UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 87

April 1986

Title:

Handling Non-printing Characters in EMAS Files

Author: Arthur Wilson

Contact: Advisory service

Software Support Category: See Note 15

Synopsis

EMAS 2900 and EMAS-A text files can contain non-printing characters. This Note describes methods for finding these non-printing characters, highlighting them, inserting, removing and re-locating them.

Keywords

ADDNP, BECCE, CHEF, CODECHANGE, COUNT, #DEC, DEGROT, DETAB, ECCE, EM, FIXCODE, #HEX, Non-printing characters, NOPRINT, SUPERSNAP.

TABLE OF CONTENTS

1. INTRODUCTION

Files on the EMAS operating system consist of a set of accepted characters; this is referred to as the Internal Character Set and is based on the set defined by the International Standards Organization (ISO).

Each character is represented in the computer by a number in the range 0 to 127. Thus, for example, when you type a capital A your terminal sends the number 65 to the computer; and when a line printer receives the number 65 it prints a capital A. These numbers are called *Character Codes*. All the usual printing characters - upper and lower case letters, numbers, punctuation signs etc. - have codes between 32 and 126 (they are called *printing characters* because they produce a visible result when sent to a printer or to your terminal). The Internal Character Set and its character codes are shown in tabular form in Appendix I.

The other character codes - 0 to 31 and 127 - are known as *control characters* or *non-printing characters*, and these are used for special functions in the computer or on an output device. For example, a line printer is positioned to a new line by a carriage return followed by a line feed - codes 13 and 10 - or to a new page by a form feed character - code 12. The complete set of all codes is shown in tabular form in Appendix I; the names and effects of some of the control codes are summarized in Appendix V.

It is worth mentioning in passing that, while humans use the decimal numbering system, computers do not. The main alternative numbering system is Hexadecimal (HEX), which works to the base of 16 rather than 10 (see Appendix II for a description of the HEX numbering system). In this Note, HEX numbers are preceded by the letter X, while decimal numbers have no prefix.

This Note explains non-printing characters and describes methods for handling them. If all you want to know about them is how to find out if you have any in your file and then how to remove or correct them, see the section headed *NON-PRINTING CHARACTER PROBLEMS AND HOW TO SOLVE THEM* on page 5. If you want to perform more complex functions then see the section headed *EDITORS FOR THE EXPERTS* on page 11.

To access the programs described in this Note, you must issue the following commands once only:

Command: **OPTION SEARCHDIR=CONLIB.GENERAL** {on EMAS 2900}
Command: **OPTION SEARCHDIR=KNTLIB.GENERAL** {on EMAS 2900}

Command: **SEARCHDIR KNTLIB:GENERAL** {on EMAS-A}
Command: **SEARCHDIR ERCLIB:GENERAL** {on EMAS-A}

You are advised always to have used the command TERMINALTYPE at the start of any session when you are using any of the commands described in this Note.

In most of the examples in this Note, I have used the file named CONLIB.EXAMPLES_CTRLCHAR1 which all users on EMAS or BUSH can access. On EMAS-A this file is called ERCLIB:EXAMPLES_CTRLCHAR1.

This file contains a mixture of printing and non-printing characters; the sequence of characters is as follows:

a. a form feed character (decimal 12, X0C);

b. 11 spaces (decimal 32, X12);

c. a horizontal tab character (decimal 9, X09);

d. the letters H, E, L and P;

e. a line feed character (decimal 10, X0A).

If you were to list this file on your terminal the output would look like this:

Command: **LIST CONLIB.EXAMPLES_CTRLCHAR1**
HELP
Command:

You will notice that the form feed, horizontal tab and line feed characters are not visible in your output, this is because they are non-printing characters.

This file is permitted to all users, so you can copy it into your process and try out some of the examples shown in this Note. All these examples were carried out on EMAS or BUSH. If you wish to try them out on EMAS-A, you should copy the file ERCLIB:EXAMPLES_CTRLCHAR1. To copy this file you would issue the following command:

Command: **COPY CONLIB.EXAMPLES_CTRLCHAR1,TEST** {on EMAS 2900}
Command: **COPY ERCLIB:EXAMPLES_CTRLCHAR1,TEST** {on EMAS-A}

The three non-printing characters contained in this file are not the only ones that can be handled by the commands described in this Note. The most common ones and their functions are described in Appendix V.

