

https://dspace.org/

Research into the usability of software produced in a utility for the utility and consultants

Stanford, Gareth

2008-05-12T12:30:26Z
http://hdl.handle.net/10539/4794

Downloaded from a DSpace repository. https://dspace.org/

RESEARCH INTO THE USABILITY OF SOFTWARE PRODUCED IN A UTILITY FOR THE UTILITY AND CONSULTANTS.

Gareth Stanford

A research report submitted to the faculty of Engineering and the Built Environment, of the University of the Witwatersrand, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg 2007

DECLARATION

I declare that this research report is my own unaided work. It is being submitted to the Degree of Masters of Science in Engineering at the University of Witwatersrand, Johannesburg. It has not been submitted before for any other degree or examination at any other university.

……
(Signature of Candidate)

…….. day of …….. year ……...

ABSTRACT

RSAT (Reticulation Sag And Tension) software is a tool available for use as part of the medium voltage Eskom Distribution standard. This software is scrutinised for its usability and for errors such that it can be revised to improve the usability of the tool.

The type of software being studied is used to ensure optimum design performance and reduce the probability of a conductor failure on distribution lines. The algorithms for calculating tension, sag and clearance values take into account research into optimum design tensions. This ensures low bending forces due to conductor vibration. An algorithm for creep is designed based on common life expectancy of line conductors. The design methodology and algorithms were then put into software form as RSAT. Review and alterations include the changing of compiler, user interface, data storage mechanisms and the inclusion of options allowing the addition of new data.

Acknowledgements

I would like to thank the Eskom Industry Association Resource Centre management team for allowing me this opportunity to complete this research with their help and resources.

The Eskom Distribution lines committee for their input to the research throughout the research.

To my supervisor, Professor Barry Dwolatzky for your guidance and support in refining ideas and methodologies employed during the course of the research.

Lastly I would like to thank the users of RSAT for your support and feedback into this research.

TABLE OF CONTENTS

DECLARATION **2**

ABSTRACT **3**

ACKNOWLEDGEMENTS **4**

TABLE OF CONTENTS **5**

1. FORWARD **7**

2. INTRODUCTION **8**

3. BACKGROUND **10**

4. M.Sc. PAPER – G. STANFORD **12**

5. CONCLUSION **26**

APPENDIX A: LITERATURE SURVEY **30**

APPENDIX B: DESIGN SPECIFICATION **54**

APPENDIX C: IMPLEMENTATION **86**

APPENDIX D: TEST RESULTS **90**

APPENDIX E: COMPLETE SOURCE CODE **100**

TABLE OF FIGURES

Figure 1: RSAT limiting criteria page 38

Figure 2: Sag and tension chart 43

Figure 3: Stringing Chart 44

Figure 4: Clearance chart 45

Figure 5: Uplift verses Catenary 47

Figure 6: RSAT critical span graph 47

Figure 8: RSAT's old interface 70

Figure 9: Old conductor form 71

Figure 10: Old conductor data form 71

Figure 11: Old result form for sag and tension algorithm 72
Figure 12: RSAT's new interface..... 73
Figure 13: New interface with a conductor selected..... 74
Figure 14: New result form for selecting sag and tension algorithm..... 75

1. FORWARD

The intention of any research is that it be presented to peers and an educational institution with the idea of continuing knowledge in the industry. This dissertation will therefore not follow the conventional format for a dissertation. The main body of the document is written as a conference paper of the research completed. More detail concerning the research, specific algorithms and software design can be found in the appendixes of the document. The intention is that the paper be presented at a conference or in a technical journal subsequent to its completion. The paper details the philosophies and methodologies used in producing usable software. The completing of the research involved applying the philosophies and methodologies into designing and producing a new version of the RSAT software.

2. INTRODUCTION

Stringing of conductor for distribution lines is done by contractors or utility field staff. The exercise is often completed with the team leader (clerk of works) checking the tension.

The tension of the conductor is a parameter that is defined by its characteristic catenary (parabolic like) shape. The catenary shape is similar to a parabola and does not change much from span to span or between different conductors. Hence experienced field staff or contractors use their “experienced eye” to tell when the conductor is at this characteristic catenary. The conductor tension is thus checked by the approximate proportion of sag in the conductor. Hence the terminology commonly used is "sagging in the line". This, being common practice, has led to many a conductor being tensioned incorrectly as the “experienced eye” is not an accurate method of tensioning conductors.

Incorrectly tensioned lines present the following possible problems:

- overstressed or un-optimised selection of line conductor,
- overstressed or un-optimised selection of line equipment,
- overstressed or un-optimised selection of towers,
- possible vibration of conductor, ultimately causing damage to the conductor and line equipment,
- possible conductor clearance not within statutory limits.

The problems above may lead to premature line failure. Failure of the line presents the utility with a high cost in terms of replacement, cost of un-served energy and possibly consequential damage. The worst case scenario would be where the consequential damage is in the form of a person coming into contact with low hanging conductor or a broken conductor. Low hanging conductors often do not hang low enough to touch the ground. Since there is no earth fault for the protection to clear the conductor remains live. Contact incidents of this type present the electrical utility with a high risk.

These types of incidents should be avoided at all costs, even though the probability of such an incident occurring is low. Hence the importance of having accurate and easily available information with respect to line tension when designing or maintaining a line is of great importance.

Another factor presenting a possible design flaw is the factor that conductor stretches when under normal loading and abnormal loading. Stretch in the conductor is referred to as the “creep” of the conductor. If underestimated, the conductor creep could lead to a conductor sagging below safe limits and cause a low hanging conductor as described above.

Due to these high risks electrical utilities must ensure designs are done such that conductors are strung correctly. Because the equations for conductor tension, temperature variance and creep are complex, software is used to complete the algorithms quickly and accurately. Sag and tension software is commercially available but it is expensive and not freely available to Eskom designers and contractors.

Advantages of software being produced in-house allow specific user requirements to be met with a relatively low cost of development. The software can therefore be made available to Eskom designers and contractors freely.

The disadvantage of the in-house software is that it is not viable for bigger full line design packages. In house software is not necessarily produced by software engineers but by engineering staff with more background in line design. This may lead to an accurate piece of software but it can rate low on the usability scales.

Standards aim to reduce risk and promote cost saving. Therefore user friendliness is of utmost importance; else the standard may not gain wide acceptance and adoption. Therefore research into the usability of the RSAT software, which forms part of the Eskom standard, is deemed to be a high priority.

3. BACKGROUND

Power lines have been used to transport electrical energy since the late 1800's. The power line is basically made up of supporting structures, insulators, line hardware and conductors. Structures can be made of materials such as steel, wood or concrete and typically range from 4 metres to 28 metres in the distribution of electricity. Insulators are used to insulate the conductor from the structure. Line hardware is used to damp, clamp and suspend the conductor with respect to the structure. Conductors used for power lines range in size and material depending on the application.

It is the job of line designers to select structures, insulators, hardware and conductor suitable for each application. The selection is made based on the design values for power transmission capability required, physical terrain needed to be covered, annual prevailing weather conditions and the statutory requirements for the voltage rating. A tool is required to account for all the above conditions such that a mechanical withstand required to overcome the conditions can be found. This design tool is known as the "Sag and Tension Calculator". This is a calculator capable of setting up design conditions and criteria to be met, while calculating tensions and sags of distribution lines.

Sag And Tension (SAT) software packages are commercially available to design a power line from the structure to each small component that comprises the modern power line. Design packages available are large elaborate design packages that require great volumes of data, training and experience to create a simple conductor tension plan for small and medium power lines. Generally these packages cater for full design lines which are engineered with great precision by engineers with a wealth of experience and data. This kind of resource is not needed and is not cost effective for use with small and medium power lines.

SAT packages are available but are costly and do not fit the user requirements in all respects. A software package was needed to produce the information simply, to a specific Eskom define user requirement and cost effectively.

RSAT was developed to fulfil this requirement. The development of algorithms and a usable program within these parameters is the challenge taken up in this research.

RSAT has evolved from MS dos based software, to Windows based software with each revision upgrading the software to the latest operating system. In staying with this modus operandi, the software will be upgraded as part of the exercise. However this upgrade will include the recommendations of the usability study since usability has always been RSAT's Achilles heel. Standard techniques for determining usability will be explored and new techniques will be suggested where traditional techniques could not be used.

4. M.Sc. PAPER – G. STANFORD

Utilising a Usability Study to Review Sag and Tension Software

G. Stanford

School of Electrical and Information Engineering

University of the Witwatersrand

Abstract - Power lines have been used to transport electrical energy since the late 1800's. These are constructed using varied sizes of conductors, clamps, insulators and held above the ground by structures made from various materials. Designers of power lines take into account material characteristics, power transfer capability, statutory requirements and prevailing weather conditions to ensure optimum design of lines. Calculating sag and tension of a span is a complex process. The process involves the use of algorithms that require iterative techniques and decision making. Software running the algorithms quickly and accurately allows the designer to do the job with relative ease.

The software needs to be compact and include all the requirements of the utility for the calculations. The commercially available packages do not fulfil this requirement and so the software is produced in-house to fulfil all the utility requirements.

Producing software in-house can be problematic in that the product is not produced by a software engineering team. RSAT (Reticulation Sag and Tension software) is produced by a single electrical engineer with limited experience in the production of software. In early revisions of the software the emphasis was clearly in the accuracy of algorithms and solution output. It did not focus on user interaction making it cumbersome to use. This paper presents a usability study done to resolve this situation with the RSAT software produced in the Electrical Utility Eskom.

Keywords: Power Line Design, Sag, Tension, Reticulation, User Interface, Software Development, Conductor Design, Clearance, Installation, Stringing charts, and Usability.

1. INTRODUCTION

Software intended for the engineering environment is not always the most user friendly software. Possible reasons for this are:

- the designers of the software are often engineers in the field of the target software (not software engineers);
- the software tends to assume a certain level of understanding of the domain;
- the software is not programmed with the users needs or preferences in mind (the designers assume the needs of the user are the same as his perceived user requirement);
- the focus of the software is a numerical solution and no thought of how to get to the solution may have been given priority.

RSAT (Eskom's in-house "Reticulation Sag and Tension" software) users are made of field technicians, project engineers and line consultants. Considering the broad scope of users and the perceived simplicity of the software (engineering calculator), training for the software has never been considered. However the support required by users on the current version of RSAT would suggest that training is required. Training would be more in the use of the software and less in the field of how to apply results. To avoid this type of training a more intuitive user interface is required, an interface that "behaves exactly how the user thought it would"[1]

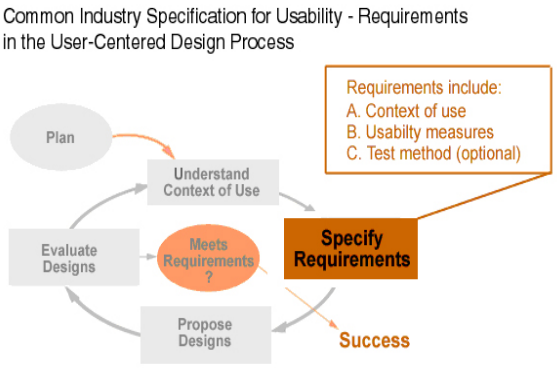
2. Software Usability requirement

Usability is: "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context use" [2]. The issue of usability of software has existed since the first software was developed. One only has to view the progression from no graphical operating interfaces like Microsoft DOS to the multi-tasking graphical user interfaces of the modern operating systems to see the effect and benefits of software usability. Even the methods of developing software have moved from the complexity of machine code, to the ease of the modern compiler which completed the more mundane tasks for the designer. However new operating systems and compilers with improved usability do not ensure that the software being used on the operating systems has been improved in usability at the same rate.

The frustration of poor software usability is a common factor for many packages used in business. This affects business productivity which in turn affects the profits of the business. It has become of such great importance that before software packages are purchased, some purchasers are now insisting that a usability study be completed.[4] The obvious advantage of completing a usability study is that the best suited software can be chosen for the business. Coupled with this, if the study is done during development, the software can be revised before implementation into a business, should changes be required. Improved usability of the software used in business will help to increase accuracy and efficiency of how business is done. If not the expenditure on such software is counterproductive.

3. Common industry specification

A common industry specification for usability [2] has been developed for the standardisation of specifying usability of software. It outlines the methodologies required to plan, understand context of use, specify requirements, propose a design and evaluate the design. There are three levels of compliance. Level 1 provides the information on context of use. Level 2 is to define performance measures, satisfaction measures and criteria for specific scenarios of use. Level 3 is the final procedure for testing the requirement. This user centred design process is summarised in the following diagram:



The redesign of the RSAT software was done taking these steps into account.

4. Level 1 - Context of use

Context of use is defined as the users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used. The context of use of RSAT can be defined as follows:

Users: Eskom designers, project engineers and design consultants.

Tasks: to get data on sag and tension, stringing charts and clearance calculations.

Equipment: consists of standard PC's linked to internet.

Physical and social environments in which a product is used: the software is generally used in a medium and low voltage line design office.

5. Level 2a - Performance and Satisfaction Measures

Performance and satisfaction are to be defined by the users. A common way to elicit information concerning user requirements for usability studies has been to setup a computer laboratory and observe the use of the software.

This had the disadvantage that it forced the user to use foreign software, on a foreign machine, in a foreign environment. Recently this has been replaced with the use of field testing to gain the information. Field testing puts the user in a normal atmosphere on the users own PC and therefore gives a realistic view of issues with the software.

With RSAT a standard path of communication has always been used to communicate with the users. The software is downloaded from a website where the designers contact details are listed. Most communication comes via telephone or by e-mail. This form of direct feedback and support has given the designer good feedback on the software and its usability. Over time an e-mail list of users has been put together. User elicitation ranged form the odd error in the software to full length installation and usage guidance. This allowed the developer establish contact with the users and scrutinise each use case. The user would be asked to complete the task causing the problem and describe each step of the task. This way the user's thought process and expectations of how things should work was logged.

Another way of communicating with the users was in a working group of users. The working group was held where specific issues could be raised with respect to the software interface and general design. Problems that were evident were the following:

1. The installation process was too complex
2. The methodology of selecting a conductor did not allow for comparison of conductors.

3. Conductor data display needed to be more easily accessed.
4. The sequential nature of selection of conductor, conductor data and the selection of algorithm to be run took too long. (Selection of a new conductor meant going through the whole process again.)
5. No allowance was given for the addition of new conductors thus limiting the user's capability with the software.
6. Limiting criteria could not be varied in line with prevailing conditions for the area.
7. Mistaken input, such as inserting characters where an integer should be, would cause the software to crash. A warning should be given as to the input errors.
8. Stringing charts could not be corrected for the first section once the next section was selected. Full editing of spans in any section should be allowed at any time, as well an update in the calculation.
9. Selected examples would cause the software to "hang" for no apparent reason.
10. The user interface gave no indication of the critical span.

Many of the criteria had already been discovered through the telephonic and e-mail elicitation.

Although the laboratory technique of eliciting usability has its flaws it was used on a small scale during beta testing and prior to final release of the software. It was used to confirm results gained from other methods of elicitation. Finally a Beta test copy of the software was released for field test and results from this testing were logged.

6. Level 2b - Criteria for specific scenarios of use

In design and testing it always a good practice to set goals and desirable outcomes for the process being undergone. In this research the criteria for good usable sag and tension programs were not available and so had to be drawn up at the beginning of the process. The criteria for a number of specific scenarios were setup. The scoring system used to grade each criteria had three possible results: a worst case, planned case and best case. [5] This simple grading system would present the designer with a quick means to see if the software achieved the goals set upfront.

The criteria are as follows:

1. The installation process

Worst case: no change to the installation file.
Planned case: a more intuitive installation process with directions given to all possible scenarios of outcome.
Best Case: The installation program must for all intensive purposes run itself with one possible installation file and one execution required on one file.

2. The methodology of selecting a conductor did not allow for comparison of conductors.

Worst case: no change.

Planned case: Comparison of two conductors can be made for one particular algorithm.

Best Case: A number of conductors can be compared for any algorithm in the same view.

3. Conductor data display needs to be more easily accessed.

Worst case: no change.

Planned case: Conductor selected should be displayed at all times.

Best Case: Conductor selected and its respective data should remain on screen at all times.

4. The sequential process of selection of conductor, conductor data and the selection of algorithm is too long.

Worst case: no change.

Planned case: a new conductor selection process that is non-sequential not involving different forms per task.

Best Case: One touch conductor selection form on screen selection without having to choose type, data, algorithm etc.

5. No allowance was given for addition of new conductors thus limiting the user's capability with the software.

Worst case: no change.

Planned case: A customised conductor function.

Best Case: A process which allows conductors to be added to a type, or new types to be added as well as new conductors, diagrams of conductor configuration and creep constant.

6. Limiting criteria could not be varied in line with prevailing conditions for the area, conductor or structure technology.

Worst case: no change.

Planned case: a facility must be added which allows the user to change the limiting criteria.

Best Case: The facility allows limits to be changed per task and allows the criteria unit to be altered. (For example C-Value can be changed to a percentage of ultimate tensile strength or to a tension.)

7. Incorrect input (such as characters where an integer is expected) should not cause the software to crash.

Worst case: no change.

Planned case: the software does not crash when incorrect input is inserted.

Best Case: the software does not crash when incorrect input is inserted and informs the user of the error with possible remedies.

8. Stringing charts could not be corrected for the first section once the next section was selected.

Worst case: no change.

Planned case: allow the user to reset the software back a section to correct errors.

Best Case: allow correction of errors without resetting any sections and allow the calculation to be redone for all sections at once.

9. Selected examples would cause the software to "hang" for no apparent reason.

Worst case: no change.

Planned case: the software is resolved for these examples.

Best Case: the reason for these problems is established.

10. The user interface has no indication of the critical span (the optimum span where tension is not required to be reduced in order to fulfil the limiting conditions).

Worst case: no change.

Planned case: a critical span value for the conductor selected is given.

Best Case: the critical span is calculated and a graph of everyday tension verses span length shows the critical span where the graph shows the conductor to have maximum tension.

Once all the upgrades were made, general functionality speed needed to be checked between the new and old software. The speed in completing tasks by a user would indicate if the usability had improved. The following test cases were added to ten tasks above to indicate the speeds of operation:

11. Select a Mink conductor, run a sat chart, set the temperature limits to 0 to 30 with an increment of 5 degrees and the span length to 45 to 160 with an increment of 10 metre. This would test the speed of use and of the computation of the algorithms.

12. Compare the results of task 11 with that for the Hare with the same criteria.

13. Count the number of mouse clicks to get from start to finish.

14. Sum the distance required for the mouse to move to complete task 11.

15. The time needed to compare Squirrel and Acacia conductors for a string chart having:
- Section 1 with spans 100, 150,125 metre and
- Section 2 with spans 230, 150, 45 and 190 metre.

16. The times need to compare Pine and 35 for a clearance chart with a 12 metre pole, a 13 metre pole, a test object placed half way down the 200 metre equivalent span of height 5 metre and at a temperature of 50 degrees.

17. The speeds of accessing the help for clearance charts.

7. Detail of changes

The changes below were attained through user forums, old revision feedback, laboratory testing and comments on subsequent prototypes produced:

- The installation program was simplified with a more intuitive interface.
- Stringing charts : added the ability to edit old section data and recalculate the whole lines stringing tensions.
- Clearance calculator: included the tension at the structures with horizontal and vertical components included.
- The user interface was a simplified interface. The conductor selected and respective conductor data being visible at all times.
- Selection of a new conductor was upgraded to allow selection without clearing old limits or criteria setup for the previously selected conductor.
- A graphical critical span indication was added such that conductor selection can be optimized.
- An option is provided whereby conductor data files can be updated with new conductor types, conductor diagrams, conductor data and conductor names.
- The default “Every Day Tension” limit was moved to “Final Stringing Tension” and is at a C value of 1425 for ACSR and AAAC so as to avoid the unnecessary use of vibration dampers.
- The default limits and ranges for the software have upgraded and can now be customised from a limits page.
- The use of setup password protection was removed from the installation program.

In line with the changes as set out above the forms based user interface was traded for a tab style user interface.

This allows the user to change between algorithms, help, limits and graphs without reselecting a conductor. The conductor selection is done by means of two list boxes – one for the type of conductor and one for the specific conductor per type. These boxes remain visible on the main form at all times such that the conductor can be changed at any time and the algorithm results can be compared. In initial Beta test version all algorithms were updated on all tabs such that if a tab was selected the updated results would be reflected. This proved to slow the software down since this meant completing calculations for all algorithms every time a conductor was selected. Therefore this was changed to only complete the algorithm for the tab being viewed. On the selection of a new tab, the tab algorithm for the respective tab was run and the tab updated before displaying the tab. The changes resulting from the process were good and process times were a fraction of previous versions. To see if the changes improved the software, it needed to be tested in line with the criteria setup earlier.

8. Testing

Testing took place in a controlled and uncontrolled exercise.

8.1 Controlled testing

The controlled testing was done with the design criteria above being checked between new and old software. The results are as follows:

1. The installation process

The Planned case was achieved in that the installation of the .Net framework will be required for the software to run. This was incorporated into the software but the installation needed to run again to install RSAT.

2. Conductor selecting.

Planned Case was achieved as a number of conductors can be compared for any algorithm. The conductor selection windows are also close to the tab selection area and so that very little mouse movement is required to change the conductor and algorithm.

3. Conductor data display.

Best Case: Conductor data remains on screen at all times.

4. Sequential selection process.

Best case: one operation is required to select a conductor without opening new windows sequentially. The type of conductor need not be selected first.

5. Addition of new conductors.

Best Case: the new process allows conductors to be added to an existing type or a new type can be added.

6. Limiting criteria.

Best Case: The facility allows limits to be changed per task and the defaults limits to be revised. It also allows the user to define the units for tension from C-value, percentage of ultimate tensile strength or a tension.

7. Incorrect input causing software to crash.

Best Case: the software was upgraded such that it does not crash when characters are inserted instead of numbers. The software uses an error message system to inform the user of the possible error and possible recourse.

8. String chart correction.

Best Case: the software upgrade allows correction of errors in previous sections without resetting any sections. It also recalculates the “all” section each time "calculate" is pressed.

9. Selected examples hang.

Planned case: the software resolved all known examples that coursed previous versions of the software to hang.

10. The user interface gave no indication of the critical span.

Best Case: A critical span graph was added with “Every Day Tension” verses span length showing the critical span.

The speed test tasks were done by candidates of a high level of competence as well as candidates of first time user competence. These were done in a controlled environment.

A controlled environment allowed for the observation of the user and his natural expectation of how the software should operate. Viewing of the less competent user provided good feedback on how to improve the intuitiveness of the software.

The results for both high and low competent users will be reported below except for task 14. In task 14, the shortest distance for the mouse to travel is given as the result, since measuring the distance travelled of the first time user's mouse pointer could not be done. It is assumed that a more usable piece of software should render times that converge since the difference between users should only be engineering competence and not software knowledge. Hence the results will be judged on this time to complete. The results are as follows:

11. Production of a Mink conductor SAT chart with certain limits.

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	43 seconds	65 seconds
5	37 seconds	65 seconds

An average improvement in time to complete was recorded as 7 %. For a new user of lower competence there was no improvement. This is indicative of the general misunderstanding of software, engineering principles, and of the designers misunderstanding of what the user expects the procedure is for completing a task. However, the object of the research was to rule out this factor.

Therefore, it can be seen that there is still room for improvement on the usability and self explanatory aspects of the software. Viewing the users helped with ideas on how to make the software more intuitive.

12. Compare the sag and tension results of Mink with Hare

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	8 seconds	8 seconds
5	3 seconds	8 seconds

The time improvement is 63 % for a competent user but again no change for a new user. More importantly it is easier to do a comparison with RSAT 5 since the sag and tension data for Mink was only removed to replace with Hare. For RSAT 4.2 the data was removed such that conductor selection could take place, then a conductor data sheet is displayed and then only is the sag and tension data displayed. Therefore a print out or a good memory would be required.

13. Number of mouse clicks required to complete the task of producing sag and tension charts:

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	10 clicks	18 clicks
5	5 clicks	11 clicks

The number of clicks required to produce a sag and tension chart improved by 50 % for the competent user. First time users showed an improvement of only 39 %. However if looked at in terms of actual numbers the improvement for lower competence users was 7 clicks while for higher competence users it was only 5 clicks. These improvements reflect an improvement in mechanical efficiency.

14. Sum the distance for the mouse to move to complete case 1 for both versions of software. (on screen assuming short cuts and tab keys are used where possible)

Result:

RSAT Version	Competence Higher/Lower
4.2	750 mm
5	130 mm

Improvement = 83 %. Competence is irrelevant since the distance between controls is a constant. However a general comment would be that for lower competence user's movements of the mouse would be further than that of a higher competence user.

15. Times for comparing Stringing charts

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	90 seconds	230 seconds
5	50 seconds	97 seconds

The improvements for high and low competency are 45 % and 57 % respectively.

The results were taken for the first time the input was correctly input. In order to get results for a second conductor, RSAT 4.2, needs to be restarted as the stringing chart does not reset. To inexperienced users this may not be obvious and the time taken substantially increased.

16. Compare the times needed to produce results for Pine conductor and 35 conductor, for a clearance chart with:

- a 12 metre pole,
- a 13 metre pole,
- a 5 metre test object,
- the object placed 100 metres from pole A,
- a 200 metre equivalent span and
- at a temperature of 50 degrees.

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	50 seconds	114 seconds
5	30 seconds	86 seconds

The resulting improvements in time were 40 % and 24 % for higher and lower competencies respectively. This test showed a problem with the interface in that when a new conductor is selected the results were not updated until an attribute was updated for the clearance calculator other than the conductor. This problem was resolved and further improved times by up to ten seconds.

17. Speeds of accessing help

Result:

RSAT Version	Competence	
	Higher	Lower
4.2	5 seconds	9
5	4 seconds	9

Improvement = 10 %

This result shows that there were very few changes made to the help system as it has never been the cause for comment or concern. Hence the same type of help system is run on both pieces of software. The only improvement that may have been gained in time would be due to the improved access to the help function selection.

Overall the improvement for competent users was better than for first time users in laboratory testing. The laboratory did prove useful in exposing errors in assumptions as to what the changes needed to be in ordered to improve the first time user's experience of the software.
The software was improved in areas allowing users to reset calculation tabs. Not allowing the user to reset a calculation tab, would mean when new data or a new chart wanted to be entered, the software would need to be restarted. Then important function buttons, fonts and tabs were increased in size to improve their visibility to users. A "Suggested Action" help window to guide users to next possible action was added. This window prompted the user as to possible steps to take.

The algorithms were updated to do calculations automatically after input was completed. It was found that some inputs, if completed incorrectly, could send the software into an extremely long calculation that was not required. An example of this is editing a span length from 300 metre to 100 metre but inserting the 100 metre before deleting the 300 metre may lead to the software calculating up to a span of 100300 metre.

8.2 Uncontrolled testing

The uncontrolled test was completed in the form of a Beta release being made available to users via the Distribution Technology website. The Beta test version was available for eight months with user feedback coming telephonically and via e-mail. The results from the Beta testing showed the new software had achieved the goal of becoming more user-friendly. There were two major feedback items related to usability of the software. The first was that the error/ exception handling could be improved. The suggestion given was to guide the user away from making mistakes when inserting data, by having a message facility that gave possible corrective actions required at the point of error.
The second comment related to the size of the forms, menus, conductor boxes, chart windows and all the items presented for the user to see. Fonts and selection options had all been increased in size after the user feedback in the laboratory testing. Further prompting lead to the software being upgraded to resize based on different resolution settings of the users' screens. The comment was that graphics were too small. This occurred at high resolutions.

The software was revised to resize the software's desk top to the size of the main application form. All other comments were related to small errors in the software that were overlooked when designing the Beta test version.

9. Conclusions

Results from the testing showed that of the ten design criteria set, six criteria were met with best case results while the remaining four met the planned result. In the speed testing an overall improvement of 42 % on the old software was recorded. The results can be attributed to redesign criteria being user-based and to the prototype loop employed to setup design requirements. Improvements are still being made to software based on user feedback of the published revision of the software. What the testing did not emphasise is the way comparison tasks have become easier to complete without restarting the software to complete the same task with another conductor. The software now allows for recovery after an error in the input is made, thus making for a much more usable piece of software. The research on usability recommended direct interaction with users in the form of a user laboratory or direct interviews. However these results show that interaction with users via a working group and representatives, telephone task analysis and e-mail does serve as a good addition to a regular usability study. Users were observed in the final testing to see the usability issues that were not perceived in the initial working group, telephone and e-mail communications. The end product from the process used is deemed to be a worthy upgrade and successor to RSAT 4.2.

10. References

[1] Joel Spolsky, *UI for Programmers,*2001, http://www.joelonsoftware.com/uibook/fog0000000249.html, last accessed on 27 November 2006

[2] Nigel Bevan, *Common Industry Specification for Usability Requirements,* Draft Version 0.85, 18 March 2006.

[3] Michael Good, *Software Usability Engineering*, Digital Technical Journal No 6 February 1988,125-133, Compaq Computer Corporation, 1988.

[4] Patrick Thibodeau, *Users begin to demand software usability tests*, http://www.computerworld.com/software topics/software/story/0,10801,76154,00.html, Last accessed 24 November 06

[5] Michael Good, *Software Usability Engineering*, Digital Technical Journal No 6 February 1988,125-133, Compaq Computer Corporation, 1988.

11. Bibliography

[1] Hans Van Vliet, *Software Engineering, Principles and Practice*, Wiley, Second edition 2000

[2] Dr. Sharon Laskowski, *Increase the visibility of software usability*, http://zing.ncsl.nist.gov/iusr/index.html, Last accessed 24 November 06

[3] Joel Spolsky, *Usability Test with Morae,* http://www.joelsoftware.com/articles/UsabilityTestingwithMorae.html, Last accessed 14 November 06.

[4] SANS 0280 Edition 1.1, *Overhead power lines for conditions prevailing in South Africa,* 2001

[5 Ian Stevenson, *.Net core architecture,* Elektron, May 2004

[6] CIGRÉ. *Sag-Tension Calculation Methods for Overhead Lines Draft 9 Revision 3.* CIGRÉ Work Group B2.12, 7 August 2005.

[7] Dale Douglass, *Sag-Tension Calculations,* IEEE TP&C Tutorial, 2005

[8] ANSI/INCITS 354-2001, *The common Industry Format for Usability Test Reports,* December 2001.

[9] Jacques Hamian and Yair Berenstein, *Sag-Tension Calculations: Refinements and Enhancements Made by Pondera*, 2005

5. CONCLUSION

The RSAT software has been redesigned with the aim of producing an application that is user friendly, quicker to install and run, with a more intuitive interface than the previous version of RSAT. The new software was written in Microsoft .Net C# for greater compatibility with platforms such as operating systems used on hand held devices. The software will become available for hand held type of devices in subsequent versions.

Results from the testing showed that of the ten design criteria set, seven criteria were met with results that were of a level that was predicted to be the best outcome possible. The remaining three criteria met the planned result expected before redesigning the software. The overall result with respect to specific criteria setup before redesigning the software shows the software upgrade has achieved in each criteria.

In speed testing an overall improvement of 45 % on the old software was recorded. The results show the criteria setup for the redesign were user centred and that the prototype methodology used to refine the software did improve the software. An area of concern to be reviewed is the improvement of the software usage to be more intuitive to a first time user. The design and feedback process have suggested possible ways to improve this and these steps have been employed in the new software. However continued research into this area is required.

The testing criteria did not cover the “ease of use”, since this was seen as a possible user preference issue. However, the comments that were received on the software, were that tasks have become easier to complete without restarting and the software stability has improved. The ability of the software to allow recovery after an error input, has made for a much more usable piece of software. The implementation of error messages guides the user to the error and the correct type of input for the field being specified.

The research on usability recommended direct interaction with users in the form of a user laboratory or direct interviews. This type of interaction was completed and was found to be useful. However, the results gained from interaction with users via a working group and representatives, telephone task analysis and e-mail was vital in the initial stages of design. The iterative process of reviewing the user's needs also proved to refine the user requirement. With this in mind, the research has shown that in order to complete a usability study on software, the scope of testing and sources of information must be as wide as possible. Broadening the scope for feedback allows the designer to interact with users at a variety of competence levels and with respect to many aspects of the software which may otherwise not be covered in simulated laboratory type testing.

User interaction, whether direct or indirect, gives the designer a different perspective on the requirement for algorithms, interfaces and functional aspects of a software design package. Different forms of user interaction may be needed based on the domain the software is in and the type of user being targeted. User interaction is vital and should not be overlooked because of the lack of time and facilities or, because of perceived knowledge of the domain.

Improvement of the usability software is not a finite process. It requires the designer to listen carefully to the user. To improve the usability continually, means using an iterative technique of sourcing user feedback. It also requires the designer to be creative in dealing with the user in terms of setting up testing laboratories, feedback forums and help communication channels. The redesign of RSAT has improved the software significantly but more importantly it has set in place lines of communication and a culture of testing and feedback forums. The software still requires further revision to become a more usable tool. However, continual improvement of software should always be a goal of the software designer. Software is never complete – it is on a journey towards the ideal of perfection.

REFERENCES

[1] CIGRÉ. *Safe design Tension with respect to aeolian vibrations, single unprotected conductors.* CIGRÉ Study Committee 22 Work Group 11 Task force 4 Conductor tension design guide: final report 2nd draft, August 1997.

[2] John Berry and Patrick Wainwright. *Foundation Mathematics For Engineers*, The Newton-Raphson iterative formula (Pg. 400), Macmillan 1991

[3] John McCombe and F.R. Haigh. *Overhead Line Practice,* Sag and Tension Calculations (pg. 224), 1966 Macdonald & Co. (publishers) Ltd.

[4] Linesoft USA technical manual, *PLSCADD – version 5,* 2001.

[5] IEC 61089, *Round wire concentric lay overhead electrical stranded conductors.*

[6] Jacques Hamian and Yair Berenstein, *Sag-Tension Calculations: Refinements and Enhancements Made by Pondera,* 2005

[7] Mickey Williams, *Microsoft Visual C#,* Microsoft Press, 2002

[8] Hans Van Vliet, *Software Engineering, Principles and Practice,* Wiley, Second edition 2000

[9] A Luchmaya, *A GIS electrification planning tool that considers terrain information*, A dissertation submitted to the School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, South Africa, 2002

[10] Patrick Thibodeau, *Users begin to demand software usability tests,* http://www.computerworld.com/software topics/software/story/0,10801,76154,00.html, Last accessed 24 November 06

[11] Dr. Sharon Laskowski, *Increase the visibility of software usability*, http://zing.ncsl.nist.gov/iusr/index.html, Last accessed 24 November 06

[12] Michael Good, *Software Usability Engineering*, Digital Technical Journal No 6 February 1988,125-133, Compaq Computer Corporation, 1988.

[13] Joel Spolsky, *Usability Test with Morae,* http://www.joelsoftware.com/articles/UsabilityTestingwithMorae.html, Last accessed 14 November 06.

[14] Nigel Bevan, *Common Industry Specification for Usability Requirements,* Draft Version 0.85, 18 March 2006.

[15] SANS 0280 Edition 1.1, *Overhead power lines for conditions prevailing in South Africa*, 2001

[16] Ian Stevenson, *.Net core architecture,* Elektron, May 2004

[17] CIGRÉ. *Sag-Tension Calculation Methods for Overhead Lines Draft 9 Revision 3.* CIGRÉ Work Group B2.12, 7 August 2005.

[18] Joel Spolsky, *UI for Programmers,* 2001, http://www.joelonsoftware.com/uibook/fog0000000249.html, last accessed on 27 November 2006

[19] Dale Douglass, *Sag-Tension Calculations,* IEEE TP&C Tutorial, 2005

[20] ANSI/INCITS 354-2001, *The common Industry Format for Usability Test Reports*, December 2001.

[21] Occupation Heath and Safety Act, Act 85, 1993

[22] D R Theron, *SCSASABE7 General information and requirements for overhead lines up to 33 kV with conductors up to Hare/Oak*, Eskom Medium Voltage Distribution Standard, 2001

APPENDIX A

LITERATURE SURVEY

To launch a research project, a literature survey needs to be undertaken to establish the current knowledge in the research area . Eskom holds a great wealth of expertise on line design within its engineering staff as well as in a well equipped and up to date library. These resources were explored concerning latest knowledge on algorithms for sag and tension software. The designer, having designed the Distribution sag and tension software for a decade was already well averse with terms, definitions and standard practices in the field. Still a large volume of information was reviewed with respect to line design and, in particular, sags and tension of electrical medium voltage lines.

Usability of software was not as easy to find information on, within the resources available in Eskom and its library. Therefore, other libraries and the internet were a good source of information for this section of the literature survey. Before getting into the detail of the survey, some definitions need to be clarified in order to make the task possible.

Definitions

C value: Catenary value which is a constant defined per conductor
Creep: the time-dependent increase in strain between two points under a constant stress.
Critical span: the optimum span where tension is not required to be reduced in order to fulfil the limiting conditions.
Every Day Tensions (EDT): The average daily tension that a conductor will experience over the conductor lifetime.
LV: Low Voltage
Line design software: large elaborate software used to design complete lines including line gradient, obstructions, geographic information and caters for all possible options and customisations.

MV: Medium Voltage

RSAT: Reticulation Sag and Tension, the name of the software being used at Eskom to resolve the values for sag and tension of distribution conductors.

SAT software: small compact software specifically designed to give the user sag and tension data, clearance calculations and stringing charts with the emphasis on simplicity.

Software usability: the ease with which end users can be trained on and operate the software product.

Tension section: a section of line between two strain poles or dead ends, with suspension poles in between.

UTS: Ultimate Tensile Strength – usually refers to the strength of the conductor.

SAG AND TENSION

Design of transmission and distribution lines is a well established topic with extensive literary resources available. Added to this, are the utility's adopted philosophies and the statutory requirements of the Occupation Heath and Safety Act [21]. Research was focused on documentation that contained specific detail on the sag and tension of the transmission and distribution lines.

The mechanical performance of overhead power lines is dependent on the correct installation of conductor. Vital to the installation of the conductor is clamping the conductor at the correct line tension. Maximum allowable conductor tension is limited by the strengths of structures, hardware, insulators and the conductor. However, tension can not be chosen based on short term mechanical criteria only. This would not take into account long term fatigue, conductor vibration or clearance values. Optimum design takes these aspects into account.

Conductor vibration occurs when wind blows across the conductor exciting natural harmonics. To combat conductor vibration the conductor tension can be reduced such that the natural harmonics of the conductor are not excited or are damped by the inter-strand friction. Inter-strand friction increases at lower tensions improving the self damping characteristic of the conductor. The natural harmonic of the conductor is determined by the span length, conductor stranding, the conductor diameter and the conductor weight. Should all mechanical design criteria not be adhere to for allowable conductor tensions, the result can be structure, hardware, insulator or conductor damage. If the conditions are severe enough the damage may lead to failure of the respective component. Failure of any of the components can put the public in danger of fire hazards and/or electrocution. These consequences present a high risk to a utility and come with the high cost of consequential damages.

Another method of combating conductor vibration is to apply vibration dampers to the line. The cost of applying vibration dampers needs to be offset against the saving from getting longer spans with higher tensions and hence less structures, insulators and hardware. These decisions can be made by running sag and tension software such that the two scenarios can be designed and respective quantities for span lengths can be calculated. The option of not using vibration dampers may not be available in certain circumstances. An example is when a line needs to cross an obstacle at a particular clearance and has a fixed span length requirement. SAT packages can help to optimise such spans and ensure safe designs. A disadvantage of applying vibration dampers is that, if incorrectly applied, vibration dampers can enhance the natural harmonic frequency and cause more damage than if they were not applied.

RSAT currently gives the horizontal component (at middle span) of tension of the conductor as the tension on the clearance calculator. This is usually a good estimate because at optimum span length on flat land, the vertical component of tension adds little to the tension at the structure. However, for long or angled spans (over or down a valley), where optimum span length can not be used, the conductor weight increases the vertical tension substantially at structure. Since the correct way to check tensions is at the structure with a dynamometer, it is important to know what the total tension (horizontal plus vertical components of tension) are at the structure. The vibration damper type and placement is also dependent on the maximum tension in the conductor. Therefore the tower tension is a necessity for good design of lines. Hence, a necessary upgrade to the current for of RSAT is to give the tension at the structure (including horizontal and vertical components of tension).

Over time conductors under line tension permanently elongate or creep as a consequence of the long term mechanical stress. The creep of a conductor is affected by the average ambient temperature of the conductor, extreme conditions (such as short circuits, ice loading) and winds during the conductor’s life. Predictions of creep are required to ensure that the sag of the conductor is not increased to a point where critical clearances of the conductor to ground or objects under the line are compromised.

If the creep can be predicted accurately, the stringing tension can increased to compensate for the creep elongation. Critical clearances (below which the conductor must not sag) are stipulated in the Occupation Health and Safety Act [21] and in the Eskom Distribution Medium Voltage Standard [22].

Methodologies for predicting creep (conductor elongation) are prevalent in the mechanical engineering environment. However, the study of creep in a multi-stranded, non-homogeneous material, such as ACSR (Aluminium Conductor Steel Reinforced), is not common. Of the many sources of mechanical fatigue information that were studied, only a few dealt with the specific creep found in conductor. Research sources for creep were PLS-CADD [4], CIGRÉ Work Group B2.12 [17], Cigré Sc 22 WG 11 TF4 August 1997 [1], McCombe and Haigh 1966 [3] and Stress- Strain testing completed on South African manufactured conductors at CSIR to IEC 61089 [5].

Conductor length also varies with respect to temperature change. A change in conductor length, from temperature, in turn produces a change in conductor tension. This again requires an algorithm to ensure this change is catered for. Therefore the temperature the conductor is strung at must be catered for such that when the temperature changes the conductor is at the correct tension.

The temperature elongation algorithm being used at present is the standard algorithm used in texts for predicting line sag and tensions. These algorithms and conductor criteria have been scrutinised to ensure that all results give accurate results when compared with line design packages such as PLSCADD.

PLSCADD is a large line design package used by Eskom for complete line designs. It is not intended to produce sag and tension data for small projects. Producing sag and tension charts in PLSCADD is cumbersome and can not be done without extensive training and a PLS-CADD license. However the results from PLSCADD are a good representation of reality. Hence the necessity to align RSAT results with PLSCADD.

