Apertum

Airviro Specification

$$\frac{\partial c}{\partial t}+u\frac{\partial c}{\partial x}+v\frac{\partial c}{\partial y}+w\frac{\partial c}{\partial z}=\frac{1}{\rho}\frac{\partial}{\partial z}\left[\rho\left(K_z\frac{\partial c}{\partial z}+w_s c\right)\right]+\frac{Q}{\rho}$$

Airviro Specification v5.00

Part II: Appendices

Airviro Specification v5.00

Part II: Appendices

Amendments

Version	Date changed	Cause of change	Signature
2.30	130706	Changed paragraph E3.3	KH
3.00	141226	Changes to update to 3.0 functionallity	PI
3.11	150303	Update to 3.11	GS
3.20	161206	Update to 3.20	GS
4.00	181008	Update to 4.00	GS
4.00	180907	Review	GS
5.00	220830	Update	GS

Contents

Appendix B: Data Interfaces to Airviro

B1 Introduction

All of the Airviro databases that contain information input by the system users have ASCII interfaces enabling the possibility of import and export of data. In this section the possibilities for the most commonly used databases are described.

B2 Data Import and Export from the Time Series Database

B2.1 The ASCII Interface to the Airviro Time Series Database

valuedb is a program that provides an ASCII interface to the Airviro time series database. The value records are represented in a well-defined ASCII format. This format can be used for input/output of values to/from the time series database. The format is described in detail in Appendix D2.2 “Standard Time Series Data File”.

valuedb is used with combinations of the following options:

- On extraction, the daylight savings time can be printed according to the value of the TZ (time zone) environment variable (normally daylight savings time is ignored).
- On extraction, it is possible to only select data that has status corresponding to bits that are set in a specified mask.
- A time offset can be added to the data on storage.

Note: On extraction of data, it is possible to mix data from several stations, but when data is imported to the time series database only one station is possible per file. When importing data it is not written directly to the time series database but instead goes to a spool file.

For a general description of the format used to describe the data to be extracted see Appendix D2.2 “The standard time series data file”. Note that it is possible to extract not only the actual data value but also any associated values (e.g. standard deviation, maximum value or status value).

B2.2 Data Export Using the Report Module

The Report module can extract data from the time series database at regular intervals (hourly, daily, weekly, monthly, yearly or a specific event) and save it to a file. Eight standard report generators are provided, see *Airviro Specification v5.00.Part I: Functions in Airviro.* 8. Indico Report.

B3 Data Collection and Distribution

B3.1 The Database Manager

Writing to the Airviro time series database is handled by **avdbm**, the Airviro database manager. Incoming data is stored in a spool directory and **avdbm** scans the spool directory at regular intervals. **avdbm** is able to serve multiple databases if necessary. Spool files that for some reason were unable to be spooled are not deleted and are instead moved to a directory for invalid spool files.

