FFI RAPPORT

A GRAPHICS-INDEPENDENT COMPUTER PROGRAM FOR RADIO COMMUNICATIONS CALCULATIONS

ÅSEN, Walther

FFI/RAPPORT-2006/03461

A GRAPHICS-INDEPENDENT COMPUTER PROGRAM FOR RADIO COMMUNICATIONS CALCULATIONS

ÅSEN, Walther

FFI/RAPPORT-2006/03461

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
P O Box 25, NO-2027 Kjeller, Norway

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

UNCLASSIFIED

P O BOX 25
N0-2027 KJELLER, NORWAY

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER FFI/RAPPORT-2006/03461	2) SECURITY CLASSIFICATION UNCLASSIFIED	3) NUMBER OF PAGES 14
1a) PROJECT REFERENCE FFI-II/1028/913	2a) DECLASSIFICATION/DOWNGRADING SCHEDULE -	

4) TITLE
A GRAPHICS-INDEPENDENT COMPUTER PROGRAM FOR RADIO COMMUNICATIONS CALCULATIONS

5) NAMES OF AUTHOR(S) IN FULL (surname first)
ÅSEN, Walther

6) DISTRIBUTION STATEMENT
Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

	IN ENGLISH:		IN NORWEGIAN:
a)	Radiowave propagation	a)	Radio bølgeutbredelse
b)	Interference	b)	Interferens
c)	Jamming	c)	Jamming
d)	Radiocommunication	d)	Radiokommunikasjon
e)		e)	

THESAURUS REFERENCE:

8) ABSTRACT
A simple program written in the C programming language is documented. The code is given the name CalcRadio and calculates the effects of radiowave propagation, antenna gain, interference and jamming. The program treats both digital and analogue waveforms, provided that conversion tables between SNR and BER values for the particular waveforms are defined.

9) DATE 2006-11-06	AUTHORIZED BY This page only Vidar S. Andersen	POSITION Director

ISBN 82-464-1043-1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

CONTENTS

A GRAPHICS-INDEPENDENT COMPUTER PROGRAM FOR RADIO COMMUNICATIONS CALCULATIONS

1 INTRODUCTION

The purpose of this report is twofold. Firstly, it documents in detail how to use a specific piece of software developed for radio communications calculations, including directory and file structures. We have named this software package ‘CalcRadio’. Secondly, it serves as an example of how to construct a graphics-independent and database-independent computer program which can be easily integrated into other software (user programs) using a file format interface. The user programs, which are the end products, will of course rely on graphics and possibly database representations.

There exist many computer applications that use radio wave calculations, involving terrain (diffraction) effects at VHF and UHF frequencies. It is often problematic to reuse the propagation part of the software in different applications because the propagation software developed often assumes some sort of graphical interface to the application. Reuse is restricted by complex computer programming knowledge, and in particular knowledge of the chosen ‘graphics software’.

In addition to the pure radio wave propagation part, it is often of interest to calculate interference between different systems which are operating at the same radio frequency. It then becomes important to take the effect of antenna diagrams into account. Antenna diagrams may be presented in a flexible way, preferably as look-up-tables on file, in order to make calculations or measurement methods of antenna diagrams independent of radio calculations. Antenna diagrams are in principal 3-dimensional objects, but are by antenna manufacturers often represented by two cuts, one in each of the horizontal and vertical planes. A method is therefore necessary to transform the two 2-dimensional cuts into a 3-dimensional representation.

With digital communications becoming increasingly important, it is essential in radio calculations to take into account required quality in terms of the Bit Error Rate (BER) that a particular system will tolerate. The problem may be treated in a general way where BER is simply translated into required Signal to Noise Ratio (SNR) for each modulation+coding of the system. Such translations from BER to SNR for different modulations, modulation indices and (error correcting) codes should preferably be kept as look-up-tables on file, in order to make calculations or measurement methods of BER to SNR translations independent of the radio calculation.

In a system for communication planning (or communications Electronic Warfare), a mechanism for calculating the effect of a jamming signal should also be contained within the radio calculations. A jamming signal may be used by an operator to prevent someone else’s communications, and may have entirely different properties from an interfering signal. The jamming signal may be designed to be efficient against the waveform it jams, and the victim (receiver) being jammed will maybe not escape the problem just by changing the transmission frequency. A general tool for jamming calculations should assume that a jammer focuses all its power, in a continuous manner, on the entire received bandwidth of the receiver, and that it jams with a waveform similar to that being used by the communication receiver. A more intelligent jamming calculation would be to consider the power distribution of the jammer, and calculate the effect of optimal operation on a victim. This requires detailed knowledge of the waveform, coding and/ or the communication protocols of the victim, which may not usually be the case. Such detailed calculations may also lead to a high security classification of the program code, which again will make it less attractive for reuse.

Wave propagation calculations are statistical in nature. The '50 % of the time/ 50 % of the locations' values is what you usually get from a wave propagation calculation. You rarely see tools that calculate the 99 % value or the 99.9 % value, which really represents the required up-time that you are interested in. A radio calculation tool should take this statistical variation into account and thus allow for a requirement on the up-time of your communications links.

2 OPERATION OF CalcRadio

The computer program CalcRadio described in this document will provide a useful tool to address most of the requirements stated in the previous chapter.

The emphasis of CalcRadio is on simplicity and flexibility. Therefore it is kept as simple as can be: No manual interaction. CalcRadio takes all of its input from files (the entry point/ main input file is 'inputfile.txt'), makes its calculations and then produces its overall output to another single file, the result file ('resultfile.txt').

In the input file the data on receivers and transmitters must be paired as links. Information about each link is contained in one single line of text of the input file. For jammers this means a dummy receiver is required, (with receiver modulation code set to 0). It is required that links on the same frequency are grouped together. It is also required that links on same 'nets' are grouped together. Being part of the same 'net' means that interference is not calculated internally between those emitters and receivers that belong to that 'net'.

The 'inputfile.txt' may look like this (example):

```
6
61.000 11.143 60.950 11.977 10.0 03.0 98.7 90.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 1 0.045 -110.0 1 4 2 7
61.000 11.143 60.882 11.677 10.0 03.0 98.7 90.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 1 0.045 -110.0 1 4 2 7
61.000 11.143 60.648 10.797 10.0 03.0 98.7 90.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 1 0.045 -110.0 1 4 2 7
61.000 11.143 60.552 10.966 10.0 03.0 98.7 30.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 2 0.045 -110.0 1 4 2 7
61.000 11.143 60.437 11.115 10.0 03.0 98.7 30.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 2 0.045 -110.0 1 4 2 7
61.000 11.143 60.237 10.993 10.0 03.0 98.7 30.0 1 1 000.0 000.0 000.0 000.0 000.0 000.0 2 0.045 -110.0 1 4 2 7
```

The number on the first line (6 in this example) indicates the total number of links (data elements or number of lines of text) of the file.

From left to right on the following lines the C programming variables to be read from 'inputfile.txt' are (ARRLEN is the maximum number of lines/records and may be set to a value in main.c, and may be restricted by memory limitations):

```
float aemit_lat[ARRLEN]; /* Longitude of emitter, in  degrees */
float aemit_lon[ARRLEN]; /* Lattitude of emitter, in  degrees */
float arec_lat[ARRLEN];  /* Longitude of receiver, in degrees */
float arec_lon[ARRLEN];  /* Lattitude of receiver, in  degrees */
float aEmitAntH[ARRLEN]; /* Emitter antenna height, in  meters */
float aRecAntH[ARRLEN];   /* Receiver antenna height, in  meters */
float aEmitFrequency[ARRLEN]; /* Frequency for link, in  MHz */
float aEmitPower[ARRLEN];     /* Emitter power without antenna gain, in dBm */
int   aRecAntTypeNo[ARRLEN];  /* Receiver antenna number, according to antenna lookup
table file */
int   aEmitAntTypeNo[ARRLEN]; /* Emitter antenna number, according to antenna lookup
table file */
float aEmitAntDir[ARRLEN];      /* Emitter antenna azimuth direction */
float aRecAntDir[ARRLEN];       /* Receiver antenna azimuth direction */
float aEmitAntElev[ARRLEN];      /* Emitter antenna elevation */
float aRecAntElev[ARRLEN];       /* Receiver antenna elevation */       float
aEmitAntRoll[ARRLEN];      /* Emitter antenna roll, FOR FUTURE USE */
float aRecAntRoll[ARRLEN];       /* Receiver antenna roll,FOR FUTURE USE */
int   arec_net_assoc[ARRLEN]; /* Net association number, must be different for different nets,
nets must be grouped *
float arec_bw[ARRLEN];        /* Receiver bandwidth, in MHz */
float arec_sens[ARRLEN];     /* Receiver sensitivity, in dBm */
int   arec_mod[ARRLEN];      /* Receiver modulation, (if =0, this means emitter in link is
jammer), according to modulation lookup table */
int   arec_modindex[ARRLEN]; /* Receiver modulation index for the chosen modulation,
k=1,2,3 ....etc */
int   arec_ber_tolerance[ARRLEN]; /* Receiver bit error tolerance, 1 means 10^-2 .., 8 means
10^-9 */
int   arec_uncert_no[ARRLEN]; /* Propagation uncertainty
margin,1=0.1%,2=1%,3=10%,4=90%,5=99%,6=99.9%,7=50% */
int   noof_links;            /* The total number of links scanned from xml file */
```

