Lexical Analysis	Lexical analysis is the process of decomposing an input sequence of characters into one or more sequences of symbols called *lexical tokens* or just *tokens*. The purpose of producing these tokens is usually to pass them as input to another program, such as a *parser*.
Lexical Analyzer	A lexical analyzer is a sub-routine that performs the lexical analysis process. It is usually composed of two stages (i.e. *scanner* and *evaluator*). The scanner and evaluator stages are often integrated, for efficiency reasons. The *scanner* is responsible for accessing the string of data and for breaking the input sequence of characters into *lexemes*. The scanner is usually based on a finite state machine (FSM). It has encoded within it information on the possible sequences of characters that can be contained within any of the types of tokens it handles. The *evaluator* is responsible for classifying a lexeme into a token. The token is composed of type and lexeme. Type classifies the lexeme so that the parser has an easier job with syntactical analysis (e.g. key word name, variable name, number, special character, character string, hexadecimal string, etc.). Sometimes evaluators suppress a lexeme entirely, concealing it from the parser, which is useful for white-space and comments. A *lexeme* is a string of characters that can be defined as a subset of some alphabet. That alphabet may be a subset of the EBCDIC character set. The scanner may be written to decompose an input sequence of characters under the control of a *template* instead of using the finite state machine method. Following are the major categories of this type of scanner. • Tokenize via words • Tokenize via string pattern • Tokenize via numeric pattern A template list can contain any combination of the listed categories. The next section “String Actions” describes a set of specific implementations of scanners and/or lexical analyzers. This type of scanner seldom has an evaluator stage associated with it. Additionally, it is usually not responsible for accessing the input data stream. The caller generally provides the data to be scanned.
Tokenize via words	The simplest method is a template consisting of a list of result fields. The source field is decomposed into sub-strings (i.e. lexeme) delimited by the declared delimiter character (e.g. blank) and a list of optional delimiters, if specified. Each sub-string from the source field is assigned to a result field (i.e. lexeme) in the template list, in order of occurrence. Each result lexeme will have both leading and trailing strip characters (e.g. blank) removed. If there are not enough result fields to satisfy the complete consumption of the source

	field, the last result field will contain the remaining sub-string(s). In this case, the leading and trailing strip characters will not be removed. If there are more result fields in the template list than there are lexemes to fill them, the superfluous results fields will have length fields of zero.
Tokenize via String Pattern	This category consists of an input template containing a string pattern used to decompose the source field. The source sub-string immediately prior to the matched input template string pattern (not inclusive) is assigned to the previously specified result field template, if one exists. The source sub-string following the matched input template string pattern is assigned to the result field template following the input template string pattern, if one exists. Leading and trailing strip characters are not removed from the result templates. When a relative numeric pattern template follows the input template string pattern, the cursor position of the located character string becomes the reference cursor for all following relative numeric pattern templates. If the input template string pattern cannot be found in the source field, the previous result template is set to the remaining data in the source field, if it exists. The following result templates, if specified, are set to a zero length value. There are two sub-categories of this type pattern. • Literal string pattern • Variable string pattern
Literal String Pattern	This category consists of a literal string pattern included in the input template to decompose a source field. This category follows the rules previously specified.
Variable String Pattern	This category consists of a variable string pattern (address of) included in the input template to decompose a source field. This category follows the rules previously specified. The input template contains a negative length field (2-byte signed integer), and a 4-byte string containing the address of the desired string pattern. The length field will contain the length of the desired string pattern multiplied by minus 1.
Tokenize via Numeric Pattern	This general category of input template uses a signed or unsigned zone decimal column number included in the template to decompose a source field. The source sub-string prior to the column (not inclusive) is assigned to the previously specified result template, if one exists. The source sub-string beginning at the specified column number will be assigned to the result template following the numeric pattern template, if one exists. Leading and trailing strip characters are not removed from the result templates. If the numeric pattern references a position prior to the current cursor, the token

