A RUSSIAN-TO-ENGLISH TRANSLATION SYSTEM FOR SCIENTIFIC ABSTRACTS

A RUSSIAN-TO-ENGLISH TRANSLATION SYSTEM FOR SCIENTIFIC ABSTRACTS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

By

DANIEL TAPPAN, B.A.
Arizona State University, 1992

May 1996
University of Arkansas

This thesis is approved for
recommendation to the
Graduate Council

Thesis Director:

_____
Dr. Daniel Berleant

Thesis Committee:

_____
Dr. Susan Mengel

_____
Dr. Mitchell Thornton

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed _____

Refused _____

ACKNOWLEDGEMENTS

I would like to express my thanks to my major advisor, Dr. Daniel Berleant, and to my other committee members, Dr. Susan Mengel and Dr. Mitchell Thornton. I also thank everyone in the department for making my stay in Arkansas both profitable and enjoyable.

Thanks to Dr. Steve Johnson for his seemingly never-ending project which supported me at critical times and allowed me to buy food (although it never gave enough free time to eat it).

Special thanks go out to Snafy Couch for his years of questionable teaching, mentoring, and friendship which may have contributed in some minuscule way to my educational development.

I thank my family and Tippy for all their love and support throughout my studies.

And finally, I express my gratitude to Elena Kupriyanova for many contributions.

TABLE OF CONTENTS

TABLES AND FIGURES

CHAPTER 1

Introduction

"Computers stand up well to a grand master when it comes to the logic of chess, but they can't match the skills of a 7-year-old when it comes to language."

—Bernard E. Scott [35]

Garry Kasparov, the current world chess champion, recently went head-to-chip with the latest and greatest chess-playing computer, IBM's *Deep Blue*. Although Kasparov eventually claimed victory, the machine presented a definite challenge. Teaching computers to play chess has been an on-going project since they were invented. Gradual progress has resulted in this latest incarnation, which shows that it is only a matter of time before machines can emulate some human actions as well as—or even better than—their creators.

For nearly the same amount of time that chess has been a project of computer scientists, work has been going on to have computers process natural language, especially translate it. Although no machine is currently at a level of language use near that of the youngest bilingual child, significant progress has been made [15]. As the need for such machines increases, undoubtedly, so too will their performance.

The world continues to shrink as global communication systems connect even the remotest places. The technology of transferring information allows practically anyone to access anything nearly instantaneously from anywhere. However, after

the high-tech layers are peeled off this information, it is clear that most communication is still done with human languages.

The language barrier still effectively separates people, ideas, and information. It is not a new problem by any means, but it has become a problem of far greater consequence as a result of exploding global information technology. The shear volume of foreign-language text can no longer be processed entirely by human translators [23].

For decades, research has been conducted in the areas of Natural Language Processing (NLP) and Machine Translation (MT). Most problems have been clearly identified, but relatively few have been adequately solved [37]. Nevertheless, the machine translation industry has mushroomed into a multi-million dollar industry and promises to continue this expansion.

Project Motivation

This project was undertaken to create a prototype machine translation system to translate Russian scientific abstracts into English. It is an interactive system designed to demonstrate many aspects of machine translation and natural language processing. It is not intended to translated every possible Russian scientific abstract.

Tsujii [40] characterizes machine translation projects by the following two statements: (1) they never fail, and (2) they never succeed. The first statement means that any serious machine translation project will be able to solve *some* problems extremely well, maybe even find a perfect solution. The second statement, however, says that regardless of the effort and complexity, etc., no machine translation system can solve *all* problems well.

Selecting a subset of machine translation problems seemed to be the best choice for this project. Knight [16] sums this up well: "...the nice thing about

translation is that most any aspect of NLP is fair game for research." Along with Tsujii's statements, this instilled confidence that this project, regardless of how it turned out, would be worthwhile.

Abstracts

By some estimates, more than half of the world's scientific information is being published in languages other than English [6]. To stay informed in one's field, it is necessary to read a great number of articles. In many fields, a large volume of those written in foreign languages is not translated into English [17]. Moreover, often even the abstracts remain untranslated [37]. This presents a difficult situation for researchers who may not be aware of work in their field.

Abstracts are extremely useful for at least three reasons [32]: to decide if an article is important; to get a brief overview of an article before reading it; or to keep abreast of published work without actually reading the articles. Abstracts can also help to decide if foreign language articles warrant translation.

Abstracts also lend themselves well to processing by this system for a number of reasons:

- **Efficiency**: their small size means that even inefficient processing still produces a translation in a reasonable amount of time. This relieved some of the pressure to write fast, compact code; instead, more time was invested in designing a system that works well and can be relatively easily modified and enhanced.
- **Content**: they are concise, self-contained bodies of text that generally do not have complex pronominal references between sentences. For example, in the following sentences, to what does 'this' specifically refer?

Designing programs in a modular fashion is important. This makes them simpler to design and debug.

- **Formatting**: they usually consist of a single paragraph of arbitrary style, so there is no need to adhere to the original format (e.g., fonts, margins, spacing, tabs, etc.)
- **Non-Textual Information**: they do not normally contain charts, graphs, diagrams, flowcharts, tables, pictures, etc. Processing these takes considerable care and patience even for human translators [42].

Scientific Text

Scientific text is usually descriptive and explanatory. Its goal is to discuss a topic as clearly and concisely as possible. Whereas many other types of writing make readers ponder the topic and draw their own conclusions, scientific writing is supposed to be inherently clear, straightforward, and not open to wide interpretation [32]. Literary translation, for example, tends to focus on style, emotion, and interpretation, whereas in technical translation, fidelity is of greatest concern [37].

Cognitive interpretation, especially when ambiguity is involved, is an area where computers currently perform abysmally. Scientific text was chosen for this project because it minimizes the amount of ambiguity, and therefore the computational complexity necessary to interpret it. Schupbach [34] writes that "[i]n general [technical Russian] combines a high degree of grammatical and logical complexity with a surprisingly low level of ambiguous expression." This relative lack of ambiguity in scientific Russian is further shown by Belonogov, et al. [4] in their work on constructing a Russian-English dictionary for their system: of its

approximately 250,000 entries, roughly 80% had a single English equivalent and 13% had two equivalents. The mean number of equivalents was 1.37, with a maximum of 15.

The grammatical and logical complexity of scientific text may rival or even exceed that of other types of text [36]. However, since the major problem of ambiguity is minimized, more emphasis can be placed on the processing of grammar, syntax, and sentence structures. This program translates purely on the basis of these.

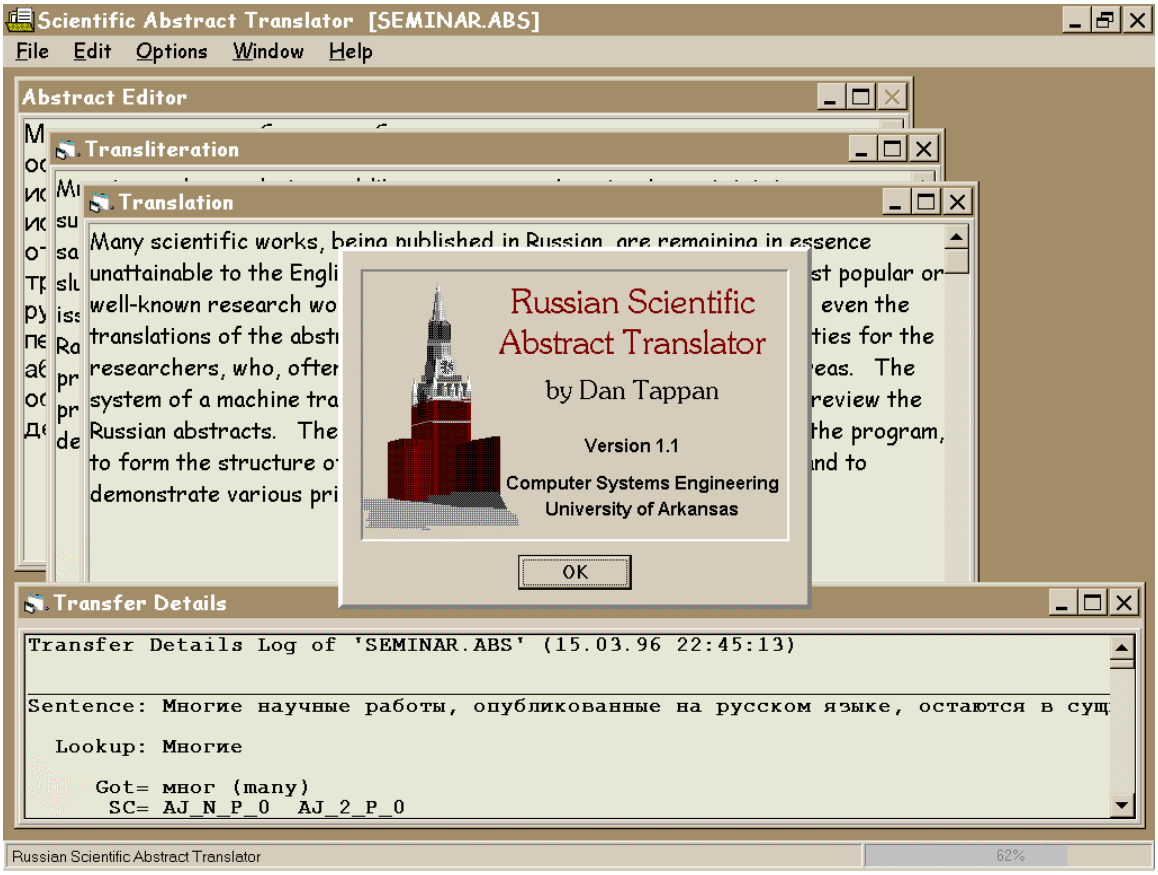
System Overview

The entire 500-page program[†] was written in Microsoft Visual Basic® 3.0 under Windows 95®. While BASIC is not normally the programming language of preference for natural language processing projects, it did nevertheless satisfy the requirements of this one. Execution speed and program compactness were less important than ease, flexibility, and clarity of programming, especially in a graphical environment. The visual interface plays an important interactive role between the program and the user. The program was designed to be relatively easy to use, and especially to demonstrate how machine translation works by giving as much feedback as possible.

Figure 1-1 shows the main interface with several windows displayed. The multiple document interface (MDI) contains all the windows and allows the user to resize and reposition them freely. The status bar along the bottom of the screen provides a description and/or help on whichever object the mouse pointer is over. A graphical status meter continually displays progress when the program is busy.

[†] Due to its large size, the source code is not listed here. For further information, contact Dr. Daniel Berleant at djb@engr.uark.edu

Figure 1-1 Main Interface



CHAPTER 2

Translation Background

On a planet of nearly six billion people speaking an estimated 4,000 to 10,000 languages and 20,000 to 50,000 dialects, the difficulties in communication are immense [44]. Since the advent of writing, there has been the need to translate between many of these languages. The world of the past, however, was far simpler and more isolated with a slower pace of life; people and information did not move around much. This is no longer the case, as global communications technology overwhelms the world with information.

Fundamentals of Translation

The philosopher I. A. Richards [31] writes that "translation is probably the most complex type of event yet produced in the evolution of the cosmos." Translation is less a science and more an art. In many respects, it does not lend itself to heuristic approaches any more than someone can list the steps to paint a masterpiece or write a best-seller.

Language is a means of giving or exchanging information. For information to be of any value, it must mean something. Therefore, language is a way to convey meaning. When languages are translated, it is not normally the words that are considered, but their meaning. Meaning falls into four categories [3 and 22]:

- Syntactic: how a word functions in a sentence (e.g., nouns, verbs, etc.)
- Semantic: what something means independent of context, such as its dictionary entry (e.g., a *lemon* is a sour yellow citrus fruit).

- **Pragmatic**: what something means in particular contexts, especially social, cultural, and historical. Pragmatic meaning elicits an emotional response in the reader. Its influence may extend over a single word, a sentence, paragraph, or even an entire work. Consider, for example, the feelings that the word "holocaust" evokes in many people.

- **Real-World**: common sense knowledge of what seems reasonable. For example, people know that a lemon is small, lightweight, semi-spherical, etc., even though this is not typically part of its dictionary description. They would react rather strangely to the announcement that a lemon had broken loose and squashed a car because it does not fit common expectation (i.e., normal lemon behavior).

Computers deal with syntactic and semantic meaning relatively well. Pragmatic meaning is too difficult to describe formally, and is therefore not normally considered in machine translation. Real-world knowledge is assimilated by living beings gradually over their lifetime. For machines, however, this knowledge is far more difficult to acquire. While it is not particularly hard to quantify the size, weight, and shape of a lemon, for example, the sheer number of such seemingly trivial attributes for common objects is difficult to process: humans must usually decide what is relevant information, how it relates to other objects, how to encode it, etc. This is a monumental undertaking.

Meaning and knowledge are often intertwined. An understanding of both is essential in translation. Arnold, et al. [3] list five types of knowledge humans use to translate. Computers current perform well on the first three points, but the fourth is still inadequately solved. The fifth point is not considered at all.

1. Knowledge of the source language.
2. Knowledge of the target language. This allows them to produce texts that are acceptable in the target language.
3. Knowledge of various correspondences between source language and target language (at the simplest level, this is knowledge of how individual words can be translated).
4. Knowledge of the subject matter, including ordinary general knowledge and 'common sense.' This, along with knowledge of the source language, allows them to understand what the text to be translated means.
5. Knowledge of the culture, social conventions, customs, and expectations, etc. of the speakers of the source and target languages.

Importance of Machine Translation

It is estimated that between 20 and 30 *b*illion dollars are spent worldwide each year on translation, and the demand continues to grow [9 and 10]. By some accounts, 40-45% of the European Community's (EC) running costs are language-related (i.e., translating and interpreting) [27]. In the United States, as of 1992 there were over 50,000 significant research centers supplying translations of scientific literature alone [6]. Slocum [37] sums up this predicament well:

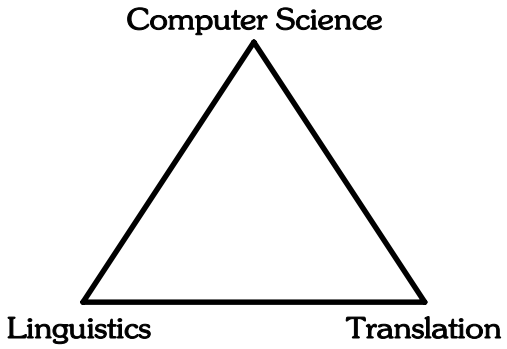
> The demand for technical translation is staggering in sheer volume; moreover, the acquisition, maintenance, and consistent use of valid technical terminology is an enormous problem. Worse, in many technical fields there is a distinct shortage of qualified human translators, and it is obvious that the problem will never be alleviated by measures such as greater incentives for translators...

Clearly the amount of material to be translated is not going to diminish. To meet the ever-increasing demand, something must take part of the load off the

already overburdened human translators. Machine translation appears to be the only practical and economically feasible solution to this problem.

Machine translation is a multidisciplinary subject bringing together the fields of computer science, linguistics, and translation, as well as software engineering, artificial intelligence, and psychology [20 and 23]. As the processing of language and translation gets deeper into the analysis of meaning, other fields may also play significant roles, such as philosophy, anthropology, and history.

Figure 2-1 Primary MT Disciplines



Machine translation systems offer several direct advantages over human translators [29]:

- They are often easier to train and keep up-to-date on terminology, etc.
- They usually translate two to ten times faster.
- Their cost per word is substantially lower: about 2.5¢ per word versus 5-40¢ per word for humans (both figures in 1986 dollars).
- Additional systems can be brought on-line more easily than new translators can be hired.

Furthermore, they have indirect advantages [29]:

- A considerable amount of typing work is eliminated.
- Output can be used as direct input to other information technology (e.g., communications systems, databases, etc.)

Perhaps the only disadvantage to machine translation systems is their present inability to translate as well as humans. Considerable work is being done world-wide

to improve their performance to a comparable level. Machine translation being such a multidisciplinary subject, progress in one area benefits many others.

Historical Overview

The roots of machine translation (or *mechanical* translation, as it was often called) can be traced back to the 17th century [13]. At that time, it was purely a theoretical suggestion. It was not until the 1930's that interest developed in designing and actually building devices to accomplish this [42]. World War II brought onto the scene electronic computers. Their primary use was for encryption and code-breaking. After the war, these machines were decommissioned for other projects, one of which was machine translation.

The post-war years brought tremendous advances in computer technology. It was also the beginning of the Cold War. The military and intelligence communities in both the United States and Soviet Union invested heavily in machine translation research to keep tabs on each other [37]. American research tended to be of a practical nature. The Soviet work, owing to limited access to significantly inferior computers, remained mostly theoretical, or was sometimes simulated by hand [9 and 38].

In August 1957, the United States was given another motivation to advance its ability to translate large volumes of Russian scientific work quickly: Sputnik [21]. This period became its high point of machine translation work, during which significant progress was made.

In 1964, despite the ever-intensifying space-race, the Automatic Language Processing Advisory Committee (ALPAC) was established to determine the feasibility of continuing funding for machine translation research in the United States [3]. Its findings, released in 1966, essentially drove a stake through the heart of MT. It

claimed that there was no shortage of qualified human translators, and that MT was a dead end [2]. For the next decade, often referred to as the 'Dark Ages of Machine Translation,' funding was practically non-existent. Europe had followed the ALPAC findings, so MT work was equally unpopular there [9]. Only Japan continued serious research [9].

In the late 1970's, interest was reborn. With this resurgence came about many new philosophies and methods of translation, as well as fresh funding. As of 1992, there were about 40 MT systems under development or in operation [9]. Slocum [37] provides a well-detailed history of many of the notable systems.

Machine Translation Quality

Many factors influence the quality of machine translations (e.g., type and complexity of the subject matter, speed of translation, etc.) While perfect translations are the ideal goal, it is widely agreed that this cannot be attained for general purpose, large-scale systems; correct translation of 80-90% of typical text is more reasonable and realistic [9]. It is indeed possible to achieve 100% accuracy with a *very* restricted range of text, but this is not practical for normal use.

High quality translation, however, may not be necessary for many purposes. For example, experts in technical fields can use rough translations to guess about the overall content of articles, or to decide if they should be 'properly' translated by some other means [29]. The former is especially true if the readers have some rudimentary background in the language. One study has shown that 98% of all roughly translated sentences are done so sufficiently well to be understood by experts [29]. Low-quality, high-volume machine translation output makes good economic sense.

Word-for-Word Translation

The earliest attempts at machine translation employed the simplest approach: word-for-word translation [26]. Since most early computer applications dealt with code-breaking, it seemed natural to approach machine translation as a similar task. Warren Weaver, commonly considered one of the fathers of machine translation, pondered this approach in 1949 [18]:

> One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say, 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'

Unfortunately, 'decoding' in foreign language processing did not achieve a fraction of the success it enjoyed in cryptography. Oettinger [24] provides as an example the following output of an early word-for-word translation system.

- **Russian Sentence**:

  8 cbe_ZYaZZ WdZ`s Y_s UaU_]\U ] e]afZ\U dZ_Zab%^bafU^faoi q_Z^fd]kZ e^]i eiZ` cUdU__Z_pab%cbe_ZYbWUfZ_pab½ eWs\] e gecZib` cb_p\gZfes UccUdUf Vg_ZWU U_XZVdo&

- **Proper Translation**:

  In recent times Boolean algebra has been successfully employed in the analysis of relay networks of the series-parallel type.

- **Word-for-Word Translation I**:

  (In, At, Into, To, For, On) (last, latter, new, latest, lowest, worst) (time, tense) for analysis and synthesis relay-contact electrical (circuit, diagram, scheme) parallel-(series, successive, consecutive, consistent) (connection, junction, combination) (with, from) (success, luck) (to be utilized, to be taken advantage of) apparatus Boolean algebra.

- **Word-for-Word Translation II**:

  In latest time for analysis and synthesis relay-contact electrical circuit parallel-series connection with success to be utilized apparatus Boolean algebra.

The translation process employed in Translation I is referred to as an *exhaustive lexical retrieval* [26]: every possible translation of each word is given.

Presumably, the reader is supposed to choose the best selection for each. This is clearly not a viable approach to machine translation. Even in the best case where the correct word translation is chosen automatically, no attention is paid to the grammar or word order of either language, as Translation II shows.

The main problems with word-for-word translation are that words are not independent; they interact with each other based on grammar and syntax, neither of which the translation properly reflects. Papegaaij, et al., [26] list additional deficiencies:

- Many words are translated differently in different contexts.
- In translation, word-order can change dramatically.
- Any language contains numerous idiomatic expressions and phrases which cannot be translated word for word, but have to be treated as single units.

Direct Translation

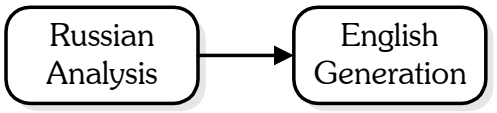
At the next level of complexity is direct translation (also referred to as a *transformer architecture* [3]). This works much as the word-for-word approach does, but on more advanced levels. Specifically, it deals more in-depth with grammar and context analysis, sentence restructuring, and compound noun and idiom processing. However, the source language text is still generally analyzed only to a level necessary to translate components and reconstruct sentences in the target language based on blind reordering rules [13]: no semantic analysis is done. Tucker [41] elaborates on this:

> No general linguistic theory or parsing principles are necessarily present for direct translation to work; these systems depend instead on well-developed dictionaries, morphological analysis, and text processing software to gain credible translations of the source text into a series of reasonably equivalent words and phrases in the target language.

He also lists ten steps characteristic of most direct translation systems:

1. Source text dictionary lookup and morphological analysis.
2. Identification of homographs
3. Identification of compound nouns
4. Identification of noun and verb phrases
5. Processing of idioms
6. Processing of prepositions
7. Subject-predicate identification
8. Syntactic ambiguity identification
9. Synthesis and morphological processing of target language
10. Rearrangement of words and phrases in target language.

Figure 2-2 Direct Approach



This project uses the direct approach, employing all these processing steps to varying degrees (but not necessarily in this order). Steps 3 and 5 are handled on a post-processing level which is not explicitly part of the translation engine (see Chapter 9, Post-Translation Cleanup Filters).

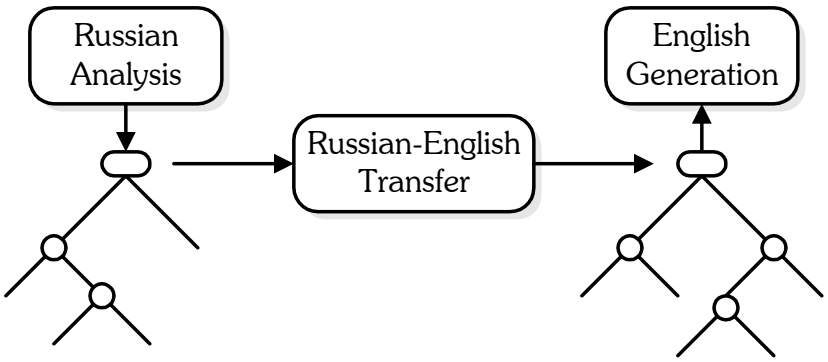
The direct approach is claimed to be most suitable for simple or restricted texts of limited vocabulary and a defined style [9 and 12]. Nevertheless, it has been employed with great success by the SYSTRAN system for decades to translate Russian technical material for the US Air Force [5]. In fact, SYSTRAN has been adapted to date for 15 language pairs and is in widespread commercial use in Europe [13]. The newer versions have evolved from a pure direct translation system into a complex hybrid incorporating many aspects of the transfer approach [12].

Transfer Translation

The transfer approach is a more complex method of direct translation. The major difference is in how sentence structure and word order are handled [13]. In

direct translation, the source language word order is generally preserved and reused in the translation, with only some superficial reordering done to fit the grammatical requirements of the target language. In the transfer approach, the source language sentence is parsed to generate an annotated grammatical tree structure which shows the role of each sentence component and its relationship to the others. From the structure and content of the source language tree, a second tree is built using the target language grammar and translated components. These trees serve as an intermediate representation between the source and target language sentences [23]. Finally, the target language tree is traversed to generate the translated sentence.

Figure 2-3 Transfer Approach



The transfer approach is the most commonly used [9]. A major advantage over direct translation is that the generation module knows the grammar of the target language. It has a much higher chance of producing grammatically correct and suitably formed translations [3]. Its major disadvantage is that the intermediate representations are language-dependent [13]: each language pair requires a separate transfer module (Figure 2-3). This problem will be discussed in more detail under *Interlingual Translation*.

Statistical Translation

Students of foreign language often build their foundation vocabulary with lists of words, or maybe with flash cards. This provides an efficient mechanism to learn foreign words and how they translate. Unfortunately, it tends to ignore one of the critical tenets of language use: context. Not only is it important to know *what* the translations of words are, but also *when* each is used. This skill normally is developed by reading (i.e., seeing words in context).

Most translation systems are 'taught' their vocabulary in a formal manner; they are spoon-fed words with translations, as well as their complex grammatical descriptions and whatever else the systems need to know about them [3]. Context is, for the most part, ignored. Simply having access to the entire word stock of a language and all possible translations, however, does not necessarily mean a machine translation system will perform well if it does not know when (i.e., in which context) to use what it knows.

Context-sensitive translation has shown promising results through the statistical approach. Large bodies of text with translation are analyzed statistically to determine the probability of certain words or translations occurring in specific contexts. These probabilities usually become part of the words' lexicon entries. In many respects, this is the computer equivalent to learning by reading.

The statistical analyses are normally done on large (i.e., several megabytes) parallel bilingual texts aligned by sentences or even words. Fortunately, several such corpora already exist between certain language pairs. For example, Canadian parliamentary debates are recorded sentence-by-sentence in English and French [13].

Knowing the probability of words in context allows a machine translation system to determine—albeit probabilistically and still with considerable uncertainty—

whether its particular translation makes 'sense' [3]. For instance, a system unsure whether "man-eating" is a reasonable translation could base the decision on the probability of its use in context: it tends to occur at a substantially higher frequency with "shark" than it does with "goldfish."

Unfortunately, obtaining significantly large aligned parallel corpora for most language pairs is rather difficult. In many cases, the text is also less than ideal: it can be argued that the language of parliamentary proceedings is not representative of the language as a whole.

Furthermore, problems in the sentence alignment between texts undermine their statistical relationships [3]. For example, scientific Russian sentences in translation tend to be broken apart for clarity or easier reading in English. In effect, these separate sentences are no longer statistically related to the original sentence.

Example-Based Translation

Translation by example can be considered a derivative of the statistical approach. Large corpora of bilingual text are analyzed to generate relational data, but on a sentence or phrase level [13]. The goal is to compile a database of commonly used templates and their partially translated forms. Within each template, those words which can easily be substituted with other words are replaced with variables; all others remain fixed. When a new sentence is encountered which matches the template, the variables are translated and substituted. The following example has two variables, *X* and *Y*:

- **Template**:
  Open *X* and put *Y* in.
- **Example Substitutions**:
  Open *the closet* and put *the shoes* in.

Open *the refrigerator* and put *the milk* in.
Open *the warp core* and put *the dilithium crystals* in.

Hutchins and Somers [13] aptly describe this as a linguistically sophisticated foreign language phrase book used by tourists. Alone it is not a viable approach to large-scale translation; however, it could be used to augment other approaches. A good application appears to be the translation of instruction manuals, where a limited set of sentence templates is often repeated.

Interlingual Translation

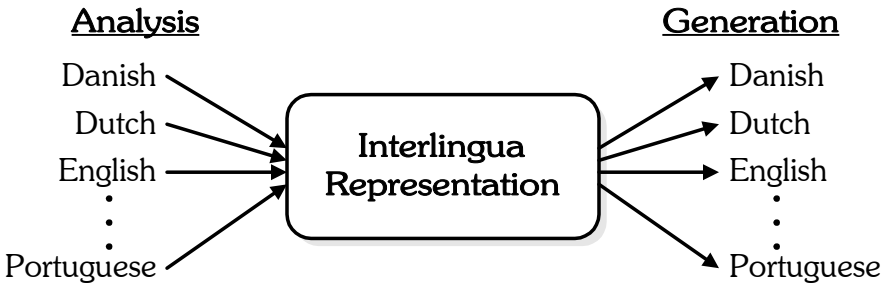
Most machine translation systems have highly language-dependent designs. An effective method for translating from German to English, for example, would most likely not be applicable for translating from German to French, or even from English back to German. In other words, bi-directional translation of each language pair usually requires two unique systems based on language-specific features.

This complexity is a major consideration when translations must be provided in many other languages. For example, the European Community currently conducts its proceedings in nine official languages: Danish, Dutch, English, French, German, Greek, Italian, Spanish, and Portuguese. With language-dependent translation systems, these nine languages would require $n(n\text{-}1)$, or 72 unique designs—not an attractive proposition.

The major bottleneck in language-dependent translation is the lack of structural commonality between the languages supported; each language conveys meaning with different grammatical constructs, etc. The *meaning*, however, should be virtually the same. By using the meaning itself as a commonality, and not grammatical or linguistic foundations, it is theoretically possible to describe anything written in any language [13]. This is known as the *interlingual approach*.

The source language is analyzed and converted into an abstract semantic representation known as *interlingua*. This is a precise and unambiguous representation that contains all the information needed to generate the given text (i.e., meaning) in any other supported language [10]. Figure 2-4 shows the basic structure of an interlingual system. The economic benefits of this approach are obvious: only $2n$ translation modules are necessary (i.e., n analysis modules and n generation modules). Properly written interlingua should also reduce or eliminate many types of ambiguity, and therefore improve translation quality [3].

Figure 2-4 Interlingual Translation in the EC



Unfortunately, much of the work on interlingual machine translation systems remains theoretical. The analysis of meaning is essential, but computers are still not up to this task. Interlingual translation is an ambitious idea, but it is still fraught with many shortcomings and often foiled by common language use. Take, for instance, the seemingly straightforward concept *to cook*. Nagao [21] lists no fewer than 18 different ways to cook something (i.e., bake, toast, roast, broil, fry, boil, steam, etc.) Choosing the correct meaning for a given context is actually far from straightforward.

The meaning of text is also heavily dependent on a culture's perception of the world [21]. Consider, for example, the great differences in Eastern and Western thought. Is 'meaning' really so common between cultures of radically different histories and philosophies?

Furthermore, a suitable interlingua capable of conveying such aspects and other semantic information has yet to be developed [3]. Esperanto has been used, but, being a quasi-natural language itself, it does not adequately solve certain problems of ambiguity [21 and 26].

Finally, an interesting and potentially valuable aspect of interlingual systems is their ability to translate between the same language. English text, for example, can be converted to interlingua, from which English text can be generated. This provides a method of testing the system, especially the interlingua: the original and 'translated' text might not be structurally identical, but they should convey the same meaning [3].

CHAPTER 3

Overview of the Russian Language

Russian is a heavily inflected, relatively free-structured language which lends itself extremely well to syntactic analysis. Whereas many other natural languages require considerable computational investment to tag their parts of speech, Russian comes essentially pre-tagged. This reduces the amount and complexity of processing required, as well as many common problems of syntactic ambiguity.

This chapter discusses the main features of Russian, especially those important to its processing. However, it is beyond the scope of this thesis to delve into the complex grammar of Russian. Any introductory text book would provide such details.

Cyrillic Alphabet

Russian is written with the Cyrillic alphabet. Its 33 letters have a very simple letter-to-sound correspondence which makes for relatively easy spelling and consistent pronunciation; almost without exception, Russian words are pronounced as they are written. Russian consonant clusters, however, often combine letters which are unfamiliar to English-speakers—for example, the present active verbal adjective, \Um]mUrm]½es (which/who is defending), is pronounced *zashchishchajushchijsja*. See Appendix Table B-1 for a description of the Cyrillic alphabet, as well as the keyboard layout used.

Grammatical Cases

The grammatical role of each word in a Russian sentence is indicated by its *case*. Verbs, prepositions, as well as several other parts of speech and common

constructions demand that the words they modify be in a particular case. The strict government and agreement between sentence components greatly simplifies determining how words interact to each other. Russian has the following cases (see the Glossary for further details):

- **Nominative**: almost without exception, the case of the subject.
- **Genitive**: indicates possession and negation, as well as the object of most prepositions.
- **Dative**: typically the case of the indirect object (i.e., the object indirectly receiving the action of the verb).
- **Accusative**: typically the case of the direct object (i.e., the object being acted upon by the verb).
- **Instrumental**: often the case of the object used to perform the action of a verb.
- **Prepositional**: always the object of several prepositions.

Two additional cases are encountered less frequently:

- **Locative**: used exclusively to indicate location.
- **Genitive Partitive**: used to limit the quantity of an object.

Inflections

The grammatical cases are indicated by the endings on words, known as *inflections*. Each part of speech typically has a different set of inflections, although several have none at all. Most words have multiple inflected forms which depend on their role in a sentence. Nouns, for example, usually have 12 forms (six singular and six plural); adjectives may have as many as 36 (nine for each of three genders and plural).

Table 3-1 shows the inflected forms of the word ebVU^U (dog). The inflections are underlined. Note that for the genitive and accusative plural forms, the

word is inflected by actually *removing* letters. Appendix C lists the 400+ sets of inflection patterns defined in this program.

Table 3-1 Inflections on 'Dog'

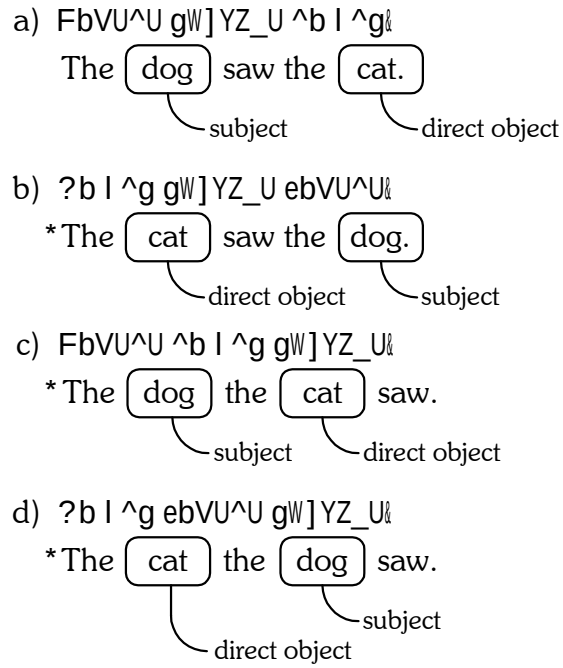
Case	Number	
	Singular	Plural
Nominative	собак**а**	собак**и**
Genitive	собак**и**	собак
Dative	собак**е**	собак**ам**
Accusative	собак**у**	собак
Instrumental	собак**ой**	собак**ами**
Prepositional	собак**е**	собак**ах**

Just by examining how a word is spelled, a wealth of syntactic information may be extracted. Unfortunately, this information is often ambiguous. Note that inflections are not necessarily *unique* endings: in this example, the genitive singular and nominative plural forms have identical inflections, as well as the dative and prepositional singular forms and genitive and accusative plural forms. Further analysis is necessary to resolve such ambiguity.

Word Order

Since the inflections indicate a word's grammatical role, and not its position in a sentence, Russian sentences are not usually constrained to any particular word order. For example, in the English sentences, "the dog saw the cat" and "the cat saw the dog," it is clear from the word order which animal saw which. Russian, however, does not use the word order for this. This allows practically any order to mean that the cat was in trouble, as shown by Figure 3-1 (the function of the inflections has been simulated in the English sentences with a caption):

Figure 3-1 Example Case Roles[†]



Regardless of where the cat and dog appear in the Russian sentences above, it is clear from the inflections which is the subject and which is the direct object. Although words can be ordered practically randomly, there are many conventional ways of structuring sentences. However, since these orders have no grammatical basis, they cannot be used reliably for sentence processing.

[†] Ungrammatical, unacceptable, or anomalous sentences are conventionally preceded by an asterisk.

CHAPTER 4

Import Filters

In order to process Russian text, clearly the program must be able to read it. While a standard character set mapping for the Roman alphabet (i.e., ASCII) has been in use for decades, Cyrillic mappings still vary greatly. Many variants are commonly used on-line [25]. To provide flexibility in reading Russian text in, the program has several built-in character mapping converters known as *import filters*. In addition to these, user-defined filters can be created. Appendix Table B-1 shows the built-in import filters.

Import Filter Editor

This editor provides a simple interface for defining how character sets are mapped. Each character mapping is described as a FROM → TO expression which converts a character or string to a single Cyrillic character in the internal program format (KOI-8). The FROM mapping is entered in one of three formats:

- **Character String Mappings**: these consist of one to seven printable characters, such as `a` or `zh` or `shch`. These mappings are case sensitive, so care must be taken when mapping capitalized strings. For example, is the capitalized mapping of `zh` to be entered as `Zh` or `ZH`?

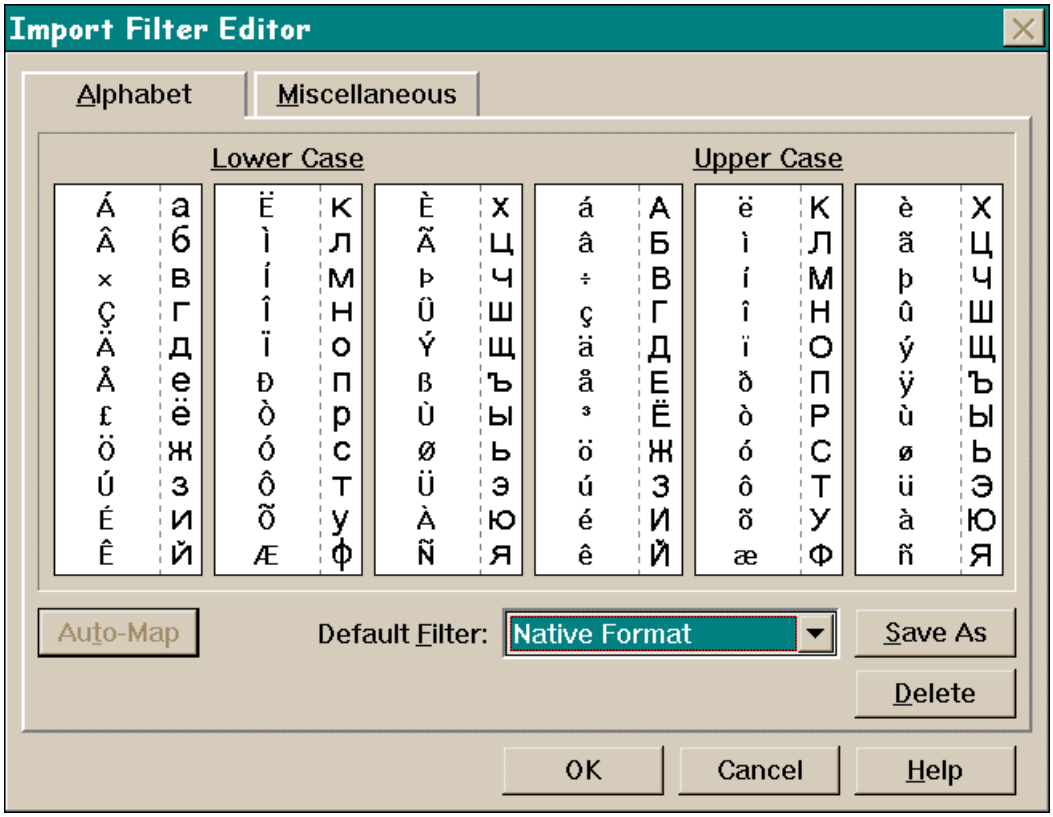
- **1-Byte Numeric Code Mappings**: these consist of a single decimal or hexadecimal number within the range of 0 to 255 or $0 to $FF, respectively. This corresponds to the character code used in the Russian character set of the abstract. The 1-byte numeric codes are useful for mapping special or non-printable characters. For example, it is not

possible to enter a space directly as a character string mapping; instead its decimal ASCII code 32 (or $20 hex) is used.

- **2-Byte Numeric Code Mappings**: these are extended 1-byte codes. They provide compatibility with existing and future multilingual character sets which use 2-byte character codes (e.g., ISO 6937/2, Unicode, etc.) A colon is used to separate the first and second numbers, both of which must be within the range of 0 to 255, or $0 to $FF.

Only one mapping can be assigned to each Cyrillic character. If the same mapping is given to more than one, only the mapping greater in length and higher in alphabetical order is processed; the others are ignored.

Figure 4-1 Import Filter Editor



Transliterations

To those who cannot read the Cyrillic alphabet, Russian text is essentially incomprehensible. Even words which are identical in both languages are usually unrecognizable (compare *компьютер* and *computer*). In order to provide some degree of readability, Cyrillic characters can be converted into the Roman alphabet (*компьютер* would be written as *komp'juter*). This is done by running the Russian text through a special transliteration filter. The transliteration can then be edited, saved, printed, or exported to another program. See Appendix Figure F-3 for a sample transliteration.

Abstract and E-Mail Modes

One of the most convenient uses of transliteration is for e-mail messages. Currently, Internet mail does not support foreign (i.e., non-ASCII) character sets well [11]. Writing e-mail in Russian with the Roman alphabet is tedious and prone to bizarre spelling errors. As a sub-feature for users who commonly send e-mail in Russian, the program provides a simple way to create, transliterate, and paste Russian text into e-mail messages.

The program operates in either *abstract mode* or *e-mail mode*. Both offer the same features; they differ only in how they deal with errors in the source document:

- **Abstract Mode**: abstracts are the primary document type. Strict error checking is enforced since unknown characters, misspellings, English words, or other formatting problems interfere with translation.
- **E-Mail Mode**: e-mail messages are the primary document type. These are not intended to be translated, so error checking is minimal. This is especially useful because no attempt is made to transliterate embedded English words, etc.

CHAPTER 5

Bilingual Lexicon

Vocabulary plays an essential role in translation. Humans and machines are similar in that they both need to know the words they are translating. They differ, however, in how much grammatical information about the words they need. Humans rely heavily on context, common sense, and feeling to translate, with the underlying grammatical constructions often playing a secondary role. However, most machine translation systems—including this one—are still rigidly constrained to translate based mainly on grammar [21]. Therefore, the more grammatical information provided for each word, the better the translations are going to turn out.

The *bilingual lexicon* is the program's vocabulary (see Appendix E for a full listing). Each Russian word has a separate entry with its grammatical information. Different parts of speech require different information, but each entry is structured identically (see Appendix Table B-6 for complete details):

Russian Word Information

- **Stem**: the root form of this entry without its inflections. Lexicon lookups are done on the stem only. This eliminates the need to have a separate lexicon entry for each inflected form of a word (up to 36).
- **Part of Speech Code**: an indicator of which of the 17 recognized parts of speech (e.g., noun, verb, adjective, adverb, etc.) this entry is. See Appendix Table B-2 for more details.
- **Primary Attributes Code**: a more detailed description of the part of speech. For example, a noun would be indicated as either masculine,

feminine, neuter, or plural-only. See Appendix Table B-2 for more details.

- **Secondary Attributes Code**: a description of how this word acts upon the word or words it modifies. For example, some verbs require objects in cases other than the usual accusative. Such a grammatical construction can then be expected with these verbs. Without this description, the program probably would not be able to resolve the role of the non-accusative objects. See Appendix Table B-2 for more details.
- **Inflection Pattern Number**: an indicator of which of the 400+ sets of inflections should be appended to the stem of this entry to derive all its inflected forms. See Appendix C for the inflection patterns.
- **Disambiguation Expression**: a logical expression indicating which translation of this entry should be chosen if more than one is possible. See Chapter 8, and Appendix Tables B-7 and B-8 for more details.

English Word Information

- **Translated Word(s)**: the equivalent English word or word forms. This field depends on the part of speech of the entry. For example, adjectives have a single translation, but verbs require many translated forms (e.g., present, simple past, past participle, etc.) See Appendix Table B-4 for more details.
- **Transfer Attributes Code**: this field, which is applicable for nouns only, describes how to use this noun in an English sentence. Three types of information are indicated (see Chapter 8 and Appendix Table B-3 for full details):

1. **Associations**: whether this noun is a person, place, thing, animal, or figurative/idiomatic. This is used for disambiguation purposes (see Chapter 8, Homograph Disambiguation).
2. **Article Usage**: which articles are used with this noun. This essentially eliminates articles which *do not* apply (e.g., **an* electricity).
3. **Plural Form**: how this noun is rendered in the plural (e.g., dog □ dogs, mouse □ mice, etc.)

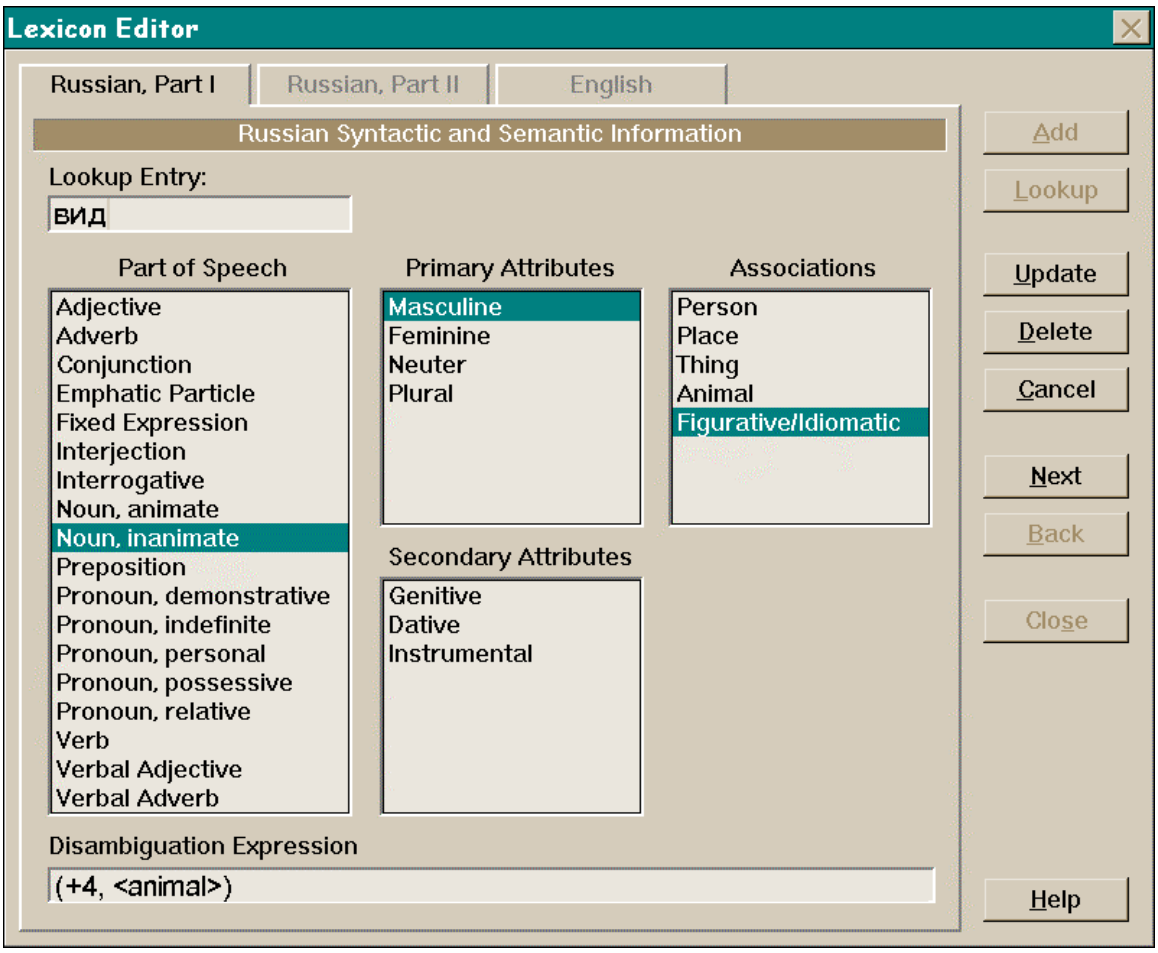
Lexicon Editor

The Lexicon Editor provides a simple interface for adding, looking up, and modifying lexicon entries. It hides the complexity of each entry's program-representation. The user selects graphically from available options and fills out simple forms of requested information. This is a three-stage process, with the options on each stage depending on the input to the previous stages.

Stage 1: Russian Attributes

In this stage, the Russian part of the entry is described. Its stem, part of speech, primary and secondary attributes, transfer attributes (for nouns), and any disambiguation expression are defined. Error checking is done to prevent invalid grammatical combinations. Figure 5-1 shows the first stage screen.

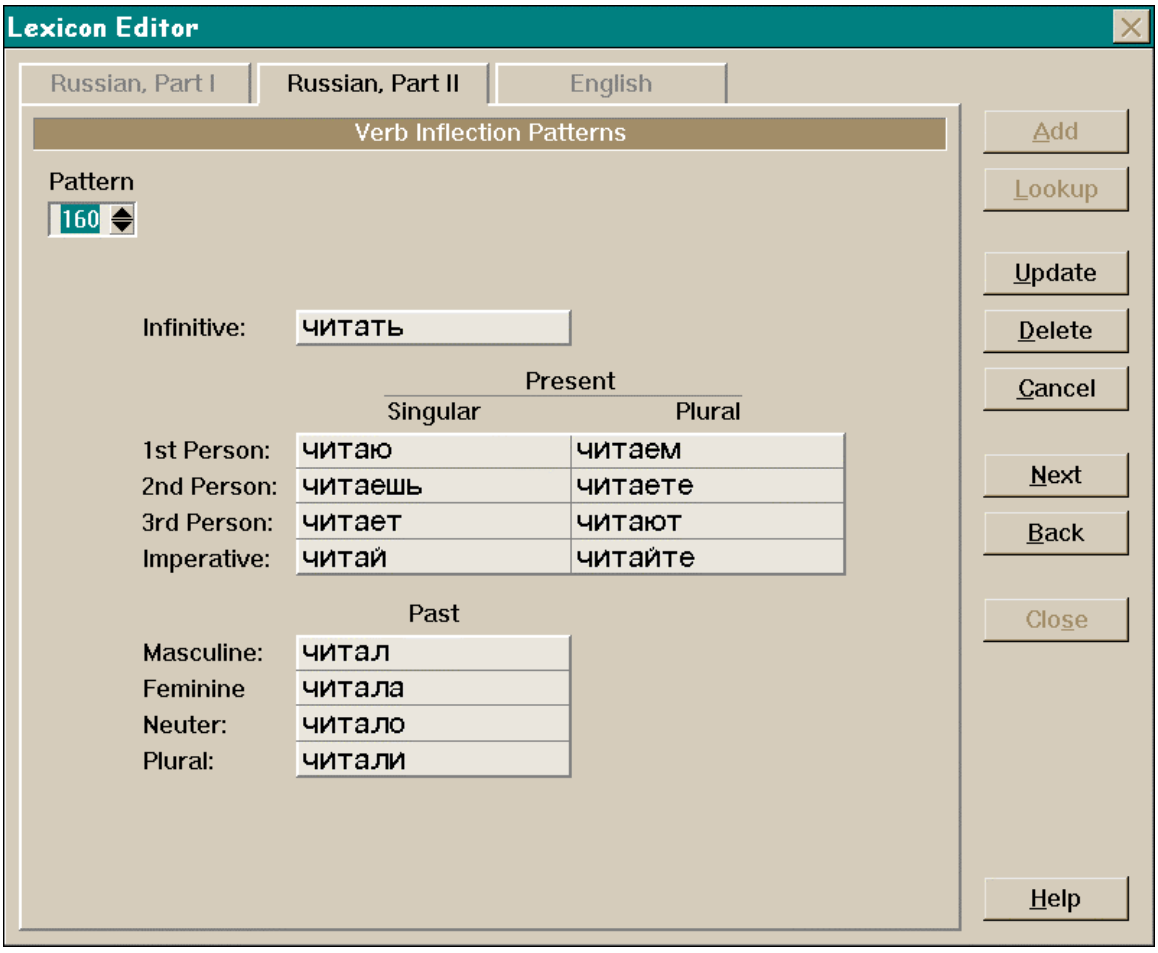
Figure 5-1 Lexicon Editor, Stage 1



Stage 2: Inflection Patterns

This stage has only one input: the inflection pattern number. All the inflected forms of the current entry are shown based on the chosen set of inflections. The structure of this entry depends on the part of speech. Figure 5-2 shows the screen for a verb entry.

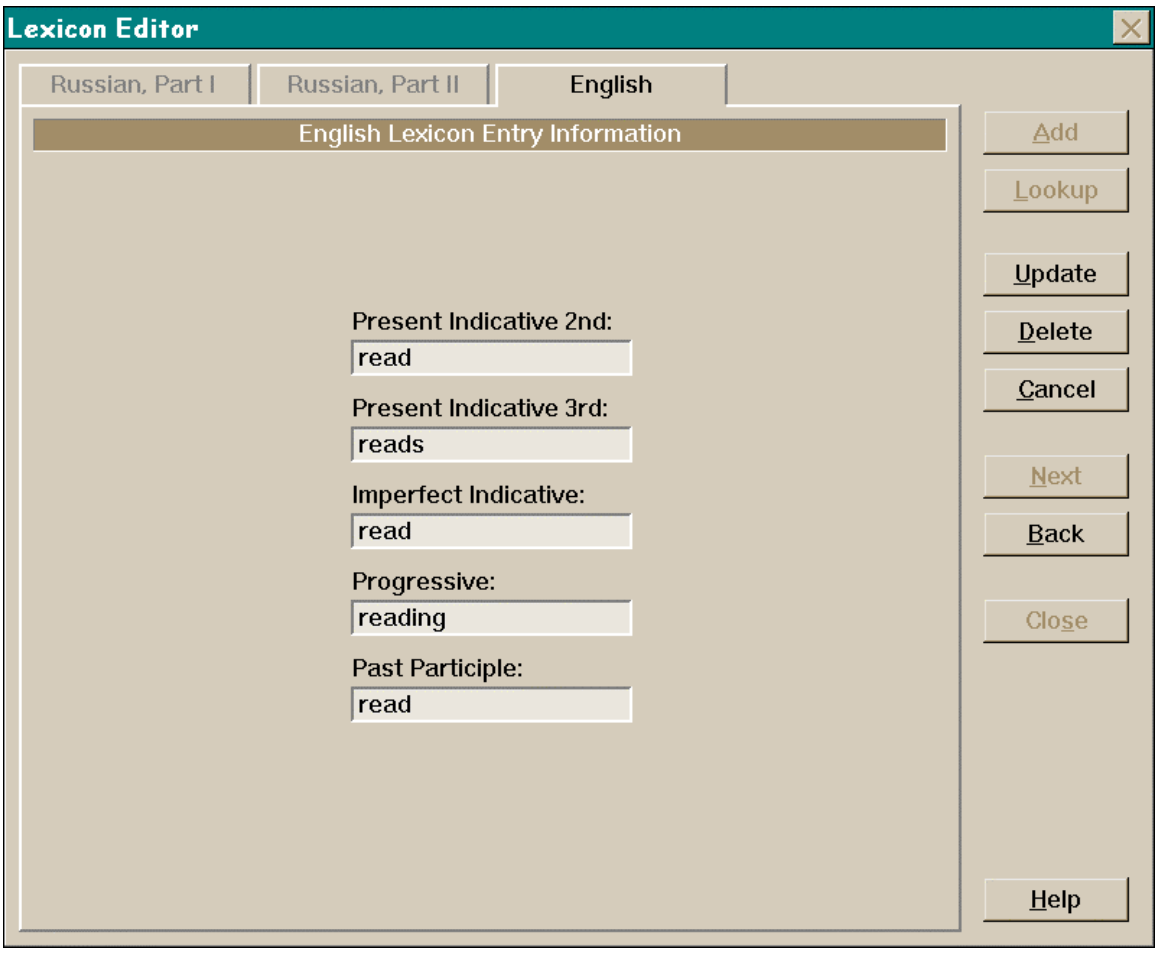
Figure 5-2 Lexicon Editor, Stage 2



Stage 3: English Attributes

This final stage describes the translated form or forms of the current entry. Depending on the part of speech, different English forms must be given. Figure 5-3 shows the screen for a verb entry.

Figure 5-3 Lexicon Editor, Stage 3



CHAPTER 6

Symbolic Constants

The grammatical information associated with each Russian word is most efficiently processed when it is in a form easily manipulated. To accommodate this, each word is assigned intermediate representations known as *symbolic constants*. Over 1,700 symbolic constants are defined, each of which concisely describes a unique syntactic form of a part of speech.

Symbolic Constant Usage

Many Russian words have complex syntactic descriptions and act upon other words. A symbolic constant representation simplifies determining the role of each word in the sentence. For example, "^b l ^U Wa] `UfZ_pab e_g l UZf kZ_bWZ^U$ bVnseas r mZXb cdUW]_U aZcbe_g l ab½ eb VU^Z" translates as "the cat is listening attentively to the person who is explaining the rules to the disobedient dog." The fifth word, bVnseas r mZXb, is described by the symbolic constant **VJ_PA_A_M_D** as a present active verbal adjective modifying a singular masculine noun in the accusative case and itself demanding an accusative object and optionally a dative object. This means that bVnseas r mZXb will be translated as a present tense relative clause construction (e.g., who is explaining) which modifies something accusative and dative in the sentence. The subject of this verbal adjective is being acted upon by something else that requires an accusative object.

From this detailed description of just a single word, it is possible to decompose most of the sentence. For instance, the nearest unbound dative object

(i.e., aZcbe_g l ab½ ebVU^Z) is likely going to be the one modified by the verbal adjective bVnseas r mZXb. An 'unbound' object is one whose modifier has not yet been determined. This verbal adjective also probably refers back to the closest accusative object kZ_bWZ^U. Symbolic constant analysis of other words will indicate that the verb e_g l UZf causes kZ_bWZ^U to be accusative.

As each symbolic constant is processed, the possible syntactic interpretations of the sentence decrease rapidly. Only one interpretation should be left after the entire sentence has been decomposed. If there is more than one, then each will be translated as a separate sentence, leaving it up to the user to decide which makes more sense.

Symbolic Constant Syntax

Each part of speech has a slightly different symbolic constant format. This format dictates the order of the components and their field position and length. The use of fixed-field strings makes symbolic constants very easy to dissect and compare. All symbolic constants for adjectives, for example, have **AJ** at the first position, a grammatical case indicator (i.e., **N**, **G**, **D**, **A**, **I**, **P**, or **L**) at the fourth position, a gender or number indicator (i.e., **M**, **F**, **N**, or **P**) at the sixth position, and a government indicator (i.e., **0**, **N**, **G**, **D**, **A**, **I**, or **P**) at the eighth position. Fields are delimited by an underscore. Table 6-1 shows the syntax for several parts of speech (for a complete listing of all symbolic constants, see Appendix Table B-10).

Table 6-1 Sample Symbolic Constant Syntax

Part of Speech	Component Fields				
	Field 1	Field 2	Field 3	Field 4	Field 5
Adjective	AJ	Case	Gender or Number	Government	
Adverb	AV	Type	Government		
Interrogative	IN	Case			
Noun	NO	Case	Gender or Number	Government	
Personal Pronoun	PP	Person	Gender or Number		
Preposition	EP	Government			
Relative Pronoun	PR	Case	Gender or Number		
Verb	VB	Person or Number	Gender or Number or Form	Government	
Verbal Adjective	VJ	Type	Case	Gender or Number	Government
Verbal Adverb	VV	Type	Government		

Symbolic Constant Generation

Most symbolic constant fields are derived from a word's bilingual lexicon entry. These are referred to as *static components* because they do not change depending on the form of the word in the sentence. For example, an adjective is always described by the symbolic constant component **AJ**, regardless of what it does in the sentence. The following lexicon entry fields generate static symbolic constant fields:

- **Part of Speech**: generates the *part of speech* component (i.e., first field).
- **Primary Attribute**: generates the *gender* component for nouns, and the *type* component for verbs.
- **Secondary Attribute**: generates the *government* component.

Dynamic components, on the other hand, depend on the form of the word in the sentence. These symbolic constant fields are derived from the inflections. For example, it is possible to determine the *case*, *gender*, and *number* components for nouns, or the *person*, *number*, or *form* components for verbs.

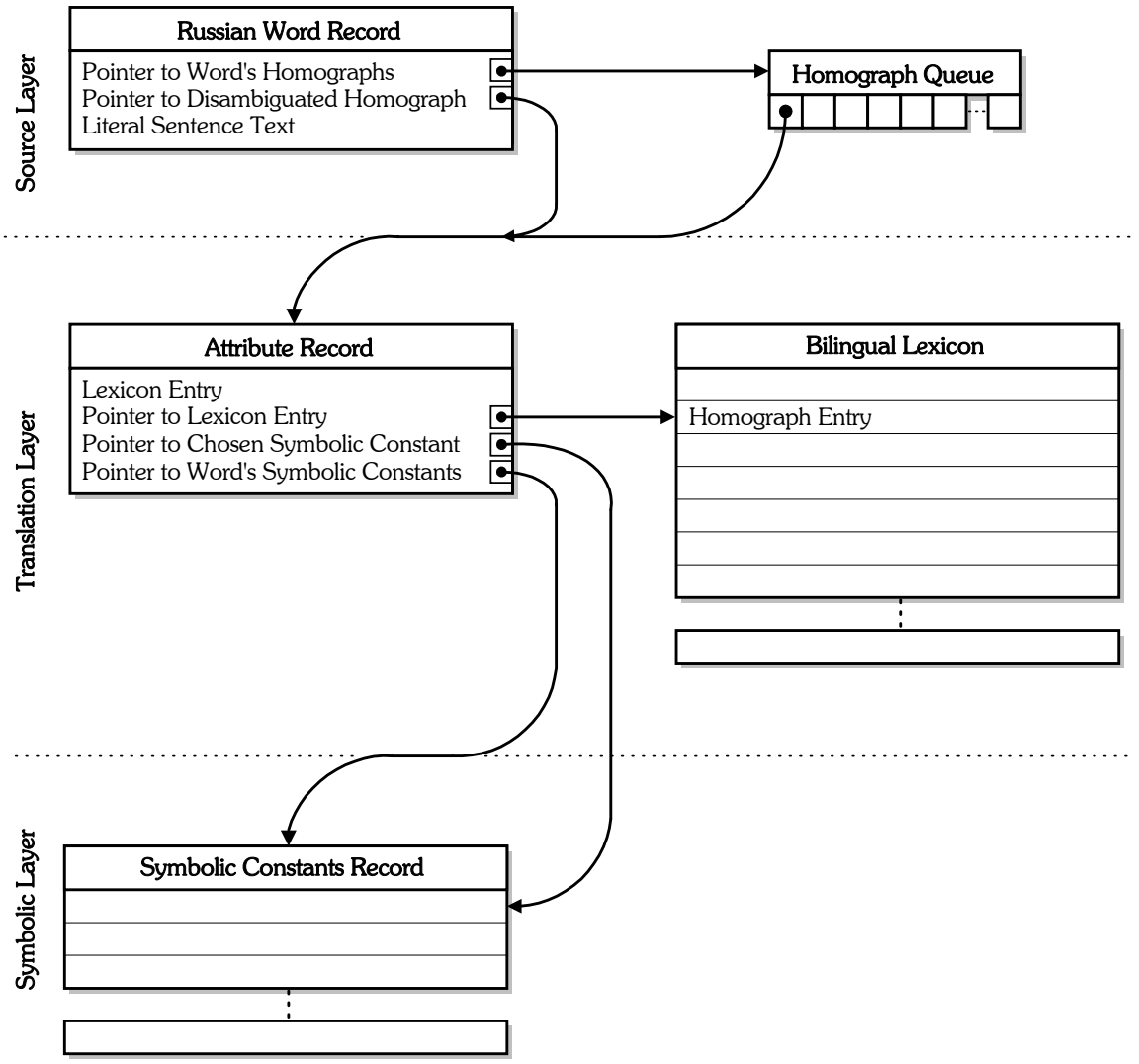
Sentence Representation Layers

Each Russian word in a sentence is described by three layers of information: the source layer, the translation layer, and the symbolic layer. Together the layers contain all the information extracted and generated for a single word:

- **Source Layer**: maintains pointers to all the possible translations of this word (i.e., homographs). It also keeps a record of the literal Russian text, including punctuation.
- **Translation Layer**: lists all homographs. Most words have a single translation, but some can have over a dozen. Each homograph is given a separate record with all its information. The *homograph queue* on the Source Layer keeps track of these records. During translation, the record which translates best will be chosen (see Chapter 8, Disambiguation).
- **Symbolic Layer**: lists all symbolic constants generated for each homograph. A pointer from the Translation Layer is assigned to the homograph chosen to describe this word's role in the sentence.

Figure 6-1 shows the three-layered word structure. In this example, there is only one translation of the word. If more were present, the homograph queue would have pointers to each translation, and the pointer to the disambiguated homograph would refer to the one chosen as the best.

Figure 6-1 Three-Layered Word Structure



While this structure is very efficient and convenient for the program, it often requires the programmer to jump through hoops to get to the desired field. The following statement returns the symbolic constant chosen to represent a single word:

```
SymbolicConstant =
   gSentence(ComponentPtr).Match(gSentence(ComponentPtr).MatchPtr).
   SymbolicConstants.SymbolicConstant(gSentence(ComponentPtr).
   Match(gSentence(ComponentPtr).MatchPtr).SymbolicConstantPtr)
```

where `ComponentPtr` points to the word in the sentence.

CHAPTER 7

Transfer Rules

> The Babel fish is small, yellow, leech-like, and probably the oddest thing in the Universe. It feeds on brain-wave energy received not from its own carrier but from those around it. It absorbs all unconscious mental frequencies from this brain-wave energy to nourish itself with. It then excretes into the mind of its carrier a telepathic matrix formed by combining the conscious thought frequencies with nerve signals picked up from the speech centers of the brain which has supplied them. The practical upshot of all this is that if you stick a Babel fish in your ear, you can instantly understand anything said to you in any form of language. The speech patterns you actually hear decode the brain-wave matrix which has been fed into your mind by your Babel fish.
>
> —Douglas Adams [1]

Alas, interfacing the computer with a Babel fish still remains unworkable. But the idea of using the source language to decode a translation matrix is quite feasible. No "brain-wave matrix" is employed, though; instead translation is done by using the symbolic constant representation of a sentence to decode a set of reorganization constructs known as *transfer rules*. These rules describe how a sentence is grammatically structured in Russian, and how it is restructured to form a rough English translation.

Transfer Rule Structure

Transfer rules can be very simple or extremely complex. Often it is more practical to break large, unwieldy rules into several smaller ones. Regardless of the rule complexity, however, each transfer rule consists of four components:

- **Comment Field**: an arbitrary English description of the transfer rule. This is not used for translation; it is simply helpful to keep the transfer rules organized.
- **Russian Expression**: a grammatical description of a Russian sentence or series of grammatically related sentences based on the symbolic constant representation.
- **English Expression**: the corresponding English grammatical description of the Russian sentence or sentences.
- **Frequency of Reference**: the number of times the transfer rule has been successfully applied to translate a sentence. The transfer rules are sorted by this field so more commonly used rules are tried first.

Transfer rules exist in two forms:

- **Compressed Form**: the rules as they are entered and modified from the Transfer Rule Editor, as well as how they are stored in the transfer rule file. Rules in this form can have nested optional parameters. See Appendix D for a complete listing of the compressed transfer rules.
- **Uncompressed Form**: the expanded form of each compressed rule. All combinations of required and optional parameters are generated. A single complex compressed rule can generate thousands of uncompressed rules. This is how they are represented in memory. When the program starts, the compressed rules are automatically uncompressed and written to the file T-RULES.LOG.

Transfer Rule Syntax

The *Russian expression* is used only to match a sentence to the appropriate transfer rule. This matching is done by comparing the structure of the current sentence to every defined sentence structure. The comparisons are done on two symbolic constant levels, both of which must pass to have a transfer rule selected. The first-level analysis quickly determines whether the more complex and time-consuming second-level analysis is warranted:

1. **Term Matching**: the order of the parts of speech from the sentence must correspond exactly to the order of the transfer rule. For example, the expression **`1:AV_N_* 2:AJ_(1)_(2) 3:NO_(1)_(2)_*`** would match on any adverb–adjective–noun constructions (e.g., the really cool dog, a terribly ferocious lion, etc.)

2. **Parameter Matching**: this is a more complex analysis to determine how the symbolic constants modify each other. In the above expression, the case and gender parameters, **`(1)`** and **`(2)`**, respectively, must be identical between the adjective and the noun it modifies. This parameter binding helps resolve more convoluted Russian sentences (see the "dog sees cat" example in Chapter 3).

The *English expression* describes how to form each translated word and where to place it in the English sentence. Each English term refers to an enumerated term in the Russian sentence. This is also referred to as a *slot grammar* [19]. The English expression for the above example would be: **`3A 1% 2% 3N`**. The first term, **`3A`**, indicates that the article associated with Russian Term #3, the noun, should be in English Slot #1. Terms **`1%`** and **`2%`** require no special processing of the translation since these refer to the adverb and the adjective of Russian Terms #1

and #2 and take English Slots #2 and #3, respectively. Finally, **3N** indicates that the proper form of the noun (i.e., singular or plural) should be placed in English Slot #4.

Translations of each term are processed according to the English term parameter. In this example, **A**, **%**, and **N** are used to process an article, anything requiring no processing (known as a *straight replacement*), and a noun, respectively. See Appendix Table B-9 for full syntax details.

Nesting of optional transfer rule terms is possible, but limited to two levels (i.e., one set of optional terms within another set). To simplify designing and processing nested sets, outer sets are enclosed in brackets (i.e., []), and any set nested within it uses braces (i.e., {}). Optional terms allow a single rule to describe many grammatically related Russian sentence structures.

Two levels proved to be sufficient to describe most Russian sentences; for any few exceptions, separate rules can be created. This decision has also been made in other machine translation systems for Russian: Sigurd and Gawronska-Werngren [36] write that "[f]ixed format rules with a restricted number of slots are preferred. Recursive rules allowing any number of infinitely long constituents at almost any place in the sentence, although mathematically interesting, are avoided." Such constructions are extremely difficult to manage, especially since each transfer rule in this program has two separate expressions to process. Term errors in either compressed expression usually make all uncompressed forms of a rule unusable.

The following example shows each component of a transfer rule. Most rules are far more complex, usually describing various subject–verb–predicate constructions. This example is for illustrative purposes only:

Comment:

```
[{adverb} + adjective] + [adjective] + noun
```

Meaning: a noun with zero, one, or two adjectives, and possibly an adverb if at least one adjective is present.

Compressed Russian Expression:

```
[{AV_N_*}   AJ_(1)_(2)_*]  [AJ_(1)_(2)_*]  NO_(1)_(2)_*
```

Compressed English Expression:

```
4A   [{1%}   2%]   [3%]   4N
```

Uncompressed Rules (with example translations):

1) `4:NO_(1)_(2)_* → 4A 4N`

The/a collie.

2) `2:AJ_(1)_(2)_* 4:NO_(1)_(2)_* → 4A 2% 4N`

The/a border collie.

3) `3:AJ_(1)_(2)_* 4:NO_(1)_(2)_* → 4A 3% 4N`

The/an awesome collie.

Note that in this rule, the adjective refers to Term #3, whereas the previous rule refers to Term #2. Both rules process identical constructions.

4) `1:AV_N_* 2:AJ_(1)_(2)_* 4:NO_(1)_(2)_* → 4A 1% 2% 4N`

The totally awesome collie.

5) `2:AJ_(1)_(2)_* 3:AJ_(1)_(2)_* 4:NO_(1)_(2)_* → 4A 2% 3% 4N`

The/an awesome border collie.

6) `1:AV_N_* 2:AJ_(1)_(2)_* 3:AJ_(1)_(2)_* 4:NO_(1)_(2)_* →`
`4A 1% 2% 3% 4N`

The/a totally awesome border collie.

Transfer rules are very good at reordering and translating components which already exist in the Russian sentence. Most components have a one-to-one translation correspondence (e.g., ebVU^U □ dog) or a one-to-many correspondence (e.g., k]fU_ □ has been reading) which provides all the necessary information to form the English sentence properly. Transfer rules, however, do not deal well with none-to-one or none-to-many correspondences. This occurs because Russian rarely

uses the present tense form of the verb 'to be.' The sentence "T GUd\Ua" translates literally as "*I Tarzan": the first word maps to 'I' and the second word to 'Tarzan.' Clearly, the verb form 'am' is necessary in the English sentence, but nothing in the Russian structure indicates that. In such cases, the English verb is omitted.

Luckily, scientific Russian often uses various other verbal constructions in place of 'to be' (referred to as *circumlocution*) [28]: to appear, to exist, to be located, to occur, etc. Furthermore, descriptions of location in Russian usually use more specific verbs than in English. For example, "the picture is on the wall," and "the chair is against the wall" would often be rendered in Russian literally as "[the] picture hangs on [the] wall" and "[the] chair stands against [the] wall," respectively. In any case, sentences like "*I Tarzan, you Jane" do occur with some frequency, even in scientific Russian. Although their translations might be more appropriate in the jungle, they are nevertheless completely understandable.

Another downside of transfer rules is that they rely entirely on grammar and syntax to translate. As a result, they are potentially capable of producing convoluted translations when the original sentence is structured in an unusual or unexpected manner. Such translations are often referred to as 'word salads.' Luckily, properly formed transfer rules usually prevent this from occurring.

Transfer Engine

Chapter 6 mentioned that each Russian word is assigned usually more than one symbolic constant which describes the word's role in the sentence. Multiple symbolic constants are a result of not every different inflection having a unique spelling. For example, the word ebVU^Z (dog) is in either the dative singular or prepositional singular case. At the time of symbolic constant assignment, no

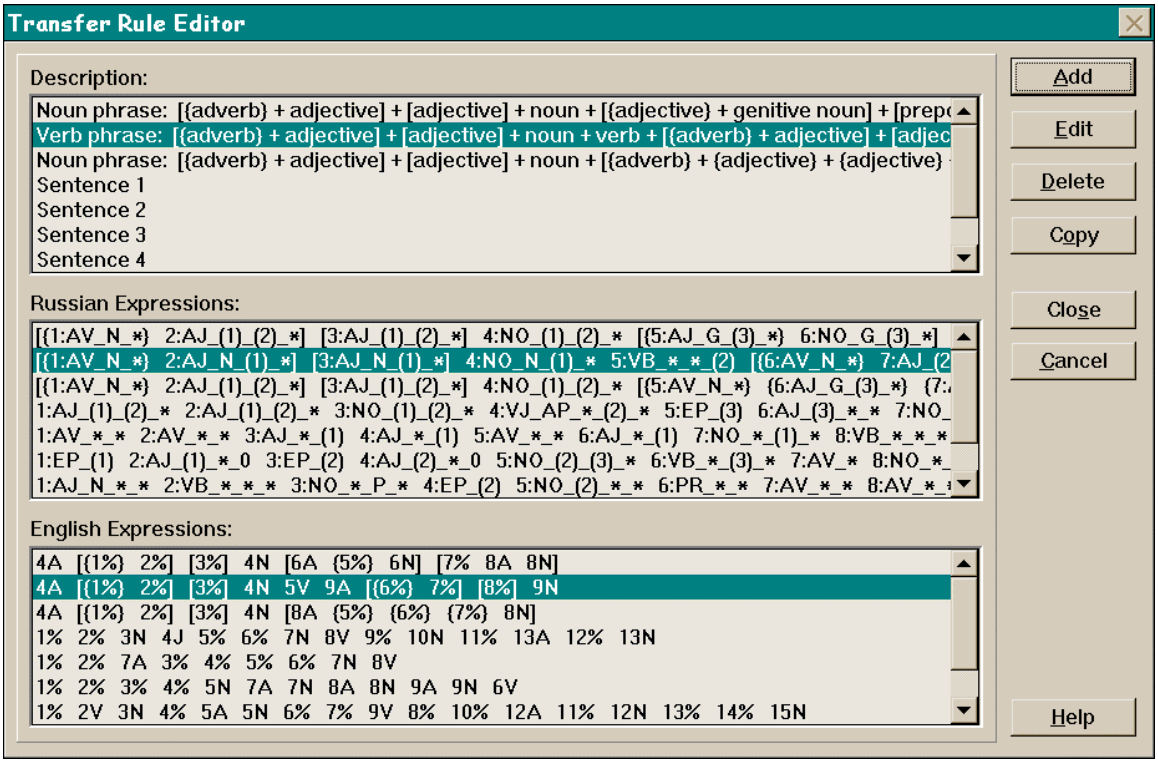
information was available to choose which constant fits best. It is the responsibility of the transfer engine to select the applicable symbolic constant.

The program uses a brute-force search method to resolve symbolic constant ambiguity: all possible combinations are generated and compared against every transfer rule. This method is computationally intensive, at one point employing 32 nested loops (32 is the maximum number of words allowed in a sentence). However, the list of transfer rules is ordered by frequency of reference, so more commonly used rules are encountered first. Typical Russian sentences require only a couple hundred iterations—less than a second of processing time—to find a match. The number of iterations is large only if no match can be found, in which case, every combination must be processed.

Transfer Rule Editor

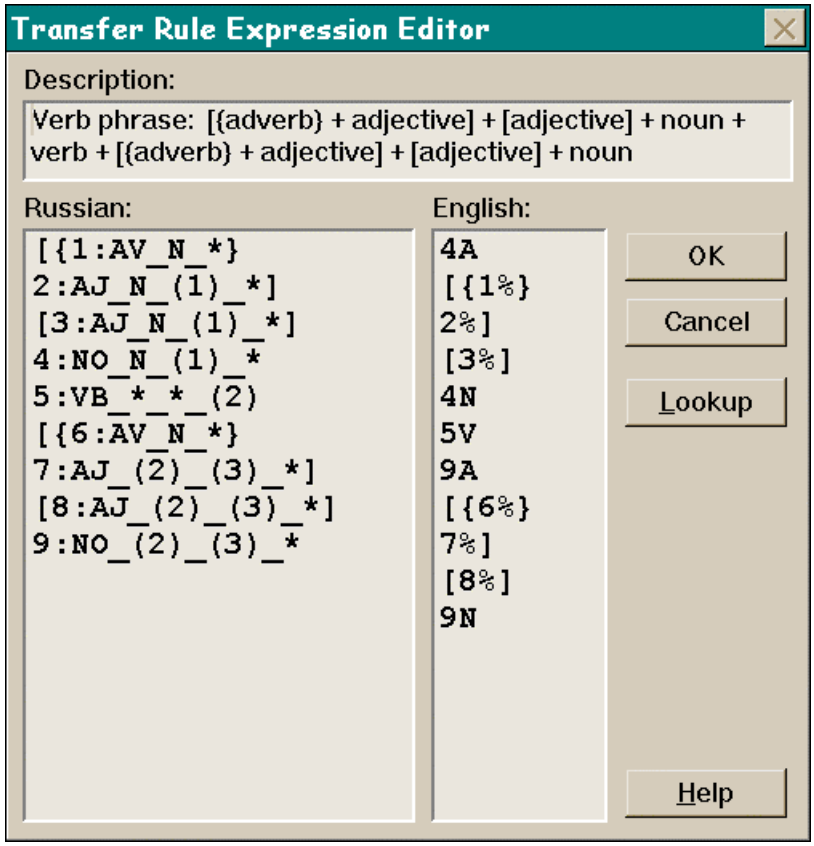
This editor provides an interface for creating and modifying transfer rules. The main screen displays the three components of every transfer rule: the comment field, the Russian expression, and the English expression. All three windows are linked together, so scrolling one correspondingly scrolls the other two.

Figure 7-1 Transfer Rule Editor



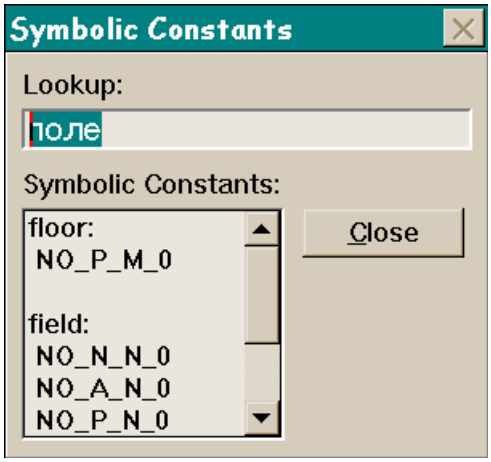
To create or modify a transfer rule, the Transfer Rule Expression Editor is used. Each transfer rule component appears in a separate field where the typical editing functions are available (i.e., insert, delete, cut, paste, etc.) Error checking ensures that the *syntax* of a transfer rule is correct; it cannot check whether a rule will work as intended, however.

Figure 7-2 Transfer Rule Expression Editor



Transfer rule terms are intimately coupled with the structure of the symbolic constants they process. These constants, however, are not in a particularly friendly form for the user. In an effort to simplify working with transfer rules, a special lookup can be done on any entry in the bilingual lexicon. This lookup returns the symbolic constants for the given form of the entry. All matches are returned, including any homographs. Figure 7-3 shows a symbolic lookup on the word form `cb_Z` (floor, gender, or field).

Figure 7-3 Symbolic Constant Lookup



CHAPTER 8

Translation

The program translates an abstract sentence by sentence, and each sentence word by word. Chapter 7 discussed how each word is assigned an intermediate representation of symbolic constants, and how these are used to reorder a sentence. This chapter goes into more detail on the reordering process, especially on how the English words are formed.

English Verbal Constructions

Translating Russian verbs into English presents a difficult problem. Although both languages have many verb forms in common (*e.g.*, past, present, future, *etc.*), their translations can be worded in different ways (*see* Appendix Table B-5 for a list of supported verb translations). For instance, the third person, plural, present tense form of the imperfective verb `k]fUfp` (to read) can be translated as:

- they read (present indicative)
- they are reading (present progressive)
- they have been reading (present perfect progressive)

Each of these translations, while conveying the same idea of 'reading,' has a slightly different connotation. However, since the Russian verb does not distinguish between them, the transfer engine has no information from which to choose the best form for the English sentence. This problem has been solved partially by the use of pseudo-random selection based on the frequency of each construction in a small sample of translated Russian scientific text. The probabilities for this example are

roughly 0.6, 0.3, and 0.1, respectively. The sample size was approximately 7,500 words taken from personal translations of [7] and [8].

A better method for verb translation might be partial context processing of the Russian sentence. Although the verb itself does not generally provide enough information, nearby words may help narrow down the choices. This could improve the translation quality, but there would always be some instances where the wrong English construction would be chosen. Many choices in translation by humans are based on what sounds and feels good—abilities the program cannot rely on.

English Article Selection

Machine translation systems perform poorly on linguistic phenomena which cannot be formally described well [21]. Among these phenomena are articles, which are notoriously problematical in Russian to English translations [36]. Since Russian has no articles, and there are few grammatical rules for their use in English, this presents difficulties in generating smooth translations. Luckily, articles—or the lack thereof—do not undermine the content of a translation.

In an effort to provide a heuristic method of article selection, each English noun has articles associated with it in the bilingual lexicon (see Table B-3, Transfer Attribute Details). This association indicates which articles make sense with the noun (e.g., *a* or *the* computer), and which do not (e.g., **an* electricity). A pseudo-random number generator chooses between 'a/an' and 'the' when both are possible. Trial and error has shown that 'the' should be given a higher probability (about 0.66).

Homograph Disambiguation

If natural languages were like computer languages, there would never be any uncertainty in their interpretation. As Ratner [30] writes: "when was the last time you encountered a `while` loop that meant something other than what it always

means!" Ambiguity is common in natural language and manifests itself as either a lexical or structural problem [13]:

Lexical Ambiguity

Ambiguity of this type involves multiple interpretations of a single word:

- **Categorical**: a word is more than one part of speech. For example, in English, *light* can be a noun (i.e., source of illumination), an adjective (i.e., not heavy), or a verb (i.e., to ignite).
- **Homographical**: a word has more than one meaning. For example, *bank* can be a financial institution or the edge of a river.
- **Polysemitical**: a word has a language-specific metaphorical relationship. For example: the *mouth* of a river. In Russian, this expression is written as a single word 'gefpZ' which has nothing to do with a mouth.
- **Translational**: a single source language word translates into more than one different target language word. See Table 8-1 for an example.

Structural Ambiguity

Ambiguity of this type involves multiple interpretations of an entire sentence:

- **Real**: a sentence has an unclear meaning. For example: the man saw the woman with the telescope. Did the man use the telescope to see the woman, or did he see the woman who has the telescope? Note that in Russian, this is rendered with two different constructions: `g[m]aUgW]YZ_ [Zam]agWfZ_Ze^bc and `g[m]aUgW]YZ_ [Zam]agefZ_Ze^bcb`.

- **Accidental**: parts of a sentence have categorical lexical ambiguities which make it unclear what agrees with or modifies what. For example: the man noticed her shaking hands. Was she giving someone a handshake, or were her hands unsteady? Again the Russian sentences are unambiguous: `g[m]aU\U`Zf]_$^U^ baU cb[ ]`U_U dg^] ([the] man noticed how she was shaking hands) versus `g[m]aU\U`Zf]_$^U^ Ydb[U_] Z¼ dg^] ([the] man noticed how her hands were shaking).

As these examples demonstrate, clarifying many ambiguities is difficult even for humans. The level of programmatic complexity necessary to deal with the more obscure problems would have been excessive for this project. Disambiguation, therefore, was limited to categorical and translational problems.

Categorical ambiguities were relatively easily solved with the use of symbolic constants. In nearly all Russian transfer rule constructions, only one part of speech correctly fits into a word slot. Also, since Russian parts of speech are identified by different sets of inflections, it is extremely rare for the role of a word to be confused.

Translational ambiguities presented a tougher problem. When a single Russian word has more than one possible English translation, the program must decide which translation to use. These decisions are based on contextual clues in the Russian text and its intermediate transfer representation. Each possible translation has an associated disambiguation expression which describes when it should be chosen over the other translations.

Disambiguation expressions work by searching forward, backward, or in both directions, from the word to be disambiguated (known as the *fulcrum word*). The distance (i.e., number of words) from the fulcrum word is known as the *search radius*. Disambiguation expression terms can be combined with AND/OR/XOR/NOT

operators to form compound logical expressions (see Table B-8 for the function operators). Three types of searches can be done within the given radius (see Table B-7 for the parameter syntax):

1. **Literal**: the Russian text is searched for the given string.
2. **Syntactic**: the symbolic constants are searched for the given part of speech or grammatical construction.
3. **Associative**: the Russian nouns are searched for the given association (i.e., person, place, thing, animal, or figurative/idiomatic).

Table 8-1 shows some possible translations of the word 'W]Y' [14]. Each requires a separate bilingual lexicon entry and disambiguation expression.

Table 8-1 Some Translations of 'W]Y'

Translates As	When Fulcrum Word
appearance	cannot be determined by any other disambiguation expression†
aspect (grammatical)	followed by the word 'X_UXb_' (verb)
condition	in the prepositional case AND preceded by an adjective
form	preceded by the preposition 'W' AND followed by a genitive noun
kind	followed by a genitive noun
mind	in the locative case AND preceded by the verb ']`Zfp' (to have)
prospects	in plural
sight	preceded by the preposition 'cd]'
species	followed by an animal
view	followed immediately by the preposition 'aU'

† Default translation

Transfer Details Log

One of the goals of this project was to demonstrate Machine Translation and Natural Language Processing principles. To this end, the program logs all important translation details:

- Each sentence.
- Each word and its symbolic constant representation(s).
- The set of symbolic constants chosen for the each sentence.

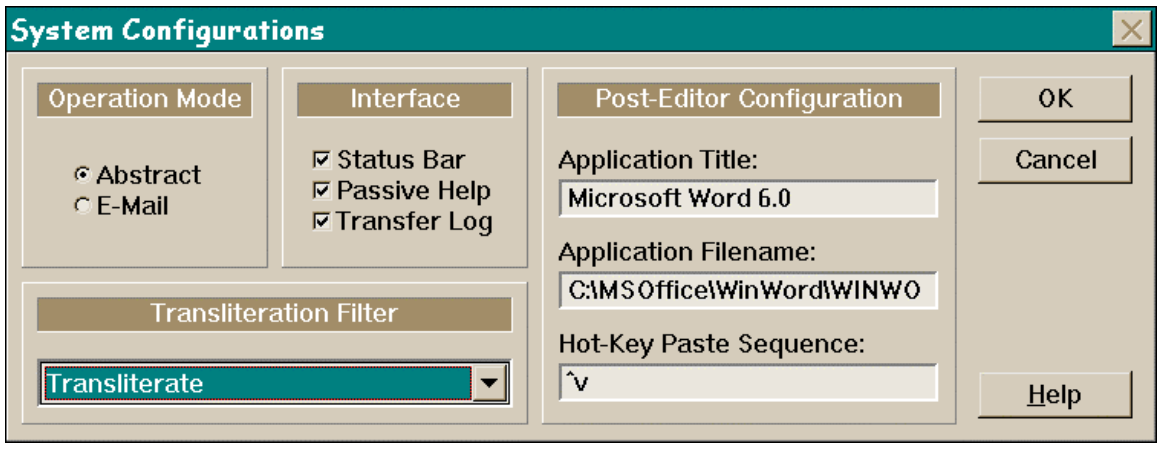
- Any disambiguation decisions.
- The transfer rule selected for each sentence.
- Any post-translation cleanup filter replacements.
- The final translation.
- Translation statistics (e.g., percent translated, translation time, etc.)

See Appendix Table F-5 for a complete listing of a transfer details log.

Translation Post-Editor

Translations can be edited, saved, and printed directly from the program; however, these functions are very limited. For added flexibility, the current translation can automatically be inserted into an external word processor. The default word processor is Microsoft Word® 6.0, but any Windows®-compatible program should work. The full filename of its executable program is necessary, as well as the sequence of hot-keys used to paste from the clipboard. Microsoft® products consistently use CONTROL-V (indicated as ^V). Figure 8-1 shows the Post-Editor Configuration dialog.

Figure 8-1 Post-Editor Configuration



CHAPTER 9

Post-Translation Cleanup Filters

Transfer rules are generally successful in translating content, but the English sentences they generate may be less than desirable; often they are choppy, verbose, or even partially unintelligible. *Post-translation cleanup filters* provide a simple, yet powerful way to improve raw translations. These are implemented using a straightforward search-and-replace algorithm. Two types of systematic transfer problems can be reduced or eliminated: cosmetic and idiomatic.

