binary file only contains either GPS or sensor data, but not both (the intention is that eventually both types of data will be combined), but the commonality allows the same general file reading programs to work with all types of data.

The typical procedure in using res_expt is:

1. Use option 3 to count the number of lines in the ASCII input file. The count total includes the first line which is usually a set of column labels (sensor data) or a line of bad data (GPS data). The number of data lines is taken to be one less than the count given by res_expt.

2. Use option 2 (sensor data) or 5 (GPS data) to create a binary file of data from the csv ASCII input file. The user specifies the file to be read (and its directory), the output file name, and the number of records to read from the source file. The program creates the binary file and gives a count of the errors found (these are errors found by comparing two versions of the GPS time and checking that the frame count value increments by 1 between frames). A valid GPS time is assumed to be any value > 10000 since the usual error value starts at 0 and increments by 1 each second; the maximum experiment length is about 1200 s, so if no GPS frame is ever correctly found the value of GPS time should never be much above 1200 s.

Option 6 of res_expt allows a binary file (usually the GPS data) to be searched to identify data segments within a user-specified region defined by limits in x and y position. The search results are written to a text file.

The other options of res_expt are:

1. Create a file of synthesised data to test the display routines. This option has not been updated and may not correctly implement the latest features used by the real data formats.

4. Display values stored in a binary file. Lists the contents of the binary file header and the first 5 data records.

7. Convert GPS time to conventional format. This option (its position on the menu changes as new functions are added, remaining the last available) tests the function to convert the long integer used to store GPS time into a more conventional format. The date and time decoded are formatted as two long integers and displayed.

The header information stored by res_expt in the binary file's first record is:

Variable name	Remark	# bytes
	number of data records in the file	
GPStime0	Original format GPS time of first valid record. Long integer.	4

date	long integer,	4
time0	long integer	4
program_ver	float indicating the version of res_expt used to write the file.	4

The header and data record structures are declared as (res_expt v 1.30):

```
typedef struct  {  long num_rec;
                   int hr0, min0;
                   float sec0;
                   long GPStime0, date, time0, recs_skipped, padding[4];
                   float ver;
                 } head_rec;                      /* 48 bytes */
typedef struct  {       float GPSt, GPSx, GPSy, GPSz, GPSu, GPSv, GPSw;
                   short DN_press, DN_rh, DN_T[3], DN_I[4], flag;
                 } data_rec;                      /* 48 bytes */
```

4. Initial Data Displays

Programs for data display have been written using Matlab. The initial versions were developed on synthesised data generated using the progam res_expt.

4.1. Standard Quick-Look

Two versions of the quick-look programs have been developed: quick.m displays sensor data values (in particular the temperature and humidity values) and quickpos.m displays the GPS data (position and velocity). Both include an error flag display which is an indication of data quality (however, data quality problems usually show up in the main parameter displays as well).

For both quick and quickpos, the user chooses the (binary) file to display and the start time (in minutes). The start time is converted to seconds and then record number offset assuming no transmission errors - there is sometimes a discrepancy between the chosen and displayed offset due to counter / timing errors in the data.

The two programs are saved as (ASCII) m-files. The files used by each program are:

quick (sensor data quick-look)		quickpos (GPS data quick-look)	
quick.m	main calling code; defines constants	quickpos.m	main calling code; defines constants
synread2.m	reads GPS and sensor data from the binary file format (includes file header data).	synread2.m	as for quick.m
dataprep.m	calculated derived variables for enhanced display purposes	dataprep.m	as for quick.m
synplot1.m	plots sensor summary plots (1 page) and fluctuations of temp (Humitter and PRT values) and humidity (second page).	plotpos.m	plots summary of aircraft locus (1 page) and of other GPS data (second page).

Problems may be found with (1) the barometric height displayed if the assumed reference pressure, temperature are significantly wrong, or (2) the GPS height or horizontal position (due to jumps in the apparent GPS position). In these cases, the relevant plotting code (synplot1.m or plotpos.m) may be edited to alter some of the axis scaling values. Modified versions (e.g. plotpos1.m) of these files are usually available for minor customisation of plots to avoid corrupting the main plotting code.

4.2. Customised Previews

Several additional plotting programs have been developed to help study particular portions of an experiment. These programs depend on the corresponding quick-look generation program (i.e. quick or quickpos) having been run previously to load all the required data values.

The programs developed are:

plotdat2.m Adds markers to the usual summary plot of sensor data.

plotseg1.m Plots expanded segments of sensor data.

plotseg2.m Plots expanded segments of aircraft track.

Both these m-files call seginit.m to define constants specifying the flight segments

to plot.

Both programs use the following information in addition to that provided by the standard quick-look programs:

tlim An array of the times (s, relative to the start of the sensor data recording) for the start and end of each data segment, e.g. tlim=[259 278 325 340 … 579 597] for the cross-boundary transects of Flight 10.

ymark An array the same size as tlim but filled with 0's. Used to write marks at t=tlim(n), y=0 to indicate the start and end of data segments visually.

nseg a constant which gives the number of data segments (the size of tlim, ymark should be 2*nseg).

del_t0 The time in seconds which the sensor recording starts before the GPS recording according to the binary files (e.g. for flight 10, using files f10.dat, f10pos.dat gives del_t0 = 13).

4.3. Time Synchronisation

Several different time / counter systems are involved in the experiments. The principal ones are:

GPS time	Adopted as the common reference system for the experiment.
Video camera clock	Approximately BST (within a few minutes).
Time relative to start of sensor data recording	The sensor data time is in s relative to the first valid record found in the sensor data file (e.g. Flit10.csv).
Time relative to start of GPS data recording	GPS data use times relative to the first valid record found in the GPS data file (e.g. Flit10.pos).

Both the relative times above also have the actual GPS time of the first record stored in the binary file header (and decoded by res_expt into conventional date and time format).

In addition to these clock times are the video counter clocks used to indicate position along the (S)VHS tape.

The different times can by synchronised using events detectable in both the recordings to be synchronised, e.g. take-offs and landings.

5. Specialised Data Processing

Several data processing steps are required to provide data which can be analysed to study the main scientific objectives of the experiments.

5.1. Test Site Surface Elevation Models

Since GPS measures aircraft position relative to some reference datum (e.g. WGS84) but the significant vertical position is the height of the aircraft above the local surface, a model of the local surface is required.

