# Why Patent Searchers (And Others) Need KEDIT When They Already Have A Word Processor; Or, Post-Processing At The Power Level

by Sandra Unger, Ph.D.
Staff Chemist
Exxon Research and Engineering

I have frequently been asked why I use KEDIT to edit downloaded patent abstracts instead of the powerful new word processors. Editors, such as KEDIT, and word processors are different types of tools that were developed to meet different sets of requirements. Word processors excel at creating well-formatted documents. KEDIT more or less ignores the format of the text, and instead provides powerful tools for manipulating the text itself. To demonstrate the power of KEDIT as well as the limitations of word processors, a macro will be examined that modifies downloaded records to meet the requirements of certain personal databases. The sample macro will delete the type of extra information that ORBIT has recently added to Derwent patent equivalents. The requirement is to delete selectively, an *unknown string of text* from an *unknown number of patent equivalents* in an *unknown number of patent records.* It would be extremely difficult simply to record a word processor keystroke macro that could deal with this level of uncertainty. I will show that the combination of KEDIT and a modern word processor is much more powerful than using either tool alone.

## PROGRAM DIFFERENCES

Editors and word processors are different types of tools that were developed to meet different sets of requirements. Editors, such as KEDIT, were originally developed to meet the needs of computer programmers in writing source code. Word processors were developed to meet the needs of writers and secretaries in producing well-formatted documents. As these tools developed, they became more and more sophisticated and now often include a macro language that allows repetitious tasks to be automated. KEDIT excels at tasks that involve the modification of variable length blocks of text based on position, or other sophisticated methods of recognition. Word processors excel at tasks that involve the formatting of existing text into well-formed documents.

The most obvious difference between editors and word processors is the type of files they manipulate. Editors generally operate on ASCII files. This is again due to the historical use of editors by programmers. The programmers plan to send their source code to a compiler and do not want any formatting getting between them

KEDIT excels at tasks that involve the modification of variable length blocks of text based on position, or other sophisticated methods of recognition.

and the compiler. A word processor, on the other hand, contains enormous amounts of formatting stored in each and every document. For example, a simple ASCII file containing the word "hello" may occupy 7 bytes. A corresponding word processor document containing the word "hello" may be nearly 2,000 bytes. Most of the size difference is due to the formatting that the word processor stores to make the document print in the desired font and with the desired headings and using the desired template.

There are many other more subtle differences. To illustrate some differences between an editor and a word processor, I will consider the problem of deleting unwanted text from the patent number field of a Derwent patent record. For example, ORBIT recently began adding the number of pages, the language, and the patent classification to each equivalent in a Derwent patent family (Figure 1). While this type of additional data may prove valuable for certain types of evaluations, it is undesirable when building a personal database. Manually deleting this extra information from a hundred records could be an extremely tedious process. Automating this task in a word processor would prove to be difficult. However in an editor such as KEDIT, it is easy to write a macro that identifies the unwanted text and then deletes it.

This task is difficult to automate for several reasons. First, the text to be deleted is not defined by a simple string of letters. Secondly, the text to be deleted exists on several different lines. Thirdly, and perhaps most importantly, your macro must not mistakenly delete any information in any other field. Here the task is to delete selectively, an *unknown string of text* from an *unknown number of equivalents* in an *unknown number of patent records.* That is a lot of uncertainty to deal with. The only selection criterion is the position of the text within each patent record. If you are accustomed to automating procedures by recording keystrokes in a word processor, this might seem an impossible task. However, by using the KEXX macro language that comes with KEDIT I was able to automate this procedure easily.

Here the task is to delete selectively, an *unknown string of text* from an *unknown number of equivalents* in an *unknown number of patent records*. That is a lot of uncertainty to deal with.

To design this macro, first consider the sequence of steps that would be needed to delete *manually* the added information from a *single* patent record. First, you would need to identify those lines of text that constitute the patent number field. Next, you would need to identify the block of text that constitutes the extra information that has been added to each patent equivalent. Finally, you would need to move your cursor to the correct column and line and delete all the text to the right of this position. This last step would need to be repeated for each equivalent containing this added information.