Cosmetic Problems

Russian scientific terms tend to be more descriptive in nature than their English equivalents. Since the bilingual lexicon does not support entries for compound words or fixed expressions, the program always generates word translations. For complex terms, this often results in very long-winded English. For example, the computer term 'time-of-day clock' could be written and translated at least three ways [45]:

- XZaZdUfbd]`cg_pebW ]ef]aabXb WdZ`Za]
  [a] generator of [the] impulses of [the] true time
- YUfk]^]ef]aabXb WdZ`Za]
  [an] indicator of [the] true time
- kUeo]ef]aabXb WdZ`Za]
  [a] clock of [the] true time

Such wordy translations can be replaced with the proper English terms or phrases. Depending on the desired translation quality and the time a user is willing to invest in compiling cleanup filters, it is possible to create an entire phrase-replacement dictionary.

Idiomatic Problems

Cosmetic problems in a translation may make the text more difficult to read, but they generally do not undermine the content. Cleanup filters play a more important role in correcting improperly translated phrases and expressions: they rely on the fact that improper translation will always be translated the same incorrect way. These errors are then replaced with the appropriate English expressions.

Although scientific text is generally less idiomatic than other text, some idioms are still commonly used. Human translators usually perceive idioms as single units, not literally. The program, however, adheres strictly to the grammatical transfer rules and translates accordingly. This often results in nonsensical translations. For example, the idiomatic expression '`W^bajZ^bajbW`' means 'finally' or 'ultimately.' Literally, it translates as '*at [the] end of [the] ends' (note that '`W`' here has been correctly disambiguated to 'at,' not 'in').

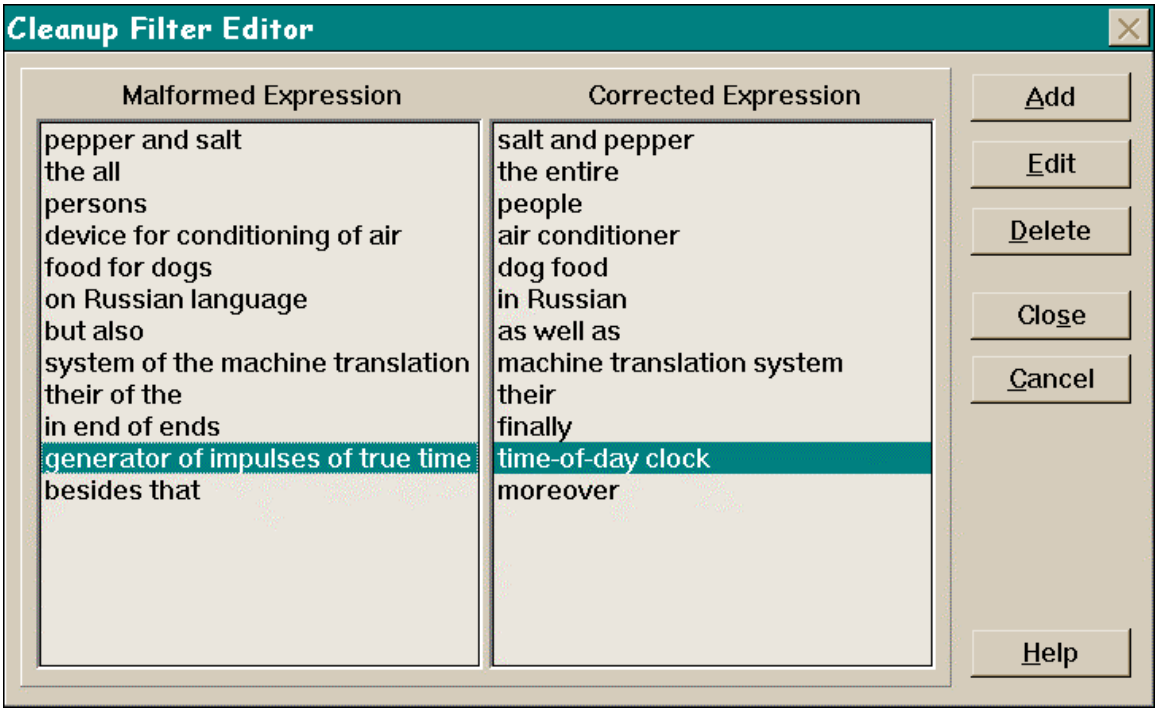
No transfer rule can correctly describe when such constructions should be taken literally, and when idiomatically. Consider '`W^bajZYas`,' which has the exact same grammatical structure. Here literal translation works fine: 'at [the] end of [the] day.'

Hard and Soft Cleanup Filters

Post-translation cleanup filters can be either *hard* or *soft*. Both types of filters function identically; they differ only in where they are defined. Hard cleanup filters are coded directly into the program and cannot be changed by the user. These filters take care of systematic translation problems which the user should never see or need to change. For example, English spelling rules dictate that the indefinite article 'a' should be written as 'an' if the following word begins with a vowel sound. A dedicated hard filter replaces the string ' a a' with ' an a'. This takes care of cases

like '* a apple' □ 'an apple'. Soft cleanup filters are user-definable through the Post-Translation Cleanup Filter Editor (Figure 9-1):

Figure 9-1 Post-Translation Cleanup Filter Editor



CHAPTER 10

Evaluating Machine Translations

Evaluating the quality of any translation is a subjective process. Judging a translation solely on its appearance is not a particularly effective method [36]. Perfectly smooth and grammatically correct translations may not be correct in meaning. Likewise, sloppy, grammatically weak translations may convey the content of the original very well. This chapter discusses some of the criteria used to evaluate translation quality.

Analysis of Quality

One popular method to evaluate quality is to look at sentence structure. Nagao [21] writes that sentence translations can be separated into the following categories from highest to lowest quality:

1. Properly formed sentences.
2. Properly formed sentences which include grammatical errors.
3. Sentences which are difficult to read unless certain portions are rearranged.
4. Sentences which are partly correct, but partly incorrect.
5. Sentences in which the word order is incorrect and a properly formed sentence is not produced.
6. Only fragmented translation with omissions.
7. No translated output.

Nagao [21] further evaluates a sentence for the following two characteristics, again in descending order of quality:

Grammaticality and Ease of Understanding

1. The sentence has a clear meaning, with no possibility of misunderstanding. Appropriate grammar, terminology, and sentence structure are employed, and post-editing is not necessary.
2. The meaning of the sentence is clear, but there are slight problems in the sentence structure, grammar, or terminology. Post-editing would easily be accomplished by human hand.
3. The overall meaning can be grasped, but there are small ambiguities due to grammar or terminology. Post-editing would best be done with reference to the original work.
4. The quality of the translation is poor, and there are many problems of grammar and terminology. Some understanding of the sentence can be inferred after lengthy study of the translation, but it would be better to undertake the translation anew by human translator than to post-edit the machine translation.
5. No understanding of the contents of the translated sentence is possible. Human translation is required.

Fidelity

1. The entire contents of the original text have been translated into an easily readable form.
2. The entire contents of the original text have been faithfully reproduced and [are] easily understood in the translation, but minor post-editing is required.
3. The translation is generally faithful to the original, but some reorganization of the word-order is required.

4. The translation is generally faithful to the original text, but there are errors in the relationships between phrases, tenses, singular/plural, position of adverbs, etc. Post-editing requires structural transformation.
5. The structure and contents of the original work are not well preserved in translation, with portions of the original text incorrectly translated or the referent of words and phrases incorrect.
6. The structure and contents of the original are poorly translated. Phrases and clauses are missing, but valid sentence structure is achieved.
7. The translation reflects neither the structure nor the contents of the original text. Valid sentence structure is not achieved because of missing subjects, predicates, etc.

Some degree of error in fidelity must be expected because translations, especially non-technical, are supposed to convey not only meaning, but emphasis, intensity, feeling, etc. Without direct consultation with the author, even human translations are based heavily on the translator's interpretation of the work [43].

Appendix F shows an example translation done by this program. Figure F-1 is the English abstract posted for the seminar presentation of this project. It was hand-translated into Russian (Figure F-2) so the program could translate it back (Figure F-4). Note that some slight differences are present between the back-translation and the original English text. This is to be expected with any translation, machine or human. In this case, the original text was changed on two occasions because it was translated from English to Russian and back to English. This was done to allow those not able to read Russian to judge the quality of the translation. A better demonstration of the program's abilities, however, is to compare Figures F-2 and F-4.

This program was designed to provide rough translations for users wanting to know the content of abstract. As such, small errors in translation, and choppy or wordy English constructions are acceptable, as long as the content is reasonably correct.

Modifying this System

No machine translation system is ever complete: it goes through a life-cycle much as any other software evolves [21]. Over time, features are added, improvements are made, and the system is continually fine-tuned to perform better. In this program, such modifications can be made in five places, the first three of which require no changes to the internal workings of the program:

1. **Disambiguation Expressions**: words which are translated incorrectly in certain contexts can be given multiple lexicon entries, each with a specific disambiguation expression.

2. **Transfer Rules**: unusually constructed sentences which are translated badly can be described with special transfers rules. A substantial set of exception-processing rules can be built up over time.

3. **Soft Post-Translation Cleanup Filters**: these user-definable filters are extremely powerful in correcting badly translated idiomatic expressions or compound nouns. As with the transfer rules, a large set of cleanup filters can greatly improve overall translation quality.

4. **Hard Post-Translation Cleanup Filters**: these hard-coded cleanup filters are useful in correcting errors known at compile time.

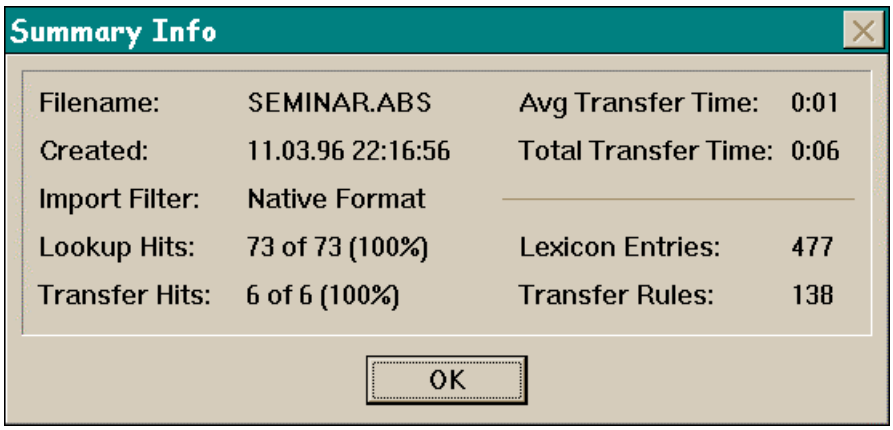
5. **Transfer Engine Modifications**: this is the most radical type of change. Additional functionality can be designed into the system to take care of

cases where the above four methods are not applicable. Also, existing programming statements can be adjusted for better performance, such as the probabilities for pseudo-random selections.

System Statistics

The program monitors several performance characteristics (see Figure 10-1), allowing the user to see at a glance how well it is working. Most of the information pertains to the current abstract translated, such as the lookup and translation success rates, and the translation time. The only system information recorded is the number of lexicon entries and transfer rules. For more details on translation performance, see Chapter 6, Transfer Details Log and Appendix F.

Figure 10-1 Summary Info



CHAPTER 11

Future Enhancements

"If we knew what we were doing, it
wouldn't be called research."

—Anonymous

Throughout the evolution of this research project, many decisions had to be made on which direction to take it, and, especially, how to get it there. Usually these were based on previous work, personal experience and intuition, and on the more-than-occasional "hmm, maybe this'll work..." In retrospect, the choices made resulted in a system that, in many respects, greatly exceeded the original expectations. This chapter discusses some improvements that could be made to it.

Translation Improvements

The transfer engine is the heart of the translation system. Any improvements to it would be reflected at every level of the translation process.

Transfer Rules

- **Deeper Nesting of Terms**: current rules are limited to two levels of nesting. More levels would allow fewer transfer rules to describe more Russian sentence structures.
- **Embedded Sub-Rules**: common sentence constituents could be assigned head structures (e.g., noun phrase NP → [{adverb} + adjective] + [adjective] + noun) These heads could then be referenced from other rules, eliminating the need to re-enter them. This is typical of others systems [33].

- **Use of Regular Expressions**: transfer rules must currently describe each term slot explicitly. The regular expression operators * and + (i.e., zero-or-more and one-or-more, respectively) would allow repeating terms to be described more concisely. This would reduce the number of rules and their complexity, as well as increase their readability.
- **Use of Logical Operators**: transfer rules could use logical operators (i.e., AND/OR/XOR/NOT) to describe grammatical constructions.
- **Simplification of Rule Design**: creating a transfer rule currently requires a great deal of patience and intimate knowledge of Russian grammar. A better interface could be designed—possibly similar to the lexicon editor—where the user could be led through a series of building stages. Ideally, actual sentences could be used as the basis of a new rule.

Transfer Engine

- **On-the-Fly Transfer Rule Evaluation**: the current method of uncompressing all the transfer rules at program startup allocates a large amount of memory. Evaluating the compressed rules would use far less memory and probably reduce translation time.
- **Computational Complexity**: although portions of the major transfer algorithms have been redesigned five times already, they still could be greatly improved. Far too many nested loops are employed for the current system to be practical with a full-sized transfer rule base and lexicon.
- **Unknown Word Processing**: unknown words currently prevent the proper translation of an entire sentence. If the program could skip the unknown

slot, only that particular word would go untranslated. This would be especially useful for proper nouns (i.e., names, places, etc.), which typically need no translation or lexicon entry.

- **Rule Matching**: the current transfer rule matching algorithm is first-fit. Other algorithms, such as best-fit, second-best-fit, etc., might provide more flexibility in translating. A selection mechanism could be integrated in as another level of processing.
- **Verbal Constructions**: when more than one English verbal construction is possible, selection is made pseudo-randomly based on simple probability. Superficial context analysis of the Russian sentence components might help narrow down the choices.
- **Article Selection**: as with verbal constructions, article selection is also done pseudo-randomly, with the definite article 'the' being favored. Similar context analysis, but of the English sentence, might provide a better basis for selection.
- **Feedback**: improvements in translation are now made by manually identifying problems and refining parts of the transfer process. This could be semi-automated with a post-translation analysis interface. For example, the user could be prompted to identify problems and indicate corrections. The program could automatically generate exception-handling transfer rules for problem constructions, or re-adjust the pseudo-random selection weights [3].
- **Disambiguation**: the program could automatically keep track of how often a lexicon entry is chosen, especially with respect to surrounding

words [26]. This could be the basis of a statistical disambiguation algorithm which could augment the current method.

Post-Translation Cleanup Filters

The use of these filters to improve the raw translation has proven exceptionally powerful. However, currently they are limited to straight search-and-replaces on literal text only. Greater flexibility could be added to the post-translation cleanup process if the rules allowed expression evaluation with logical operators. Furthermore, searches could be done on more than just the literal text, similar to the way disambiguation expressions can do literal, symbolic, and associative searches.

Compound Word Processing

While post-translation cleanup filters do an adequate job of resolving compound Russian nouns (e.g., "time-of-day clock" from "[a] generator of [the] impulses of [the] true time) and idiomatic expressions (e.g., "finally" from "*in end of ends"), this is far from an ideal solution. More processing needs to be done in the earlier stages of translation. Specifically, the bilingual lexicon needs to be significantly redesigned to support compound entries. The Russian scientific word stock contains thousands of frequently used constructions which could be incorporated (Zimmerman and Vedeneeva [46] is an exceptional source).

CHAPTER 12

Conclusions

The direct approach to translation used for this project has shown favorable results. Its most significant advantages are the following:

Flexibility

The program will always generate translations for any sentence structure it recognizes, provided that it knows all the words. For sentences which it does not recognize, it provides a direct word translation which usually conveys the content of the original in rough form. A user with no knowledge of Russian should always be better off with the translation from this system than with the original Russian abstract.

Expandability

There is a finite number of ways to form typical Russian sentences in scientific abstracts. Each time a new transfer rule is added to recognize an unknown sentence, that particular structure and any grammatically related sentences become a permanent part of the program's memory. Eventually most sentence structures will be described. This ability to modify the system continually is extremely important, as Nagao [21] writes:

> Stated bluntly, language is a massive conglomeration of exceptions and any competent machine translation system must therefore be able to incorporate the exceptional aspects of language simply and easily at every stage in its construction and function. In other words, the entire structure of the system must be open ended and evolutionary... During its construction and use, when a new or exceptional phenomenon is encountered, it is possible to insert that phenomenon directly into the system. The time required for improving the quality of such a system in this piecemeal fashion is of course extremely lengthy, and we anticipate that any rudimentary system will require years of refinement before it is truly useful. Such machine translation systems are virtually never fully completed.

Quality

Direct translations have relatively correct grammar with an acceptable English word order. Errors are generally cosmetic and require little or minor post-editing. Testing has shown that even badly translated abstracts still convey enough the original content to be useful.

Ambiguity

Translation by syntactic analysis naturally solves many complex problems of ambiguity, which is one of the major obstacles in the development of fully-functional machine translation systems [36]. This method of analysis, in combination with others, such as knowledge-based or statistical disambiguation, could form the basis of a very powerful disambiguation engine.

Far from being the ideal machine translation system, this project exhibits several disadvantages which make it unsuitable for practical use:

Lack of Flexibility

The fixed-form transfer rules are overly rigid. A sentence processed 99% correctly still fails if even a single word is unknown or in an unexpected position. Although a direct word-for-word translation is done in this case, it is essentially a 'dump' of everything extracted from the sentence. More leeway needs to be granted in matching transfer rules with sentence structures.

Lack of Expandability

Transfer rules are difficult to design and require intimate knowledge of Russian grammar. This prevents the typical user from expanding the system. Even adding new vocabulary—an extremely common procedure in machine translation

systems—requires the user to know the Russian words and to be able to describe them in grammatical terms.

Transfer rules do not lend themselves well to empirical designing and testing [3]. Usually rules are written in an *ad hoc* manner. As new rules are added, they may impact on existing rules, possibly degrading system performance. Thurmair [39] mentions this as a major concern: "One of the biggest problems in rule-based systems is that the application of rules has many side-effects, that they can feed or also block each other, and that rules written for a certain context fire in other contexts as well." For the small, prototype system developed for this project, such problems did not present themselves. However, in a full-blown version, efficiency (i.e., translation quality and time) would surely suffer.

In conclusion, this project has been a worthwhile endeavor which fulfilled or exceeded all personal expectations. The time and effort put into its research, development, and discussion provided a wonderful opportunity to learn more about the many and varied aspects of work done in the field of Natural Language Processing and Machine Translation.

Dan Tappan
23 April 1996

BIBLIOGRAPHY

[1] Adams, Douglas, *The Hitchhikers Guide to the Galaxy*, Harmony Books, New York, 1980.

[2] ALPAC, *Language and machines: computers in translation and linguistics.* A report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Research Council. National Academy of Sciences, Washington DC, 1966.

[3] Arnold, D., et al., *Machine Translation: An Introductory Guide*, Blackwell Publishers, Cambridge, MA, 1994.

[4] Belonogov, G. G., et al., "An Interactive Russian-English and English-Russian Machine Translation System for Multisubject Scientific and Technical Texts", *Nauchno-Tekhnicheskaya Informatsiya*, Vol. 27, No. 3, 1993, pp. 20-27.

[5] Bostad, Dale A., "Aspects of Machine Translation in the United States Air Force", *AGARD Lecture Series,* No. 171, June 1990, pp. 6.1-6.5.

[6] Carlson, Bob, "English-Only Orientation Limits Opportunities in Today's High-Tech World", *Computer*, Vol. 25, March 1992, pp. 81-82.

[7] Couch, Sanford, *Fizicheskii i nauchnii mir vokrug nas [The Physical and Scientific Word around Us]*, Condor Publications, Scottsdale, AZ, 1978.

[8] Couch, Sanford, *Nauka i tekhnika v nashei zhizni [Science and Technology in Our Life]*, Condor Publications, Scottsdale, AZ, 1979.

[9] Cynkin, Christopher, "Machine Translation and the Documentation Process", *Aslib Proceedings*, Vol. 44, No. 3, March 1992, pp. 121-125.

[10] Dane, Abe, "Computers Break the Language Barrier", *Popular Mechanics*, Vol. 169, April 1992, pg. 138.

[11] Fenwick, Peter, "Recent Developments in IT Standards of Interest to Translators", in *Translating and the Computer 8*, Aslib Proceedings, 1986.

[12] Hovy, Eduard, "How MT Works: MT systems tackle translation in several ways—each has its own benefits and trade-offs", *Byte*, Vol. 18, Jan. 1993, pp. 167-175.

[13] Hutchins, W. J., and H. L. Somers, *An Introduction to Machine Translation*, Academic Press, San Diego, 1992.

[14] Katzner, Kenneth, *English-Russian/Russian-English Dictionary,* John-Wiley & Sons, 1984.

[15] King, Margaret, ed., *Machine Translation Today: The State of the Art*, Edinburgh University Press, Edinburgh, 1987.

[16] Knight, Kevin [knight@ISI.EDU]. E-mail correspondence. 10 Nov. 1995.

[17] Lawson, Veronica, and Muriel Vasconcellos, "Forty Ways to Skin a Cat: Users Report on Machine Translation", *Aslib Proceedings*, Vol. 46, No. 3, March 1994, pp. 83-87.

[18] Locke, W. N., and A. D. Booth, eds., *Machine Translation of Languages*, Technology Press of MIT and Wiley, New York, 1955.

[19] McCord, M. C., "New Version of the Machine Translation System LMT", *Literary and Linguistic Computing*, Vol. 4, No. 3, 1989, pp. 218-229.

[20] Meyer, Ingrid, "The Role and Design of Computer Studies in a Research-Oriented Translation Program", *Computers and the Humanities 25*, 1991, pp. 297-301.

[21] Nagao, Makoto, *Machine Translation: How Far Can It Go?* Oxford University Press, New York, 1989.

[22] Nida, Eugene A., *Toward a Science of Translating*, Leiden, The Netherlands, 1964.

[23] Nirenburg, Sergei, ed., *Machine Translation: Theoretical and Methodological Issues*, Cambridge University Press, Cambridge, England, 1987.

[24] Oettinger, A. G., "The Design of an Automatic Russian-English Technical Dictionary", *Machine Translation of Languages*, Technology Press of MIT and Wiley, New York, 1955.

[25] Paine, Richard. [rpaine@cc.colorado.edu]. "Russian articles available in various fonts." In FRIENDS [friends@solar.rtd.utk.edu] 21 July 1995.

[26] Papegaaij, B. C., et al., *Word Expert Semantics: An Interlingual Knowledge-Based Approach*, Foris Publications, Dordrecht, The Netherlands, 1986.

[27] Patterson, B., "Multilingualism in the European Community", *Multilingua*, Vol. 1, No. 1, pp. 4-15, 1982.

[28] Perry, James, *Scientific Russian: A Textbook for Classes and Self-Study*, Interscience Publishers, New York, 1950.

[29] Pigott, Ian M., "Essential Requirements for a Large-Scale Operational Machine-Translation System", *Computers and Translation 1*, 1986, pp. 67-72.

[30] Ratner, Phillip, "An Introduction to Machine Translation" [Book Review], *Computer*, Vol. 25, Dec. 1992, pg. 118.

[31] Richards, I. A., *Toward a Theory of Translating*, in *Studies in Chinese Thought* by A. F. Wright, ed., 1953.

[32] Sandman, Peter M., et al., *Scientific and Technical Writing*, Holt, Rinehart and Winston, Inc., Fort Worth, Texas, 1985.

[33] Schneider, Thomas, "State of the Art in West Germany", *AGARD Lecture Series*, No. 171, June 1990.

[34] Schupbach, Richard, "On Technical Russian and the Teaching Thereof", *Slavic and East European Journal*, Vol. 32, No. 3, 1988, pp. 445-460.

[35] Scott, Bernard E., "The Five Layers of Ambiguity", *Byte*, Vol. 18, Jan. 1993, pp. 174-175.

[36] Sigurd, Bengt, and Barbara Gawronska-Werngren, "The Potential of Swetra—A Multilanguage MT System", *Computers and Translation 3*, 1988, pp. 237-250.

[37] Slocum, Jonathan, ed., *Machine Translation Systems*, Cambridge University Press, Cambridge, England, 1988.

[38] Smith, Peter D., *An Introduction to Text Processing*, MIT Press, Cambridge, MA, 1990.

[39] Thurmair, Gregor, "Recent Developments in Machine Translation", *Computers and the Humanities 25*, 1991, pp. 115-128.

[40] Tsujii, Jun-Ichi, "Machine Translation in Natural Language Understanding", *Literary and Linguistic Computing*, Vol. 4, No. 3, 1989, pp. 214-217.

[41] Tucker, Allen B., "Current strategies in machine translation research and development." In Nirenburg, Sergei, ed., *Machine Translation: Theoretical and Methodological Issues*, Cambridge University Press, Cambridge, England, 1987.

[42] Vasconcellos, Muriel, "Machine Translation: Translating the languages of the world on a desktop computer comes of age", *Byte*, Vol. 18, Jan. 1993, pp. 153-161.

[43] Wilss, Wolfram, *The Science of Translation: Problems and Methods*, Gunter Narr Verlag, Tübingen, Germany, 1982.

[44] Wright, John W., ed., *The Universal Almanac 1995*, Andrews and McMeel, Kansas City, 1995.

[45] Zejdenberg, V. K., et al., *Anglo-russkii slovar' po vychislitel'noi tekhnike [English-Russian Dictionary of Computer Science]*, Pergamon Press, USSR, 1984.

[46] Zimmerman, Mikhail, and Claudia Vedeneeva, *Russko-angliiskii nauchno-tekhnicheskii slovar' perevodchika [Russian-English Translator's Dictionary]*, John Wiley & Sons, 1991.

GLOSSARY

A

Adjective, compound comparative
two-word form of an adjective indicating that its quality is more than that of something else. For example: more interesting (Vb_ZZ]afZdZeao½). See also *Adjective, simple comparative*.

Adjective, compound superlative
two-word form of an adjective conveying the meaning of *most of all*. For example: most beautiful (eU`o½ ^dUe]Wo½). See also *Adjective, simple superlative*.

Adjective, long form
adjective which can be declined to indicate case, gender, and number. See also *Adjective, short form; Case;* and *Inflections*.

Adjective, predicate
adjective, short or long form nominative, which appears in the predicate (non-subject) portion of a sentence. For example: The dog is *big* (meaning: the dog is *a big dog*). See also *Adjective, short form; Adjective, long form; Case; Declension; Inflection;* and *Predicate*.

Adjective, short form
predicate adjective which can indicate gender and number, but only in the nominative case. See also *Adjective, long form; Case; Inflections;* and *Predicate*.

Adjective, simple comparative
single word form of an adjective indicating that its quality is more than that of something else. For example: faster (VoefdZZ). See also *Adjective, compound comparative*.

Adjective, simple superlative
single word form of an adjective conveying the meaning of *most of all*. For example: greatest (WZ_]kU½ l]½). See also *Adjective, compound superlative*.

Agreement
Russian enforces very strict grammatical requirements on its components. They must agree with each other in case, number, gender, and person. See also *Case, Gender, Government, Inflection, Number,* and *Person*.

Antecedent
word or phrase to which a pronoun refers back. For example: The dog is really friendly. She [i.e., the dog] also likes to play catch.

Article, definite
English determiner indicating specific reference to something (i.e., the). Compare: There is *the* house vs. there is *a* house. See also *Article, indefinite;* and *Target article*.

Article, indefinite
English determiner indicating non-specific reference to something (i.e., *a* and *an*). Compare: There is *a* house vs. there is *the* house. See also *Article, definite;* and *Target article*.

Aspect, imperfective
see *Verb, imperfective*.

Aspect, perfective
see *Verb, perfective*.

Attribute (code), primary
encoded form of syntactic information which differs for various parts of speech. For nouns, it indicates gender; for verbs, whether they are main or auxiliary, imperfective or perfective, reflexive or non-reflexive, determinate or indeterminate, etc. See Appendix Table B-2 for complete encoding details. See also *Attribute (code), secondary*.

Attribute (code), secondary
encoded form of the case(s) demanded by a preposition, noun, verb, adjective, adverb, verbal adjective, or verbal adverb. See Appendix Table B-2 for complete encoding details. See also *Attribute (code), primary; Cases;* and *Government*.

C

Case, accusative (direct object)
case of the agent being acted upon by a verb. For example: To throw the dog *a ball* (Vdbe]fp `sk ebVU^Z).

also used to indicate the object of prepositions demanding the accusative case: about (cdb), onto (aU), into (W), etc.

Case, dative (indirect object)

case of the agent receiving the action of a verb. For example: To throw *the dog* a ball (Vdbe]fp ebVU^Z `sk).

also used to indicate the object of prepositions demanding the dative case: toward (^), contrary to (WbcdZ^]), etc.

Case, genitive

Russian case indicating a relation of possession or ownership. Russian has no 'apostrophe-s' construction; instead, it uses the genitive case and an inverted word order. For example: The dog's bone □ The bone of the dog (^befp ebVU^]. Literally: Bone dog-of).

also used to indicate the object of prepositions demanding the genitive case: near (b^b_b), from (]\, bf), at (g), no (aZf), etc.

Case, genitive partitive

infrequently used Russian case indicating small quantities. Whereas English requires support words like *some*, *a [wee] bit of*, *just a pinch of,* etc., Russian can render this idea with a single word in the genitive partitive case. Most commonly used with items like tea, milk, and sugar.

Case, instrumental

Russian case of the agent being used to perform the action of a verb. For example: To hit the dog *with the ball*. In Russian, *with the ball* can be rendered with just the word ball (`sk) in the instrumental case (`skb`). English usually requires the prepositional phrase *with*, *using*, or *by means of* something.

also used to indicate the object of prepositions demanding the instrumental case: above (aUY), below (cbY), in front of (cZdZY), behind (\U), with (e), etc.

Case, locative

alternate Russian prepositional case used with some nouns to indicate location only. For example: aU cb_g (on the floor), not aU cb_Z.

Case, nominative (subject)

case of the agent performing the action of a verb. For example: *Lena* threw the dog a ball (*Лена* Vdbe]_U `sk ebVU^Z).

Case, oblique

any non-nominative case (i.e., genitive, genitive partitive, dative, accusative, instrumental, prepositional, or locative).

Case, prepositional

Russian case used with the prepositions about (b), on (aU), in (W), in the presence of (cd]), etc. Only prepositions can demand this case.

Conjugation
see *Verb, conjugation.*

Conjunction
any word, except a relative pronoun, used to connect words, phrases, or clauses (e.g., and, but, while, although, etc.)

Corpus (plural: Corpora)
for this project, a body of Russian scientific text from which examples of common sentence constructions were extracted. Most transfer rules are based on these examples. See also *Transfer rule.*

D

Declension
ending put onto nouns, adjectives, pronouns, etc. to indicate case, number, and gender. See also *Case; Inflection; Number;* and *Verb, conjugation.*

Declinable
noun, adjective, pronoun, etc. which can take a declension. See also *Case; Declension; Indeclinable; Inflection; Number;* and *Verb, conjugation.*

Diminutive
variation on a word conveying the meaning of small or unimportant, or to express affection. For example: cat ▫ kitty (^bf ▫ ^bf¼ab^).

Disambiguation expression
logical expression used to decide whether to choose one English word or phrase translation over another when there is ambiguity. For example, two translations of the Russian word W]Y are: view and species. A disambiguation expression might dictate that *species* is the correct translation if there is an animal nearby in the sentence. See also *Source association.*

F

Fixed expression
group of words or phrase which is best considered a single unit and should not translated word for word. For example, W^bajZ^bajbW taken as a fixed expression translates as *finally.* However, a literal word-for-word translation results in **in end of ends.*

G

Gender

set of three grammatical categories (masculine, feminine, and neuter) into which all Russian nouns fit. For living things, the gender is usually the same as its real-life gender (i.e., man is masculine, woman is feminine, etc.). However, for inanimate things, there is usually no pattern to gender assignments.

Government

all Russian prepositions, verbs, and some adjectives and nouns require that any word(s) they modify or act upon be in a certain case. See also *Agreement* and *Case*.

H

Homograph

words with the same spelling but different in meaning or part of speech. For example: bank (financial institution) versus bank (edge of a river), or saw (cutting tool) versus saw (to have seen).

I

Import filter

conversion table between different Cyrillic character sets and the one used internally by the program. Four import filters are built in: KOI-7, Native Format (KOI-8), Phonetic Unique, and Transliteration. See Appendix Table B-1 for complete mapping details.

Indeclinable

noun, adjective, pronoun, etc. which cannot take a declension. See also *Case; Declension; Declinable; Inflection; Number;* and *Verb, conjugation*.

Infinitive

non-conjugated (uninflected) form a verb, always preceded in English by *to*. For example: to say (XbWbd]fp–e^U\Ufp). Nearly all Russian verbs come in pairs or triples, so their infinitive forms are usually written as imperfective–perfective, or indeterminate–determinate–perfective, respectively. See also *Inflection; Verb, conjugation; Verb, determinate; Verb, imperfective; Verb, indeterminate;* and *Verb, perfective*.

Inflections

ending put on words to indicate different forms, such as person, number, and tense for verbs, or case, number, and gender for nouns, adjectives, etc. See also *Case; Declension; Number; Person; Tense;* and *Verb, conjugation*.

Interjection
see *Emphatic particle*.

Interrogative
question word, either declinable (e.g., ^fb, kfb, ^U^b½, etc.) or indeclinable (e.g., XYZ, ^bXYU, ^U^, etc.) See also *Declinable* and *Indeclinable*.

L

Lexicon
vocabulary of the program, with each entry's grammatical information and translation equivalents. See also *Attribute (code), primary; Attribute (code), secondary attribute; Source association;* and *Target*.

M

Mood, indicative
condition of a verb dealing with statements, assertions, and actual events (e.g., The dog never *bites*). See also *Mood, subjunctive;* and *Tense, present indicative*.

Mood, subjunctive
condition of a verb dealing with uncertainties, suppositions, and hypothetical statements (e.g., I doubt the dog *would bite*). See also *Mood, indicative*.

N

Negation
process of making a statement false, or eliminating the presence or existence of something (e.g., The dog *does not* bite, or, there *is no* mall in this town).

Number, plural
verb conjugation where there is more than one subject (e.g., we *are*, people *talk*). See also *Verb, conjugation*.

Number, singular
verb conjugation where there is just one subject (e.g., I *am*, the program *works*). See also *Verb, conjugation*.

O

Object, direct
see *Case, accusative (direct object)*.

Object, indirect
see *Case, indirect (indirect object)*.

P

Part of speech
grammatical role a word plays (e.g., noun, adjective, adverb, verb, etc.) See Appendix Table B-2 for complete encoding details.

Particle
word or short phrase (e.g., wow, holy cow, [Z, g [, etc.) used to intensify or add flavor to nearby words, or used in semi-idiomatic roles (to give *up*).

Past participle
verb used with the present and past perfect tenses, and adverbial participle expressions (e.g., The author has/had *written* a novel). See also *Tense, Verbal adjective,* and *Verbal adverb*.

Pattern (code)
unique set of inflections which are appended to the stem of a word to generate all its forms. See Appendix C for complete encoding details. See also *Declension; Inflection; Stem;* and *Verb, conjugation*.

Person
verb conjugation referring to the subject: *I/we* is first person, *you* second, and *he/she/it* third. See also *Verb, conjugation*.

Predicate
portion of a sentence not devoted to the subject. In the example, *the dog chased the cat all over the yard*, the subject is *the dog*, and the rest is the predicate. See also *Adjective, predicate*.

Preposition
word (non-verb) or group of words that governs the case of a noun and any words modifying it (e.g., *behind* the house, *under* the table, *in* the car, *on account of* rain, etc.) See also *Case, Government, Number,* and *Declension*.

Pronoun, demonstrative
determiner that references a specific item among more than one (e.g., *This* house, or *that* car, or *this* computer is faster than *that* one).

Pronoun, indefinite
word serving to replace an unknown noun or noun phrase (e.g., somebody, something, anybody, anything, etc.)

Pronoun, personal
word serving to replace a noun or noun phrase already mentioned (e.g., he, she, it, they, etc.) See also *Agreement, Case, Person,* and *Antecedent*.

Pronoun, possessive

noun modifier indicating ownership (e.g., my, your, his, her, etc.)

Pronoun, relative

word used to bind two clauses with a shared subject or object. English uses *who(m)* and *which (that)*, for people and things, respectively. Russian combines these into a single word, ^bfbdo½. See also *Verbal adjective*.

R

Reflexive, active

Russian reflexive verb which performs an action only on a non-accusative object. For example: gkUfpes dgee^b`g s\o^g (to study the Russian language). See also *Verb, non-reflexive;* and *Verb, reflexive*.

Reflexive, passive

Russian reflexive verb which performs an action on its subject(s) and has no objects (e.g., to wash *oneself*). See also *Verb, non-reflexive;* and *Verb, reflexive*.

S

Source abstract

Russian abstract after conversion to the character set used internally by the program (Native Format or KOI-8).

Source association

general description of a Russian word's context used to disambiguate its translation (i.e., person, place, thing, animal, or figurative/idiomatic). More than one can be assigned to a word (e.g., *canyon* is both a place and a thing). See Appendix Table B-3 for complete encoding details. See also *Disambiguation Expression* and *Transfer attribute*.

Stem

base part of a word as it appears in the bilingual lexicon. By appending the appropriate inflection patterns to the stem, any form of the word can be generated. See also *Inflection* and *Pattern*.

Symbolic constant

concise encoded word description generated from the available syntactic information. For example: **VJ_PA_G_M_I** represents a present active verbal adjective modifying a masculine noun in the genitive case and itself demanding an instrumental object (e.g., W_UYZ r mZXb). See Appendix Table B-10 for complete encoding details. See also *Verbal adjective*.

T

Target (noun, adjective, etc.)
English translation of a Russian word. See also *Transfer*.

Target article (code)
valid article usage for a target noun (i.e., a, an, or the). Used mainly to eliminate non-existent combinations (e.g., *The* Europe, *An* electricity, etc.) Russian has no articles; therefore it is difficult for the program (and Russians!) to use the correct article(s) in English. See Appendix Table B-3 for complete encoding details. See also *Article,* and *Transfer attribute*.

Target plural (code)
inflection pattern appended to the stem of an English word to make it plural. The most common patterns are *-s* or *-es*. See Appendix Table B-3 for complete encoding details. See also *Transfer attribute*.

Tense, imperfect indicative
simple past tense form of a verb (e.g., The dog *ate* my thesis).

Tense, present indicative
simple present tense form of a verb, used mainly for making statements (e.g., The dog *sleeps* all day).

Tense, present perfect
compound past tense form an English verb (e.g., The dog *has eaten* my thesis). See also *Past participle*.

Tense, progressive
verbal form conveying the meaning of an on-going process (e.g., The *dog has been chasing* its tail for a week now). Also known as the continuous tense.

Transfer
process by which the English target is generated from the Russian source. See also *Source* and *Target*.

Transfer attribute (code)
packed description of the context and syntax of a source and target word. See also *Source association*, *Target article*, and *Target plural*.

Transfer engine
part of the program which decides how to translate a sentence from Russian to English.

Transfer expression
modified regular expression describing how a sentence is structured. See also *Transfer rule*.

Transfer rule
set of transfer expressions combined to form the grammatical description of how to translate a sentence. See also *Transfer expression*.

Transfer rule base
ordered compilation of all uncompressed transfer rules which can be applied to translate a Russian document. See also *Transfer rule*.

Translation details
step-by-step account of the information extracted and the decisions made by the transfer engine in translating a Russian document. See also *Transfer Engine*.

Transliteration
transcription of the letters from one alphabet to another. See also *Import filter*.

V

Verb, auxiliary
verb used to indicate the voice, tense, or mood of another verb that cannot do it itself. In the example, *Lena will go to the shoe store*, the verb *to go* has no forms of its own by which a future tense can be indicated. For simple past tense, a form does exist (i.e., went); however, a form of the auxiliary verb *to have* is necessary to indicate present perfect tense (i.e., Lena *has* gone to the shoe store). See also *Verb, main*.

Verb, conjugation
ending put on verbs to indicate person, number, and tense. See also *Inflection, Number, Person,* and *Tense*.

Verb, determinate
Russian verb of locomotion conveying the meaning of motion in progress. See also *Tense, progressive; Verb, indeterminate;* and *Verb, perfective*.

Verb, imperative
verb form used for commands.

Verb, imperfective
Russian verb conveying the idea of a repeated, habitual, or general action, or an action that lasts for a specified duration. For example: For three days the dog did not eat (i.e., was not eating). See also *Verb, perfective*.

Verb, indeterminate
form of Russian verbs of locomotion conveying at least four meanings: general ability (e.g., the baby can walk), general or random motion, habitual motion (e.g., commuting), or a single round trip in the past tense. See also *Verb, determinate; Verb, imperfective;* and *Verb, perfective*.

Verb, intransitive
verb which cannot perform an action on any direct object (e.g., to sleep). See also *Verb, transitive*.

Verb, main
verb that does not function as an auxiliary verb. See *Verb, auxiliary*.

Verb, perfective
Russian verb used to indicate a simple action which comes to an end: For example: I read (i.e., finished) the book. See also *Verb, imperfective*.

Verb, support
verb which conveys very little or no meaning. For example, in *to take a walk*, *take* has nothing to do with gaining possession of something. Also known as light verbs.

Verb, reflexive
verb which performs an action on its subject(s). All Russian reflexive verbs end in -es or -ep. For example: `ofpes (to wash *oneself*). See also *Reflexive*.

Verb, transitive
verb which performs an action on an object (e.g., to write a letter). See also *Verb, intransitive*.

Verbal adjective, past active
single Russian word substitute for a past active relative clause construction. For example: The man who was photographing/photographed the dog... In Russian *who was photographing/photographed* can be rendered by a single word derived from the verb *to photograph* (i.e., hbfbXdUh]dbWUWl]½/ehbfbXdUh%]dbWUWl]½).

Verbal adjective, past passive
single Russian word substitute for a past passive relative clause construction. For example: The dog which was photographed/being photographed by the man... In Russian *which was photographed/being photographed* can be rendered by a single word derived from the verb *to photograph* (i.e., ehbfb%XdUh]dbWUaaUs/hbfbXdUh]dbWUaaUs).

Verbal adjective, present active

single Russian word substitute for a present active relative clause construction. For example: The man who is photographing the dog... In Russian *who is photographing* can be rendered by a single word derived from the verb *to photograph* (i.e., hbfbXdUh]dgrm]½).

Verbal adjective, present passive

single Russian word substitute for a present passive relative clause construction. For example: The dog which is being photographed by the man... In Russian *which is being photographed* can be rendered by a single word derived from the verb *to photograph* (i.e., hbfbXdUh]dgZ`Us).

Verbal adverb, sequential

single Russian word substitute for an adverbial clause describing a sequence of events. For example: Having buried the bone, the dog fell asleep. In Russian, *having buried* can be rendered by a single word derived from the verb *to bury* (i.e., \U^bcUW or \U^bcUWl]).

Verbal adverb, simultaneous

single word substitute for an adverbial clause describing two events occurring at the same time. For example: While burying the bone, the dog often barked. In Russian, *while burying* can be rendered by a single word derived from the verb *to bury* (i.e., \U^UcoWUs).

Voice, active

verb construction where the subject is performing the action (e.g., The dog is eating the bone). See also *Voice, passive*.

Voice, passive

verb construction when the action is being performed by the subject, not necessarily named (e.g., The bone is being eaten [by the dog]). See also *Voice, active*.

APPENDIX A

User's Manual

File Menu

Open

Opens a Russian abstract or e-mail message. Supported character mappings are automatically converted to the program format.

How to...

Open an abstract

1. Make sure 'Open Abstract' appears on the title bar. If it does not, click CANCEL and choose Configure from the Options menu to switch to Abstract Mode.
2. Select the drive, directory, and document filename.
3. Click OK.
4. Wait while the default import filter is applied to the abstract. On slower computers, this may take 30 seconds or more. Progress is continually updated graphically in the lower-right corner of the screen.

   If the abstract cannot be converted with the default import filter, the Manual Filter Selection dialog appears. Select any of the filters listed and click OK. If it is unclear which filter(s) to choose, click SELECT ALL and the correct filter, if it is defined, is applied automatically. If none of the selected filters succeeds in converting the abstract, a new import filter must be created (See Import Filter Editor on the Edit menu).

Open an e-mail message

1. Make sure 'Open E-Mail Message' appears on the title bar. If it does not, click CANCEL and choose Configure from the Options menu to switch to E-Mail Mode.
2. Select the drive, directory, and document filename.
3. Click OK.
4. Wait while the default import filter is applied to the e-mail message. On slower computers, this may take 30 seconds or more. Progress is continually updated graphically in the lower-right corner of the screen.

Commands

Open Abstract Dialog

OK

Opens the currently selected document and attempts to convert its character mappings to the program format.

CANCEL

Aborts trying to open a document.

HELP

Displays help on opening documents.

FILENAME

Enter the name of the Russian document to open, or click on any of the files listed.

LIST FILES OF TYPE

All Files (.*)*

Lists all files in the current directory.

Source Abstract Files (.ABS)*

Lists all files with the `.ABS` extension in the current directory. Source abstracts are text files in any supported Cyrillic character set.

Converted Abstract Files (.CVT)*

Lists all files with the `.CVT` extension in the current directory. Converted abstracts are text files in the KOI-8 character set.

E-Mail Message Files (.TXT)*

Lists all e-mail message files with the `.TXT` extension in the current directory. The default import filter for e-mail messages is KOI-8.

Text Files (.TXT)*

Lists all text files with the `.TXT` extension in the current directory.

DRIVES

Lists the available drives.

DIRECTORIES

Lists the directories on the current drive.

Manual Filter Selection Dialog

FILTER LIST

Shows all available import filters. More than one can be selected.

OK

Applies the selected import filter or filters to the current document. At least one import filter must be selected.

CANCEL

Aborts trying to apply any import filter to the current document. It cannot be read in or translated without the appropriate import filter.

HELP

Displays help on manually selecting an import filter.

SELECT ALL

Selects all available import filters. Use this when it is unclear which import filter(s) to apply to the current document.

Close

Closes the current document and any associated windows (e.g., transliteration, translation details, etc.). It prompts to save any unsaved changes.

Save

Individually saves components of the current document:

- TRANSLATION

Saves the current translation.

- TRANSLITERATION

Saves the current transliteration.

- CONVERTED ABSTRACT

Saves the current abstract in the KOI-8 character set.

- E-MAIL MESSAGE

Saves the current e-mail message in the KOI-8 character set.

- TRANSLATION DETAILS

Saves a step-by-step account of information extracted and decisions made by the program during translation of the current document.

Printer Setup

Sets the default printer and its configurations for paper source, paper size, page orientation, etc.

Print

Individually prints components of the current document:

- TRANSLATION

Prints the current translation.

- TRANSLITERATION

Prints the current transliteration.

- CONVERTED ABSTRACT

Prints the current abstract in Russian.

- **E-MAIL MESSAGE**

Prints the current e-mail message in Russian.

- **TRANSLATION DETAILS**

Prints a step-by-step account of information extracted and decisions made by the program during translation of the current document.

Exit

Terminates the program. Its prompts to save any unsaved changes.

Edit Menu

Lexicon Editor

Provides an tool for describing and editing lexicon entries using the specific format and grammatical terms the program understands. For complete details on these terms, see the glossary, appendices, and thesis text.

How to...

Add a new lexicon entry

1. Click **ADD**.
2. Simultaneously press the CONTROL and SHIFT keys to change the keyboard layout to Russian. See Table B-1 for the key assignments.
3. In the *New Stem* field, enter the stem of the Russian entry.
4. Again simultaneously press the CONTROL and SHIFT keys to return the keyboard layout to English.
5. From the *Part of Speech* list, select the part of speech for this entry.
6. From the *Primary Attribute* list, select the primary attributes(s) for this entry. Some parts of speech have no primary attributes.
7. From the *Secondary Attribute* list, select the secondary attributes(s) for this entry. Some parts of speech have no secondary attributes.
8. For nouns, select from the *Associations* list the association(s) for this entry.
9. In the *Disambiguation Expression* field, enter an optional disambiguation expression.
10. Click **NEXT**.
11. In the *Pattern* field, enter the inflection pattern number. Use the up- and down-arrows on the keyboard or click on the arrow buttons to change the forms of the entry. Some inflection patterns are undefined.
12. Click **NEXT**.
13. In the fields shown, enter the English word forms requested. When all requirements are satisfied, the **UPDATE** button becomes available.
14. Click **UPDATE**.

Look up a lexicon entry

1. Click ADD.
2. Simultaneously press the CONTROL and SHIFT keys to change the keyboard layout to Russian. See Table B-1 for the key assignments.
3. In the *Lookup Entry* field, enter legal form of the Russian entry to find.
4. Again simultaneously press the CONTROL and SHIFT keys to return the keyboard layout to English.
5. If only one entry matches the given stem, its details are displayed immediately. The entry can now be modified or deleted.
6. If more than one entry matches the given stem, the Lookup Results dialog appears. From either the *Russian Entry* or *English Entry* list, select the unique entry to look up. Entries with inflections have ... appended. Inflections on ambiguous matches are displayed in square brackets.
7. Click OK.
8. The entry can now be modified or deleted.

Modify a lexicon entry

1. Follow the steps in 'Look up a lexicon entry' above.
2. Change any of the details for this entry. When all requirements are satisfied, the UPDATE button becomes available.
3. Click UPDATE.

Remove a lexicon entry

1. Follow the steps in 'Look up a lexicon entry' above.
2. Click DELETE.
3. Click YES.

Commands

Lexicon Editor

ADD

Adds a new entry to the lexicon.

LOOKUP

Searches the lexicon for the given entry and, if found, returns its record. Lookups can be done on any legal form of an entry.

UPDATE

Completes the current operation and commits any changes.

DELETE

Removes the current lexicon entry.

CANCEL

Aborts the current operation and discards any changes.

NEXT

Opens the next folder.

BACK

Opens the previous folder.

CLOSE

Updates the lexicon, regenerates its index, and exits.

HELP

Displays help on editing lexicon entries.

Lookup Results Dialog

OK

Looks up the selected entry.

CANCEL

Aborts the current lookup.

HELP

Displays help on selecting from multiple lexicon entries.

Transfer Rule Editor

Provides an editing environment for assembling the transfer rules used to translate Russian sentences to English. See Table B-9 for details on the syntax of transfer rules.

How to...

Add a new transfer rule

1. Click ADD.
2. In the *Description* field, enter a brief description of the transfer rule.
3. In the *Russian* field, enter Russian expression of the transfer rule.
4. In the *English* field, enter English expression of the transfer rule.
5. Click OK.

Modify an existing transfer rule

1. Select from any list the transfer rule to modify.
2. Click EDIT.
3. Modify the description and Russian and English expressions.
4. Click OK.

Determine the symbolic constant(s) for a word

1. Click **LOOKUP**.
2. In the *Lookup* field, enter the word form in question.
3. All the symbolic constants matching this word and its inflections appear in the *Symbolic Constants* list. Double-click on any entry to paste it into the Russian expression of the transfer rule.

Copy an existing transfer rule

1. Select from any list the transfer rule to copy.
2. Click **COPY**.
3. Click **EDIT** to modify the copied transfer rule.

Remove an existing transfer rule

1. Select from any list the transfer rule to remove.
2. Click **DELETE**.
3. Click **YES**.

Commands

ADD

Adds a new transfer rule.

EDIT

Modifies the currently selected transfer rule.

DELETE

Removes the currently selected transfer rule.

COPY

Copies the currently selected transfer rule.

CLOSE

Saves any changes and exits.

CANCEL

Exits without saving any changes.

LOOKUP

Shows the symbolic constants for the lookup entry.

HELP

Displays help on editing transfer rules.

Import Filter Editor

Import filters are used to convert documents written with different Cyrillic character sets to the native program format. This editor allows the user to define and modify how characters sets are mapped.

How to...

Select a default import filter

1. Select the default import filter from the *Default Filter* list.
2. Click **OK**.

Create a new filter

1. From the *Default Filter* list, select a filter similar to the one to create. Filters usually have many mappings in common (e.g., numbers and punctuation), so the quickest and easiest way to create a new filter is to base it on a similar existing one.
2. Click **SAVE AS**.
3. In the *New Filter Name* field, enter the name of the new filter.
4. Click **OK**.
5. Define the mappings according to 'Change how a filter is defined' below.

Change how a filter is defined

1. From the *Default Filter* list, select the import filter to change.
2. Click the *Alphabet* or *Miscellaneous* folder tab to display its table of character mappings.
3. Click any of the mapping fields.
4. Enter or modify a mapping (see the Chapter 4 for format details). Use the CONTROL key plus the arrow keys to move between mappings.
5. Repeat steps 1 through 4 until all mappings are defined. Unused mappings must be assigned a zero.

Copy a filter

1. From the *Default Filter* list, select the filter to copy.
2. Click **SAVE AS**.
3. In the *New Filter Name* field, enter the name of the copy.
4. Click **OK**.

- The copy automatically becomes the default filter. To change the default filter to another filter, see 'Select a default import filter' above.

Rename a filter

1. From the *Default Filter* list, select the filter to rename.

2. Click SAVE AS.
3. In the *New Filter Name* field, enter the new name of the filter.
4. Click OK.
5. From the *Default Filter* list, select the old filter.
6. Click DELETE.
7. Click YES.

- No default filter is defined after a rename. To set the default filter, see 'Select a default import filter' above.

Remove a filter

1. From the *Default Filter* list, click on the filter to remove.
2. Click DELETE.
3. Click YES.

- No default filter is defined after a delete. To set the default filter, see 'Select a default import filter' above.

- Although deleting a filter removes it immediately from the list of available filters, changes do not become permanent until OK is clicked. If a filter is accidentally deleted, click CANCEL. All other changes made during this filter setup session are also discarded.

- Native Format (KOI-8), Phonetic Unique, Transliteration, and KOI-7 are built-in filters; they cannot be modified or deleted.

Commands

SAVE AS

Makes a copy of the default filter and assigns the copy a new name.

DELETE

Removes the default filter. A new default filter needs to be selected.

OK

Saves any changes and exits.

CANCEL

Exits without saving any changes.

HELP

Displays help on editing import filters.

AUTO-MAP

This function is not implemented in the current version. Its intended purpose is to determine the filter mapping automatically for unknown character sets.

DEFAULT FILTER

Shows the default filter automatically applied to all documents read in. The list shows all available filters.

ALPHABET

Upper Case and *Lower Case* are mapping tables of all uppercase and lowercase alphabetic characters, respectively.

MISCELLANEOUS

Special Characters is a mapping table of all numeric characters, plus punctuation, space, etc.

Extended Characters are mappings reserved for future expansion.

Cleanup Filter Editor

Provides a simple editor for commonly mistranslated expressions and their corrections.

How to...

Add a new filter

1. Click **ADD**.
2. In the *Malformed Expression* field, enter the incorrect expression exactly as it appears after translation.
3. In the *Corrected Expression* field, enter the properly constructed expression.
4. Click **OK**.

Modify an existing filter

1. Select from either list the expression to modify.
2. Click **EDIT**.
3. Modify the malformed and corrected expressions.
4. Click **OK**.

Remove an existing filter

1. Select from either list the expression to remove.
2. Click **DELETE**.
3. Click **YES**.

Commands

ADD

Adds a new expression to the list.

EDIT

Modifies the currently selected expression.

DELETE

Removes the currently selected expression.

CLOSE

Saves any changes and exits.

CANCEL

Exits without saving any changes.

HELP

Displays help on editing cleanup filters.

Post-Edit Translation

Opens the default external word processor to post-edit the current translation. See Configure on the Options menu for details on how to change the default word processor.

Options Menu

Translate

Translates the current document into English.

Transliterate

Converts the current document from the Cyrillic to the Latin alphabet.

How to...

Transliterate a document

1. Choose Transliterate from the Options menu.

Change the transliteration filter

1. Choose Configure from Options the menu.
2. From the *Transliteration Filter* list, select a new transliteration filter.
3. Click OK.

Transliterate and send an e-mail message written in Cyrillic

1. Write the e-mail message using any word processor supporting Cyrillic characters (Microsoft Word® 6.0 with a KOI-8 font works well).
2. Save the e-mail message as 'Text only with line breaks.' If this is not available, save it as plain 'ASCII Text.'

3. Choose Open from the File menu.
4. Choose Transliterate from the Options menu.
5. Click the right mouse button on the Transliteration window. From the pop-up menu, choose SelectAll.
6. Again click the right mouse button on the Transliteration window. From the pop-up menu, choose Copy.
7. Paste the transliteration directly into the e-mail document. This procedure varies according the e-mail and terminal software used. Consult their user's manuals for help.

Summary Info

Displays information and performance statistics on the current abstract and the translation system. Most statistics are available only after the current abstract has been translated.

Filename

Shows the filename of the current abstract.

Created

Shows the time and date of the last modification to the current abstract.

Import Filter

Shows the import filter used to convert the current abstract.

Lookup Hits

Shows how many words in the current abstract were found in the lexicon.

Transfer Hits

Shows how many sentences of the current abstract were translated.

Transfer Time

Shows the total time taken to translate the current abstract.

Average Transfer Time

Shows the average time taken to translate a single sentence in the current abstract.

Lexicon Entries

Shows the number of entries in the lexicon.

Transfer Rules

Shows the number of uncompressed transfer rules defined.

Configure

Allows default system configurations and interface preferences to be modified.

How to...

Set the post-editor word processing package

1. In the *Application Title* field, enter the name of the word processor to use as the post-editor (e.g., Microsoft Word® 6.0, Word Perfect® 5.0, etc.).
2. In the *Application Filename* field, enter the full path and filename of the word processor's executable file.
3. In the *Hot-Key Paste Sequence* field, enter the string of characters used to paste clipboard contents into the selected word processor. For most Windows® applications, use ^v.

- Control keys are indicated by +, ^, and %, for SHIFT, CONTROL, and ALT, respectively. Function keys are entered in braces: {F1}, {F2}, etc.

Commands

OK

Saves any changes and exits.

CANCEL

Exits without saving any changes.

HELP

Displays help on modifying system configurations.

Operation Mode

Abstract

Sets the primary document type to abstracts. Strict error checking is enforced on abstracts since unknown characters, misspellings, or formatting problems prevent translation.

E-Mail

Sets the primary document type to e-mail messages. These documents are intended to be transliterated, possibly edited, and pasted into an outgoing e-mail document. Error checking is minimal.

E-Mail messages are expected to be in KOI-8 (Native) format.

Interface

Status Bar

Shows or hides the status bar at the bottom of the main interface screen.

Passive Help

Shows or hides the passive help text on the status bar.

Transfer Log

Enables or disables the log of translation details. The translation process is marginally faster when the log is disabled.

Post-Editor Configuration

Application Title

Name of the post-editor word processing software.

Application Filename

Full path and filename of the post-editor's executable file.

Hot-Key Paste Sequence

String of hot-keys used to paste clipboard contents into the post-editor.

Transliteration Filter

Filter used to transliterate Cyrillic characters to Latin.

Window Menu

Transfer Details

Displays a step-by-step account of information extracted and decisions made by the program during translation.

Cascade

Sets all the currently open windows to the same size, then stacks them offset on top of each other starting at the top-left corner of the workspace. The currently active window appears on top.

Tile Horizontally

Resizes and repositions all the currently open windows so that they each occupy the entire workspace height, but do not overlap horizontally. If only one window is open, it occupies the entire workspace area.

Tile Vertically

Resizes and repositions all the currently open windows so that they each occupy the entire workspace width, but do not overlap vertically. If only one window is open, it occupies the entire workspace area.

Arrange Icons

Aligns the icons of all minimized windows into a single row at the bottom of the workspace.

Help Menu

Contents

Brings up the table of contents of this user's manual.

Search For Help On

Searches this user's manual for a specific word or phrase.

About Scientific Abstract Translator

Displays general information on this program.

APPENDIX B

Encoding Tables

Table B-1 Character Set Mappings

Cyrillic Alphabet[1]		Built-In Import Filters				
Character	Pronunciation	Native Format[2]	Phonetic Unique	KOI-7	Transliteration[3]	Keyboard Assignment[4]
á Á	far	E1 C1	A a	A a	A a	A a
â Â	bat	E2 C2	B b	B b	B b	B b
÷ ×	vat	F7 D7	V v	W w	V v	V v
ç Ç	goose	E7 C7	G g	G g	G g	G g
ä Ä	dog	E4 C4	D d	D d	D d	D d
å Å	yet	E5 C5	E e	E e	E e	E e
³ £	yoke	B3 A3	YO yo	# $	Jo jo	{ [
ö Ö	pleasure	F6 D6	ZH zh	V v	Zh zh	}]
ú Ú	zero	FA DA	Z z	Z z	Z z	Z z
é É	tree	E9 C9	I i	I i	I i	I i
ê Ê	boy	EA CA	J j	J j	J j	J j
ë Ë	kangaroo	EB CB	K k	K k	K k	K k
ì Ì	ball	EC CC	L l	L l	L l	L l
í Í	monkey	ED CD	M m	M m	M m	M m
î Î	north	EE CE	N n	N n	N n	N n
ï Ï	open/October	EF CF	O o	O o	O o	O o
ð Ð	pencil	F0 D0	P p	P p	P p	P p
ò Ò	trust[5]	F2 D2	R r	R r	R r	R r
ó Ó	circle	F3 D3	S s	S s	S s	S s
ô Ô	town	F4 D4	T t	T t	T t	T t
õ Õ	stool	F5 D5	U u	U u	U u	U u
æ Æ	fish	E6 C6	F f	F f	F f	F f
è È	hockey[6]	E8 C8	KH kh	H h	X x	X x
ã Ã	cats	E3 C3	TZ tz	C c	Ts ts	S s
þ Þ	chicken	FE DE	CH ch	^ ~	Ch ch	H h
û Û	fish	FB DB	SH sh	[{	Sh sh	W w
ý Ý	fresh-cheese	FD DD	SHCH shch	] }	Shch shch	Q q
ÿ ß	no sound[7]	FF DF	` `	_ "	` `	" '
ù Ù	chop suey[8]	F9 D9	Y y	Y y	Y y	Y y
ø Ø	no sound[7]	F8 D8	' '	X x	' '	: ;
ü Ü	let	FC DC	EH eh	\ |	E e	_ -
à À	unit	E0 C0	YU yu	@ `	Ju ju	+ =
ñ Ñ	yacht	F1 D1	YA ya	Q q	Ja ja	| \

[1] Non-alphabetic characters generally need no conversion table.

[2] Hexidecimal representation of the KOI-8 font mapping.

[3] Serves as the default export filter for transliteration.

[4] Non-standard keyboard layout.

[5] Rolled or trilled as a Scot would.

[6] Coughed mildly from the back of the throat.

[7] a) Enhances the sound of the preceding letter.
b) Capital forms are used only in titles.

[8] Uttered quickly as a sea lion would.

Table B-2 Part of Speech Encoding

Part of Speech		Primary Attributes		Secondary Attributes	
Code	Description	Code†	Description	Code†	Description
1	Noun, inanimate	1	Masculine	1	Demands genitive
2	Noun, animate	2	Feminine	2	Demands dative
		3	Neuter	4	Demands instrumental
		4	Plural		
3	Pronoun, personal	1	1st person, singular		
		2	2nd person, singular		
		3	3rd person, singular, masculine		
		4	3rd person, singular, feminine		
		5	3rd person, singular, neuter		
		6	1st person, plural		
		7	2nd person, plural		
		8	3rd person, plural		
4	Pronoun, possessive	1	Declinable (Part of Speech Code 8, Primary Attribute 1)		
		2	3rd person, singular, masculine *or* neuter, indeclinable		
		3	3rd person, singular, feminine, indeclinable		
		4	3rd person, plural, indeclinable		
5	Pronoun, indefinite	1	Masculine		
		3	Neuter		
6	Pronoun, relative				
7	Pronoun, demonstrative				
8	Verb	1	Main	1	Demands nominative
		2	Auxiliary	2	Demands genitive
		4	Imperfective	4	Demands dative
		8	Perfective	8	Demands accusative
		16	Non-reflexive	16	Demands instrumental
		32	Reflexive, active		
		64	Reflexive, passive		
		128	Indeterminate (left-column verb of locomotion)		
		256	Determinate (middle-column verb of locomotion)		

Table B-2 Part of Speech Encoding, Continued

Part of Speech		Primary Attributes		Secondary Attributes	
Code	Description	Code'	Description	Code'	Description
9	Adjective	1	Normal, long form	1	Demands genitive
		2	Normal, short form	2	Demands dative
		3	Comparative	4	Demands instrumental
		4	Superlative (simple)		
10	Adverb	1	Normal	1	Demands genitive
		2	Comparative	2	Demands dative
		3	Superlative	4	Demands instrumental
11	Preposition			1	Demands genitive
				2	Demands dative
				4	Demands accusative
				8	Demands instrumental
				16	Demands prepositional
12	Interrogative	1	Masculine		
		2	Neuter		
		3	Indeclinable		
13	Interjection or Emphatic particle	1	Retain		
		2	Ignore		
14	Verbal adverb	1	Non-reflexive	1	Demands nominative
		2	Reflexive, active	2	Demands genitive
		4	Reflexive, passive	4	Demands dative
		8	Imperfective	8	Demands accusative
		16	Perfective	16	Demands instrumental

Table B-2 Part of Speech Encoding, Continued

Part of Speech		Primary Attributes		Secondary Attributes	
Code	Description	Code†	Description	Code†	Description
15	Verbal adjective	1	Non-reflexive	1	Demands nominative
		2	Reflexive, active	2	Demands genitive
		4	Reflexive, passive	4	Demands dative
		8	Present active	8	Demands accusative
		16	Present passive	16	Demands instrumental
		32	Past active		
		64	Past passive		
		128	Imperfective		
		256	Perfective		
16	Conjunction				
17	Fixed expression				

† Combinations within a block are represented by summing the applicable codes.

Table B-3 Transfer Attribute Encoding

Category	Hex Code†	Description	Example
Source Associations	1	Person	
	2	Place	
	4	Thing	
	8	Animal	
	10	Figurative or Idiomatic	
Target Articles	20	None	
	40	A	
	80	An	
	1000	The	
Target Plurals	200	Plural same or not used	Species ⇒ Species
	300	Append 's'	Dog ⇒ Dogs
	400	Append 'es'	Dish ⇒ Dishes
	500	Drop last letter; append 'ies'	Party ⇒ Parties
	600	Drop last letter; append 'ves'	Leaf ⇒ Leaves
	700	Drop last 2 letters; append 'ves'	Knife ⇒ Knives
	800	Drop last 2 letters; append 'a'	Criterion ⇒ Criteria
	900	Drop last 2 letters; append 'i'	Alumnus ⇒ Alumni
	A00	Drop last 2 letters; append 'en'	Woman ⇒ Women
	B00	Drop last 4 letters; append 'ice'	Mouse ⇒ Mice
	C00	Drop last 4 letters; append 'eese'	Goose ⇒ Geese

† Combinations are represented by summing the applicable codes.

Table B-4 Target Verb Encoding

Part of Speech	Type	Part of Speech Code	Primary Attribute Code†	Encoding Scheme
Verb	Imperfective, non-reflexive	8	20	Present Indicative 2nd Person • Present Indicative 3rd Person • Imperfect Indicative • Progressive • Past Participle
	Perfective, non-reflexive	8	24	Present Indicative 2nd Person • Imperfect Indicative • Past Participle
	Active Reflexive	8	33	Present Indicative 2nd Person • Present Indicative 3rd Person • Imperfect Indicative • Progressive • Past Participle
	Passive Reflexive	8	65	Past Participle
Verbal Adverb	Non-reflexive	14	1	Progressive • Past Participle
	Active Reflexive	14	2	Progressive • Past Participle
	Passive Reflexive	14	4	Past Participle
Verbal Adjective	Non-reflexive	15	1	Present Indicative 2nd Person • Present Indicative 3rd Person • Imperfect Indicative • Progressive • Past Participle
	Active Reflexive	15	2	Present Indicative 2nd Person • Present Indicative 3rd Person • Imperfect Indicative • Progressive • Past Participle
	Passive Reflexive	15	4	Past Participle

† Bitwise logical OR.

Table B-5 Target Verb Constructions

Part of Speech	Type	Form	Aspect	Transfer Expression[†]
Verb	Any	Infinitive	Any	'to' + Present Indicative 2nd Person
	Non-reflexive	Present	Imperfective	a) {'am'|'is'|'are'} + Progressive b) {'has'|'have'} + 'been' + Progressive c) {Present Indicative 2nd Person|Present Indicative 3rd Person}
			Indeterminate	{Present Indicative 2nd Person|Present Indicative 3rd Person}
			Determinate	{'am'|'is'|'are'} + Progressive
		Past	Imperfective	{'was'|'were'} + Progressive
			Indeterminate	Imperfective Indicative
			Determinate	{'was'|'were'} + Progressive
			Perfective	a) {'has'|'have'} + Past Participle b) Imperfect Indicative
		Future	Imperfective or Indeterminate	a) 'will be' + Progressive b) 'will' + Present Indicative 2nd Person
			Determinate	'will be' + Progressive
			Perfective	'will' + Present Indicative 2nd Person
	Active Reflexive	Present	Imperfective	a) {'am'|'is'|'are'} + Progressive b) {'has'|'have'} + 'been' Progressive c) {Present Indicative 2nd Person|Present Indicative 3rd Person}
		Past	Imperfective	{'was'|'were'} + Progressive
			Perfective	a) {'has'|'have'} + Past Participle b) Imperfect Indicative
		Future	Imperfective	a) 'will be' + Progressive b) 'will' + Present Indicative 2nd Person
			Perfective	'will' + Present Indicative 2nd Person
	Passive Reflexive	Present	Imperfective or Determinate	a) {'am'|'is'|'are'} + Past Participle b) {'am'|'is'|'are'} + 'being' + Past Participle
		Past	Imperfective or Determinate	a) {'was'|'were'} + Past Participle b) {'was'|'were'} + 'being' + Past Participle
			Perfective	{'was'|'were'} + Past Participle
		Future	Imperfective or Perfective	'will be' + Past Participle

Table B-5 Target Verb Constructions, Continued

Part of Speech	Type	Form	Aspect	Transfer Expression[†]
Verbal Adverb	Non-reflexive	Simultaneous	Imperfective	'while' + Progressive
		Sequential	Imperfective	'having been' + Progressive
			Perfective	'having' + Past Participle
	Active Reflexive	Simultaneous	Imperfective	'while' + Progressive
		Sequential	Imperfective	'having been' + Progressive
			Perfective	'having' + Past Participle
	Passive Reflexive	Simultaneous	Imperfective	'while being' + Past Participle
		Sequential	Imperfective or Perfective	'having been' + Past Participle
Verbal Adjective	Non-reflexive	Present Active	Imperfective	a) Progressive b) {'which'|'who'|'whom'} + {'is'|'are'} + Progressive c) {'which'|'who'|'whom'} + {'has'|'have'} + 'been' + Progessive d) {'which'|'who'|'whom'} + {Present Indicative 2nd Person|Present Indicative 3rd Person}
		Past Active	Imperfective	{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
			Perfective	a) {'which'|'who'|'whom'} + Imperfect Indicative b) {'which'|'who'|'whom'} + {'has'|'have'} + Past Participle
	Active Reflexive	Present Active	Imperfective	a) Progressive b) {'which'|'who'|'whom'} + {'is'|'are'} + Progressive c) {'which'|'who'|'whom'} + {'has'|'have'} + 'been' + Progessive d) {'which'|'who'|'whom'} + {Present Indicative 2nd Person|Present Indicative 3rd Person}
		Past Active	Imperfective	{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
			Perfective	a) {'which'|'who'|'whom'} + Imperfect Indicative b) {'which'|'who'|'whom'} + {'has'|'have'} + Past Participle
	Passive Reflexive	Present Active	Imperfective	{'which'|'who'} + {'is'|'are'} + 'being' + Past Participle
		Past Active	Imperfective	{'which'|'who'} + {'was'|'were'} + 'being' + Past Participle
			Perfective	{'which'|'who'} + {'was'|'were'} + Past Participle
	Passive	Present Passive	Imperfective	a) 'being' + Past Participle b) {'which'|'who'} + {'is'|'are'} + 'being' + Past Participle
		Past Passive	Imperfective	a) 'being' + Past Participle b) {'which'|'who'} + {'was'|'were'} + 'being' + Past Participle
			Perfective	a) Past Participle b) {'which'|'who'} + {'was'|'were'} + Past Participle

Table B-5 Target Verb Constructions, Continued

Part of Speech	Type	Form	Aspect	Transfer Expression†
Relative Pronoun	Non-reflexive	Present	Imperfective	a) Progressive b) {'which'|'who'|'whom'} + {'am'|'is'|'are'} + Progressive c) {'which'|'who'|'whom'} + {'has'|'have'} + 'been' + Progressive d) {'which'|'who'|'whom'} + {Present Indicative 2nd Person|Present Indicative 3rd Person}
			Indeterminate	{'which'|'who'|'whom'} + {Present Indicative 2nd Person|Present Indicative 3rd Person}
			Determinate	{'which'|'who'|'whom'} + {'am'|'is'|'are'} + Progressive
		Past	Imperfective	{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
			Indeterminate	{'which'|'who'|'whom'} + Imperfective Indicative
			Determinate	{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
			Perfective	a) {'which'|'who'|'whom'} + {'has'|'have'} + Past Participle b) {'which'|'who'|'whom'} + Imperfect Indicative
		Future	Imperfective or Indeterminate	a) {'which'|'who'|'whom'} + 'will be' + Progressive b) {'which'|'who'|'whom'} + 'will' + Present Indicative 2nd Person
			Determinate	{'which'|'who'|'whom'} + 'will be' + Progressive
			Perfective	{'which'|'who'|'whom'} + 'will' + Present Indicative 2nd Person
	Active Reflexive	Present	Imperfective	a) {'which'|'who'|'whom'} + {'am'|'is'|'are'} + Progressive b) {'which'|'who'|'whom'} + {'has'|'have'} + 'been' Progressive c) {'which'|'who'|'whom'} + {Present Indicative 2nd Person|Present Indicative 3rd Person}
		Past	Imperfective	{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
			Perfective	a) {'which'|'who'|'whom'} + {'has'|'have'} + Past Participle b) {'which'|'who'|'whom'} + Imperfect Indicative
		Future	Imperfective	a) {'which'|'who'|'whom'} + 'will be' + Progressive b) {'which'|'who'|'whom'} + 'will' + Present Indicative 2nd Person
			Perfective	{'which'|'who'|'whom'} + 'will' + Present Indicative 2nd Person
	Passive Reflexive	Present	Imperfective or Determinate	a) {'which'|'who'} + {'am'|'is'|'are'} + Past Participle b) {'which'|'who'} + {'am'|'is'|'are'} + 'being' + Past Participle
		Past	Imperfective or Determinate	a) {'which'|'who'} + {'was'|'were'} + Past Participle b) {'which'|'who'} + {'was'|'were'} + 'being' + Past Participle
			Perfective	{'which'|'who'} + {'was'|'were'} + Past Participle
		Future	Imperfective or Perfective	{'which'|'who'} + 'will be' + Past Participle

† Where more than one verbal construction is possible, each has a slightly different connotation.

Table B-6 Lexicon Record Structures

Part of Speech	Field 1	Field 2[1]	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8
Noun	Stem	1,2	Target Noun	Pattern Code	Primary Attribute Code	Secondary Attribute Code	Transfer Attribute Code	Disambiguation Expression
Pronoun, personal	Stem	3	Target Pronoun	Pattern Code	Primary Attribute Code			Disambiguation Expression
Pronoun, possessive	Stem	4	Target Pronoun	Pattern Code	Primary Attribute Code			Disambiguation Expression
Pronoun, indefinite	Stem	5	Target Pronoun	Pattern Code	Primary Attribute Code			Disambiguation Expression
Pronoun, relative	Stem	6	Target Pronoun	Pattern Code				
Pronoun, demonstrative	Stem	7	Target Pronoun	Pattern Code				
Verb	Stem	8	Target Verb Encoding[2]	Pattern Code	Primary Attribute Code	Secondary Attribute Code		Disambiguation Expression
Adjective	Stem	9	Target Adjective	Pattern Code	Primary Attribute Code	Secondary Attribute Code		Disambiguation Expression
Adverb	Full Entry	10	Target Adverb		Primary Attribute Code			Disambiguation Expression
Preposition	Full Entry	11	Target Preposition			Secondary Attribute Code		Disambiguation Expression
Interrogative	Stem	12	Target Interrogative	Pattern Code	Primary Attribute Code			Disambiguation Expression
Emphatic Particle or Interrogative	Full Entry	13	Target Particle		Primary Attribute Code			Disambiguation Expression
Participle, verbal adverb	Stem	14	Target Verb Encoding[2]	Pattern Code	Primary Attribute Code	Secondary Attribute Code		Disambiguation Expression
Participle, verbal adjective	Stem	15	Target Verb Encoding[2]	Pattern Code	Primary Attribute Code	Secondary Attribute Code		Disambiguation Expression
Conjunction	Full Entry	16	Target Conjunction					
Fixed Expression	Full Entry	17	Target Expression					Disambiguation Expression

[1] Part of speech codes from Table B-2.

[2] See Table B-4 for encoding scheme.

Table B-7 Disambiguation Parameters

Parameter	Search Type
' '	Literal string
< >	Association
[ ]	Symbolic constant

Table B-8 Disambiguation Operators

Operator	Function
&	AND
|	OR
%	XOR
~	NOT

Table B-9 Transfer Rule Operators

Operator	Translation Type
V	Verb
A	Article
N	Noun
J	Verbal Adjective
D	Verbal Adverb
%	Straight replacement
~	Ignore term

Table B-10 Symbolic Constants

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol†
Noun		Singular	Masculine		Nominative	NO_N_M_*
					Genitive	NO_G_M_*
					Dative	NO_D_M_*
					Accusative	NO_A_M_*
					Instrumental	NO_I_M_*
					Prepositional	NO_P_M_*
					Locative	NO_L_M_*
			Feminine		Nominative	NO_N_F_*
					Genitive	NO_G_F_*
					Dative	NO_D_F_*
					Accusative	NO_A_F_*
					Instrumental	NO_I_F_*
					Prepositional	NO_P_F_*
			Neuter		Nominative	NO_N_N_*
					Genitive	NO_G_N_*
					Dative	NO_D_N_*
					Accusative	NO_A_N_*
					Instrumental	NO_I_N_*
					Prepositional	NO_P_N_*
		Plural	Any		Nominative	NO_N_P_*
					Genitive	NO_G_P_*
					Dative	NO_D_P_*
					Accusative	NO_A_P_*
					Instrumental	NO_I_P_*
					Prepositional	NO_P_P_*
Pronoun	Personal	Singular	Any	1st person	Nominative	PP_N_1_S
					Genitive	PP_G_1_S
					Dative	PP_D_1_S
					Accusative	PP_A_1_S
					Instrumental	PP_I_1_S
					Prepositional	PP_P_1_S
				2nd person	Nominative	PP_N_2_S
					Genitive	PP_G_2_S
					Dative	PP_D_2_S
					Accusative	PP_A_2_S
					Instrumental	PP_I_2_S
					Prepositional	PP_P_2_S
			Masculine	3rd person	Nominative	PP_N_3_M
					Genitive	PP_G_3_M
					Dative	PP_D_3_M
					Accusative	PP_A_3_M
					Instrumental	PP_I_3_M
					Prepositional	PP_P_3_M
			Feminine	3rd person	Nominative	PP_N_3_F
					Genitive	PP_G_3_F
					Dative	PP_D_3_F
					Accusative	PP_A_3_F
					Instrumental	PP_I_3_F
					Prepositional	PP_P_3_F
			Neuter	3rd person	Nominative	PP_N_3_N
					Genitive	PP_G_3_N
					Dative	PP_D_3_N
					Accusative	PP_A_3_N
					Instrumental	PP_I_3_N
					Prepositional	PP_P_3_N

Table B-10 Symbolic Constants, Continued

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol[1]
		Plural	Any	1st person	Nominative	PP_N_1_P
					Genitive	PP_G_1_P
					Dative	PP_D_1_P
					Accusative	PP_A_1_P
					Instrumental	PP_I_1_P
					Prepositional	PP_P_1_P
				2nd person	Nominative	PP_N_2_P
					Genitive	PP_G_2_P
					Dative	PP_D_2_P
					Accusative	PP_A_2_P
					Instrumental	PP_I_2_P
					Prepositional	PP_P_2_P
				3rd person	Nominative	PP_N_3_P
					Genitive	PP_G_3_P
					Dative	PP_D_3_P
					Accusative	PP_A_3_P
					Instrumental	PP_I_3_P
					Prepositional	PP_P_3_P
	Possessive	Any	Any	Any	Declinable - see Adjective, normal	
		Singular	Masculine	3rd person	Indeclinable	PO_X_B
			Neuter	3rd person		PO_X_B
			Feminine	3rd person		PO_X_F
		Plural	Any	3rd person		PO_X_P
	Indefinite		Masculine		Nominative	NO_N_M_0
					Genitive	NO_G_M_0
					Dative	NO_D_M_0
					Accusative	NO_A_M_0
					Instrumental	NO_I_M_0
					Prepositional	NO_P_M_0
			Neuter		Nominative	NO_N_N_0
					Genitive	NO_G_N_0
					Dative	NO_D_N_0
					Accusative	NO_A_N_0
					Instrumental	NO_I_N_0
					Prepositional	NO_P_N_0
	Relative	Singular	Masculine		Nominative	PR_N_M
					Genitive	PR_G_M
					Dative	PR_D_M
					Accusative Animate	PR_1_M
					Accusative Inanimate	PR_2_M
					Instrumental	PR_I_M
					Prepositional	PR_P_M
			Feminine		Nominative	PR_N_F
					Genitive	PR_G_F
					Dative	PR_D_F
					Accusative Animate	PR_A_F
					Accusative Inanimate	PR_A_F
					Instrumental	PR_I_F
					Prepositional	PR_P_F
			Neuter		Nominative	PR_N_N
					Genitive	PR_G_N
					Dative	PR_D_N
					Accusative Animate	PR_A_N
					Accusative Inanimate	PR_A_N

Table B-10 Symbolic Constants, Continued

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol[1]
					Instrumental	PR_I_N
					Prepositional	PR_P_N
		Plural	Any		Nominative	PR_N_P
					Genitive	PR_G_P
					Dative	PR_D_P
					Accusative Animate	PR_1_P
					Accusative Inanimate	PR_2_P
					Instrumental	PR_I_P
					Prepositional	PR_P_P
	Demonstrative	see Adjective, normal				
Verb	Active			Infinitive		VB_I_*
		Singular	Any	1st Person		VB_1_S_*
				2nd Person		VB_2_S_*
				3rd Person		VB_3_S_*
				Imperative		VB_3_K_*
			Masculine	Past		VB_P_M_*
			Feminine			VB_P_F_*
			Neuter			VB_P_N_*
		Plural	Any	1st Person	Normal	VB_1_P_*
					Imperative	VB_1_K_*
				2nd Person	Normal	VB_2_P_*
					Imperative	VB_2_K_*
				3rd Person		VB_3_P_*
				Past		VB_P_P_*
	Auxiliary			Infinitive		VA_I_*
		Singular	Any	1st Person		VA_1_S_*
				2nd Person		VA_2_S_*
				3rd Person		VA_3_S_*
				Imperative		VA_3_K_*
			Masculine	Past		VA_P_M_*
			Feminine			VA_P_F_*
			Neuter			VA_P_N_*
		Plural	Any	1st Person	Normal	VA_1_P_*
					Imperative	VA_1_K_*
				2nd Person	Normal	VA_2_P_*
					Imperative	VA_2_K_*
				3rd Person		VA_3_P_*
				Past		VA_P_P_*
Adjective	Normal	Singular	Masculine		Nominative	AJ_N_M_*
					Nominative, short form	AJ_S_M_*
					Genitive	AJ_G_M_*
					Dative	AJ_D_M_*
					Accusative Animate	AJ_1_M_*
					Accusative Inanimate	AJ_2_M_*
					Instrumental	AJ_I_M_*
					Prepositional	AJ_P_M_*
			Feminine		Nominative	AJ_N_F_*
					Nominative, short form	AJ_S_F_*
					Genitive	AJ_G_F_*
					Dative	AJ_D_F_*
					Accusative Animate	AJ_A_F_*
					Accusative Inanimate	AJ_A_F_*
					Instrumental	AJ_I_F_*
					Prepositional	AJ_P_F_*

Table B-10 Symbolic Constants, Continued

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol[1]
			Neuter		Nominative	**AJ_N_N_***
					Nominative, short form	**AJ_S_N_***
					Genitive	**AJ_G_N_***
					Dative	**AJ_D_N_***
					Accusative Animate	**AJ_A_N_***
					Accusative Inanimate	**AJ_A_N_***
					Instrumental	**AJ_I_N_***
					Prepositional	**AJ_P_N_***
		Plural	Any		Nominative	**AJ_N_P_***
					Nominative, short form	**AJ_S_P_***
					Genitive	**AJ_G_P_***
					Dative	**AJ_D_P_***
					Accusative Animate	**AJ_1_P_***
					Accusative Inanimate	**AJ_2_P_***
					Instrumental	**AJ_I_P_***
					Prepositional	**AJ_P_P_***
	Comparative					**AJ_C**
	Superlative	see Adjective, normal				
Adverb	Normal					**AV_N_***
	Comparative					**AV_C**
	Superlative					**AV_S**
Preposition					Genitive	**EP_G**
					Dative	**EP_D**
					Accusative	**EP_A**
					Instrumental	**EP_I**
					Prepositional	**EP_P**
Interrogative					Nominative	**IN_N**
					Genitive	**IN_G**
					Dative	**IN_D**
					Accusative	**IN_A**
					Instrumental	**IN_I**
					Prepositional	**IN_P**
Verbal Adverb	Simultaneous					**VV_S_***
	Sequential					**VV_Q_***
Verbal Adjective	Present Active	Singular	Masculine		Nominative	**VJ_PA_N_M_***
					Genitive	**VJ_PA_G_M_***
					Dative	**VJ_PA_D_M_***
					Accusative Animate	**VJ_PA_1_M_***
					Accusative Inanimate	**VJ_PA_2_M_***
					Instrumental	**VJ_PA_I_M_***
					Prepositional	**VJ_PA_P_M_***
			Feminine		Nominative	**VJ_PA_N_F_***
					Genitive	**VJ_PA_G_F_***
					Dative	**VJ_PA_D_F_***
					Accusative Animate	**VJ_PA_A_F_***
					Accusative Inanimate	**VJ_PA_A_F_***
					Instrumental	**VJ_PA_I_F_***
					Prepositional	**VJ_PA_P_F_***
			Neuter		Nominative	**VJ_PA_N_N_***
					Genitive	**VJ_PA_G_N_***
					Dative	**VJ_PA_D_N_***
					Accusative Animate	**VJ_PA_A_N_***
					Accusative Inanimate	**VJ_PA_A_N_***
					Instrumental	**VJ_PA_I_N_***

Table B-10 Symbolic Constants, Continued

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol[1]
					Prepositional	VJ_PA_P_N_*
		Plural	Any		Nominative	VJ_PA_N_P_*
					Genitive	VJ_PA_G_P_*
					Dative	VJ_PA_D_P_*
					Accusative Animate	VJ_PA_1_P_*
					Accusative Inanimate	VJ_PA_2_P_*
					Instrumental	VJ_PA_I_P_*
					Prepositional	VJ_PA_P_P_*
	Present Passive	Singular	Masculine		Nominative	VJ_PP_N_M_*
					Genitive	VJ_PP_G_M_*
					Dative	VJ_PP_D_M_*
					Accusative Animate	VJ_PP_1_M_*
					Accusative Inanimate	VJ_PP_2_M_*
					Instrumental	VJ_PP_I_M_*
					Prepositional	VJ_PP_P_M_*
			Feminine		Nominative	VJ_PP_N_F_*
					Genitive	VJ_PP_G_F_*
					Dative	VJ_PP_D_F_*
					Accusative Animate	VJ_PP_A_F_*
					Accusative Inanimate	VJ_PP_A_F_*
					Instrumental	VJ_PP_I_F_*
					Prepositional	VJ_PP_P_F_*
			Neuter		Nominative	VJ_PP_N_N_*
					Genitive	VJ_PP_G_N_*
					Dative	VJ_PP_D_N_*
					Accusative Animate	VJ_PP_A_N_*
					Accusative Inanimate	VJ_PP_A_N_*
					Instrumental	VJ_PP_I_N_*
					Prepositional	VJ_PP_P_N_*
		Plural	Any		Nominative	VJ_PP_N_P_*
					Genitive	VJ_PP_G_P_*
					Dative	VJ_PP_D_P_*
					Accusative Animate	VJ_PP_1_P_*
					Accusative Inanimate	VJ_PP_2_P_*
					Instrumental	VJ_PP_I_P_*
					Prepositional	VJ_PP_P_P_*
	Past Active	Singular	Masculine		Nominative	VJ_AA_N_M_*
					Genitive	VJ_AA_G_M_*
					Dative	VJ_AA_D_M_*
					Accusative Animate	VJ_AA_1_M_*
					Accusative Inanimate	VJ_AA_2_M_*
					Instrumental	VJ_AA_I_M_*
					Prepositional	VJ_AA_P_M_*
			Feminine		Nominative	VJ_AA_N_F_*
					Genitive	VJ_AA_G_F_*
					Dative	VJ_AA_D_F_*
					Accusative Animate	VJ_AA_A_F_*
					Accusative Inanimate	VJ_AA_A_F_*
					Instrumental	VJ_AA_I_F_*
					Prepositional	VJ_AA_P_F_*
			Neuter		Nominative	VJ_AA_N_N_*
					Genitive	VJ_AA_G_N_*
					Dative	VJ_AA_D_N_*
					Accusative Animate	VJ_AA_A_N_*

Table B-10 Symbolic Constants, Continued

Part of Speech	Attribute	Number	Gender	Conjugation	Part of Speech Role	Symbol[1]
					Accusative Inanimate	VJ_AA_A_N_*
					Instrumental	VJ_AA_I_N_*
					Prepositional	VJ_AA_P_N_*
		Plural	Any		Nominative	VJ_AA_N_P_*
					Genitive	VJ_AA_G_P_*
					Dative	VJ_AA_D_P_*
					Accusative Animate	VJ_AA_1_P_*
					Accusative Inanimate	VJ_AA_2_P_*
					Instrumental	VJ_AA_I_P_*
					Prepositional	VJ_AA_P_P_*
	Past Passive	Singular	Masculine		Nominative	VJ_AP_N_M_*
					Nominative, short I	VJ_AP_S_M_*
					Nominative, short II	AJ_S_M
					Genitive	VJ_AP_G_M_*
					Dative	VJ_AP_D_M_*
					Accusative Animate	VJ_AP_1_M_*
					Accusative Inanimate	VJ_AP_2_M_*
					Instrumental	VJ_AP_I_M_*
					Prepositional	VJ_AP_P_M_*
			Feminine		Nominative	VJ_AP_N_F_*
					Nominative, short I	VJ_AP_S_F_*
					Nominative, short II	AJ_S_F
					Genitive	VJ_AP_G_F_*
					Dative	VJ_AP_D_F_*
					Accusative Animate	VJ_AP_A_F_*
					Accusative Inanimate	VJ_AP_A_F_*
					Instrumental	VJ_AP_I_F_*
					Prepositional	VJ_AP_P_F_*
			Neuter		Nominative	VJ_AP_N_N_*
					Nominative, short I	VJ_AP_S_N_*
					Nominative, short II	AJ_S_N
					Genitive	VJ_AP_G_N_*
					Dative	VJ_AP_D_N_*
					Accusative Animate	VJ_AP_A_N_*
					Accusative Inanimate	VJ_AP_A_N_*
					Instrumental	VJ_AP_I_N_*
					Prepositional	VJ_AP_P_N_*
		Plural	Any		Nominative	VJ_AP_N_P_*
					Nominative, short I	VJ_AP_S_P_*
					Nominative, short II	AJ_S_P
					Genitive	VJ_AP_G_P_*
					Dative	VJ_AP_D_P_*
					Accusative Animate	VJ_AP_1_P_*
					Accusative Inanimate	VJ_AP_2_P_*
					Instrumental	VJ_AP_I_P_*
					Prepositional	VJ_AP_P_P_*
Interjection						IJ
Emphatic Particle						IJ
Conjunction						CJ

Table B-10 Symbolic Constants, Continued

† Due to space restrictions, the following cases demanded by the given part of speech are not listed above with their respective entries. Any of these six symbols can replace an asterisk.

