Paper 87-2019

# People Keep ASCIIng Me About These Characters

Dan Konkler, Rebecca Bowermaster, and Stephanie Ann Sanchez, Roche Molecular Solutions

## ABSTRACT

Every programmer has run into an issue at least once that was caused by the presence of a control character or extended ASCII character in their dataset. You get extra records in your dataset because there's a line feed character within a CSV field. Your Excel output has "unreadable content" because your data contain "<, >, or &". Something went wrong in your RTF file when your trusty escape character wound up in your dataset. Your fancy validation program that reads in the production output for programmatic comparison fails because it can't process the extended ASCII or control characters. These issues can cause hours of time lost to searching for the problematic character. In this paper, we present two macros that can quickly identify potentially problematic characters in your dataset library and point you to the specific records and fields that need your attention.

## INTRODUCTION

There are many situations in which the characters and encodings associated with a dataset require careful attention. When data is crossing systems or software, character encodings in particular can lead to mis-translation of text, in particular when the characters used go beyond standard ASCII. In addition, depending on the program being used and interfacing with other software programs, standard characters can cause issues if they act as a control character in either the source or destination. Below is a list of four such situations the authors have encountered, though more exist which the reader has likely encountered:

1. Unexpected results or missing characters may occur in your report if your specified ODS ESCAPECHAR appears in the body of your ODS output.
2. An XML special character (, &,',") in ODS EXCEL destination will cause unreadable content errors if option PROTECTSPECIALCHARS=OFF (Sanchez, 2018).
3. Control characters (e.g. line feed) within source data can interfere with data processing (e.g. signal process to prematurely end or split the record).
4. Extended ASCII characters can translate incorrectly on output and/or read in from XML and HTML files and well as between systems using different encodings or character sets.

For context, consider the ASCII character map (Appendix 1: ASCII Table), which provides the standard 128 codes plus an additional 128 extended ASCII characters as generated in SAS using the default WLATIN1 session encoding. The standard 128 ASCII characters include 33 control characters (including 5 space characters) and 95 printable characters (including space) ("ASCII", Wikipedia: The Free Encyclopedia). While the first 128 characters are standard across encoding, they can cause issues with data processing (see examples 1-3 above). The UTF-8 encoding extends the character set by use of up to 4 bytes to encode the additional characters. This causes risk of truncation during transcoding, resulting in translation errors or lost information. Additionally, data represented in languages other than English or using characters outside of the standard ASCII character set may not be acceptable for submission to regulatory bodies such as FDA (Stackhouse, 2016).

Base SAS® software includes several useful character functions which can be used to identify and handle target characters at the variable level. The "ANY" and "NOT" family of character functions allow the user to search for the first instance of a character either in or not in a specified class of characters. Defined character classes include ALNUM (alphanumeric characters), ALPHA (alphabetic characters), CNTRL (control characters), DIGIT (digits), FIRST (characters valid as the first character in a SAS variable name under VALIDVRNAME=V7), GRAPH (graphical characters), LOWER (lowercase letters), NAME (characters valid in a SAS variable name under VALIDVARNAME=V7), PRINT (printable characters), PUNCT (punctuation characters), SPACE (Whitespace characters), UPPER (uppercase letters), and

XDIGIT (hexadecimal characters that represent a digit). The character classes each ASCII character belongs to is included in the attached character map.

Additional character functions allow you to keep or remove specified characters or character classes (COMPRESS), count instances of specified characters (COUNTC), find if specified characters are present (FINDC), return the first position of any of specified characters (INDEXC), or replace specific characters in a character expression (TRANSLATE). By combining these functions, it is possible to identify and, if desired, keep or delete any specified character in a dataset, though doing so can be cumbersome, especially when processing multiple data fields across a library of datasets.

## SUMMARIZE CHARACTER CONTENT– CHARCOUNT MACRO

Macro CHARCOUNT (Appendix 2: CHARCOUNT Macro) was developed to count the number of occurrences of each character in your data. The macro allows you to search an individual dataset or entire library and print a summary of how many times each character appear in the specified dataset(s). This macro is useful for e.g. determining if your favorite ESCAPECHAR is safe to use or if you might have any dataset portability issues when sending data across systems. The macro has only one required parameter LIB=*libname*.

The default functionality of this macro to search all datasets within the library and report the number of instances of each character as well as the number of instances of control, white space, printable, extended ASCII, and standard ASCII character sets. Optional parameter MEM=*dataset* allows you to specify a specific dataset to summarize while parameter MAXMEM=*num* (default=999) allows you to restrict the number of datasets included in the summary. Following identification of the presence of the character(s) of interest, setting the parameter RECORD=yes will provide a summary at the record level, allowing you to identify individual records containing the character of interest. To identify the specific records and fields, use the FINDCHAR macro described in the next section.

## IDENTIFY AND HANDLE CHARACTERS OF INTEREST – FINDCHAR MACRO

Macro FINDCHAR (Appendix 3: FINDCHAR Macro) can be used to identify and keep, delete, or replace all specified characters from a given dataset. This macro is useful in both identifying and handling data fields that contain problematic characters. The macro has two required parameters; DS=input-dataset and at least one of the following: KEEPCHARS= *character-class or -list*, DELCHARS=*character-class or -list*, REPCHARS=*character-list* or FUNCTION=*string-function*. These parameters allow you to do one of the following actions:

- Compress out characters not in the list or class (KEEPCHARS)
- Compress out characters in the list or class (DELCHARS)
- Apply the specified string function to all character variables (FUNCTION)
- Replace characters in the list or class specified using DELCHARS (REPCHARS)

The –CHARS parameters allow the user to specify a standard character set (ALPHA, CTRL, DIGIT, …), “Extended” characters (characters beyond ASCII 127), “Hex” characters (printable standard ASCII characters), or a user-specified quoted string containing ASCII or hex characters of interest (e.g, “~!^*”).

At least one and only one of KEEPCHARS, DELCHARS, or FUNCTION may be specified. If DELCHARS is specified, adding REPCHARS=*list* replaces the deleted characters with the specified replacement character(s) using the TRANSLATE function after sorting and DELCHARS string and removing any duplicate characters. The REPCHARS value must be either a single character or the string length must match the DELCHARS string length. DELCHARS list should have values presented in alphanumeric order; REPCHARS should be listed in corresponding replacement order. When a single REPCHAR character is specified, that single character replaces all characters in the DELCHARS list.

