

SToMP

Developer's

Guide

Version 2.2

The SToMP Consortium
December 1997

© 1997 The SToMP Consortium

About this document

This document has been created by members of the SToMP consortium as a result of extensive discussions and criticisms of proposed styles, frequently exemplified by documents and applications. Everyone involved in the project has contributed to this debate, and no-one has had their work spared from the criticism of their peers. This document, originally produced in 1993 as an encapsulation of the results of these deliberations, ensured a degree of homogeneity within the first versions produced. It has been continually updated since then, so that SToMP materials can continue to be produced in a homogenous style, even though that style develops and changes over the years.

Version 2 contains a major re-write of chapter 5 to accommodate the new features of the testing tool.

New SToMP Material

The SToMP project welcomes the generation of new materials to the specifications laid down in this document. You are invited to discuss proposed developments with the SToMP team to ensure coordination with work being carried out elsewhere. Initial enquiries should be made to the Project Director,

Dr. R A Bacon,

Department of Physics,
University of Surrey,
Guildford, GU2 5XH
UK

phone: +1483 259414

fax: +1483 259501

email: stomp@surrey.ac.uk

CONTENTS

| | |
|---|----|
| 1 Introduction | 1 |
| 1.1 The SToMP Consortium (1992 - 1995)..... | 1 |
| 1.2 SToMP developers (1995 - 1996)..... | 1 |
| 1.3 SToMP Philosophy | 1 |
| 1.4 SToMP Multimedia Documents | 2 |
| 1.5 Compatibility | 2 |
| 2 Script authoring | 3 |
| 2.1. Preparing a script -- housekeeping issues | 3 |
| 2.1.1 Copyright | 3 |
| 2.1.2 Template | 3 |
| 2.1.3 Footnotes..... | 4 |
| 2.1.4 Interfacing with activities..... | 4 |
| 2.1.5 Filenames | 5 |
| 2.1.6 Navigation..... | 6 |
| 2.2. Writing a script -- pedagogics..... | 6 |
| 2.2.1 Preamble and prerequisites | 6 |
| 2.2.2 Summary | 7 |
| 2.2.3 General author style | 7 |
| 2.2.4 Assessment..... | 7 |
| 2.2.5 Activities | 8 |
| 2.2.6 Equations..... | 9 |
| 2.2.7 How to emphasise | 9 |
| 2.2.8 Indentation | 9 |
| 2.2.9 Biographies | 9 |
| 2.2.10 Data tables..... | 10 |
| 2.3. Writing a script - technical..... | 10 |
| 2.3.1 Buttons links | 10 |
| 2.3.2 Associated documents (zero selection links) | 11 |
| 2.3.3 Variables | 12 |
| 2.3.4 Equation styles | 12 |
| 2.3.5 Colour | 12 |
| 2.3.6 Figures..... | 12 |
| 2.3.7 Tables | 13 |
| 2.3.8 Hidden files..... | 13 |
| 2.3.9 Activity coding scheme..... | 13 |
| 2.4 Authoring techniques | 14 |
| Note on numbering in Word2 | 14 |
| 3 Styles of Programmed Activities, | 15 |
| 3.1 General observations..... | 15 |
| 3.2 Associated documents and links. | 15 |
| 3.3 Help..... | 17 |
| 3.4 Author and user modes. | 17 |
| 3.5 Controls..... | 17 |
| 3.6 Menu bar | 18 |
| 3.7 The Window | 20 |
| 3.8 The 'about' box. | 21 |
| 3.9 Dialogue boxes | 21 |

| | |
|---|----|
| 3.10 Text | 21 |
| 3.11 Pictures | 21 |
| 3.12 Colours | 21 |
| 3.12.1 Windows colours | 21 |
| 3.12.2 SToMP colours for programmed activities. | 22 |
| 3.12.3 Colour example 1. | 23 |
| 3.12.4 Colour example 2 | 23 |
| 3.12.5 Colour example 3 | 25 |
| 3.12.6 Colour example 4 | 26 |
| 3.13 DLL's, components and other support features | 26 |
| 3.13.1. Toolbar DLL..... | 26 |
| 3.13.2. Aboutbox DLL | 26 |
| 3.13.3. Colour DLL | 26 |
| 3.13.4. mcs file format..... | 27 |
| 4 Editing guidelines | 29 |
| 4.1 Introduction | 29 |
| 4.2 Guidelines | 29 |
| 4.3 Addressing the student | 30 |
| 4.4 Example..... | 30 |
| 5 Testing style guide | 31 |
| 5.0 Re-implementation of the testing tool..... | 31 |
| 5.1 Changes | 31 |
| Changes between version 4 and version 4.2..... | 31 |
| Changes between version 3 and version 4..... | 31 |
| 5.2 Student assessment in SToMP. | 32 |
| 5.3 In line testing..... | 32 |
| 5.4 The testing package..... | 32 |
| 5.5 Question Files..... | 34 |
| 5.5.1 Multiple choice or rank ordering questions..... | 34 |
| 5.5.2 Pair matching questions | 35 |
| 5.5.3 Textual and numeric questions | 35 |
| 5.5.4 Random numeric questions | 35 |
| 5.6 The Associated Data file | 36 |
| 5.7 Help (physics)..... | 39 |
| 5.8 Handling responses | 40 |
| 5.9 Working in the Answer Sheet. | 40 |
| 5.10 Using the testing tool within Microcosm..... | 41 |
| 5.11 Filenames | 42 |
| 6 Creating a unit..... | 43 |
| 6.1 New schemes for hiding documents..... | 43 |
| 6.2 New graph activity type..... | 43 |
| 6.3 Organising your unit material..... | 43 |
| 6.3.1 Pictures, diagrams, videos and applications | 43 |
| 6.4 A check list for scripts before integration | 43 |
| 6.5 Before integrating your material you must:..... | 43 |
| 6.6 Integration of your material into a unit..... | 43 |
| 6.7 Note:..... | 46 |
| 7. stscript.dot macros | 47 |
| 7.1 Recommended usage..... | 47 |

| | |
|--------------------------------|----|
| 7.2 st1format | 48 |
| 7.3 st2comment | 49 |
| 7.4 st3activity | 49 |
| 7.5 st4symbol | 49 |
| 7.6 stCleanActs | 49 |
| 7.7 stCleanEquns | 49 |
| 7.8 stCleanNormal | 49 |
| 7.9 stDelHidden | 49 |
| 7.10 stRunAll | 49 |
| 7.11 stSplitFile | 49 |
| 7.12 General points | 49 |
| Appendix A | 51 |
| Virtual links: | 51 |
| Appendix B | 51 |
| Hidden documents: | 51 |
| Introduction | 51 |
| Microcosm Implementation | 51 |
| Suggested Hierarchy | 52 |

1 Introduction

The SToMP project (Software Teaching of Modular Physics) is a TLTP project (Teaching and Learning Technology Programme) funded from August 1992 until July 1995 by the Higher Education Funding Councils. Nine university Physics Departments have been involved in developing the material, with additional assistance being provided by the National Physical Laboratory, by the IBM UK Scientific Centre and by Queensland University of Technology, Australia, Keele University and the University of Guelph, Canada.

The project has produced materials for the teaching of Waves and Vibrations, and the Treatment of Measurement Uncertainty.

1.1 The SToMP Consortium (1992 - 1995)

University of Surrey. - Drs Dick Bacon, David Lancefield, Tony Cartwright, Terry Hinton, Messrs Justin Watkins, Dan Blanchard, Richard Moran, Zhouxon Li, Erwin Flick and Mrs Sheila Dickens.

University of Cambridge.- Drs Robert Harding, Atta Chui, Mr Mike Gilbert

University of Wales, College of Cardiff.. - Drs. Susan Burke, Stephen Webb, Phil Richards

University of St Andrews. - Drs Roger Edwin, Bruce Sinclair, Mr Jonathan Armitage, Drs Alistair Gillies, Wendy Webster.

University of Salford. - Drs Graham Keeler, Brian James, Graham Duffy, Ms Pauline Green

University of Manchester. - Drs Ross Barnett, Andrew Tyler, Messrs James Youngman, Ian Porter and Robert Pitt.

Queen's University, Belfast. - Dr Martin Lamb, Mr Tony McElhinney

University of Portsmouth. - Messrs Colin White, Andrews Stosiek, Ms Stella Herridge

The Open University - Drs Stephen Swithenby, Robert Hasson.

University of Keele - Dr Gayle Calverly.

Together with - Dr Stephen Rake (IBM & Southampton University), Mr Oliver Dewdney (Southampton University), Mr John Davies (Queensland University of Technology), Dr. Bruno Lemke (Nelson Polytechnic), Prof. Jim Hunt (University of Guelph, Canada)

1.2 SToMP developers (1995 - 1996)

Queensland University of Technology, Messrs - John Davies, Mike Jackson.

The Open University - Drs Stephen Swithenby

University of Guelph - Canada. Prof Jim Hunt.

University of Portsmouth. - Mr Colin White

University of St Andrews. - Drs Roger Edwin, Bruce Sinclair

1.3 SToMP Philosophy

The philosophy of the SToMP project is that the user (the student) should be provided with a complete and open study environment that is suitably hyper-linked so that minimal outside help or resourcing should be required for the study of a complete course. As far as possible the material should be self motivating, self explanatory and self contained. The user interface should be simple to use, entirely self consistent across the package and based upon *de facto* standards.

1.4 SToMP Multimedia Documents

SToMP documents are of physical types text (.rtf and .txt) picture (.enc, .bmp and .jpg), audio (.wav), video (.avi) and executable (.exe/.mcs). Pedagogically these are structured as main, extension or foundation scripts, pictures, audio, video, activities, tests and tools. The tests include both formative (self tests) and summative (assessed tests). Logically the documents are structured as a linear learning path consisting of scripts (equivalent to lecture notes) and activities, and reference material containing textbooks (electronic versions of relevant chapters), data books (containing physical data) and other items. The linking mechanism provides preferred routes through the material as well as ready access to relevant reference and other ancillary items.

1.5 Compatibility.

As new version of windows come into wider use (win95, NT win98?) we have additional issues involving compatibility and maintenance.

Until we abandon support for Win31 (not yet planned!) it is necessary that we **only use DOS compatible directory names and file names**. This means: names must be 8 characters or less, must contain no non-alphanumeric characters other than underscore, and extensions may only be three characters.

There are also some problems with the equations, in that to maintain compatibility with the existing equations we have to use an equation editor that appears to be only compatible with Word 2. This introduces editorial problems with documents authored in word 6, because equations have to be re-entered.

2 Script authoring¹

2.1. Preparing a script -- housekeeping issues

2.1.1 Copyright

Each author is responsible for identifying the need for copyright clearance on documents being used and if possible for establishing the holder of the copyright. A centrally prepared copyright request form is available.. Fill this in and send it to Surrey who will organise the clearance. It is quite appropriate to contact the copyright holder informally but this must be followed up by a formal request. We will be seeking world English language rights for use within the SToMP context.

2.1.2 Template

Scripts should be prepared as storyboards using the *story.dot* Word2 template or the *story6.doc* Word6 template, including comments and activity descriptions. The storyboards should then be used as master documents for a unit. Any modifications should be made to these storyboards. The RTF documents will be generated from these storyboards using the *stscript.dot* macros.

These macros will strip out the activities (leaving an identification number as hidden text) as well as the comments and underlines that you should use to identify buttons. Consult chapter 6 for details of how to use the stscript template.

Use the appropriate paragraph styles in your storyboards as listed below, rather than manufacturing your own layout devices.

- **Heading 1** -- three lines at the top of each main and all additional scripts, the first line giving the application name e.g. **Waves and Vibrations** the second giving unit number e.g. **Unit 1.7**, and the third giving the title e.g. **Non-linear oscillations**. The second line should be followed by a footnote specifying the copyright owner and the author. All support documents in a unit (except Equations and Symbols) are identified in sequence by adding (a), (b) etc after the unit number, e.g. Unit1.7(a) for the first subsidiary document, followed by the the copyright footnote
- **Heading 2** -- sections within the unit are to be numbered, e.g. **1 Introduction** and **2 Using a ruler**. etc. Note that the unit number and application letter does not appear. Use just two spaces between the number and the title, and do not end with a full stop. This style produces dark red text.
- **Heading 3** -- sub-sections within unit sections - number these sub-sections as **2.1 Using wooden rulers**, **2.2 Using metal rulers**. Again, no unit number or letter, and just two spaces between number and description. No terminating full stop.
- **comment** - use this style to advise other consortium members on anything you like, e.g. aide-memoirs, to develop the structure, as script ideas, information about links etc.
- **activity** - this style (which puts text in a box on the right) is used to specify the activity and to briefly describe it.

¹ v2.2, July 1994. Martin Lamb, Steve Swithenby, Dick Bacon.

Script authoring

- **prereqs** - use this style for the bulleted prerequisites paragraph.
- **greyback** - use this style for the grey-background paragraphs in the preamble and in paragraphs containing the questions icon (in line questions and the self and assessed tests).
- **Normal indent** - should be used where a sentence requires special emphasis. It can also be italicised..
- **Normal** - should be used for all other text.

These styles will be modified by the stscript macros so that all the final scripts display the same styles. There is no need to modify any of the styles in the storyboards.


All text should be black, other than the heading 2 style described above which is dark red. Text to be used as a button link should be underlined in the storyboard but left in black. This text becomes coloured (blue by default) when the link is generated within Microcosm.

Keep the information on the front page of your storyboards up to date (dates, version number etc.) but do not alter the layout since the stscript.dot macros use this layout to recognise and delete this title page.

Each activity must carry an identifier in a storyboard. The codes to be used for these are defined at the end of this document in the appendix. The activity identifier (e.g. WV.2.3.4g) must be the first item to appear in the activity box in a storyboard and it must be followed by a paragraph break (i.e. it must be the only item on the first line). A concise summary of the activity should be given in the second paragraph. Only one activity should be put into each box and adjacent activities should have separate boxes, i.e. insert blank lines in the normal style between the activities. All blank lines (i.e. empty paragraphs) will be removed by the stscript macros.

2.1.3 Footnotes

When displayed by the rtf viewer a footnote identifier will appear as an information icon in the main text. The footnote text will then pop-up in a box when the information icon is double-clicked. It is not possible to link out from footnote text (since a single click dismisses the pop-up). Thus, footnotes should contain nothing that is likely to require further explanation (e.g. via generic links). For example only the simplest of in-line-question answers should be in footnote form.

Each footnote reference should have a space before it and a space after it (to allow room for the information icon ). This icon appears circular in the rtf viewer.

Every script must have a footnote at the end of the second line of the main title that contains a copyright notice containing the following information:

© 1994 University of wherever
Written by A.N.Other.

2.1.4 Interfacing with activities

Each activity will have a button associated with it. There are two styles of activity button, but in the storyboard the actual text that will form the button in either style should be underlined.

1) When the button destination is a sufficiently important part of the development of the script that the link should always be followed, then the button should be put into a separate paragraph with an appropriate icon at the start of the paragraph. The text should describe what

the user should be doing having followed the link, but the link destination type does not have to be explicitly stated. e.g.



Look at this model of a plucked string and see how the spectral content changes with time.

Or, if the preceding paragraph has made clear what is expected of the user, a simple description can be used, e.g.



Critically damped motion.

2) When the button destination is not part of the main argument, but provides additional information or insight that the user might want to pursue then the button should be in the body of a paragraph with the surrounding text describing its purpose. e.g. "if you cannot remember how to use complex numbers, look at this revision material"

The button is indicated in the storyboard by underlining. This underlining will not appear in the final scripts, but the storyboard version will be used as the definitive guide to the position of buttons when MCM links are being set up.

In a storyboard the destination of a button link should always be specified uniquely. This should normally be obvious since the activity box will be near the underlined text. If there is any ambiguity, then put the destination as hidden text next to the underlined button text (select text, choose **format text** and check the **hidden** box under **style**).

In many cases it will be appropriate for there to be an introductory script to describe how to use an activity. This script could contain both the background theory and instructions about how to use the activity. This introduction can be made to display automatically when the activity is started.(see section 3.2)

2.1.5 Filenames

The following scheme for the filenames of documents aids the organisation and recognition of the large number of files involved. Please keep to this convention. Until there is no possibility of any user wanting to run SToMP under Win3.1, please retain the conventional DOS filename conventions. Also, please refrain from including any non-ISO filename characters (like &, -, \$, etc) that have to be changed before they can be written to CD.

Filenames of scripts and of their support documents should use the following coding scheme based upon the unit numbers themselves. Thus the script for waves unit 1.6 will be w1_6.rtf. Subsidiary scripts for this unit would be in files w1_6a.rtf, w1_6b.rtf, etc. The a,b,c.. are to be in the order that a student following all buttons would see them. The script for waves unit 1.6a will be in w1_6_a.rtf, with subsidiaries w1_6_aa.rtf etc. These same identifiers (with stops instead of underlines) are to be used as document descriptors in the docuvers and in the second line of the heading of each document. In these last two cases, however, the lower case letter should be put in braces. i.e. Unit W1.6(a) would be the second document seen in Unit W1.6, and Unit W1.6A(c) would be the fourth seen in Unit W1.6a.

If you run out of characters then drop an underline separator (e.g. op1_6aa.rtf), but endeavour to keep the coding unambiguous.

Filenames for unit equation and symbol documents are obtained by preceding the main script filename with 'e' or 's'. Thus the equation document for a unit w2.1 is called ew2_1.rtf and the

symbol document is called sw2_1.rtf. For Optics unit 2.1 the equation file is eop2_1.rtf and the symbols file is sop2_1.rtf.

2.1.6 Navigation

At the end of every script, after the summary (see below), and the symbols and equations buttons, there will be three additional buttons giving navigational aid i.e.:

Symbols

Equations

Contents

Back

Next

Other documents will just finish with the three buttons. The Back button will link back to whatever was previous to the current document, e.g. main script, previous unit or contents page. If there is more than one possibility then they should both be implemented, so that they appear in an 'available links' box. Next will be the next logical document, e.g. next unit script, or the main script.

2.2. Writing a script -- pedagogics

2.2.1 Preamble and prerequisites

Each unit should start with two short sections, **prerequisites** and **preamble**, the prerequisites will be a bulleted list of topics followed by the unit numbers (in braces) in which each topic is covered. The bulleted list is indented, as specified in style **prereq**. Prerequisites and preamble headings should be in style **heading 2**. The preamble paragraph is to be on a light grey background - use the style **greyback** e.g.

Prerequisites

- Units (M1.2)
- Handling Numbers (M1.3)
- Random and Systematic Uncertainties (M2.1)
- Histograms (M2.2)

If there are no identifiable units that can be stated here, then use the phrase:

- A general science education to 'A' level or equivalent.

Preamble

The preamble should be no more than one or two short paragraphs incorporating the study guide and containing

- information about the study context (i.e. how the ideas link from earlier units)
- advice on the length and level of difficulty of the material
- advice on any special features that might influence a student's study pattern
- advice on how to cope if short of time.

The preamble should provide immediate context for those jumping into the unit out of sequence and/or provide study guide information.

2.2.2 Summary

At the end of each unit script there will be a bulleted list (where appropriate) of a few lines re-emphasising the key points of the unit. Just listing the contents of the unit is not appropriate here. Authors may find it useful to draft the summary early in the process to ensure the script delivers all the outcomes intended.

2.2.3 General author style

Authors are requested to:

- keep text concise (avoid repetition)
- introduce pictures and sound instead of words where appropriate
- include lots of exciting activities
- avoid the use of "we" unless you really mean "we all . . . ", or possibly "we (the author and the student) conclude from this . . . ". e.g. "In the following sections we look at ..." can be put "In the following sections you will see..."
- avoid using "he" or "she". You can use "the user" or "you" or recast the sentence to avoid the syntax altogether.

2.2.4 Assessment

Assessment can be;

1. to allow the student to check understanding of the main script with immediate answers.
2. for formative assessment (self test) at the end of the unit.
3. for summative assessment (assessed test) at the end of the unit.
4. problems suitable for formative or summative assessment at the end of each unit

All questions and tests will be indicated by the question icon, problems will be in separate documents indicated by the document icon. In-line questions will be followed by the question number (within the unit) and the text of the question on grey background. e.g.



3. What is the ratio of the lengths of two otherwise identical simple pendula that will have their frequencies in the ratio of 2:1? Run [the pendulum simulation](#) if you wish.

An answer for an in-line question may be a link to another document (preferred), a pop-up footnote (only where possible - see 2.1.3 above) or 'in-line', in the unit text (not encouraged).

The answer will be indicated by the answer icon followed in the case of a footnote, by the word "Answer", or in the case of a link by the text "Answer to question *n*" as shown below. The answer icon and text will be on the default background, and not on the grey background of the question e.g.



Answer to question 3

An answer document will have the same type of header as other documents, e.g. for file M2_1e.rtf

| |
|---|
| <p style="text-align: center;">Measurement and Uncertainty Unit 2.1(e) ² Answer to question 3</p> |
|---|

²© 1994 University of St Andrews

Written by R. P. Edwin, A. D. Gillies and W. J. Webster

Solutions to unstable watches scenario

A formative assessment activity will be indicated by the question icon followed by the words "self test"



Self-test for unit 2.3

and a summative assessment activity will be similar, but the icon will be followed by "assessed test"



Assessed test for unit 3.3

Refer to chapter 5 for information on how to author all types of questions, and how to produce questions for the multiple choice tester.

The multiple choice testing tool can be used for both summative and formative assessment. However, you may wish to include assessment that is of essay/project/activity type.

Our aim is to have at least 3 alternative versions of entire question sets, each of which should be sufficient to occupy students for 20% of their study time. For a typical unit this might require 3 or more lots of 5 multiple choice questions for both the formative and the summative tests, plus other relevant essays etc.

Problems will be in a subsidiary document, accessed from the main script just after the self test and assessed tests above.



Problems

The problems will be any suitable physics problems, and they can (but need not) take advantage of the tools and other resources available within the whole package.

The problem file should start with the usual three line heading with copyright footnote. The question numbers should be preceded by the question icon. e.g.



1.

Replacing the question icon with a problem icon is currently under consideration. e.g.



1.

2.2.5 Activities

activity icons currently available are



text



pictures



audio



video



activity



derivations



graph



tests.

They will be used only against the left hand margin and will be followed by a single space.

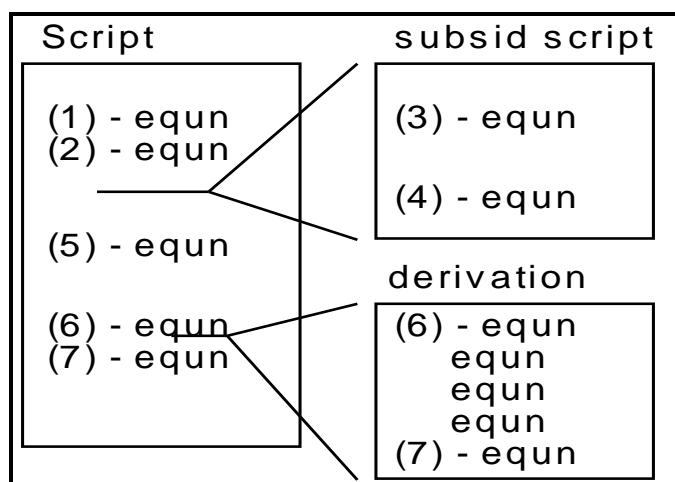
An alternative for the graph style, is to make the equation of the graph itself the button. In this case, the rtf viewer will put a blue (or button colour) border around the equation.

2.2.6 Equations

Equations should be authored in Mathtype and inserted one tab stop from the left whether they are numbered or not. For numbered equations the number should be enclosed in round brackets (5) starting at the left margin followed by a tab and then the equation. DO NOT use automatic equation numbering. Equation numbers should increment sequentially within the unit. If the unit is broken down hierarchically into smaller scripts then the sequential numbering will follow on into the subsidiary documents, as shown in the diagram.

Where a derivation is offered between two expressions (where the script does not go into enough detail), then the equations in the derivation will not need to be numbered, except for the first and the last which will have the same numbers as in the script.

If it is necessary to refer back to an earlier equation within a unit then use the form "Equation(5) above is restricted to one-dimensional motion. More



generally we can. . .". Here the word Equation(5) would be a specific link back to the original Equation 5. From outside the unit the equation would be referred to as Equation(W4.8.5), again using a specific link.

2.2.7 How to emphasise

Italics should be used to emphasise text. The text should be in Normal style and italicised (*not* in Comment style). An important sentence (e.g. a definition) could be italicised and indented (use Normal indent and italicise).

Important equations should be emphasised by a surrounding box. The advice here is to insert the equation in a single cell Word table. Initially the RTF viewer will not display this border, but once the borders of the table display is included then boxed equations will display as required.

2.2.8 Indentation

The use of indented text should be avoided because of the wasted space. Two exceptions are emphasised sentences (see above) and equations (see below).

2.2.9 Biographies

The names of important people will be generic links to biographical information. Steve Swithenby (OU) is collating a document of such biography entries including photographs and can assist in obtaining the required information from the OU. Authors should identify requirements and send a list to Steve and a copy to Erwin Flick at Surrey. Also include the list in *comment style* at the end of the unit script, together with the external link information.

2.2.10 Data tables

Tables should be produced using Word tables with the appropriate numbers of columns. Tables are of fixed size and cannot be 'line wrapped'. It is therefore obvious that tables should be kept as narrow as possible because they will otherwise put constraints on the user's use of screen space. Where possible, split large tables. As far as possible avoid allowing the text to use more than one line in a cell. If necessary, use markers (daggers, asterisks, crosses, etc.) to refer to text beneath the table, rather than increase the table size for one or two text-based cells. Remember that the viewer might take a little more space for the text, and so always check the appearance of tables in the viewer before finalising column widths.

In general, the first one or two rows of the table should identify the quantities tabulated and give the units. The table should be identified (below) by a number and caption. Do not indent tables.

2.3. Writing a script - technical

A good scheme used by many SToMP authors is to prepare all the documents required for a unit in one Word document. Each document is separated by a page or "section break on new page", and the main scripts with derivations, answers, activity introductions, equation document, symbol document and test sheets are all included. Word macros distributed with the SToMP templates can then be used to convert this one file into the separate rtf files required, and write them into the required file names. Details of how to do this are given in chapter 7.

2.3.1 Buttons links

Buttons links should be used (see above) where the author wishes to invite the student to follow a particular link before continuing. The main context for buttons is activities (including animations, simulations, pictures, video, biography etc.) but includes invitations to tributaries, advanced material and diversions. If it is clear that not all users would need to follow a link (e.g. if the button links to a tributary document), then that should be made clear in the surrounding text. e.g. "you should recall from unit W1.2 that vectors are represented by two values, for example magnitude and direction" as opposed to ". . .now watch this model of a rotating vector to see the relation between . . .".

The aim when using a button link is to ensure that double clicking on a button reaches the required destination without having to make any further decisions or selections. This means that the action "select button text and then follow link" (synthesised by double clicking on the button text) must not conflict with any other link. It also means that the "Auto-Follow Single Links" option under the **available links** option button must be checked (this is currently the default).

The best way to achieve this aim is to use buttons composed of three words. Two word buttons may be used if one of the words is non-technical (i.e. the two words together are never going to be used as generic, local or specific link sources), but three word buttons are preferred. It is important that the text of a button is unique within its document, and this can sometimes be achieved only by using more than three words. Generally you should not use more words than are necessary.

2.3.2 Generic links (keyword links)

Glossary terms will be generic links. On the first occurrence of the term it should be emphasised by the use of **bold text**. It may be appropriate to add the definition of the

glossary term in the text at this point. The glossary entry will obviously be similar but may provide an expansion or generalisation. A list of glossary terms should be included in *comment style* at the end of the unit script. These terms together with their suggested glossary definitions should be shipped to Surrey where they will be collated and edited into the common glossary documents.

Data book items will also be destinations of generic links. Suggestions for inclusion into the data book should be discussed with module developers. In general the data book is aimed at being as inclusive as possible and it is current policy to make the same data book available to all modules..

In most cases an item will be the destination of more than one such link, with subject-synonyms being used. The text books will also be the destinations of generic links, so that a given word, for example **antinode**, will be the source of generic links into the glossary and the text book (to sections on strings and room acoustics), and possibly the glossary entry on standing waves.

Single word generic links are preferred. If multiple word roots have to be used, then they should be emboldened..Where multiple word generic links cannot be avoided, as in the case of **simple harmonic motion**, make the three words a source, then choose two of the three **simple harmonic** and then a single word **harmonic**. There is a registry entry for 'Show links' to operate on word pairs. To check this option is operating, from the SToMP floating toolbar select the restore icons command under the View menu. and select (single click) the Show link icon. Make sure under Options that 'Word pair' box is checked.

4. **Specific links** - these should be used where the user will have a high expectation of the existence of one. Thus, equation and diagram references will be specific links. In these cases, the code part will be part of the link source. e.g. equation(W3.4.1), or in a textbook, Figure(10.4). In this way the link sources will be unique, thus minimising confusion when using the link editor, and no conflict will occur with generic links.

Specific links should be single words only.

A feature of single word generic and specific links is that the word can be selected by double clicking. Otherwise, mouse dragging or keyboard selection has to be used which is much less convenient for the user. It also means that follow link can be used with confidence, which is quicker than show links.

Each author should identify the button, specific and generic links out of the unit that is being prepared. This list should be placed in a *comment style* at the end of the unit.

2.3.2 Associated documents (zero selection links)

Each script document will have a number of associated documents that will be either specific to that document (e.g. equations and symbols used) or general (e.g. quick help, data book contents).. A generalised method of reaching these documents has been defined in the **docs/policy/doclinks** document from Drs Barnett, Tyler and Rake. The documents proposed for this linking scheme are:

1. Symbols used in Document (use **local** link)
2. Equations used in this Document (use **local** link)
3. Module Contents page (generic link).
4. The data book Contents index (generic link)

Script authoring

5. Text book Contents index (generic link)
6. Training material Quick reference guide (generic link)

The **local links** from documents imported into microcosm should be added by authors. Make the links to these two documents as normal, except that *no text* should be selected. Make sure that you select the **local** link type.

The **generic links** for the Module Contents, Data book contents, textbook contents and Quick reference guide are added at Surrey by the SToMP editor during module integration.

2.3.3 Variables

Within text, variables and simple expressions should be authored using the Symbol font as far as possible. The reason is that MathType equations cannot be copied and pasted into other documents together with text. The inclusion of MathType objects within paragraphs thus makes copying blocks of text much more difficult for the user. If part of an expression has to be in MathType, however, then the whole of should be.

2.3.4 Equation styles

Within Mathtype, the font sizes should be set to:

| | | | |
|---------------------------|----------|----------------|----------|
| normal text..... | 13 point | symbols | 15 point |
| sub and superscripts..... | 11 point | sub-symbols .. | 12 point |
| sub-sub and sub-super ... | 9 point | | |

Note that Mathtype equations do not necessarily produce sufficient spacing between symbols, and that the 'nudge' feature (cntrl-arrow) should be used in these cases.

2.3.5 Colour

All text will be in black except for section headings which will be dark red (heading style 2 - the colour will be inserted automatically by the **stscript** macros).

The default background colour should be left white in all storyboards, the colouring of the background will be handled by the rtf viewer.

A light grey background (available by using the style **greyback**) will be used for the preamble paragraph (not heading) in each document and for paragraphs starting with the Question icon. for in-line questions and the self and assessed tests.. A light grey background may also be used for paragraph emphasis and for important equation emphasis. This last feature should be used very sparingly, i.e. less than one per script.

2.3.6 Figures

Small thumbnail sketches can be included in the script (as Microsoft Draw objects or as small bitmaps). More detailed figures will be in separate bitmap documents and accessed by a specific link. The figure(*number*) will be the link root.

e.g. Figure(2) The electromagnetic spectrum. (as a caption)

or

The various regions of the electromagnetic spectrum are shown in Figure(2).

If referred to later in the same unit then a specific link could be used. From outside the unit the figure would be referred to as Figure(W5.1.2), again as a specific link.

Alternatively an important diagram may be treated as an activity, with a picture icon paragraph (as described in section 2.2.5)

2.3.7 Tables

Tables will follow the same scheme as for diagrams and figures. If they are in the same document, then a specific link from, for example `table(3)` will be used. From outside the unit the reference will include the unit number, e.g. `table(W4.7.3)`.

2.3.8 Hidden files

When documents are imported into the multi-media data base, it is desirable that some of them are not visible to student users in the data-base index (Select a document box). Such files will be called **hidden**, and include:

- Question sheets for formative tests,
- Question sheets for summative tests
- Answers to 'in-line' questions,
- Introductions to activities.

Files that are to be hidden should be imported into the logical type *Application(e.g. Waves)/Files/Hidden*. This logical type will be removed during system integration.

2.3.9 Activity coding scheme

The coding system for activities is *XY.a.b.c[g]*

where

X is the Package

W - waves
M - measurement
Op - Optics
Th - Thermodynamics
As - Astronomy
Me - Mechanics
VI - Schools 'A' level

Y type of "activity"

A - animation/simulation (irrespective of development language)
Q - bitmap sequence
I - image (picture or diagram)
S - sound
V - video
B - biography
M - multiple choice test
T - text file

a is the block number.

b is the unit number in the block.

c is the (sequential) activity number within the unit. These are to be unique within a unit without reference to the activity type and need not be sequential.

The g would signify that this is a generic activity that would be expected to appear in several units, or in different guises in the same unit.

2.4 Authoring techniques

Note on numbering in Word2

This note is for those authors not familiar with numbering in Word.2

Type your script using the story.dot template in the View - normal or Page Layout mode, including the correct heading styles. Save to a filename.doc file. View in Outline with the number of levels required. Select all the sections to be numbered from say Introduction in heading 2 style to the final paragraph to be numbered in heading 3 style. Under the Tools menu select Bullets and numbering, then outline, legal auto update, OK. To update numbering View in Outline reduce to the number of levels required, select the sections that require updating. Under the Tools menu select Bullets and numbering and autoupdate.

3 Styles of Programmed Activities³,

3.1 General observations

- a) The intention is that programmed activities should look as independent of their development environments as possible.
- b) Programmed activities should always start up with an interesting set of parameter values (if appropriate). If a decision has to be made to achieve this, then it should be forced onto the user by starting the application with a modal selection box, where a selection must be made before the box can be removed.
- c) Activity windows should not be larger than 1/3 of the area of the screen at 800x600 resolution, and should always fit completely within a 640x480 screen.
- d) Keyboard equivalents of mouse actions should be supported where possible (i.e. where a suitable simple protocol can be devised).
- e) All activities should provide the user with a toolbar by means of the toolbar DLL. This will always display the **exit**, **link-to-main script**, **cross references** and **help** buttons. Where there is an introduction document, the **link-to-introduction** button should also be displayed. Other buttons will be available as appropriate
- f) Where activities can **proceed**, **stop**, **pause** and **restart**, then this will be controlled by means of VCR style buttons in the toolbar, as well as by means of drop-down menu items.
- g) ToolBook style **next**, **previous**, **first page** and **last page** buttons are available in the toolbar if required, and thus do not all need to be provided elsewhere in an application window. However, control buttons for the primary actions (e.g. next page) should always be included in the activity. If control buttons involving similar symbols (but for other functions) are required within an application (e.g. for moving a line), then the icons used must be different to those used in the toolbar.
- h) All activities should start up with the bookmark in the **locked** state.

3.2 Associated documents and links.

Every programmed activity will have a link from at least one script. In addition, many interactive activities will have introductory documents that describe what to do with them. An introductory script might suggest one or more 'experiments' that could be carried out, together with instructions on how to use the activity to do the experiments. The style of these instructions could be that of the so called 'tutorial introductions'.

In addition, in each programmed activity there will be a set of links from the activity to any relevant reference material, by means of an appropriate set of generic link words.

Applications will provide access to the script from which it was called by means of a "Links" menu item "Link to main text". This will usually be by means of the generic link to be provided for each unit, of the form "W1.4". If the unit is long, a special generic link pointing to the place in the unit from which the activity was called can be used, and this should be of the form "w1.4NameOfActivity" where *NameOfActivity* is preferably an abbreviation to keep it

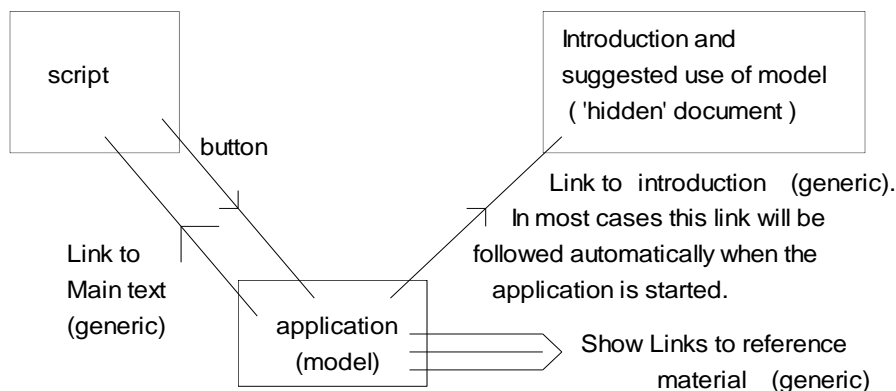
³Version 3.2 July 1994. Dick Bacon, Graham Keeler & Steve Rake

Styles of Applications

short. The preferred method for the activity to obtain the name of this and other links is from the mcs file. Thus, if the same activity is called from different units different mcs files should be used that have the appropriate link roots specified in them.

Those programmed activities that have an introduction document will provide access to it by means of an Action menu item 'Link to introduction' This generic link will be the file name (including extension) of the introduction document. The preferred means of reaching the description file is for the activity to automatically start it when the activity is started.

The reference set of generic link words will be accessed from the action menu via a "Cross references" item.



There are four links in this scenario

| Link | Description |
|--------------------------|--|
| Script to Application | This is a Microcosm button |
| Application to Script | This is a Follow Link, where the link is Generic. The root of the link is to be the unit number, possibly plus the activity name. e.g. m2.3 or m2.3NameOfActivity. |
| Application to Intro. | This is a Follow Link, where the link is Generic. The root of the link is to be the filename of the intro. document. This filename should follow the conventions described in section 2.1.5. |
| Application to Reference | This is a Show Links, where the text passed to Show Links contains words that are in the Reference material and are relevant to the application (Notes 1 and 3) |

Note 1: Follow Link and Show Links messages can be sent to Microcosm using the mcmsg.dll code. In ToolBook the messages can be sent directly (stevetst.tbk has an example of a Show Links)

Note 2: By using the Generic link approach, the application does not need to become Microcosm aware. The Generic Link root can be defined using the Selection filter. The root must be chosen so that it is unique to this link (if it is not unique, an Available Links box will appear and confusion will reign!).

Note 3: Show Links allows a number of words to be passed to Microcosm as roots of links. The text string in this case should contain words that are roots of generic links into the reference material.

N.B. The toolbar DLL now provides all the functionality to read the link information from the MCS file and follow the appropriate links as necessary.

3.3 Help

Windows style Help should be available for every programmed activity. The information in these help files should be aimed at helping the user with the windows style interface, explaining the functionality of menu commands, buttons and other controls. A brief description of the activity should always be supplied within an introductory topic. Normally the introductory document should be used to describe the physical significance of the activity, and the help file to describe how to control it. If for some reason it is not felt necessary to provide an introductory document then there might be some description of the physics in the help file.

3.4 Author and user modes.

Simulations that have start-up data files for when they are used as animations should be defined to have two modes of operation - author mode or user mode. In author mode it will be possible to generate start up files interactively. This will involve changing the model's parameter values until the model operates in the required manner, and then saving these values as a start-up data file. In user mode the parameters will be adjustable in a similar way, but the user will not be able to save the data to a file.

The preferred scheme to facilitate this is to have **Load file** and **Save file** commands under the **File** menu option in author mode, (or when not started with a preset data file), but only the **Reload data file** command in the **file** menu when in user mode (or when it *has* been started with a preset data file).

The parameters should be available via the **option** menu item.

The registry can be interrogated during launching by means of the .mcs file passing a special syntax command line parameter within %% symbols.. The registry can also be accessed from within an application, see the document **iniman.doc**.

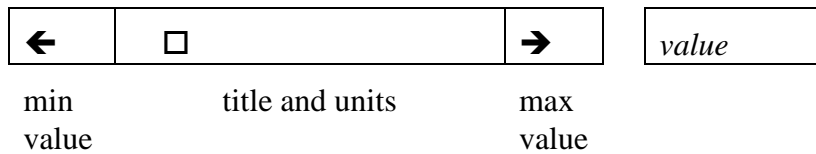
3.5 Controls.

There is clearly a wide range of required styles of control and it is difficult to be prescriptive about how these should be organised. The following notes should be followed in detail if possible or in spirit if not. If in doubt, please discuss specific cases with unit authors and the project multimedia editor.

a) There will always be a toolbar available, with as a minimum an **exit** button, follow link to **Main text** and **Cross Reference** buttons and a **Help** button. If there is an introductory document, then there will be a follow link to **Introduction** button, If required, controls of **start**, **pause**, **reset**, **stop** style should be in the toolbar as discussed above (VCR) or the ToolBook style **forward**, **back beginning**, and **end** icons. The **Help** icon should always be on the right. The **exit** icon should always be on the left.

b) Where parameters have to be adjusted then scroll bars should be used, in conjunction with a value box, where the value can be entered directly. These should be arranged like this

Styles of Applications



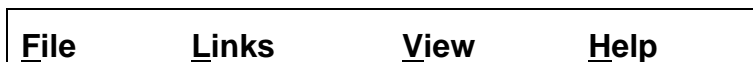
c) Controls that can have immediate effect on a static view should be available (i.e. visible) all the time. If the controls affect the starting conditions of a progressive model (i.e. one that uses the VCR buttons) then they should be available only on request (via the toolbar or menus), and when the model is stopped.

d) Controls should usually be in a separate (floating) window. In some special cases (e.g. where there are only one or two controls) the implementor may wish to have these controls permanently on screen in the main window. As a rule of thumb, this is acceptable if the area taken by the controls is no more than 1/3 of this window area.

e) Controls not supported by the VCR buttons or the **next**, **previous** etc. toolbar buttons, will be offered under the **options** menu item. In certain activities, however, it is accepted that their use will require them to be actual buttons, and that they might need to be permanently visible. In this case they should be arranged so as to conform to the 1/3 window area rule (above). Buttons with relief indications are preferred but not mandatory.

