log2ps

Technical Manual

Ottar Kvindesland, LA9IHA

This document is intended for those of you who enjoys tinkering with software. It's purpose is to provide an easier access to the code of *log2ps*. The program is under contious development, and contributions are very welcome, both as ideas and new code.

The reader is assumed having a good understanding and practice in C++ programming. Also, the detailed specification is the code itself.

The author is not an experienced C-programmer, and this project has also been a learning exercise for the use of pointers. I have tried to avoid pointers, but it seems to be hard to avoid them.

Program Overview

The program is written in Borland Turbo C++ 3.0. It performs the following sequence for production of cards:

- Read the configuration file
- Create PostScript header
- Read a line from the log output
- Add a QSL card to the PostScript tail
- Continue until log is completely read
- Write configurations back to file.

It is composed of the following CPP and H files:

`CFG.CPP`	Maintain the configuration paramters and support of such.
`L2P_COM.CPP`	Functions for converting log file to QSL cards in the PostScript tail.
`LOG2PS.CPP`	Contains main function.

LOG-GEN.CPP	Reads the file generated by your log program and pass on accepted entries to QSL card-producing functions.
MAN.CPP	Prints a short manual and presents legal switches on screen.
O_FILES.CPP	Functions regarding file handeling
O_STR.CPP	Functions regarding string handeling
PS.CPP	Functions to create PostScript header in output file.
ICONS.CPP	Functions to create place EPS icons on the card.
STRUCT.H	Structures and datastructures
L2P_GLOB.H	External definitions of configuration parameters.

Important data Structures and Variables

Structures are defined in `struct.h`. Global variables are defined in `l2p_glob.h`.

tlpos
Structure holds first and last position on substrings in source file from log. This is station worked, time, date, frequency, mode and report.

The positions are held as integers and a char value describes type of definitions. Legal definitions are A, G and E.

cfgElType
The configuration parameter has a character string for data. The type describes where data should be stored, i.e. int, float or boolean. Min and Max values are used for validation purposes.

cfgType
A structure holding all parameters used in log2ps.

Hence the vital global data are:

```
tlogtbl c_qp;
cfgType cfg;
```

Project File

```
-[_]---------------------- Project: LOG2PS ---------------------1-[•
  File name     Location                        Lines    Code    Data
  LOG-GEN.CPP   .                                 327    1552     728
  LOG2PS.CPP    .                                  75     128    7247
  L2P_COM.CPP   .                                 315    2262    1121
```

```
MAN.CPP       .                394    2429    9812
PS.CPP        .                699    3943   14843
O_STR.CPP     .                536    1408       7
CFG.CPP       .               1257    9943    3263
O_FILES.CPP   .                 24      56       2
ICONS.CPP     .                 91     504     458
```

Execution Path of *log2ps*

The main program is found in the file `log2ps.cpp`. It starts with reading the config file. If arguments are passed on the command line theese are detected by the function `getCfg`. This will in this case return false and the program will place all configuration data onto the config file.

When the program is executed without command line parameters the program will follow the normal card producing sequence, and filally fill the last sheets with empty cards if needed.

Reading Configuration
See `cfg.cpp`, function `getCfg`.
First the config variable `cfg` is initially, then the config file is read and data is placed into `cfg`. Any switches read as command line parameters are then read and checked in the function `redefCfg`.

Printing Header
See `ps.cpp`, function `PS_print_header`.
First the output file is defined. Then the header is composed in `PS_print_header` where the global variable `cfg` is reffered to when necessary to select which parts of the standard header that has to be adopted, or may be excluded.

Reading Log
See `log-gen.cpp`, function `other_readLog`.
First the input file is defined and the existence is established. One line is then read from the log at a time. The elements station worked, year, day, hour, minute and frequency are checked and validated. If succsessful the line is forwarded to function `printCard` in `l2p_com.cpp`, where the appropriate substrings are selected, and the PostScript statement is created in the function `makeQSL`. Then next log line is fetched and so on.

PostScript Code

The reader is reffered to my contribution to Elektor Software 96-97 where it is described in greater detail as QSL.PS.

Post Script is a stack oriented language. Readers familiar with other similar languages like Lisp and Prolog should have an advantage. If the language is unfamiliar, relevant documentation by Adobe inc should be studied.

The script is composed of a header and a tail. The header defines the layout of the card, and how it is placed on the sheet. The tail is set up of calls to this header with parameters as elements from the QSO. This is station worked, time, date, rpt etc.

The header performs the following tasks in sequence:

- Initial definitions set out fonts and papersize and directions. For fine details please refer to literature 2).
- The ISO vector allows for Norwegian letters. Literature 2) gives the details.
- Various relationships and definitions contain simple procedures which assign relationships between instants on the QSL card, e.g coordinates, strings, sizes of boxes, shades of the logo etc.
- PosLittleBox.... finds the properties of the little boxes containing the particulares of the QSO.
- posNqslAZ gives the coordinates of the QSL card. The code describes number of cards on a sheet, and the position of the cards on the sheet.
- Boxes and corners. You can mark the corners of the QSL card, or you can draw a box around. it. The small boxes containing details of the QSO is also drawn in this section.
- Information section. Here all fixed strings like 'To Radio', 'OPR' etc is written.
- Font section. Declares the various fonts. For simplicity only standard fonts have been used. When a font definition ending with '_n' is chosen the ISO-vector will be implemented.
- QSO data section. Data from the QSO is present on stack. A box is drawn and title and date is inserted into it. The field To Radio gets the signature of the station worked.
- Print standard. Calls the procedures needed to print an empty standard QSL card.
- Print local. Places operators own particulars on the QSL card.
- Extra information. Procedures to print special event banner on the right hand side of the QSL card, QTH for portable operation, and a logo for NRRL, the Norwegian Amateur Radio Society.
- Print cards. With all particulares from the QSO on the stack the procedures required for the QSL card will be invoked. Print1Card will cause the a sheet of QSL cards to be printed out on the postscript printer.

log2ps will, based on configuration data tailor the postscript to the QSL card defined. Hence some procedures may only be defined with certain configurations set.

Further developments

As with all software there are ideas to implement. I have so far thought of a few.

- JAVA, GUI config setting. The setting of configurations are well suited for a GUI interface. Java beeing the coming thing, and write once - run everywhere may be a good alternative to Borland C++ 3.0 for DOS. Another advantage for free is the lack of pointers.

- More Efficient and Intelligent PostScript Code. PostScript beeing such a powerful tool lends it self to be more used. E.g. more dynamic card production, so that say only your 40m GP will appear on your card if you are QSL'ing a 7MHz QSO. You could also have greetings in the language of the receiving station and many other not that useful, but fun things on the card.

Literature

1) *Postscript Language, Tutorial and Cookbook* by Adobe Systems Incorporated (the blue book) is a natural place to start the study of Postscript.

2) *Postscript Language Reference Manual* by Adobe Systems Incorporated (the red book)gives all the details of the language ignored by the book above. An understanding of Postscript is essential before buying the Reference Manual.

3) *Encapsulated PostScript File Format Specification, version 3.0* by Adobe Systems Incorporated describes the EPS format, and how to include them into a PostScript document. Available at `http://www.adobe.com`.

ERROR: undefined
OFFENDING COMMAND:

STACK: