```
// TYPESET.WCM — Replace ASCII quotation, apostrophe, dash and hyphen marks with typesetter-style
//                                        marks, using characters from WP charsets 1 and 4, and arrange for correct hypenation
//                                        at hyphens and dashes.
//
//                              "foo"       becomes     “foo”
//                              'foo'       becomes     ‘foo’
//                              dad's       becomes     dad’s
//                              Niels'      becomes     Niels’
//                              foo--bar    becomes     foo—bar
//                              foo-bar     becomes     foo–bar
//                              75-78       remains     75-78
//
// WARNINGS:
//      1. This screws up on nested quotation marks of the same kind (e.g. doublequotes within
//          doublequotes).  This could be fixed, but the speed penalty makes it probably not
//          worthwhile.  Doublequotes within single quotes, and vice versa, are handled
//          correctly.
//      2. Apostrophes that are not followed by an "s" (e.g. in "the three cats' whiskers") that occur after
//          an opening singlequote can't be distinguished from the closing singlequote—humans
//          do this semantically, not syntactically.  It is best to type these as proper apostrophes
//          when you input the document; this will keep the macro from getting confused.
//      3. If you use ASCII squotes as mathematical prime symbols, this macro will screw up.
//          Type your document with real prime ′ ([6,45]) and doubleprime ″ ([6,46]) symbols
//           to avoid trouble.
//
// AUTHOR
//      Richard Reiner <rreiner@nexus.yorku.ca>
//
// REVISION HISTORY
//      12 Jun 1992     RjR     Initial version
//      16 Jun 1992     RjR     Logic to distinguish between singlequotes and apostrophes;
//                                                  loop unrolling for speed.
//                                                  Released as v 1.0.
//      25 Jun 1992     RjR     Added conversion of dashes; changed name to
```

```
        TYPESET.WCM.
        //                                              Released as v 1.1.
        //      16 Oct 1992       RjR    Added extra passes to insert [HyphSRt] codes
        after Emdashes and
        //                                              Endashes.
        //      23 Nov 1992       RjR    Improved accuracy and speed of
        singlequote/apostrophe heuristic;
        //                                              improved usage notes.
        //      25 Nov 1992       RjR    Added front-end dialog to make conversions
        selective; added conversion
        //                                              of hyphens.
        //                                              Released as v. 1.3.
        //

        APPLICATION(WP;WPWP;Default)

        //
        **********************************************************************
        **
        // Main
        //
        CALL(Init@)
        CALL(WhichConversions@)

        IF (DoSquote=1)
CALL(ProcessSq@)
        ENDIF
        IF (DoDquote=1)
CALL(ProcessDq@)
        ENDIF
        IF (DoDash=1)
CALL(ProcessDash@)
        ENDIF
        IF (DoHyphen=1)
CALL(ProcessHyphen@)
        ENDIF
        IF (DoHyphSrt=1)
CALL(ProcessHyphSRt@)
        ENDIF
        CALL(End@)
        // NOTREACHED
        QUIT
```

```
        //
        ********************************************************************************
        **
        // Init@
        //
        // Initialization
        //
        LABEL(Init@)
ONERROR(End@)
ONCANCEL(End@)
GetWPData(MacroVariable:RevCodes;SystemVariable:RevealCodesActive!)
SelectMode(Off!)
RevealCodes(Off!)
MacroStatusPrompt(Off!)
Display(Off!)