3.6 Menu bar

A menu bar should be provided, and it should always contain at least



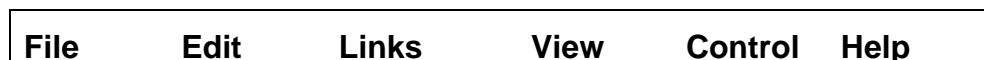
The **Help** item should be left adjusted with the other items in the menu bar, (not right adjusted as in some older windows applications).

If there is a requirement for an edit item, then it should go between **file** and **links**

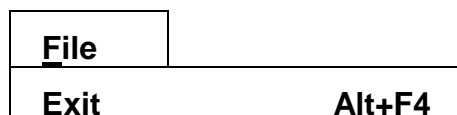


The **edit** item should be used, among other things, whenever 'cut and paste' functionality is required.

If other menu items are required, such as **control** or **options**, then they should be put between the **View** and the **Help** items.



The file menu should contain at least



or, if there are associated data files:

Styles of Applications

| | |
|------------------------------|------------------|
| <u>F</u>ile | |
| <u>O</u>pen | Ctrl+F12 |
| <u>S</u>ave | Shift+F12 |
| S<u>a</u>ve <u>a</u>s | F12 |
| <u>E</u>xit | Alt+F4 |

or, if started with a preset data file:

| | |
|--------------------------------|---------------|
| <u>F</u>ile | |
| <u>R</u>eload data file | |
| <u>E</u>xit | Alt+F4 |

The links menu should contain, with the introduction item optional:

| | |
|------------------------------------|--|
| <u>L</u>inks | |
| Link to <u>m</u>ain text | |
| Link to <u>i</u>ntroduction | |
| Cross <u>r</u>eferences | |

Link to main text will follow a single source word chosen to be the unit number generic link to the script document from which the application was accessed. In general such links will already exist and will be called the name of the unit, e.g. W3.5 or M1.2. **Link To introduction** will (if appropriate) follow a single word generic link to the introduction document. **Cross references** will perform a 'show links' on one or more suitable words that match entries in the textbook, glossary, data book, etc.

If required, a control menu item would contain

| | |
|---------------------------------|--|
| <u>C</u>ontrol | |
| <u>S</u>tart/<u>S</u>top | |
| <u>P</u>ause | |

The view menu should contain at least:

| | |
|-------------------------------|--|
| <u>V</u>iew | |
| <u>T</u>oolbar | |
| <u>S</u>tatus bar | |
| <u>U</u>ppdate colours | |

This menu may contain other viewing options suitable for the application, such as zoom. The update colours option should re-obtain the colours from the colour DLL, so that a user can change an application's colour (using the palette tool) without leaving and restarting it.

The help menu should contain:

| |
|--------------------|
| <u>H</u>elp |
|--------------------|

| |
|---------------------------------|
| <u>C</u>ontents |
| <u>H</u>ow to use help |
| <u>A</u>bout <i>Name</i> |

The aim here is that the contents menu will lead to only one contents page in which all the options are listed. The contents page is to contain the following headings,

| | |
|-------------------------|--|
| Introduction | (introduction to the program) |
| Using the program | (optional summary of features) |
| <i>Other headings</i> ★ | (these will be the help items for the program) |
| half line space | |
| Link to Main text | (describes how to get to main text) |
| Link to Introduction★ | (describes how to get to introductory document) |
| Cross-References | (describes how to reach cross references) |
| half line space | |
| Menu Bar | (how to use the menu bar features) |
| Toolbar | (how to use the toolbar features <u>used in this program</u>) |

★ - where appropriate.

'How to use help' may be renamed 'Using help'.

An **options** menu item should be a feature of most model based simulations, and would offer the user the possibility of changing the model's parameters.

3.7 The Window

a) The style of a window can be simple, with just a graphical display that is controllable via toolbar or menu bar.

b) It can be composite, with areas containing different types of output (text, graphic, picture, etc.). In this case consideration should be given to putting the different output types into different windows.

c) The style can be multi-paged ('ToolBook' style).

In the case of b) or c) use can be made of the 3D (or relief) style of window appearance using light and dark greys (in a similar manner to buttons).

d) The text to appear in an application's title bar should be taken from the description in the MCM docuverse. Information can be obtained from the registry by means of the MCM application launcher. Alternatively, this text can be obtained from the mcs file (see section 3.15.5)

e) The mcs file can be used to define many of the startup parameters. It is styled as a windows "ini" file, and the details of comonly used parameters are given in section 3.13.5.

f) The preferred style is for a window contents to rescale when it is resized. In some cases this is not practical (e.g. with a composite window) and in such windows: i) the window should not be allowed to be resized larger than the full size of the application display, and ii) if resized smaller the area of maximum interest should be kept in view as far as possible. This will often mean that the area of most interest will need to be placed in the top left. If a diagram or graph is resized, the lines should wherever possible change thickness to keep the relative line weight constant.

f) The position of a new window on the screen should be chosen with care so that it matches the size and position of the script from which it would normally be called. The preferred way of doing this is to pass such information in the mcs file (see section 3.13.5)

g) Error boxes should default to the centre of the screen, not the centre of the current window.

3.8 The 'about' box.

A standard SToMP "about" box has been produced as a DLL and should be used for all applications. Title, dates and names of authors can be passed in the usual way. This is described in section 3.13.3

3.9 Dialogue boxes

When values are being changed by the user by means of a dialogue box, then

- a) the program should constrain input numbers to sensible/possible values.
- b) a **Modal** dialogue box should be used when restarting the model is implied.
- c) in static displays the box should be allowed to remain on screen (see above).
- d) If there are two levels of options, then the first level should have an **options** button to get to the next level of parameters.
- e) If an error box is produced, then it should be centred on the screen (see above).

3.10 Text

- a) Text in menu bars, drop down menus, etc. should match normal windows practice.
- b) Text in an application window should normally be 12 point New Times Roman. Symbols and equations should be in 13 point. At the author's discretion, however, text may be up to two points larger or one point smaller, and where contrast is required, the Arial True-Type font may be used sparingly. Within any one application the fonts and sizes used must be consistent.
- c) 12 point Arial is the standard for graphs, and MS sans serif for buttons.

3.11 Pictures

a) Some pictures are being used that are not true representations of physical situations (e.g. experimental apparatus chosen for their visual effect rather than authenticity). Such pictures should be titled, and the word **schematic** should be included in the title. It is also preferred in such situations, that a true representation (e.g. a photograph or diagram) is available that can be called up and viewed by the interested student.

3.12 Colours

3.12.1 Windows colours

The windows colours for:

| | |
|-----------------------|-------------------------|
| Menu Bar | Menu text |
| Active title bar | Inactive title bar |
| Active title bar text | Inactive title bar text |

Styles of Applications

| | |
|------------------|-----------------------|
| Active border | Inactive border |
| Window frame | Scroll Bars |
| Button face | Button shadow |
| Button text | Button highlight |
| Disabled text | highlight |
| highlighted text | application workspace |

should not be changed from the windows default values. These colours will then be under the control of the user via the standard windows **main/control panel/colors** control, which is also where you should look if you are in doubt about where these colours are used.

The following two windows colours are not to be used directly within SToMP programmed applications.

window text
window background

They will probably be used later as a basis for generating some of the SToMP colours to be described below.

3.12.2 SToMP colours for programmed activities.

| | |
|--|------------------|
| application background 1 | application text |
| application background 2 | application draw |
| panel background | panel text |
| graph background | graph line1 |
| graph text | graph line2 |
| graph axes | graph line3 |
| graph axes text (including annotation) | graph line4 |

The colour sets, application, panel and graph will contain colours that contrast with the background in each case. These represent potential colours - in many cases the same colour will be used in several different places (e.g. application text/draw, panel text, graph text/axes/axes text/line1 may all start out as black).

Two default colour schemes have been defined, one with dark objects on a light background, the other with light objects on a dark background. An automatic choice is made between these at start-up time, depending upon the windows colours being used. The SToMP colour palette utility allows the user to change the colours used in all conforming applications.

A diagram (static or dynamic) within an application can use the application colours or the graph colours.

The background colours should only be used for background, but the 'application text' colour may also be used for drawing.

A graph using multiple lines should always have a key, and this should be on the graphics background colour. This is to enable reference to the different lines from an accompanying script, without the script writer being aware of what the actual display colours will be. It follows that such references **must** be to the line descriptor used in the key.

Where representational colours are required as in a picture or schematic diagram, then any colours from the 256 colour windows palette may used, but **do** remember to look at it in 16 colour mode to make sure it does not then convey a different message!

Styles of Applications

A DLL has been produced that interrogates this SToMP colour palette allowing all applications to conform to this model.

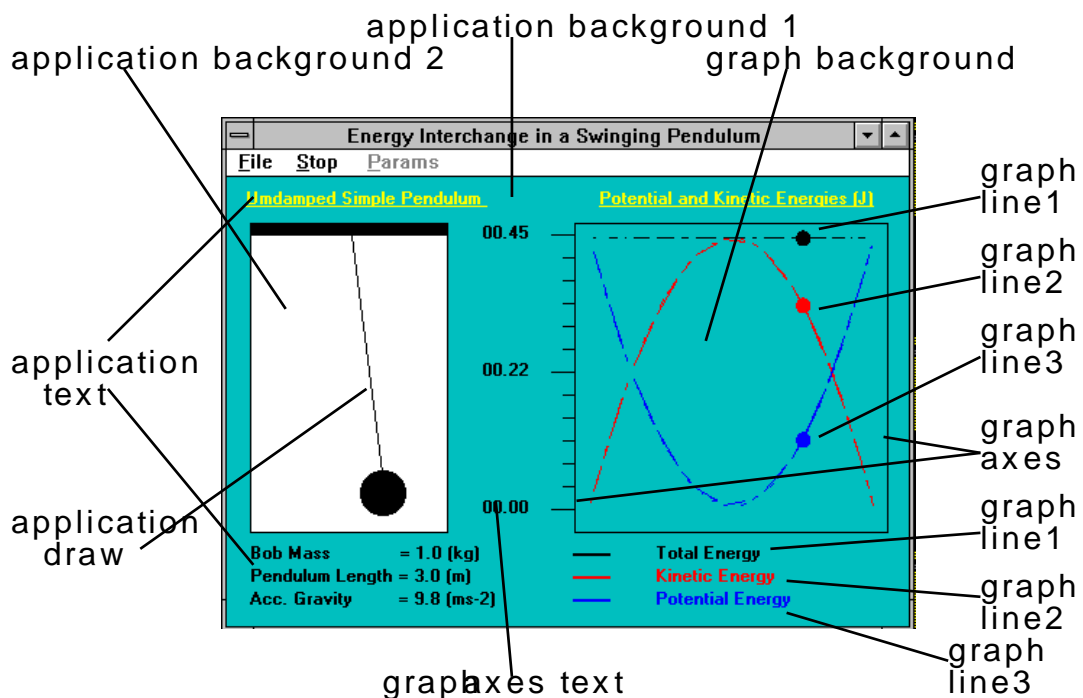
Applications using an MDI window should use the windows application workspace colour.

Some applications have already been prepared using a 3D (or relief) effect, and for some applications this may well be appropriate. In such cases the relief colour scheme is dictated by the application generator.

The following examples have been selected to demonstrate how the colours should be used. The examples do not always conform to the new prescribed colour styles and as you will see, the new style is not always even prescriptive. The main aim is to provide sets of colours that contrast foreground with background, and in a scheme that will provide some consistency across our various applications.

Please note also that the following examples are only being used for the colour - there are other features that have been the subject of debate and will be covered soon in another specification document (or two).

3.12.3 Colour example 1.

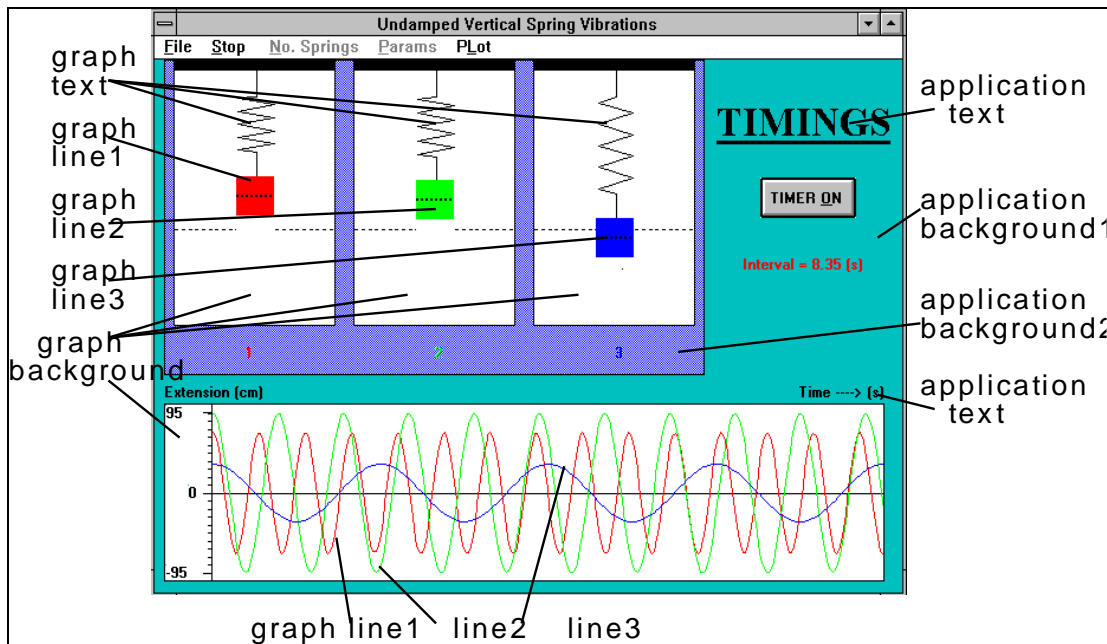


In this example it is not clear where the graphics area ends, so that 'graph background' might be better than 'application background1' for the whole window. 'Graph text' would then be used in place of 'application text'.