Government	Symbol
None	**0**
Nominative	**N**
Genitive	**G**
Dative	**D**
Accusative	**A**
Instrumental	**I**

APPENDIX C

Inflection Patterns

Table C-1 Hard Masculine Noun Patterns

		Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5
Singular	Nominative	пол	угол	лёд	глаз	лес
	Genitive	пола	угла	льда	глаза	леса
	Dative	полу	углу	льду	глазу	лесу
	Accusative	пол	угол	лёд	глаз	лес
	Instrumental	полом	углом	льдом	глазом	лесом
	Prepositional	поле	угле	льде	глазе	лесе
	Locative	полу	углу	льду	глазу	лесу
Plural	Nominative	полы	углы	льды	глаза	леса
	Genitive	полов	углов	льдов	глаз	лесов
	Dative	полам	углам	льдам	глазам	лесам
	Accusative	полы	углы	льды	глаза	леса
	Instrumental	полами	углами	льдами	глазами	лесами
	Prepositional	полах	углах	льдах	глазах	лесах

		Pattern 6	Pattern 7	Pattern 8	Pattern 9	Pattern 10
Singular	Nominative	стол	рынок	брат	врач	доктор
	Genitive	стола	рынка	брата	врача	доктора
	Dative	столу	рынку	брату	врачу	доктору
	Accusative	стол	рынок	брата	врача	доктора
	Instrumental	столом	рынком	братом	врачом	доктором
	Prepositional	столе	рынке	брате	враче	докторе
Plural	Nominative	столы	рынки	братья	врачи	доктора
	Genitive	столов	рынков	братьев	врачей	докторов
	Dative	столам	рынкам	братьям	врачам	докторам
	Accusative	столы	рынки	братьев	врачей	докторов
	Instrumental	столами	рынками	братьями	врачами	докторами
	Prepositional	столах	рынках	братьях	врачах	докторах

		Pattern 11	Pattern 12	Pattern 13	Pattern 14	Pattern 15
Singular	Nominative	друг	отец	тигр	индюк	римлянин
	Genitive	друга	отца	тигра	индюка	римлянина
	Dative	другу	отцу	тигру	индюку	римлянину
	Accusative	друга	отца	тигра	индюка	римлянина
	Instrumental	другом	отцом	тигром	индюком	римлянином
	Prepositional	друге	отце	тигре	индюке	римлянине
Plural	Nominative	друзья	отцы	тигры	индюки	римляне
	Genitive	друзей	отцов	тигров	индюков	римлян
	Dative	друзьям	отцам	тиграм	индюкам	римлянам
	Accusative	друзей	отцов	тигров	индюков	римлян
	Instrumental	друзьями	отцами	тиграми	индюками	римлянами
	Prepositional	друзьях	отцах	тиграх	индюках	римлянах

Table C-1 Hard Masculine Noun Patterns, Continued

		Pattern 16	Pattern 17	Pattern 18	Pattern 19	Pattern 20
Singular	Nominative	сын	муж	ковёр	мышонок	телёнок
	Genitive	сына	мужа	ковра	мышонка	телёнка
	Dative	сыну	мужу	ковру	мышонку	телёнку
	Accusative	сына	мужа	ковёр	мышонка	телёнка
	Instrumental	сыном	мужем	ковром	мышонком	телёнком
	Prepositional	сыне	муже	ковре	мышонке	телёнке
Plural	Nominative	сыновья	мужья	ковры	мышата	телята
	Genitive	сыновей	мужей	ковров	мышат	телят
	Dative	сыновьям	мужьям	коврам	мышатам	телятам
	Accusative	сыновей	мужей	ковры	мышат	телят
	Instrumental	сыновьями	мужьями	коврами	мышатами	телятами
	Prepositional	сыновьях	мужьях	коврах	мышатах	телятах

		Pattern 21†	Pattern 22	Pattern 23	Pattern 24	Pattern 25
Singular	Nominative	человек	стул	шток	вечер	Даника
	Genitive	человека	стула	штока	вечера	Даника
	Dative	человеку	стулу	штоку	вечеру	Данику
	Accusative	человека	стул	шток	вечер	Даника
	Instrumental	человеком	стулом	штоком	вечером	Даником
	Prepositional	человеке	стуле	штоке	вечере	Данике
Plural	Nominative		стулья	штоки	вечера	Даники
	Genitive	человек	стульев	штоков	вечеров	Даников
	Dative		стульям	штокам	вечерам	Даникам
	Accusative		стулья	штоки	вечера	Даников
	Instrumental		стульями	штоками	вечерами	Даниками
	Prepositional		стульях	штоках	вечерах	Даниках

† Plural forms except the genitive after numbers are rendered by ÌÀÄÉ (Pattern 112)

		Pattern 26†
Singular	Nominative	кенгуру
	Genitive	кенгуру
	Dative	кенгуру
	Accusative	кенгуру
	Instrumental	кенгуру
	Prepositional	кенгуру
Plural	Nominative	кенгуру
	Genitive	кенгуру
	Dative	кенгуру
	Accusative	кенгуру
	Instrumental	кенгуру
	Prepositional	кенгуру

† Indeclinable

Table C-2 Soft Masculine Noun Patterns

		Pattern 30	Pattern 31	Pattern 32	Pattern 33	Pattern 34
Singular	Nominative	край	дождь	музей	киль	день
	Genitive	края	дождя	музея	киля	дня
	Dative	краю	дождю	музею	килю	дню
	Accusative	край	дождь	музей	киль	день
	Instrumental	краем	дождём	музеем	килем	днём
	Prepositional	крае	дожде	музее	киле	дне
	Locative	краю				
Plural	Nominative	краи	дожди	музеи	кили	дни
	Genitive	краёв	дождей	музеев	килей	дней
	Dative	краям	дождям	музеям	килям	дням
	Accusative	краи	дожди	музеи	кили	дни
	Instrumental	краями	дождями	музеями	килями	днями
	Prepositional	краях	дождях	музеях	килях	днях

		Pattern 35	Pattern 36	Pattern 37	Pattern 38	Pattern 39
Singular	Nominative	путь	кашель	гость	герой	уголь
	Genitive	пути	кашля	гостя	героя	угля
	Dative	пути	кашлю	гостю	герою	углю
	Accusative	путь	кашель	гостя	героя	уголь
	Instrumental	путём	кашлем	гостем	героем	углем
	Prepositional	пути	кашли	госте	герое	угли
Plural	Nominative	пути	кашли	гости	герои	угли
	Genitive	путей	кашлей	гостей	героев	углей
	Dative	путям	кашлям	гостям	героям	углям
	Accusative	пути	кашли	гостей	героев	угли
	Instrumental	путями	кашлями	гостями	героями	углями
	Prepositional	путях	кашлях	гостях	героях	углях

		Pattern 40
Singular	Nominative	судья
	Genitive	судьи
	Dative	судье
	Accusative	судью
	Instrumental	судьёй
	Prepositional	судье
Plural	Nominative	судьи
	Genitive	судьей
	Dative	судьям
	Accusative	судьей
	Instrumental	судьями
	Prepositional	судьях

Table C-3 Hard Feminine Noun Patterns

		Pattern 50	Pattern 51	Pattern 52	Pattern 53	Pattern 54
Singular	Nominative	газета	штанга	мама	собака	кошка
	Genitive	газеты	штанги	мамы	собаки	кошки
	Dative	газете	штанге	маме	собаке	кошке
	Accusative	газету	штангу	маму	собаку	кошку
	Instrumental	газетой	штангой	мамой	собакой	кошкой
	Prepositional	газете	штанге	маме	собаке	кошке
Plural	Nominative	газеты	штанги	мамы	собаки	кошки
	Genitive	газет	штанг	мам	собак	кошек
	Dative	газетам	штангам	мамам	собакам	кошкам
	Accusative	газеты	штанги	мам	собак	кошек
	Instrumental	газетами	штангами	мамами	собаками	кошками
	Prepositional	газетах	штангах	мамах	собаках	кошках

		Pattern 55	Pattern 56	Pattern 57	Pattern 58	Pattern 59
Singular	Nominative	сестра	жена	точка	покупка	щека
	Genitive	сестры	жены	точки	покупки	щеки
	Dative	сестре	жене	точке	покупке	щеке
	Accusative	сестру	жену	точку	покупку	щёку
	Instrumental	сестрой	женой	точкой	покупкой	щекой
	Prepositional	сестре	жене	точке	покупке	щеке
Plural	Nominative	сёстры	жёны	точки	покупки	щёки
	Genitive	сестёр	жён	точек	покупок	щёк
	Dative	сёстрам	жёнам	точкам	покупкам	щекам
	Accusative	сестёр	жён	точки	покупки	щёки
	Instrumental	сёстрами	жёнами	точками	покупками	щеками
	Prepositional	сёстрах	жёнах	точках	покупках	щеках

		Pattern 60	Pattern 61
Singular	Nominative	звезда	девица
	Genitive	звезды	девицы
	Dative	звезде	девице
	Accusative	звезду	девицу
	Instrumental	звездой	девицей
	Prepositional	звезде	девице
Plural	Nominative	звёзды	девицы
	Genitive	звёзд	девиц
	Dative	звёздам	девицам
	Accusative	звёзды	девиц
	Instrumental	звёздами	девицами
	Prepositional	звёздах	девицах

Table C-4 Soft Feminine Noun Patterns

		Pattern 70	Pattern 71	Pattern 72	Pattern 73	Pattern 74
Singular	Nominative	неделя	лекция	дверь	песня	тётя
	Genitive	недели	лекции	двери	песни	тёти
	Dative	неделе	лекции	двери	песне	тёте
	Accusative	неделю	лекцию	дверь	песню	тётю
	Instrumental	неделей	лекцией	дверью	песней	тётей
	Prepositional	неделе	лекции	двери	песне	тёте
Plural	Nominative	недели	лекции	двери	песни	тёти
	Genitive	недель	лекций	дверей	песен	тётей
	Dative	неделям	лекциям	дверям	песням	тётям
	Accusative	недели	лекции	двери	песни	тётей
	Instrumental	неделями	лекциями	дверями	песнями	тётями
	Prepositional	неделях	лекциях	дверях	песнях	тётях

		Pattern 75	Pattern 76	Pattern 77	Pattern 78	Pattern 79
Singular	Nominative	статья	дочь	мать	церковь	мышь
	Genitive	статьи	дочери	матери	церкви	мыши
	Dative	статьи	дочери	матери	церкви	мыши
	Accusative	статью	дочь	мать	церковь	мышь
	Instrumental	статьёй	дочерью	матерью	церковью	мышью
	Prepositional	статьи	дочери	матери	церкви	мыши
Plural	Nominative	статьи	дочери	матери	церкви	мыши
	Genitive	статьей	дочерей	матерей	церквей	мышей
	Dative	статьям	дочерям	матерям	церквам	мышам
	Accusative	статьи	дочерей	матерей	церкви	мышей
	Instrumental	статьями	дочерьми	матерями	церквами	мышами
	Prepositional	статьях	дочерях	матерях	церквах	мышах

Table C-5 Neuter Noun Patterns

		Pattern 90	Pattern 91	Pattern 92	Pattern 93	Pattern 94
Singular	Nominative	место	здание	поле	кольцо	масло
	Genitive	места	здания	поля	кольца	масла
	Dative	месту	зданию	полю	кольцу	маслу
	Accusative	место	здание	поле	кольцо	масло
	Instrumental	местом	зданием	полем	кольцом	маслом
	Prepositional	месте	здании	поле	кольце	масле
Plural	Nominative	места	здания	поля	кольца	масла
	Genitive	мест	зданий	полей	колец	масел
	Dative	местам	зданиям	полям	кольцам	маслам
	Accusative	места	здания	поля	кольца	масла
	Instrumental	местами	зданиями	полями	кольцами	маслами
	Prepositional	местах	зданиях	полях	кольцах	маслах

		Pattern 95	Pattern 96	Pattern 97	Pattern 98	Pattern 99
Singular	Nominative	время	небо	село	дерево	яблоко
	Genitive	времени	неба	села	дерева	яблока
	Dative	времени	небу	селу	дереву	яблоку
	Accusative	время	небо	село	дерево	яблоко
	Instrumental	временем	небом	селом	деревом	яблоком
	Prepositional	времени	небе	селе	дереве	яблоке
Plural	Nominative	времена	небеса	сёла	деревья	яблоки
	Genitive	времён	небес	сёл	деревьев	яблок
	Dative	врененам	небесам	сёлам	деревьям	яблокам
	Accusative	времена	небеса	сёла	деревья	яблоки
	Instrumental	временами	небесами	сёлами	деревьями	яблоками
	Prepositional	временах	небесах	сёлах	деревьях	яблоках

		Pattern 100†	Pattern 101	Pattern 102	Pattern 103
Singular	Nominative	метро	облако	ружьё	ущелье
	Genitive	метро	облака	ружья	ущелья
	Dative	метро	облаку	ружью	ущелью
	Accusative	метро	облако	ружьё	ущелье
	Instrumental	метро	облаком	ружьём	ущельем
	Prepositional	метро	облаке	ружье	ущелье
Plural	Nominative	метро	облака	ружья	ущелья
	Genitive	метро	облаков	ружьей	ущелий
	Dative	метро	облакам	ружьям	ущельям
	Accusative	метро	облака	ружья	ущелья
	Instrumental	метро	облаками	ружьями	ущельями
	Prepositional	метро	облаках	ружьях	ущельях

† Indeclinable

Table C-6 Plural Only Noun Patterns

	Pattern 110	Pattern 111	Pattern 112	Pattern 113	Pattern 114
Nominative	очки	деньги	люди	ребята	трусы
Genitive	очков	денег	людей	ребят	трусов
Dative	очкам	деньгам	людям	ребятам	трусам
Accusative	очки	деньги	людей	ребят	трусы
Instrumental	очками	деньгами	людьми	ребятами	трусами
Prepositional	очках	деньгах	людях	ребятах	трусах

Table C-7 Hard Adjective Patterns

	Pattern 120			
	Masculine	Feminine	Neuter	Plural
Nominative	красивый	красивая	красивое	красивые
Genitive	красивого	красивой	красивого	красивых
Dative	красивому	красивой	красивому	красивым
Accusative Animate	красивого	красивую	красивое	красивых
Accusative Inanimate	красивый	красивую	красивое	красивые
Instrumental	красивым	красивой	красивым	красивыми
Prepositional	красивом	красивой	красивом	красивых
Short Form	красив	красива	красиво	красивы

	Pattern 121			
	Masculine	Feminine	Neuter	Plural
Nominative	хороший	хорошая	хорошее	хорошие
Genitive	хорошего	хорошей	хорошего	хороших
Dative	хорошему	хорошей	хорошему	хорошим
Accusative Animate	хорошего	хорошую	хорошее	хороших
Accusative Inanimate	хороший	хорошую	хорошее	хорошие
Instrumental	хорошим	хорошей	хорошим	хорошими
Prepositional	хорошем	хорошей	хорошем	хороших
Short Form	хорош	хороша	хорошо	хороши

	Pattern 122			
	Masculine	Feminine	Neuter	Plural
Nominative	плохой	плохая	плохое	плохие
Genitive	плохого	плохой	плохого	плохих
Dative	плохому	плохой	плохому	плохим
Accusative Animate	плохого	плохую	плохое	плохих
Accusative Inanimate	плохой	плохую	плохое	плохие
Instrumental	плохим	плохой	плохим	плохими
Prepositional	плохом	плохой	плохом	плохих
Short Form	плох	плоха	плохо	плохи

	Pattern 123			
	Masculine	Feminine	Neuter	Plural
Nominative	близкий	близкая	близкое	близкие
Genitive	близкого	близкой	близкого	близких
Dative	близкому	близкой	близкому	близким
Accusative Animate	близкого	близкую	близкое	близких
Accusative Inanimate	близкий	близкую	близкое	близкие
Instrumental	близким	близкой	близким	близкими
Prepositional	близком	близкой	близком	близких
Short Form	близок	близка	близко	близки

Table C-7 Hard Adjective Patterns, Continued

Pattern 124				
	Masculine	Feminine	Neuter	Plural
Nominative	важный	важная	важное	важные
Genitive	важного	важной	важного	важных
Dative	важному	важной	важному	важным
Accusative Animate	важного	важную	важное	важных
Accusative Inanimate	важный	важную	важное	важные
Instrumental	важным	важной	важным	важными
Prepositional	важном	важной	важном	важных
Short Form	важен	важна	важно	важны

Pattern 125				
	Masculine	Feminine	Neuter	Plural
Nominative	широкий	широкая	широкое	широкие
Genitive	широкого	широкой	широкого	широких
Dative	широкому	широкой	широкому	широким
Accusative Animate	широкого	широкую	широкое	широких
Accusative Inanimate	широкий	широкую	широкое	широкие
Instrumental	широким	широкой	широким	широкими
Prepositional	широком	широкой	широком	широких
Short Form	широк	широка	широко	широки

Pattern 126				
	Masculine	Feminine	Neuter	Plural
Nominative	сильный	сильная	сильное	сильные
Genitive	сильного	сильной	сильного	сильных
Dative	сильному	сильной	сильному	сильным
Accusative Animate	сильного	сильную	сильное	сильных
Accusative Inanimate	сильный	сильную	сильное	сильные
Instrumental	сильным	сильной	сильным	сильными
Prepositional	сильном	сильной	сильном	сильных
Short Form	силен	сильна	сильно	сильны

Table C-8 Soft Adjective Patterns

	Pattern 130			
	Masculine	Feminine	Neuter	Plural
Nominative	синий	синяя	синее	синие
Genitive	синего	синей	синего	синих
Dative	синему	синей	синему	синим
Accusative Animate	синего	синюю	синее	синих
Accusative Inanimate	синий	синюю	синее	синие
Instrumental	синим	синей	синим	синими
Prepositional	синем	синей	синем	синих

	Pattern 131			
	Masculine	Feminine	Neuter	Plural
Nominative	третий	третья	третье	третьи
Genitive	третьего	третьей	третьего	третьих
Dative	третьему	третьей	третьему	третьим
Accusative Animate	третьего	третью	третье	третьих
Accusative Inanimate	третий	третью	третье	третьи
Instrumental	третьим	третьей	третьим	третьими
Prepositional	третьем	третьей	третьем	третьих

Table C-9 Miscellaneous Adjective Patterns

Pattern 140				
	Masculine	Feminine	Neuter	Plural
Nominative	весь	вся	всё	все
Genitive	всего	всей	всего	всех
Dative	всему	всей	всему	всем
Accusative Animate	всего	всю	всё	всех
Accusative Inanimate	весь	всю	всё	все
Instrumental	всем	всей	всем	всеми
Prepositional	всём	всей	всём	всех

Pattern 141				
	Masculine	Feminine	Neuter	Plural
Nominative	чей	чья	чьё	чьи
Genitive	чьего	чьей	чьего	чьих
Dative	чьему	чьей	чьему	чьим
Accusative Animate	чьего	чью	чьё	чьих
Accusative Inanimate	чьей	чью	чьё	чьи
Instrumental	чьим	чьей	чьим	чьими
Prepositional	чьём	чьей	чьём	чьих

Pattern 142				
	Masculine	Feminine	Neuter	Plural
Nominative	один	одна	одно	одни
Genitive	одного	одной	одного	одних
Dative	одному	одной	одному	одним
Accusative Animate	одного	одну	одно	одних
Accusative Inanimate	один	одну	одно	одни
Instrumental	одним	одной	одним	одними
Prepositional	одном	одной	одном	одних

Pattern 143			
	Masculine	Feminine	Neuter
Nominative	оба	обе	оба
Genitive	обоих	обеих	обоих
Dative	обоим	обеим	обоим
Accusative Animate	обоих	обеих	оба
Accusative Inanimate	оба	обе	оба
Instrumental	обоими	обеими	обоими
Prepositional	обоих	обеих	обоих

Table C-10 Possessive Pronoun Patterns

	Pattern 150			
	Masculine	Feminine	Neuter	Plural
Nominative	мой	моя	моё	мои
Genitive	моего	моей	моего	моих
Dative	моему	моей	моему	моим
Accusative Animate	моего	мою	моё	моих
Accusative Inanimate	мой	мою	моё	мои
Instrumental	моим	моей	моим	моими
Prepositional	моём	моей	моём	моих

	Pattern 151			
	Masculine	Feminine	Neuter	Plural
Nominative	наш	наша	наше	наши
Genitive	нашего	нашей	нашего	наших
Dative	нашему	нашей	нашему	нашим
Accusative Animate	нашего	нашу	наше	наших
Accusative Inanimate	наш	нашу	наше	наши
Instrumental	нашим	нашей	нашим	нашими
Prepositional	нашем	нашей	нашем	наших

Table C-11 Demonstrative Pronoun Patterns

	Pattern 155			
	Masculine	Feminine	Neuter	Plural
Nominative	этот	эта	это	эти
Genitive	этого	этой	этого	этих
Dative	этому	этой	этому	этим
Accusative Animate	этого	эту	этого	этих
Accusative Inanimate	этот	эту	это	эти
Instrumental	этим	этой	этим	этими
Prepositional	этом	этой	этом	этих

	Pattern 156			
	Masculine	Feminine	Neuter	Plural
Nominative	тот	та	то	те
Genitive	того	той	того	тех
Dative	тому	той	тому	тем
Accusative Animate	того	ту	того	тех
Accusative Inanimate	тот	ту	то	те
Instrumental	тем	той	тем	теми
Prepositional	том	той	том	тех

Table C-12 Personal Pronoun Patterns

	Pattern 390	Pattern 391	Pattern 392	Pattern 393	Pattern 394
Nominative	я		ты	он	
Genitive		меня	тебя		его
Dative		мне	тебе		ему
Accusative		меня	тебя		его
Instrumental		мной	тобой		
Prepositional		мне	тебе		

	Pattern 395	Pattern 396	Pattern 397	Pattern 398	Pattern 399
Nominative			она		
Genitive				её	
Dative				ей	
Accusative				её	
Instrumental	им			ей	
Prepositional		нём			ней

	Pattern 400	Pattern 401	Pattern 402	Pattern 403	Pattern 404
Nominative	мы		они		
Genitive		нас		их	
Dative		нам		им	
Accusative		нас		их	
Instrumental		нами		ими	
Prepositional		нас			них

Table C-13 Relative Pronoun Patterns

	Pattern 410			
	Masculine	Feminine	Neuter	Plural
Nominative	который	которая	которое	которые
Genitive	которого	которой	которого	которых
Dative	которому	которой	которому	которым
Accusative Animate	которого	которую	которое	которых
Accusative Inanimate	который	которую	которое	которые
Instrumental	которым	которой	которым	которыми
Prepositional	котором	которой	котором	которых

Table C-14 Interrogative Patterns

	Pattern 380	Pattern 381
Nominative	кто	что
Genitive	кого	чего
Dative	кому	чему
Accusative	кого	что
Instrumental	кем	чем
Prepositional	ком	чём

Table C-15 Non-Reflexive Verb Patterns

		Pattern 160	Pattern 161	Pattern 162	Pattern 163
	Infinitive	читать	говорить	сказать	жить
Singular	1st Person	читаю	говорю	скажу	живу
	2nd Person	читаешь	говоришь	скажешь	живёшь
	3rd Person	читает	говорит	скажет	живёт
	Imperative	читай	говори	скажи	живи
Plural	1st Person	читаем	говорим	скажем	живём
	2nd Person	читаете	говорите	скажете	живёте
	3rd Person	читают	говорят	скажут	живут
	Imperative	читайте	говорите	скажите	живите
Past	Masculine	читал	говорил	сказал	жил
	Feminine	читала	говорила	сказала	жила
	Neuter	читало	говорило	сказало	жило
	Plural	читали	говорили	сказали	жили

		Pattern 164	Pattern 165	Pattern 166	Pattern 167
	Infinitive	давать	ответить	писать	купить
Singular	1st Person	даю	отвечу	пишу	куплю
	2nd Person	даёшь	ответишь	пишешь	купишь
	3rd Person	даёт	ответит	пишет	купит
	Imperative	дай	ответь	пиши	купи
Plural	1st Person	даём	ответим	пишем	купим
	2nd Person	даёте	ответите	пишете	купите
	3rd Person	дают	ответят	пишут	купят
	Imperative	дайте	ответьте	пишите	купите
Past	Masculine	давал	ответил	писал	купил
	Feminine	давала	ответила	писала	купила
	Neuter	давало	ответило	писало	купило
	Plural	давали	ответили	писали	купили

		Pattern 168	Pattern 169	Pattern 170	Pattern 171
	Infinitive	забыть	видеть	закрыть	класть
Singular	1st Person	забуду	вижу	закрою	кладу
	2nd Person	забудешь	видишь	закроешь	кладёшь
	3rd Person	забудет	видит	закроет	кладёт
	Imperative	забудь	види	закрой	клади
Plural	1st Person	забудем	видим	закроем	кладём
	2nd Person	забудете	видите	закроете	кладёте
	3rd Person	забудут	видят	закроют	кладут
	Imperative	забудьте	видите	закройте	кладите
Past	Masculine	забыл	видел	закрыл	клал
	Feminine	забыла	видела	закрыла	клала
	Neuter	забыло	видело	закрыло	клало
	Plural	забыли	видели	закрыли	клали

Table C-15 Non-Reflexive Verb Patterns, Continued

		Pattern 172	Pattern 173	Pattern 174	Pattern 175
	Infinitive	положить	дать	стать	помочь
Singular	1st Person	положу	дам	стану	помогу
	2nd Person	положишь	даш	станешь	поможешь
	3rd Person	положит	даст	станет	поможет
	Imperative	положи	дай	стань	помоги
Plural	1st Person	положим	дадим	станем	поможем
	2nd Person	положите	дадите	станете	поможете
	3rd Person	положат	дадут	станут	помогут
	Imperative	положите	дайте	станьте	помогите
Past	Masculine	положил	дал	стал	помог
	Feminine	положила	дала	стала	помогла
	Neuter	положило	дало	стало	помогло
	Plural	положили	дали	стали	помогли

		Pattern 176	Pattern 177	Pattern 178	Pattern 179
	Infinitive	понять	лечь	начать	есть
Singular	1st Person	пойму	лягу	начну	ем
	2nd Person	поймёшь	ляжешь	начнёшь	ешь
	3rd Person	поймёт	ляжет	начнёт	есть
	Imperative	пойми	ляг	начни	ешь
Plural	1st Person	поймём	ляжем	начнём	едим
	2nd Person	поймёте	ляжете	начнёте	едите
	3rd Person	поймут	лягут	начнут	едят
	Imperative	поймите	лягте	начните	ешьте
Past	Masculine	понял	лёг	начал	ел
	Feminine	поняла	легла	начала	ела
	Neuter	поняло	легло	начало	ело
	Plural	поняли	легли	начали	ели

		Pattern 180	Pattern 181	Pattern 182	Pattern 183
	Infinitive	дить	деть	сесть	послать
Singular	1st Person	пью	пою	сяду	пошлю
	2nd Person	пьёшь	поёшь	сядешь	пошлёшь
	3rd Person	пьёт	поёт	сядет	пошлёт
	Imperative	пьей	пой	сядь	пошли
Plural	1st Person	пьём	поём	сядем	пошлём
	2nd Person	пьёте	поёте	сядете	пошлёте
	3rd Person	пьют	поют	сядут	пошлют
	Imperative	пьейте	пойте	сядьте	пошлите
Past	Masculine	пил	пел	сел	послал
	Feminine	пила	пела	села	послала
	Neuter	пило	пело	село	послало
	Plural	пили	пели	сели	послали

Table C-15 Non-Reflexive Verb Patterns, Continued

		Pattern 184	Pattern 185	Pattern 186	Pattern 187
	Infinitive	брать	взять	миновать	ночевать
Singular	1st Person	беру	возьму	миную	ночую
	2nd Person	берёшь	возьмёшь	минуешь	ночуешь
	3rd Person	берёт	возьмёт	минует	ночует
	Imperative	бери	возьми	минуй	ночуй
Plural	1st Person	берём	возьмём	минуем	ночуем
	2nd Person	берёте	возьмёте	минуете	ночуете
	3rd Person	берут	возьмут	минуют	ночуют
	Imperative	берите	возьмите	минуйте	ночуйте
Past	Masculine	брал	взял	миновал	ночевал
	Feminine	брала	взяла	миновала	ночевала
	Neuter	брало	взяло	миновало	ночевало
	Plural	брали	взяли	миновали	ночевали

		Pattern 188	Pattern 189	Pattern 190	Pattern 191
	Infinitive	искать	спать	ходить	носить
Singular	1st Person	ищу	сплю	хожу	ношу
	2nd Person	ищешь	спишь	ходишь	носишь
	3rd Person	ищет	спит	ходит	носит
	Imperative	ищи	спи	ходи	носи
Plural	1st Person	ищем	спим	ходим	носим
	2nd Person	ищете	спите	ходите	носите
	3rd Person	ищут	спят	ходят	носят
	Imperative	ищите	спите	ходите	носите
Past	Masculine	искал	спал	ходил	носил
	Feminine	искала	спала	ходила	носила
	Neuter	искало	спало	ходило	носило
	Plural	искали	спали	ходили	носили

		Pattern 192[1]	Pattern 193[1]	Pattern 194[2]	Pattern 195
	Infinitive	идти		ехать	вести
Singular	1st Person	иду		еду	веду
	2nd Person	идёшь		едешь	ведёшь
	3rd Person	идёт		едет	ведёт
	Imperative	иди			веди
Plural	1st Person	идём		едим	ведём
	2nd Person	идёте		едите	ведёте
	3rd Person	идут		едут	ведут
	Imperative	идите			ведите
Past	Masculine		шёл	ехал	вёл
	Feminine		шла	ехала	вела
	Neuter		шло	ехало	вело
	Plural		шли	ехали	вели

[1] Present and past tense forms have no common stem.

[2] Imperative ÐÏÄÚÖÁÊ(ÔÅ) is not related to the stem.

Table C-15 Non-Reflexive Verb Patterns, Continued

		Pattern 196	Pattern 197	Pattern 198	Pattern 199
	Infinitive	лететь	поднять	ждать	хотеть
Singular	1st Person	лечу	подниму	жду	хочу
	2nd Person	летишь	поднимешь	ждёжь	хочешь
	3rd Person	летит	поднимет	ждёт	хочет
	Imperative	лети	подними	жди	хоти
Plural	1st Person	летим	поднимем	ждём	хотим
	2nd Person	летите	поднимете	ждёте	хотите
	3rd Person	летят	поднимут	ждут	хотят
	Imperative	летите	поднимите	ждите	хотите
Past	Masculine	летел	поднял	ждал	хотел
	Feminine	летела	подняла	ждала	хотела
	Neuter	летело	подняло	ждало	хотело
	Plural	летели	подняли	ждали	хотели

Table C-16 Non-Reflexive Verb Patterns

		Pattern 210	Pattern 211	Pattern 212
	Infinitive	читаться	говориться	сказаться
Singular	1st Person	читаюсь	говорюсь	скажусь
	2nd Person	читаешься	говоришься	скажешься
	3rd Person	читается	говорится	скажется
	Imperative	читайся	говорись	скажись
Plural	1st Person	читаемся	говоримся	скажемся
	2nd Person	читаетесь	говоритесь	скажетесь
	3rd Person	читаются	говорятся	скажутся
	Imperative	читайтесь	говоритесь	скажитесь
Past	Masculine	читался	говорился	сказался
	Feminine	читалась	говорилась	сказалась
	Neuter	читалось	говорилось	сказалось
	Plural	читались	говорились	сказались

		Pattern 213	Pattern 214	Pattern 215
	Infinitive	ужиться	отдаваться	писаться
Singular	1st Person	уживусь	отдаюсь	пишусь
	2nd Person	уживёшься	отдаёшься	пишешься
	3rd Person	уживётся	отдаётся	пишется
	Imperative	уживись	отдайся	пишись
Plural	1st Person	уживёмся	отдаёмся	пишемся
	2nd Person	уживётесь	отдаётесь	пишетесь
	3rd Person	уживутся	отдаются	пишутся
	Imperative	уживитесь	отдайтесь	пишитесь
Past	Masculine	ужился	отдался	писался
	Feminine	ужилась	отдалась	писалась
	Neuter	ужилось	отдалось	писалось
	Plural	ужились	отдались	писались

		Pattern 216	Pattern 217	Pattern 218
	Infinitive	видеться	закрыться	питься
Singular	1st Person	вижусь	закроюсь	пьюсь
	2nd Person	видишься	закроешься	пьёшься
	3rd Person	видится	закроется	пьётся
	Imperative	видись	закройся	пьейся
Plural	1st Person	видимся	закроемся	пьёмся
	2nd Person	видитесь	закроетесь	пьётесь
	3rd Person	видятся	закроются	пьются
	Imperative	видитесь	закройтесь	пьейтесь
Past	Masculine	виделся	закрылся	пился
	Feminine	виделась	закрылась	пилась
	Neuter	виделось	закрылось	пилось
	Plural	виделись	закрылись	пились

Table C-16 Non-Reflexive Verb Patterns, Continued

		Pattern 219	Pattern 220	Pattern 221
	Infinitive	положиться	отдаться	статься
Singular	1st Person	положусь	отдамся	станусь
	2nd Person	положишься	отдашся	станешься
	3rd Person	положится	отдастся	станется
	Imperative	положись	отдайся	станься
Plural	1st Person	положимся	отдадимся	станемся
	2nd Person	положитесь	отдадитесь	станетесь
	3rd Person	положатся	отдадутся	станутся
	Imperative	положитесь	отдайтесь	станьтесь
Past	Masculine	положился	отдался	стался
	Feminine	положилась	отдалась	сталась
	Neuter	положилось	отдалось	сталось
	Plural	положились	отдались	стались

		Pattern 222	Pattern 223	Pattern 224
	Infinitive	хотеться	начаться	носиться
Singular	1st Person	хочусь	начнусь	ношусь
	2nd Person	хочешься	начнёшься	носишься
	3rd Person	хочется	начнётся	носится
	Imperative	хотись	начнись	носись
Plural	1st Person	хотимся	начнёмся	носимся
	2nd Person	хотитесь	начнётесь	носитесь
	3rd Person	хотятся	начнутся	носятся
	Imperative	хотитесь	начнитесь	носитесь
Past	Masculine	хотелся	начался	носился
	Feminine	хотелась	началась	носилась
	Neuter	хотелось	началось	носилось
	Plural	хотелись	начались	носились

		Pattern 225	Pattern 226	Pattern 227
	Infinitive	выспаться	браться	взяться
Singular	1st Person	высплюсь	берусь	возьмусь
	2nd Person	выспишься	берёшься	возьмёшься
	3rd Person	выспится	берётся	возьмётся
	Imperative	выспись	берись	возьмись
Plural	1st Person	выспимся	берёмся	возьмёмся
	2nd Person	выспитесь	берётесь	возьмётесь
	3rd Person	выспятся	берутся	возьмутся
	Imperative	выспитесь	беритесь	возьмитесь
Past	Masculine	выспался	брался	взялся
	Feminine	выспалась	бралась	взялась
	Neuter	выспалось	бралось	взялось
	Plural	выспались	брались	взялись

Table C-16 Non-Reflexive Verb Patterns, Continued

		Pattern 228	Pattern 229	Pattern 230
	Infinitive	вестись	разлететься	подняться
Singular	1st Person	ведусь	разлечусь	поднимусь
	2nd Person	ведёшься	разлетишься	поднимешься
	3rd Person	ведётся	разлетится	поднимется
	Imperative	ведись	разлетись	поднимись
Plural	1st Person	ведёмся	разлетимся	поднимемся
	2nd Person	ведётесь	разлетитесь	поднимете сь
	3rd Person	ведутся	разлетятся	поднимутся
	Imperative	ведитесь	разлетитесь	поднимитесь
Past	Masculine	вёлся	разлетелся	поднялся
	Feminine	велась	разлетелась	поднялась
	Neuter	велось	разлетелось	поднялось
	Plural	велись	разлетелись	поднялись

		Pattern 231	Pattern 232	Pattern 233
	Infinitive	влюбиться	любоваться	водиться
Singular	1st Person	влюблюсь	любуюсь	вожусь
	2nd Person	влюбишься	любуешься	водишься
	3rd Person	влюбится	любуется	водится
	Imperative	влюбись	любуйся	водись
Plural	1st Person	влюбимся	любуемся	водимся
	2nd Person	влюбитесь	любуетесь	водитесь
	3rd Person	влюбятся	любуются	водятся
	Imperative	влюбитесь	любуйтесь	водитесь
Past	Masculine	влюбился	любовался	водился
	Feminine	влюбилась	любовалась	водилась
	Neuter	влюбилось	любовалось	водилось
	Plural	влюбились	любовались	водились

		Pattern 234	Pattern 235†
	Infinitive	разойтись	разъехаться
Singular	1st Person	разойдусь	разъедусь
	2nd Person	разойдёшься	разъедешься
	3rd Person	разойдётся	разъедется
	Imperative	разойдись	
Plural	1st Person	разойдёмся	разъедимся
	2nd Person	разойдётесь	разъедитесь
	3rd Person	разойдутся	разъедутся
	Imperative	разойдитесь	
Past	Masculine	разошёлся	разъехался
	Feminine	разошлась	разъехалась
	Neuter	разошлось	разъехалось
	Plural	разошлись	разъехались

† Imperatives ÒÁÚßÂÚÖÁÊÓÑ and ÒÁÚßÂÚÖÁÂÔÂÓØ are not related to the stem.

Table C-17 Non-Reflexive Verbal Adverb Patterns

	Pattern 250	Pattern 251	Pattern 252	Pattern 253
Infinitive†	читать	слышать	образовать	нести
Simultaneous	читая	слыша	образуя	неся
Sequential I	читав	слышав	образовав	
Sequential II	читавши	слышавши	образовавши	нёсши

	Pattern 254	Pattern 255	Pattern 256	Pattern 257
Infinitive†	говорить	видеть	спешать	выжить
Simultaneous	говоря	видя	спеша	выживя
Sequential I	говорив	видев	спешив	выжив
Sequential II	говоривши	видевши	спешивши	выживши

† Included here for reference to the lexicon entry stem, but not part of the pattern.

Table C-18 Reflexive Verbal Adverb Patterns

	Pattern 270	Pattern 271	Pattern 272	Pattern 273
Infinitive†	стараться	видеться	образоваться	нестись
Simultaneous	стараясь	видясь	образуясь	несясь
Sequential	старавшись	видевшись	образовавшись	нёсшись

	Pattern 274	Pattern 275
Infinitive†	мешаться	ужиться
Simultaneous	мешась	уживясь
Sequential	мешившись	ужившись

† Included here for reference to the lexicon entry stem, but not part of the pattern.

Table C-19 Non-Reflexive Present Active Verbal Adjective Patterns

	Pattern 290			
	Masculine	Feminine	Neuter	Plural
Nominative	спешащий	спешащая	спешащее	спешащие
Genitive	спешащего	спешащей	спешащего	спешащих
Dative	спешащему	спешащей	спешащему	спешащим
Accusative Animate	спешащего	спешащую	спешащее	спешащих
Accusative Inanimate	спешащий	спешащую	спешащее	спешащие
Instrumental	спешащим	спешащей	спешащим	спешащими
Prepositional	спешащем	спешащей	спешащем	спешащих

	Pattern 291			
	Masculine	Feminine	Neuter	Plural
Nominative	говорящий	говорящая	говорящее	говорящие
Genitive	говорящего	говорящей	говорящего	говорящих
Dative	говорящему	говорящей	говорящему	говорящим
Accusative Animate	говорящего	говорящую	говорящее	говорящих
Accusative Inanimate	говорящий	говорящую	говорящее	говорящие
Instrumental	говорящим	говорящей	говорящим	говорящими
Prepositional	говорящем	говорящей	говорящем	говорящих

	Pattern 292			
	Masculine	Feminine	Neuter	Plural
Nominative	пишущий	пишущая	пишущее	пишущие
Genitive	пишущего	пишущей	пишущего	пишущих
Dative	пишущему	пишущей	пишущему	пишущим
Accusative Animate	пишущего	пишущую	пишущее	пишущих
Accusative Inanimate	пишущий	пишущую	пишущее	пишущие
Instrumental	пишущим	пишущей	пишущим	пишущими
Prepositional	пишущем	пишущей	пишущем	пишущих

	Pattern 293			
	Masculine	Feminine	Neuter	Plural
Nominative	читающий	читающая	читающее	читающие
Genitive	читающего	читающей	читающего	читающих
Dative	читающему	читающей	читающему	читающим
Accusative Animate	читающего	читающую	читающее	читающих
Accusative Inanimate	читающий	читающую	читающее	читающие
Instrumental	читающим	читающей	читающим	читающими
Prepositional	читающем	читающей	читающем	читающих

	Pattern 294			
	Masculine	Feminine	Neuter	Plural
Nominative	образующий	образующая	образующее	образующие
Genitive	образующего	образующей	образующего	образующих
Dative	образующему	образующей	образующему	образующим
Accusative Animate	образующего	образующую	образующее	образующих
Accusative Inanimate	образующий	образующую	образующее	образующие
Instrumental	образующим	образующей	образующим	образующими
Prepositional	образующем	образующей	образующем	образующих

Table C-20 Non-Reflexive Present Passive Verbal Adjective Patterns

Pattern 310				
	Masculine	Feminine	Neuter	Plural
Nominative	читаемый	читаемая	читаемое	читаемые
Genitive	читаемого	читаемой	читаемого	читаемых
Dative	читаемому	читаемой	читаемому	читаемым
Accusative Animate	читаемого	читаемую	читаемое	читаемых
Accusative Inanimate	читаемый	читаемую	читаемое	читаемые
Instrumental	читаемым	читаемой	читаемым	читаемыми
Prepositional	читаемом	читаемой	читаемом	читаемых

Pattern 311				
	Masculine	Feminine	Neuter	Plural
Nominative	говоримый	говоримая	говоримое	говоримые
Genitive	говоримого	говоримой	говоримого	говоримых
Dative	говоримому	говоримой	говоримому	говоримым
Accusative Animate	говоримого	говоримую	говоримое	говоримых
Accusative Inanimate	говоримый	говоримую	говоримое	говоримые
Instrumental	говоримым	говоримой	говоримым	говоримыми
Prepositional	говоримом	говоримой	говоримом	говоримых

Pattern 312				
	Masculine	Feminine	Neuter	Plural
Nominative	даваемый	даваемая	даваемое	даваемые
Genitive	даваемого	даваемой	даваемого	даваемых
Dative	даваемому	даваемой	даваемому	даваемым
Accusative Animate	даваемого	даваемую	даваемое	даваемых
Accusative Inanimate	даваемый	даваемую	даваемое	даваемые
Instrumental	даваемым	даваемой	даваемым	даваемыми
Prepositional	даваемом	даваемой	даваемом	даваемых

Pattern 313				
	Masculine	Feminine	Neuter	Plural
Nominative	образуемый	образуемая	образуемое	образуемые
Genitive	образуемого	образуемой	образуемого	образуемых
Dative	образуемому	образуемой	образуемому	образуемым
Accusative Animate	образуемого	образуемую	образуемое	образуемых
Accusative Inanimate	образуемый	образуемую	образуемое	образуемые
Instrumental	образуемым	образуемой	образуемым	образуемыми
Prepositional	образуемом	образуемой	образуемом	образуемых

Table C-21 Non-Reflexive Past Active Verbal Adjective Patterns

Pattern 320				
	Masculine	Feminine	Neuter	Plural
Nominative	читавший	читавшая	читавшее	читавшие
Genitive	читавшего	читавшей	читавшего	читавших
Dative	читавшему	читавшей	читавшему	читавшим
Accusative Animate	читавшего	читавшую	читавшее	читавших
Accusative Inanimate	читавший	читавшую	читавшее	читавшие
Instrumental	читавшим	читавшей	читавшим	читавшими
Prepositional	читавшем	читавшей	читавшем	читавших

Pattern 321				
	Masculine	Feminine	Neuter	Plural
Nominative	говоривший	говорившая	говорившее	говорившие
Genitive	говорившего	говорившей	говорившего	говоривших
Dative	говорившему	говорившей	говорившему	говорившим
Accusative Animate	говорившего	говорившую	говорившее	говоривших
Accusative Inanimate	говоривший	говорившую	говорившее	говорившие
Instrumental	говорившим	говорившей	говорившим	говорившими
Prepositional	говорившем	говорившей	говорившем	говоривших

Pattern 322				
	Masculine	Feminine	Neuter	Plural
Nominative	видевший	видевшая	видевшее	видевшие
Genitive	видевшего	видевшей	видевшего	видевших
Dative	видевшему	видевшей	видевшему	видевшим
Accusative Animate	видевшего	видевшую	видевшее	видевших
Accusative Inanimate	видевший	видевшую	видевшее	видевшие
Instrumental	видевшим	видевшей	видевшим	видевшими
Prepositional	видевшем	видевшей	видевшем	видевших

Pattern 323				
	Masculine	Feminine	Neuter	Plural
Nominative	мешавший	мешавшая	мешавшее	мешавшие
Genitive	мешавшего	мешавшей	мешавшего	мешавших
Dative	мешавшему	мешавшей	мешавшему	мешавшим
Accusative Animate	мешавшего	мешавшую	мешавшее	мешавших
Accusative Inanimate	мешавший	мешавшую	мешавшее	мешавшие
Instrumental	мешавшим	мешавшей	мешавшим	мешавшими
Prepositional	мешавшем	мешавшей	мешавшем	мешавших

Table C-21 Non-Reflexive Past Active Verbal Adjective Patterns, Continued

	Pattern 324			
	Masculine	Feminine	Neuter	Plural
Nominative	живший	жившая	жившее	жившие
Genitive	жившего	жившей	жившего	живших
Dative	жившему	жившей	жившему	жившим
Accusative Animate	жившего	жившую	жившее	живших
Accusative Inanimate	живший	жившую	жившее	жившие
Instrumental	жившим	жившей	жившим	жившими
Prepositional	жившем	жившей	жившем	живших

	Pattern 325			
	Masculine	Feminine	Neuter	Plural
Nominative	миновавший	миновавшая	миновавшее	миновавшие
Genitive	миновавшего	миновавшей	миновавшего	миновавших
Dative	миновавшему	миновавшей	миновавшему	миновавшим
Accusative Animate	миновавшего	миновавшую	миновавшее	миновавших
Accusative Inanimate	миновавший	миновавшую	миновавшее	миновавшие
Instrumental	миновавшим	миновавшей	миновавшим	миновавшими
Prepositional	миновавшем	миновавшей	миновавшем	миновавших

Table C-22 Non-Reflexive Past Passive Verbal Adjective Patterns

Pattern 340				
	Masculine	Feminine	Neuter	Plural
Nominative	разбитый	разбитая	разбитое	разбитые
Genitive	разбитого	разбитой	разбитого	разбитых
Dative	разбитому	разбитой	разбитому	разбитым
Accusative Animate	разбитого	разбитую	разбитое	разбитых
Accusative Inanimate	разбитый	разбитую	разбитое	разбитые
Instrumental	разбитым	разбитой	разбитым	разбитыми
Prepositional	разбитом	разбитой	разбитом	разбитых
Short Form	разбит	разбита	разбито	разбиты

Pattern 341				
	Masculine	Feminine	Neuter	Plural
Nominative	закрытый	закрытая	закрытое	закрытые
Genitive	закрытого	закрытой	закрытого	закрытых
Dative	закрытому	закрытой	закрытому	закрытым
Accusative Animate	закрытого	закрытую	закрытое	закрытых
Accusative Inanimate	закрытый	закрытую	закрытое	закрытые
Instrumental	закрытым	закрытой	закрытым	закрытыми
Prepositional	закрытом	закрытой	закрытом	закрытых
Short Form	закрыт	закрыта	закрыто	закрыты

Pattern 342				
	Masculine	Feminine	Neuter	Plural
Nominative	одетый	одетая	одетое	одетые
Genitive	одетого	одетой	одетого	одетых
Dative	одетому	одетой	одетому	одетым
Accusative Animate	одетого	одетую	одетое	одетых
Accusative Inanimate	одетый	одетую	одетое	одетые
Instrumental	одетым	одетой	одетым	одетыми
Prepositional	одетом	одетой	одетом	одетых
Short Form	одет	одета	одето	одеты

Pattern 343				
	Masculine	Feminine	Neuter	Plural
Nominative	раненный	раненная	раненное	раненные
Genitive	раненного	раненной	раненного	раненных
Dative	раненному	раненной	раненному	раненным
Accusative Animate	раненного	раненную	раненное	раненных
Accusative Inanimate	раненный	раненную	раненное	раненные
Instrumental	раненным	раненной	раненным	раненными
Prepositional	раненном	раненной	раненном	раненных
Short Form I	ранен	ранена	ранено	ранены
Short Form II	ранен	раненна	раненно	раненны

Table C-22 Non-Reflexive Past Passive Verbal Adjective Patterns, Contined

	Pattern 344			
	Masculine	Feminine	Neuter	Plural
Nominative	купленный	купленная	купленное	купленные
Genitive	купленного	купленной	купленного	купленных
Dative	купленному	купленной	купленному	купленным
Accusative Animate	купленного	купленную	купленное	купленных
Accusative Inanimate	купленный	купленную	купленное	купленные
Instrumental	купленным	купленной	купленным	купленными
Prepositional	купленном	купленной	купленном	купленных
Short Form I	куплен	куплена	куплено	куплены
Short Form II	куплен	купленна	купленно	купленны

	Pattern 345			
	Masculine	Feminine	Neuter	Plural
Nominative	увиденный	увиденная	увиденное	увиденные
Genitive	увиденного	увиденной	увиденного	увиденных
Dative	увиденному	увиденной	увиденному	увиденным
Accusative Animate	увиденного	увиденную	увиденное	увиденных
Accusative Inanimate	увиденный	увиденную	увиденное	увиденные
Instrumental	увиденным	увиденной	увиденным	увиденными
Prepositional	увиденном	увиденной	увиденном	увиденных
Short Form I	увиден	увидена	увидено	увидены
Short Form II	увиден	увиденна	увиденно	увиденны

	Pattern 346			
	Masculine	Feminine	Neuter	Plural
Nominative	удивлённый	удивлённая	удивлённое	удивлённые
Genitive	удивлённого	удивлённой	удивлённого	удивлённых
Dative	удивлённому	удивлённой	удивлённому	удивлённым
Accusative Animate	удивлённого	удивлённую	удивлённое	удивлённых
Accusative Inanimate	удивлённый	удивлённую	удивлённое	удивлённые
Instrumental	удивлённым	удивлённой	удивлённым	удивлёнными
Prepositional	удивлённом	удивлённой	удивлённом	удивлённых
Short Form I	удивлён	удивлёна	удивлёно	удивлёны
Short Form II	удивлён	удивлённа	удивлённо	удивлённы

	Pattern 347			
	Masculine	Feminine	Neuter	Plural
Nominative	образованный	образованная	образованное	образованные
Genitive	образованного	образованной	образованного	образованных
Dative	образованному	образованной	образованному	образованным
Accusative Animate	образованного	образованную	образованное	образованных
Accusative Inanimate	образованный	образованную	образованное	образованные
Instrumental	образованным	образованной	образованным	образованными
Prepositional	образованном	образованной	образованном	образованных
Short Form I	образован	образована	образовано	образованы
Short Form II	образован	образованна	образованно	образованны

Table C-23 Reflexive Present Active Verbal Adjective Patterns

Pattern 360				
	Masculine	Feminine	Neuter	Plural
Nominative	учащийся	учащаяся	учащееся	учащиеся
Genitive	учащегося	учащейся	учащегося	учащихся
Dative	учащемуся	учащейся	учащемуся	учащимся
Accusative Animate	учащегося	учащуюся	учащееся	учащихся
Accusative Inanimate	учащийся	учащуюся	учащееся	учащиеся
Instrumental	учащимся	учащейся	учащимся	учащимися
Prepositional	учащемся	учащейся	учащемся	учащихся

Pattern 361				
	Masculine	Feminine	Neuter	Plural
Nominative	длящийся	длящаяся	длящееся	длящиеся
Genitive	длящегося	длящейся	длящегося	длящихся
Dative	длящемуся	длящейся	длящемуся	длящимся
Accusative Animate	длящегося	длящуюся	длящееся	длящихся
Accusative Inanimate	длящийся	длящуюся	длящееся	длящиеся
Instrumental	длящимся	длящейся	длящимся	длящимися
Prepositional	длящемся	длящейся	длящемся	длящихся

Pattern 362				
	Masculine	Feminine	Neuter	Plural
Nominative	пишущийся	пишущаяся	пишущееся	пишущиеся
Genitive	пишущегося	пишущейся	пишущегося	пишущихся
Dative	пишущемуся	пишущейся	пишущемуся	пишущимся
Accusative Animate	пишущегося	пишущуюся	пишущееся	пишущихся
Accusative Inanimate	пишущийся	пишущуюся	пишущееся	пишущиеся
Instrumental	пишущимся	пишущейся	пишущимся	пишущимися
Prepositional	пишущемся	пишущейся	пишущемся	пишущихся

Pattern 363				
	Masculine	Feminine	Neuter	Plural
Nominative	занимающийся	занимающаяся	занимающееся	занимающиеся
Genitive	занимающегося	занимающейся	занимающегося	занимающихся
Dative	занимающемуся	занимающейся	занимающемуся	занимающимся
Accusative Animate	занимающегося	занимающуюся	занимающееся	занимающихся
Accusative Inanimate	занимающийся	занимающуюся	занимающееся	занимающиеся
Instrumental	занимающимся	занимающейся	занимающимся	занимающимися
Prepositional	занимающемся	занимающейся	занимающемся	занимающихся

Pattern 364				
	Masculine	Feminine	Neuter	Plural
Nominative	образующийся	образующаяся	образующееся	образующиеся
Genitive	образующегося	образующейся	образующегося	образующихся
Dative	образующемуся	образующейся	образующемуся	образующимся
Accusative Animate	образующегося	образующуюся	образующееся	образующихся
Accusative Inanimate	образующийся	образующуюся	образующееся	образующиеся
Instrumental	образующимся	образующейся	образующимся	образующимися
Prepositional	образующемся	образующейся	образующемся	образующихся

Table C-24 Reflexive Past Active Verbal Adjective Patterns

Pattern 370				
	Masculine	Feminine	Neuter	Plural
Nominative	учившийся	учившаяся	учившееся	учившиеся
Genitive	учившегося	учившейся	учившегося	учившихся
Dative	учившемуся	учившейся	учившемуся	учившимся
Accusative Animate	учившегося	учившуюся	учившееся	учившихся
Accusative Inanimate	учившийся	учившуюся	учившееся	учившиеся
Instrumental	учившимся	учившейся	учившимся	учившимися
Prepositional	учившемся	учившейся	учившемся	учившихся

Pattern 371				
	Masculine	Feminine	Neuter	Plural
Nominative	говорившийся	говорившаяся	говорившееся	говорившиеся
Genitive	говорившегося	говорившейся	говорившегося	говорившихся
Dative	говорившемуся	говорившейся	говорившемуся	говорившимся
Accusative Animate	говорившегося	говорившуюся	говорившееся	говорившихся
Accusative Inanimate	говорившийся	говорившуюся	говорившееся	говорившиеся
Instrumental	говорившимся	говорившейся	говорившимся	говорившимися
Prepositional	говорившемся	говорившейся	говорившемся	говорившихся

Pattern 372				
	Masculine	Feminine	Neuter	Plural
Nominative	решавшийся	решавшаяся	решавшееся	решавшиеся
Genitive	решавшегося	решавшейся	решавшегося	решавшихся
Dative	решавшемуся	решавшейся	решавшемуся	решавшимся
Accusative Animate	решавшегося	решавшуюся	решавшееся	решавшихся
Accusative Inanimate	решавшийся	решавшуюся	решавшееся	решавшиеся
Instrumental	решавшимся	решавшейся	решавшимся	решавшимися
Prepositional	решавшемся	решавшейся	решавшемся	решавшихся

Pattern 373				
	Masculine	Feminine	Neuter	Plural
Nominative	ужившийся	ужившаяся	ужившееся	ужившиеся
Genitive	ужившегося	ужившейся	ужившегося	ужившихся
Dative	ужившемуся	ужившейся	ужившемуся	ужившимся
Accusative Animate	ужившегося	ужившуюся	ужившееся	ужившихся
Accusative Inanimate	ужившийся	ужившуюся	ужившееся	ужившиеся
Instrumental	ужившимся	ужившейся	ужившимся	ужившимися
Prepositional	ужившемся	ужившейся	ужившемся	ужившихся

Pattern 374				
	Masculine	Feminine	Neuter	Plural
Nominative	образовавшийся	образовавшаяся	образовавшееся	образовавшиеся
Genitive	образовавшегося	образовавшейся	образовавшегося	образовавшихся
Dative	образовавшемуся	образовавшейся	образовавшемуся	образовавшимся
Accusative Animate	образовавшегося	образовавшуюся	образовавшееся	образовавшихся
Accusative Inanimate	образовавшийся	образовавшуюся	образовавшееся	образовавшиеся
Instrumental	образовавшимся	образовавшейся	образовавшимся	образовавшимися
Prepositional	образовавшемся	образовавшейся	образовавшемся	образовавшихся

APPENDIX D

Transfer Rules

- Noun phrase: [{adverb} + adjective] + [adjective] + noun + [{adjective} + genitive noun] + [preposition + noun]

  [{1:AV_N_*} 2:AJ_(1)_(2)_*] [3:AJ_(1)_(2)_*] 4:NO_(1)_(2)_* [{5:AJ_G_(3)_*} 6:NO_G_(3)_*] [7:EP_(4) 8:NO_(4)_(5)_*]

  4A [{1%} 2%] [3%] 4N [6A {5%} 6N] [7% 8A 8N]

- Verb phrase: [{adverb} + adjective] + [adjective] + noun + verb + [{adverb} + adjective] + [adjective] + noun

  [{1:AV_N_*} 2:AJ_N_(1)_*] [3:AJ_N_(1)_*] 4:NO_N_(1)_* 5:VB_*_*_(2) [{6:AV_N_*} 7:AJ_(2)_(3)_*] [8:AJ_(2)_(3)_*] 9:NO_(2)_(3)_*

  4A [{1%} 2%] [3%] 4N 5V 9A [{6%} 7%] [8%] 9N

- Adverb + adverb + adjective + adjective + adverb + adjective + noun + verb

  1:AV_*_* 2:AV_*_* 3:AJ_*_(1) 4:AJ_*_(1) 5:AV_*_* 6:AJ_*_(1) 7:NO_*_(1)_* 8:VB_*_*_*

  1% 2% 7A 3% 4% 5% 6% 7N 8V

- Adjective + verb + noun + preposition + noun + relative pronoun + adverb + adverb + verb + preposition + adjective + noun + preposition + possessive pronoun + noun

  1:AJ_N_*_* 2:VB_*_*_* 3:NO_*_P_* 4:EP_(2) 5:NO_(2)_*_* 6:PR_*_* 7:AV_*_* 8:AV_*_* 9:VB_*_*_* 10:EP_(4) 11:AJ_(4)_*_* 12:NO_(4)_*_* 13:EP_(5) 14:PO_X_P 15:NO_(5)_*_*

  1% 2V 3N 4% 5A 5N 6% 7% 9V 8% 10% 12A 11% 12N 13% 14% 15N

- Adjective + noun + adjective.genitive + noun.genitive + verb + noun + verb + adjective + noun

  1:AJ_(1)_(2)_* 2:NO_(1)_(2)_* 3:AJ_G_M_* 4:NO_G_M_* 5:VB_*_*_(3) 6:NO_(3)_*_* 7:VB_*_(4) 8:AJ_(5)_*_* 9:NO_(5)_*_*

  2A 2N 4A 3% 4N 1% 5V 6A 6N 7V 9A 8% 9N

- Noun phrase: [{adverb} + adjective] + [adjective] + noun + [{adverb} + {adjective} + {adjective} + genitive noun]

  [{1:AV_N_*} 2:AJ_(1)_(2)_*] [3:AJ_(1)_(2)_*] 4:NO_(1)_(2)_* [{5:AV_N_*} {6:AJ_G_(3)_*} {7:AJ_G_(3)_*} 8:NO_G_(3)_*]

  4A [{1%} 2%] [3%] 4N [8A {5%} {6%} {7%} 8N]

- Noun + adverb + verb + verb.infinitive + noun + noun.genitive + verb + noun + noun.genitive + adverb + adverb + preposition.prepositional + noun + adverb + verb + adjective + noun + adjective.genitive + noun.genitive

  1:NO_N_*_* 2:AV_*_* 3:VB_*_*_* 4:VB_*_(1) 5:NO_(1)_*_* 6:NO_G_*_* 7:VB_*_(2) 8:NO_(2)_*_* 9:NO_G_*_*
  10:AV_*_* 11:AV_*_* 12:PO_X_P 13:NO_*_*_* 14:AV_*_* 15:VB_*_* 16:AJ_*_*_* 17:NO_*_*_* 18:AJ_G_*_*
  19:NO_G_M_*

  1A 1N 2% 3V 4V 5A 5N 6A 6N 7V 8A 8N 9A 9N 10% 11% 12% 13A 13N 14% 15V 16%
  17N 19A 18% 19N

- Adjective + adjective + noun + verbal adjective.past passive + preposition + adjective + noun + verb + preposition + noun + adjective + adjective + noun

  1:AJ_(1)_(2)_* 2:AJ_(1)_(2)_* 3:NO_(1)_(2)_* 4:VJ_AP_*_(2)_* 5:EP_(3) 6:AJ_(3)_*_* 7:NO_(3)_*_* 8:VB_*_*_* 9:EP_(4)
  10:NO_(4)_*_* 11:AJ_(5)_*_* 12:AJ_*_*_* 13:NO_*_P_*

  1% 2% 3N 4J 5% 6% 7N 8V 9% 10N 11% 13A 12% 13N

- Preposition + adjective + preposition + adjective + noun + verb + adverb + noun + noun.genitive

  1:EP_(1) 2:AJ_(1)_*_0 3:EP_(2) 4:AJ_(2)_*_0 5:NO_(2)_(3)_* 6:VB_*_(3)_* 7:AV_* 8:NO_*_*_* 9:NO_G_*_*

  1% 2% 3% 4% 5N 7A 7N 8A 8N 9A 9N 6V

- Simple Noun phrase: [{adverb} + adjective] + [adjective] + noun

  [{1:AV_N_*} 2:AJ_(1)_(2)_*] [3:AJ_(1)_(2)_*] 4:NO_(1)_(2)_*

  4A [{1%} 2%] [3%] 4N

APPENDIX E

Bilingual Lexicon

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
а	10	but	0	1	0	0	
абсолют	9	absolute	124	1	0	0	
абстракт	1	abstract	6	1	0	4996	
аппарат	1	apparatus	6	1	0	5252	
актив	9	active	124	1	0	0	
анализ	1	analysis	6	1	0	5268	
анализир	8	analyze.analyzes.analyzed.analyzing.analyzed	186	21	8	0	
американск	9	American	125	1	0	0	
англоговорящ	9	English-speaking	121	1	0	0	
амплитуд	1	amplitude	50	2	0	5044	
анод	1	anode	6	1	0	4996	
автомат	1	automatic device	6	1	0	4996	
автоматическ	9	automatic	125	1	0	0	
автомобил	1	automobile	33	1	0	4996	
атом	1	atom	6	1	0	4996	
б	8	take.takes.took.taking.taken	184	21	8	0	
бел	9	white	120	1	0	0	
бел	9	white	120	1	0	0	
биологическ	9	biological	125	1	0	0	
близ	9	near	123	1	0	0	
больш	9	big	122	1	0	0	
брат	2	brother	8	1	0	4929	
бумаг	1	paper	51	2	0	4900	
быва	8	happen.happens.happened.happening.happened	160	21	1	0	
быстро	10	quickly	0	1	0	0	
в	11	in	0	0	16	0	
в	8	conduct.conducts.conducted.conducting.conducted	195	273	8	0	
в	8	conducted	228	321	0	0	
в	8	take.took.taken	185	25	8	0	
в	9	all/entire	140	1	0	0	
ва	3	you	401	7	0	0	
велик	9	great	121	1	0	0	
вес	1	weight	6	1	0	4212	
важ	9	important	124	1	0	0	
веществ	1	substance	90	3	0	4932	
вечер	1	evening	24	1	0	5008	
ви	8	see.sees.saw.seeing.seen	169	21	8	0	
ви	8	seen	216	69	0	0	
вид	1	species	6	1	0	4176	(+4, <animal>)
вид	1	view	1	1	0	4944	(+1, 'на')
вид	14	seeing.seen	255	1	8	0	
вид	15	see.sees.saw.seeing.seen	322	33	8	0	
во	11	in	0	0	16	0	
во	8	conduct.conducts.conducted.conducting.conducted	190	145	8	0	
вод	1	water	50	2	0	4900	
вне	11	outside	0	0	1	0	
вопрос	1	question	6	1	0	4932	
волн	1	wave	50	2	0	4932	
воздух	1	air	6	1	0	4132	
всегда	10	always	0	1	0	0	
врем	1	time	95	3	0	4208	
врач	2	doctor	9	1	0	4929	
вы	3	you	400	7	0	0	
выжив	14	surviving.survived	257	1	0	0	
высок	9	high	125	1	0	0	
высот	1	height	50	2	0	4932	
выуч	8	learn.learned.learned	219	41	4	0	
выучив	15	learn.learns.learned.learning.learned	374	290	4	0	
вычислител	9	computational	126	1	0	0	
геро	2	hero	38	1	0	5185	
газ	1	gas	6	1	0	5220	
газет	1	newspaper	50	2	0	4932	
глав	9	main	124	1	0	0	
глаз	1	eye	4	1	0	4996	
говор	14	speaking.spoken	254	1	12	0	
говор	15	speak.speaks.spoke.speaking.spoken	291	9	12	0	
говор	15	speak.speaks.spoke.spoking.spoken	310	17	4	0	
говор	15	speak.speaks.spoke.spoking.spoken	311	17	12	0	
говор	8	speak.speaks.spoke.speaking.spoken	161	21	12	0	
город	1	city	10	1	0	5446	
гост	2	guest	37	1	0	4929	
градус	1	degree	6	1	0	4948	
групп	1	group	50	2	0	4932	
д	1	day	34	1	0	4944	
да	15	give.gives.gave.giving.given	312	17	4	0	
да	8	give.gave.given	173	25	12	0	
да	8	give.gives.gave.giving.given	164	21	12	0	

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
дела	8	do.does.did.doing.did	160	21	12	0	
Даник	2	Danik	25	1	0	4961	
демонстрир	8	demonstrate.demonstrates.demonstrated.demonstrating.demonstrated	186	21	8	0	
девиц	2	girl	61	2	0	4929	
давно	10	long ago	0	1	0	0	
даже	10	even	0	1	0	0	
дерев	1	tree	98	3	0	4932	
дерев	1	wood	98	3	0	4132	
датчик	1	indicator	14	1	0	4996	
диск	1	disk	14	1	0	4964	
добр	9	nice	120	1	0	0	
длин	1	length	50	2	0	4976	
доктор	2	doctor	10	1	0	4929	
для	11	for	0	0	1	0	
довольно	10	quite	0	1	0	0	
дожд	1	rain	31	1	0	4900	
дорог	1	road	51	2	0	4934	
доч	2	daughter	76	2	0	4936	
двер	1	door	72	2	0	4932	
дру	2	friend	11	1	0	4929	
е	3	her	398	4	0	0	
е	3	him	394	3	0	0	
е	3	it	394	5	0	0	
е	8	eat.eats.ate.eating.eaten	179	21	8	0	
е	8	go.goes.went.going.gone	194	277	0	0	
если	12	if	0	3	0	0	
ж	2	wife	56	2	0	5953	
жд	8	wait.waits.waited.waiting.waited	198	21	10	0	
жи	8	live.lives.lived.living.lived	163	21	0	0	
жидкост	1	liquid	72	2	0	4964	
жив	15	live.lives.lived.living.lived	324	33	0	0	
журнал	1	journal	6	1	0	4164	
заб	8	forget.forgot.forgotten	168	25	8	0	
здани	1	building	91	3	0	4934	
запас	1	reserve	6	1	0	4932	
закр	15	close.closes.close.closing.closed	341	65	0	0	
закр	8	close.closed.closed	170	25	8	0	
заряд	1	charge	6	1	0	4948	
зачастую	10	often	0	1	0	0	
зна	8	know.knows.knew.knowing.knew	160	21	8	0	
зв	1	star	60	2	0	4934	
звук	1	sound	14	1	0	4932	
зрени	1	view	91	3	0	4948	
и	10	and	0	1	0	0	
и	8	search.searches.searched.searching.searched	188	21	10	0	
ид	8	go.goes.went.going.gone	192	277	0	0	
их	4	their	150	4	0	0	
им	3	him	395	3	0	0	
им	3	it	395	5	0	0	
индюк	2	turkey	14	1	0	4936	
импульс	1	impulse	6	1	0	5012	
или	10	or	0	1	0	0	
иногда	10	sometimes	0	1	0	0	
интерес	9	interesting	124	1	0	0	
истинн	9	true	120	1	0	0	
исследовани	1	research work	91	3	0	4980	
исследовател	2	researcher	33	1	0	4929	
избыт	1	abundance	7	1	0	5008	
извест	9	well-known	124	1	0	0	
к	12	who	380	1	0	0	
кажд	9	every	120	1	0	0	
катод	1	cathode	6	1	0	4932	
каш	1	cough	36	1	0	4932	
кил	1	keel	33	1	0	4932	
кла	8	put.puts.put.putting.put	171	21	8	0	
книг	1	book	51	2	0	4932	
кон	1	end	12	1	0	5012	
кол	1	ring	93	3	0	4932	
колеба	8	oscillate.oscillates.oscillated.oscillating.oscillated	160	21	8	0	
колеба	8	oscillated	210	69	0	0	
ков	1	carpet	18	1	0	4964	
котор	6	which	410	0	0	0	~(5,<PERSON>)
котор	6	who	410	0	0	0	(5,<PERSON>)
кош	2	cat	54	2	0	4936	
кра	1	edge	30	1	0	4870	
красив	9	beautiful	120	1	0	0	
куп	15	buy.buys.bought.buying.bought	340	65	0	0	

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
л	1	ice	3	1	0	4132	
л	8	lay.laid.lain	177	9	0	0	
ле	8	fly.flies.flew.flying.flown	196	277	0	0	
люд	2	people	112	4	0	4129	
лекци	1	lecture	71	2	0	4934	
Лен	2	Lena	50	2	0	4961	
лес	1	forest	5	1	0	4934	
линз	1	lens	50	2	0	5188	
литературу	1	literature	50	2	0	4100	
лун	1	moon	50	2	0	4934	
м	3	me	391	1	0	0	
мам	2	mom	52	2	0	865	
мальчик	2	boy	10	1	0	4929	
мас	1	oil	94	3	0	5028	
мат	2	mother	77	2	0	4961	
металл	1	metal	6	1	0	4964	
материал	1	material	6	1	0	4932	
масс	1	mass	50	2	0	5232	
мест	1	place	90	3	0	4930	
метро	1	subway	100	3	0	4868	
машин	1	machine	50	2	0	4932	
машинн	9	machine	120	1	0	0	
миханизм	1	mechanism	6	1	0	4932	
мил	9	gentle	120	1	0	0	
мо	4	my	150	1	0	0	
мног	9	many	125	1	0	0	
мор	1	sea	92	3	0	4934	
мотор	1	motor	6	1	0	4932	
мощн	9	powerful	120	1	0	0	
муж	2	husband	17	1	0	4929	
музе	1	museum	32	1	0	4934	
мы	3	we	390	6	0	0	
мыш	2	baby mouse	19	1	0	6984	
мыш	2	mouse	79	2	0	6984	
н	14	carrying.carried	253	1	8	0	
нём	3	him	396	3	0	0	
нём	3	it	396	5	0	0	
не	10	not	0	1	0	0	
на	11	of	0	0	4	0	(-1,'вид')
на	11	on	0	0	16	0	(3,[_P_])
на	11	onto	0	0	4	0	
на	3	us	401	6	0	0	
неб	1	sky	96	3	0	5382	
неб	1	sky	96	3	0	5446	
недел	1	week	70	2	0	4944	
наблюдени	1	observation	91	3	0	5008	
недоступ	9	unattainable	124	1	0	0	
ней	3	her	399	4	0	0	
невыгод	9	unsuitable	124	1	0	0	
нет	10	there are no	0	1	0	0	(+3,[_G_P])
неудач	1	failure	51	2	0	4980	
наук	1	science	51	2	0	4208	
настоящ	9	present	121	1	0	0	
научн	9	scientific	120	1	0	0	
наш	4	our	151	1	0	0	
нач	8	begin.began.begun	178	25	8	0	
них	3	them	404	8	0	0	
но	8	carry.carries.carried.carrying.carried	191	149	8	0	
о	11	about	0	0	16	0	
од	15	dress.dresses.dressed.dressing.dressed	342	65	0	0	
об	9	both	143	1	0	0	
од	9	one	142	1	0	0	
облак	1	cloud	101	3	0	4932	
област	1	area	72	2	0	5012	
оформля	8	form.forms.formed.forming.formed	160	21	8	0	
образ	1	form	6	1	0	4948	
образ	14	forming.formed	252	1	8	0	
образ	15	form.forms.formed.forming.formed	294	9	8	0	
образ	15	form.forms.formed.forming.formed	313	17	0	0	
образ	15	form.forms.formed.forming.formed	325	33	8	0	
образ	15	form.forms.formed.forming.formed	340	65	0	0	
образ	15	formed	364	12	0	0	
образ	8	form.forms.formed.forming.formed	186	21	8	0	
образовав	15	formed	374	164	0	0	
обычно	10	normally	0	1	0	0	
опять	10	again	0	1	0	0	
определённ	9	established	120	1	0	0	

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
океан	1	ocean	6	1	0	4998	
он	3	he	390	3	0	0	
она	3	she	390	4	0	0	
оно	3	it	393	5	0	0	
от	2	father	12	1	0	4961	
отличи	1	difference	91	3	0	4944	
отличн	9	excellent	120	1	0	0	
основ	1	basis	50	2	0	4112	
основ	9	basic	122	1	0	0	
отве	8	answer.answered.answered	165	25	4	0	
оста	8	remain.remains.remained.remaining.remained	214	37	16	0	
отрицател	9	negative	126	1	0	0	
отсутств	8	missing	186	69	8	0	
ошиб	1	error	58	2	0	4996	
очень	10	very	0	1	0	0	
очевидно	10	obviously	0	1	0	0	
очк	1	glasses	110	4	0	4132	
п	8	drink.drinks.drank.drinking.drunk	180	21	8	0	
п	8	sing.sings.sang.singing.sung	181	21	8	0	
пес	1	song	73	2	0	4932	
перево	8	translate.translates.translated.translating.translated	190	145	8	0	
перево	8	translated	236	197	0	0	
перевод	1	translation	6	1	0	4980	
перв	9	first	120	1	0	0	
пи	8	write.writes.wrote.writing.written	166	21	12	0	
пищ	1	food	51	2	0	4196	
по	11	about	0	0	2	0	
по	8	send.sent.sent	183	25	12	0	
по	8	understand.understood.understood	176	25	8	0	
по-английски	10	in English	0	1	0	0	
подержани	1	support	91	3	0	4944	
план	1	plan	6	1	0	4948	
подн	8	lift.lifted.lifted	197	25	8	0	
подобно	10	similar	0	1	2	0	
подтверждени	1	confirmation	91	3	0	4976	
популяр	9	popular	124	1	0	0	
покуп	1	purchase	58	2	0	4932	
пол	1	field	92	3	0	4934	
пол	1	floor	1	1	0	4934	
пол	1	gender	6	1	0	4880	
плох	9	bad	122	1	0	0	
помо	8	help.helped.helped	175	25	4	0	
половин	1	half	50	2	0	5716	
полож	8	put.put.put	172	25	8	0	
положител	9	positive	126	1	0	0	
помощ	1	help	72	2	0	4148	
пользовател	2	user	37	1	0	4929	
повышени	1	increase	91	3	0	5008	
пожалуй	10	perhaps	0	1	0	0	
потер	1	loss	70	2	0	5232	
последовател	9	successful	126	1	0	0	
поряд	1	order	7	1	0	5044	
постоянно	10	constantly	0	1	0	0	
поступател	9	progressive	126	1	0	0	
позволя	8	allow.allows.allowed.allowing.allowed	160	21	4	0	
почти	10	almost	0	1	0	0	
предел	1	limit	6	1	0	4948	
предмет	1	object	6	1	0	4996	
предложени	1	sentence	91	3	0	4932	
представлени	1	hypothesis	91	3	0	4180	
представля	8	present.presents.presented.presenting.presented	160	21	12	0	
правд	1	truth	50	2	0	4976	
превосходн	9	superb	120	1	0	0	
прибор	1	device	6	1	0	4932	
принцип	1	principle	6	1	0	4944	
принцип	1	principle	6	1	0	4944	
пример	1	example	6	1	0	4996	
примерно	10	approximately	0	1	2	0	
природ	1	nature	50	2	0	4134	
присоединени	1	combination	91	3	0	4928	
процесс	1	process	6	1	0	5200	
программ	1	program	50	2	0	4932	
провод	1	conductor	6	1	0	4932	
просматрыва	8	review.reviews.reviewed.reviewing.reviewed	160	21	8	0	
прост	9	simple	122	1	0	0	
просто	10	simply	0	1	0	0	
пространств	1	space	90	3	0	4902	

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
прочита	15	read.reads.read.reading.read	343	321	4	0	
прочита	8	read	210	73	0	0	
прочита	8	read.read.read	160	25	12	0	
прочитав	15	read	374	292	4	0	
пут	1	way	35	1	0	4944	
пут	1	way	35	1	0	4946	
ред	9	rare	123	1	0	0	
реакци	1	reaction	71	2	0	4948	
работ	1	work	50	2	0	4916	
ребят	2	children	113	4	0	4129	
ракет	1	rocket	50	2	0	4932	
ране	15	injure.injures.injured.injuring.injured	343	65	0	0	
рав	9	equal	124	1	0	0	
раствор	1	solution	6	1	0	4164	
разби	15	break.breaks.broke.breaking.broken	340	65	0	0	
размер	1	dimension	6	1	0	4880	
разнообраз	9	various	124	1	0	0	
разност	1	difference	72	2	0	4944	
разработанн	9	developed	120	1	0	0	
римлян	2	Roman	15	1	0	4929	
рол	1	role	33	1	0	4948	
ряд	1	series	1	1	0	4180	
ружь	1	rifle	102	3	0	4932	
русск	9	Russian	125	1	0	0	
рын	1	market	7	1	0	4946	
рыб	2	fish	52	2	0	5224	
с	1	village	97	3	0	4934	
с	2	sister	55	2	0	4929	
с	8	sit.sat.sat	182	25	0	0	
сам	9	most	120	1	0	0	
самолёт	1	airplane	6	1	0	4996	
середин	1	center	50	2	0	4870	
сп	8	sleep.sleeps.slept.sleeping.slept	189	21	0	0	
спеш	15	hurry.hurries.hurried.hurrying.hurried	290	9	0	0	
способ	1	means	6	1	0	4180	
ска	8	say.said.said	162	25	12	0	
схем	1	scheme	50	2	0	4932	
сил	1	strength	50	2	0	4912	
син	9	blue	130	1	0	0	
сил	9	strong	126	1	0	0	
систем	1	system	50	2	0	4932	
собак	2	dog	53	2	0	4936	
согласи	1	agreement	91	3	0	5008	
сохранени	1	conservation	91	3	0	4916	
сло	1	layer	38	1	0	4934	
слов	1	word	90	3	0	4932	
словар	1	dictionary	31	1	0	5444	
совсем	10	completely	0	1	0	0	
случа	1	case	32	1	0	4944	
слыша	14	hearing.heard	251	1	8	0	
свет	1	light	1	1	0	4964	
сво	9	own	121	1	0	0	
свобод	9	free	124	1	0	0	
связ	1	connection	72	2	0	4948	
ста	8	become.became.become	174	25	16	0	
сред	1	medium	50	2	0	6228	
судь	2	judge	40	1	0	4929	
стать	1	article	75	2	0	4996	
стол	1	table	6	1	0	4932	
стул	1	chair	22	1	0	4932	
стро	8	build.builds.built.building.built	161	21	8	0	
структур	1	structure	50	2	0	4932	
структур	1	structure	50	2	0	4948	
сущност	1	essence	79	2	0	4912	
сын	2	son	16	1	0	4929	
т	3	you	392	2	0	0	
т	7	that	156	0	0	0	~(1,[_P_])
т	7	those	156	0	0	0	(1,[_P_])
тёт	2	aunt	74	2	0	4993	
так	10	so	0	1	0	0	
техник	1	technology	51	2	0	5424	
также	10	also	0	1	0	0	
тел	2	calf	20	1	0	5704	
температур	1	temperature	50	2	0	4884	
теори	1	theory	71	2	0	5460	
термин	1	term	6	1	0	4932	
тигр	2	tiger	13	1	0	4936	

Source Stem	Part of Speech	Target Stem	Inflection Pattern	Primary Attribute	Secondary Attribute	Transfer Attribute	Disambiguation Expression
только	10	only	0	1	0	0	
точ	1	point	54	2	0	4932	
точ	1	point	57	2	0	4948	
тяжёл	9	heavy	120	1	0	0	
трет	9	third	131	1	0	0	
труб	1	tube	58	2	0	4932	
трудност	1	difficulty	72	2	0	1392	
трус	1	underwear	114	4	0	4132	
у	11	at	0	0	1	0	
удивл	15	surprise.surprises.surprised.surprising.surprised	346	65	4	0	
удоб	9	comfortable	124	1	0	0	
уг	1	coal	39	1	0	4900	
уг	1	corner	2	1	0	4932	
университет	1	university	6	1	0	5446	
уви	8	see.sees.saw.seeing.seen	216	73	0	0	
увид	15	see.sees.saw.seeing.seen	345	65	0	0	
уже	10	already	0	1	0	0	
успех	1	success	23	1	0	5232	
уров	1	level	34	1	0	4948	
устойчив	9	stable	120	1	0	0	
устройств	1	construction	90	3	0	4932	
ущел	1	canyon	103	3	0	4934	
уч	15	study.studies.studied.studying.studied	360	10	4	0	
уч	8	study.studies.studied.studying.studied	219	37	4	0	
уча	15	study.studies.studied.studying.studied	370	162	4	0	
участи	1	participation	91	3	0	4132	
факт	1	fact	6	1	0	4944	
форм	1	form	50	2	0	4980	
функци	1	function	70	2	0	4932	
функционир	8	function.functions.functioned.functioning.functioned	186	21	8	0	
характер	9	characteristic	124	1	0	0	
химическ	9	chemical	125	1	0	0	
хо	8	go.goes.went.going.gone	190	145	0	0	
хо	8	want.wants.wanted.wanting.wanted	199	21	8	0	
холод	9	cold	124	1	0	0	
хорош	9	good	121	1	0	0	
цел	1	goal	72	2	0	4948	
церк	1	church	78	2	0	5222	
ч	12	what	381	2	0	0	
ч	9	whose	141	1	0	0	
чёрн	9	black	120	1	0	0	
человек	2	person	21	1	0	4929	
час	1	clock	114	4	0	4164	
черепах	2	turtle	51	2	0	4936	
частиц	1	particle	50	2	0	4932	
часто	10	often	0	1	0	0	
частот	1	frequency	50	2	0	5444	
чита	14	reading.read	250	9	12	0	
чита	14	reading.read	270	12	4	0	
чита	15	read.reads.read.reading.read	293	137	12	0	
чита	15	read.reads.read.reading.read	310	145	4	0	
чита	15	read.reads.read.reading.read	320	161	12	0	
чита	15	read.reads.read.reading.read	343	193	4	0	
чита	8	read	210	69	0	0	
чита	8	read.reads.read.reading.read	160	21	12	0	
читав	15	read	374	164	4	0	
чист	9	pure	120	1	0	0	
что	16	that	0	0	0	0	(-2,',')
ш	8	/./.went.gone	193	277	0	0	
шаг	1	step	1	1	0	4948	
ширин	1	width	50	2	0	4948	
широк	9	wide	125	1	0	0	
штанг	1	rod	51	2	0	4932	
шум	1	noise	6	1	0	4976	
шток	1	rod	23	1	0	4932	
щ	1	cheek	59	2	0	4932	
эксперимент	1	experiment	6	1	0	4996	
электрод	1	electrode	6	1	0	4996	
элемент	1	element	6	1	0	4996	
энерги	1	energy	70	2	0	5428	
эт	7	it	155	0	0	0	
эт	7	this	155	0	0	0	
я	3	I	390	1	0	0	
яблок	1	apple	99	3	0	4996	
явлени	1	phenomenon	91	3	0	6228	
ярк	9	bright	125	1	0	0	
ясн	9	clear	120	1	0	0	

APPENDIX F

Sample Translation

Figure F-1: Original English Abstract

A large volume of scientific work published in Russian remains essentially unavailable to the English-speaking world. Often only the most popular or well-known work is translated. Moreover, in many cases, even the abstracts remain untranslated. This presents a difficult situation for researchers who may not be aware of Russian work in their field. This machine translation system allows users to survey Russian abstracts. It also provides various tools to build the program's vocabulary, to design how sentences are structured and translated, and to demonstrate many machine translation concepts.

Figure F-2: Abstract in Russian

AabX]Z aUgkaoZ dUVbfo$ bcgV_]^bWUaaoZ aU dgee^b` s\o^Z$ befU r fes W e gmabef] aZYbefgcao`] UaX_bXbWbdsm]` ]ee_ZYbWUfZ_s`& LUefb fb_p^b eU`oZ c bcg_sdaoZ ]_] ]\WZefaoZ ]ee_ZYbWUa]s cZdZWbYsfes& ?db`Z fbXb$ Wb `abX]i e_gk Usi bfegfefWg r f YU[Z cZdZWbYo UVefdU^fbW& Rfb cdZYefUW_sZf fdgYabef] Y_s ]e e_ZYbWUfZ_Z½$ ^bfbdoZ$ \UkUefg r aZ \aU r f b dgee^]i dUVbfUi W eWb]i bV_Uefsi& EU\dUVbfUaaUs e]efZ`U `U l ]aabXb cZdZWbYU cb\Wb_sZf cb_p\bWUfZ_s` cdbe`Ufd oWUfp dgee^]Z UVefdU^fo& F]efZ`U fU^[Z cb\Wb_sZf efdb]fp e_bWUdp cdbXdU``o $ bhbd`_sfp efdg^fgdg cdZY_b[Za]½$ U fU^[Z]i cZdZWbYbW ] YZ`baefd]dbWUfp d U\abbVdU\aoZ cd]aj]co `U l]aabXb cZdZWbYU&

Figure F-3: Program-Generated Transliteration from Russian

Mnogie nauchnye raboty, opublikovannye na russkom jazyke, ostajutsja v sushchnosti nedostupnymi anglogovorjashchim issledovateljam. Chasto tol'ko samye populjarnye ili izvestnye issledovanija perevodjatsja. Krome togo, vo mnogix sluchajax otsutstvujut dazhe perevody abstraktov. Eto predstavljaet trudnosti dlja issledovatelej, kotorye, zachastuju ne znajut o russkix rabotax v ix oblastjax. Razrabotannaja sistema mashinnogo perevoda pozvoljaet pol'zovateljam prosmatryvat' russkie abstrakty. Sistema takzhe pozvoljaet stroit' slovar' programmy, oformljat' strukturu predlozhenij, a takzhe ix perevodov i demonstrirovat' raznoobraznye printsipy mashinnogo perevoda.