This is a much more sophisticated task than simply changing misspelled words. Here, we do not know what text will be added to each patent. Any number of pages, language or patent classification could be added to each equivalent. The text we want to delete is not defined by a string of characters. Instead, the text we want to delete is *defined by its position within a record.*

To automate this procedure, the software tool must be able to identify the patent number field first by identifying the "PN -" field-tag and secondly, by recognizing that the patent number field extends across many lines of text until a new field-tag is reached. For each line of text in this field, any text in columns 27 to 80 should be deleted. We need a software tool that can:

**FIGURE 1**
**Derwent Patent Record**

```
                                           patent class||
                                                       ||
                                  language||           ||
                                          ||           ||
                             pages||      ||           ||
                                  ||      ||           ||
( -- other fields --)             \/      \/           \/
PN  - US5084197-A     92.01.28 (9207)
      EP-482759-A     92.04.29 (9218) 21p  E
      NO9103691-A     92.03.23 (9221)              C10M
      AU9184567-A     92.03.26 (9222)              C10M-145/32
      CA2051495-A     92.03.22 (9223)              C10M-145/32
      FI9104430-A     92.03.22 (9225)              C10M-145/32
( --  other fields -- )
```

1) uniquely identify the field tag,
2) operate within the block of text that defines the patent number field,
3) manipulate the text that lies within a certain range of columns.

There are several unique features that make KEDIT the correct tool for this type of sophisticated editing. I will first describe in a general way why each of these features is important in designing the macro, and then describe how each feature differs from that found in a word processor. As one side note, many word processors include sophisticated macro languages, however it would probably be necessary to simulate versions of the following features that are missing from many word processors. The following list summarizes these unique features, each of which will be discussed in detail.

- **Lines of text as the unit record**—By treating each line of text as a separate entity, there is no danger of wrapping your highly-formatted patent record into a paragraph containing many different kinds of data including the field tags.
- **Rectangular blocks of text**—The ability to mark a rectangular block of text allows the entire block of characters to be marked, and deleted as a unit.
- **Persistent blocks**—The persistence of marked blocks allows the upper left-hand corner of the block to be marked and then a sequence of complicated steps completed, to identify the lower right-hand corner of the block.
- **ZONE (columns of text)**—The use of the ZONE command allows the patent number field tag "PN -" to be identified uniquely. This avoids the problem of falsely identifying a title containing the text "prePN -" as the beginning of the patent number field. If a title word was falsely identified as a field tag, the macro might mistakenly delete a rectangular block of text from the title field.

I will first demonstrate how these features are used to design the macro. Following the description of this sample macro, I'll describe how differently the corresponding features are handled within a word processor. The following steps are required to delete the unwanted text from the patent number field:

1) Locate the patent number field tag "PN -"
2) Mark the upper left hand corner of the block of text to be deleted
3) Locate the end of the patent number field (i.e., locate the next tagged line and then move up one line)
4) Mark the lower right-hand corner of the block of text to be deleted
5) Delete the block of text

These steps define the minimum needed to recognize the block of text and delete it from a single patent record. Using the KEXX macro language, these steps could be accomplished by the sequence of commands shown in Figure 2.

As shown later, this sequence of steps could be repeated automatically for a hundred or a thousand patent records.

Why would this sequence of steps be difficult to automate in a word processor? The difficulty lies in the four underlying features of an editor, which are handled completely differently in a word processor. The following discussion of each of these features describes the difference between the implementation within an editor and the way corresponding features are handled within a word processor.

**Lines Of Text As The Unit Record**

In this context, a unit record defines the smallest block of text that the software tool will treat as a discrete entity. In a word processor, the unit record is a paragraph. Any word processor will attempt to wrap text within the limits of a paragraph, where a paragraph is defined as text that exists within either blank lines or paragraph symbols. For an editor, the unit record is a line of text. An editor will not wrap text from one line to another unless explicitly instructed to do so.

When editing a downloaded patent record, the searcher does not want the accession number field, the inventor field and the title field to wrap into one large and difficult to read "paragraph." Therefore, when editing a patent record with a word processor, the first thing you must do is to add "hard returns" or "paragraph symbols" to the end of every line. Otherwise, any changes to the text will cause the entire record to wrap into a large ungainly mess. Anyone who has ever attempted to edit a downloaded record in a word processor is familiar with this phenomenon. In an editor, there is no need to add these "hard returns." Each line of text is treated as a separate record. An editor makes no attempt to wrap the highly-formatted text into a paragraph.