RSAT is a simple software package that:

- produces sag and tension charts easily,
- without extensive training and
- without a license since the package is written in-house by Eskom Distribution Technology.

The intension for the software is to make it easy to use without any boundaries to use. Use of the software promotes the current standard options of the utility and the use of standard design practices. Creating boundaries to use may mean lines will be designed to other standards or no standard at all. This presents a high risk to the utility and therefore producing the software at no cost to users with no boundaries to use is of utmost importance.

SAG AND TENSION EQUATIONS

RSAT produces sag and tension information for reticulation lines. The equations required to produce sag and tension data are commonly referred to as the catenary equations. The shape of a freely hanging conductor is a catenary shape which is very similar to the shape defined by parabolic equations. For spans typical of reticulation lines (a span with a ratio of span to sag of greater than 20:1, with flat underlying land, and homogenous conductor) the parabolic formulae are accurate to about 0.5%. Since this application is intended to be a simple reference for the reticulation designer the parabolic equations are used for Rsat calculations. The two main equations ((1) and (2)) are used for sag and tension values respectively:

$$D = \frac{WL^2}{8T} \tag{1}$$

$$T_2^3 + T_2^2\left[AE\left(\frac{W_1^2 L^2}{24T_1^2} + \alpha(\theta_2 - \theta_1)\right) - T_1\right] - \frac{AEW_2^2 L^2}{24} = 0 \tag{2}$$

D = Sag (Meters)

T = Tension (Newton)

A = Area of the conductor cross-section (Square Meters)

E = Young's modulus (Pascal)

α =Temperature coefficient-efficient of linear expansion (Per Degree Celsius)

θ = Temperature (Degrees)

W = weight of conductor (Newton per Meter)

L = Span length (Meters)

As can be seen above there are conductor attributes which are required to calculate the equations. These conductor attributes are loaded into RSAT from a conductor data file when a conductor is chosen. The types of conductors are loaded onto type selection box on initialisation of the program.

The attributes stored in the data file are as follows:

- Conductor type
- Conductor name
- C value
- Area
- Temperature coefficient
- Initial modulus
- Final modulus
- Mass
- Ultimate Tensile strength
- Diameter
- Creep constant
- Path for bitmap

When selecting a conductor in the program these characteristics are downloaded into the data area of the page. A path to bitmaps allows the program the access to a bitmap of the conductor stranding which is viewed when a conductor is selected.

The starting point for all algorithms is the every day conditions and statutory limiting criteria (as listed in the occupation health and safety act) for South Africa. An every day average temperature for South Africa is 15 degrees Celsius. This is the average temperature used for average calculations for creep, vibration criteria and a comparison point for all limit calculations. The default everyday tension point(for most tensioning software) is set on final tensions at 15 degrees and will be kept as such for RSAT. The everyday tension point is set-up by using the tension calculated for the C value.

The C value is the catenary constant and is defined to be the every day tension divided by the conductor weight. The constant holds for the full range of ACSR and AAAC conductors. The C value for ACSR and AAAC of 1425 was taken from a *CIGRÉ work group report on Aeolian vibrations*. [1] This C value then sets up the first limiting condition at everyday tension.

The other "worst case" limiting conditions that are used for calculations are:

2. on initial at -5 degrees with a 700 Pascal wind tension must be limited to less than 50 % of the UTS of the conductor

3. on final at -5 degrees with a 700 Pascal wind tension must be limited to less than 40 % of the UTS of the conductor.

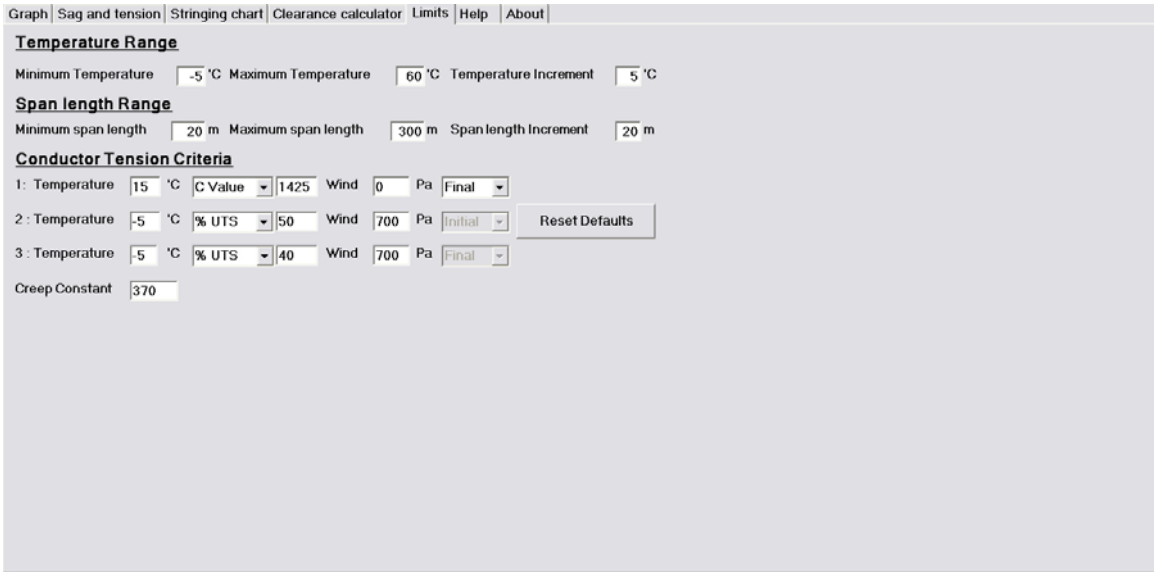


Figure 1: RSAT limiting criteria page

The limits page of RSAT presents the three tension limits, the span limits and temperature limits for the sag and tension charts.

Before starting to calculate tensions, the ruling limiting condition must be determined. To find out which condition is the ruling limiting condition all limits must be brought to a common base. The common base used is the everyday tension point: 15 degrees with no wind and at final tension. Limiting conditions 2 and 3 are converted to this everyday tension point. Conversion involves taking the limiting tensions and accounting for the conductor length variations from the temperature and creep to get to the everyday tension point. If one of the conditions converts to 15 °C with a lower tension than that given by the C value; then this condition is the limiting condition for the calculation.

Converting the final worst case scenario 3 to 15 °C is done using equation (2). The equation has two subscripts (1& 2). If the known worst case initial conditions are placed in to the constants with subscripts of one and all known every day constants into that with a subscript of two then all that is left is a polynomial in T_2. Since the polynomial is of the form $AT^3_2 + BT^2_2 + C = 0$, it can not be solved by regular methods. This requires an iterative method that converges to the solution. For the iterations, Newton-Raphson [2] iterative formula is used to get a solution. The formula is as follows:

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \qquad (3)$$

The extrapolation is done and then it can be discovered which of the two final conditions 1 & 3 the ruling condition could be. However, it is not possible to compare the initial limiting condition 2 with these final conditions without a creep equation to convert the initial tension to a final tension. Limiting condition 2 can be converted to 15 degrees with no wind initial as shown above for limiting condition 3. The conversion from initial to final to compare on the same base is discussed below.

CONDUCTOR CREEP

Creep is defined as the time-dependent increase in strain between two points under a constant stress. In other words it is the elongation of a material which is under constant load. Conductors, due to their twisted construction, are loosely packed when they arrive on site to be strung. This loosening is due to the transport and handling vibrations and movement. The constant load the conductor is placed under when it is then strung gradually pulls the conductor tight. This form of creep elongation is quick and will be pulled out of the conductor in a short period after stringing.

The conductor materials that are used elongate over time under the constant force. This is a common feature of materials under constant force and is a long term process. These elongations to the conductor over time cause the conductor to sag below the initial clearance the conductor was sagged to. This can cause tremendous damage to lines if conductors in the subsequent phases also elongated swing in the wind and clash with each other. Another risk presented by the elongation of the conductors is that elongation can be to such an extent that the conductors sag down onto or too close to an object. A remedy may be re-tension the lines regularly. However, this is a labour intensive costly solution to countering conductor creep. Alternatively the designer can choose to tension the conductor to higher tensions that result in conductor sag above the critical clearance. Then when conductor elongation does occur the clearance will not be compromised. However, pulling the conductor to higher tensions is only acceptable if these tensions meet the limiting criteria. The limits and creep work against each other. This is the main focus of the designer and determines if a conductor is suitable for certain spans. The margin required needs to calculated by calculating the conductor creep under the conditions the conductor is subjected and converting this into a sag value. Then the critical sagged conductor length can be converted back into some initial conductor length for a time span and conductor relaxation. The designer needs initial and final value for sag. For this reason a sag including the creep is calculated in sag and tension charts and is called final sag. The creep used to work out final sag is calculated to match that produced by the commonly used PLSCADD line design package.

The creep was derived because equations given in published literature do not give the values for permanent elongation that are produced by PLSCADD.

The creep equation derived to match PLSCADD is:

$$\Delta S_c = \frac{kL^{1.8}}{200}\left(\frac{T_2}{T_{20}}\right)^2 \qquad (4)$$

$\boldsymbol{S_c}$ = Conductor length

$\boldsymbol{k}$ = Creep Constant

$\boldsymbol{L}$ = Span Length

$\boldsymbol{T_2}$ = Final Tension

$\boldsymbol{T_{20}}$ = Tension at 20°C (Creep constant was for a 20°C tension)

The equation for the length of the conductor required in the span is:

$$S = L + \frac{W^2 L^3}{24T^2} \qquad (5)$$

$\boldsymbol{S}$ = True Conductor Length

$\boldsymbol{L}$ = Span Length

$\boldsymbol{W}$ = Weight per unit length of conductor

$\boldsymbol{T}$ = Tension of conductor

The change in length of conductor is:

$$\Delta S_c = S_2 - S_1 \quad (6)$$

S_1 = Length of conductor at stringing tension

S_2 = Length of conductor after creep with reduced tension

Making the relevant substitutions of equations (4) and (5) into equation (6) an equation in T_1 and T_2 is derived. This equation has powers of 2 and 4 and therefore can not be solved with regular techniques. Once again the Newton-Raphson method (equation (3)) is used to resolve this equation from final tension to initial tension or from initial tension to final tension.

The capability of converting to final values (including elongation) for twenty years allows the program to impose a limiting condition of 50 % of UTS on worst case initial tension conditions. Conversion of this limit to final allows the overall ruling condition to be discovered. Based on the final limit being found the table of initial and final values for tension from -5°C with wind to 60°C can be calculated using equations (2), (3) and (6). The relating sags can then be calculated from these tensions using equation (1).

Sag and tension charts are not ideal when considering the distribution line is commonly comprised of section lengths under constant tension that is not dependent on the specific length of the span but on an equivalent span. For this instance RSAT can produce stringing charts that use equivalent span as a ruling factor for producing sags.

Graph | Sag and tension | Stringing chart | Clearance calculator | Limits | Help | About

Sag and Tension Chart

Span (m)	Temperature (°C)	Stringing tension (N	Stringing sag (m)	Final tension (N)	Final sag (m)
EDT =	1191				
Limit = 1	I = -5 F = -5	2391.64 (700 PA)	0.44	2537.03 (700 PA)	0.42
100	-5	1760.87	0.6	1737.22	0.61
100	0	1649.35	0.64	1585.29	0.66
100	5	1541.08	0.68	1442.57	0.73
100	10	1436.69	0.73	1310.69	0.8
100	15	1336.83	0.79	*1191.01	0.88
100	20	1242.19	0.85	1084.31	0.97
100	25	1153.42	0.91	990.68	1.06
100	30	1071.05	0.98	909.47	1.15
100	35	995.44	1.05	839.55	1.25
100	40	926.73	1.13	779.49	1.34
100	45	864.82	1.21	727.87	1.44
100	50	809.4	1.3	683.32	1.53
100	55	760	1.38	644.68	1.63
100	60	716.07	1.46	610.95	1.71
EDT =	1191				
Limit = 1	I = -5 F = -5	2493 (700 PA)	0.61	2663.1 (700 PA)	0.57
120	-5	1687.48	0.9	1666.71	0.91
120	0	1584.75	0.95	1531.43	0.99
120	5	1486.23	1.02	1406.57	1.07
120	10	1392.47	1.09	1292.98	1.17
120	15	1303.95	1.16	*1191.01	1.27
120	20	1221.1	1.24	1100.5	1.37

Figure 2: Sag and tension chart

STRINGING CHARTS

A disadvantage of using sag and tension charts is that the sag and tension information is only related to a single span length or many spans of exactly the same length. However, this is hardly ever the scenario that a designer is able to use. Designs generally have spans of variable length due to various obstacles or due to servitude rights. The nature of design and cost also means that pulling each span to an individual tension with termination structures each side of the span is not possible. Terminal structures are more costly than suspension structures and so the designer uses suspension towers as often as possible. A section of line is made of tension sections which are suspension spans with termination structures each side of the section. The tension in any span within this section is related to that of all the other spans within a tension section. The sagging in of lines is done between the two strain poles with rollers placed on each suspension pole (between the strain poles) in order to equal out the tension in all spans. Stringing charts are aimed at producing tables with the type of information to design line between strainer sections.

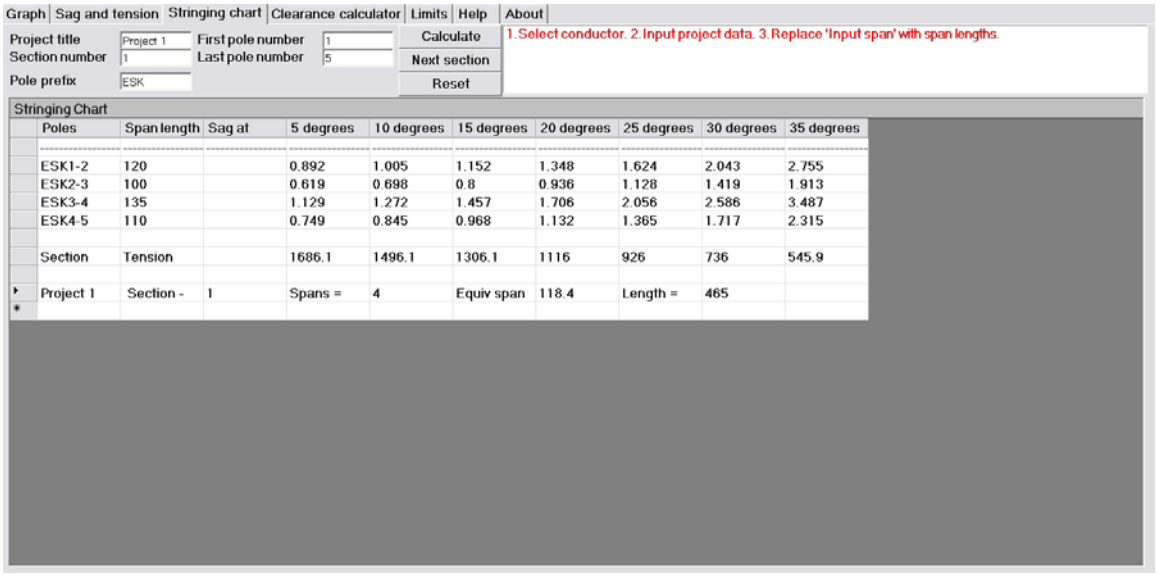


Figure 3: Stringing Chart

The tension through out the strainer section is constant and calculated using an equivalent span. Equivalent span is approximately a span length that if multiplied by the number of spans will equal the total length between strainers or in other words it is an average span length:

$$nl_e \approx \sum_{i=1}^{n} l_i \tag{7}$$

n = Number of spans

l_e = Equivalent span length

L_i = individual span lengths

The difference is that the equivalent span compensates for extremes in span length as these are the spans where the equivalent span will be most noticeably the incorrect tension. The equation to find this equivalent span length is:

$$l_e = \sqrt{\frac{\sum l_i^3}{\sum l_i}} \tag{8}$$

Thus with this equation an equivalent span can be calculated and the ruling limiting condition for the equivalent span can be found. The constant section tension is calculated at different temperatures. The sags for each individual span length are calculated from the tension and checked against the clearance requirements per span. It must be noted that, for higher accuracy, spans in a section should not have a great range of span lengths. It is good design practice to keep span length differences within 25 % range of each other if possible.

THE CLEARANCE CALCULATOR

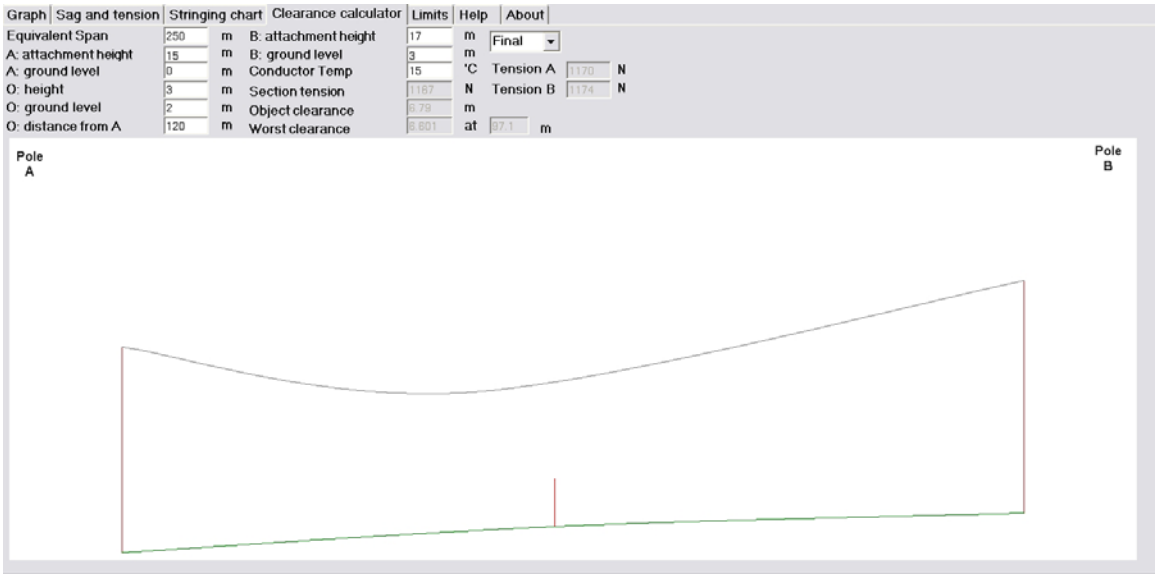


Figure 4: Clearance chart

A clearance calculator is a useful tool for finding the clearance of the conductor to a object. It is also used to find the clearance and tension at the structure for unusual spans (long spans over a valley, down a hill, over a road, over a building, etc).

The characteristics of the span needed to get the required clearances are:

- the attachment heights,
- the ground levels, and
- the detail of the object (to which clearance is calculated).

The tension quoted in SAT and stringing charts is the horizontal tension at the lowest point of the conductor catenary. For the clearance calculator it is needed to give the tensions at the attachment points and at the horizontal tension at the lowest point of the conductor catenary. The calculator is used the string unusual spans which can have significant vertical components of tension. Another reason attachment tension is required is that the tension at the attachment point is the tension measured by the dynamometer when clamping in the line. This tension will determine the strength requirement of the clamping, hardware and towers. This increased tension may indicate to the designer that vibration dampers are required for the particular span. If horizontal tension is used the designer may overlook this requirement.

Calculating the attachment tension is done by summing the horizontal tension and vertical tension vector components for the conductor. The horizontal component is the component generated in the normal sag and tension equations. The vertical component is worked from the conductor weight of the conductor from the attachment point to the lowest point of the catenary.

The clearance calculator is setup to allow the designer to consider different attachment heights, land levels and object heights. This tool gives the designer flexibility in terms of real life situation design in particular where clearance or uplift can present a challenge to the designer. The uplift case is where the lower of two structures is significantly lower than the higher structure that the conductor pulls the bottom structure up. This means that the structure falls below the natural catenary profile.

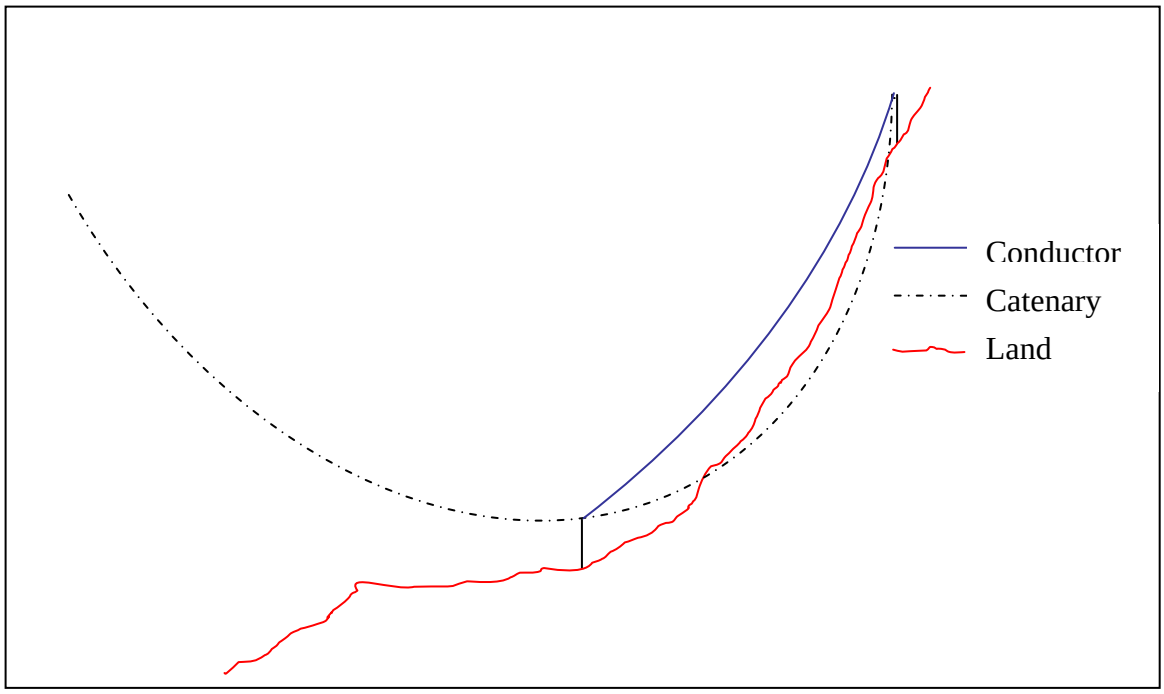


Figure 5: Uplift verses Catenary

Critical span graph

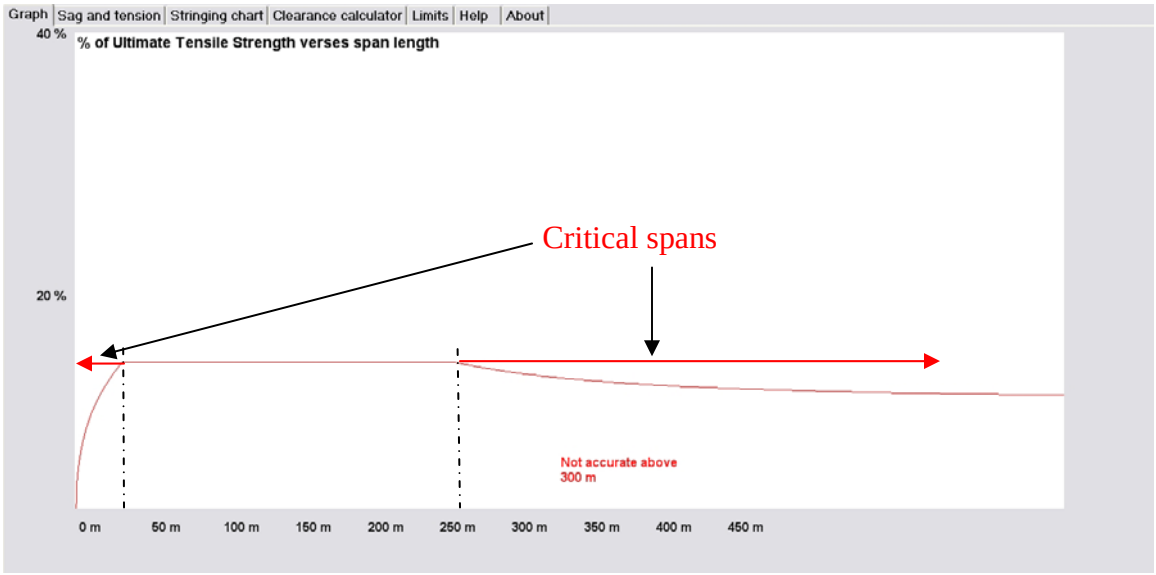


Figure 6: RSAT critical span graph

The critical span graph is a presentation of the tension (in percentage of ultimate tensile strength) against the span length. This relation gives the designer the range of spans for which the conductor will and will not be optimised.

In the case given above spans below 25 meters require the tension be reduced considerably. This reduction is caused by the low ratio of conductor length to percentage of elongation due to creep and temperature change. In particular the reduction total length of conductor due to temperature presents a significant value. Hence the everyday tension is limited by the low temperature statutory limits and does not allow for optimum tension of the conductor.

The upper critical span (above 250 meters in the example above) marks the point beyond which the limiting criteria change from the C-value to the high wind limit. The longer the span length the greater the effect of the winds force on the span. The resultant weight of the conductor increases with span until the maximum allowable tension is reached in one of the statutory requirements. Then this limit will reduce the every day tension, not allowing for the optimum tensioning of the conductor. Both short and long span length limitations of tension present the designer with an inefficient tension. In order to optimise line design, the designer may need to reduce the number of spans by increasing the tension in the conductor. This may require that the designer choose another conductor if the span lengths need to be reduced to satisfy the limits.

SOFTWARE USABILITY

Usable software “behaves exactly how the user thought it would” [18]. The design of the software must seek to ensure that the interface makes sense, operates with ease and presents the path of least resistance as a design tool. The amount of time to complete a task should be perceived to be short and the software should motivate the user to use the software.

A common problem with engineering software is that the usability is not prioritised as an outcome of the development team. The designers of the software may no be the users or may not be competent in software design (usually from the field expertise that the application is going to employed in). The priorities of the development team are therefore accuracy and functionality. The software is deemed to be intuitive by the developer but never really checked to ascertain the usability of the software. This means the software may be correct but not usable. This is because the software is not programmed with the user’s needs or preferences in mind. The user requirement is replaced with the designers introspective answers to design questions and preferences.

RSAT was initially developed in order to replace an old obsolete (in terms of operating system support) SAT application. The time given to develop the software was short. The modus operandi was to produce a small accurate calculator of sag and tension. There was no research completed into what compiler to use or the best development practice. The development was rapid and focused on the calculated solutions. The result was an accurate but “difficult to use” program. This feedback on the interface proved there was a need to redesign the software. The redesign needed to take into account the user’s perspective, usability and aim to remove possible boundaries that would cause users not to use the software.

The users of the RSAT software are field technicians, project engineers and line consultants. With this wide scope of users the need for training on software is not only difficult but would serve as a boundary to its acceptance and use. Hence a desirable property of the software is that the users need not be educated. This means having an intuitive user interface. In the past no training was been available for RSAT. The result that users that persisted grasped the concepts required to use the software. If the concepts were not grasped the user would contact the designer for assistance with respect to usage.

Support of the software was the starting point for the usability research. Feedback on the software was high and hours guiding users with concepts of design has lead to much of the changes presented in this dissertation. However, usability is not only the ease of use of the software but the amount of time to complete a task and the fact that it motivates the user to use the software (characteristics desirable for the RSAT software).

The literature survey revealed that the issue of usability of software is not new. The problem has persisted since the first software was produced in non-graphical operating interfaces like Microsoft DOS. The advent of new more usable operating system and complier interfaces shows the strides that have been taken to improve the situation. However, new operating systems with improved usability do not ensure that software being used on the operating systems have being improved at the same rate. A key to improving usability of an application is to note how these new operating systems have gone about improving usability. Designing the interface in a similar manor as the operating systems interface means that the user is already familiar with the interface. Research into new tools offered guided the developer to more usable functions, graphics and interfaces options.

Software usability has become of such great importance that before software packages are purchased, a purchaser can now insist that a usability study must be completed on the software before the software is purchased. The advantage of completing a usability study are that the best suited software can be chosen for the business and if there are revisions required, these can be completed before implementation into a business.

The uniqueness of this usability study is that it is done by the developer on who is also the owner of the software.

A pertinent question at this point is: How can a study be done such that it can be repeated for various different packages, compare them on a consistent test and reporting criteria? To this end a specification for testing criteria based on the fundamentals was setup by software customers internationally. The result can be found in ISO TC159/SC4-JTC1/SC7/JWG1 – a joint working group on common industry format that was formed to develop the standards for usability requirements. The group has drafted a Common Industry Specification for Usability Requirements. An industry standard format for reporting on usability has been established and is called the Common Industry Format (CIF). The common Industry Format for usability Test Reports became a standard in December 2001 in ANSI/INCITS 354-2001. Other interest groups looking into the question of software usability are the Association for Computing Machinery's Special Interest Group on Computer-Human Interaction (ACM SIGCHI) and the Computer Systems Group of the Human Factors Society. With business software and licensing costs increasing the demand for such groups and related specifications is increasing. The standards are still crude and adoption is not universal at this point. The concepts embodied in these standards will tested further in this study. Some new concepts will be added and reviewed.

Usability specifications suggested setting up a user laboratory where users can be observed using the software. The standard usability test centres spared no expense with video cameras setup to record the user actions and responses to the software. The disadvantage of using this methodology is that it does not clearly depict the software being used under normal conditions and is expensive. Under normal conditions the user has his/her own computer, with his/her own system set-up and in his/her own office. Thus, in a laboratory, the environment and equipment being used is unfamiliar to the testing user. This may cast doubt over the validity of the results for usability which is based on intuitively of the software and is supposed to be independent of other environmental factors.

The first testing completed on RSAT was done through field testing. This allowed the user to use the software on his or her personal machine in a familiar natural environment; making the only variable the software. This technique allows for various forms of feedback. Direct feedback was gained by visiting the user and watching the user in his daily work using the software. A less direct method of using webcams/video conferencing could have been used as an alternative but this required an elaborate system of web cams and microphones. This type of equipment was not available to the average user of RSAT. Lastly the user was interviewed with respect to their experience of the software in a field test. The emphasis was on getting direct feedback on the system. The earlier the feedback is acquired the better as the feedback can direct the future efforts and point out short comings. A user specification was then initiated. The specification focused more on the user interface, speed of use and the ability of the software to naturally point the user in the right direction.

The software design process then took on an evolutionary or a prototyping type of process. A number of interviews were taken as a means of user requirement feedback. Then the feedback loop process was completed until the amount of users feedback showed the software to be at an acceptable level. Note that this is not always an ideal solution since the number of iterations and numbers of interviews per iteration may be limited by the budget afforded a development team.
Interviews need to concentrate on:

- what the user naturally does to complete tasks,
- the bottle necks that slow user progress,
- the redundant sections of software,
- redundant actions and
- suggestions on how to make tasks easier while not sacrificing on achieving the goal of the software.

The interpretation of the user interviews, questionnaires, comments and ultimately the usability requirement was put together by the lines working group. It was important that the team could speak openly about this requirement with as little as possible left to the each team member's perception of the requirement. This was done through a workshop of the requirements.

Measuring usability is not something that one can perceive or see without some sort of scoring system. Therefore it is important in the usability specification to set usability criteria. These are usually set around the time to complete tasks, the ease with which a new user can use the software, error recovery and the user's opinion of the software. Michael Good [12], of the Compaq Computer Corporation, suggests that each attribute consist of five items:

- the measuring technique,(interview, questionnaire, etc)
- metric, (how the attribute is expressed as a measurable quantity)
- worst case level, (a range of results which do not meet the specification)
- planned level (a range of results which meet the specification)
- best case level. (a range of results which exceed the specification)

The risk in setting up a usability specification is that the broad scope of usability can be traded for the user specifications narrow definition of usability. This is an acceptable trade off if the specification is setup correctly and defines the usability well. Even a poor usability specification is better than none at all, but this may mean that resources are invested in the wrong direction or in only a small section of the whole problem.

APPENDIX B

DESIGN SPECIFICATION

B1 SYSTEM REQUIREMENTS

The program must:

- run in conjunction with Microsoft Windows XP.
- be downloadable from the Eskom internet site in less than 5 minutes.
- have a graphics driven interface that resizes depending on the monitor resolution.
- be easily installed on current imaged Eskom PC's and by Eskom contractors.
- have a programming language and compiler that allow for transfer to other platforms as the need arises and suit the application requirements.

B2 ACQUIRING THE USER REQUIREMENT

The user requirement techniques need to be defined for the software with the users specific needs in mind. The standard flow of information can be seen in figure 1 one below.

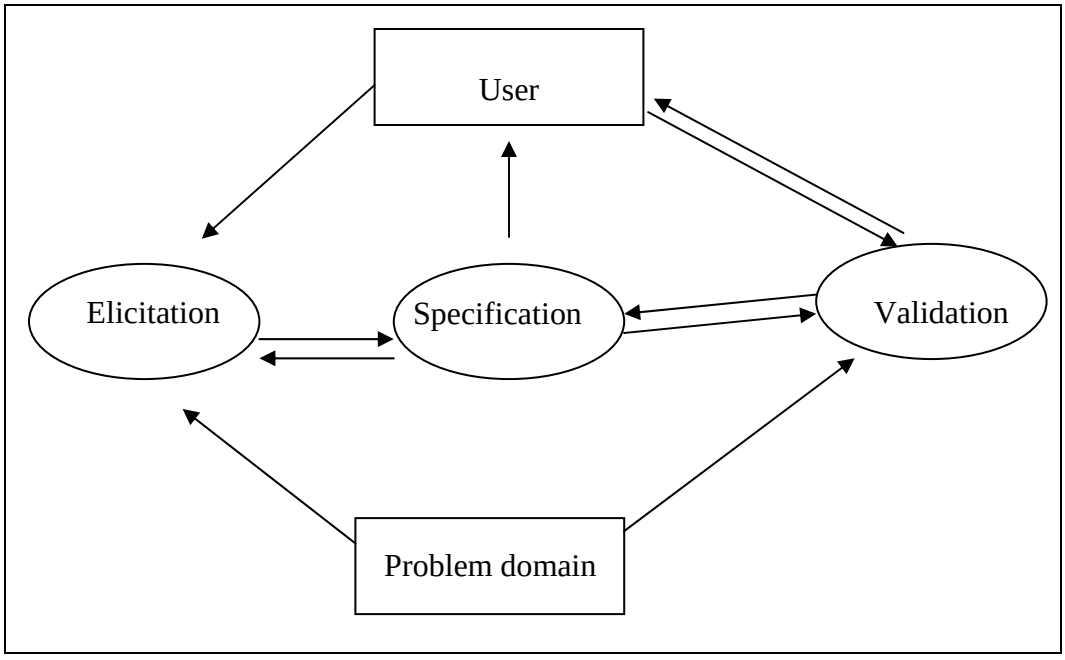


Figure 7: Information flow to acquire technical specification

Each technique used to facilitate the flow of information to form a user requirement is now discussed.

B2.1. THE DELPHI TECHNIQUE

The Delphi technique is a technique where the development team get the users to write requirements and ideas for the software down.

Each member of the Eskom Distribution Lines study committee was requested to give a set of minimum requirements for the sag and tension software package. Each member represented a region of the power system. Thus requirements were received from all regions customers resided in.

B2.2. BRAIN STORMING TECHNIQUE

The brain storming technique is best described a group discussion of ideas with a scribe writing them up on a board.

The requirements obtained through the Delphi technique as well as newly proposed ideas were discussed at a study committee meeting with the developer putting ideas up on a projector. The requirements were thoroughly discussed and filtered down to a concise list.

B2.3. DOMAIN ANALYSES

The domain analysis is a study of the domain and not the actual system to become conversant with norms and terminology.

The domain for the software is familiar to the developer as the developer is employed in the domain – the electrical utility. Terminology and norms for the domain are discussed later.

B2.4. TASK ANALYSIS

The task analysis seeks to define tasks that the system is expected to perform.

The list of requirements from the study committee, gained in the Delphi and Brain storming techniques, were then reviewed and broken down into a group of tasks expected to be performed by the software.

B2.5. USE CASE ANALYSIS

Use case analysis involves specific instances of the system at work. Examples should be run through to note the system flow process and results required.

Specific instances of sag and tension, stringing charts, clearance calculators, limitation changes, help files and critical graphs were drawn up to ensure all pertinent issues are captured. Flow process and results were correlated to the user expectations.

B2.6. SYNTHESES FROM EXISTING SYSTEMS

This involves the study of other software packages available for methodologies of meeting requirements.

PLSCADD, Sag10, Vsat and a number of papers written on the topic of sag and tension were scrutinised for possible methodologies of meeting the user requirements.

B2.7. PROTOTYPING

Prototyping involves creating a model of what you are going to do rapidly. Then present the model to users to see if the concept is a favourable solution to customer. It presents the customer with a broad view of the concepts and checking mechanism to see interpretation of the requirements.

The user requirement was transferred rapidly into preliminary design software with limited functionality. Once accepted a beta version was programmed and released for users to start using. Users were given means of feeding back issues, comments, errors and possible improvements to the software. Comments and improvements were user to update the user requirement.

The user requirement below is the final outcome of a number of iterations through each step. In going through certain steps in the process it was necessary to revisit previous steps to refine the results.

B3 USER REQUIREMENTS

The user requirements are as follows:

- Sag and Tension charts: with a default temperature range and span range.
- The ranges of span and temperature should be editable by the user.
- Stringing charts: with a project name, section number, customized pole numbers, the ability to print more than one section per print out and the ability to edit old section data.
- Clearance calculator: with worst case clearance and specific case clearance to an object. The option should be available for clearance calculation with initial or final tension.
- Print outs of the relevant data, charts and calculators.

- The user interface should be a simple interface with a conductor selection and data being available and editable at all times without having to restart input of data from when changing conductor.
- A critical span indication is required such that conductor selection can be optimized.
- Conductor data files must be created for all Eskom Distribution standard conductors.
- An option must be provided whereby conductor data files can be updated with new conductors
- A save file option must be provided that results can be stored.
- Tensions at the structure must be produced for the clearance calculator.
- The default Every Day Tension limit must be a final stringing tension and be at a C value of 1425 for 6/1 strand ACSR and 7 strand AAAC so as to avoid the unnecessary use of vibration dampers.
- The option to change the default C-value should be made available to the user.
- The default initial tension statutory limit must be set-up as 50 % of ultimate tensile strength of the conductor at -5 degrees with a 700 Pa wind on the conductor.
- The default final tension statutory limit must be set-up as 40 % of ultimate tensile strength of the conductor at -5 degrees with a 700 Pa wind on the conductor.
- The option to change the limiting criteria should be made available to the user.
- Conductor elongation due to creep and temperature variation should be correlated to PLSCADD.
- The use of setup password protection of software as a means to protect the financial interests of the code owner was initially part of the user requirement. However, in the process of refining the user specification this requirement was removed. The reasoning was that the use of standardised software by Eskom and contractor personnel is more important than the financial gain from sale of the software.

B4 DOMAIN MODEL

The domain model of terminology extracts from the user requirement as follows:

Sag	Tension	Span	Chart	Clearance
Print	Conductor	Critical span graph	Input	Simple interface
Data files	Save	Structure tension	Default	Everyday tension
C value	Limits	PLSCADD	Pole	Print Preview
ACSR	AAAC	Graphics	User	Selection
Vibration	1425	40% UTS	50% UTS	700 Pa wind
Structure	Clamping tension	Wind pressure	Temperature	Increment
Calculator	Damper	Aluminium	Galvanised Steel wire	Standard conductor
Algorithm	ABC	Service cable	Data	Ultimate Tensile Strength
Creep	Equivalent span			

B4.1 SIFTING THE LIST

Synonyms

Structure = Pole

Algorithm = Calculator

Vibration = Damper

Conductor = Standard conductor

Wind pressure = 700 Pa wind

Structure tension = Clamping tension

Redundant

Simple interface

User

Selection

Structure

Vibration

C value

Data files,

Outside scope

PLSCADD

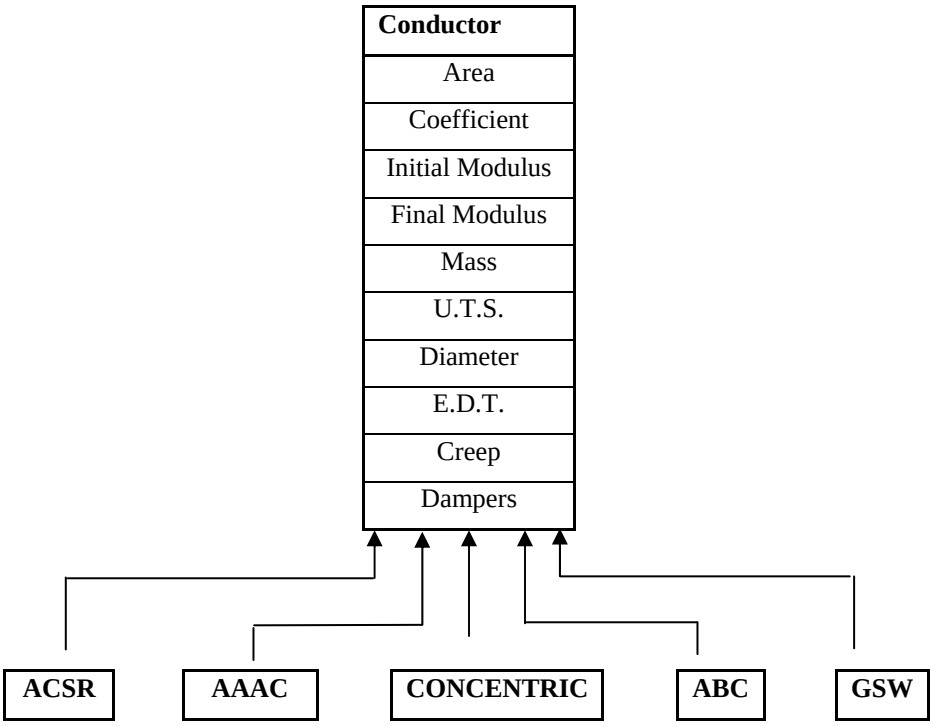
Final sifted list

Sag	Tension	Span	Chart	Clearance
Print	Critical span graph	Input	Save	Structure tension
Default	Everyday tension	Limits	Print Preview	ACSR
AAAC	Graphics	ABC	Creep	Ultimate Tensile Strength
1425	40% UTS	50% UTS	Wind pressure	Data preview
Temperature	Increment	Aluminium	Service cable	Galvanised Steel wire
Conductor	Algorithm	Equivalent span		

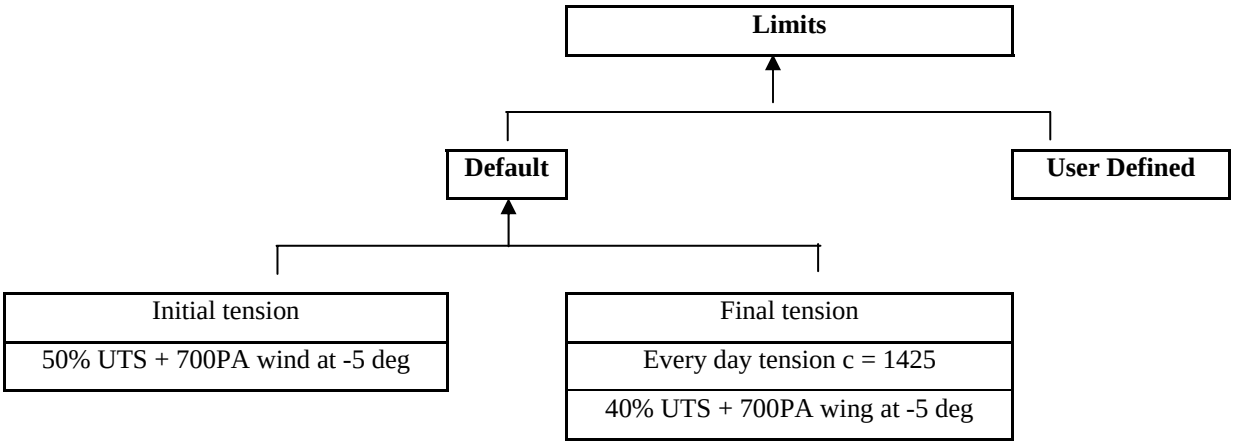
B4.2 GROUPING

Terms were then grouped and with additional terms (**in bold**) required to complete the picture in terms of required relationships.