Figure F-4: Program-Generated Translation from Russian

Many scientific works, being published in Russian, remain in essence unattainable to the English-speaking researchers. Often only the most popular or well-known research works are translated. Moreover, in many cases even the translations of the abstracts are missing. This presents difficulties for the researchers, who, often know not about the Russian work in their areas. The machine translation system developed allows to the users to review the Russian abstracts. The system also allows to build the dictionary of the program, to form the structure of the sentences, as well as their translations and to demonstrate various principles of a machine translation.

Figure F-5: Transfer Details

```
Transfer Details Log of 'SEMINAR.ABS' (11.03.96 22:22:28)
_____________________________________________________________________________
Sentence: Многие научные работы, опубликованные на русском языке, остаются в
          сущности недоступными англоговорящим исследователям.
                                                                        [0:00]
  Lookup: Многие

     Got= мног (many)
      SC= AJ_N_P_0  AJ_2_P_0

  Lookup: научные

     Got= научн (scientific)
      SC= AJ_N_P_0  AJ_2_P_0

  Lookup: работы

     Got= работ (work)
      SC= NO_G_F_0  NO_N_P_0  NO_A_P_0

  Lookup: опубликованные

     Got= опублик (publish.publishes.published.publishing.published)
      SC= VJ_AP_N_P_0  VJ_AP_2_P_0

  Lookup: на

     Got= на (onto)
      SC= EP_A

     Got= на (of)
      SC= EP_A

     Got= на (on)
      SC= EP_P

  Lookup: русском

     Got= русск (Russian)
      SC= AJ_P_M_0  AJ_P_N_0

  Lookup: языке

     Got= язык (language)
      SC= NO_P_M_0

  Lookup: остаются

     Got= оста (remain.remains.remained.remaining.remained)
      SC= VB_3_P_I

  Lookup: в

     Got= в (in)
      SC= EP_P

  Lookup: сущности

     Got= сущност (essence)
      SC= NO_G_F_0  NO_D_F_0  NO_P_F_0  NO_N_P_0

  Lookup: недоступными

     Got= недоступ (unattainable)
      SC= AJ_I_P_0
  Lookup: англоговорящим
```

```
   Got= англоговорящ (English-speaking)
    SC= AJ_I_M_0  AJ_I_N_0  AJ_D_P_0

Lookup: исследователям

   Got= исследовател (researcher)
    SC= NO_D_P_0

Sentence disambiguation:

   (на) disambiguated as (on).

Sentence has 576 possible symbolic constant interpretations.

Sentence transfer:

  Symbolic Constants:                                              [0:01]
    AJ_N_P_0  AJ_N_P_0  NO_N_P_0  VJ_AP_N_P_0  EP_P  AJ_P_M_0  NO_P_M_0
    VB_3_P_I  EP_P  NO_G_F_0  AJ_I_P_0  AJ_I_M_0  NO_D_P_0

  Transfer Rule #130 matched:
     Russian= 1:AJ_(1)_(2)_* 2:AJ_(1)_(2)_* 3:NO_(1)_(2)_* 4:VJ_AP_*_(2)_*
              5:EP_(3) 6:AJ_(3)_*_* 7:NO_(3)_*_* 8:VB_*_*_* 9:EP_(4)
              10:NO_(4)_*_* 11:AJ_(5)_*_* 12:AJ_*_*_* 13:NO_*_P_*
     English= 1%  2%  3N  4J  5%  6%  7N  8V  9%  10N  11%  13A  12%  13N
                                                                   [0:01]
Raw Translation:

  Many scientific works, being published on Russian language, remain in
  essence unattainable to the English-speaking researchers.
```

```
Sentence: Часто только самые популярные или известные исследования
          переводятся.
                                                                   [0:01]
Lookup: Часто

   Got= часто (often)
    SC= AV_N_0

Lookup: только

   Got= только (only)
    SC= AV_N_0

Lookup: самые

   Got= сам (most)
    SC= AJ_N_P_0  AJ_2_P_0

Lookup: популярные

   Got= популяр (popular)
    SC= AJ_N_P_0  AJ_2_P_0

Lookup: или

   Got= или (or)
    SC= AV_N_0

Lookup: известные

   Got= извест (well-known)
    SC= AJ_N_P_0  AJ_2_P_0



Lookup: исследования
```

```
   Got= исследовани (research work)
    SC= NO_G_N_0  NO_N_P_0  NO_A_P_0

Lookup: переводятся

   Got= перево (translated)
    SC= VB_3_P_0

Sentence disambiguation:

   No ambiguous words.

Sentence has 24 possible symbolic constant interpretations.

Sentence transfer:

  Symbolic Constants:                                                [0:02]
    AV_N_0  AV_N_0  AJ_N_P_0  AJ_N_P_0  AV_N_0  AJ_N_P_0  NO_N_P_0
    VB_3_P_0

  Transfer Rule #126 matched:
    Russian= 1:AV_*_* 2:AV_*_* 3:AJ_*_(1) 4:AJ_*_(1) 5:AV_*_* 6:AJ_*_(1)
             7:NO_*_(1)_* 8:VB_*_*_*
    English= 1%  2%  7A  3%  4%  5%  6%  7N  8V
                                                                     [0:02]
Raw Translation:

  Often only the most popular or well-known research works are translated.
_____________________________________________________________________________
Sentence: Кроме того, во многих случаях отсутствуют даже переводы
          абстрактов.
                                                                     [0:02]
Lookup: Кроме

   Got= кроме (besides)
    SC= EP_G

Lookup: того

   Got= т (those)
    SC= AJ_G_M_0  AJ_1_M_0  AJ_G_N_0  AJ_A_N_0

   Got= т (that)
    SC= AJ_G_M_0  AJ_1_M_0  AJ_G_N_0  AJ_A_N_0

Lookup: во

   Got= во (in)
    SC= EP_P

Lookup: многих

   Got= мног (many)
    SC= AJ_G_P_0  AJ_1_P_0  AJ_P_P_0

Lookup: случаях

   Got= случа (case)
    SC= NO_P_P_0

Lookup: отсутствуют

   Got= отсутств (missing)
    SC= VB_3_P_A


Lookup: даже

   Got= даже (even)
```

```
     SC= AV_N_0

  Lookup: переводы

     Got= перевод (translation)
      SC= NO_N_P_0  NO_A_P_0

  Lookup: абстрактов

     Got= абстракт (abstract)
      SC= NO_G_P_0

  Sentence disambiguation:

     (того,) disambiguated as (that).

  Sentence has 24 possible symbolic constant interpretations.

  Sentence transfer:

    Symbolic Constants:                                              [0:02]
      EP_G  AJ_G_M_0  EP_P  AJ_G_P_0  NO_P_P_0  VB_3_P_A  AV_N_0  NO_N_P_0
      NO_G_P_0

    Transfer Rule #127 matched:
      Russian= 1:EP_(1) 2:AJ_(1)_*_0 3:EP_(2) 4:AJ_(2)_*_0 5:NO_(2)_(3)_*
               6:VB_*_(3)_* 7:AV_* 8:NO_*_*_* 9:NO_G_*_*
      English= 1%  2%  3%  4%  5N  7A  7N  8A  8N  9A  9N  6V
                                                                     [0:02]
  Raw Translation:

    Besides that, in many cases  even the translations of the abstracts are
    missing.
_____________________________________________________________________________
Sentence: Это представляет трудности для исследователей, которые, зачастую
          не знают о русских работах в их областях.
                                                                     [0:02]
  Lookup: Это

     Got= эт (this)
      SC= AJ_N_N_0  AJ_A_N_0

     Got= эт (it)
      SC= AJ_N_N_0  AJ_A_N_0

  Lookup: представляет

     Got= представля (present.presents.presented.presenting.presented)
      SC= VB_3_S_D  VB_3_S_A

  Lookup: трудности

     Got= трудност (difficulty)
      SC= NO_G_F_0  NO_D_F_0  NO_P_F_0  NO_N_P_0  NO_A_P_0

  Lookup: для

     Got= для (for)
      SC= EP_G

  Lookup: исследователей

     Got= исследовател (researcher)
      SC= NO_G_P_0

  Lookup: которые

     Got= котор (who)
      SC= PR_N_P  PR_2_P
```

```
   Got= котор (which)
    SC= PR_N_P  PR_2_P

Lookup: зачастую

   Got= зачастую (often)
    SC= AV_N_0

Lookup: не

   Got= не (not)
    SC= AV_N_0

Lookup: знают

   Got= зна (know.knows.knew.knowing.knew)
    SC= VB_3_P_A

Lookup: о

   Got= о (about)
    SC= EP_P

Lookup: русских

   Got= русск (Russian)
    SC= AJ_G_P_0  AJ_1_P_0  AJ_P_P_0

Lookup: работах

   Got= работ (work)
    SC= NO_P_P_0

Lookup: в

   Got= в (in)
    SC= EP_P

Lookup: их

   Got= их (their)
    SC= PO_X_P

Lookup: областях

   Got= област (area)
    SC= NO_P_P_0

Sentence disambiguation:

   (Это) randomly disambiguated as (this).
   (которые,) disambiguated as (who).

Sentence has 120 possible symbolic constant interpretations.

Sentence transfer:

  Symbolic Constants:                                              [0:03]
    AJ_N_N_0  VB_3_S_D  NO_N_P_0  EP_G  NO_G_P_0  PR_N_P  AV_N_0  AV_N_0
    VB_3_P_A  EP_P  AJ_P_P_0  NO_P_P_0  EP_P  PO_X_P  NO_P_P_0



  Transfer Rule #128 matched:
    Russian= 1:AJ_N_*_* 2:VB_*_*_* 3:NO_*_P_* 4:EP_(2) 5:NO_(2)_*_*
             6:PR_*_* 7:AV_*_* 8:AV_*_* 9:VB_*_*_* 10:EP_(4) 11:AJ_(4)_*_*
             12:NO_(4)_*_* 13:EP_(5) 14:PO_X_P 15:NO_(5)_*_*
```

```
      English= 1%  2V  3N  4%  5A  5N  6%  7%  9V  8%  10%  12A  11%  12N
               13%  14%  15N
                                                                      [0:03]
Raw Translation:

    This presents difficulties for the researchers, who, often know not
    about the Russian works in their areas.
_____________________________________________________________________________
Sentence: Разработанная система машинного перевода позволяет пользователям
          просматривать русские абстракты.
                                                                      [0:03]
  Lookup: Разработанная

     Got= разработанн (developed)
      SC= AJ_N_F_0

  Lookup: система

     Got= систем (system)
      SC= NO_N_F_0

  Lookup: машинного

     Got= машинн (machine)
      SC= AJ_G_M_0  AJ_1_M_0  AJ_G_N_0

  Lookup: перевода

     Got= перевод (translation)
      SC= NO_G_M_0

  Lookup: позволяет

     Got= позволя (allow.allows.allowed.allowing.allowed)
      SC= VB_3_S_D

  Lookup: пользователям

     Got= пользовател (user)
      SC= NO_D_P_0

  Lookup: просматривать

     Got= просматрыва (review.reviews.reviewed.reviewing.reviewed)
      SC= VB_I_A

  Lookup: русские

     Got= русск (Russian)
      SC= AJ_N_P_0  AJ_2_P_0

  Lookup: абстракты

     Got= абстракт (abstract)
      SC= NO_N_P_0  NO_A_P_0

  Sentence disambiguation:

     No ambiguous words.

  Sentence has 12 possible symbolic constant interpretations.


  Sentence transfer:

    Symbolic Constants:                                               [0:03]
      AJ_N_F_0  NO_N_F_0  AJ_G_M_0  NO_G_M_0  VB_3_S_D  NO_D_P_0  VB_I_A
      AJ_N_P_0  NO_N_P_0
```

```
    Transfer Rule #131 matched:
      Russian= 1:AJ_(1)_(2)_* 2:NO_(1)_(2)_* 3:AJ_G_M_* 4:NO_G_M_*
               5:VB_*_*_(3) 6:NO_(3)_*_* 7:VB_*_(4) 8:AJ_(5)_*_*
               9:NO_(5)_*_*
      English= 2A  2N  4A  3%  4N  1%  5V  6A  6N  7V  9A  8%  9N
                                                                    [0:03]
  Raw Translation:

    The system of the machine translation developed allows to the users to
    review the Russian abstracts.
_____________________________________________________________________________
Sentence: Система также позволяет строить словарь программы, оформлять
          структуру предложений, а также их переводов и демонстрировать
          разнообразные принципы машинного перевода.
                                                                    [0:03]
  Lookup: Система

     Got= систем (system)
      SC= NO_N_F_0

  Lookup: также

     Got= также (also)
      SC= AV_N_0

  Lookup: позволяет

     Got= позволя (allow.allows.allowed.allowing.allowed)
      SC= VB_3_S_D

  Lookup: строить

     Got= стро (build.builds.built.building.built)
      SC= VB_I_A

  Lookup: словарь

     Got= словар (dictionary)
      SC= NO_N_M_0  NO_A_M_0

  Lookup: программы

     Got= программ (program)
      SC= NO_G_F_0  NO_N_P_0  NO_A_P_0

  Lookup: оформлять

     Got= оформля (form.forms.formed.forming.formed)
      SC= VB_I_A

  Lookup: структуру

     Got= структур (structure)
      SC= NO_A_F_0

  Lookup: предложений

     Got= предложени (sentence)
      SC= NO_G_P_0



  Lookup: а

     Got= а (but)
      SC= AV_N_0

  Lookup: также
```

```
   Got= также (also)
    SC= AV_N_0

Lookup: их

   Got= их (their)
    SC= PO_X_P

Lookup: переводов

   Got= перевод (translation)
    SC= NO_G_P_0

Lookup: и

   Got= и (and)
    SC= AV_N_0

Lookup: демонстрировать

   Got= демонстрир (demonstrate.demonstrates.demonstrated.demonstrating.
                    demonstrated)
    SC= VB_I_A

Lookup: разнообразные

   Got= разнообраз (various)
    SC= AJ_N_P_0  AJ_2_P_0

Lookup: принципы

   Got= принцип (principle)
    SC= NO_N_P_0  NO_A_P_0

Lookup: машинного

   Got= машинн (machine)
    SC= AJ_G_M_0  AJ_1_M_0  AJ_G_N_0

Lookup: перевода

   Got= перевод (translation)
    SC= NO_G_M_0

Sentence disambiguation:

   No ambiguous words.

Sentence has 72 possible symbolic constant interpretations.

Sentence transfer:

  Symbolic Constants:                                                [0:03]
    NO_N_F_0  AV_N_0  VB_3_S_D  VB_I_A  NO_N_M_0  NO_G_F_0  VB_I_A
    NO_A_F_0  NO_G_P_0  AV_N_0  AV_N_0  PO_X_P  NO_G_P_0  AV_N_0  VB_I_A
    AJ_N_P_0  NO_N_P_0  AJ_G_M_0  NO_G_M_0

  Transfer Rule #129 matched:
    Russian= 1:NO_N_*_* 2:AV_*_* 3:VB_*_*_* 4:VB_*_(1) 5:NO_(1)_*_*
             6:NO_G_*_* 7:VB_*_(2) 8:NO_(2)_*_* 9:NO_G_*_* 10:AV_*_*
             11:AV_*_* 12:PO_X_P 13:NO_*_*_* 14:AV_*_* 15:VB_*_*
             16:AJ_*_*_* 17:NO_*_*_* 18:AJ_G_*_* 19:NO_G_M_*
    English= 1A  1N  2%  3V  4V  5A  5N  6A  6N  7V  8A  8N  9A  9N  10%
             11%  12%  13A  13N  14%  15V  16%  17N  19A  18%  19N
                                                                   [0:03]
Raw Translation:

  A system also allows to build the dictionary of the program, to form the
  structure of the sentences, but also their of the translations and to
```

demonstrate various principles of a machine translation.

Applying post-translation cleanup filters:

'on Russian language' replaced with 'in Russian'
'Besides that' replaced with 'Moreover'
'but also' replaced with 'as well as'
'system of the machine translation' replaced with 'machine translation system'
'their of the' replaced with 'their'

Translation complete: [0:04]

Many scientific works, being published in Russian, remain in essence unattainable to the English-speaking researchers. Often only the most popular or well-known research works are translated. Moreover, in many cases even the translations of the abstracts are missing. This presents difficulties for the researchers, who, often know not about the Russian work in their areas. The machine translation system developed allows to the users to review the Russian abstracts. The system also allows to build the dictionary of the program, to form the structure of the sentences, as well as their translations and to demonstrate various principles of a machine translation.

Words Translated: 73 of 73 (100%)
Sentences Translated: 6 of 6 (100%)
Avg Transfer Time: 0:01
Total Transfer Time: 0:04

APPENDIX G

Support Files

Table G-1 File List

Filename†	Extension	Description
CLEANUP	CLN	Post-Translation Cleanup Filters
CONFIG	CNF	Configuration Settings
HELP1	XLP	Cleanup Filter Editor Help
HELP2	XLP	Configuration Settings Help
HELP3	XLP	Import Filter Editor Help
HELP4	XLP	Lexicon Editor Help
HELP5	XLP	Transfer Rule Editor Help
IMPORT FILTERS	FTR	Import Filters
K8ARIAL	TTF	Cyrillic True-Type Font
KBDUSX	KBD	Cyrillic Keyboard Layout
LEXICON	LEX	Bilingual Lexicon
T-RULES	LOG	Uncompressed Transfer Rule
TRANSFER RULES	TRR	Compressed Transfer Rules
XLATE	XLG	Translation Details Log

† Extended Windows 95® filenames.