The surface model is obtained by (1) reading height values from maps (IGN 1:25000) of the local area - contours and spot heights, and then (2) using these heights to generate a model of the local surface elevation. The height readings from the map are irregularly spaced, so an important step to provide a useful surface height function is to generate a regular look-up table (LUT) of surface heights (and local slopes in the approach adopted here).

5.1.1. Modelling the local surface elevation

The algorithm used is implemented in program res_expt (v 1.40 and later). It is based on ideas given in Sabin, M.A., Contouring - a review of methods for scattered data, in Brodlie, K.W., (ed.), Mathematical Methods in Computer Graphics and Design, Academic Press, 1980.

The first step is to generate the look-up table (stored as a binary file). This is done by fitting a weighted least squares plane at each grid point of the LUT, and storing the parameters of that plane in the LUT. The algorithm minimises the sum G (defined below) by varying the parameters z_0, a and b for each grid point ($x_{0,i}$ and $y_{0,i}$ are the coordinates of the local grid point for which the plane's parameters are being calculated).

$$G = \sum_{i=1}^{n} w_i \left(z_i - \hat{z}_i \right)^2$$
$$\hat{z}_i = z_{0,i} + ax'_i + by'_i$$
$$x'_i = \left(x_i - x_{0,i} \right)$$
$$y'_i = \left(y_i - y_{o,i} \right)$$

The least square solution for these parameters is given by;

$$\begin{pmatrix} z_0 \\ a \\ b \end{pmatrix} = \begin{pmatrix} \Sigma w_i & \Sigma w_i x'_i & \Sigma w_i y'_i \\ \Sigma w_i x'_i & \Sigma w_i x'^2_i & \Sigma w_i x'_i y'_i \\ \Sigma w_i y' & \Sigma w_i x'_i y'_i & \Sigma w_i y'^2_i \end{pmatrix}^{-1} \begin{pmatrix} \Sigma w_i z_i \\ \Sigma w_i x'_i z_i \\ \Sigma w_i y'_i z_i \end{pmatrix}$$

The w_i are weights assigned to each data point controlling their contribution to the sums.

An estimate of the height uncertainty is obtained by taking G to be the sum of the square of the residuals, and then dividing this "variance" by the number of degrees of freedom and taking the square root of the result. The number of degrees of freedom is taken as the sum of the normalised weights less the number of parameters fitted, i.e. 3 (but the sum of the weights is sometimes less than 4, in which case its value is set to 4 to allow the following height uncertainty estimate to be used).

$$\delta z = \sqrt{\frac{G}{\Sigma w_i - 3}}$$

This height uncertainty is stored in the LUT.

5.1.2. Definition of weights

Several alternative methods of defining the weights w_i are available. A specification often involves two steps: (1) choosing a subset of neighbouring data points, (2) applying some weighting function to this subset which generally peaks at the point of interest and decays to a small value at the furthest point.

5.1.2.1.Choice of subset

The subset of n_total points closest to the grid point for which the plane is being fitted is chosen, ensuring that at least n_sector points are in each quadrant (to ensure the algorithm interpolates rather than extrapolating if the point is just to one side of a dense cluster of points). The weighting function is set to 0 for all points not in this subset.

5.1.2.2.Radial weighting function

The following function gives a surface which is continuous in height and first derivative (if appropriately applied - the method used here probably does not ensure continuity of height). If rmax is the range to the furthest point of the subset, then the weight is initially defined as

$$w_i = \left(\frac{r_{\max} - r}{r_{\max}} \right)^2$$

This in general gives weights which are all less than 1. These initial weights are then scaled such that the largest weight is exactly equal to 1.

Alternative functions can be used which will in general lead to other properties of the fitted surface. Non-radial functions could be useful if the surface has second derivatives which are strongly non-isotropic.

The integral of w(r) over the region divided by the region's area gives the proportion of points contributing to the sums. For this function, if n points are contained within $r < r_{max}$, then only n/6 points effectively contribute to the sums. Thus if the sum of weights is required to be at least 4 (to give a valid definition of the height uncertainty) then a total of approximately 24 points should be used. (The method of normalisation of the weights and the fact that one point is by definition at $r = r_{max}$, mean that this analysis is only approximately correct.)

It is not clear what an appropriate estimate of the number of degrees of freedom is in the case that the statistical sums are weighted as is done here, although this approach seems "reasonable".

5.1.3. Evaluating the model

The model is evaluated by finding the grid point closest to the position of interest and then using the slope of the local fitted plane and the displacement from the grid point to calculate a correction to the height at the grid point. This algorithm means that only one entry in the LUT is required. Note that this method does not ensure continuity of height or slope (although all higher derivatives are zero) at the midpoint between neighbouring grid points, but the density of grid points should be chosen to ensure that any height discontinuities are less than the inherent height uncertainty.

5.2. Composite Height Record

Two measurements of aircraft height are available: barometer and GPS. The barometer is expected to be more accurate (approx. 2 m vertically) but due to a poor electrical connection the barometer reading is very noisy at times (e.g. when the engine was operating at high power). For the periods when the barometer is not reliable, GPS height data will have to be used. GPS is less accurate vertically (approx. 5 m when obtained differentially), and may be corrupted by slow drifts of a few metres over the length of a flight.

If the drift of the GPS data is modelled as follows, then the same solution method outlined above can be applied, identifying the variable x with t, and y with t^2.

$$h = h_0 + at' + bt'^2$$
$$t' = t - t_0$$

It is probably appropriate to apply a windowing function of some sort with tapers to 0 at times sufficiently far from t_0, effectively defining a weighting term as used above.

6. Program Reseda

The program reseda (written in C) is used for data preprocessing. It performs several tasks, including:

1. converts the original data into a form suitable for the quick-look displays discussed above.
2. creates a regular grid of site elevations from a table of scattered elevation data across the test site
3. converts the raw aircraft GPS data (not completely regular) into a regular binary table suitable for later analysis stages, and allows errors, gaps, and glitches in the data to be fixed.
4. converts the raw payload sensor data (not completely regular) into a regular binary table suitable for later analysis stages, and allows errors, gaps, and glitches in the data to be fixed.
5. defines a standard time reference (based on GPS time) and reference coordinate for the experiment.
6. the aircraft position data and site elevation models can be combined to give aircraft position and height above the ground as a function of time through the experiment.
7. the aircraft position, site elevation, and payload data can be combined to give a full set of experiment parameters for the experiment.

