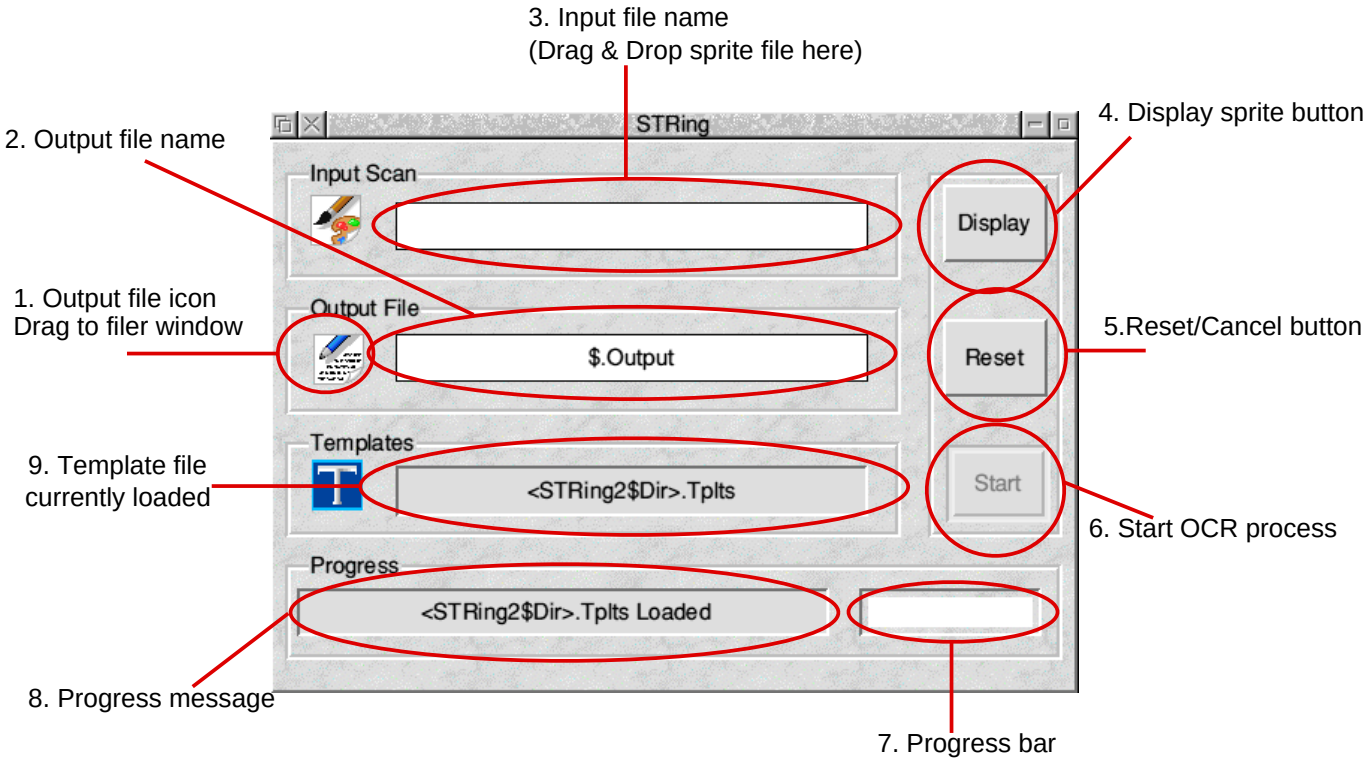
Program:	**!STRing**
Purpose:	**Scanned Text Recognition**
Version:	**2.2.4 / January 2011**
Tested on:	**RPC/SA233MHz/RISC OS 4.39**
Author:	**M. R. Beerling**
Date:	**4/1/11**



Summary

Scanned Text Recognition. Program *!STRing* loads an image of "scanned" text as a sprite file templates, based on Acorns Trinity, Homerton & Corpus fonts, are used to match the characters in the sprite. The output is a text file of the text in the scanned sprite.

Inputs: One "scan" sprite file. B&W only, per Drag & Drop or Obey file.
Outputs: Text file of text in scanned image.

The Author can be contacted by EMail on **beerling@online.de,** please use 'STRing' in subject line to get through the filters.