APPENDIX H

Source Code

Dan Tappan
15 March 1996

CSEG Master's Thesis

Russian-to-English Scientific Abstract Translator

This program demonstrates a prototype translation system. It provides a variety of editors (e.g., lexicon, transfer rule, character mapping, etc.) Translation is done using the direct approach to reorganize Russian sentence components into English.

```
Option Explicit

'_____________________________________________________________________________________________________
'                                           CONSTANTS
'_____________________________________________________________________________________________________

'_____ common dialog ________________________________________________________________________________
Global Const CMFILEDIALOG_HIDEREADONLY = &H4&                   'no READ-ONLY option
Global Const CMFILEDIALOG_NOVALIDATE = &H100&                   'don't check filename validity (for Win95)
Global Const CMFILEDIALOG_SHOWHELP = &H10&                      'include Help button
Global Const CMFILEDIALOG_PATHMUSTEXIST = &H800&                'path must exist for attempt to open
Global Const CMFILEDIALOG_FILEMUSTEXIST = &H1000&               'file must exist for attempt to open
Global Const CMFILEDIALOG_OVERWRITEPROMPT = &H2&                'confirm replace
Global Const CMDIALOG_CANCEL_ERROR = 32755                      'error returned by Cancel click

Global Const CMPRINTDIALOG_NOPAGENUMS = &H8&                    'don't prompt for page numbers to print
Global Const CMPRINTDIALOG_NOSELECTION = &H4&                   'no range can be selected
Global Const CMPRINTDIALOG_PRINTSETUP = &H40&

'_____ label properties ______________________________________________________________________________
Global Const STYLE_TRANSPARENT = 0                              'sets label background color to whatever it's on
Global Const STYLE_OPAQUE = 1                                   'sets label background to given color

'_____ error codes ___________________________________________________________________________________
Global Const FILEACCESS_FILENOTFOUND = 53                       'file not found error
Global Const FILEACCESS_PATHNOTFOUND = 76                       'path not found error

'_____ message box ___________________________________________________________________________________
Global Const MB_OK = 0                                          'display OK button only.
Global Const MB_OKCANCEL = 1                                    'display OK and Cancel buttons.
Global Const MB_ABORTRETRYIGNORE = 2                            'display Abort, Retry, and Ignore buttons.
```

```
Global Const MB_YESNOCANCEL = 3                         'display Yes, No, and Cancel buttons.
Global Const MB_YESNO = 4                               'display Yes and No buttons.
Global Const MB_RETRYCANCEL = 5                         'display Retry and Cancel buttons.

Global Const MB_ICONSTOP = 16                           'stop icon
Global Const MB_ICONQUESTION = 32                       'question mark icon
Global Const MB_ICONEXCLAMATION = 48                    'exclamation point icon
Global Const MB_ICONINFORMATION = 64                    'general information icon

Global Const MB_DEFBUTTON1 = 0                          'first button default
Global Const MB_DEFBUTTON2 = 256                        'second button default
Global Const MB_DEFBUTTON3 = 512                        'third button  default

Global Const ID_OK = 1                                  'OK button selected.
Global Const ID_CANCEL = 2                              'Cancel button selected.
Global Const ID_ABORT = 3                               'Abort button selected.
Global Const ID_RETRY = 4                               'Retry button selected.
Global Const ID_IGNORE = 5                              'Ignore button selected.
Global Const ID_YES = 6                                 'Yes button selected.
Global Const ID_NO = 7                                  'No button selected.

'----- colors ------------------------------------------------------------------------------------------------------
Global Const WHITE = &HFFFFFF
Global Const BLACK = &H0
Global Const DKGRAY = &H808080
Global Const LTGRAY = &HC0C0C0
Global Const LTSAND = &HBBCCD5
Global Const CREAM = &H80000005
Global Const SAND = &H688DA2
Global Const BACKGROUND_COLOR = LTSAND                  'background color on all forms

'----- program values ----------------------------------------------------------------------------------------------
Global Const PROGRAM_TITLE = "Scientific Abstract Translator" 'program name

Global Const NUM_MAPPINGS = 98                          'num characters-1 in mapping filter set

Global Const SUBSTITUTION_CHARACTER = "□"                'char to substitute for unknown mapping

Global Const INFLECTION_DELIMITER = "."                 'separator between case endings in packed string

Global Const CONJUGATION_DELIMITER = "."                'separator between target verb conjugations

Global Const MAX_ABSTRACT_CHARS = 50000                 'max chars in abstract

Global Const MAX_LEXICON_ENTRIES = 500                  'max entries in lexicon
```

```
Global Const FILTER_NAME = 0                          'list box control array pointers for
Global Const FILTER_MAP = 1                           'Filter Setup form

Global Const FULL_MATCH = 1                           'full match required on lookup
Global Const STEM_MATCH = 2                           'stem match        "

Global Const UNUSED = -1                              'unused index entry

Global Const IGNORE_WORD = -1                         'ignore word in sentence

Global Const NUM_INFLECTION_PATTERNS = 410            'num inflection patterns

Global Const TRANSLATION = 1                          'code indicating which document to print
Global Const TRANSLITERATION = 2
Global Const CONVERTED_ABSTRACT = 3
Global Const TRANSLATION_DETAILS = 4

Global Const ABSTRACT_MODE = 1                        'program operates in two modes.  E-mail used
Global Const EMAIL_MODE = 2                           'mainly for transliteration.

Global Const MAX_HEADER_LINES = 10                    'max header lines in lexicon

Global Const MAX_ASCII = 255                          'max chararacter set length

Global Const MAX_TRANSFER_RULES = 8192                'max supportable transfer rules

Global Const MAX_SENTENCE_COMPONENTS = 31             'max words-1 per sentence

Global Const MAX_MATCHES = 15                         'max number of ambiguous translations per word

Global Const RULE_COMPLEXITY = 2048                   'max decodable forms of single transfer rule

Global Const VERSION_NUMBER = "1.1"                   'current version number

'_____ part of speech codes before conversion to symbolic representation ____________________________________
Global Const NOUN_INANIMATE = 1                         'nouns
Global Const NOUN_ANIMATE = 2

Global Const PRONOUN_PERSONAL = 3                     'pronouns
Global Const PRONOUN_POSSESSIVE = 4
Global Const PRONOUN_INDEFINITE = 5
Global Const PRONOUN_RELATIVE = 6
Global Const PRONOUN_DEMONSTRATIVE = 7

Global Const VERB = 8                                 'verbs
```

```
Global Const ADJECTIVE = 9                                  'adjectives

Global Const ADVERB = 10                                    'adverbs

Global Const PREPOSITION = 11                               'prepositions

Global Const INTERROGATIVE = 12                             'interrogatives

Global Const INTERJECTION = 13                              'interjections

Global Const EMPHATIC_PARTICLE = 13                         'emphatic particles

Global Const VERBAL_ADVERB = 14                             'verbal adverbs

Global Const VERBAL_ADJECTIVE = 15                          'verbal adjectives

Global Const CONJUNCTION = 16                               'conjunctions

Global Const FIXED_EXPRESSION = 17                          'fixed expressions

'————— part of speech primary attributes before conversion to symbolic representation ——————————————————
Global Const NOUN_MASCULINE = 1                             'nouns
Global Const NOUN_FEMININE = 2
Global Const NOUN_NEUTER = 3
Global Const NOUN_PLURAL = 4

Global Const PERSONAL_PRONOUN_1ST_SINGULAR = 1              'personal pronouns
Global Const PERSONAL_PRONOUN_2ND_SINGULAR = 2
Global Const PERSONAL_PRONOUN_3RD_SINGULAR_MASCULINE = 3
Global Const PERSONAL_PRONOUN_3RD_SINGULAR_FEMININE = 4
Global Const PERSONAL_PRONOUN_3RD_SINGULAR_NEUTER = 5
Global Const PERSONAL_PRONOUN_1ST_PLURAL = 6
Global Const PERSONAL_PRONOUN_2ND_PLURAL = 7
Global Const PERSONAL_PRONOUN_3RD_PLURAL = 8

Global Const POSS_PRONOUN_DECLINABLE = 1                    'possessive pronouns
Global Const POSS_PRONOUN_INDECLINABLE_3RD_MASC_NEUT = 2
Global Const POSS_PRONOUN_INDECLINABLE_3RD_FEMININE = 3
Global Const POSS_PRONOUN_INDECLINABLE_3RD_PLURAL = 4

Global Const VERB_MAIN = 1                                  'verbs
Global Const VERB_AUXILIARY = 2
Global Const VERB_IMPERFECTIVE = 4
Global Const VERB_PERFECTIVE = 8
Global Const VERB_NONREFLEXIVE = 16
Global Const VERB_REFLEXIVE_ACTIVE = 32
```

```
Global Const VERB_REFLEXIVE_PASSIVE = 64
Global Const VERB_1ST_IMPERATIVE = 6
Global Const VERB_INDETERMINATE = 128
Global Const VERB_DETERMINATE = 256

Global Const ADJECTIVE_LONG_FORM = 1                    'adjectives
Global Const ADJECTIVE_SHORT_FORM = 2
Global Const ADJECTIVE_COMPARATIVE = 3
Global Const ADJECTIVE_SUPERLATIVE = 4

Global Const ADVERB_NORMAL = 1                          'adverbs
Global Const ADVERB_COMPARATIVE = 2
Global Const ADVERB_SUPERLATIVE = 3

Global Const INTERROGATIVE_NOMINATIVE = 1               'interrogatives
Global Const INTERROGATIVE_GENITIVE = 2
Global Const INTERROGATIVE_DATIVE = 3
Global Const INTERROGATIVE_ACCUSATIVE = 4
Global Const INTERROGATIVE_INSTRUMENTAL = 5
Global Const INTERROGATIVE_PREPOSITIONAL = 6
Global Const INTERROGATIVE_MASCULINE = 1
Global Const INTERROGATIVE_NEUTER = 2
Global Const INTERROGATIVE_INDECLINABLE = 3

Global Const INTERJECTION_RETAIN = 1                    'interjections
Global Const INTERJECTION_IGNORE = 2

Global Const VERBAL_ADVERB_SIMULTANEOUS = 1             'verbal adverbs
Global Const VERBAL_ADVERB_SEQUENTIAL = 2
Global Const VERBAL_ADVERB_NON_REFLEXIVE = 1
Global Const VERBAL_ADVERB_REFLEXIVE_ACTIVE = 2
Global Const VERBAL_ADVERB_REFLEXIVE_PASSIVE = 4
Global Const VERBAL_ADVERB_IMPERFECTIVE = 8
Global Const VERBAL_ADVERB_PERFECTIVE = 16

Global Const VERBAL_ADJECTIVE_NON_REFLEXIVE = 1         'verbal adjectives
Global Const VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE = 2
Global Const VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE = 4
Global Const VERBAL_ADJECTIVE_PRESENT_ACTIVE = 8
Global Const VERBAL_ADJECTIVE_PRESENT_PASSIVE = 16
Global Const VERBAL_ADJECTIVE_PAST_ACTIVE = 32
Global Const VERBAL_ADJECTIVE_PAST_PASSIVE = 64
Global Const VERBAL_ADJECTIVE_IMPERFECTIVE = 128
Global Const VERBAL_ADJECTIVE_PERFECTIVE = 256

Global Const NOT_APPLICABLE = 0                         'ignore (applies to any attribute)
```

```
'———— part of speech secondary attributes before conversion to symbolic representation ——————————————————
Global Const DEMAND1_NO_DEMANDS = 0                          'type 1 case demands (nouns, adjectives, adverbs)
Global Const DEMAND1_GENITIVE = 1
Global Const DEMAND1_DATIVE = 2
Global Const DEMAND1_INSTRUMENTAL = 4

Global Const DEMAND2_NO_DEMANDS = 0                             'type 2 case demands (verbs, verbal adjectives and
adverbs)
Global Const DEMAND2_NOMINATIVE = 1
Global Const DEMAND2_GENITIVE = 2
Global Const DEMAND2_DATIVE = 4
Global Const DEMAND2_ACCUSATIVE = 8
Global Const DEMAND2_INSTRUMENTAL = 16

Global Const DEMAND3_NO_DEMANDS = 0                          'type 3 case demands (prepositions)
Global Const DEMAND3_GENITIVE = 1
Global Const DEMAND3_DATIVE = 2
Global Const DEMAND3_ACCUSATIVE = 4
Global Const DEMAND3_INSTRUMENTAL = 8
Global Const DEMAND3_PREPOSITIONAL = 16

Global Const ADJECTIVE_SHORT_FORM_MASCULINE = 1              'adjectives
Global Const ADJECTIVE_SHORT_FORM_FEMININE = 2
Global Const ADJECTIVE_SHORT_FORM_NEUTER = 3
Global Const ADJECTIVE_SHORT_FORM_PLURAL = 4

'———— article codes ————————————————————————————————————————————————————————————————————————————————
Global Const ARTICLE_NONE = 32                               'no article
Global Const ARTICLE_A = 64                                  'a
Global Const ARTICLE_AN = 128                                'an
Global Const ARTICLE_THE = 4096                              'the

'———— noun association codes ———————————————————————————————————————————————————————————————————————
Global Const ASSOCIATION_PERSON = 1
Global Const ASSOCIATION_PLACE = 2
Global Const ASSOCIATION_THING = 4
Global Const ASSOCIATION_ANIMAL = 8
Global Const ASSOCIATION_FIGURATIVE = 16

'———— plural nouns codes ———————————————————————————————————————————————————————————————————————————
Global Const PLURAL1 = 512                                   '$200, plural same or not used
Global Const PLURAL2 = 768                                   '$300, add 's'
Global Const PLURAL3 = 1024                                  '$400, add 'es'
Global Const PLURAL4 = 1280                                  '$500, add 'ies'
```

```
Global Const PLURAL5 = 1536                                 '$600, drop last, add 'ves'
Global Const PLURAL6 = 1792                                 '$700, drop last 2, add 'ves'
Global Const PLURAL7 = 2048                                 '$800, drop last 2, add 'a'
Global Const PLURAL8 = 2304                                 '$900, drop last 2, add 'i'
Global Const PLURAL9 = 2560                                 '$A00, drop last 2, add 'en'
Global Const PLURAL10 = 2816                                '$B00, drop last 4, add 'ice'
Global Const PLURAL11 = 3072                                '$C00, drop last 4, add 'eese'

'----- symbol table ranges ------------------------------------------------------------------------------------------
Global Const MASCULINE_NOUN_START = 1                       'nouns
Global Const MASCULINE_NOUN_END = 40

Global Const FEMININE_NOUN_START = 50
Global Const FEMININE_NOUN_END = 79

Global Const NEUTER_NOUN_START = 90
Global Const NEUTER_NOUN_END = 103

Global Const PLURAL_NOUN_START = 110
Global Const PLURAL_NOUN_END = 114

Global Const ADJECTIVE_START = 120                          'adjectives
Global Const ADJECTIVE_END = 143

Global Const POSSESSIVE_PRONOUN_START = 150                 'possessive pronouns
Global Const POSSESSIVE_PRONOUN_END = 151

Global Const DEMONSTRATIVE_PRONOUN_START = 155              'demonstrative pronouns
Global Const DEMONSTRATIVE_PRONOUN_END = 156

Global Const PERSONAL_PRONOUN_START = 390                   'personal pronouns
Global Const PERSONAL_PRONOUN_END = 404

Global Const INDEFINITE_PRONOUN_START = 380                 'indefinite pronouns
Global Const INDEFINITE_PRONOUN_END = 381

Global Const VERB_START = 160                               'verbs
Global Const VERB_END = 236

Global Const VERBAL_ADVERB_NON_REFLEXIVE_START = 250        'verbal adverbs
Global Const VERBAL_ADVERB_NON_REFLEXIVE_END = 257

Global Const VERBAL_ADVERB_REFLEXIVE_START = 270
Global Const VERBAL_ADVERB_REFLEXIVE_END = 276

Global Const VERBAL_ADJECTIVE_PA1_START = 290               'verbal adjectives
```

```
Global Const VERBAL_ADJECTIVE_PA1_END = 294
Global Const VERBAL_ADJECTIVE_PA2_START = 360
Global Const VERBAL_ADJECTIVE_PA2_END = 364

Global Const VERBAL_ADJECTIVE_PP_START = 310
Global Const VERBAL_ADJECTIVE_PP_END = 313

Global Const VERBAL_ADJECTIVE_AA1_START = 320
Global Const VERBAL_ADJECTIVE_AA1_END = 325
Global Const VERBAL_ADJECTIVE_AA2_START = 370
Global Const VERBAL_ADJECTIVE_AA2_END = 375

Global Const VERBAL_ADJECTIVE_AP_START = 340
Global Const VERBAL_ADJECTIVE_AP_END = 347

Global Const INTERROGATIVE_START = 380                          'interrogatives
Global Const INTERROGATIVE_END = 381

Global Const RELATIVE_PRONOUN_AT = 410                          'relative pronoun

'_____ program files ____________________________________________________________________________________________
Global Const LEXICON_FILENAME = "C:\Transl~1\lexicon.LEX"           'bilingual lexicon

Global Const HELP1_FILENAME = "C:\Transl~1\Help1.XLP"               'topical help files
Global Const HELP3_FILENAME = "C:\Transl~1\Help2.XLP"
Global Const HELP4_FILENAME = "C:\Transl~1\Help3.XLP"
Global Const HELP5_FILENAME = "C:\Transl~1\Help4.XLP"
Global Const HELP7_FILENAME = "C:\Transl~1\Help5.XLP"

Global Const FILTER_FILENAME = "C:\Transl~1\import~1.FTR"            'import filter file

Global Const CLEANUP_FILTER_FILENAME = "C:\Transl~1\cleanup.CLN"     'cleanup filter file

Global Const CONFIG_FILENAME = "C:\Transl~1\config.CNF"              'configuration settings

Global Const TRANSFER_RULE_FILENAME = "C:\Transl~1\transf~1.TRR"     'transfer rules file

Global Const TRANSFER_RULE_LOG_FILENAME = "C:\Transl~1\t-rules.LOG" 'expanded transfer rules log

Global Const DETAILS_LOG_FILENAME = "c:\Transl~1\xlate.XLG"          'translation details log file

'_______________________________________________________________________________________________________________
'                                          DATA STRUCTURES
'_______________________________________________________________________________________________________________

Type FilterType                                                 '-- import filter structure
```

```
     InCode                                  As String           'non-native character(s)
     OutCode                                 As String           'native cyrillic character
End Type

Type ConstantType                                                '-- results of analysis on single word structure
     SymbolicConstant(1 To MAX_SENTENCE_COMPONENTS) As String 'list of all symbols extracted from word
     NumSymbolicConstants                    As Integer          'pointer to end of list
End Type

Type LexiconType                                                 '-- lexicon structure
     Stem                                    As String           'word stem
     PartOfSpeech                            As Integer          'part of speech code
     Target                                  As String           'english stem
     InflectionPattern                       As Integer          'inflection pattern code
     PrimaryAttributes                       As Integer          'primary attributes
     SecondaryAttributes                     As Integer          'secondary attributes
     TransferAttributes                      As Long             'english syntax info (article,plurals,etc.)
     Disambiguation                          As String           'ambiguity disambiguation expression
End Type

Type TransferAttributeType                                       '-- structure to accomodate multiple homonyms
     LexiconEntry                            As LexiconType      'full lexicon entry for each homonym
     LexiconEntryPtr                         As Integer          'pointer to record in lexicon
     SymbolicConstants                       As ConstantType     'symbolic constants derived from each homonym
End Type

Type RawRuleType                                                    '-- internal temp structure for expanding transfer
rules
     Parameter(MAX_SENTENCE_COMPONENTS)      As String           'single side of transfer expression
     NumParameters                           As Integer          'number of terms in this side of expression
End Type

Type TransferType                                                '-- symbolic constant structure for each word
     Match(1 To MAX_MATCHES)                   As TransferAttributeType 'list of all possible matches found and their
attributes
     MatchPtr                                As Integer          'pointer to actual element to use in match list
     NumMatches                              As Integer          'pointer to last element of match list
     LiteralText                             As String           'word as it appear in abstract (punctuation, etc)
End Type

Type TransferRuleType                                            '-- transfer component structure for each word
     NumRussianComponents                    As Integer          'num components of russian rule
     Russian(MAX_SENTENCE_COMPONENTS)        As String           'russian expression of transfer rule
     NumEnglishComponents                    As Integer          'num components of english rule
     English(MAX_SENTENCE_COMPONENTS)        As String           'english expression of transfer rule
End Type
```

```
'_____________________________________________________________________________________________________
'                                        GLOBAL VARIABLES
'_____________________________________________________________________________________________________

Global gRawAbstract                         As String           'unconverted abstract
Global gAbstract                            As String           'converted abstract

Global gPackDelimiter                             As String * 1         'delimiter constant for filter packing (see
frmMain_Load)

Global gAbstractFilename                    As String           'abstract full filename
Global gAbstractTitle                       As String           'abstract filename without path, etc.
Global gAbstractDate                        As String           'date of creation or last modification
Global gAbstractWords                       As Integer          'num words in abstract
Global gLookups                             As Integer          'num words looked up in document
Global gLookupHits                          As Integer          'num words found in document

Global gTransferAttempts                    As Integer          'num sentences in document
Global gTransferHits                        As Integer          'num sentences in document translated

Global gTranslation                         As String           'raw translation text

Global gAbort                               As Integer          'flag indicating escape key hit

Global gShowHelp                            As Integer          'config setting to display passive help
Global gTransferDetails                     As Integer          'config setting to generate transfer log

Global gAbstractModified                    As Integer          'flags indicating changes made
Global gTranslationModified                 As Integer
Global gTransliterationModified             As Integer

Global gTransliterationFilter               As String           'default transliteration filter

Global gOperationMode                       As Integer          'mode of operation {TRANSLATION_MODE,EMAIL_MODE}

Global gPostEditorAPIName                   As String           'application name of post-translation editor
Global gPostEditorFilename                  As String           'filename of post-translation editor executable
Global gPostEditorPasteSequence             As String           'escape sequence used to paste into post-edit document

Global gPattern(NUM_INFLECTION_PATTERNS)    As String           'inflection patterns

Global gInputFilter(NUM_MAPPINGS)           As FilterType       'active character set filter

Global gLexicon(MAX_LEXICON_ENTRIES)        As LexiconType      'resident bilingual lexicon
```

```
Global gLexiconEndPtr                     As Integer          'pointer to end of lexicon structure

Global gLexiconIndex(MAX_ASCII)           As Integer          'index to lexicon entries

Global gSortOrder(MAX_ASCII)              As Integer          'lookup table for Cyrillic alphabet sort

Global gHeader(MAX_HEADER_LINES)          As String           'header lines (comments) in lexicon.  Stored
Global gHeaderPtr                         As Integer          'so they can be written back at lexicon update.

Global gTransferRule(MAX_TRANSFER_RULES)  As TransferRuleType 'transfer rule base
Global gTransferRulePtr                   As Integer          'pointer to last transfer rule

Global gSentence(MAX_SENTENCE_COMPONENTS) As TransferType     'single sentence structure
Global gWordPtr                           As Integer          'num words in sentence

Global gEventTick                         As Integer          'num seconds elapsed during translation
Global gEventTime                         As String           'seconds time converted to minutes:seconds

Global MsgBoxAnswer                            As Integer            'reply to all message boxes (value not shared
globally)

Global gNounConstant(1 To 13)             As String           'symbolic constant structures
Global gAdjectiveConstant(1 To 33)        As String
Global gAdverbConstant(1 To 3)            As String
Global gActiveVerbConstant(1 To 13)       As String
Global gAuxiliaryVerbConstant(1 To 13)    As String
Global gVerbalAdverbConstant(1 To 3)      As String
Global gRelPronounConstant(1 To 28)       As String
Global gPersPronounConstant(1 To 6)       As String
Global gVAdjPresActConstant(1 To 28)      As String
Global gVAdjPresPasConstant(1 To 28)      As String
Global gVAdjPastActConstant(1 To 28)      As String
Global gVAdjPastPasConstant(1 To 36)      As String
Global gInterrogativeConstant(1 To 6)     As String
Global gPrepositionCase(1 To 5)           As String
Global gConjunctionConstant(1 To 1)       As String


'_____________________________________________________________________________________________________
'update transfer details
'_____________________________________________________________________________________________________
Sub AddDetail (Detail As String, TimeStamp As Integer)

    'process only if transfer details enabled
    If gTransferDetails Then
       'add detail
```

```
       Print #5, Chr(34); Detail;

       'indicate time
       If TimeStamp Then
        Print #5, Tab(78); "["; gEventTime; "]"; Chr(34)
       Else
        Print #5, Chr(34)
       End If
    End If

End Sub

'_____________________________________________________________________________________________
'do automatic cleanup of translation based on cleanup filters
'_____________________________________________________________________________________________
Function ApplyCleanupFilters (ByVal DirtyText As String)

    Dim NumFilters   As Integer
    Dim TextPtr      As Integer
    Dim FilterPtr    As Integer
    Dim ComparePtr   As Integer
    Dim Replacements As Integer
    Dim CompareText  As String
    Dim ReplaceText  As String
    Dim CleanText    As String

    Const HARD_REPLACEMENTS = 5

    Static CText(HARD_REPLACEMENTS) As String
    Static RText(HARD_REPLACEMENTS) As String


    'display status info during load
    MousePointerWait
    NumFilters = frmCleanupFilterEditor.lstCleanupFilter(0).ListCount - 1
    DisplayWait -NumFilters, "Applying post-translation cleanup filters..."

    'log details
    AddDetail "", False
    AddDetail "_____________________________________________________________________________", False
    AddDetail "Applying post-translation cleanup filters:", False
    AddDetail "", False

    'compare each filter entry against dirty text
    For FilterPtr = 0 To NumFilters
       'get filter entry
```

```
      CompareText = frmCleanupFilterEditor.lstCleanupFilter(0).List(FilterPtr)
      ReplaceText = frmCleanupFilterEditor.lstCleanupFilter(1).List(FilterPtr)
      'until no more, replace
      ComparePtr = 1
      Do
         ComparePtr = InStr(ComparePtr, DirtyText, CompareText, 1)
         'replace
         If ComparePtr Then
            DirtyText = Left(DirtyText, ComparePtr - 1) & ReplaceText & Mid(DirtyText, ComparePtr +
Len(CompareText))
            AddDetail "  '" & CompareText & "' replaced with '" & ReplaceText & "'", False
            Replacements = Replacements + 1
            ComparePtr = ComparePtr + 1
         End If
      Loop Until ComparePtr = 0
      DisplayWait 1, ""
   Next FilterPtr

   'hard-coded replacements:
   'articles
   CText(0) = " a a": RText(0) = " an a"
   CText(1) = " a e": RText(1) = " an e"
   CText(2) = " a i": RText(2) = " an i"
   CText(3) = " a o": RText(3) = " an o"
   CText(4) = " a u": RText(4) = " an u"
   'punctuation
   CText(5) = ", that ": RText(5) = " that "

   'compare each hard-coded filter entry against dirty text
   For FilterPtr = 0 To HARD_REPLACEMENTS
      'get filter entry
      CompareText = CText(FilterPtr)
      ReplaceText = RText(FilterPtr)
      'until no more, replace
      ComparePtr = 1
      Do
         ComparePtr = InStr(ComparePtr, DirtyText, CompareText, 1)
         'replace
         If ComparePtr Then
            DirtyText = Left(DirtyText, ComparePtr - 1) & ReplaceText & Mid(DirtyText, ComparePtr +
Len(CompareText))
            ComparePtr = ComparePtr + 1
         End If
      Loop Until ComparePtr = 0
   Next FilterPtr
```

```
    'hide status
    DisplayWait 0, ""
    MousePointerContinue

    'log details
    If Replacements = 0 Then
       AddDetail "  No replacements made.", False
    End If
    AddDetail "", False

    'return cleaned up text
    ApplyCleanupFilters = DirtyText

End Function

'_____________________________________________________________________________________________________
'write final details to transfer log and close it
'_____________________________________________________________________________________________________
Sub CloseTransferLog ()

    Dim LookupRate As Integer

    On Error Resume Next


    'write final log details
    AddDetail "_____________________________________________________________________________", False
    AddDetail "Translation complete:", True
    AddDetail "", False

    'write translation
    DumpFinalTranslation

    AddDetail "", False

    'lookup rate on this abstract
    If gLookups Then
       LookupRate = (gLookupHits / gLookups) * 100
    Else
       LookupRate = 0
    End If
    AddDetail "  Words Translated:     " & gLookupHits & " of " & gLookups & " (" & LookupRate & "%)", False

    'transfer rate on this abstract
    If gTransferAttempts Then
       LookupRate = (gTransferHits / gTransferAttempts) * 100
```

```
    Else
       LookupRate = 0
    End If
    AddDetail "  Sentences Translated: " & gTransferHits & " of " & gTransferAttempts & " (" & LookupRate & "%)",
False

    AddDetail "  Avg Transfer Time:    " & Mid(TimeSerial(0, 0, gEventTick / gTransferHits), 5), False
    AddDetail "  Total Transfer Time:  " & gEventTime, False
    AddDetail "", False

    'terminate log session
    frmMain.timerTranslation.Enabled = False

    Close 5

End Sub

'_____________________________________________________________________________________________________
'proceed according to success of abstract conversion
'_____________________________________________________________________________________________________
Function ConversionOK (Abstract As String)

    'if conversion routine returned failure code, fail conversion process
    If Abstract = "[FAILED]" Then
       Abstract = ""
       ConversionOK = False
       Exit Function
    End If

    'conversion successful
    ConversionOK = True

End Function

'_____________________________________________________________________________________________________
'run abstract through selected available filters with intention of converting to native format
'_____________________________________________________________________________________________________
Function ConvertAbstract (InText As String, ShowStatus As Integer, InfoLine As String) As String

    Dim AbstractPtr     As Integer
    Dim FilterPtr       As Integer
    Dim InCode          As String
    Dim Found           As Integer
    Dim OutText         As String
    Dim UpdateTally     As Integer
```

```
Dim AbstractLen     As Integer
Dim NumChars        As Integer
Dim NumUnknownChars As Integer
Dim plural          As String
Dim Suggestion      As String
Dim PercentBad      As Single
Dim SourceChar      As String
Dim i               As Integer

Const THRESHOLD = 3  'acceptable threshold of unknown characters
                '(translation mode; email mode has no limit)


'set starting and ending points of abstract
AbstractPtr = 1
AbstractLen = Len(InText)

'reset word count
gAbstractWords = 1

'hide/display status info as conversion takes place
If ShowStatus Then
   DisplayWait -AbstractLen, InfoLine
End If

'for each character or character group in abstract...
Do
   Found = False

   '...try matching filter mappings
   For FilterPtr = 0 To NUM_MAPPINGS
       InCode = gInputFilter(FilterPtr).InCode
       'if found, stop looking
       SourceChar = Mid(InText, AbstractPtr, Len(InCode))

       If SourceChar = InCode And InCode <> "" Then
          Found = True
          'if space, increment word count (multiple spaces were already removed, so this is a valid count)
          If SourceChar = " " Then
            gAbstractWords = gAbstractWords + 1
          End If
          Exit For
       End If
   Next FilterPtr

   'if found, add converted character(s) to converted abstract and move to next character(s) in original...
```

```
   If Found Then
      OutText = OutText & gInputFilter(FilterPtr).OutCode
      AbstractPtr = AbstractPtr + Len(InCode)
   '...otherwise mark this character unknown and move on (translation cannot be done, though)
   Else
      'deal with unknown characters:
      If gOperationMode = ABSTRACT_MODE Then
         'in translation mode, mark them as unknown
         OutText = OutText & SUBSTITUTION_CHARACTER
      Else
         'deal with carriage return (for text box, must also have linefeed)
         If SourceChar = Chr(13) Then
           SourceChar = Chr(13) & Chr(10)
         End If
         'in email mode, pass them right through untouched (ignores embedded English)
         OutText = OutText & SourceChar
      End If

      NumUnknownChars = NumUnknownChars + 1
      AbstractPtr = AbstractPtr + 1
   End If
   UpdateTally = UpdateTally + Len(InCode)
   NumChars = NumChars + 1

   'if status bar displayed, update at every 5% change in work done
   If UpdateTally > (AbstractLen - 1) / 20 And ShowStatus Then
      DisplayWait UpdateTally, ""
      UpdateTally = 0
   End If
Loop Until AbstractPtr > AbstractLen

'hide status, if shown
If ShowStatus Then
   DisplayWait 0, ""
End If

'determine if singular/plural unknown characters
If NumUnknownChars = 1 Then
   plural = ""
Else
   plural = "s"
End If

'if conversion successful or email mode engaged, return abstract;
If NumUnknownChars = 0 Or gOperationMode = EMAIL_MODE Then
   ConvertAbstract = OutText
```

```
    'otherwise determine degree of conversion failure
    Else
       'too many mapping errors
       If NumUnknownChars > THRESHOLD Then
        ConvertAbstract = "[FAILED]"
       'enough to continue, but inform user to clean them up
       Else
        Beep
        MsgBox ("This abstract contains " & NumUnknownChars & " character" & plural & " which could not be
converted.  Unconverted characters are represented by the '" & SUBSTITUTION_CHARACTER & "' character and can be
fixed in the Source Abstract Editor.  Translation is possible without fixing these characters, but it may not be
complete or correct."), MB_ICONINFORMATION, ("Incomplete Mapping")
        ConvertAbstract = OutText
       End If
    End If

End Function

'_____________________________________________________________________________________________________________
'map active filter to standardized internal program representation
'_____________________________________________________________________________________________________________
Sub ConvertFilterMap ()

    Dim ColonAt As Integer
    Dim Temp    As FilterType
    Dim IValue  As String
    Dim IValue1 As String
    Dim IValue2 As String
    Dim i, j    As Integer


    'map or convert three possible filter entries (letters, 1-byte codes, and 2-byte codes) to
    'single-byte internal program representation.  1-byte and 2-byte codes can be any combination
    'of hex and decimal formats
    For i = 0 To NUM_MAPPINGS
      ColonAt = InStr(frmFilter.txtLatin(i), ":")

      'map compound numeric decimal or hex 2-byte codes to single character
      If ColonAt > 0 Then
         IValue1 = Left(frmFilter.txtLatin(i), ColonAt - 1)
         IValue1 = Chr(HexToDecimal(IValue1))
         IValue2 = Mid(frmFilter.txtLatin(i), ColonAt + 1)
         IValue2 = Chr(HexToDecimal(IValue2))
         gInputFilter(i).InCode = IValue1 & IValue2

      'convert simple numeric decimal 1-byte code to single character
```

```
      ElseIf IsNumeric(frmFilter.txtLatin(i)) Then
         gInputFilter(i).InCode = Chr(Val(frmFilter.txtLatin(i)))

      'convert simple numeric hex 1-byte (lower nibble) code to single character
      ElseIf Left(frmFilter.txtLatin(i), 1) = "$" And Len(frmFilter.txtLatin(i)) > 1 Then
         IValue = Chr(HexToDecimal("" & frmFilter.txtLatin(i)))
         gInputFilter(i).InCode = IValue

      'map letters(s) to single character
      Else
         IValue = frmFilter.txtLatin(i)
         gInputFilter(i).InCode = IValue
      End If

      'assign mapping
      gInputFilter(i).OutCode = frmFilter.lblCyrillic(i)
   Next i

   'sort mapping by size then alphabetized within groups of same size.  This is done so that larger
   'groupings are checked first in decoding.  For example, the mapping 'shch > q' must be found
   'before 's > c'; otherwise 'shch' would be interpreted as several separate characters, and not
   'as a single group.  This shell sort is not the most efficient algorithm, but only 66 entries
   'need to be arranged
   For i = 0 To NUM_MAPPINGS
      For j = i To NUM_MAPPINGS
         If  (Len(gInputFilter(j).InCode)  >  Len(gInputFilter(i).InCode))  Or  ((Len(gInputFilter(j).InCode)  =
Len(gInputFilter(i).InCode)) And (gInputFilter(j).InCode < gInputFilter(i).InCode)) Then
            Temp = gInputFilter(i)
            gInputFilter(i) = gInputFilter(j)
            gInputFilter(j) = Temp
         End If
      Next j
   Next i

End Sub

'_____________________________________________________________________________________________
'convert Windows 95 filenames to Windows 3.1 (ie. DOS) filenames.  The DOS representation is
'referred to as augmented DOS 6+2 format here because extended Win95 filenames are reduced to
'first 6 characters plus the following: ~1.  Some filenames may not be converted properly
'because of extraneous characters not supported by Visual Basic I/O.  In this case, file access
'needs to be done using standard method of truncated entries in file list boxes.
'_____________________________________________________________________________________________
Function ConvertWin95Filename (Win31Filename As String) As String
```

```
Dim FilenameChar   As String * 1
Dim TempFileName   As String
Dim TempFilePart   As String
Dim Win95Filename  As String
Dim FilenameStart  As Integer
Dim ExtensionStart As Integer
Dim EntryLength    As Integer
Dim CountOn        As Integer
Dim i              As Integer


'trim leading/trailing spaces from path and filename (lowercase conversion not nesc. for Win95)
Win31Filename = LCase(Trim(Win31Filename))

'remove spaces from path and filename
For i = 1 To Len(Win31Filename)
   FilenameChar = Mid(Win31Filename, i, 1)
   If FilenameChar <> " " Then
      TempFileName = TempFileName & FilenameChar
   End If
Next i

'check each path entry and condense extended Win95 entries to DOS 6+2 augmented entries
CountOn = False
For i = 1 To Len(TempFileName)
   FilenameChar = Mid(TempFileName, i, 1)
   'check for start of entry
   If FilenameChar = "\" Then
      If CountOn Then
         'reduce to augmented DOS name format if extended...
         If EntryLength > 8 Then
           Win95Filename = Win95Filename & "\" & Mid(TempFileName, i - EntryLength, 6) + "~1"
         '...otherwise take filename as is
         Else
           Win95Filename = Win95Filename & "\" & Mid(TempFileName, i - EntryLength, EntryLength)
         End If
         EntryLength = 0
      End If
      CountOn = True
   'count characters in entry
   ElseIf CountOn Then
      EntryLength = EntryLength + 1
   End If
Next i
'append path divider
Win95Filename = Win95Filename & "\"
```

```
    'extract filename extension
    ExtensionStart = InStr(TempFileName, ".")

    'search for start of filename working from end of full path
    For i = Len(TempFileName) To 1 Step -1
      FilenameChar = Mid(TempFileName, i, 1)
      'if found, record position
      If FilenameChar = "\" Then
         FilenameStart = i + 1
         Exit For
      End If
    Next i

    'if no paths given, whole entry is filename in current directory
    If FilenameStart < 1 Then
       FilenameStart = 1
       Win95Filename = ""
    End If

    'extract filename
    TempFilePart = Mid(TempFileName, FilenameStart, ExtensionStart - FilenameStart)

    'condense filename if necessary
    If Len(TempFilePart) > 8 Then
       TempFilePart = Left(TempFilePart, 6) + "~1"
    End If

    'combine condensed pathnames, filename, and extension
    TempFilePart = TempFilePart & Mid(TempFileName, ExtensionStart)
    Win95Filename = Win95Filename & TempFilePart

    'return DOS filename from Windows 95 filename
    ConvertWin95Filename = Win95Filename

End Function

'_____________________________________________________________________________________________
'take single sentence and extract symbolic constant(s) for each word
'_____________________________________________________________________________________________
Sub DecomposeSentence (ByVal Sentence As String)

    Dim WordPtr         As Integer
    Dim BreakAt         As Integer
    Dim MatchPtr        As Integer
    Dim SymbolPtr       As Integer
```

```
Dim LogPtr1         As Integer
Dim LogPtr2         As Integer
Dim Word            As String
Dim FullWord        As String
Dim SymbolicDetails As String

Dim TransferDetails As TransferType

MousePointerWait
'reinitialize structures
For WordPtr = 0 To MAX_SENTENCE_COMPONENTS
   gSentence(WordPtr).NumMatches = 0
   gSentence(WordPtr).MatchPtr = 0
   'blank words
   For MatchPtr = 1 To MAX_MATCHES
       gSentence(WordPtr).Match(MatchPtr).LexiconEntry.Stem = ""
       gSentence(WordPtr).Match(MatchPtr).LexiconEntry.Target = "<UNKNOWN WORD>"
       gSentence(WordPtr).Match(MatchPtr).SymbolicConstants.NumSymbolicConstants = 0
       'blank symbolic constants
       For SymbolPtr = 1 To MAX_SENTENCE_COMPONENTS
          gSentence(WordPtr).Match(MatchPtr).SymbolicConstants.SymbolicConstant(SymbolPtr) = ""
       Next SymbolPtr
   Next MatchPtr
Next WordPtr

'strip leading/trailing spaces.  Append one to terminate do/loop
Sentence = Trim(Sentence) & " "

'update translation details
AddDetail "_____________________________________________________________________________", False
AddDetail ("Sentence: " & Sentence), True
AddDetail "", False

'loop through sentence word by word
WordPtr = 1
gWordPtr = -1
Do
   'get end of word, skipping multiple spaces
   Do
     BreakAt = InStr(WordPtr, Sentence, " ")
     'multiple space, so skip this character
     If BreakAt = WordPtr Then
        WordPtr = WordPtr + 1
     'found actually end, so exit
     Else
        Exit Do
```

```
        End If
      Loop
      'if there's an end, extract word; otherwise...
      If BreakAt > 0 Then
        'strip whitespace, but leave punctuation
        Word = Trim(Mid(Sentence, WordPtr, BreakAt - WordPtr))

        'strip prepended punctuation
        If Left(Word, 1) < "ю" Then
           Word = Mid(Word, 2)
        End If

        'record punctuation
        FullWord = Word

        'strip appended punctuation
        If (Right(Word, 1) < "ю") And (Len(Word)) Then
           Word = Left(Word, Len(Word) - 1)
        End If

RetryLookup:
        'look up word
        LookupWord (Word), TransferDetails, True
        gLookups = gLookups + 1

        'AddDetail "  ______", False
        'if word not found, get user to deal with it
        If TransferDetails.NumMatches = 0 Then
           AddDetail ("  Lookup: " & Word & " (unknown)"), False
           AddDetail "", False
           Beep
           MousePointerContinue
           frmUnknownWord.lblUnknownWord = Word
           frmUnknownWord.Show 1
           'if just added to lexicon, look up again; otherwise...
           If frmUnknownWord.Tag = "[ADD]" Then
             AddDetail "          Word added to lexicon", False
             AddDetail "", False
             GoTo RetryLookup 'ick, a GoTo!
           '...user opted to ignore, so tag this entry as unknown
           Else
             AddDetail "          Word ignored", False
             AddDetail "", False
             gWordPtr = gWordPtr + 1
             gSentence(gWordPtr).NumMatches = IGNORE_WORD
           End If
```

```
            MousePointerWait
         Else
            AddDetail ("  Lookup: " & Word), False
            AddDetail "", False
            'record everything for this word
            gLookupHits = gLookupHits + 1
            gWordPtr = gWordPtr + 1
            gSentence(gWordPtr) = TransferDetails
            gSentence(gWordPtr).LiteralText = FullWord

            'write details to log, if active
            If gTransferDetails Then
              'write all matches
              For LogPtr1 = 1 To gSentence(gWordPtr).NumMatches
                  AddDetail  ("           Got=  "  &  TransferDetails.Match(LogPtr1).LexiconEntry.Stem  &  "  ("  &
TransferDetails.Match(LogPtr1).LexiconEntry.Target) & ")", False
                  SymbolicDetails = ""
                  'write details of each match
                  For LogPtr2 = 1 To gSentence(gWordPtr).Match(LogPtr1).SymbolicConstants.NumSymbolicConstants
                    SymbolicDetails                    =                    SymbolicDetails                    &
gSentence(gWordPtr).Match(LogPtr1).SymbolicConstants.SymbolicConstant(LogPtr2) & "  "
                  Next LogPtr2
                  AddDetail ("      SC= " & SymbolicDetails), False
                  AddDetail "", False
              Next LogPtr1
            End If
         End If

         'move to next word
         WordPtr = BreakAt + 1
       '...done processing
       Else
         Exit Do
       End If
    Loop

End Sub

'_____________________________________________________________________________________________________
'define inflections for all Russian words.  Gaps in array for additional rare inflections
'overlooked at initial design time
'_____________________________________________________________________________________________________
Sub DefineInflectionPatterns ()

    'nouns:
    '(format "nom_s.gen_s.dat_s.acc_s.ins_s.pre_s.loc_s.
```

```
'         nom_p.gen_p.dat_p.acc_p.ins_p.pre_p")

'hard masculine nouns
gPattern(1) = "#.а.у.#.ом.е.у.ы.ов.ам.ы.ами.ах"
gPattern(2) = "ол.ла.лу.ол.лом.ле.лу.лы.лов.лам.лы.лами.лах"
gPattern(3) = "ёд.ьда.ьду.ёд.ьдом.ьде.ьу.ьды.ьдов.ьдам.ьды.ьдами.ьдах"
gPattern(4) = "#.а.у.#.ом.е.у.а.#.ам.а.ами.ах"
gPattern(5) = "#.а.у.#.ом.е.у.а.ов.ам.а.ами.ах"
gPattern(6) = "#.а.у.#.ом.е./.ы.ов.ам.ы.ами.ах"
gPattern(7) = "ок.ка.ку.ок.ком.ке./.ки.ков.кам.ки.ками.ках"
gPattern(8) = "#.а.у.а.ом.е./.ья.ьев.ьям.ьев.ьями.ьях"
gPattern(9) = "#.а.у.а.ом.е./.и.ей.ам.ей.ами.ах"
gPattern(10) = "#.а.у.а.ом.е./.а.ов.ам.ов.ами.ах"
gPattern(11) = "г.га.гу.га.гом.ге./.зья.зей.зьям.зей.зьями.зьях"
gPattern(12) = "ец.ца.цу.ца.цом.це./.цы.цов.цам.цов.цами.цах"
gPattern(13) = "#.а.у.а.ом.е./.ы.ов.ам.ов.ами.ах"
gPattern(14) = "#.а.у.а.ом.е./.и.ов.ам.ов.ами.ах"
gPattern(15) = "ин.ина.ину.ина.ином.ине./.е.#.ам.#.ами.ах"
gPattern(16) = "#.а.у.а.ом.е./.овья.овей.овьям.овей.овьями.овьях"
gPattern(17) = "#.а.у.а.ем.е./.ья.ей.ьям.ей.ьями.ьях"
gPattern(18) = "ёр.ра.ру.ёр.ром.ре./.ры.ров.рам.ры.рами.рах"
gPattern(19) = "онок.онка.онку.онка.онком.онке./.ата.ат.атам.ат.атами.атах"
gPattern(20) = "ёнок.ёнка.ёнку.ёнка.ёнком.ёнке./.ята.ят.ятам.ят.ятами.ятах"
gPattern(21) = "#.а.у.а.ом.е././.#./././."
gPattern(22) = "#.а.у.#.ом.е./.ья.ьев.ьям.ья.ьями.ьях"
gPattern(23) = "#.а.у.#.ом.е./.и.ов.ам.и.ами.ах"
gPattern(24) = "#.а.у.#.ом.е./.а.ов.ам.а.ами.ах"
gPattern(25) = "а.а.у.а.ом.е./.и.ов.ам.ов.ами.ах"
gPattern(26) = "#.#.#.#.#.#./.#.#.#.#.#.#"

'soft masculine nouns
gPattern(30) = "й.я.ю.й.ем.е.ю.и.ёв.ям.и.ями.ях"
gPattern(31) = "ь.я.ю.ь.ём.е./.и.ей.ям.и.ями.ях"
gPattern(32) = "й.я.ю.й.ем.е./.и.ев.ям.и.ями.ях"
gPattern(33) = "ь.я.ю.ь.ем.е./.и.ей.ям.и.ями.ях"
gPattern(34) = "ень.ня.ню.ень.нём.не./.ни.ней.ням.ни.нями.нях"
gPattern(35) = "ь.и.и.ь.ём.и./.и.ей.ям.и.ями.ях"
gPattern(36) = "ель.ля.лю.ель.лем.ли./.ли.лей.лям.ли.лями.лях"
gPattern(37) = "ь.я.ю.я.ем.е./.и.ей.ям.ей.ями.ях"
gPattern(38) = "й.я.ю.я.ем.е./.и.ев.ям.ев.ями.ях"
gPattern(39) = "оль.ля.лю.оль.лем.ли./.ли.лей.лям.ли.лями.лях"
gPattern(40) = "я.и.е.ю.ёй.е./.и.ей.ям.ей.ями.ях"

'hard feminine nouns
gPattern(50) = "а.ы.е.у.ой.е./.ы.#.ам.ы.ами.ах"
gPattern(51) = "а.и.е.у.ой.е./.и.#.ам.и.ами.ах"
```

```
gPattern(52) = "а.ы.е.у.ой.е./.ы.#.ам.#.ами.ах"
gPattern(53) = "а.и.е.у.ой.е./.и.#.ам.#.ами.ах"
gPattern(54) = "ка.ки.ке.ку.кой.ке./.ки.ек.кам.ек.ками.ках"
gPattern(55) = "естра.естры.естре.естру.естрой.естре./.ёстры.естёр.ёстрам.естёр.ёстрами.ёстрах"
gPattern(56) = "ена.ены.ене.ену.еной.ене./.ёны.ён.ёнам.ён.ёнами.ёнах"
gPattern(57) = "ка.ки.ке.ку.кой.ке./.ки.ек.кам.ки.ками.ках"
gPattern(58) = "ка.ки.ке.ку.кой.ке./.ки.ок.кам.ки.ками.ках"
gPattern(59) = "ека.еки.еке.ёку.екой.еке./.ёки.ёк.екам.ёки.еками.еках"
gPattern(60) = "езда.езды.езде.езду.ездой.езде./.ёзды.ёзд.ёздам.ёзды.ёздами.ёздах"
gPattern(61) = "а.ы.е.у.ей.е./.ы.#.ам.#.ами.ах"

'soft feminine nouns
gPattern(70) = "я.и.е.ю.ей.е./.и.ь.ям.и.ями.ях"
gPattern(71) = "я.и.и.ю.ей.и./.и.й.ям.и.ями.ях"
gPattern(72) = "ь.и.и.ь.ью.и./.и.ей.ям.и.ями.ях"
gPattern(73) = "ня.ни.не.ню.ней.не./.ни.ен.ням.ни.нями.нях"
gPattern(74) = "я.и.е.ю.ей.е./.и.ей.ям.ей.ями.ях"
gPattern(75) = "я.и.и.ю.ёй.и./.и.ей.ям.и.ями.ях"
gPattern(76) = "ь.ери.ери.ь.ерью.ери./.ери.ерей.ерям.ерей.ерьми.ерях"
gPattern(77) = "ь.ери.ери.ь.ерью.ери./.ери.ерей.ерям.ерей.ерями.ерях"
gPattern(78) = "овь.ви.ви.овь.овью.ви./.ви.вей.вам.ви.вами.вах"
gPattern(79) = "ь.и.и.ь.ью.и./.и.ей.ам.ей.ами.ах"

'neuter nouns
gPattern(90) = "о.а.у.о.ом.е./.а.#.ам.а.ами.ах"
gPattern(91) = "е.я.ю.е.ем.и./.я.й.ям.я.ями.ях"
gPattern(92) = "е.я.ю.е.ем.е./.я.ей.ям.я.ями.ях"
gPattern(93) = "ьцо.ьца.ьцу.ьцо.ьцом.ьце./.ьца.ец.ьцам.ьца.ьцами.ьцах"
gPattern(94) = "ло.ла.лу.ло.лом.ле./.ла.ел.лам.ла.лами.лах"
gPattern(95) = "я.ени.ени.я.енем.ени./.ена.ён.енам.ена.енами.енах"
gPattern(96) = "о.а.у.о.ом.е./.еса.ес.есам.еса.есами.есах"
gPattern(97) = "ело.ела.елу.ело.елом.еле./.ёла.ёл.ёлам.ёла.ёлами.ёлах"
gPattern(98) = "о.а.у.о.ом.е./.ья.ьев.ьям.ья.ьями.ьях"
gPattern(99) = "о.а.у.о.ом.е./.и.#.ам.и.ами.ах"
gPattern(100) = "#.#.#.#.#.#./.#.#.#.#.#.#"
gPattern(101) = "о.а.у.о.ом.е./.а.ов.ам.а.ами.ах"
gPattern(102) = "ё.я.ю.ё.ём.е./.я.ей.ям.я.ями.ях"
gPattern(103) = "ье.ья.ью.ье.ьем.ье./.ья.ий.ьям.ья.ьями.ьях"

'plural only nouns
gPattern(110) = "/././././././.и.ов.ам.и.ами.ах"
gPattern(111) = "/././././././.ьги.ег.ьгам.ьги.ьгами.ьгах"
gPattern(112) = "/././././././.и.ей.ям.ей.ьми.ях"
gPattern(113) = "/././././././.а.#.ам.#.ами.ах"
gPattern(114) = "/././././././.ы.ов.ам.ы.ами.ах"
```

```
    'adjectives:
    '(format "nom_m.gen_m.dat_m.aca_m.aci_m.ins_m.pre_m.
    '         nom_f.gen_f.dat_f.aca_f.aci_f.ins_f.pre_f.
    '         nom_n.gen_n.dat_n.aca_n.aci_n.ins_n.pre_n.
    '         nom_p.gen_p.dat_p.aca_p.aci_p.ins_p.pre_p")

    'hard adjectives
    gPattern(120) =
"ый.ого.ому.ого.ый.ым.ом.#.ая.ой.ой.ую.ую.ой.ой.а.ое.ого.ому.ое.ое.ым.ом.о.ые.ых.ым.ых.ые.ыми.ых.ы"
    gPattern(121) =
"ий.его.ему.его.ий.им.ем.#.ая.ей.ей.ую.ую.ей.ей.а.ее.его.ему.ее.ее.им.ем.о.ие.их.им.их.ие.ими.их.и"
    gPattern(122) =
"ой.ого.ому.ого.ой.им.ом.#.ая.ой.ой.ую.ую.ой.ой.а.ое.ого.ому.ое.ое.им.ом.о.ие.их.им.их.ие.ими.их.и"
    gPattern(123) =
"кий.кого.кому.кого.кий.ким.ком.ок.кая.кой.кой.кую.кую.кой.кой.ка.кое.кого.кому.кое.кое.ким.ком.ко.кие.ких.ким.ких.к
ие.кими.ких.ки"
    gPattern(124) =
"ный.ного.ному.ного.ный.ным.ном.ен.ная.ной.ной.ную.ную.ной.ной.на.ное.ного.ному.ное.ное.ным.ном.но.ные.ных.ным.ных.н
ые.ными.ных.ны"
    gPattern(125) =
"ий.ого.ому.ого.ий.им.ом.#.ая.ой.ой.ую.ую.ой.ой.а.ое.ого.ому.ое.ое.им.ом.о.ие.их.им.их.ие.ими.их.и"
    gPattern(126) =
"ьный.ьного.ьному.ьного.ьный.ьным.ьном.ен.ьная.ьной.ьной.ьную.ьную.ьной.ьной.ьна.ьное.ьного.ьному.ьное.ьное.ьным.ьн
ом.ьно.ьные.ьных.ьным.ьных.ьные.ьными.ьных.ьны"

    'soft adjectives
    gPattern(130) =
"ий.его.ему.его.ий.им.ем./.яя.ей.ей.юю.юю.ей.ей./.ее.его.ему.ее.ее.им.ем./.ие.их.им.их.ие.ими.их./"
    gPattern(131) =
"ий.ьего.ьему.ьего.ий.ьим.ьем./.ья.ьей.ьей.ью.ью.ьей.ьей./.ье.ьего.ьему.ье.ье.ьим.ьем./.ьи.ьих.ьим.ьих.ьи.ьими.ьих"

    'miscellaneous adjectives
    gPattern(140) =
"есь.сего.сему.сего.есь.сем.сём./.ся.сей.сей.сю.сю.сей.сей./.сё.сего.сему.сё.сё.сем.сём./.се.сех.сем.сех.се.семи.сех
./"
    gPattern(141) =
"ей.ьего.ьему.ьего.ьей.ьим.ьём./.ья.ьей.ьей.ью.ью.ьей.ьей./.ьё.ьего.ьему.ьё.ьё.ьим.ьём./.ьи.ьих.ьим.ьих.ьи.ьими.ьих
./"
    gPattern(142) =
"ин.ного.ному.ного.ин.ним.ном./.на.ной.ной.ну.ну.ной.ной./.но.ного.ному.но.но.ним.ном./.ни.них.ним.них.ни.ними.них./
"
    gPattern(143) = "а.оих.оим.оих.а.оими.оих./.е.еих.еим.еих.е.еими.еих./.а.оих.оим.а.а.оими.оих./././././././././"

    'possesive pronouns
    gPattern(150) = "й.его.ему.его.й.им.ём./.я.ей.ей.ю.ю.ей.ей./.ё.его.ему.ё.ё.им.ём./.и.их.им.их.и.ими.их./"
    gPattern(151) = "#.его.ему.его.#.им.ем./.а.ей.ей.у.у.ей.ей./.е.его.ему.е.е.им.ем./.и.их.им.их.и.ими.их./"
```

```
'demonstrative pronouns
gPattern(155) = "от.ого.ому.ого.от.им.ом./.а.ой.ой.у.у.ой.ой./.о.ого.ому.ого.о.им.ом./.и.их.им.их.и.ими.их./"
gPattern(156) = "от.ого.ому.ого.от.ем.ом./.а.ой.ой.у.у.ой.ой./.о.ого.ому.ого.о.ем.ом./.е.ех.ем.ех.е.еми.ех./"

'verbs:
'(format "inf.1st_s.2nd_s.3rd_s.imp_s.
'         1st_p.2nd_p.3rd_p.imp_p.
'         past_m.past_f.past_n.past_p")

'non-reflexive
gPattern(160) = "ть.ю.ешь.ет.й.ем.ете.ют.йте.л.ла.ло.ли"
gPattern(161) = "ить.ю.ишь.ит.и.им.ите.ят.ите.ил.ила.ило.или"
gPattern(162) = "зать.жу.жешь.жет.жи.жем.жете.жут.жите.зал.зала.зало.зали"
gPattern(163) = "ть.ву.вёшь.вёт.ви.вём.вёте.вут.вите.л.ла.ло.ли"
gPattern(164) = "вать.ю.ёшь.ёт.й.ём.ёте.ют.йте.вал.вала.вало.вали"
gPattern(165) = "тить.чу.тишь.тит.ть.тим.тите.тят.тьте.тил.тила.тило.тили"
gPattern(166) = "сать.шу.шешь.шет.ши.шем.шете.шут.шите.сал.сала.сало.сали"
gPattern(167) = "ить.лю.ишь.ит.и.им.ите.ят.ите.ил.ила.ило.или"
gPattern(168) = "ыть.уду.удешь.удет.удь.удем.удете.удут.удьте.ыл.ыла.ыло.ыли"
gPattern(169) = "деть.жу.дишь.дит.ди.дим.дите.дят.дите.дел.дела.дело.дели"
gPattern(170) = "ыть.ою.оешь.оет.ой.оем.оете.оют.ойте.ыл.ыла.ыло.ыли"
gPattern(171) = "сть.ду.дёшь.дёт.ди.дём.дёте.дут.дите.л.ла.ло.ли"
gPattern(172) = "ить.у.ишь.ит.и.им.ите.ат.ите.ил.ила.ило.или"
gPattern(173) = "ть.м.ш.ст.й.дим.дите.дут.йте.л.ла.ло.ли"
gPattern(174) = "ть.ну.нешь.нет.нь.нем.нете.нут.ньте.л.ла.ло.ли"
gPattern(175) = "чь.гу.жешь.жет.ги.жем.жете.гут.гите.г.гла.гло.гли"
gPattern(176) = "нять.йму.ймёшь.ймёт.ймём.ймёте.ймут.ймите.нял.няла.няло.няли"
gPattern(177) = "ечь.ягу.яжешь.яжет.яг.яжем.яжете.ягут.ягте.ёг.егла.егло.егли"
gPattern(178) = "ать.ну.нёшь.нёт.ни.нём.нёте.нут.ните.ал.ала.ало.али"
gPattern(179) = "сть.м.шь.сть.шь.дим.дите.дят.шьте.л.ла.ло.ли"
gPattern(180) = "ить.ью.ьёшь.ьёт.ьей.ьём.ьёте.ьют.ьейте.ил.ила.ило.или"
gPattern(181) = "еть.ою.оёшь.оёт.ой.оём.оёте.оют.ойте.ел.ела.ело.ели"
gPattern(182) = "есть.яду.ядешь.ядет.ядь.ядем.ядете.ядут.ядьте.ел.ела.ело.ели"
gPattern(183) = "слать.шлю.шлёшь.шлёт.шли.шлём.шлёте.шлют.шлите.слал.слала.слало.слали"
gPattern(184) = "рать.еру.ерёшь.ерёт.ери.ерём.ерёте.ерут.ерите.рал.рала.рало.рали"
gPattern(185) = "зять.озьму.озьмёшь.озьмёт.озьми.озьмём.озьмёте.озьмут.озьмите.зял.зяла.зяло.зяли"
gPattern(186) = "овать.ую.уешь.ует.уй.уем.уете.уют.уйте.овал.овала.овало.овали"
gPattern(187) = "евать.ую.уешь.ует.уй.уем.уете.уют.уйте.евал.евала.евало.евали"
gPattern(188) = "скать.щу.щешь.щет.щи.щем.щете.щут.щите.скал.скала.скало.скали"
gPattern(189) = "ать.лю.ишь.ит.и.им.ите.ят.ите.ал.ала.ало.али"
gPattern(190) = "дить.жу.дишь.дит.ди.дим.дите.дят.дите.дил.дила.дило.дили"
gPattern(191) = "сить.шу.сишь.сит.си.сим.сите.сят.сите.сил.сила.сило.сили"
gPattern(192) = "ти.у.ёшь.ёт.и.ём.ёте.ут.ите./././."
gPattern(193) = "/././././././././.ёл.ла.ло.ли"
gPattern(194) = "хать.ду.дешь.дет./.дим.дите.дут./.хал.хала.хало.хали"
```

```
    gPattern(195) = "ести.еду.едёшь.едёт.еди.едём.едёте.едут.едите.ёл.ела.ело.ели"
    gPattern(196) = "теть.чу.тишь.тит.ти.тим.тите.тят.тите.тел.тела.тело.тели"
    gPattern(197) = "ять.иму.имешь.имет.ими.имем.имете.имут.имите.ял.яла.яло.яли"
    gPattern(198) = "ать.у.ёжь.ёт.и.ём.ёте.ут.ите.ал.ала.ало.али"
    gPattern(199) = "теть.чу.чешь.чет.ти.тим.тите.тят.тите.тел.тела.тело.тели"

    'reflexive
    gPattern(210) = "ться.юсь.ешься.ется.йся.емся.етесь.ются.йтесь.лся.лась.лось.лись"
    gPattern(211) = "иться.юсь.ишься.ится.ись.имся.итесь.ятся.итесь.ился.илась.илось.ились"
    gPattern(212) = "заться.жусь.жешься.жется.жись.жемся.жетесь.жутся.житесь.зался.залась.залось.зались"
    gPattern(213) = "ться.вусь.вёшься.вётся.вись.вёмся.вётесь.вутся.витесь.лся.лась.лось.лись"
    gPattern(214) = "ваться.юсь.ёшься.ётся.йся.ёмся.ётесь.ются.йтесь.лся.лась.лось.лись"
    gPattern(215) = "саться.шусь.шешься.шется.шись.шемся.шетесь.шутся.шитесь.сался.салась.салось.сались"
    gPattern(216) = "деться.жусь.дишься.дится.дись.димся.дитесь.дятся.дитесь.делся.делась.делось.делись"
    gPattern(217) = "ыться.оюсь.оешься.оется.ойся.оемся.оетесь.оются.ойтесь.ылся.ылась.ылось.ылись"
    gPattern(218) = "иться.ьюсь.ьёшься.ьётся.ьейся.ьёмся.ьётесь.ьются.ьейтесь.ился.илась.илось.ились"
    gPattern(219) = "иться.усь.ишься.ится.ись.имся.итесь.атся.итесь.ился.илась.илось.ились"
    gPattern(220) = "ться.мся.шся.стся.йся.димся.дитесь.дутся.йтесь.лся.лась.лось.лись"
    gPattern(221) = "ться.нусь.нешься.нется.нься.немся.нетесь.нутся.ньтесь.лся.лась.лось.лись"
    gPattern(222) = "теться.чусь.чешься.чется.тись.тимся.титесь.тятся.титесь.телся.телась.телось.телись"
    gPattern(223) = "аться.нусь.нёшься.нётся.нись.нёмся.нётесь.нутся.нитесь.ался.алась.алось.ались"
    gPattern(224) = "ситься.шусь.сишься.ится.сись.симся.ситесь.сятся.ситесь.сился.силась.силось.сились"
    gPattern(225) = "аться.люсь.ишься.ится.ись.имся.итесь.ятся.итесь.ался.алась.алось.ались"
    gPattern(226) = "раться.ерусь.ерёшься.ерётся.ерись.ерёмся.ерётесь.етутся.еритесь.рался.ралась.ралось.рались"
    gPattern(227) =
"зяться.озьмусь.озьмёшься.озьмётся.озьмись.озьмёмся.озьмётесь.озьмутся.озьмитесь.зялся.зялась.зялось.зялись"
    gPattern(228) = "естись.едусь.едёшься.едётся.едись.едёмся.едётесь.едутся.едитесь.ёлся.елась.елось.елись"
    gPattern(229) = "теться.чусь.тишься.тится.тись.тимся.титесь.тятся.титесь.телся.телась.телось.телись"
    gPattern(230) = "яться.имусь.имешься.имется.имись.имемся.иметесь.имутся.имитесь.ялся.ялась.ялось.ялись"
    gPattern(231) = "иться.люсь.ишься.ится.ись.имся.итесь.ятся.итесь.ился.илась.илось.ились"
    gPattern(232) = "оваться.уюсь.уешься.уется.уйся.уемся.уетесь.уются.уйтесь.овался.овалась.овалось.овались"
    gPattern(233) = "ебаться.уюсь.уешься.уется.уйся.уемся.уетесь.уются.уйтесь.ебался.ебалась.ебалось.ебались"
    gPattern(234) = "йтись.йдусь.йдёшься.йдётся.йдись.йдёмся.йдётесь.йдутся.йдитесь.шёлся.шлась.шлось.шлись"
    gPattern(235) = "хаться.дусь.дешься.дется.дись.димся.дитесь.дутся.дитесь.хался.халась.халось.хались"
    gPattern(236) = "диться.жусь.дишься.дится.дись.димся.дитесь.дятся.дитесь.дился.дилась.дилось.дились"

    'verbal adverbs:
    '(format "simultaneous.sequential1.sequential2")

    'non-reflexive
    gPattern(250) = "я.в.вши"
    gPattern(251) = "#.в.вши"
    gPattern(252) = "уя.овав.овавши"
    gPattern(253) = "еся./.ёсши"
    gPattern(254) = "я.ив.ивши"
    gPattern(255) = "я.ев.евши"
```

```
    gPattern(256) = "а.ив.ивши"
    gPattern(257) = "я.#.ши"

    'reflexive
    gPattern(270) = "ясь.вшись./"
    gPattern(271) = "ясь.евшись./"
    gPattern(272) = "уясь.овавшись./"
    gPattern(273) = "есясь.ёсшись./"
    gPattern(274) = "ясь.ившись./"
    gPattern(275) = "ась.ившись./"
    gPattern(276) = "ясь.шись./"

    'verbal adjectives:
    '(format "nom_m.gen_m.dat_m.aca_m.aci_m.ins_m.pre_m.
    '         nom_f.gen_f.dat_f.aca_f.aci_f.ins_f.pre_f.
    '         nom_n.gen_n.dat_n.aca_n.aci_n.ins_n.pre_n.")

    'non-reflexive present active
    gPattern(290)                                                                                  =
"ащий.ащего.ащему.ащего.ащий.ащим.ащем.ащая.ащей.ащей.ащую.ащую.ащей.ащей.ащее.ащего.ащему.ащее.ащее.ащим.ащем.ащие.ащих
.ащим.ащих.ащие.ащими.ащих"
    gPattern(291)                                                                                  =
"ящий.ящего.ящему.ящего.ящий.ящим.ящем.ящая.ящей.ящей.ящую.ящую.ящей.ящей.ящее.ящего.ящему.ящее.ящее.ящим.ящем.ящие.ящих
.ящим.ящих.ящие.ящими.ящих"
    gPattern(292)                                                                                  =
"ущий.ущего.ущему.ущего.ущий.ущим.ущем.ущая.ущей.ущей.ущую.ущую.ущей.ущей.ущее.ущего.ущему.ущее.ущее.ущим.ущем.ущие.ущи
х.ущим.ущих.ущие.ущими.ущих"
    gPattern(293)                                                                                  =
"ющий.ющего.ющему.ющего.ющий.ющим.ющем.ющая.ющей.ющей.ющую.ющую.ющей.ющей.ющее.ющего.ющему.ющее.ющее.ющим.ющем.ющие.ющих
.ющим.ющих.ющие.ющими.ющих"
    gPattern(294)                                                                                  =
"ующий.ующего.ующему.ующего.ующий.ующим.ующем.ующая.ующей.ующей.ующую.ующую.ующей.ующей.ующее.ующего.ующему.ующее.ующее
.ующим.ующем.ующие.ующих.ующим.ующих.ующие.ующими.ующих"

    'non-reflexive present passive
    gPattern(310)                                                                                  =
"емый.емого.емому.емого.емый.емым.емом.емая.емой.емой.емую.емую.емой.емой.емое.емого.емому.емое.емое.емым.емом.емые.
емых.емым.емых.емые.емыми.емых"
    gPattern(311)                                                                                  =
"имый.имого.имому.имого.имый.имым.имом.имая.имой.имой.имую.имую.имой.имой.имое.имого.имому.имое.имое.имым.имом.имые.
имых.имым.имых.имые.имыми.имых"
    gPattern(312)                                                                                  =
"ваемый.ваемого.ваемому.ваемого.ваемый.ваемым.ваемом.ваемая.ваемой.ваемой.ваемую.ваемую.ваемой.ваемой.ваемое.ваемог
о.ваемому.ваемое.ваемое.ваемым.ваемом.ваемые.ваемых.ваемым.ваемых.ваемые.ваемыми.ваемых"
```

```
    gPattern(313)                                                                                        =
"уемый.уемого.уемому.уемого.уемый.уемым.уемом.уемая.уемой.уемой.уемую.уемую.уемой.уемой.уемое.уемого.уемому.уемое.у
емое.уемым.уемом.уемые.уемых.уемым.уемых.уемые.уемыми.уемых"

    'non-reflexive past active
    gPattern(320)                                                                                        =
"вший.вшего.вшему.вшего.вший.вшим.вшем.вшая.вшей.вшей.вшую.вшую.вшей.вшей.вшее.вшего.вшему.вшее.вшее.вшим.вшем.вшие
.вших.вшим.вших.вшие.вшими.вших"
    gPattern(321)                                                                                        =
"явший.ившего.ившему.ившего.ивший.ившим.ившем.ившая.ившей.ившей.ившую.ившую.ившей.ившей.ившее.ившего.ившему.ившее.и
вшее.ившим.ившем.ившие.ивших.ившим.ивших.ившие.ившими.ивших"
    gPattern(322)                                                                                        =
"евший.евшего.евшему.евшего.евший.евшим.евшем.евшая.евшей.евшей.евшую.евшую.евшей.евшей.евшее.евшего.евшему.евшее.е
вшее.евшим.евшем.евшие.евших.евшим.евших.евшие.евшими.евших"
    gPattern(323)                                                                                        =
"авший.авшего.авшему.авшего.авший.авшим.авшем.авшая.авшей.авшей.авшую.авшую.авшей.авшей.авшее.авшего.авшему.авшее.а
вшее.авшим.авшем.авшие.авших.авшим.авших.авшие.авшими.авших"
    gPattern(324)                                                                                        =
"ший.шего.шему.шего.ший.шим.шем.шая.шей.шей.шую.шую.шей.шей.шее.шего.шему.шее.шее.шим.шем.шие.ших.шим.ших.шие.шими.ш
их"
    gPattern(325)                                                                                        =
"овавший.овавшего.овавшему.овавшего.овавший.овавшим.овавшем.овавшая.овавшей.овавшей.овавшую.овавшую.овавшей.овавше
й.овавшее.овавшего.овавшему.овавшее.овавшее.овавшим.овавшем.овавшие.овавших.овавшим.овавших.овавшие.овавшими.овавш
их"

    'past passive
    gPattern(340)                                                                                        =
"тый.того.тому.ого.тый.тым.том.т./.тая.той.той.тую.тую.той.той.та./.тое.того.тому.тое.тое.тым.том.то./.тые.тых.тым.т
ых.тые.тыми.тых.ты./"
    gPattern(341)                                                                                        =
"ытый.ытого.ытому.ытого.ытый.ытым.ытом.ыт./.ытая.ытой.ытой.ытую.ытую.ытой.ытой.ыта./.ытое.ытого.ытому.ытое.ытое.ытым
.ытом.ыто./.ытые.ытых.ытым.ытых.ытые.ытыми.ытых.ыты./"
    gPattern(342)                                                                                        =
"етый.етого.етому.етого.етый.етым.етом.ет./.етая.етой.етой.етую.етую.етой.етой.ета./.етое.етого.етому.етое.етое.етым
.етом.ето./.етые.етых.етым.етых.етые.етыми.етых.еты./"
    gPattern(343)                                                                                        =
"нный.нного.нному.нного.нный.нным.нном.н.н.нная.нной.нной.нную.нную.нной.нной.на.нна.нное.нного.нному.нное.нное.нным
.нном.но.нно.нные.нных.нным.нных.нные.нными.нных.ны.нны"
    gPattern(344)                                                                                        =
"ленный.ленного.ленному.ленного.ленный.ленным.ленном.лен.лен.ленная.ленной.ленной.ленную.ленную.ленной.ленной.лена.л
енна.ленное.ленного.ленному.ленное.ленное.ленным.ленном.лено.ленно.ленные.ленных.ленным.ленных.ленные.ленными.ленных
.лены.ленны"
    gPattern(345)                                                                                        =
"енный.енного.енному.енного.енный.енным.енном.ен.ен.енная.енной.енной.енную.енную.енной.енной.ена.енна.енное.енного.
енному.енное.енное.енным.енном.ено.енно.енные.енных.енным.енных.енные.енными.енных.ены.енны"
```

```
    gPattern(346)                                                                                    =
"ённый.ённого.ённому.ённого.ённый.ённым.ённом.ён.ён.ённая.ённой.ённой.ённую.ённую.ённой.ённой.ёна.ённа.ённое.ённого.
ённому.ённое.ённое.ённым.ённом.ёно.ённо.ённые.ённых.ённым.ённых.ённые.ёнными.ённых.ёны.ённы"
    gPattern(347)                                                                                    =
"ованный.ованного.ованному.ованного.ованный.ованным.ованном.ован.ован.ованная.ованной.ованной.ованную.ованную.ованн
ой.ованной.ована.ованна.ованное.ованного.ованному.ованное.ованное.ованным.ованном.овано.ованно.ованные.ованных.ован
ным.ованных.ованные.ованными.ованных.ованы.ованны"

    'reflexive present active
    gPattern(360)                                                                                    =
"ащийся.ащегося.ащемуся.ащегося.ащийся.ащимся.ащемся.ащаяся.ащейся.ащейся.ащуюся.ащуюся.ащейся.ащейся.ащееся.ащегося.ащ
емуся.ащееся.ащееся.ащимся.ащемся.ащиеся.ащихся.ащимся.ащихся.ащиеся.ащимися.ащихся"
    gPattern(361)                                                                                    =
"ящийся.ящегося.ящемуся.ящегося.ящийся.ящимся.ящемся.ящаяся.ящейся.ящейся.ящуюся.ящуюся.ящейся.ящейся.ящееся.ящегося.ящ
емуся.ящееся.ящееся.ящимся.ящемся.ящиеся.ящихся.ящимся.ящихся.ящиеся.ящимися.ящихся"
    gPattern(362)                                                                                    =
"ущийся.ущегося.ущемуся.ущегося.ущийся.ущимся.ущемся.ущаяся.ущейся.ущейся.ущуюся.ущуюся.ущейся.ущейся.ущееся.ущегося.у
щемуся.ущееся.ущееся.ущимся.ущемся.ущиеся.ущихся.ущимся.ущихся.ущиеся.ущимися.ущихся"
    gPattern(363)                                                                                    =
"ющийся.ющегося.ющемуся.ющегося.ющийся.ющимся.ющемся.ющаяся.ющейся.ющейся.ющуюся.ющуюся.ющейся.ющейся.ющееся.ющегося.ющ
емуся.ющееся.ющееся.ющимся.ющемся.ющиеся.ющихся.ющимся.ющихся.ющиеся.ющимися.ющихся"
    gPattern(364)                                                                                    =
"ующийся.ующегося.ующемуся.ующегося.ующийся.ующимся.ующемся.ующаяся.ующейся.ующейся.ующуюся.ующуюся.ующейся.ующейся.ую
щееся.ующегося.ующемуся.ующееся.ующееся.ующимся.ующемся.ующиеся.ующихся.ующимся.ующихся.ующиеся.ующимися.ующихся"

    'reflexive past active
    gPattern(370)                                                                                    =
"вшийся.вшегося.вешмуся.вшегося.вшийся.вшимся.вшемся.вшаяся.вшейся.вшейся.вшуюся.вшуюся.вшейся.вшейся.вшееся.вшегос
я.вшемуся.вшееся.вшееся.вшимся.вшемся.вшиеся.вшихся.вшимся.вшихся.вшиеся.вшимися.вшихся"
    gPattern(371)                                                                                    =
"ившийся.ившегося.ивешмуся.ившегося.ившийся.ившимся.ившемся.ившаяся.ившейся.ившейся.ившуюся.ившуюся.ившейся.ившейся
.ившееся.ившегося.ившемуся.ившееся.ившееся.ившимся.ившемся.ившиеся.ивших ся.ившимся.ившихся.ившиеся.ившимися.ившихся
"
    gPattern(372)                                                                                    =
"евшаяся.евшейся.евшейся.евшуюся.евшуюся.евшейся.евшейся.евшаяся.евшейся.евшейся.евшуюся.евшуюся.евшейся.евшейся.ев
шееся.евшегося.евшемуся.евшееся.евшееся.евшимся.евшемся.евшиеся.евшихся.евшимся.евшихся.евшиеся.евшимися.евшихся"
    gPattern(373)                                                                                    =
"авшийся.авшегося.авешмуся.авшегося.авшийся.авшимся.авшемся.авшаяся.авшейся.авшейся.авшуюся.авшуюся.авшейся.авшейся
.авшееся.авшегося.авшемуся.авшееся.авшееся.авшимся.авшемся.авшиеся.авшихся.авшимся.авшихся.авшиеся.авшимися.авшихся
"
    gPattern(374)                                                                                    =
"шийся.шегося.ешмуся.шегося.шийся.шимся.шемся.шаяся.шейся.шейся.шуюся.шуюся.шейся.шейся.шееся.шегося.шемуся.шееся.ше
еся.шимся.шемся.шиеся.шихся.шимся.шихся.шиеся.шимися.шихся"
    gPattern(375)                                                                                    =
"овавшийся.овавшегося.овавешмуся.овавшегося.овавшийся.овавшимся.овавшемся.овавшаяся.овавшейся.овавшейся.овавшуюся.о
```

```
вавшуюся.овавшейся.овавшейся.овавшееся.овавшегося.овавшемуся.овавшееся.овавшееся.овавшимся.овавшемся.овавшиеся.овав
шихся.овавшимся.овавшихся.овавшиеся.овавшимися.овавшихся"

    'interrogatives:
    '(format "nom.gen.dat.acc.ins.pre")
    gPattern(380) = "то.ого.ому.ого.ем.ом"
    gPattern(381) = "то.его.ему.то.ем.ём"

    'personal pronouns:
    '(format "nom.gen.dat.acc.ins.pre")
    gPattern(390) = "#./././././"
    gPattern(391) = "/.еня.не.еня.ной.не"
    gPattern(392) = "ы.ебя.ебе.ебя.обой.ебе"
    gPattern(393) = "#./././././"
    gPattern(394) = "/.го.му.го./"
    gPattern(395) = "/./././.#./"
    gPattern(396) = "/././././.#"
    gPattern(397) = "#./././././"
    gPattern(398) = "/.ё.й.ё.й./"
    gPattern(399) = "/././././.#"
    gPattern(400) = "#./././././"
    gPattern(401) = "/.с.м.с.ми.с"
    gPattern(402) = "#./././././"
    gPattern(403) = "/.х.м.х.ми./"
    gPattern(404) = "/././././.#"

    'relative pronoun
    gPattern(410) = "ый.ого.ому.ого.ый.ым.ом.ая.ой.ой.ую.ую.ой.ой.ое.ого.ому.ое.ое.ым.ом.ые.ых.ым.ых.ые.ыми.ых"

    'make sure no inflection patterns are duplicated
    VerifyUniqueInflectionPatterns

End Sub



'_____________________________________________________________________________________________________
'define lookup table to order cyrillic characters
'_____________________________________________________________________________________________________
Sub DefineSortOrder ()

    '      lowercase         :        uppercase
    gSortOrder(Asc("а")) = 33: gSortOrder(Asc("А")) = 33
    gSortOrder(Asc("б")) = 32: gSortOrder(Asc("Б")) = 32
    gSortOrder(Asc("в")) = 31: gSortOrder(Asc("В")) = 31
```

```
    gSortOrder(Asc("г")) = 30: gSortOrder(Asc("Г")) = 30
    gSortOrder(Asc("д")) = 29: gSortOrder(Asc("Д")) = 29
    gSortOrder(Asc("е")) = 28: gSortOrder(Asc("Е")) = 28
    gSortOrder(Asc("ё")) = 27: gSortOrder(Asc("Ё")) = 27
    gSortOrder(Asc("ж")) = 26: gSortOrder(Asc("Ж")) = 26
    gSortOrder(Asc("з")) = 25: gSortOrder(Asc("З")) = 25
    gSortOrder(Asc("и")) = 24: gSortOrder(Asc("И")) = 24
    gSortOrder(Asc("й")) = 23: gSortOrder(Asc("Й")) = 23
    gSortOrder(Asc("к")) = 22: gSortOrder(Asc("К")) = 22
    gSortOrder(Asc("л")) = 21: gSortOrder(Asc("Л")) = 21
    gSortOrder(Asc("м")) = 20: gSortOrder(Asc("М")) = 20
    gSortOrder(Asc("н")) = 19: gSortOrder(Asc("Н")) = 19
    gSortOrder(Asc("о")) = 18: gSortOrder(Asc("О")) = 18
    gSortOrder(Asc("п")) = 17: gSortOrder(Asc("П")) = 17
    gSortOrder(Asc("р")) = 16: gSortOrder(Asc("Р")) = 16
    gSortOrder(Asc("с")) = 15: gSortOrder(Asc("С")) = 15
    gSortOrder(Asc("т")) = 14: gSortOrder(Asc("Т")) = 14
    gSortOrder(Asc("у")) = 13: gSortOrder(Asc("У")) = 13
    gSortOrder(Asc("ф")) = 12: gSortOrder(Asc("Ф")) = 12
    gSortOrder(Asc("х")) = 11: gSortOrder(Asc("Х")) = 11
    gSortOrder(Asc("ц")) = 10: gSortOrder(Asc("Ц")) = 10
    gSortOrder(Asc("ч")) = 9:  gSortOrder(Asc("Ч")) = 9
    gSortOrder(Asc("ш")) = 8:  gSortOrder(Asc("Ш")) = 8
    gSortOrder(Asc("щ")) = 7:  gSortOrder(Asc("Щ")) = 7
    gSortOrder(Asc("ъ")) = 6:  gSortOrder(Asc("Ъ")) = 6
    gSortOrder(Asc("ы")) = 5:  gSortOrder(Asc("Ы")) = 5
    gSortOrder(Asc("ь")) = 4:  gSortOrder(Asc("Ь")) = 4
    gSortOrder(Asc("э")) = 3:  gSortOrder(Asc("Э")) = 3
    gSortOrder(Asc("ю")) = 2:  gSortOrder(Asc("Ю")) = 2
    gSortOrder(Asc("я")) = 1:  gSortOrder(Asc("Я")) = 1

End Sub

'_____________________________________________________________________________________________________________
'define internal constants used to convert numeric inflection codes in lexicon file to internal
'symbolic representation.  Symbolic is slower to process, but easier to understand and program
'_____________________________________________________________________________________________________________
Sub DefineSymbolicConstantTable ()

    'define noun code table
    gNounConstant(1) = "NO_N_"              'nominative singular (gender and demand case to
    gNounConstant(2) = "NO_G_"              'genitive singular    be appended when determined)
    gNounConstant(3) = "NO_D_"              'dative singular                  "
    gNounConstant(4) = "NO_A_"              'accusative singular              "
    gNounConstant(5) = "NO_I_"              'instrumental singular            "
    gNounConstant(6) = "NO_P_"              'prepositional singular           "
```

```
gNounConstant(7) = "NO_L_"              'locative singular                "

gNounConstant(8) = "NO_N_P"             'nominative plural
gNounConstant(9) = "NO_G_P"             'genitive plural
gNounConstant(10) = "NO_D_P"            'dative plural
gNounConstant(11) = "NO_A_P"            'accusative plural
gNounConstant(12) = "NO_I_P"            'instrumental plural
gNounConstant(13) = "NO_P_P"            'prepositional plural

'define adjective code table
gAdjectiveConstant(1) = "AJ_N_M_"       'nominative masculine
gAdjectiveConstant(2) = "AJ_G_M_"       'genitive masculine
gAdjectiveConstant(3) = "AJ_D_M_"       'dative masculine
gAdjectiveConstant(4) = "AJ_1_M_"       'accusative animate masculine
gAdjectiveConstant(5) = "AJ_2_M_"       'accusative inanimate masculine
gAdjectiveConstant(6) = "AJ_I_M_"       'instrumental masculine
gAdjectiveConstant(7) = "AJ_P_M_"       'prepositional masculine
gAdjectiveConstant(8) = "AJ_S_M_"       'short form masculine

gAdjectiveConstant(9) = "AJ_N_F_"       'nominative feminine
gAdjectiveConstant(10) = "AJ_G_F_"      'genitive feminine
gAdjectiveConstant(11) = "AJ_D_F_"      'dative feminine
gAdjectiveConstant(12) = "AJ_A_F_"      'accusative animate feminine
gAdjectiveConstant(13) = "AJ_A_F_"      'accusative inanimate feminine
gAdjectiveConstant(14) = "AJ_I_F_"      'instrumental feminine
gAdjectiveConstant(15) = "AJ_P_F_"      'prepositional feminine
gAdjectiveConstant(16) = "AJ_S_F_"      'short form feminine

gAdjectiveConstant(17) = "AJ_N_N_"      'nominative neuter
gAdjectiveConstant(18) = "AJ_G_N_"      'genitive neuter
gAdjectiveConstant(19) = "AJ_D_N_"      'dative neuter
gAdjectiveConstant(20) = "AJ_A_N_"      'accusative animate neuter
gAdjectiveConstant(21) = "AJ_A_N_"      'accusative inanimate neuter
gAdjectiveConstant(22) = "AJ_I_N_"      'instrumental neuter
gAdjectiveConstant(23) = "AJ_P_N_"      'prepositional neuter
gAdjectiveConstant(24) = "AJ_S_N_"      'short form neuter

gAdjectiveConstant(25) = "AJ_N_P_"      'nominative plural
gAdjectiveConstant(26) = "AJ_G_P_"      'genitive plural
gAdjectiveConstant(27) = "AJ_D_P_"      'dative plural
gAdjectiveConstant(28) = "AJ_1_P_"      'accusative animate plural
gAdjectiveConstant(29) = "AJ_2_P_"      'accusative inanimate plural
gAdjectiveConstant(30) = "AJ_I_P_"      'instrumental plural
gAdjectiveConstant(31) = "AJ_P_P_"      'prepositional plural
gAdjectiveConstant(32) = "AJ_S_P_"      'short form plural
```

```
gAdjectiveConstant(33) = "AJ_C"          'comparative

'define active verb code table
gActiveVerbConstant(1) = "VB_I_"         'infinitive (demand case to be appended when determined)

gActiveVerbConstant(2) = "VB_1_S_"       '1st person singular             "
gActiveVerbConstant(3) = "VB_2_S_"       '2nd person singular             "
gActiveVerbConstant(4) = "VB_3_S_"       '3rd person singular             "
gActiveVerbConstant(5) = "VB_3_K_"       'imperative singular             "

gActiveVerbConstant(6) = "VB_1_P_"       '1st person plural (+imperative) "
gActiveVerbConstant(7) = "VB_2_P_"       '2nd person plural               "
gActiveVerbConstant(8) = "VB_3_P_"       '3rd person plural               "
gActiveVerbConstant(9) = "VB_2_K_"       'imperative plural               "

gActiveVerbConstant(10) = "VB_P_M_"      'masculine past tense            "
gActiveVerbConstant(11) = "VB_P_F_"      'feminine past tense             "
gActiveVerbConstant(12) = "VB_P_N_"      'neuter past tense               "
gActiveVerbConstant(13) = "VB_P_P_"      'plural past tense               "

'define auxilliary verb code table
gAuxiliaryVerbConstant(1) = "VA_I_"      'infinitive (demand case to be appended when determined)

gAuxiliaryVerbConstant(2) = "VA_1_S_"    '1st person singular             "
gAuxiliaryVerbConstant(3) = "VA_2_S_"    '2nd person singular             "
gAuxiliaryVerbConstant(4) = "VA_3_S_"    '3rd person singular             "
gAuxiliaryVerbConstant(5) = "VA_3_K_"    'imperative singular             "

gAuxiliaryVerbConstant(6) = "VA_1_P_"    '1st person plural (+imperative) "
gAuxiliaryVerbConstant(7) = "VA_2_P_"    '2nd person plural               "
gAuxiliaryVerbConstant(8) = "VA_3_P_"    '3rd person plural               "
gAuxiliaryVerbConstant(9) = "VA_2_K_"    'imperative plural               "

gAuxiliaryVerbConstant(10) = "VA_P_M_"   'masculine past tense            "
gAuxiliaryVerbConstant(11) = "VA_P_F_"   'feminine past tense             "
gAuxiliaryVerbConstant(12) = "VA_P_N_"   'neuter past tense               "
gAuxiliaryVerbConstant(13) = "VA_P_P_"   'plural past tense               "

'define adverb code table
gAdverbConstant(1) = "AV_N"              'normal
gAdverbConstant(2) = "AV_C"              'comparative
gAdverbConstant(3) = "AV_S"              'superlative

'define verbal adverb code table
gVerbalAdverbConstant(1) = "VV_S_"       'simultaneous
gVerbalAdverbConstant(2) = "VV_Q_"       'sequential, form I
```

```
gVerbalAdverbConstant(3) = "VV_Q_"      'sequential, form II

'define relative pronoun code table
gRelPronounConstant(1) = "PR_N_M"       'nominative masculine
gRelPronounConstant(2) = "PR_G_M"       'genitive masculine
gRelPronounConstant(3) = "PR_D_M"       'dative masculine
gRelPronounConstant(4) = "PR_1_M"       'accusative animate masculine
gRelPronounConstant(5) = "PR_2_M"       'accusative inanimate masculine
gRelPronounConstant(6) = "PR_I_M"       'instrumental masculine
gRelPronounConstant(7) = "PR_P_M"       'prepositional masculine

gRelPronounConstant(8) = "PR_N_F"       'nominative feminine
gRelPronounConstant(9) = "PR_G_F"       'genitive feminine
gRelPronounConstant(10) = "PR_D_F"      'dative feminine
gRelPronounConstant(11) = "PR_A_F"      'accusative animate feminine
gRelPronounConstant(12) = "PR_A_F"      'accusative inanimate feminine
gRelPronounConstant(13) = "PR_I_F"      'instrumental feminine
gRelPronounConstant(14) = "PR_P_F"      'prepositional feminine

gRelPronounConstant(15) = "PR_N_N"      'nominative neuter
gRelPronounConstant(16) = "PR_G_N"      'genitive neuter
gRelPronounConstant(17) = "PR_D_N"      'dative neuter
gRelPronounConstant(18) = "PR_A_N"      'accusative animate neuter
gRelPronounConstant(19) = "PR_A_N"      'accusative inanimate neuter
gRelPronounConstant(20) = "PR_I_N"      'instrumental neuter
gRelPronounConstant(21) = "PR_P_N"      'prepositional neuter

gRelPronounConstant(22) = "PR_N_P"      'nominative plural
gRelPronounConstant(23) = "PR_G_P"      'genitive plural
gRelPronounConstant(24) = "PR_D_P"      'dative plural
gRelPronounConstant(25) = "PR_1_P"      'accusative animate plural
gRelPronounConstant(26) = "PR_2_P"      'accusative inanimate plural
gRelPronounConstant(27) = "PR_I_P"      'instrumental plural
gRelPronounConstant(28) = "PR_P_P"      'prepositional plural

'define personal pronoun code table
gPersPronounConstant(1) = "PP_N_"       'nominative (person, gender, and number to be appended when determined)
gPersPronounConstant(2) = "PP_G_"       'genitive
gPersPronounConstant(3) = "PP_D_"       'dative
gPersPronounConstant(4) = "PP_A_"       'accusative
gPersPronounConstant(5) = "PP_I_"       'instrumental
gPersPronounConstant(6) = "PP_P_"       'prepositional

'define verbal adjective code table

'present active
```

```
gVAdjPresActConstant(1) = "VJ_PA_N_M_"  'nominative masculine
gVAdjPresActConstant(2) = "VJ_PA_G_M_"  'genitive masculine
gVAdjPresActConstant(3) = "VJ_PA_D_M_"  'dative masculine
gVAdjPresActConstant(4) = "VJ_PA_1_M_"  'accusative animate masculine
gVAdjPresActConstant(5) = "VJ_PA_2_M_"  'accusative inanimate masculine
gVAdjPresActConstant(6) = "VJ_PA_I_M_"  'instrumental masculine
gVAdjPresActConstant(7) = "VJ_PA_P_M_"  'prepositional masculine

gVAdjPresActConstant(8) = "VJ_PA_N_F_"  'nominative feminine
gVAdjPresActConstant(9) = "VJ_PA_G_F_"  'genitive feminine
gVAdjPresActConstant(10) = "VJ_PA_D_F_" 'dative feminine
gVAdjPresActConstant(11) = "VJ_PA_A_F_" 'accusative animate feminine
gVAdjPresActConstant(12) = "VJ_PA_A_F_" 'accusative inanimate feminine
gVAdjPresActConstant(13) = "VJ_PA_I_F_" 'instrumental feminine
gVAdjPresActConstant(14) = "VJ_PA_P_F_" 'prepositional feminine

gVAdjPresActConstant(15) = "VJ_PA_N_N_" 'nominative neuter
gVAdjPresActConstant(16) = "VJ_PA_G_N_" 'genitive neuter
gVAdjPresActConstant(17) = "VJ_PA_D_N_" 'dative neuter
gVAdjPresActConstant(18) = "VJ_PA_A_N_" 'accusative animate neuter
gVAdjPresActConstant(19) = "VJ_PA_A_N_" 'accusative inanimate neuter
gVAdjPresActConstant(20) = "VJ_PA_I_N_" 'instrumental neuter
gVAdjPresActConstant(21) = "VJ_PA_P_N_" 'prepositional neuter

gVAdjPresActConstant(22) = "VJ_PA_N_P_" 'nominative plural
gVAdjPresActConstant(23) = "VJ_PA_G_P_" 'genitive plural
gVAdjPresActConstant(24) = "VJ_PA_D_P_" 'dative plural
gVAdjPresActConstant(25) = "VJ_PA_1_P_" 'accusative animate plural
gVAdjPresActConstant(26) = "VJ_PA_2_P_" 'accusative inanimate plural
gVAdjPresActConstant(27) = "VJ_PA_I_P_" 'instrumental plural
gVAdjPresActConstant(28) = "VJ_PA_P_P_" 'prepositional plural

'present passive
gVAdjPresPasConstant(1) = "VJ_PP_N_M_"  'nominative masculine
gVAdjPresPasConstant(2) = "VJ_PP_G_M_"  'genitive masculine
gVAdjPresPasConstant(3) = "VJ_PP_D_M_"  'dative masculine
gVAdjPresPasConstant(4) = "VJ_PP_1_M_"  'accusative animate masculine
gVAdjPresPasConstant(5) = "VJ_PP_2_M_"  'accusative inanimate masculine
gVAdjPresPasConstant(6) = "VJ_PP_I_M_"  'instrumental masculine
gVAdjPresPasConstant(7) = "VJ_PP_P_M_"  'prepositional masculine

gVAdjPresPasConstant(8) = "VJ_PP_N_F_"  'nominative feminine
gVAdjPresPasConstant(9) = "VJ_PP_G_F_"  'genitive feminine
gVAdjPresPasConstant(10) = "VJ_PP_D_F_" 'dative feminine
gVAdjPresPasConstant(11) = "VJ_PP_A_F_" 'accusative animate feminine
gVAdjPresPasConstant(12) = "VJ_PP_A_F_" 'accusative inanimate feminine
```

```
gVAdjPresPasConstant(13) = "VJ_PP_I_F_" 'instrumental feminine
gVAdjPresPasConstant(14) = "VJ_PP_P_F_" 'prepositional feminine

gVAdjPresPasConstant(15) = "VJ_PP_N_N_" 'nominative neuter
gVAdjPresPasConstant(16) = "VJ_PP_G_N_" 'genitive neuter
gVAdjPresPasConstant(17) = "VJ_PP_D_N_" 'dative neuter
gVAdjPresPasConstant(18) = "VJ_PP_A_N_" 'accusative animate neuter
gVAdjPresPasConstant(19) = "VJ_PP_A_N_" 'accusative inanimate neuter
gVAdjPresPasConstant(20) = "VJ_PP_I_N_" 'instrumental neuter
gVAdjPresPasConstant(21) = "VJ_PP_P_N_" 'prepositional neuter

gVAdjPresPasConstant(22) = "VJ_PP_N_P_" 'nominative plural
gVAdjPresPasConstant(23) = "VJ_PP_G_P_" 'genitive plural
gVAdjPresPasConstant(24) = "VJ_PP_D_P_" 'dative plural
gVAdjPresPasConstant(25) = "VJ_PP_1_P_" 'accusative animate plural
gVAdjPresPasConstant(26) = "VJ_PP_2_P_" 'accusative inanimate plural
gVAdjPresPasConstant(27) = "VJ_PP_I_P_" 'instrumental plural
gVAdjPresPasConstant(28) = "VJ_PP_P_P_" 'prepositional plural

'past active
gVAdjPastActConstant(1) = "VJ_AA_N_M_"  'nominative masculine
gVAdjPastActConstant(2) = "VJ_AA_G_M_"  'genitive masculine
gVAdjPastActConstant(3) = "VJ_AA_D_M_"  'dative masculine
gVAdjPastActConstant(4) = "VJ_AA_1_M_"  'accusative animate masculine
gVAdjPastActConstant(5) = "VJ_AA_2_M_"  'accusative inanimate masculine
gVAdjPastActConstant(6) = "VJ_AA_I_M_"  'instrumental masculine
gVAdjPastActConstant(7) = "VJ_AA_P_M_"  'prepositional masculine

gVAdjPastActConstant(8) = "VJ_AA_N_F_"  'nominative feminine
gVAdjPastActConstant(9) = "VJ_AA_G_F_"  'genitive feminine
gVAdjPastActConstant(10) = "VJ_AA_D_F_" 'dative feminine
gVAdjPastActConstant(11) = "VJ_AA_A_F_" 'accusative animate feminine
gVAdjPastActConstant(12) = "VJ_AA_A_F_" 'accusative inanimate feminine
gVAdjPastActConstant(13) = "VJ_AA_I_F_" 'instrumental feminine
gVAdjPastActConstant(14) = "VJ_AA_P_F_" 'prepositional feminine

gVAdjPastActConstant(15) = "VJ_AA_N_N_" 'nominative neuter
gVAdjPastActConstant(16) = "VJ_AA_G_N_" 'genitive neuter
gVAdjPastActConstant(17) = "VJ_AA_D_N_" 'dative neuter
gVAdjPastActConstant(18) = "VJ_AA_A_N_" 'accusative animate neuter
gVAdjPastActConstant(19) = "VJ_AA_A_N_" 'accusative inanimate neuter
gVAdjPastActConstant(20) = "VJ_AA_I_N_" 'instrumental neuter
gVAdjPastActConstant(21) = "VJ_AA_P_N_" 'prepositional neuter

gVAdjPastActConstant(22) = "VJ_AA_N_P_" 'nominative plural
gVAdjPastActConstant(23) = "VJ_AA_G_P_" 'genitive plural
```

```
gVAdjPastActConstant(24) = "VJ_AA_D_P_" 'dative plural
gVAdjPastActConstant(25) = "VJ_AA_1_P_" 'accusative animate plural
gVAdjPastActConstant(26) = "VJ_AA_2_P_" 'accusative inanimate plural
gVAdjPastActConstant(27) = "VJ_AA_I_P_" 'instrumental plural
gVAdjPastActConstant(28) = "VJ_AA_P_P_" 'prepositional plural

'past passive
gVAdjPastPasConstant(1) = "VJ_AP_N_M_"  'nominative masculine
gVAdjPastPasConstant(2) = "VJ_AP_G_M_"  'genitive masculine
gVAdjPastPasConstant(3) = "VJ_AP_D_M_"  'dative masculine
gVAdjPastPasConstant(4) = "VJ_AP_1_M_"  'accusative animate masculine
gVAdjPastPasConstant(5) = "VJ_AP_2_M_"  'accusative inanimate masculine
gVAdjPastPasConstant(6) = "VJ_AP_I_M_"  'instrumental masculine
gVAdjPastPasConstant(7) = "VJ_AP_P_M_"  'prepositional masculine
gVAdjPastPasConstant(8) = "VJ_AP_S_M_"  'short form I masculine
gVAdjPastPasConstant(9) = "AJ_S_M"      'short form II masculine (given adjective symbol)

gVAdjPastPasConstant(10) = "VJ_AP_N_F_" 'nominative feminine
gVAdjPastPasConstant(11) = "VJ_AP_G_F_" 'genitive feminine
gVAdjPastPasConstant(12) = "VJ_AP_D_F_" 'dative feminine
gVAdjPastPasConstant(13) = "VJ_AP_A_F_" 'accusative animate feminine
gVAdjPastPasConstant(14) = "VJ_AP_A_F_" 'accusative inanimate feminine
gVAdjPastPasConstant(15) = "VJ_AP_I_F_" 'instrumental feminine
gVAdjPastPasConstant(16) = "VJ_AP_P_F_" 'prepositional feminine
gVAdjPastPasConstant(17) = "VJ_AP_S_F_" 'short form I feminine
gVAdjPastPasConstant(18) = "AJ_S_F"     'short form II feminine (given adjective symbol)

gVAdjPastPasConstant(19) = "VJ_AP_N_N_" 'nominative neuter
gVAdjPastPasConstant(20) = "VJ_AP_G_N_" 'genitive neuter
gVAdjPastPasConstant(21) = "VJ_AP_D_N_" 'dative neuter
gVAdjPastPasConstant(22) = "VJ_AP_A_N_" 'accusative animate neuter
gVAdjPastPasConstant(23) = "VJ_AP_A_N_" 'accusative inanimate neuter
gVAdjPastPasConstant(24) = "VJ_AP_I_N_" 'instrumental neuter
gVAdjPastPasConstant(25) = "VJ_AP_P_N_" 'prepositional neuter
gVAdjPastPasConstant(26) = "VJ_AP_S_N_" 'short form neuter
gVAdjPastPasConstant(27) = "AJ_S_N"     'short form II neuter (given adjective symbol)

gVAdjPastPasConstant(28) = "VJ_AP_N_P_" 'nominative plural
gVAdjPastPasConstant(29) = "VJ_AP_G_P_" 'genitive plural
gVAdjPastPasConstant(30) = "VJ_AP_D_P_" 'dative plural
gVAdjPastPasConstant(31) = "VJ_AP_1_P_" 'accusative animate plural
gVAdjPastPasConstant(32) = "VJ_AP_2_P_" 'accusative inanimate plural
gVAdjPastPasConstant(33) = "VJ_AP_I_P_" 'instrumental plural
gVAdjPastPasConstant(34) = "VJ_AP_P_P_" 'prepositional plural
gVAdjPastPasConstant(35) = "VJ_AP_S_P_" 'short form plural
gVAdjPastPasConstant(36) = "AJ_S_P"     'short form II plural (given adjective symbol)
```

```
    'define interrogative code table
    gInterrogativeConstant(1) = "IN_N"      'nominative
    gInterrogativeConstant(2) = "IN_G"      'genitive
    gInterrogativeConstant(3) = "IN_D"      'dative
    gInterrogativeConstant(4) = "IN_A"      'accusative
    gInterrogativeConstant(5) = "IN_I"      'instrumental
    gInterrogativeConstant(6) = "IN_P"      'prepositional

    'define preposition code table
    gPrepositionCase(1) = "EP_G"            'genitive
    gPrepositionCase(2) = "EP_D"            'dative
    gPrepositionCase(3) = "EP_A"            'accusative
    gPrepositionCase(4) = "EP_I"            'instrumental
    gPrepositionCase(5) = "EP_P"            'prepositional

    'define conjunction code
    gConjunctionConstant(1) = "CJ"          'conjunction

End Sub

'_____________________________________________________________________________________________________
'delete lexicon entry and regenerate lexicon index
'_____________________________________________________________________________________________________
Sub DeleteLexiconEntry (DeletePtr As Integer)

    Dim LexiconPtr As Integer


    'decrease lexicon length by one element
    gLexiconEndPtr = gLexiconEndPtr - 1

    'delete entry
    For LexiconPtr = DeletePtr To gLexiconEndPtr
      gLexicon(LexiconPtr) = gLexicon(LexiconPtr + 1)
    Next LexiconPtr

    'regenerate index
    GenerateLexiconIndex

End Sub

'_____________________________________________________________________________________________________
'for words where more than one translation possible, try to pick correct one
'_____________________________________________________________________________________________________
```

```
Sub DisambiguateSentence ()

    Dim WordPtr        As Integer
    Dim MatchPtr       As Integer
    Dim WordChoice     As Integer
    Dim AmbiguousWords As Integer

    Dim i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15, i16 As Integer
    Dim i17, i18, i19, i20, i21, i22, i23, i24, i25, i26, i27, i28, i29, i30, i31 As Integer


    'log details
    AddDetail "  Sentence disambiguation:", False
    AddDetail "", False

    'check each word for multiple translations
    For WordPtr = 0 To MAX_SENTENCE_COMPONENTS
       If gSentence(WordPtr).NumMatches > 1 Then
          'evaluate each disambiguation expression
          WordChoice = 0
          AmbiguousWords = AmbiguousWords + 1
          For MatchPtr = 1 To gSentence(WordPtr).NumMatches
              'evaluate disambiguation expression:
              'choose this word; otherwise, don't choose this word
              If     EvaluateDisambiguationExpression(gSentence(WordPtr).Match(MatchPtr).LexiconEntry.Disambiguation,
WordPtr) Then
               WordChoice = MatchPtr
               Exit For
              End If
          Next MatchPtr
          'not decision made, so make random choice; otherwise...
          If WordChoice = 0 Then
             Randomize
             WordChoice = Int(Rnd * (gSentence(WordPtr).NumMatches - 1)) + 1
             AddDetail  "          ("  &  gSentence(WordPtr).LiteralText  &  ")  randomly  disambiguated  as  ("  &
gSentence(WordPtr).Match(WordChoice).LexiconEntry.Target & ").", False
          '...take chosen word
          Else
             AddDetail   "             ("   &   gSentence(WordPtr).LiteralText   &   ")   disambiguated   as   ("   &
gSentence(WordPtr).Match(WordChoice).LexiconEntry.Target & ").", False
          End If

          'mark chosen word
          gSentence(WordPtr).MatchPtr = WordChoice
       Else
          'no disambiguation necessary word
```

```
         gSentence(WordPtr).MatchPtr = 1
      End If
   Next WordPtr


   'log details
   If AmbiguousWords = 0 Then
      AddDetail "     No ambiguous words.", False
   End If
   AddDetail "", False

End Sub

'_____________________________________________________________________________________________________
'put up status information for time-consuming processes.  Multiple operations are supported here by
'varying parameters
'_____________________________________________________________________________________________________
Sub DisplayWait (Instruction As Integer, WaitLabel As String)

   On Error Resume Next

   Static EndPoint         As Integer
   Static CurrentPoint     As Integer
   Static ScrollLockStatus As Integer

   Dim Percent As Single
   Dim i       As Integer


   'passing in negative number sets range for percent bar and displays Wait indicator
   'with message
   DoEvents
   If Instruction < 0 Then
      MousePointerWait
      EndPoint = -Instruction
      CurrentPoint = 0
      'describe wait period, unless suppressed
      PassiveHelp WaitLabel

      frmMain.pnlStatus.FloodPercent = 0
      frmMain.pnlStatus.FloodShowPct = True
      DoEvents
   'zero deactivates Wait indicator
   ElseIf Instruction = 0 Then
      frmMain.pnlStatus.FloodPercent = 100
      PassiveHelp ""
```

```
      'slight delay to allow 100% to be shown
      For i = 1 To 7000
        DoEvents
      Next i
      frmMain.pnlStatus.FloodShowPct = False
      frmMain.pnlStatus.FloodPercent = 0
      MousePointerContinue
   'positive updates percent as (CurrentPoint/EndPoint)*100 by incrementing CurrentPoint
   'with Instruction value passed in
   Else
      CurrentPoint = CurrentPoint + Instruction
      Percent = (CurrentPoint / EndPoint) * 100
      'keep percent within legal range
      If Percent > 100 Then
       Percent = 100
      End If
      frmMain.pnlStatus.FloodPercent = Percent
   End If

End Sub

'_____________________________________________________________________________________________________
'write final translation to log file
'_____________________________________________________________________________________________________
Sub DumpFinalTranslation ()

    Dim CharPtr     As Integer
    Dim StartPtr    As Integer
    Dim Char        As String
    Dim Sentence    As String
    Dim TRANSLATION As String


    'get final translation
    TRANSLATION = frmTranslation.txtTranslation & Chr(1)

    'don't dump empty translation
    If TRANSLATION = "" Then
       Exit Sub
    End If

    'dump each sentence
    StartPtr = 1
    Do
       'extract sentence
       For CharPtr = StartPtr To Len(TRANSLATION)
```

```
         Char = Mid(TRANSLATION, CharPtr, 1)
         If Char = Chr(1) Then
            Exit Do
         End If
         Sentence = Sentence & Char
         'found sentence terminator
         If InStr(".!?", Char) Then
            StartPtr = CharPtr + 1
            Exit For
         End If
      Next CharPtr

      'dump sentence
      AddDetail "  " & Trim(Sentence), False
      Sentence = ""
   Loop Until StartPtr >= Len(TRANSLATION) - 2

End Sub

'_____________________________________________________________________________________________
'evaluate logical expression describing when to choose this translation of a word
'_____________________________________________________________________________________________
Function EvaluateDisambiguationExpression (Expression As String, CurrentWordPtr As Integer) As Integer

   Dim CharPtr        As Integer
   Dim CharPtr2       As Integer
   Dim WordPtr        As Integer
   Dim SymbolPtr      As Integer
   Dim MatchPtr       As Integer
   Dim Range          As Integer
   Dim Direction      As Integer
   Dim ParameterCode  As Integer
   Dim ParameterStart As Integer
   Dim ParameterEnd   As Integer
   Dim ParameterType  As Integer
   Dim TermEvaluation As Integer
   Dim TempEvaluation As Integer
   Dim StartPtr       As Integer
   Dim EndPtr         As Integer
   Dim Char           As String * 1
   Dim Parameter      As String
   Dim Operation      As String

   Const BACKWARD = 1
   Const FORWARD = 2
   Const BOTH = 3
```

```
Const LITERAL = 1
Const SYMBOLIC = 2
Const ASSOCIATIVE = 3

On Error GoTo ExpressionError


'default is to fail
TempEvaluation = False

'evaluate expression character by character
For CharPtr = 1 To Len(Expression)
   Char = Mid(Expression, CharPtr, 1)

   'build logical operation
   Operation = Operation & Trim(Char)

   'start of term
   If Char = "(" Then
      ParameterStart = 0
      ParameterEnd = 0

      Operation = Left(Operation, Len(Operation) - 1)
      Range = Abs(Val(Mid(Expression, CharPtr + 1)))
      Char = Mid(Expression, CharPtr + 1, 1)
      'check 'Range' words after
      If Char = "+" Then
         Direction = FORWARD
      'check 'Range' words before
      ElseIf Char = "-" Then
         Direction = BACKWARD
      'check 'Range' words before and after
      Else
         Direction = BOTH
      End If

      'get compare parameter
      For CharPtr2 = CharPtr + 2 To Len(Expression)
          Char = Mid(Expression, CharPtr2, 1)
          'literal compare
          If Char = "'" Then
            'term start
            If ParameterStart = 0 Then
               ParameterType = LITERAL
               ParameterStart = CharPtr2 + 1
```

```
      'term end
      Else
         ParameterEnd = CharPtr2
         Exit For
      End If
     'symbolic compare:
     'parameter start
     ElseIf Char = "[" Then
      ParameterType = SYMBOLIC
      ParameterStart = CharPtr2 + 1
     'parameter end
     ElseIf Char = "]" Then
      ParameterEnd = CharPtr2
      Exit For
     'associative compare:
     'parameter start
     ElseIf Char = "<" Then
      ParameterType = ASSOCIATIVE
      ParameterStart = CharPtr2 + 1
     'parameter end
     ElseIf Char = ">" Then
      ParameterEnd = CharPtr2
      Exit For
     End If
Next CharPtr2

'expression error
If (ParameterStart = 0) Or (ParameterEnd = 0) Or (ParameterStart >= ParameterEnd) Then
   Error 32767
End If

'extract compare parameter
Parameter = Mid(Expression, ParameterStart, ParameterEnd - ParameterStart)

'find end of term
For CharPtr2 = ParameterEnd + 1 To Len(Expression)
     If Mid(Expression, CharPtr2, 1) = ")" Then
      CharPtr = CharPtr2 + 1
      Exit For
     End If
Next CharPtr2

'set search radius from current word:
'forward
If Direction = FORWARD Then
  StartPtr = CurrentWordPtr
```

```
          EndPtr = CurrentWordPtr + Range
        'backward
        ElseIf Direction = BACKWARD Then
          StartPtr = CurrentWordPtr - Range
          EndPtr = CurrentWordPtr
        'backward and forward
        Else
          StartPtr = CurrentWordPtr - Range
          EndPtr = CurrentWordPtr + Range
        End If

        'don't start before sentence
        If StartPtr < 0 Then
           StartPtr = 0
        End If
        'don't end beyond sentence limit
        If EndPtr > MAX_SENTENCE_COMPONENTS Then
           EndPtr = MAX_SENTENCE_COMPONENTS
        End If

        'generate transfer attribute code from associative parameter
        If ParameterType = ASSOCIATIVE Then
           Select Case UCase(Parameter):
                Case "PERSON":     ParameterCode = 1
                Case "PLACE":      ParameterCode = 2
                Case "THING":      ParameterCode = 4
                Case "ANIMAL":     ParameterCode = 8
                Case "FIGURATIVE": ParameterCode = 16
                Case Else
                    Error 32767
           End Select
        End If

        'evaluate expression terms
        TermEvaluation = False
        For WordPtr = StartPtr To EndPtr
            Select Case ParameterType
                 Case LITERAL:
                     'match literal string against all word combinations within search radius
                     For MatchPtr = 1 To gSentence(WordPtr).NumMatches
                        If     (InStr(gSentence(WordPtr).Match(MatchPtr).LexiconEntry.Stem,     Parameter))     Or
(InStr(gSentence(WordPtr).LiteralText, Parameter)) Then
                          TermEvaluation = True
                          Exit For
                        End If
                     Next MatchPtr
```

```
                    Case SYMBOLIC:
                        'match symbolic string against all word/symbolic constant combinations within search radius
                        For MatchPtr = 1 To gSentence(WordPtr).NumMatches
                            For                    SymbolPtr                    =                    1                    To
gSentence(WordPtr).Match(MatchPtr).SymbolicConstants.NumSymbolicConstants
                               If
InStr(gSentence(WordPtr).Match(MatchPtr).SymbolicConstants.SymbolicConstant(SymbolPtr), Parameter) Then
                                  TermEvaluation = True
                                  Exit For
                               End If
                            Next SymbolPtr
                        Next MatchPtr

                    Case ASSOCIATIVE:
                        'match association code against all word association combinations within search radius
                        For MatchPtr = 1 To gSentence(WordPtr).NumMatches
                            If (gSentence(WordPtr).Match(MatchPtr).LexiconEntry.TransferAttributes And ParameterCode)
Then
                             TermEvaluation = True
                             Exit For
                           End If
                        Next MatchPtr

                    Case Else
                        Error 32767
                End Select
            Next WordPtr

            'evaluate no operation (implied OR)
            If (Operation = "") Then
               If TermEvaluation Then
                TempEvaluation = True
               End If

            'evaluate AND operation
            ElseIf (Operation = "&") Then
               If TermEvaluation Then
                EvaluateDisambiguationExpression = False
                Exit Function
               End If

            'evaluate AND-NOT operation
            ElseIf (Operation = "&~") Then
               If Not (TermEvaluation) Then
                EvaluateDisambiguationExpression = False
```

```
            Exit Function
          End If

        'evaluate OR operation
        ElseIf (Operation = "|") Then
          If TermEvaluation Then
            EvaluateDisambiguationExpression = True
            Exit Function
          End If

        'evaluate OR-NOT operation
        ElseIf (Operation = "|~") Then
          If Not (TermEvaluation) Then
            EvaluateDisambiguationExpression = True
            Exit Function
          End If

        'evaluate NOT operation
        ElseIf (Operation = "~") Then
          EvaluateDisambiguationExpression = Not TermEvaluation
          Exit Function

        'evalulate XOR operation
        ElseIf (Operation = "%") Then
          EvaluateDisambiguationExpression = True

        'unknown operation
        Else
          Error 32767
        End If

        'reset operation string
        Operation = ""
      End If

    Next CharPtr

    'return expression as evaluate
    EvaluateDisambiguationExpression = TempEvaluation

    Exit Function


ExpressionError:

    'error handler
```

```
    Beep
    MsgBox ("Syntax error in disambiguation expression" & Chr(13) & "'" & Expression & "'."), MB_ICONEXCLAMATION,
("Lexicon Entry Error")
    Exit Function

End Function

'_____________________________________________________________________________________________________
'explode modified regular expression rules to full form
'_____________________________________________________________________________________________________
Sub ExpandCompressedTransferRule (Term() As String, NumTerms As Integer, IsRussianExpression As Integer)

    Dim TermPtr        As Integer
    Dim InnerTermPtr   As Integer
    Dim RawRulePtr     As Integer
    Dim BlockPtr       As Integer
    Dim NextBindPtr    As Integer
    Dim ParameterPtr   As Integer
    Dim RulePtr        As Integer
    Dim Matched        As Integer
    Dim DuplicateRule  As Integer
    Dim IterationCount As Integer
    Dim j, k, l        As Integer
    Dim CurrentTerm    As String
    Dim Rule           As String

    Const UNBOUND = -7
    Const UNRESOLVED = -24

    Static RawRule(MAX_SENTENCE_COMPONENTS)        As String
    Static SiblingBinder(MAX_SENTENCE_COMPONENTS)  As Integer
    Static ParentBinder(MAX_SENTENCE_COMPONENTS)   As Integer
    Static TermIteration(MAX_SENTENCE_COMPONENTS)  As Integer
    Static i(MAX_SENTENCE_COMPONENTS)              As Integer
    Static FinalRule(RULE_COMPLEXITY)              As RawRuleType

    Dim i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15, i16 As Integer
    Dim i17, i18, i19, i20, i21, i22, i23, i24, i25, i26, i27, i28, i29, i30, i31 As Integer

    On Error GoTo ErrorOnRuleDecode

    'build rule string
    For TermPtr = 0 To NumTerms - 1
      Rule = Rule & Term(TermPtr) & " "
    Next TermPtr
    Rule = Rule & Term(TermPtr)
```

```
'initialize terms
For TermPtr = 0 To MAX_SENTENCE_COMPONENTS
   TermIteration(TermPtr) = 1
   SiblingBinder(TermPtr) = UNBOUND
   ParentBinder(TermPtr) = UNBOUND
Next TermPtr

'set term iteration counts and bind parent siblings
NextBindPtr = 0
For TermPtr = 0 To NumTerms
   'start of optional block; otherwise...
   Matched = False
   If InStr(Term(TermPtr), "[") Then
      'mark all terms through end of block
      For BlockPtr = TermPtr To NumTerms
          'bind as siblings
          SiblingBinder(BlockPtr) = NextBindPtr
          'mark as variable
          TermIteration(NextBindPtr) = 0
          CurrentTerm = Term(BlockPtr)
          'end of optional block
          If InStr(Term(BlockPtr), "]") Then
            NextBindPtr = NextBindPtr + 1
            TermPtr = BlockPtr
            Matched = True
            Exit For
          End If
      Next BlockPtr

      'syntax error in transfer expression
      If Not Matched Then
         Error 32600
      End If
   '...not member of block
   Else
      'bind to self
      SiblingBinder(TermPtr) = NextBindPtr
      NextBindPtr = NextBindPtr + 1
   End If
Next TermPtr

'resolve child siblings, first pass
For TermPtr = 0 To NumTerms
   'start of embedded optional block
   Matched = False
```

```
   If InStr(Term(TermPtr), "{") Then
      'mark all terms through end of block
      For BlockPtr = TermPtr To NumTerms
          'pre-bind to parent
          ParentBinder(BlockPtr) = UNRESOLVED
          'bind as siblings
          SiblingBinder(BlockPtr) = NextBindPtr
          'mark as variable
          TermIteration(NextBindPtr) = 0
          'end of optional block
          If InStr(Term(BlockPtr), "}") Then
            Matched = True
            Exit For
          End If
      Next BlockPtr
      NextBindPtr = NextBindPtr + 1
      'syntax error in transfer expression
      If Not Matched Then
         Error 32600
      End If
      TermPtr = BlockPtr
   End If
Next TermPtr

'bind child siblings, second pass
For TermPtr = 0 To NumTerms
   'start of optional block
   If InStr(Term(TermPtr), "[") Then
      'find first unbound term in block
      For BlockPtr = TermPtr To NumTerms
          'got it, or end of optional block
          If (ParentBinder(BlockPtr) = UNBOUND) Or (InStr(Term(BlockPtr), "]")) Then

            'set each unresolved term in block to unbound parent
            For InnerTermPtr = TermPtr To NumTerms
                If ParentBinder(InnerTermPtr) = UNRESOLVED Then
                  ParentBinder(InnerTermPtr) = SiblingBinder(BlockPtr)
                End If
                'end of block reached, so all resolved
                If InStr(Term(InnerTermPtr), "]") Then
                  Exit For
                End If
            Next InnerTermPtr
            Exit For
          End If
      Next BlockPtr
```

```
      TermPtr = BlockPtr
   End If
Next TermPtr

'loop through all combinations of terms, both required and optional
IterationCount = 0
For i0 = 1 To TermIteration(0) Step -1
  i(0) = i0
  For i1 = 1 To TermIteration(1) Step -1
   i(1) = i1
   For i2 = 1 To TermIteration(2) Step -1
     i(2) = i2
     For i3 = 1 To TermIteration(3) Step -1
       i(3) = i3
       For i4 = 1 To TermIteration(4) Step -1
         i(4) = i4
         For i5 = 1 To TermIteration(5) Step -1
          i(5) = i5
          For i6 = 1 To TermIteration(6) Step -1
            i(6) = i6
            For i7 = 1 To TermIteration(7) Step -1
              i(7) = i7
              For i8 = 1 To TermIteration(8) Step -1
                i(8) = i8
                For i9 = 1 To TermIteration(9) Step -1
                 i(9) = i9
                 For i10 = 1 To TermIteration(10) Step -1
                   i(10) = i10
                   For i11 = 1 To TermIteration(11) Step -1
                     i(11) = i11
                     For i12 = 1 To TermIteration(12) Step -1
                       i(12) = i12
                       For i13 = 1 To TermIteration(13) Step -1
                        i(13) = i13
                        For i14 = 1 To TermIteration(14) Step -1
                          i(14) = i14
                          For i15 = 1 To TermIteration(15) Step -1
                            i(15) = i15
                            For i16 = 1 To TermIteration(16) Step -1
                              i(16) = i16
                              For i17 = 1 To TermIteration(17) Step -1
                               i(17) = i17
                               For i18 = 1 To TermIteration(18) Step -1
                                 i(18) = i18
                                 For i19 = 1 To TermIteration(19) Step -1
                                   i(19) = i19
```

```
                                    For i20 = 1 To TermIteration(20) Step -1
                                      i(20) = i20
                                      For i21 = 1 To TermIteration(21) Step -1
                                       i(21) = i21
                                       For i22 = 1 To TermIteration(22) Step -1
                                         i(22) = i22
                                         For i23 = 1 To TermIteration(23) Step -1
                                           i(23) = i23
                                           For i24 = 1 To TermIteration(24) Step -1
                                             i(24) = i24
                                             For i25 = 1 To TermIteration(25) Step -1
                                              i(25) = i25
                                              For i26 = 1 To TermIteration(26) Step -1
                                                i(26) = i26
                                                For i27 = 1 To TermIteration(27) Step -1
                                                  i(27) = i27
                                                  For i28 = 1 To TermIteration(28) Step -1
                                                    i(28) = i28
                                                    For i29 = 1 To TermIteration(29) Step -1
                                                     i(29) = i29
                                                     For i30 = 1 To TermIteration(30) Step -1
                                                       i(30) = i30
                                                       For i31 = 1 To TermIteration(31) Step -1
                                                         i(31) = i31
'build transfer rule from current combination of parameters
ParameterPtr = -1
For TermPtr = 0 To NumTerms
   '...current term on
   If i(SiblingBinder(TermPtr)) Then
      'if bound to parent...
      If ParentBinder(TermPtr) <> UNBOUND Then
        '...generate only if parent on
        If i(ParentBinder(TermPtr)) Then
           ParameterPtr = ParameterPtr + 1
           RawRule(ParameterPtr) = RemoveJunk(Term(TermPtr))
        End If
      '...otherwise fixed term, so always generate
      Else
        ParameterPtr = ParameterPtr + 1
        RawRule(ParameterPtr) = RemoveJunk(Term(TermPtr))
      End If
   End If
 Next TermPtr

 'determine if current rule already in rule base
 DuplicateRule = False
```

```
   For RulePtr = 0 To RawRulePtr - 1
      'check only rules with same number of parameters
