Determining Word Selection: An Example

This section provides an example of how a script system determines word selection.

Note: The definition of a word in the Roman Script System may vary slightly with localization. The Table below provides the class numbers, character classes, and explanations of the class names for the U.S. word selection algorithm.

U.S. word selection algorithm

Class number	**Class name**	**Used for**
0	break	Everything not included below
1	nonBreak	Nonbreaking spaces
2	letter	Letters, ligatures, and accents
3	number	Digits
4	midLetter	Hyphen
5	midLetNum	Apostrophe (vertical or right single quote)
6	preNum	$ £ ¥ €
7	postNum	% ‰ ¢
8	midNum	, ⁄
9	preMidNum	.
10	blank	Space, tab, null
11	cr	Return

The **NFindWord** table in the U.S. 'itl2' resource defines words as any of the following configurations of the classes listed in the Table above:

- A sequence of letters, possibly separated by a hyphen, apostrophe, or period and possibly followed by a sequence of numbers (defined next). Some examples are *ultra-cool*, *Bob's*, and *record.field.*

- A sequence of numbers, possibly separated by a comma, fraction sign, apostrophe, or period; possibly preceded by a decimal point or currency sign; and possibly followed by a percentage sign or by a sequence of letters (defined previously). Some examples are *1.234*, *$23.14*, *.70*, and *12ea-b.*

- A sequence of spaces, tabs, or nulls, possibly followed by a *cr*.

- Characters that are words by themselves and not included as part of the above definitions.

Note: With system software version 7.0, the treatment of a sequence of one or more nonbreaking spaces has changed. If there is a non-whitespace character (that is, a character that is neither *blank* nor *cr*) on either side of the sequence, the sequence becomes part of the word that includes the non-whitespace character. Thus, if the sequence is between non-whitespace characters, it joins the words of which they are a part. Otherwise, the nonbreaking spaces are treated as blanks.

The Table below shows where word breaks occur in various character sequences. In the table, *other* denotes any character that is not *blank, cr,* or *nonBreak.*

Occurrence of word breaks in various character sequences

Character	Word break	Character	Word break	Character
blank	no	*nonBreak*	no	*blank*
blank	no	*nonBreak*	no	*cr*
blank	yes	*nonBreak*	no	*other*
cr	yes	*nonBreak*	no	*blank*
cr	yes	*nonBreak*	no	*cr*
cr	yes	*nonBreak*	no	*other*
other	no	*nonBreak*	yes	*blank*
other	no	*nonBreak*	yes	*cr*
other	no	*nonBreak*	no	*other*

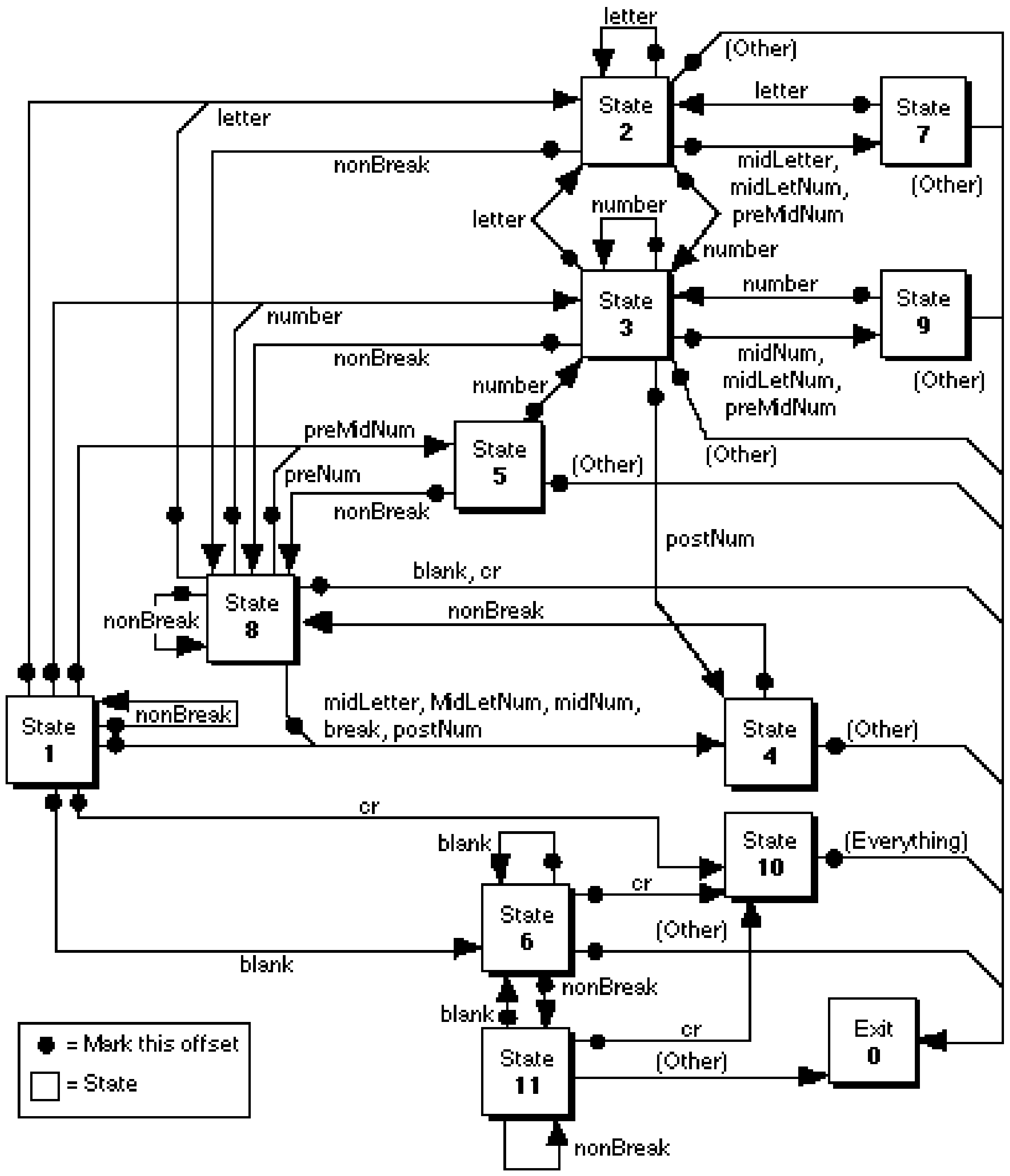
The Table below describes the meaning of each state number shown in the figure below.