It is anticipated that later stages of the analysis will use programs such as MATLAB, IDL, etc. for the scientific analysis based on the synchronised, regular, coordinated dataset produced by reseda.

6.1. Program Version History

Program reseda has evolved significantly over the months. The initial version was written using the Microsoft C compiler v 5.1 (program versions 0.xx), but more recent developments have used Microsoft Visual C++ v 4.0 (program versions 1.6x onwards). The Visual C++ versions are only for 32-bit operating systems (e.g. Windows 95 and NT).

Version	Released	Remarks
0.61		As supplied to INRA.
1.65	9/10/98	Includes functions for reading in the differential GPS corrections provided by Jeff Stevens and combining them with the aircraft C/A code data to give differentially-corrected GPS data.
1.70	27/5/99	Able to read in the DGPS differential corrections. Change to GPS look-up table format to include location of site origin and reference antenna.

Table: Program version history.

The MS-DOS versions of the program contain the basic functions (developed early on), but the policy now is only to develop the 32-bit version. The programs require some attention to transfer between the operating systems, mainly because data are stored in 16-bit chunks under MS-DOS, and 32-bit chunks under Windows 95 etc.

6.2. Initial Program Release Version

As an initial release version, program reseda (as res_xdos v 0.61 - for MS-DOS) with a set of standard input files for Flight 10 and an example output file is supplied to INRA.

File type	ASCII /	size (kb)	name	remarks

	binary			
program	bin	104	reseda.exe	version 1.62 is current (12/1/98). Runs under Windows 95 or NT.
	bin	112	res_xdos.exe	v 0.62 current (12/1/98); for MS-DOS
input	ASCII	3	lacrau0.prn	spot heights for La Crau test site
	bin	8	lacrau0.d25	site elevation model for La Crau test site
	ASCII	109	f10pos.csv	raw aircraft GPS data (inc. glitches, gaps, etc.)
	bin	29	f10pos.lut	aircraft position data table (does NOT include differential GPS corrections)
	ASCII	1298	f10sen.csv	raw sensor data (inc. glitches, gaps, etc.)
	bin	323	f10sen.lut	payload sensor data table
output	ASCII	138	f10test.txt	output file (data every 1 s through the experiment), can be read by Excel etc., produced using option 4.7
	ASCII	9	lacrau_z.m	MATLAB compatible version of elevation model for La Crau test site

Table: Sample files provided for initial program testing.

6.3. Running the Program

reseda is organised using a menu structure as for conventional MS-DOS command line programs. The user types one of the indicated numbers to choose the desired option.

A minor problem exists (only for the 32-bit version, i.e. not MS-DOS) when entering file or directory names as strings. The user has to enter the required name (or press <return> directly if the default name is satisfactory), and then confirm the choice by pressing 'y' (upper or lower case, any other character cancels the selection). When the 'y' key is pressed, the character is also written to the start of the next file name asked for, and generally has to be deleted using the <backspace> key.

6.3.1. Program Help Information

Option 0 from any menu leads to some pages giving information about the operation of all functions in the program.

6.3.2. Compatibility - MS-DOS / Windows 95 / NT

MS-DOS and the 32-bit PC operating systems (Windows 95 / NT) require slightly different file sizes to store information in general. The struct definitions of the MS-DOS program (res_xdos v 0.61 and later) have been modified (except as specified below) to make them compatible with the way Windows 95 interprets them. This means that binary files produced by either program version are interchangeable.

There are some differences in the way the `scanf()` (and related functions) function is interpreted by the two different operating system versions of the compiler, such that the input text files may require more comments for the Windows 95 / NT version. E.g. file `lacrau0.prn` has a comment on the last data line to enable the leading character `'E'` on the following line to be read correctly.

This policy has not yet (24/11/97) been applied to the synthesised aircraft data files because (1) these options are not significant in current program use, and (2) any changes would probably require modifications to the MATLAB code used to read the files for the quick-look displays.

6.3.3. An example session

The following commands show how to carry out some of the basic program operations.

Not yet implemented.

6.4. Binary Look-Up Table (LUT) Files

The program creates several different binary files containing look-up tables which are used to store datasets for easier access than the original ASCII formats used for most data. The binary files are usually able to be quality checked, and to have any erroneous data corrected using basic error detection and editing functions (implemented in reseda).

6.4.1. File formats

Not yet implemented.

6.4.2. Data quality flags

The binary files of GPS and sensor data contain a parameter `flag` used to indicate data quality (flag is of type character and takes values 0 to 255). `flag` may be an array of two characters: the intended use is `flag[0]` to indicate data quality and `flag[1]` to act as a marker for data segments of particular interest.

`flag[0]` value	**When loaded**	**Remarks**
`0`	when valid data are loaded	good data
`1`	when valid data are loaded but the sequence counter is anomalous	only 16 (= sample rate) records are expected per second; the flag is set to 1 if more than 16 records are written for the same nominal GPS time (in whole seconds).
63	when overwriting data	these values were entered manually
127	when copying data	this record is a copy of another
191 (-65)	when interpolating data	this record is interpolated from nearby neighbours
255 (-1)	at initialisation	no data record found for this time

Table: Values of data quality flag in binary tables of GPS and sensor data. Depending on how the character value is printed, values > 127 may appear negative (value in brackets).

One means of detecting a potential error in a data file is for the program to look for values of flag not equal to zero.

`flag[2]` value	**When loaded**	**Remarks**
`0`	when valid data are loaded	no differential data
`1`	when valid data are loaded but the sequence counter is anomalous	differential data loaded
2	when overwriting data	values copied to start from 1st good value
3	when copying data	values copied to end from last good value

4	when interpolating data	this record is interpolated from nearby neighbours

Table: Data quality flags for differentially-corrected GPS (horizontal) positions.

One means of detecting a potential error in a data file is for the program to look for values of flag not equal to zero.

6.5. MATLAB test plots

reseda produces output files of the site topography (and height uncertainty) ready for direct input to MATLAB for checking (options 2.3 and 2.4). If the output file name is lacrau_z.m, then the MATLAB commands required to view the surface (assuming the file lacrau_z.m is in the current directory) are:

```
lacrau_z
view(45,10)
```

(The view() command alters the viewpoint from the MATLAB default, and can be reset by the user to give the most useful view of the surface.)