	will be assigned the remaining source field. If the numeric pattern template references a position beyond the source field, the result field template will be assigned the remaining source field, no padding will occur. There are two sub-categories of this type pattern. • Positional numeric pattern • Relative numeric pattern
Positional numeric pattern	This category of template specifies an unsigned zone decimal number as a numeric pattern in the template; the zone decimal number refers to a particular column in the source field, relative to the initial cursor position. The zone decimal number specifies the start column for the following token. This category follows the rules previously specified.
Relative numeric pattern	This category of template specifies a signed zone decimal number as a numeric pattern in the template; the zone decimal number refers to the number of columns to the left (-n) or right (+n) of the current cursor position, which is the cursor position of the previous token, in the source field. This category follows the rules previously specified.

String Actions	Define a set of string services similar to those provided by REXX built-in-functions (BIF) to act on character strings. In most cases, the following string actions are limited to acting on source fields that are 1:256 bytes in length. Some special cases support 1:32,767 bytes in length. All source fields are assumed to contain only single byte character set (SBCS) extended binary coded decimal interchange code (EBCDIC) characters. When specifying the AS LA Source field, the supplied data can be the source data or the address of the source field if the length field is negative. The source field is never modified in the storage that it resides. When specifying the AS LA Template input field(s), the supplied data can be the pattern specification field or the address of the pattern specification field when the length field is negative. If the AS LA Template result field contains a maximum field length value of zero, it will be used as a placeholder (i.e. dummy variable) and no data is returned. If the value is too small to hold the lexeme, a zero length result field is returned along with the appropriate error. All results fields will have a used length returned via the template. In some cases, the caller can specify an optional delimiter string as the first input template item. This string can contain up to 16 characters, used to

determine the end of a lexeme. The default *delimiter* character specified in the AS LA Request Options is also used.

The actions STRFPOS and STRLPOS present a special case. The first input template item can contain the least frequently used (LFU) frequency vector. This vector must be 256 bytes in length. Each byte represents a valid EBCDIC character. The value placed in each byte (one byte integer), is the frequency (0:255) that character is used. The STRFPOS and STRLPOS actions use this value to determine which byte in the search string to scan the source field for. The character with the lowest frequency value is used. The intent is to provide the fastest string search sub-routine possible. If not provided, a default LFU frequency vector will be used.

The IBM Language Environment (LE) Feedback Area is a 12-byte data structure (ADT) that contains the condition token. If any error conditions are encountered by the sub-routine, it will be placed in this, data structure. If this parameter is omitted the sub-routine will signal the condition and if the condition code is three or higher an ABEND will occur. Otherwise, the condition is returned to the caller and the caller can determine what actions are necessary to resolve the error.

Parameter list:

- AS Communication Area (data structure)
- AS LA Action (4-byte integer)
- AS LA Request Options (data structure)
- AS LA Source field (variable length string)
- AS LA Template count (4-byte integer)
- AS LA Template input or result (data structure)
  one to many occurrences as reflected by the AS LA Template count
- (optional) LE Feedback Area (data structure)

Request Options:

- Default characters:
  - *Delimiter* character (input): character that terminates sub-strings within the source character string. The default is a blank.
  - *Strip* character (input): character that precedes and /or succeeds the source character string. The default is a blank.
  - *Pad* character (output): character used to separate sub-strings within the result lexeme or precede and/or succeed the result lexeme depending on the action requested. The default is a blank.
- *Upper-case translation* of the source field is requested. The source field is copied to an internal work area prior to translation.
- *Case-insensitive* string match for Boyer-Moore.
- Remove *leading* and/or *trailing* strip characters from the source field.
- *Continue* the previous request.
- Initial *start column* or *word* in source field.
- *Count* field, number of words or characters.
- Persistent Storage Area (PSA) a variable length string (ADT). Contains