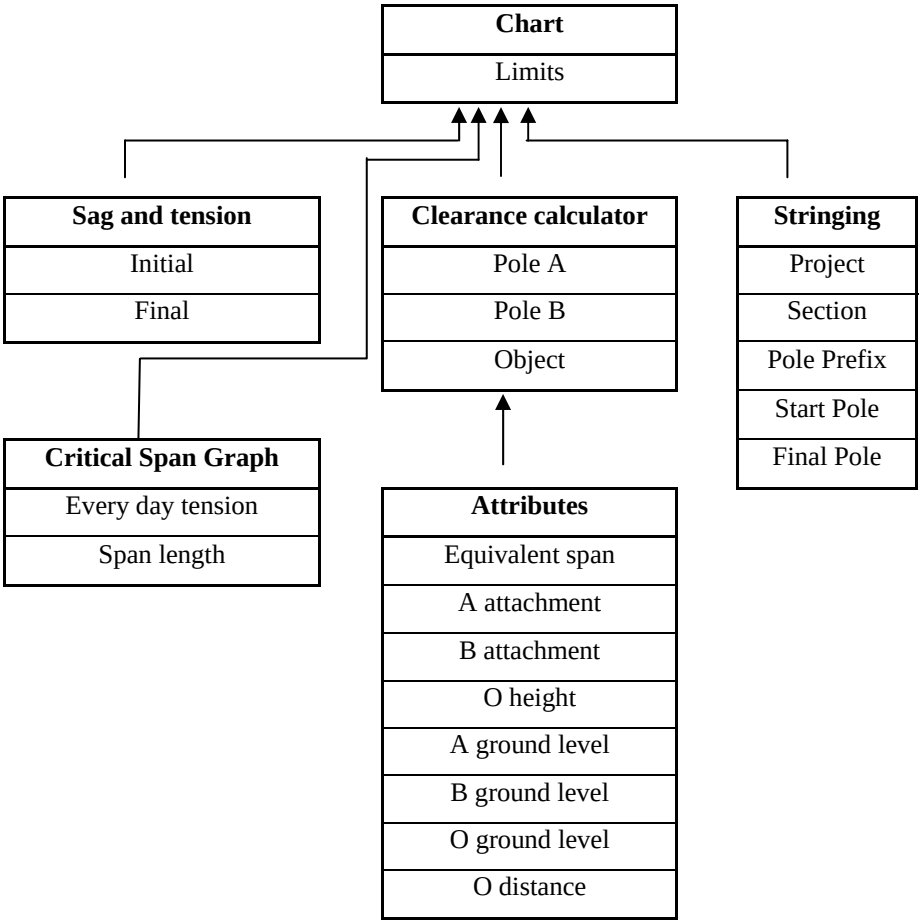
Conductor: Types: ACSR, AAAC, Steel, Aluminium, ABC, Service cable, Ultimate Tensile Strength, **Area, Coefficient, Initial Modulus, Final Modulus, Mass, Diameter, Every Day Tension, Creep and Dampers**



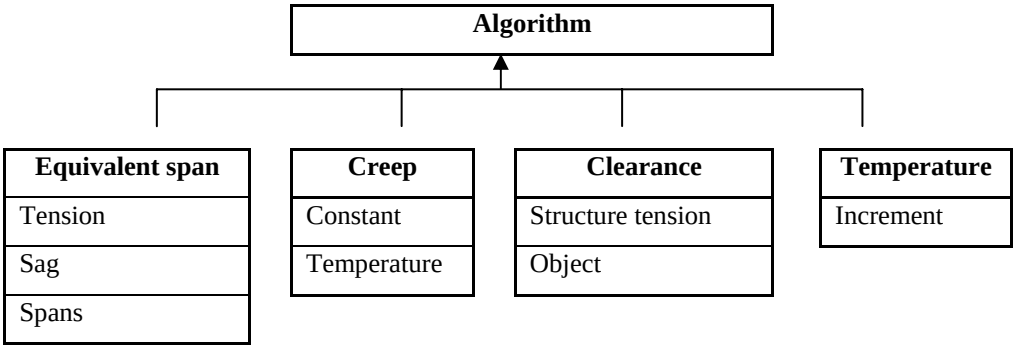
Limits: 1425, 40 % UTS, 50 % UTS, Everyday tension, Wind pressure, Default, **User defined**, **Initial Tension and Final Tension**



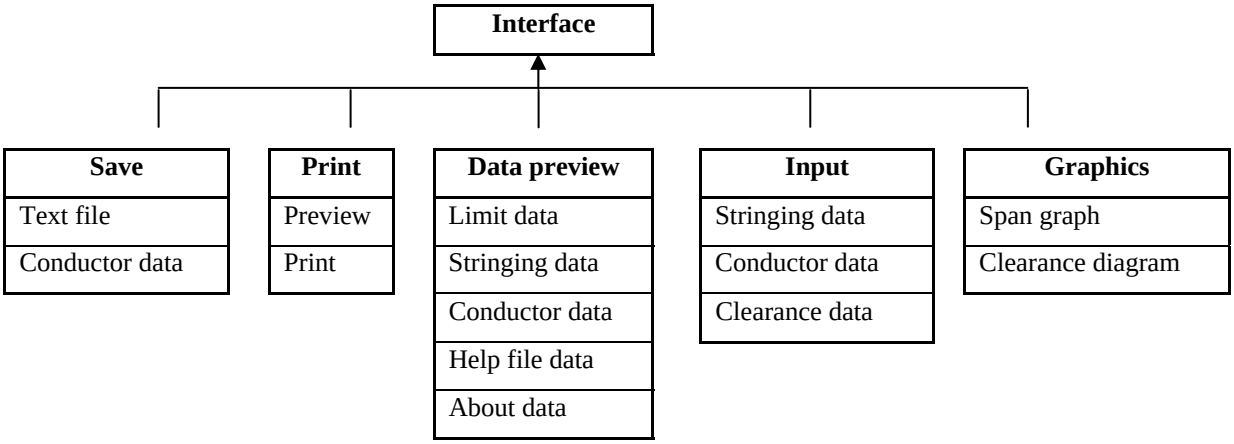
Charts: Sag and Tension, Limits, Stringing, Clearance, Graphics, Critical span, Critical span graph, **Initial, Final, Pole A, Pole B, Object, Project, Section, Pole Prefix, Start Pole, Final Pole, Every Day Tension, Span Length, Equivalent span, Pole A Attachment, Pole B Attachment, Object height, Pole A ground level, Pole B ground level, Object ground level and Object distance from Pole A.**



Algorithm: Temperature, Increment, Structure tension, Creep, Equivalent span, Clearance, Tension, Sag, Spans, **Constant and Object**



Simple user interface: Graphics, Limit, Stringing, Conductor, Input, Print, Data preview, Print preview, Save, **Text File, Conductor Data, About and Graph.**



B5 SOFTWARE DESIGN

B5.1 Compiler

The old software design of RSAT needed to be reviewed entirely. The software ran slowly and was cumbersome to use. The old compiler – Borland C++ Builder is no longer supported on the XP operating system used by Eskom power Utility RSAT is designed for. The software required an overhaul from compiler to the basic code for the software.

A compiler needed to chosen that suited the dynamic nature of the software. The software needs to flexible in terms of platform installed on. Eskom runs an imaged set of standard software. For this reason it may be necessary at a later stage to have a web based calculator as opposed to a stand alone application. The modernisation of the field staff equipment may mean that in future the field staff have hand held devices that can run small applications such as sag and tension or clearance calculators. Hence the compiler chosen in .NET represents a compiler that allows compilation of code to different type of application for different platforms.

The RSAT software is an in-house developed application distributed freely to project engineers and consultants. There is no assurance that the software will maintained by the same development team from one version to the next. The ability of the compiler to compile code in different languages for add on components and have a universal tool set would be the best compiler for the project. ".Net" presented the development team with these characteristics.

Since it was deemed that the software would be upgraded, for use on handheld devices, in future, the language of choice was changed from C++ to C#. C # presented the developer with a challenge initially in terms of the new syntax and methodologies of completing simple operations. However, this setback was seen as minor compared to the boundaries that will be faced in producing the software for handheld devices.

B5.2 Source Code

The code design was written around the group terminology given in section B4 above. The code was written quickly and prototypes were produced and tested. Initial prototypes used classes for conductors, algorithms and charts. There is no necessity for the software store more than one conductor at any given time. The user feedback was that the more information must be displayed continuously. This made the access of this information easy and hence the need for class use was not great and only the conductor class was retained. Simple methods were used for the algorithms required for the software. The methods used are as follows (extracted directly from the source code):

```
1. public void ResetStringing()
   //Resets the string chart if a new project is to be strung.
2. public void DefaultLimits()
   // Resets the limits on the limit page to the default values -
   default values are stored in the limits XML file.
3. public double ToInitial(double L,double t1,double w1,double
   c1,double t2,double A,double Coef,double Imod, double Fmod,
   double Wc, double Diam, double T20, double L12, double k)
   //Convert to initial at convert temp.
   //Arguments: L = span length, t1 = temp 1, w1 = wind 1, c1 =
   tension 1, convert at temp = t2, A = cond area, Coef = cond temp
   coef, Imod = initial modulus, Fmod = Final Modulus, Wc = cond
   weight, Diam = conductor diameter, T20 = 20 % of UTS tension, L12
   = L1.2(required for calculations) and k = creep constant.
```

```
4. public double ToFinal(double L,double t1,double w1,double
   c1,double t2,double A,double Coef,double Imod, double Fmod,
   double Wc, double Diam, double T20, double L12, double k)
   //Convert to final at convert temp.
   //Arguments: L = span length, t1 = temp 1, w1 = wind 1, c1 =
   tension 1, convert at temp = t2, A = cond area, Coef = cond temp
   coef, Imod = initial modulus, Fmod = Final Modulus, Wc = cond
   weight, Diam = conductor diameter, T20 = 20 % of UTS tension, L12
   = L1.2(required for calculations) and k = creep constant.
5. public double TempConvert(double L,double t1,double w1,double
   T1,double t2,double w2,string IF,double A,double Coef,double
   Imod, double Fmod, double Wc, double Diam, double T20, double
   L12, double k )
   //Converts the tension for any change in temperature.
    //Arguments:L = span length, t1 = temp 1, w1 = wind 1, T1 =
    tension 1, t2 = temp 2, w2 = wind 2, IF = initial or final calc,
    A = cond area, Coef = cond temp coef, Imod = initial modulus,
    Fmod = Final Modulus, Wc = cond weight, Diam = conductor
    diameter, T20 = 20 % of UTS tension, L12 = L1.2(required for
    calculations) and k = creep constant.
6. public int findlimit(double L,double A,double Coef,double Imod,
   double Fmod, double Wc, double Diam, double T20, double L12,
   double k )
   //Determine which of the limits is the ruling limit producing the
   lowest tension at the every day conditions.
    Arguments: L = span length, A = cond area, Coef = cond temp coef,
    Imod = initial modulus, Fmod = Final Modulus, Wc = cond weight,
    Diam = conductor diameter, T20 = 20 % of UTS tension, L12 =
    L1.2(required for calculations) and k = creep constant.
7. public void equivspan()
   // Algorithm to determine the stringing charts output
8. public void stringupdate()
   // Update sag and tension chart when changes to any project input
   is made for the current section is made.
9. public void graphupdate()
   // Updates the graph when the conductor is changed
```

```
10. public void satupdate()
    // Updates the sag and tension chart when changes to any variable
 is made.
11. public void clearance()
    // Updates the clearance calculator page when changes to any
    variable is made.
12. public void clpicture()
    // Updates the graphic on the clearance page
13. private void ExportToText (string fileName, bool
    hasColumnHeaders, double columnWidth)
    // Copies displayed data to a file
    // Arguments: filename = the file name to be saved to,
    hasColumnHeaders = has the data got column headers, columnWidth =
    column width
14. public void tabupdate(object sender, System.EventArgs e)
    //update tab based on selection.
    // Arguments: capture the selected tab
15. public void screensize()
    // Updates size of the form based on screen resolution.
```

The use of the above methods reduced the source code required considerably and made coding easier to read and produce.

The rest of the code produced consists of code and methods produced by the compiler split into the following sections:

- The conductor class
- Windows form designer generated code and the application run command.
- General functionality menus and buttons
- Changing conductors and new conductor parameters
- Changing string chart parameters
- Changing clearance chart parameter

The complete code is given in Annex D.

B5.3 The user interface

The user interface is the area that was are an area of focus, in order to achieve the goals of producing a more usable application.

The old interface

The old interface presented at start-up is not an intuitive first operation.

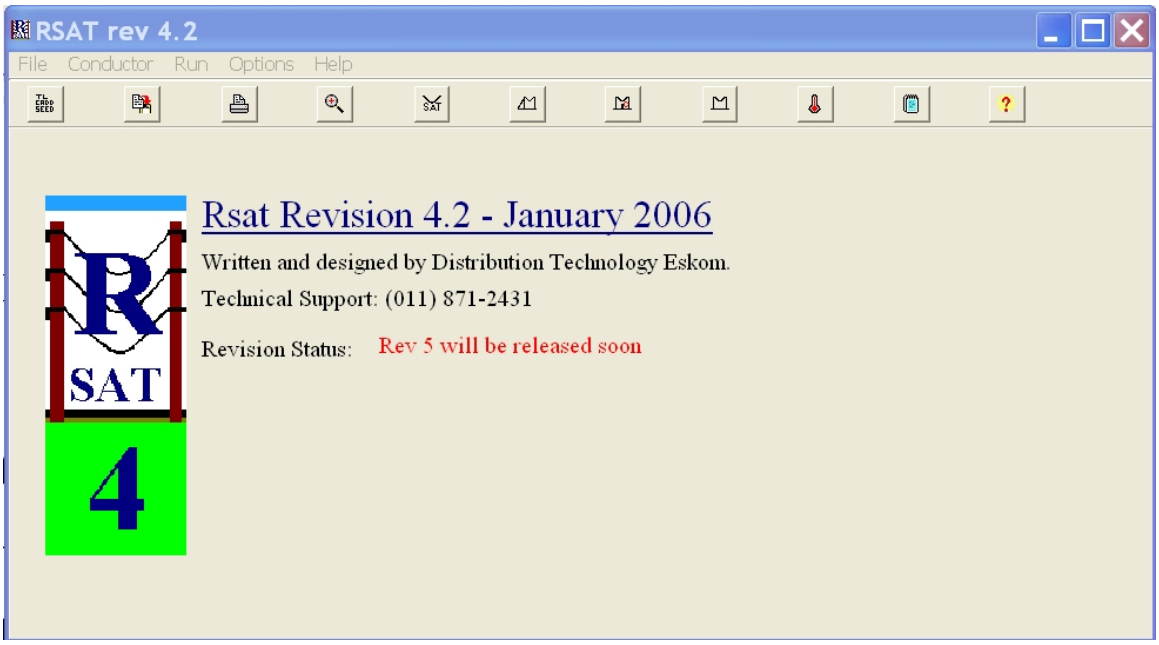


Figure 8: RSAT's old interface

By selecting Help the new user can find out how to proceed. This is not a desirable first operation. A possible improvement on this was to insert guiding messages but even this did not work if the user does not recognise the message box as an indication of what needs to be done.

The first operation required is to select the conductor menu option. This presents the user with a drop down of the types of conductor that can be selected. Once a selection is made, a new model conductor form is opened.

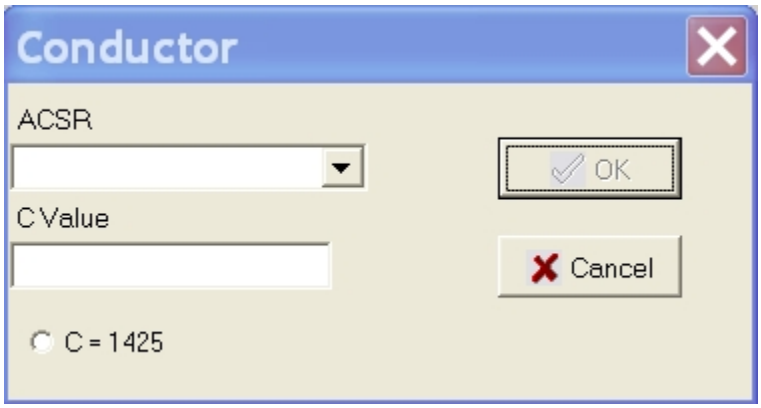


Figure 9: Old conductor form

This form is intuitive with the conductor to be selected from the drop down. The C-value is displayed and then the “OK” or “Cancel” selection can be made. If a conductor and “Ok” are selected, the respective conductor data form is displayed.

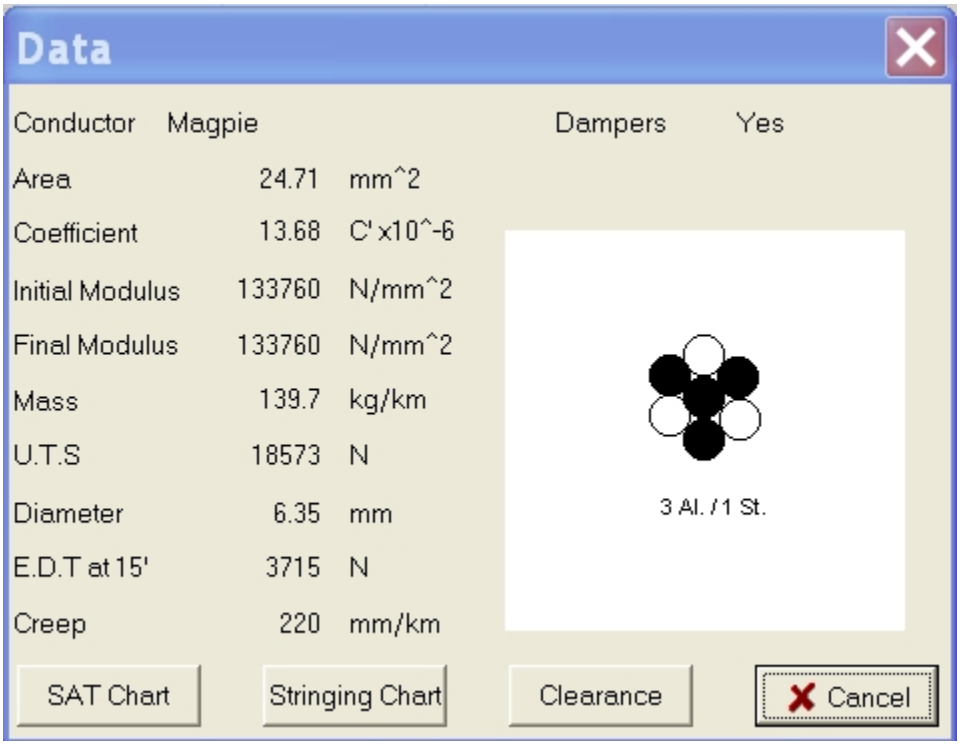
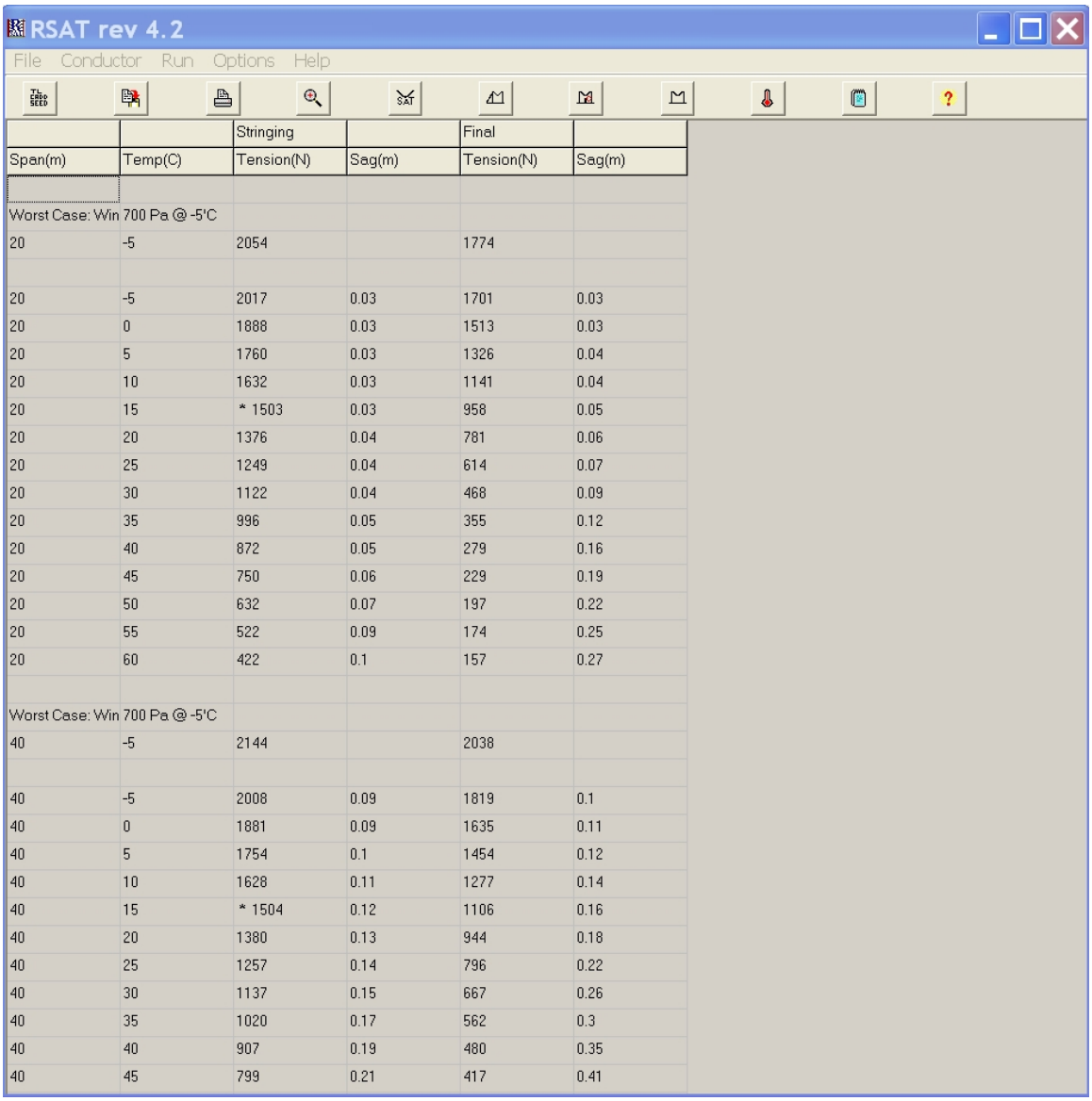


Figure 10: Old conductor data form

Finally the type of algorithm can be selected at the base of the form. This brings up the result display on the original form.



		Stringing		Final	
Span(m)	Temp(C)	Tension(N)	Sag(m)	Tension(N)	Sag(m)
Worst Case: Win 700 Pa @ -5'C					
20	-5	2054		1774	
20	-5	2017	0.03	1701	0.03
20	0	1888	0.03	1513	0.03
20	5	1760	0.03	1326	0.04
20	10	1632	0.03	1141	0.04
20	15	* 1503	0.03	958	0.05
20	20	1376	0.04	781	0.06
20	25	1249	0.04	614	0.07
20	30	1122	0.04	468	0.09
20	35	996	0.05	355	0.12
20	40	872	0.05	279	0.16
20	45	750	0.06	229	0.19
20	50	632	0.07	197	0.22
20	55	522	0.09	174	0.25
20	60	422	0.1	157	0.27
Worst Case: Win 700 Pa @ -5'C					
40	-5	2144		2038	
40	-5	2008	0.09	1819	0.1
40	0	1881	0.09	1635	0.11
40	5	1754	0.1	1454	0.12
40	10	1628	0.11	1277	0.14
40	15	* 1504	0.12	1106	0.16
40	20	1380	0.13	944	0.18
40	25	1257	0.14	796	0.22
40	30	1137	0.15	667	0.26
40	35	1020	0.17	562	0.3
40	40	907	0.19	480	0.35
40	45	799	0.21	417	0.41

Figure 11: Old result form for sag and tension algorithm

To select an alternative algorithm the quick buttons or menus can be used and the algorithm will be shown for the selected conductor. To change the conductor the whole process is stated from the beginning.

The new interface

The new interface, presented at start-up, is intuitive as to the first operation required.



Figure 12: RSAT's new interface

The new interface has the conductor types displayed in the top left list box. A suggestion window also prompts the user to select a type of conductor. Once selected the conductor list box fills with possible conductors to select. The suggestion then prompts the user to select a conductor. On selection of a conductor the displayed graph displays the corresponding critical span graph and all the conductor data to the right of all the conductor list box.

Note: for quick access a conductor can be selected directly from the conductor box without selecting the type filter.

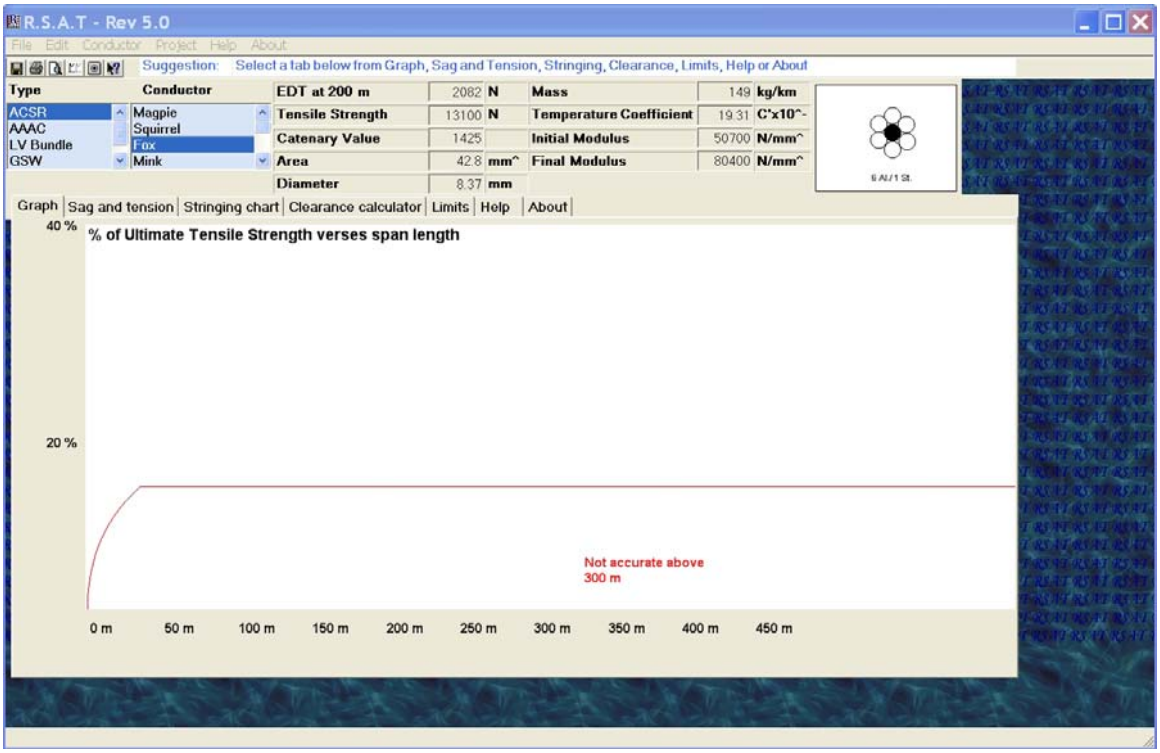
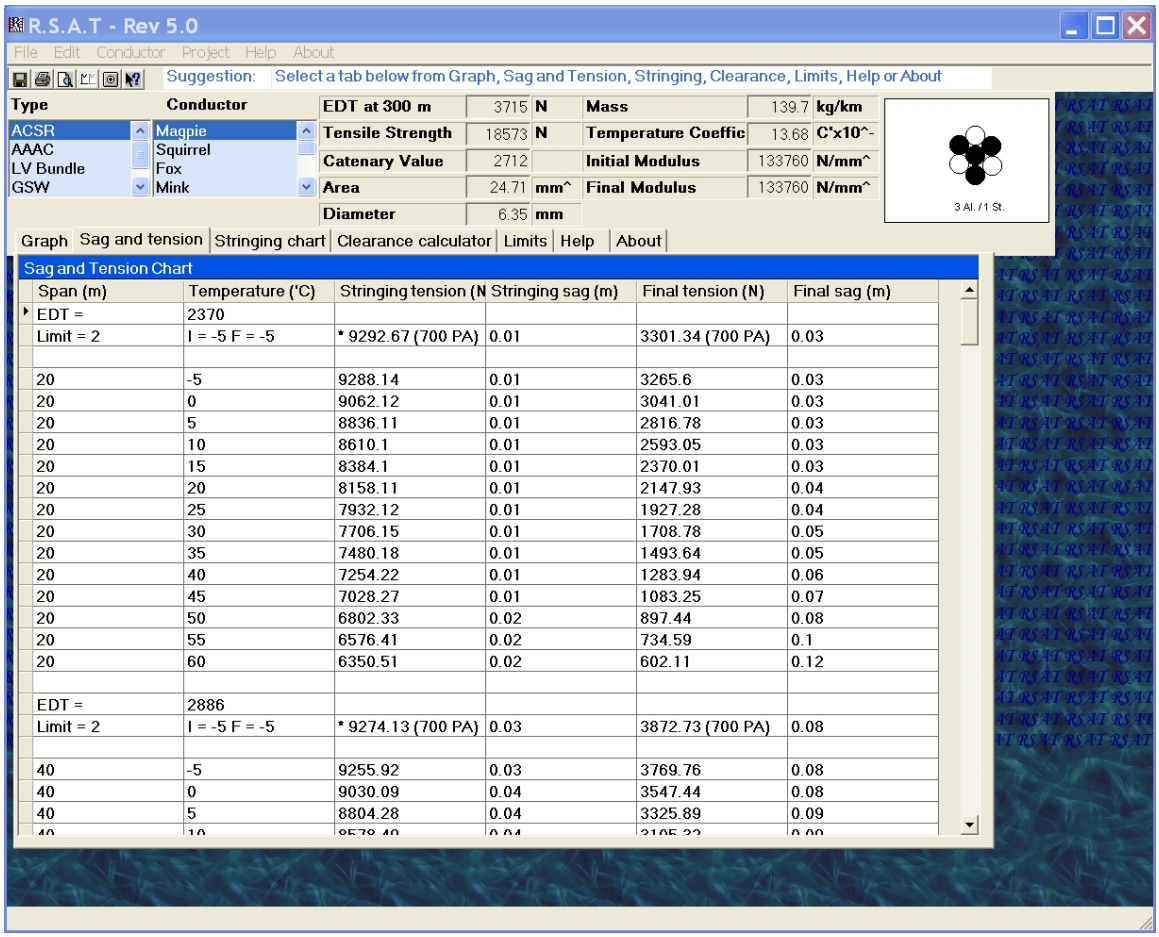


Figure 13: New interface with a conductor selected

To select an algorithms simply select the tab of the algorithm that is required. The suggestion prompts the user at this point to select a tab on screen.



Span (m)	Temperature ('C)	Stringing tension (N	Stringing sag (m)	Final tension (N)	Final sag (m)
EDT =	2370				
Limit = 2	I = -5 F = -5	* 9292.67 (700 PA)	0.01	3301.34 (700 PA)	0.03
20	-5	9288.14	0.01	3265.6	0.03
20	0	9062.12	0.01	3041.01	0.03
20	5	8836.11	0.01	2816.78	0.03
20	10	8610.1	0.01	2593.05	0.03
20	15	8384.1	0.01	2370.01	0.03
20	20	8158.11	0.01	2147.93	0.04
20	25	7932.12	0.01	1927.28	0.04
20	30	7706.15	0.01	1708.78	0.05
20	35	7480.18	0.01	1493.64	0.05
20	40	7254.22	0.01	1283.94	0.06
20	45	7028.27	0.01	1083.25	0.07
20	50	6802.33	0.02	897.44	0.08
20	55	6576.41	0.02	734.59	0.1
20	60	6350.51	0.02	602.11	0.12
EDT =	2886				
Limit = 2	I = -5 F = -5	* 9274.13 (700 PA)	0.03	3872.73 (700 PA)	0.08
40	-5	9255.92	0.03	3769.76	0.08
40	0	9030.09	0.04	3547.44	0.08
40	5	8804.28	0.04	3325.89	0.09

Figure 14: New result form for selecting sag and tension algorithm

To select an alternative algorithm an alternative tab can be selected.

B6 AREAS OF CONCERN

B6.1 Usability

Satisfaction with the performance of the software, with respect to its general usability, must be defined by the users. The common way to elicit information concerning user satisfaction in usability studies is to setup a computer laboratory, observe the use of the software and then question the user on there general satisfaction.

The users of RSAT were observed using the old and new revisions of RSAT. This proved useful in finding the issues with usability and general satisfaction. User interaction showed that reset and default buttons were needed for certain algorithms. The laboratory also showed that a suggestion window would guide the user in decision making with respect to usage. The laboratory also revealed overlooked errors in functionality such as algorithms not updating when a new conductor was selected. Another error revealed was that if an algorithm was updated prematurely, it could cause system instability. For example when a high figure was inserted in a field, because the default was not deleted before inputting new data, the application would complete a great number of tasks before the error could be resolved. These little but vital error had not been picked up in initial testing done since these methods of input were not the norm.

Field testing of a Beta version also proved useful as errors and oversights were reported to the developer. The field testing liberated user preferences due to the fact that it allows the user a normal atmosphere on the users own PC were custom requirements become more important to the user.

Over time an e-mail list of users has been put together. This list of RSAT users was encouraged to test the Beta version of RSAT. The Beta software was downloaded from a website and installed. This process also proved the installation process was acceptable as the there was no feedback with respect to installation problems. However, the installation process and software was improved in linen with the new compilers setup facility. Another advantage is that the process was simplified as far as possible through a installer messaging system that guides the user through each step.

User feedback ranged from the odd error in the software to full length installation and usage guidance. Telephone, e-mail and visits allowed the developer establish contact with the users. In each problem case the user would be requested to complete the task causing problem and describe each step of the task. This way standard thought process and users expectations of how things should work was noted.

Finally the software was presented to a working group of users. The working group was held where specific issues could be raised with respect to the old software interface and general design. Problems that were evident were the following:

1. The password installation process needed further simplification
2. The methodology of selecting conductors did not allow for comparison of conductors.
3. Conductor data display needs to be more easily accessed.
4. The sequential nature of selection of conductor, conductor data and the selection of algorithm to be run took too long. Selection of a new conductor meant going through the whole process again.
5. No allowance was given for addition of new conductors limited the user's capability with the software.
6. Limiting criteria could not be varied in line with prevailing conditions for the area,
7. Incorrect input such as characters where an integer should be would case the software to crash.
8. Stringing charts could not be corrected for the first section once the next section was selected.
9. Selected examples would hang the software for no apparent reason.
10. The user interface gave no indication of the critical span.

B6 USABILITY IMPROVEMENT CRITERIA

The usability criteria are taken directly from comments from the working group. The three possible results: a worst case, planned case and best case were then defined for the particular criteria.

1. The installation process

The installation process of an application is the first impression a user gets of the application. It sets the tone for the user's experience of the application and is vital in terms of keeping the users interest in the software. If too cumbersome or difficult to understand the user may not install the software at all. Therefore the aim of the process should be to almost install itself without any user intervention. By the same token there are users of the software that are conversant with software installation and prefer to customise PC memory and application installation. The installation should seek to satisfy both types of user.

Criteria results

The **Worst case** result of testing for this criterion would be for there to be no change to the installation process.

It is **planned** to have a more intuitive installation process that has an intuitive process and explanations of all actions to be taken and guidance all the way through the process.

The **best** result would be to have an installation program that is intuitive in all aspects with one installation file and one execution required on one file.

2. The methodology of selecting conductors did not allow for comparison of conductors.

Comparison of conductor results is a requirement that comes from the more competent users of RSAT. The function will allow the designer the ability to optimise the selection of conductor easily. Currently the user needs to repeat a full set of operations to get subsequent conductor results. A single operation or dual view would present a better solution.

Criteria results

Again the **Worst case result is for there to be** no change to the process of getting a new conductor displayed.

The **planned** result would be the ability to select a new conductor for an algorithm with a single action.

The **best** outcome for comparing conductors would be to have a dual view arrangement where two or more conductors can be viewed at the same time.

3. Conductor data display needs to be more easily accessed.

If the software is used extensively and in particular in comparing conductors the need to see the current selected conductor is something the old version of RSAT omitted. An indication of the conductor being used and possibly it related data may be useful.

Criteria results

The **Worst case** would be to have no change made to the software.

The **planned** result would be to have the selected conductor displayed on the screen at all times.

The **best** outcome would be for the conductor and conductor data to be displayed on the screen at all times.

4. The sequential process of selection of conductor, conductor data and the selection of algorithm is too long.

Since the selection of the conductor is the first operation the user is expected to do each time the software is used - a process of conductor selection is required that is simple and quick. This will instil confidence in the software and ensure speedy and accurate use of the software.

Criteria results

The **Worst case** is for there to be no change made to the sequential selection process.

The **Planned** result is a new conductor selection process that is non-sequential and that does not involving different forms to be traversed in order to complete one task.

The best result that can be expected is a process where only one operation is required to select a conductor without having to choose type, data, algorithm etc.

5. An allowance needs to be given for the addition of new conductors.

Currently the standard set of conductors limits the user's capability with the software. The standard sets of conductors were only available in the first versions of RSAT. The philosophy employed was that users should be forced to use the standard conductors for new projects. However, it has been found that maintenance of lines requires a wider variety of conductors. The necessity for users to be able to add conductors become a requirement.

Criteria results

The **Worst case** is for no change to be made to the software.

The **planned** result would be to a customised conductor function which allows the user to store conductor data into a customised file.

The **best** outcome would be to have a process which allows conductors to be added to the current data files for specific conductor types. In addition new types of conductors should also be catered for.

6. Limiting criteria can not be varied in line with prevailing conditions for the area,

The limiting criteria setup for the application algorithms are sourced from the occupation health and safety act and the everyday C-value tension is from the most recently published papers on the subject. However, this only takes into account the average prevailing conditions for South Africa. Extreme conditions such as high winds may be common in some areas. The current software does not cater for these and should be revised to allow for at least one condition change.

Criteria results

The **Worst case** scenario would be to make no changes to the software.

The **Planned** change is to include a facility that allows the user to change one limiting criteria.

The best outcome would be for a new facility allowing all limits to be changed per task and the defaults limits to be restored or revised.

7. Incorrect input (e.g. inserting characters where an integer should be inserted) would case the software to crash.

Proper exception handling should be coupled to all inputs for the application. This will improve stability and reduce the frustration the users have with the application.

Criteria results

The **Worst case** would be to leave the software as is.

The **Planned** outcome would be to improve the software by not accepting the incorrect input.

The **best** outcome of the software would be to program proper exception handling with error messages directing the user to correct input values.

8. The first section of stringing charts could not be corrected once the next section was selected.

An oversight of previous versions of the software was that - the user would not mistakes in a early section of stringing charts before moving onto the next section. This fact could be very frustrating if a number of spans were input into the third section of a line and then having to restart input because there is no error correction mechanism to correct an error made in the first section.

Criteria results

The **Worst case** for the software would be to no change this oversight.

The **Planned** changes will allow the user to reset the software back a section to correct errors and recalculate the section separately without deleting spans already input.

Best Case: allow correction of errors without resetting any sections and recalculating all sections simultaneously.

9. Selected examples would hang the software for no apparent reason.

On occasion feedback will reveal special cases where the software hangs or gives incorrect solutions. These need to be checked against improved algorithms.

Criteria results

The **Worst case** outcome would be for the special cases to give the same results.

The planned outcome would be for the software is resolved for these examples.

The **best** outcome would be for the reasons for these problems to be found and be part of resolving the issues.

10. The user interface gave no indication of the critical span.

In line design, the conductor used is critical in terms of the span lengths required and the allowable clearance. Longer span lengths mean less structure and hardware and so this may be a necessary requirement of the designer up front. Indication of the critical span will allow the designer, up front, to determine what the limitations of the selected conductor are.