Similar commands allow the height uncertainty surface to be viewed for information / checking.

6.6. Composite Height Data

The barometer data (accurate but noisy) is combined with the GPS height (smoother but liable to drift) to create a composite height with the smoothness of the GPS data but the long term accuracy of the (good portions of the) barometer data. The accuracy of the composite height trace appears to be a few metres at worst.

6.7. Differential GPS Data

A pseudo-different GPS correction has been calculated by forcing the reference receiver to use only the satellites viewed by the aircraft. The results of this are not yet satisfactory.

Program DGPS (see below) has been developed to provide independent calculation of GPS positions and thus to make the corrections directly rather than depending on proprietary software (e.g. the Trimble software appears to make undocumented assumptions in its processing of the GPS data which compromise further analysis).

6.8. Composite Output File

Reseda.exe option 4.9 creates a combined listing of the GPS, map, sensor and GPS height drift model. This is the main file type that most science users of the data will require since it contains the raw data synchronised and error-checked.

A full listing of the whole flight is available (`fnnall0.txt`) but this is a large file with much unimportant information. It may be more convenient to create smaller files focussed just on portions of the flight of particular interest. To do this program reseda.exe should be run but the appropriate record range and step size can be chosen by the user (e.g. find start and stop record numbers for portions of low-level transects, and output data only every 4 records (every 0.25 s) during the transect).

The data values listed in the composite output file are:

Variable	**Comments**
`rec`	record number used to index data in the file
`t/s`	time in seconds from the start of the experiment. The GPS time (in seconds from 00:00:00 6 Jan 1980) gives the absolute time of the start of the experiment.

x, y, z / m	This is the (differential) GPS position of the aircraft relative to the GPS reference receiver plus the position of the reference relative to the local map origin
u, v, w / ms^{-1}	These are the velocity components copied from the origin aircraft GPS C/A code receiver output. Note that a different GPS algorithm is used in the receiver than is used by program DGPS.
el / m	This is the site surface elevation at position x, y derived from the site elevation model created by reseda.exe.
z-drift	z-drift (dz) is the mean difference between the barometer height reading (z_{baro}) and the GPS height reading corrected for the reference receiver height (z). $$dz = z_{baro} - z = z_{baro} - \left(z_{GPS} + offset\right)$$ If the barometer reading is correct, then the height of the aircraft (h) above the surface is given in terms of the drift model (dz), reported vertical position (z), and the site elevation (el) by: $$h = z_{baro} - el = dz + z - el$$
stdev	This is an estimate of the uncertainty of the stated z-drift in metres.
press, rh, T0, T1, T2, I0, I1, I2, I3	These are the raw digital number (DN) values for each of the payload sensor channels. The chapter on Sensor Calibration describes how to convert the DN values into physical units for each sensor.
flag0, 1	These are the two flag values used to indicate the quality / history of the data record. The flag values are described in the section above on Binary Look-Up Table Files.

Table: Explanation of the values tabulated in the combined results listing (reseda.exe option 4.9).

A header to the data file gives information about the files from which the composite file has been created, and parameter values used to create the file.

6.9. Example Data Files and Processing

A set of example data files has been compiled for parallel work at INRA Avignon. The files supplied include the source data files (ASCII) for the aircraft payload data and the aircraft GPS position and velocity data, elevation and map data files for each of the test sites, and example files for flight 10. The Windows 95/NT version of the data processing program reseda.exe is also provided.

File name	**Input for reseda functions (option #)**	**Contents**
alp_elev.lut	2.2 - 2.4, 3.6, 4.7, 4.9	Site surface elevation model for main ReSeDA test sites ("Alpilles")
ermiteht.lut	2.2 - 2.4, 3.6, 4.7, 4.9	Site surface elevation model for Mas l'Hermite test site
lacrauht.lut	2.2 - 2.4, 3.6, 4.7, 4.9	Site surface elevation model for La Crau test site
f10pos1.lut	3.4 - 3.8, 4.7, 4.9	Look-up table of aircraft position data (C/A GPS data); edited to remove errors.
f10sen4.lut	3.8, 4.5 - 4.9	Look-up table of aircraft payload data; edited to remove errors.
f10drift.dat	4.7, 4.9	Model of GPS vertical drift during the flight for

		correction of C/A GPS heights
f10drift.txt		ASCII version of f10drift.dat
CU_core_data_1of2.zip		Raw data from aircraft payload sensors and aircraft GPS for flights 2-6
CU_core_data_2of2.zip		Raw data from aircraft payload sensors and aircraft GPS for flights 7-10
reseda.exe		Data processing program (Windows 95/NT)
mapdata\Alp_ht1.txt	2.1	Alpilles site surface elevation data ready for use by reseda (Easting / m, Northing / m, elevation / m)
gridfit.m		Matlab m-file to calculate (2nd order) transformation from paper mm units to map grid reference units
gridfunc.m		Matlab m-file which uses coefficients calculated by gridfit.m to convert paper mm units to grid reference units.
gridelev.m		Matlab m-file which converts a file of x,y,height values from mm to grid units (for input to reseda option 2.1)
gridconv.m		Matlab m-file which converts a file of x,y,label map data from mm to grid units
mapdata\Alpmap.m		Matlab m-file to find quadratic coefficient to convert mm readings to grid values (now use gridfit.m)
mapdata\Altrax1.txt		Alpilles site map data (Easting / m, Northing / m, label); field boundaries, tracks, etc.
mapdata\Ermiteht.txt		Mas l'Hermite site spot heights and contours in mm units
mapdata\Ermitexy.txt		Grid points and their mm coords for Mas l'Hermite site
mapdata\Ermitht1.txt	2.1	Mas l'Hermite site surface elevation data ready for use by reseda (Easting / m, Northing / m, elevation / m)
mapdata\Ermitmp1.txt		Mas l'Ermite site map data (Easting / m, Northing / m, label); field boundaries, tracks, etc.
mapdata\Lacrauht.txt	2.1	La Crau site surface elevation data ready for use by reseda (Easting / m, Northing / m, elevation / m)
mapdata\lacraump.txt		La Crau map data (Easting / m, Northing / m, label)

Table: Sample data files supplied and their function.