	a 2-byte integer length field and data area. If required, the minimum length is 32-bytes, the maximum length required is 288-bytes.
APPEND (05) ASUDAPND	Return a result field containing the source field appended to the result field. The result field can contain preexisting data. The result field can be up to 32,767 bytes in length. If the result field used length is zero, the result field will be initialized to the default *pad* character prior to any other action. If the *count* field is greater than zero, then the rest of the result field will be initialized using the default *pad* character prior to appending the source field. The used length of the result field will be returned. Request option(s), source field *upper case translation* and *count* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
CENTER (06) ASUDCNTR	Return a result field containing the source field centered within the bounds of the result field maximum size value. The used length of the result field will be returned. The default *pad* character is used to bracket the result field. It must be specified via the AS LA Request Options data structure. Request option(s), source field *upper case translation* and *start* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03H – 113 Too many arguments

	• ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
COUNT (07) ASUDSCNT	Return a result field containing the source field length (4-byte integer), character count (4-byte integer), and word count (4-byte integer). If two result fields are specified, a vector containing each valid delimited sub-string (offset, length) within the source field is returned. Each vector entry is four bytes in length. The offset is a 2-byte integer. The length is a 2-byte integer. The word count result field contains the number of vector entries. The internal vector used to keep track of each sub-string in the source string is a fixed length of 512 bytes. Which is enough space to track 128 sub-strings. The internal vector will be copied to the optional supplied result template, if present. Therefore, make sure that the second result field is a minimum of 512 bytes in length. Request option(s), *start* is supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template result field • (optional) AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
DEEDIT (08) ASUDDEDT	Return a result string containing the source field less the valid edit characters (i.e. blank, monetary sign, minus sign, comma, decimal point, and trailing characters G, M, K). The source string must be a valid edited numeric field. The result string format is controlled via the AS LA Template input field. The de-editing is managed by a finite state machine (FSM) macro routine. The used length of the result field will be returned. Request option(s), source field *upper case translation* is supported.

	Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03P – 121 Syntax error
EDIT (09)	
JUSTIFY (10) ASUDJSTF	Return a result field containing the source field extended to the bounds of the result field maximum size value, by adding an equal number of default *pad* character(s) between each of the words until the source field fills up the result field. The right most word in the source field may fall short of the right bounds of the result field. The used length of the result field will be returned. Request option(s), source field *upper case translation* is supported. Other actions referenced: • COUNT Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
LEFT (11)	Return a result field containing the source field with the leading default *strip*

ASUDLEFT	characters removed and trailing default *pad* characters added as needed to fulfill the result field maximum size value specified. The used length of the result field will be returned. Request option(s), source field *upper case translation* is supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
LEX (12)	
RIGHT (13) ASUDRGHT	Return a result field containing the source field with the trailing default *strip* characters removed and leading default *pad* characters added as needed to fulfill the result field maximum size value specified. The used length of the result field will be returned. Request option(s), source field *upper case translation* and *start* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow

SPACE (14) ASUDSPCE	Return a result field with the specified number of default *pad* characters between each word of the source field. The used length of the result field will be returned. The number/*count* of default *pad* characters must be specified via the AS LA Request Options data structure. Request option(s), source field *upper case translation* and *count* are supported. Other actions referenced: • COUNT Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
STRDEL (15) ASUDSDEL	Return a result field containing the source field less the string selected to be deleted. The string pattern to be deleted from the source field can be selected in one of two ways. • Specify the *start*ing position and number/*count* of characters to be deleted via the AS LA Request Options data structure. If the *count* (number of characters) to be deleted is zero, the rest of the source field will be deleted. • Specify a Template input field containing the string pattern to locate in the source field. The length of the string pattern will be used as the *count* or number of characters to delete. The used length of the result field will be returned. Request option(s), source field *upper case translation* or *case-insensitive* and *start*, *count* are supported. Other actions referenced: • STRBPOS Parameter list: • AS Communication Area

	• AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03E – 110 Not found • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
STRFPOS (16) ASUDFPOS	Return a result field containing the position (4-byte integer) of the string pattern specified in the AS LA Template input field. The used length of the result field will be returned. A not found condition will result in a zero value (4-byte integer) in the result field. All scans are from front-to-back. The source field is limited to 32,767 bytes. Additional calls for the same target string pattern will allow you to find all occurrences of the string pattern within the source field. A not found condition will result in a zero value (4-byte integer) in the result field. This requires a minimal 32-byte PSA to be provided via the AS LA Request Options data structure. An optional least frequently used (LFU) frequency vector can be specified as the first input template field. Otherwise, the default LFU frequency vector will be used. The length must be 256 bytes. Note: http://en.wikipedia.org/wiki/Letter_frequency Request option(s), *start* and *PSA* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ LFU frequency vector (256 bytes) • AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template result field • (optional) LE Feedback Area

	Conditions: • ASU039 – 105 Argument is invalid • ASU03C – 108 Contains invalid data • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range
STRIP (17) ASUDSTRP	Return a result field containing the source field less the leading and/or trailing default *strip* characters specified via the AS LA Request Options data structure. The used length of the result field will be returned. Request option(s), source field *upper case translation* is supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
STRISRT (18) ASUDSINS	Return a result field containing the source field plus the string pattern specified in the Template input field, inserted at the selected position. Specify the *start* position via the AS LA Request Options data structure. The used length of the result field will be returned. Request option(s), source field *upper case translation* and *start* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template input field • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid

	• ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
STRLPOS (19) ASUDLPOS	Return a result field containing the position (4-byte integer) in the source string of the string pattern specified in the Template input field. The used length of the result field will be returned. A not found condition will result in a zero value (4-byte integer) in the result field. All scans are from back-to-front. The source field is limited to 32,767 bytes. Additional calls for the same target string pattern will allow you to find all occurrences of the string pattern within the source field. A not found condition will result in a zero value (4-byte integer) in the result field. This requires a minimal 32-byte PSA to be provided via the AS LA Request Options data structure. An optional least frequently used (LFU) frequency vector can be specified as the first input template field. Otherwise, the default LFU frequency vector will be used. The length must be 256 bytes. Note: http://en.wikipedia.org/wiki/Letter_frequency Request option(s), *start* and *PSA* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ LFU frequency vector (256 bytes) • AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03C – 108 Contains invalid data • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range
STRBPOS (20) ASUDBPOS	Using the Boyer-Moore string matching algorithm, return a result field containing the position (4-byte integer) of the string pattern specified in the AS LA Template input field. The search argument can not exceed 255 bytes. The

	used length of the result field will be returned. A not found condition will result in a zero value (4-byte integer) in the result field. All scans are from front-to-back. The source field is limited to 32,767 bytes. Additional calls for the same target string pattern will allow you to find all occurrences of the string pattern within the source field. A not found condition will result in a zero value (4-byte integer) in the result field. You can optionally match an upper-case search argument to a mixed-case source string (case insensitive match), by specifying *case insensitive* in the AS LA Request Options data structure. *Case insensitive* match is not available when the search argument is one byte long. The AS LA Request Options data structure must contain a 288-byte PSA. Request option(s), source field *case insensitive*, *start* and *PSA* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03C – 108 Contains invalid data • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range
STRREPL (21) ASUDSREP	Returns a result field containing the source field with the selected string replaced. The string pattern to be replaced in the source field can be selected in one of two ways. • Specify the *start* position and number/*count* of characters to be replaced via the AS LA Request Options data structure. If the *count* (number of characters) to be replaced is zero, the rest of the source field will be replaced / truncated. • Specify a Template input field containing the string pattern to locate in the source field. If not found, no data is moved to the result field. The length of the string pattern will be used as the *count* or number of characters to replace. The used length of the result field will be returned. Request option(s), source field upper *case translation* or *case-insensitive* and

	start, *count* are supported. Other actions referenced • STRBPOS Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template input field ○ String pattern template (replacement string) • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03E – 110 Not found • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
SUBSTR (22) ASUDSSUB	Return a result field containing the part of the source field specified by the *start* position and number/*count* of characters to be selected via the AS LA Request Options data structure. If the *count* (number of characters) to be selected is zero the rest of the source field will be selected. The used length of the result field will be returned. Request option(s), source field *upper case translation*, *start* and *count* are supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow

VAR / SCAN (23) ASUDSVAR	Return one or more result fields containing part of the source field specified by one or more of the three major parsing methods (word, string pattern, or numeric pattern). The template list may contain only result fields if parsing via words, a combination of pattern and result fields if parsing via string pattern and/or numeric pattern. The template list is limited to 16 occurrences made up of any of the previously defined methods. If the last template occurrence is a dummy result field, you may specify a 32-byte *PSA* to hold the necessary data for a returning call to re-position your cursor in the source field to continue parsing. Request options(s), source field *upper case translation* or *case-insensitive* and *start* are supported. A 32-byte *PSA* is optional. Other actions referenced: • STRBPOS Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template(s) consisting of at most 16 alternating pattern specifications and result fields or just result fields • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03D – 109 Branch index is invalid • ASU03E – 110 Not found • ASU03G – 112 Not numeric • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
VHEX (24) ASUDVHEX	Return three result fields. The first containing all or part of the source field. The second containing a printable zone nibble of each character in the first result field. The third containing a printable number nibble of each character in the first result field. The result fields can be up to 256 bytes in length. The source field can be up to 32,767 bytes in length. Specify *start*, to position yourself for subsequent calls, via the AS LA Request Options data structure. An optional alphabet translation vector can be specified as the first input template field. This will be used to translate the first result field. Otherwise,

	the default EBCDIC printable character vector will be used to translate all unprintable characters to blank. The length must be 256 bytes Request option(s), *start* is supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Translation vector (256 bytes) • AS LA Template result field (character) • AS LA Template result field (byte zone) • AS LA Template result field (byte number) • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
WORDDEL (25) ASUDWDEL	Return a result field containing the source field less the word(s) selected to be deleted. Specify the *start* word and number/*count* of words to delete, via the AS LA Request Options data structure. If the count (number of words) to delete is zero the rest of the source field will be deleted. The result field will have leading, trailing and superfluous delimiter characters removed. One default *pad* character will exist between each word. The used length of the result field will be returned. Request option(s), source field *upper case translation*, *start* and *count* are supported. Other actions referenced: • COUNT Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template result field • (optional) LE Feedback Area

	Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
(26)	
WORDPOS (27) ASUDWPOS	Returns a result field containing the position (4-byte integer) of the selected word(s) in the source field. A second result field, if present, contains the word(s) selected from the source field, with leading, trailing and superfluous delimiter characters removed. One default *pad* character will exist between each word. Selections can be made by specifying the *start* word and number/*count* of word(s) via the AS LA Request Options data structure. If the *count* of word(s) is zero, all following words will be copied to the second result field if present. The used length of the result fields will be returned. Request option(s), source field *upper case translation*, *start* and *count* are supported. Other actions referenced: • COUNT Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Delimiter vector (16 bytes max) • AS LA Template result field • (optional) AS LA Template result field • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
HHEX (28) ASUDHHEX	Return two result fields. The first containing all or part of the source field. The second containing a printable hexadecimal character of each character in the first result field. The second result field must be twice the length of the first result field. Therefore, the first result field can not exceed 16,383 bytes in

	length. The first result field may be shorter than the source field. The source field can be up to 32,767 bytes. Specify *start*, to position yourself for subsequent calls, via the AS LA Request Options data structure. An optional alphabet translation vector can be specified as the first input template field. This will be used to translate the first result field. Otherwise, the default EBCDIC printable character vector will be used to translate all unprintable characters to blank. The length must be 256 bytes Request option(s), *start* is supported. Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • (optional) AS LA Template input field ○ Translation vector (256 bytes) • AS LA Template result field (character) • AS LA Template result field (hexadecimal) • (optional) LE Feedback Area Conditions: • ASU039 – 105 Argument is invalid • ASU03H – 113 Too many arguments • ASU03I – 114 Too few arguments • ASU03J – 115 Numeric data is out of range • ASU03L – 117 Overflow
****************	
ASUDNOFB	Return a result field (4-byte integer) containing the number of times the search argument is found in the source field. If not found then the value is zero. If two result fields are specified, a vector containing the position (2-byte integer) of each occurrence of the search argument in the source field. The internal vector used to keep track of each find in the source field is a fixed length of 512 bytes. Which is enough space to track 256 finds. The internal vector will be copied to the optional supplied result template, if present. Therefore, make sure that the second result field is a minimum of 512 bytes in length. Request option(s), source field *case insensitive*, and *start* are supported. Other actions referenced: • STRBPOS

	Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template input field ○ String pattern template to locate (search argument) • AS LA Template result field • (optional) AS LA Template result field Conditions: • LE/COBOL RETURN-CODE = +4 when template count < 2 • LE/COBOL RETURN-CODE = +8 when primary result field has not been specified correctly • LE/COBOL RETURN-CODE = +8 when the optional result field has not been specified correctly • LE/COBOL RETURN-CODE = +8 when the length used of the optional result field requires a maximum length greater than provided by the caller
ASUDN2TB	Returns a result field containing the source field (some number) converted to text. The source field can be provided in zone decimal, pack decimal, or edited numeric format. Use *count* via the AS LA Request Options data structure to specify the number of digits right of decimal for a source field provided in zone decimal or pack decimal format. The source field supports 18 digits, of which a maximum of 9 digits may be right of decimal. Output format: • word – text [AND] text • currency – text [DOLLARS] [AND] text [CENTS] no more than 2 decimal digits may be specified • check – text [AND] nn[/100] [DOLLARS] value can not be negative The used length of the result field will be returned. Request option(s), *output text format* and *count* are supported. Other actions referenced: • APPEND • DEEDIT Parameter list: • AS Communication Area • AS LA Request Options • AS LA Source field • AS LA Template count • AS LA Template result field

	• (optional) LE Feedback Area Conditions: • result field (4-byte integer) value of zero is returned if any errors exist • LE/COBOL RETURN-CODE = +4 when template count < 1 • LE/COBOL RETURN-CODE = +4 when output format is currency and the number of digits right of decimal is greater than 2 • LE/COBOL RETURN-CODE = +4 when output format is check and the source number is negative • LE/COBOL RETURN-CODE = +8 when the length used of the result field exceeds the maximum length provided by the caller • LE/COBOL RETURN-CODE = +16 when the source format can not be determined
ASUMDHEX	Print (CEEMOUT) data in hexadecimal dump format. Other actions referenced: • HHEX Parameter list: • AS Communication Area • AS5DMPRB Request block • Input area
ASUMVHEX	Print (CEEMOUT) data in vertical hexadecimal format. Other actions referenced: • VHEX Parameter list: • AS Communication Area • AS5DMPRB Request block • Input area
	Things like Soundex and Metaphone 3 http://en.wikipedia.org/wiki/Moby_Project http://en.wikipedia.org/wiki/Phonetic_algorithm **Soundex** is a phonetic algorithm for indexing names by sound, as pronounced in English. The goal is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling. Soundex was created by Robert C. Russell of Pittsburgh, Pennsylvania. He received U.S. patent 1,261,167 on April 2, 1918 on for it.

	http://en.wikipedia.org/wiki/Soundex http://en.wikipedia.org/wiki/New_York_State_Identification_and_Intelligence_System http://en.wikipedia.org/wiki/Daitch%E2%80%93Mokotoff_Soundex http://stevemorse.org/phonetics/bmpm.htm http://en.wikipedia.org/wiki/International_Phonetic_Alphabet https://www.internationalphoneticassociation.org/redirected_home http://www.searchforancestors.com/utility/soundex.html