The coloured title text is not supported in the new scheme - the panel scheme is really for blocks of text and is not intended for highlighting titles.

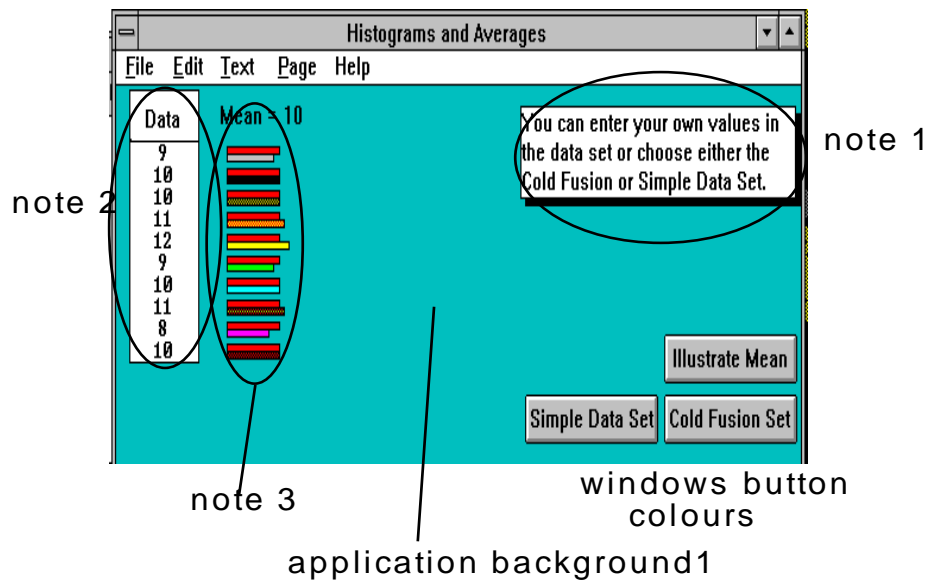
Note the graphic key has text in the colour of the line (this is not necessarily the preferred style). The key must be on the graphic background colour - another argument in favour of most of this window being graphics background.

3.12.4 Colour example 2



In this example, the text under the timer button could use the 'application draw' colour. None of the graphics colours would be appropriate for this text because they would not necessarily contrast with the 'application background1'. Similarly, the windows 'disabled text' is not appropriate because the text is not on the 'menu bar' background. Note that the panel text should only be used with panel background, but that another option in this case would be to put the whole timer button and text into a panel using the panel colours.

3.12.5 Colour example 3

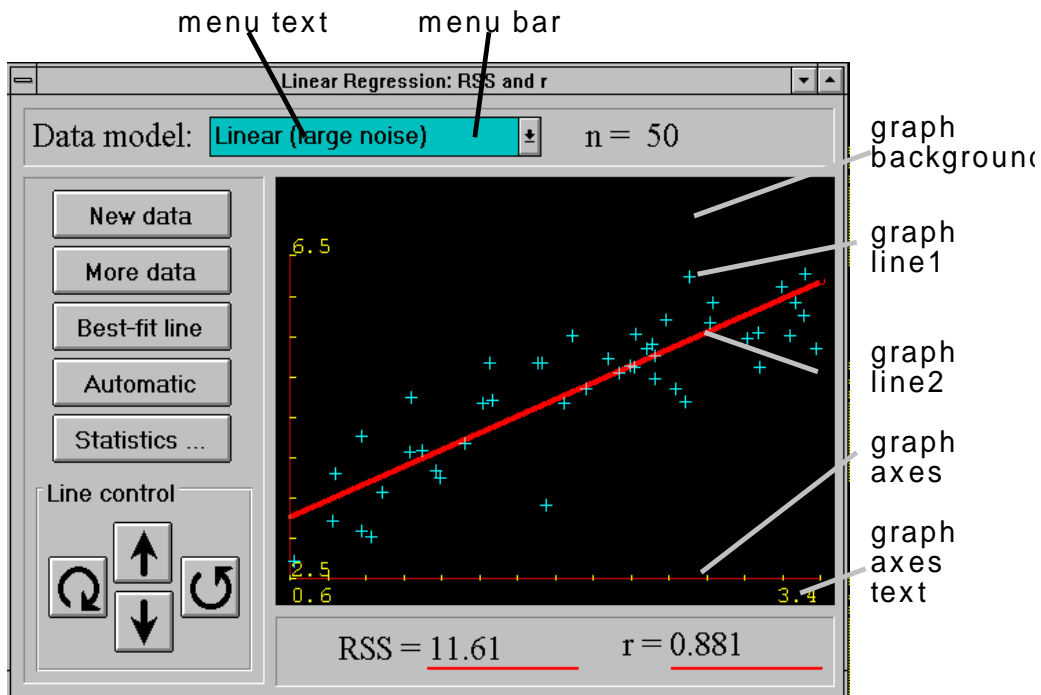


Note 1. This text panel could either be in application text on application background2, or in panel text on panel background. In the general case the choice would depend upon what other text the window contained and what other use was already being made of these colours.

Note 2. This could be the same as the text panel, or could make use of graphics text and background.

Note 3. This specification does not cover the use of colours in this way. When such a requirement occurs then use your best judgement, remembering that sometimes it will be viewed with only 16 colours available. Note the convention used here where each area is outlined in black. This ensures that each light coloured area is seen, whether the background is light or dark, but does not necessarily achieve this with a dark area on a dark background.

3.12.6 Colour example 4



Note how the windows 'menu bar' and 'menu text' colours are to be used in the combo-box. The 'menu highlight' and menu 'disabled text' colours should be used in the drop-down list where appropriate.

3.13 DLL's, components and other support features

3.13.1. Toolbar DLL

The Toolbar DLL and the Status Bar DLL supply software developers with a set of easy to use functions for creating and manipulating a Toolbar and a Status Bar within each of their own applications. The functionality can be accessed from programs written using Pascal for Windows, Visual Basic and Visual C++. Toolbook applications can also access the DLLs by means of the stumpsys.tbk startup book prepared by Alsiatir Gillies. The toolbar and status bar DLL's are available within Delphi as Delphi components by means of some code prepared by Chris Meenan.

The toolbar DLL supports the standard SToMP buttons, exit, link to main, introduction and cross references, the VCR buttons, the page turning buttons, and some others. In addition developers may define their own buttons and create their own status-bar messages for them.

See the SToMP Toolbar and Status-bar DLL's User Guide by Richard Moran.

3.13.2. Aboutbox DLL

Get from Richard's summary

3.13.3. Colour DLL

Get from Richard's doc

3.13.4. mcs file format

Each activity is launched by means of an '.mcs' file, and it is this mcs file that is imported into the Microcosm docuvers. One program can have many different mcs files, that start it up in different modes, with different data files or with different introductory documents.

When an activity is launched from Mirocosm, the mcs file is read by microcosm, and the command line found there is exected. After the activity is started, it can also open the mcs file to obtain link information (to main script, to introductory document, etc), the name of the program's help file, and other information laid out in this definition.

The mcs file is organised as a conventional windows .ini file.

Any line starting with a semicolon is a comment.

Under the heading [**Microcosm script**], there are two possible entries:

Command= this is mandatory, and specifies the DOS style comand line required to launch the application. There is a convention within SToMP aplciations that the second parameter is the name of the mcs file, in order for the application to open it to obtain the data under the later headings.

LoadDLL= this is optional, and allows specific dll files to be loaded before the application is launched. Since the path to the stomp/bin directory can be specified (as shown in the next paragraph) this obviates the need for dll's to be placed in a system directory.

Within the above two entry's strings, any text enclosed by percent signs (%antext%) are treated as entries in the SToMP registry, and the entry text subsituted. Thus, to run a ToolBook application where the toolbook run files are in the stomp/bin directory, the following command entry is used:

```
Command=%/system/stomp/stbin%mtb30run.exe $CurDir$wfa4_4.tbk $Description$
```

The text "/system/stomp/stbin" is the full entry name for the registry entry containing the absolute path to the stomp/bin directory.

This example also illustrates two further features of the mcs files. There are some specific text strings which, when placed inside \$ \$ symbols are expaned before launching. Thus, "\$CurDir\$" expands to the full path of the directory containing the mcs file being processed. The \$Description\$ string expands to the text description of hte mcs document in the applciation docuvers (i.e. the description given when the mcs file was imported).

In a similar way, the string \$Filename\$ expands to the filename of the current mcs file.

In the toolbook application shown above, the documnet description was used in the title bar of the program. This is an old standard that has been superseded. The text to go into the title bar is now normaly obtained from further entries in the mcs file, and for this purpose the mcs file is specified in the command line. An example is shown here:

```
Command=string.exe wd4_6_31.mcs
```

This comes fromt he mcs file wd4_6_31.mcs which has further standard entries:

```
[Application]
Title = Clarinet model
MainText=W4.6a
Introduction = w4.6o
```

CrossReference = standing clarinet

IntroOnStartup = False

HelpFileName = String.HLP

DataFileName = clari2.dat

The title entry appears in the title bar.

The Maintext entry is the text of the generic link to the script from which the application was linked.

The Introduction entry is the text of the generic link to the introductory document.

The CrossReference entry is the text that will be passed to a Microcosm 'show links' action when the cross-references button is pressed.

The IntroOnStartup entry selects whether or not the introductory document should be automatically started when the program starts.

HelpFileName is self descriptive.

The DataFileName entry is used in this case to specify the data file the application is to use on startup.

These entries are used as required, and their presence and processing (by the program) is optional. New entries may be specified for particular features of any new program. If a new entry is likely to be used by others it is helpful to discuss its syntax before finalising it.

each window that a program uses should also have a set of entries, e.g.

[Window 1]

Left=0

Top=300

Height = 300

Width = 400

;Caption = Main Window

These must be processed by the program. They are needed by the integration team to finalise window positioning. Window1 is the main window, and so the caption entry is not required (it was specified in the title entry above) and so it is 'commented out'.

Some programs have many more entries (e.g. the ripple tank program specifies the colour mapping in the mcs file), but the main entries under the application header are now processed automatically by the toolbar.dll code.

4 Editing guidelines⁴

4.1 Introduction

These guidelines are for the informal use of the editing panel and subsequent editors.

The editing operation will cover all scripts and also those applications that have significant amounts of text (e.g. many of the ToolBook applications).

Scripts should be edited from the original storyboards, using the revision marks feature of Word. (under Tools/Revision marks).

Applications should be obtained as executables and with a hard copy of each successive screens. Editing suggestions will be written onto the hard copy.

Editing is being undertaken:

- for clarity
- for the correction of errors
- to create consistency (within each document or unit)
- to address the student in an agreeable manner

but as far as possible gross restructuring should be avoided.

4.2 Guidelines

2. Tenses - generally the present tense is to be used.
3. Uniformity - we should look for uniformity within each script for items not covered by these guidelines (e.g. for some capitalisation).
4. colloquialisms - avoid the use of "let's" and "here's".
5. Policy on numbers - the integers one to ten should be written as words, all other numbers including decimals to be written as digits. In some cases it will be necessary to write the number 1-10 as digits.
6. Sentence length The first paragraph in a document, and the first sentence in each paragraph are more important than the others. Thus they should be short and clear. There is no reason why a first sentence should not be short even if other sentences in a paragraph are (relatively) long.
7. SToMP technical terms - these should not be used. e.g. rtf, bitmap, Visual Basic, etc.
8. Paragraphs - in general paragraphs should be short, because of their appearance when a window is narrow
9. Hyphens - these should be used only when necessary, not to tie-words-together.
10. Colon - valid use is to separate two balanced halves of a sentence (for contrast) or at the start of a list of semicolon separated clauses.
- 11 Units - look out for the use of units, they should be present, correct, and consistent.
- 12 Significance - check that the significance of numbers used is consistent.

⁴Version 1.1 August 1994. John Davies, Steve Swithenby, Dick Bacon.

Editing guidelines

- 13 Upper case - main headings (document heading) will use upper case on main words. In section headings only the first letter of the first word.
- 14 &, i.e. and e.g. - the ampersand should not be used. "i.e." and "e.g." should only be used in bracketed phrases.
- 15 Addressing the student - this should be done with care, see below.

4.3 Addressing the student

Use of 'you' and 'we':

Using **you** in phrases like 'you should now run the simulation...' or 'you can see from equation 4...' is fine, but beware of the style 'you will be dealing with ..' which can easily have an aggressive or even a condescending feel to it.

The use of **we** is to be minimised. When used it can mean 'the scientific community' (we all agree..), 'user and author' (we will shortly be considering..) or in some rare cases 'the course developers'. It should never be used when 'you' is meant (we will now run the simulation..)

4.4 Example

The following two paragraphs come from unit M2.3A, and introduce the section on the **Mean value of a data set**. They give a good example of 'we' being overused, and also being used inconsistently (i.e. referring to different groups).

We shall be dealing in all our work with sets of experimental data, and these arise in different situations. The first that we shall consider is a set of repeated readings of the same quantity, for example the resistance of a coil, or the mass of an object.

We may also repeat the same experiment a number of times. The experiment may be a simple one of finding the voltage drop and the current flow through a resistor or it may be much more elaborate. We may also measure a set of nominally identical quantities. Few of us would doubt that for the last two cases the results in the data set may differ somewhat. What may come as a surprise is that even repeated measurements on the same quantity will differ by amounts close to the accuracy of the measurement.