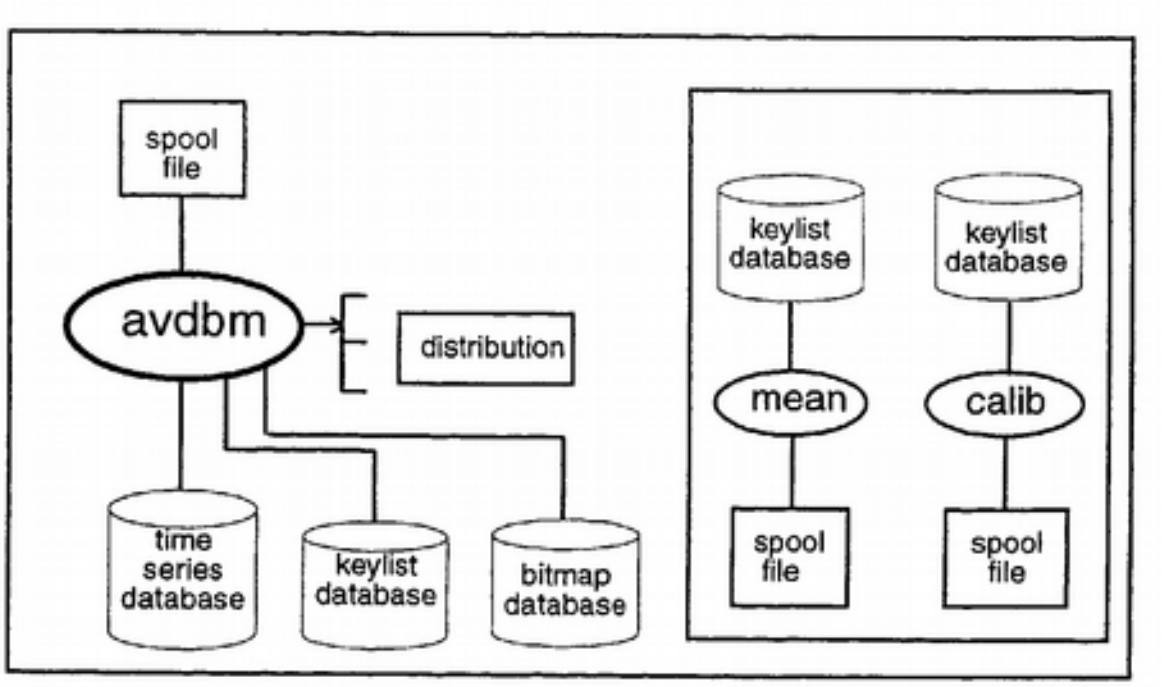

Figure B3.1.i: The use of the different databases with time series data

Different flags can be set in the incoming data indicating that further post processing of the data is required. **avdbm** sets these flags in the keylist database which is scanned by programs such as mean (for calculation of means over a longer time interval) or calib (for calculation of calibrated values). These calculated values are then respooled ready to be processed by avdbm. From Airviro 5.00 it is possible to do all configuration for post processing and distribution in a central configuration file for avdbm.

B3.2 The Data Collection Daemon and the External Protocol

The data collection daemon (**cold**) coordinates data collection activities and keeps track of what is happening. The main tasks of the data collection daemon are as follows:

- controlling data collection activities as a function of time
- sharing common resources (e.g. modems and telephone lines) in an optimal way
- maintaining statistics for data collection activities.

Whereas the **cold** daemon is used to manage resources for all data collection, there is a specific external protocol program for each type of measurement station. The external protocol reads a description of the configuration of each station (telephone number, measured parameters/channels, password, baud rate, alarm limits per channel, etc.). The

program contacts the station when ordered to do so by the data collection daemon and downloads data from the station, producing alarms when necessary for measurement or station status. The purpose of the external protocol is only to extract measurement data from the station, not to perform maintenance activities.

A measurement site which is to be included in an Airviro data collection network must meet the following requirements:

- The measurement data from the instruments for at least the most recent month should be stored at the measurement site in a computer or data logger.
- The data logger should allow a dialogue (described by a well-defined protocol) with the computer collecting data, including the facility to order the transfer of all measured data in engineering units between two points in time. Data should be transferred with checksums, to allow the data collection computer to reorder blocks of data that have arrived corrupt.
- All other interactions with the logger (such as configuration and calibration) can be carried out using software provided by the logger manufacturer.

The task of connecting a new type of measurement station into an Airviro data collection network includes the development of an external protocol program specific to the type of measurement station. See appendix A7 for a list of supported data loggers.