Criteria results

The **Worst case** result would be for there to be no changes made to the current software and have no critical span indication.

The **planned** result would be to have at least a critical span value for the conductor selected.

The **best** outcome would be for the critical span to be viewed in a graph of everyday tension verses span length.

B7 USE CASE ANALYSIS

The use case analysis of the new and old software needed to be tested in order to determine if there are improvements in speed and general intuitiveness of the new interface. A number of use cases were put together in order to fully compare the two packages. The speed in completing tasks by a user will indicate if the usability has in actual fact improved. The use cases were tested by competent and beginner users in a testing laboratory. The test cases are as follows:

1. Select a Mink conductor. Run a sat chart. Set the temperature limits to 0 to 30 with an increment of 5 degrees and the span length to 45 to 160 with an increment of 10 metre.

This will test the speed of use and the speed of the computation of the algorithms. The use case is a typical simple function that the application will have to perform repetitively.

2. With this task completed repeat the process but for Hare conductor. Do this task directly after the first task in order that the resulting time of changing to another conductor can be seen. This will be the case when two results are to be compared. Ideally this task should be quick such that the results from the first task are not forgotten before seeing the results of the second task.

3. The next test is intended to determine the mechanical stress imposed on the user. This stress is in the form of the number of clicks the user needs to make to complete functions. Count the number of mouse clicks to get from start to finish with both versions of software for case 1.

4. Another mechanical stress imposed on the user is the distance the mouse needs to travel to complete the task. Sum the distance required for the mouse to move to complete case 1 for both versions of software.

5. With the first algorithm tested the next algorithm is put to the test. A Comparison of the times needed to complete stringing charts for both Squirrel and Acacia conductors. The project data is as follows:

Section 1

Spans: 100, 150 and125 metres.

Section 2

Spans: 230, 150, 45 and 190 metres.

6. The final algorithm which gives the clearances to an object is the last of the algorithm speed tests. This is done by compare the times need to complete the clearance charts for Pine conductor and 35 conductor. The span consists of:

Pole A = 12 metre pole,

Pole B = 13 metre pole,

Object distance from A = 100 metres

Object height = 5 metres

Equivalent span = 200 metres

Temperature = 50 degrees.

7. The last test that could be completed on functionality that is common to both versions of software was the test of the help systems. The test was to compare the speeds of accessing the help for clearance charts.

APPENDIX C

IMPLEMENTATION

C1 THE USER INTERFACE

As can be seen in the software design section B5.3 the interface moved from a form based user interface to a tab style user interface. The advantage is that the user can change between algorithms, help, limits and graphs without reselecting a conductor or going though a number of forms. The conductor selection is done by means of two list boxes – one for the type of conductor and one for the specific conductor per type. Both boxes are populated with conductor types and conductors respectively. This means that the user can select a conductor immediately without filtering for type. The conductor list is extensive though and the filter will make the conductor list shorter and easier to traverse. The most popular conductors were listed at the top of the conductor listing such that filtering will not be necessary for common conductors. The selection boxes remain visible on the main form at all times such that the conductor can be changed at any time and algorithm results can be compared quickly.

C2 THE ALGORITHMS

In the Beta test version of RSAT, all algorithms were updated when a conductor or a limit was changed. This change was made such that if an algorithm tab was selected the updated results would already be reflected on the tab. This proved to slow the software down with all the calculation algorithms needing to be completed. Therefore this was changed to only calculate the algorithm for the tab that is visible to the user. On the selection of a new tab the algorithm for the respective tab was done and the tab updated before displaying the tab. The changes resulting from the process improved the software stability and time to complete updates of the visible algorithm.

The initial algorithms used to calculate the sag and tension charts, stringing charts and clearance calculator were derived based on the literature survey and past versions of RSAT. The algorithms were then compared with the results given for PLSCADD (the large line design package used by Eskom). Based on the comparison the algorithms were tweaked and adjusted to give results that correlate closely with PLSCADD.

C3 DATA FILES

RSAT requires data for conductors and the help file. While redesigning the software it was discovered that two extra data sources were required for the default limits and the default ranges. In the old version of RSAT the limits were not listed in particular. The only evidence that these limits were being used was on sag and tension sheets were the limit results were given for each span. The feedback form users was that the limits should be available to view and change for special environments with special conditions. It was decided to accomplish this by including a tab for the limits. This limits tab includes:

- the minimum, maximum and increment for span ranges,
- the minimum, maximum and increment for temperature ranges,
- three standard limiting criteria with tension, temperature, wind values and whether the attributes relate to the initial or final results and
- the C value.

In beta testing it was also suggested that a default button, that recalls defaults after changes have been made, would be useful. This was then implemented into the software. In final testing a user requested that the defaults be changeable. In order to do this the limits could not be stored inside the source code but had to be called from a source that the user can see and update. Therefore it was decided that the limits and ranges should be stored in a data file. (C values are already stored as part of the conductor data files.)

In the previous version of RSAT the data files were simple text files that were loaded into the software by an algorithm in the source code. This method of storing the data proved cumbersome when the amount of data increased. For this reason the conductor data was split into different type files to reduce this complexity. The files were still not ideal and so when the new version was researched it was decided to change the file format to XML standard file format to store the data. XML files are specifically designed to store data and can be viewed as tables if the format is correct. Therefore the XML file system proved to be a far better method store the data for RSAT. The data files RSAT utilises are:

- Conductors.xml;
- Help.xml;
- Limits.xml and
- Ranges.xml

C4 THE SET-UP PROGRAM

The previous compiler chosen to write RSAT (Borland C++ Builder) did not provide a set-up file programming facility. Therefore this feature had to be programmed in C++. The set-up program was designed such that illegal copying of the program could be prevented. In order to accomplish this, a password facility was included. The system renewed the password every time the setup program was run. User feedback on the setup showed the setup to be cumbersome and not very usable.

The software and supporting files for RSAT 4.2 were kept in a zip file. The zip file was not easily recognised by users by the fact that the windows operating system file manager explorer was upgraded to view inside zip files as though they are normal folders. Further confusing the user is the fact that explorer was also upgraded to hide the file extensions. This hid the fact that the executable file and its supporting data files were still compressed. Thus when the user ran the RSAT executable file, without extracting the files through the setup facility, an error message would appear saying that the conductor data file could not be found.

The conductor data file is the first file to be loaded into the desktop interface of the software and it was still zipped if this message appeared. Hence the setup file was not intuitive and needed to be upgraded.

The design and supply philosophy for the RSAT software has changed since the RSAT 4.2. The philosophy being supported by the Eskom business is to line up with the standardisation drive of the utility. This is to ensure standardization of the designs being employed in the field. To achieve this, the working group driving the project decided to remove all boundaries from use of the software. This included the removal of the password protection and sale of the RSAT software. The philosophy is that if the software is used, the cost saving in getting good, well designed lines, will save far more than could be charged for the software.

The new compiler (Microsoft.Net) chosen to write RSAT incorporated a setup file writer making the design and programming easier. This allowed the files to be compressed and decompressed easily in the background while a standard installation “front-end” guides the user. The standard setup interface is the same as used by most other software available. This made for a more intuitive process.

CROSS PLATFORM

A factor that could lead to inaccuracies and possible omission of sag and tension data is the fact that tension charts needed to be printed out before going to sight. This presents a problem, in that sight conditions are not known at time of printout. For this reason it was suggested that software be made available for hand held devices. Although this was not in the scope of this research it had to factored into the choice in the compiler and language that were chosen for the software.

To see if the implementation of the changes improves the software it is needed to be tested in line the criteria setup earlier.

APPENDIX D

TEST RESULTS

Testing took place in controlled and uncontrolled environments. The controlled test was done in a laboratory type test with the set out test plan. Experienced and beginner users were asked to test the software to determine if the software had improved. This meant that testing needed to be completed on both the new and old software so the comparison on the same tasks could be completed. While the users were doing testing on the software, they were observed for natural tendencies, mouse movements, interpretation of instructions and errors that could be resolved by improving the software. After the testing the user was questioned with respect to the software and its ease of use. Comments and suggestions were collected for possible updates to the software.

D.1 CONTROLLED TESTING

The controlled testing was done with the design criteria given in B6. First some simple requirements needed to be met in the new software that were not met in the old software. Then the performance was checked between new and old software. The results are as follows:

1. The setup process

The planned case: was achieved. The installation of the .Net framework will be required for the software to run. This was incorporated into the software but the setup needed to run twice to the install the framework and RSAT. However, the guidance on the setup interface does explain all that the user is required to do in terms of the framework and the installation of the software. If only one file needed to be run and all else was completed without the users intervention then the setup process would have be to the **best** possible outcome.

2. Conductor selection.

The best result would have to have had a dual view with two conduct results displayed. This was not possible with the size requirements for the user to see result effectively. It was decided instead to design an interface where a number of conductors can be compared for any algorithm consecutively. The process was made quick such that the user would remember the results from the last conductor or the conductors could be switched back and forth to compare. Else the two conductors could be printed and compared. Therefore the **planned result** was achieved.

3. Conductor data display.

The best case result required that the selected conductor and its respective data remains on screen at all times. This was achieved in the new software.

4. Sequential selection process.

The software has been upgraded to display conductor select boxes on screen at all times. This allows for selection without opening new windows sequentially. The upgrade was even fine tuned after feedback to allow one touch selection without selecting the conductor type first. This constitutes the **best case** result.

5. Addition of new conductors.

A conductor addition facility was added to the software that lets the user update the conductor data files with new conductors of a type already saved or the user can define a new type. This outcome represents the **best case result.**

6. Limiting criteria

The new software includes a facility that allows limits to be changed per task and the defaults limits to be revised or restored. This outcome represents the **best case result.**

7. Incorrect input causing software to crash.

An exception handling function of C# was discovered in the reprogramming. This was used to avoid having the software crash when incorrect input was fed into the software. In addition an error messaging system was added to inform the user of the error and possible remedies. This outcome represents the **best case result.**

8. String chart correction.

The software upgrade allows correction of errors in previous sections without resetting any sections and recalculates the all section each time "calculate" is pressed. This outcome represents the **best case result.**

9. Selected examples hang.

The software resolved all known examples that hung RSAT 4.2. However, the reasons for the problems on the old software were not discovered. Therefore this result represents only a **planned case** result.

10. The user interface gave no indication of the critical span.

A critical span graph was added with everyday tension verses span length that shows the critical spans. This outcome represents the **best case result.**

Now the results of speed test tasks comparing old and new software are listed:

11. The production of a sat chart for a Mink conductor with particular limits.
The result for the two applications is:

RSAT Version	Competence	
	Higher	Lower
4.2	43 seconds	65 seconds
5	37 seconds	65 seconds

An average improvement in time to complete the task of getting a Mink conductor sag and tension chart was recorded as 7 %. This is not a vast improvement and in particular new users of lower competence had no improvement. This is indicative of the general misunderstanding of software, engineering principles and of the designers misunderstanding of what the user expects to the procedure for completing a task. It should be noted though that this function of RSAT is the simplest. Small changes to improve the software did improve the time of the competent users and therefore usability was improved even if it was only for a section of the users.

12. The next test was to compare the sag and tension results of Mink conductor (completed in the first test) with the results for Hare conductor. The results are as follows:

RSAT Version	Competence	
	Higher	Lower
4.2	8 seconds	8 seconds
5	3 seconds	8 seconds

The time improvement is 63 % for a competent user but again there was no change for a beginner type user. It is easier to do the actual comparison with RSAT 5 since the sag and tension data for Mink was only removed to replace with Hare. This meant that the

results could be viewed back and forth quickly without needing a good memory or a printout of the results.
For RSAT 4.2 the data is removed for the view such that the conductor selection form can be displayed. After selection a conductor data sheet is displayed and then only is the sag and tension data displayed. Therefore a printout or a good memory would be required to compare the result of the two conductors.

13. The number of mouse clicks required to complete the sag and tension charts for Mink by the different competence users will be different as the beginner user may get to a solution via a longer route due to misunderstanding of simplest methods to get to the solution. The results reflect this clearly:

RSAT Version	Competence	
	Higher	Lower
4.2	10 clicks	18 clicks
5	5 clicks	11 clicks

The number of clicks required to produce a sag and tension chart improved by 50 % for the competent user. First time users showed an improvement of only 39 %. However, if looked at in terms of actual numbers the improvement for lower competence users was 7 clicks while for higher competence users it was only 5 clicks. The figures show a greater numerical improvement form the beginner users. These improvements reflect an improvement in mechanical efficiency and in turn the usability of the software.

14. Another means to test the mechanical efficiency of the software is to compare the sum of the distances the mouse is required to move to complete a simple task such as producing a sag ad tension chart for Mink. The test will be completed for both versions of software by noting the minimum distances possible. The results will not be taken for the two different competencies as the laboratory did not have facilities the tracked distance on screen. Therefore the minimum on screen distance was measured and short cuts and tab keys are used where possible to reduce the need to move the mouse. The results for minimum distance between the two versions of RSAT are:

RSAT Version	Competence Higher/Lower
4.2	750 mm
5	130 mm

An improvement of 83 % on the original distance is recorded. However, a general comment would be that for lower competence users movements of the mouse would be further than that for a higher competence user. This result is probably an inflated result and a result in 70's would probably be more realistic. This still represents a good improvement over the old software.

16. The next algorithm to be tested was the times to complete stringing charts. This algorithm is more complex than that of the sag and tension charts. It requires more input from the user. Therefore the result are expected to be improved for the beginner users:

RSAT Version	Competence	
	Higher	Lower
4.2	90 seconds	230 seconds
5	50 seconds	97 seconds

The suspicions were realised in the results as the beginner user results reflect the complexity of the algorithm in the 230 seconds required to complete the task in the old version of the software. The improvements for high and low competency are 45 % and 57 % respectively. This shows the improvement in the software is substantial. The results were taken when the input was correctly input as there were many attempts which had to be restarted by the lower competency users. The process required for RSAT 4.2, to compare the results of the first conductor with that of the second, was to restart the program to get the second conductors results, as the stringing chart did not reset.

To inexperienced users this may not be obvious and the time taken increased due to this misunderstanding. The reset button added to the new revision proved invaluable in making the software more usable.

16. The last of the algorithms still needed to be tested by comparing the times need to calculate the clearance charts for Pine and 35 conductors for a particular span. Being a complex algorithm it is expected the results between new and old versions of the software should liberate difference in completion times. The results are as follows:

RSAT Version	Competence	
	Higher	Lower
4.2	50 seconds	114 seconds
5	30 seconds	86 seconds

Again the improvements in performance between the versions are evident. The resulting improvements in time were 40 % and 24 % for higher and lower competencies respectively. The importance of this test was not only the times but the test showed a problem with the interface. When a new conductor was selected the results were not updated until an attribute on the clearance chart was changed. This meant that the user had to change the conductor and a clearance attribute even if the same clearance case was required. This did not make for easy comparison of results between conductors. The problem was resolved and further improved times by ten seconds. This made the final improvements 60 % and 33 % for high and low competencies respectively.

17. The speed of accessing the help may prove the software capable of holding the users interest in using the product or not. The test involved accessing the same help function within the two different versions of software. The results were:

RSAT Version	Competence	
	Higher	Lower
4.2	5 seconds	9
5	4 seconds	9

Improvement = 10 %

This result was shows that there very few changes made to the help system as it has never been the cause for comment or concern. Hence the same type of help system is run on both pieces of software. The only improvement that may have been gained in time would be due to the improved access to the help function selection. In hind sight this test may have been omitted had the changes been known before the tests were set-up.

Overall the improvement for competent users was better than for first time users in laboratory testing. The laboratory did prove useful in exposing errors in assumptions as to what the changes needed to be in ordered to improve the first time users' experience of the software.

The software has improved in areas allowing users to reset stringing calculation tab. Not allowing the user to reset this calculation would mean when new data or a new chart wanted to be entered the software would need to be restarted.

Then function buttons, fonts and tabs titles were increased in size to improve their visibility to users. This was done after it was noticed that the size was not legible to all users completing the testing.

It was found that lower competency users in general needed guidance through the process until the basics of the software were learnt. Since the software is basic and can be learnt fairly easily a “Suggested action” help window to guide users to next possible action was added. This window prompts the user as to possible next steps to take until the user become familiar with the software.

The event capture functions were updated to do calculations automatically for the algorithm visible. However, it was also noticed that if automatic update occurred before the input was completed, it could cause instability of the software. For example some input, if completed incorrectly, could send the software into an extremely long calculation that was not required. This could be seen when editing a maximum span length from 300 metres to 100 metres. Inputting the 100 metres before deleting the 300 metres may lead to the software calculating up to a span of 100300 metres. If the increment for calculation is 20 metres and the starting span 20 metres then the number spans is 5015. If the temperature range is -5 to 60 degrees with an increment of 5 degrees. Then the number of temperature calculations (including the worst calculation) is 15. Now the number of calculations is the number of temperatures times the spans which gives 75225 calculations. Not to mention that to get to each result involved a number of iterations of its own.

D2 UNCONTROLLED TESTING

The testing is uncontrolled because it was not completed in a laboratory. The test was completed in form of a Beta release being made available to users via the Distribution Technology website. This meant that any user of the website could down load the software and use it freely. A question and comments return e-mail and contact number for the developer was left on the website for feedback. The Beta test version was available for eight months with user feedback coming via e-mail and telephonically. The results from the Beta testing showed the new software had achieved the goal of becoming more user-friendly. There were 2 major feedback items related to usability of the software. The first was that the error/ exception handling could be improved to guide the user away from making mistakes when inserting data. The resolution to this comment has already been discussed and was accepted by users.

The second comment related to the size of the form and the size menus, conductor boxes, chart windows and all the items presented for the user to see. This was a repeat of the feedback from the laboratory testing. Fonts and selection option had all been increased in size after the user feedback in the laboratory testing. Further prompting during the Beta testing was that graphics were too small at high resolutions. The software was revised to resize the software's desk top to the size of the main form.

Other comments were related to small errors in the software that were overlooked when designing the Beta test version. The rest of the comments confirmed that the users were happy with the new software.

APPENDIX E

COMPLETE SOURCE CODE

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Xml;
using System.Xml.XPath;
using System.Xml.Schema;
using System.Xml.Xsl;
using System.IO;
/*using CrystalDecisions.Shared;*/



namespace RsatDraft0
{
    /// <summary>
    /// 1 Dec 2005 met with Prof Dwol. Started XML data file comms and conductor class usage.
    /// Rev A 6 Dec 2005 got the xml to first drop down correct.
    /// Rev B 7 Dec start work on down loading conductors data.
    /// Rev B Complete downloading type and conductor into two list boxes
    /// Rev C Replaced data lists with text boxes such that input will be easier when adding conductors.Load data into text boxes.
    /// 20 Dec Added update SAT Chart to"private void CLB_SelectedIndexChanged(object sender, System.EventArgs e)"
    /// 3 Jan Added method for Sat calcs to conductor class.
    ///   </summary>

    public class MainView : System.Windows.Forms.Form
    {
        int line = 0;
        int page = 1;
```

```
private System.Windows.Forms.Label label1;
private System.Windows.Forms.MainMenu mainMenu1;
private System.Windows.Forms.MenuItem menuItem1;
private System.Windows.Forms.MenuItem menuItem3;
private System.Windows.Forms.MenuItem menuItem4;
private System.Windows.Forms.MenuItem menuItem9;
private System.Windows.Forms.MenuItem menuItem10;
private System.Windows.Forms.MenuItem menuItem11;
private System.Windows.Forms.MenuItem menuItem14;
private System.Windows.Forms.MenuItem menuItem15;
private System.Windows.Forms.MenuItem menuItem18;
private System.Windows.Forms.MenuItem menuItem21;
private System.Windows.Forms.MenuItem menuItem29;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.StatusBar statusBar1;
private System.Windows.Forms.ToolBar toolBar1;
private System.Windows.Forms.ImageList imageList1;
private System.Windows.Forms.ToolBarButton toolBarButton2;
private System.Windows.Forms.ToolBarButton toolBarButton3;
private System.Windows.Forms.ToolBarButton toolBarButton4;
private System.Windows.Forms.ToolBarButton toolBarButton6;
private System.Windows.Forms.ToolBarButton toolBarButton7;
private System.Windows.Forms.ToolBarButton toolBarButton8;
private System.Windows.Forms.TabControl tabControl1;
private System.Windows.Forms.TabPage tabPage1;
private System.Windows.Forms.TabPage tabPage2;
private System.Windows.Forms.TabPage tabPage3;
private System.Windows.Forms.TabPage tabPage4;
private System.Windows.Forms.DataGrid dataGrid1;
```

```
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.Label label15;
private System.Windows.Forms.Label label16;
private System.Windows.Forms.TabPage tabPage5;
private System.Windows.Forms.Label label19;
private System.Windows.Forms.Label label20;
internal System.Data.DataSet SagAndTension;
private System.Data.DataTable dataTable1;
private System.Data.DataColumn dataColumn1;
private System.Data.DataColumn dataColumn2;
private System.Data.DataColumn dataColumn3;
private System.Data.DataColumn dataColumn4;
private System.Data.DataColumn dataColumn5;
private System.Data.DataColumn dataColumn6;
private System.Data.DataView dataView1;
private System.Windows.Forms.ListBox CTLB;
private System.Windows.Forms.ListBox CLB;
private System.Windows.Forms.PictureBox Pict;
private System.Windows.Forms.TextBox Area;
private System.Windows.Forms.TextBox IMod;
private System.Windows.Forms.TextBox FMod;
private System.Windows.Forms.TextBox Mass;
private System.Windows.Forms.TextBox UTS;
private System.Windows.Forms.TextBox EDT;
private System.Windows.Forms.TextBox Diameter;
private System.Windows.Forms.Label label28;
```

```
private System.Windows.Forms.Label label29;
private System.Windows.Forms.Label label30;
private System.Windows.Forms.Label label32;
private System.Windows.Forms.Label label34;
private System.Windows.Forms.Label label35;
private System.Windows.Forms.Label label39;
private System.Windows.Forms.TextBox Coefficient;
private System.Windows.Forms.TabPage tabPage6;
private System.Windows.Forms.TextBox tmin;
private System.Windows.Forms.Label label33;
private System.Windows.Forms.Label label40;
private System.Windows.Forms.Label label41;
private System.Windows.Forms.Label label42;
private System.Windows.Forms.Label label43;
private System.Windows.Forms.Label label44;
private System.Windows.Forms.Label label45;
private System.Windows.Forms.Label label46;
private System.Windows.Forms.TextBox tinc;
private System.Windows.Forms.TextBox linc;
private System.Windows.Forms.TextBox lmax;
private System.Windows.Forms.TextBox lmin;
private System.Windows.Forms.Label label47;
private System.Windows.Forms.Label label48;
private System.Windows.Forms.Label label50;
private System.Windows.Forms.Label label49;
private System.Windows.Forms.Label label51;
private System.Windows.Forms.Label label52;
private System.Windows.Forms.Label label53;
private System.Windows.Forms.Label label54;
private System.Windows.Forms.Label label55;
private System.Windows.Forms.Label label56;
```

```
private System.Windows.Forms.Label label57;
private System.Windows.Forms.Label label58;
private System.Windows.Forms.Label label59;
private System.Windows.Forms.Label label60;
private System.Windows.Forms.Label label61;
private System.Windows.Forms.Label label62;
private System.Windows.Forms.TextBox tmax;
private System.Windows.Forms.Label label63;
private System.Windows.Forms.Label label64;
private System.Windows.Forms.Label label65;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.TextBox Creep;
private System.Windows.Forms.PictureBox picture;
private System.Windows.Forms.TextBox ntype;
private System.Windows.Forms.TextBox ncond;
private System.Windows.Forms.Button Save;
private System.Windows.Forms.Button Cancel;
private System.Windows.Forms.OpenFileDialog openFileDialog1;
private System.Windows.Forms.TextBox image;
private System.Windows.Forms.SaveFileDialog saveFileDialog1;
private System.Windows.Forms.Label Clabel;
private System.Windows.Forms.TextBox ncreep;
private System.Windows.Forms.Label label73;
private System.Windows.Forms.Label label74;
private System.Windows.Forms.Label label75;
private System.Windows.Forms.Label label76;
private System.Windows.Forms.Label label77;
private System.Windows.Forms.Label label78;
private System.Windows.Forms.Label label79;
private System.Windows.Forms.Label label80;
private System.Windows.Forms.Label label81;
```

```
private System.Windows.Forms.TabPage tabPage7;
private System.Windows.Forms.MenuItem menuItem16;
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.Label label82;
private System.Windows.Forms.Label label83;
private System.Windows.Forms.Label label84;
private System.Windows.Forms.Label label85;
private System.Windows.Forms.Label label86;
private System.Windows.Forms.TextBox Ww1;
private System.Windows.Forms.TextBox Wt1;
private System.Windows.Forms.TextBox Wc1;
private System.Windows.Forms.TextBox CValue;
private System.Windows.Forms.TextBox Ww3;
private System.Windows.Forms.TextBox Wc3;
private System.Windows.Forms.TextBox Wt3;
private System.Windows.Forms.TextBox Ww2;
private System.Windows.Forms.TextBox Wc2;
private System.Windows.Forms.TextBox Wt2;
private System.Windows.Forms.ComboBox Ini1;
private System.Windows.Forms.ComboBox Ini3;
private System.Windows.Forms.ComboBox Ini2;
private System.Windows.Forms.ComboBox CT2;
private System.Windows.Forms.ComboBox CT1;
private System.Windows.Forms.DataGrid dataGrid2;
private System.Data.DataTable dataTable2;
private System.Data.DataColumn dataColumn7;
private System.Data.DataColumn dataColumn8;
private System.Data.DataColumn dataColumn9;
private System.Data.DataColumn dataColumn10;
private System.Data.DataColumn dataColumn11;
private System.Data.DataColumn dataColumn12;
```

```
private System.Data.DataColumn dataColumn13;
private System.Data.DataColumn dataColumn14;
private System.Data.DataColumn dataColumn15;
private System.Windows.Forms.TextBox polef;
private System.Windows.Forms.TextBox poles;
private System.Windows.Forms.TextBox section;
private System.Windows.Forms.TextBox project;
private System.Windows.Forms.TextBox polep;
private System.Windows.Forms.Button calculate;
private System.Windows.Forms.Button nsection;
private System.Data.DataColumn dataColumn16;
private System.Windows.Forms.PictureBox pictureBox2;
private System.Windows.Forms.Label label87;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.Label label17;
private System.Windows.Forms.Label label99;
private System.Windows.Forms.TextBox condtemp;
private System.Windows.Forms.TextBox Tension;
private System.Windows.Forms.TextBox AttachA;
private System.Windows.Forms.TextBox GroundA;
private System.Windows.Forms.TextBox GroundO;
private System.Windows.Forms.TextBox GroundDist;
private System.Windows.Forms.TextBox HeightO;
private System.Windows.Forms.TextBox AttachB;
private System.Windows.Forms.TextBox GroundB;
private System.Windows.Forms.TextBox ESpan;
private System.Windows.Forms.ComboBox IF;
private System.Windows.Forms.TextBox Worst;
private System.Windows.Forms.TextBox nclearance;
private System.Windows.Forms.TextBox DWorst;
private System.Windows.Forms.PictureBox pictureBox3;
```

```
private System.Windows.Forms.Label label18;
private System.Windows.Forms.Label label21;
private System.Windows.Forms.Label TWorst;
private System.Windows.Forms.RichTextBox Helptext;
private System.Windows.Forms.ListBox helpindex;
private System.Windows.Forms.Label label24;
private System.Windows.Forms.Label label25;
private System.Windows.Forms.Label label88;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.Label label89;
private System.Windows.Forms.Label label90;
private System.Windows.Forms.Label label91;
private System.Windows.Forms.Label label93;
private System.Windows.Forms.Label label95;
private System.Windows.Forms.Label label96;
private System.Windows.Forms.Label label97;
private System.Windows.Forms.Label label22;
private System.Windows.Forms.Label label92;
private System.Windows.Forms.Label label94;
private System.Windows.Forms.Label label23;
private System.Windows.Forms.Label label98;
private System.Windows.Forms.Label label102;
private System.Windows.Forms.Label label104;
private System.Windows.Forms.Label label105;
private System.Windows.Forms.Label label106;
private System.Windows.Forms.Label label107;
private System.Windows.Forms.Label label108;
private System.Windows.Forms.Label label109;
private System.Windows.Forms.Label label27;
private System.Windows.Forms.ComboBox CT3;
```

```
    private System.Windows.Forms.RichTextBox richTextBox1;
    private System.Windows.Forms.MenuItem PrintPreview;
    private System.Windows.Forms.MenuItem Print;
    private System.Windows.Forms.PrintPreviewDialog printPreviewDialog1;
    private System.Drawing.Printing.PrintDocument printDocument1;
    private System.Windows.Forms.Label label26;
    private System.Windows.Forms.Label label36;
    private System.Windows.Forms.Label label66;
    private System.Windows.Forms.TextBox TensionA;
    private System.Windows.Forms.Label label67;
    private System.Windows.Forms.TextBox TensionB;
    private System.Windows.Forms.Label Error;
    private System.Windows.Forms.Label LimError;
/*  private CrystalDecisions.CrystalReports.Engine.ReportDocument Report;*/
    //private System.Windows.Forms.RichTextBox clipboard;
    private System.Windows.Forms.Button Defaults;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Label label31;
    private System.Windows.Forms.Label emsg;
    private System.Windows.Forms.RichTextBox clipboard;
    private System.Windows.Forms.Label label37;
    private System.Windows.Forms.Panel panel1;
    private System.Windows.Forms.Label label68;
    private System.Windows.Forms.Label suggest;
    private System.ComponentModel.IContainer components;



    public MainView()
    {
        //
```

```
// Required for Windows Form Designer support
//
InitializeComponent();

//
// TODO: Add any constructor code after InitializeComponent call
// Open the Conductor Xml file and reader the conductor types in the type box
XmlTextReader cr = new XmlTextReader(@"Conductors.xml");//Open a reader
cr.WhitespaceHandling = WhitespaceHandling.None;// White space handling

if (cr.Read())
{
    // Moves the reader to the 1st node.
    cr.MoveToContent();

    // Moves the reader to the next node.
    cr.MoveToContent();
    //string to type

    while(cr.Read())
    {

        // Load types
        if(cr.Name == "Type")// loop to the types specified until the end of the file
        {
            cr.Read();;// move to the type names
            while(CTLB.FindString(cr.Value)==-1 && cr.Value!="")
                this.CTLB.Items.Add(cr.Value);
        // Load conductors
                cr.Read();cr.Read();cr.Read();// move to the names
```

```
                    //Load conductors
                        this.CLB.Items.Add(cr.Value);
            }

        }

        cr.Close();
}
// Open the Help Xml file and reader the Help index in the helpindex box
XmlTextReader hr = new XmlTextReader(@"Help.xml");//Open a reader
hr.WhitespaceHandling = WhitespaceHandling.None;// White space handling

if (hr.Read())
{
        // Moves the reader to the 1st node.
        hr.MoveToContent();

        // Moves the reader to the next node.
        hr.MoveToContent();
        //string to type

        while(hr.Read())
        {

            // Load types
            if(hr.Name == "L0")// loop to the types specified until the end of the file
            {
                hr.Read();;// move to the type names
                while(helpindex.FindString(hr.Value)==-1 && hr.Value!="")
                    this.helpindex.Items.Add(hr.Value);
            }
```

```
                }

                hr.Close();
            }
            ResetStringing();
        }
        public void ResetStringing()//resets the string chart if a new project is to be strung
        {

        try
         {
            Error.Text="";
            int Poles = Convert.ToInt16(poles.Text);
            int Polef = Convert.ToInt16(polef.Text);
            dataTable2.Rows.Clear();
            if(dataTable2.Rows.Count>0)
            {
                while(dataTable2.Rows[dataTable2.Rows.Count-1]["5 degrees"].ToString()!="----------")
                {dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);}
                dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);

            }
            int count = 0;
            if(Polef>Poles)
            {
                dataTable2.Rows.Add(new Object[] {"----------","----------","----------","
```

```
                    dataTable2.Rows.Add(new Object[] {polep.Text+(count + Poles)+"-"+(count + Poles+1), "Input span", "", "", "",
"","","","
```

```
}
        public void DefaultLimits()// Resets the limits on the limit page to the efault values - default values are stored in the limits XML
file.
        {
            //Read default limit data from Limits XML file
            XmlTextReader lt = new XmlTextReader(@"Limits.xml");//Open a reader
            lt.WhitespaceHandling = WhitespaceHandling.None;// White space handling

            if (lt.Read())
            {
                // Moves the reader to the 1st node.
                lt.MoveToContent();

                // Moves the reader to the next node.
                lt.MoveToContent();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();
                // Load data
                Wt1.Text=lt.Value;
                lt.Read();lt.Read();lt.Read();
                CT1.Text=lt.Value;
                lt.Read();lt.Read();lt.Read();
                Wc1.Text=lt.Value;
                lt.Read();lt.Read();lt.Read();
                Ww1.Text=lt.Value;
                lt.Read();lt.Read();lt.Read();
                Ini1.Text=lt.Value;
                // Moves the reader to the next node.
                lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();
                // Load data
                Wt2.Text=lt.Value;
                lt.Read();lt.Read();lt.Read();
                CT2.Text=lt.Value;
```

```
        lt.Read();lt.Read();lt.Read();
        Wc2.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        Ww2.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        Ini2.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();lt.Read();
        // Load data
        Wt3.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        CT3.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        Wc3.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        Ww3.Text=lt.Value;
        lt.Read();lt.Read();lt.Read();
        Ini3.Text=lt.Value;
        lt.Close();
}
//Read default range data from Ranges XML file
        XmlTextReader rg = new XmlTextReader(@"Ranges.xml");//Open a reader
        rg.WhitespaceHandling = WhitespaceHandling.None;// White space handling

if (rg.Read())
{
        // Moves the reader to the 1st node.
        rg.MoveToContent();

        // Moves the reader to the next node.
        rg.MoveToContent();rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();
        // Load data
```

```
            tmin.Text=rg.Value;
            rg.Read();rg.Read();rg.Read();
            tmax.Text=rg.Value;
            rg.Read();rg.Read();rg.Read();
            tinc.Text=rg.Value;
            // Moves the reader to the next node.
            rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();rg.Read();
            // Load data
            lmin.Text=rg.Value;
            rg.Read();rg.Read();rg.Read();
            lmax.Text=rg.Value;
            rg.Read();rg.Read();rg.Read();
            linc.Text=rg.Value;
            rg.Close();

            }
    }
    public double ToInitial(double L,double t1,double w1,double c1,double t2,double A,double Coef,double Imod, double Fmod,
double Wc, double Diam, double T20, double L12, double k)//Convert to initial at convert temp.
        //t1 = temp 1, w1 = wind 1, c1 = tension 1, convert at temp t2
    {
        // convert final values to ref temp without wind
        double T1 = TempConvert(L,t1,w1,c1,t2,0,"Final",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

        // Creep convertion find initial at ref temp
        double Ti = T20;// initial guess
        double Tin = 0;
        int count = 0;
        while(Tin - Ti>0.1||Tin - Ti<-0.1)
        {
            Tin = Ti;
```

```
                Ti = (25/3*Wc*Wc*L12*T20*T20/k*Tin*Tin-
Math.Pow(T1,4)*Tin*Tin+25/3*Wc*Wc*L12*T20*T20/k*T1*T1)/(50/3*Wc*Wc*L12*T20*T20/k*Tin-2*Math.Pow(T1,4)*Tin);
                count++;
                if (Ti<0)
                    Ti = -0.5*Ti;
                if (count == 10000)
                    break;

            }

            return (Ti);
        }
        public double ToFinal(double L,double t1,double w1,double c1,double t2,double A,double Coef,double Imod, double Fmod,
double Wc, double Diam, double T20, double L12, double k) //Convert to final at convert temp.
            //t1 = temp 1, w1 = wind 1, c1 = tension 1, convert at temp t2
        {
            // convert initial values to 15 deg without wind
            double T = TempConvert(L,t1,w1,c1,t2,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

            // Creep convertion to final EDT at convert temp (def 15)

            //double Ti = Tfw;//initial tension as worked
            double Tfw = T20;//initial geuss
            double Tnw = 0;

            while(Tnw - Tfw>1||Tnw - Tfw<-1)
            {
                Tnw = Tfw;
                Tfw = (-25/3*Wc*Wc*L12*T20*T20/k*Tnw*Tnw-3*Math.Pow(Tnw,4)*T*T-25/3*Wc*Wc*L12*T20*T20/k*T*T)/(-
50/3*Wc*Wc*L12*T20*T20/k*Tnw-4*Math.Pow(Tnw,3)*T*T);
            }
```

```
            return (Tfw);

        }
        public double TempConvert(double L,double t1,double w1,double T1,double t2,double w2,string IF,double A,double Coef,double Imod, double Fmod, double Wc, double Diam, double T20, double L12, double k ) //Converts tension with change in temp.
            //Arguments:L = span length, t1 = temp 1, w1 = wind 1, T1 = tension 1, t2 = temp 2, w2 = wind 2, IF = initial or final calc, A = cond area, Coef = cond temp coef, Imod = initial modulus, Fmod = Final Modulus, Wc = cond weight, Daim = conductor diameter, T20 = 20 % of UTS tension, L12 = L1.2(required for calculations) and k = creep constant.
        {
            double w = Math.Pow(Wc,2)+Math.Pow(w1,2);
            double W1 = Math.Sqrt(w);
            w = Math.Pow(Wc,2)+Math.Pow(w2,2);
            double W2 = Math.Sqrt(w);
            double b=0;
            double c=0;
            if(IF=="Initial")
            {
                b = Imod*A*((Math.Pow(W1,2)*Math.Pow(L,2))/(24*Math.Pow(T1,2))+Coef*(t2 -t1))-T1;//with wind
                c = (Imod*A*Math.Pow(W2,2)*Math.Pow(L,2))/24;  // without wind
            }
            if(IF=="Final")
            {
                b = Fmod*A*((Math.Pow(W1,2)*Math.Pow(L,2))/(24*Math.Pow(T1,2))+Coef*(t2 -t1))-T1;//with wind
                c = (Fmod*A*Math.Pow(W2,2)*Math.Pow(L,2))/24;  // without wind
            }

            double Tnw = 100;
            double Tfw = Convert.ToDouble(UTS.Text)*0.4;// initial geuss

            while((Tnw-Tfw)>0.1||(Tnw-Tfw)<-0.1)
            {
```

```
            Tnw = Tfw;
            Tfw= Tnw - ((Math.Pow(Tnw,3) + (Math.Pow(Tnw,2)*b)-c)/(3*Math.Pow(Tnw,2)+2*Tnw*b));
        }
        return(Tfw);

    }
    public int findlimit(double L,double A,double Coef,double Imod, double Fmod, double Wc, double Diam, double T20, double
L12, double k ) //Determine which of the limits is the ruling limit producing the lowest tension at the every day conditions
    //Arguments:L = span length, A = cond area, Coef = cond temp coef, Imod = initial modulus, Fmod = Final Modulus, Wc = cond
weight, Daim = conductor diameter, T20 = 20 % of UTS tension, L12 = L1.2(required for calculations) and k = creep constant.
    {
        int worst = 0;// setup counter to hold worst case number
        int ct = 15;//conversion temp

        // Limits Default EDT C @ 15 deg Final
        // Limit 1.
        double wt1 = Convert.ToDouble(Wt1.Text);//weather temp 1
        double ww1 = Convert.ToDouble(Ww1.Text)*0.6*Diam;//weather weight 1
        double wc1=0;

        //convert weather c 1 from c or % uts to tension
        if(CT1.Text=="C Value")
            wc1 = Convert.ToDouble(Wc1.Text)*Wc;
        if(CT1.Text=="% UTS")
            wc1 = Convert.ToDouble(Wc1.Text)*Convert.ToDouble(UTS.Text)/100;
        if(CT1.Text=="Tension")
            wc1 = Convert.ToDouble(Wc1.Text);//weather c 1

        // get limits to common base 15 deg 0 Pa final
        double L1=0;
        if (Ini1.Text == "Initial")// Convert limit to final 15 deg 0 wind
```

```
            L1 = ToFinal(L,wt1,ww1,wc1,ct,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);// convert to final at convert temp ct default
15'C and zero wind etc.
        else
            L1 = TempConvert(L,wt1,ww1,wc1,15,0,"Final",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

        // Limit 2. Default: Final 40 % -5 700 pa
        double wt2 = Convert.ToDouble(Wt2.Text);//weather temp 2
        double ww2 = Convert.ToDouble(Ww2.Text)*0.6*Diam;//weather weight 2
        double wc2=0;
        //convert weather c 2 from c or % uts to tension
        if(CT2.Text=="C Value")
            wc2 = Convert.ToDouble(Wc2.Text)*Wc;
        if(CT2.Text=="% UTS")
            wc2 = Convert.ToDouble(Wc2.Text)*Convert.ToDouble(UTS.Text)/100;
        if(CT2.Text=="Tension")
            wc2 = Convert.ToDouble(Wc2.Text);//weather c 1
        // get limits to common base 15 deg 0 Pa final
        double L2=0;
        if (Ini2.Text == "Initial")// Convert limit to final 15 deg 0 wind
            L2 = ToFinal(L,wt2,ww2,wc2,ct,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);// convert to final at convert temp ct default
15'C and zero wind etc.