The 'resultfile.txt' will look like the 'inputfile.txt', except that it will **not** contain the first line which tells the number of input records, and except that it will contain **one extra element at the end of each line:**

```
int   astate[ARRLEN];  /* The state of each of the links after calculation -2 means jammed,-1
means interfered,1 means OK, 0 communications power too low */
```

3 FILE STRUCTURE

'C:\CalcRadio\' contains all information necessary to run and change the computer program, including all input and output files.

Subdirectories:

3.1 ‘C:\CalcRadio\Program’

At this level one finds all *.c – files and *.h – files that are used in the making of the *.obj - files and *.exe – files, when using a C compiler.

There is no ‘makefile’ structure provided here, but the file ‘linker.txt’ is a file which lists helpful commands to execute for compilation and linking in a simple command prompt window. (These may be copied and pasted into the command window).

If you run the lcc command (to compile files), the compiler will tell you that it is not happy, since you have not used a proper installation procedure for the compiler. Just write C:\CalcRadio\SimpleCompiler\include as response to this (each time you execute lcc) and it will do the compilation anyway.

This level also contains the ‘inputfile.txt’ needed to run the program, and the output file ‘resultfile.txt’.

3.2 ‘C:\CalcRadio\SimpleCompiler’

Contains a simple ‘freely downloaded’ C compiler that is able to compile and link the code (even though it gives a lot of warnings with the current state of the C code).

3.3 ‘C:\CalcRadio\dted’

This directory contains digital terrain elevation data (dted). For convenience we have loaded some example files to this directory. One must provide the dted for the complete geographical area where the calculations are to be run.

Each dted file contains elevation data for an area of one degree latitude by one degree longitude. The filenames must be constructed in the following manner, using the geographical lat/ lon coordinate of the south-west corner of the data area: The first two positions of the filename are made from the latitude (integer), starting with zero if latitude is less than 10. Next position is the letter ‘n’ (meaning ‘north’). Next three positions are longitude, starting with zero if longitude is less than 100. Last position of the file name is the letter ‘e’ (meaning ‘east’). Examples of valid file names: 59n007e, 33n123e.

The files are really just containing a long array of ‘unsigned short’ integers, which are heights in meters, and can be read from the file running from west to east, keeping latitude constant. When the values for one latitude is finished, the next one to the north is read, again running from west to east, and so on until the whole one by one degree area is covered. Thus the long array of ‘unsigned short’ should really be interpreted/ addressed in this way as a 2-dimensional matrix.

Up to and including the data for 69 degree latitude, the array contains 600 points east by 1200 points north. From 70 degrees north and northwards, the array contains 400 points east by 1200 points north. So the one by one degree files become smaller from 70 degree latitude onwards. This ensures that the density of height data remains approximately 1 point pr 100 meters in each direction. The file structure is similar to what is usually supplied as dted files, except that it is stripped for some additional information.

For historical reasons the two bytes which make up each unsigned short in the dted have switched places, and they are therefore switched back by the code during execution. It might be a good idea to make a small C program to turn the bytes of all the terrain height data into the order in which they could be used directly by the calculation program CalcRadio. Then one must also remember to change (and thus simplify) the code that reads the height data.

3.4 'C:\CalcRadio\Antennas'

This directory contains the file 'antennas.txt' which lists all the antennas that are being used by the CalcRadio program. On each line this file contains an antenna number, which must be equal to the line number of the file, and a corresponding antenna file name.

The line number is the identifier which is the reference to the chosen antenna, and is input through 'inputfile.txt' and stored in the (array) variable arec_mod of the C program.

In order to introduce a new antenna pattern, a new line must be entered and a file name for the antenna diagram must be given.

All antenna diagram files must reside on this directory.

An example of 'antennas.txt' contents:

```
1 rec_ant_pattern_95.txt
2 rec_ant_pattern_96.txt
```

An example of antenna diagram file contents:

```
NAME GEORG1
FREQUENCY 1
GAIN 0.0 dBi
TILT
HORIZONTAL
10.0000    5.8000
30.00      10.4000
90.0       4.6000
180.0000   4.2000
270.0000   0.4000
360.000    20.0
VERTICAL
0.0     0.0
```

The first 4 lines are not currently used for anything and are just ignored by the CalcRadio program, but must be present in the file. The figures of the table are usually provided by the antenna manufacturers, or they may be calculated by special software or measured.