      If FinalRule(RulePtr).NumParameters = ParameterPtr Then
         DuplicateRule = True
         'compare each parameter
         For TermPtr = 0 To ParameterPtr
            If FinalRule(RulePtr).Parameter(TermPtr) <> RawRule(TermPtr) Then
               DuplicateRule = False
               Exit For
            End If
         Next TermPtr
         'exit when duplicate found
         If DuplicateRule Then
           Exit For
         End If
      End If
   Next RulePtr

   'if this rule unique, add it to temporary rule base
   If Not DuplicateRule Then
     For TermPtr = 0 To ParameterPtr
         FinalRule(RawRulePtr).Parameter(TermPtr) = RawRule(TermPtr)
     Next TermPtr
     FinalRule(RawRulePtr).NumParameters = ParameterPtr
     RawRulePtr = RawRulePtr + 1
   End If

   'keep track of computational complexity
   IterationCount = IterationCount + 1

Next i31: Next i30: Next i29: Next i28: Next i27: Next i26: Next i25: Next i24
Next i23: Next i22: Next i21: Next i20: Next i19: Next i18: Next i17: Next i16
Next i15: Next i14: Next i13: Next i12: Next i11: Next i10: Next i9:  Next i8
Next i7:  Next i6:  Next i5:  Next i4:  Next i3:  Next i2:  Next i1:  Next i0

'export final rules from simplest to most complex:
'russian expression; otherwise...
If IsRussianExpression Then
   'write each rule
   For RulePtr = RawRulePtr - 1 To 0 Step -1
      'don't process blank rules
      'If FinalRule(RulePtr).Parameter(0) <> "" Then
         gTransferRule(gTransferRulePtr).NumRussianComponents = FinalRule(RulePtr).NumParameters
         'write each parameter of current rule
         For TermPtr = 0 To FinalRule(RulePtr).NumParameters
            gTransferRule(gTransferRulePtr).Russian(TermPtr) = FinalRule(RulePtr).Parameter(TermPtr)
```

```
            Next TermPtr

            'increment transfer rule pointer
            gTransferRulePtr = gTransferRulePtr + 1
         'End If
      Next RulePtr
   '...english expression
   Else
      'write each rule
      For RulePtr = RawRulePtr - 1 To 0 Step -1
         'don't process blank rules
         'If FinalRule(RulePtr).Parameter(0) <> "" Then
            gTransferRule(gTransferRulePtr).NumEnglishComponents = FinalRule(RulePtr).NumParameters
            'write each parameter of current rule
            For TermPtr = 0 To FinalRule(RulePtr).NumParameters
               gTransferRule(gTransferRulePtr).English(TermPtr) = FinalRule(RulePtr).Parameter(TermPtr)
            Next TermPtr

            'increment transfer rule pointer
            gTransferRulePtr = gTransferRulePtr + 1
         'End If
      Next RulePtr
   End If

   Exit Sub


ErrorOnRuleDecode:

   Beep

   'rule too complex; otherwise...
   If Err = 9 Then
      If RulePtr >= RULE_COMPLEXITY Then
       MsgBox ("Formula too complex in transfer rule:" & Chr(13) & " '" & Rule & "'." & Chr(13) & Chr(13) & "It
was only partially decoded."), MB_ICONSTOP, ("Tranfer Rule Error")
      Else
       MsgBox ("The maximum number of transfer rules has been exceeded.  Some rules will be not be available."),
MB_ICONEXCLAMATION, ("Program Limitation")
        'dump current rule
        Exit Sub
      End If
      'dump currently decoded forms
      RawRulePtr = 0
   '...general error in decoding
   Else
```

```
       MsgBox ("Syntax error in transfer rule:" & Chr(13) & "'" & Rule & "'."), MB_ICONSTOP, ("Tranfer Rule Error")
    End If

    Resume Next

End Sub

'_____________________________________________________________________________________________________________
'determine start of each letter segment in resident flat-file lexicon.  Lexicon sorted by first letter
'only since inflected entries mess up sort order
'_____________________________________________________________________________________________________________
Sub GenerateLexiconIndex ()

    Dim LexiconPtr As Integer
    Dim IndexPtr   As Integer
    Dim IndexChar  As String * 1


    'indicate delay
    PassiveHelp "Regenerating lexicon index..."

    'initialize index
    For IndexPtr = 0 To MAX_ASCII
      gLexiconIndex(IndexPtr) = UNUSED
    Next IndexPtr

    'loop through entire lexicon
    For LexiconPtr = 0 To gLexiconEndPtr
      'get first char of entry
      IndexChar = LowerCase(Left(gLexicon(LexiconPtr).Stem, 1))
      'determine position in index array
      IndexPtr = Asc(IndexChar)
      'if not defined yet, this is first, so set segment starting position (ignore entries marked for deletion)
      If (gLexiconIndex(IndexPtr) = UNUSED) Then
         gLexiconIndex(IndexPtr) = LexiconPtr
      End If
    Next LexiconPtr

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
'return if valid noun and extract all possible cases and number
'_____________________________________________________________________________________________________________
```

```
Function GetSymbolicConstant (ByVal StemWord As String, ByVal AbstractWord As String, Word As LexiconType, WordCase
As ConstantType) As Integer

    Dim CasePtr          As Integer
    Dim Ending           As String
    Dim FieldPtr         As Integer
    Dim BreakStart       As Integer
    Dim BreakEnd         As Integer
    Dim DelimiterAt      As Integer
    Dim InflectionField  As String
    Dim InflectionString As String
    Dim Augment          As String
    Dim DemandAugment    As String
    Dim i, j             As Integer


    'string whitespace from start and end of words
    StemWord = LowerCase(Trim(StemWord))
    AbstractWord = LowerCase(Trim(AbstractWord))

    'if stem split (inflection between stem and suffix)
    If InStr(AbstractWord, "-") Then
       'strip suffix of stem word
       DelimiterAt = InStr(StemWord, "~")
       If DelimiterAt > 0 Then
         StemWord = Left(StemWord, DelimiterAt - 1)
       End If
       'strip suffix of abstract word
       AbstractWord = Left(AbstractWord, InStr(AbstractWord, "-") - 1)
    End If

    'extract case ending from stem and abstract word
    Ending = Mid(AbstractWord, Len(StemWord) + 1)

    'initialize list of symbols to empty
    WordCase.NumSymbolicConstants = 0

    'decode preposition and assign case requirement(s); otherwise...
    If Word.PartOfSpeech = PREPOSITION Then

       'genitive
       If Word.SecondaryAttributes And DEMAND3_GENITIVE Then
         WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
         WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "EP_G"
       End If
```

```
    'dative
    If Word.SecondaryAttributes And DEMAND3_DATIVE Then
      WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
      WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "EP_D"
    End If

    'accusative
    If Word.SecondaryAttributes And DEMAND3_ACCUSATIVE Then
      WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
      WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "EP_A"
    End If

    'instrumental
    If Word.SecondaryAttributes And DEMAND3_INSTRUMENTAL Then
      WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
      WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "EP_I"
    End If

    'prepositional
    If Word.SecondaryAttributes And DEMAND3_PREPOSITIONAL Then
      WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
      WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "EP_P"
    End If

'...if indeclinable possessive pronoun, decode (use NOT_APPLICABLE as InflectionPattern); otherwise...
ElseIf Word.PartOfSpeech = PRONOUN_POSSESSIVE And Word.PrimaryAttributes <> POSS_PRONOUN_DECLINABLE Then
    Select Case Word.PrimaryAttributes

          '3rd person masculine or neuter singular indeclinable
          Case POSS_PRONOUN_INDECLINABLE_3RD_MASC_NEUT:
              WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
              WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "PO_X_B"

          '3rd person feminine singular indeclinable
          Case POSS_PRONOUN_INDECLINABLE_3RD_FEMININE:
              WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
              WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "PO_X_F"

          '3rd person plural indeclinable
          Case POSS_PRONOUN_INDECLINABLE_3RD_PLURAL:
              WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
              WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "PO_X_P"

          'error in attribute record
          Case Else:
              Beep
```

```
               MsgBox ("Illegal POS attribute (" & Word.PrimaryAttributes & ") on possessive pronoun '" &
AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
       End Select

    '...if interjection/emphatic particle, assign symbol; otherwise...
    ElseIf Word.PartOfSpeech = INTERJECTION Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "IJ"

    '...if conjunction, assign symbol; otherwise...
    ElseIf Word.PartOfSpeech = CONJUNCTION Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "CJ"

    '...if comparative adjective, assign symbol; otherwise...
    ElseIf Word.PartOfSpeech = ADJECTIVE And Word.PrimaryAttributes = ADJECTIVE_COMPARATIVE Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "AJ_C"

    '...if adverb, decode and assign symbol; otherwise...
    ElseIf Word.PartOfSpeech = ADVERB Then
       Select Case Word.PrimaryAttributes

            Case ADVERB_NORMAL:      Augment = "AV_N_"
                                  GoSub DecodeSecondaryAttribute
            Case ADVERB_COMPARATIVE: WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                                  WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "AV_C"
            Case ADVERB_SUPERLATIVE: WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                                  WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "AV_S"

            'error in attribute record
            Case Else:
                Beep
                MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on adverb '" & AbstractWord &
"'."), MB_ICONSTOP, ("Lexicon Entry Error")
       End Select

    '...decode inflection patterns
    Else

       'verify that valid inflection pattern chosen
       If (Word.InflectionPattern = NOT_APPLICABLE) Or (Word.InflectionPattern > NUM_INFLECTION_PATTERNS) Then
         Beep
         MsgBox ("Invalid inflection pattern selected (" & Word.InflectionPattern & ") for '" & AbstractWord &
"'."), MB_ICONSTOP, ("Lexicon Entry Error")
       End If
```

```
       'verify that defined inflection pattern chosen (pattern array not contiguous)
       If gPattern(Word.InflectionPattern) = "" Then
        Beep
        MsgBox ("Unused inflection pattern selected (" & Word.InflectionPattern & ") for '" & AbstractWord & "'."),
MB_ICONSTOP, ("Lexicon Entry Error")
       End If

       'modify inflection pattern to allow easy searching
       InflectionString = INFLECTION_DELIMITER & gPattern(Word.InflectionPattern) & INFLECTION_DELIMITER
       FieldPtr = 1
       CasePtr = 1
       WordCase.NumSymbolicConstants = 0

       'find enumerated position of all occurrences of ending
       Do
        BreakStart = InStr(FieldPtr, InflectionString, INFLECTION_DELIMITER) + 1
        BreakEnd = InStr(BreakStart, InflectionString, INFLECTION_DELIMITER) - 1
        'if inflection pattern exhausted, exit loop
        If BreakEnd < 0 Then
           Exit Do
        End If
        'extract next field from inflection pattern
        InflectionField = Mid(InflectionString, BreakStart, BreakEnd - BreakStart + 1)
        'if extracted field matches pattern, add this case/number to compiled list for this abstract word
        If (Ending = InflectionField) Or (Ending = "" And InflectionField = "#") Then
           'determine which symbolic constant table to reference
           Select Case Word.PartOfSpeech
                'nouns
                Case NOUN_INANIMATE, NOUN_ANIMATE, PRONOUN_INDEFINITE:

                    'decode demand augment on non-special nouns only...
                    If Word.PartOfSpeech <> PRONOUN_INDEFINITE Then
                       'append demand case
                       Select Case Word.SecondaryAttributes
                            Case DEMAND1_NO_DEMANDS:   DemandAugment = "_0"
                            Case DEMAND1_GENITIVE:     DemandAugment = "_G"
                            Case DEMAND1_DATIVE:       DemandAugment = "_D"
                            Case DEMAND1_INSTRUMENTAL: DemandAugment = "_I"
                            Case Else:
                                Beep
                                MsgBox ("Illegal secondary attribute (" & Word.SecondaryAttributes & ") on noun '"
& AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                       End Select
                    '...special nouns all receive no demand code
                    Else
```

```
                    DemandAugment = "_0"
                End If
                Augment = ""
                'append gender only if not already determined as plural
                If CasePtr <= 7 Then
                    Select Case Word.PrimaryAttributes
                         Case NOUN_MASCULINE: Augment = "M"
                         Case NOUN_FEMININE:  Augment = "F"
                         Case NOUN_NEUTER:    Augment = "N"
                         Case NOUN_PLURAL:    Augment = "P"
                         Case Else:
                             Beep
                             MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on noun '" &
AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                    End Select
                End If
                WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = gNounConstant(CasePtr) & Augment &
DemandAugment

            'demonstative and possessive pronouns
            Case PRONOUN_POSSESSIVE, PRONOUN_DEMONSTRATIVE:
                WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = gAdjectiveConstant(CasePtr) & "0"

            'adjectives
            Case ADJECTIVE:
                'decode adjective type
                Select Case Word.PrimaryAttributes:

                     Case ADJECTIVE_LONG_FORM, ADJECTIVE_SUPERLATIVE:
                          Augment = gAdjectiveConstant(CasePtr)
                          GoSub DecodeSecondaryAttribute
                     Case ADJECTIVE_SHORT_FORM:
                          'decode gender/number of short form adjective
                          Select Case Word.SecondaryAttributes:

                               Case ADJECTIVE_SHORT_FORM_MASCULINE:
                                  Augment = "AJ_S_M_"

                               Case ADJECTIVE_SHORT_FORM_FEMININE:
                                  Augment = "AJ_S_F_"

                               Case ADJECTIVE_SHORT_FORM_NEUTER:
                                  Augment = "AJ_S_N_"
```

```
                                    Case ADJECTIVE_SHORT_FORM_PLURAL:
                                        Augment = "AJ_S_P_"

                                    Case Else
                                        Beep
                                        MsgBox ("Illegal secondary attribute (" & Word.SecondaryAttributes & ") on
short form adjective '" & AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                                End Select
                                GoSub DecodeSecondaryAttribute

                            Case ADJECTIVE_COMPARATIVE:
                                WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                                WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = "AJ_C"

                            'error in attribute record
                            Case Else
                                Beep
                                MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on adjective '" &
AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                        End Select

                    'personal pronouns
                    Case PRONOUN_PERSONAL:
                        'append person, gender, and number
                        Select Case Word.PrimaryAttributes
                            Case PERSONAL_PRONOUN_1ST_SINGULAR:           Augment = "1_S"
                            Case PERSONAL_PRONOUN_2ND_SINGULAR:           Augment = "2_S"
                            Case PERSONAL_PRONOUN_3RD_SINGULAR_MASCULINE: Augment = "3_M"
                            Case PERSONAL_PRONOUN_3RD_SINGULAR_FEMININE:  Augment = "3_F"
                            Case PERSONAL_PRONOUN_3RD_SINGULAR_NEUTER:    Augment = "3_N"
                            Case PERSONAL_PRONOUN_1ST_PLURAL:             Augment = "1_P"
                            Case PERSONAL_PRONOUN_2ND_PLURAL:             Augment = "2_P"
                            Case PERSONAL_PRONOUN_3RD_PLURAL:             Augment = "3_P"
                            Case Else:
                                Beep
                                MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on personal
pronoun '" & AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                        End Select
                        WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                        WordCase.SymbolicConstant(WordCase.NumSymbolicConstants)   =   gPersPronounConstant(CasePtr)   &
Augment

                    'verbs
                    Case VERB:
                        'decode primary attribute
                        'active (non-auxiliary)
```

```
                    If Word.PrimaryAttributes And VERB_MAIN Then
                       Augment = gActiveVerbConstant(CasePtr)
                       'decode secondary attribute
                       GoSub DecodeSecondaryAttribute
                       '1st person plural plays dual role:  'my' and imperative
                       If CasePtr = VERB_1ST_IMPERATIVE Then
                          Augment = "VB_1_K_"
                          'decode secondary attribute again
                          WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1   '???????
                          GoSub DecodeSecondaryAttribute
                       End If
                    'auxiliary
                    ElseIf Word.PrimaryAttributes And VERB_AUXILIARY Then
                       Augment = gAuxiliaryVerbConstant(CasePtr)
                       'decode secondary attribute
                       GoSub DecodeSecondaryAttribute
                       '1st person plural plays dual role:  'my' and imperative
                       If CasePtr = VERB_1ST_IMPERATIVE Then
                          Augment = "VA_1_K_"
                          'decode secondary attribute again
                          WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1   '??????
                          GoSub DecodeSecondaryAttribute
                       End If
                    Else
                       Beep
                       MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on verb '" &
AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                    End If

                 'verbal adverbs
                 Case VERBAL_ADVERB:
                    Augment = gVerbalAdverbConstant(CasePtr)
                    'decode secondary attribute
                    GoSub DecodeSecondaryAttribute

                 'verbal adjectives
                 Case VERBAL_ADJECTIVE:
                    'decode primary attribute
                    If (Word.PrimaryAttributes And VERBAL_ADJECTIVE_PRESENT_ACTIVE) Then
                       Augment = gVAdjPresActConstant(CasePtr)
                    ElseIf (Word.PrimaryAttributes And VERBAL_ADJECTIVE_PRESENT_PASSIVE) Then
                       Augment = gVAdjPresPasConstant(CasePtr)
                    ElseIf (Word.PrimaryAttributes And VERBAL_ADJECTIVE_PAST_ACTIVE) Then
                       Augment = gVAdjPastActConstant(CasePtr)
                    ElseIf (Word.PrimaryAttributes And VERBAL_ADJECTIVE_PAST_PASSIVE) Then
                       Augment = gVAdjPastPasConstant(CasePtr)
```

```
                    'error in attribute record
                    Else
                       Beep
                       MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on verbal adjective '" &
AbstractWord & "'."), MB_ICONSTOP, ("Lexicon Entry Error")
                    End If

                    'check if true verbal adjective selected (forms expected to be appended to
                    'all terminate with underscore); otherwise...
                    If Right(Augment, 1) = "_" Then
                       'decode secondary attribute
                       GoSub DecodeSecondaryAttribute
                    '...short form adjective derived from verbal adjective selected
                    Else
                       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment
                    End If

                 'interrogatives
                 Case INTERROGATIVE:
                    WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                    WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = gInterrogativeConstant(CasePtr)

                 'relative pronoun
                 Case PRONOUN_RELATIVE:
                    WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
                    WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = gRelPronounConstant(CasePtr)

                 'error in attribute record
                 Case Else:
                    Beep
                    MsgBox ("Illegal primary attribute (" & Word.PrimaryAttributes & ") on '" & AbstractWord &
"'."), MB_ICONSTOP, ("Lexicon Entry Error")
              End Select
           End If

           'move pointers to next field and compiled list slot
           FieldPtr = BreakEnd + 1
           CasePtr = CasePtr + 1
        'fail-safe loop exit for when no case found (ie. error in word or inflection pattern)
        Loop Until FieldPtr > Len(InflectionString)
     End If

     'remove any duplicate symbols
     If WordCase.NumSymbolicConstants Then
        'tag duplicates for removal
```

```
      For i = 1 To WordCase.NumSymbolicConstants
        For j = i + 1 To WordCase.NumSymbolicConstants
            If WordCase.SymbolicConstant(i) = WordCase.SymbolicConstant(j) Then
              WordCase.SymbolicConstant(j) = "*"
            End If
        Next j
      Next i
      'removed tagged entries
      For i = 1 To WordCase.NumSymbolicConstants
        If WordCase.SymbolicConstant(i) = "*" Then
           For j = WordCase.NumSymbolicConstants To i Step -1
             WordCase.SymbolicConstant(j) = WordCase.SymbolicConstant(j + 1)
             WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants - 1
           Next j
        End If
      Next i
    End If

    'return result of symbol generation
    GetSymbolicConstant = WordCase.NumSymbolicConstants

    Exit Function


DecodeSecondaryAttribute:

    'decode secondary attribute
    'none
    If Word.SecondaryAttributes = DEMAND2_NO_DEMANDS Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "0"
    End If

    'nominative
    If Word.SecondaryAttributes And DEMAND2_NOMINATIVE Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "N"
    End If

    'genitive
    If Word.SecondaryAttributes And DEMAND2_GENITIVE Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "G"
    End If

    'dative
```

```
    If Word.SecondaryAttributes And DEMAND2_DATIVE Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "D"
    End If

    'accusative
    If Word.SecondaryAttributes And DEMAND2_ACCUSATIVE Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "A"
    End If

    'instrumental
    If Word.SecondaryAttributes And DEMAND2_INSTRUMENTAL Then
       WordCase.NumSymbolicConstants = WordCase.NumSymbolicConstants + 1
       WordCase.SymbolicConstant(WordCase.NumSymbolicConstants) = Augment & "I"
    End If

    Return

End Function


'_____________________________________________________________________________________________________
'convert hex string to decimal value
'_____________________________________________________________________________________________________
Function HexToDecimal (HexValue As String) As Long

    Dim HValue As String
    Dim DValue As Integer
    Dim Sum    As Long
    Dim i      As Integer


    'if no value given, return zero (null)
    If Len(HexValue) = 0 Then
       HexToDecimal = 0
       Exit Function
    End If

    'convert hex value to uppercase
    HexValue = UCase(HexValue)

    'strip hex indicator if present
    If Left(HexValue, 1) = "$" Then
       HexValue = Mid(HexValue, 2)
    'otherwise don't convert to decimal; just return value since it's already decimal
```

```
    Else
       HexToDecimal = Val(HexValue)
       Exit Function
    End If

    'prepend zeros to make it 4-digit hex
    Do Until Len(HexValue) = 4
       HexValue = "0" & HexValue
    Loop

    'convert each nibble to hex
    Sum = 0
    For i = 1 To 4
      HValue = Mid(HexValue, i, 1)
      'convert A..F to 10..15
      If Not IsNumeric(HValue) Then
         DValue = Asc(HValue) - 55
      'leave 0..9 alone
      Else
         DValue = Val(HValue)
      End If
      'sum values in place to build decimal result
      Sum = Sum + DValue * 2 ^ ((4 - i) * 4)
    Next i

    HexToDecimal = Sum

End Function

'---------------------------------------------------------------------------------------------------------------
'perform insertion sort of new entry into lexicon
'---------------------------------------------------------------------------------------------------------------
Sub InsertLexiconEntry (Entry As LexiconType)

    Dim LexiconPtr As Integer
    Dim InsertPtr  As Integer


    'deal with first insertion into empty lexicon
    If gLexiconEndPtr <= 0 Then
       gLexicon(0) = Entry
       gLexiconEndPtr = 1
       Exit Sub
    End If

    'append trailer node for insertion at end
```

```
    gLexicon(gLexiconEndPtr + 1).Stem = ""

    'search entire lexicon
    For LexiconPtr = 0 To gLexiconEndPtr + 1
       'if insert entry is larger than current lexicon entry...
       If SortLookup(Entry.Stem) > SortLookup(gLexicon(LexiconPtr).Stem) Then
          'increase lexicon length by one element
          gLexiconEndPtr = gLexiconEndPtr + 1

          'make room for insertion
          For InsertPtr = gLexiconEndPtr To LexiconPtr + 1 Step -1
              gLexicon(InsertPtr) = gLexicon(InsertPtr - 1)
          Next InsertPtr

          'insert entry
          gLexicon(LexiconPtr) = Entry

          'done with this entry
          Exit Sub
       End If
    Next LexiconPtr

End Sub

'_____________________________________________________________________________________________________
'load post-translation cleanup filters
'_____________________________________________________________________________________________________
Sub LoadCleanupFilters ()

    Dim Source      As String
    Dim Destination As String
    Dim i             As Integer

    On Error GoTo ErrorOnCleanupRead


    'clear current filters
    frmCleanupFilterEditor.lstCleanupFilter(0).Clear
    frmCleanupFilterEditor.lstCleanupFilter(1).Clear

    'load filters and build resident list
    MousePointerWait
    PassiveHelp "Loading post-translation cleanup filters..."
    Open CLEANUP_FILTER_FILENAME For Input As 1
    Do Until EOF(1)
       Input #1, Source, Destination
```

```
       frmCleanupFilterEditor.lstCleanupFilter(0).AddItem Source
       frmCleanupFilterEditor.lstCleanupFilter(1).AddItem Destination
    Loop
    Close 1
    MousePointerContinue
    PassiveHelp ""

    Exit Sub


ErrorOnCleanupRead:

    'prompt user to deal with store fail
    Beep
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to read the cleanup filter file '" & UCase(CLEANUP_FILTER_FILENAME) & "':" &
Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to store it again; otherwise...
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'don't load filters
    Else
       MsgBox ("No post-translation cleanup can be performed."), MB_ICONINFORMATION, ("Cleanup Filters Missing")
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________
'get system configuration settings
'_____________________________________________________________________________________________________
Sub LoadConfigSettings ()

    Dim StatusBar As Integer

    On Error GoTo ErrorOnConfigLoad:


    'read config file
    MousePointerWait
    Open CONFIG_FILENAME For Input As 1
    Input #1, StatusBar
    Input #1, gShowHelp
    Input #1, gTransferDetails
    Input #1, gTransliterationFilter
    Input #1, gPostEditorAPIName
```

```
    Input #1, gPostEditorFilename
    Input #1, gPostEditorPasteSequence
    Close 1
    MousePointerContinue

    'initialize interface and primary system operation
    frmMain.pnlStatusBar.Visible = StatusBar
    gOperationMode = ABSTRACT_MODE

    Exit Sub


ErrorOnConfigLoad:

    MousePointerContinue
    Beep

    MsgBoxAnswer = MsgBox("Unable to load configuration settings:" & Chr(13) & Error(Err), MB_ICONSTOP +
MB_RETRYCANCEL, "File Error")
    'if user wants to try again, attempt to re-execute operation...
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    '...otherwise abort operation and set defaults
    Else
       'initialize help options
       gShowHelp = True

       'set primary mode to translation
       gOperationMode = ABSTRACT_MODE

       'set default transliteration filter
       gTransliterationFilter = "Transliterate"

       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________
'load bilingual lexicon from file and build internal lexicon structure
'_____________________________________________________________________________________________________
Sub LoadLexicon ()

    Dim UpdateAt     As Single
    Dim LexiconLen   As Long
    Dim LexiconEntry As LexiconType
```

```
    On Error GoTo LoadLexiconError


    'display status info during load
    MousePointerWait
    LexiconLen = FileLen(LEXICON_FILENAME)
    UpdateAt = LexiconLen / 400
    DisplayWait -LexiconLen / UpdateAt, "Loading Russian-English lexicon..."

    'load header
    gHeaderPtr = -1
    Open LEXICON_FILENAME For Input As 1
    Do While Not EOF(1)
       gHeaderPtr = gHeaderPtr + 1
       Line Input #1, gHeader(gHeaderPtr)
       'exit loop when header terminator detected
       If gHeader(gHeaderPtr) = "#." Then
        Exit Do
       End If
    Loop

    'load lexicon entries
    gLexiconEndPtr = 0
    Do While Not EOF(1)
       'get record line
       Input #1, LexiconEntry.Stem

       'check for record sequencing error
       If IsNumeric(LexiconEntry.Stem) Then
        Beep
        MousePointerContinue
        MsgBox ("The bilingual lexicon is corrupted:" & Chr(13) & "Sequencing error detected at line " &
gLexiconEndPtr & "."), MB_ICONSTOP, ("Lexicon Entry Error")
        End
       End If

       'ignore blank lines
       If LexiconEntry.Stem <> "" Then
        Input     #1,     LexiconEntry.PartOfSpeech,     LexiconEntry.Target,     LexiconEntry.InflectionPattern,
LexiconEntry.PrimaryAttributes,          LexiconEntry.SecondaryAttributes,          LexiconEntry.TransferAttributes,
LexiconEntry.Disambiguation

        'insert entry into resident lexicon (uses insertion sort which is acceptable since
        'lexicon file should already be sorted)
        InsertLexiconEntry LexiconEntry
```

```
        'update status by average lexicon line length (not exact, but good enough for status bar)
        If gLexiconEndPtr Mod UpdateAt = 0 Then
           DisplayWait 33, ""
        End If
      End If
    Loop
    Close 1

    'move pointer back to last entry
    gLexiconEndPtr = gLexiconEndPtr - 1

    'index lexicon
    GenerateLexiconIndex

    'hide status
    DisplayWait 0, ""
    MousePointerContinue

    Exit Sub

'_____________________________________________________________________________________________________________
LoadLexiconError:

    'prompt user to deal with missing lexicon
    Beep
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to load the bilingual lexicon '" & UCase(LEXICON_FILENAME) & "':" & Chr(13) &
Error(Err) & ".", MB_ICONEXCLAMATION + MB_ABORTRETRYIGNORE, "File Error")
    'retry attempts to load it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'ignore skips problem (may result other problems later, though)
    ElseIf MsgBoxAnswer = ID_IGNORE Then
       Resume Next
    'otherwise program cannot run
    Else
       End
    End If

End Sub

'_____________________________________________________________________________________________________________
'load transfer rules from file
'_____________________________________________________________________________________________________________
```

```
Sub LoadTransferRules ()

    Dim Component   As String
    Dim TempRulePtr As Integer
    Dim NumRules    As Integer
    Dim NumTerms    As Integer
    Dim RulePtr     As Integer
    Dim TermPtr     As Integer
    Dim Writing     As Integer

    Static Term(MAX_SENTENCE_COMPONENTS) As String

    On Error GoTo ErrorOnTransferRuleRead


    'load transfer rules and build resident list
    MousePointerWait
    Writing = False
    Open TRANSFER_RULE_FILENAME For Input As 1
    Input #1, NumRules
    DisplayWait -NumRules * 1.2, "Loading and expanding transfer rules..."

    gTransferRulePtr = 0
    For RulePtr = 0 To NumRules

       'get russian components of current rule
       Input #1, NumTerms
       For TermPtr = 0 To NumTerms
          Input #1, Component
          Term(TermPtr) = Component
       Next TermPtr

       'expand compressed rule
       TempRulePtr = gTransferRulePtr
       ExpandCompressedTransferRule Term(), (NumTerms), True

       'get english components of current rule
       Input #1, NumTerms
       For TermPtr = 0 To NumTerms
          Input #1, Component
          Term(TermPtr) = Component
       Next TermPtr

       'expand compressed rule
       gTransferRulePtr = TempRulePtr
       ExpandCompressedTransferRule Term(), (NumTerms), False
```

```
      'get comments
      Input #1, Component

      DisplayWait 1, ""
   Next RulePtr
   Close 1

   'set rule pointer to last rule
   gTransferRulePtr = gTransferRulePtr - 1

   'hide status
   DisplayWait 0, ""
   MousePointerContinue

   'write log of all transfer rules generated
   Writing = True
   PassiveHelp "Writing transfer rule log..."
   Open TRANSFER_RULE_LOG_FILENAME For Output As 7
   For RulePtr = 0 To gTransferRulePtr
      'write russian expression
      Print #7, RulePtr; Tab(8);
      For TermPtr = 0 To gTransferRule(RulePtr).NumRussianComponents
          Print #7, gTransferRule(RulePtr).Russian(TermPtr); ", ";
      Next TermPtr
      Print #7, "=> ";
      'write english expression
      For TermPtr = 0 To gTransferRule(RulePtr).NumEnglishComponents
          Print #7, gTransferRule(RulePtr).English(TermPtr); ", ";
      Next TermPtr
      Print #7,
   Next RulePtr
   PassiveHelp ""
   Close 7

   Exit Sub


ErrorOnTransferRuleRead:

   'prompt user to deal with store fail
   Beep
   MousePointerContinue
   PassiveHelp ""
   'error on read
   If Not Writing Then
```

```
      MsgBoxAnswer = MsgBox("Unable to read the transfer rule file '" & UCase(TRANSFER_RULE_FILENAME) & "':" &
Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
      'retry attempts to store it again; otherwise...
      If MsgBoxAnswer = ID_RETRY Then
       Resume
      'don't load rules
      Else
       MsgBox ("No translation can be performed."), MB_ICONINFORMATION, ("Transfer Rules Missing")
       Close 1
       Exit Sub
      End If
   Else
      MsgBox ("Unabled to write transfer rule log file '" & UCase(TRANSFER_RULE_LOG_FILENAME) & "':" & Chr(13) &
Error(Err) & "."), MB_ICONEXCLAMATION, ("File Error")
      Close 7
      Exit Sub
   End If
End Sub

'_____________________________________________________________________________________________________
'look up selected word and return symbolic constants
'_____________________________________________________________________________________________________
Sub LookupWord (SearchWord As String, TransferDetails As TransferType, FullLookup As Integer)

   Dim Symbols      As ConstantType
   Dim MatchType    As Integer
   Dim Stem         As String
   Dim LexiconPtr   As Integer
   Dim SegmentStart As Integer
   Dim SegmentID    As String * 1


   'convert search word to lowercase
   SearchWord = LowerCase(Trim(SearchWord))

   'index into appropriate segment of lexicon
   SegmentID = Left(SearchWord, 1)
   SegmentStart = gLexiconIndex(Asc(SegmentID))
   'if no segment, word cannot be in lexicon, so fail immediately
   If SegmentStart = UNUSED Then
      Exit Sub
   End If

   'loop through segment
   TransferDetails.NumMatches = 0
   For LexiconPtr = SegmentStart To gLexiconEndPtr
```

```
      Stem = LowerCase(gLexicon(LexiconPtr).Stem)
      'if no longer in lexicon letter segment, stop searching
      If Left(Stem, 1) <> SegmentID Then
         Exit For
      End If
      DoEvents
      'determine type of match to make (full or stem)
      Select Case gLexicon(LexiconPtr).PartOfSpeech
             Case PREPOSITION, INTERJECTION, EMPHATIC_PARTICLE, ADVERB, CONJUNCTION:
                MatchType = FULL_MATCH
             Case Else:
                MatchType = STEM_MATCH
      End Select
      'attempt match
      If StemMatch(SearchWord, Stem, MatchType) Then

         '...during full lookup, all info retrieved
         If FullLookup Then
            'if complete entry produces valid noun pattern(s), success
            If GetSymbolicConstant(Stem, SearchWord, gLexicon(LexiconPtr), Symbols) Then
              'record all details for this match
              TransferDetails.NumMatches = TransferDetails.NumMatches + 1
              TransferDetails.Match(TransferDetails.NumMatches).SymbolicConstants = Symbols
              TransferDetails.Match(TransferDetails.NumMatches).LexiconEntry = gLexicon(LexiconPtr)
              TransferDetails.Match(TransferDetails.NumMatches).LexiconEntryPtr = LexiconPtr
            End If

         'during Lexicon Editor lookup, symbolic constant info not retrieved, just lexicon record entry;
otherwise...
         Else
            'record partial details for this match
            TransferDetails.NumMatches = TransferDetails.NumMatches + 1
            TransferDetails.Match(TransferDetails.NumMatches).LexiconEntry = gLexicon(LexiconPtr)
            TransferDetails.Match(TransferDetails.NumMatches).LexiconEntryPtr = LexiconPtr
         End If
      End If
   Next LexiconPtr

End Sub

'_____________________________________________________________________________________________
'convert uppercase cyrillic characters in string to lowercase
'_____________________________________________________________________________________________
Function LowerCase (InString As String) As String

   Dim TempString As String
```

```
    Dim CharCode   As Integer
    Dim i          As Integer


    'don't process empty string
    If InString = "" Then
       LowerCase = ""
       Exit Function
    End If

    'loop through all characters in string
    For i = 1 To Len(InString)
       'extract single character
       CharCode = Asc(Mid(InString, i, 1))
       'special case to convert 'YO' to 'yo'
       If CharCode = 179 Then
          CharCode = 163
       'if uppercase, convert to lower and copy to output string;
       ElseIf CharCode >= 224 Then
          TempString = TempString & Chr(CharCode - 32)
       'otherwise just copy as is
       Else
          TempString = TempString & Chr(CharCode)
       End If
    Next i

    'return converted string
    LowerCase = TempString

End Function

'_____________________________________________________________________________________________________
'enable or disable menu controls dealing with MDI child forms depending on their presence
'_____________________________________________________________________________________________________
Sub MDIChildControls (State As Integer)

    'dis/enable child form controls
    frmMain.menuCascade.Enabled = State
    frmMain.menuTileHorizontally.Enabled = State
    frmMain.menuTileVertically.Enabled = State
    frmMain.menuArrangeIcons.Enabled = State

End Sub

'_____________________________________________________________________________________________________
Sub MousePointerContinue ()
```

```
    'set mouse pointer to normal (ie. arrow)
    Screen.MousePointer = 0

End Sub

'_____________________________________________________________________________________________________________
Sub MousePointerWait ()

    'set mouse pointer to hourglass to indicate wait
    Screen.MousePointer = 11

End Sub

'_____________________________________________________________________________________________________________
Sub OpenTransferLog ()

    On Error GoTo ErrorOnWriteDetails

    frmMain.menuTransferDetails.Enabled = True
    'begin log session
    Open DETAILS_LOG_FILENAME For Output As 5

    'write log header
    Write #5, "Transfer Details Log of '" & gAbstractTitle & "' (" & Date & " " & Time & ")"
    Write #5,
    If Not gTransferDetails Then
       Write #5, "[Transfer Details Logging DISABLED]"
    End If

    'reset translation events
    gLookups = 0
    gLookupHits = 0
    gTransferHits = 0
    gTransferAttempts = 0
    gEventTick = 0
    gEventTime = "0:00"

    'enable event timer
    frmMain.timerTranslation.Enabled = True

    Exit Sub


ErrorOnWriteDetails:
```

```
    'prompt user to deal with write fail
    Beep
    Close 5
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to write to the transfer details log file '" & UCase(DETAILS_LOG_FILENAME) & "':"
& Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to open it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'otherwise abort log, but continue translation
    Else
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________________
'strip leading parenthesis from parameter string if present; otherwise just return parameter string
'leave trailing paranthesis to indicate this is still variable parameter and not literal
'_____________________________________________________________________________________________________________
Function Parameterize (Parameter As String) As String

    'strip
    If Left(Parameter, 1) = "(" Then
       Parameterize = Mid(Parameter, 2)
    'ignore
    Else
       Parameterize = Parameter
    End If

End Function

'_____________________________________________________________________________________________________________
'write to the status line help info for object pointer currently on
'_____________________________________________________________________________________________________________
Sub PassiveHelp (HelpString As String)

    'display if help line enabled
    If gShowHelp Then
       frmMain.pnlInfo.Caption = " " & HelpString
    End If

End Sub

'_____________________________________________________________________________________________________________
```

```
'remove formatting characters from exploded transfer rules
'_____________________________________________________________________________________________
Function RemoveJunk (Term As String) As String

   Dim JunkAt As Integer


   JunkAt = InStr(Term, "[")
   If JunkAt Then
      Mid(Term, JunkAt) = " "
   End If
   JunkAt = InStr(Term, "]")
   If JunkAt Then
      Mid(Term, JunkAt) = " "
   End If
   JunkAt = InStr(Term, "{")
   If JunkAt Then
      Mid(Term, JunkAt) = " "
   End If
   JunkAt = InStr(Term, "}")
   If JunkAt Then
      Mid(Term, JunkAt) = " "
   End If

   RemoveJunk = Trim(Term)

End Function

'_____________________________________________________________________________________________
'compare first letter of cyrillic string against sort lookup table
'_____________________________________________________________________________________________
Function SortLookup (EntryWord As String) As Integer

    'perform lookup
    SortLookup = gSortOrder(Asc(Left(EntryWord & " ", 1) & ""))

End Function

'_____________________________________________________________________________________________
'order transfer rules by description.  Ordering by rule frequency would be more efficient, but at this time
'frequency data have not been determined
'_____________________________________________________________________________________________
Sub SortTransferRules ()

    Dim TempRule As TransferRuleType
```

```
    Dim i        As Integer
    Dim j        As Integer


    MsgBox ("SOMEBODY IS SORTING TRANSFER RULES!")

    Exit Sub

    'shell sort since number of transfer rules relatively small and this need be done only once
    For i = 0 To gTransferRulePtr
       For j = i + 1 To gTransferRulePtr
           'swap entries
           'If gTransferRule(i).Description > gTransferRule(j).Description Then
              TempRule = gTransferRule(i)
              gTransferRule(i) = gTransferRule(j)
              gTransferRule(j) = TempRule
           'End If
       Next j
    Next i

End Sub

'_____________________________________________________________________________________________________
'compare search word with stem.  Takes into account split words like что-нибудь where inflections
'appear between stem and suffix
'_____________________________________________________________________________________________________
Function StemMatch (ByVal SearchWord As String, ByVal Stem As String, Compare As Integer) As Integer

    Static SearchWordDelimiterAt As Integer
    Static StemDelimiterAt       As Integer
    Static SearchWordStem        As String
    Static SearchWordSuffix      As String
    Static StemStem              As String
    Static StemSuffix            As String


    'full match (all chars must match)
    If Compare = FULL_MATCH Then
       StemMatch = (SearchWord = Stem)

    'stem match (only stem must match)
    Else
       'if search word split, do compound match (stem and suffix, ignoring imbedded inflection); otherwise...
       SearchWordDelimiterAt = InStr(SearchWord, "-")
       If SearchWordDelimiterAt Then
         'check that stem splits also
```

```
        StemDelimiterAt = InStr(Stem, "~")
        If StemDelimiterAt Then
           'extract stem stem (yeah, this sounds goofy, but think about it!)
           StemStem = Left(Stem, StemDelimiterAt - 1)
           'extract stem suffix
           StemSuffix = Mid(Stem, StemDelimiterAt + 1)
        'if this fails, return no match
        Else
           StemMatch = False
           Exit Function
        End If

        'extract search word stem
        SearchWordStem = Left(SearchWord, Len(StemStem))
        'extract search word suffix
        SearchWordSuffix = Mid(SearchWord, SearchWordDelimiterAt + 1)

        'compare stems and suffixes
        StemMatch = (SearchWordStem = StemStem) And (SearchWordSuffix = StemSuffix)

      '...try matching just the stems
      Else
       StemMatch = (Left(SearchWord, Len(Stem)) = Stem)
      End If
    End If

End Function

'_____________________________________________________________________________________________________
'write sorted and updated lexicon back to file
'_____________________________________________________________________________________________________
Sub StoreLexicon ()

    Dim EntryPtr As Integer
    Dim UpdateAt As Single

    On Error GoTo StoreLexiconError


    'determine update frequency
    UpdateAt = gLexiconEndPtr / 10

    'display status info during store
    DisplayWait -(gLexiconEndPtr / UpdateAt), "Storing the Russian-to-English lexicon..."

    'rewrite header
```

```
    MousePointerWait
    Open LEXICON_FILENAME For Output As 1
    For EntryPtr = 0 To gHeaderPtr
       'stamp update date and time on second line; otherwise...
       If EntryPtr = 1 Then
          Print #1, "# Last updated: "; Date & " " & Time
       '...write old header info
       Else
          Print #1, gHeader(EntryPtr)
       End If
    Next EntryPtr

    'store lexicon entries
    For EntryPtr = 0 To gLexiconEndPtr
       'if entry not marked for deletion, write it
       If gLexicon(EntryPtr).Stem <> "" Then
          Write    #1,    gLexicon(EntryPtr).Stem,    gLexicon(EntryPtr).PartOfSpeech,    gLexicon(EntryPtr).Target,
gLexicon(EntryPtr).InflectionPattern, gLexicon(EntryPtr).PrimaryAttributes, gLexicon(EntryPtr).SecondaryAttributes,
gLexicon(EntryPtr).TransferAttributes, gLexicon(EntryPtr).Disambiguation
       End If

       'update status periodically
       If EntryPtr Mod UpdateAt = 0 Then
          DisplayWait 1, ""
       End If
    Next EntryPtr
    Close 1

    'hide status
    DisplayWait 0, ""
    MousePointerContinue

    Exit Sub


StoreLexiconError:

    'prompt user to deal with store fail
    Beep
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to store the bilingual lexicon '" & UCase(LEXICON_FILENAME) & "':" & Chr(13) &
Error(Err) & ".", MB_ICONEXCLAMATION + MB_ABORTRETRYIGNORE, "File Error")
    'retry attempts to store it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'ignore skips problem (may result other problems later, though)
```

```
    ElseIf MsgBoxAnswer = ID_IGNORE Then
       Resume Next
    'otherwise program cannot run
    Else
       End
    End If

End Sub

'_____________________________________________________________________________________________________
'collapse multiple spaces to single space and convert tabs to single spaces
'_____________________________________________________________________________________________________
Function StripWhitespace (InString As String) As String

    Dim SpacesAt As Integer
    Dim i        As Integer


    'trim spaces off both ends of string
    InString = Trim(InString)

    'replace tabs with single space
    For i = 1 To Len(InString)
      If Mid(InString, i, 1) = Chr(9) Then
         Mid(InString, i, 1) = " "
      End If
    Next i

    'collapse multiple-space substrings into single space
    Do
       SpacesAt = InStr(InString, "  ")
       'if no such substrings, exit
       If SpacesAt = 0 Then
        Exit Do
       End If
       InString = Left(InString, SpacesAt) & Mid(InString, SpacesAt + 2)
    Loop Until SpacesAt >= Len(InString)

    'return string
    StripWhitespace = InString

End Function

'_____________________________________________________________________________________________________
'generate English from Russian sentence (such a simple explanation for ALL THIS!)
```

```
'_____________________________________________________________________________________________________
Function TransferSentence () As String

    Dim WordPtr               As Integer
    Dim RulePtr               As Integer
    Dim ComponentPtr          As Integer
    Dim Success               As Integer
    Dim i                     As Integer
    Dim ColonAt               As Integer
    Dim ExtractPtr            As Integer
    Dim DelimiterAt           As Integer
    Dim ParameterPtr          As Integer
    Dim MatchPtr              As Integer
    Dim RegisterPtr           As Integer
    Dim TermReferent          As Integer
    Dim TermPreReferent       As Integer
    Dim ReferentBinder        As Integer
    Dim SingularLength        As Integer
    Dim AssociationCode       As Integer
    Dim SCCombinations        As Integer
    Dim VerbPrimaryAttribute  As Integer
    Dim VerbCode              As Integer
    Dim ArticleNone           As Integer
    Dim ArticleThe            As Integer
    Dim ArticleA              As Integer
    Dim ArticleAn             As Integer
    Dim SearchPtr             As Integer
    Dim TermFunction          As String * 1
    Dim Gender                As String * 1
    Dim VerbalAdjectiveType   As String
    Dim VerbalAdjectiveCase   As String
    Dim VerbalAdjectiveGender As String
    Dim Article               As String
    Dim TempTarget            As String
    Dim Noun                  As String
    Dim VerbSymbolicConstant  As String
    Dim VERB                  As String
    Dim VerbPerson            As String
    Dim VerbNumber            As String
    Dim VerbPresent2nd        As String
    Dim VerbPresent3rd        As String
    Dim VerbImperfect         As String
    Dim VerbProgressive       As String
    Dim VerbParticiple        As String
    Dim Sentence              As String
    Dim SecondarySentence     As String
```

```
    Dim Parameter             As String
    Dim Pronoun               As String
    Dim Rule                  As String
    Dim Detail                As String
    Dim TransferComponent     As String
    Dim SentenceComponent     As String
    Dim SymbolicConstantPOS   As String
    Dim TransferRule          As String
    Dim SingularForm          As String

    Dim i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12, i13, i14, i15, i16 As Integer
    Dim i17, i18, i19, i20, i21, i22, i23, i24, i25, i26, i27, i28, i29, i30, i31 As Integer

    Static CombinationPtr(MAX_SENTENCE_COMPONENTS)   As Integer
    Static NumCombinations(MAX_SENTENCE_COMPONENTS)  As Integer
    Static Parameters(MAX_SENTENCE_COMPONENTS)       As String
    Static RegisterStore(MAX_SENTENCE_COMPONENTS)    As String
    Static SymbolicConstant(MAX_SENTENCE_COMPONENTS) As String

    Const DEFINITE_ARTICLE_PROBABILITY = .75

    On Error GoTo TransferError


    MousePointerWait

    'record number of symbolic constants generated for each word calculate number of symbolic constant combinations
    SCCombinations = 1
    DoEvents
    For WordPtr = 0 To MAX_SENTENCE_COMPONENTS
       NumCombinations(WordPtr)                                                                                    =
gSentence(WordPtr).Match(gSentence(WordPtr).MatchPtr).SymbolicConstants.NumSymbolicConstants
       'unused word slots must be processed because of rigid looping constaints
       If NumCombinations(WordPtr) = 0 Then
          NumCombinations(WordPtr) = 1
       End If
       'record combinational complexity
       SCCombinations = SCCombinations * NumCombinations(WordPtr)
    Next WordPtr

    'build contingency sentence in case no translation possible
    SecondarySentence = ""
    For WordPtr = 0 To gWordPtr
       SecondarySentence                          =                          SecondarySentence                          &
gSentence(WordPtr).Match(gSentence(WordPtr).MatchPtr).LexiconEntry.Target & " "
    Next WordPtr
```

```
'log details
Sentence = "  Sentence has " & SCCombinations & " possible symbolic constant interpretation"
If SCCombinations > 1 Then
   Sentence = Sentence & "s"
End If
AddDetail (Sentence & "."), False
AddDetail "", False

'compare all combinations sentence structure against every transfer rule
'(talk about nesting!  This is why max 32-word sentences are supported!)
'All unused word slots add just one iteration each to the loop complexity, so this isn't at all
'as bad as it looks
Do
   For i0 = 1 To NumCombinations(0)
    CombinationPtr(0) = i0
    For i1 = 1 To NumCombinations(1)
      CombinationPtr(1) = i1
      For i2 = 1 To NumCombinations(2)
        CombinationPtr(2) = i2
        For i3 = 1 To NumCombinations(3)
          CombinationPtr(3) = i3
          For i4 = 1 To NumCombinations(4)
           CombinationPtr(4) = i4
           For i5 = 1 To NumCombinations(5)
             CombinationPtr(5) = i5
             For i6 = 1 To NumCombinations(6)
               CombinationPtr(6) = i6
               For i7 = 1 To NumCombinations(7)
                 CombinationPtr(7) = i7
                 For i8 = 1 To NumCombinations(8)
                  CombinationPtr(8) = i8
                  For i9 = 1 To NumCombinations(9)
                    CombinationPtr(9) = i9
                    For i10 = 1 To NumCombinations(10)
                      CombinationPtr(10) = i10
                      For i11 = 1 To NumCombinations(11)
                        CombinationPtr(11) = i11
                        For i12 = 1 To NumCombinations(12)
                         CombinationPtr(12) = i12
                         For i13 = 1 To NumCombinations(13)
                           CombinationPtr(13) = i13
                           For i14 = 1 To NumCombinations(14)
                             CombinationPtr(14) = i14
                             For i15 = 1 To NumCombinations(15)
                               CombinationPtr(15) = i15
```

```
                    For i16 = 1 To NumCombinations(16)
                     CombinationPtr(16) = i16
                     For i17 = 1 To NumCombinations(17)
                       CombinationPtr(17) = i17
                       For i18 = 1 To NumCombinations(18)
                         CombinationPtr(18) = i18
                         For i19 = 1 To NumCombinations(19)
                           CombinationPtr(19) = i19
                           For i20 = 1 To NumCombinations(20)
                            CombinationPtr(20) = i20
                            For i21 = 1 To NumCombinations(21)
                              CombinationPtr(21) = i21
                              For i22 = 1 To NumCombinations(22)
                                CombinationPtr(22) = i22
                                For i23 = 1 To NumCombinations(23)
                                  CombinationPtr(23) = i23
                                  For i24 = 1 To NumCombinations(24)
                                   CombinationPtr(24) = i24
                                   For i25 = 1 To NumCombinations(25)
                                     CombinationPtr(25) = i25
                                     For i26 = 1 To NumCombinations(26)
                                       CombinationPtr(26) = i26
                                       For i27 = 1 To NumCombinations(27)
                                         CombinationPtr(27) = i27
                                         For i28 = 1 To NumCombinations(28)
                                          CombinationPtr(28) = i28
                                          For i29 = 1 To NumCombinations(29)
                                            CombinationPtr(29) = i29
                                            For i30 = 1 To NumCombinations(30)
                                              CombinationPtr(30) = i30
                                              For i31 = 1 To NumCombinations(31)
                                                CombinationPtr(31) = i31
'escape key aborts translation
If gAbort Then
   TransferSentence = "[TRANSLATION ABORTED]"
   Exit Function
End If

'compare each symbolic sentence representation against each transfer rule
DoEvents
For RulePtr = 0 To gTransferRulePtr
    'clear register store
    For RegisterPtr = 0 To MAX_SENTENCE_COMPONENTS
      RegisterStore(RegisterPtr) = ""
    Next RegisterPtr
```

```
            'compare each component for part of speech matches
            Rule = ""
            Success = True
            For ComponentPtr = 0 To gWordPtr
               MatchPtr = gSentence(ComponentPtr).MatchPtr
               'if part of speech component of transfer rule match fails, skip rest of this rule
               SymbolicConstantPOS                                                                    =
Left(gSentence(ComponentPtr).Match(MatchPtr).SymbolicConstants.SymbolicConstant(CombinationPtr(ComponentPtr)), 2)
               TransferRule = gTransferRule(RulePtr).Russian(ComponentPtr)
               If InStr(TransferRule, SymbolicConstantPOS) = 0 Then
                  Success = False
                  Exit For
               End If
               'record rule building
               Rule = Rule & gTransferRule(RulePtr).Russian(ComponentPtr) & " "
            Next ComponentPtr

            'initial matching successful; now do in-depth matching
            If Success Then
              'clear variable slots
              For i = 0 To MAX_SENTENCE_COMPONENTS
                  Parameters(i) = ""
              Next i

              Success = True
              'try to match all components
              For ComponentPtr = 0 To gTransferRule(RulePtr).NumRussianComponents
                  TransferComponent = gTransferRule(RulePtr).Russian(ComponentPtr)
                  ColonAt = InStr(TransferComponent, ":")
                  'strip parameter binder
                  If ColonAt Then
                    TransferComponent = Mid(TransferComponent, ColonAt + 1)
                  'if none present, indicate error
                  Else
                    Beep
                    MsgBox ("Syntax error in transfer rule" & Chr(13) & "'" & Rule & "'."), MB_ICONEXCLAMATION,
("Translation Failure")
                  End If
                  SentenceComponent                                                                    =
gSentence(ComponentPtr).Match(gSentence(ComponentPtr).MatchPtr).SymbolicConstants.SymbolicConstant(CombinationPtr(Co
mponentPtr))

                  'extract sentence component slices
                  ExtractPtr = 1
                  ParameterPtr = -1
                  Do
```

```
  DelimiterAt = InStr(ExtractPtr, SentenceComponent & "_", "_")
  'process slice; otherwise...
  If DelimiterAt Then
     ParameterPtr = ParameterPtr + 1
     Parameters(ParameterPtr) = Mid(SentenceComponent, ExtractPtr, DelimiterAt - ExtractPtr)
     ExtractPtr = DelimiterAt + 1
  '...all done with this component
  Else
     Exit Do
  End If
Loop

'extract transfer component slices
ExtractPtr = 1
ParameterPtr = -1
Do
  DelimiterAt = InStr(ExtractPtr, TransferComponent & "_", "_")
  'process slice; otherwise...
  If DelimiterAt Then
     ParameterPtr = ParameterPtr + 1
     Parameter = Parameterize(Mid(TransferComponent, ExtractPtr, DelimiterAt - ExtractPtr))

     'if parameter is variable, bind it to sentence component slice; otherwise...
     If Right(Parameter, 1) = ")" Then
        'if variable not in use, assign it to its slot; otherwise...
        If RegisterStore(Val(Parameter)) = "" Then
          RegisterStore(Val(Parameter)) = Parameters(ParameterPtr)
        '...if slot not already assigned this variable, flag mismatch
        ElseIf RegisterStore(Val(Parameter)) <> Parameters(ParameterPtr) Then
          Success = False
          Exit Do
        End If
     '...compare parameter as literal; otherwise parameter is wildcard, so don't compare at all
     ElseIf Parameter <> "*" Then
        'if literals don't match, flag mismatch
        If Parameter <> Parameters(ParameterPtr) Then
          Success = False
          Exit Do
        End If
     End If
     ExtractPtr = DelimiterAt + 1
  '...all done with this component
  Else
     Exit Do
  End If
Loop
```

```
                'stop matching if anything fails
                If Not Success Then
                  Exit For
                End If
            Next ComponentPtr

            'if still success, rule has matched
            If Success Then

               'log details
               AddDetail ("  Sentence transfer:"), False
               AddDetail "", False

               'detail symbolic constants chosen
               Detail = ""

               'list chosen symbolic constants
               For ComponentPtr = 0 To gTransferRule(RulePtr).NumRussianComponents
                 Detail                              =                              Detail                                &
gSentence(ComponentPtr).Match(gSentence(ComponentPtr).MatchPtr).SymbolicConstants.SymbolicConstant(CombinationPtr(Co
mponentPtr)) & "  "
                 SymbolicConstant(ComponentPtr)                                                                            =
gSentence(ComponentPtr).Match(gSentence(ComponentPtr).MatchPtr).SymbolicConstants.SymbolicConstant(CombinationPtr(Co
mponentPtr))
               Next ComponentPtr

               If gTransferDetails Then
                  AddDetail ("    Symbolic Constants: "), True
                  AddDetail ("      " & Detail), False
                  AddDetail "", False

                  'detail russian side string
                  AddDetail ("    Transfer Rule #" & RulePtr & " matched:"), False
                  AddDetail ("      Russian= " & Rule), False
                  Rule = ""

                  'build english side string
                  For ComponentPtr = 0 To gTransferRule(RulePtr).NumEnglishComponents
                    Rule = Rule & gTransferRule(RulePtr).English(ComponentPtr) & "  "
                  Next ComponentPtr
                  Rule = Trim(Rule)
                  AddDetail ("      English= " & Rule), False
               End If

               'exit alls loops gracefully
```

```
                Exit Do
              End If
            End If
         Next RulePtr

      Next i31: Next i30: Next i29: Next i28: Next i27: Next i26: Next i25: Next i24
      Next i23: Next i22: Next i21: Next i20: Next i19: Next i18: Next i17: Next i16
      Next i15: Next i14: Next i13: Next i12: Next i11: Next i10: Next i9:  Next i8
      Next i7:  Next i6:  Next i5:  Next i4:  Next i3:  Next i2:  Next i1:  Next i0

   'provide a properly structured looping arrangement that doesn't corrupt the stack when exited
   Loop Until True

   'no applicable transfers rules found; this sentence cannot be translated
   gTransferAttempts = gTransferAttempts + 1
   If Not Success Then
      AddDetail ("  ** No applicable transfer rule found -- This sentence was translated literally **"), True
      'end untranslated sentence with period
      TransferSentence = "[" & Left(SecondarySentence, Len(SecondarySentence) - 1) & ".]"
      Exit Function

   'continue transfer
   Else

      'process each term of english side
      Sentence = ""
      For ComponentPtr = 0 To gTransferRule(RulePtr).NumEnglishComponents

         'decode current english side term for russian term referent and function
         TermPreReferent = Val(gTransferRule(RulePtr).English(ComponentPtr))
         TermFunction = Right(gTransferRule(RulePtr).English(ComponentPtr), 1)

         'bind term pre-referent to actual term
         TermReferent = -1
         For ReferentBinder = 0 To gTransferRule(RulePtr).NumEnglishComponents
             If TermPreReferent = Val(gTransferRule(RulePtr).Russian(ReferentBinder)) Then
               TermReferent = ReferentBinder
               Exit For
             End If
         Next ReferentBinder
         'flag error if not bind made
         If TermReferent = -1 Then
            Beep
            MsgBox ("Unresolved parameter reference in English expression of rule" & Chr(13) & "'" & Rule & "'."),
MB_ICONEXCLAMATION, ("Translation Failure")
            TermReferent = 0
```

```
        End If

        'for nouns and articles, decode transfer associations
        If (TermFunction = "A") Or (TermFunction = "N") Then
           'get gender or plural
           Gender                                                                                    =
Mid(gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).SymbolicConstants.SymbolicConstant(CombinationPt
r(TermReferent)), 6, 1)

           'get articles
           AssociationCode                                                                           =
gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.TransferAttributes

           'set target article 'none'
           ArticleNone = False
           If (gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.TransferAttributes And
ARTICLE_NONE) Then
            'decrement extracted code from transfer code
            AssociationCode = AssociationCode - ARTICLE_NONE
            ArticleNone = True
           End If

           'plurals are always optional
           If (Gender = "P") Then
            ArticleNone = True
           End If

           'set target article 'the'
           If (gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.TransferAttributes And
ARTICLE_THE) Then
            'decrement extracted code from transfer code
            AssociationCode = AssociationCode - ARTICLE_THE
            ArticleThe = True
           Else
            ArticleThe = False
           End If

           'set target article 'a'
           ArticleA = False
           ArticleAn = False
           If (gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.TransferAttributes And
ARTICLE_A) And (Gender <> "P") Then
            'decrement extracted code from transfer code
            AssociationCode = AssociationCode - ARTICLE_A
            ArticleA = True
           'set target article 'an'
```

```
            ElseIf  (gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.TransferAttributes
And ARTICLE_AN) And (Gender <> "P") Then
             'decrement extracted code from transfer code
             AssociationCode = AssociationCode - ARTICLE_AN
             ArticleAn = True
            End If

            'prepare to generate plural form of noun
            SingularForm = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
            SingularLength = Len(SingularForm)

            'decode plural by subtraction since bitwise operations not applicable; otherwise...
            If Gender = "P" Then
             If (AssociationCode - PLURAL11) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 4) & "eese"
             ElseIf (AssociationCode - PLURAL10) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 4) & "ice"
             ElseIf (AssociationCode - PLURAL9) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 2) & "en"
             ElseIf (AssociationCode - PLURAL8) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 2) & "i"
             ElseIf (AssociationCode - PLURAL7) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 2) & "a"
             ElseIf (AssociationCode - PLURAL6) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 2) & "ves"
             ElseIf (AssociationCode - PLURAL5) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 1) & "ves"
             ElseIf (AssociationCode - PLURAL4) >= 0 Then
                Noun = Left(SingularForm, SingularLength - 1) & "ies"
             ElseIf (AssociationCode - PLURAL3) >= 0 Then
                Noun = SingularForm & "es"
             ElseIf (AssociationCode - PLURAL2) >= 0 Then
                Noun = SingularForm & "s"
             ElseIf (AssociationCode - PLURAL1) >= 0 Then
                Noun = SingularForm
             End If

            '...noun is singular
            Else
             Noun = SingularForm
            End If

            'set articles
            Article = ""
            Randomize
            'randomly pick from the/a when both present
```

```
            If (ArticleThe) And ((ArticleA) Or (ArticleAn)) Then
              If (Rnd > 1 - DEFINITE_ARTICLE_PROBABILITY) Then
                 Article = Article & "the"
              Else
                 If ArticleA Then
                    Article = Article & "a"
                 Else
                    Article = Article & "an"
                 End If
              End If
            ElseIf ArticleThe Then
              Article = Article & "the"
            End If

            'set indefinite article
            If (ArticleA) And (Article = "") Then
              Article = Article & "a"
            ElseIf (ArticleAn) And (Article = "") Then
              Article = Article & "an"
            End If
            'set none
            If (ArticleNone) And (Article <> "") Then
              'Article = "[" & Article & "]"
            End If

         'for verbs, decode forms and generate correct english
         ElseIf (TermFunction = "V") Or (TermFunction = "D") Or (TermFunction = "J") Then
            VERB = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target

            'extract verbal information
            VerbSymbolicConstant = SymbolicConstant(TermReferent)
            VerbPrimaryAttribute                                                                                =
gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.PrimaryAttributes

            'extract verb forms:
            'passive reflexive
            If ((VerbPrimaryAttribute And VERB_REFLEXIVE_PASSIVE) And TermFunction = "V") Or ((VerbPrimaryAttribute
And VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE) And TermFunction = "J") Then
              VerbParticiple = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target

            'imperfective or indeterminate verb
            ElseIf ((VerbPrimaryAttribute And VERB_IMPERFECTIVE) Or (VerbPrimaryAttribute And VERB_INDETERMINATE))
And (TermFunction <> "D") Or (((VerbPrimaryAttribute And VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE) = 0) And (TermFunction
= "J")) Then
              'extract field 1
              TempTarget = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
```

```
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbPresent2nd = Left(TempTarget, DelimiterAt - 1)

  'extract field 2
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbPresent3rd = Left(TempTarget, DelimiterAt - 1)

  'extract field 3
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbImperfect = Left(TempTarget, DelimiterAt - 1)

  'extract field 4
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbProgressive = Left(TempTarget, DelimiterAt - 1)

  'extract field 5
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  VerbParticiple = TempTarget

'perfective verb
ElseIf (VerbPrimaryAttribute And VERB_PERFECTIVE) And (TermFunction <> "D") Then
  'extract field 1
  TempTarget = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbPresent2nd = Left(TempTarget, DelimiterAt - 1)

  'extract field 2
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbImperfect = Left(TempTarget, DelimiterAt - 1)

  'extract field 3
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  VerbParticiple = TempTarget
Else
  'extract field 1
  TempTarget = gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
  DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
  VerbProgressive = Left(TempTarget, DelimiterAt - 1)

  'extract field 2
  TempTarget = Mid(TempTarget, DelimiterAt + 1)
  VerbParticiple = TempTarget
```

```
            End If

            'get person/tense
            VerbPerson = Mid(VerbSymbolicConstant, 4, 1)

            'get number
            VerbNumber = Mid(VerbSymbolicConstant, 6, 1)

            'get verbal adjective attributes
            VerbalAdjectiveType = Mid(VerbSymbolicConstant, 4, 2)
            VerbalAdjectiveCase = Mid(VerbSymbolicConstant, 7, 1)
            VerbalAdjectiveGender = Mid(VerbSymbolicConstant, 9, 1)

            'verb processing, stage 1:  generate verb codes:
            Randomize
            If (TermFunction = "V") Then

              '----- verb, infinitive
              If (VerbPerson = "I") Then
                 VerbCode = 14

              '----- verb, non-reflexive, present, imperfective
              ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_IMPERFECTIVE)) = (VERB_NONREFLEXIVE +
VERB_IMPERFECTIVE)) And (VerbPerson <> "P") Then
                 'randomly choose between type A, B, and C
                 If Rnd < .2 Then
                    VerbCode = 1
                 ElseIf Rnd < .2 Then
                    VerbCode = 2
                 Else
                    VerbCode = 3
                 End If

              '----- verb, non-reflexive, present, indeterminate
              ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_INDETERMINATE)) = (VERB_NONREFLEXIVE +
VERB_INDETERMINATE)) And (VerbPerson <> "P") Then
                 VerbCode = 3

              '----- verb, non-reflexive, present, determinate
              ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_DETERMINATE)) = (VERB_NONREFLEXIVE +
VERB_DETERMINATE)) And (VerbPerson <> "P") Then
                 VerbCode = 1

              '----- verb, non-reflexive, past, imperfective
              ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_IMPERFECTIVE)) = (VERB_NONREFLEXIVE +
VERB_IMPERFECTIVE)) And (VerbPerson = "P") Then
```

```
                VerbCode = 4

            '----- verb, non-reflexive, past, indeterminate
            ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_INDETERMINATE)) = (VERB_NONREFLEXIVE +
VERB_INDETERMINATE)) And (VerbPerson = "P") Then
                VerbCode = 5

            '----- verb, non-reflexive, past, determinate
            ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_DETERMINATE)) = (VERB_NONREFLEXIVE +
VERB_DETERMINATE)) And (VerbPerson = "P") Then
                VerbCode = 4

            '----- verb, non-reflexive, past, perfective
            ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_PERFECTIVE)) = (VERB_NONREFLEXIVE +
VERB_PERFECTIVE)) And (VerbPerson = "P") Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 5
                Else
                   VerbCode = 6
                End If

            '----- verb, non-reflexive, future, imperfective or indeterminate
            '      unsupported

            '----- verb, non-reflexive, future, determinate
            '      unsupported

            '----- verb, non-reflexive, future, perfective
            ElseIf ((VerbPrimaryAttribute And (VERB_NONREFLEXIVE + VERB_PERFECTIVE)) = (VERB_NONREFLEXIVE +
VERB_PERFECTIVE)) And (VerbPerson <> "P") Then
                VerbCode = 8

            '----- verb, active reflexive, present, imperfective
            ElseIf    ((VerbPrimaryAttribute    And    (VERB_REFLEXIVE_ACTIVE    +    VERB_IMPERFECTIVE))    =
(VERB_REFLEXIVE_ACTIVE + VERB_IMPERFECTIVE)) And (VerbPerson <> "P") Then
                'randomly choose between type A, B, and C
                If Rnd < .2 Then
                   VerbCode = 1
                ElseIf Rnd < .2 Then
                   VerbCode = 2
                Else
                   VerbCode = 3
                End If

            '----- verb, active reflexive, past, imperfective
```

```
            ElseIf     ((VerbPrimaryAttribute     And     (VERB_REFLEXIVE_ACTIVE     +     VERB_IMPERFECTIVE))     =
(VERB_REFLEXIVE_ACTIVE + VERB_IMPERFECTIVE)) And (VerbPerson = "P") Then
               VerbCode = 4

            '----- verb, active reflexive, past, perfective
            ElseIf ((VerbPrimaryAttribute And (VERB_REFLEXIVE_ACTIVE + VERB_PERFECTIVE)) = (VERB_REFLEXIVE_ACTIVE
+ VERB_PERFECTIVE)) And (VerbPerson = "P") Then
               'choose between type A and B
               If Rnd < .5 Then
                  VerbCode = 5
               Else
                  VerbCode = 6
               End If

            '----- verb, active reflexive, future, imperfective
            '      unsupported

            '----- verb, active reflexive, future, perfective
            ElseIf ((VerbPrimaryAttribute And (VERB_REFLEXIVE_ACTIVE + VERB_PERFECTIVE)) = (VERB_REFLEXIVE_ACTIVE
+ VERB_PERFECTIVE)) And (VerbPerson <> "P") Then
               VerbCode = 8

            '----- verb, passive reflexive, present, imperfective
            ElseIf     ((VerbPrimaryAttribute     And     (VERB_REFLEXIVE_PASSIVE     +     VERB_IMPERFECTIVE))     =
(VERB_REFLEXIVE_PASSIVE + VERB_IMPERFECTIVE)) And (VerbPerson <> "P") Then
               'choose between type A and B
               If Rnd < .75 Then
                  VerbCode = 10
               Else
                  VerbCode = 11
               End If

            '----- verb, passive reflexive, present, determinate
            ElseIf     ((VerbPrimaryAttribute     And     (VERB_REFLEXIVE_PASSIVE     +     VERB_DETERMINATE))     =
(VERB_REFLEXIVE_PASSIVE + VERB_DETERMINATE)) And (VerbPerson <> "P") Then
               'choose between type A and B
               If Rnd < .75 Then
                  VerbCode = 10
               Else
                  VerbCode = 11
               End If

            '----- verb, passive reflexive, past, imperfective
            ElseIf     ((VerbPrimaryAttribute     And     (VERB_REFLEXIVE_PASSIVE     +     VERB_IMPERFECTIVE))     =
(VERB_REFLEXIVE_PASSIVE + VERB_IMPERFECTIVE)) And (VerbPerson = "P") Then
               'choose between type A and B
```

```
                If Rnd < .5 Then
                   VerbCode = 15
                Else
                   VerbCode = 12
                End If

             '----- verb, passive reflexive, past, determinate
             ElseIf    ((VerbPrimaryAttribute    And    (VERB_REFLEXIVE_PASSIVE    +    VERB_DETERMINATE))    =
(VERB_REFLEXIVE_PASSIVE + VERB_DETERMINATE)) And (VerbPerson = "P") Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 10
                Else
                   VerbCode = 12
                End If

             '----- verb, passive reflexive, past, perfective
             ElseIf    ((VerbPrimaryAttribute    And    (VERB_REFLEXIVE_PASSIVE    +    VERB_PERFECTIVE))    =
(VERB_REFLEXIVE_PASSIVE + VERB_PERFECTIVE)) And (VerbPerson = "P") Then
                VerbCode = 12

             '----- verb, passive reflexive, future, perfective
             ElseIf    ((VerbPrimaryAttribute    And    (VERB_REFLEXIVE_PASSIVE    +    VERB_PERFECTIVE))    =
(VERB_REFLEXIVE_PASSIVE + VERB_PERFECTIVE)) And (VerbPerson <> "P") Then
                VerbCode = 13
             End If

          'verbal adverbs
          ElseIf (TermFunction = "D") Then

             '----- verbal adverb, non-reflexive, simultaneous
             If (VerbPrimaryAttribute And (VERBAL_ADVERB_NON_REFLEXIVE)) And (VerbPerson = "S") Then
                VerbCode = 20

             '----- verbal adverb, non-reflexive, sequential, imperfective
             ElseIf  ((VerbPrimaryAttribute  And  (VERBAL_ADVERB_NON_REFLEXIVE  +  VERBAL_ADVERB_IMPERFECTIVE))  =
(VERBAL_ADVERB_NON_REFLEXIVE + VERBAL_ADVERB_IMPERFECTIVE)) And (VerbPerson = "Q") Then
                VerbCode = 21

             '----- verbal adverb, non-reflexive, sequential, perfective
             ElseIf   ((VerbPrimaryAttribute   And   (VERBAL_ADVERB_NON_REFLEXIVE   +   VERBAL_ADVERB_PERFECTIVE))   =
(VERBAL_ADVERB_NON_REFLEXIVE + VERBAL_ADVERB_PERFECTIVE)) And (VerbPerson = "Q") Then
                VerbCode = 22

             '----- verbal adverb, active reflexive, simultaneous, imperfective
             ElseIf (VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_ACTIVE)) And (VerbPerson = "S") Then
```

```
               VerbCode = 20

            '----- verbal adverb, active reflexive, sequential, imperfective
            ElseIf ((VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_ACTIVE + VERBAL_ADVERB_IMPERFECTIVE)) =
(VERBAL_ADVERB_REFLEXIVE_ACTIVE + VERBAL_ADVERB_IMPERFECTIVE)) And (VerbPerson = "Q") Then
               VerbCode = 21

            '----- verbal adverb, non-reflexive, sequential, imperfective
            ElseIf ((VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_ACTIVE + VERBAL_ADVERB_PERFECTIVE)) =
(VERBAL_ADVERB_REFLEXIVE_ACTIVE + VERBAL_ADVERB_PERFECTIVE)) And (VerbPerson = "Q") Then
               VerbCode = 22

            '----- verbal adverb, passive reflexive, simultaneous, imperfective
            ElseIf ((VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_IMPERFECTIVE)) =
(VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_IMPERFECTIVE)) And (VerbPerson = "S") Then
               VerbCode = 23

            '----- verbal adverb, passive reflexive, sequential, imperfective
            ElseIf ((VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_IMPERFECTIVE)) =
(VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_IMPERFECTIVE)) And (VerbPerson = "Q") Then
               VerbCode = 24

            '----- verbal adverb, passive reflexive, sequential, perfective
            ElseIf ((VerbPrimaryAttribute And (VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_PERFECTIVE)) =
(VERBAL_ADVERB_REFLEXIVE_PASSIVE + VERBAL_ADVERB_PERFECTIVE)) And (VerbPerson = "Q") Then
               VerbCode = 24
            End If

          'verbal adjectives
          ElseIf (TermFunction = "J") Then
            '----- verbal adjective, non-reflexive, present active, imperfective
            If (VerbalAdjectiveType = "PA") And (VerbPrimaryAttribute And (VERBAL_ADJECTIVE_NON_REFLEXIVE)) Then
               'choose between type A, B, C, and D
               If Rnd < .25 Then
                  VerbCode = 30
               ElseIf Rnd < .25 Then
                  VerbCode = 31
               ElseIf Rnd < .25 Then
                  VerbCode = 32
               Else
                  VerbCode = 33
               End If

            '----- verbal adjective, non-reflexive, past active, imperfective
            ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_NON_REFLEXIVE +
VERBAL_ADJECTIVE_IMPERFECTIVE)) = (VERBAL_ADJECTIVE_NON_REFLEXIVE + VERBAL_ADJECTIVE_IMPERFECTIVE)) Then
```

```
                VerbCode = 34

            '----- verbal adjective, non-reflexive, past active, perfective
            ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_NON_REFLEXIVE +
VERBAL_ADJECTIVE_PERFECTIVE)) = (VERBAL_ADJECTIVE_NON_REFLEXIVE + VERBAL_ADJECTIVE_PERFECTIVE)) Then
                'choose between type A, B, C, and D
                If Rnd < .5 Then
                   VerbCode = 35
                Else
                   VerbCode = 36
                End If

            '----- verbal adjective, active reflexive, present active, imperfective
            ElseIf (VerbalAdjectiveType = "PA") And (VerbPrimaryAttribute And (VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE))
Then
                'choose between type A, B, C, and D
                If Rnd < .25 Then
                   VerbCode = 30
                ElseIf Rnd < .25 Then
                   VerbCode = 31
                ElseIf Rnd < .25 Then
                   VerbCode = 32
                Else
                   VerbCode = 33
                End If

            '----- verbal adjective, active reflexive, past active, imperfective
            ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE
+ VERBAL_ADJECTIVE_IMPERFECTIVE)) = (VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE + VERBAL_ADJECTIVE_IMPERFECTIVE)) Then
                VerbCode = 34

            '----- verbal adjective, active reflexive, past active, perfective
            ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE
+ VERBAL_ADJECTIVE_PERFECTIVE)) = (VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE + VERBAL_ADJECTIVE_PERFECTIVE)) Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 35
                Else
                   VerbCode = 36
                End If

            '----- verbal adjective, passive, present passive, imperfective
            ElseIf (VerbalAdjectiveType = "PP") And (VerbPrimaryAttribute And VERBAL_ADJECTIVE_IMPERFECTIVE) Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 37
```

```
                Else
                   VerbCode = 38
                End If

             '----- verbal adjective, passive, past passive, imperfective
             ElseIf (VerbalAdjectiveType = "AP") And (VerbPrimaryAttribute And VERBAL_ADJECTIVE_IMPERFECTIVE) Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 37
                Else
                   VerbCode = 38
                End If

             '----- verbal adjective, passive, past passive, perfective
             ElseIf (VerbalAdjectiveType = "AP") And (VerbPrimaryAttribute And VERBAL_ADJECTIVE_PERFECTIVE) Then
                'choose between type A and B
                If Rnd < .5 Then
                   VerbCode = 40
                Else
                   VerbCode = 41
                End If

             '----- verbal adjective, passive reflexive, present active, imperfective
             ElseIf       (VerbalAdjectiveType       =       "PA")       And       (VerbPrimaryAttribute       And
(VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE)) Then
                VerbCode = 38

             '----- verbal adjective, passive reflexive, past active, imperfective
             ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE
+ VERBAL_ADJECTIVE_IMPERFECTIVE)) = (VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE + VERBAL_ADJECTIVE_IMPERFECTIVE)) Then
                VerbCode = 39

             '----- verbal adjective, passive reflexive, past active, perfective
             ElseIf (VerbalAdjectiveType = "AA") And ((VerbPrimaryAttribute And (VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE
+ VERBAL_ADJECTIVE_PERFECTIVE)) = (VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE + VERBAL_ADJECTIVE_PERFECTIVE)) Then
                VerbCode = 40
             End If
          End If

          'verb processing, stage 2:  generate verb forms:
          Select Case VerbCode
               'verbs
               Case 1:  '{'am'|'is'|'are'} + Progressive
                      If (VerbPerson = "1") And (VerbNumber = "S") Then
                        VERB = "am " & VerbProgressive
                      ElseIf (VerbPerson = "3") And (VerbNumber = "S") Then
```

```
          VERB = "is " & VerbProgressive
        Else
          VERB = "are " & VerbProgressive
        End If

Case 2:  '{'has'|'have'} + 'been' + Progressive
        If (VerbPerson = "3") And (VerbNumber = "S") Then
          VERB = "has been " & VerbProgressive
        Else
          VERB = "have been " & VerbProgressive
        End If

Case 3:  '{Present Indicative 2nd|Present Indicative 3rd}
        If (VerbPerson = "3") And (VerbNumber = "S") Then
          VERB = VerbPresent3rd
        Else
          VERB = VerbPresent2nd
        End If

Case 4:  '{'was'|'were'} + Progressive
        If (VerbNumber = "P") Then
          VERB = "were " & VerbProgressive
        Else
          VERB = "was " & VerbProgressive
        End If

Case 5:  'imperfect indicative
        VERB = VerbImperfect

Case 6:  '{'has'|'have'} + Past Participle
        If (VerbNumber = "P") Then
          VERB = "have " & VerbParticiple
        Else
          VERB = "has " & VerbParticiple
        End If

Case 7:  'will be' + Progressive
        VERB = "will be " & VerbProgressive

Case 8:  'will' + Present Indicative 2nd
        VERB = "will " & VerbPresent2nd

Case 10: '{'am'|'is'|'are'} + Past Participle
        If (VerbPerson = "1") And (VerbNumber = "S") Then
          VERB = "am " & VerbParticiple
        ElseIf (VerbPerson = "3") And (VerbNumber = "S") Then
```

```
          VERB = "is " & VerbParticiple
        Else
          VERB = "are " & VerbParticiple
        End If

Case 11: '{'am'|'is'|'are'} + 'being' + Past Participle
        If (VerbPerson = "1") And (VerbNumber = "S") Then
         VERB = "am being " & VerbParticiple
        ElseIf (VerbPerson = "3") And (VerbNumber = "S") Then
          VERB = "is being " & VerbParticiple
        Else
          VERB = "are being " & VerbParticiple
        End If

Case 12: '{'was'|'were'} + 'being' + Past Participle
        If (VerbNumber = "P") Then
          VERB = "were being " & VerbParticiple
        Else
          VERB = "was being " & VerbParticiple
        End If

Case 13: 'will be' + Past Participle
        VERB = "will be " & VerbParticiple

Case 14: VERB = "to " & VerbPresent2nd

Case 15: '{'was'|'were'} + Past Participle
        If (VerbNumber = "P") Then
          VERB = "were " & VerbParticiple
        Else
          VERB = "was " & VerbParticiple
        End If

'verbal adverbs
Case 20: 'while' + Progressive
        VERB = "while " & VerbProgressive

Case 21: 'having been' + Progressive
        VERB = "having been " & VerbProgressive

Case 22: 'having' + Past Participle
        VERB = "having " & VerbParticiple

Case 23: 'while being' + Past Participle
        VERB = "while being " & VerbParticiple
```

```
Case 24: 'having been' + Past Participle
       VERB = "having been " & VerbParticiple

'verbal adjectives
Case 30: 'Progressive
       VERB = VerbProgressive

Case 31: '{'which'|'who'|'whom'} + {'is|are'} + Progressive
       If (VerbalAdjectiveGender <> "P") Then
         VERB = "is " & VerbProgressive
       Else
         VERB = "are " & VerbProgressive
       End If

Case 32: '{'which'|'who'|'whom'} + {'has'|'have'} + 'been' + Progressive
       If (VerbalAdjectiveGender <> "P") Then
         VERB = "has been " & VerbProgressive
       Else
         VERB = "have been " & VerbProgressive
       End If

Case 33: '{'which'|'who'|'whom'} + {Present Indicative 2nd|Present Indicative 3rd}
       If (VerbalAdjectiveGender <> "P") Then
         VERB = VerbPresent3rd
       Else
         VERB = VerbPresent2nd
       End If

Case 34: '{'which'|'who'|'whom'} + {'was'|'were'} + Progressive
       If (VerbalAdjectiveGender = "P") Then
         VERB = "were " & VerbProgressive
       Else
         VERB = "was " & VerbProgressive
       End If

Case 35: '{'which'|'who'|'whom'} + Imperfect Indicative
       VERB = VerbImperfect

Case 36: '{'which'|'who'|'whom'} + {'has'|'have'} + Past Participle
       If (VerbalAdjectiveGender = "P") Then
         VERB = "have " & VerbParticiple
       Else
         VERB = "has " & VerbParticiple
       End If

Case 37: 'being' + Past Participle
```

```
            VERB = "being " & VerbParticiple

        Case 38: '{'which'|'who'} + {'is|are'} + 'being' + Past Participle
            If (VerbalAdjectiveGender = "P") Then
              VERB = "are being " & VerbParticiple
            Else
              VERB = "is being " & VerbParticiple
            End If

        Case 39: '{'which'|'who'} + {'was'|'were'} + 'being' + Past Participle
            If (VerbalAdjectiveGender = "P") Then
              VERB = "were being " & VerbParticiple
            Else
              VERB = "was being " & VerbParticiple
            End If

        Case 40: '{'which'|'who'} + {'was'|'were'} + Past Participle
            If (VerbalAdjectiveGender = "P") Then
              VERB = "were " & VerbParticiple
            Else
              VERB = "was " & VerbParticiple
            End If

        Case 41: 'Past Participle
            VERB = VerbParticiple

        'unknown verbal construction
        Case Else
           Beep
           MsgBox ("Illegal verb code '" & VerbCode & "'."), MB_ICONEXCLAMATION, ("Encoding Error")
   End Select
End If

Select Case TermFunction
     Case "~": 'ignore term

     Case "%": 'straight-through literal transfer
            'form dative construction
            If InStr(SymbolicConstant(TermReferent), "_D_") Then
               If TermReferent > 0 Then
                 If (Left(SymbolicConstant(TermReferent - 1), 2) <> "EP") Then
                    Sentence = Sentence & "to "
                 End If
               End If
            End If
```

```
                    Sentence                          =                          Sentence                          &
gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
                    GoSub InsertPunctuation

            Case "A": 'article transfer
                    'don't insert prepositions if one already present
                    If TermReferent > 0 Then
                       If Left(SymbolicConstant(TermReferent - 1), 2) <> "EP" Then
                         'form dative construction
                         If InStr(SymbolicConstant(TermReferent), "NO_D_") Then
                            Sentence = Sentence & "to "
                         'form genitive construction
                         ElseIf InStr(SymbolicConstant(TermReferent), "NO_G_") Then
                            Sentence = Sentence & "of "
                         End If
                       End If
                    End If

                    'add article(s)
                    Sentence = Sentence & Article

            Case "N": 'noun transfer
                    Sentence = Sentence & Noun
                    GoSub InsertPunctuation

                    'verb and verbal adverb transfer
            Case "V", "D":
                    Sentence = Sentence & VERB
                    GoSub InsertPunctuation

            Case "J": 'verbal adjective transfer
                    If (VerbCode <> 30) And (VerbCode <> 37) And (VerbCode <> 41) Then
                       'determine whether person or non-person
                       Pronoun = ""
                       For SearchPtr = TermReferent - 5 To TermReferent + 5
                          'observe limits
                          If SearchPtr < 0 Then
                             SearchPtr = 0
                          ElseIf SearchPtr > MAX_SENTENCE_COMPONENTS Then
                             SearchPtr = MAX_SENTENCE_COMPONENTS
                          End If
                          'look for person nearby
                          If
(gSentence(SearchPtr).Match(gSentence(SearchPtr).MatchPtr).LexiconEntry.TransferAttributes  And  ASSOCIATION_PERSON)
Then
                             Pronoun = "who "
```

```
                              Exit For
                           End If
                        Next SearchPtr
                        'no person found, so assign non-person
                        If Pronoun = "" Then
                          Pronoun = "which "
                        End If
                     Else
                        Pronoun = ""
                     End If
                     Sentence = Sentence & Pronoun & VERB
                     GoSub InsertPunctuation

               Case "R": 'relative pronoun transfer
                     Pronoun                                                                                    =
gSentence(TermReferent).Match(gSentence(TermReferent).MatchPtr).LexiconEntry.Target
                     'form dative construction
                     If InStr(SymbolicConstant(TermReferent), "_D_") Then
                        'don't insert prepositions if one already present
                        If TermReferent > 0 Then
                          If Left(SymbolicConstant(TermReferent - 1), 2) <> "EP" Then
                             Sentence = Sentence & "to "
                          End If
                        End If
                        'form object case of who if person
                        If Pronoun = "who" Then
                          Pronoun = "whom"
                        End If
                     'form genitive construction
                     ElseIf InStr(SymbolicConstant(TermReferent), "_G_") Then
                        'don't insert prepositions if one already present
                        If TermReferent > 0 Then
                          If Left(SymbolicConstant(TermReferent - 1), 2) <> "EP" Then
                             Sentence = Sentence & "of "
                          End If
                        End If
                        'form object case of who if person
                        If Pronoun = "who" Then
                          Pronoun = "whom"
                        End If
                     End If
                     Sentence = Sentence & Pronoun

               Case Else: 'unknown function
                    Beep
```

```
                    MsgBox ("Invalid Function '" & TermFunction & "' on Term " & ComponentPtr & " of transfer rule:
" & Chr(13) & Rule), MB_ICONEXCLAMATION, ("Transfer Rule Syntax Error")
         End Select

         'capitalize first word of sentence
         Sentence = Sentence & " "
         If ComponentPtr = 0 Then
            'skip bracket and capitalize second letter; otherwise...
            If Left(Sentence, 1) = "[" Then
             Mid(Sentence, 2, 1) = UCase(Mid(Sentence, 2, 1))
            '...capitalize first letter
            Else
             If Sentence <> "" Then
                Mid(Sentence, 1, 1) = UCase(Mid(Sentence, 1, 1))
             End If
            End If
         End If
       Next ComponentPtr

       'end sentence with period if none
       Sentence = Trim(Sentence)
       If Right(Sentence, 1) <> "." Then
        Sentence = Sentence & "."
       End If
       Sentence = Sentence & "  "

       'write transfer details
       gTransferHits = gTransferHits + 1
       AddDetail (""), True
       AddDetail ("  Raw Translation:"), False
       AddDetail (""), False
       AddDetail ("    " & Sentence), False

       'return translation
       TransferSentence = Sentence
    End If

    Exit Function


InsertPunctuation:

    'don't place period in middle of sentence
    If      (Right(gSentence(TermReferent).LiteralText,      1)      =      ".")      And      (ComponentPtr      <>
gTransferRule(RulePtr).NumEnglishComponents) Then
       Return
```

```
    End If

    'insert punctuation
    If InStr(",.!?:;>)-]}:", Right(gSentence(TermReferent).LiteralText, 1)) Then
       Sentence = Sentence & Right(gSentence(TermReferent).LiteralText, 1)
    End If
    Return

TransferError:

    'return complete failure
    AddDetail "  ** [SENTENCE GENERATED RUN-TIME ERROR: " & Error(Err) & "]", False
    TransferSentence = "[COMPLETELY UNTRANSLATABLE SENTENCE OMITTED]"
    Exit Function

End Function

'_____________________________________________________________________________________________________
'translate current document
'_____________________________________________________________________________________________________
Sub Translate ()

    'reset statistical counters
    gTransferAttempts = 0
    gTransferHits = 0
    gLookupHits = 0
    gLookups = 0
    gAbort = False

    'initiate log session
    MousePointerWait
    OpenTransferLog

    'translate sentence by sentence
    TranslateSentences

    'clean raw translation
    frmTranslation.txtTranslation = ApplyCleanupFilters(gTranslation)

    'terminate log session
    CloseTransferLog

End Sub

'_____________________________________________________________________________________________________
'isolate each sentence and translate
```

```
'_____________________________________________________________________________________________
Sub TranslateSentences ()

    Dim CharPtr  As Integer
    Dim StartPtr As Integer
    Dim Progress As Integer
    Dim Char     As String
    Dim Sentence As String


    'don't process empty abstract
    gAbstract = Trim(gAbstract)
    If gAbstract = "" Then
       Exit Sub
    End If

    'display status info as transliteration takes place
    frmMain.pnlStatus.FloodPercent = 0
    frmMain.pnlStatus.FloodShowPct = True
    PassiveHelp "Translating..."

    'if abstract doesn't end with period, append one
    If Right(gAbstract, 1) <> "." Then
       gAbstract = gAbstract & "."
    End If

    'process each sentence
    gTranslation = ""
    StartPtr = 1
    Do
       'escape key aborts translation
       If gAbort Then
        Exit Do
       End If

       'extract sentence
       For CharPtr = StartPtr To Len(gAbstract)
          Char = Mid(gAbstract, CharPtr, 1)
          Sentence = Sentence & Char
          'found sentence terminator
          If InStr(".!?", Char) Then
             StartPtr = CharPtr + 1
             Exit For
          End If
       Next CharPtr
```

```
        'strip leading and trailing whitespace
        Sentence = Trim(Sentence)

        'extract symbolic constants from sentence
        DecomposeSentence (Sentence)

        'disambiguate sentence
        DisambiguateSentence

        'translate sentence
        gTranslation = gTranslation & TransferSentence() & " "
        Sentence = ""

        'update status bar after every sentence
        Progress = (StartPtr / Len(gAbstract)) * 100
        If Progress > 100 Then
          Progress = 100
        End If
        frmMain.pnlStatus.FloodShowPct = True
        frmMain.pnlStatus.FloodPercent = Progress
    Loop Until StartPtr >= Len(gAbstract)

    'hide status
    DisplayWait 0, ""

End Sub

'_____________________________________________________________________________________________
'apply import filter backwards to transliterate Cyrillic text into Roman
'_____________________________________________________________________________________________
Function Transliterate (InText As String) As String

    Dim FilterPtr     As Integer
    Dim Found         As Integer
    Dim AbstractPtr   As Integer
    Dim AbstractLen   As Integer
    Dim UpdateTally   As Integer
    Dim DefaultFilter As Integer
    Dim CurrentChar   As String
    Dim InCode        As String
    Dim OutText       As String


    'set starting and ending points of abstract
    AbstractLen = Len(InText)
```

```
    'display status info as transliteration takes place
    DisplayWait -AbstractLen, "Transliterating from the Cyrillic alphabet to the Roman alphabet..."

    'remember default filter for this abstract
    DefaultFilter = frmFilter.lstFilters(FILTER_NAME).ListIndex

    'locate selected transliteration filter in import filter list
    Found = False
    For FilterPtr = 0 To frmFilter.lstFilters(FILTER_NAME).ListCount - 1
       If frmFilter.lstFilters(FILTER_NAME).List(FilterPtr) = gTransliterationFilter Then
          Found = True
          Exit For
       End If
    Next FilterPtr

    'abort if filter not located; otherwise...
    If Not Found Then
       Beep
       MsgBox ("Cannot locate import filter '" & gTransliterationFilter & "'."), MB_ICONSTOP, ("Transliteration
Error")

    '...proceed with transliteration
    Else
       'temporarily change default filter to selected filter
       frmFilter.lstFilters(FILTER_NAME).ListIndex = FilterPtr
       UnpackCurrentFilter
       ConvertFilterMap

       'for each Cyrillic character in abstract...
       For AbstractPtr = 1 To AbstractLen
          Found = False
          '...find its Transliterate equivalent filter mapping
          For FilterPtr = 0 To NUM_MAPPINGS
              InCode = gInputFilter(FilterPtr).OutCode
              'if found, stop looking
              CurrentChar = Mid(InText, AbstractPtr, 1)
              If CurrentChar = InCode And InCode <> "" Then
                Found = True
                Exit For
              End If
          Next FilterPtr

          'if found, add transliterated character(s) to transliterated abstract and move to next character(s) in
original...
          If Found Then
             'only transliterate mapped characters (ie. ignore null)
```

```
            If gInputFilter(FilterPtr).InCode > Chr(0) Then
              OutText = OutText & gInputFilter(FilterPtr).InCode
            End If
          '...otherwise pass unknown character straight through untransliterated (it must be Roman already)
          Else
             OutText = OutText & CurrentChar
          End If

          'update status bar at every 5% change in work done
          UpdateTally = UpdateTally + 1
          If UpdateTally > (AbstractLen - 1) / 20 Then
             DisplayWait UpdateTally, ""
             UpdateTally = 0
          End If
       Next AbstractPtr

       'reset default filter
       frmFilter.lstFilters(FILTER_NAME).ListIndex = DefaultFilter
    End If

    'hide status
    DisplayWait 0, ""

    'return transliterated text
    Transliterate = OutText

End Function

'_____________________________________________________________________________________________________
'take packed string representation of character set mappings from filter list box and expand
'them into edit fields.  Packed form eliminates external data structures and their management
'_____________________________________________________________________________________________________
Sub UnpackCurrentFilter ()


    Dim i        As Integer
    Dim FieldEnd As Integer
    Dim Entry    As String
    Dim PackLine As String


    'skip over filter name.  It's present only for alphabetization purposes.
    PackLine   =   Mid(frmFilter.lstFilters(FILTER_MAP),   InStr(frmFilter.lstFilters(FILTER_MAP),   ":")   +   1)   &
gPackDelimiter

    'pull out each entry and load respective text box with it
```

```
    For i = 0 To NUM_MAPPINGS
       FieldEnd = InStr(PackLine, gPackDelimiter)
       Entry = Left(PackLine, FieldEnd - 1)
       PackLine = Mid(PackLine, FieldEnd + 1)
       frmFilter.txtLatin(i) = Entry
    Next i

End Sub

'_____________________________________________________________________________________________________
'make sure each noun pattern is unique (exceptions allowed)
'_____________________________________________________________________________________________________
Sub VerifyUniqueInflectionPatterns ()

    Dim i As Integer
    Dim j As Integer


    MousePointerWait

    'compare each noun inflection pattern against all others
    For i = 1 To NUM_INFLECTION_PATTERNS
       For j = i + 1 To NUM_INFLECTION_PATTERNS
           'if match, display message. Don't check upper patterns (personal pronouns have intentional duplicates)
           If (gPattern(i) = gPattern(j)) And (gPattern(i) <> "") And i < 390 And j <> 410 And j <> 26 And j <> 100
Then
              Beep
              MsgBox ("Inflection patterns " & i & " and " & j & " are identical."), MB_ICONSTOP, ("Duplicate
Entries")
              End
           End If
       Next j
    Next i

    MousePointerContinue

End Sub

'_____________________________________________________________________________________________________
'perform page formatting and print text
'_____________________________________________________________________________________________________
Sub WordWrap (TextBody As String, LeftMargin As Integer, TopMargin As Integer, LineWidth As Integer)

    Dim CharPtr      As Integer
    Dim BackPtr      As Integer
```

```
Dim LineStartPtr As Integer
Dim TextSpan     As Integer
Dim TextHigh     As Integer
Dim LineCounter  As Integer

Const PAGE_HEIGHT = 12250
Const HOT_ZONE_FLUFF = 350  'num twips allowed past right margin before forced wrap


'set left margin
Printer.CurrentX = LeftMargin

'loop through text to print
LineStartPtr = 1
For CharPtr = 1 To Len(TextBody)

   'get size of current line
   TextSpan = Printer.TextWidth(Mid(TextBody, LineStartPtr, CharPtr - LineStartPtr))
   TextHigh = Printer.TextHeight("A")

   'if current line goes into hot-zone and can be broken, do it; otherwise...
   If (TextSpan > LineWidth) And (Mid(TextBody, CharPtr, 1) = " " Or Mid(TextBody, CharPtr, 1) = "-") Then
      'print current line
      Printer.Print Trim(Mid(TextBody, LineStartPtr, CharPtr - LineStartPtr + 1))

      'reset left margin
      Printer.CurrentX = LeftMargin

      'insert page break if nesc.
      GoSub PageBreak
      LineStartPtr = CharPtr + 1

   'if it can't be broken in hot-zone, find where it can be broken
   ElseIf TextSpan > LineWidth + HOT_ZONE_FLUFF Then

      'back search from current character
      For BackPtr = CharPtr To CharPtr - 25 Step -1

          'when space found, make this end of line
          If (Mid(TextBody, BackPtr, 1) = " ") Or (Mid(TextBody, CharPtr, 1) = "-") Then
            'print as much of current line as possible
            Printer.Print Trim(Mid(TextBody, LineStartPtr, BackPtr - LineStartPtr + 1))

            'reset left margin
            Printer.CurrentX = LeftMargin
```

```
                'insert page break if nesc.
                GoSub PageBreak

                'update pointers
                LineStartPtr = BackPtr + 1
                CharPtr = BackPtr + 1
                Exit For
              End If
          Next BackPtr
       End If
    Next CharPtr

    'print last line
    Printer.CurrentX = LeftMargin
    Printer.Print Trim(Mid(TextBody, LineStartPtr))

    Exit Sub


PageBreak:
    'increment vertical position
    LineCounter = LineCounter + TextHigh
    'force new page if current one exceeded
    If LineCounter > PAGE_HEIGHT Then
       Printer.NewPage
       'reset for new page
       LineCounter = 0
       Printer.CurrentY = TopMargin
    End If
    Return

End Sub

'_____________________________________________________________________________________________________
'(frmAbout): shows general program and copyright information
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    'close About form
    PassiveHelp ""
    frmAbout.Hide

End Sub
```

```
'_____________________________________________________________________________________________________
Sub Form_Activate ()

    'set form colors according to user preferences
    pnlRoot.BackColor = BACKGROUND_COLOR
    pnlTitle.BackColor = BACKGROUND_COLOR
    cmdOK.BackColor = BACKGROUND_COLOR

    'center modal About form on MDI parent space
    Left = frmMain.Left + frmMain.ScaleWidth / 2 - Width / 2
    Top = frmMain.Top + frmMain.ScaleHeight / 2 - Height / 2.6
    lblVersion = "Version " & VERSION_NUMBER

End Sub

'_____________________________________________________________________________________________________
Sub Form_Deactivate ()

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'(frmCleanupFilterEditor): provides method of fixing consistent translation errors
'_____________________________________________________________________________________________________
Option Explicit

Dim fChangesMade As Integer    'flag indicating whether changes made

'_____________________________________________________________________________________________________
Sub cmdAdd_Click ()

    'init and bring up cleanup expression dialog editor
    frmCleanupExpressionEditor.txtMalformedExpression = ""
    frmCleanupExpressionEditor.txtCorrectedExpression = ""
    frmCleanupExpressionEditor.Show 1

    'if expression not canceled, add it to filter lists
    If frmCleanupExpressionEditor.Tag <> "[CANCEL]" Then
       lstCleanupFilter(0).AddItem frmCleanupExpressionEditor.txtMalformedExpression
       lstCleanupFilter(1).AddItem frmCleanupExpressionEditor.txtCorrectedExpression
       'update editor
       cmdCancel.Enabled = True
       fChangesMade = True
```

```
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdAdd_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Add a new malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'if changes made, prompt to confirm cancel
    If fChangesMade Then
       Beep
       MsgBoxAnswer = MsgBox("Cancel will discard all the changes made." & Chr(13) & "Continue?", MB_ICONQUESTION +
MB_YESNO + MB_DEFBUTTON2, "Confirm Cancel")
       If MsgBoxAnswer = ID_NO Then
          Exit Sub
       End If
    End If

    'overwrite changes
    LoadCleanupFilters

    'close cleanup editor and return to main interface
    timerScroll.Enabled = False
    cmdCancel.Enabled = False