        RETURN
```

```
        //
        ********************************************************************************
        **
        // WhichConversions@
        //
        // Front end dialog
        //
        LABEL(WhichConversions@)
ConvDlg:=1
DoSquote:=1
DoDquote:=1
DoDash:=1
DoHyphen:=1
DoHyphSrt:=1
DialogDefine(ConvDlg; 50; 50; 160; 136; 1+2+16; "TypeSet.WCM: Select
Conversions")
DialogAddCheckBox(ConvDlg; 1000; 12; 8; 90; 12; "Convert singlequotes"; DoSquote)
DialogAddCheckBox(ConvDlg; 1001; 12; 24; 90; 12; "Convert doublequotes";
DoDquote)
DialogAddCheckBox(ConvDlg; 1002; 12; 40; 90; 12; "Convert ASCII dashes"; DoDash)
DialogAddCheckBox(ConvDlg; 1003; 12; 56; 90; 12; "Convert ASCII hyphens";
DoHyphen)
DialogAddCheckBox(ConvDlg; 1004; 12; 72; 130; 12; "Insert [HyphSrt] codes after WP
dashes"; DoHyphSrt)
DialogDisplay(ConvDlg; 1)

IF (MacroDialogResult = 2)
    CALL(End@)
ENDIF

DialogDestroy(ConvDlg)
        RETURN
```

```
            //
            ********************************************************************************
            **
            // ProcessSq@
            //
            // Convert singlequotes and apostrophes.  Note that this *must* be called
            *before* ProcessDq, because
            //       it relies on the fact that any dquote characters are in ASCII.
            //
            LABEL(ProcessSq@)
PosDocTop()
ONNOTFOUND(SqDone@)
// Unwind the following loop by one, so we can use SearchNext() in the body.  This
            cuts almost 50%
//          off the function's execution time.
SearchText(SearchString:"'";SearchDirection:Forward!;SearchScope:Extended!)
PosCharPrevious()
GetWPData(MacroVariable:LeftCh;SystemVariable:LeftChar!)
PosCharNext()
// We are not in a quote when we start; so the found char is an squote iff the char to
            it's
//          left is a code, one of certain punctuation marks, or a space.
STRPOS(TargPos; LeftCh; "([{-–—"" ")
IF ((LeftCh="") or (TargPos > 0))
    // It's a singlequote
    DeleteCharPrevious()
    TypeChar(CharacterSet:4;CharacterOffset:27)
    InQuote:=True
ELSE
    // It's an apostrophe
    DeleteCharPrevious()
    TypeChar(CharacterSet:1;CharacterOffset:9)
    InQuote:=False
ENDIF
// The only way out of this loop is the ONNOTFOUND trap
WHILE(True)
    SearchNext()
    PosCharPrevious()
    GetWPData(MacroVariable:LeftCh;SystemVariable:LeftChar!)
    PosCharNext()
    GetWPData(MacroVariable:RightCh;SystemVariable:RightChar!)
    IF (InQuote)
        // If we are in a quote, the found char is an squote iff the char to it's right is a
            code,
        //  one of certain punctuation marks, or a space.
```

```
        STRPOS(TargPos; RightCh; ".?!,;:)]}-–—"" ")
        IF ((RightCh="") or (TargPos > 0))
            // It's a singlequote
            DeleteCharPrevious()
            TypeChar(CharacterSet:4;CharacterOffset:28)
            InQuote:=False
        ELSE
            // It's an apostrophe
            DeleteCharPrevious()
            TypeChar(CharacterSet:1;CharacterOffset:9)
        ENDIF
    ELSE // not InQuote
        // If we are not in a quote, the found char is an squote iff the char to it's left is a
            code,
        //   one of certain punctuation marks, or a space.
        STRPOS(TargPos; LeftCh; "([{-–—"" ")
        IF ((LeftCh="") or (TargPos > 0))
            // It's a singlequote
            DeleteCharPrevious()
            TypeChar(CharacterSet:4;CharacterOffset:27)
            InQuote:=True
        ELSE
            // It's an apostrophe
            DeleteCharPrevious()
            TypeChar(CharacterSet:1;CharacterOffset:9)
        ENDIF
    ENDIF
ENDWHILE
LABEL(SqDone@)
            RETURN
```

```
            //
            ********************************************************************************
            **
            // ProcessDq@
            //
            // Convert all doublequotes.  This is much easier than the singlequotes since
            the ASCII character
            //       is not overloaded.
            //
            LABEL(ProcessDq@)
PosDocTop()
ONNOTFOUND(DqDone@)
// Unwind the following loop by one, so we can use SearchNext() in the body.  This
            cuts almost 50%
//          off the function's execution time.
SearchText(SearchString:"""";SearchDirection:Forward!;SearchScope:Extended!)