This macro produces an output dataset FIND_FIELD which contains a record for each field containing one or more characters in character list and variable NEW_DATA which contained the value of the field

with all characters in character list either kept, deleted, or replaced. An additional optional parameter OUT=*output-dataset* specifies that an output dataset be generated in which the original data are replaced

## REAL LIFE EXAMPLES

This three-part example will demonstrate use of both of the above macros. First, we use CHARCOUNT to locate ASCII characters of the Extended class among the library of SAS Help data sets. Then, we follow-up with the FINDCHAR macro to identify each instance of Extended class characters within one of these datasets. After identification, we demonstrate how to utilize FINDCHAR to replace the present Extended characters with standard ASCII characters. The examples below use the global variable “current” to hold the path of the directory where the macros are stored.

### EXAMPLE 1 – SUMMARIZE CHARACTERS USED IN SAS HELP LIBRARY DATASETS

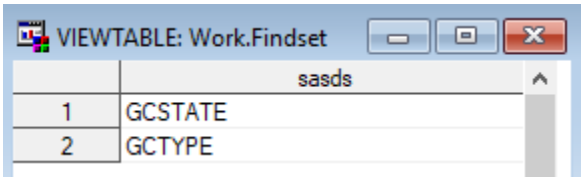
In the first example (see Input 1) we load the CHARCOUNT macro then use it to characterize the contents of the SAS Help library. CHARCOUNT creates the number of instances of each character in a dataset, using the decimal value of the Extended class characters as variable names (i.e. _128). We use a data step to find the subset of SAS Help datasets with any Extended class characters, using the standard output dataset CHARCOUNT_H.

```
*Macro call.;
%INCLUDE "&current.\macro_charcount.sas";
%charcount(lib=sashelp);

*Format output to find datasets with extended ASCII characters.;
DATA findset;
    SET charcount_h;
    test=SUM(OF _128-_255); *Total counts of Extended characters.;
    IF test^=.; *Keep observations from datasets with Extended characters;
    KEEP sasds; *Keep dataset name(s).;
RUN;
```

**Input 1: Using CHARCOUNT to Find Extended ASCII Characters in the SAS Help Datasets.**

The resulting data set includes two observations: GCSTATE and GCTYPE. See Display 1.




**Display 1: SAS Help Datasets Containing Extended ASCII Characters**

### EXAMPLE 2 – FIND FIELDS CONTAINING EXTENDED ASCII CHARACTERS

In the second example (see Input 2) we load the FINDCHAR macro then utilize it to find fields containing the Extended class of ASCII characters in the SAS Help dataset GCSTATE.

```
*Macro inclusion.;
%INCLUDE "&current.\macro_findchar.sas";

*Open our target data set, SAS Help dataset GCSTATE.;
DATA example;
    SET sashelp.gcstate;
RUN;

*Macro call to remove Extended characters in a new output datasets NOEX.;
%findchar(ds=example, out=noex, delchars=Extended);
```

**Input 2: Finding Extended Characters in SAS Help Dataset GCSTATE**

The call in Input 2 creates two datasets, FIND_FIELD (default) and NOEX (specified in macro call). FIND_FIELD details which observations (RECORDNUMBER) of which fields (FIELD) had Extended characters in them, the original data value (ORIG_DATA), and the new data value after deleting Extended characters (NEW_DATA) (see Display 2). The user-specified output dataset NOEX contains an updated version of the input dataset, with Extended characters deleted (see Display 3).

VIEWTABLE: Work.Find_field

| | recordnumber | field | new_data | remove_chars_hex | orig_data |
|---|---|---|---|---|---|
| 1 | 39 | StateAlias | N.S. \| N.-. | C9 | N.S. \| N.-É. |
| 2 | 49 | StateAlias | P.E.I. \| .P.. | CEC9 | P.E.I. \| Î.P.É. |
| 3 | 50 | MapIDName | Qubec | E9 | Québec |
| 4 | 50 | MapIDName2 | QUBEC | C9 | QUÉBEC |
| 5 | 50 | StateAlias | P.Q. \| Qu. \| Que. \| QU \| Quebec | E9 | P.Q. \| Qué. \| Que. \| QU \| Quebec |

**Display 2: FIND_FIELD Dataset Identifying Extended Characters in SAS Help Dataset GCSTATE**

VIEWTABLE: Work.Noex

| | State/province name | Normalized state/province name | State/province abbreviation | ISO country name | ISO alpha2 country code | ISO alpha3 country code | State abbreviations other than postal standards |
|---|---|---|---|---|---|---|---|
| 49 | Prince Edward Island | PRINCEEDWARDISLAND | PE | Canada | CA | CAN | P.E.I. \| .P.. |
| 50 | Qubec | QUBEC | QC | Canada | CA | CAN | P.Q. \| Qu. \| Que. \| QU \| Quebec |

**Display 3: NOEX Dataset with Extended Characters Removed**

## EXAMPLE 3 – TRANSLATE ALL INSTANCES OF EXTENDED ASCII TO STANDARD ASCII

In Display 2, we see three characters being removed: capital E acute (hex C9), capital I circumflex (hex CE), and small e acute (E9). In our final example (see Input 3), we utilize FINDCHAR, the hex identifier for the Extended characters identified in Example 2, and a suitable replacement standard ASCII character to maintain but clean the data content.

In this use of FINDCHAR, the DELCHARS and REPCHARS parameters take a list of single-byte characters which will be exchanged, in order. Table 1 below demonstrates the conversions to be performed. Note in the following example, the DELCHARS characters are presented in alphanumeric order, with the REPCHARS presented in the corresponding replacement order.

| Hex Code | Original Character | Replacement Character |
|---|---|---|
| C9 | É | E |
| CE | Î | I |
| E9 | é | e |

**Table 1: Extended Characters with Replacement Standard Characters**

```
*Starting with EXAMPLE data set, which is SAS Help data set GCSTATE.;
%findchar(ds=example,out=replace,delchars='C9CEE9'x,repchars='EIe');
```

**Input 3: Replacing Extended Characters in SAS Help Dataset GCSTATE**

The call in Input 3 creates dataset FIND_FIELD which displays the original data with the edited data (see Display 4). The FIND_FIELD macro checks all observations and all fields for the delete characters, so that each cell is documented in the output dataset. Dataset REPLACE is created as the output dataset with the characters replaced within the data (see Display 5).

VIEWTABLE: Work.Find_field

| | recordnumber | field | orig_data | new_data | remove_chars_hex | contents |
|---|---|---|---|---|---|---|
| 1 | 39 | StateAlias | N.S. \| N.-É. | N.S. \| N.-E. | C9 | Élé characters replaced |
| 2 | 49 | StateAlias | P.E.I. \| Î.P.É. | P.E.I. \| I.P.E. | CEC9 | Élé characters replaced |
| 3 | 50 | MapIDName | Québec | Quebec | E9 | Élé characters replaced |
| 4 | 50 | MapIDName2 | QUÉBEC | QUEBEC | C9 | Élé characters replaced |
| 5 | 50 | StateAlias | P.Q. \| Qué. \| Que. \| QU \| Quebec | P.Q. \| Que. \| Que. \| QU \| Quebec | E9 | Élé characters replaced |

**Display 4: FIND_FIELD Dataset Displaying Replacement Standard Characters**

VIEWTABLE: Work.Replace

| | State/province name | Normalized state/province name | State/province abbreviation | ISO country name | ISO alpha2 country code | ISO alpha3 country code | State abbreviations other than postal standards |
|---|---|---|---|---|---|---|---|
| 49 | Prince Edward Island | PRINCEEDWARDISLAND | PE | Canada | CA | CAN | P.E.I. \| I.P.E. |
| 50 | Quebec | QUEBEC | QC | Canada | CA | CAN | P.Q. \| Que. \| Que. \| QU \| Quebec |

**Display 5: REPLACE Dataset Edited to Include Standard Characters**

## CONCLUSION

There are many situations in which the characters or character encoding used to store data can cause issues in processing or transmittal of the data. Multiple functions and data processing steps must be combined to identify and handle problematic characters. The CHARCOUNT and FINDCHAR macros are simple yet useful tools to trouble shoot, identify, and handle problematic characters quickly and with minimal programmer effort.

## REFERENCES


"ASCII". *Wikipedia, The Free Encyclopedia*. Retrieved 08-Jul-2019. Available at https://en.wikipedia.org/wiki/ASCII#cite_note-Maini_2007-9.

Sanchez, Stephanie Ann. 2018. "Improving Listing Generation with ODS Excel". *Proceedings of the PharmaSUG 2018 Conference*, Seattle, WA: PharmaSUG. Available at https://www.pharmasug.org/proceedings/2018/QT/PharmaSUG-2018-QT02.pdf.

Stackhouse, Michael. 2016. "UTF What? A Guide to Using UTF-8 Encoded Data in a SDTM Submission". *Proceedings of the PharmaSUG2016 Conference,* Denver, COBB: PharmaSUG. Available at https://www.pharmasug.org/proceedings/2016/BB/PharmaSUG-2016-BB16.pdf


## ACKNOWLEDGMENTS


The authors would like to thank our many associates, colleagues, and collaborators who provided the data files that inspired the creation of the CHARCOUNT and FINDCHAR macros.


## RECOMMENDED READING

- Sample 52832: SAS® Clinical Standards Toolkit - _cstFindCharacters and _cstFixCharacters SAS® macros to find and remove ASCII extended characters. (12-Jun-2014). Available at: http://support.sas.com/kb/52/832.html

- SAS® 9.4 National Language Support (NLS): Reference Guide, Fifth Edition. Available at: https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=nlsref&docsetTarget=titlepage.htm&locale=en

## CONTACT INFORMATION

Dan Konkler
Roche Molecular Solutions
dan.konkler@roche.com

## APPENDIX 1: ASCII TABLE

The table below describes the 256 single byte ASCII and extended ASCII characters encoded using WLATIN1.

| Hex | Decimal | Printable Character | Description | | SAS Character Classes |
|---|---|---|---|---|---|
| 00 | 0 | | NUL | null | CNTRL |
| 01 | 1 | | SOH | start of heading | CNTRL |
| 02 | 2 | | STX | start of text | CNTRL |
| 03 | 3 | | EOT | end of text | CNTRL |
| 04 | 4 | | EOT | end of transmission | CNTRL |
| 05 | 5 | | ENQ | enquiry | CNTRL |
| 06 | 6 | | ACK | acknowledge | CNTRL |
| 07 | 7 | | BEL | bell | CNTRL |
| 08 | 8 | | BS | backspace | CNTRL |
| 09 | 9 | | HT | horizontal tab | CNTRL SPACE |
| 0A | 10 | | LF,NL | line feed, new line | CNTRL SPACE |
| 0B | 11 | | VT | vertical tab | CNTRL SPACE |
| 0C | 12 | | FF,NP | form feed, new page | CNTRL SPACE |
| 0D | 13 | | CR | carriage return | CNTRL SPACE |
| 0E | 14 | | SO | shift out | CNTRL |
| 0F | 15 | | SI | shift in | CNTRL |
| 10 | 16 | | DLE | data link escape | CNTRL |
| 11 | 17 | | DC1 | device control 1 | CNTRL |
| 12 | 18 | | DC2 | device control 2 | CNTRL |
| 13 | 19 | | DC3 | device control 3 | CNTRL |
| 14 | 20 | | DC4 | device control 4 | CNTRL |
| 15 | 21 | | NAK | negative acknowledge | CNTRL |
| 16 | 22 | | SYN | synchronous idle | CNTRL |
| 17 | 23 | | ETB | end of transmission block | CNTRL |
| 18 | 24 | | CAN | cancel | CNTRL |
| 19 | 25 | | EM | end of medium | CNTRL |
| 1A | 26 | | SUB | substitute | CNTRL |
| 1B | 27 | | ESC | escape | CNTRL |
| 1C | 28 | | FS | file separator | CNTRL |
| 1D | 29 | | GS | group separator | CNTRL |
| 1E | 30 | | RS | record separator | CNTRL |
| 1F | 31 | | US | unit separator | CNTRL |
| 20 | 32 | | space | | PRINT SPACE |
| 21 | 33 | ! | exclamation mark | | GRAPH PRINT PUNCT |
| 22 | 34 | " | double quotation mark | | GRAPH PRINT PUNCT |
| 23 | 35 | # | number sign, pound | | GRAPH PRINT PUNCT |
| 24 | 36 | $ | dollar sign | | GRAPH PRINT PUNCT |
| 25 | 37 | % | percent sign | | GRAPH PRINT PUNCT |
| 26 | 38 | & | ampersand | | GRAPH PRINT PUNCT |
| 27 | 39 | ' | apostrophe, single quote mark | | GRAPH PRINT PUNCT |

| Hex | Decimal | Printable Character | Description | SAS Character Classes |
|---|---|---|---|---|
| 28 | 40 | ( | left parenthesis | GRAPH PRINT PUNCT |
| 29 | 41 | ) | right parenthesis | GRAPH PRINT PUNCT |
| 2A | 42 | * | asterisk | GRAPH PRINT PUNCT |
| 2B | 43 | + | plus sign | GRAPH PRINT PUNCT |
| 2C | 44 | , | comma | GRAPH PRINT PUNCT |
| 2D | 45 | - | minus sign, hyphen | GRAPH PRINT PUNCT |
| 2E | 46 | . | period, decimal point, full stop | GRAPH PRINT PUNCT |
| 2F | 47 | / | slash, virgule, solidus | GRAPH PRINT PUNCT |
| 30 | 48 | 0 | digit 0 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 31 | 49 | 1 | digit 1 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 32 | 50 | 2 | digit 2 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 33 | 51 | 3 | digit 3 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 34 | 52 | 4 | digit 4 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 35 | 53 | 5 | digit 5 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 36 | 54 | 6 | digit 6 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 37 | 55 | 7 | digit 7 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 38 | 56 | 8 | digit 8 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 39 | 57 | 9 | digit 9 | ALNUM DIGIT GRAPH NAME PRINT XDIGIT |
| 3A | 58 | : | colon | GRAPH PRINT PUNCT |
| 3B | 59 | ; | semicolon | GRAPH PRINT PUNCT |
| 3C | 60 | < | less-than sign | GRAPH PRINT PUNCT |
| 3D | 61 | = | equal sign | GRAPH PRINT PUNCT |
| 3E | 62 | > | greater-than sign | GRAPH PRINT PUNCT |
| 3F | 63 | ? | question mark | GRAPH PRINT PUNCT |
| 40 | 64 | @ | commercial at sign | GRAPH PRINT PUNCT |
| 41 | 65 | A | capital A | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 42 | 66 | B | capital B | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 43 | 67 | C | capital C | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 44 | 68 | D | capital D | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 45 | 69 | E | capital E | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 46 | 70 | F | capital F | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER XDIGIT |
| 47 | 71 | G | capital G | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 48 | 72 | H | capital H | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 49 | 73 | I | capital I | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4A | 74 | J | capital J | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4B | 75 | K | capital K | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4C | 76 | L | capital L | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4D | 77 | M | capital M | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4E | 78 | N | capital N | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 4F | 79 | O | capital O | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 50 | 80 | P | capital P | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 51 | 81 | Q | capital Q | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 52 | 82 | R | capital R | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 53 | 83 | S | capital S | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |

| Hex | Decimal | Printable Character | Description | SAS Character Classes |
|---|---|---|---|---|
| 54 | 84 | T | capital T | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 55 | 85 | U | capital U | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 56 | 86 | V | capital V | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 57 | 87 | W | capital W | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 58 | 88 | X | capital X | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 59 | 89 | Y | capital Y | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 5A | 90 | Z | capital Z | ALNUM ALPHA FIRST GRAPH NAME PRINT UPPER |
| 5B | 91 | [ | left square bracket | GRAPH PRINT PUNCT |
| 5C | 92 | \ | backslash, reverse solidus | GRAPH PRINT PUNCT |
| 5D | 93 | ] | right square bracket | GRAPH PRINT PUNCT |
| 5E | 94 | ^ | spacing circumflex accent | GRAPH PRINT PUNCT |
| 5F | 95 | _ | spacing underscore, low line, horizontal b | FIRST GRAPH NAME PRINT PUNCT |
| 60 | 96 | ` | spacing grave accent, back apostrophe | GRAPH PRINT PUNCT |
| 61 | 97 | a | small a | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 62 | 98 | b | small b | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 63 | 99 | c | small c | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 64 | 100 | d | small d | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 65 | 101 | e | small e | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 66 | 102 | f | small f | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT XDIGIT |
| 67 | 103 | g | small g | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 68 | 104 | h | small h | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 69 | 105 | i | small i | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6A | 106 | j | small j | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6B | 107 | k | small k | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6C | 108 | l | small l | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6D | 109 | m | small m | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6E | 110 | n | small n | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 6F | 111 | o | small o | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 70 | 112 | p | small p | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 71 | 113 | q | small q | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 72 | 114 | r | small r | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 73 | 115 | s | small s | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 74 | 116 | t | small t | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 75 | 117 | u | small u | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 76 | 118 | v | small v | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 77 | 119 | w | small w | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 78 | 120 | x | small x | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 79 | 121 | y | small y | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 7A | 122 | z | small z | ALNUM ALPHA FIRST GRAPH LOWER NAME PRINT |
| 7B | 123 | { | left brace, left curly bracket | GRAPH PRINT PUNCT |
| 7C | 124 | \| | vertical bar | GRAPH PRINT PUNCT |
| 7D | 125 | } | right brace, right curly bracket | GRAPH PRINT PUNCT |
| 7E | 126 | ~ | tilde accent | GRAPH PRINT PUNCT |
| 7F | 127 | • | DEL delete | CNTRL |

| Hex | Decimal | Printable Character | Description | SAS Character Classes |
|---|---|---|---|---|
| 80 | 128 | € | | GRAPH PRINT PUNCT |
| 81 | 129 | • | | CNTRL |
| 82 | 130 | ‚ | | GRAPH PRINT PUNCT |
| 83 | 131 | ƒ | | ALNUM ALPHA GRAPH LOWER PRINT |
| 84 | 132 | „ | | GRAPH PRINT PUNCT |
| 85 | 133 | … | | GRAPH PRINT PUNCT |
| 86 | 134 | † | | GRAPH PRINT PUNCT |
| 87 | 135 | ‡ | | GRAPH PRINT PUNCT |
| 88 | 136 | ˆ | | GRAPH PRINT PUNCT |
| 89 | 137 | ‰ | | GRAPH PRINT PUNCT |
| 8A | 138 | Š | | ALNUM ALPHA GRAPH PRINT UPPER |
| 8B | 139 | ‹ | | GRAPH PRINT PUNCT |
| 8C | 140 | Œ | | ALNUM ALPHA GRAPH PRINT UPPER |
| 8D | 141 | • | | CNTRL |
| 8E | 142 | Ž | | ALNUM ALPHA GRAPH PRINT UPPER |
| 8F | 143 | • | | CNTRL |
| 90 | 144 | • | | CNTRL |
| 91 | 145 | ‘ | | GRAPH PRINT PUNCT |
| 92 | 146 | ’ | | GRAPH PRINT PUNCT |
| 93 | 147 | “ | | GRAPH PRINT PUNCT |
| 94 | 148 | ” | | GRAPH PRINT PUNCT |
| 95 | 149 | • | | GRAPH PRINT PUNCT |
| 96 | 150 | – | | GRAPH PRINT PUNCT |
| 97 | 151 | — | | GRAPH PRINT PUNCT |
| 98 | 152 | ˜ | | GRAPH PRINT PUNCT |
| 99 | 153 | ™ | | GRAPH PRINT PUNCT |
| 9A | 154 | š | | ALNUM ALPHA GRAPH LOWER PRINT |
| 9B | 155 | › | | GRAPH PRINT PUNCT |
| 9C | 156 | œ | | ALNUM ALPHA GRAPH LOWER PRINT |
| 9D | 157 | • | | CNTRL |
| 9E | 158 | ž | | ALNUM ALPHA GRAPH LOWER PRINT |
| 9F | 159 | Ÿ | | ALNUM ALPHA GRAPH PRINT UPPER |
| A0 | 160 | | non-breaking space | PRINT |
| A1 | 161 | ¡ | inverted exclamation mark | GRAPH PRINT PUNCT |
| A2 | 162 | ¢ | cent sign | GRAPH PRINT PUNCT |
| A3 | 163 | £ | pound sterling sign | GRAPH PRINT PUNCT |
| A4 | 164 | ¤ | general currency sign | GRAPH PRINT PUNCT |
| A5 | 165 | ¥ | yen sign | GRAPH PRINT PUNCT |
| A6 | 166 | ¦ | broken vertical bar | GRAPH PRINT PUNCT |
| A7 | 167 | § | section sign | GRAPH PRINT PUNCT |
| A8 | 168 | ¨ | spacing dieresis or umlaut | GRAPH PRINT PUNCT |
| A9 | 169 | © | copyright sign | GRAPH PRINT PUNCT |
| AA | 170 | ª | feminine ordinal sign | ALNUM ALPHA GRAPH LOWER PRINT |
| AB | 171 | « | left double angle quote or guillemet | GRAPH PRINT PUNCT |

| Hex | Decimal | Printable Character | Description | SAS Character Classes |
|---|---|---|---|---|
| AC | 172 | ¬ | logical not sign | GRAPH PRINT PUNCT |
| AD | 173 | - | soft hyphen | CNTRL |
| AE | 174 | ® | registered trademark sign | GRAPH PRINT PUNCT |
| AF | 175 | ¯ | spacing macron long accent | GRAPH PRINT PUNCT |
| B0 | 176 | ° | degree sign | GRAPH PRINT PUNCT |
| B1 | 177 | ± | plus-or-minus sign | GRAPH PRINT PUNCT |
| B2 | 178 | ² | superscript 2 | GRAPH PRINT PUNCT |
| B3 | 179 | ³ | superscript 3 | GRAPH PRINT PUNCT |
| B4 | 180 | ´ | spacing acute accent | GRAPH PRINT PUNCT |
| B5 | 181 | µ | micro sign, mu | ALNUM ALPHA GRAPH LOWER PRINT |
| B6 | 182 | ¶ | paragraph sign, pilcrow sign | GRAPH PRINT PUNCT |
| B7 | 183 | · | middle dot, centered dot | GRAPH PRINT PUNCT |
| B8 | 184 | ¸ | spacing cedilla | GRAPH PRINT PUNCT |
| B9 | 185 | ¹ | superscript 1 | GRAPH PRINT PUNCT |
| BA | 186 | º | masculine ordinal indicator | ALNUM ALPHA GRAPH LOWER PRINT |
| BB | 187 | » | right double angle quote or guillemet | GRAPH PRINT PUNCT |
| BC | 188 | ¼ | fraction 1/4 | GRAPH PRINT PUNCT |
| BD | 189 | ½ | fraction 1/2 | GRAPH PRINT PUNCT |
| BE | 190 | ¾ | fraction 3/4 | GRAPH PRINT PUNCT |
| BF | 191 | ¿ | inverted question mark | GRAPH PRINT PUNCT |
| C0 | 192 | À | capital A grave | ALNUM ALPHA GRAPH PRINT UPPER |
| C1 | 193 | Á | capital A acute | ALNUM ALPHA GRAPH PRINT UPPER |
| C2 | 194 | Â | capital A circumflex | ALNUM ALPHA GRAPH PRINT UPPER |
| C3 | 195 | Ã | capital A tilde | ALNUM ALPHA GRAPH PRINT UPPER |
| C4 | 196 | Ä | capital A dieresis or umlaut | ALNUM ALPHA GRAPH PRINT UPPER |
| C5 | 197 | Å | capital A ring | ALNUM ALPHA GRAPH PRINT UPPER |
| C6 | 198 | Æ | capital AE ligature | ALNUM ALPHA GRAPH PRINT UPPER |
| C7 | 199 | Ç | capital C cedilla | ALNUM ALPHA GRAPH PRINT UPPER |
| C8 | 200 | È | capital E grave | ALNUM ALPHA GRAPH PRINT UPPER |
| C9 | 201 | É | capital E acute | ALNUM ALPHA GRAPH PRINT UPPER |
| CA | 202 | Ê | capital E circumflex | ALNUM ALPHA GRAPH PRINT UPPER |
| CB | 203 | Ë | capital E dieresis or umlaut | ALNUM ALPHA GRAPH PRINT UPPER |
| CC | 204 | Ì | capital I grave | ALNUM ALPHA GRAPH PRINT UPPER |
| CD | 205 | Í | capital I acute | ALNUM ALPHA GRAPH PRINT UPPER |
| CE | 206 | Î | capital I circumflex | ALNUM ALPHA GRAPH PRINT UPPER |
| CF | 207 | Ï | capital I dieresis or umlaut | ALNUM ALPHA GRAPH PRINT UPPER |
| D0 | 208 | Ð | capital ETH | ALNUM ALPHA GRAPH PRINT UPPER |
| D1 | 209 | Ñ | capital N tilde | ALNUM ALPHA GRAPH PRINT UPPER |
| D2 | 210 | Ò | capital O grave | ALNUM ALPHA GRAPH PRINT UPPER |
| D3 | 211 | Ó | capital O acute | ALNUM ALPHA GRAPH PRINT UPPER |
| D4 | 212 | Ô | capital O circumflex | ALNUM ALPHA GRAPH PRINT UPPER |
| D5 | 213 | Õ | capital O tilde | ALNUM ALPHA GRAPH PRINT UPPER |
| D6 | 214 | Ö | capital O dieresis or umlaut | ALNUM ALPHA GRAPH PRINT UPPER |
| D7 | 215 | × | multiplication sign | GRAPH PRINT PUNCT |

| Hex | Decimal | Printable Character | Description | SAS Character Classes |
|---|---|---|---|---|
| D8 | 216 | Ø | capital O slash | ALNUM ALPHA GRAPH PRINT UPPER |
| D9 | 217 | Ù | capital U grave | ALNUM ALPHA GRAPH PRINT UPPER |
| DA | 218 | Ú | capital U acute | ALNUM ALPHA GRAPH PRINT UPPER |
| DB | 219 | Û | capital U circumflex | ALNUM ALPHA GRAPH PRINT UPPER |
| DC | 220 | Ü | capital U dieresis or umlaut | ALNUM ALPHA GRAPH PRINT UPPER |
| DD | 221 | Ý | capital Y acute | ALNUM ALPHA GRAPH PRINT UPPER |
| DE | 222 | Þ | capital THORN | ALNUM ALPHA GRAPH PRINT UPPER |
| DF | 223 | ß | small sharp s, sz ligature | ALNUM ALPHA GRAPH LOWER PRINT UPPER |
| E0 | 224 | à | small a grave | ALNUM ALPHA GRAPH LOWER PRINT |
| E1 | 225 | á | small a acute | ALNUM ALPHA GRAPH LOWER PRINT |
| E2 | 226 | â | small a circumflex | ALNUM ALPHA GRAPH LOWER PRINT |
| E3 | 227 | ã | small a tilde | ALNUM ALPHA GRAPH LOWER PRINT |
| E4 | 228 | ä | small a dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |
| E5 | 229 | å | small a ring | ALNUM ALPHA GRAPH LOWER PRINT |
| E6 | 230 | æ | small ae ligature | ALNUM ALPHA GRAPH LOWER PRINT |
| E7 | 231 | ç | small c cedilla | ALNUM ALPHA GRAPH LOWER PRINT |
| E8 | 232 | è | small e grave | ALNUM ALPHA GRAPH LOWER PRINT |
| E9 | 233 | é | small e acute | ALNUM ALPHA GRAPH LOWER PRINT |
| EA | 234 | ê | small e circumflex | ALNUM ALPHA GRAPH LOWER PRINT |
| EB | 235 | ë | small e dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |
| EC | 236 | ì | small i grave | ALNUM ALPHA GRAPH LOWER PRINT |
| ED | 237 | í | small i acute | ALNUM ALPHA GRAPH LOWER PRINT |
| EE | 238 | î | small i circumflex | ALNUM ALPHA GRAPH LOWER PRINT |
| EF | 239 | ï | small i dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |
| F0 | 240 | ð | small eth | ALNUM ALPHA GRAPH LOWER PRINT |
| F1 | 241 | ñ | small n tilde | ALNUM ALPHA GRAPH LOWER PRINT |
| F2 | 242 | ò | small o grave | ALNUM ALPHA GRAPH LOWER PRINT |
| F3 | 243 | ó | small o acute | ALNUM ALPHA GRAPH LOWER PRINT |
| F4 | 244 | ô | small o circumflex | ALNUM ALPHA GRAPH LOWER PRINT |
| F5 | 245 | õ | small o tilde | ALNUM ALPHA GRAPH LOWER PRINT |
| F6 | 246 | ö | small o dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |
| F7 | 247 | ÷ | division sign | GRAPH PRINT PUNCT |
| F8 | 248 | ø | small o slash | ALNUM ALPHA GRAPH LOWER PRINT |
| F9 | 249 | ù | small u grave | ALNUM ALPHA GRAPH LOWER PRINT |
| FA | 250 | ú | small u acute | ALNUM ALPHA GRAPH LOWER PRINT |
| FB | 251 | û | small u circumflex | ALNUM ALPHA GRAPH LOWER PRINT |
| FC | 252 | ü | small u dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |
| FD | 253 | ý | small y acute | ALNUM ALPHA GRAPH LOWER PRINT |
| FE | 254 | þ | small thorn | ALNUM ALPHA GRAPH LOWER PRINT |
| FF | 255 | ÿ | small y dieresis or umlaut | ALNUM ALPHA GRAPH LOWER PRINT |

## APPENDIX 2: CHARCOUNT MACRO

```
%macro charcount(lib=,mem=,record=no,maxmem=999,delwork=yes);

/*-------------------------------------------------------------------------------------------------------------------------------
Function: Count occurrences of every ASCII character in a file or library.

   lib=      =[sas libname]      required
   mem=      =[sas member name]  optional
   record    =[yes | no]         default is no
   maxmem    =[nnn]              default is 999, maximum number of members from lib to process
   delwork   =[yes | no]         default is yes, will delete all intermediate work files

Input: Any SAS library and optionally any sas member

Output: charcount_h  [Horizontal layout]
           output fields:
           sasds            the sas member name
           record           total or the record number within the member
           control_00_1F    total count of characters between '00'x and '1F'x
           spaces_20        total count of space characters in the file
           printable_21_7F  total count of printable characters in the file
           extended_7F_FF   total count of extended characters in the file
           _0 to _255       total count of each ASCII character in the file

Output: charcount_v [Vertical layout]
           output fields:
           dec              decimal ASCII character
           character        ASCII character
           ...              one variable per member with counts for each character in that member

Example 1 - output file charcount_v will have 256 obs, one for each possible ASCII character
            each record will have an overall count(the variable will have the name of the mem= parm)
            and a column for each record in the file like _1 to _99 (if there were 99 records)
%charcount(lib=sashelp,mem=cars,record=yes);

Example 2 - same as above but the count for each record in the file will not be produced
%charcount(lib=sashelp,mem=cars,record=no);

Example 3 - output file charcount_v will have 256 obs, one for each possible ASCII character
```

```
             each record will have the total for each of the first 10(because maxmem is specified) SAS files
%charcount(lib=sashelp,maxmem=10,record=yes,delwork=no);

Example 4 - output file charcount_v will have 256 obs, one for each possible ASCII character
             each record will have the total for each of the SAS files
             the record=yes parameter will be ignored because there are more than 10 SAS files in sashelp
%charcount(lib=sashelp,record=yes);

Programmer: Dan Konkler
-----------------------------------------------------------------------------------------------------------------------------------------*/

proc datasets nowarn; delete reccount out_: charcount: ; quit;

%if &mem=%str() and &lib ne %str() %then %do;
    proc sql noprint;  select memname, count(*) into :MEMBER separated by '/' , :nummem
              from sashelp.vtable where libname="%upcase(&lib)" and memtype='DATA'; quit;
%end;
%else %do; %let nummem=1; %let member=&mem ; %end;
%if &maxmem lt &nummem %then %let nummem = &maxmem ;
%put &nummem &member ;

%if &mem=%str() and &nummem gt 10 %then %let record=no;   * force record=no if more than 10 members ;


%if &lib = %str() %then %let lib=work;

%let processed = 0;
%do i = 1 %to &nummem;
%let processed = %eval(&processed + 1);

options varinitchk=nonote;
data charcount;
  %let thismem = %scan(&member , &i , /) ;
  if nobs=0 then do; call symput('numobs','0'); put "*** File &lib..&thismem has no observations"; stop; end;
  set &lib..&thismem end=lastrec nobs=nobs;
  array ch(*) $ _character_;
  array asccharS(*) 8 c000-c255;
  array asccharx(*) 8 x000-x255;
  if _n_ = 1 then do;
    x=strip(put(dim(ch),best.)); call symput('numobs',x); drop x;
    if x=0 then do; put "*** File &lib..&thismem has no character variables"; stop; end;
```

```
  end;
  do i = 1 to dim(ascchars); ascchars(i)=.; end;
  do i = 1 to dim(ch);
    do loop1 = 1 to length(ch(i));
       char= char(ch(i),loop1);  hexval = input(put(char(ch(i),loop1),hex.),hex2.); * put char= hexval=;
       ascchars(hexval+1)++1;
       asccharx(hexval+1)++1;
    end;
  end;
  recnum = _n_;
  %if %upcase(&record)=YES %then output; ;
  if lastrec then do; recnum=99999999; do i = 1 to dim(ascchars); ascchars(i)=asccharx(i); end; output; end;
run;
%if &numobs gt 0 %then %do;
  proc transpose data=charcount out=charcount2;
    by recnum ;
    var c000-c255;
  run;
  proc sort data=charcount2; by _name_ recnum; run;
  proc transpose data=charcount2 out=charcount3;
    by _name_ ;
    var col1;
    id recnum;
  run;
  data charcount_v_&i;
    retain dec character _99999999;
    length character $ 1;
    set charcount3;
    dec = input(substr(_name_,2),best.);
    character = byte(dec);
    drop _name_;
    rename _99999999=&thismem ;
  run;
  proc transpose data=charcount_v_&i out=charcount5;
    var &thismem  %if %upcase(&record)=YES %then _: ; ;
    id dec;
    idl character ;
  run;
  data out_&i ; retain sasds record ; sasds="&thismem";
    set charcount5; if _name_="&thismem" then record ='Total'; else record=substr(_name_,2);
    drop _name_;
```

```
  run;
%end;
%else %let processed = %eval(&processed - 1);
%end;

%put **** Number of files processed is &processed;
%if &processed gt 0 %then %do;

  options nodsnferr;
  data charcount_h;
  retain sasds record control_00_1F spaces_20 printable_21_7E extended_7F_FF;
  length sasds $ 32; set out_: ;
  control_00_1F   = sum(of _0-_31);
  spaces_20       = sum(of _32);
  printable_21_7E = sum(of _33-_126);
  extended_7F_FF  = sum(of _127-_255);
  run;
  data charcount_v;
    merge charcount_v_: ;
    by dec;
  run;
  options dsnferr;

  %if %upcase(&record) = YES %then %do;
    data reccount;
      length sasds $ 32;
      set out_: ;
    run;
  %end;

%end;

%if &delwork = yes %then %do; proc datasets nowarn; delete out_: charcount_v_: charcount charcount2 charcount3 charcount5 ; quit;
%end;

%mend charcount;
```

## APPENDIX 3: FINDCHAR MACRO

```
%macro findchar(ds=,out=,delchars=,keepchars=,repchars=,function=,runid=);

/*-------------------------------------------------------------------------------------------------------------------------
Function: Find characters that are either in or not in a specified character group.

   ds        =[sas dataset] required
   out       =[new sas dataset with fields updated] optional

   at least one of these:
   delchars  =[sas character class or a literal character string (hex or ASCII)] all these characters will be compressed from
              the variable
   keepchars =[sas character class or a literal character string (hex or ASCII)] all characters not in this string will be compressed
              from the variable
   repchars  =[literal character string (hex or ASCII)] - optional, used only if delchars is specified
   function  =[one of the SAS string functions] REVERSE STRIP LOWCASE UPCASE PROPCASE COMPBL COMPRESS TRIM TRIMN LEFT SOUNDEX FIRST

   runid     =[suffix identifier added to find_field and out= dataset names] optional, i.e. if out=ABC and runid=1 then output is ABC1

   character classes: ALNUM ALPHA CNTRL DIGIT GRAPH LOWER PRINT PUNCT UPPER QUOTES EXTENDED HEX, the last 3 are user defined in macro

Input: Any SAS dataset

Output: work.find_field
        output fields:
           recordnumber     original recordnumber
           field            variable name
           orig_data        original data from the data field
           new_data         values of characters that are kept
           remove_chars_hex hex values of characters that are removed
           contents         description of the change made to the data

Output: work.&out - if out parameter is specified
        output fields:
           all records in the input file with all character fields updated that are listed in the find_field file

Programmer: Dan Konkler
-------------------------------------------------------------------------------------------------------------------------*/
```

```
%let dsorig = &ds;
%let ds     = %upcase(&ds);
%let dropchars= %str(recordnumber field orig_data new_data remove_chars_hex contents hexkeep pos i remove_chars);
%if &repchars ne %str() %then %let dropchars= %str(&dropchars repcharslen repcharstr);

%if &function ne %str() %then %do;
     %let charlist = %upcase(&function) ;
%end;
%else %do;
  %if &keepchars ne %str() %then %do;
     %let negate = %str();
     %let charlist = &keepchars ;
     %let keeprem=kept;
     %let repchars = %str(); * ignore if keepchars is specified ;
  %end;
  %else %do;
    %if &delchars ne %str() %then %do;
      %if &repchars = %str() %then %let negate = not;     %else %let negate = %str();
      %if &repchars = %str() %then %let keeprem= removed; %else %let keeprem= replaced;
      %let charlist = &delchars ;
    %end;
  %end;
%end;

%let dspart = %scan(&ds.,1, ()) ;
  %if %index(&dspart.,.) %then %do; %let lib = %scan(&dspart.,1,.); %let mem = %scan(&dspart.,2,.); %end;
                         %else %do; %let lib = WORK;                %let mem = &dspart.;            %end;
proc sql noprint;
     select count(name) into :charvarcount           from sashelp.vcolumn where libname="&lib" and memname="&mem" and type='char';
quit;

%if &charvarcount le 0 %then %do;
  %put %str(*** File &dsorig has no character variables or file does not exist.);
%end;
%else %do;
data                           find_field&runid(keep=recordnumber field contents orig_data new_data remove_chars_hex )
     %if &out ne %str() %then &out.&runid      (drop=&dropchars) ; ;

  retain recordnumber field orig_data new_data remove_chars_hex contents ;

  set &dsorig ;
```

```
  array ch(*) $ _character_;
  length field $ 32 contents $ 300 hexkeep $ 256;

  recordnumber=_n_;

  %if %index(ALNUM ALPHA CNTRL DIGIT GRAPH LOWER PRINT PUNCT UPPER QUOTES EXTENDED HEX, %upcase(&charlist.)) %then %do;
      %let charlist = %upcase(&charlist); %let parmtype=name; %end;
  %else %if %index(REVERSE STRIP LOWCASE UPCASE PROPCASE COMPBL COMPRESS TRIM TRIMN LEFT SOUNDEX FIRST, %upcase(&charlist.)) %then
%do;
      %let parmtype=function; %end;
  %else %let parmtype=quotedstring;

    %if &parmtype = name         %then contents="&charlist characters &keeprem"; ;
    %if &parmtype = function     %then contents="&charlist function"; ;
    %if &parmtype = quotedstring %then contents=&charlist !! " characters &keeprem"; ;
    retain pos 0 hexkeep ;

    * INITIALIZE THE ARRAY OF CHARACTERS TO KEEP OR REMOVE ;
    if _n_=1 then do;
    hexkeep='';
    %if &parmtype ne function %then %do;
        do i = 0 to 255;

          * ADD YOUR OWN CHARACTER GROUPS HERE, LEAVE THE ELSE CLAUSE TO PROCESS THE ANY.... TYPES ;
                %if &parmtype=quotedstring        %then %do; if &negate index(&charlist.,byte(i)) then do; %end;
          %else %if %upcase(&charlist)=QUOTES     %then %do; if &negate index('222760'x ,byte(i)) then do; %end;
          %else %if %upcase(&charlist)=EXTENDED   %then %do; if &negate (i ge 128)                then do; %end;
          %else %if %upcase(&charlist)=HEX        %then %do; if &negate (i le 31 or i ge 127)     then do; %end;
          %else                                         %do; if &negate (any&charlist.(byte(i)))  then do; %end;

          pos++1;
          substr(hexkeep,pos,1) = byte(i); end;
        end;
        %if &repchars ne %str() %then %do;
            repcharslen = length(&repchars) ;
            if repcharslen = pos or repcharslen = 1 then do;
              if repcharslen = 1 then repcharstr = repeat(&repchars.,pos-1);
                                 else repcharstr = &repchars.;
                   retain repcharslen repcharstr ;
            end;
            else do;
```

```
                put '*** REPCHARS PARAMETER LENGTH MUST EITHER BE 1 OR THE LENGTH OF THE CHARACTER DELETE STRING ***' ;
                put "*** Length of replacement string:" repcharslen "length of delete string:" pos;
                stop;
            end;
        %end;
    %end;
    end;

    do i = 1 to dim(ch);

      orig_data        = ch(i);
      %if &parmtype = function %then %do;
           new_data        = &charlist.(ch(i));
           remove_chars    = ' ';
      %end;
      %else %if &repchars = %str() %then %do;
           new_data        = compress(ch(i),' '!!hexkeep,'OK');
           remove_chars    = compress(ch(i),hexkeep);
      %end;
      %else %do;
           new_data        = translate(ch(i),repcharstr,hexkeep);
           remove_chars    = compress(ch(i),hexkeep,'OK');
      %end;

      if new_data ne orig_data then do;
        remove_chars_hex = put(strip(remove_chars),hex.);
        field            = vname(ch(i));
        output find_field&runid ;
      end;
        %if &out ne %str() %then ch(i) = new_data; ;

    end;

    %if &out ne %str() %then output &out.&runid ;  ;

run;
%end;
%mend findchar;
```