All the commands described in this Note work on character files and character members of partitioned files.

If you are using Fortran programs from DEC systems, you will find that they are spaced out using the horizontal tab (decimal 9, X09) character. There is a program which removes these characters and inserts the equivalent amount of spaces. This program is called DETAB: for more details see section 10 on page 9.

2. Acknowledgements

I would like to thank Nick Stroud for helping me to make this User Note a more helpful one to both the novice and the expert. The structure of the Note and some of the text were suggested by him. I would also like to thank Neil Hamilton-Smith for suggesting this Note and for giving me his comments at various stages. John Murison and Malcolm Brown also gave me comments on various drafts and I would like to take this opportunity to thank them for their help.

3. Non-printing character problems

The main problem with non-printing characters arises when they appear in your file where they are not wanted. This can cause the terminal or the printer you are using to behave in a peculiar way. It may also adversely affect the package or program you are using.

Their appearance can be caused by a variety of reasons, some of which are described below.

(1) Non-printing characters inserted by the BACK SPACE key

If you were to use the BACK SPACE key while editing a file, you would move the cursor back one space over the offending character. This removes the previous character from the screen, but not from the file. What it does is insert a BACK SPACE character in your file, so instead of deleting a character you are adding one.

For this reason avoid using the BACK SPACE key as it can insert unwanted non-printing characters in your file.

(2) Non-printing characters inserted by EMAS programs

If you created your file by using a text formatting program such as SCRIBE or LAYOUT, it is possible that these programs will have inserted certain non-printing characters. The form feed character will be inserted when the program asks for a new page, and the carriage return when underlining is requested.

(3) Non-printing characters in imported files

In files imported via tape or microcomputer, the occurrence of non-printing characters is common. This is usually because:

a. the Internal Character Set of the remote system is not based on ISO but on another system such as Extended Binary Coded Decimal Interchange Code (EBCDIC);

b. on a micro, a word processing package may have been used on the file, so inserting control characters or creating characters with codes above 127;

c. the parity of the remote system or micro may differ from EMAS, or it may have been set wrongly by the method used to import the file. This will result in the most significant bit of some or every byte being set (see Appendix IV headed *The most significant bit* on page 24 for more details).

4. How to find them

These characters may appear anywhere in your file. If you want a report giving details of what they are and where they are situated, you should use the NOPRINT command as described in section 6 on page 7.

5. How to remove them

a. To remove all non-printing characters - use DEGROT (page 8), DETAB (page 9) or FIXCODE (page 8),

b. To change the character code system (e.g. EBCDIC to ISO) - use CODECHANGE (page 10),

c. To correct characters that have had the most significant bit set - use FIXCODE (page 8),

d. To remove all non-printing characters but preserve tabulation settings created by the horizontal tab character - use DETAB (page 9).

6. The NOPRINT Command

This command looks at every character in a file. If it finds a non-printing one, it displays its location and HEX value.

The only non-printing character not picked up by this command is line feed (decimal 10, X0A), which is a vital component of all text files, and is therefore valid.

The NOPRINT command takes one parameter which is the name of the file you want to check. Here is an example of how you could issue the command and the results it would produce.

Command: **NOPRINT TEST**
Line 1 Character 1 is value X0C
Line 1 Character 13 is value X09

File TEST contains 2 non-printing characters

Note that the form feed character (12, X0C) is part of line 1 and the horizontal tab character (9, X09) is on the same line. The values given are in HEX; to see which characters they represent, see the table in Appendix I.

7. The COUNT Command

You can use this command to check a file for the occurrence of any particular character. It takes two parameters:

a. the name of the file to be checked;

b. the code of the character to be searched for.

You could check the file TEST for the occurrence of the form feed character like this:

Command: **COUNT TEST,12**

where 12 is the decimal code for the form feed character.

The above command would produce the following output:

Finished! - 'TEST' contains 18 characters -
1 of them have the value 12!

Note that COUNT uses decimal code values and not HEX.

The COUNT command is not yet available on EMAS-A.

8. The DEGROT Command

This command can be used to remove all non-printing characters (except line feed) from a text file. It does not delete the characters, but makes a copy of the file with the non-printing characters omitted.