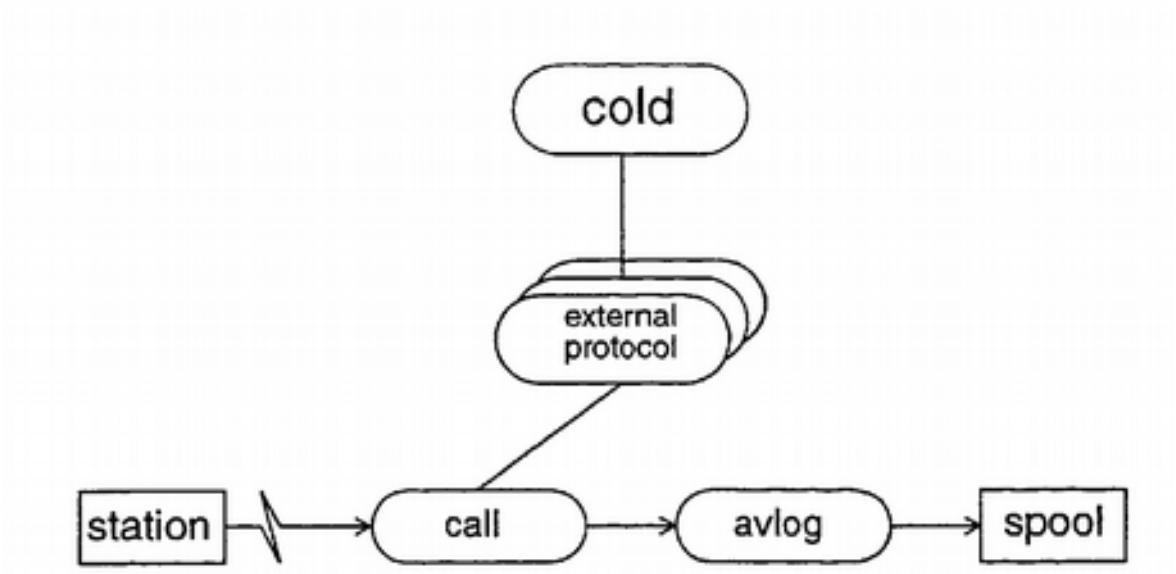

Figure B3.2.i: shows how data is transferred from a station to the time series database. The call process calls the station and starts the relevant scripts. The ***spool*** *process loads the data into the time series database via* ***avdbm*** *(see figure B3.1.i).*

B3.3 The Data Collection Process

A data source (station) from which measurement data are to be collected can be anything from a single measuring device with data transfer functions to a data logger or computer containing measurement data from a large number of instruments. The information necessary for Airviro to contact each station is mainly stored in two places - the station database and the external protocol.

This station database mostly contains information about when to call the station and includes:

- The communications protocol used by the station
- The call times and frequency
- A record of successive bad calls along with bad call alarm limits

The more specific information needed to obtain the data required from a station is stored with the external protocol. This helps to keep the station database clear, because there are different requirements and formats for each type of protocol. This includes:

- The parameters measured at the station
- The station telephone number
- How each parameter is to be processed in the system (storage of values, comparison to validity limits)

The date and time at which data were last successfully retrieved by Airviro are also stored in the station database, so that the correct amount of data is retrieved after a loss of contact between Airviro and the measuring station. The data collection daemon scans the station database at regular intervals and starts the external protocol for each station at the times specified in the station database for downloading of measurement data. If data cannot be successfully retrieved from a certain station, a counter for unsuccessful calls is incremented in the station database for that station. An alarm is generated as soon as this counter reaches a user-defined level. If more unsuccessful calls follow, the counter reaches a second user-defined level where another alarm is generated and the system stops attempting to call that particular station. Once the problem has been rectified the user can reset the number of bad calls for that particular station and restart the data collection. Data collection begins from the time of the last update, as long as there is sufficient memory in the logging equipment.

B3.4 Creation of New External Protocols

Apertum owns a specially developed Logger Communication Language (LCL). It does not need to be compiled and is well structured so that anyone familiar with LCL can easily alter or adapt an existing program even if they were not the original programmer.

LCL is not meant as a general-purpose language although it includes functionality from languages such as C and AWK. The main purpose of LCL is to communicate over a serial line with some kind of computer, usually a more or less intelligent logger. The language is script-based and compiled at run time like AWK and it has a similar way of declaring functions, which gives it the strength of AWK, but it has type checking (with convenient relaxations) and local variables like C. LCL is also built on states since serial communication often brings up the need to restart from a certain point.