DeleteCharPrevious()
TypeChar(CharacterSet:4;CharacterOffset:30)
InQuote:=True
// The only way out of this loop is the ONNOTFOUND trap
WHILE(True)
    SearchNext()
    DeleteCharPrevious()
    IF (InQuote)
        TypeChar(CharacterSet:4;CharacterOffset:31)
        InQuote:=False
    ELSE
        TypeChar(CharacterSet:4;CharacterOffset:30)
        InQuote:=True
    ENDIF
ENDWHILE
LABEL(DqDone@)
            RETURN
```

```
            //
            ********************************************************************************
            **
            // ProcessDash@
            //
            // Convert ASCII "--" dashes to Emdashes.  This one's easy.
            //
            LABEL(ProcessDash@)
PosDocTop()
SearchReplace(SearchString:"";
        SearchDirection:Forward!;
        ReplacementScope:Extended!;
        ReplacementString:"—";
        ReplacementAction:ReplaceAll!)
            RETURN
```

```
            //
            ********************************************************************************
            **
            // ProcessHyphen@
            //
            // Convert hyphens.  If the characters left or right of the thing are digits,
            leave it as ASCII "-"; otherwise
            //       make it an Endash.
            //
            LABEL(ProcessHyphen@)
PosDocTop()
ONNOTFOUND(HyphenDone@)
// Unwind the following loop by one, so we can use SearchNext() in the body.  This
            cuts almost 50%
//          off the function's execution time.
SearchText(SearchString:""; SearchDirection:Forward!; SearchScope:Extended!)
PosCharPrevious()
GetWPData(MacroVariable:LeftCh;SystemVariable:LeftChar!)
PosCharNext()
GetWPData(MacroVariable:RightCh;SystemVariable:RightChar!)
STRPOS(LTargPos; LeftCh; "1234567890")
STRPOS(RTargPos; RightCh; "1234567890")
// If there's a code on either side, or neither of the surrounding chars is a digit
IF ((LeftCh="") or (RightCh="") or ((LTargPos = 0) and (RTargPos = 0)))
    // It's an EnDash
    DeleteCharPrevious()
    TypeChar(CharacterSet:4; CharacterOffset:33)
ELSE
    // It's a minus, leave it alone
ENDIF
// The only way out of this loop is the ONNOTFOUND trap
WHILE(True)
    SearchNext()
    PosCharPrevious()
    GetWPData(MacroVariable:LeftCh;SystemVariable:LeftChar!)
    PosCharNext()
    GetWPData(MacroVariable:RightCh;SystemVariable:RightChar!)
    STRPOS(LTargPos; LeftCh; "1234567890")
    STRPOS(RTargPos; RightCh; "1234567890")
    // If there's a code on either side, or neither of the surrounding chars is a digit
    IF ((LeftCh="") or (RightCh="") or ((LTargPos = 0) and (RTargPos = 0)))
        // It's an EnDash
        DeleteCharPrevious()
        TypeChar(CharacterSet:4; CharacterOffset:33)
    ELSE
```

```
        // It's a minus, leave it alone
    ENDIF
ENDWHILE
LABEL(HyphenDone@)
            RETURN
```

```
            //
            ********************************************************************************
            **
            // ProcessHyphSRt@
            //
            // Insert [HyphSRt] codes after Emdashes and Endashes.  This one's easy too.
            //
            LABEL(ProcessHyphSRt@)
// Do Emdashes
PosDocTop()
SearchReplace(SearchString:"—";
        SearchDirection:Forward!;
        ReplacementScope:Extended!;
        ReplacementString:"—";
        ReplacementAction:ReplaceAll!)
// Do Endashes
PosDocTop()
SearchReplace(SearchString:"–";
        SearchDirection:Forward!;
        ReplacementScope:Extended!;
        ReplacementString:"–";
        ReplacementAction:ReplaceAll!)
            RETURN
```

```
        //
        ********************************************************************************
        **
        // End@
        //
        // Wrap things up and terminate
        //
        LABEL(End@)
IF(RevCodes=True)
    RevealCodes(On!)
ENDIF
PosDocTop()
QUIT
```