    PassiveHelp ""
    Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Discard any changes made here and return to the main interface."

End Sub

'_____________________________________________________________________________________________________________
```

```
Sub cmdClose_Click ()

    Dim i As Integer

    On Error GoTo ErrorOnCleanupWrite


    'update changes only if any made
    If fChangesMade Then
       fChangesMade = False
       MousePointerWait
       Open CLEANUP_FILTER_FILENAME For Output As 1
       For i = 0 To lstCleanupFilter(0).ListCount - 1
           Write #1, lstCleanupFilter(0).List(i), lstCleanupFilter(1).List(i)
       Next i
       Close 1
       MousePointerContinue
    End If

    'close cleanup editor and return to main interface
    timerScroll.Enabled = False
    cmdCancel.Enabled = False
    PassiveHelp ""
    Hide

    Exit Sub


ErrorOnCleanupWrite:

    'prompt user to deal with store fail
    Beep
    Close 1
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to write to the cleanup filter file '" & UCase(CLEANUP_FILTER_FILENAME) & "':" &
Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to store it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'otherwise sacrifice changes
    Else
       Exit Sub
    End If

End Sub
```

```
'_____________________________________________________________________________________________________
Sub cmdClose_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Save any changes made here and return to the main interface."

End Sub

'_____________________________________________________________________________________________________
Sub cmdDelete_Click ()

    'confirm delete
    Beep
    MsgBoxAnswer = MsgBox("Are you sure you want to delete the current entry?", MB_ICONQUESTION + MB_YESNO +
MB_DEFBUTTON2, "Confirm Delete")
    If MsgBoxAnswer = ID_YES Then
       'delete current entry
       lstCleanupFilter(0).RemoveItem lstCleanupFilter(0).ListIndex
       lstCleanupFilter(1).RemoveItem lstCleanupFilter(1).ListIndex
       'update editor
       cmdEdit.Enabled = False
       cmdDelete.Enabled = False
       cmdCancel.Enabled = True
       fChangesMade = True
    End If

End Sub

'_____________________________________________________________________________________________________
Sub cmdDelete_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Delete the selected malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________
Sub cmdEdit_Click ()

    'init and bring up cleanup expression dialog editor
    frmCleanupExpressionEditor.txtMalformedExpression = lstCleanupFilter(0)
    frmCleanupExpressionEditor.txtCorrectedExpression = lstCleanupFilter(1)
    frmCleanupExpressionEditor.Show 1

    'if expression not canceled, process it
```

```
    If frmCleanupExpressionEditor.Tag <> "[CANCEL]" Then
       'delete current entry
       lstCleanupFilter(0).RemoveItem lstCleanupFilter(0).ListIndex
       lstCleanupFilter(1).RemoveItem lstCleanupFilter(1).ListIndex
       'add modified entry
       lstCleanupFilter(0).AddItem frmCleanupExpressionEditor.txtMalformedExpression
       lstCleanupFilter(1).AddItem frmCleanupExpressionEditor.txtCorrectedExpression
       'update editor
       cmdEdit.Enabled = False
       cmdDelete.Enabled = False
       cmdCancel.Enabled = True
       fChangesMade = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdEdit_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Edit the selected malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP1_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on Cleanup Filter Editor."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set default command availabilty
```

```
    cmdClose.Enabled = True
    timerScroll.Enabled = True
    cmdAdd.SetFocus

End Sub

'_____________________________________________________________________________________________________________
Sub lstCleanupFilter_Click (Index As Integer)

    'set focus to both lists simultaneously
    lstCleanupFilter(1 - Index).ListIndex = lstCleanupFilter(Index).ListIndex

    'edit and delete now available
    cmdEdit.Enabled = True
    cmdDelete.Enabled = True

End Sub

'_____________________________________________________________________________________________________________
Sub lstCleanupFilter_DblClick (Index As Integer)

    cmdEdit_Click

End Sub

'_____________________________________________________________________________________________________________
Sub lstCleanupFilter_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'set focus to both lists simultaneously
    lstCleanupFilter(1 - Index).ListIndex = lstCleanupFilter(Index).ListIndex

End Sub

'_____________________________________________________________________________________________________________
Sub lstCleanupFilter_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select an existing malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""
```

```
End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot2_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'deal with scrolling both lists simultaneously
'_____________________________________________________________________________________________________
Sub timerScroll_Timer ()

    Static PrevTop0 As Integer
    Static PrevTop1 As Integer


    'if first list changed, reposition second
    If lstCleanupFilter(0).TopIndex <> PrevTop0 Then
       PrevTop0 = lstCleanupFilter(0).TopIndex
       lstCleanupFilter(1).TopIndex = lstCleanupFilter(0).TopIndex
    'if second list changed, reposition first
    ElseIf lstCleanupFilter(1).TopIndex <> PrevTop1 Then
       PrevTop1 = lstCleanupFilter(1).TopIndex
       lstCleanupFilter(0).TopIndex = lstCleanupFilter(1).TopIndex
    End If

End Sub

'_____________________________________________________________________________________________________
'(frmConvert): manual conversion dialog requires user to select import filter from list
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'confirm Cancel
    Beep
    PassiveHelp "'Yes' returns to the main interface without loading the abstract; 'No' allows you to select new
filter(s)."
    MsgBoxAnswer = MsgBox("Without an input filter, this abstract cannot be read in and translated." & Chr(13) &
Chr(13) & "Do you really want to cancel?", MB_ICONINFORMATION + MB_YESNO + MB_DEFBUTTON2, "Info")
```

```
    PassiveHelp ""

    'if user does not want to continue, then abort Open
    If MsgBoxAnswer = ID_YES Then
       'return to main form with failure
       Tag = "[CANCEL]"
       frmConvert.Hide
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Cancel trying to load this abstract."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP4_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on converting abstracts from unknown formats."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_Click ()

    Dim Found As Integer
    Dim i     As Integer


    'determine if any filters chosen
    Found = False
```

```
    For i = 0 To lstFilters.ListCount - 1
        If lstFilters.Selected(i) Then
           Found = True
           Exit For
        End If
    Next i

    'if no filters chosen, display error message
    If Not Found Then
       Beep
       PassiveHelp "Only those filters which you select will be applied to the abstract.  You must select at least
one."
       MsgBox ("You must select at least one filter." & Chr(13) & Chr(13) & "If you are unsure of which filter to
apply, click Select All and the correct filter (if available) for this abstract format will be determined
automatically."), MB_ICONSTOP, ("Missing Input")
       PassiveHelp ""
       Exit Sub
    End If

    'apply selected filters to abstract
    MousePointerWait
    Found = False
    For i = 0 To lstFilters.ListCount - 1
        If lstFilters.Selected(i) Then
           frmFilter.lstFilters(FILTER_NAME).ListIndex = i
           UnpackCurrentFilter
           ConvertFilterMap

           'apply filter to convert abstract to internal format
           gAbstract     =     Trim(ConvertAbstract(gRawAbstract,     True,     "Applying     filter     '"     &
frmFilter.lstFilters(FILTER_NAME).List(i)  &  "'  to  the  abstract  in  an  attempt  to  convert  it  to  a  readable
format..."))

           'check first few abstract words against lexicon to verify correct conversion
           If ConversionOK(gAbstract) Then
              Found = True
              Exit For
           End If
        End If
    Next i
    MousePointerContinue

    'if conversion still failed, prompt user for next step
    If Not Found Then
       Beep
```

```
       PassiveHelp "'Retry' allows you to reselect the filters to apply; 'Cancel' aborts trying to load this
abstract."
       MsgBoxAnswer = MsgBox("None of the selected filters resulted in successful conversion.", MB_ICONEXCLAMATION +
MB_RETRYCANCEL, "Conversion Unsuccessful")
       PassiveHelp ""
       If MsgBoxAnswer = ID_RETRY Then
          'return to select filter(s)
          Tag = "[RETRY]"
          Exit Sub
       End If
       'return to main form with cancel code
       Tag = "[CANCEL]"
       frmConvert.Hide
       Exit Sub
    End If

    'return to main form with success
    PassiveHelp ""
    Tag = "[OK]"
    frmConvert.Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Apply the selected filter(s) to the abstract in an attempt to convert it to a readable format."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdSelectAll_Click ()

    Dim i As Integer


    'select all available filters
    For i = 0 To lstFilters.ListCount - 1
        lstFilters.Selected(i) = True
    Next i

End Sub

'_____________________________________________________________________________________________________________
Sub cmdSelectAll_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    'show topical help on info line
    PassiveHelp "Select all the available filters shown.  Use this if you are unsure of which filters to apply."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    Dim i As Integer


    Beep
    MousePointerContinue

    'get available filter names
    lstFilters.Clear
    For i = 0 To frmFilter.lstFilters(FILTER_NAME).ListCount - 1
        lstFilters.AddItem frmFilter.lstFilters(FILTER_NAME).List(i)
    Next i

    'deselect all filters
    For i = 0 To lstFilters.ListCount - 1
        lstFilters.Selected(i) = False
    Next i

End Sub

'_____________________________________________________________________________________________________________
Sub lstFilters_DblClick ()

    'double-click selects filter and OK
    cmdOK_Click

End Sub

'_____________________________________________________________________________________________________________
Sub lstFilters_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select the filter(s) which might be used to convert the unknown format."

End Sub

'_____________________________________________________________________________________________________________
```

```
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'(frmCustomize): allows general program settings to be modified
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub chkClock_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Hide or display the clock on the status bar."

End Sub

'_____________________________________________________________________________________________________
Sub chkPassiveHelp_Click ()

    'if passive help selected, make sure status bar enabled
    If chkPassiveHelp Then
       chkStatusBar = 1
    End If

End Sub

'_____________________________________________________________________________________________________
Sub chkPassiveHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Hide or display this type of passive help message on the status bar."

End Sub

'_____________________________________________________________________________________________________
Sub chkStatusBar_Click ()

    'if no status bar selected, disable passive help and clock
    If chkStatusBar = 0 Then
       chkPassiveHelp = 0
    End If
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub chkStatusBar_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Hide or display the status bar at the bottom of the screen."

End Sub

'_____________________________________________________________________________________________________________
Sub chkTransferLog_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Enable or disable logging of translation details.   When disabled, translation is moderately
faster."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'close customize dialog
    Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Discard any changes made to the system configuration settings and return to the main interface."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP3_FILENAME
    frmHelp.Show 1

End Sub
```

```
'_____________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on setting system configurations."

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    On Error GoTo ErrorOnConfigWrite


    'update changes:
    'status bar
    frmMain.pnlStatusBar.Visible = (chkStatusBar = 1)

    'passive help
    PassiveHelp ""
    gShowHelp = (chkPassiveHelp = 1)

    'transfer details log
    gTransferDetails = (chkTransferLog = 1)

    'have list box to select transliterate filter
    gTransliterationFilter = lstTransliterationFilter.Text

    'operation mode
    If optOperationMode(0) Then
       gOperationMode = ABSTRACT_MODE
       frmMain.menuSaveConvertedAbstract.Caption = "Converted &Abstract"
       frmMain.menuPrintConvertedAbstract.Caption = "Converted &Abstract"
    Else
       gOperationMode = EMAIL_MODE
       frmMain.menuSaveConvertedAbstract.Caption = "&E-Mail Message"
       frmMain.menuPrintConvertedAbstract.Caption = "&E-Mail Message"
    End If

    'post-editor configurations
    gPostEditorAPIName = txtApplicationTitle
    gPostEditorFilename = txtApplicationFilename
    gPostEditorPasteSequence = txtPasteSequence

    'save changes
    MousePointerWait
```

```
    Open CONFIG_FILENAME For Output As 1
    Print #1, (chkStatusBar = 1)
    Print #1, gShowHelp
    Print #1, gTransferDetails
    Write #1, gTransliterationFilter
    Write #1, gPostEditorAPIName
    Write #1, gPostEditorFilename
    Write #1, gPostEditorPasteSequence
    Close 1
    MousePointerContinue

    'close customize dialog
    Hide

    Exit Sub


ErrorOnConfigWrite:

    MousePointerContinue
    Beep

    MsgBoxAnswer = MsgBox("Unable to write configuration settings:" & Chr(13) & Error(Err), MB_ICONSTOP +
MB_RETRYCANCEL, "File Error")
    'if user wants to try again, attempt to re-execute operation...
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    '...otherwise abort operation
    Else
       cmdOK.Enabled = False
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Accept these system configuration settings and return to the main interface."

End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()
```

```
    Dim i As Integer


    'get current filter list
    lstTransliterationFilter.Clear
    For i = 0 To frmFilter.lstFilters(0).ListCount - 1
        lstTransliterationFilter.AddItem frmFilter.lstFilters(0).List(i)
    Next i

    'initialize form components
    cmdOK.Enabled = True

    'status bar
    chkStatusBar = Abs(frmMain.pnlStatusBar.Visible)

    'transfer log
    chkTransferLog = Abs(gTransferDetails)

    'passive help
    PassiveHelp ""
    chkPassiveHelp = Abs(gShowHelp)

    'search list box for transliterate filter
    lstTransliterationFilter.ListIndex = -1
    For i = 0 To lstTransliterationFilter.ListCount - 1
        If lstTransliterationFilter.List(i) = gTransliterationFilter Then
           lstTransliterationFilter.ListIndex = i
           Exit For
        End If
    Next i

    'deal with missing filter
    If lstTransliterationFilter.ListIndex = -1 Then
       Beep
       MsgBox ("Unable to find transliteration filter '" & gTransliterationFilter & "'."), MB_ICONSTOP, ("Missing
Filter")
    End If

    'operation mode
    If gOperationMode = ABSTRACT_MODE Then
       optOperationMode(0) = True
    Else
       optOperationMode(1) = True
    End If

    'post-editor configurations
```

```
    txtApplicationTitle = gPostEditorAPIName
    txtApplicationFilename = gPostEditorFilename
    txtPasteSequence = gPostEditorPasteSequence

End Sub

'_____________________________________________________________________________________________________________
Sub optOperationMode_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    If Index = 0 Then
       PassiveHelp "Set up the system primarily for translating abstracts."
    Else
       PassiveHelp "Set up the system primarily for transliterating e-mail messages."
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub pnlFilter_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlFilterShadow_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select the filter for transliterating Cyrillic to Latin characters."

End Sub

'_____________________________________________________________________________________________________________
Sub pnlInterface_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlOperationMode_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub pnlPostEditor_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub txtApplicationFilename_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return move to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPasteSequence.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtApplicationFilename_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Enter the path and executable filename (.exe) of the post-editor software."

End Sub

'_____________________________________________________________________________________________________________
Sub txtApplicationTitle_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
```

```
      KeyAscii = Asc("'")
   End If

   'carriage return move to next field
   If KeyAscii = 13 Then
      KeyAscii = 0
      txtApplicationFilename.SetFocus
   End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtApplicationTitle_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'show topical help on info line
   PassiveHelp "Enter the name of the post-editor software (i.e. Microsoft Word 6.0, Word Perfect 5.0, etc.)."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPasteSequence_KeyPress (KeyAscii As Integer)

   'convert double-quotes to single-quotes
   If KeyAscii = 34 Then
      KeyAscii = Asc("'")
   End If

   'carriage return move to ok button
   If KeyAscii = 13 Then
      KeyAscii = 0
      cmdOK.SetFocus
   End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPasteSequence_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'show topical help on info line
   PassiveHelp "Enter the sequence of hotkeys used to paste from the clipboard into the post-editor document."

End Sub

'_____________________________________________________________________________________________________________
'(frmCleanupExpressionEditor): simple entry dialog for cleanup filter editor
```

```
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'return to cleanup editor with abort code
    Tag = "[CANCEL]"
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Discard the current malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    'return to cleanup editor with ok code
    Tag = "[OK]"
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Accept the current malformed expression and its correction."

End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()

    'set initial focus to first entry field
    txtMalformedExpression.SelStart = 0
    txtMalformedExpression.SelLength = Len(txtMalformedExpression)
    txtMalformedExpression.SetFocus
```

```
End Sub

'_____________________________________________________________________________________________________
Sub txtCorrectedExpression_Change ()

    'ok command availability dependent on completing both entries
    cmdOK.Enabled = ((txtMalformedExpression <> "") And (txtCorrectedExpression <> ""))

End Sub

'_____________________________________________________________________________________________________
Sub txtCorrectedExpression_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'if ok button available, set focus to it
       If cmdOK.Enabled Then
          cmdOK.SetFocus
       '...otherwise go to first incomplete field
       ElseIf txtMalformedExpression = "" Then
          txtMalformedExpression.SetFocus
       Else
          txtCorrectedExpression.SetFocus
       End If
    End If

End Sub

'_____________________________________________________________________________________________________
Sub txtMalformedExpression_Change ()

    'ok command availability dependent on completing both entries
    cmdOK.Enabled = ((txtMalformedExpression <> "") And (txtCorrectedExpression <> ""))

End Sub

'_____________________________________________________________________________________________________
Sub txtMalformedExpression_KeyPress (KeyAscii As Integer)
```

```
    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtCorrectedExpression.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________
'(frmFilter): import filter editor, used to create, modify, and delete Cyrillic conversion filters
'_____________________________________________________________________________________________________
Option Explicit


'_____________________________________________________________________________________________________
Sub cmdAutoMap_Click ()

    'display info box about unsupported feature.  This code should never be called, though, since
    'the 'Auto-Map' command button is permanently disabled.
    Beep
    PassiveHelp ""
    MsgBox ("The Auto-Map feature is not supported in Version " & VERSION_NUMBER & " of this program."),
MB_ICONINFORMATION, (PROGRAM_TITLE & " v" & VERSION_NUMBER)

End Sub

'_____________________________________________________________________________________________________
Sub cmdAutoMap_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "_Filter_Setup_AUTOMAP"

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'reload available filters to negate any changes made
    LoadFilters
```

```
    'convert entries to active filter mapping
    ConvertFilterMap

    'close Filter form without saving changes the current filter
    frmFilter.Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Return to the main interface without saving any changes made here."

End Sub

'_____________________________________________________________________________________________________
Sub cmdDelete_Click ()

    Dim i As Integer


    'don't allow built-in filters to be deleted
    If      (lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)      =      "Native      Format")      Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)           =           "Transliterate")           Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)              =              "KOI-7")              Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex) = "Phonetic Unique") Then
       Beep
       MsgBox ("This is a built-in filter and cannot be deleted."), MB_ICONSTOP, ("Illegal Operation")
       Exit Sub
    End If

    'confirm filter delete
    Beep
    PassiveHelp "'Yes' will delete this filter from the list of available filters; 'No' will keep it."
    MsgboxAnswer     =     MsgBox("Are     you     sure     you     want     to     delete     filter     '"     &
lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)  &  "'?",  MB_ICONQUESTION  +  MB_YESNO,  "Confirm
Delete")
    PassiveHelp ""

    'if confirmed...
    If MsgboxAnswer = ID_YES Then
       'remove current filter from both lists
       lstFilters(FILTER_NAME).RemoveItem (lstFilters(FILTER_NAME).ListIndex)
       lstFilters(FILTER_MAP).RemoveItem (lstFilters(FILTER_MAP).ListIndex)
```

```
       'disable delete button
       cmdDelete.Enabled = False

       'blank filter mappings because no active filter selected
       For i = 0 To NUM_MAPPINGS
           txtLatin(i) = ""
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdDelete_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Remove filter '" & lstFilters(FILTER_NAME).Text & "' from the list of available filters."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP4_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on Import Filter Editor."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_Click ()

    'don't allow OK if no filter selected
    If lstFilters(FILTER_NAME).Text = "" Then
       Beep
       PassiveHelp "A default filter must be selected from the list of available filters."
       MsgBox ("No default filter is selected."), MB_ICONSTOP, ("Setup Error")
       PassiveHelp ""
```

```
      Exit Sub
   End If

   'error check entries
   If MappingError() Then
      Exit Sub
   End If

   'pack currently active filter so it's saved as current name
   frmFilterSave.txtFilterName = lstFilters(FILTER_NAME).Text
   PackCurrentFilter (True)

   'save filters to filter file
   SaveFilters

   'convert entries to active filter mapping
   ConvertFilterMap

   'close Filter form
   frmFilter.Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'show topical help on info line
   PassiveHelp "Save any changes made here and return to the main interface."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdSaveAs_Click ()

   'error check entries
   If MappingError() Then
      Exit Sub
   End If

   'call up filter name entry form.  Note: in order to rename a filter, it must be saved under a
   'new name and old filter must be deleted
   frmFilterSave.Show 1

   'if OK selected, pack current filter with its name
   If frmFilterSave.Tag = "[OK]" Then
      PackCurrentFilter (True)
```

```
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdSaveAs_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Save the currently displayed filter values with a new name."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set form colors according to user preferences
    pnlRoot.BackColor = BACKGROUND_COLOR
    pnlMain.BackColor = BACKGROUND_COLOR
    pnlPanel(0).BackColor = BACKGROUND_COLOR
    pnlPanel(1).BackColor = BACKGROUND_COLOR

    'adjust 3d list box to current system configurations
    pnlFilterName.Height = lstFilters(FILTER_NAME).Height + 50
    pnlFilterName.Width = lstFilters(FILTER_NAME).Width + 50

    'open default folder
    pnlFolderTab_Click (Val(Tag))

    'if opening filter form from main, set focus to filter list box, not first mapping field
    If Tag = "00" Then
       Tag = "0"
       lstFilters(FILTER_NAME).SetFocus
    End If

    MousePointerContinue

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Deactivate ()

    PassiveHelp ""

End Sub
```

```
'_____________________________________________________________________________________________________
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    'alt keys intercepted to open folders or set focus to default filter
    If Shift = 4 Then
       'alt-a opens 'Alphabet' folder
       If KeyCode = 65 Then
          pnlFolderTab_Click (0)
       'alt-m opens 'Miscellaneous' folder
       ElseIf KeyCode = 77 Then
          pnlFolderTab_Click (1)
       'alt-f sets focus to default filter
       ElseIf KeyCode = 70 Then
          lstFilters(FILTER_NAME).SetFocus
       End If
       KeyCode = 0
    End If

End Sub

'_____________________________________________________________________________________________________
Sub Form_Load ()

    Dim i       As Integer
    Dim YOffset As Integer


    DoEvents

    'hide background on each object.  Backgrounds are shown for alignment purposes at design time
    For i = 0 To NUM_MAPPINGS
        lblCyrillic(i).BackStyle = STYLE_TRANSPARENT
        txtLatin(i).BackColor = WHITE
    Next i
    DoEvents

    'align text boxes and labels
    YOffset = 50
    For i = 0 To 10
        txtLatin(i).Top = YOffset
        lblCyrillic(i).Top = YOffset - 60
        txtLatin(i + 11).Top = YOffset
        lblCyrillic(i + 11).Top = YOffset - 60
        txtLatin(i + 22).Top = YOffset
        lblCyrillic(i + 22).Top = YOffset - 60
        txtLatin(i + 33).Top = YOffset
```

```
        lblCyrillic(i + 33).Top = YOffset - 10
        txtLatin(i + 44).Top = YOffset
        lblCyrillic(i + 44).Top = YOffset - 10
        txtLatin(i + 55).Top = YOffset
        lblCyrillic(i + 55).Top = YOffset - 10
        txtLatin(i + 66).Top = YOffset
        lblCyrillic(i + 66).Top = YOffset - 10
        txtLatin(i + 77).Top = YOffset
        lblCyrillic(i + 77).Top = YOffset - 10
        txtLatin(i + 88).Top = YOffset
        lblCyrillic(i + 88).Top = YOffset - 10
        YOffset = YOffset + 260
    Next i
    DoEvents

    'get available filters and set default
    LoadFilters
    ConvertFilterMap

End Sub

'_____________________________________________________________________________________________________
'load available filters from single filter file
'_____________________________________________________________________________________________________
Sub LoadFilters ()

    Dim FilterLine    As String
    Dim DefaultFilter As Integer

    On Error GoTo LoadFiltersError


    'clear filter lists
    lstFilters(FILTER_NAME).Clear
    lstFilters(FILTER_MAP).Clear

    'open filter file
    MousePointerWait
    Open FILTER_FILENAME For Input As 1

    'get pointer to default filter
    If Not EOF(1) Then
       Input #1, DefaultFilter
    'flag filter file format error
    Else
       DefaultFilter = -1
```

```
    End If

    'read filter entries from file
    Do Until EOF(1)
       Line Input #1, FilterLine
       'strip leading quote and trailing quotes
       FilterLine = Mid(FilterLine, 2)
       FilterLine = Left(FilterLine, Len(FilterLine) - 1)
       'extract filter name from entry and add to filter name list
       lstFilters(FILTER_NAME).AddItem Trim(Left(FilterLine, InStr(FilterLine, ":") - 1))
       'store filter map in map list
       lstFilters(FILTER_MAP).AddItem FilterLine
    Loop

    'set default filter (or none if no filters available)
    lstFilters(FILTER_NAME).ListIndex = DefaultFilter

    Close 1
    MousePointerContinue

    Exit Sub


LoadFiltersError:

    Close 1

    MousePointerContinue
    Beep

    'prompt user to deal with error
    PassiveHelp "An error has occurred while getting the available filters.  'Retry' will attempt to get them
again."
    MsgboxAnswer = MsgBox("Unable to load the import filters '" & UCase(FILTER_FILENAME) & "':" & Chr(13) &
Error(Err), MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    PassiveHelp ""
    If MsgboxAnswer = ID_RETRY Then
       Resume
    End If

    'inform that 'Open' function no longer available, then disable it
    PassiveHelp ""
    MsgBox ("Without any filters, it will not be possible to load abstracts.  However, most other functions will
still be available."), MB_ICONINFORMATION, ("Fatal Error")
    frmMain.menuOpenAbstract.Enabled = False
```

```
    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub lstFilters_Click (Index As Integer)

    'make sure both lists always syncronized
    If Index = FILTER_NAME Then
       'record currently open filter
       lstFilters(FILTER_MAP).ListIndex = lstFilters(FILTER_NAME).ListIndex
       'display filter entries
       UnpackCurrentFilter
    End If

    'enable delete button when filter selected
    cmdDelete.Enabled = True

End Sub

'_____________________________________________________________________________________________________________
'error-check character set mappings entered.  Filter cannot be saved with errors
'_____________________________________________________________________________________________________________
Function MappingError () As Integer

    Dim ColonAt As Integer
    Dim LPram   As String
    Dim RPram   As String
    Dim i       As Integer


    MappingError = True

    'error check entry fields
    For i = 0 To NUM_MAPPINGS
        'check for omissions
        If txtLatin(i) = "" Then
           Beep
           'open misc folder if focus on one of its text fields
           If i > 65 Then
              pnlFolderTab_Click (1)
           End If
           txtLatin(i).SetFocus
           PassiveHelp "All mapping entries must have a value."
           MsgBox ("Entries cannot be left blank.  Enter a zero if this mapping is to be ignored."), MB_ICONSTOP,
("Missing Input")
```

```
            PassiveHelp ""
            Exit Function
         End If

         'since it's a likely error to try to map the digit 0 to the character zero, verify that
         'the user understands what this means.  The digit 0 is interpreted as ASCII 0.  This holds
         'true for all digits 0-9, but only 0 is checked:  if they misunderstood 0, then they surely
         'misunderstood them all
         If i = 67 And Val(txtLatin(67)) < 10 Then
            Beep
            PassiveHelp ""
            MsgboxAnswer = MsgBox("Mappings entered as numbers are interpreted as ASCII codes.  Normally the digits
0-9 have mappings of 48-57.  Are you sure that your mappings for the digits 0-9 are correct?", MB_ICONQUESTION +
MB_YESNO, "Query")
            'if user answers no, set focus to field in question
            If MsgboxAnswer = ID_NO Then
               pnlFolderTab_Click (1)
               txtLatin(67).SetFocus
               Exit Function
            End If
         End If

         'auto-convert lone $ to ASCII value so no confusion with hex indicator
         If txtLatin(i) = "$" Then
            txtLatin(i) = "38"
         End If

         'auto-convert lone : to ASCII value so no confusion with two-byte separator
         If txtLatin(i) = ":" Then
            txtLatin(i) = "58"
         End If

         'check for codes larger than 255
         ColonAt = InStr(txtLatin(i), ":")
         If ColonAt = 0 Then
            'check invalid range
            If IsNumeric(txtLatin(i)) Or Left(txtLatin(i), 1) = "$" Then
               If HexToDecimal("" & txtLatin(i)) > MAX_ASCII Then
                  Beep
                  'open misc folder if focus on one of its text fields
                  If i > 65 Then
                     pnlFolderTab_Click (1)
                  End If
                  'highlight invalid entry
                  txtLatin(i).SelStart = 0
                  txtLatin(i).SelLength = Len(txtLatin(i))
```

```
                PassiveHelp ""
                MsgBox ("Simple entries have a valid range of 0 to 255 ($00 to $FF)."), MB_ICONSTOP, ("Invalid
Input")
                txtLatin(i).SetFocus
                Exit Function
             End If
          End If
       'check for valid compound entries
       Else
          'missing first parameter; missing last parameter; invalid first; invalid last
          LPram = Left(txtLatin(i), ColonAt - 1)
          RPram = Mid(txtLatin(i), ColonAt + 1)
          If ColonAt = 1 Or ColonAt = Len(txtLatin(i)) Or HexToDecimal(LPram) > MAX_ASCII Or HexToDecimal(RPram) >
MAX_ASCII Then
             Beep
             'open misc folder if focus on one of its text fields
             If i > 65 Then
                pnlFolderTab_Click (1)
             End If
             'highlight invalid entry
             txtLatin(i).SelStart = 0
             txtLatin(i).SelLength = Len(txtLatin(i))
             PassiveHelp "For example:  0,65 or $00:$41"
             MsgBox ("Compound entries are entered as:  VALUE:VALUE" & Chr(13) & "where VALUE has a valid range of
0 to 255 ($00 to $FF)."), MB_ICONSTOP, ("Invalid Input")
             PassiveHelp ""
             txtLatin(i).SetFocus
             Exit Function
          End If
       End If
    Next i

    'if flow reaches this, then no error was located
    MappingError = False

End Function

'_____________________________________________________________________________________________________
'take character set mappings in edit fields and condense them down to string form.  Tack this
'string to filter name in list box.  This eliminates external data structures and their management
'_____________________________________________________________________________________________________
Sub PackCurrentFilter (MakeDefault As Integer)

    Dim Map As String
    Dim i   As Integer
```

```
    If frmFilterSave.txtFilterName = "" Then
       frmFilterSave.txtFilterName = lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)
    End If

    'if filter already in lists, remove it (serves to update existing filter)
    For i = 0 To lstFilters(FILTER_NAME).ListCount - 1
        If lstFilters(FILTER_NAME).List(i) = frmFilterSave.txtFilterName Then
           lstFilters(FILTER_NAME).RemoveItem i
           lstFilters(FILTER_MAP).RemoveItem i
           Exit For
        End If
    Next i

    'add filter name to list
    lstFilters(FILTER_NAME).AddItem frmFilterSave.txtFilterName

    'create packed mapping
    For i = 0 To NUM_MAPPINGS
        Map = Map & txtLatin(i) & gPackDelimiter
    Next i

    'add filter mapping to list.  Prepend filter name so both lists sorted in same order
    lstFilters(FILTER_MAP).AddItem frmFilterSave.txtFilterName & Space(32 - Len(frmFilterSave.txtFilterName)) & ":"
& Map

    'make just added filter active filter, if desired
    If MakeDefault Then
       For i = 0 To lstFilters(FILTER_NAME).ListCount - 1
           If lstFilters(FILTER_NAME).List(i) = frmFilterSave.txtFilterName Then
              lstFilters(FILTER_NAME).ListIndex = i
              Exit For
           End If
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________
Sub pctExtendedCharacters_Click (Index As Integer)

    'inform of unsupported feature
    Beep
    PassiveHelp "Available in " & PROGRAM_TITLE & " Version 2.0..."
    MsgBox ("These character mappings are reserved for future expansion and are not supported in this version."),
MB_ICONINFORMATION, ("Unsupported Feature")
```

```
    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pctExtendedCharacters_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)

    'show topical help on info line
    PassiveHelp "These character mappings are reserved for future expansion and are not supported in this version."

End Sub

'_____________________________________________________________________________________________________________
Sub pnlFilterName_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select from the available filters."

End Sub

'_____________________________________________________________________________________________________________
Sub pnlFolderTab_Click (Index As Integer)

    'obscure bottom of tab to simulate actively open folder
    pctTabOverlay.Left = pnlFolderTab(Index).Left - pnlMain.Left + 20

    'display contents of selected folder
    pnlPanel(Index).ZOrder 0

    'record folder opened
    If Tag <> "00" Then
       Tag = "" & Index
    End If

    'set focus to first entry field
    If Index Then
       txtLatin(66).SetFocus
    Else
       txtLatin(0).SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
```

```
Sub pnlFolderTab_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help for each tab on info line
    If Index = 0 Then
       PassiveHelp "Display filter mappings for alphabetic characters."
    Else
       PassiveHelp "Display filter mappings for miscellaneous characters."
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub pnlMain_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlPanel_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
'take all packed filter mappings and write to filter file
'_____________________________________________________________________________________________________________
Sub SaveFilters ()

    Dim NumFilters As Integer
    Dim i          As Integer

    On Error GoTo SaveFiltersError


    'determine how many filters to save
```

```
    NumFilters = lstFilters(FILTER_MAP).ListCount - 1

    'open file to write filters
    MousePointerWait
    Open FILTER_FILENAME For Output As 1

    'record which is default filter (ie. current)
    Print #1, lstFilters(FILTER_NAME).ListIndex

    'if any filters to save, then do it; otherwise file is made empty
    If NumFilters >= 0 Then
       For i = 0 To NumFilters
           Write #1, lstFilters(FILTER_MAP).List(i)
       Next i
    End If

    Close 1
    MousePointerContinue

    Exit Sub


SaveFiltersError:

    Close 1
    MousePointerContinue

    Beep

    'prompt user to deal with error
    PassiveHelp "An error has occurred while saving the available filters.  'Retry' will attempt to save them
again."
    MsgboxAnswer = MsgBox("Unable to save the list of available of filters:" & Chr(13) & Error(Err),
MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    PassiveHelp ""
    If MsgboxAnswer = ID_RETRY Then
       Resume
    End If

    Exit Sub

End Sub

'____________________________________________________________________________________________________
Sub txtLatin_Click (Index As Integer)
```

```
    'don't allow read-only filters to be modified
    If       (lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)       =       "Transliterate")       Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)             =             "KOI-7")             Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex)        =        "Native        Format")        Or
(lstFilters(FILTER_NAME).List(lstFilters(FILTER_NAME).ListIndex) = "Phonetic Unique") Then
       Beep
       MsgBox ("This is a built-in filter and cannot be modified.  To make changes to it, save it as a new name,
then modify that filter."), MB_ICONSTOP, ("Illegal Operation")
       cmdSaveAs.SetFocus
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtLatin_KeyDown (Index As Integer, KeyCode As Integer, Shift As Integer)

    'control key in combination with cursor keys selects next cell
    If Shift = 2 Then
       'cursor down moves down one cell
       If KeyCode = 40 Then
          Index = Index + 1
          'wrap around on 'Alphabet' folder
          If Index = 66 Then
             Index = 0
          'wrap around on 'Miscellaneous' folder
          ElseIf Index > NUM_MAPPINGS Then
             Index = 66
          End If
          'go to new cell
          txtLatin(Index).SetFocus
          KeyCode = 0
       ElseIf KeyCode = 38 Then
       'cursor up moves up one cell
          Index = Index - 1
          'wrap around on 'Miscellaneous' folder
          If Index = 65 Then
             Index = NUM_MAPPINGS
          'wrap around on 'Alphabet' folder
          ElseIf Index < 0 Then
             Index = 65
          End If
          'go to new cell
          txtLatin(Index).SetFocus
          KeyCode = 0
```

```
      'cursor left moves left of column
      ElseIf KeyCode = 37 Then
         'wrap around 'Miscellaneous' folder
         If Index > 65 Then
            Index = Index - 11
            'wrap to third column
            If Index < 66 Then
               Index = Index + 33
            End If
         Else
            Index = Index - 11
            'wrap around on 'Alphabet' folder
            If Index < 0 Then
               Index = Index + 66
            End If
         End If
         'go to new cell
         txtLatin(Index).SetFocus
         KeyCode = 0
      'cursor right moves right of column
      ElseIf KeyCode = 39 Then
         'wrap around 'Miscellaneous' folder
         If Index > 65 Then
            Index = Index + 11
            'wrap to first column
            If Index > NUM_MAPPINGS Then
               Index = Index - 33
            End If
         Else
            Index = Index + 11
            'wrap around on 'Alphabet' folder
            If Index > 65 Then
               Index = Index - 66
            End If
         End If
         'go to new cell
         txtLatin(Index).SetFocus
         KeyCode = 0
      End If
   End If

End Sub

'____________________________________________________________________________________________________________
Sub txtLatin_KeyPress (Index As Integer, KeyAscii As Integer)
```

```
    Dim i As Integer


    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       'if not at end of alphabet folder list, go to next field...
       If Index < 65 Then
          Index = Index + 1
      '...otherwise
       Else
          'if in misc folder...
          If Index >= 66 Then
             'if not at end of misc folder, go to next field...
             If Index < NUM_MAPPINGS Then
                Index = Index + 1
             '...otherwise wrap to start of misc folder
             Else
                Index = 66
             End If
          '...otherwise wrap to start of alphabet folder
          Else
             Index = 0
          End If
       End If
       'set focus to new field
       txtLatin(Index).SetFocus
       Exit Sub
    End If

    'allow backspace to be processed normally
    If KeyAscii = 8 Then
       Exit Sub
    End If

    'field cannot start with colon
    If KeyAscii = Asc(":") And Len(txtLatin(Index)) = 0 Then
       Beep
       KeyAscii = 0
       PassiveHelp "The colon is used to separate numeric values in two-byte mappings; for example:   0:65 or
$00:$41."
       MsgBox ("This field cannot begin with a colon.  If you are attempting to enter the colon as a mapping
character, enter 58 instead (i.e. its ASCII code)"), MB_ICONSTOP, ("Invalid Input")
       PassiveHelp ""
       Exit Sub
    End If
```

```
    'if colon present, mask any more colons
    If InStr(txtLatin(Index), ":") And KeyAscii = Asc(":") Then
       Beep
       KeyAscii = 0
       PassiveHelp ""
       MsgBox ("Only one colon is allowed in this field."), MB_ICONSTOP, ("Invalid Input")
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtLatin_LostFocus (Index As Integer)

    'if single space entered,
    If txtLatin(Index) = " " Then
       Beep
       PassiveHelp ""
       MsgBox ("The space character cannot be entered directly.   Its ASCII code will be substituted."),
MB_ICONINFORMATION, ("Automatic Conversion")
       txtLatin(Index) = "32"
    End If

    'remove leading/trailing spaces
    txtLatin(Index) = Trim(txtLatin(Index))

End Sub

'_____________________________________________________________________________________________________________
Sub txtLatin_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Input the letters(s) from the abstract which will be interpreted as the Russian letters shown."

End Sub

'_____________________________________________________________________________________________________________
'(frmHelp): displays topical help from given file
'_____________________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________________
Sub cmdClose_Click ()
```

```
    frmHelp.Hide

End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()

    Dim HelpFunction As String
    Dim Description  As String
    Dim HowTo        As String
    Dim Commands     As String
    Dim FileLine     As String

    On Error GoTo LoadHelpError


    'open help file passed in form tag field
    Open Tag For Input As 10

       'get field info
       Line Input #10, HelpFunction
       Line Input #10, Description

       'get to 'how to' field
       Do
          Line Input #10, FileLine
       Loop Until (FileLine = "~") Or EOF(10)
       'extract lines
       Do
          Line Input #10, FileLine
          If FileLine <> "~" Then
             HowTo = HowTo & FileLine & Chr(13) & Chr(10)
          Else
             Exit Do
          End If
       Loop Until EOF(10)

       'extract 'commands' lines
       Do
          Line Input #10, FileLine
          Commands = Commands & FileLine & Chr(13) & Chr(10)
       Loop Until EOF(10)

    Close 10

    'display info in fields
```

```
    pnlFunction.Caption = HelpFunction
    pnlDescription.Caption = Description
    txtHowDoI.Text = HowTo
    txtHowDoI.SelStart = 0
    txtCommands.Text = Commands
    txtCommands.SelStart = 0

    Exit Sub


LoadHelpError:

    'prompt user to deal with load fail
    Beep
    MousePointerContinue
    MsgBox ("Unable to load help file '" & UCase(Tag) & "':" & Chr(13) & Error(Err) & "."), MB_ICONEXCLAMATION,
("File Error")
    Hide
    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub txtCommands_KeyPress (KeyAscii As Integer)

    'prevent editing of help info
    KeyAscii = 0

End Sub

'_____________________________________________________________________________________________________________
Sub txtCommands_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'prevent editing of help info
    txtCommands.SelLength = 0

End Sub

'_____________________________________________________________________________________________________________
Sub txtHowDoI_KeyPress (KeyAscii As Integer)

    'prevent editing of help info
    KeyAscii = 0

End Sub
```

```
'_____________________________________________________________________________________________________
Sub txtHowDoI_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'prevent editing of help info
    txtHowDoI.SelLength = 0

End Sub

'_____________________________________________________________________________________________________
'(frmLookup): symbolic constant lookup dialog, takes Russian word and shows symbolic constant(s) generated from
known word info
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp "Display help on how to look up a word or phrase."

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    'return to main window
    PassiveHelp ""
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp "Close this dialog and returns to the main interface."

End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()

    'make sure entry field is empty and gets default focus
    txtSearchString.Height = 252
    txtSearchString = ""
    txtSearchString.SetFocus
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub lstResults_DblClick ()

    'insert symbolic constant into russian expression
    frmTransferRuleEntry.txtRussian = frmTransferRuleEntry.txtRussian & Trim(lstResults)

End Sub

'_____________________________________________________________________________________________________________
Sub lstResults_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp "Lists the symbolic constants extracted from the search word or phrase."

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_DragDrop (Source As Control, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub txtSearchString_Change ()

    'clear current list when new search word entered
    lstResults.Clear

End Sub

'_____________________________________________________________________________________________________________
Sub txtSearchString_KeyPress (KeyAscii As Integer)

    Dim Details As TransferType
    Dim Hits    As Integer
    Dim i, j    As Integer


    'carriage return initiates search
    If KeyAscii = 13 Then
       KeyAscii = 0
       'verify something entered
       If txtSearchString = "" Then
```

```
          Beep
          MsgBox ("You must enter a word or phrase to look up."), MB_ICONSTOP, ("Missing Entry")
          Exit Sub
       End If

       'do lookup
       LookupWord ("" & txtSearchString), Details, True

       'if it fails, return message; otherwise...
       If Details.NumMatches = 0 Then
          Beep
          MsgBox ("This word or the given inflection for it is not present in the lexicon."), MB_ICONEXCLAMATION,
("Lookup Failed")
       '...load list box with returned symbolic constants
       Else
          lstResults.Clear
          For i = 1 To Details.NumMatches
              lstResults.AddItem Details.Match(i).LexiconEntry.Target & ":"
              For j = 1 To Details.Match(i).SymbolicConstants.NumSymbolicConstants
                  lstResults.AddItem " " & Details.Match(i).SymbolicConstants.SymbolicConstant(j)
              Next j
              lstResults.AddItem ""
          Next i
       End If

       'highlight lookup word and set focus
       txtSearchString.SelStart = 0
       txtSearchString.SelLength = Len(txtSearchString)
       txtSearchString.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtSearchString_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help in info line
    PassiveHelp "Enter a word or phrase to look up.  To switch keyboard layouts, press CTRL+SHIFT."

End Sub

'_____________________________________________________________________________________________________________
'(frmLookupResults): lists results of lexicon lookup when more than one match found
'_____________________________________________________________________________________________________________
Option Explicit
```

```
'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'return code indicating cancel selected
    Tag = "[CANCEL]"
    timerScroll.Enabled = False
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Abort this lookup and return to the Lexicon Editor."

End Sub

'_____________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP5_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display topical help on multiple lexicon entry lookups."

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    'return code indicating ok selected
    Tag = "[OK]"
    timerScroll.Enabled = False
    Hide

End Sub
```

```
'_____________________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Accept the selected entry and display its information."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set form components
    lstResults(0).Height = 2380
    lstResults(1).Height = 2380
    timerScroll.Enabled = True

    'initialize form return code
    Tag = ""

    'set default to first list element
    lstResults(0).ListIndex = 0
    lstResults(0).SetFocus

End Sub

'_____________________________________________________________________________________________________________
Sub lstResults_Click (Index As Integer)

    'set focus to both lists simultaneously
    lstResults(1 - Index).ListIndex = lstResults(Index).ListIndex

End Sub

'_____________________________________________________________________________________________________________
Sub lstResults_DblClick (Index As Integer)

    'double-click selects current entry and clicks OK
    cmdOK_Click

End Sub

'_____________________________________________________________________________________________________________
Sub lstResults_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'set focus to both lists simultaneously
```

```
    lstResults(1 - Index).ListIndex = lstResults(Index).ListIndex

End Sub

'_____________________________________________________________________________________________________
Sub lstResults_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select the lexicon entry you want to display."

End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'deal with scrolling both lists simultaneously
'_____________________________________________________________________________________________________
Sub timerScroll_Timer ()

    Static PrevTop0 As Integer
    Static PrevTop1 As Integer


    'if first list changed, reposition second
    If lstResults(0).TopIndex <> PrevTop0 Then
       PrevTop0 = lstResults(0).TopIndex
       lstResults(1).TopIndex = lstResults(0).TopIndex
    'if second list changed, reposition first
    ElseIf lstResults(1).TopIndex <> PrevTop1 Then
       PrevTop1 = lstResults(1).TopIndex
       lstResults(0).TopIndex = lstResults(1).TopIndex
    End If

End Sub

'_____________________________________________________________________________________________________
'(frmMain): MDI parent form of entire program.  Home to main interface, menus, and all child forms and dialogs
'_____________________________________________________________________________________________________
Option Explicit
```

```
'_____________________________________________________________________________________________
'determine if any changes made to documents.  If so, prompt to save
'_____________________________________________________________________________________________
Function ConfirmClose () As Integer

    'if changes made to source abstract, prompt to save
    If gAbstractModified Then
       Beep
       MsgBoxAnswer = MsgBox("The converted abstract has been modified." & Chr(13) & Chr(13) & "Do you want to save
the changes?", MB_YESNOCANCEL + MB_ICONEXCLAMATION, "Confirm Close")
       'if confirmed, initiate save procedure; otherwise...
       If MsgBoxAnswer = ID_YES Then
        menuSaveConvertedAbstract_Click
       '...if canceled, fail function
       ElseIf MsgBoxAnswer = ID_CANCEL Then
        ConfirmClose = False
        Exit Function
       End If
    End If

    'if changes made to transliteration, prompt to save
    If gTransliterationModified Then
       Beep
       MsgBoxAnswer = MsgBox("The transliteration of this abstract has been modified." & Chr(13) & Chr(13) & "Do you
want to save the changes?", MB_YESNOCANCEL + MB_ICONEXCLAMATION, "Confirm Close")
       'if confirmed, initiate save procedure; otherwise...
       If MsgBoxAnswer = ID_YES Then
        menuSaveTransliteration_Click
       '...if canceled, fail function
       ElseIf MsgBoxAnswer = ID_CANCEL Then
        ConfirmClose = False
        Exit Function
       End If
    End If

    'if changes made to translation, prompt to save
    If gTranslationModified Then
       Beep
       MsgBoxAnswer = MsgBox("The translation of this abstract has been modified." & Chr(13) & Chr(13) & "Do you
want to save the changes?", MB_YESNOCANCEL + MB_ICONEXCLAMATION, "Confirm Close")
       'if confirmed, initiate save procedure; otherwise...
       If MsgBoxAnswer = ID_YES Then
        menuSaveTranslation_Click
       '...if canceled, fail function
```

```
       ElseIf MsgBoxAnswer = ID_CANCEL Then
         ConfirmClose = False
         Exit Function
       End If
    End If

    'go ahead with close or exit
    ConfirmClose = True

End Function

'_____________________________________________________________________________________________________________
Sub MDIForm_Activate ()

    Static OneShot As Integer
    Dim i As Integer


    'run this code only on startup AFTER screen displayed
    If Not OneShot Then
       OneShot = True

       'display title form while initialization performed
       DoEvents
       MousePointerWait
       frmStartup.Show

       'small delay loop to allow startup form to be drawn
       For i = 1 To 5000
         DoEvents
       Next i

       'get configuration settings
       LoadConfigSettings

       'load default system filter
       PassiveHelp "Loading and unpacking import filters..."
       Load frmFilter

       'set up Cyrillic lookup table for sorting
       DefineSortOrder

       'initialize fixed translation tables
       PassiveHelp "Defining inflection and constant tables..."
       DefineInflectionPatterns
       DefineSymbolicConstantTable
```

```
        'load bilingual lexicon
        LoadLexicon

        'load lexicon editor
        MousePointerWait
        PassiveHelp "Generating lexicon index..."
        DoEvents
        Load frmLexiconEditor

        'load cleanup filters
        LoadCleanupFilters

        'load transfer rules
        LoadTransferRules

        'get rid of title form
        Unload frmStartup

        'go
        PassiveHelp ""
        MousePointerContinue
        menuFile.Enabled = True
        menuEdit.Enabled = True
        menuOptions.Enabled = True
        menuWindow.Enabled = True
        menuHelp.Enabled = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub MDIForm_Load ()

    'delimiter char to separate mapping entries in filter.  This should be a constant, but Visual
    'Basic constant declarations don't allow functions (ie. Chr).  The delimiter must be a non-
    'printable character, so it is defined here
    gPackDelimiter = Chr(7)
    DoEvents

End Sub

'_____________________________________________________________________________________________________________
Sub MDIForm_Resize ()

    'set form colors according to user preferences
```

```
    pnlStatusBar.BackColor = BACKGROUND_COLOR
    pnlInfo.BackColor = BACKGROUND_COLOR

    'place status bar at bottom of main window and scale to current screen dimensions
    pnlStatus.Width = ScaleWidth * .25
    pnlStatus.Left = ScaleWidth - pnlStatus.Width - 30
    pnlInfo.Width = pnlStatus.Left - 60
    pnlStatusBar.Top = 350

    'preload certain forms to avoid update delays
    Load frmAbout
    DoEvents

    'revert mouse pointer from hourglass back to arrow
    MousePointerContinue

End Sub

'_____________________________________________________________________________________________________________
Sub MDIForm_Unload (Cancel As Integer)

    'intercept mouse click on close icon and route control to menu option Exit
    menuExit_Click

End Sub

'_____________________________________________________________________________________________________________
Sub menuAbout_Click ()

    'display 'About' form modally
    frmAbout.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuArrangeIcons_Click ()

    'arrange minimized child forms on MDIParent
    frmMain.Arrange 3

End Sub

'_____________________________________________________________________________________________________________
Sub menuCascade_Click ()
```

```
    'cascade child forms on MDIParent
    frmMain.Arrange 0

End Sub

'_____________________________________________________________________________________________________________
Sub menuCleanupFilterEditor_Click ()

    'bring up cleanup filter editor
    frmCleanupFilterEditor.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuCloseAbstract_Click ()

    'prompt to save any unsaved changes
    If ConfirmClose() Then
       'make functions unavailable
       menuCloseAbstract.Enabled = False
       menuPrint.Enabled = False
       menuSave.Enabled = False

       'reset globals
       ResetAbstract

       'close forms
       Unload frmTransliteration
       Unload frmAbstract
       Unload frmTranslation
       Unload frmTranslationDetails
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub menuContents_Click ()

    Beep
    MsgBox ("Sorry, the on-line help is not available in this version." & Chr(13) & "See the thesis text for further
information."), MB_ICONINFORMATION, ("No Help")

End Sub

'_____________________________________________________________________________________________________________
```

```
Sub menuCustomize_Click ()

    'bring up customize dialog
    frmCustomize.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuExit_Click ()

    'prompt to save any unsaved changes
    If ConfirmClose() Then
       'terminate
       End
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub menuImportFilterSetup_Click ()

    MousePointerWait

    'set 'Alphabet' folder to default. Double-zero indicates set focus on filter form to filter list
    frmFilter.Tag = "00"

    'call up filter editor
    frmFilter.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuLexiconEditor_Click ()

    'call up lexicon editor form
    MousePointerWait
    frmLexiconEditor.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuOpenAbstract_Click ()

    On Error GoTo ErrorOnOpen
```

```
    Dim InString    As String
    Dim LineBreak   As String
    Dim AbstractLen As Long


    'confirm save changes before closing to open new abstract
    menuCloseAbstract_Click

    'call up common dialog to open abstract file
    CMDialog.Flags   =   CMFILEDIALOG_HIDEREADONLY   Or   CMFILEDIALOG_FILEMUSTEXIST   Or   CMFILEDIALOG_SHOWHELP   'Or
CMFILEDIALOG_NOVALIDATE
    CMDialog.Filename = ""

    'determine how to format abstract:
    'translation mode expects .abs files and strips line breaks
    If gOperationMode = ABSTRACT_MODE Then
       LineBreak = " "
       CMDialog.DialogTitle = "Open Abstract"
       CMDialog.DefaultExt = "abs"
       CMDialog.Filter  =  "Source  Abstract  (*.abs)|*.abs|Converted  Abstract  (*.cvt)|*.cvt|Text  (*.txt)|*.txt|All
Files|*.*"
    'email mode expects .txt and retains line breaks
    Else
       LineBreak = Chr(13)
       CMDialog.DialogTitle = "Open E-Mail Message"
       CMDialog.DefaultExt = "txt"
       CMDialog.Filter   =   "E-mail   Message   (*.txt)|*.txt|Source   Abstract   (*.abs)|*.abs|Converted   Abstract
(*.cvt)|*.cvt|Text (*.txt)|*.txt|All Files|*.*"
    End If

    CMDialog.Action = 1

    'unload current abstract forms
    Unload frmAbstract
    Unload frmTransliteration

    'attempt to access file; if this fails, error handler takes over
    MousePointerWait
    PassiveHelp "Opening '" & CMDialog.Filetitle & "'..."
    Open (CMDialog.Filename) For Input As 1

    PassiveHelp "Reading '" & CMDialog.Filetitle & "'..."
    'check length of abstract
    AbstractLen = FileLen(CMDialog.Filename)
    'if it's too big, abort
    If AbstractLen > MAX_ABSTRACT_CHARS Then
```

```
      Error 14
   End If

   'read abstract in original format
   gRawAbstract = ""
   Do Until EOF(1)
      Line Input #1, InString
      gRawAbstract = gRawAbstract & InString & LineBreak
   Loop
   Close 1
   gRawAbstract = gRawAbstract & LineBreak

   'assume conversion success until shown otherwise
   frmConvert.Tag = "[OK]"

   'apply default filter to convert abstract to internal format
   gAbstract = ConvertAbstract(StripWhitespace(gRawAbstract), True, "Applying the default filter to convert the
abstract...")

   'verify that conversion did not detect any unmapped characters.  Then check first few words in
   'abstract against lexicon to determine if actual Russian words are generated by conversion.
   If Not ConversionOK(gAbstract) Then
      'if this fails, get input from user on which filter(s) to apply
      frmConvert.Show 1
   End If

   'if conversion successful, bring up abstract
   If frmConvert.Tag = "[OK]" Then
      'set current document name
      gAbstractFilename = CMDialog.Filename
      gAbstractTitle = UCase(CMDialog.Filetitle)
      gAbstractDate = FileDateTime(gAbstractFilename)
      Caption = PROGRAM_TITLE & "  [" & gAbstractTitle & "]"

      'display abstract in source abstract window
      frmAbstract.txtAbstract = gAbstract
      frmAbstract.Show

      'enable Cascade, Tile, etc.
      MDIChildControls (True)

      'make functions available
      menuCloseAbstract.Enabled = True
      menuPrint.Enabled = True
      menuSave.Enabled = True
   Else
```

```
        'blank current document name
        gAbstractFilename = ""
        gAbstractTitle = ""
        Caption = PROGRAM_TITLE

        'make functions unavailable
        menuCloseAbstract.Enabled = False
        menuPrint.Enabled = False
        menuSave.Enabled = False

        'disable Cascade, Tile, etc.
        MDIChildControls (False)
    End If

    MousePointerContinue

    Exit Sub



ErrorOnOpen:

    Close 1
    MousePointerContinue
    pnlStatus.FloodPercent = 0

    'if 'Cancel' clicked, abort Open...
    If Err = CMDIALOG_CANCEL_ERROR Then
        Exit Sub

    '...otherwise inform of error
    Else
        Beep
        PassiveHelp ""
        Select Case Err
             'file is too big
             Case 14 'out of string space
                 MsgBox ("This file is too big to be an abstract." & Chr(13) & "It cannot be translated."),
MB_ICONSTOP, ("Invalid File")
                 Exit Sub
             'file not found
             Case FILEACCESS_FILENOTFOUND:
                 MsgBox ("The requested abstract file '" & CMDialog.Filename & "' cannot be found." & Chr(13) & "Make
sure you have specified the correct path and filename."), MB_ICONEXCLAMATION, ("File Error")
                  'open common dialog again
                  menuOpenAbstract_Click
```

```
               Exit Sub
            'path not found
            Case FILEACCESS_PATHNOTFOUND:
               MsgBox ("The requested directory cannot be found." & Chr(13) & "Make sure you have specified the
correct path."), MB_ICONEXCLAMATION, ("File Error")
               'open common dialog again
               menuOpenAbstract_Click
               Exit Sub
            'generic error message
            Case Else:
               MsgBoxAnswer = MsgBox("Upon trying to open the abstract file '" & CMDialog.Filename & "', a " &
LCase(Error) & " error occurred.", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
               'if user wants to try again, attempt to re-execution operation...
               If MsgBoxAnswer = ID_RETRY Then
                  Resume
               '...otherwise abort operation
               Else
                  'open common dialog again
                  Exit Sub
               End If
      End Select
   End If

   Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub menuPageSetup_Click ()

   On Error GoTo ErrorOnSetup


   'call up common dialog to set print up
   CMDialog.Flags = CMPRINTDIALOG_PRINTSETUP
   CMDialog.DialogTitle = "Print"
   CMDialog.Action = 5

   Exit Sub



ErrorOnSetup:

   'if 'Cancel' clicked, abort setup...
   If Err = CMDIALOG_CANCEL_ERROR Then
```

```
       Exit Sub

    '...otherwise inform of error
    Else
       Beep
       PassiveHelp ""
       MsgBoxAnswer = MsgBox("Upon trying to set up the printer, a " & LCase(Error) & " error occurred.",
MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
       'if user wants to try again, attempt to re-execution operation...
       If MsgBoxAnswer = ID_RETRY Then
        Resume
       End If
    End If

    Exit Sub

End Sub

'_____________________________________________________________________________________________________
Sub menuPostEditTranslation_Click ()

    Dim Status As Long

    On Error GoTo ShellError


    'copy translation
    frmTranslation.txtTranslation.SelStart = 0
    frmTranslation.txtTranslation.SelLength = Len(frmTranslation.txtTranslation)
    SendKeys "^C", True
    frmTranslation.txtTranslation.SelLength = 0

    'attempt to launch selected editor
    MousePointerWait
    Status = Shell(gPostEditorFilename, 3)
    DoEvents

    'paste translation into newly opened document
    SendKeys gPostEditorPasteSequence, True
    MousePointerContinue

    Exit Sub


ShellError:
```

```
    'error handler
    Beep
    MousePointerContinue
    MsgBox ("Unable to open " & gPostEditorAPIName & " (" & UCase(gPostEditorFilename) & "):" & Chr(13) &
Error(Err)), MB_ICONSTOP, ("Post-Edit Failure")
    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub menuPrintConvertedAbstract_Click ()

    'print converted abstract
    PrintDocuments (CONVERTED_ABSTRACT)

End Sub

'_____________________________________________________________________________________________________________
Sub menuPrintTranslation_Click ()

    'print translation
    PrintDocuments (TRANSLATION)

End Sub

'_____________________________________________________________________________________________________________
Sub menuPrintTranslationDetails_Click ()

    'print transfer details
    PrintDocuments (TRANSLATION_DETAILS)

End Sub

'_____________________________________________________________________________________________________________
Sub menuPrintTransliteration_Click ()

    'print transliteration
    PrintDocuments (TRANSLITERATION)

End Sub

'_____________________________________________________________________________________________________________
Sub menuSaveConvertedAbstract_Click ()

    On Error GoTo ErrorOnSaveConvertedAbstract
```

```
    'call up common dialog to save converted abstract file
    CMDialog.Flags = CMFILEDIALOG_SHOWHELP Or CMFILEDIALOG_OVERWRITEPROMPT Or CMFILEDIALOG_HIDEREADONLY
    CMDialog.DialogTitle = "Save Converted Abstract"
    CMDialog.Filename = StripExtension("" & CMDialog.Filetitle)
    CMDialog.DefaultExt = "cvt"
    CMDialog.Filter = "Converted Abstract (*.cvt)|*.cvt|Text (*.txt)|*.txt|All Files|*.*"
    CMDialog.Action = 2

    'create file; if this fails, error handler takes over
    MousePointerWait
    PassiveHelp "Opening '" & CMDialog.Filetitle & "'..."
    Open (CMDialog.Filename) For Output As 1
    PassiveHelp "Writing '" & CMDialog.Filetitle & "'..."
    Print #1, frmAbstract.txtAbstract
    Close 1
    PassiveHelp ""
    MousePointerContinue

    Exit Sub



ErrorOnSaveConvertedAbstract:

    Close 1
    MousePointerContinue

    'if 'Cancel' clicked, abort Open...
    If Err = CMDIALOG_CANCEL_ERROR Then
       Exit Sub

    '...otherwise inform of error
    Else
       Beep
       PassiveHelp ""
       MsgBoxAnswer = MsgBox("Upon trying to save the converted abstract file '" & CMDialog.Filename & "', a " &
LCase(Error) & " error occurred.", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
       'if user wants to try again, attempt to re-execution operation
       If MsgBoxAnswer = ID_RETRY Then
        Resume
       End If
    End If

    Exit Sub
```

```
End Sub

'_____________________________________________________________________________________________________
Sub menuSaveTranslation_Click ()

    On Error GoTo ErrorOnSaveTranslation


    'call up common dialog to save converted abstract file
    CMDialog.Flags = CMFILEDIALOG_SHOWHELP Or CMFILEDIALOG_OVERWRITEPROMPT Or CMFILEDIALOG_HIDEREADONLY
    CMDialog.DialogTitle = "Save Translation"
    CMDialog.Filename = StripExtension("" & CMDialog.Filetitle)
    CMDialog.DefaultExt = "trn"
    CMDialog.Filter = "Translation (*.trn)|*.trn|Text (*.txt)|*.txt|All Files|*.*"
    CMDialog.Action = 2

    'create file; if this fails, error handler takes over
    MousePointerWait
    PassiveHelp "Opening '" & CMDialog.Filetitle & "'..."
    Open (CMDialog.Filename) For Output As 1
    PassiveHelp "Writing '" & CMDialog.Filetitle & "'..."
    'Print #1, frmSourceAbstract.txtAbstract
    Close 1
    PassiveHelp ""
    MousePointerContinue

    Exit Sub



ErrorOnSaveTranslation:

    Close 1
    MousePointerContinue

    'if 'Cancel' clicked, abort Open...
    If Err = CMDIALOG_CANCEL_ERROR Then
       Exit Sub

    '...otherwise inform of error
    Else
       Beep
       PassiveHelp ""
       MsgBoxAnswer = MsgBox("Upon trying to save the translation file '" & CMDialog.Filename & "', a " &
LCase(Error) & " error occurred.", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
```

```
       'if user wants to try again, attempt to re-execution operation
       If MsgBoxAnswer = ID_RETRY Then
        Resume
       End If
    End If

    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub menuSaveTranslationDetails_Click ()

    On Error GoTo ErrorOnSaveTranslationDetails


    'call up common dialog to save translation details file
    CMDialog.Flags = CMFILEDIALOG_SHOWHELP Or CMFILEDIALOG_OVERWRITEPROMPT Or CMFILEDIALOG_HIDEREADONLY
    CMDialog.DialogTitle = "Save Translation Details"
    CMDialog.Filename = StripExtension("" & CMDialog.Filetitle)
    CMDialog.DefaultExt = "tdt"
    CMDialog.Filter = "Translation Details (*.tdt)|*.tdt|Text (*.txt)|*.txt|All Files|*.*"
    CMDialog.Action = 2

    'copy log file; if this fails, error handler takes over
    MousePointerWait
    PassiveHelp "Opening '" & CMDialog.Filetitle & "'..."
    FileCopy DETAILS_LOG_FILENAME, CMDialog.Filename
    PassiveHelp "Writing '" & CMDialog.Filetitle & "'..."
    PassiveHelp ""
    MousePointerContinue

    Exit Sub



ErrorOnSaveTranslationDetails:

    MousePointerContinue

    'if 'Cancel' clicked, abort Open...
    If Err = CMDIALOG_CANCEL_ERROR Then
       Exit Sub

    '...otherwise inform of error
    Else
```

```
        Beep
        PassiveHelp ""
        MsgBoxAnswer = MsgBox("Upon trying to save the translation details file '" & CMDialog.Filename & "', a " &
LCase(Error) & " error occurred.", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
        'if user wants to try again, attempt to re-execution operation
        If MsgBoxAnswer = ID_RETRY Then
          Resume
        End If
    End If

    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub menuSaveTransliteration_Click ()

    On Error GoTo ErrorOnSaveTransliteration


    'call up common dialog to save transliteration file
    CMDialog.Flags = CMFILEDIALOG_SHOWHELP Or CMFILEDIALOG_OVERWRITEPROMPT Or CMFILEDIALOG_HIDEREADONLY
    CMDialog.DialogTitle = "Save Transliteration"
    CMDialog.Filename = StripExtension("" & CMDialog.Filetitle)
    CMDialog.DefaultExt = "tlt"
    CMDialog.Filter = "Transliteration (*.tlt)|*.tlt|Text (*.txt)|*.txt|All Files|*.*"
    CMDialog.Action = 2

    'create file; if this fails, error handler takes over
    MousePointerWait
    PassiveHelp "Opening '" & CMDialog.Filetitle & "'..."
    Open (CMDialog.Filename) For Output As 1
    PassiveHelp "Writing '" & CMDialog.Filetitle & "'..."
    Print #1, frmTransliteration.txtTransliteration
    Close 1
    PassiveHelp ""
    MousePointerContinue

    Exit Sub



ErrorOnSaveTransliteration:

    Close 1
    MousePointerContinue
```

```
    'if 'Cancel' clicked, abort Open...
    If Err = CMDIALOG_CANCEL_ERROR Then
       Exit Sub

    '...otherwise inform of error
    Else
       Beep
       PassiveHelp ""
       MsgBoxAnswer = MsgBox("Upon trying to save the transliteration file '" & CMDialog.Filename & "', a " &
LCase(Error) & " error occurred.", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
       'if user wants to try again, attempt to re-execution operation
       If MsgBoxAnswer = ID_RETRY Then
        Resume
       End If
    End If

    Exit Sub

End Sub

'_____________________________________________________________________________________________________________
Sub menuSearchForHelpOn_Click ()

    Beep
    MsgBox ("Sorry, the on-line help is not available in this version." & Chr(13) & "See the thesis text for further
information."), MB_ICONINFORMATION, ("No Help")

End Sub

'_____________________________________________________________________________________________________________
Sub menuSummaryInfo_Click ()

    'call up summary form
    frmSummaryInfo.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuTileHorizontally_Click ()

    'tile child forms horizontally on MDIParent
    frmMain.Arrange 2

End Sub
```

```
'_____________________________________________________________________________________________________
Sub menuTileVertically_Click ()

    'tile child forms vertically on MDIParent
    frmMain.Arrange 1

End Sub

'_____________________________________________________________________________________________________
Sub menuTransferDetails_Click ()

    Dim Detail As String

    On Error GoTo ErrorOnOpenTranslationDetails


    'clear old details
    frmTranslationDetails.lstDetails.Clear

    'load current details from log file
    Open DETAILS_LOG_FILENAME For Input As 5
    Do Until EOF(5)
       Input #5, Detail
       frmTranslationDetails.lstDetails.AddItem Detail
    Loop
    Close 5

    'display transfer details
    frmTranslationDetails.Show

    Exit Sub


ErrorOnOpenTranslationDetails:

    'prompt user to deal with read fail
    Beep
    Close 5
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to read the transfer details log file '" & UCase(DETAILS_LOG_FILENAME) & "':" &
Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to open it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'otherwise abort log, but continue translation
    Else
```

```
       frmTranslationDetails.lstDetails.AddItem "No transfer details available."
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub menuTransferRuleEditor_Click ()

    'bring up transfer rule editor
    frmTransferRuleEditor.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub menuTranslate_Click ()

    'translate current document
    Translate

    'display translation window
    frmTranslation.Show

End Sub

'_____________________________________________________________________________________________________________
Sub menuTransliterate_Click ()

    Dim TRANSLITERATION As String


    'attempt to transliterate Russian abstract text
    TRANSLITERATION = Transliterate(gAbstract)

    'if successful, display; otherwise...
    If TRANSLITERATION <> "" Then
       frmTransliteration.txtTransliteration = TRANSLITERATION
       frmTransliteration.Show

    '...inform user of failure
    Else
       Beep
       MsgBox ("Unabled to transliterate the Russian abstract."), MB_ICONEXCLAMATION, ("Transliteration Failure")
    End If

End Sub
```

```
'_____________________________________________________________________________________________________
Sub pnlStatusBar_MouseMove (Button As Integer, Shift As Integer, x As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub pnlTime_MouseMove (Button As Integer, Shift As Integer, x As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'format and print selected documents
'_____________________________________________________________________________________________________
Sub PrintDocuments (DocumentCode As Integer)

    Dim Detail   As String
    Dim Printing As String

    Const TOP_MARGIN = 1000
    Const LEFT_MARGIN = 1000
    Const LINE_WIDTH = 9100

    On Error GoTo ErrorOnPrint


    'determine which document to print
    Select Case DocumentCode

          'english translation
          Case TRANSLATION:
              Printing = "the English translation"

          'latin transliteration
          Case TRANSLITERATION:
              Printing = "the transliterated text"

              'set page attributes
              Printer.FontName = "Arial"
              Printer.FontSize = 13
```

```
    'print header
    Printer.CurrentX = LEFT_MARGIN
    Printer.CurrentY = TOP_MARGIN
    Printer.Print "Abstract ["; gAbstractTitle; "]"
    Printer.CurrentX = LEFT_MARGIN
    Printer.Print "Transliterated Text:"
    Printer.FontSize = 12
    Printer.Print

    'format margins and print transliteration
    WordWrap ("" & frmTransliteration.txtTransliteration), LEFT_MARGIN, TOP_MARGIN, LINE_WIDTH

    'spool output
    Printer.EndDoc

'russian abstract
Case CONVERTED_ABSTRACT:
    Printing = "the Russian abstract"

    'set page attributes
    Printer.FontName = "Arial"
    Printer.FontSize = 13

    'print header
    Printer.CurrentX = LEFT_MARGIN
    Printer.CurrentY = TOP_MARGIN
    Printer.Print "Abstract ["; gAbstractTitle; "]"
    Printer.FontSize = 12
    Printer.Print
    Printer.FontName = "K8 Arial"

    'format margins and print abstract
    WordWrap ("" & frmAbstract.txtAbstract), LEFT_MARGIN, TOP_MARGIN, LINE_WIDTH

    'spool output
    Printer.EndDoc

'transfer details
Case TRANSLATION_DETAILS
    Printing = "the transfer details"

    'set page attributes
    Printer.FontName = "K8 Kurier"
    Printer.FontSize = 10

    'dump details log to printer
```

```
            MousePointerWait
            Open DETAILS_LOG_FILENAME For Input As 5
            Do Until EOF(5)
               Input #5, Detail
               Printer.Print Detail
            Loop
            Close 5

            'spool output
            Printer.EndDoc
            MousePointerContinue
    End Select

    DoEvents

    Exit Sub



ErrorOnPrint:

   Beep
   Close 5
   MousePointerContinue

   MsgBoxAnswer = MsgBox("Upon trying to print " & Printing & ", a " & LCase(Error) & " error occurred.",
MB_ICONEXCLAMATION + MB_ABORTRETRYIGNORE + MB_DEFBUTTON2, "File Error")
   'if user wants to try again, attempt to re-execution operation...
   If MsgBoxAnswer = ID_RETRY Then
      Resume
      '...abort skips this document and goes to next, if any...
   ElseIf MsgBoxAnswer = ID_IGNORE Then
      Resume Next
      '...otherwise abort operation
   Else
      Exit Sub
   End If

End Sub

'_____________________________________________________________________________________________________
'when abstract closed or new loaded, reinitialize globals
'_____________________________________________________________________________________________________
Sub ResetAbstract ()

    'reset globals
```

```
    gAbstractFilename = ""
    gAbstractTitle = ""
    gAbstractDate = ""
    gAbstractWords = 0
    gLookups = 0
    gLookupHits = 0
    gTransferAttempts = 0
    gTransferHits = 0
    gEventTime = ""
    gEventTick = 0
    gAbstractModified = False
    gTransliterationModified = False
    gTranslationModified = False

    'reset interface
    frmMain.Caption = PROGRAM_TITLE

End Sub

'_____________________________________________________________________________________________________
'strip extension from filename
'_____________________________________________________________________________________________________
Function StripExtension (FilenameString As String) As String

    StripExtension = Left(FilenameString, InStr(FilenameString & ".", ".") - 1)

End Function

'_____________________________________________________________________________________________________
Sub timerTranslation_Timer ()

    'update clock to time translation events
    gEventTick = gEventTick + 1
    gEventTime = Mid(TimeSerial(0, 0, gEventTick), 5)

End Sub

'_____________________________________________________________________________________________________
'(frmFilterSave): gets new or renamed filter name from user
'_____________________________________________________________________________________________________
Option Explicit


'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()
```

```
    'return to filter form with cancel flag indicating do not except file name just entered
    frmFilterSave.Tag = "[CANCEL]"
    frmFilterSave.Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Ignore this entry and leave the filter name as it already is (i.e. '" &
frmFilter.lstFilters(FILTER_NAME).Text & "')."

End Sub

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    Dim NumFilters As Integer
    Dim i          As Integer


    'remove leading/trailing spaces from filter name
    frmFilterSave.txtFilterName = Trim(frmFilterSave.txtFilterName)

    'verify that this filter name isn't already is use
    NumFilters = frmFilter.lstFilters(FILTER_NAME).ListCount
    If NumFilters > 0 Then
       For i = 0 To NumFilters
           'if filter name on list, then this is a duplicate, so inform user and abort
           If LCase(txtFilterName) = LCase(frmFilter.lstFilters(FILTER_NAME).List(i)) Then
              Beep
              PassiveHelp ""
              MsgBox ("This filter name is already on the list of available filters.  Duplicate names are not
allowed.  To save any changes to this filter, click OK on the Import Filter Editor screen."), MB_ICONSTOP,
("Duplicate Entry")
              txtFilterName.SetFocus
              Exit Sub
           End If
       Next i
    End If

    'return to filter form with success flag indicating filter name just entered is to be processed
    frmFilterSave.Tag = "[OK]"
    frmFilterSave.Hide
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Assign this name to the currently displayed filter."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set default name to selected filter name.  Allows user easily to append 2, 3, etc. to make
    'versions of filters.  If user decides not to use current name, upon entering first character
    'of new name, current name disappears
    txtFilterName = frmFilter.lstFilters(FILTER_NAME).List(frmFilter.lstFilters(FILTER_NAME).ListIndex)
    txtFilterName.SelStart = 0
    txtFilterName.SelLength = txtFilterName.MaxLength

    'adjust 3d text box to current system configurations
    pnlText.Height = txtFilterName.Height + 50
    pnlText.Width = txtFilterName.Width + 50

    'focus is on entry field
    txtFilterName.SetFocus

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub txtFilterName_Change ()

    'ok button enabled only when filter name field not empty.  Prevents user from clicking OK if
    'nothing has been entered
    cmdOK.Enabled = Len(txtFilterName) > 0
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub txtFilterName_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtFilterName_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Enter a new name for the currently displayed filter."

End Sub

'_____________________________________________________________________________________________________________
'(frmAbstract): displays converted abstract and provides simple editing operations
'_____________________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________________
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    'escape key aborts translation
    If KeyCode = 27 Then
       gAbort = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Load ()

    'size window to portion of parent form
    Height = Screen.Height / 1.75
    Width = Screen.Width / 1.28

    'position offset slightly from top-left
    Left = Screen.Width * .01
    Top = Screen.Height * .01
```

```
    'enable functions available when source abstract form loaded
    frmMain.menuTranslate.Enabled = True
    frmMain.menuTransliterate.Enabled = True
    frmMain.menuSaveConvertedAbstract.Enabled = True
    frmMain.menuPrintConvertedAbstract.Enabled = True
    frmMain.menuTranslate.Enabled = True

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Resize ()

    'fit display field vertically in window
    If Height > 450 Then
       pnlRoot.Height = Height - 450
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Unload (Cancel As Integer)

    'make functions unavailable when source abstract form unloaded
    frmMain.menuTranslate.Enabled = False
    frmMain.menuTransliterate.Enabled = False
    frmMain.menuPrintConvertedAbstract.Enabled = False
    frmMain.menuSaveConvertedAbstract.Enabled = False

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub txtAbstract_Change ()

    Static OneShot As Integer
```

```
    'update changes made
    gAbstract = txtAbstract


    'ignore first change  when text loaded
    If Not OneShot Then
       OneShot = True
       Exit Sub
    Else
       'indicate that changes have been made
       gAbstractModified = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtAbstract_KeyDown (KeyCode As Integer, Shift As Integer)

    'escape key aborts translation
    If KeyCode = 27 Then
       gAbort = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtAbstract_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on abstract window
    PassiveHelp "This windows shows the source abstract in Russian."

End Sub

'_____________________________________________________________________________________________________________
'(frmStartup): shows title screen at startup while lexicon, etc. being loaded.
'_____________________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set form colors according to user preferences
    pnlRoot.BackColor = BACKGROUND_COLOR
    pnlTitle.BackColor = BACKGROUND_COLOR
```

```
    'center modal About form on MDI parent space
    Left = Screen.Width / 2 - Width / 2
    Top = Screen.Height / 2 - Height / 2
    lblVersion = "Version " & VERSION_NUMBER

End Sub

'_____________________________________________________________________________________________________
'(frmSummary): displays system performance info on current document and history of all translations done
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub cmdOK_Click ()

    'return to main interface
    PassiveHelp ""
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()

    Dim LookupRate As Integer

    On Error Resume Next


    'display help
    PassiveHelp "Summary information about the current abstract and system performance."

    'update statistics
    If gAbstractTitle <> "" Then
       lblFilename.Caption = gAbstractTitle
       lblCreated.Caption = gAbstractDate
    Else
       lblFilename.Caption = "n/a"
       lblCreated.Caption = "n/a"
    End If

    lblLexiconEntries.Caption = gLexiconEndPtr + 1
    lblTransferRules.Caption = gTransferRulePtr + 1
    lblImportFilter.Caption = frmFilter.lstFilters(0).List(frmFilter.lstFilters(1).ListIndex)
```

```
    'total transfer time
    If gEventTime <> "" Then
       lblTotalTransferTime.Caption = gEventTime
    Else
       lblTotalTransferTime.Caption = "n/a"
    End If

    'lookup rate on this abstract
    If gLookups Then
       LookupRate = (gLookupHits / gLookups) * 100
       lblLookupHits.Caption = gLookupHits & " of " & gLookups & " (" & LookupRate & "%)"
    Else
       LookupRate = 0
       lblLookupHits.Caption = "n/a"
    End If

    'transfer rate on this abstract
    If gTransferAttempts Then
       LookupRate = (gTransferHits / gTransferAttempts) * 100
       lblAverageTransferTime.Caption = Mid(TimeSerial(0, 0, gEventTick / gTransferHits), 5)
       lblTransferHits.Caption = gTransferHits & " of " & gTransferAttempts & " (" & LookupRate & "%)"
    Else
       LookupRate = 0
       lblTransferHits.Caption = "n/a"
       lblAverageTransferTime.Caption = "n/a"
    End If

End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'(frmTransferRuleEditor): simple editing environment for transfer rules
'_____________________________________________________________________________________________________
Option Explicit

Dim fChangesMade As Integer    'flag indicating whether changes made
Dim fRegenerate  As Integer    'flag indicating that rules should be regenerated upon close

Const PACK_DELIMITER = "   "   'delimiter between transfer rule components
```

```
'_____________________________________________________________________________________________________________
Sub cmdAdd_Click ()

    'init and bring up transfer rule expression dialog editor
    frmTransferRuleEntry.txtRussian = ""
    frmTransferRuleEntry.txtEnglish = ""
    frmTransferRuleEntry.txtDescription = ""
    frmTransferRuleEntry.Show 1

    'if expression not canceled, add it to filter lists
    If frmTransferRuleEntry.Tag <> "[CANCEL]" Then
       lstTransferRule(0).AddItem PackRule(frmTransferRuleEntry.txtRussian)
       lstTransferRule(1).AddItem PackRule(frmTransferRuleEntry.txtEnglish)
       lstTransferRule(2).AddItem frmTransferRuleEntry.txtDescription
       'update editor
       cmdCancel.Enabled = True
       fChangesMade = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdAdd_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Add a new transfer rule."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'if changes made, prompt to confirm cancel
    If fChangesMade Then
       Beep
       MsgBoxAnswer = MsgBox("Cancel will discard all the changes made." & Chr(13) & "Continue?", MB_ICONQUESTION +
MB_YESNO + MB_DEFBUTTON2, "Confirm Cancel")
       If MsgBoxAnswer = ID_NO Then
          Exit Sub
       End If
    End If

    'overwrite changes
    LoadTransferRules
```

```
    'close transfer rule editor and return to main interface
    timerScroll.Enabled = False
    cmdCancel.Enabled = False
    PassiveHelp ""
    Unload frmTransferRuleEditor

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Discard any changes made here and return to the main interface."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdClose_Click ()

    Dim UpdateAt     As Single
    Dim RulePtr      As Integer
    Dim RuleCount    As Integer
    Dim ComponentPtr As Integer
    Dim DelimiterAt  As Integer
    Dim i            As Integer

    Static Component(MAX_SENTENCE_COMPONENTS) As String

    On Error GoTo ErrorOnTransferRuleWrite


    'update only if changes made
    If fChangesMade Then
       fChangesMade = False
       'determine update frequency
       UpdateAt = (gTransferRulePtr + 10) / 10

       'display status info during store
       DisplayWait -(gTransferRulePtr / UpdateAt), "Storing the transfer rules..."

       'update changes
       MousePointerWait
       Open TRANSFER_RULE_FILENAME For Output As 1

       'write number of rules
       Print #1, lstTransferRule(0).ListCount - 1
```

```
       'write rules
       For RulePtr = 0 To lstTransferRule(0).ListCount - 1

           'unpack russian rule
           ComponentPtr = 1
           RuleCount = 0
           'separate each component
           Do
              DelimiterAt = InStr(ComponentPtr, lstTransferRule(0).List(RulePtr), PACK_DELIMITER)
              'got another component; otherwise...
              If DelimiterAt Then
                 Component(RuleCount)  =  Trim(Mid(lstTransferRule(0).List(RulePtr),  ComponentPtr,  DelimiterAt  -
ComponentPtr))
                 ComponentPtr = DelimiterAt + Len(PACK_DELIMITER)
                 RuleCount = RuleCount + 1
              '...done
              Else
                 Exit Do
              End If
           Loop

           'record number of components in russian rule
           Print #1, RuleCount - 1; ",";
           For i = 0 To RuleCount - 1
               Print #1, Chr(34); Component(i); Chr(34); ",";
           Next i

           'unpack english rule
           ComponentPtr = 1
           RuleCount = 0
           'separate each component
           Do
              DelimiterAt = InStr(ComponentPtr, lstTransferRule(1).List(RulePtr), PACK_DELIMITER)
              'got another component; otherwise...
              If DelimiterAt Then
                 Component(RuleCount)  =  Trim(Mid(lstTransferRule(1).List(RulePtr),  ComponentPtr,  DelimiterAt  -
ComponentPtr))
                 ComponentPtr = DelimiterAt + Len(PACK_DELIMITER)
                 RuleCount = RuleCount + 1
              '...done
              Else
                 Exit Do
              End If
           Loop
```

```
            'record number of components in english rule
            Print #1, RuleCount - 1; ",";
            For i = 0 To RuleCount - 1
                Print #1, Chr(34); Component(i); Chr(34); ",";
            Next i

            'record comments
            Write #1, lstTransferRule(2).List(RulePtr)

            'update status periodically
            If RulePtr Mod UpdateAt = 0 Then
               DisplayWait 1, ""
            End If

        Next RulePtr
        Close 1

        'hide status
        DisplayWait 0, ""

        'regenerate rules if control+r pressed during session
        If fRegenerate Then
           LoadTransferRules
        End If

        MousePointerContinue
    End If

    'close transfer rule editor and return to main interface
    timerScroll.Enabled = False
    cmdCancel.Enabled = False
    PassiveHelp ""

    'close editor
    Unload frmTransferRuleEditor

    Exit Sub


ErrorOnTransferRuleWrite:

    'prompt user to deal with store fail
    Beep
    Close 1
    MousePointerContinue
```

```
    MsgBoxAnswer = MsgBox("Unable to write to the transfer rule file '" & UCase(TRANSFER_RULE_FILENAME) & "':" &
Chr(13) & Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to store it again;
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'otherwise sacrifice changes
    Else
       Exit Sub
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdClose_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Save any changes made here and return to the main interface."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCopy_Click ()

    'copy current entry
    lstTransferRule(0).AddItem lstTransferRule(0)
    lstTransferRule(1).AddItem lstTransferRule(1)
    lstTransferRule(2).AddItem lstTransferRule(2)

End Sub

'_____________________________________________________________________________________________________________
Sub cmdDelete_Click ()

    'confirm delete
    Beep
    MsgBoxAnswer = MsgBox("Are you sure you want to delete the current transfer rule?", MB_ICONQUESTION + MB_YESNO +
MB_DEFBUTTON2, "Confirm Delete")
    If MsgBoxAnswer = ID_YES Then
       'delete current entry
       lstTransferRule(0).RemoveItem lstTransferRule(0).ListIndex
       lstTransferRule(1).RemoveItem lstTransferRule(1).ListIndex
       lstTransferRule(2).RemoveItem lstTransferRule(2).ListIndex
       'update editor
       cmdEdit.Enabled = False
       cmdDelete.Enabled = False
       cmdCopy.Enabled = False
```

```
       cmdCancel.Enabled = True
       fChangesMade = True
    End If

End Sub

'_____________________________________________________________________________________________________
Sub cmdDelete_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Delete the current transfer rule."

End Sub

'_____________________________________________________________________________________________________
Sub cmdEdit_Click ()

    Dim Rule         As String
    Dim DelimiterAt  As Integer
    Dim ComponentPtr As Integer


    'unpack russian rule
    Rule = ""
    ComponentPtr = 1
    'get each component and place on separate line
    Do
       DelimiterAt = InStr(ComponentPtr, lstTransferRule(0), PACK_DELIMITER)
       'got another component; otherwise...
       If DelimiterAt Then
          Rule = Rule & Trim(Mid(lstTransferRule(0), ComponentPtr, DelimiterAt - ComponentPtr)) & Chr(13) & Chr(10)
          ComponentPtr = DelimiterAt + Len(PACK_DELIMITER)
       '...done
       Else
          Exit Do
       End If
    Loop
    'display rule, stripping last CR+LF
    frmTransferRuleEntry.txtRussian = Left(Rule, Len(Rule) - 2)

    'unpack english rule
    Rule = ""
    ComponentPtr = 1
    'get each component and place on separate line
    Do
       DelimiterAt = InStr(ComponentPtr, lstTransferRule(1), PACK_DELIMITER)
```

```
      'got another component; otherwise...
      If DelimiterAt Then
         Rule = Rule & Trim(Mid(lstTransferRule(1), ComponentPtr, DelimiterAt - ComponentPtr)) & Chr(13) & Chr(10)
         ComponentPtr = DelimiterAt + Len(PACK_DELIMITER)
      '...done
      Else
         Exit Do
      End If
   Loop
   'display rule, stripping last CR+LF
   frmTransferRuleEntry.txtEnglish = Left(Rule, Len(Rule) - 2)

   'display comments
   frmTransferRuleEntry.txtDescription = lstTransferRule(2)

   'bring up transfer rule dialog editor
   frmTransferRuleEntry.Show 1

   'if rule not canceled, process it
   If frmTransferRuleEntry.Tag <> "[CANCEL]" Then
      'delete current entry
      lstTransferRule(0).RemoveItem lstTransferRule(0).ListIndex
      lstTransferRule(1).RemoveItem lstTransferRule(1).ListIndex
      lstTransferRule(2).RemoveItem lstTransferRule(2).ListIndex
      'add modified entry
      lstTransferRule(0).AddItem PackRule(frmTransferRuleEntry.txtRussian)
      lstTransferRule(1).AddItem PackRule(frmTransferRuleEntry.txtEnglish)
      lstTransferRule(2).AddItem frmTransferRuleEntry.txtDescription
      'update editor
      cmdEdit.Enabled = False
      cmdDelete.Enabled = False
      cmdCopy.Enabled = False
      cmdCancel.Enabled = True
      fChangesMade = True
   End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdEdit_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'show topical help on info line
   PassiveHelp "Edit the selected transfer rule."

End Sub
```

```
'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP7_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on Transfer Rule Editor."

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    'set default command availabilty
    cmdClose.Enabled = True
    timerScroll.Enabled = True
    cmdAdd.SetFocus

End Sub

'_____________________________________________________________________________________________________________
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    'control+r regenerates transfer rules upon closing (undocumented feature)
    If (KeyCode = 82) And (Shift = 2) Then
       PassiveHelp "Transfer Rule auto-regeneration enabled."
       cmdClose_Click
       fRegenerate = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Load ()

    LoadUnexpandedTransferRules

End Sub
```

```
'_____________________________________________________________________________________________________
'get transfer rules in compressed form
'_____________________________________________________________________________________________________
Sub LoadUnexpandedTransferRules ()

    Dim TempRulePtr As Integer
    Dim NumRules    As Integer
    Dim NumTerms    As Integer
    Dim RulePtr     As Integer
    Dim TermPtr     As Integer
    Dim Component   As String
    Dim Rule        As String

    On Error GoTo ErrorOnTransferRuleRead


    'load transfer rules and build resident list
    MousePointerWait
    Open TRANSFER_RULE_FILENAME For Input As 1
    Input #1, NumRules
    DisplayWait -NumRules * 1.2, "Loading unexpanded transfer rules..."

    For RulePtr = 0 To NumRules

        'get russian components of current rule
        Rule = ""
        Input #1, NumTerms
        For TermPtr = 0 To NumTerms
            Input #1, Component
            'generate string of russian rule components
            Rule = Rule & Component & PACK_DELIMITER
        Next TermPtr
        lstTransferRule(0).AddItem Rule

        'get english components of current rule
        Input #1, NumTerms
        Rule = ""
        For TermPtr = 0 To NumTerms
            Input #1, Component
            'generate string of english rule components
            Rule = Rule & Component & PACK_DELIMITER
        Next TermPtr
        lstTransferRule(1).AddItem Rule

        'get comments
```

```
        Input #1, Component
        'add comments
        lstTransferRule(2).AddItem Component

        DisplayWait 1, ""
    Next RulePtr
    Close 1

    'hide status
    DisplayWait 0, ""
    MousePointerContinue

    Exit Sub


ErrorOnTransferRuleRead:

    'prompt user to deal with store fail
    Beep
    MousePointerContinue
    MsgBoxAnswer = MsgBox("Unable to read the transfer rule file '" & UCase(TRANSFER_RULE_FILENAME) & "':" & Chr(13)
& Error(Err) & ".", MB_ICONEXCLAMATION + MB_RETRYCANCEL, "File Error")
    'retry attempts to store it again; otherwise...
    If MsgBoxAnswer = ID_RETRY Then
       Resume
    'don't load rules
    Else
       MsgBox ("No editing can be performed."), MB_ICONINFORMATION, ("Transfer Rules Missing")
       Unload frmTransferRuleEditor
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstTransferRule_Click (Index As Integer)

    'set focus to all lists simultaneously
    lstTransferRule(0).ListIndex = lstTransferRule(Index).ListIndex
    lstTransferRule(1).ListIndex = lstTransferRule(Index).ListIndex
    lstTransferRule(2).ListIndex = lstTransferRule(Index).ListIndex

    'edit and delete now available
    cmdEdit.Enabled = True
    cmdDelete.Enabled = True
    cmdCopy.Enabled = True
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub lstTransferRule_DblClick (Index As Integer)

    'edit current entry
    If cmdEdit.Enabled Then
       cmdEdit_Click
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstTransferRule_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'set focus to all lists simultaneously
    lstTransferRule(0).ListIndex = lstTransferRule(Index).ListIndex
    lstTransferRule(1).ListIndex = lstTransferRule(Index).ListIndex
    lstTransferRule(2).ListIndex = lstTransferRule(Index).ListIndex

End Sub

'_____________________________________________________________________________________________________________
Sub lstTransferRule_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Select an existing transfer rule."

End Sub

'_____________________________________________________________________________________________________________
'pack unpacked transfer rule components
'_____________________________________________________________________________________________________________
Function PackRule (UnpackedRule)

    Dim PackPtr     As Integer
    Dim DelimiterAt As Integer
    Dim PackedRule  As String
    Dim Parameter   As String


    PackPtr = 1
    'loop through unpacked rule adding pack delimiters
    Do
       DelimiterAt = InStr(PackPtr, UnpackedRule & Chr(13) & Chr(10), Chr(13) & Chr(10))
```

```
        'process another component; otherwise...
        If DelimiterAt Then
           'skip blank lines
           Parameter = Mid(UnpackedRule, PackPtr, DelimiterAt - PackPtr) & PACK_DELIMITER
           If Parameter <> PACK_DELIMITER Then
              PackedRule = PackedRule & Parameter
              PackPtr = DelimiterAt + 2
           Else
              Exit Do
           End If
        '...done
        Else
           Exit Do
        End If
     Loop

     'return packed rule
     PackRule = PackedRule

End Function

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

     PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot2_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

     PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
'deal with scrolling both lists simultaneously
'_____________________________________________________________________________________________________________
Sub timerScroll_Timer ()

     Static PrevTop0 As Integer
     Static PrevTop1 As Integer
     Static PrevTop2 As Integer
```

```
    'if first list changed, reposition other two
    If lstTransferRule(0).TopIndex <> PrevTop0 Then
       PrevTop0 = lstTransferRule(0).TopIndex
       lstTransferRule(1).TopIndex = lstTransferRule(0).TopIndex
       lstTransferRule(2).TopIndex = lstTransferRule(0).TopIndex

    'if second list changed, reposition other two
    ElseIf lstTransferRule(1).TopIndex <> PrevTop1 Then
       PrevTop1 = lstTransferRule(1).TopIndex
       lstTransferRule(0).TopIndex = lstTransferRule(1).TopIndex
       lstTransferRule(2).TopIndex = lstTransferRule(1).TopIndex

    'if third list changed, reposition other two
    ElseIf lstTransferRule(2).TopIndex <> PrevTop2 Then
       PrevTop2 = lstTransferRule(2).TopIndex
       lstTransferRule(0).TopIndex = lstTransferRule(2).TopIndex
       lstTransferRule(1).TopIndex = lstTransferRule(2).TopIndex
    End If

End Sub

'_____________________________________________________________________________________________________________
'(frmCleanupExpressionEditor): simple entry dialog for cleanup filter editor
'_____________________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'return to cleanup editor with abort code
    Tag = "[CANCEL]"
    Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Discard the expression."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()
```

```
    'bring up help
    frmHelp.Tag = HELP7_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Display help on Transfer Rule Editor."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdLookup_Click ()

    'bring up lookup dialog
    frmLookup.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_Click ()

    Dim CharPtr        As Integer
    Dim BracketCount   As Integer
    Dim BraceCount     As Integer
    Dim ParameterCount As Integer
    Dim OpenBracket    As Integer
    Dim OpenBrace      As Integer
    Dim PackPtr        As Integer
    Dim DelimiterAt    As Integer


    'error check parameters in russian expression
    ParameterCount = 0
    BracketCount = 0
    BraceCount = 0
    OpenBracket = False
    OpenBrace = False
    For CharPtr = 1 To Len(txtRussian)
        Select Case Mid(txtRussian, CharPtr, 1)
               Case Chr(10): ParameterCount = ParameterCount + 1
               Case ":":     ParameterCount = ParameterCount - 1
```

```
            Case "[":     BracketCount = BracketCount + 1
                          If OpenBracket Then
                             Beep
                             txtRussian.SelStart = CharPtr - 1
                             txtRussian.SelLength = 1
                             MsgBox ("Bracketed terms not closed properly in Russian expression."), MB_ICONSTOP,
("Syntax Error")
                             txtRussian.SetFocus
                             Exit Sub
                          End If
                          OpenBracket = True
            Case "]":     BracketCount = BracketCount - 1
                          If Not OpenBracket Then
                             Beep
                             txtRussian.SelStart = CharPtr - 1
                             txtRussian.SelLength = 1
                             MsgBox ("Invalid bracket nesting in Russian expression."), MB_ICONSTOP, ("Syntax
Error")
                             txtRussian.SetFocus
                             Exit Sub
                          End If
                          OpenBracket = False
            Case "{":     BraceCount = BraceCount + 1
                          If OpenBrace Then
                             Beep
                             txtRussian.SelStart = CharPtr - 1
                             txtRussian.SelLength = 1
                             MsgBox ("Braced terms not closed properly in Russian expression."), MB_ICONSTOP,
("Syntax Error")
                             txtRussian.SetFocus
                             Exit Sub
                          End If
                          OpenBrace = True
            Case "}":     BraceCount = BraceCount - 1
                          If Not OpenBrace Then
                             Beep
                             txtRussian.SelStart = CharPtr - 1
                             txtRussian.SelLength = 1
                             MsgBox ("Invalid brace nesting in Russian expression."), MB_ICONSTOP, ("Syntax
Error")
                             txtRussian.SetFocus
                             Exit Sub
                          End If
                          OpenBrace = False
        End Select
    Next CharPtr
```

```
    'mismatched brackets
    If BracketCount <> 0 Then
       Beep
       MsgBox ("Mismatched []'s in Russian expression."), MB_ICONSTOP, ("Syntax Error")
       txtRussian.SetFocus
       Exit Sub
    End If

    'mismatched braces
    If BraceCount <> 0 Then
       Beep
       MsgBox ("Mismatched {}'s in Russian expression."), MB_ICONSTOP, ("Syntax Error")
       txtRussian.SetFocus
       Exit Sub
    End If

    'not every russian term has colon
    If ParameterCount <> -1 Then
       Beep
       MsgBox ("Parameter binder (#:) missing in Russian expression."), MB_ICONSTOP, ("Syntax Error")
       txtRussian.SetFocus
       Exit Sub
    End If

    'error check parameters in english expression
    ParameterCount = 0
    BracketCount = 0
    BraceCount = 0
    OpenBracket = False
    OpenBrace = False
    For CharPtr = 1 To Len(txtEnglish)