Ideas have been fetched from a number of different areas, from commit()/rollback of SQL to chat scripts of UUCP. Other parts stem from experience of frustration trying to get some structure out of code for serial communication which needs to be adjusted each time the other end updates it’s software.

LCL has few types and no way to derive new types although this is a possible extension. Note however that the string type has unlimited dynamic length and that there is a time type which is very convenient for data retrieving purposes.

LCL has its basic control flow in common with most modern languages: statement grouping, decision making (if-else), loop (for), but also includes “hunters” which is similar to C’s switch.

Functions can return any or no (void) type and they can be called recursively. Local variables can be declared at the beginning of each statement block.

As strings play an important part in serial communication, LCL has several string manipulation functions that are tailored for this use, support for arbitrary contents (NULL etc.) and slicing to mention two.

Normally LCL should work on a specified device (like /dev/cul00) and therefore a lot of the built-in functions are tailored for this. Standard input and output is mainly used for trace during development.

In keeping LCL similar to existing languages it is hoped that a programmer should be able to learn to take advantage of it quickly. Shortening the actual development of a serial protocol type product by a factor of 5-10 is not unrealistic, but the main advantage lies in that later adjustments can be done easily by someone else than the original programmer.

B3.5 Data Distribution

During the process of storing data, the database manager (avdbm) will store data in internal (spool-file) format for distribution to other Airviro systems. The configuration of the distribution is made in a file stored under *rsrc* directory in the domain. Rules are set up specifying what time series that should be distributed to other systems. The spool files are then stored in sub directories with the name of the distribution site under the *dist* directory in the domain. These files can then be copied to other Airviro systems using the Dist (Dial up), FDist (FTP), HDist (Http) and EDist (email) protocols. The task of the distribution protocol is to connect to the computer to which data are to be sent, transfer the data and delete the files from the distribution directory in the local computer.

B4 Data Import and Export from the Emission Database

B4.1 The EDB ASCII Interface

An EDB is partly made up of databases that store emission sources (point, area, road and grid) and partly of databases that store significant information about these sources. The latter category contains substances, searchkeys, fuels, formulae, road types and vehicles, which are referred to as underlying databases.

When you create an EDB it is important that the underlying databases are defined before you begin to enter emission sources. Normally these are defined via menus and windows except for grid sources, which have to be defined in files. The primary key to these databases is an index. All references from the databases containing the sources to the underlying databases use the indices. You can find out the index to the entries in the underlying databases with the help of the ASCII interface for the respective database.

The contents of these databases are described in Appendix C3.

B4.1.1 Substances and Searchkeys

Substances and searchkeys are the most fundamental parts of the underlying databases. The substance database can contain up to 1023 substances. Apertum has a substance list that is recommended for customers to use, so that it is possible to combine databases.

There are 5 groups of searchkeys. The first two contain 128 records while the other three are limited to 32. For point sources the two first search keys are faster when they are used for searching.

For the searchkeys and substance database there is an ASCII interface called **subdb**.

These substances and searchkeys can be used to search in the EDB.

B4.1.2 Substance Groups, Fuels and Formulae

Substance groups and fuels are stored in the same database. The primary key to this database is an index. This must therefore be unique both for the substance groups and the fuels. The primary key to the formula database is also an index, which must be unique.

If you change a formula, substance group or fuel all the emissions for the emission sources that reference these will be altered. To easily be able to change the emissions for groups of sources these should reference the same group of entries.

The ASCII interface to the substance group and fuel database is the program **fueldb**, and to the formula database the program **formuladb**.

B4.1.3 Point and Area Sources

Point and area sources are stored in the same database. The primary key for the point and area database is the bottom left coordinate pair. This must therefore be unique for each source. For point sources the upper right coordinate pair is set to 0.

The difference between the point and area sources, apart from their geographical extension, is that a point source stores the characteristics describing the emission. These characteristics include the height of the chimney, the inner and outer diameter of the chimney as well as the velocity and temperature of the exhaust gas. All this data must be given so that a correct plume can be calculated. In addition the height and width of the surrounding buildings can be specified. For area sources all these data are 0.