Significance of the state numbers in the Roman word selection algorithm

State number	**Meaning**
1	Start, or has detected initial *nonBreak* sequence
2	Has detected a *letter*
3	Has detected a *number*
4	Has detected a non-whitespace character that should stand alone; now anything but *nonBreak* generates an exit
5	Has detected *preMidNum* or *preNum;* now anything but *number* or *nonBreak* generates an exit
6	Has detected a *blank*
7	Has detected a *letter* followed by *midLetter, midLetNum,*or *preMidNum;* now anything but *letter* generates an exit
8	Has detected a non-whitespace character followed by *nonBreak* (the *nonBreak* should be treated as non-whitespace)
9	Has detected a *number* followed by *midNum, midLetNum,* or *preMidNum;* now anything but *number* generates an exit

10	Marks current offset (include one more character), then exits
11	Has detected *blank* followed by *nonBreak* (the *nonBreak* should be treated as *blank*)

The Figure below illustrates the process of determining whether a sequence of characters is to be selected as a word. It shows the possible paths through the states of the Roman word selection algorithm defined in the Table below.



Roman word selection state transitions

The Table below shows the U.S. word selection transition table for forward processing. Each column shows the action codes for a current state.

> **Note:** In this table, the first column must be all nonzero values with '*'.

U.S. word select transition table for forward processing

Class No.	Class name	State number 1	2	3	4	5	6	7	8	9	10	11
0	break	*4	*0	*0	*0	*0	*0	0	*4	0	*0	0
1	nonBreak	*8	*8	*8	*8	*8	*0	0	*8	0	*0	11
2	letter	*2	*2	*2	*0	*0	*0	*2	*2	0	*0	0
3	number	*3	*3	*3	*0	*3	*0	0	*3	*3	*0	0
4	midLetter	*4	*7	*0	*0	*0	*0	0	*4	0	*0	0
5	midLetNum	*4	*7	*9	*0	*0	*0	0	*4	0	*0	0
6	preNum	*5	*0	*0	*0	*0	*0	0	*5	0	*0	0
7	postNum	*4	*0	*4	*0	*0	*0	0	*4	0	*0	0
8	midNum	*4	*0	*9	*0	*0	*0	0	*4	0	*0	0
9	preMidNum	*5	*7	*9	*0	*0	*0	0	*5	0	*0	0
10	blank	*6	*0	*0	*0	*0	*6	0	*0	0	*0	*6
11	cr	*10	*0	*0	*0	*0	*10	0	*0	0	*0	*10

* Means "mark the offset before this character"

Optimized Word Break Tables

The code example below shows how to obtain a copy of the word selection table from the default 'itl2' resource. You may want to do this so that you can pass the word selection table directly to **NFindWord** or **FindWord** in a loop. This example assumes that inside the loop you are trying to find word boundaries. Outside the loop, you set up for this task by getting a copy of the table needed for the word break routine. See the section entitled **Accessing the International Resources** for more on the **IUGetItlTable** procedure and as well as the section entitled **Localizing Word Selection and Line Break Tables** for details on the **NFindWord** procedure.

Obtaining optimized word break tables

```
// Assuming inclusion of <MacHeaders>

#include <Packages.h>

Ptr GetWordSelect (void);

Ptr GetWordSelect ()
{
    Handle  itlHandle;
    long    tableOffset;
    long    tableLength;
    Ptr     wordBreakPtr;
    Ptr     tempPtr;

    wordBreakPtr = nil;                     // assume failure

    IUGetItlTable(iuSystemScript, iuWordSelectTable, &itlHandle,
            &tableOffset,  &tableLength);
    // If script is incorrect or the table is not available, the handle
    //  will return as nil.

    if (itlHandle) {
        wordBreakPtr = NewPtr(tableLength);
        if (wordBreakPtr) {
            tempPtr = (*itlHandle) + tableOffset;
            BlockMove(tempPtr, wordBreakPtr, tableLength);
        }
    }
    return wordBreakPtr;
}
```