Form Data Postprocessor Application Specification

The Form Data Postprocessor Application (FDPA) will be designed as a constant-presence Windows application. It will scan a given target directory for OCR for Forms output files and process them as they appear. It will need to be multithreaded so that multiple jobs can be handled at once and so that the control console is kept free.

The FDPA will take in raw OCR for Forms data files (in a format which has yet to be specified) and process them based on form family and form type, using predefined macro scripts written in the form postprocessing language (specified in the Form Postpocessing Language Specification document). It will output result files to a specified directory in the format specified in the Form Postpocessing Language Specification document.

Interface Requirements

The FDPA interface should be implemented as a dialog-based control. It will be available in the system tray and should be intended for independent startup on machine initialization.
Necessary controls are:

- Start/Stop service
  - (Suspends/Resumes scanning of source directory – all in process threads are allowed to finish)
- Specify Source file directory
  - (Specifies where OCR for Forms output files will reside)
  - Recursive Directories?
    - (Check box allowing subdirectories to be scanned)
- Specify Target file Directory (for each type of form for which there is a defined macro)
  - (Determines a location to write .DAT files)
- Specify Macro location for a given form
  - (Determines where to look for the macro file for a given form type)
- Kill current threads and continue
  - (Terminates all in-process threads and continues scanning)
- Test Parse Macro File
  - (Runs a macro file through the parser to assure that syntax is correct)
- Reprocess Form Data File
  - (Processes an OCR for Forms data file immediately as if it was just added to the scanning directory)

It will be useful to use a preferences file to specify the macro file name, destination directory, and form type pairings. This file should be plaintext so that it is modifiable outside of the dialog (so that it is not necessary to go through the dialog control to add every form type when adding a large number of forms at once).

Processing Model

The specification for the engine is fairly open. For each form type which needs to be processed there will be a defined macro file which will handle capturing the necessary data from the form. This macro file will always have at least the Main macro – this is the entry point of the processing routine. A parser will be used to directly process the macro file and manipulate the data. The parser/processor will be spun off by the main process as an independent thread; successful termination will be signaled by the creation of a processed output .DAT file in the specified output file directory.

The data specification is as follows: Each form page will appear on a single line of the file. Every block of data (a block is a logical separation of data on the form and can generally be recognized by a change in the number of columns in a dataset) will begin with a header specifying the name of the block, followed by the number of columns in the block. This header will be prefixed and terminated by the special string

**$

and will be "|" delimited.

Following the header information the data from individual cells in row/column sets will be stored and will be "|" δελιμιτεδ. Data will be stored row-wise across a matrix; so given a matrix that appears as:

```
AAAAAA      BBBBBB      CCCCCC
DDDDDD      EEEEEE      FFFFFF
GGGGGG      HHHHHH      IIIIII
```

the file will store the information as:

```
AAAAAA|BBBBBB|CCCCCC|DDDDDD|EEEEEE|FFFFFF|GGGGGG|HHHHHH|IIIIII
```

Thus an entire block of data will appear as:

```
|**$|BTHeader1|3|**$|AAAAAA|BBBBBB|CCCCCC|DDDDDD|EEEEEE|FFFFFF|GGGGGG|HHHHHH|IIIIII
```

The end of a page will be determined, as stated previously, by a CR/LF. All blocks terminate at the end of a page. Blocks are also terminated when a new header is recognized.