For use with the US EPA AERMOD model it is possible to specify information about surrounding houses in 18 different directions.

Application specific data can be stored in ASCII large objects. It is up to the application to interpret these data. Special web pages need to

be implemented in order to input the data in a graphical user interface.

The longest side of an area source must be less than 10 kilometres.

A point source refers to a string of underlying databases. These are the substance, searchkey, fuel and formula databases. These can also be used as search criteria.

Search strings: The name, info and info2 strings are search strings. This means that by using pattern matching you can set restrictions when searching. Information that you may want to search by but which is not stored in the search keys should be stored here. If more

than one type of information is stored in a search string then a special symbol should be used to distinguish between them. Note that the search time is greater for search strings than for searchkeys.

The ASCII interface for the point and area database is the program **datadb**.

B4.1.4 Grid Sources

The program **griddb** can be used to create or export an emission grid layer, which is part of the emission database.

B4.1.5 Vehicles and Road Types

The primary key to the vehicle and road type databases is an index. This must be unique.

If you change a vehicle or a road type then all emissions for the emission sources that reference these will be altered. To easily be able to change the emissions for groups of sources these should reference the same group of entries.

The ASCII interface to the vehicle database is the program **vehdb**, and to the road type database the program **roadtypedb**.

B4.1.6 Road Sources

The primary key to the road database is the coordinate pair for the bounding rectangle containing the set of vectors describing the road. This is calculated when the source is saved in the EDB.

A road source references the road type database, which in turn references the vehicle database. You can use both road types and vehicles as search criteria.

Searchkeys and search strings work in the same way as for point and area sources.

Congested traffic situations are dealt with by specifying a congested threshold on the link. If the threshold is passed for any hour of the emission simulation, the simulation will use an alternative congested speed when calculating the emission for that link and hour.

Noise factors can be added to road links and a noise emission map can be presented. The noise emission is presented as colored links.

Application specific data can be stored in ASCII large objects. It is up to the application to interpret these data. Special web pages need to be implemented in order to input the data in a graphical user interface.

Information about physical attributes of the road as well as information about the street canyon in which the road resides can be specified for use with canyon and other dispersion models.

The ASCII interface for the road database is the program **roaddb**.

B4.1.7 Miscellaneous

When you enter sources via the ASCII interface you can check that you do not reference entries that do not exist by using the program **chkedb**.

Note that the ASCII interface just adds entries to the existing databases. If you want to empty a database you have to copy an empty database. The database files for the global EDB are stored in /usr/airviro/data/NAME/dba/edb. The empty database files are stored in /usr/airviro/data/NAME/emp/edb. NAME is the name of the database.

It is also possible for each user to create one or several personal EDBs, which he can choose to work with instead of using the global EDB. These can be used for test scenarios or for separate projects. It is possible to create a new personal EDB by either copying an existing EDB or by copying the empty EDB.

B4.2 The EDB Report Generator

Reports about the contents of an EDB (either the global EDB or a personal EDB) can be generated from within the search subwindow of the EDB module.

Reports can be created in four different ways:

1. **Database report:** All records from a specified database are output. Reports can be made for the fuel, formula, substance, searchkey, vehicle and road type databases.

2. **Static report for emission sources:** Records from the point, area, road and grid databases are output. Search criteria can be placed on the output and it is also possible to choose which fields should appear in each record.

3. **Dynamic report for emission sources:** Records from the point, area, road and grid databases are output. Search criteria can be placed on the output and it is also possible to choose which fields should appear in each record. In addition the emission figures can be obtained which depend upon what time and temperature restrictions have been placed. This report can also be produced in a more compact SKV semicolon separated format that is intended for export. The SKV report can also be produced in shape format.

4. **Grid report:** A two dimensional array is created that describes the emission from a region. Search conditions can be placed as well as time and temperature restrictions. It is also possible to specify two dates and hours and then create a number of grids (hour or day mean) between the dates. Output is either in grid, GIS or shape format.