            // Extapolate limit to 15 deg with no wind final.
        else
            L2 = TempConvert(L,wt2,ww2,wc2,15,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

        // Limit 3. Default: Initial 40 % -5 700 pa
        double wt3 = Convert.ToDouble(Wt3.Text);//weather temp 3
        double ww3 = Convert.ToDouble(Ww3.Text)*0.6*Diam;//weather weight 3
        double wc3=0;
        //convert weather c 3 from c or % uts to tension
```

```
            if(CT3.Text=="C Value")
                wc3 = Convert.ToDouble(Wc3.Text)*Wc;
            if(CT3.Text=="% UTS")
                wc3 = Convert.ToDouble(Wc3.Text)*Convert.ToDouble(UTS.Text)/100;
            if(CT3.Text=="Tension")
                wc3 = Convert.ToDouble(Wc3.Text);//weather c 1
            // get limits to common base 15 deg 0 Pa final
            double L3=0;
            if (Ini3.Text == "Initial")// Convert limit to final 15 deg 0 wind
                L3 = ToFinal(L,wt3,ww3,wc3,ct,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);// convert to final at convert temp ct default
15'C and zero wind etc.
                // Extapolate limit to 15 deg with no wind final.
            else
                L3 = TempConvert(L,wt3,ww3,wc3,15,0,"Final",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);
            //Convert EDT to integer
            // Compare all limits at 15 final with no wind L1, L2 & L3
            if (L1<L2&&L1<L3)
            {worst = 1;EDT.Text = L1.ToString();}
            if (L2<L1&&L2<L3)
            {worst = 2;EDT.Text = L2.ToString();}
            if (L3<L2&&L3<L1)
            {worst = 3;EDT.Text = L3.ToString();}
            label28.Text="EDT at " + L + " m";
            //Convert EDT to integer
            EDT.Text = Convert.ToString(Math.Round(Convert.ToDouble(EDT.Text),0));

            return(worst);
        }
        public void equivspan()// Algorithm to determine the stringing charts output
        {
            try
```

```
{
    Error.Text="";

    // Convert Conductor data for easy use
    double A = Convert.ToDouble(Area.Text)/1000000;//convert into sq meters
    double Coef = (Convert.ToDouble(Coefficient.Text)*Math.Pow(10,-6)); //
    double Imod = Convert.ToDouble(IMod.Text)*Math.Pow(10,6); //
    double Fmod = (Convert.ToDouble(FMod.Text)*Math.Pow(10,6)); //
    double Wc = (Convert.ToDouble(Mass.Text)*9.80665/1000); //convert to m/N
    double Diam = (Convert.ToDouble(Diameter.Text)/1000); //convert to meters
    double T20 = 0.2*Convert.ToDouble(UTS.Text);
    double k = Convert.ToDouble(Creep.Text)/1000000; // Creep
    double L12 = 0;
    int Poles;
    int Polef;
    Poles= Convert.ToInt16(poles.Text);
    Polef= Convert.ToInt16(polef.Text);
    int sa=0;
    int count =1;
    int rcount = 1;
    double sb=0;
    double L = 0;
    bool flag = true;//setup flag on input spans
    // Check all spans are in and flag if not
    while (rcount<dataTable2.Rows.Count-1)
    {
        count=rcount;
        while(dataTable2.Rows[count+1]["Poles"].ToString()!="Section"&&dataTable2.Rows[count]["Span
Lengths"].ToString()!="")
        {
            if (dataTable2.Rows[count]["Span Lengths"].ToString()!="Input span")
```

```
                    count++;
                else
                {
                    flag = false;
                    count++;
                }
            }
            count = rcount;

            // get total section length and equivalent span
            if(flag == true)
            {
                sa = 0;sb = 0;
                while(dataTable2.Rows[count+1]["Poles"].ToString()!="Section"&&dataTable2.Rows[count]["Span
Lengths"].ToString()!="")
                {
                    sa = sa + Convert.ToInt16(dataTable2.Rows[count]["Span Lengths"]);
                    sb = sb + Math.Pow(Convert.ToInt16(dataTable2.Rows[count]["Span Lengths"]),3);
                    count++;
                }

                L = Math.Round(Math.Sqrt(sb/sa),1);//equiv span
                L12 = Math.Pow(L,1.2);
                // the edt for equivalent span
                findlimit(L,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k );
                // convert to initial
                double iedt = ToInitial(L,15,0,Convert.ToDouble(EDT.Text),15,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

                // output sags
                count = rcount;
                while(dataTable2.Rows[count+1]["Poles"].ToString()!="Section")
```

```
                {
                    dataTable2.Rows[count]["5 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,5,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100;
                    dataTable2.Rows[count]["10 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,10,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100 ;
                    dataTable2.Rows[count]["15 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span Lengths"]),2)/(8*iedt)*100)/100;
                    dataTable2.Rows[count]["20 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,20,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100 ;
                    dataTable2.Rows[count]["25 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,25,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100 ;
                    dataTable2.Rows[count]["30 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,30,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100 ;
                    dataTable2.Rows[count]["35 degrees"]=
Math.Ceiling(Wc*Math.Pow(Convert.ToDouble(dataTable2.Rows[count]["Span
Lengths"]),2)/(8*TempConvert(L,15,0,iedt,35,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k))*100)/100;

                    count++;
                }
                // output tensions
                dataTable2.Rows[count + 1]["5 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,5,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100;
                dataTable2.Rows[count + 1]["10 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,10,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100;
                dataTable2.Rows[count + 1]["15 degrees"]= Math.Ceiling(iedt*100)/100;
```

```
                    dataTable2.Rows[count + 1]["20 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,20,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100 ;
                    dataTable2.Rows[count + 1]["25 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,25,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100 ;
                    dataTable2.Rows[count + 1]["30 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,30,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100 ;
                    dataTable2.Rows[count + 1]["35 degrees"]=
Math.Ceiling(TempConvert(L,15,0,iedt,35,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k)*100)/100 ;
                    //

                    // OUTPUT SECTION DETAIL
                    dataTable2.Rows[count + 3]["Poles"]= project.Text;
                    dataTable2.Rows[count + 3]["Span lengths"]= " Section - ";
                    dataTable2.Rows[count + 3]["5 degrees"]= "Spans = ";
                    dataTable2.Rows[count + 3]["10 degrees"]= Polef-Poles;
                    dataTable2.Rows[count + 3]["15 degrees"]= "Equiv span = ";
                    dataTable2.Rows[count + 3]["20 degrees"]= "      " + L;
                    dataTable2.Rows[count + 3]["25 degrees"]= "Length = ";
                    dataTable2.Rows[count + 3]["30 degrees"]= sa ;

                }count= count +5;rcount = count;
            }
            count = dataTable2.Rows.Count;
            rcount = 0;
        }
        catch
        {
            Error.Text = "Input error.";
        }
    }
```

```
        public void stringupdate() //update sag and tension chart when changes to any project input is made for the current section is
made.
        {
            try
            {
                Error.Text="";
                Convert.ToInt16(poles.Text);
                Convert.ToInt16(polef.Text);

                if(dataTable2.Rows.Count>0)
                {
                    while(dataTable2.Rows[dataTable2.Rows.Count-1]["5 degrees"].ToString()!="----------------------------------------
--")
                    {dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);}
                    dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);

                }
                int Poles= Convert.ToInt16(poles.Text);
                int Polef= Convert.ToInt16(polef.Text);
                int count = 0;
                if(Polef>Poles)
                {
                    dataTable2.Rows.Add(new Object[] {"----------","----------
-----","----------","----------","----------","----
----------","----------","----------","----------
----------","----------"});
                    while (count<(Polef-Poles))
                    {
                        dataTable2.Rows.Add(new Object[] {polep.Text+(count + Poles)+"-"+(count + Poles+1), "Input span", "", "", "",
"","","","
```

```
                    }
                }
                else
                {
                    dataTable2.Rows.Add(new Object[] {"-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------"});
                    while (count<(Poles-Polef))
                    {
                        dataTable2.Rows.Add(new Object[] {polep.Text+(Poles - count)+"-"+(Poles - count - 1), "Input span", "", "", "", "","","","","",""});
                        count++;
                    }
                }

                dataTable2.Rows.Add(new Object[] {"","","","","
```

```
try
  {
      Graphics g = pictureBox2.CreateGraphics();

      g.Clear(System.Drawing.Color.White);
      // Convert Conductor data for easy use
      double A = Convert.ToDouble(Area.Text)/1000000;//convert into sq meters
      double Coef = (Convert.ToDouble(Coefficient.Text)*Math.Pow(10,-6)); //
      double Imod = Convert.ToDouble(IMod.Text)*Math.Pow(10,6); //
      double Fmod = (Convert.ToDouble(FMod.Text)*Math.Pow(10,6)); //
      double Wc = (Convert.ToDouble(Mass.Text)*9.80665/1000); //convert to m/N
      double Diam = (Convert.ToDouble(Diameter.Text)/1000); //convert to meters
      double T20 = 0.2*Convert.ToDouble(UTS.Text);
      double k = Convert.ToDouble(Creep.Text)/1000000; // Creep

      //y max - 60, 20 % = 310
      int maxy = pictureBox2.Height + 60;
      // setup labels
      int H20 = Convert.ToInt32(maxy - maxy*0.5);
      label10.Top = H20;
      int H40 = 0;
      label9.Top = H40;


      //x 100 - 900 0m = 100, 50m = 250, 100m = 400, 150m = 550, 200m = 700, 250m = 850, 300m = 1000, 350m = 1150, 400m
= 1300, 450m = 1450, 500m = 1600

      int y0 = maxy;//0%
      int y1 = 0;//0%
      int x = 0;// 1 m
      int maxs = 1000;// max span lengths
```

```
    int incr = 1; // increment
    if (CTLB.Text=="LV Bundle")// LV Bundle does not work with same accuracy and does not get strung so far.
        if (CLB.Text=="35 ABC")
        {x = 0; maxs= 200;incr = 10;}
        else
        {x = 0; maxs= 200;incr = 10;}

    while(x < maxs)
    {
        findlimit(x,A,Coef,Imod,Fmod,Wc,Diam,T20,Math.Pow(x,1.2),k);
        y1 = Convert.ToInt32(Math.Ceiling(maxy - maxy*0.5/0.2*Convert.ToInt32(EDT.Text)/Convert.ToInt32(UTS.Text)));
        Point []pointArray3 = {new Point(2*(x-1),y0),new Point(2*(x-1+incr),y1)};
        y0 =y1;

        g.DrawCurve(System.Drawing.Pens.Red, pointArray3);
        x = x + incr;
    }
    g.Dispose();
    x = 200;
    findlimit(x,A,Coef,Imod,Fmod,Wc,Diam,T20,Math.Pow(x,1.2),k);
  }
    catch
  {
        emsg.Text = "Select a conductor.";
  }
}
public void satupdate() //update sag and tension chart when changes to any variable is made.
{

            if(CLB.SelectedIndex!=-1) // check to see if a cable is selected
            {
```

```
LimError.Text = "";
try
{
    // put selected conductor into cond
//  Conductor cond = new Conductor();
//  cond.Name(CLB.SelectedItem.ToString());//Load the name

    // get SAT Chart variables for selected conductor (defaults of 20 to 300 m (20 m) -5 to 60 deg (5 deg))

    dataTable1.Clear();
    double tmini = Convert.ToDouble(tmin.Text);
    double tmaxi = Convert.ToDouble(tmax.Text);
    double tinci = Convert.ToDouble(tinc.Text);
    double lmini = Convert.ToDouble(lmin.Text);
    double lmaxi = Convert.ToDouble(lmax.Text);
    double linci = Convert.ToDouble(linc.Text);
    double iten = 0;
    doubleften = 0;
    int worst = 0;
    double edt = 0;
    double IEDT = 0;


    //initial conditions
    double L = lmini;
    double t2 = Convert.ToDouble(tmin.Text) ;
    //int ct =15;// set temperature at which creep conversions are done at 15 deg.
    // Convert Conductor data for easy use
    double A = Convert.ToDouble(Area.Text)/1000000;//convert into sq meters
    double Coef = (Convert.ToDouble(Coefficient.Text)*Math.Pow(10,-6)); //
    double Imod = Convert.ToDouble(IMod.Text)*Math.Pow(10,6); //
```

```
                    double Fmod = (Convert.ToDouble(FMod.Text)*Math.Pow(10,6)); //
                    double Wc = (Convert.ToDouble(Mass.Text)*9.80665/1000); //convert to m/N
                    double Diam = (Convert.ToDouble(Diameter.Text)/1000); //convert to meters
                    double T20 = 0.2*Convert.ToDouble(UTS.Text);
                    double k = Convert.ToDouble(Creep.Text)/1000000; // Creep
                    double L12 = 0;
                    //
                    //
                    //
                    while (L<=lmaxi)
                    {
                        L12 = Math.Pow(L,1.2);
                        worst = 0;
                        worst = findlimit(L,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k );
                        //Convert EDT into IEDT
                        IEDT = ToInitial(L,15,0,Convert.ToDouble(EDT.Text),15,A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);
                        edt = Convert.ToDouble(EDT.Text);
                        dataTable1.Rows.Add(new Object[] {"EDT = ",edt," "," "," ", " "});
                        //
                        //
                        //
                        //If you want add a column use e.g for comments column: dataTable1.Columns.Add("Comments",
typeof(double));//to add an optional column for say %uts etc

                        //Output Limiting weather cases
                        //wind

                        ften
=TempConvert(L,15,0,edt,Convert.ToDouble(Wt2.Text),(Convert.ToDouble(Ww2.Text)*0.6*Diam),"Final",A,Coef,Imod,Fmod,Wc,Dia
m,T20,L12,k);
```

```
                                iten
=TempConvert(L,15,0,IEDT,Convert.ToDouble(Wt3.Text),(Convert.ToDouble(Ww3.Text)*0.6*Diam),"Initial",A,Coef,Imod,Fmod,Wc,
Diam,T20,L12,k);


                                if (worst == 1)
                                    dataTable1.Rows.Add(new Object[] {"Limit = 1 ","I = "+Wt2.Text+" F =
"+Wt3.Text,Math.Ceiling(iten*100)/100+" (" + Ww2.Text +"
PA)",Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,Math.Ceiling(ften*100)/100 +" ("+ Ww3.Text +" PA)",
Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});
                                if (worst == 2)
                                    dataTable1.Rows.Add(new Object[] {"Limit = 2","I = "+Wt2.Text+" F = "+Wt3.Text,"* " +
Math.Ceiling(iten*100)/100+" (" + Ww2.Text +"
PA)",Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,Math.Ceiling(ften*100)/100+" (" + Ww3.Text +" PA)",
Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});
                                if (worst == 3)
                                    dataTable1.Rows.Add(new Object[] {"Limit = 3","I = "+Wt2.Text+" F =
"+Wt3.Text,Math.Ceiling(iten*100)/100+" (" + Ww2.Text +" PA)", Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,"* " +
Math.Ceiling(ften*100)/100+" (" + Ww3.Text +" PA)",Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});

                                dataTable1.Rows.Add(new Object[] {"","","","","","",""});
                                t2 = tmini;
                                while(t2<=tmaxi)
                                {
                                    ften
=TempConvert(L,15,0,Convert.ToDouble(EDT.Text),t2,0,"Final",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);
                                    iten =TempConvert(L,15,0,IEDT,t2,0,"Initial",A,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);
                                    if(t2.ToString() == Wt1.Text&&worst ==1&&Ini1.Text=="Final")
                                        dataTable1.Rows.Add(new Object[] {L,t2,
Math.Ceiling(iten*100)/100,Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,"*"+Math.Ceiling(ften*100)/100 ,
Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});
```

```
                                    else
                                        if(t2.ToString() == Wt1.Text&&worst ==1&&Ini1.Text=="Initial")
                                        dataTable1.Rows.Add(new Object[] {L,t2, "*"+
Math.Ceiling(iten*100)/100,Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,Math.Ceiling(ften*100)/100 ,
Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});
                                        else
                                        dataTable1.Rows.Add(new Object[] {L,t2,
Math.Ceiling(iten*100)/100,Math.Ceiling(Wc*Math.Pow(L,2)/(8*iten)*100)/100,Math.Ceiling(ften*100)/100 ,
Math.Ceiling(Wc*Math.Pow(L,2)/(8*ften)*100)/100});
                                        t2 = t2 + tinci;

                                    }
                                    dataTable1.Rows.Add(new Object[] {"","", "", "", "", ""});
                                    L=L+linci;
                                }
                            }
                            catch
                            {LimError.Text = "Input error";}
                        }
                    }

        public void clearance()// Updates the clearance calculator page when changes to any variable is made.
        {
            try
            {
                // Convert Conductor data for easy use
                double area = Convert.ToDouble(Area.Text)/1000000;//convert into sq meters
                double Coef = (Convert.ToDouble(Coefficient.Text)*Math.Pow(10,-6)); //
                double Imod = Convert.ToDouble(IMod.Text)*Math.Pow(10,6); //
                double Fmod = (Convert.ToDouble(FMod.Text)*Math.Pow(10,6)); //
                double Diam = (Convert.ToDouble(Diameter.Text)/1000); //convert to meters
```

```
            double T20 = 0.2*Convert.ToDouble(UTS.Text);
            double k = Convert.ToDouble(Creep.Text)/1000000; // Creep
            double L = Convert.ToDouble(ESpan.Text);
            double HA = Convert.ToDouble(AttachA.Text);
            double A = Convert.ToDouble(GroundA.Text);
            double O = Convert.ToDouble(GroundO.Text);
            double x = Convert.ToDouble(GroundDist.Text);
            double HO = Convert.ToDouble(HeightO.Text);
            double HB = Convert.ToDouble(AttachB.Text);
            double B = Convert.ToDouble(GroundB.Text);
            double Wc = Convert.ToDouble(Mass.Text)*9.80665/1000; //
            double x1;
            double tens;
            double HT;
            double L12 = Math.Pow(L,1.2);
            TWorst.Text  = "";
            findlimit(L,area,Coef,Imod,Fmod,Wc,Diam,T20,Math.Pow(L,1.2),k);
            if (IF.Text=="Final")
                tens= Convert.ToDouble(EDT.Text);
            else
                tens= ToInitial(L,15,0,Convert.ToDouble(EDT.Text),15,area,Coef,Imod,Fmod,Wc,Diam,T20,L12,k);

            double T =
Convert.ToDouble(Math.Round(TempConvert(L,15,0,tens,Convert.ToDouble(condtemp.Text),0,IF.Text,area,Coef,Imod,Fmod,Wc,Diam,
T20,L12,k ),2));//tension at higher support
            double D = Wc*L*L/(8*T);
            bool flag = false;
            // Swop a and b if b higher than a
            if((HB+B)>(HA+A))
            {
                flag = true;
```

```
        double H1 = HA;
        double G1 = A ;
        HA = HB;
        A  = B;
        HB = H1;
        B  = G1;
        x = L-x;
    }
    double H = Math.Abs((HA+A) - (HB+B));;
    if((1-(H/(4*D)))<0)
    {
        nclearance.Text = "Uplift case" ;
        TWorst.Text =  "Uplift case" ;
        Worst.Text ="0";
    }
    else
    {
        x1 = (L/2)+(T*H/(Wc*L)); //worst distance from Pole A
        HT = T - Math.Pow(Wc*x1,2)/(2*T);//converting to horizontal tension
        double D1 =HT/Wc*(Math.Cosh(Wc*x1/HT)-1);//Sag check to see if accurate base max tension to horizontal
        double x11 = L/2+HT*H/(Wc*L);
        while(Math.Abs(x11-x1)/L>0.1)//1 % of length accuracy
        {
            //check x1
            x1 = x11;
            D1 =HT/Wc*(Math.Cosh(Wc*x1/HT)-1);//Sag check to see if accurate base max tension to horizontal
            x11 = L/2+HT*H/(Wc*L);
        }

        if( Math.Round((HA+A-D1-HO-O),3)>=0)
            Worst.Text  = Convert.ToString(Math.Round((HA+A-D1-HO-O),3));
```

```
else
{
     TWorst.Text  = "Worst case clearance Violation";
     Worst.Text = Convert.ToString(Math.Round((HA+A-D1-HO-O),3));
}
if(flag)//B is higher than A
{
     DWorst.Text = Convert.ToString(Math.Round(10*Math.Round((L-x11),3))/10);
     TensionB.Text = Convert.ToString(Math.Round(HT*Math.Cosh(Wc*(x11)/HT)));
     TensionA.Text = Convert.ToString(Math.Round(HT*Math.Cosh(Wc*(L-x11)/HT)));

}
else
{
     DWorst.Text  = Convert.ToString(Math.Round(10*Math.Round(x11,3))/10);
     TensionA.Text=Convert.ToString(Math.Round(HT*Math.Cosh(Wc*(x11)/HT)));
     TensionB.Text=Convert.ToString(Math.Round(HT*Math.Cosh(Wc*(L-x11)/HT)));
}

// Now work out difference between this sag and that of object
double Xsag= Math.Abs(x11-x); //get distance from object to low point of catenary
double Dsag= HT/Wc*(Math.Cosh(Wc*Xsag/HT)-1);
nclearance.Text = Convert.ToString(Math.Round(100*(Convert.ToDouble(Worst.Text)+Dsag))/100);
Tension.Text=Convert.ToString(Math.Round(HT));

if (x==L)
{
     if(flag)
          nclearance.Text  = Convert.ToString(Math.Round((HB+B-(HO+O)),3));
     else
          nclearance.Text  = Convert.ToString(Math.Round((HA+A-(HO+O)),3));
```

```
                }

                if (Convert.ToDouble(nclearance.Text)<0)
                {
                    TWorst.Text  = "Object clearance Violation";
                }


            }
        }
        catch
        {
            TWorst.Text = "Are inputs correct?";
        }


    }

    public void clpicture()// Updates the graphic on the clearance page
    {
        if(ESpan.Text!=""&&CLB.Text!=""&&AttachA.Text!=""&&GroundA.Text!=""&&GroundA.Text!="-
"&&GroundO.Text!=""&&GroundO.Text!="-"&&GroundO.Text!="-
"&&GroundDist.Text!=""&&HeightO.Text!=""&&AttachB.Text!=""&&GroundB.Text!=""&&GroundB.Text!="-
"&&GroundB.Text!="-")
        {
            // update picture on clearance page
            Graphics g = pictureBox3.CreateGraphics();
            double L = Convert.ToDouble((Math.Round(Convert.ToDouble(ESpan.Text))));


            g.Clear(System.Drawing.Color.White);
```

```
int ymax =   pictureBox3.Height;
//y 480(ymax) - 0, depends on pole lengths and ground levels, max pole height = 80(ymax/6) leaving 400 for ground diffs
int xmax = pictureBox3.Width;
//x 0 - 1600(xmax),  L = 1400 (xmax*0.8)
// Graph dimensions
// constants
int Ax = Convert.ToInt32(0.1*xmax);// Pole A x = 200 m 10 % xmax
int Bx = Convert.ToInt32(0.9*xmax);// Pole B x = 200 m 90 % xmax
// Declare varaibles
int Ay = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundA.Text))*ymax/32+10);//Pole foot co ord y = 0
%
int By = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundB.Text))*ymax/32+10);//Pole foot co ordy = 0
%
int Oy = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundO.Text))*ymax/32+10);//Object foot
int Ox = Ax + (Convert.ToInt32(Math.Round(Convert.ToDouble(GroundDist.Text)/L*xmax*0.8)));// Object distance
int Ah = Convert.ToInt32(Math.Round(Convert.ToDouble(AttachA.Text)))*ymax/32+10;// Pole att height 10 = 60
int Bh = Convert.ToInt32(Math.Round(Convert.ToDouble(AttachB.Text)*ymax/32)+10);// Pole att height 10 = 60
int Oh = Convert.ToInt32(Math.Round(Convert.ToDouble(HeightO.Text)*ymax/32)+10);// Object height 10 = 60
Point PAg = new Point(Ax,Ay);  //Pole A ground
Point PAa = new Point(Ax,Ay-Ah); //Pole A attach
Point POg = new Point(Ox,Oy);  //Object ground
Point POa = new Point(Ox,Oy-Oh); //Object attach
Point PBg = new Point(Bx,By);  //Object ground
Point PBa = new Point(Bx,By-Bh); //Object attach

//Pole A
g.DrawLine(System.Drawing.Pens.Brown,PAg,PAa);
//Object
g.DrawLine(System.Drawing.Pens.Red,POg,POa);
//Pole B
g.DrawLine(System.Drawing.Pens.Brown,PBg,PBa);
```

```
                // conductor
                Point PSh;
                int Wy = Oy - Oh;
                int Wx = Ax+ xmax/2;
                //sag at 2 x worst dist get sag for normal span
                double Wc = Convert.ToDouble(Mass.Text)*9.80665/1000;
                double T = Convert.ToDouble(Tension.Text);


                if (Worst.Text!="Uplift Case"||Worst.Text!="Clearance Violation"||Worst.Text!=""||DWorst.Text!="Uplift
Case"||DWorst.Text!="Clearance Violation"||DWorst.Text!="")
                {
                    Wx = Ax + (Convert.ToInt32(Math.Round(Convert.ToDouble(DWorst.Text)/L*xmax*0.8)));// Object distance

                    Wy = Oy - Oh - Convert.ToInt32(Math.Round(Convert.ToDouble(Worst.Text)*ymax/32));// Object height 10 = 60

                }
                PSh = new Point(Wx ,Wy);

                Point []pointArray2 = {PAa,PSh,PBa};
                // Draw Conductor
                g.DrawCurve(System.Drawing.Pens.Gray,pointArray2);
                // Ground
                Point []pointArray3 = {PAg,POg,PBg};
                g.DrawCurve(System.Drawing.Pens.Green, pointArray3);
                g.Dispose();
            }
        }

        private void ExportToText (string fileName, bool hasColumnHeaders, double columnWidth)// Copies displayed data to a file
```

```
            // Arguments: filename = the file name to be saved to, hasColumnHeaders = has the data got column headers, columnWidth =
column width
        {
            clipboard.Clear();
            // Page set-up for all print functions
            clipboard.AppendText("RSAT 5.1                                          "+DateTime.Now+" ");
            clipboard.AppendText(CLB.Text+" data");
            clipboard.AppendText("
_____________________________________________________________________________________________");
            clipboard.AppendText("UTS: "+UTS.Text+" N   ");
            clipboard.AppendText("Diameter: "+Diameter.Text+" mm     ");
            clipboard.AppendText("C Value: "+CValue.Text+"    ");
            clipboard.AppendText("Mass: "+Mass.Text+" kg/km ");
            clipboard.AppendText("Area: "+Area.Text+" mm^2 ");
            clipboard.AppendText("Final mod: "+FMod.Text+" N/mm^2 ");
            clipboard.AppendText("Initial mod: "+IMod.Text+" N/mm^2");
            clipboard.AppendText("Temp Coeff: "+Coefficient.Text+"'Cx10^-6    ");
            clipboard.AppendText("
_____________________________________________________________________________________________");
            clipboard.AppendText("Limits                                                    ");
            clipboard.AppendText("Limit 1: Temperature = " + Wt1.Text +", "+ CT1.Text + Wc1.Text+ ", Wind "+Ww1.Text+" PA, on
" + Ini1.Text+"                          ");
            clipboard.AppendText("Limit 2: Temperature = " + Wt2.Text +", "+ CT2.Text + Wc2.Text+ ", Wind "+Ww2.Text+" PA, on
" + Ini2.Text+"                          ");
            clipboard.AppendText("Limit 3: Temperature = " + Wt3.Text +", "+ CT3.Text + Wc3.Text+ ", Wind "+Ww3.Text+" PA, on
" + Ini3.Text);
            clipboard.AppendText("
_____________________________________________________________________________________________");

            //Print graph
            int pline = 0;
```

```
            //print page setting for sag and  tension charts

    if(tabControl1.SelectedIndex==0||tabControl1.SelectedIndex==1||tabControl1.SelectedIndex==4||tabControl1.SelectedIndex==5||tabC
ontrol1.SelectedIndex==6)
                {
                    clipboard.AppendText("Span      ");
                    clipboard.AppendText("Temp     ");
                    clipboard.AppendText("Stringing tension      ");
                    clipboard.AppendText("Stringing sag      ");
                    clipboard.AppendText("Final tension      ");
                    clipboard.AppendText("Final sag");
                    clipboard.AppendText("
_____");
                    int max = dataTable1.Rows.Count;
                    while (line<max)
                    {
                        // format EDT and limit lines
                        if (dataTable1.Rows[line]["Span (m)"].ToString()=="EDT = ")
                        {
                            clipboard.AppendText(dataTable1.Rows[line]["Span (m)"].ToString()+"   ");
                            clipboard.AppendText(dataTable1.Rows[line]["Temperature ('C)"].ToString()+"
                ");
                            line++;pline++;
                            clipboard.AppendText(dataTable1.Rows[line]["Span (m)"].ToString()+" (");
                            clipboard.AppendText(dataTable1.Rows[line]["Temperature ('C)"].ToString()+")  ");
                            clipboard.AppendText(dataTable1.Rows[line]["Stringing tension (N)"].ToString()+"    ");
                            clipboard.AppendText(dataTable1.Rows[line]["Stringing sag (m)"].ToString()+"   ");
                            clipboard.AppendText(dataTable1.Rows[line]["Final tension (N)"].ToString()+"    ");
                            clipboard.AppendText(dataTable1.Rows[line]["Final sag (m)"].ToString()+"   ");
                        }
                        // format blank line
```

```
            else
        {
            if (dataTable1.Rows[line]["Span (m)"].ToString()=="")
                clipboard.AppendText("                                                                ");
            else
            {
                    clipboard.AppendText(dataTable1.Rows[line]["Span (m)"].ToString()+"   ");
                clipboard.AppendText(dataTable1.Rows[line]["Temperature ('C)"].ToString()+"          ");
                clipboard.AppendText(dataTable1.Rows[line]["Stringing tension (N)"].ToString()+"               ");
                clipboard.AppendText(dataTable1.Rows[line]["Stringing sag (m)"].ToString()+"        ");
                clipboard.AppendText(dataTable1.Rows[line]["Final tension (N)"].ToString()+"        ");
                clipboard.AppendText(dataTable1.Rows[line]["Final sag (m)"].ToString()+"                    ");
            }
        }
        line++;pline++;
    }



}
if(tabControl1.SelectedIndex==2)
{

    clipboard.AppendText("Poles     ");
    clipboard.AppendText("Spans    ");
    clipboard.AppendText("Sag at    ");
    clipboard.AppendText("5 degs   ");
    clipboard.AppendText("10 degs  ");
    clipboard.AppendText("15 degs  ");
    clipboard.AppendText("20 degs  ");
    clipboard.AppendText("25 degs  ");
```

```
            clipboard.AppendText("30 degs  ");
            clipboard.AppendText("35 degs  ");

            int max = dataTable2.Rows.Count;
            while (line<max)
            {
                if(dataTable2.Rows[line]["Poles"].ToString()=="----------------------------------------------")
                {
                    clipboard.AppendText("
_____________________________________________________________________________________________________");
                    line++;
                }
                clipboard.AppendText(dataTable2.Rows[line]["Poles"].ToString()+"   ");
                clipboard.AppendText(dataTable2.Rows[line]["Span lengths"].ToString()+"    ");
                clipboard.AppendText(dataTable2.Rows[line]["Sag at"].ToString()+" ");
                clipboard.AppendText(dataTable2.Rows[line]["5 degrees"].ToString()+"   ");
                clipboard.AppendText(dataTable2.Rows[line]["10 degrees"].ToString()+"");
                clipboard.AppendText(dataTable2.Rows[line]["15 degrees"].ToString()+"");
                clipboard.AppendText(dataTable2.Rows[line]["20 degrees"].ToString()+"");
                clipboard.AppendText(dataTable2.Rows[line]["25 degrees"].ToString()+"");
                clipboard.AppendText(dataTable2.Rows[line]["30 degrees"].ToString()+"");
                clipboard.AppendText(dataTable2.Rows[line]["35 degrees"].ToString()+"                    ");

                line++;pline++;
            }



        }
        if(tabControl1.SelectedIndex==3)
        {
```

```
            // Data line 1
            clipboard.AppendText("General                                                    ");
            clipboard.AppendText("Span = " +ESpan.Text+" m              ");
            clipboard.AppendText("Temperature = " +condtemp.Text+" 'C    ");
            clipboard.AppendText("Centre Tension = " +Tension.Text+" N  ");
            clipboard.AppendText("
_____________________________________________________________________________________");
            clipboard.AppendText("Pole A                                                     ");
            clipboard.AppendText("Attachement = "+AttachA.Text+" m       ");
            clipboard.AppendText("Ground = "+GroundA.Text+" m        ");
            clipboard.AppendText("Tension = "+TensionA.Text+" N      ");
            clipboard.AppendText("
_____________________________________________________________________________________");
            clipboard.AppendText("Pole B                                                     ");
            clipboard.AppendText("Attachement = "+AttachB.Text+" m       ");
            clipboard.AppendText("Ground = "+GroundB.Text+" m        ");
            clipboard.AppendText("Tension = "+TensionB.Text+" N       ");
            clipboard.AppendText("
_____________________________________________________________________________________");
            clipboard.AppendText("Object                                                     ");
            clipboard.AppendText("Height = "+HeightO.Text+" m              ");
            clipboard.AppendText("Ground = "+GroundO.Text+" m       ");
            clipboard.AppendText("Distance from A = "+GroundDist.Text+" m         ");
            clipboard.AppendText("
_____________________________________________________________________________________");
            clipboard.AppendText("Clearance                                                  ");
            clipboard.AppendText("Object = "+nclearance.Text+" m            ");
            clipboard.AppendText("Worst = "+Worst.Text+" m        ");
            clipboard.AppendText("Worst distance = "+DWorst.Text+" m               ");
            clipboard.AppendText("
_____________________________________________________________________________________");
```

```
    }               clipboard.SaveFile(fileName);
}

public void tabupdate(object sender, System.EventArgs e) //update tab based on selection.
    // Arguments: capture the selected tab
{
    if(tabControl1.SelectedIndex==0)
    {
        try
        {graphupdate();}
        catch
        {Error.Text="try again";}
    }
    if(tabControl1.SelectedIndex==1)
        satupdate();
    if(tabControl1.SelectedIndex==2)
    {
            equivspan();
        Error.Text = "1.Select conductor. 2.Input project data. 3.Replace 'Input span' with span lengths.";
    }
    if(tabControl1.SelectedIndex==3)
        {
            clearance();
            clpicture(); //update picture
        }
    }
public void screensize() //update size of the form based on screen resolution.
{

        int bot = tabControl1.Bottom;
```

```
                int rit = tabControl1.Right;
                int factor = 450;
                int offset =300;
                //EDT.Text = bot.ToString();
                //UTS.Text = rit.ToString();

                label28.Width = label29.Width = label30.Width = label31.Width = label32.Width = 80*rit/factor;
                EDT.Left = UTS.Left = CValue.Left = Area.Left = Diameter.Left = offset + label28.Width;
                EDT.Width = UTS.Width = CValue.Width = Area.Width = Diameter.Width = 70;//64*rit/factor;
                label78.Left =label77.Left =label75.Left =label74.Left =label73.Left = offset + label28.Width + EDT.Width;
                //label78.Width =label77.Width =label75.Width =label74.Width =label73.Width = 58*rit/factor;
                image.Left = label34.Left = label35.Left = label39.Left = label6.Left = offset + label28.Width + EDT.Width +
label73.Width;
                image.Width = label34.Width = label35.Width = label39.Width = label6.Width = 75*rit/factor;
                Mass.Left = Coefficient.Left = IMod.Left = FMod.Left = offset + label28.Width + EDT.Width + label73.Width +
label6.Width;
                Mass.Width = Coefficient.Width = IMod.Width = FMod.Width = 70;
                label79.Left =label80.Left =label81.Left =label76.Left  = offset + label28.Width + EDT.Width + label73.Width +
label6.Width + FMod.Width;
                //label79.Width =label80.Width =label81.Width =label76.Width = 78*rit/factor;
                picture.Left  = offset + label28.Width + EDT.Width + label73.Width + label6.Width + FMod.Width + label76.Width + 5;
              panel1.Width = offset + label28.Width + EDT.Width + label73.Width + label6.Width + FMod.Width + label76.Width +
picture.Width+ 12;
        }
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
```

```
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        System.Configuration.AppSettingsReader configurationAppSettings = new System.Configuration.AppSettingsReader();
        System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(MainView));
        this.label1 = new System.Windows.Forms.Label();
        this.mainMenu1 = new System.Windows.Forms.MainMenu();
        this.menuItem1 = new System.Windows.Forms.MenuItem();
        this.menuItem3 = new System.Windows.Forms.MenuItem();
        this.menuItem4 = new System.Windows.Forms.MenuItem();
        this.PrintPreview = new System.Windows.Forms.MenuItem();
        this.Print = new System.Windows.Forms.MenuItem();
        this.menuItem9 = new System.Windows.Forms.MenuItem();
        this.menuItem10 = new System.Windows.Forms.MenuItem();
        this.menuItem11 = new System.Windows.Forms.MenuItem();
        this.menuItem15 = new System.Windows.Forms.MenuItem();
        this.menuItem21 = new System.Windows.Forms.MenuItem();
        this.menuItem29 = new System.Windows.Forms.MenuItem();
        this.menuItem14 = new System.Windows.Forms.MenuItem();
        this.menuItem18 = new System.Windows.Forms.MenuItem();
```

```
this.menuItem16 = new System.Windows.Forms.MenuItem();
this.CTLB = new System.Windows.Forms.ListBox();
this.label2 = new System.Windows.Forms.Label();
this.label3 = new System.Windows.Forms.Label();
this.CLB = new System.Windows.Forms.ListBox();
this.label6 = new System.Windows.Forms.Label();
this.statusBar1 = new System.Windows.Forms.StatusBar();
this.toolBar1 = new System.Windows.Forms.ToolBar();
this.toolBarButton2 = new System.Windows.Forms.ToolBarButton();
this.toolBarButton3 = new System.Windows.Forms.ToolBarButton();
this.toolBarButton4 = new System.Windows.Forms.ToolBarButton();
this.toolBarButton6 = new System.Windows.Forms.ToolBarButton();
this.toolBarButton7 = new System.Windows.Forms.ToolBarButton();
this.toolBarButton8 = new System.Windows.Forms.ToolBarButton();
this.imageList1 = new System.Windows.Forms.ImageList(this.components);
this.tabControl1 = new System.Windows.Forms.TabControl();
this.tabPage1 = new System.Windows.Forms.TabPage();
this.emsg = new System.Windows.Forms.Label();
this.label7 = new System.Windows.Forms.Label();
this.label88 = new System.Windows.Forms.Label();
this.label25 = new System.Windows.Forms.Label();
this.label24 = new System.Windows.Forms.Label();
this.label87 = new System.Windows.Forms.Label();
this.pictureBox2 = new System.Windows.Forms.PictureBox();
this.label17 = new System.Windows.Forms.Label();
this.label16 = new System.Windows.Forms.Label();
this.label15 = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.label13 = new System.Windows.Forms.Label();
this.label11 = new System.Windows.Forms.Label();
this.label8 = new System.Windows.Forms.Label();
```

```
this.label9 = new System.Windows.Forms.Label();
this.label10 = new System.Windows.Forms.Label();
this.label99 = new System.Windows.Forms.Label();
this.clipboard = new System.Windows.Forms.RichTextBox();
this.tabPage2 = new System.Windows.Forms.TabPage();
this.dataGrid1 = new System.Windows.Forms.DataGrid();
this.dataTable1 = new System.Data.DataTable();
this.dataColumn1 = new System.Data.DataColumn();
this.dataColumn2 = new System.Data.DataColumn();
this.dataColumn3 = new System.Data.DataColumn();
this.dataColumn4 = new System.Data.DataColumn();
this.dataColumn5 = new System.Data.DataColumn();
this.dataColumn6 = new System.Data.DataColumn();
this.tabPage3 = new System.Windows.Forms.TabPage();
this.button1 = new System.Windows.Forms.Button();
this.nsection = new System.Windows.Forms.Button();
this.Error = new System.Windows.Forms.Label();
this.label86 = new System.Windows.Forms.Label();
this.calculate = new System.Windows.Forms.Button();
this.dataGrid2 = new System.Windows.Forms.DataGrid();
this.dataTable2 = new System.Data.DataTable();
this.dataColumn7 = new System.Data.DataColumn();
this.dataColumn8 = new System.Data.DataColumn();
this.dataColumn9 = new System.Data.DataColumn();
this.dataColumn10 = new System.Data.DataColumn();
this.dataColumn11 = new System.Data.DataColumn();
this.dataColumn12 = new System.Data.DataColumn();
this.dataColumn13 = new System.Data.DataColumn();
this.dataColumn14 = new System.Data.DataColumn();
this.dataColumn15 = new System.Data.DataColumn();
this.dataColumn16 = new System.Data.DataColumn();
```

```
this.polef = new System.Windows.Forms.TextBox();
this.poles = new System.Windows.Forms.TextBox();
this.polep = new System.Windows.Forms.TextBox();
this.section = new System.Windows.Forms.TextBox();
this.project = new System.Windows.Forms.TextBox();
this.label85 = new System.Windows.Forms.Label();
this.label84 = new System.Windows.Forms.Label();
this.label83 = new System.Windows.Forms.Label();
this.label82 = new System.Windows.Forms.Label();
this.tabPage4 = new System.Windows.Forms.TabPage();
this.label37 = new System.Windows.Forms.Label();
this.Worst = new System.Windows.Forms.TextBox();
this.nclearance = new System.Windows.Forms.TextBox();
this.Tension = new System.Windows.Forms.TextBox();
this.condtemp = new System.Windows.Forms.TextBox();
this.GroundB = new System.Windows.Forms.TextBox();
this.AttachB = new System.Windows.Forms.TextBox();
this.label67 = new System.Windows.Forms.Label();
this.TensionB = new System.Windows.Forms.TextBox();
this.label66 = new System.Windows.Forms.Label();
this.TensionA = new System.Windows.Forms.TextBox();
this.label36 = new System.Windows.Forms.Label();
this.label26 = new System.Windows.Forms.Label();
this.label97 = new System.Windows.Forms.Label();
this.label105 = new System.Windows.Forms.Label();
this.label104 = new System.Windows.Forms.Label();
this.label106 = new System.Windows.Forms.Label();
this.label4 = new System.Windows.Forms.Label();
this.label107 = new System.Windows.Forms.Label();
this.label108 = new System.Windows.Forms.Label();
this.label109 = new System.Windows.Forms.Label();
```