Licence

Contents

1. Introduction

A picture may paint a thousand words but text in picture form is difficult to use, especially for a computer. A picture of text from a book or newspaper cannot be edited in a word processor or dropped into a spreadsheet, it must first be converted to text. Hence the need for Scanned Text Recognition which is also known as OCR - Optical Character Recognition. Program *!STRing* allows the conversion of a picture of the text to actual text characters which can then be manipulated by a word processor.

Pictures of text may come from a fax, camera or scanner, here they will be referred to as "scans" because a scanner is the most common source.

Written in C/C++ this version has been compiled using GCC V4.1.1 Release2 on a RPC/SA233MHz/ROS 4.39. The program was linked with the OSLib library version 7.

2. Using the program

!STRing can be run in two ways:

(i) By double clicking on its filer icon.

An icon will appear on the icon bar and a window will open. To start the program a scanned sprite file is dragged & dropped onto either the baricon or the window. An output text file name must be given in the windows output file writable icon. To help, the text file icon to the left of the writable icon can be dragged & dropped onto a filer window where the file will be saved.

The input sprite can be displayed in its own window by clicking on the **Display** button. The displayed sprite window has a control pain which is used to change the displayed sprite or to zoom in or out.

The main window also has a **Start** button, a **Reset** button, a status line & percent progress bar. When the **Start** button is clicked the program starts to match the sprites. The status line displays the current task & the bar shows how far the task has been completed.

Cache data is built up in the program from scans already matched. The cache data is used to match subsequent scans and allows faster and more accurate matching.

Clicking on the **Reset** button stops the matching process and resets the cached data.

A template file (file type &071) can be dragged & dropped on the main window. The templates will be loaded and used for the next scan to be matched. These templates replace those which were previously loaded and the cache data is reset.

The text output will be saved in the file named in the output file writable icon.

The iconbar icon & the main window have a menu option for 'Settings'. These are two flags to disable algorithms which correct the text after matching. For most users they can be left alone but they are explained further under Hints & Tips.

(ii) Batch mode from an Obey file or task window. The abbreviation 'STR' has been defined for this purpose.

Syntax:

```
STR -S<Sprite file> -O<Output file> -T<Tplts file> -M<Messages file> -F<flags>
```

Default values are:

-O	STRingOp
-T	<STRing2$Dir>.Tplts
-M	<STRing2$Dir>.XMessages
-F	0x00

E.g.:

```
DIR <Obey$Dir>
STR  -SScan1 -OText
```

The flags option -F requires a hex number in the form 0x03. Bit 0 set means 'Disable letter checks' and bit 1 set means 'disable number checks'. See Hits & Tips below.

3. Hints & Tips

Memory use:
Whilst idle on the iconbar *!STRing* requires 640k. A task is started which takes 640k and then loads a template file of around 760k along with the sprite. Working memory required is about 40% of the size of the sprite.

So, approximately: max. sprite size = (total memory - (760k+640k+640k))/1.4

Scan Resolution:
For good matching results scans should not be less than 300dots per inch (dpi). More than 600dpi may result in large characters being mistaken for pictures and being filtered out. Photocopies and faxes do not match well because the effective resolution is too low.

Technique:
Use *!Paint* to export text areas of a scan to a new sprite file (don't forget to give each sprite a unique name). Multiple sprites in a file can be handled (see below). Save the resulting sprite file and drag & drop it onto the *!STRing* iconbar icon. The resulting text output must be saved as a file (RAM transfers not supported). Correct the output text by dropping it into a word processor and spell checking it.
For dark texts (e.g. Newspapers) try increasing the scan brilliance to help distinguish the individual letters. This will increase match accuracy.

Multiple Scans:
Several sprites can be placed in a sprite file. *!STRing* will work through each of the sprites in the order found in the sprite file. Each sprite's text output is written to the single text file but will be separated by a form feed character. This is faster than processing the sprites individually but requires more memory since all the sprites are loaded at once.
Multi column text can be handled using *!Paints* copy & export function (the camera icon). Again, don't forget to give each sprite a unique name. Each column is exported to a new sprite file and that sprite file is then saved and dropped onto the *!STRing* iconbar icon.

Pictures, boxes and noise:
Filtering for noise, pictures and boxes is performed by *!STRing* but for best results these should be avoided. *!Paint* or another pixel editor can be used to extract the text area from the scan.