Reports can be made for the following databases:

- the data database (point and area sources) as a plain data report, a data report with extended fuel information, a data report with extended formula information or a data report with both extended fuel and formula information.
- the fuel database
- the formula database
- the substance list with index numbers
- each searchkey list
- the road database as a plain road report, a road report with extended road type information or a road report with extended road type with extended vehicle

information. These 3 report types can also take an option to include the vector chain describing each road.

- the road type database as a plain report or as a road type report with extended vehicle information.
- the vehicle database.

The report can be made for any specified edb, with or without and empty fields, and it is also possible to print each record on a new page. There are also options for alphabetic sorting, sorting by emission size or no sorting.

By default the output is to a viewer but it can be redirected to a printer or file. This can be further influenced with the help of filters. Customised filters can be requested from Apertum as a supplement to the system.

B4.2.1 Examples of Output Formats

1. Database report:

The following example shows the first few records from the fuel database:

```
Fuelname:                   Rosenlund_HP1-5_oil(16)
Energy value:               42.70 MJ/kg
           Substance   Emission
             NOx(3)    (0.1 g/MJ)
             SO2(15)   (0.40000%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Fuelname:                   Rosenlund_MT_gas(17)
Energy value:               51.90 MJ/kg
           Substance   Emission
             NOx(3)    (0.1 g/MJ)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Fuelname:                   Rosenlund_MT_oil(18)
Energy value:               42.70 MJ/kg
           Substance   Emission
             NOx(3)    (0.1 g/MJ)
             SO2(15)   (0.40000%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
...
```

2. Static report for emission sources:

The following example shows the first two records for the point database. Only certain fields have been selected:

```
Name:                       ANGEREDS VÄRMECENTRUM
Chimney:   Height:          70 m
           Out:             4.80 m
           In:              1.90 m
           Gas temp (abs):  1503C
                 Gas flow:  5 m/s
Substance:
              NOx(3)    7.00000 ton/year
              SO2(15)   10.00000 ton/year
Formula:    STANDARD (1)
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Name:                        BEROL NOBEL MÖLN:1
Chimney:   Height:           13 m
           Out:              0.48 m
           In:               0.20 m
           Gas temp (abs):   2003C
                 Gas flow:   0 m/s
Substance:
             NOx(3)     3.10000 ton/year
             SO2(15)    0.70000 ton/year
Formula:    STANDARD (1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
...
```

3. Dynamic report for emission sources:

The following example is in the more compact SKV format but has the same field selection and search conditions as the static report. The difference is that the emission figures are also given:

```
"Name";"Chimney height";"Chimney outer diameter";"Chimney inner
diameter";"Chimney gas temperature (absolute)";"Chimney gas
temperature (relative)";"Chimney gasflow";"Fuel index";"Fuel
name";"Formula index";"Formula name";
"ANGEREDS VÄRMECENTRUM";70;4.8;1.9;150;0;5;0;"";1;"STANDARD";
"BEROL NOBEL MÖLN:1";13;0.48;0.2;200;0;0;0;"";1;"STANDARD";
"BJÖRNDAMMENS HETV.";42;1.9;0.99;180;0;0;0;"";1;"STANDARD";
"CHALMERS VÄRMEC:1";31;1.2;1;150;0;4;0;"";1;"STANDARD";
"CHALMERS VÄRMEC:2";31;1.2;1;150;0;4;0;"";1;"STANDARD";
"ERICSSON RADAR:1";38;1.89;0.53;150;0;3;0;"";1;"STANDARD";
"GRAAB SÄVENÄS";126;6;2.8;75;0;16;0;"";1;"STANDARD";
"GÖTEBORGS STADS BOS";50;0;0;175;0;15;0;"";1;"STANDARD";
"HEDLUNDS PAPPER:1";9;0;0;0;0;0;0;"";1;"STANDARD";
"MARCONICENTRALEN";65;2.5;1.4;100;0;4;0;"";1;"STANDARD";
"MÖLNLYCKE:1";55;1.8;0.74;240;0;13;0;"";1;"STANDARD";
"NYNÄS SUPPLY:1";19;1.4;1.2;225;0;4;0;"";1;"STANDARD";
...
```