```
this.label23 = new System.Windows.Forms.Label();
this.label98 = new System.Windows.Forms.Label();
this.label102 = new System.Windows.Forms.Label();
this.label94 = new System.Windows.Forms.Label();
this.label92 = new System.Windows.Forms.Label();
this.label22 = new System.Windows.Forms.Label();
this.label95 = new System.Windows.Forms.Label();
this.label96 = new System.Windows.Forms.Label();
this.label93 = new System.Windows.Forms.Label();
this.label90 = new System.Windows.Forms.Label();
this.label89 = new System.Windows.Forms.Label();
this.label12 = new System.Windows.Forms.Label();
this.TWorst = new System.Windows.Forms.Label();
this.label21 = new System.Windows.Forms.Label();
this.label18 = new System.Windows.Forms.Label();
this.DWorst = new System.Windows.Forms.TextBox();
this.IF = new System.Windows.Forms.ComboBox();
this.GroundDist = new System.Windows.Forms.TextBox();
this.GroundO = new System.Windows.Forms.TextBox();
this.HeightO = new System.Windows.Forms.TextBox();
this.GroundA = new System.Windows.Forms.TextBox();
this.AttachA = new System.Windows.Forms.TextBox();
this.label20 = new System.Windows.Forms.Label();
this.label19 = new System.Windows.Forms.Label();
this.pictureBox3 = new System.Windows.Forms.PictureBox();
this.label27 = new System.Windows.Forms.Label();
this.ESpan = new System.Windows.Forms.TextBox();
this.label91 = new System.Windows.Forms.Label();
this.tabPage6 = new System.Windows.Forms.TabPage();
this.lmin = new System.Windows.Forms.TextBox();
this.tmax = new System.Windows.Forms.TextBox();
```

```
this.lmax = new System.Windows.Forms.TextBox();
this.Defaults = new System.Windows.Forms.Button();
this.LimError = new System.Windows.Forms.Label();
this.Creep = new System.Windows.Forms.TextBox();
this.label5 = new System.Windows.Forms.Label();
this.Ini3 = new System.Windows.Forms.ComboBox();
this.Ini2 = new System.Windows.Forms.ComboBox();
this.Ini1 = new System.Windows.Forms.ComboBox();
this.label63 = new System.Windows.Forms.Label();
this.label64 = new System.Windows.Forms.Label();
this.label65 = new System.Windows.Forms.Label();
this.label60 = new System.Windows.Forms.Label();
this.label61 = new System.Windows.Forms.Label();
this.label62 = new System.Windows.Forms.Label();
this.label57 = new System.Windows.Forms.Label();
this.label58 = new System.Windows.Forms.Label();
this.label59 = new System.Windows.Forms.Label();
this.label56 = new System.Windows.Forms.Label();
this.label55 = new System.Windows.Forms.Label();
this.label54 = new System.Windows.Forms.Label();
this.CT3 = new System.Windows.Forms.ComboBox();
this.Ww3 = new System.Windows.Forms.TextBox();
this.label52 = new System.Windows.Forms.Label();
this.label53 = new System.Windows.Forms.Label();
this.Wc3 = new System.Windows.Forms.TextBox();
this.Wt3 = new System.Windows.Forms.TextBox();
this.CT2 = new System.Windows.Forms.ComboBox();
this.Ww2 = new System.Windows.Forms.TextBox();
this.label49 = new System.Windows.Forms.Label();
this.label51 = new System.Windows.Forms.Label();
this.Wc2 = new System.Windows.Forms.TextBox();
```

```
this.Wt2 = new System.Windows.Forms.TextBox();
this.CT1 = new System.Windows.Forms.ComboBox();
this.label47 = new System.Windows.Forms.Label();
this.Ww1 = new System.Windows.Forms.TextBox();
this.label48 = new System.Windows.Forms.Label();
this.label50 = new System.Windows.Forms.Label();
this.Wc1 = new System.Windows.Forms.TextBox();
this.Wt1 = new System.Windows.Forms.TextBox();
this.label43 = new System.Windows.Forms.Label();
this.linc = new System.Windows.Forms.TextBox();
this.label44 = new System.Windows.Forms.Label();
this.label45 = new System.Windows.Forms.Label();
this.label46 = new System.Windows.Forms.Label();
this.label42 = new System.Windows.Forms.Label();
this.tinc = new System.Windows.Forms.TextBox();
this.label41 = new System.Windows.Forms.Label();
this.label40 = new System.Windows.Forms.Label();
this.label33 = new System.Windows.Forms.Label();
this.tmin = new System.Windows.Forms.TextBox();
this.tabPage5 = new System.Windows.Forms.TabPage();
this.helpindex = new System.Windows.Forms.ListBox();
this.Helptext = new System.Windows.Forms.RichTextBox();
this.tabPage7 = new System.Windows.Forms.TabPage();
this.richTextBox1 = new System.Windows.Forms.RichTextBox();
this.pictureBox1 = new System.Windows.Forms.PictureBox();
this.SagAndTension = new System.Data.DataSet();
this.dataView1 = new System.Data.DataView();
this.Area = new System.Windows.Forms.TextBox();
this.IMod = new System.Windows.Forms.TextBox();
this.FMod = new System.Windows.Forms.TextBox();
this.Mass = new System.Windows.Forms.TextBox();
```

```
this.UTS = new System.Windows.Forms.TextBox();
this.Diameter = new System.Windows.Forms.TextBox();
this.EDT = new System.Windows.Forms.TextBox();
this.label28 = new System.Windows.Forms.Label();
this.label29 = new System.Windows.Forms.Label();
this.label30 = new System.Windows.Forms.Label();
this.CValue = new System.Windows.Forms.TextBox();
this.label31 = new System.Windows.Forms.Label();
this.label32 = new System.Windows.Forms.Label();
this.label34 = new System.Windows.Forms.Label();
this.label35 = new System.Windows.Forms.Label();
this.label39 = new System.Windows.Forms.Label();
this.Coefficient = new System.Windows.Forms.TextBox();
this.picture = new System.Windows.Forms.PictureBox();
this.ntype = new System.Windows.Forms.TextBox();
this.ncond = new System.Windows.Forms.TextBox();
this.Save = new System.Windows.Forms.Button();
this.Cancel = new System.Windows.Forms.Button();
this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
this.image = new System.Windows.Forms.TextBox();
this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
this.ncreep = new System.Windows.Forms.TextBox();
this.Clabel = new System.Windows.Forms.Label();
this.label73 = new System.Windows.Forms.Label();
this.label74 = new System.Windows.Forms.Label();
this.label75 = new System.Windows.Forms.Label();
this.label76 = new System.Windows.Forms.Label();
this.label77 = new System.Windows.Forms.Label();
this.label78 = new System.Windows.Forms.Label();
this.label79 = new System.Windows.Forms.Label();
this.label80 = new System.Windows.Forms.Label();
```

```
            this.label81 = new System.Windows.Forms.Label();
            this.printPreviewDialog1 = new System.Windows.Forms.PrintPreviewDialog();
            this.printDocument1 = new System.Drawing.Printing.PrintDocument();
            this.panel1 = new System.Windows.Forms.Panel();
            this.suggest = new System.Windows.Forms.Label();
            this.label68 = new System.Windows.Forms.Label();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.dataGrid1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.dataTable1)).BeginInit();
            this.tabPage3.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.dataGrid2)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.dataTable2)).BeginInit();
            this.tabPage4.SuspendLayout();
            this.tabPage6.SuspendLayout();
            this.tabPage5.SuspendLayout();
            this.tabPage7.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.SagAndTension)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.dataView1)).BeginInit();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.BackColor = System.Drawing.Color.Transparent;
            this.label1.Font = new System.Drawing.Font("Monotype Corsiva", 15F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label1.ForeColor = System.Drawing.SystemColors.HotTrack;
            this.label1.Location = new System.Drawing.Point(-10, -37);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(2400, 1154);
```

```
            this.label1.TabIndex = 0;
            this.label1.Text = "RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT
RSAT R" +
                "SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT
```

"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +

"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
"SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +

```
    "SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
    "SAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT RSAT R" +
    "";
//
// mainMenu1
//
this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
                                                                        this.menuItem1,
                                                                        this.menuItem11,
                                                                        this.menuItem21,
                                                                        this.menuItem14,
                                                                        this.menuItem18,
                                                                        this.menuItem16});
//
// menuItem1
//
this.menuItem1.Index = 0;
this.menuItem1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
                                                                        this.menuItem3,
                                                                        this.menuItem4,
                                                                        this.PrintPreview,
                                                                        this.Print,
                                                                        this.menuItem9,
                                                                        this.menuItem10});
this.menuItem1.Text = "&File";
//
// menuItem3
//
this.menuItem3.Enabled = false;
this.menuItem3.Index = 0;
this.menuItem3.Text = "&Save";
```

```
this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
//
// menuItem4
//
this.menuItem4.Index = 1;
this.menuItem4.Text = "-";
//
// PrintPreview
//
this.PrintPreview.Index = 2;
this.PrintPreview.Text = "Print Pre&view";
this.PrintPreview.Click += new System.EventHandler(this.PrintPreview_Click);
//
// Print
//
this.Print.Index = 3;
this.Print.Text = "&Print";
this.Print.Click += new System.EventHandler(this.Print_Click);
//
// menuItem9
//
this.menuItem9.Index = 4;
this.menuItem9.Text = "-";
//
// menuItem10
//
this.menuItem10.Index = 5;
this.menuItem10.Text = "E&xit";
this.menuItem10.Click += new System.EventHandler(this.menuItem10_Click);
//
// menuItem11
```

```
//
this.menuItem11.Index = 1;
this.menuItem11.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
                                                                        this.menuItem15});
this.menuItem11.Text = "&Edit";
//
// menuItem15
//
this.menuItem15.Index = 0;
this.menuItem15.Text = "&Spans/Temperatures/Limit";
this.menuItem15.Click += new System.EventHandler(this.menuItem15_Click);
//
// menuItem21
//
this.menuItem21.Index = 2;
this.menuItem21.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
                                                                        this.menuItem29});
this.menuItem21.Text = "&Conductor";
//
// menuItem29
//
this.menuItem29.Index = 0;
this.menuItem29.Text = "&New";
this.menuItem29.Click += new System.EventHandler(this.menuItem29_Click);
//
// menuItem14
//
this.menuItem14.Index = 3;
this.menuItem14.Text = "&Project";
this.menuItem14.Click += new System.EventHandler(this.menuItem14_Click);
//
```

```
            // menuItem18
            //
            this.menuItem18.Index = 4;
            this.menuItem18.Text = "&Help";
            this.menuItem18.Click += new System.EventHandler(this.menuItem18_Click);
            //
            // menuItem16
            //
            this.menuItem16.Index = 5;
            this.menuItem16.Text = "About";
            this.menuItem16.Click += new System.EventHandler(this.menuItem16_Click);
            //
            // CTLB
            //
            this.CTLB.BackColor = System.Drawing.SystemColors.InactiveCaptionText;
            this.CTLB.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.CTLB.ForeColor = System.Drawing.SystemColors.WindowText;
            this.CTLB.ItemHeight = 25;
            this.CTLB.Location = new System.Drawing.Point(1, 62);
            this.CTLB.Name = "CTLB";
            this.CTLB.Size = new System.Drawing.Size(168, 129);
            this.CTLB.TabIndex = 0;
            this.CTLB.SelectedIndexChanged += new System.EventHandler(this.TypeBox_SelectedIndexChanged);
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label2.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label2.Location = new System.Drawing.Point(1, 35);
```

```
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(210, 27);
            this.label2.TabIndex = 2;
            this.label2.Text = "Type";
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label3.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label3.Location = new System.Drawing.Point(182, 35);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(638, 27);
            this.label3.TabIndex = 3;
            this.label3.Text = "Conductor";
            //
            // CLB
            //
            this.CLB.BackColor = System.Drawing.SystemColors.InactiveCaptionText;
            this.CLB.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.CLB.ForeColor = System.Drawing.SystemColors.WindowText;
            this.CLB.ItemHeight = 25;
            this.CLB.Location = new System.Drawing.Point(169, 62);
            this.CLB.Name = "CLB";
            this.CLB.Size = new System.Drawing.Size(192, 129);
            this.CLB.TabIndex = 1;
            this.CLB.SelectedIndexChanged += new System.EventHandler(this.CLB_SelectedIndexChanged);
            //
            // label6
            //
```

```
            this.label6.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label6.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label6.Location = new System.Drawing.Point(578, 36);
            this.label6.Name = "label6";
            this.label6.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.label6.Size = new System.Drawing.Size(104, 27);
            this.label6.TabIndex = 13;
            this.label6.Text = "Mass";
            //
            // statusBar1
            //
            this.statusBar1.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.statusBar1.Location = new System.Drawing.Point(0, 432);
            this.statusBar1.Name = "statusBar1";
            this.statusBar1.Size = new System.Drawing.Size(944, 26);
            this.statusBar1.TabIndex = 14;
            //
            // toolBar1
            //
            this.toolBar1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.toolBar1.Buttons.AddRange(new System.Windows.Forms.ToolBarButton[] {
                                                                                this.toolBarButton2,
                                                                                this.toolBarButton3,
                                                                                this.toolBarButton4,
                                                                                this.toolBarButton6,
                                                                                this.toolBarButton7,
                                                                                this.toolBarButton8});
            this.toolBar1.DropDownArrows = true;
```

```
this.toolBar1.ImageList = this.imageList1;
this.toolBar1.Location = new System.Drawing.Point(0, 0);
this.toolBar1.Name = "toolBar1";
this.toolBar1.ShowToolTips = true;
this.toolBar1.Size = new System.Drawing.Size(944, 30);
this.toolBar1.TabIndex = 15;
this.toolBar1.ButtonClick += new System.Windows.Forms.ToolBarButtonClickEventHandler(this.toolBar1_ButtonClick);
//
// toolBarButton2
//
this.toolBarButton2.Enabled = false;
this.toolBarButton2.ImageIndex = 10;
this.toolBarButton2.ToolTipText = "Save";
//
// toolBarButton3
//
this.toolBarButton3.ImageIndex = 8;
this.toolBarButton3.ToolTipText = "Print";
//
// toolBarButton4
//
this.toolBarButton4.ImageIndex = 7;
this.toolBarButton4.Pushed = ((bool)(configurationAppSettings.GetValue("toolBarButton4.Pushed", typeof(bool))));
this.toolBarButton4.ToolTipText = "Print Preview";
//
// toolBarButton6
//
this.toolBarButton6.ImageIndex = 15;
this.toolBarButton6.ToolTipText = "Limits";
//
// toolBarButton7
```

```
            //
            this.toolBarButton7.ImageIndex = 21;
            this.toolBarButton7.Pushed = ((bool)(configurationAppSettings.GetValue("toolBarButton7.Pushed", typeof(bool))));
            this.toolBarButton7.ToolTipText = "Create conductor ";
            //
            // toolBarButton8
            //
            this.toolBarButton8.ImageIndex = 0;
            this.toolBarButton8.ToolTipText = "Help";
            //
            // imageList1
            //
            this.imageList1.ImageSize = new System.Drawing.Size(16, 16);
            this.imageList1.ImageStream =
((System.Windows.Forms.ImageListStreamer)(resources.GetObject("imageList1.ImageStream")));
            this.imageList1.TransparentColor = System.Drawing.Color.Transparent;
            //
            // tabControl1
            //
            this.tabControl1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Controls.Add(this.tabPage3);
            this.tabControl1.Controls.Add(this.tabPage4);
            this.tabControl1.Controls.Add(this.tabPage6);
            this.tabControl1.Controls.Add(this.tabPage5);
            this.tabControl1.Controls.Add(this.tabPage7);
```

```
            this.tabControl1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.tabControl1.Location = new System.Drawing.Point(8, 185);
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(916, 231);
            this.tabControl1.TabIndex = 2;
            this.tabControl1.Click += new System.EventHandler(this.tabupdate);
            //
            // tabPage1
            //
            this.tabPage1.Controls.Add(this.emsg);
            this.tabPage1.Controls.Add(this.label7);
            this.tabPage1.Controls.Add(this.label88);
            this.tabPage1.Controls.Add(this.label25);
            this.tabPage1.Controls.Add(this.label24);
            this.tabPage1.Controls.Add(this.label87);
            this.tabPage1.Controls.Add(this.pictureBox2);
            this.tabPage1.Controls.Add(this.label17);
            this.tabPage1.Controls.Add(this.label16);
            this.tabPage1.Controls.Add(this.label15);
            this.tabPage1.Controls.Add(this.label14);
            this.tabPage1.Controls.Add(this.label13);
            this.tabPage1.Controls.Add(this.label11);
            this.tabPage1.Controls.Add(this.label8);
            this.tabPage1.Controls.Add(this.label9);
            this.tabPage1.Controls.Add(this.label10);
            this.tabPage1.Controls.Add(this.label99);
            this.tabPage1.Controls.Add(this.clipboard);
            this.tabPage1.Location = new System.Drawing.Point(4, 34);
            this.tabPage1.Name = "tabPage1";
```

```
            this.tabPage1.Size = new System.Drawing.Size(908, 193);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Graph";
            this.tabPage1.Paint += new System.Windows.Forms.PaintEventHandler(this.Form_Paint);
            //
            // emsg
            //
            this.emsg.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.emsg.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.emsg.ForeColor = System.Drawing.SystemColors.HotTrack;
            this.emsg.Location = new System.Drawing.Point(106, 48);
            this.emsg.Name = "emsg";
            this.emsg.Size = new System.Drawing.Size(656, 56);
            this.emsg.TabIndex = 21;
            //
            // label7
            //
            this.label7.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.label7.Font = new System.Drawing.Font("Arial", 14.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label7.Location = new System.Drawing.Point(96, 9);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(758, 33);
            this.label7.TabIndex = 0;
            this.label7.Text = "% of Ultimate Tensile Strength verses span length";
            //
            // label88
            //
            this.label88.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
```

```
            this.label88.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label88.Location = new System.Drawing.Point(1000, 135);
            this.label88.Name = "label88";
            this.label88.Size = new System.Drawing.Size(66, 28);
            this.label88.TabIndex = 20;
            this.label88.Text = "450 m";
            //
            // label25
            //
            this.label25.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label25.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label25.Location = new System.Drawing.Point(900, 135);
            this.label25.Name = "label25";
            this.label25.Size = new System.Drawing.Size(66, 28);
            this.label25.TabIndex = 19;
            this.label25.Text = "400 m";
            //
            // label24
            //
            this.label24.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label24.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label24.Location = new System.Drawing.Point(800, 135);
            this.label24.Name = "label24";
            this.label24.Size = new System.Drawing.Size(66, 28);
            this.label24.TabIndex = 18;
            this.label24.Text = "350 m";
```

```
            //
            // label87
            //
            this.label87.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label87.BackColor = System.Drawing.SystemColors.Window;
            this.label87.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label87.ForeColor = System.Drawing.Color.Red;
            this.label87.Location = new System.Drawing.Point(768, 48);
            this.label87.Name = "label87";
            this.label87.Size = new System.Drawing.Size(182, 56);
            this.label87.TabIndex = 13;
            this.label87.Text = "Not accurate above 300 m";
            //
            // pictureBox2
            //
            this.pictureBox2.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)));
            this.pictureBox2.BackColor = System.Drawing.SystemColors.Window;
            this.pictureBox2.Location = new System.Drawing.Point(96, 8);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(1373, 112);
            this.pictureBox2.TabIndex = 12;
            this.pictureBox2.TabStop = false;
            this.pictureBox2.Click += new System.EventHandler(this.pictureBox2_Click);
            //
            // label17
            //
```

```
            this.label17.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label17.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label17.Location = new System.Drawing.Point(600, 135);
            this.label17.Name = "label17";
            this.label17.Size = new System.Drawing.Size(66, 28);
            this.label17.TabIndex = 10;
            this.label17.Text = "250 m";
            //
            // label16
            //
            this.label16.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label16.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label16.Location = new System.Drawing.Point(500, 135);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(66, 28);
            this.label16.TabIndex = 9;
            this.label16.Text = "200 m";
            //
            // label15
            //
            this.label15.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label15.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label15.Location = new System.Drawing.Point(300, 135);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(66, 28);
```

```
            this.label15.TabIndex = 8;
            this.label15.Text = "100 m";
            //
            // label14
            //
            this.label14.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label14.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label14.Location = new System.Drawing.Point(200, 135);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(54, 28);
            this.label14.TabIndex = 7;
            this.label14.Text = "50 m";
            //
            // label13
            //
            this.label13.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label13.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label13.Location = new System.Drawing.Point(400, 135);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(66, 28);
            this.label13.TabIndex = 6;
            this.label13.Text = "150 m";
            //
            // label11
            //
            this.label11.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
```

```
            this.label11.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label11.Location = new System.Drawing.Point(100, 135);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(48, 28);
            this.label11.TabIndex = 4;
            this.label11.Text = "0 m ";
            //
            // label8
            //
            this.label8.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label8.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label8.Location = new System.Drawing.Point(352, 214);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(170, 26);
            this.label8.TabIndex = 1;
            this.label8.Text = "Span Length (m)";
            //
            // label9
            //
            this.label9.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label9.Location = new System.Drawing.Point(40, 8);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(56, 28);
            this.label9.TabIndex = 2;
            this.label9.Text = "40 %";
            //
            // label10
```

```
            //
            this.label10.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label10.Location = new System.Drawing.Point(40, 120);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(56, 28);
            this.label10.TabIndex = 3;
            this.label10.Text = "20 %";
            //
            // label99
            //
            this.label99.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.label99.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label99.Location = new System.Drawing.Point(700, 135);
            this.label99.Name = "label99";
            this.label99.Size = new System.Drawing.Size(66, 28);
            this.label99.TabIndex = 5;
            this.label99.Text = "300 m";
            //
            // clipboard
            //
            this.clipboard.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.clipboard.Location = new System.Drawing.Point(0, 160);
            this.clipboard.Name = "clipboard";
            this.clipboard.Size = new System.Drawing.Size(1296, 72);
            this.clipboard.TabIndex = 22;
            this.clipboard.Text = "";
            this.clipboard.Visible = false;
```

```
            //
            // tabPage2
            //
            this.tabPage2.Controls.Add(this.dataGrid1);
            this.tabPage2.Location = new System.Drawing.Point(4, 34);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Size = new System.Drawing.Size(908, 193);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Sag and tension";
            //
            // dataGrid1
            //
            this.dataGrid1.AllowDrop = true;
            this.dataGrid1.AllowSorting = false;
            this.dataGrid1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.dataGrid1.BackgroundColor = System.Drawing.SystemColors.Control;
            this.dataGrid1.CaptionText = "Sag and Tension Chart";
            this.dataGrid1.DataMember = "";
            this.dataGrid1.DataSource = this.dataTable1;
            this.dataGrid1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.dataGrid1.GridLineColor = System.Drawing.SystemColors.ControlDarkDark;
            this.dataGrid1.HeaderForeColor = System.Drawing.SystemColors.ControlText;
            this.dataGrid1.Location = new System.Drawing.Point(0, 0);
            this.dataGrid1.Name = "dataGrid1";
            this.dataGrid1.PreferredColumnWidth = 160;
            this.dataGrid1.ReadOnly = true;
            this.dataGrid1.RowHeaderWidth = 15;
```

```
this.dataGrid1.Size = new System.Drawing.Size(899, 188);
this.dataGrid1.TabIndex = 0;
//
// dataTable1
//
this.dataTable1.Columns.AddRange(new System.Data.DataColumn[] {
                                                                this.dataColumn1,
                                                                this.dataColumn2,
                                                                this.dataColumn3,
                                                                this.dataColumn4,
                                                                this.dataColumn5,
                                                                this.dataColumn6});
this.dataTable1.TableName = "Table1";
//
// dataColumn1
//
this.dataColumn1.Caption = "Span (m)";
this.dataColumn1.ColumnName = "Span (m)";
this.dataColumn1.MaxLength = 20;
this.dataColumn1.ReadOnly = true;
//
// dataColumn2
//
this.dataColumn2.Caption = "Temperature (\'C)";
this.dataColumn2.ColumnName = "Temperature (\'C)";
this.dataColumn2.MaxLength = 20;
this.dataColumn2.ReadOnly = true;
//
// dataColumn3
//
this.dataColumn3.Caption = "Stringing tension (N)";
```

```
this.dataColumn3.ColumnName = "Stringing tension (N)";
this.dataColumn3.ReadOnly = true;
//
// dataColumn4
//
this.dataColumn4.Caption = "Stringing sag (m)";
this.dataColumn4.ColumnName = "Stringing sag (m)";
this.dataColumn4.ReadOnly = true;
//
// dataColumn5
//
this.dataColumn5.Caption = "Final Tension (N)";
this.dataColumn5.ColumnName = "Final tension (N)";
this.dataColumn5.ReadOnly = true;
//
// dataColumn6
//
this.dataColumn6.Caption = "Final Sag (m)";
this.dataColumn6.ColumnName = "Final sag (m)";
this.dataColumn6.ReadOnly = true;
//
// tabPage3
//
this.tabPage3.Controls.Add(this.button1);
this.tabPage3.Controls.Add(this.nsection);
this.tabPage3.Controls.Add(this.Error);
this.tabPage3.Controls.Add(this.label86);
this.tabPage3.Controls.Add(this.calculate);
this.tabPage3.Controls.Add(this.dataGrid2);
this.tabPage3.Controls.Add(this.polef);
this.tabPage3.Controls.Add(this.poles);
```

```
this.tabPage3.Controls.Add(this.polep);
this.tabPage3.Controls.Add(this.section);
this.tabPage3.Controls.Add(this.project);
this.tabPage3.Controls.Add(this.label85);
this.tabPage3.Controls.Add(this.label84);
this.tabPage3.Controls.Add(this.label83);
this.tabPage3.Controls.Add(this.label82);
this.tabPage3.Location = new System.Drawing.Point(4, 34);
this.tabPage3.Name = "tabPage3";
this.tabPage3.Size = new System.Drawing.Size(908, 193);
this.tabPage3.TabIndex = 2;
this.tabPage3.Text = "Stringing chart";
//
// button1
//
this.button1.Location = new System.Drawing.Point(547, 63);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(144, 32);
this.button1.TabIndex = 14;
this.button1.Text = "Reset";
this.button1.Click += new System.EventHandler(this.button1_Click);
//
// nsection
//
this.nsection.Location = new System.Drawing.Point(547, 32);
this.nsection.Name = "nsection";
this.nsection.Size = new System.Drawing.Size(144, 31);
this.nsection.TabIndex = 12;
this.nsection.Text = "Next section";
this.nsection.Click += new System.EventHandler(this.nsection_Click);
//
```

```
            // Error
            //
            this.Error.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.Error.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.Error.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.Error.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Error.ForeColor = System.Drawing.Color.Red;
            this.Error.Location = new System.Drawing.Point(692, 0);
            this.Error.Name = "Error";
            this.Error.Size = new System.Drawing.Size(210, 92);
            this.Error.TabIndex = 13;
            this.Error.Text = "Replace \"Input span\" with span lengths";
            //
            // label86
            //
            this.label86.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label86.Location = new System.Drawing.Point(6, 63);
            this.label86.Name = "label86";
            this.label86.Size = new System.Drawing.Size(114, 24);
            this.label86.TabIndex = 4;
            this.label86.Text = "Pole prefix";
            //
            // calculate
            //
            this.calculate.Location = new System.Drawing.Point(547, 0);
            this.calculate.Name = "calculate";
            this.calculate.Size = new System.Drawing.Size(144, 32);
```

```
            this.calculate.TabIndex = 11;
            this.calculate.Text = "Calculate";
            this.calculate.Click += new System.EventHandler(this.calculate_Click);
            //
            // dataGrid2
            //
            this.dataGrid2.AllowDrop = true;
            this.dataGrid2.AllowSorting = false;
            this.dataGrid2.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.dataGrid2.CaptionText = "Stringing Chart";
            this.dataGrid2.DataMember = "";
            this.dataGrid2.DataSource = this.dataTable2;
            this.dataGrid2.HeaderForeColor = System.Drawing.SystemColors.ControlText;
            this.dataGrid2.Location = new System.Drawing.Point(8, 96);
            this.dataGrid2.Name = "dataGrid2";
            this.dataGrid2.PreferredColumnWidth = 105;
            this.dataGrid2.Size = new System.Drawing.Size(891, 94);
            this.dataGrid2.TabIndex = 10;
            //
            // dataTable2
            //
            this.dataTable2.Columns.AddRange(new System.Data.DataColumn[] {
                                                                        this.dataColumn7,
                                                                        this.dataColumn8,
                                                                        this.dataColumn9,
                                                                        this.dataColumn10,
                                                                        this.dataColumn11,
                                                                        this.dataColumn12,
```

```
this.dataColumn13,
this.dataColumn14,
this.dataColumn15,
this.dataColumn16});
this.dataTable2.MinimumCapacity = 75;
this.dataTable2.TableName = "Table2";
//
// dataColumn7
//
this.dataColumn7.ColumnName = "Poles";
//
// dataColumn8
//
this.dataColumn8.Caption = "Span lengths";
this.dataColumn8.ColumnName = "Span lengths";
//
// dataColumn9
//
this.dataColumn9.Caption = "Sag at";
this.dataColumn9.ColumnName = "Sag at";
//
// dataColumn10
//
this.dataColumn10.Caption = "5 degrees";
this.dataColumn10.ColumnName = "5 degrees";
//
// dataColumn11
//
this.dataColumn11.Caption = "10 degrees";
this.dataColumn11.ColumnName = "10 degrees";
//
```

```
            // dataColumn12
            //
            this.dataColumn12.Caption = "15 degrees";
            this.dataColumn12.ColumnName = "15 degrees";
            //
            // dataColumn13
            //
            this.dataColumn13.Caption = "20 degrees";
            this.dataColumn13.ColumnName = "20 degrees";
            //
            // dataColumn14
            //
            this.dataColumn14.Caption = "25 degrees";
            this.dataColumn14.ColumnName = "25 degrees";
            //
            // dataColumn15
            //
            this.dataColumn15.Caption = "30 degrees";
            this.dataColumn15.ColumnName = "30 degrees";
            //
            // dataColumn16
            //
            this.dataColumn16.Caption = "35 degrees";
            this.dataColumn16.ColumnName = "35 degrees";
            //
            // polef
            //
            this.polef.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.polef.Location = new System.Drawing.Point(442, 32);
            this.polef.Name = "polef";
```

```
            this.polef.Size = new System.Drawing.Size(99, 27);
            this.polef.TabIndex = 9;
            this.polef.Text = "3";
            this.polef.TextChanged += new System.EventHandler(this.polef_TextChanged);
            //
            // poles
            //
            this.poles.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.poles.Location = new System.Drawing.Point(442, 8);
            this.poles.Name = "poles";
            this.poles.Size = new System.Drawing.Size(99, 27);
            this.poles.TabIndex = 8;
            this.poles.Text = "1";
            this.poles.TextChanged += new System.EventHandler(this.poles_TextChanged);
            //
            // polep
            //
            this.polep.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.polep.Location = new System.Drawing.Point(162, 63);
            this.polep.Name = "polep";
            this.polep.TabIndex = 7;
            this.polep.Text = "ESK";
            this.polep.TextChanged += new System.EventHandler(this.polep_TextChanged);
            //
            // section
            //
            this.section.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.section.Location = new System.Drawing.Point(162, 32);
```

```
            this.section.Name = "section";
            this.section.TabIndex = 6;
            this.section.Text = "1";
            this.section.TextChanged += new System.EventHandler(this.section_TextChanged);
            //
            // project
            //
            this.project.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.project.Location = new System.Drawing.Point(162, 8);
            this.project.Name = "project";
            this.project.TabIndex = 5;
            this.project.Text = "Project 1";
            this.project.TextChanged += new System.EventHandler(this.project_TextChanged);
            //
            // label85
            //
            this.label85.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label85.Location = new System.Drawing.Point(265, 32);
            this.label85.Name = "label85";
            this.label85.Size = new System.Drawing.Size(167, 23);
            this.label85.TabIndex = 3;
            this.label85.Text = "Last pole number";
            //
            // label84
            //
            this.label84.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label84.Location = new System.Drawing.Point(265, 8);
            this.label84.Name = "label84";
```

```
            this.label84.Size = new System.Drawing.Size(168, 24);
            this.label84.TabIndex = 2;
            this.label84.Text = "First pole number";
            //
            // label83
            //
            this.label83.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label83.Location = new System.Drawing.Point(6, 32);
            this.label83.Name = "label83";
            this.label83.Size = new System.Drawing.Size(167, 23);
            this.label83.TabIndex = 1;
            this.label83.Text = "Section number";
            //
            // label82
            //
            this.label82.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label82.Location = new System.Drawing.Point(6, 8);
            this.label82.Name = "label82";
            this.label82.Size = new System.Drawing.Size(130, 23);
            this.label82.TabIndex = 0;
            this.label82.Text = "Project title";
            //
            // tabPage4
            //
            this.tabPage4.Controls.Add(this.label37);
            this.tabPage4.Controls.Add(this.Worst);
            this.tabPage4.Controls.Add(this.nclearance);
            this.tabPage4.Controls.Add(this.Tension);
            this.tabPage4.Controls.Add(this.condtemp);
```

```
this.tabPage4.Controls.Add(this.GroundB);
this.tabPage4.Controls.Add(this.AttachB);
this.tabPage4.Controls.Add(this.label67);
this.tabPage4.Controls.Add(this.TensionB);
this.tabPage4.Controls.Add(this.label66);
this.tabPage4.Controls.Add(this.TensionA);
this.tabPage4.Controls.Add(this.label36);
this.tabPage4.Controls.Add(this.label26);
this.tabPage4.Controls.Add(this.label97);
this.tabPage4.Controls.Add(this.label105);
this.tabPage4.Controls.Add(this.label104);
this.tabPage4.Controls.Add(this.label106);
this.tabPage4.Controls.Add(this.label4);
this.tabPage4.Controls.Add(this.label107);
this.tabPage4.Controls.Add(this.label108);
this.tabPage4.Controls.Add(this.label109);
this.tabPage4.Controls.Add(this.label23);
this.tabPage4.Controls.Add(this.label98);
this.tabPage4.Controls.Add(this.label102);
this.tabPage4.Controls.Add(this.label94);
this.tabPage4.Controls.Add(this.label92);
this.tabPage4.Controls.Add(this.label22);
this.tabPage4.Controls.Add(this.label95);
this.tabPage4.Controls.Add(this.label96);
this.tabPage4.Controls.Add(this.label93);
this.tabPage4.Controls.Add(this.label90);
this.tabPage4.Controls.Add(this.label89);
this.tabPage4.Controls.Add(this.label12);
this.tabPage4.Controls.Add(this.TWorst);
this.tabPage4.Controls.Add(this.label21);
this.tabPage4.Controls.Add(this.label18);
```

```
            this.tabPage4.Controls.Add(this.DWorst);
            this.tabPage4.Controls.Add(this.IF);
            this.tabPage4.Controls.Add(this.GroundDist);
            this.tabPage4.Controls.Add(this.GroundO);
            this.tabPage4.Controls.Add(this.HeightO);
            this.tabPage4.Controls.Add(this.GroundA);
            this.tabPage4.Controls.Add(this.AttachA);
            this.tabPage4.Controls.Add(this.label20);
            this.tabPage4.Controls.Add(this.label19);
            this.tabPage4.Controls.Add(this.pictureBox3);
            this.tabPage4.Controls.Add(this.label27);
            this.tabPage4.Controls.Add(this.ESpan);
            this.tabPage4.Controls.Add(this.label91);
            this.tabPage4.Location = new System.Drawing.Point(4, 34);
            this.tabPage4.Name = "tabPage4";
            this.tabPage4.Size = new System.Drawing.Size(908, 193);
            this.tabPage4.TabIndex = 3;
            this.tabPage4.Text = "Clearance calculator";
            this.tabPage4.Paint += new System.Windows.Forms.PaintEventHandler(this.Form_Paint2);
            //
            // label37
            //
            this.label37.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label37.Location = new System.Drawing.Point(634, 73);
            this.label37.Name = "label37";
            this.label37.Size = new System.Drawing.Size(31, 27);
            this.label37.TabIndex = 68;
            this.label37.Text = "N";
            //
            // Worst
```

```
            //
            this.Worst.Enabled = false;
            this.Worst.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Worst.Location = new System.Drawing.Point(557, 121);
            this.Worst.Name = "Worst";
            this.Worst.Size = new System.Drawing.Size(65, 27);
            this.Worst.TabIndex = 30;
            this.Worst.Text = "";
            //
            // nclearance
            //
            this.nclearance.Enabled = false;
            this.nclearance.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.nclearance.Location = new System.Drawing.Point(557, 97);
            this.nclearance.Name = "nclearance";
            this.nclearance.Size = new System.Drawing.Size(65, 27);
            this.nclearance.TabIndex = 29;
            this.nclearance.Text = "";
            //
            // Tension
            //
            this.Tension.Enabled = false;
            this.Tension.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Tension.Location = new System.Drawing.Point(557, 73);
            this.Tension.Name = "Tension";
            this.Tension.Size = new System.Drawing.Size(65, 27);
            this.Tension.TabIndex = 28;
            this.Tension.Text = "";
```

```
            //
            // condtemp
            //
            this.condtemp.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.condtemp.Location = new System.Drawing.Point(557, 48);
            this.condtemp.Name = "condtemp";
            this.condtemp.Size = new System.Drawing.Size(65, 27);
            this.condtemp.TabIndex = 27;
            this.condtemp.Text = "";
            this.condtemp.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // GroundB
            //
            this.GroundB.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.GroundB.Location = new System.Drawing.Point(557, 27);
            this.GroundB.Name = "GroundB";
            this.GroundB.Size = new System.Drawing.Size(65, 27);
            this.GroundB.TabIndex = 26;
            this.GroundB.Text = "";
            this.GroundB.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // AttachB
            //
            this.AttachB.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.AttachB.Location = new System.Drawing.Point(557, 1);
            this.AttachB.Name = "AttachB";
            this.AttachB.Size = new System.Drawing.Size(65, 27);
            this.AttachB.TabIndex = 25;
```

```
            this.AttachB.Text = "";
            this.AttachB.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // label67
            //
            this.label67.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label67.Location = new System.Drawing.Point(845, 72);
            this.label67.Name = "label67";
            this.label67.Size = new System.Drawing.Size(32, 28);
            this.label67.TabIndex = 67;
            this.label67.Text = "N";
            //
            // TensionB
            //
            this.TensionB.Enabled = false;
            this.TensionB.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.TensionB.Location = new System.Drawing.Point(778, 73);
            this.TensionB.Name = "TensionB";
            this.TensionB.Size = new System.Drawing.Size(63, 27);
            this.TensionB.TabIndex = 66;
            this.TensionB.Text = "";
            //
            // label66
            //
            this.label66.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label66.Location = new System.Drawing.Point(845, 48);
            this.label66.Name = "label66";
            this.label66.Size = new System.Drawing.Size(32, 28);
```

```
            this.label66.TabIndex = 65;
            this.label66.Text = "N";
            //
            // TensionA
            //
            this.TensionA.Enabled = false;
            this.TensionA.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.TensionA.Location = new System.Drawing.Point(778, 48);
            this.TensionA.Name = "TensionA";
            this.TensionA.Size = new System.Drawing.Size(63, 27);
            this.TensionA.TabIndex = 64;
            this.TensionA.Text = "";
            //
            // label36
            //
            this.label36.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label36.Location = new System.Drawing.Point(672, 74);
            this.label36.Name = "label36";
            this.label36.TabIndex = 63;
            this.label36.Text = "Tension B";
            //
            // label26
            //
            this.label26.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label26.Location = new System.Drawing.Point(672, 46);
            this.label26.Name = "label26";
            this.label26.TabIndex = 62;
            this.label26.Text = "Tension A";
```

```
            //
            // label97
            //
            this.label97.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label97.Location = new System.Drawing.Point(634, 46);
            this.label97.Name = "label97";
            this.label97.Size = new System.Drawing.Size(38, 28);
            this.label97.TabIndex = 44;
            this.label97.Text = "\'C";
            //
            // label105
            //
            this.label105.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label105.Location = new System.Drawing.Point(336, 126);
            this.label105.Name = "label105";
            this.label105.Size = new System.Drawing.Size(163, 24);
            this.label105.TabIndex = 60;
            this.label105.Text = "Worst clearance";
            this.label105.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            //
            // label104
            //
            this.label104.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label104.Location = new System.Drawing.Point(336, 102);
            this.label104.Name = "label104";
            this.label104.Size = new System.Drawing.Size(192, 24);
            this.label104.TabIndex = 61;
            this.label104.Text = "Object clearance";
```

```
            //
            // label106
            //
            this.label106.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label106.Location = new System.Drawing.Point(336, 77);
            this.label106.Name = "label106";
            this.label106.Size = new System.Drawing.Size(192, 25);
            this.label106.TabIndex = 59;
            this.label106.Text = "Section tension";
            //
            // label4
            //
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label4.Location = new System.Drawing.Point(336, 50);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(173, 27);
            this.label4.TabIndex = 37;
            this.label4.Text = "Conductor Temp";
            //
            // label107
            //
            this.label107.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label107.Location = new System.Drawing.Point(0, 2);
            this.label107.Name = "label107";
            this.label107.Size = new System.Drawing.Size(182, 25);
            this.label107.TabIndex = 58;
            this.label107.Text = "Equivalent Span";
            //
```

```
            // label108
            //
            this.label108.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label108.Location = new System.Drawing.Point(336, 27);
            this.label108.Name = "label108";
            this.label108.Size = new System.Drawing.Size(192, 23);
            this.label108.TabIndex = 57;
            this.label108.Text = "B: ground level";
            //
            // label109
            //
            this.label109.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label109.Location = new System.Drawing.Point(336, 3);
            this.label109.Name = "label109";
            this.label109.Size = new System.Drawing.Size(221, 24);
            this.label109.TabIndex = 56;
            this.label109.Text = "B: attachment height";
            //
            // label23
            //
            this.label23.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label23.Location = new System.Drawing.Point(634, 98);
            this.label23.Name = "label23";
            this.label23.Size = new System.Drawing.Size(19, 24);
            this.label23.TabIndex = 55;
            this.label23.Text = "m";
            //
            // label98
```

```
            //
            this.label98.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label98.Location = new System.Drawing.Point(298, 2);
            this.label98.Name = "label98";
            this.label98.Size = new System.Drawing.Size(27, 25);
            this.label98.TabIndex = 54;
            this.label98.Text = "m";
            //
            // label102
            //
            this.label102.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label102.Location = new System.Drawing.Point(634, 24);
            this.label102.Name = "label102";
            this.label102.Size = new System.Drawing.Size(33, 23);
            this.label102.TabIndex = 51;
            this.label102.Text = "m";
            //
            // label94
            //
            this.label94.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label94.Location = new System.Drawing.Point(298, 122);
            this.label94.Name = "label94";
            this.label94.Size = new System.Drawing.Size(26, 25);
            this.label94.TabIndex = 49;
            this.label94.Text = "m";
            //
            // label92
            //
```