It is suggested that these two paragraphs could be replaced by:-

Science involves experiments and the collection of data. Perhaps the simplest type of experiment is the repeat measurement of a single quantity, for example a series of measurement of the resistance of a coil. This is where we shall start.

In the original, the second paragraph repeats the ideas of the first, and then presumes too much about the user's ideas/knowledge. In any case, this is an 'A' stream document, so the users should be familiar with the concepts anyway.

The modified form is succinct, it introduces what is to be dealt with in the unit, and ends with a punchy, contrasting short sentence.

We should look at such paragraphs that jar in style and suggest modifications.

5 Testing style guide⁵

5.0 Re-implementation of the testing tool.

The testing tool is currently being re-implemented. The style of the tests will remain the same, with questions being in formatted text files and student responses being handled by a separate answer box, but the 'look and feel' of the answer box is being completely changed with the answer box for only one question being visible at a time (in default mode) The new version will be functionally compatible with the old but the format of the control file has been changed. However, a control file editor is being prepared that will automatically convert data from the old format to the new.

A functional version of the new testing tool and control file editor should be ready early in May (95), but since conversion of the data files is automatic, you are recommended to continue developing multi-filed self tests and assessed tests according to the following specification, and perform the conversions when the new software becomes available. The question files and the mcs files will not require any changes.

Note, Feb 1996. The re-implementation is complete, and a control file editor is available. The rest of this chapter needs minor modifications to bring it up to date, but the principles remain the same. It will be updated in the near future. A new question style has been introduced that allows numeric questions to have values randomised. This is not yet fully tested, but details and a final version should be available soon.

5.1 Changes

Changes between version 4 and version 4.2

1. The answer sheet and the control file syntax have been extended to include questions with numeric answers.
2. Multiple questions files are now supported. The number of files is specified on the command line together with the style of test.
 - If self-test (formative), then the user selects the question file
 - If assessed (summative), then a question file is randomly selected.

Changes between version 3 and version 4

1. No changes have been made to the specification of the Control (.txt) file. All existing tests will still work
2. The Microcosm linking has been extensively modified and is described below. The essential difference is that the RTF viewer is now launched 'automatically' by an 'autodispatch' message to Microcosm. There is no longer a need for a Generic link from the Test Tool to the RTF Question File. A small (optional, but a good idea) addition to the .MCS file allows the sttest.exe to be placed in mcm\bin and eliminated from the quiz directory.
3. The Answer Sheet has the rubric. The rubric can be removed from the top of the Question File.

⁵ Version 4.0. October 1994. Dick Bacon, Steve Rake.

4. There have been a number of small changes to the appearance of the Answer Sheet.
5. The feedback window is no longer the standard VB Message Box but more tailored to our needs. On its first appearance, the Feedback window is always positioned below the Answer Sheet. If the Feedback Window is moved, the new position is retained.
6. There have been changes to the Explain process.
7. When a question is correct, there is visible feedback that the student has been successful. The student can make several attempts to get a question correct.
8. The user is always prompted to verify an Exit from the test.

For all this to work it is essential to have the new version of **mcmmsg.dll** which re-implements Autodispatch of Microcosm documents. Use a version dated 6/13/94 or later. If you get a message about being unable to find the function QD_FileGot then you are not running with the latest version of **mcmmsg.dll**.

5.2 Student assessment in SToMP.

Where this style guide refers to the testing package, it is referring to the new version released with version 3.00. This can log student responses and handles four styles of multiple choice question, a free text and a free numeric response and can put random numeric values into the questions..

Three schemes for testing students are to be used, two providing feedback and the third for use in summative testing. In summative, the SToMP resource material will be available but there will be no direct support linking (i.e. linking to the SToMP/Microcosm resources) as in the other two versions.

5.3 In line testing

The first scheme is for questions to be put directly into a script, with some possible solutions being part of the question text (or immediately following). These possible solutions will be MCM buttons linking to support material as required. This will be called **in line testing**.

Both the second and third schemes will use the **testing package**, one with and one without active linking to the rest of the SToMP material.

5.4 The testing package.

The testing package has an Answer Window to handle student responses, and uses the rtf viewer to present questions (authored in Word). Each Question (**.rtf**) file has one or two associated text control files (**.qni** for all types of test, plus an **.ani** for assessed tests) that contain information to set up and control the Answer Sheet (e.g. the number of questions, the number of choices for each question, etc.) and the marking of the questions. Students will select their responses by mouse or keyboard action or by typing text into text boxes.

For *self tests* the **qni** file contains the set up information for the answer window and the answers with feedback messages.

For *assessed tests* the set up information is in the **qni** file but the answers are in the **ani** file. The **ani** files with the answers are not available to students.

Six types of response style are available:

testing style guide

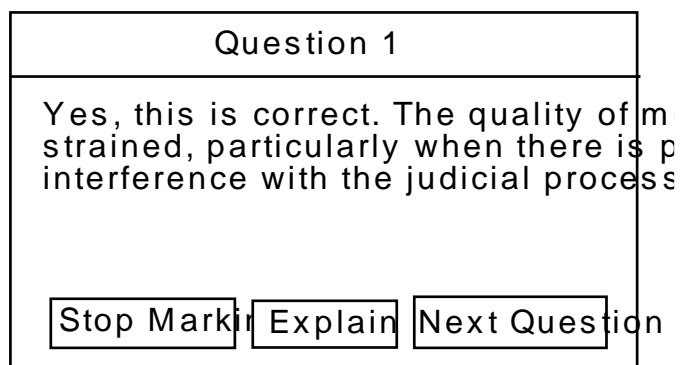
| | |
|----------------------|--|
| list (radio buttons) | in which only one item may be selected from a list. The Answer Sheet displays radio buttons for this style. List items in the Question File should be labelled a), b) etc. |
| list (check boxes) | in which one or more items may be selected from a list. The Answer Sheet displays check boxes for this style. List items in the Question File should be labelled a), b) etc. |
| rank ordering | in which a list of items in the question must be ordered. The Answer Sheet shows a list of characters (a, b, c . .) and allows the user to drag them into the required order. List items in the Question File should be labelled a), b), c) etc. and the rubric should state clearly which direction the ranking is required. |
| pair matching | in which some or all items from two lists must be matched. The Answer Sheet displays two lists, one with upper case and the other with lower case letters. The user must drag a line from each item in the top list (upper case) to items in the bottom list (lower case), to indicate pairings. The top list is labelled A), B),C) etc. and must be at least as long as the lower list which is labelled a), b), c) etc. The lower case list items may each pair with more than one upper case list items, but not vica versa. In the answer box the user can pair appropriate items by clicking first on a character in the top list and then on a character in the bottom list. |
| text | in which a text string must be typed in. The answer sheet displays one text box into which the number must be typed. The correct answer can be defined as alternative text strings containing wild characters. |
| numeric | in which the answer is one or more single values. This type is the same as the random numeric type, as described below, except that no random valued parameters may be used in the question. |
| random numeric | the answer may be a single value or multiple values (as in specifying a vector). If multiple values are required, then each is typed into a separate text box, which can be arranged in a row, a column, or in a matrix. Rubrics of the form 'x=' 'y=', 'z=' can be put before each row. The question file can contain values that will be randomised when the question is set, and the number of significant digits with which each value is to be displayed can be specified individually. Both the number of significant digits and the number of places of decimals can also be specified both as values or as ranges. The correct answer (and alternative answers) can be defined as a single value, a range of values or as an expression involving the randomised values. |

In self test mode

In the answer window an **Explain** button will be displayed if the question keywords have been edited into the control file. These keywords are passed to Microcosm with a *show link* action when the **Explain** button is pressed. These keywords can either be carefully chosen to lead to existing background materials, or they may be unique (e.g. w1.2q3.4) to lead to a specific document for a given question.

When a student has finished a question, clicking the **Mark** button marks the question and provides feedback via a Feedback Window, as illustrated in the examples shown below. It is possible to create specific feedback messages for all possible wrong answers or for classes of wrong answers.

An **Explain** button only appears in the feedback window if the Control File has the appropriate keyword entries. Helpful messages in the Feedback window should be provided in *simple text*. A message containing symbols and/or Mathtype equation(s) must be placed in a separate file accessed from the explain button through a unique generic link.



5.5 Question Files

The question files are conveniently authored within a unit's Word storyboard document. In this case the destination directory or filename must appear at the start of the first line for the *splitfile* macro to function correctly, e.g. for W2.1 unit question sheet 1

Directory c:\mcm\waves\quiz
 Filename wm2_1f0.rtf

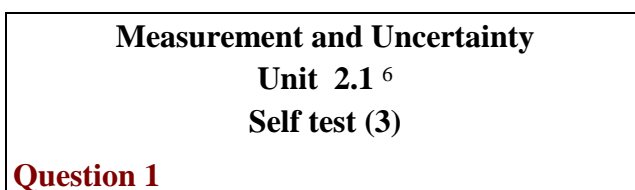
Subsequent tests for the same unit *following on* in the storyboard do not require the directory specification. They can start with:

Filename wm2_1f1.rtf

Filename wm2_2f2.rtf

Alternatively the files can be prepared in Word separately from the script (using the storyboard template), and then saved as rtf files when ready.

Each question sheet should start with a three line title (as described in chapter 2) as shown, with the copyright footnote added after a single space at the end of the second line. The number in braces after "Self test" is the number of the question sheet. Single (1) and multiple question sheets (1), (2), (3), for a unit should each be given a number. The question number is in heading2 style (i.e. in dark red).



There is no constraint on students to answer the questions in any given order, or even to answer all the questions.

5.5.1 Multiple choice or rank ordering questions

Where the question is in any of the rank ordering or multiple choice styles, care should be used in laying out the lists. Lower case letters should be used for the lists (to match the answer

⁶© 1994 University of St Andrews

Written by R. P. Edwin, A. D. Gillies and W. J. Webster

sheet), and the selection should be layed out to maximise visibility. For example, if each answer is short, the list can be put into two columns.

- (a) 22.30 ± 0.20 (d) 20.30 ± 0.02
 (b) 35.05 ± 0.00 (e) 20.30 ± 0.01
 (c) 18.00 ± 0.01 (f) 20.30 ± 0.03

or possibly on one line

- (a) 0.032 (b) 0.02 (c) 0.04 (d) 0.025

If each answer is a sentence, then use a hanging indent for each option:

- (a) The flute is closed at the blown end and and open at the far end, so that the fundamental mode has one quarter wavelength in the instrument's length.
 (b) The flute is open at the blown end and and open at the far end, so that the fundamental mode has one quarter wavelength in the instrument's length.
 (c) The flute is closed at the blown end and and open at the far end, so that the fundamental mode has one half wavelength in the instrument's length.
 (d) The flute is open at the blown end and and open at the far end, so that the fundamental mode has one half wavelength in the instrument's length.

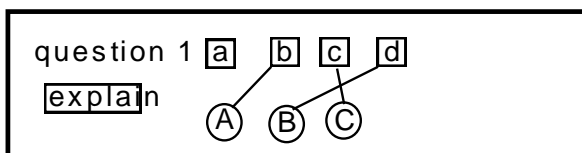
5.5.2 Pair matching questions

When using pair matching, the two lists are best arranged in two columns with the items in the list on the right being matched to the items in the list on the left. The left hand list will use lower case (a), (b), (c), etc, and the right hand list will use upper case (A), (B), (C), etc.

For example:

- (a) flute (A) brass
 (b) trumpet (B) string
 (c) timpani (C) percussion
 (d) violin

In this case, a correct response could look like this:



An item from the lower case list may match more than one item in the upper case list, but not the other way round. Either list may be shorter or longer than the other, however.

Do not use the word 'Answer' before the lists and do not indent the list if more than one column is used or if lengthy text is used.

5.5.3 Textual and numeric questions

Question files styles require no specific considerations, except that it should be remembered taht it is difficult to prepare questions that elicit a single unique text answer.

5.5.4 Random numeric questions

A random numeric question must be identified in the Word document by inserting the bitmap c:\docuverse\rtfvalq.bmp (a roundal) on the first line after the question number. N.B. this

bitmap must be *linked to file*. The question files will be processed during final integration so that this bitmap will not be visible to the user taking the test. For each quantity to be randomised insert the bitmap `c:\docuverse\rtfvalue.bmp` at the position the value is to appear. This bitmap must also be *linked to file*. The rtf file displayed to the student will have specific values in place of the variables.

There are several parameters that can be specified for this style of question: the number of significant figures and the number of decimal places of the displayed values, and the number of significant figures to be used in the answer. The answer can be specified as a single value, a column or a row vector, or a two dimensional vector. For each answer an expression involving the variables is required. Different precisions can be specified in alternative (or wrong) answers.

5.6 The Associated Data file

Formative tests (self tests)

The actual test control (qni) file produced by the test editor has the following entries:

[Header]

title= This will appear in the title bar of the answer box.

environment=stomp This is to allow the test tool to have different functionality when used with other systems.

mode= The option is **linked** or **unlinked**, for use within or outside microcosm (for normal STOMP use the state is linked)

test= The test type is defined as **formative** (self test) or **summative** (assessed test). This can be overridden by a command line declaration in the launcher (mcs) file. which includes 3F for three formative tests for a unit or 1S for one summative tests (see section 5.10 below).

question= The number of questions n in the test, is specified as an integer.

rubric= This is one of three rubrics displayed in the answer control box., should be a general instruction such as **Press the 'mark' button to check your answer** for self tests, or **Answer all the questions you can before pressing the 'submit' button** for assessed tests.

StyleRubric=text This rubric has a default form for each style, containing a suitable instruction for the style of question. The defaults can be overridden.

AlwaysOnTop= The options are false(default) or true. It can be very inconvenient at some screen resolutions for the answer box to always obscure the question window.