The DEGROT command takes two parameters:

a. the name of the file which contains the non-printing characters;

b. the name of the new file which will have the non-printing characters omitted.

The following example shows how you would strip the file TEST of non-printing characters:

Command: **DEGROT TEST,TESTNEW**
No of non-printing characters removed from EKLD91.TEST = 2

Note that the file TEST would remain unchanged, but the file TESTNEW would be a copy of TEST with the two non-printing characters omitted.

9. The FIXCODE Command

This program was written for the purpose of handling characters that have had the *most significant bit* of each or some of its *bytes* set. This can happen when a file is imported via a magnetic tape or micro computer. See section 3(3) on page 5 for more details.

If you do not understand the terms *most significant bit* and *byte*, you should refer to Appendix III and IV.

The FIXCODE command unsets the most significant bit from every byte in a file so that every character in the file will be represented by a byte with a value in the range 0 - 127 or X00 - X7F. If unsetting the bit turns the character into a control character i.e. gives it a value in the range 0-31 or 127 (X00-X1F or X7F), then the character will be removed. So like DEGROT, FIXCODE can be used to remove non-printing characters. The only difference is that FIXCODE will try to convert any non-printing characters into printing ones.

The FIXCODE command takes two parameters:

a. the name of the file which contains the non-printing characters;

b. the name of the new file which will be similar to the first, but with the non-printing characters amended or removed.

To give you an example of what FIXCODE does you could use it with the file CONLIB.EXAMPLES_CTRLCHAR2 which is accessible on EMAS and BUSH (the same file is available on EMAS-A where it is called ERCLIB:EXAMPLES_CTRLCHAR2). This file contains four characters, all with their most significant bits set. The HEX values of these characters are XCD, XC5, X88 and X8A.

Before using this file you would copy it into your own process like this:

Command: **COPY CONLIB.EXAMPLES_CTRLCHAR2,SET**

To unset the most significant bit from every character in this file, you would issue the following command:

Command: **FIXCODE SET,UNSET**

which would produce the following output on your terminal:

Characters read = 4
Control characters deleted = 1
Valid characters written = 3
Invalid characters amended = 4

The file UNSET created by the FIXCODE command would contain the characters with their most significant bits unset. The HEX values of these characters would be X4D, X45 and X0A, which represent the ISO characters M, E and line feed.

The character with HEX code value X88 reverted to X08 when the most significant bit was unset. This code represents the back space which is a control character. For this reason it was omitted from the output file.

10. The DETAB Command

You would use this command to remove all non-printing characters (except line feed) from a file, and replace any horizontal tab characters with the number of spaces needed to move the cursor or carriage to the next tab stop. The original file is not changed, but an edited copy is made.

As an example, take the file TEST, the contents of which are described in the Introduction. You could put it through the DETAB program like this:

Command: **DETAB TEST,TESTAB2**

DETAB detabbing program, version 03 (24th May 1985)
Finished! - 1 lines read.
1 TAB characters were replaced with spaces.
The following miscellaneous control characters have been removed:
1 FFs (code value 12)

The file TESTAB2 created by this example would have the form feed and horizontal tab characters removed and the word HELP moved from column 14 to 17.

In the above example the horizontal tab character was replaced by 3 spaces to move HELP to the next tab stop. By default, these tab stops are set every 8 columns, at 1, 9, 17, 25 etc.

You can change this default by giving a number, representing the distance between tabs, as the third parameter. For example:

Command: **DETAB TEST,TESTAB2,7**

would set the tab stops to 1, 8, 15, 22 etc. and would move the text HELP along to column 15.

Note that this command is not yet available on EMAS-A.

11. The CODECHANGE Command

This command converts the whole of a file from one character set to another. It takes the following two parameters (which can be typed on the same line as the command or issued after prompts):

a. the name of the file you want to change;

b. the name of the character code set you want to change it to.

At present, the valid values for the second parameter are EBCDICTOISO, ISOTOEBCDIC and CDC6600TOISO. Note that this command converts the contents of your file, it does not create a new one. If you want to preserve the contents of your file you should make a copy of it before converting it.