**FIGURE 2**
**Commands Using KEXX Macro Language**

```
reset block                             (unmark any existing block)
zone 1 6                                (limit any search to columns 1 to 6)
locate/PN -/                            (locate the patent number field tag)
zone 1 *                                (remove the column limitation)
clocate :37                             (move the cursor to the starting column)
mark box                                (mark the upper left hand corner)
down                                    (move to the first equivalent)
do forever                              (repeat for each equivalent)
    if substr(curline.3(),1,2) = ' ' then do    (test for a field tag)
        down                            (if not tagged go to the next line)
    end
    else do                             (if tagged this ends the patent field)
        up
        clocate :80
        mark box                        (mark the lower right hand corner)
        delete block                    (delete the block of text)
        leave                           (you're finished with this record)
    end
end
```

➡

KEDIT allows searches for text strings to be limited to a defined set of columns. This is especially important when attempting to find field tags, although it is also useful in many other applications.

## Rectangular Blocks Of Text

In an editor, a rectangular block of text can be marked. This means that the rectangle defined by columns 37 to 80 and lines one to five can be marked. This block of text can then be manipulated as a unit.

Many word processors can only identify continuous blocks of text, starting at any point and including all the text on every line to the ending point. A block of text beginning in line one, column 37, and going to line five, column 80, would include all the text in lines two through four. At least one word processor, Microsoft's Word for Windows, does have the capability to select columns of text on the screen by using the right mouse, however this ability is less versatile and is disabled when recording a macro.

In addition, many word processors include the capability to format text into columns. I'd like to emphasize that "formatting text into a column" is quite different from "marking a column of text." In recognizing the block of text to be deleted, the text is not being reformatted at all; instead a vertical slice of text is being marked and then manipulated. Many word processors lack the ability to mark or manipulate a vertical slice of text.

## Persistent Blocks

KEDIT has what is known as persistent blocks. Once a block is marked, it will exist until it is explicitly unmarked. Any number of other steps can be executed between the time the block is marked and the time that the block is manipulated. This feature provides great flexibility in creating macros. Likewise, one corner of the block could be marked, and then 50 other steps could be executed, and finally the other corner of the block could be marked. As shown in the above minimum version of the macro, a series of steps must be processed between the time the upper left-hand corner is marked and the time the lower right-hand corner is marked.

Word processors require that a marked block be manipulated immediately. Once a block is marked, a word processor assumes that any subsequent commands should operate on the marked block. The boundaries of the block must also be marked as one continuous action. The corners of the block must be marked one after another without any intervening actions. Furthermore, many word processors unmark blocks automatically whenever certain actions are taken. For example, hitting the <page down> key will often unmark (deselect) a block.

## ZONE (Columns Of Text)

KEDIT allows searches for text strings to be limited to a defined set of columns. This is especially important when attempting to find field tags, although it is also useful in many other applications. For example, one might think that the patent number field tag "PN -" would be a unique identifier string. However, if you do not limit the search for this tag to columns 1 to 5, you will also find lines that contain the text string "prePN -," which occurs in many Derwent titles. By using the zone feature, macros can uniquely identify lines of text that begin with any field tag. As another example, it is possible to identify quickly, all United States equivalents in a downloaded file by using the ZONE feature combined with KEDIT's selective editing feature, the ALL command (Figure 3).

```
'zone 7 8'
'all/US/'
```

It would also be possible to create a list of all U.S. or European equivalents by including Boolean logic in your ALL statement, for example,

```
all /US/ | /EP/.
```

It is also noteworthy that KEDIT can have both an "active zone" and a "marked block of text" at the same time. Since these two features are implemented separately, you can mark a block or one corner of a block while a zone is simultaneously limiting any text searches to a different column of text.

Word processors do not generally include any equivalent of the zone feature. Indeed, with today's emphasis on variable fonts, the concept of a "column" becomes almost meaningless. Depending on the font size, a character may appear on line five, column 30, or wrap to line six, column five. Without the ability to limit a text search to the first few columns, it is difficult to identify uniquely, the lines that begin with the accession number field-tag, "AN -," for example, and avoid finding lines containing phrases such as "GREATER THAN -25 DEGREES." On some word processors, it is possible to find text selectively, that begins in the first column by locating a paragraph symbol followed by the desired text. For example, in Microsoft's Word for Windows it is possible to search for the string, "^pPN -" and thereby locate a line tagged with "PN -." This is a far less elegant method and would only work for text that begins in column one. I often need the ability to limit my search to other specific columns, as in the earlier example of identifying all the U.S. patents in a listing of multiple equivalents.