For each flight, the following files are required:

Data type	File name	Comment
map	**alp_ht0.lut** **ermitht0.lut** **lacrauh0.lut**	Binary look-up table of site surface elevation (relative to datum of chosen projection, e.g. Lambert Zone II étendu). One of these files is used, depending on the site location.
	alp_map.txt ermitmp0.txt lacraum0.txt	ASCII file of map features in Lambert Zone II étendu coordinates. These allow the aircraft position to be related to other significant features of the site.
payload sensor	flitn.csv	original ASCII file of payload sensor data from Alan Stevenson, as a comma separated variable file (can be read by Excel).
	fnnsenm.csv	minor changes to above file for it to be read by reseda.exe (add line to give experiment start and stop times, and flag for end of file). No gaps or glitches are corrected, but incomplete data lines may be repaired (or deleted).

	fnnsenm.lut	regular binary look-up table of sensor data for the whole experiment period with gaps and glitches fixed
position (GPS)	flitnn.pos	ASCII .csv file from Alan Stevenson of C/A GPS aircraft position and velocity using the receiver's built-in GPS solution algorithm
	fnnposm.csv	edited version of flitnn.pos to allow reading by reseda.exe. The changes are to add information about the reference position, the experiment times, and a flag for end of file. No gaps or glitches are corrected, but broken records may be repaired or deleted.
	fnnposm.lut	look-up table of aircraft C/A position and velocity for the whole flight. Gaps and glitches can be corrected to give a complete regular dataset.
	ephemnn.dat	binary file of GPS satellite ephemeris for day nn in June 1997. These data allow the exact GPS satellite position to be calculated, which is needed to solve for the user receiver position.
	fnnrefm.rxo	edited set of RINEX GPS observations (pseudo-range to each satellite in view) for the reference receiver (flight nn, version m of the file). Unnecessary observation records are deleted.
	flitnn.gps	binary file from Trimble receiver (aircraft) giving code phases
	fnngps.txt	ASCII version of Trimble receiver data with code phases
	fnngpsm.rxo	ASCII set of pseudo-ranges in RINEX observation format derived from the Trimble code phase values
	fnndgpsm.txt	ASCII file of differential GPS positions based on the two .rxo files (which must be edited so that both contain only exactly corresponding observation records)
	fnndgpsm.lut	binary look-up table of differential GPS positions (same binary format as the C/A position look-up table, so both file types can be edited, printed, and error-checked using reseda.exe)
	fnndgpsm.err	ASCII file created with fnndgpsm.lut which lists the errors found and fixed.
height correction	**fnndzm.dat**	binary format (ASCII version created as fnndzm.txt) of the height correction function which uses barometer height to correct the GPS derived height.
	fnndzm.txt	ASCII file created with fnndzm.dat which lists the errors found and fixed.
composite	fnnallm.txt	ASCII file containing combined payload, GPS, map and height correction data, all properly synchronised and error-checked, ready for further analysis. The file contains 1 record for every sample period (1/16 second) for the whole flight. Reseda.exe allows the user to create other composite data files for shorter periods, or to sample only every n records (e.g. 1 record every 16 gives 1 record per second).

*Table. List of the main files used in the data processing for an aircraft flight. The files in bold type are those needed to create the final composite file suitable for detailed scientific analysis. Most users should only need the final composite file, although in case of queries about data quality the full set of files may be required. Two digits (*nn*) are usually used to number the flight, and file version number is indicated by the suffix (*m*).*

Most users should need only the final composite data file (e.g. f01all0.txt).

6.10. Further Work (@28/5/99)

The following topics require further work. Most of the data processing is now complete, but a few topics remain. Little scientific analysis has yet been performed.

Data processing:

- Sensor calibration (fast temperature sensor, radiometers)
- Estimate sensor footprints

Science:

- Plot temperature, humidity variations
- Estimation of heat and moisture fluxes

7. Program DGPS

Initial GPS data processing provided reformatted values of the GPS C/A position and velocity outputs of the aircraft GPS receiver (Trimble Lassen-SK8). To obtain full value from the science (aircraft payload) data it is important to have differentially corrected GPS position measurements. Attempts to calculate these by forcing the reference receiver to see the same satellites as the aircraft receiver was apparently using and then differencing the two position estimates were unsuccessful (it is assumed because of the data processing algorithm used by the Trimble receiver).

Program DGPS was written to process the GPS data independently (DGPS also carries a variety of map coordinate conversions).

The data files available are:

1. Reference receiver: RINEX format navigation and observation data for all flights. (Both are ASCII files.)

2. Aircraft receiver: Trimble proprietary format observation data for all flights (no navigation data, although these are mostly the same as the reference receiver values). (The source data file is binary; program gpslst.exe extracts relevant values to an ASCII file.)

(RINEX is a standard GPS data transfer format developed for receiver independent exchange of data.)

7.1. DGPS Objectives

Program DGPS is written to calculate differentially corrected GPS positions of the aircraft for the experiments of June 1997 for ReSeDA.

DGPS uses the RINEX data formats for navigation (satellite ephemeris) and observation (receiver pseudo-ranges) data. The main functions performed are:

- Propagates satellite orbits based on the GPS ephemeris data.

- Finds changes in satellite constellation used by the GPS receiver, and reports the satellites in use at any time.

- Solves for receiver position using 4 or more satellites (uses a non-linear least squares algorithm).

- Converts Trimble codephase values from the proprietary Trimble format (ASCII listing produced by gpslst.exe) to standard RINEX format pseudo-range (observation) file format.

- Miscellaneous time and map coordinate conversions.

The initial task of DGPS is to determine whether or not the data are able to be differentially corrected (initial work by Jeff Stevens seemed to indicate a problem if the Trimble position values are used). If it looks feasible, then DGPS will be used to calculate differentially-corrected positions.

7.2. DGPS Validation

Several specific validation tasks have been carried out: orbit propagation, position determination. Example data provided by Peter H. Dana have been used for independent validation of the algorithms.

The satellite ephemeris data from Peter Dana were copied to the Excel spreadsheet (**`DGPScheck.xls`**) used to check the orbit propagation algorithm (Table 2.3 of Kaplan, 1996) and copied into a RINEX format navigation data file (**`ephtest.rxn`**). The pseudo-ranges were converted to Trimble codephase units (1/16 of codephase chip ~ 18 m) and copied into a Trimble format observation file (**`danatest.txt`**).