For each question:

[Question q] q is an integer. The numbering must be sequential

title= This will usually be the question number which provides the title for the answer box. (an alternative entry is possible here, but it should not be wider than about 3 characters).

style= n The integer n corresponds to the following styles of question and thus answer box layout:

- | | |
|------------------------------|------------------|
| 1 list (radio button) | 5 numeric |
| 2 list (check boxes) | 6 text |

3 rank ordering

9 random numeric

4 pair matching

Choice= $l m$ The first integer l defines the number of answer choices available for question styles **list** (1 & 2), **rank ordering** and **pair matching**.

The (space separated) second integer m is *only* used in the style **pair matching** The number l is the length of the first list (i.e. the number of boxes with lower case letters) and the number m will be the length of the second list (upper case letters).

Keywords= This is a space separated list of words which need to be the sources of generic links to useful background material for the question.. Generic links to material within the contribution should be made by the author. For generic links to material outside the contribution let the editor at Surrey know what is required.

Rubric= $text$ This is one of two rubrics that are specific to each question. This message should be specific to the subject or manner of the question, rather than the style being used.

Right answer

Right= $expression$ The *expression* specifies a correct response. The *expression* depends upon the style of a question as follows:

Style 1 for **radio button list**

expression = single letter **a..j**

e.g. right=**b** radio button **b** checked

'Other conditions' may use the **xor** function defined in style 2.

Style 2 for **list(check boxes)**:

expression = single letter **a..j**

expression = *function*(string of letters)

where *function* = **and, or, xor**

e.g. right=**b** box **b** only must be checked

right=**and(df)** both **d** and **f** boxes must be checked

right=**or(abj)** any combination of boxes **a, b** and **j** may be checked

right=**xor(jdf)** one only of boxes **d, f** and **j** may be checked.

Style 3 for **rank ordering** of choice n :

expression = n digits 1 to n

expression = *function*(strings of digits)

where *function* = **or**

e.g. right=**32415** the digits must be in the order specified

right=**or(12345 54321)** either of these two rank orderings will be accepted

The meaning of the first example above, is that item 3 in the list (i.e. **c**) should come first, so that 32415 = **cbdae**

Style 4 for **pair matching** of choice $m n$:

expression = n upper case letters **A B C** etc. or the character **_** signifying a blank box.

In **IfOther** expressions the character ***** is also allowed, and stands for any letter from the second list. This allows specific pairing cases to be

testing style guide

identified without repetition. The number of characters in each list must equal the number of boxes requested - i.e. the first *choice* parameter.

expression = *function*(strings of upper case letters)

where *function* = **or**

e.g. right=DCBA the letters must be in the order specified

right=or(BCDA DBAC ACBD)

any of these three orderings will be accepted

IfOther=or(**C* *C**) would correspond to C being paired with c or b, with any other combination of letters or blanks in the other positions.

Style 5 for **numeric** answers

This style the testing tool handles *numeric* answers by evaluating and comparing with single values or ranges of values. Numbers may be entered in conventional 'computer' formats, e.g. as integers (31), decimal fractions (3.142) or in exponent format (4.9e3).

expression = single value

expression = [minimum value..maximum value]

Style 6 for **text** answers

This style handles *text* answers (which may be alpha or numeric) by matching each character in the string.

expression = 'string'

expression = **or**('string1' 'string2')

The strings must be delimited by single quotes.

The strings may contain the character '*' which is treated as a 'wild card' that can stand for any single character.

Style 9 for **random numeric**

This style accepts numeric answers by evaluating them and comparing the value with expressions which can contain random variables. The variables can be defined by range or by an explicit set of values.

expression = a pascal style expression containing the variables within square brackets.

e.g. right = [V1] * sin([V2] + 1.57078)

AnswerSig=*minsig* [*maxsig*]

The number of significant figures used in a student's response can be specified precisely, or as an inclusive range.

Vn= *rangemin rangemax* [*sigfig* [*decimals*]]

or

Vn= (*value1* [, *value2* [, *value3* [. .]]]) [*sigfig* [*decimals*]]

Variables are defined as ranges or as a list of values. The list of values must appear within braces and be comma separated. The number of variables starts from 1 for each question, and the numbering must be consecutive. The number of significant figures to be displayed and the number of decimal digits may also be specified.

If the **Explain** box is checked in the test editor and keywords specified, the answer sheet will display an *explain* button:

Rightkey=keywords Single space separated *keywords* are entered here. If the user presses the explain button, these keywords are passed to MCM show-links..

IfRight=message The *message* entered here (simple text only accepted) is put on the screen if the answer is correct.

Wrong answer:

If the **Explain** box is checked in the test editor and keywords specified the answer message box will display an *explain* button .

Wrongkey=keywords Single space separated *keywords* are entered here. If the user presses the explain button these keywords are passed to MCM show-links.

IfWrong=message The *message* entered here (simple text only accepted) is put on the screen if the answer is wrong and does not match any of the specific wrong answers specified below.

Specific wrong answers

This allows specific wrong answers to be identified and given their own helpful message and explain button keywords.

Use the *ins* (insert) button to create a new special case and the *del* (delete) button to remove a case.. The cases are tested in the order shown, so the conditions can become successively more specific. The order can be changed by means of the *up* and *down* buttons.

Conditions, messages, and keywords can be edited by selecting and using the *edit* button and appear as the following entries in the qni file:

OtherCondition1=Specific wrong answer entered using syntax outlined above in the 'Right answer' section.for each style.

OtherMessage1=message as above

OtherKey1=keywords as above

For the random numeric style the significant figures are specified using

OtherSig1= with the specification as above

OtherCondition2=

etc.

Summative tests (assessed tests)

The test and question information entered using the test editor is split between the qni and ani file. The test and answer window control data goes into a qni file and the actual answer information into the ani file. The ani files will be removed at Surrey and distributed separately from STOMP. Marks can be awarded for correct answers. There is an option for alternative answers to attract marks if entered under the 'Wrong answer' or 'Specific wrong answers' conditions.

5.7 Help (physics).

There are five different ways in which physics theory help can be obtained when using the testing tool in **linked** mode. In unlinked mode only the first of these is available.

1. Question files are viewed using the rtf viewer and conventional Microcosm linking is available as normal.
2. The **Explain** button against each question in the **answer sheet** does a 'show links' on the **keywords** provided for that question.

3. The **Right answer** feedback window displays:
 - A simple text **message** entered using the test editor (qni file **IfRight=message**).
 - An **explain** button which passes on the **Right answer keywords** (qni file **Rightkey = keywords**) to Microcosm for 'show links' . The keywords must be sources of generic links to helpful material.
4. The **Wrong answer message** (qni file **IfWrong=message**) and **Explain** button with **Wrong answer keywords** (qni file **Wrongkey = keywords**) is treated the same way.as the **Right answer** in 3.
5. The **Specific wrong answer messages** (e.g. qni file **OtherMessage1=message**) and the **Explain** button with the **Specific wrong answer keywords** (qni file **OtherKey1,2 etc**) are also treated the same as the **Right answer** in 3..

Ideally, the question **Explain** button keywords should provide general background material for the question. The **Right answer (Rightkey=)** keywords should provide links to material that would be of further interest. The **Wrong answer (Wrongkey=)** keywords should provide links to other helpful information that is more specific than the question **Explain** button material. The **Specific wrong answer** (e.g. **Otherkey1, 2 etc**) keywords should provide links to specific material for the identified error. Clearly these four should link to different material. It is appreciated that this can be very difficult to achieve and in many cases sufficient material will just not exist to be able to make sufficient discrimination.

5.8 Handling responses

Formative mode

The two buttons available to the user are **Mark** for individual questions and **Mark all** for the test. When the **Mark all** button is clicked answers are checked to see a) if any have not even been attempted, and b) whether those that have been attempted are logically wrong (i.e. not finished or containing duplicate responses). In all cases the student is asked to choose between continuing anyway and going back and correcting/finishing the questions.

If in **formative** mode each question's response will be judged right or wrong, At least two messages are available for feedback, and other messages can be displayed for special answer conditions. As stated above, when a message is displayed the user can be invited to ask for further information using the Explain button (if provided), which passes the keywords to Microcosm for 'show Links'.

Summative mode

In **summative** mode only the **Submit** button is available. when this is clicked the student's responses are saved to a data-base and the testing tool terminates. The student is warned if any questions have not been attempted, and they are given the chance of going back.

5.9 Working in the Answer Sheet.

In the answer sheet the different question styles have the following facilities for entering the answers:

list (style 1 and 2) - radio buttons (for a single answer) or check boxes (for one or more answers)(

rank (style 3) - lower case letters must be dragged (mouse) into numbered boxes.

pair matching (style 4) - Pairs are selected from a set of lower case letters a, b, etc (in boxes) and upper case letters A, B, etc (in circles). Clicking on a box and then on a circle makes the pairing, which is shown by a line.

numeric and text (style 5 and 6) - a single text entry box is provided. Text entry is standard, and numeric entry is as described below for random numeric.

random numeric a single entry box, or an array of entry boxes can be provided. Numeric input can be entered as integers, or the two standard floating formats (123, 123.456, 12.34e4). If the format is not valid, then an error message is displayed and the user can enter the number again. The text box, or the left hand column (if an array of boxes is specified) can be preceded by 'a=', or 'x=', 'y=', etc. Numeric and text answers can only be typed in via the keyboard.

Assessed tests may only be marked once, and there is no immediate feedback to the students about whether questions were right or wrong.

5.10 Using the testing tool within Microcosm.

As far as Microcosm is concerned, the testing tool is just another programmed activity. This means that for each test an **.mcs** file is required with a command line that specifies the tool and the root name of a set of question documents with the number of documents and style of the test - **F** for formative or **S** for summative. Each question document requires two files, an **rtf** that is displayed as the question File and a **.qni** that is used by the testing tool as the control file. For assessed tests, the **.ani** file is only used by the marking tool. The same base filename must be used for these two or three files. The **.rtf** and **.qni** files for each test must be in the same directory as the **.mcs** file. The command line in the **.mcs** must specify the base filename without extension. e.g., in a situation where there is only one formative (self test) question file the command line entry in mm1_3ef.mcs would be:

```
[Microcosm Script]
Command=%system/stomp/stbin/%sttest.exe mm1_3ef $Filename$ 1F
```

The single question file and control file in this case is mm1_3ef0.rtf and mm1_3ef0.qni.

Where there is a set of assessed question files the command line might be

```
[Microcosm Script]
Command=%system/stomp/stbin/%sttest.exe wm1_5cf $Filename$ 5S
```

This example would mean that there were five sets of files wm1_5cf0.rtf, wm1_5cf0.qni, wm1_5cf0.ani, wm1_5cf1.rtf, wm1_5cf1.qni, wm1_5cf1.ani . . . wm1_5cf4.rtf, wm1_5cf4.qni, wm1_5cf4.ani, and that the test is summative so that the user would be presented with a randomly selected test from the five possibilities. In a formative test the user would be asked to select a file.

Note that the last specifier on the command line (e.g. 1F, 5S) specifies the style of the test - **Summative** or **Formative**, and this overrides any specification in the qni file (see section 5.6 above).

The **.mcs** file(s) must be imported into Microcosm in the normal way, as a Launcher document type.

The **.rtf** document (the question file) must be imported in to Microcosm as an ordinary rtf document. It is no longer necessary to link the Test Tool with the **.rtf** document by a Generic Link. The file is automatically displayed by the Test Tool

All question (**.rtf**) files and associated control (**.qni** and **.ani**) files must be put into your quiz directories..

If n is the number of question sheets in the test.

For each **formative** test there are $2*n+1$ files (*base.mcs, base0.rtf, base0.qni, base1.rtf, base1.qni, base2.rtf, base2.qni, etc.*)

For each **summative** test there are $3*n+1$ files (*base.mcs, base0.rtf, base0.qni, base0.ani, base1.rtf, base1.qni, base1.ani, base2.rtf, base2.qni, base2.ani, etc.*)

The **.ani** files will be removed during module integration and distributed separately.

The Launcher replaces \$FileName\$ with the full path file name of the **.mcs** file. The Test Tool uses this information to identify the full path names of the current directory and, hence, the control (**.qni** & **.ani**) and question (**.rtf**) files.

5.11 Filenames

All problem and test files go into the quiz directory for the contribution. They are named as shown in the following examples:

Problem file for unit 2.1 in Waves is **pw2_1.rtf**, Measure **pm2_1.rtf** and Optics **pop2_1.rtf**.

Formative (self test) question sheets for Waves unit 2.1 are:

sheet 1 is **wm2_1f0.rtf**, sheet 2 is **wm2_1f1.rtf**, sheet 3 is **wm2_1f2.rtf** etc

The associated launcher file is **wm2_1f.mcs**.

The command in this mcs file will be

```
command=%/system/stomp/stbin%sttest.exe wm2_1f $Filename$ 3F
```

The data files for these three test sheets are:

wm2_1f0.qni, **wm2_1f1.qni**, **wm2_1f2.qni**

Summative (assessed test) question sheets for this unit are:

sheet 1 is **wm2_1s0.rtf**, sheet 2 is **wm2_1s1.rtf**, sheet 3 is **wm2_1s2.rtf** etc

The associated launcher file is **wm2_1s.mcs**.

The command in this mcs file will be

```
command=%/system/stomp/stbin%sttest.exe wm2_1s $Filename$ 3S
```

The data files for these three test sheets are:

wm2_1s0.qni, **wm2_1s1.qni**, **wm2_1s2.qni**

wm2_1s0.ani, **wm2_1s1.ani**, **wm2_1s2.ani**

For tests in the Measure and Optics modules the **w** should be replaced with **m** or **op**.

The question files for assessed tests are encrypted for security, the decryption can only take place when the testing tool is used to look at the question files. For additional security the ani files (which contain the answers) are also encrypted, even though these files should not be placed on any machine available to students.

6 Creating a unit

6.1 New schemes for hiding documents

A new appendix (B) describes the proposed system for hidden documents, but the actual scheme is now included in the body of this guide.

6.2 New graph activity type

If you are going to use any graphs as activities, then you should create a new directory called **graphs** (alongside `mscript`, `simul`, etc.) for the `.grv` and `.mcs` files required.