        Select Case Mid(txtEnglish, CharPtr, 1)
               Case Chr(10): ParameterCount = ParameterCount + 1
               Case ":":     ParameterCount = ParameterCount - 1
               Case "[":     BracketCount = BracketCount + 1
                             If OpenBracket Then
                                Beep
                                txtEnglish.SelStart = CharPtr - 1
                                txtEnglish.SelLength = 1
                                MsgBox ("Bracketed terms not closed properly in English expression."), MB_ICONSTOP,
("Syntax Error")
                                txtEnglish.SetFocus
                                Exit Sub
                             End If
                             OpenBracket = True
```

```
            Case "]":     BracketCount = BracketCount - 1
                          If Not OpenBracket Then
                             Beep
                             txtEnglish.SelStart = CharPtr - 1
                             txtEnglish.SelLength = 1
                             MsgBox ("Invalid bracket nesting in English expression."), MB_ICONSTOP, ("Syntax
Error")
                             txtEnglish.SetFocus
                             Exit Sub
                          End If
                          OpenBracket = False
            Case "{":     BraceCount = BraceCount + 1
                          If OpenBrace Then
                             Beep
                             txtEnglish.SelStart = CharPtr - 1
                             txtEnglish.SelLength = 1
                             MsgBox ("Braced terms not closed properly in English expression."), MB_ICONSTOP,
("Syntax Error")
                             txtEnglish.SetFocus
                             Exit Sub
                          End If
                          OpenBrace = True
            Case "}":     BraceCount = BraceCount - 1
                          If Not OpenBrace Then
                             Beep
                             txtEnglish.SelStart = CharPtr - 1
                             txtEnglish.SelLength = 1
                             MsgBox ("Invalid brace nesting in English expression."), MB_ICONSTOP, ("Syntax
Error")
                             txtEnglish.SetFocus
                             Exit Sub
                          End If
                          OpenBrace = False
        End Select
    Next CharPtr

    'mismatched brackets
    If BracketCount <> 0 Then
       Beep
       MsgBox ("Mismatched []'s in English expression."), MB_ICONSTOP, ("Syntax Error")
       txtEnglish.SetFocus
       Exit Sub
    End If

    'mismatched braces
    If BraceCount <> 0 Then
```

```
      Beep
      MsgBox ("Mismatched {}'s in English expression."), MB_ICONSTOP, ("Syntax Error")
      txtEnglish.SetFocus
      Exit Sub
   End If

   PackPtr = 1
   'look for blank entries
   Do
      DelimiterAt = InStr(PackPtr, txtEnglish & Chr(13) & Chr(10), Chr(13) & Chr(10))
      'process another component; otherwise...
      If DelimiterAt Then
         'found blank line; otherwise...
         If Mid(txtEnglish, PackPtr, DelimiterAt - PackPtr) = "" Then
            Beep
            txtEnglish.SelStart = DelimiterAt
            txtEnglish.SelLength = 1
            MsgBox ("English expression has a null term."), MB_ICONSTOP, ("Syntax Error")
            txtEnglish.SetFocus
            Exit Sub
         '...continue looking
         Else
            PackPtr = DelimiterAt + 2
         End If
      '...done
      Else
         Exit Do
      End If
   Loop

   'convert rules to uppercase
   txtRussian = UCase(txtRussian)
   txtEnglish = UCase(txtEnglish)

   'return to cleanup editor with ok code
   Tag = "[OK]"
   Hide

End Sub

'_____________________________________________________________________________________________________________
Sub cmdOK_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'show topical help on info line
   PassiveHelp "Accept the current expression."
```

```
End Sub

'_____________________________________________________________________________________________________
Sub Form_Activate ()

    'set initial focus to first entry field
    txtDescription.SetFocus

End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub txtDescription_Change ()

    'ok command availability dependent on completing all entries
    cmdOK.Enabled = ((txtRussian <> "") And (txtEnglish <> "") And (txtDescription <> ""))

End Sub

'_____________________________________________________________________________________________________
Sub txtDescription_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

End Sub

'_____________________________________________________________________________________________________
Sub txtDescription_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Enter a description of this expression."

End Sub

'_____________________________________________________________________________________________________
Sub txtEnglish_Change ()
```

```
    'ok command availability dependent on completing all entries
    cmdOK.Enabled = ((txtRussian <> "") And (txtEnglish <> "") And (txtDescription <> ""))

End Sub

'_____________________________________________________________________________________________________________
Sub txtEnglish_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtEnglish_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show topical help on info line
    PassiveHelp "Enter the components of the English expression."

End Sub

'_____________________________________________________________________________________________________________
Sub txtRussian_Change ()

    'ok command availability dependent on completing all entries
    cmdOK.Enabled = ((txtRussian <> "") And (txtEnglish <> "") And (txtDescription <> ""))

End Sub

'_____________________________________________________________________________________________________________
Sub txtRussian_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtRussian_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    'show topical help on info line
    PassiveHelp "Enter the components of the Russian expression."

End Sub

'_____________________________________________________________________________________________________
'(frmTransliteration): shows transliterated form of current abstract and provides simple editing functions
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub Form_Load ()

    'size window to portion of parent form
    Height = Screen.Height / 1.72
    Width = Screen.Width / 1.28

    'position offset slightly from top-left
    Left = Screen.Width * .04
    Top = Screen.Height * .06

    'enable functions available when transliteration form loaded
    frmMain.menuPrintTransliteration.Enabled = True
    frmMain.menuSaveTransliteration.Enabled = True

End Sub

'_____________________________________________________________________________________________________
Sub Form_Resize ()

    'fit display field vertically in window
    If Height > 450 Then
       pnlRoot.Height = Height - 450
    End If

End Sub

'_____________________________________________________________________________________________________
Sub Form_Unload (Cancel As Integer)

    'make functions unavailable when transliteration form unloaded
    frmMain.menuPrintTransliteration.Enabled = False
    frmMain.menuSaveTransliteration.Enabled = False

    PassiveHelp ""
```

```
End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub txtTransliteration_Change ()

    Static OneShot As Integer


    'ignore first change  when text loaded
    If Not OneShot Then
       OneShot = True
       Exit Sub
    Else
       'indicate that changes have been made
       gTransliterationModified = True
    End If

End Sub

'_____________________________________________________________________________________________________
Sub txtTransliteration_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on window
    PassiveHelp "This windows shows the transliterated abstract."

End Sub

'_____________________________________________________________________________________________________
'(frmTranslation): shows translation  of current abstract and provides simple editing functions
'_____________________________________________________________________________________________________
Option Explicit

'_____________________________________________________________________________________________________
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    'escape key aborts translation
    If KeyCode = 27 Then
```

```
       gAbort = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Load ()

    'size window to portion of parent form
    Height = Screen.Height / 1.72
    Width = Screen.Width / 1.28

    'position offset slightly from top-left
    Left = Screen.Width * .04
    Top = Screen.Height * .06

    'enable functions available when translation form loaded
    frmMain.menuSaveTranslation.Enabled = True
    frmMain.menuPostEditTranslation.Enabled = True

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Resize ()

    'fit display field vertically in window
    If Height > 450 Then
       pnlRoot.Height = Height - 450
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Unload (Cancel As Integer)

    'make functions unavailable when translation form unloaded
    frmMain.menuSaveTranslation.Enabled = False
    frmMain.menuPostEditTranslation.Enabled = False

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________________
Sub txtTranslation_Change ()

    Static OneShot As Integer


    'ignore first change when text loaded
    If Not OneShot Then
       OneShot = True
       Exit Sub
    Else
       'indicate that changes have been made
       gTranslationModified = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtTranslation_KeyDown (KeyCode As Integer, Shift As Integer)

    'escape key aborts translation
    If KeyCode = 27 Then
       gAbort = True
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtTranslation_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on window
    PassiveHelp "This windows shows the translated document."

End Sub

'_____________________________________________________________________________________________________________
'(frmTranslationDetails): shows transfer details of current abstract
'_____________________________________________________________________________________________________________
Option Explicit
```

```
'_____________________________________________________________________________________________________
Sub Form_Load ()

    'size window to portion of parent form
    Height = Screen.Height / 1.3
    Width = Screen.Width / 1.1

    'position offset slightly from top-left
    Left = Screen.Width * .04
    Top = Screen.Height * .06

    'enable functions available when translation form loaded
    frmMain.menuSave.Enabled = True
    frmMain.menuSaveTranslationDetails.Enabled = True
    frmMain.menuPrint.Enabled = True
    frmMain.menuPrintTranslationDetails.Enabled = True

End Sub

'_____________________________________________________________________________________________________
Sub Form_Resize ()

    'fit display field vertically in window
    If Height > 450 Then
       pnlRoot.Height = Height - 450
    End If

End Sub

'_____________________________________________________________________________________________________
Sub Form_Unload (Cancel As Integer)

    'make functions unavailable when translation form unloaded
    frmMain.menuSaveTranslationDetails.Enabled = False
    frmMain.menuPrintTranslationDetails.Enabled = False

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""
```

```
End Sub

'_____________________________________________________________________________________________________
'(frmLexiconEditor): lexicon editor, used to lookup, add, modify, and delete lexicon entries
'_____________________________________________________________________________________________________
Option Explicit

Dim fMode          As Integer  'current editor mode {LOOKUP,ADD}
Dim fSkip2ndStage  As Integer  'flag indicating whether 2nd stage skipped for part of speech

Dim fChangesMade   As Integer  'flag indicating whether changes made

Dim fBindPtr       As Integer  'pointer to current record in lexicon

Dim fUpperLimit    As Integer  'inflection table range for part of speech
Dim fLowerLimit    As Integer
Dim fUpperLimit2   As Integer  'second inflection table range for verbal adjectives (first range non-contiguous)
Dim fLowerLimit2   As Integer

Dim fInflectionNum As Integer  'current inflection pattern number (fLowerLimit<=n<=fUpperLimit)

Const LOOKUP = 1               'enumerations for fMode
Const ADD = 2

'_____________________________________________________________________________________________________
'enabled/disable associations list box
'_____________________________________________________________________________________________________
Sub Associations (State As Integer)

    Dim i As Integer


    lstAssociations.Enabled = State

    If State Then
       'enable field
       lblAssociations.ForeColor = BLACK
       lstAssociations.BackColor = CREAM
    Else
       'disable field
       lblAssociations.ForeColor = SAND
       lstAssociations.Clear
       lstAssociations.BackColor = LTSAND

       'reset any selections
```

```
       For i = 0 To lstAssociations.ListCount - 1
           lstAssociations.Selected(i) = False
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________
'reset entry fields for english verbs
'_____________________________________________________________________________________________________
Sub ClearEnglishVerbPanel ()

    txtPresentIndicative2nd = ""
    txtPresentIndicative3rd = ""
    txtImperfectIndicative = ""
    txtProgressive = ""
    txtPastParticiple = ""
    txtVAdvProgressive = ""
    txtVAdvPastParticiple = ""

End Sub

'_____________________________________________________________________________________________________
'clear entry field for general english word
'_____________________________________________________________________________________________________
Sub ClearGeneralPanel ()

    txtEnglishWord = ""

End Sub

'_____________________________________________________________________________________________________
'clear all inflection tables
'_____________________________________________________________________________________________________
Sub ClearInflectedForms ()

    Dim i As Integer


    'clear out verbal adverb inflections
    For i = 0 To 2
        lblVerbalAdverbDeclension(i).Caption = ""
    Next i

    'clear out noun inflections
```

```
    For i = 0 To 13
        lblNounDeclension(i).Caption = ""
    Next i

    'clear out adjective inflections
    For i = 0 To 35
        lblAdjectiveDeclension(i).Caption = ""
    Next i

    'clear out verb inflections
    For i = 0 To 12
        lblVerbConjugation(i).Caption = ""
    Next i

End Sub

'_____________________________________________________________________________________________________
'reset english noun entry field and article list box
'_____________________________________________________________________________________________________
Sub ClearNounPanel ()

    Dim i


    'clear noun fields
    txtSingular = ""
    txtPlural = ""

    'unselect all articles in list
    For i = 0 To lstArticles.ListCount - 1
        lstArticles.Selected(i) = False
    Next i

End Sub

'_____________________________________________________________________________________________________
'reset entry fields for english verbs
'_____________________________________________________________________________________________________
Sub ClearPerfectiveVerbPanel ()

    txtPerfectiveIndicative = ""
    txtPerfectiveImperfect = ""
    txtPerfectivePastParticiple = ""

End Sub
```

```
'_____________________________________________________________________________________________________
'initiate add operation
'_____________________________________________________________________________________________________
Sub cmdAddNew_Click ()

    'indicate add new entry
    fMode = ADD

    'set controls
    cmdAddNew.Enabled = False
    cmdLookup.Enabled = False
    cmdClose.Enabled = False
    cmdCancel.Enabled = True

    'enable stem entry field
    lblStem.Caption = "New Stem:"
    Stem True
    txtStem.SetFocus

End Sub

'_____________________________________________________________________________________________________
Sub cmdAddNew_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Define and add a new entry to the lexicon."

End Sub

'_____________________________________________________________________________________________________
'go back to previous folder
'_____________________________________________________________________________________________________
Sub cmdBack_Click ()

    cmdNext.Enabled = True

    'if at 3rd stage, return to 2nd; otherwise...
    If cmdNext.Tag = "3" And Not fSkip2ndStage Then
       cmdNext.Tag = "2"
       Setup2ndStage

    '...must be at 2nd, so go back to 1st
    Else
       cmdBack.Enabled = False
```

```
       cmdNext.Tag = "1"
       Setup1stStage
    End If

End Sub

'_____________________________________________________________________________________________________
Sub cmdBack_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Return to the previous screen."

End Sub

'_____________________________________________________________________________________________________
'abort current operation and reset lexicon editor
'_____________________________________________________________________________________________________
Sub cmdCancel_Click ()

    'reset entry fields
    Stem False
    PartOfSpeech False
    PrimaryAttributes False
    SecondaryAttributes False
    Associations False
    DisambiguationExpression False

    'clear fields on 2nd and 3rd stages
    txtInflectionNumber.Text = ""
    ClearNounPanel

    'reset available commands
    ResetCommands

    'return to 1st stage
    lblStem.Caption = "New Stem:"
    fBindPtr = 0
    cmdNext.Tag = "1"
    Setup1stStage
    cmdAddNew.SetFocus

End Sub

'_____________________________________________________________________________________________________
Sub cmdCancel_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    'display help on current object
    PassiveHelp "Ignore the changes made to the current entry."

End Sub

'_____________________________________________________________________________________________________
'close lexicon editor and return to main interface
'_____________________________________________________________________________________________________
Sub cmdClose_Click ()

    'if changes made, update lexicon file
    If fChangesMade Then
       StoreLexicon
       'reset flag to indicate changes made
       fChangesMade = False
    End If

    'close Lexicon Editor
    PassiveHelp ""
    Hide

End Sub

'_____________________________________________________________________________________________________
Sub cmdClose_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Exit the Lexicon Editor and return to the main interface."

End Sub

'_____________________________________________________________________________________________________
Sub cmdDelete_Click ()

    'confirm current entry deletion
    Beep
    MsgBoxAnswer = MsgBox("Are you sure you want to delete the current entry from the lexicon?", MB_ICONQUESTION +
MB_YESNO + MB_DEFBUTTON2, "Confirm Delete")
    If MsgBoxAnswer = ID_YES Then
       '...delete current entry
       DeleteLexiconEntry (fBindPtr)
       fChangesMade = True

       'reset Lexicon Editor
```

```
       cmdCancel_Click
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub cmdDelete_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Remove the current entry from the lexicon."

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_Click ()

    'bring up help
    frmHelp.Tag = HELP5_FILENAME
    frmHelp.Show 1

End Sub

'_____________________________________________________________________________________________________________
Sub cmdHelp_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Display help on using the Lexicon Editor."

End Sub


'_____________________________________________________________________________________________________________
'lookup russian stem word in lexicon
'_____________________________________________________________________________________________________________
Sub cmdLookup_Click ()

    'indicate looking up entry
    fMode = LOOKUP

    'set controls
    cmdAddNew.Enabled = False
    cmdLookup.Enabled = False
    cmdClose.Enabled = False
    cmdCancel.Enabled = True
```

```
    'enable stem entry field
    lblStem.Caption = "Lookup Entry:"
    Stem True
    txtStem.SetFocus

End Sub

'_____________________________________________________________________________________________________
Sub cmdLookup_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Find an entry in the lexicon and display its syntactic details."

End Sub

'_____________________________________________________________________________________________________
'go to next folder
'_____________________________________________________________________________________________________
Sub cmdNext_Click ()

    Dim POS   As Integer
    Dim Found As Integer
    Dim i     As Integer


    'get part of speech from list
    POS = lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)

    'transition from 1st stage to 2nd
    If cmdNext.Tag = "1" Then
       'on parts of speech where primary attributes required...
       If (POS <> PRONOUN_RELATIVE) And (POS <> PRONOUN_DEMONSTRATIVE) And (POS <> PREPOSITION) And (POS <>
CONJUNCTION) And (POS <> INTERJECTION) And (POS <> FIXED_EXPRESSION) Then
          '...make sure primary attribute(s) selected
          Found = False
          For i = 0 To lstPrimaryAttributes.ListCount - 1
              If lstPrimaryAttributes.Selected(i) Then
                 Found = True
                 Exit For
              End If
          Next i

          'if missing, indicate and don't proceed to next folder
          If Not Found Then
             Beep
             MsgBox ("You must select a primary attribute."), MB_ICONSTOP, ("Missing Entry")
```

```
         lstPrimaryAttributes.SetFocus
         Exit Sub
      End If
   End If

   'on parts of speech where secondary attributes required...
   If POS = PREPOSITION Then
      '...make sure primary attribute(s) selected
      Found = False
      For i = 0 To lstSecondaryAttributes.ListCount - 1
          If lstSecondaryAttributes.Selected(i) Then
             Found = True
             Exit For
          End If
      Next i

      'if missing, indicate and don't proceed to last folder
      If Not Found Then
         Beep
         MsgBox ("You must select a secondary attribute."), MB_ICONSTOP, ("Missing Entry")
         lstSecondaryAttributes.SetFocus
         Exit Sub
      End If
   End If

   'if currently on 2nd stage, go to 3rd; otherwise...
   If fSkip2ndStage Then
      cmdBack.Enabled = True
      cmdNext.Enabled = False
      cmdNext.Tag = "3"
      Setup3rdStage

   '...must be on 1st, so go to 2nd
   Else
      'setup 2nd stage folder
      Setup2ndStage

      'indicate next transition from 2nd stage to 3rd
      cmdBack.Enabled = True
      cmdNext.Tag = "2"
   End If

'transition from 2nd stage to 3rd
Else
   cmdBack.Enabled = True
   cmdNext.Enabled = False
```

```
       cmdNext.Tag = "3"
       Setup3rdStage
    End If

End Sub

'_____________________________________________________________________________________________________
Sub cmdNext_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Proceed to the next screen."

End Sub

'_____________________________________________________________________________________________________
'complete current operation, compile record for lexicon entry, then update lexicon
'_____________________________________________________________________________________________________
Sub cmdUpdate_Click ()

    Dim PrimaryAttributeCode   As Integer
    Dim SecondaryAttributeCode As Integer
    Dim TransferAttributeCode  As Integer
    Dim CharPtr                As Integer
    Dim SingularChar           As String * 1
    Dim PluralChar             As String * 1
    Dim NumDropOffs            As Integer
    Dim PluralString           As String
    Dim Target                 As String
    Dim PluralCode             As Integer
    Dim LongerForm             As Integer
    Dim i                      As Integer
    Dim LexiconEntry           As LexiconType


    'verify that article selection made
    If pnlEnglishNouns.Visible Then
       If (lstArticles.Selected(0) = False) And (lstArticles.Selected(1) = False) And (lstArticles.Selected(2) =
False) And (lstArticles.Selected(3) = False) Then
          Beep
          MsgBox ("You must select the article(s) for this English word."), MB_ICONSTOP, ("Missing Entry")
          lstArticles.SetFocus
          Exit Sub
       End If
    End If

    'build primary attribute code
```

```
For i = 0 To lstPrimaryAttributes.ListCount - 1
    If lstPrimaryAttributes.Selected(i) Then
       PrimaryAttributeCode = PrimaryAttributeCode + lstPrimaryAttributes.ItemData(i)
    End If
Next i

'build secondary attribute code
For i = 0 To lstSecondaryAttributes.ListCount - 1
    If lstSecondaryAttributes.Selected(i) Then
       SecondaryAttributeCode = SecondaryAttributeCode + lstSecondaryAttributes.ItemData(i)
    End If
Next i

'if noun, build transfer attribute
If pnlEnglishNouns.Visible Then
   'build transfer attribute code from associations
   For i = 0 To lstAssociations.ListCount - 1
       If lstAssociations.Selected(i) Then
          TransferAttributeCode = TransferAttributeCode + lstAssociations.ItemData(i)
       End If
   Next i

   'continue building transfer attribute code from target plurals
   'strip any leading/trailing whitespace
   txtSingular = Trim(txtSingular)
   txtPlural = Trim(txtPlural)

   'if forms identical, assign code; otherwise...
   If txtSingular = txtPlural Then
      PluralCode = PLURAL1

   '...decode further
   Else
      'determine which form longer
      If Len(txtSingular) > Len(txtPlural) Then
         LongerForm = Len(txtSingular)
      Else
         LongerForm = Len(txtPlural)
      End If

      'compare singular and plural forms characterwise
      For CharPtr = 1 To LongerForm
          SingularChar = LCase(Mid(txtSingular, CharPtr, 1))
          PluralChar = LCase(Mid(txtPlural, CharPtr, 1))
          'if stems no longer match...
          If PluralChar <> SingularChar Then
```

```
                '...determine how many characters to drop off singular before adding plural ending
                NumDropOffs = Len(txtSingular) - CharPtr + 1
                PluralString = Mid(txtPlural, CharPtr)
                Exit For
            End If
        Next CharPtr

        'determine drop off string
        Select Case PluralString
                Case "s":    PluralCode = PLURAL2
                Case "es":   PluralCode = PLURAL3
                Case "ies":  PluralCode = PLURAL4

                'determine code for 'ves' (leaf->leaves != knife->knives)
                Case "ves":
                             If NumDropOffs = 1 Then
                                PluralCode = PLURAL5
                             Else
                                PluralCode = PLURAL6
                             End If

                Case "a":    PluralCode = PLURAL7
                Case "i":    PluralCode = PLURAL8
                Case "en":   PluralCode = PLURAL9
                Case "ice":  PluralCode = PLURAL10
                Case "eese": PluralCode = PLURAL11

                'deal with unknown plural form (should be user's fault)
                Case Else:   Beep
                             MsgBox ("There is currently no plural encoding scheme for deriving '" & txtPlural & "'
from '" & txtSingular & "'.  No update can be performed."), MB_ICONSTOP, ("Program Limitation")
                             txtPlural.SetFocus
                             Exit Sub
        End Select
        TransferAttributeCode = TransferAttributeCode + PluralCode
      End If

      'continue building transfer attribute code from target articles
      For i = 0 To lstArticles.ListCount - 1
          If lstArticles.Selected(i) Then
             TransferAttributeCode = TransferAttributeCode + lstArticles.ItemData(i)
          End If
      Next i
   End If

   'determine target encoding:
```

```
    'nouns
    If pnlEnglishNouns.Visible Then
       Target = txtSingular
    'pronouns, adjectives, adverbs, prepositions, interrogatives, interjections, emphatic particles, conjunction,
fixed expression
    ElseIf pnlGeneral.Visible Then
       Target = txtEnglishWord
    'verbal adverbs
    ElseIf pnlVerbalAdverb.Visible Then
       Target = txtVAdvProgressive & CONJUGATION_DELIMITER & txtVAdvPastParticiple
    'perfective verbs
    ElseIf pnlPerfectiveVerbs.Visible Then
       Target = txtPerfectiveIndicative & CONJUGATION_DELIMITER & txtPerfectiveImperfect & CONJUGATION_DELIMITER &
txtPerfectivePastParticiple
    'imperfective verbs, verbal adjectives
    Else
       Target = txtPresentIndicative2nd & CONJUGATION_DELIMITER & txtPresentIndicative3rd & CONJUGATION_DELIMITER &
txtImperfectIndicative & CONJUGATION_DELIMITER & txtProgressive & CONJUGATION_DELIMITER & txtPastParticiple
    End If

    'build lexicon entry record
    LexiconEntry.Stem = Trim(txtStem)
    LexiconEntry.PartOfSpeech = lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)
    LexiconEntry.Target = Target
    LexiconEntry.InflectionPattern = Val(txtInflectionNumber)
    LexiconEntry.PrimaryAttributes = PrimaryAttributeCode
    LexiconEntry.SecondaryAttributes = SecondaryAttributeCode
    LexiconEntry.TransferAttributes = TransferAttributeCode
    LexiconEntry.Disambiguation = ConvertQuotes(txtDisambiguationExpression)

    'if updating changes to existing entry...
    If fMode = LOOKUP Then
       '...delete current entry
       DeleteLexiconEntry (fBindPtr)
    End If

    'insert entry into resident lexicon and regenerate index
    InsertLexiconEntry LexiconEntry
    GenerateLexiconIndex

    'indicate changes made
    fChangesMade = True

    'reset lexicon editor
    cmdCancel_Click
```

```
End Sub

'_____________________________________________________________________________________________________
Sub cmdUpdate_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Make the current changes to the lexicon permanent."

End Sub

'_____________________________________________________________________________________________________
'convert double-quotes to single-quotes
'_____________________________________________________________________________________________________
Function ConvertQuotes (ByVal InString As String) As String

    Dim i As Integer

    'find and replace each quote
    For i = 1 To Len(InString)
        If Mid(InString, i, 1) = Chr(34) Then
           Mid(InString, i, 1) = "'"
        End If
    Next i

    ConvertQuotes = InString

End Function

'_____________________________________________________________________________________________________
'enable/disable disambiguation expression entry field
'_____________________________________________________________________________________________________
Sub DisambiguationExpression (State As Integer)

    txtDisambiguationExpression.Enabled = State

    If State Then
       'enable field
       lblDisambiguationExpression.ForeColor = BLACK
       txtDisambiguationExpression.BackColor = CREAM
       pnlDisambiguationExpression.BackColor = CREAM
    Else
       'disable field
       lblDisambiguationExpression.ForeColor = SAND
       txtDisambiguationExpression.Text = ""
       txtDisambiguationExpression.BackColor = LTSAND
```

```
       pnlDisambiguationExpression.BackColor = LTSAND
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_Activate ()

    MousePointerContinue

    'preset form components
    txtStem.Height = 252
    txtDisambiguationExpression.Height = 252
    cmdNext.Tag = "1"
    pnlRussianPartI.ZOrder 0

    'set focus to first button, if available
    If cmdAddNew.Enabled Then
       cmdAddNew.SetFocus
    End If

    'insert unknown word if given
    If Tag <> "" Then
       txtStem = Tag
       cmdAddNew_Click
       Tag = ""
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)

    'control+l brings up hidden dialog for demonstration of symbolic constants
    If (KeyCode = 76) And (Shift = 2) Then
       frmLookup.Show 1
    End If

End Sub

'_____________________________________________________________________________________________________________
'initialize form components and static list boxes
'_____________________________________________________________________________________________________________
Sub Form_Load ()
```

```
    'load part of speech list
    lstPartOfSpeech.AddItem "Adjective"
    lstPartOfSpeech.ItemData(0) = ADJECTIVE
    lstPartOfSpeech.AddItem "Adverb"
    lstPartOfSpeech.ItemData(1) = ADVERB
    lstPartOfSpeech.AddItem "Conjunction"
    lstPartOfSpeech.ItemData(2) = CONJUNCTION
    lstPartOfSpeech.AddItem "Emphatic Particle"
    lstPartOfSpeech.ItemData(3) = INTERJECTION
    lstPartOfSpeech.AddItem "Fixed Expression"
    lstPartOfSpeech.ItemData(4) = FIXED_EXPRESSION
    lstPartOfSpeech.AddItem "Interjection"
    lstPartOfSpeech.ItemData(5) = INTERJECTION
    lstPartOfSpeech.AddItem "Interrogative"
    lstPartOfSpeech.ItemData(6) = INTERROGATIVE
    lstPartOfSpeech.AddItem "Noun, animate"
    lstPartOfSpeech.ItemData(7) = NOUN_ANIMATE
    lstPartOfSpeech.AddItem "Noun, inanimate"
    lstPartOfSpeech.ItemData(8) = NOUN_INANIMATE
    lstPartOfSpeech.AddItem "Preposition"
    lstPartOfSpeech.ItemData(9) = PREPOSITION
    lstPartOfSpeech.AddItem "Pronoun, demonstrative"
    lstPartOfSpeech.ItemData(10) = PRONOUN_DEMONSTRATIVE
    lstPartOfSpeech.AddItem "Pronoun, indefinite"
    lstPartOfSpeech.ItemData(11) = PRONOUN_INDEFINITE
    lstPartOfSpeech.AddItem "Pronoun, personal"
    lstPartOfSpeech.ItemData(12) = PRONOUN_PERSONAL
    lstPartOfSpeech.AddItem "Pronoun, possessive"
    lstPartOfSpeech.ItemData(13) = PRONOUN_POSSESSIVE
    lstPartOfSpeech.AddItem "Pronoun, relative"
    lstPartOfSpeech.ItemData(14) = PRONOUN_RELATIVE
    lstPartOfSpeech.AddItem "Verb"
    lstPartOfSpeech.ItemData(15) = VERB
    lstPartOfSpeech.AddItem "Verbal Adjective"
    lstPartOfSpeech.ItemData(16) = VERBAL_ADJECTIVE
    lstPartOfSpeech.AddItem "Verbal Adverb"
    lstPartOfSpeech.ItemData(17) = VERBAL_ADVERB

    'initialize entry fields
    Stem False
    PrimaryAttributes False
    SecondaryAttributes False
    Associations False
    DisambiguationExpression False

End Sub
```

```
'_____________________________________________________________________________________________
'unpack inflections for given pattern number and append to stem
'_____________________________________________________________________________________________
Sub GenerateInflectedForms ()

    Dim FieldPtr           As Integer
    Dim BreakStart         As Integer
    Dim BreakEnd           As Integer
    Dim POS                As Integer
    Dim Inflection         As String
    Dim InflectionString   As String
    Dim InflectionPtr      As Integer
    Dim i                  As Integer

    Static Inflections(36) As String

    On Error Resume Next


    'don't process if cancel operation selected
    If (lstPartOfSpeech.ListIndex < 0) Then
       Exit Sub
    End If

    'find enumerated position of all occurrences of ending
    FieldPtr = 1
    InflectionString = INFLECTION_DELIMITER & gPattern(fInflectionNum) & INFLECTION_DELIMITER
    POS = lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)

    'process patterns only, not gaps in record
    If Len(InflectionString) > 2 Then
       Do
          BreakStart = InStr(FieldPtr, InflectionString, INFLECTION_DELIMITER) + 1
          BreakEnd = InStr(BreakStart, InflectionString, INFLECTION_DELIMITER) - 1
          'if inflection pattern exhausted, exit loop
          If BreakEnd < 0 Then
             Exit Do
          End If
          'extract next field from inflection pattern
          Inflection = Mid(InflectionString, BreakStart, BreakEnd - BreakStart + 1)
          'deal with null ending
          If Inflection = "#" Then
             Inflection = ""
          End If
```

```
         'store current inflection in temp list to append to current stem
         Inflections(InflectionPtr) = Inflection
         InflectionPtr = InflectionPtr + 1

         'skip unused entries on adjectives and demonstrative and possessive pronouns
         If ((POS = PRONOUN_DEMONSTRATIVE) Or (POS = ADJECTIVE) Or (POS = PRONOUN_POSSESSIVE)) And ((InflectionPtr
= 8) Or (InflectionPtr = 17) Or (InflectionPtr = 26)) Then
            Inflections(InflectionPtr) = "/"
            InflectionPtr = InflectionPtr + 1

         'skip unused entries on relative pronoun
         ElseIf (POS = PRONOUN_RELATIVE) And ((InflectionPtr = 7) Or (InflectionPtr = 16) Or (InflectionPtr = 25))
Then
            Inflections(InflectionPtr) = "/"
            InflectionPtr = InflectionPtr + 1
            Inflections(InflectionPtr) = "/"
            InflectionPtr = InflectionPtr + 1

         'skip unused entries on verbal adjectives
         ElseIf (POS = VERBAL_ADJECTIVE) And ((InflectionPtr = 7) Or (InflectionPtr = 16) Or (InflectionPtr = 25))
Then
            'process all verbal adjectives except past passive
            If Not lstPrimaryAttributes.Selected(6) Then
               Inflections(InflectionPtr) = "/"
               InflectionPtr = InflectionPtr + 1
               Inflections(InflectionPtr) = "/"
               InflectionPtr = InflectionPtr + 1
            End If
         End If
         'move pointer to next field and compiled list slot
         FieldPtr = BreakEnd + 1
         'fail-safe loop exit for when no case found (ie. error in word or inflection pattern)
      Loop Until FieldPtr > Len(gPattern(fInflectionNum))

      'generate inflections
      For i = 0 To InflectionPtr - 1
          'combine stem and inflections for valid stems; otherwise...
          If Inflections(i) <> "/" Then
             'determine which table to update
             If pnlNouns.Visible Then
                lblNounDeclension(i).Caption = txtStem & Inflections(i)
             ElseIf pnlAdjectives.Visible Then
                lblAdjectiveDeclension(i).Caption = txtStem & Inflections(i)
             ElseIf POS = PRONOUN_INDEFINITE Then
                lblPronounDeclension(i).Caption = txtStem & Inflections(i) & "-"
             ElseIf POS = INTERROGATIVE Then
```

```
                lblPronounDeclension(i).Caption = txtStem & Inflections(i)
             ElseIf POS = PRONOUN_PERSONAL Then
                lblPronounDeclension(i).Caption = txtStem & Inflections(i)
             ElseIf pnlVerbalAdverbs.Visible Then
                lblVerbalAdverbDeclension(i).Caption = txtStem & Inflections(i)
             ElseIf pnlVerbs.Visible Then
                lblVerbConjugation(i).Caption = txtStem & Inflections(i)
             End If

          '...blank field
          Else
             'determine which table field to blank
             If pnlNouns.Visible Then
                lblNounDeclension(i).Caption = ""
             ElseIf pnlAdjectives.Visible Then
                lblAdjectiveDeclension(i).Caption = ""
             ElseIf pnlPronouns.Visible Then
                lblPronounDeclension(i).Caption = ""
             ElseIf pnlVerbalAdverbs.Visible Then
                lblVerbalAdverbDeclension(i).Caption = ""
             ElseIf pnlVerbs.Visible Then
                lblVerbConjugation(i).Caption = ""
             End If
          End If
       Next i
       PassiveHelp ""

    'blank current inflections
    Else
       PassiveHelp "This inflection pattern number is not currently in use.  It cannot be assigned to the new
entry."
       ClearInflectedForms
    End If

End Sub

'_____________________________________________________________________________________________________
'hide inflection panels stacked on top of each other at 3rd stage
'_____________________________________________________________________________________________________
Sub HideInflectionPanels ()

    pnlNouns.Visible = False
    pnlPronouns.Visible = False
    pnlAdjectives.Visible = False
    pnlVerbs.Visible = False
    pnlNouns.Visible = False
```

```
    pnlPronouns.Visible = False
    pnlVerbalAdverbs.Visible = False

End Sub

'_____________________________________________________________________________________________________________
Sub lstArticles_Click ()

    'enforce mutual exclusion on a/an
    If lstArticles.ListIndex = 1 And lstArticles.Selected(2) Then
       lstArticles.Selected(2) = False
    ElseIf lstArticles.ListIndex = 2 And lstArticles.Selected(1) Then
       lstArticles.Selected(1) = False
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstArticles_KeyPress (KeyAscii As Integer)

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'if update command available, set focus to it; otherwise...
       If cmdUpdate.Enabled Then
          cmdUpdate.SetFocus
       '...set focus to first entry field not filled in
       Else
          If txtSingular = "" Then
             txtSingular.SetFocus
          Else
             txtPlural.SetFocus
          End If
       End If
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstArticles_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Select the article(s) for the singular form of the current English noun."

End Sub
```

```
'_____________________________________________________________________________________________________
Sub lstAssociations_Click ()

    'reset current inflections
    ClearInflectedForms

End Sub

'_____________________________________________________________________________________________________
Sub lstAssociations_KeyPress (KeyAscii As Integer)

    'carriage return moves to 'Next' button
    If KeyAscii = 13 Then
       KeyAscii = 0
       cmdNext.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________
Sub lstAssociations_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    PassiveHelp "Select the association attribute(s) for this Russian entry."

End Sub

'_____________________________________________________________________________________________________
'load contents of each stage based on attributes relevant to selected part of speech
'_____________________________________________________________________________________________________
Sub lstPartOfSpeech_Click ()

    Dim POS As Integer


    'set no upper inflection pattern range until requested
    fLowerLimit2 = -1
    fUpperLimit2 = -1

    'clear current contents of current lists and entry fields
    lstPrimaryAttributes.Clear
    lstSecondaryAttributes.Clear
    lstAssociations.Clear
    lstArticles.Clear
    pnlEnglishNouns.Visible = False
```

```
pnlEnglishVerbs.Visible = False
pnlPerfectiveVerbs.Visible = False
pnlVerbalAdverb.Visible = False
pnlGeneral.Visible = False
txtInflectionNumber = ""
txtEnglishWord = ""
txtVAdvProgressive = ""
txtVAdvPastParticiple = ""
HideInflectionPanels

'abort if no part of speech selected (cancel generates click event without selecting)
If lstPartOfSpeech.ListIndex < 0 Then
   cmdNext.Enabled = False
   Exit Sub
End If

'prepare next stage
cmdNext.Enabled = True
fSkip2ndStage = False

'decide how to load attribute list boxes
POS = lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)
Select Case POS

       Case NOUN_ANIMATE, NOUN_INANIMATE:
            'set fields
            PrimaryAttributes True
            SecondaryAttributes True
            Associations True
            DisambiguationExpression True

            'load primary attribute list
            lstPrimaryAttributes.AddItem "Masculine"
            lstPrimaryAttributes.ItemData(0) = NOUN_MASCULINE
            lstPrimaryAttributes.AddItem "Feminine"
            lstPrimaryAttributes.ItemData(1) = NOUN_FEMININE
            lstPrimaryAttributes.AddItem "Neuter"
            lstPrimaryAttributes.ItemData(2) = NOUN_NEUTER
            lstPrimaryAttributes.AddItem "Plural"
            lstPrimaryAttributes.ItemData(3) = NOUN_PLURAL

            'load secondary attribute list
            lstSecondaryAttributes.AddItem "Genitive"
            lstSecondaryAttributes.ItemData(0) = DEMAND1_GENITIVE
            lstSecondaryAttributes.AddItem "Dative"
            lstSecondaryAttributes.ItemData(1) = DEMAND1_DATIVE
```

```
      lstSecondaryAttributes.AddItem "Instrumental"
      lstSecondaryAttributes.ItemData(2) = DEMAND1_INSTRUMENTAL

      'load source association list
      lstAssociations.AddItem "Person"
      lstAssociations.ItemData(0) = ASSOCIATION_PERSON
      lstAssociations.AddItem "Place"
      lstAssociations.ItemData(1) = ASSOCIATION_PLACE
      lstAssociations.AddItem "Thing"
      lstAssociations.ItemData(2) = ASSOCIATION_THING
      lstAssociations.AddItem "Animal"
      lstAssociations.ItemData(3) = ASSOCIATION_ANIMAL
      lstAssociations.AddItem "Figurative/Idiomatic"
      lstAssociations.ItemData(4) = ASSOCIATION_FIGURATIVE

      'load articles list
      lstArticles.AddItem "None"
      lstArticles.ItemData(0) = ARTICLE_NONE
      lstArticles.AddItem "A"
      lstArticles.ItemData(1) = ARTICLE_A
      lstArticles.AddItem "An"
      lstArticles.ItemData(2) = ARTICLE_AN
      lstArticles.AddItem "The"
      lstArticles.ItemData(3) = ARTICLE_THE

      'set up other two stages for nouns
      pnlInflectionDescriptor.Caption = "Noun Inflection Patterns"
      ClearNounPanel
      pnlNouns.Visible = True
      pnlEnglishNouns.Visible = True

Case PRONOUN_PERSONAL:
      'set fields
      PrimaryAttributes True
      SecondaryAttributes False
      Associations False
      DisambiguationExpression True

      'load primary attribute list
      lstPrimaryAttributes.AddItem "1st singular"
      lstPrimaryAttributes.ItemData(0) = PERSONAL_PRONOUN_1ST_SINGULAR
      lstPrimaryAttributes.AddItem "2nd singular"
      lstPrimaryAttributes.ItemData(1) = PERSONAL_PRONOUN_2ND_SINGULAR
      lstPrimaryAttributes.AddItem "3rd masculine"
      lstPrimaryAttributes.ItemData(2) = PERSONAL_PRONOUN_3RD_SINGULAR_MASCULINE
      lstPrimaryAttributes.AddItem "3rd feminine"
```

```
        lstPrimaryAttributes.ItemData(3) = PERSONAL_PRONOUN_3RD_SINGULAR_FEMININE
        lstPrimaryAttributes.AddItem "3rd neuter"
        lstPrimaryAttributes.ItemData(4) = PERSONAL_PRONOUN_3RD_SINGULAR_NEUTER
        lstPrimaryAttributes.AddItem "1st plural"
        lstPrimaryAttributes.ItemData(5) = PERSONAL_PRONOUN_1ST_PLURAL
        lstPrimaryAttributes.AddItem "2nd plural"
        lstPrimaryAttributes.ItemData(6) = PERSONAL_PRONOUN_2ND_PLURAL
        lstPrimaryAttributes.AddItem "3rd plural"
        lstPrimaryAttributes.ItemData(7) = PERSONAL_PRONOUN_3RD_PLURAL

        'set up other two stages for personal pronouns
        pnlInflectionDescriptor.Caption = "Personal Pronoun Inflection Patterns"
        lblGeneral.Caption = "English Pronoun:"
        fLowerLimit = PERSONAL_PRONOUN_START
        fUpperLimit = PERSONAL_PRONOUN_END
        txtInflectionNumber = fLowerLimit
        pnlPronouns.Visible = True
        pnlGeneral.Visible = True

    Case PRONOUN_POSSESSIVE:
        'set fields
        PrimaryAttributes True
        SecondaryAttributes False
        Associations False
        DisambiguationExpression True

        'load primary attribute list
        lstPrimaryAttributes.AddItem "Declinable"
        lstPrimaryAttributes.ItemData(0) = POSS_PRONOUN_DECLINABLE
        lstPrimaryAttributes.AddItem "3rd masc/neut, indec."
        lstPrimaryAttributes.ItemData(1) = POSS_PRONOUN_INDECLINABLE_3RD_MASC_NEUT
        lstPrimaryAttributes.AddItem "3rd feminine, indec."
        lstPrimaryAttributes.ItemData(2) = POSS_PRONOUN_INDECLINABLE_3RD_FEMININE
        lstPrimaryAttributes.AddItem "3rd plural, indec."
        lstPrimaryAttributes.ItemData(3) = POSS_PRONOUN_INDECLINABLE_3RD_PLURAL

        'set up other two stages for possessive pronouns
        pnlInflectionDescriptor.Caption = "Possessive Pronoun Inflection Patterns"
        lblGeneral.Caption = "English Pronoun:"
        HideInflectionPanels
        fLowerLimit = POSSESSIVE_PRONOUN_START
        fUpperLimit = POSSESSIVE_PRONOUN_END
        txtInflectionNumber = fLowerLimit
        pnlAdjectives.Visible = True
        pnlAdjectives.Visible = True
        pnlGeneral.Visible = True
```

```
Case PRONOUN_INDEFINITE:
     'set fields
     PrimaryAttributes True
     SecondaryAttributes False
     Associations False
     DisambiguationExpression True

     'load primary attribute list
     lstPrimaryAttributes.AddItem "Masculine"
     lstPrimaryAttributes.ItemData(0) = NOUN_MASCULINE
     lstPrimaryAttributes.AddItem "Neuter"
     lstPrimaryAttributes.ItemData(1) = NOUN_NEUTER

     'set up other two stages for indefinite pronouns
     pnlInflectionDescriptor.Caption = "Indefinite Pronoun Inflection Patterns"
     lblGeneral.Caption = "English Pronoun:"
     HideInflectionPanels
     fLowerLimit = INDEFINITE_PRONOUN_START
     fUpperLimit = INDEFINITE_PRONOUN_END
     txtInflectionNumber = fLowerLimit
     pnlPronouns.Visible = True
     pnlGeneral.Visible = True

Case PRONOUN_RELATIVE
     'set fields
     PrimaryAttributes False
     SecondaryAttributes False
     Associations False
     DisambiguationExpression True

     'set up other two stages for relative pronoun
     pnlInflectionDescriptor.Caption = "Relative Pronoun Inflection Patterns"
     lblGeneral.Caption = "English Pronoun:"
     fLowerLimit = RELATIVE_PRONOUN_AT
     fUpperLimit = RELATIVE_PRONOUN_AT
     txtInflectionNumber = fLowerLimit
     pnlAdjectives.Visible = True
     pnlGeneral.Visible = True

Case PRONOUN_DEMONSTRATIVE:
     'set fields
     PrimaryAttributes False
     SecondaryAttributes False
     Associations False
     DisambiguationExpression True
```

```
     'set up other two stages for demonstrative pronouns
     pnlInflectionDescriptor.Caption = "Demonstrative Pronoun Inflection Patterns"
     lblGeneral.Caption = "English Pronoun:"
     HideInflectionPanels
     fLowerLimit = DEMONSTRATIVE_PRONOUN_START
     fUpperLimit = DEMONSTRATIVE_PRONOUN_END
     txtInflectionNumber = fLowerLimit
     pnlAdjectives.Visible = True
     pnlGeneral.Visible = True

Case VERB:
     'set fields
     PrimaryAttributes True
     SecondaryAttributes True
     Associations False
     DisambiguationExpression True

     'load primary attribute list
     lstPrimaryAttributes.AddItem "Main"
     lstPrimaryAttributes.ItemData(0) = VERB_MAIN
     lstPrimaryAttributes.AddItem "Auxiliary"
     lstPrimaryAttributes.ItemData(1) = VERB_AUXILIARY
     lstPrimaryAttributes.AddItem "Imperfective"
     lstPrimaryAttributes.ItemData(2) = VERB_IMPERFECTIVE
     lstPrimaryAttributes.AddItem "Perfective"
     lstPrimaryAttributes.ItemData(3) = VERB_PERFECTIVE
     lstPrimaryAttributes.AddItem "Non-reflexive"
     lstPrimaryAttributes.ItemData(4) = VERB_NONREFLEXIVE
     lstPrimaryAttributes.AddItem "Reflexive, active"
     lstPrimaryAttributes.ItemData(5) = VERB_REFLEXIVE_ACTIVE
     lstPrimaryAttributes.AddItem "Reflexive, passive"
     lstPrimaryAttributes.ItemData(6) = VERB_REFLEXIVE_PASSIVE
     lstPrimaryAttributes.AddItem "Indeterminate"
     lstPrimaryAttributes.ItemData(7) = VERB_INDETERMINATE
     lstPrimaryAttributes.AddItem "Determinate"
     lstPrimaryAttributes.ItemData(8) = VERB_DETERMINATE

     'load secondary attribute list
     lstSecondaryAttributes.AddItem "Nominative"
     lstSecondaryAttributes.ItemData(0) = DEMAND2_NOMINATIVE
     lstSecondaryAttributes.AddItem "Genitive"
     lstSecondaryAttributes.ItemData(1) = DEMAND2_GENITIVE
     lstSecondaryAttributes.AddItem "Dative"
     lstSecondaryAttributes.ItemData(2) = DEMAND2_DATIVE
     lstSecondaryAttributes.AddItem "Accusative"
```

```
        lstSecondaryAttributes.ItemData(3) = DEMAND2_ACCUSATIVE
        lstSecondaryAttributes.AddItem "Instrumental"
        lstSecondaryAttributes.ItemData(4) = DEMAND2_INSTRUMENTAL

        'set up other two stages for verbs
        pnlInflectionDescriptor.Caption = "Verb Inflection Patterns"
        HideInflectionPanels
        ClearEnglishVerbPanel
        ClearPerfectiveVerbPanel
        fLowerLimit = VERB_START
        fUpperLimit = VERB_END
        txtInflectionNumber = fLowerLimit
        pnlVerbs.Visible = True

    Case ADJECTIVE:
        'set fields
        PrimaryAttributes True
        SecondaryAttributes True
        Associations False
        DisambiguationExpression True

        'load primary attribute list
        lstPrimaryAttributes.AddItem "Normal, long form"
        lstPrimaryAttributes.ItemData(0) = ADJECTIVE_LONG_FORM
        lstPrimaryAttributes.AddItem "Normal, short form"
        lstPrimaryAttributes.ItemData(1) = ADJECTIVE_SHORT_FORM
        lstPrimaryAttributes.AddItem "Comparative"
        lstPrimaryAttributes.ItemData(2) = ADJECTIVE_COMPARATIVE
        lstPrimaryAttributes.AddItem "Superlative (simple)"
        lstPrimaryAttributes.ItemData(3) = ADJECTIVE_SUPERLATIVE

        'load secondary attribute list
        lstSecondaryAttributes.AddItem "Genitive"
        lstSecondaryAttributes.ItemData(0) = DEMAND1_GENITIVE
        lstSecondaryAttributes.AddItem "Dative"
        lstSecondaryAttributes.ItemData(1) = DEMAND1_DATIVE
        lstSecondaryAttributes.AddItem "Instrumental"
        lstSecondaryAttributes.ItemData(2) = DEMAND1_INSTRUMENTAL

        'set up other two stages for adjectives
        pnlInflectionDescriptor.Caption = "Adjective Inflection Patterns"
        lblGeneral.Caption = "English Adjective:"
        fLowerLimit = ADJECTIVE_START
        fUpperLimit = ADJECTIVE_END
        txtInflectionNumber = fLowerLimit
        HideInflectionPanels
```

```
     pnlAdjectives.Visible = True
     pnlGeneral.Visible = True

Case ADVERB:
     'set fields
     PrimaryAttributes True
     SecondaryAttributes True
     Associations False
     DisambiguationExpression True

     'load primary attribute list
     lstPrimaryAttributes.AddItem "Normal"
     lstPrimaryAttributes.ItemData(0) = ADVERB_NORMAL
     lstPrimaryAttributes.AddItem "Comparative"
     lstPrimaryAttributes.ItemData(1) = ADVERB_COMPARATIVE
     lstPrimaryAttributes.AddItem "Superlative"
     lstPrimaryAttributes.ItemData(2) = ADVERB_SUPERLATIVE

     'load secondary attribute list
     lstSecondaryAttributes.AddItem "Genitive"
     lstSecondaryAttributes.ItemData(0) = DEMAND1_GENITIVE
     lstSecondaryAttributes.AddItem "Dative"
     lstSecondaryAttributes.ItemData(1) = DEMAND1_DATIVE
     lstSecondaryAttributes.AddItem "Instrumental"
     lstSecondaryAttributes.ItemData(2) = DEMAND1_INSTRUMENTAL

     'set up other two stages for adverbs
     lblGeneral.Caption = "English Adverb:"
     HideInflectionPanels
     fSkip2ndStage = True
     pnlGeneral.Visible = True

Case PREPOSITION:
     'set fields
     PrimaryAttributes False
     SecondaryAttributes True
     Associations False
     DisambiguationExpression True

     'load secondary attribute list
     lstSecondaryAttributes.AddItem "Genitive"
     lstSecondaryAttributes.ItemData(0) = DEMAND3_GENITIVE
     lstSecondaryAttributes.AddItem "Dative"
     lstSecondaryAttributes.ItemData(1) = DEMAND3_DATIVE
     lstSecondaryAttributes.AddItem "Accusative"
     lstSecondaryAttributes.ItemData(2) = DEMAND3_ACCUSATIVE
```

```
        lstSecondaryAttributes.AddItem "Instrumental"
        lstSecondaryAttributes.ItemData(3) = DEMAND3_INSTRUMENTAL
        lstSecondaryAttributes.AddItem "Prepositional"
        lstSecondaryAttributes.ItemData(4) = DEMAND3_PREPOSITIONAL

        'set up other two stages for prepositions
        lblGeneral.Caption = "English Preposition:"
        HideInflectionPanels
        ClearGeneralPanel
        fSkip2ndStage = True
        pnlGeneral.Visible = True

    Case INTERROGATIVE:
        'set fields
        PrimaryAttributes True
        SecondaryAttributes False
        Associations False
        DisambiguationExpression True

        'load primary attribute list
        lstPrimaryAttributes.AddItem "Masculine"
        lstPrimaryAttributes.ItemData(0) = INTERROGATIVE_MASCULINE
        lstPrimaryAttributes.AddItem "Neuter"
        lstPrimaryAttributes.ItemData(1) = INTERROGATIVE_NEUTER
        lstPrimaryAttributes.AddItem "Indeclinable"
        lstPrimaryAttributes.ItemData(2) = INTERROGATIVE_INDECLINABLE

        'set up other two stages for interrogatives
        pnlInflectionDescriptor.Caption = "Interrogative Inflection Patterns"
        lblGeneral.Caption = "English Interrogative:"
        HideInflectionPanels
        fLowerLimit = INTERROGATIVE_START
        fUpperLimit = INTERROGATIVE_END
        txtInflectionNumber = fLowerLimit
        pnlPronouns.Visible = True
        pnlGeneral.Visible = True

    Case CONJUNCTION, FIXED_EXPRESSION, EMPHATIC_PARTICLE, INTERJECTION:
        'set fields
        PrimaryAttributes False
        SecondaryAttributes False
        Associations False
        DisambiguationExpression True

        'set up other two stages
        ClearGeneralPanel
```

```
      fSkip2ndStage = True
      pnlGeneral.Visible = True

      'set title to entry field
      Select Case POS
             Case EMPHATIC_PARTICLE: lblGeneral.Caption = "English Interjection:"
             Case CONJUNCTION:       lblGeneral.Caption = "English Conjunction:"
             Case INTERJECTION:      lblGeneral.Caption = "English Interjection:"
             Case FIXED_EXPRESSION:  lblGeneral.Caption = "English Fixed Expression:"
      End Select

Case VERBAL_ADVERB:
      'set fields
      PrimaryAttributes True
      SecondaryAttributes True
      Associations False
      DisambiguationExpression True

      'load primary attribute list
      lstPrimaryAttributes.AddItem "Non-reflexive"
      lstPrimaryAttributes.ItemData(0) = VERBAL_ADVERB_NON_REFLEXIVE
      lstPrimaryAttributes.AddItem "Reflexive, active"
      lstPrimaryAttributes.ItemData(1) = VERBAL_ADVERB_REFLEXIVE_ACTIVE
      lstPrimaryAttributes.AddItem "Reflexive, passive"
      lstPrimaryAttributes.ItemData(2) = VERBAL_ADVERB_REFLEXIVE_PASSIVE
      lstPrimaryAttributes.AddItem "Imperfective"
      lstPrimaryAttributes.ItemData(3) = VERBAL_ADVERB_IMPERFECTIVE
      lstPrimaryAttributes.AddItem "Perfective"
      lstPrimaryAttributes.ItemData(4) = VERBAL_ADVERB_PERFECTIVE

      'load secondary attribute list
      lstSecondaryAttributes.AddItem "Nominative"
      lstSecondaryAttributes.ItemData(0) = DEMAND2_NOMINATIVE
      lstSecondaryAttributes.AddItem "Genitive"
      lstSecondaryAttributes.ItemData(1) = DEMAND2_GENITIVE
      lstSecondaryAttributes.AddItem "Dative"
      lstSecondaryAttributes.ItemData(2) = DEMAND2_DATIVE
      lstSecondaryAttributes.AddItem "Accusative"
      lstSecondaryAttributes.ItemData(3) = DEMAND2_ACCUSATIVE
      lstSecondaryAttributes.AddItem "Instrumental"
      lstSecondaryAttributes.ItemData(4) = DEMAND2_INSTRUMENTAL

      'set up other two stages for verbal adverbs
      pnlInflectionDescriptor.Caption = "Verbal Adverb Inflection Patterns"
      pnlVerbalAdverbs.Visible = True
      pnlVerbalAdverb.Visible = True
```

```
Case VERBAL_ADJECTIVE:
     'set fields
     PrimaryAttributes True
     SecondaryAttributes True
     Associations False
     DisambiguationExpression True

     'load primary attribute list
     lstPrimaryAttributes.AddItem "Non-reflexive"
     lstPrimaryAttributes.ItemData(0) = VERBAL_ADJECTIVE_NON_REFLEXIVE
     lstPrimaryAttributes.AddItem "Reflexive, active"
     lstPrimaryAttributes.ItemData(1) = VERBAL_ADJECTIVE_REFLEXIVE_ACTIVE
     lstPrimaryAttributes.AddItem "Reflexive, passive"
     lstPrimaryAttributes.ItemData(2) = VERBAL_ADJECTIVE_REFLEXIVE_PASSIVE
     lstPrimaryAttributes.AddItem "Present Active"
     lstPrimaryAttributes.ItemData(3) = VERBAL_ADJECTIVE_PRESENT_ACTIVE
     lstPrimaryAttributes.AddItem "Present Passive"
     lstPrimaryAttributes.ItemData(4) = VERBAL_ADJECTIVE_PRESENT_PASSIVE
     lstPrimaryAttributes.AddItem "Past Active"
     lstPrimaryAttributes.ItemData(5) = VERBAL_ADJECTIVE_PAST_ACTIVE
     lstPrimaryAttributes.AddItem "Past Passive"
     lstPrimaryAttributes.ItemData(6) = VERBAL_ADJECTIVE_PAST_PASSIVE
     lstPrimaryAttributes.AddItem "Imperfective"
     lstPrimaryAttributes.ItemData(7) = VERBAL_ADJECTIVE_IMPERFECTIVE
     lstPrimaryAttributes.AddItem "Perfective"
     lstPrimaryAttributes.ItemData(8) = VERBAL_ADJECTIVE_PERFECTIVE

     'load secondary attribute list
     lstSecondaryAttributes.AddItem "Nominative"
     lstSecondaryAttributes.ItemData(0) = DEMAND2_NOMINATIVE
     lstSecondaryAttributes.AddItem "Genitive"
     lstSecondaryAttributes.ItemData(1) = DEMAND2_GENITIVE
     lstSecondaryAttributes.AddItem "Dative"
     lstSecondaryAttributes.ItemData(2) = DEMAND2_DATIVE
     lstSecondaryAttributes.AddItem "Accusative"
     lstSecondaryAttributes.ItemData(3) = DEMAND2_ACCUSATIVE
     lstSecondaryAttributes.AddItem "Instrumental"
     lstSecondaryAttributes.ItemData(4) = DEMAND2_INSTRUMENTAL

     'set up other two stages for verbal adjectives
     pnlInflectionDescriptor.Caption = "Verbal Adjective Inflection Patterns"
     fLowerLimit = VERBAL_ADJECTIVE_PA1_START
     fUpperLimit = VERBAL_ADJECTIVE_PA1_END
     txtInflectionNumber = fLowerLimit
     ClearEnglishVerbPanel
```

```
                pnlAdjectives.Visible = True
                pnlEnglishVerbs.Visible = True
    End Select

End Sub

'_____________________________________________________________________________________________________________
Sub lstPartOfSpeech_KeyPress (KeyAscii As Integer)

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'move to next available list
       If lstPrimaryAttributes.Enabled Then
          lstPrimaryAttributes.SetFocus
       ElseIf lstSecondaryAttributes.Enabled Then
          lstSecondaryAttributes.SetFocus
       ElseIf lstAssociations.Enabled Then
          lstAssociations.SetFocus
       'no lists used for this part of speech, so go to 'Next' button
       Else
          cmdNext.SetFocus
       End If
    End If


End Sub

'_____________________________________________________________________________________________________________
Sub lstPartOfSpeech_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    PassiveHelp "Select the part of speech for this Russian entry."

End Sub

'_____________________________________________________________________________________________________________
'determine consequences of selecting primary attribute(s)
'_____________________________________________________________________________________________________________
Sub lstPrimaryAttributes_Click ()

    Dim i As Integer


    'set no upper inflection pattern range until requested
```

```
fLowerLimit2 = -1
fUpperLimit2 = -1

'reset current inflections
ClearInflectedForms

'comparative adjectives have no inflections, so skip 2nd stage
If lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = ADJECTIVE Then
   fSkip2ndStage = (lstPrimaryAttributes.ItemData(lstPrimaryAttributes.ListIndex) = ADJECTIVE_COMPARATIVE)
'indeclinable interrogatives have no inflections, so skip 2nd stage
ElseIf lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = INTERROGATIVE Then
   fSkip2ndStage = (lstPrimaryAttributes.ItemData(lstPrimaryAttributes.ListIndex) = INTERROGATIVE_INDECLINABLE)
   If fSkip2ndStage Then
      txtInflectionNumber = ""
   End If
'on declinable possessive pronouns have inflections; otherwise skip 2nd stage
ElseIf lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = PRONOUN_POSSESSIVE Then
   fSkip2ndStage = (lstPrimaryAttributes.ItemData(lstPrimaryAttributes.ListIndex) <> POSS_PRONOUN_DECLINABLE)
End If

'determine inflection pattern range depending on verbal adjective type
If lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = VERBAL_ADJECTIVE Then

   'enforce mutual exclusion on verbal adjective reflexivity and type
   Select Case lstPrimaryAttributes.ListIndex

          'non-reflexive verbal adjective so make sure reflexive unselected
          Case 0: lstPrimaryAttributes.Selected(1) = False
                  lstPrimaryAttributes.Selected(2) = False
                  'determine non-reflexive range for selected verbal adjective type
                  Select Case fLowerLimit

                         'present active reflexive becomes non-reflexive
                         Case VERBAL_ADJECTIVE_PA2_START:
                              fLowerLimit = VERBAL_ADJECTIVE_PA1_START
                              fUpperLimit = VERBAL_ADJECTIVE_PA1_END

                         'past active reflexive becomes non-reflexive
                         Case VERBAL_ADJECTIVE_AA2_START:
                              fLowerLimit = VERBAL_ADJECTIVE_PA1_START
                              fUpperLimit = VERBAL_ADJECTIVE_PA1_END
                  End Select

                  'set entry fields to non-reflexive
                  pnlGeneral.Visible = False
                  pnlEnglishVerbs.Visible = True
```

```
'active reflexive verbal adjective so make sure non-reflexive unselected
Case 1: 'don't allow switch to reflexive to either passive form selected
        If lstPrimaryAttributes.Selected(4) Or lstPrimaryAttributes.Selected(6) Then
           lstPrimaryAttributes.Selected(0) = True
           lstPrimaryAttributes.Selected(1) = False
           lstPrimaryAttributes.Selected(2) = False

        '...process switch
        Else
           lstPrimaryAttributes.Selected(0) = False
           lstPrimaryAttributes.Selected(2) = False

           'determine non-reflexive range for selected verbal adjective type
           Select Case fLowerLimit

                  'present active non-reflexive becomes reflexive
                  Case VERBAL_ADJECTIVE_PA1_START:
                       fLowerLimit = VERBAL_ADJECTIVE_PA2_START
                       fUpperLimit = VERBAL_ADJECTIVE_PA2_END

                  'past active non-reflexive becomes reflexive
                  Case VERBAL_ADJECTIVE_AA2_START:
                       fLowerLimit = VERBAL_ADJECTIVE_PA2_START
                       fUpperLimit = VERBAL_ADJECTIVE_PA2_END
           End Select
        End If

        'set entry fields to reflexive active
        pnlGeneral.Visible = False
        pnlEnglishVerbs.Visible = True

'passive reflexive verbal adjective so make sure non-reflexive unselected
Case 2: 'don't allow switch to reflexive to either passive form selected
        If lstPrimaryAttributes.Selected(4) Or lstPrimaryAttributes.Selected(6) Then
           lstPrimaryAttributes.Selected(0) = True
           lstPrimaryAttributes.Selected(1) = False
           lstPrimaryAttributes.Selected(2) = False

        '...process switch
        Else
           lstPrimaryAttributes.Selected(0) = False
           lstPrimaryAttributes.Selected(1) = False

           'determine non-reflexive range for selected verbal adjective type
           Select Case fLowerLimit
```

```
                    'present active non-reflexive becomes reflexive
                    Case VERBAL_ADJECTIVE_PA1_START:
                         fLowerLimit = VERBAL_ADJECTIVE_PA2_START
                         fUpperLimit = VERBAL_ADJECTIVE_PA2_END

                    'past active non-reflexive becomes reflexive
                    Case VERBAL_ADJECTIVE_AA2_START:
                         fLowerLimit = VERBAL_ADJECTIVE_PA2_START
                         fUpperLimit = VERBAL_ADJECTIVE_PA2_END
               End Select

               'display entry field for passive reflexive
               pnlEnglishVerbs.Visible = False
               lblGeneral.Caption = "Past Participle:"
               pnlGeneral.Visible = True
            End If

    'present passive verbal adjective
    Case 3: 'get inflections for non-reflexive...
            If lstPrimaryAttributes.Selected(0) Then
               fLowerLimit = VERBAL_ADJECTIVE_PA1_START
               fUpperLimit = VERBAL_ADJECTIVE_PA1_END
            '...or for reflexive
            Else
               fLowerLimit = VERBAL_ADJECTIVE_PA2_START
               fUpperLimit = VERBAL_ADJECTIVE_PA2_END
            End If
            'enforce mutual exclusion on verbal adjective type
            lstPrimaryAttributes.Selected(4) = False
            lstPrimaryAttributes.Selected(5) = False
            lstPrimaryAttributes.Selected(6) = False

    'present passive verbal adjective
    Case 4: fLowerLimit = VERBAL_ADJECTIVE_PP_START
            fUpperLimit = VERBAL_ADJECTIVE_PP_END
            'enforce mutual exclusion on verbal adjective type
            lstPrimaryAttributes.Selected(0) = True   'present passives must be non-reflexive
            lstPrimaryAttributes.Selected(1) = False
            lstPrimaryAttributes.Selected(2) = False
            lstPrimaryAttributes.Selected(3) = False
            lstPrimaryAttributes.Selected(5) = False
            lstPrimaryAttributes.Selected(6) = False

    'past active verbal adjective
    Case 5: 'get inflections for non-reflexive...
```

```
                    If lstPrimaryAttributes.Selected(0) Then
                       fLowerLimit = VERBAL_ADJECTIVE_AA1_START
                       fUpperLimit = VERBAL_ADJECTIVE_AA1_END
                    '...or for reflexive
                    Else
                       fLowerLimit = VERBAL_ADJECTIVE_AA2_START
                       fUpperLimit = VERBAL_ADJECTIVE_AA2_END
                    End If
                    'enforce mutual exclusion on verbal adjective type
                    lstPrimaryAttributes.Selected(3) = False
                    lstPrimaryAttributes.Selected(4) = False
                    lstPrimaryAttributes.Selected(6) = False

             'past passive verbal adjective
             Case 6: fLowerLimit = VERBAL_ADJECTIVE_AP_START
                     fUpperLimit = VERBAL_ADJECTIVE_AP_END

                     'enforce mutual exclusion on verbal adjective type
                     lstPrimaryAttributes.Selected(0) = True   'past passives must be non-reflexive
                     lstPrimaryAttributes.Selected(1) = False
                     lstPrimaryAttributes.Selected(2) = False
                     lstPrimaryAttributes.Selected(3) = False
                     lstPrimaryAttributes.Selected(4) = False
                     lstPrimaryAttributes.Selected(5) = False
      End Select

      'set starting inflection pattern number
      txtInflectionNumber = fLowerLimit

   'determine inflection pattern range depending on noun gender/number
   ElseIf         (lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)         =          NOUN_ANIMATE)          Or
(lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = NOUN_INANIMATE) Then
      'set range
      Select Case lstPrimaryAttributes.ListIndex:

             'masculine
             Case 0: fLowerLimit = MASCULINE_NOUN_START
                     fUpperLimit = MASCULINE_NOUN_END

             'feminine
             Case 1: fLowerLimit = FEMININE_NOUN_START
                     fUpperLimit = FEMININE_NOUN_END

             'neuter
             Case 2: fLowerLimit = NEUTER_NOUN_START
                     fUpperLimit = NEUTER_NOUN_END
```

```
          'plural
          Case 3: fLowerLimit = PLURAL_NOUN_START
                  fUpperLimit = PLURAL_NOUN_END
   End Select

   'set starting inflection pattern number
   txtInflectionNumber = fLowerLimit

'enforce mutual exclusion on verb primary attributes
ElseIf lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = VERB Then
   Select Case lstPrimaryAttributes.ListIndex:
          'main/auxiliary
          Case 0: lstPrimaryAttributes.Selected(1) = False
          Case 1: lstPrimaryAttributes.Selected(0) = False

          'imperfective/perfective
          Case 2: lstPrimaryAttributes.Selected(3) = False
                  pnlPerfectiveVerbs.Visible = False
                  pnlEnglishVerbs.Visible = True
          Case 3: lstPrimaryAttributes.Selected(2) = False
                  pnlEnglishVerbs.Visible = False
                  pnlPerfectiveVerbs.Visible = True

          'non-reflexive/reflexive active/reflexive passive
          Case 4: lstPrimaryAttributes.Selected(5) = False
                  lstPrimaryAttributes.Selected(6) = False
                  If lstPrimaryAttributes.Selected(3) = False Then
                     pnlEnglishVerbs.Visible = True
                  End If
          Case 5: lstPrimaryAttributes.Selected(4) = False
                  lstPrimaryAttributes.Selected(6) = False
                  If lstPrimaryAttributes.Selected(3) = False Then
                     pnlEnglishVerbs.Visible = True
                  End If
          Case 6: lstPrimaryAttributes.Selected(4) = False
                  lstPrimaryAttributes.Selected(5) = False
                  'display entry field for passive reflexive
                  pnlEnglishVerbs.Visible = False
                  lblGeneral.Caption = "Past Participle:"
                  pnlGeneral.Visible = True

          'indeterminate/determinate
          Case 7: lstPrimaryAttributes.Selected(8) = False
                  pnlPerfectiveVerbs.Visible = False
                  pnlEnglishVerbs.Visible = True
```

```
            Case 8: lstPrimaryAttributes.Selected(7) = False
                    pnlPerfectiveVerbs.Visible = False
                    pnlEnglishVerbs.Visible = True
       End Select

    'determine inflection pattern range depending on verbal adverb type
    ElseIf lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = VERBAL_ADVERB Then
       If lstPrimaryAttributes.Selected(0) Then
          fLowerLimit = VERBAL_ADVERB_NON_REFLEXIVE_START
          fUpperLimit = VERBAL_ADVERB_NON_REFLEXIVE_END
       '...or for reflexive
       Else
          fLowerLimit = VERBAL_ADVERB_REFLEXIVE_START
          fUpperLimit = VERBAL_ADVERB_REFLEXIVE_END
       End If

       'set starting inflection pattern number
       txtInflectionNumber = fLowerLimit

    'enforce mutual exclusion on personal pronouns
    ElseIf lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = PRONOUN_PERSONAL Then
       'unselect all except current
       For i = 0 To lstPrimaryAttributes.ListCount - 1
           If i <> lstPrimaryAttributes.ListIndex Then
              lstPrimaryAttributes.Selected(i) = False
           End If
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstPrimaryAttributes_KeyPress (KeyAscii As Integer)

    'carriage return moves to 'Next' button
    If KeyAscii = 13 Then
       KeyAscii = 0
       cmdNext.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstPrimaryAttributes_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

    Dim i As Integer
```

```
    'enforce mutual exclusion on primary attributes, except verb, verbal adjective, and verb adverb
    If            (lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)            <>            VERB)            And
(lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex)              <>              VERBAL_ADJECTIVE)              And
(lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) <> VERBAL_ADVERB) Then
       'reset other selections
       For i = 0 To lstPrimaryAttributes.ListCount - 1
           If i <> lstPrimaryAttributes.ListIndex Then
              lstPrimaryAttributes.Selected(i) = False
           End If
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstPrimaryAttributes_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    PassiveHelp "Select the primary attribute(s) for this Russian entry."

End Sub

'_____________________________________________________________________________________________________________
Sub lstSecondaryAttributes_Click ()

    Dim i As Integer


    'reset current inflections
    ClearInflectedForms

    'enforce mutual exclusion on preposition secondary attribute
    If lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex) = PREPOSITION Then
       'reset other selections
       For i = 0 To lstSecondaryAttributes.ListCount - 1
           If i <> lstSecondaryAttributes.ListIndex Then
              lstSecondaryAttributes.Selected(i) = False
           End If
       Next i
    End If

End Sub

'_____________________________________________________________________________________________________________
```

```
Sub lstSecondaryAttributes_KeyPress (KeyAscii As Integer)

    'carriage return moves to 'Next' button
    If KeyAscii = 13 Then
       KeyAscii = 0
       cmdNext.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub lstSecondaryAttributes_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    PassiveHelp "Select the secondary attribute(s) for this Russian entry."

End Sub

'_____________________________________________________________________________________________________________
'enabled/disable part of speech list box
'_____________________________________________________________________________________________________________
Sub PartOfSpeech (State As Integer)

    lstPartOfSpeech.Enabled = State

    If State Then
       'enable field
       lblPartOfSpeech.ForeColor = BLACK
       lstPartOfSpeech.BackColor = CREAM
    Else
       'disable field
       lstPartOfSpeech.ListIndex = -1
       lblPartOfSpeech.ForeColor = SAND
       lstPartOfSpeech.BackColor = LTSAND
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub pnlEnglish_DragDrop (Source As Control, X As Single, Y As Single)

    PassiveHelp ""

End Sub
```

```
'_____________________________________________________________________________________________________
Sub pnlEnglish_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub pnlRoot_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub pnlRussianPartI_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
Sub pnlRussianPartII_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    PassiveHelp ""

End Sub

'_____________________________________________________________________________________________________
'enabled/disable primary attribute list box
'_____________________________________________________________________________________________________
Sub PrimaryAttributes (State As Integer)

    lstPrimaryAttributes.Enabled = State

    If State Then
       'enable field
       lblPrimaryAttributes.ForeColor = BLACK
       lstPrimaryAttributes.BackColor = CREAM
    Else
       'disable field
       lblPrimaryAttributes.ForeColor = SAND
       lstPrimaryAttributes.Clear
       lstPrimaryAttributes.BackColor = LTSAND
    End If
```

```
End Sub

'_____________________________________________________________________________________________
'initialize or reset after cancel command buttons
'_____________________________________________________________________________________________
Sub ResetCommands ()

    'reset controls
    cmdAddNew.Enabled = True
    cmdLookup.Enabled = True
    cmdDelete.Enabled = False
    cmdUpdate.Enabled = False
    cmdCancel.Enabled = False
    cmdNext.Enabled = False
    cmdBack.Enabled = False
    cmdClose.Enabled = True

End Sub

'_____________________________________________________________________________________________
'enabled/disable secondary attributes list box
'_____________________________________________________________________________________________
Sub SecondaryAttributes (State As Integer)

    lstSecondaryAttributes.Enabled = State

    If State Then
       'enable field
       lblSecondaryAttributes.ForeColor = BLACK
       lstSecondaryAttributes.BackColor = CREAM
    Else
       'disable field
       lblSecondaryAttributes.ForeColor = SAND
       lstSecondaryAttributes.Clear
       lstSecondaryAttributes.BackColor = LTSAND
    End If

End Sub



'_____________________________________________________________________________________________
'make 1st stage folder active
'_____________________________________________________________________________________________
```

```
Sub Setup1stStage ()

    'setup 1st stage folder
    pnlFolderTab(0).ForeColor = BLACK
    pnlFolderTab(1).ForeColor = DKGRAY
    pnlFolderTab(2).ForeColor = DKGRAY
    timerLimit.Enabled = False
    pnlRussianPartI.ZOrder 0

    'set focus to stem entry field only if available
    If txtStem.Enabled Then
       txtStem.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________
'make 2nd stage folder active
'_____________________________________________________________________________________________________
Sub Setup2ndStage ()

    'setup 2nd stage folder
    pnlFolderTab(0).ForeColor = DKGRAY
    pnlFolderTab(1).ForeColor = BLACK
    pnlFolderTab(2).ForeColor = DKGRAY
    timerLimit.Enabled = True
    pnlRussianPartII.ZOrder 0

    'generate inflected forms of stem
    GenerateInflectedForms

    'set focus on inflection number field
    txtInflectionNumber.SelStart = 0
    txtInflectionNumber.SelLength = Len(txtInflectionNumber)
    txtInflectionNumber.SetFocus

End Sub

'_____________________________________________________________________________________________________
'make 3rd stage folder active
'_____________________________________________________________________________________________________
Sub Setup3rdStage ()

    'setup 3rd stage folder
    pnlFolderTab(0).ForeColor = DKGRAY
```

```
    pnlFolderTab(1).ForeColor = DKGRAY
    pnlFolderTab(2).ForeColor = BLACK
    timerLimit.Enabled = False
    pnlEnglish.ZOrder 0

    'set focus to first available field
    If pnlEnglishNouns.Visible Then
       txtSingular.SetFocus
    ElseIf pnlGeneral.Visible Then
       txtEnglishWord.SetFocus
    ElseIf pnlEnglishVerbs.Visible Then
       txtPresentIndicative2nd.SetFocus
    ElseIf pnlPerfectiveVerbs.Visible Then
       txtPerfectiveIndicative.SetFocus
    Else
       txtVAdvProgressive.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub spinPattern_SpinDown ()

    'decrement to previous pattern, if not at first
    If (fInflectionNum > fLowerLimit) Then
       fInflectionNum = fInflectionNum - 1
       txtInflectionNumber = Trim(Str(fInflectionNum))
    End If
    DoEvents

End Sub

'_____________________________________________________________________________________________________________
Sub spinPattern_SpinUp ()

    'increment to next pattern, if not at last
    If (fInflectionNum < fUpperLimit) Or ((fInflectionNum < fUpperLimit2) And (fUpperLimit2 > 0)) Then
       fInflectionNum = fInflectionNum + 1
       txtInflectionNumber = Trim(Str(fInflectionNum))
    End If
    DoEvents

End Sub

'_____________________________________________________________________________________________________________
'enabled/disable stem entry field
```

```
'_____________________________________________________________________________________________________
Sub Stem (State As Integer)

    txtStem.Enabled = State

    If State Then
       'enable field
       lblStem.ForeColor = BLACK
       pnlStem.BackColor = CREAM
       txtStem.BackColor = CREAM
    Else
       'disable field
       lblStem.ForeColor = SAND
       pnlStem.BackColor = LTSAND
       txtStem.Text = ""
       txtStem.BackColor = LTSAND
    End If

End Sub

'_____________________________________________________________________________________________________
'error check inflection number every second.  If it's out of range, set it to closest limit (high/low).
'This allows enough time to have illegal number with intention of fixing it (ie. 56 > 156 > 15).  Without
'this delay, as soon as the number went out of range, it would be set to the closet limit.
'_____________________________________________________________________________________________________
Sub timerLimit_Timer ()

    'highlight current entry
    txtInflectionNumber.SelStart = 0
    txtInflectionNumber.SelLength = Len(txtInflectionNumber)

    'inflection exceeds lower limit
    If (fInflectionNum < fLowerLimit) And (fInflectionNum > 0) Then
       fInflectionNum = fLowerLimit
       'update inflection number
       txtInflectionNumber = Trim(Str(fInflectionNum))
       'update inflection table with new inflection pattern
       GenerateInflectedForms
    'inflection exceeds upper limit
    ElseIf (fInflectionNum > fUpperLimit) Then
       fInflectionNum = fUpperLimit
       'update inflection number
       txtInflectionNumber = Trim(Str(fInflectionNum))
       'update inflection table with new inflection pattern
       GenerateInflectedForms
    End If
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub txtDisambiguationExpression_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to 'Next' button
    If KeyAscii = 13 Then
       KeyAscii = 0
       cmdNext.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtDisambiguationExpression_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    PassiveHelp "Enter the disambiguation expression for this Russian entry."

End Sub

'_____________________________________________________________________________________________________________
Sub txtEnglishWord_Change ()

    'prevent leading space(s)
    txtEnglishWord = LTrim(txtEnglishWord)

    'update command availability depends on field being filled
    cmdUpdate.Enabled = (txtEnglishWord <> "")

End Sub

'_____________________________________________________________________________________________________________
Sub txtEnglishWord_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If
```

```
    'carriage return moves to update command if available
    If KeyAscii = 13 Then
       KeyAscii = 0
       If cmdUpdate.Enabled Then
          cmdUpdate.SetFocus
       End If
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtEnglishWord_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the English equivalent of the Russian entry."

End Sub

'_____________________________________________________________________________________________________________
Sub txtImperfectIndicative_Change ()

    'prevent leading space(s)
    txtImperfectIndicative = LTrim(txtImperfectIndicative)

    'determine if update command available
    UpdateOnImperfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtImperfectIndicative_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtProgressive.SetFocus
    End If

End Sub
```

```
'_____________________________________________________________________________________________________
Sub txtImperfectIndicative_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the simple past tense form of the current verb.  For example: I 'swam'."

End Sub

'_____________________________________________________________________________________________________
'allow user to enter pattern number or spin up/down, skipping gaps in range
'_____________________________________________________________________________________________________
Sub txtInflectionNumber_Change ()

    fInflectionNum = Val(txtInflectionNumber)

    'record valid inflection number being entered
    If (fInflectionNum >= fLowerLimit) And (fInflectionNum <= fUpperLimit) Then
       'update inflection table with new inflection pattern
       GenerateInflectedForms
    'check if entire field deleted (ie. typed over highlighted string)
    ElseIf fInflectionNum = 0 Then
       txtInflectionNumber = ""
       ClearInflectedForms
    End If

End Sub

'_____________________________________________________________________________________________________
Sub txtInflectionNumber_KeyDown (KeyCode As Integer, Shift As Integer)

    'down-arrow decrements inflection number
    If KeyCode = 40 Then
       spinPattern_SpinDown
    'up-arrow increments inflection number
    ElseIf KeyCode = 38 Then
       spinPattern_SpinUp
    End If

End Sub

'_____________________________________________________________________________________________________
Sub txtInflectionNumber_KeyPress (KeyAscii As Integer)

    'mask invalid characters
    If InStr("0123456789" & Chr(13) & Chr(8), Chr(KeyAscii)) = 0 Then
```

```
      Beep
      KeyAscii = 0
      MsgBox ("Only numbers are valid in this field."), MB_ICONSTOP, ("Invalid Entry")
   End If

   'carriage return moves to Next command
   If KeyAscii = 13 Then
      KeyAscii = 0
      cmdNext.SetFocus
   End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtInflectionNumber_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'display help on current object
   PassiveHelp "Enter the inflection code number for this word (" & fLowerLimit & ".." & fUpperLimit & ")."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPastParticiple_Change ()

   'prevent leading space(s)
   txtPastParticiple = LTrim(txtPastParticiple)

   'determine if update command available
   UpdateOnImperfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPastParticiple_KeyPress (KeyAscii As Integer)

   'convert double-quotes to single-quotes
   If KeyAscii = 34 Then
      KeyAscii = Asc("'")
   End If

   'process carriage return
   If KeyAscii = 13 Then
      KeyAscii = 0
      'if update command available, set focus to it; otherwise...
      If cmdUpdate.Enabled Then
         cmdUpdate.SetFocus
```

```
      '...set focus to first entry field not filled in
      Else
         If txtPresentIndicative2nd = "" Then
            txtPresentIndicative2nd.SetFocus
         ElseIf txtPresentIndicative3rd = "" Then
            txtPresentIndicative3rd.SetFocus
         ElseIf txtImperfectIndicative = "" Then
            txtImperfectIndicative.SetFocus
         ElseIf txtProgressive = "" Then
            txtProgressive.SetFocus
         Else
            txtPastParticiple.SetFocus
         End If
      End If
   End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPastParticiple_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

   'display help on current object
   PassiveHelp "Enter the past participle form of the current verb.  For example: I have 'swum'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveImperfect_Change ()

   'prevent leading space(s)
   txtPerfectiveImperfect = LTrim(txtPerfectiveImperfect)

   'determine if update command available
   UpdateOnPerfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveImperfect_KeyPress (KeyAscii As Integer)

   'convert double-quotes to single-quotes
   If KeyAscii = 34 Then
      KeyAscii = Asc("'")
   End If

   'carriage return moves to next field
```

```
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPerfectivePastParticiple.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveImperfect_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the simple past tense form of the current verb.  For example: I 'swam'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveIndicative_Change ()

    'prevent leading space(s)
    txtPerfectiveIndicative = LTrim(txtPerfectiveIndicative)

    'determine if update command available
    UpdateOnPerfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveIndicative_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPerfectiveImperfect.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectiveIndicative_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    'display help on current object
    PassiveHelp "Enter the present tense 'you' form of the current verb.  For example: You 'swim'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectivePastParticiple_Change ()

    'prevent leading space(s)
    txtPerfectivePastParticiple = LTrim(txtPerfectivePastParticiple)

    'determine if update command available
    UpdateOnPerfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectivePastParticiple_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'if update command available, set focus to it; otherwise...
       If cmdUpdate.Enabled Then
          cmdUpdate.SetFocus
       '...set focus to first entry field not filled in
       Else
          If txtPerfectiveIndicative = "" Then
             txtPerfectiveIndicative.SetFocus
          ElseIf txtPerfectiveImperfect = "" Then
             txtPerfectiveImperfect.SetFocus
          Else
             txtPerfectivePastParticiple.SetFocus
          End If
       End If
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPerfectivePastParticiple_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    'display help on current object
    PassiveHelp "Enter the past participle form of the current verb.  For example: I have 'swum'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPlural_Change ()

    'prevent leading space(s)
    txtSingular = LTrim(txtSingular)

    'update command availability depends on all fields being filled
    cmdUpdate.Enabled = (txtSingular <> "") And (txtPlural <> "")

End Sub

'_____________________________________________________________________________________________________________
Sub txtPlural_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to article list
    If KeyAscii = 13 Then
       KeyAscii = 0
       lstArticles.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPlural_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the plural form of the current noun.  For example: 'dogs'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative2nd_Change ()

    'prevent leading space(s)
```

```
    txtPresentIndicative2nd = LTrim(txtPresentIndicative2nd)

    'determine if update command available
    UpdateOnImperfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative2nd_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPresentIndicative3rd.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative2nd_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the present tense 'you' form of the current verb.  For example: You 'swim'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative3rd_Change ()

    'prevent leading space(s)
    txtPresentIndicative3rd = LTrim(txtPresentIndicative3rd)

    'determine if update command available
    UpdateOnImperfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative3rd_KeyPress (KeyAscii As Integer)
```

```
    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtImperfectIndicative.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtPresentIndicative3rd_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the present tense 'he' form of the current verb.  For example: He 'swims'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtProgressive_Change ()

    'prevent leading space(s)
    txtProgressive = LTrim(txtProgressive)

    'determine if update command available
    UpdateOnImperfectiveVerb

End Sub

'_____________________________________________________________________________________________________________
Sub txtProgressive_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPastParticiple.SetFocus
    End If
```

```
End Sub

'_____________________________________________________________________________________________________________
Sub txtProgressive_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the progressive tense form of the current verb.  For example: I am 'swimming'."

End Sub

'_____________________________________________________________________________________________________________
Sub txtSingular_Change ()

    'prevent leading space(s)
    txtSingular = LTrim(txtSingular)

    'update command availability depends on all fields being filled
    cmdUpdate.Enabled = (txtSingular <> "") And (txtPlural <> "")

End Sub

'_____________________________________________________________________________________________________________
Sub txtSingular_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to plural field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtPlural.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtSingular_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the singular form of the current noun.  For example: 'dog'."

End Sub
```

```
'_____________________________________________________________________________________________
'make part of speech list box available when stem entered
'_____________________________________________________________________________________________
Sub txtStem_Change ()

    Static PrevStem As String


    'prevent leading space(s)
    txtStem = LTrim(txtStem)

    'if adding new entry
    If fMode = ADD Then
       'make part of speech list box available only if stem present
       If txtStem <> "" Then
          If PrevStem = "" Then
             'enable only once to prevent flashing
             lstPartOfSpeech.ListIndex = 0
             PartOfSpeech True
          End If
       'deactivate because stem deleted
       Else
          PartOfSpeech False
          PrimaryAttributes False
          SecondaryAttributes False
          Associations False
          DisambiguationExpression False
       End If

       'record current stem
       PrevStem = txtStem
    End If

End Sub

'_____________________________________________________________________________________________
Sub txtStem_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'show help on info line
    If fMode = ADD Then
       PassiveHelp "Enter a word to add to the lexicon.  To switch keyboard layouts, press CTRL+SHIFT."
    Else
       PassiveHelp "Enter a word to look up in the lexicon.  To switch keyboard layouts, press CTRL+SHIFT."
    End If

End Sub
```

```
'____________________________________________________________________________________________________
Sub txtVAdvPastParticiple_Change ()

    'prevent leading space(s)
    txtVAdvPastParticiple = LTrim(txtVAdvPastParticiple)

    'update command availability depends on all fields being filled
    cmdUpdate.Enabled = (txtVAdvProgressive <> "") And (txtVAdvPastParticiple <> "")

End Sub

'____________________________________________________________________________________________________
Sub txtVAdvPastParticiple_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'if update command available, set focus to it; otherwise...
       If cmdUpdate.Enabled Then
          cmdUpdate.SetFocus
       '...set focus to first entry field not filled in
       Else
          If txtVAdvProgressive = "" Then
             txtVAdvProgressive.SetFocus
          Else
             txtVAdvPastParticiple.SetFocus
          End If
       End If
    End If

End Sub

'____________________________________________________________________________________________________
Sub txtVAdvPastParticiple_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the past participle form of the current verb.  For example: I have 'swum'."

End Sub
```

```
'_____________________________________________________________________________________________________________
Sub txtVAdvProgressive_Change ()

    'prevent leading space(s)
    txtVAdvProgressive = LTrim(txtVAdvProgressive)

    'update command availability depends on all fields being filled
    cmdUpdate.Enabled = (txtVAdvProgressive <> "") And (txtVAdvPastParticiple <> "")

End Sub

'_____________________________________________________________________________________________________________
Sub txtVAdvProgressive_KeyPress (KeyAscii As Integer)

    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'carriage return moves to next field
    If KeyAscii = 13 Then
       KeyAscii = 0
       txtVAdvPastParticiple.SetFocus
    End If

End Sub

'_____________________________________________________________________________________________________________
Sub txtVAdvProgressive_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

    'display help on current object
    PassiveHelp "Enter the progressive tense form of the current verb.  For example: I am 'swimming'."

End Sub

'_____________________________________________________________________________________________________________
'update command availability depends on all english verb form fields being completed
'_____________________________________________________________________________________________________________
Sub UpdateOnImperfectiveVerb ()

    cmdUpdate.Enabled   =   (txtPresentIndicative2nd   <>   "")   And   (txtPresentIndicative3rd   <>   "")   And
(txtImperfectIndicative <> "") And (txtProgressive <> "") And (txtPastParticiple <> "")

End Sub
```

```
'_____________________________________________________________________________________________
'update command availability depends on all english verb form fields being completed
'_____________________________________________________________________________________________
Sub UpdateOnPerfectiveVerb ()

    cmdUpdate.Enabled    =    (txtPerfectiveIndicative    <>    "")    And    (txtPerfectiveImperfect    <>    "")    And
(txtPerfectivePastParticiple <> "")

End Sub

'_____________________________________________________________________________________________
'process carriage return:  initiate Add New or Lookup
'_____________________________________________________________________________________________
Sub txtStem_KeyPress (KeyAscii As Integer)

    Dim BindPtr          As Integer
    Dim ListPtr          As Integer
    Dim MatchPtr         As Integer
    Dim Attribute        As Integer
    Dim PluralCode       As Integer
    Dim DelimiterAt      As Integer
    Dim SingularLength   As Integer
    Dim SingularForm     As String
    Dim TempTarget       As String
    Dim Entry            As String
    Dim SampleInflection As String
    Dim TransferDetails  As TransferType

    On Error GoTo FormattingError


    'convert double-quotes to single-quotes
    If KeyAscii = 34 Then
       KeyAscii = Asc("'")
    End If

    'process carriage return
    If KeyAscii = 13 Then
       KeyAscii = 0
       'continue only when text present
       If txtStem <> "" Then

          'if adding new entry; otherwise...
          If fMode = ADD Then
             'set focus to part of speech list, if available
```

```
            If lstPartOfSpeech.Enabled Then
               lstPartOfSpeech.SetFocus
            End If
            Exit Sub

         '...looking up existing entry
         Else
            LookupWord (txtStem), TransferDetails, True

            'single match, so display entry info in lexicon editor; otherwise...
            If TransferDetails.NumMatches = 1 Then
               'get first and only entry
               BindPtr = 1
               fBindPtr = TransferDetails.Match(BindPtr).LexiconEntryPtr
               GoSub DisplayEntryInfo

               'delete now available
               cmdDelete.Enabled = True

            '...if more than one match, build and display selection list; otherwise...
            ElseIf TransferDetails.NumMatches > 1 Then
               'clear previous list
               frmLookupResults.lstResults(0).Clear
               frmLookupResults.lstResults(1).Clear
               'load lists with current results
               For MatchPtr = 1 To TransferDetails.NumMatches
                   'set stem as start of possible concentation sequence
                   Entry = TransferDetails.Match(MatchPtr).LexiconEntry.Stem
                   'if entry takes ending, append ... (peculiar expression avoids huge if-then)
                   Select Case TransferDetails.Match(MatchPtr).LexiconEntry.PartOfSpeech:
                          Case PREPOSITION, INTERJECTION, EMPHATIC_PARTICLE, ADVERB, CONJUNCTION
                          Case Else:
                              'get first inflection to indicate more than just stem on list
                               SampleInflection                    =                    "["                        &
Left(gPattern(TransferDetails.Match(MatchPtr).LexiconEntry.InflectionPattern),
InStr(gPattern(TransferDetails.Match(MatchPtr).LexiconEntry.InflectionPattern), INFLECTION_DELIMITER) - 1) & "]"
                               'if first inflection null or unused, don't print it
                               If (SampleInflection = "[#]") Or (SampleInflection = "[/]") Then
                                  SampleInflection = ""
                               End If
                               'build remainder of sequence
                               Entry = Entry & SampleInflection & "..."
                   End Select
                   frmLookupResults.lstResults(0).AddItem " " & Entry
                   frmLookupResults.lstResults(1).AddItem " " & TransferDetails.Match(MatchPtr).LexiconEntry.Target
               Next MatchPtr
```

```
            frmLookupResults.Show 1

            'if user ok'd lookup, display selected entry
            If frmLookupResults.Tag = "[OK]" Then
               'get selected entry
               BindPtr = frmLookupResults.lstResults(0).ListIndex + 1
               fBindPtr = TransferDetails.Match(BindPtr).LexiconEntryPtr
               GoSub DisplayEntryInfo
            End If

            'delete now available
            cmdDelete.Enabled = True

         '...no match
         Else
            Beep
            MsgBox ("This entry is not present in lexicon."), MB_ICONSTOP, ("Lookup Failure")
         End If
      End If

   'inform of empty entry
   Else
      Beep
      MsgBox ("You must enter a stem."), MB_ICONSTOP, ("Missing Entry")
   End If
End If

Exit Sub


DisplayEntryInfo:

    'fill in Lexicon Editor fields with current entry info
    txtStem = TransferDetails.Match(BindPtr).LexiconEntry.Stem
    PluralCode = TransferDetails.Match(BindPtr).LexiconEntry.TransferAttributes

    'set part of speech
    PartOfSpeech True
    For ListPtr = 0 To lstPartOfSpeech.ListCount - 1
        If lstPartOfSpeech.ItemData(ListPtr) = TransferDetails.Match(BindPtr).LexiconEntry.PartOfSpeech Then
           lstPartOfSpeech.ListIndex = ListPtr
           Exit For
        End If
    Next ListPtr

    'set primary attribute(s)
```

```
    Select Case lstPartOfSpeech.ItemData(lstPartOfSpeech.ListIndex):

           'set primary attribute for combinational (non-mutually exclusive) attributes
           Case VERB, VERBAL_ADVERB, VERBAL_ADJECTIVE:
                For ListPtr = 0 To lstPrimaryAttributes.ListCount - 1
                    Attribute = TransferDetails.Match(BindPtr).LexiconEntry.PrimaryAttributes
                    If (Attribute And lstPrimaryAttributes.ItemData(ListPtr)) Then
                       lstPrimaryAttributes.Selected(ListPtr) = True
                       lstPrimaryAttributes.ListIndex = ListPtr
                    End If
                Next ListPtr

           'set primary attribute for non-combinational (mutually exclusive) attributes
           Case Else:
                For ListPtr = 0 To lstPrimaryAttributes.ListCount - 1
                    Attribute = TransferDetails.Match(BindPtr).LexiconEntry.PrimaryAttributes
                    'select attribute
                    If lstPrimaryAttributes.ItemData(ListPtr) = Attribute Then
                       lstPrimaryAttributes.Selected(ListPtr) = True
                       lstPrimaryAttributes.ListIndex = ListPtr
                    'clear unselected
                    Else
                       lstPrimaryAttributes.Selected(ListPtr) = False
                    End If
                Next ListPtr
    End Select

    'set secondary attribute(s)
    For ListPtr = 0 To lstSecondaryAttributes.ListCount - 1
        'select attribute
        If                    (TransferDetails.Match(BindPtr).LexiconEntry.SecondaryAttributes                    And
lstSecondaryAttributes.ItemData(ListPtr)) Then
           lstSecondaryAttributes.Selected(ListPtr) = True
           lstSecondaryAttributes.ListIndex = ListPtr
        'clear unselected
        Else
           lstSecondaryAttributes.Selected(ListPtr) = False
        End If
    Next ListPtr

    'set association attribute(s)
    For ListPtr = 0 To lstAssociations.ListCount - 1
        'select attribute
        If  (TransferDetails.Match(BindPtr).LexiconEntry.TransferAttributes  And  lstAssociations.ItemData(ListPtr))
Then
           lstAssociations.Selected(ListPtr) = True
```

```
            lstAssociations.ListIndex = ListPtr
            'decrement extracted code from transfer code
            PluralCode = PluralCode - lstAssociations.ItemData(ListPtr)
        'clear unselected
        Else
            lstAssociations.Selected(ListPtr) = False
        End If
    Next ListPtr

    'set target article attribute(s)
    For ListPtr = 0 To lstArticles.ListCount - 1
        'select attribute
        If (TransferDetails.Match(BindPtr).LexiconEntry.TransferAttributes And lstArticles.ItemData(ListPtr)) Then
            lstArticles.Selected(ListPtr) = True
            lstArticles.ListIndex = ListPtr
            'decrement extracted code from transfer code
            PluralCode = PluralCode - lstArticles.ItemData(ListPtr)
        'clear unselected
        Else
            lstArticles.Selected(ListPtr) = False
        End If
    Next ListPtr

    'set disambiguation expression
    txtDisambiguationExpression = TransferDetails.Match(BindPtr).LexiconEntry.Disambiguation

    'set inflection pattern
    txtInflectionNumber = TransferDetails.Match(BindPtr).LexiconEntry.InflectionPattern

    'set english forms
    Select Case TransferDetails.Match(BindPtr).LexiconEntry.PartOfSpeech:

            'pronouns, adjectives, adverbs, prepositions, interrogatives, interjections, emphatic particles,
conjunction, fixed expression
            Case    ADJECTIVE,    PRONOUN_PERSONAL,    PRONOUN_POSSESSIVE,    PRONOUN_INDEFINITE,    PRONOUN_RELATIVE,
PRONOUN_DEMONSTRATIVE,   ADVERB,   PREPOSITION,   INTERROGATIVE,   INTERJECTION,   EMPHATIC_PARTICLE,   CONJUNCTION,
FIXED_EXPRESSION:
                txtEnglishWord = TransferDetails.Match(BindPtr).LexiconEntry.Target

            Case VERBAL_ADVERB:
                txtVAdvProgressive            =            Left(TransferDetails.Match(BindPtr).LexiconEntry.Target,
InStr(TransferDetails.Match(BindPtr).LexiconEntry.Target, CONJUGATION_DELIMITER) - 1)
                txtVAdvPastParticiple           =           Mid(TransferDetails.Match(BindPtr).LexiconEntry.Target,
InStr(TransferDetails.Match(BindPtr).LexiconEntry.Target, CONJUGATION_DELIMITER) + 1)

            Case NOUN_INANIMATE, NOUN_ANIMATE:
```

```
      txtSingular = TransferDetails.Match(BindPtr).LexiconEntry.Target

      'generate plural
      SingularForm = TransferDetails.Match(BindPtr).LexiconEntry.Target
      SingularLength = Len(TransferDetails.Match(BindPtr).LexiconEntry.Target)

      'decode by subtraction since bitwise operations not applicable to plural decoding
      If (PluralCode - PLURAL11) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 4) & "eese"
      ElseIf (PluralCode - PLURAL10) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 4) & "ice"
      ElseIf (PluralCode - PLURAL9) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 2) & "en"
      ElseIf (PluralCode - PLURAL8) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 2) & "i"
      ElseIf (PluralCode - PLURAL7) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 2) & "a"
      ElseIf (PluralCode - PLURAL6) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 2) & "ves"
      ElseIf (PluralCode - PLURAL5) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 1) & "ves"
      ElseIf (PluralCode - PLURAL4) >= 0 Then
         txtPlural = Left(SingularForm, SingularLength - 1) & "ies"
      ElseIf (PluralCode - PLURAL3) >= 0 Then
         txtPlural = SingularForm & "es"
      ElseIf (PluralCode - PLURAL2) >= 0 Then
         txtPlural = SingularForm & "s"
      Else
         txtPlural = SingularForm
      End If

Case VERB, VERBAL_ADJECTIVE:
      'any passive reflexive
      If ((TransferDetails.Match(BindPtr).LexiconEntry.PartOfSpeech = VERB) And
          (lstPrimaryAttributes.Selected(6)) Or TransferDetails.Match(BindPtr).LexiconEntry.PartOfSpeech
          = VERBAL_ADJECTIVE)) And (lstPrimaryAttributes.Selected(2)) Then
         txtEnglishWord = TransferDetails.Match(BindPtr).LexiconEntry.Target

      'verbal adjective or imperfective verb
      ElseIf (lstPrimaryAttributes.Selected(2)) Or TransferDetails.Match(BindPtr).LexiconEntry.
              PartOfSpeech = VERBAL_ADJECTIVE) Then
         'extract field 1
         TempTarget = TransferDetails.Match(BindPtr).LexiconEntry.Target
         DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
         txtPresentIndicative2nd = Left(TempTarget, DelimiterAt - 1)
```

```
                'extract field 2
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
                txtPresentIndicative3rd = Left(TempTarget, DelimiterAt - 1)

                'extract field 3
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
                txtImperfectIndicative = Left(TempTarget, DelimiterAt - 1)

                'extract field 4
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
                 txtProgressive = Left(TempTarget, DelimiterAt - 1)

                'extract field 5
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                txtPastParticiple = TempTarget

            'perfective verb
            Else
                'extract field 1
                TempTarget = TransferDetails.Match(BindPtr).LexiconEntry.Target
                DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
                txtPerfectiveIndicative = Left(TempTarget, DelimiterAt - 1)

                'extract field 2
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                DelimiterAt = InStr(TempTarget, CONJUGATION_DELIMITER)
                txtPerfectiveImperfect = Left(TempTarget, DelimiterAt - 1)

                'extract field 3
                TempTarget = Mid(TempTarget, DelimiterAt + 1)
                txtPerfectivePastParticiple = TempTarget
            End If
    End Select

    Return


FormattingError:

    'exception handler
    Beep
    MsgBox ("The current entry is incorrectly formatted." & Chr(13) & "Please recheck its information."),
           MB_ICONSTOP, ("Lexicon Entry Error")
```

```
    Resume Next
End Sub
```