The validation files are held in directory `e:\reseda\dgps\validation`.

7.2.1. Orbit propagation

The Excel spreadsheet values for satellite position agreed to within 2 m. At early stages of program development, the spreadsheet and DGPS were shown to agree to about the level of the double precision rounding error.

DGPS converted the RINEX navigation data (`ephtest.rxn`) to its own binary format for ephemeris data (`ephtest.dat`). The orbit propagation algorithm then calculated satellite positions for all 4 satellites given in the Dana test data, with agreement to 2 m or better (to cm level agreement for 2 of the 4 satellites).

A check of the DGPS orbit against valued derived using STK with tle data showed agreement only to the level of about 20 km. This discrepancy may be due to an orbit maneouvre between the time of generating the tle values and the time used to compare satellite positions. The discrepancy has not been resolved definitively yet (DGPS v 0.20).

7.2.2. Position determination

DGPS was used to convert the Trimble format observation data (`danatest.txt`) to RINEX format observations (`danatest.rxo`), and then these were used to calculate user position with the ephemeris values (`ephtest.dat`). The results show good agreement (position agreement to 0.6 m or better) IF the allowance in function `dmodel()` for signal transit time from the satellite to the user is disabled.

Notes:

1. The default for DGPS is to enable the allowance for signal transit time.

2. An approximately correct value for user position is required to convert the codephase to pseudo-range and also to initiate the solution algorithm. This is given in function `enter_first_guess()`, and needs to be edited if data from different regions are to be processed.

7.2.3. Coordinate conversions

Jean-Pierre Pirat (IGN) supplied a set of validation data for Lambert Zone II data, and some conversions have been read from other IGN 1:25000 maps (Série Bleue). The checks for the La Crau and IGN data seem fine (the other points have not yet been checked).

From J.-P. Pirat (IGN) (coordinate conversion check)

Lambert Zone II: X = 632 542.058 m, Y = 180 804.145 m
NTF: 51.8072313 gr N, 0.4721669 gr E Paris

Note that the coordinates on the map have the Northing preceeded by a 1 for Zone I, 2 for Zone 2 and 3 for Zone III.

Region	Coordinate system	Easting	Northing
Boulogne	LZI	545	1321
	LZII étendu	544.9	2621.5
	NTF	1° 33.6' E	50° 35.2' N
Quiberon	LZII étendu	185	2289
	NTF	3° 10.3' W	47° 28.25' N
Annecy	LZII étendu	888	2089
	NTF	6° 2.0' E	45° 44.4'N
La Crau	LZII étendu	790	1845
	LZIII	789.70	3145.08
	NTF	4° 41.2' E	43° 34.9' N

Table. Test points used to check grid reference conversions.

7.3. Differential Corrections

The method proposed is similar to that attempted by Jeff Stevens: both receivers will be forced to use the same set of satellites, and then the drift in the reference receiver position is used to correct the aircraft receiver position estimate. The difference is that both receivers will be forced to use the same position estimation algorithm (whose assumptions we know) - the one implemented in program DGPS.

8. Sensor Calibration

The principal sensor calibration results are summarised in the following table.

Sensor (channel)	**Measurand DN = 0**	**Measurand DN = fsd**	**Full scale deflect^{n} (fsd)**	**Calibration function**	**Remarks**
pressure (ADC0)	p = 1060 hPa	p = 800 hPa	4095	$p = 1060 - DN.\left(\frac{260}{4095}\right)$ (1 DV = 6.35 Pa = 0.54 m)	Vaisala barometer; gives pressure in hPa
humidity (ADC1)	rh = 100 %	rh = 0 %	2047	$rh = 100 - DN.\left(\frac{100}{2047}\right)$	Vaisala Humitter; gives rh in %
temperature (T_{slow}, ADC2)	T = 60 °C	T = -40 °C	2047	$T = 60 - DN.\left(\frac{100}{2047}\right)$	Vaisala Humitter; gives T in °C

	Calibration	**Comments**
temperature (T_{fast}, ADC3 and ADC15 are two versions of this signal, with different gains)	ADC3: 1 lsb = 5.55 mK ADC15: 1lsb = 11.1 mK	ADC15 should have coarser resolution than ADC3 by a factor of 2. Check calibrations by comparing low frequency behaviour with the slow temperature sensor (ADC2).
Radiation (ADC 4-7; sensors unreliable for later flights)	300 nm UV 1.71 mW m^{-2} nm^{-1} lsb^{-1} 440 nm UV 0.98 mW m^{-2} nm^{-1} lsb^{-1} 440 nm Vis 0.59 mW m^{-2} nm^{-1} lsb^{-1} 670 nm Vis 0.23 mW m^{-2} nm^{-1} lsb^{-1} 870 nm Vis 0.18 mW m^{-2} nm^{-1} lsb^{-1} 940 nm Vis 0.17 mW m^{-2} nm^{-1} lsb^{-1}	The four physical channels of the radiometer can be used in several different ways by changing filters and aperture masks. These calibrations are derived below (Radiation sensor description).

Table: Payload sensor calibration summary.

GPS position and velocity are assumed to have the typical GPS accuracies: several ms^{-1} for the C/A velocity, about 100 m (95%) for horizontal C/A position, about 150 m for vertical C/A position. The accuracy of the differential GPS position seems to be better than 5 m rms (horizontal), and is expected to be about 1.5 times this for height (portions of data when the aircraft is stationary before take-off can be used to check this).

8.1. Pressure

The calibration is based on the manufacturer's stated performance characteristics and careful calibration of the signal amplifiers.

Pressure is used to measure aircraft altitude. This requires knowledge of the local air density at the time of the flight, which depends in air temperature, pressure and humidity.

From Garrett (1992, p.22), Calder (1990, Appendix),

$$\rho = \frac{p}{R_w T} = \frac{p}{R_d\left(1+0.61q\right)T}$$

$$q = \frac{rh(\%)}{100}\cdot q_s\left(p,T\right)$$

$$q_s = 0.622\cdot\frac{e_s(T)}{p-0.378e_s(T)}$$

$$e_s = 610.78\exp\left(\frac{17.269T_C}{T_C+237.30}\right)$$

p is pressure (Pa), T is absolute temperature (K), ρ is density (kg m^{-3}), and R_d is the specific dry air gas constant (= R/m = 8.31/0.029 = 287 J kg^{-1} K^{-1}). T_C is the temperature in °C, q is the specific humidity (mass of water per unit mass of moist air), and e_s is the (saturation) vapour pressure of water in air. Height and pressure changes are then related by

$$\delta h = -\frac{\delta p}{\rho g}$$

This is used to calculate the barometric height of the aircraft.

8.2. Temperature

The calibration of the slow response temperature sensor (T_{slow}) is based on the manufacturer's stated performance characteristics and careful calibration of the signal amplifiers. A fast response temperature sensor (T_{fast}) is also carried which can be calibrated by (1) comparison with T_{slow}, or (2) noting the signal change for a change of "1 lsb" in the resistor bias network used to bring the mean signal within the dynamic range of the output amplifier / digitiser.

8.2.1. Pt wire fast temperature sensor

Kaimal and Finnigan (1994) describe the basic circuit and use of the platinum wire temperature sensor. Initial investigations suggest that the time response of T_{fast} should be better than 10-20 Hz.

The figure below shows the circuit for the Pt wire temperature sensor. The circuit values used are:

Parameter	**Value**	**Parameter**	**Value**
Vs	10 V	R1b3	800 R
R1a	4k3	R2	5k1
R1b	0 - 1k5	R3	150 R
R1b0	100 R	R4	150 R (Pt wire sensor)
R1b1	200 R	Amplifier gain	400 (bandwidth ~ DC - 100 Hz)
R1b2	400 R		

R1 to R3 are precision metal film resistors (temperature stability = 15 ppm K^{-1}). R1b is a binary switchable resistor which allows the zero to be set over the full working range and also provides a calibration signal (by switching R1b0 in and out). The amplifier must have good temperature stability (input offset voltage temperature sensitivity < 1 μV K^{-1}, e.g. AD 524 BD).

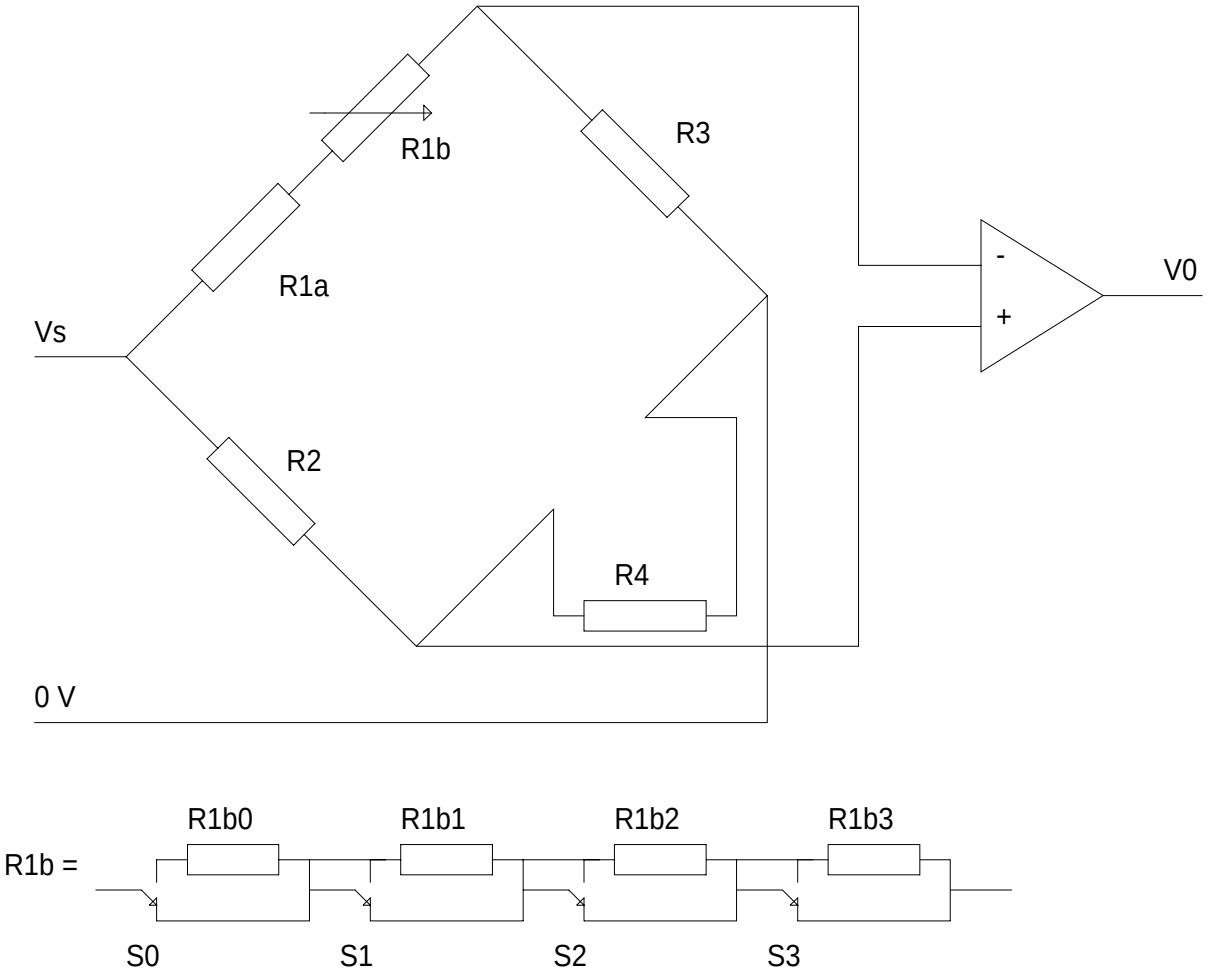

Figure. Fast response Pt wire temperature sensor circuit.

The circuit has been designed so that the operating current through the Pt wire sensor is about 2 mA, which gives a temperature rise of less than 0.1 K. Since temperature fluctuations, not the absolute level, are being measured this is not significant (circuit noise will be much less than the measurement resolution). The fine wire can be exposed to direct sunlight without creating a significant signal: the power absorbed by the wire for 500 W m^{-2} incident radiation is about 50 μW, which gives a temperature rise of 4 mK (the design measurement resolution is approximately 10 mK).

The amplifier output voltage is digitised to 12 bit resolution over a signal range of -5.0 - +5.0 V, i.e. 1 lsb corresponds to 2.44 mV of output signal. For amplifier gain A, ADC voltage resolution Δ, and temperature sensor temperarature coefficient of resistance α, the temperature resolution δT is

$$\delta T = \frac{\Delta\left(R_2 + R_4\right)}{AV_s R_4 \alpha}$$

For the values above this gives δT = 5.55 mK. (This suggests that moving from strong sunlight to shade should just be detectable.)

8.3. Humidity

The calibration is based on the manufacturer's stated performance characteristics and careful calibration of the signal amplifiers along with some simple experiments to measure the time response to a step humidity change.

8.4. Radiation

These sensors were operating with bad power and / or signal connections during most of the experiments (all but the first two flights) and provide no usable data. Further ground-based operation is required. Calibrations are based on the stated detector performance, the (manufacturer-supplied) filter characteristics, and sensor geometry. No comprehensive system calibration has yet been performed.

The circuit for one radiometer channel is shown below. Four channels were built to run in parallel. The parameter values for each channel are given in the following table.

Channel		**Gain**	**R1**	**R2**	**R3 (=R2)**	**R4**	**R5**
1	UV (300 or 440 nm)	-4	24k9	100k	100k	16k5	5k1
2	V1 (440 or 940 nm)	-0.9	18k2	16k2	16k2	5k62	5k1
3	V2 (870 nm)	-0.9	18k2	16k2	16k2	5k62	5k1
4	V3 (670 nm)	-0.9	18k2	16k2	16k2	5k62	5k1

Table. Component values for each of the four radiometer channels. A voltage reference of 4.1 V (e.g. LM9140BYZ-4.1)and bias current of 0.25 mA through R3 is assumed in choosing R3 and R5 (note that R5 should be reduced slightly, e.g. to 4k7, if the same Vref is used for all four channels, instead of providing a separate Vref for each channel).

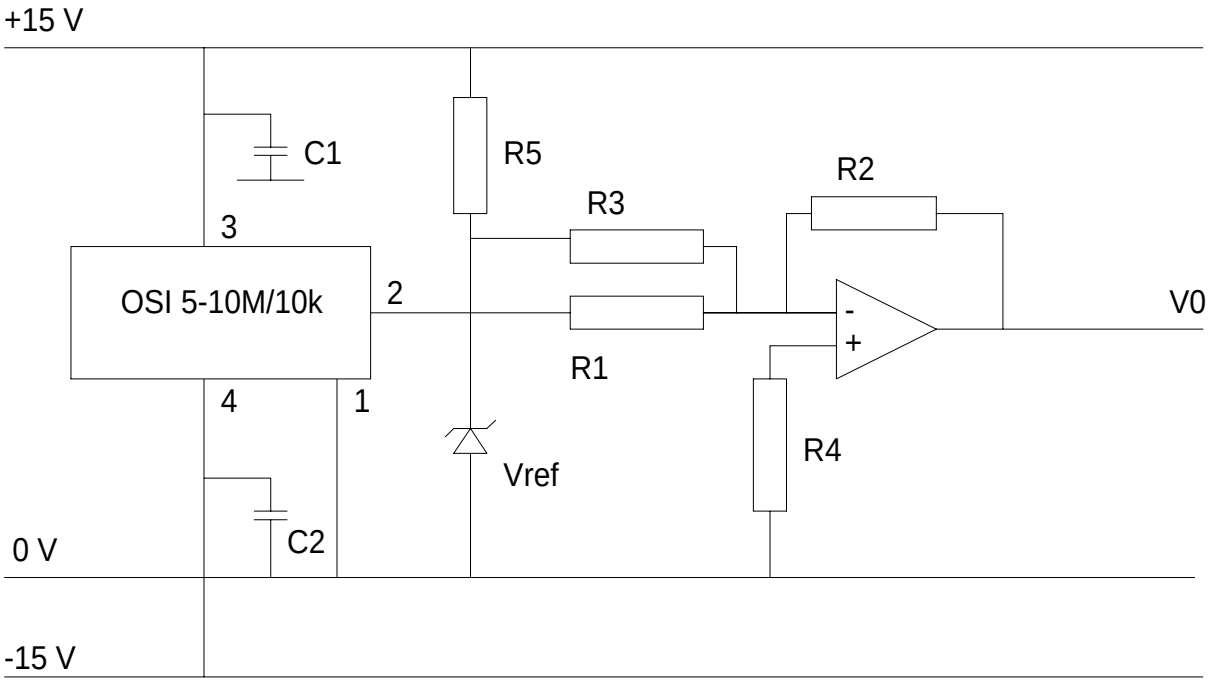

Figure. Circuit for one radiometer channel.

If the channel detector sensitivity is s (V μW^{-1}), the exposed sensor area A (= aperture area, mm^2), filter bandwidth B (nm) and filter transmissivity across the bandwidth , then for a voltage signal V (V), the spectral irradiance (I) is

$$I = \frac{V}{sAB\tau}$$

The following table gives the sensitivities of the different radiometer channels available.

Channel nm	Detector	Sensitivity V μW^{-1}	Aperture mm^2	Filter bw nm	Filter trans.	Irradiance W m^{-2} nm^{-1} V^{-1}	Irradiance mW m^{-2} nm^{-1} lsb^{-1}
300	UV	0.19	5.0	10	0.15	0.70	1.71
440	UV	1.50	0.495	7.5	0.45	0.40	0.98
440	Vis	2.43	,,	7.5	0.45	0.25	0.59
670	Vis	4.00	,,	10.8	0.50	0.094	0.23
870	Vis	5.05	,,	12.2	0.45	0.073	0.18
940	Vis	5.00	,,	13.2	0.45	0.068	0.17

Table. Radiometer channel nominal sensitivities and calibration. 12-bit digitisation of signal over the range -5.0 - +5.0 V has been assumed. Note that the final column gives spectral irradiance per lsb in mW (not W) m^{-2} nm^{-1}.

The four physical radiometer channels can be reconfigured by changing the interference filters and aperture masks; the physical UV channel is normally used for 300 nm or 440 nm.

S.E. Hobbs, 25/8/99