The keyword HORIZONTAL must be listed on a separate line, followed by float number pairs, one on each line, giving horizontal antenna diagram angles (between 0 to 360 degrees) with respective gain in dBs in different directions relative to an isotropic antenna.

The same procedure is used to present the vertical antenna diagram, but with vertical angles limited to numbers between -90 and 90 degrees.

3.5 ‘C:\CalcRadio\Modulations’

This directory contains the file ‘modulations.txt’ which lists all the modulations that can be currently handled by the CalcRadio program. On each line this file contains a modulation number, which must be equal to the line number of the file, and a corresponding modulation file name.

The line number is the identifier which is the reference to the chosen modulation, and which is input through ‘inputfile.txt’ and stored in the (array) variable arec_mod of the C program.

In order to introduce a new modulation, a new line must be entered and a file name for the modulation matrix must be given.

All modulation files must reside on this directory.

An example of ‘modulations.txt’ contents:

```
0 ber_mpsk.txt
1 ber_mpsk_diff.txt
```

An example of the contents of a modulation matrix file:

```
 4.4   6.9   8.6   9.8  10.8  11.6  12.0  12.9
 7.4   9.9  11.6  12.8  13.8  14.7  15.0  15.9
12.1  14.8  16.7  17.9  18.9  19.8  20.5  21.0
17.5  20.5  22.3  23.7  24.7  25.6  26.0  26.9
23.0  26.2  28.0  29.5  30.6  31.4  32.0  32.9
28.7  32.0  33.9  35.4  36.5  37.3  38.0  38.8
```

All the numbers of this matrix are SNR values (in dBs). They give the SNR threshold needed for a combination of required modulation index $M=2^k$ (increasing M vertically from top down) and Bit Error Rate, $BER= 10^{-(1+x)}$ (increasing BER from left to right). k is running from 1 to 6, and x is running from 1 to 8. k and x are two of the inputs of ‘inputfile.txt’ and are stored in the (arrays of) variables arec_modindex and arec_ber_tolerance respectively of the C program.

4 ADOPTING THE PROGRAM FOR YOUR USE

In order to use this program you first of all need to supply **antenna files for the antennas that are to be used by the communications and jamming** equipment that you investigate, and you must also supply **BER to SNR conversion tables for modulations** that are relevant for your communications equipment. These ‘communications hardware’ related input files are in general useful, and once they are obtained they may be used for all later scenarios. Examples of files are supplied.

Each country has its own **digital maps of terrain elevation data (dted)**. These maps may be in different digital formats, and/or they may be similar in format except for preambles. In order for the program to work in your country you have to either reshape your data into the form described in chapter 3, or you need to rewrite the C function

```
int InterHeight(float longitude, float latitude, unsigned short *height)
```

which is contained in the file elevation.c to make it fit with the format you supply.

Independently of the format of your terrain elevation data it should be easy to write such a function which returns altitude above sea level in meters for a given longitude and latitude of position.

Now that all these basic input files are supplied, the rest is quite easy.

The scenarios that are run by the program are input from a single input file, as specified in Chapter 2. To produce this file you may use a text editor, but what is more elegant is to use your favourite Geographical Information System (GIS) infrastructure to input the data. The data must then be stored in the required input file format, and CalcRadio can be executed from the GIS. After CalcRadio has done its job you may want your GIS to read the output file. The results may then be viewed by the GIS. You may for instance choose to use the GIS to indicate communications status between emitters and receivers (not OK, OK, jammed, interfered) by drawing lines of different colours between them.

The computer C code is made available, together with a simple compiler, as described in Chapter 3, so if you intend to use for instance an xml-file structure for input and output, you simply supply your own code for reading and writing the necessary xml input and output files (to replace the calls to fscanf and fprintf of main.c).

5 REFERENCES

(1) The International Telecommunications Union (ITU): ”Propagation by diffraction”, ITU Recommendation ITU-R P.526-7

(2) Walther Åsen: “Comparison of measurements with prediction methods for propagation by diffraction at 88 – 108 MHz” , IEEE Transactions on Antennas and Propagation , vol. 52, NO. 6, June 2004

(3) Walther Åsen, Jørn Kårstad: “Protection of Network Centric Military Radio Communications”, FFI rapport 2004/0248, ISBN 82-464-0860-7

(4) Asta Villanger; Walther Åsen: ”Teknisk beskrivelse av biblioteksrutiner i VULCANO”, FFI/RAPPORT-93/7024