```
            this.label92.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label92.Location = new System.Drawing.Point(298, 98);
            this.label92.Name = "label92";
            this.label92.Size = new System.Drawing.Size(26, 24);
            this.label92.TabIndex = 48;
            this.label92.Text = "m";
            //
            // label22
            //
            this.label22.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label22.Location = new System.Drawing.Point(298, 74);
            this.label22.Name = "label22";
            this.label22.Size = new System.Drawing.Size(26, 24);
            this.label22.TabIndex = 47;
            this.label22.Text = "m";
            //
            // label95
            //
            this.label95.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label95.Location = new System.Drawing.Point(298, 50);
            this.label95.Name = "label95";
            this.label95.Size = new System.Drawing.Size(26, 24);
            this.label95.TabIndex = 46;
            this.label95.Text = "m";
            //
            // label96
            //
```

```
            this.label96.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label96.Location = new System.Drawing.Point(298, 24);
            this.label96.Name = "label96";
            this.label96.Size = new System.Drawing.Size(33, 23);
            this.label96.TabIndex = 45;
            this.label96.Text = "m";
            //
            // label93
            //
            this.label93.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label93.Location = new System.Drawing.Point(0, 98);
            this.label93.Name = "label93";
            this.label93.Size = new System.Drawing.Size(163, 24);
            this.label93.TabIndex = 43;
            this.label93.Text = "O: ground level";
            //
            // label90
            //
            this.label90.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label90.Location = new System.Drawing.Point(0, 74);
            this.label90.Name = "label90";
            this.label90.Size = new System.Drawing.Size(144, 24);
            this.label90.TabIndex = 40;
            this.label90.Text = "O: height";
            //
            // label89
            //
```

```
            this.label89.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label89.Location = new System.Drawing.Point(0, 50);
            this.label89.Name = "label89";
            this.label89.Size = new System.Drawing.Size(173, 24);
            this.label89.TabIndex = 39;
            this.label89.Text = "A: ground level";
            //
            // label12
            //
            this.label12.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label12.Location = new System.Drawing.Point(0, 27);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(211, 21);
            this.label12.TabIndex = 38;
            this.label12.Text = "A: attachment height";
            //
            // TWorst
            //
            this.TWorst.BackColor = System.Drawing.SystemColors.Window;
            this.TWorst.Font = new System.Drawing.Font("Microsoft Sans Serif", 13.8F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.TWorst.ForeColor = System.Drawing.Color.Red;
            this.TWorst.Location = new System.Drawing.Point(173, 157);
            this.TWorst.Name = "TWorst";
            this.TWorst.Size = new System.Drawing.Size(659, 48);
            this.TWorst.TabIndex = 36;
            //
            // label21
            //
```

```
            this.label21.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label21.Location = new System.Drawing.Point(739, 126);
            this.label21.Name = "label21";
            this.label21.Size = new System.Drawing.Size(34, 17);
            this.label21.TabIndex = 35;
            this.label21.Text = "m";
            //
            // label18
            //
            this.label18.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label18.Location = new System.Drawing.Point(634, 123);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(24, 21);
            this.label18.TabIndex = 34;
            this.label18.Text = "at";
            //
            // DWorst
            //
            this.DWorst.Enabled = false;
            this.DWorst.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.DWorst.Location = new System.Drawing.Point(672, 121);
            this.DWorst.Name = "DWorst";
            this.DWorst.Size = new System.Drawing.Size(55, 27);
            this.DWorst.TabIndex = 32;
            this.DWorst.Text = "";
            //
            // IF
            //
```

```
            this.IF.Items.AddRange(new object[] {
                                                    "Final",
                                                    "Initial"});
            this.IF.Location = new System.Drawing.Point(672, 6);
            this.IF.Name = "IF";
            this.IF.Size = new System.Drawing.Size(100, 30);
            this.IF.TabIndex = 31;
            this.IF.Text = "Final";
            this.IF.SelectedIndexChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // GroundDist
            //
            this.GroundDist.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.GroundDist.Location = new System.Drawing.Point(221, 121);
            this.GroundDist.Name = "GroundDist";
            this.GroundDist.Size = new System.Drawing.Size(63, 27);
            this.GroundDist.TabIndex = 24;
            this.GroundDist.Text = "";
            this.GroundDist.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // GroundO
            //
            this.GroundO.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.GroundO.Location = new System.Drawing.Point(221, 97);
            this.GroundO.Name = "GroundO";
            this.GroundO.Size = new System.Drawing.Size(63, 27);
            this.GroundO.TabIndex = 23;
            this.GroundO.Text = "";
            this.GroundO.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
```

```
            //
            // HeightO
            //
            this.HeightO.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.HeightO.Location = new System.Drawing.Point(221, 73);
            this.HeightO.Name = "HeightO";
            this.HeightO.Size = new System.Drawing.Size(63, 27);
            this.HeightO.TabIndex = 22;
            this.HeightO.Text = "";
            this.HeightO.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // GroundA
            //
            this.GroundA.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.GroundA.Location = new System.Drawing.Point(221, 48);
            this.GroundA.Name = "GroundA";
            this.GroundA.Size = new System.Drawing.Size(63, 27);
            this.GroundA.TabIndex = 21;
            this.GroundA.Text = "";
            this.GroundA.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // AttachA
            //
            this.AttachA.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.AttachA.Location = new System.Drawing.Point(221, 27);
            this.AttachA.Name = "AttachA";
            this.AttachA.Size = new System.Drawing.Size(63, 27);
            this.AttachA.TabIndex = 20;
```

```
            this.AttachA.Text = "";
            this.AttachA.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // label20
            //
            this.label20.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.label20.BackColor = System.Drawing.SystemColors.Window;
            this.label20.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label20.ForeColor = System.Drawing.Color.Black;
            this.label20.Location = new System.Drawing.Point(819, 157);
            this.label20.Name = "label20";
            this.label20.Size = new System.Drawing.Size(56, 0);
            this.label20.TabIndex = 1;
            this.label20.Text = "PoleB";
            this.label20.TextAlign = System.Drawing.ContentAlignment.TopCenter;
            //
            // label19
            //
            this.label19.BackColor = System.Drawing.SystemColors.Window;
            this.label19.Font = new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label19.ForeColor = System.Drawing.Color.Black;
            this.label19.Location = new System.Drawing.Point(8, 165);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(57, 50);
            this.label19.TabIndex = 0;
            this.label19.Text = "PoleA";
            this.label19.TextAlign = System.Drawing.ContentAlignment.TopCenter;
```

```
            //
            // pictureBox3
            //
            this.pictureBox3.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.pictureBox3.BackColor = System.Drawing.SystemColors.Window;
            this.pictureBox3.Location = new System.Drawing.Point(8, 150);
            this.pictureBox3.Name = "pictureBox3";
            this.pictureBox3.Size = new System.Drawing.Size(879, 31);
            this.pictureBox3.TabIndex = 33;
            this.pictureBox3.TabStop = false;
            //
            // label27
            //
            this.label27.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label27.Location = new System.Drawing.Point(634, 1);
            this.label27.Name = "label27";
            this.label27.Size = new System.Drawing.Size(24, 27);
            this.label27.TabIndex = 50;
            this.label27.Text = "m";
            //
            // ESpan
            //
            this.ESpan.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.ESpan.Location = new System.Drawing.Point(221, 1);
            this.ESpan.Name = "ESpan";
            this.ESpan.Size = new System.Drawing.Size(63, 27);
```

```
            this.ESpan.TabIndex = 19;
            this.ESpan.Text = "";
            this.ESpan.TextChanged += new System.EventHandler(this.condtemp_TextChanged);
            //
            // label91
            //
            this.label91.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label91.Location = new System.Drawing.Point(0, 122);
            this.label91.Name = "label91";
            this.label91.Size = new System.Drawing.Size(192, 25);
            this.label91.TabIndex = 41;
            this.label91.Text = "O: distance from A";
            this.label91.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            //
            // tabPage6
            //
            this.tabPage6.Controls.Add(this.lmin);
            this.tabPage6.Controls.Add(this.tmax);
            this.tabPage6.Controls.Add(this.lmax);
            this.tabPage6.Controls.Add(this.Defaults);
            this.tabPage6.Controls.Add(this.LimError);
            this.tabPage6.Controls.Add(this.Creep);
            this.tabPage6.Controls.Add(this.label5);
            this.tabPage6.Controls.Add(this.Ini3);
            this.tabPage6.Controls.Add(this.Ini2);
            this.tabPage6.Controls.Add(this.Ini1);
            this.tabPage6.Controls.Add(this.label63);
            this.tabPage6.Controls.Add(this.label64);
            this.tabPage6.Controls.Add(this.label65);
            this.tabPage6.Controls.Add(this.label60);
```

```
this.tabPage6.Controls.Add(this.label61);
this.tabPage6.Controls.Add(this.label62);
this.tabPage6.Controls.Add(this.label57);
this.tabPage6.Controls.Add(this.label58);
this.tabPage6.Controls.Add(this.label59);
this.tabPage6.Controls.Add(this.label56);
this.tabPage6.Controls.Add(this.label55);
this.tabPage6.Controls.Add(this.label54);
this.tabPage6.Controls.Add(this.CT3);
this.tabPage6.Controls.Add(this.Ww3);
this.tabPage6.Controls.Add(this.label52);
this.tabPage6.Controls.Add(this.label53);
this.tabPage6.Controls.Add(this.Wc3);
this.tabPage6.Controls.Add(this.Wt3);
this.tabPage6.Controls.Add(this.CT2);
this.tabPage6.Controls.Add(this.Ww2);
this.tabPage6.Controls.Add(this.label49);
this.tabPage6.Controls.Add(this.label51);
this.tabPage6.Controls.Add(this.Wc2);
this.tabPage6.Controls.Add(this.Wt2);
this.tabPage6.Controls.Add(this.CT1);
this.tabPage6.Controls.Add(this.label47);
this.tabPage6.Controls.Add(this.Ww1);
this.tabPage6.Controls.Add(this.label48);
this.tabPage6.Controls.Add(this.label50);
this.tabPage6.Controls.Add(this.Wc1);
this.tabPage6.Controls.Add(this.Wt1);
this.tabPage6.Controls.Add(this.label43);
this.tabPage6.Controls.Add(this.linc);
this.tabPage6.Controls.Add(this.label44);
this.tabPage6.Controls.Add(this.label45);
```

```
this.tabPage6.Controls.Add(this.label46);
this.tabPage6.Controls.Add(this.label42);
this.tabPage6.Controls.Add(this.tinc);
this.tabPage6.Controls.Add(this.label41);
this.tabPage6.Controls.Add(this.label40);
this.tabPage6.Controls.Add(this.label33);
this.tabPage6.Controls.Add(this.tmin);
this.tabPage6.Location = new System.Drawing.Point(4, 34);
this.tabPage6.Name = "tabPage6";
this.tabPage6.Size = new System.Drawing.Size(908, 193);
this.tabPage6.TabIndex = 5;
this.tabPage6.Text = "Limits";
//
// lmin
//
this.lmin.Location = new System.Drawing.Point(230, 129);
this.lmin.Name = "lmin";
this.lmin.Size = new System.Drawing.Size(48, 30);
this.lmin.TabIndex = 3;
this.lmin.Text = "20";
this.lmin.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
this.lmin.TextChanged += new System.EventHandler(this.lmin_TextChanged);
//
// tmax
//
this.tmax.Location = new System.Drawing.Point(542, 55);
this.tmax.Name = "tmax";
this.tmax.Size = new System.Drawing.Size(40, 30);
this.tmax.TabIndex = 1;
this.tmax.Text = "60";
this.tmax.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
```

```
            this.tmax.TextChanged += new System.EventHandler(this.tmax_TextChanged);
            //
            // lmax
            //
            this.lmax.Location = new System.Drawing.Point(534, 129);
            this.lmax.Name = "lmax";
            this.lmax.Size = new System.Drawing.Size(48, 30);
            this.lmax.TabIndex = 4;
            this.lmax.Text = "300";
            this.lmax.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.lmax.TextChanged += new System.EventHandler(this.lmax_TextChanged);
            //
            // Defaults
            //
            this.Defaults.Location = new System.Drawing.Point(710, 240);
            this.Defaults.Name = "Defaults";
            this.Defaults.Size = new System.Drawing.Size(192, 46);
            this.Defaults.TabIndex = 60;
            this.Defaults.Text = "Reset Defaults";
            this.Defaults.Click += new System.EventHandler(this.Defaults_Click);
            //
            // LimError
            //
            this.LimError.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.LimError.ForeColor = System.Drawing.Color.Red;
            this.LimError.Location = new System.Drawing.Point(278, 9);
            this.LimError.Name = "LimError";
            this.LimError.Size = new System.Drawing.Size(615, 37);
            this.LimError.TabIndex = 59;
            //
```

```
// Creep
//
this.Creep.Location = new System.Drawing.Point(173, 342);
this.Creep.Name = "Creep";
this.Creep.Size = new System.Drawing.Size(67, 30);
this.Creep.TabIndex = 57;
this.Creep.Text = "370";
this.Creep.TextChanged += new System.EventHandler(this.Creep_TextChanged);
//
// label5
//
this.label5.Location = new System.Drawing.Point(10, 342);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(159, 23);
this.label5.TabIndex = 58;
this.label5.Text = "Creep Constant";
//
// Ini3
//
this.Ini3.AllowDrop = true;
this.Ini3.Enabled = false;
this.Ini3.Items.AddRange(new object[] {
                                                "Initial",
                                                "Final"});
this.Ini3.Location = new System.Drawing.Point(605, 295);
this.Ini3.Name = "Ini3";
this.Ini3.Size = new System.Drawing.Size(96, 30);
this.Ini3.TabIndex = 52;
this.Ini3.Text = "Final";
//
// Ini2
```

```
//
this.Ini2.AllowDrop = true;
this.Ini2.Enabled = false;
this.Ini2.Items.AddRange(new object[] {
                                                "Initial",
                                                "Final"});
this.Ini2.Location = new System.Drawing.Point(605, 249);
this.Ini2.Name = "Ini2";
this.Ini2.Size = new System.Drawing.Size(96, 30);
this.Ini2.TabIndex = 51;
this.Ini2.Text = "Initial";
//
// Ini1
//
this.Ini1.AllowDrop = true;
this.Ini1.Items.AddRange(new object[] {
                                                "Initial",
                                                "Final"});
this.Ini1.Location = new System.Drawing.Point(605, 203);
this.Ini1.Name = "Ini1";
this.Ini1.Size = new System.Drawing.Size(96, 30);
this.Ini1.TabIndex = 50;
this.Ini1.Text = "Final";
//
// label63
//
this.label63.Location = new System.Drawing.Point(883, 129);
this.label63.Name = "label63";
this.label63.Size = new System.Drawing.Size(55, 22);
this.label63.TabIndex = 49;
this.label63.Text = "m";
```

```
//
// label64
//
this.label64.Location = new System.Drawing.Point(581, 129);
this.label64.Name = "label64";
this.label64.Size = new System.Drawing.Size(21, 22);
this.label64.TabIndex = 48;
this.label64.Text = "m";
//
// label65
//
this.label65.Location = new System.Drawing.Point(278, 129);
this.label65.Name = "label65";
this.label65.Size = new System.Drawing.Size(28, 22);
this.label65.TabIndex = 47;
this.label65.Text = "m";
//
// label60
//
this.label60.Location = new System.Drawing.Point(883, 55);
this.label60.Name = "label60";
this.label60.Size = new System.Drawing.Size(41, 23);
this.label60.TabIndex = 44;
this.label60.Text = "\'C";
//
// label61
//
this.label61.Location = new System.Drawing.Point(581, 55);
this.label61.Name = "label61";
this.label61.Size = new System.Drawing.Size(33, 23);
this.label61.TabIndex = 43;
```

```
this.label61.Text = "\'C";
//
// label62
//
this.label62.Location = new System.Drawing.Point(278, 55);
this.label62.Name = "label62";
this.label62.Size = new System.Drawing.Size(29, 23);
this.label62.TabIndex = 42;
this.label62.Text = "\'C";
//
// label57
//
this.label57.Location = new System.Drawing.Point(221, 295);
this.label57.Name = "label57";
this.label57.Size = new System.Drawing.Size(29, 23);
this.label57.TabIndex = 39;
this.label57.Text = "\'C";
//
// label58
//
this.label58.Location = new System.Drawing.Point(221, 249);
this.label58.Name = "label58";
this.label58.Size = new System.Drawing.Size(32, 23);
this.label58.TabIndex = 38;
this.label58.Text = "\'C";
//
// label59
//
this.label59.Location = new System.Drawing.Point(221, 203);
this.label59.Name = "label59";
this.label59.Size = new System.Drawing.Size(32, 23);
```

```
this.label59.TabIndex = 37;
this.label59.Text = "\'C";
//
// label56
//
this.label56.Location = new System.Drawing.Point(566, 295);
this.label56.Name = "label56";
this.label56.Size = new System.Drawing.Size(33, 23);
this.label56.TabIndex = 36;
this.label56.Text = "Pa";
//
// label55
//
this.label55.Location = new System.Drawing.Point(566, 249);
this.label55.Name = "label55";
this.label55.Size = new System.Drawing.Size(33, 23);
this.label55.TabIndex = 35;
this.label55.Text = "Pa";
//
// label54
//
this.label54.Location = new System.Drawing.Point(566, 203);
this.label54.Name = "label54";
this.label54.Size = new System.Drawing.Size(33, 23);
this.label54.TabIndex = 34;
this.label54.Text = "Pa";
//
// CT3
//
this.CT3.AllowDrop = true;
this.CT3.ItemHeight = 25;
```

```
this.CT3.Items.AddRange(new object[] {
                                                "C Value",
                                                "% UTS",
                                                "Tension"});
this.CT3.Location = new System.Drawing.Point(259, 295);
this.CT3.Name = "CT3";
this.CT3.Size = new System.Drawing.Size(115, 30);
this.CT3.TabIndex = 33;
this.CT3.Text = "% UTS";
//
// Ww3
//
this.Ww3.Location = new System.Drawing.Point(510, 295);
this.Ww3.Name = "Ww3";
this.Ww3.Size = new System.Drawing.Size(52, 30);
this.Ww3.TabIndex = 30;
this.Ww3.Text = "700";
//
// label52
//
this.label52.Location = new System.Drawing.Point(442, 295);
this.label52.Name = "label52";
this.label52.Size = new System.Drawing.Size(67, 23);
this.label52.TabIndex = 32;
this.label52.Text = "Wind";
//
// label53
//
this.label53.Location = new System.Drawing.Point(10, 295);
this.label53.Name = "label53";
this.label53.Size = new System.Drawing.Size(153, 23);
```

```
this.label53.TabIndex = 31;
this.label53.Text = "3 : Temperature";
//
// Wc3
//
this.Wc3.Location = new System.Drawing.Point(376, 295);
this.Wc3.Name = "Wc3";
this.Wc3.Size = new System.Drawing.Size(58, 30);
this.Wc3.TabIndex = 29;
this.Wc3.Text = "40";
//
// Wt3
//
this.Wt3.Location = new System.Drawing.Point(173, 295);
this.Wt3.Name = "Wt3";
this.Wt3.Size = new System.Drawing.Size(38, 30);
this.Wt3.TabIndex = 28;
this.Wt3.Text = "-5";
//
// CT2
//
this.CT2.AllowDrop = true;
this.CT2.Items.AddRange(new object[] {
                                                "C Value",
                                                "% UTS",
                                                "Tension"});
this.CT2.Location = new System.Drawing.Point(259, 249);
this.CT2.Name = "CT2";
this.CT2.Size = new System.Drawing.Size(114, 30);
this.CT2.TabIndex = 27;
this.CT2.Text = "% UTS";
```

```
//
// Ww2
//
this.Ww2.Location = new System.Drawing.Point(510, 249);
this.Ww2.Name = "Ww2";
this.Ww2.Size = new System.Drawing.Size(52, 30);
this.Ww2.TabIndex = 24;
this.Ww2.Text = "700";
//
// label49
//
this.label49.Location = new System.Drawing.Point(442, 249);
this.label49.Name = "label49";
this.label49.Size = new System.Drawing.Size(66, 23);
this.label49.TabIndex = 26;
this.label49.Text = "Wind ";
//
// label51
//
this.label51.Location = new System.Drawing.Point(10, 249);
this.label51.Name = "label51";
this.label51.Size = new System.Drawing.Size(153, 23);
this.label51.TabIndex = 25;
this.label51.Text = "2 : Temperature";
//
// Wc2
//
this.Wc2.Location = new System.Drawing.Point(376, 249);
this.Wc2.Name = "Wc2";
this.Wc2.Size = new System.Drawing.Size(58, 30);
this.Wc2.TabIndex = 23;
```

```
            this.Wc2.Text = "50";
            //
            // Wt2
            //
            this.Wt2.Location = new System.Drawing.Point(173, 249);
            this.Wt2.Name = "Wt2";
            this.Wt2.Size = new System.Drawing.Size(43, 30);
            this.Wt2.TabIndex = 22;
            this.Wt2.Text = "-5";
            //
            // CT1
            //
            this.CT1.AllowDrop = true;
            this.CT1.Items.AddRange(new object[] {
                                                  "C Value",
                                                  "% UTS",
                                                  "Tension"});
            this.CT1.Location = new System.Drawing.Point(259, 203);
            this.CT1.Name = "CT1";
            this.CT1.Size = new System.Drawing.Size(114, 30);
            this.CT1.TabIndex = 21;
            this.CT1.Text = "C Value";
            this.CT1.SelectedIndexChanged += new System.EventHandler(this.CT1_SelectedIndexChanged);
            //
            // label47
            //
            this.label47.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F,
((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Underline))),
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label47.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label47.Location = new System.Drawing.Point(10, 166);
```

```
this.label47.Name = "label47";
this.label47.Size = new System.Drawing.Size(499, 32);
this.label47.TabIndex = 20;
this.label47.Text = "Conductor Tension Criteria";
//
// Ww1
//
this.Ww1.Location = new System.Drawing.Point(510, 203);
this.Ww1.Name = "Ww1";
this.Ww1.Size = new System.Drawing.Size(52, 30);
this.Ww1.TabIndex = 16;
this.Ww1.Text = "0";
//
// label48
//
this.label48.Location = new System.Drawing.Point(442, 203);
this.label48.Name = "label48";
this.label48.Size = new System.Drawing.Size(66, 23);
this.label48.TabIndex = 19;
this.label48.Text = "Wind ";
//
// label50
//
this.label50.Location = new System.Drawing.Point(10, 203);
this.label50.Name = "label50";
this.label50.Size = new System.Drawing.Size(153, 23);
this.label50.TabIndex = 17;
this.label50.Text = "1:  Temperature ";
//
// Wc1
//
```

```
            this.Wc1.Location = new System.Drawing.Point(376, 203);
            this.Wc1.Name = "Wc1";
            this.Wc1.Size = new System.Drawing.Size(58, 30);
            this.Wc1.TabIndex = 15;
            this.Wc1.Text = "1800";
            this.Wc1.TextChanged += new System.EventHandler(this.Wc1_TextChanged);
            //
            // Wt1
            //
            this.Wt1.Location = new System.Drawing.Point(173, 203);
            this.Wt1.Name = "Wt1";
            this.Wt1.Size = new System.Drawing.Size(43, 30);
            this.Wt1.TabIndex = 14;
            this.Wt1.Text = "15";
            //
            // label43
            //
            this.label43.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F,
((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Underline))),
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label43.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label43.Location = new System.Drawing.Point(10, 92);
            this.label43.Name = "label43";
            this.label43.Size = new System.Drawing.Size(364, 33);
            this.label43.TabIndex = 13;
            this.label43.Text = "Span length Range";
            //
            // linc
            //
            this.linc.Location = new System.Drawing.Point(846, 129);
            this.linc.Name = "linc";
```

```
this.linc.Size = new System.Drawing.Size(36, 30);
this.linc.TabIndex = 5;
this.linc.Text = "20";
this.linc.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
this.linc.TextChanged += new System.EventHandler(this.linc_TextChanged);
//
// label44
//
this.label44.Location = new System.Drawing.Point(614, 129);
this.label44.Name = "label44";
this.label44.Size = new System.Drawing.Size(221, 22);
this.label44.TabIndex = 11;
this.label44.Text = "Span length Increment";
//
// label45
//
this.label45.Location = new System.Drawing.Point(307, 129);
this.label45.Name = "label45";
this.label45.Size = new System.Drawing.Size(223, 22);
this.label45.TabIndex = 10;
this.label45.Text = "Maximum span length";
//
// label46
//
this.label46.Location = new System.Drawing.Point(10, 129);
this.label46.Name = "label46";
this.label46.Size = new System.Drawing.Size(222, 22);
this.label46.TabIndex = 9;
this.label46.Text = "Minimum span length";
//
// label42
```

```
            //
            this.label42.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F,
((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Underline))),
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label42.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label42.Location = new System.Drawing.Point(10, 9);
            this.label42.Name = "label42";
            this.label42.Size = new System.Drawing.Size(364, 33);
            this.label42.TabIndex = 6;
            this.label42.Text = "Temperature Range";
            //
            // tinc
            //
            this.tinc.Location = new System.Drawing.Point(846, 55);
            this.tinc.Name = "tinc";
            this.tinc.Size = new System.Drawing.Size(36, 30);
            this.tinc.TabIndex = 2;
            this.tinc.Text = "5";
            this.tinc.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.tinc.TextChanged += new System.EventHandler(this.tinc_TextChanged);
            //
            // label41
            //
            this.label41.Location = new System.Drawing.Point(614, 55);
            this.label41.Name = "label41";
            this.label41.Size = new System.Drawing.Size(221, 23);
            this.label41.TabIndex = 4;
            this.label41.Text = "Temperature Increment";
            //
            // label40
            //
```

```
this.label40.Location = new System.Drawing.Point(307, 55);
this.label40.Name = "label40";
this.label40.Size = new System.Drawing.Size(229, 23);
this.label40.TabIndex = 3;
this.label40.Text = "Maximum Temperature";
//
// label33
//
this.label33.Location = new System.Drawing.Point(10, 55);
this.label33.Name = "label33";
this.label33.Size = new System.Drawing.Size(222, 23);
this.label33.TabIndex = 2;
this.label33.Text = "Minimum Temperature";
//
// tmin
//
this.tmin.Location = new System.Drawing.Point(238, 55);
this.tmin.Name = "tmin";
this.tmin.Size = new System.Drawing.Size(40, 30);
this.tmin.TabIndex = 0;
this.tmin.Text = "-5";
this.tmin.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
this.tmin.TextChanged += new System.EventHandler(this.tmin_TextChanged);
//
// tabPage5
//
this.tabPage5.Controls.Add(this.helpindex);
this.tabPage5.Controls.Add(this.Helptext);
this.tabPage5.Location = new System.Drawing.Point(4, 34);
this.tabPage5.Name = "tabPage5";
this.tabPage5.Size = new System.Drawing.Size(908, 193);
```

```
            this.tabPage5.TabIndex = 4;
            this.tabPage5.Text = "Help";
            //
            // helpindex
            //
            this.helpindex.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                | System.Windows.Forms.AnchorStyles.Left)));
            this.helpindex.Font = new System.Drawing.Font("Arial", 13.8F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.helpindex.ItemHeight = 26;
            this.helpindex.Location = new System.Drawing.Point(8, 8);
            this.helpindex.Name = "helpindex";
            this.helpindex.Size = new System.Drawing.Size(248, 104);
            this.helpindex.TabIndex = 0;
            this.helpindex.SelectedIndexChanged += new System.EventHandler(this.helpindex_SelectedIndexChanged);
            //
            // Helptext
            //
            this.Helptext.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
            this.Helptext.Font = new System.Drawing.Font("Arial", 13.8F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Helptext.Location = new System.Drawing.Point(264, 8);
            this.Helptext.Name = "Helptext";
            this.Helptext.ReadOnly = true;
            this.Helptext.Size = new System.Drawing.Size(482, 151);
            this.Helptext.TabIndex = 1;
            this.Helptext.Text = "";
```

```
            //
            // tabPage7
            //
            this.tabPage7.Controls.Add(this.richTextBox1);
            this.tabPage7.Controls.Add(this.pictureBox1);
            this.tabPage7.Location = new System.Drawing.Point(4, 34);
            this.tabPage7.Name = "tabPage7";
            this.tabPage7.Size = new System.Drawing.Size(908, 193);
            this.tabPage7.TabIndex = 6;
            this.tabPage7.Text = "About";
            //
            // richTextBox1
            //
            this.richTextBox1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.richTextBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.richTextBox1.Location = new System.Drawing.Point(168, 8);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.ReadOnly = true;
            this.richTextBox1.Size = new System.Drawing.Size(578, 181);
            this.richTextBox1.TabIndex = 1;
            this.richTextBox1.Text = @"RSAT is a design tool for the production of sag and tension data. RSAT is not intended to
replace design packages such as PLSCADD which are recommended for the design of high voltage lines.  Rather RSAT serves as a
simple checking tool for large designs. RSAT is recommended for smaller flat line designs. RSAT was developed by the Industry
Association Resource Centre. Technical Queries can be forwarded to G. Stanford email: gareth.stanford@Eskom.co.za. Tel: +27 11 871
2431";
            //
            // pictureBox1
```

```
            //
            this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(24, 8);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(149, 334);
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // SagAndTension
            //
            this.SagAndTension.DataSetName = "SagAndTension";
            this.SagAndTension.Locale = new System.Globalization.CultureInfo("en-ZA");
            this.SagAndTension.Tables.AddRange(new System.Data.DataTable[] {
                                                                this.dataTable1,
                                                                this.dataTable2});
            //
            // dataView1
            //
            this.dataView1.Table = this.dataTable1;
            //
            // Area
            //
            this.Area.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Area.Location = new System.Drawing.Point(463, 125);
            this.Area.Name = "Area";
            this.Area.ReadOnly = true;
            this.Area.Size = new System.Drawing.Size(64, 30);
            this.Area.TabIndex = 6;
            this.Area.Text = "Area";
            this.Area.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
```

```
            this.Area.TextChanged += new System.EventHandler(this.Area_TextChanged);
            //
            // IMod
            //
            this.IMod.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.IMod.Location = new System.Drawing.Point(684, 95);
            this.IMod.Name = "IMod";
            this.IMod.ReadOnly = true;
            this.IMod.Size = new System.Drawing.Size(59, 30);
            this.IMod.TabIndex = 10;
            this.IMod.Text = "IMod";
            this.IMod.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.IMod.TextChanged += new System.EventHandler(this.IMod_TextChanged);
            //
            // FMod
            //
            this.FMod.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.FMod.Location = new System.Drawing.Point(684, 125);
            this.FMod.Name = "FMod";
            this.FMod.ReadOnly = true;
            this.FMod.Size = new System.Drawing.Size(59, 30);
            this.FMod.TabIndex = 11;
            this.FMod.Text = "FMod";
            this.FMod.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.FMod.TextChanged += new System.EventHandler(this.FMod_TextChanged);
            //
            // Mass
            //
```

```
            this.Mass.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Mass.Location = new System.Drawing.Point(684, 35);
            this.Mass.Name = "Mass";
            this.Mass.ReadOnly = true;
            this.Mass.Size = new System.Drawing.Size(59, 30);
            this.Mass.TabIndex = 8;
            this.Mass.Text = "Mass";
            this.Mass.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.Mass.TextChanged += new System.EventHandler(this.Mass_TextChanged);
            //
            // UTS
            //
            this.UTS.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.UTS.Location = new System.Drawing.Point(463, 65);
            this.UTS.Name = "UTS";
            this.UTS.ReadOnly = true;
            this.UTS.Size = new System.Drawing.Size(64, 30);
            this.UTS.TabIndex = 4;
            this.UTS.Text = "U.T.S";
            this.UTS.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.UTS.TextChanged += new System.EventHandler(this.UTS_TextChanged);
            //
            // Diameter
            //
            this.Diameter.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Diameter.Location = new System.Drawing.Point(463, 155);
            this.Diameter.Name = "Diameter";
            this.Diameter.ReadOnly = true;
```

```
            this.Diameter.Size = new System.Drawing.Size(64, 30);
            this.Diameter.TabIndex = 7;
            this.Diameter.Text = "Diameter";
            this.Diameter.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.Diameter.TextChanged += new System.EventHandler(this.Diameter_TextChanged);
            //
            // EDT
            //
            this.EDT.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.EDT.Location = new System.Drawing.Point(463, 35);
            this.EDT.Name = "EDT";
            this.EDT.ReadOnly = true;
            this.EDT.Size = new System.Drawing.Size(64, 30);
            this.EDT.TabIndex = 45;
            this.EDT.Text = "EDT";
            this.EDT.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label28
            //
            this.label28.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label28.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label28.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label28.Location = new System.Drawing.Point(362, 36);
            this.label28.Name = "label28";
            this.label28.Size = new System.Drawing.Size(101, 27);
            this.label28.TabIndex = 27;
            this.label28.Text = "Every Day Tension";
            //
            // label29
```

```
            //
            this.label29.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label29.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label29.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label29.Location = new System.Drawing.Point(362, 66);
            this.label29.Name = "label29";
            this.label29.Size = new System.Drawing.Size(101, 27);
            this.label29.TabIndex = 28;
            this.label29.Text = "Tensile Strength";
            //
            // label30
            //
            this.label30.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label30.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label30.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label30.Location = new System.Drawing.Point(362, 97);
            this.label30.Name = "label30";
            this.label30.Size = new System.Drawing.Size(101, 26);
            this.label30.TabIndex = 29;
            this.label30.Text = "Catenary Value";
            //
            // CValue
            //
            this.CValue.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.CValue.Location = new System.Drawing.Point(463, 95);
            this.CValue.Name = "CValue";
            this.CValue.ReadOnly = true;
            this.CValue.Size = new System.Drawing.Size(64, 30);
```

```
            this.CValue.TabIndex = 5;
            this.CValue.Text = "C Value";
            this.CValue.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.CValue.TextChanged += new System.EventHandler(this.CValue_TextChanged);
            //
            // label31
            //
            this.label31.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label31.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label31.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label31.Location = new System.Drawing.Point(362, 127);
            this.label31.Name = "label31";
            this.label31.Size = new System.Drawing.Size(101, 26);
            this.label31.TabIndex = 31;
            this.label31.Text = "Area";
            //
            // label32
            //
            this.label32.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label32.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label32.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label32.Location = new System.Drawing.Point(362, 156);
            this.label32.Name = "label32";
            this.label32.Size = new System.Drawing.Size(101, 27);
            this.label32.TabIndex = 32;
            this.label32.Text = "Diameter";
            //
            // label34
            //
```

```
            this.label34.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label34.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label34.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label34.Location = new System.Drawing.Point(578, 127);
            this.label34.Name = "label34";
            this.label34.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.label34.Size = new System.Drawing.Size(104, 26);
            this.label34.TabIndex = 36;
            this.label34.Text = "Final Modulus";
            //
            // label35
            //
            this.label35.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label35.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label35.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label35.Location = new System.Drawing.Point(578, 97);
            this.label35.Name = "label35";
            this.label35.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.label35.Size = new System.Drawing.Size(104, 26);
            this.label35.TabIndex = 35;
            this.label35.Text = "Initial Modulus";
            //
            // label39
            //
            this.label39.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label39.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label39.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label39.Location = new System.Drawing.Point(578, 66);
```

```
            this.label39.Name = "label39";
            this.label39.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.label39.Size = new System.Drawing.Size(243, 27);
            this.label39.TabIndex = 38;
            this.label39.Text = "Temperature Coefficient";
            //
            // Coefficient
            //
            this.Coefficient.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Coefficient.Location = new System.Drawing.Point(684, 65);
            this.Coefficient.Name = "Coefficient";
            this.Coefficient.ReadOnly = true;
            this.Coefficient.Size = new System.Drawing.Size(59, 30);
            this.Coefficient.TabIndex = 9;
            this.Coefficient.Text = "Coefficient";
            this.Coefficient.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            this.Coefficient.TextChanged += new System.EventHandler(this.Coefficient_TextChanged);
            //
            // picture
            //
            this.picture.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.picture.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.picture.Image = ((System.Drawing.Image)(resources.GetObject("picture.Image")));
            this.picture.Location = new System.Drawing.Point(444, 39);
            this.picture.Name = "picture";
            this.picture.Size = new System.Drawing.Size(192, 139);
            this.picture.TabIndex = 39;
            this.picture.TabStop = false;
            this.picture.Click += new System.EventHandler(this.pictureBox1_Click);
```

```
            //
            // ntype
            //
            this.ntype.Font = new System.Drawing.Font("Arial", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.ntype.Location = new System.Drawing.Point(1, 62);
            this.ntype.Name = "ntype";
            this.ntype.Size = new System.Drawing.Size(168, 32);
            this.ntype.TabIndex = 0;
            this.ntype.Text = "";
            this.ntype.Visible = false;
            this.ntype.TextChanged += new System.EventHandler(this.ntype_TextChanged);
            //
            // ncond
            //
            this.ncond.Font = new System.Drawing.Font("Arial", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.ncond.Location = new System.Drawing.Point(168, 62);
            this.ncond.Name = "ncond";
            this.ncond.Size = new System.Drawing.Size(192, 32);
            this.ncond.TabIndex = 3;
            this.ncond.Text = "";
            this.ncond.Visible = false;
            this.ncond.TextChanged += new System.EventHandler(this.ncond_TextChanged);
            //
            // Save
            //
            this.Save.Enabled = false;
            this.Save.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Save.Location = new System.Drawing.Point(240, 99);
```

```
            this.Save.Name = "Save";
            this.Save.Size = new System.Drawing.Size(92, 27);
            this.Save.TabIndex = 12;
            this.Save.Text = "Save";
            this.Save.Visible = false;
            this.Save.Click += new System.EventHandler(this.Save_Click);
            //
            // Cancel
            //
            this.Cancel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Cancel.Location = new System.Drawing.Point(240, 145);
            this.Cancel.Name = "Cancel";
            this.Cancel.Size = new System.Drawing.Size(92, 26);
            this.Cancel.TabIndex = 13;
            this.Cancel.Text = "Cancel";
            this.Cancel.Visible = false;
            this.Cancel.Click += new System.EventHandler(this.Cancel_Click);
            //
            // image
            //
            this.image.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.image.Location = new System.Drawing.Point(581, 157);
            this.image.Name = "image";
            this.image.Size = new System.Drawing.Size(230, 27);
            this.image.TabIndex = 45;
            this.image.Text = "";
            this.image.Visible = false;
            //
            // ncreep
```

```
            //
            this.ncreep.Font = new System.Drawing.Font("Arial", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.ncreep.Location = new System.Drawing.Point(29, 136);
            this.ncreep.Name = "ncreep";
            this.ncreep.Size = new System.Drawing.Size(135, 32);
            this.ncreep.TabIndex = 1;
            this.ncreep.Text = "";
            this.ncreep.Visible = false;
            this.ncreep.TextChanged += new System.EventHandler(this.ncreep_TextChanged);
            //
            // Clabel
            //
            this.Clabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.Clabel.ForeColor = System.Drawing.SystemColors.ControlText;
            this.Clabel.Location = new System.Drawing.Point(4, 92);
            this.Clabel.Name = "Clabel";
            this.Clabel.Size = new System.Drawing.Size(178, 44);
            this.Clabel.TabIndex = 46;
            this.Clabel.Text = "Creep - ACSR = 370, AAAC = 465";
            this.Clabel.Visible = false;
            //
            // label73
            //
            this.label73.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label73.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label73.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label73.Location = new System.Drawing.Point(521, 97);
            this.label73.Name = "label73";
```

```
            this.label73.Size = new System.Drawing.Size(59, 26);
            this.label73.TabIndex = 49;
            //
            // label74
            //
            this.label74.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label74.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label74.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label74.Location = new System.Drawing.Point(521, 66);
            this.label74.Name = "label74";
            this.label74.Size = new System.Drawing.Size(59, 27);
            this.label74.TabIndex = 48;
            this.label74.Text = "N";
            //
            // label75
            //
            this.label75.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label75.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label75.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label75.Location = new System.Drawing.Point(521, 36);
            this.label75.Name = "label75";
            this.label75.Size = new System.Drawing.Size(59, 27);
            this.label75.TabIndex = 47;
            this.label75.Text = "N";
            //
            // label76
            //
            this.label76.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
```

```
            this.label76.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label76.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label76.Location = new System.Drawing.Point(742, 36);
            this.label76.Name = "label76";
            this.label76.Size = new System.Drawing.Size(78, 27);
            this.label76.TabIndex = 52;
            this.label76.Text = "kg/km";
            //
            // label77
            //
            this.label77.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label77.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label77.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label77.Location = new System.Drawing.Point(521, 156);
            this.label77.Name = "label77";
            this.label77.Size = new System.Drawing.Size(59, 27);
            this.label77.TabIndex = 51;
            this.label77.Text = "mm";
            //
            // label78
            //
            this.label78.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label78.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label78.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label78.Location = new System.Drawing.Point(521, 127);
            this.label78.Name = "label78";
            this.label78.Size = new System.Drawing.Size(59, 26);
            this.label78.TabIndex = 50;
```

```
            this.label78.Text = "mm^2";
            //
            // label79
            //
            this.label79.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label79.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label79.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label79.Location = new System.Drawing.Point(742, 127);
            this.label79.Name = "label79";
            this.label79.Size = new System.Drawing.Size(78, 26);
            this.label79.TabIndex = 55;
            this.label79.Text = "N/mm^2";
            //
            // label80
            //
            this.label80.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label80.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label80.ForeColor = System.Drawing.SystemColors.ControlText;
            this.label80.Location = new System.Drawing.Point(742, 97);
            this.label80.Name = "label80";
            this.label80.Size = new System.Drawing.Size(78, 26);
            this.label80.TabIndex = 54;
            this.label80.Text = "N/mm^2";
            //
            // label81
            //
            this.label81.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
            this.label81.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
```