Settings Flag 'Disable Letter Checks'
In some fonts, e.g. 'Homerton' (Helvetica), it is impossible to distinguish a lower case 'l' from a capital 'I'. Normally *!STRing* checks for the context of 'l' or 'I' and corrects accordingly. For example an 'I' can only occur with other capitals or at the start of a word when the next character is not a vowel. Otherwise it must be an 'l'. This algorithm works well for English, German & French but fails for languages like Welsh where many words start with 'll'. In this case the algorithm can be disabled by ticking the box. The status of this flag is stored when the program quits.

Settings Flag 'Disable Numeric Checks'
In some fonts it is impossible or difficult to distinguish between the letters 'O', 'o' and 'l' and the numbers '0' and '1'. Normally *!STRing* checks for the context of '1' or '0' and 'O', 'o' or 'l' and corrects accordingly. For example an '0' is assumed to occur with other numbers where as a 'O' will be found with other letters. This algorithm works well except when letters & numbers are often mixed (e.g. in car number plates). In this case the algorithm can be disabled by ticking the box. The status of this flag is stored when the program quits.

4. Error messages

File not found	A file required by *!STRing* has been deleted or renamed.
You cannot load a directory	A directory was dragged & dropped on the *!STRing* baricon.
File cannot be loaded	For some reason the file could not be opened & loaded. Memory?
Memory could not be allocated	Not enough memory for the sprite
No sprites loaded	No Sprites in the sprite file.
No sprite found	Program error - shouldn't occur.
Sprite not 2 colour mode	Sprites must be a two colour mode - see Limitations.
File name is too long	The file name including file path is too long. Copy file to a higher directory and try again from there.
File is not a sprite file	Only sprite files can be loaded.
Input Parameters Wrong	The parameters given in an Obey file or task window were incorrect.
Can't tell sprite background	All four corner pixels must be the same colour (background) colour.
Program Error	Something unexpected has happened.

5. Limitations

A sprite file which is to be used as a scan must contain only two colour sprites which are modes 0, 3, 4, 6, 18, 23, 25, 29, 33, 37, 41 or 44. The pixels in the four corners of the sprite must be of background colour. The sprite name and the presence of a mask are not important (although less memory is used without a mask). Scans must have an effective resolution of at least 300dpi. Faxes and photocopies do not produce good results because they have a lower effective resolution, even when scanned in at a higher resolution. Scans made at more than 600dpi may also give poor results because of filters used to remove pictures and limitations on the size of objects that can be matched (253 pixels in x or y directions).

For good matching the image of text contained in the scan sprite must have the following characteristics:

- Original image must be clean & of a good print quality.
- Text must not be deformed.
- Lines of text must be straight and not curved.
- Text characters must be upright.
- Text characters must be separate from one another.
- Text characters must be whole and not broken.
- At least a 1 pixel border of background colour should exist around the text.

The template file has character templates for Trinity, Homerton & Corpus fonts including bold and italic as well as for German & French characters and common ligatures. Other non standard characters are not present in the template file.

6. Files Released

The files that make up this release are as follows.

```
!Boot        WR/     Obey      16:06:36 18-Apr-2004  107  bytes
!Help        WR/     Text      19:34:07 06-May-2009 1418  bytes
!Run         WR/     Obey      18:41:29 01-Jan-2011 1518  bytes
!RunImage    WR/R    ELF       11:46:32 02-Jan-2011  270 Kbytes
!Sprites     WR/     Sprite    11:01:42 03-Jan-2011 1756  bytes
!Sprites22   WR/     Sprite    16:40:26 04-Jan-2011    4 Kbytes
Messages     WR/     Text      11:38:17 02-Jan-2011  410  bytes
Res          WR/     Resource  10:45:10 04-Jan-2011    4 Kbytes
Settings     WR/     Text      10:44:15 04-Jan-2011   26  bytes
STRing       WR/R    ELF       15:54:27 27-Dec-2010  359 Kbytes
STRLine      WR/R    ELF       16:05:51 24-Dec-2010  359 Kbytes
Tplts        WR/     &071      09:33:34 05-Oct-2004  743 Kbytes
XMessages    WR/     Text      14:02:28 27-Dec-2010  771  bytes
```

7. Acknowledgements

ArcSite for the web space
http://www.arcsite.de

The good work done by the GCC and OSLib teams.

Thanks to Bernard Veasey for !Sprites22