## A COMPLETE WORKING MACRO (PARTIALLY IN PSEUDO-CODE)

The previously shown minimum version of the macro will execute exactly once. In a finished version, this subset of commands is embedded in a loop that will execute once for each patent record. Furthermore, the working version of this macro has been extended to both highlight and display the block of text and wait for approval before deleting each block. The user may view each highlighted block

**FIGURE 3**
**U.S. Equivalents Identified by KEDIT**

```
PN  - US5118835-A    92.06.02 (9225) 28p     C07C-381/00
PN  - US5118325-A    92.06.02 (9225)  6p     C10M-133/16
PN  - US5116522-A    92.05.26 (9224)  6p     C10M-145/14
PN  - US5110963-A    92.05.05 (9221)  3p
      US5118875-A    92.06.02 (9225) 16p     C07C-037/20
PN  - US5108462-A    92.04.28 (9220)  5p
PN  - US5102427-A    92.04.07 (9217)  6p
PN  - US5099046-A  92.03.24 (9215)
PN  - US5095171-A  92.03.10 (9213)
PN  - US5095170-A  92.03.10 (9213)
```

**FIGURE 4**
**A Complete Working Macro**

```
*************************************************************
*  Delete page-numbers, language and classification information
*  from Derwent patent equivalents (downloaded from ORBIT)
*
*  Note: This macro highlights the text just BEFORE it is deleted!
*
*  Hit  ENTER  once to highlight the extra data
*  Hit  ENTER  again to delete the highlighted data
*   -or-
*  EXIT  the macro without deleting the highlighted data
*------------------------------------------------------------------------
--------

'set autosave off'
( --- set any system parameters  ---)

( --- repeat the following section of commands for each patent abstract --- )
      'reset block'
      'zone 1 6'
      'nomsg locate/PN  - /'
      ( ---  test if this is the last patent record and exit the loop if it is the
last --- )
      'zone 1 *'
      'clocate :37'
      'mark box'
      'down'
      do forever
           if substr(curline.3(),1,2) = '  ' then do
                'down'
           end
           else do
                'up'
                'clocate :80'
                'mark box'
                ( ---  highlight and display the marked block of data ---)
                ( ---  ask the user to verify the deletion of this block of data --- )
                ( ---   if the user does not verify the deletion then exit this macro
--- )
                'delete block'
                 leave
           end
      end
( ---  go to the next patent abstract  --- )

( --- reset system parameters and "clean up" after the macro ---)
```

before it is deleted. If the macro falsely identifies an incorrect block of text, the user may optionally exit the macro without deleting the marked text (Figure 4).

Caution is always advised when learning a powerful tool such as KEDIT. Preferably, you should always test your macros on duplicate copies of your downloaded files. When first testing a macro, make a sample file that includes new records, ten-year-old records and twenty-year-old records. Remember that the format for patent records has changed significantly over the years, and a macro that functions perfectly on new records may have strange effects on records that are twenty-years old and have slightly different data structures. As one obvious example, a macro that tests for an author field will not function properly on early Derwent records that do not contain an author field.

## A POWER TOOL

This article has examined a few of the unique features of the editor KEDIT, which make this a powerful tool for patent searchers. Of course, KEDIT can be used on any type of ASCII file, including data downloaded from chemical or business databases. Although I have emphasized the strengths of KEDIT, I'd like to mention a few of its "weaknesses." KEDIT lacks the type of "keystroke record" feature that is used by many word processors to create macros. KEDIT also does not include any type of spelling checker, and makes no attempt at sophisticated formatting, such as headers or footers. In short, KEDIT lacks some important word processing features that would be used to produce a final, professional looking document. KEDIT is a superb editor, and was never designed to be a word processor. I use KEDIT whenever I want to evaluate, or selectively edit a downloaded file. To produce a final report, I import this evaluated and edited file into a word processor. For the patent searcher, the combination of KEDIT plus a modern word processor gives the best of both worlds.

## ACKNOWLEDGEMENT

My thanks to Dr. Thomas E. Wolff for reviewing this manuscript and for offering several helpful suggestions. Dr. Wolff wrote on KEDIT for post-processing searches in the June 1992 issue of *DATABASE*.

*Sandra S. Unger is a Staff Chemist in the Information Research and Analysis group with Exxon Research & Engineering. She received her B.S. (Chemistry) from Michigan State University in 1974 and Ph.D. (Organometallic Chemistry) from Michigan State University in 1979. She also received a Masters in Computer Science from the New Jersey Institute of Technology in 1991. She has been with Exxon since 1979 and in the Information Research and Analysis group since 1986, providing chemical and patent studies for clients throughout Exxon.*

*Communications to the author should be addressed to Sandra S. Unger, Exxon Research and Engineering, P.O. Box 121, Linden, NJ 07036; 908/474-6605; Internet—SandraUnger@delphi.com.*