```
this.label81.ForeColor = System.Drawing.SystemColors.ControlText;
this.label81.Location = new System.Drawing.Point(742, 66);
this.label81.Name = "label81";
this.label81.Size = new System.Drawing.Size(78, 27);
this.label81.TabIndex = 53;
this.label81.Text = "C\'x10^-6";
//
// printPreviewDialog1
//
this.printPreviewDialog1.AutoScrollMargin = new System.Drawing.Size(0, 0);
this.printPreviewDialog1.AutoScrollMinSize = new System.Drawing.Size(0, 0);
this.printPreviewDialog1.ClientSize = new System.Drawing.Size(400, 300);
this.printPreviewDialog1.Document = this.printDocument1;
this.printPreviewDialog1.Enabled = true;
this.printPreviewDialog1.Icon = ((System.Drawing.Icon)(resources.GetObject("printPreviewDialog1.Icon")));
this.printPreviewDialog1.Location = new System.Drawing.Point(126, 159);
this.printPreviewDialog1.MinimumSize = new System.Drawing.Size(375, 250);
this.printPreviewDialog1.Name = "printPreviewDialog1";
this.printPreviewDialog1.TransparencyKey = System.Drawing.Color.Empty;
this.printPreviewDialog1.Visible = false;
//
// panel1
//
this.panel1.Location = new System.Drawing.Point(-1, 28);
this.panel1.Name = "panel1";
this.panel1.Size = new System.Drawing.Size(1038, 157);
this.panel1.TabIndex = 56;
//
// suggest
//
```

```
            this.suggest.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.suggest.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.suggest.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.suggest.ForeColor = System.Drawing.Color.RoyalBlue;
            this.suggest.Location = new System.Drawing.Point(308, 3);
            this.suggest.Name = "suggest";
            this.suggest.Size = new System.Drawing.Size(605, 25);
            this.suggest.TabIndex = 57;
            this.suggest.Text = "Select a type of conductor in the left box below";
            //
            // label68
            //
            this.label68.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.label68.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label68.ForeColor = System.Drawing.Color.RoyalBlue;
            this.label68.Location = new System.Drawing.Point(182, 3);
            this.label68.Name = "label68";
            this.label68.Size = new System.Drawing.Size(125, 25);
            this.label68.TabIndex = 58;
            this.label68.Text = "Suggestion:";
            //
            // MainView
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(6, 15);
            this.BackgroundImage = ((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage")));
            this.ClientSize = new System.Drawing.Size(944, 458);
            this.Controls.Add(this.label68);
```

```
this.Controls.Add(this.suggest);
this.Controls.Add(this.Diameter);
this.Controls.Add(this.Area);
this.Controls.Add(this.CValue);
this.Controls.Add(this.UTS);
this.Controls.Add(this.EDT);
this.Controls.Add(this.FMod);
this.Controls.Add(this.IMod);
this.Controls.Add(this.Coefficient);
this.Controls.Add(this.ncreep);
this.Controls.Add(this.image);
this.Controls.Add(this.ncond);
this.Controls.Add(this.ntype);
this.Controls.Add(this.Mass);
this.Controls.Add(this.label79);
this.Controls.Add(this.label80);
this.Controls.Add(this.label81);
this.Controls.Add(this.label76);
this.Controls.Add(this.label77);
this.Controls.Add(this.label78);
this.Controls.Add(this.label73);
this.Controls.Add(this.label74);
this.Controls.Add(this.label75);
this.Controls.Add(this.Clabel);
this.Controls.Add(this.toolBar1);
this.Controls.Add(this.Cancel);
this.Controls.Add(this.Save);
this.Controls.Add(this.label39);
this.Controls.Add(this.label34);
this.Controls.Add(this.label35);
this.Controls.Add(this.label32);
```

```
this.Controls.Add(this.label31);
this.Controls.Add(this.label30);
this.Controls.Add(this.label29);
this.Controls.Add(this.label28);
this.Controls.Add(this.tabControl1);
this.Controls.Add(this.statusBar1);
this.Controls.Add(this.label6);
this.Controls.Add(this.CLB);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.CTLB);
this.Controls.Add(this.picture);
this.Controls.Add(this.panel1);
this.Controls.Add(this.label1);
this.HelpButton = true;
this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
this.Menu = this.mainMenu1;
this.Name = "MainView";
this.Text = "R.S.A.T - Rev 5.1";
this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
this.SizeChanged += new System.EventHandler(this.MainView_SizeChanged);
this.tabControl1.ResumeLayout(false);
this.tabPage1.ResumeLayout(false);
this.tabPage2.ResumeLayout(false);
((System.ComponentModel.ISupportInitialize)(this.dataGrid1)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.dataTable1)).EndInit();
this.tabPage3.ResumeLayout(false);
((System.ComponentModel.ISupportInitialize)(this.dataGrid2)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.dataTable2)).EndInit();
this.tabPage4.ResumeLayout(false);
this.tabPage6.ResumeLayout(false);
```

```
        this.tabPage5.ResumeLayout(false);
        this.tabPage7.ResumeLayout(false);
        ((System.ComponentModel.ISupportInitialize)(this.SagAndTension)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.dataView1)).EndInit();
        this.ResumeLayout(false);

}
//Graphics for the Graphs
private void Form_Paint(object sender,System.Windows.Forms.PaintEventArgs e)
{
}
//Graphics for the Clearance Calculator
private void Form_Paint2(object sender,System.Windows.Forms.PaintEventArgs e)
{
}

#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{

        Application.Run(new MainView());

}


// General functionality menus and buttons
```

```
    private void toolBar1_ButtonClick(object sender, System.Windows.Forms.ToolBarButtonClickEventArgs e)
    {
        // Evaluate the Button property to determine which button was clicked.
        switch(toolBar1.Buttons.IndexOf(e.Button))
        {
            case 0:
                // Initialize the SaveFileDialog to specify the RTF extension for the file.
                saveFileDialog1.DefaultExt = "*.doc";
                saveFileDialog1.Filter = "Word File|*.doc";


                // Determine if the user selected a file name from the saveFileDialog.
                if(saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&
                    saveFileDialog1.FileName.Length > 0)
                {
                    ExportToText (saveFileDialog1.FileName, true, 30);//XmlDocument myDoc = new XmlDocument();

                }
                break;
            case 1:
                page = 1;
                line = 0;

                this.printDocument1.PrintPage += new
System.Drawing.Printing.PrintPageEventHandler(this.printDocument1_PrintPage);
                printDocument1.DocumentName = "Rsat";
                if(tabControl1.SelectedIndex==2||tabControl1.SelectedIndex==3)
                    printDocument1.DefaultPageSettings.Landscape = true;
                else
                    printDocument1.DefaultPageSettings.Landscape = false;
                printDocument1.Print();
```

```
                    printDocument1.Dispose();
                    break;
                case 2:
                    // Printpreview the file.
                    page = 1;
                    line = 0;

                    this.printDocument1.PrintPage += new
    System.Drawing.Printing.PrintPageEventHandler(this.printDocument1_PrintPage);
                    printDocument1.DocumentName = "Rsat";
                    printPreviewDialog1.Document = printDocument1;
                    if(tabControl1.SelectedIndex==2||tabControl1.SelectedIndex==3)
                        printDocument1.DefaultPageSettings.Landscape = true;
                    else
                        printDocument1.DefaultPageSettings.Landscape = false;
                    printPreviewDialog1.ShowDialog();
                    printDocument1.Dispose();
                    break;
                case 3:
                    tabControl1.SelectedIndex=4;
                    break;
                case 4:
                    // change status of visibilty of buttons
                    Save.Visible = Cancel.Visible = true;CTLB.ClearSelected();
                    // Add a new type to conductor type list box CTLB
                    CTLB.Items.Insert(0,"New Type");
                    //Clear Conductor data
                    CValue.Text = CValue.Text = Area.Text = Coefficient.Text = IMod.Text = FMod.Text = Mass.Text = UTS.Text =
Diameter.Text = Creep.Text = EDT.Text = "";
                    picture.Image = Image.FromFile("intro image.jpg");
                    ncond.Visible = true;
```

```
                CLB.Visible = false;
                toolBar1.Enabled = menuItem29.Enabled= false;
                // Insert code to open the file.
                break;
            case 5:
                tabControl1.SelectedIndex=5;
                // Insert code to open the file.
                break;
        }

    }

    private void menuItem15_Click(object sender, System.EventArgs e)
    {
        tabControl1.SelectedIndex=4;
    }

    private void menuItem29_Click(object sender, System.EventArgs e)
    {
        // change status of visibilty of buttons
        Save.Visible = Cancel.Visible = true;CTLB.ClearSelected();
        // Add a new type to conductor type list box CTLB
        CTLB.Items.Insert(0,"New Type");
        //Clear Conductor data
        CValue.Text = CValue.Text = Area.Text = Coefficient.Text = IMod.Text = FMod.Text = Mass.Text = UTS.Text =
Diameter.Text = Creep.Text = EDT.Text = "";
        picture.Image = Image.FromFile("intro image.jpg");
        ncond.Visible = true;
        CLB.Visible = false;
        toolBar1.Enabled = menuItem29.Enabled=false;
        // Insert code to open the file.
```

```
}

private void menuItem17_Click(object sender, System.EventArgs e)
{
    tabControl1.SelectedIndex=2;
}

private void menuItem19_Click(object sender, System.EventArgs e)
{
    tabControl1.SelectedIndex=5;
}

private void menuItem5_Click(object sender, System.EventArgs e)
{
    saveFileDialog1.ShowDialog();
    // Insert code to save the file.
}

private void menuItem6_Click(object sender, System.EventArgs e)
{
    saveFileDialog1.ShowDialog();
    // Insert code to save the file.
}

private void menuItem14_Click(object sender, System.EventArgs e)
{
    tabControl1.SelectedIndex=2;

}

private void menuItem18_Click(object sender, System.EventArgs e)
```

```
{
    tabControl1.SelectedIndex=5;

}

private void menuItem16_Click(object sender, System.EventArgs e)
{
    tabControl1.SelectedIndex=6;

}

private void menuItem10_Click(object sender, System.EventArgs e)
{
    Close();
}


// Changing conductors and new conductor parameters
private void TypeBox_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if(Save.Visible==false)
    {//Clear Conductor List Box
        suggest.Text="Select a conductor in the conductor box below";
        this.CLB.Items.Clear();
        CValue.Text = "";
        Area.Text = "";
        Coefficient.Text = "";
        IMod.Text = "";
        FMod.Text = "";
        Mass.Text = "";
        UTS.Text = "";
```

```
Diameter.Text = "";
Creep.Text = "";
EDT.Text = "";
picture.Image = Image.FromFile("intro image.jpg");

// Open the Conductor Xml file and reader cr
XmlTextReader cr = new XmlTextReader(@"Conductors.xml");
cr.WhitespaceHandling = WhitespaceHandling.None;
Conductor cond = new Conductor();
cond.Type(CTLB.SelectedItem.ToString());//

if (cr.Read())
{
    // Moves the reader to the 1st node.
    cr.MoveToContent();

    // Moves the reader to the next node.
    cr.MoveToContent();
    //string selected = CTLB.SelectedItem;

    while(cr.Read())
    {

        // Load conductors
        if(cr.Value== cond.Type())// loop to the type specified until the end of the file
        {
            cr.Read();cr.Read();cr.Read();// move to the conductor names
            this.CLB.Items.Add(cr.Value);
        }
    }
```

```
                    cr.Close();
                }
            }
            else
            {
                if(CTLB.SelectedIndex==0)
                {
                    ntype.Visible=ncreep.Visible=Clabel.Visible=true;
                    CTLB.Visible=false;
                    //Make Readonly false
                    CValue.ReadOnly = CValue.ReadOnly = Area.ReadOnly = Coefficient.ReadOnly = IMod.ReadOnly =
FMod.ReadOnly = Mass.ReadOnly = UTS.ReadOnly = Diameter.ReadOnly = Creep.ReadOnly = false;

                }
                if(CTLB.SelectedIndex!=-1)
                {
                    ntype.Visible=ncreep.Visible=Clabel.Visible=true;
                    CTLB.Visible=false;
                    ntype.Text = CTLB.Text;
                    //Make Readonly false
                    CValue.ReadOnly = CValue.ReadOnly = Area.ReadOnly = Coefficient.ReadOnly = IMod.ReadOnly =
FMod.ReadOnly = Mass.ReadOnly = UTS.ReadOnly = Diameter.ReadOnly = Creep.ReadOnly = false;

                }
            }
        }

        private void CLB_SelectedIndexChanged(object sender, System.EventArgs e)
        {
            try
            {
```

```
//Clear Conductor data
emsg.Text="";
suggest.Text="Select a tab below from Graph, Sag and Tension, Stringing, Clearance, Limits, Help or About";
CValue.Text = "";
CValue.Text = "";
Area.Text = "";
Coefficient.Text = "";
IMod.Text = "";
FMod.Text = "";
Mass.Text = "";
UTS.Text = "";
Diameter.Text = "";
Creep.Text = "";
EDT.Text = "";
picture.Image = Image.FromFile("intro image.jpg");

// Open the Conductor Xml file and reader cr
XmlTextReader cr = new XmlTextReader(@"Conductors.xml");
cr.WhitespaceHandling = WhitespaceHandling.None;
Conductor cond = new Conductor();
cond.Name(CLB.SelectedItem.ToString());//Load the name

cr.MoveToContent(); // Moves the reader to the 1st node.
cr.MoveToContent();// Moves the reader to the next node.
cr.Read();cr.Read();cr.Read();cr.Read();cr.Read();// move to the 1st c value
// Load data
while(cr.Value!= cond.Name())// Find the conductor
{
     cr.Read();
}
```

```
// Load data
cr.Read();cr.Read();cr.Read();
CValue.Text =  cr.Value;
cr.Read();cr.Read();cr.Read();
Area.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
Coefficient.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
IMod.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
FMod.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
Mass.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
UTS.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
Diameter.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
Creep.Text = cr.Value;
cr.Read();cr.Read();cr.Read();
if(cr.Value=="")
      picture.Image = Image.FromFile("intro image.jpg");
else
      picture.Image = Image.FromFile(cr.Value);

// update sat chart and EDT
Wc1.Text=CValue.Text;
cr.Close();
tabupdate(sender,e);//UPDATE TABLE IN FOCUS
graphupdate();
```

```
                // enable save buttons
                toolBarButton2.Enabled=true;
                menuItem3.Enabled = true;
            }
            catch
            {
                emsg.Text="Try again";
            }
        }

        private void CValue_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void Mass_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void pictureBox1_Click(object sender, System.EventArgs e)
```

```
        {

            if(Save.Visible==true)
                if(openFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    System.IO.StreamReader sr = new
                        System.IO.StreamReader(openFileDialog1.FileName);
                    picture.Image =Image.FromFile(openFileDialog1.FileName);
                    image.Text = openFileDialog1.FileName;
                    image.Visible = true;
                    sr.Close();
                }
        }

        private void create_Click(object sender, System.EventArgs e)
        {
            // Add a new type to conductor type list box CTLB
            if(Area.ReadOnly==true)
                CTLB.Items.Insert(0,"New Type");
            //Clear Conductor data
            CValue.Text = CValue.Text = Area.Text = Coefficient.Text = IMod.Text = FMod.Text = Mass.Text = UTS.Text =
Diameter.Text = Creep.Text = EDT.Text = "";
            picture.Image = Image.FromFile("intro image.jpg");
            ncond.Visible = true;
            CLB.Visible = false;
            //Make Readonly false
            CValue.ReadOnly = CValue.ReadOnly = Area.ReadOnly = Coefficient.ReadOnly = IMod.ReadOnly = FMod.ReadOnly =
Mass.ReadOnly = UTS.ReadOnly = Diameter.ReadOnly = Creep.ReadOnly = EDT.ReadOnly = false;

        }
```

```
        private void Cancel_Click(object sender, System.EventArgs e)
        {
            //enable toolbar and disable save
            Save.Enabled = false;
            toolBar1.Enabled = menuItem29.Enabled= true;
            // Remove edit windows and replace boxs
            CTLB.Items.Remove("New Type");ncond.Visible = ntype.Visible = ncreep.Visible = Clabel.Visible = false;CTLB.Visible =
CLB.Visible = true;

            //Make Readonly false
            CValue.ReadOnly = CValue.ReadOnly = Area.ReadOnly = Coefficient.ReadOnly = IMod.ReadOnly = FMod.ReadOnly =
Mass.ReadOnly = UTS.ReadOnly = Diameter.ReadOnly = Creep.ReadOnly = EDT.ReadOnly = true;
            picture.Image = Image.FromFile("intro image.jpg");
            // change status of visibilty of buttons
            Save.Visible = Cancel.Visible = image.Visible=false;
            //Clear Conductor List Box, data and image
            this.CLB.Items.Clear();  CValue.Text = Area.Text = Coefficient.Text = IMod.Text = FMod.Text = Mass.Text = UTS.Text =
Diameter.Text = Creep.Text = EDT.Text = ncond.Text= CValue.Text= CValue.Text=ntype.Text=ncreep.Text="";
            picture.Image = Image.FromFile("intro image.jpg");
        }

        private void Save_Click(object sender, System.EventArgs e)
        {
            //enable toolbar and disable save
            Save.Enabled = false;
            toolBar1.Enabled = menuItem29.Enabled= true;
            //Open file
            XmlDocument myDoc = new XmlDocument();
            myDoc.Load("Conductors.xml");
            XmlNode root = myDoc.DocumentElement;
            //Create a new conductor
```

```
XmlElement conductor = myDoc.CreateElement("Conductor");

// Add new nodes
if(CTLB.Visible == true)
      ntype.Text = CTLB.SelectedItem.ToString();
XmlElement type = myDoc.CreateElement("Type");type.InnerText = ntype.Text;
XmlElement cond = myDoc.CreateElement("Name");cond.InnerText= ncond.Text;
XmlElement c = myDoc.CreateElement("C");c.InnerText=CValue.Text;
XmlElement area = myDoc.CreateElement("Area");area.InnerText=Area.Text;
XmlElement coeff = myDoc.CreateElement("Coefficient");coeff.InnerText=Coefficient.Text;
XmlElement imod = myDoc.CreateElement("IMod");imod.InnerText=IMod.Text;
XmlElement fmod = myDoc.CreateElement("FMod");fmod.InnerText=FMod.Text;
XmlElement mass = myDoc.CreateElement("Mass");mass.InnerText=Mass.Text;
XmlElement uts = myDoc.CreateElement("UTS");uts.InnerText=UTS.Text;
XmlElement diam = myDoc.CreateElement("Diameter");diam.InnerText=Diameter.Text;
XmlElement creep = myDoc.CreateElement("Creep");creep.InnerText=ncreep.Text;
XmlElement pict = myDoc.CreateElement("Image");pict.InnerText=image.Text;

//Add the node to the document.
conductor.PrependChild(pict);
conductor.PrependChild(creep);
conductor.PrependChild(diam);
conductor.PrependChild(uts);
conductor.PrependChild(mass);
conductor.PrependChild(fmod);
conductor.PrependChild(imod);
conductor.PrependChild(coeff);
conductor.PrependChild(area);
conductor.PrependChild(c);
conductor.PrependChild(cond);
conductor.PrependChild(type);
```

```
            root.InsertAfter(conductor,root.LastChild);
            myDoc.Save("Conductors.xml");

            // Remove edit windows and replace boxs
            CTLB.Items.Remove("New Type");ncond.Visible = ntype.Visible = ncreep.Visible = Clabel.Visible = false;CTLB.Visible =
CLB.Visible = true;

            //Make Readonly false
            CValue.ReadOnly = CValue.ReadOnly = Area.ReadOnly = Coefficient.ReadOnly = IMod.ReadOnly = FMod.ReadOnly =
Mass.ReadOnly = UTS.ReadOnly = Diameter.ReadOnly = Creep.ReadOnly = EDT.ReadOnly = true;
            picture.Image = Image.FromFile("intro image.jpg");
            // change status of visibilty of buttons
            Save.Visible = Cancel.Visible = image.Visible=false;
            //Clear Conductor List Box, data and image
            this.CLB.Items.Clear();  CValue.Text = Area.Text = Coefficient.Text = IMod.Text = FMod.Text = Mass.Text = UTS.Text =
Diameter.Text = Creep.Text = EDT.Text = ncond.Text= CValue.Text= CValue.Text=ntype.Text=ncreep.Text="";
            picture.Image = Image.FromFile("intro image.jpg");

            // Open the Conductor Xml file and reader the conductor types in the type box
            XmlTextReader cr = new XmlTextReader(@"Conductors.xml");//Open a reader
            cr.WhitespaceHandling = WhitespaceHandling.None;// White space handling

            if (cr.Read())
            {
                // Moves the reader to the 1st node.
                cr.MoveToContent();

                // Moves the reader to the next node.
                cr.MoveToContent();
                //string to type
```

```
                while(cr.Read())
                {

                    // Load types
                    if(cr.Name == "Type")// loop to the types specified until the end of the file
                    {
                        cr.Read();;// move to the type names
                        while(CTLB.FindString(cr.Value)==-1 && cr.Value!="")
                            this.CTLB.Items.Add(cr.Value);
                    }
                }

                cr.Close();
            }


        }

        private void ntype_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void ncreep_TextChanged(object sender, System.EventArgs e)
        {
```

```
    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void ncond_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void UTS_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void Area_TextChanged(object sender, System.EventArgs e)
        {
```

```
    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void Diameter_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void Coefficient_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void IMod_TextChanged(object sender, System.EventArgs e)
        {
```

```
    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }

        private void FMod_TextChanged(object sender, System.EventArgs e)
        {

    if(ntype.Text!=""&&ncond.Text!=""&&ncreep.Text!=""&&UTS.Text!=""&&CValue.Text!=""&&Area.Text!=""&&Diameter.Text!
=""&&Mass.Text!=""&&Coefficient.Text!=""&&IMod.Text!=""&&FMod.Text!="")
                Save.Enabled = true;
            else
                Save.Enabled = false;
        }


        // Changing limit parameters
        private void tmin_TextChanged(object sender, System.EventArgs e)
        {
            try
            {
                LimError.Text="";
                Convert.ToDouble(tmin.Text);
            }
            catch
            {LimError.Text="Check minimum temperature";}
        }
```

```
private void tmax_TextChanged(object sender, System.EventArgs e)
{
    try
    {
        LimError.Text="";
        Convert.ToDouble(tmax.Text);
    }
    catch
    {LimError.Text="Check maximum temperature";}

}

private void tinc_TextChanged(object sender, System.EventArgs e)
{
    try
    {
        LimError.Text="";
        if(Convert.ToDouble(tinc.Text)<0)
        {
            LimError.Text="Temperature increament must be a positive number";
            tinc.Text="";
        }
    }
    catch{   LimError.Text="Temperature increament must be a positive number";}
}

private void lmin_TextChanged(object sender, System.EventArgs e)
{
    try
    {
        LimError.Text="";
```

```
            Convert.ToDouble(lmin.Text);
        }
        catch
        {LimError.Text="Check minimum length";}
}

private void lmax_TextChanged(object sender, System.EventArgs e)
{
        try
        {
            LimError.Text="";
            Convert.ToDouble(lmax.Text);
        }
        catch
        {LimError.Text="Check maximum length";}
}

private void linc_TextChanged(object sender, System.EventArgs e)
{
        try
        {
            LimError.Text="";
            if(Convert.ToDouble(linc.Text)<0)
            {
                linc.Text = "";
                LimError.Text="Length increament must be a positive number";
            }
        }
        catch{   LimError.Text="Length increament must be a positive number";}

}
```

```
private void CT1_SelectedIndexChanged(object sender, System.EventArgs e)
{
    label30.Text =  CT1.Text;
}
private void Wc1_TextChanged(object sender, System.EventArgs e)
{
    if(Wc1.Text!="")
        CValue.Text=Wc1.Text;
}


// Changing string chart parameters
private void project_TextChanged(object sender, System.EventArgs e)
{
    dataTable2.Rows[dataTable2.Rows.Count-1]["Poles"]= project.Text;
}

private void section_TextChanged(object sender, System.EventArgs e)
{
    try
    {
        Error.Text = "";
        Convert.ToDouble(section.Text);
        dataTable2.Rows[dataTable2.Rows.Count-1]["Sag at"]= section.Text;
    }
    catch
    {
        Error.Text = "The Section must be a number only. No text.";
    }
}
```

```
        private void polep_TextChanged(object sender, System.EventArgs e)
        {
            int Poles= Convert.ToInt16(poles.Text);
            int Polef= Convert.ToInt16(polef.Text);
            int count = 0;
            if(Polef>Poles)
            {
                while (count<(Polef-Poles))
                {
                    dataTable2.Rows[dataTable2.Rows.Count-4-(Polef-Poles)+count]["Poles"] = polep.Text+(Poles + count)+"-"+(Poles
+ count +1);
                    count++;
                }
            }
            else
            {
                while (count<(Poles-Polef))
                {
                    dataTable2.Rows[dataTable2.Rows.Count-4-(Poles-Polef)+count]["Poles"] = (polep.Text+(Poles - count)+"-"+(Poles
- count-1));
                    count++;
                }
            }

        }

        private void poles_TextChanged(object sender, System.EventArgs e)
        {
            stringupdate();
        }
```

```
        private void polef_TextChanged(object sender, System.EventArgs e)
        {
            stringupdate();
        }

        private void calculate_Click(object sender, System.EventArgs e)
        {
            if(polep.Text!=""&&poles.Text!=""&&polef.Text!=""&&section.Text!=""&&project.Text!=""&&UTS.Text!="U.T.S")
                equivspan();
            else
                Error.Text="Check all fields are completed.";
        }

        private void nsection_Click(object sender, System.EventArgs e)
        {
            equivspan();
            dataTable2.Rows.Add(new Object[] {"----------","----------","----------","----------","----------","----------","----------","----------","----------","----------"});
            stringupdate();
            section.Text = Convert.ToString(Convert.ToDouble(section.Text)+1);
            double spole = (Convert.ToDouble(poles.Text));
            poles.Text= Convert.ToString(Convert.ToDouble(polef.Text)+1);
            polef.Text= Convert.ToString(2*(Convert.ToDouble(polef.Text))-spole+1);
        }


        // Changing clearance chart parameter
        private void condtemp_TextChanged(object sender, System.EventArgs e)
```

```
        {
            try
            {
                TWorst.Text="";
                //Check for numbers

    Convert.ToDouble(condtemp.Text);Convert.ToDouble(ESpan.Text);Convert.ToDouble(AttachA.Text);Convert.ToDouble(AttachB.T
ext);Convert.ToDouble(GroundA.Text);Convert.ToDouble(GroundB.Text);Convert.ToDouble(GroundO.Text);Convert.ToDouble(Height
O.Text);Convert.ToDouble(GroundDist.Text);
                clearance();clpicture(); //update picture
            }
            catch
            {
                TWorst.Text="Are all inputs filled in and are entries all numbers?";
            }

        }

        private void helpindex_SelectedIndexChanged(object sender, System.EventArgs e)
        {
        {//Clear Conductor List Box
            this.Helptext.Clear();

            // Open the Help.Xml file and reader cr
            XmlTextReader cr = new XmlTextReader(@"Help.xml");
            cr.WhitespaceHandling = WhitespaceHandling.All;

            while(cr.Read())
            {
                // Load help text
                if(helpindex.SelectedItem.ToString() == cr.Value)// loop to the type specified until the end of the file
```

```
            {
                while(cr.Value!= "end")
                {
                    Helptext.AppendText(cr.Value);
                    cr.Read();// move to the conductor names
                }

            }
            //cr.Read();
        }

        cr.Close();

}

}

private void pictureBox2_Click(object sender, System.EventArgs e)
{
    graphupdate();
}

private void PrintPreview_Click(object sender, System.EventArgs e)
{
    page = 1;
    line = 0;

    this.printDocument1.PrintPage += new    System.Drawing.Printing.PrintPageEventHandler(this.printDocument1_PrintPage);
    printDocument1.DocumentName = "Rsat";
    printPreviewDialog1.Document = printDocument1;
    if(tabControl1.SelectedIndex==2||tabControl1.SelectedIndex==3)
```

```
            printDocument1.DefaultPageSettings.Landscape = true;
        else
            printDocument1.DefaultPageSettings.Landscape = false;
        printPreviewDialog1.ShowDialog();
        //    printDocument1.Dispose();
    }
    private void printDocument1_PrintPage(object sender,System.Drawing.Printing.PrintPageEventArgs e)
    {
        e.Graphics.Clear(Color.White);
        // Page set-up for all print functions
        e.Graphics.DrawString("                RSAT 5.1                                    " , new Font("Arial", 20,
FontStyle.Underline), Brushes.Blue, 10, 10);
        e.Graphics.DrawString(CLB.Text+" data      " + "Page "+page.ToString()+"        Printed: "+DateTime.Now, new
Font("Arial", 14, FontStyle.Regular), Brushes.Maroon, 10, 40);
        e.Graphics.DrawString("UTS:"+UTS.Text+" N", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon, 10, 70);
        e.Graphics.DrawString("C Value:"+CValue.Text+" ", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon, 160, 70);
        e.Graphics.DrawString("Area:"+Area.Text+" mm^2", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon, 380, 70);
        e.Graphics.DrawString("Diameter:"+Diameter.Text+" mm", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon, 580,
70);
        e.Graphics.DrawString("Mass:"+Mass.Text+" kg/km", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon, 10, 90);
        e.Graphics.DrawString("Temp Coeff:"+Coefficient.Text+"'Cx10^-6 ", new Font("Arial", 10, FontStyle.Regular),
Brushes.Maroon, 160, 90);
        e.Graphics.DrawString("Initial mod:"+IMod.Text+" N/mm^2", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon,
380, 90);
        e.Graphics.DrawString("Final mod:"+FMod.Text+" N/mm^2", new Font("Arial", 10, FontStyle.Regular), Brushes.Maroon,
580, 90);
        e.Graphics.DrawLine(Pens.Maroon,10,110,1200,110);
        e.Graphics.DrawString("Limits                                                    ", new Font("Arial", 14, FontStyle.Regular),
Brushes.BlueViolet, 10, 110);
        e.Graphics.DrawString("Limit 1: Temperature = " + Wt1.Text +", "+ CT1.Text + " " + Wc1.Text+ ", Wind "+Ww1.Text+"
PA, on " + Ini1.Text+"                    ", new Font("Arial", 10, FontStyle.Regular), Brushes.BlueViolet, 10, 130);
```

```
            e.Graphics.DrawString("Limit 2: Temperature = " + Wt2.Text +", "+ CT2.Text + " " + Wc2.Text+ ", Wind "+Ww2.Text+"
PA, on " + Ini2.Text+"                          ", new Font("Arial", 10, FontStyle.Regular), Brushes.BlueViolet, 10, 150);
            e.Graphics.DrawString("Limit 3: Temperature = " + Wt3.Text +", "+ CT3.Text + " " + Wc3.Text+ ", Wind "+Ww3.Text+"
PA, on " + Ini3.Text+"                          ", new Font("Arial", 10, FontStyle.Regular), Brushes.BlueViolet, 10, 170);
            e.Graphics.DrawLine(Pens.BlueViolet,10,190,1200,190);

            //Print graph
            int pline = 0;
            //print page setting for sag and  tension charts

    if(tabControl1.SelectedIndex==0||tabControl1.SelectedIndex==1||tabControl1.SelectedIndex==4||tabControl1.SelectedIndex==5||tabC
ontrol1.SelectedIndex==6)
            {
                e.Graphics.DrawString("Span", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 20, 200);
                e.Graphics.DrawString("Temp", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 140, 200);
                e.Graphics.DrawString("Stringing tension", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 200, 200);
                e.Graphics.DrawString("Stringing sag", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 370, 200);
                e.Graphics.DrawString("Final tension", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 520, 200);
                e.Graphics.DrawString("Final sag", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 650, 200);
                e.Graphics.DrawLine(Pens.Black,10,220,1200,220);

                int max = dataTable1.Rows.Count;
                while (line < e.MarginBounds.Height*page/22&&line<max)
                {
                    e.Graphics.DrawString(dataTable1.Rows[line]["Span (m)"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 20, pline*20+230);
                    e.Graphics.DrawString(dataTable1.Rows[line]["Temperature ('C)"].ToString(), new Font("Arial", 10,
FontStyle.Bold), Brushes.Black, 160, pline*20+230);
                    e.Graphics.DrawString(dataTable1.Rows[line]["Stringing tension (N)"].ToString(), new Font("Arial", 10,
FontStyle.Bold), Brushes.Black, 240, pline*20+230);
```

```
                    e.Graphics.DrawString(dataTable1.Rows[line]["Stringing sag (m)"].ToString(), new Font("Arial", 10,
FontStyle.Bold), Brushes.Black, 390, pline*20+230);
                    e.Graphics.DrawString(dataTable1.Rows[line]["Final tension (N)"].ToString(), new Font("Arial", 10,
FontStyle.Bold), Brushes.Black, 520, pline*20+230);
                    e.Graphics.DrawString(dataTable1.Rows[line]["Final sag (m)"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 680, pline*20+230);
                    line++;pline++;
                }

                if(line < max)
                {
                    e.HasMorePages = true;page++;}
                else
                {    e.HasMorePages = false;page = 1; line =0;}

            }
            if(tabControl1.SelectedIndex==2)
            {
                e.Graphics.DrawString("Poles ", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 10, 200);
                e.Graphics.DrawString("Spans", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 100,200);
                e.Graphics.DrawString("Sag at", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 200, 200);
                e.Graphics.DrawString("5 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 300, 200);
                e.Graphics.DrawString("10 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 400, 200);
                e.Graphics.DrawString("15 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 500, 200);
                e.Graphics.DrawString("20 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 600, 200);
                e.Graphics.DrawString("25 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 700, 200);
                e.Graphics.DrawString("30 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 800, 200);
                e.Graphics.DrawString("35 degs", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 900, 200);
                e.Graphics.DrawLine(Pens.Black,10,220,1200,220);

                int max = dataTable2.Rows.Count;
```

```
                while (line < e.MarginBounds.Height*page/22&&line<max)
                {
                    e.Graphics.DrawString(dataTable2.Rows[line]["Poles"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 10, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["Span lengths"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 100, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["Sag at"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 200, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["5 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 300, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["10 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 400, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["15 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 500, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["20 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 600, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["25 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 700, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["30 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 800, pline*20+220);
                    e.Graphics.DrawString(dataTable2.Rows[line]["35 degrees"].ToString(), new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 900, pline*20+220);

                    line++;pline++;
                }

                if(line < max)
                {
                    e.HasMorePages = true;page++;}
                else
                {   e.HasMorePages = false;page = 1; line =0;}
```

```
            }
            if(tabControl1.SelectedIndex==3)
            {
                e.Graphics.DrawLine(Pens.Maroon,10,200,1200,200);
                // Data line 1
                e.Graphics.DrawString("General", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 10,210);
                e.Graphics.DrawString("Pole A", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 220,210);
                e.Graphics.DrawString("Pole B", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 430,210);
                e.Graphics.DrawString("Object", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 640,210);
                e.Graphics.DrawString("Clearance", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 850,210);
                e.Graphics.DrawLine(Pens.Black,10,230,1200,230);
                // Data line 2
                e.Graphics.DrawString("Span = " +ESpan.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 10, 240);
                e.Graphics.DrawString("Attachement = "+AttachA.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
220,240);
                e.Graphics.DrawString("Attachement = "+AttachB.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
430,240);
                e.Graphics.DrawString("Height = "+HeightO.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
640,240);
                e.Graphics.DrawString("Object = "+nclearance.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
850,240);
                // Data line 3
                e.Graphics.DrawString("Temperature = " +condtemp.Text+" 'C", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
10, 260);
                e.Graphics.DrawString("Ground = "+GroundA.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
220,260);
                e.Graphics.DrawString("Ground = "+GroundB.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
430,260);
                e.Graphics.DrawString("Ground = "+GroundO.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
640,260);
```

```
            e.Graphics.DrawString("Worst = "+Worst.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black, 850,260);
            // Data line 4
            e.Graphics.DrawString("Centre Tension = " +Tension.Text+" N", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
10, 280);
            e.Graphics.DrawString("Tension = "+TensionA.Text+" N", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
220,280);
            e.Graphics.DrawString("Tension = "+TensionB.Text+" N", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
430,280);
            e.Graphics.DrawString("Distance from A = "+GroundDist.Text+" m", new Font("Arial", 10, FontStyle.Bold),
Brushes.Black, 640,280);
            e.Graphics.DrawString("Worst distance = "+DWorst.Text+" m", new Font("Arial", 10, FontStyle.Bold), Brushes.Black,
850,280);
            e.Graphics.DrawLine(Pens.Black,10,300,1200,300);


            double L = Convert.ToDouble((Math.Round(Convert.ToDouble(ESpan.Text))));
            int ymax =  800;
            //y 800(ymax) - 0, depends on pole lengths and ground levels, max pole height = 80(ymax/6) leaving 400 for ground diffs
            int xmax = 1000;
            //x 0 - 1000(xmax),  L = 1400 (xmax*0.8)
            // Graph dimensions
            // constants
            int Ax = Convert.ToInt32(0.1*xmax);// Pole A x = 200 m 10 % xmax
            int Bx = Convert.ToInt32(0.9*xmax);// Pole B x = 200 m 90 % xmax
            // Declare varaibles
            int Ay = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundA.Text))*ymax/32+10);//Pole foot co ord y = 0
%
            int By = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundB.Text))*ymax/32+10);//Pole foot co ordy = 0
%
            int Oy = ymax - Convert.ToInt32(Math.Round(Convert.ToDouble(GroundO.Text))*ymax/32+10);//Object foot
            int Ox = Ax + (Convert.ToInt32(Math.Round(Convert.ToDouble(GroundDist.Text)/L*xmax*0.8)));// Object distance
```

```
int Ah = Convert.ToInt32(Math.Round(Convert.ToDouble(AttachA.Text)))*ymax/32+10;// Pole att height 10 = 60
int Bh = Convert.ToInt32(Math.Round(Convert.ToDouble(AttachB.Text)*ymax/32)+10);// Pole att height 10 = 60
int Oh = Convert.ToInt32(Math.Round(Convert.ToDouble(HeightO.Text)*ymax/32)+10);// Object height 10 = 60
Point PAg = new Point(Ax,Ay);  //Pole A ground
Point PAa = new Point(Ax,Ay-Ah); //Pole A attach
Point POg = new Point(Ox,Oy);  //Object ground
Point POa = new Point(Ox,Oy-Oh); //Object attach
Point PBg = new Point(Bx,By);  //Object ground
Point PBa = new Point(Bx,By-Bh); //Object attach

//Pole A
e.Graphics.DrawLine(System.Drawing.Pens.Brown,PAg,PAa);
//Object
e.Graphics.DrawLine(System.Drawing.Pens.Red,POg,POa);
//Pole B
e.Graphics.DrawLine(System.Drawing.Pens.Brown,PBg,PBa);

// conductor
Point PSh;
int Wy = Oy - Oh;
int Wx = Ax+ xmax/2;
//sag at 2 x worst dist get sag for normal span
double Wc = Convert.ToDouble(Mass.Text)*9.80665/1000;
double T = Convert.ToDouble(Tension.Text);


if (Worst.Text!="Uplift Case"||Worst.Text!="Clearance Violation"||Worst.Text!=""||DWorst.Text!="Uplift
Case"||DWorst.Text!="Clearance Violation"||DWorst.Text!="")
{
    Wx = Ax + (Convert.ToInt32(Math.Round(Convert.ToDouble(DWorst.Text)/L*xmax*0.8)));// Object distance
```

```
            Wy = Oy - Oh - Convert.ToInt32(Math.Round(Convert.ToDouble(Worst.Text)*ymax/32));// Object height 10 = 60

        }
        PSh = new Point(Wx ,Wy);

        Point []pointArray2 = {PAa,PSh,PBa};
        // Draw Conductor
        e.Graphics.DrawCurve(System.Drawing.Pens.Gray,pointArray2);
        // Ground
        Point []pointArray3 = {PAg,POg,PBg};
        e.Graphics.DrawCurve(System.Drawing.Pens.Green, pointArray3);
    }
}

private void Print_Click(object sender, System.EventArgs e)
{
    page = 1;
    line = 0;

    this.printDocument1.PrintPage += new    System.Drawing.Printing.PrintPageEventHandler(this.printDocument1_PrintPage);
    printDocument1.DocumentName = "Rsat";
    if(tabControl1.SelectedIndex==2||tabControl1.SelectedIndex==3)
        printDocument1.DefaultPageSettings.Landscape = true;
    else
        printDocument1.DefaultPageSettings.Landscape = false;
    printDocument1.Print();
    printDocument1.Dispose();
}

private void Creep_TextChanged(object sender, System.EventArgs e)
{
```

```
    if(Creep.Text!="")
    {satupdate();equivspan();}
}

private void menuItem3_Click(object sender, System.EventArgs e)
{

    // Initialize the SaveFileDialog to specify the RTF extension for the file.
    saveFileDialog1.DefaultExt = "*.doc";
    saveFileDialog1.Filter = "Word File|*.doc";


    // Determine if the user selected a file name from the saveFileDialog.
    if(saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&
        saveFileDialog1.FileName.Length > 0)
    {
        ExportToText (saveFileDialog1.FileName, true, 30);//XmlDocument myDoc = new XmlDocument();

    }
}

private void MainView_SizeChanged(object sender, System.EventArgs e)
{
    screensize();
}

private void Defaults_Click(object sender, System.EventArgs e)
{
    DefaultLimits();
}
```

```
        private void button1_Click(object sender, System.EventArgs e)
        {
            ResetStringing();try
            {
                Error.Text="";
                int Poles = Convert.ToInt16(poles.Text);
                int Polef = Convert.ToInt16(polef.Text);
                dataTable2.Rows.Clear();
                if(dataTable2.Rows.Count>0)
                {
                    while(dataTable2.Rows[dataTable2.Rows.Count-1]["5 degrees"].ToString()!="----------------------------------------------
--")
                    {dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);}
                    dataTable2.Rows.RemoveAt(dataTable2.Rows.Count-1);

                }
                int count = 0;
                if(Polef>Poles)
                {
                    dataTable2.Rows.Add(new Object[] {"----------------------------------------------","----------------------------------------------
-----","----------------------------------------------","----------------------------------------------","----------------------------------------------","----
----------------------------------------","----------------------------------------------","----------------------------------------------","----------------
--------------------------------","----------------------------------------------"});
                    while (count<(Polef-Poles))
                    {
                        dataTable2.Rows.Add(new Object[] {polep.Text+(count + Poles)+"-"+(count + Poles+1), "Input span", "", "", "",
"","","","","","});
                        count++;
                    }
                }
                else
```

```
                {
                    dataTable2.Rows.Add(new Object[] {"-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------","-----------------------------------------"});
                    while (count<(Poles-Polef))
                    {
                        dataTable2.Rows.Add(new Object[] {polep.Text+(Poles - count)+"-"+(Poles - count - 1), "Input span", "", "", "", "","","","
```