6.3 Organising your unit material

6.3.1 Pictures, diagrams, videos and applications

Filenames should generally be based upon the code number for the activity in the storyboard. Exceptions are where one executable is used for several activities. In such cases the executable name might be descriptive (`raytrace.exe`) and the `.mcs` files (Microcosm scripts) for its multiple instances would follow the code

6.4 A check list for scripts before integration

- Are all icons inserted with the 'link to file' box checked?
- Are all glossary items sent to Belfast (i.e. at least every word that appears in bold)?
- Are all biographies names sent to OU?
- Are all Databook requirements sent to Manchester?
- Have you checked with Surrey about any items that might require copyright clearance?
- Is the storyboard spell checked?
- Has your unit been checked by your academic checker?
- Has your storyboard been checked by your editor? (It is helpful here to send storyboards via marie, but to put your activity's pages onto paper as well as sending the executables, or as agreed with the editor)
- Have you responded to the editor's comments and suggestions.
- Are self test and assessed test question files prepared?
- Are your problem questions prepared?

6.5 Before integrating your material you must:

- Convert storyboards to scripts (using StScript macros).
- Add copyright legends to third party pictures (rubric from Surrey)
- Have control files prepared for testing tool tests
- Have `mcs` files prepared for all applications and tests
- Have help files prepared for each application.

6.6 Integration of your material into a unit

It is important that documents are imported into the docuverse in the same way that they will finally be distributed. To do this, not only must the physical files be in the correct directories,

Creating a unit

but the logical types of the documents should be imported into the correct logical structure and the correct descriptors given, before any links are made.

To achieve this, you should use the registry file (mcsystem.reg) that is in the bin directory under the release name (e.g. third). It is not sufficient just to use the logical types under a different application name.

If you wish to keep your current mcm setup, then you will have to copy it all to a temporary mcmback directory (for example), and set up a new mcm directory with mcm/bin-docuvers-etc as required. The bin, docuvers and user directories can be copied directly from the backed up version - but only if these are from the most recent release. Otherwise, obtain the most recent versions from the server.

Having obtained the registry file, change the machine number (in the /System/Docuverse branch) to that of the machine you are using (using maintain/registry).

You must ensure that the following files are in place, and in the state indicated:

c:\mcm\bin\mcsystem.reg as described above
c:\mcm\docuvers\docuvers.dra this should be edited to be empty.
c:\mcm\measure\measlink.raw edited to one comment line only

or

c:\mcm\waves\wavelink.raw edited to one comment line only.

The existing files

c:\mcm\docuvers\docuvers.ddf and
c:\mcm\waves\wavelink.ddf or
c:\mcm\measure\measlink.ddf are required, and should be left as distributed..

The existing files

c:\mcm\docuvers\docuvers.nix and
c:\mcm\waves\wavelink.nix or
c:\mcm\measure\measlink.nix are best deleted.

The distributed registry file has been set up so that new links that you make are put into the wavelink.raw or measlink.raw files and not into the 'user' linkbases.

1) Import your new .rtf, .bmp, .mcs etc files into the **Files** section of the logical document structure. Use the **File/import document** menu command of the **select a document** box, but make sure that you have selected the appropriate logical document type for the file you are importing, before you import it.

Test question files, in-line question answer files and activity introduction documents should be put into the logical type **Files/Hidden**.

2) Using the **File/Edit Document Information** menu command to change the current descriptions to the correct ones for each document as you import them.

a. Main script descriptions should be in the form 'Unit 5.4: Regression models' (N.B. do not put the module descriptor letter before the unit number, the module is already defined in the logical tree structure).

b. Subsidiary scripts and all other application descriptions do not have unit numbers, except for equation and symbol files which are of the form 'Equations/Symbols used in

Creating a unit

unit M5.4' and for the test files which should be of the form 'Self test for unit W6.2' and 'Assessed test for unit W6.2'.

- c. Descriptions should not contain words like 'video', 'activity', etc.
- 3) Copy all the documents (except **Hidden** files) into the **contents** logical document structure by dragging the descriptor of each document and dropping it into the desired logical document type with the control key held down. Main scripts go into the block type, all others go into the unit type
- 4) Insert your links for the unit:
 - a. main unit → simulations
 - b. simulations → introductions
 - c. Unit name (e.g. W6.3) generic link → main unit.
 - d. subject specific and position specific generic links into the main unit
 - e. main unit → self and assessed tests
 - f. main unit → other documents (e.g. text / audio / video / pictures)
 - g. back/next links if you are developing some consecutive units.
 - h. specific zero-selection links from script to equations and symbols files
 - i. virtual links - (see appendix A).
- N.B. Links from introduction documents to applications should be removed.
- 5) Position the windows of the RTF files (and of applications if the position data is in the mcs file).
- 6) Using Maintain, export the file sizes from the branch of the registry Users/testing/wavemeasure/RTFViewer into a file **sizes.ini**.
- 7) Open every picture and ensure that the window is opened to the full size of the picture (including credit) before closing.
- 8) Using Maintain, save this picture startup information from the Users/testing/wavemeasure/settings/BmpView branch of the registry into **bitmaps.ini**

You should now test the unit(s) to make sure that everything functions correctly, and that text files and applications open where you want them

Before you zip up your source documents, make sure that only the files actually required are in the directories stompuni/.. .To actually zip up your source documents, from the c:\mcm\wavesmeasure directory use the DOS command

```
pkzip -rp zipfile stompuni\*.*
```

Please use the lower case p, since using the upper case P leaves us no flexibility in how we unzip!

Add the other necessary files to the unit zip file

| | | |
|--------------|---------------|-----------------------|
| The docuvers | pkzip zipfile | docuvers\docuvers.dra |
| The linkbase | pkzip zipfile | measure\measlink.raw |
| or | pkzip zipfile | waves\wavelink.raw |

Creating a unit

File sizes and positions `pkzip zipfile bin\sizes.ini`

Bitmap sizes `pkzip zipfile bin\bitmaps.ini`

This zip file should then be placed on the server in the appropriate version/uni/mcm directory.

Storyboards should preferably be zipped into a different zip file, and placed into the uni/master directory.

6.7 Note:

There are many equivalent ways of producing the above set of zip files. Some methods are not so equivalent as others, so unless you **know** that another method is equivalent, please stick to the one above. Differences that are known to be **not** equivalent include: using different logical types whilst links are being made, putting in descriptors after links are made and importing into the logical structure in a different order.

7. stscript.dot macros

Always keep your original storyboard file.

The Word templates used by SToMP authors *story.dot* and *stscript.dot* are on the server under system\template, these should be placed on the local hard disc in the c:\winword directory.

7.1 Recommended usage

1. Load the storyboard.
2. *Save-As* the name you want the new script to be.
3. use File/Template to attach to the template **stscript**

In all of the following macros, choose **No** if offered to continue searching from the start of the document.

Then either:

4. use Tools/Macro, select the macro **stRunAll** and then the **run** button,
or
4. use Tools/Macro, select the macro **st1format** and then the **run** button.
5. use Tools/Macro, select the macro **st2comment** and then the **run** button.
6. use Tools/Macro, select the macro **st3activity** and then the **run** button.
7. use Tools/Macro, select the macro **st4symbol** and then the **run** button.

If any frames remain that look like **activity** boxes, then click on them and see if they claim to be **activity** style. If they do, then start again by closing the file without saving and then loading it in again. This time, run the macro **stCleanActs** after **st1format** but before the others.

If this does not do the trick, then start again, but this time after running **st1format** go through the document selecting each offending ex-activity box and force it to the **activity** style by selecting the **activity** style in the ribbon style-box. (If it claims to be **activity** then it really is fibbing, since after running **st1format** the style is no longer in a box!) Then carry on with **st2comment**, etc.

There are two other macros of this style, **stCleanNormal** and **stDelHidden** that may be run, and the best time is between **st1format** and **st2comment**.

stCleanNormal will clean up the Normal style in the same way that **stCleanActs** cleans up the activity styles. i.e. it is possible to have paragraphs that claim to be *style* without necessarily having all the attributes of *style*. These two macros force them to conform.

stDelHidden deletes hidden text that you may have in your storyboard. Such text will not be needed in the script, and its presence will slow down the opening of the rtf viewer.

The **stRunAll** macro will run all the above macros, which takes rather longer than you probably need, but it might be convenient for some. It is not certain to deal with all situations where the original **story** and **story2** styles have not been kept to.

Creating a unit

Before running the macros, you must ensure that in each activity box the activity code occurs in the first paragraph, and preferably that is the only item in the first paragraph. If more than one code is used then they should all be in the first paragraph, or in separate activity boxes.

One further macro has been developed to help with situations where more than one script file is contained within one storyboard file. The macro is called **stSplitFile** and it should be the last macro run because it actually writes the rtf files. In order for this macro to 'know' where to split the document into separate files, you should separate into separate sections what you want to be written to separate files. This is quickly and easily done in the storyboard by inserting section breaks which may then be left permanently in your storyboards. Any form of section break will work, but it must be inserted just before the first character required in each file and this should be the title of the document except as described below.

Filenames can be specified, but by default they are written to the predefined file names, script1.rtf, script2.rtf etc. and they are written to the currently selected directory. The current directory may be changed by the directive "Directory" which must be the first word of the storyboard proper following the title page break. This must be followed by one space and the required directory e.g.

Directory c:\mcm\measure\stompou\mscript
Filename m1_1.rtf

The "Filename" must similarly be followed by one space and the required filename (with .rtf extension).

It is currently recommended that the macro stDelHidden is run after using stRunAll.

7.2 stlformat

1. sets margins to 2.5cm.
2. style 'normal' set to : "Times New Roman", 12 Point, Justified, 0.5 spacing before paragraph.
3. style 'activity' temporarily set to Normal + 10 point, 0.5 cm left and right indent. (i.e. it is moved from the box into the main body of the text, but is still identifiable visually and by style.)
4. style 'heading 1' set to Normal + 13 Point + bold, centred, keep-with-next, 1 line space before.
5. style 'heading 2' set to heading1 +12 Point + Dark red + left justified.
6. style 'heading 3' set to Normal +bold + left justified, keep with next.
7. style 'Normal indent' set to Normal + 0.5cm indent.
8. style 'greyback' set to Normal + light grey background
9. style 'prereq' set to normal + 1.27cm indent + -0.63 first line indent
10. Search for underlined text and remove the underlining.
11. delete the first storyboard page.
 - check first word of document is 'SToMP' - delete the line.
 - check next word is 'Storyboard' - delete the line.
 - delete the next 6 lines
 - check next word is 'Date' - delete the line or output an error message.

Creating a unit

N.B. if the layout of the title page has been altered then this part of the macro will fail - in this case just ensure that the whole of the title page is deleted by some other means, and continue.

7.3 st2comment

This deletes any text in 'comment' style, then deletes the style 'comment' and then removes all empty paragraphs.

7.4 st3activity

This converts the first paragraph of each set of adjacent 'activity' paragraphs to hidden text (normal style, but 10 point), and deletes subsequent adjacent paragraphs. Hidden text is displayed during the execution of the macro and then hidden at the end.

7.5 st4symbol

This finds each 'SYMBOL' field and checks if the font used is 'symbol'. If so the field is replaced by the equivalent character from the 'symbol' font but at 13 point.

7.6 stCleanActs

cleans up any ambiguous activity paragraphs, which can otherwise lead to macro failure.

7.7 stCleanEquns

cleans up equations, and forces them into the current Mathtype formats and sizes. This is useful if you have changed Mathtype setting during script preparation, and wish to check that all equations now conform.

7.8 stCleanNormal

This macro cleans up any ambiguous normal text, forcing it into the normal style. It is occasionally useful in documents where the normal style has been (accidentally) changed.

7.9 stDelHidden

This searches for all hidden text and when found it is deleted. This is to enable you to keep hidden text in your original storyboards if you want, without it getting through to final scripts where it would just slow down the rtf viewer.

7.10 stRunAll

This runs all the above macros (except stCleanEquns) in a suitable order to correctly create the desired script files in most cases.

7.11 stSplitFile

This macro splits up a file containing multiple script files as described above.

7.12 General points.

The storyboard should always remain the definitive version of a unit - any changes should be made to this and the conversion process repeated.

Creating a unit

Any storyboard that contains more than one unit should be split into the individual units by section breaks. The title pages are not necessary - the **stlformat** macro will not delete a first page that is not a title page.

The macros are not very fast - but only chronically so if the individual scripts are long.

Appendix A

Virtual links:

If you wish to have links from your documents to others in another site's material, then set up 'virtual' links as follows:

In your stompxxx directory create a file called "virtlink.txt". This file is to consist of one line for each virtual link, and each line should contain

- a) the link description
- b) the destination document filename
- c) the destination document description.

If b) and c) are not known then include sufficient information for the file to be uniquely identified.

Import **virtlink.txt** into your docuvers into the highest level logical type unique to your contribution. Then make the required links to this virtlink.txt. As each link is made, ensure that the link description (in the complete link box) is changed to match that in virtlink.txt.

Zip the file virtlink.txt up with the rest of your files when shipping to Surrey.

Appendix B

Hidden documents:

Introduction

It is proposed that the following types of document should not be visible to student users via the 'select a document box':

- The question sheets of self tests and assessed tests.
- Answers to in-line questions
- Introductory documents for activities

Microcosm Implementation

When hidden branches have been implemented (hopefully in the next MCM release, which is soon) they will operate as follows

1. Hidden branches will be hidden to normal users but seen by authors. Use Maintain to make a user an author.
2. Hidden branches in the hierarchy of Logical Types will be created by authors. All descendent branches will also be hidden.
3. Hidden branches can be anywhere in the hierarchy.
4. The hidden branches will be created from the Select a Document window. When a new branch in the Logical Hierarchy is created, the user (the author) will have the option to create it a Hidden.
5. Documents can be moved to a hidden branch by drag/drop in the SaD window.

Suggested Hierarchy

Documents to be hidden from users should be put into a branch called Hidden, under the Files branch of the appropriate application. This branch can be created using the maintain utility.

I propose that we introduce a 'hidden' branch in each application, below the Files branch. For instance, in the Waves part of the hierarchy, the hidden branch will be placed as shown below.