4. Grid report:

The following is a 10x10 grid output for average annual NOx emissions:

```
1266075 6399709
10 10
1000 1000
1
1.658623E+00 9.427182E-01 1.615039E-01 2.595479E+00 0.000000E+00
3.373053E-01 4.559744E-01 3.195540E+00 3.610402E-02 4.220088E-01
2.337220E+00 9.580520E-01 4.517313E+00 3.329525E+00 1.541839E-01
1.169524E+00 7.306970E-01 3.608456E+00 1.625000E+00 7.821343E-01
1.659858E-02 4.748488E+00 1.316224E+00 4.864694E-01 3.319005E+00
1.760191E+00 2.112431E+00 5.193089E+00 0.000000E+00 0.000000E+00
0.000000E+00 3.483438E+00 2.828047E+00 1.010176E+00 3.606980E+00
3.579331E+00 6.382233E+00 1.361631E+00 1.387961E-01 0.000000E+00
4.546407E+00 4.063155E-01 3.996959E-01 2.790251E+00 6.302989E+00
5.600060E+00 6.494990E+00 6.686031E-01 8.347538E-01 0.000000E+00
8.998015E-01 2.634292E+00 1.581233E+00 1.770756E+00 1.429001E+00
4.339622E+00 6.380318E+00 2.389216E+00 1.222308E+00 4.324739E-01
```

```
4.966676E+00 1.111098E+00 1.463058E+00 2.334930E+00 4.645815E+00
1.960136E+00 6.070888E+00 4.259719E+00 1.026278E+00 2.042155E+00
3.303026E-01 3.994831E-01 7.701138E-01 1.370270E+00 1.542747E+00
2.876454E+00 2.824568E+00 3.502115E+00 1.540404E+00 6.769530E-01
0.000000E+00 7.611246E-01 4.044795E-01 3.744966E-01 3.987195E-01
4.574746E-01 2.452919E+00 2.544028E+00 5.208084E-01 4.584284E-01
5.782919E-01 3.518259E-01 1.228893E-01 6.693470E-01 5.028050E-01
7.999193E-02 2.624649E+00 1.986067E+00 3.861975E-01 1.351577E+00
0
```

B4.3 Wedbed

Wedbed is a tool that integrates the Airviro emission database into MS Excel®. It is possible to export the data to Excel, make changes and import the data back to Airviro or store an emission database in your PC as a workbook.

With Wedbed you can make an extensive consistency check on the loaded emission database, merge EDB:s and split EDB:s. See *Appendix 1D Wedbed. Volume 1 Working with the emission database.*for a more detailed description.

B5 Import and Export of Fields in the Dispersion Module

An interface is provided that can be used to export the field currently displayed to a file. The field can be a combination of several existing dispersion results. The same interface can also be used to import a field as an external result.

An import/export file has the following format:

```
#AREA 1261750 1281750 6394750 6410750
#DESC This is a small grid for a clear illustration
#GRID 4 3
0 1 2 1
1 2 4 2
0 1 2 1
```

The #AREA line describes the coordinates for the area on the map that this grid corresponds to, in the format x(left) x(right) y(bottom) y(top). The unit is metres. Most map coordinate systems have the origin in the lower left corner.

The #DESC line is optional. When a user exports a file it is possible but not compulsory to type in a description. If #DESC does not exist in an import file, the user will be prompted to enter a description.

The #GRID line describes the format of the grid that follows. The first digit gives the number of columns (Nx) and the second gives the number of rows (Ny). The lines that follow should contain Nx*Ny values - not necessarily Ny lines with Nx figures on each line, although this is strongly recommended as the grid then matches the representation on the screen. The first figure of the grid represents the grid square for the top left hand corner, the second represents the grid square to the right of the first one and so on. New lines are not significant - the second row of the grid begins when Nx figures have been read.

Note: The given area describes the grid squares, i.e. the given grid values are considered to be in the centre of each grid square.