Note that any non-printing characters contained in your file will be converted to the equivalent non-printing characters in the new character set.

As an example you could convert the characters in the file TEST from ISO to EBCDIC by issuing this command:

Command: **CODECHANGE TEST,ISOTOEBCDIC**

EBCDIC codes are not described in this Note, but details of them can be obtained from the ERCC Advisory service.

Note that this command is not yet available on EMAS-A.

12. The ADDNP Command

You can use this command to add the form feed character to the top and bottom of a file. If one is already in place at these positions then no action is taken.
The command could be issued like this:

Command: **ADDNP TEST**

The effect is to throw a page at the beginning and end of your file when you are listing it on your printer. This means that your file will begin on an new page and the closing banner will be moved over onto a new page.

EDITORS FOR THE EXPERTS

This section gives details on how you can use certain editors to handle non-printing characters. If you want to:

a. display the ISO values of non-printing characters as they occur in the file - use EM;

b. display the HEX values of non-printing characters as they occur in the file - use CHEF or EM;

c. search for non-printing characters - use EDIT, ECCE, BECCE or SUPERSNAP;

d. insert non-printing characters - use EDIT, ECCE, BECCE or SUPERSNAP;

e. delete non-printing characters - use EDIT, ECCE, BECCE or SUPERSNAP.

Many users will be familiar with the EMAS Subsystem Editor (EDIT) and the Edinburgh Compatible Context Editor (ECCE), both of which can handle non-printing characters. For more details see the section on EDIT (page 14) and the one on ECCE (page 18).

13. The EM Command

EM is an editor imported from the University of Kent at Canterbury. It is not yet available on EMAS-A, but is available on EMAS 2900.

You could use it to indicate the non-printing characters which occur in a part or the whole of a file. You have the option of displaying the ISO name or the HEX value of these characters.

To start the program you would type the EM command with the name of the file as its parameter. For example:

Command: **EM TEST**

The first thing the program does is to print the message *Editor* followed by the number of lines in the file, followed by the editor prompt (which is the > character). For example:

```
Editor
1
>
```

After this prompt, you should give:

a. the number of the first line you want to display (followed by a comma);

b. the number of the last line you want to display (to display to the end of a file, you would give an asterisk instead of the line number);

c. the letter L for the List command.

For example:

```
> 1,1L
```

would display the first line of the file which would look like this:

```
{ff}          {tab}HELP{nl}
>
```

In the above example, the non-printing characters are indicated by letters representing their ISO names, for example: {tab} represents the horizontal tab (HT) character and {nl} represents the line feed (LF) character. The most common of these characters are described in Appendix V.

If you wanted to display their HEX codes instead of their ISO names you would add the letter H to the command line. For example:

```
> 1,1LH
```

would produce the following output

```
{0C}          {09}HELP{0A}
>
```

To find out which characters these HEX codes represent, see the table in Appendix I.

You would exit from the program by typing the letter Q on its own, for example:

```
> Q
```

A document describing EM (Ref K2.7/1) is available from the ERCC Advisory service.

14. The CHEF Command

CHEF is an editor imported from the University of Kent at Canterbury. It is not yet available on EMAS-A, but can be accessed on EMAS 2900.

CHEF can be used to indicate the non-printing characters which occur in a section or all of a file. It displays these characters in HEX and precedes them with the # character.

You would start the program by issuing the CHEF command with the name of the file as its parameter, for example:

Command: **CHEF TEST**

CHEF will then: display the number of characters in the file; give details of available help information; display the CHEF editor prompt and then ring the bell of your terminal (if it has one). For example:

18
Enter H for help (Q for quit)
>

You should reply with a command consisting of four items in the following order:

a. the number of the first line you want to display (followed by a comma);

b. the number of the last line you want to display (to display to the end of the file, you would give the full stop symbol . instead of a line number);

c. the letter P for the print command;

d. the letter L to tell the editor to print in lucid mode, which means that non-printing characters will be printed in HEX and preceded by the # character.

For example:

> 1,1PL

would produce the following:

 # 0C # 09HELP
>

See the table in Appendix I to find out which characters these HEX codes represent.

To exit from CHEF you would issue the Q command after the prompt like this:

> Q

The line feed character (decimal 10, X0A) is not treated as a non-printing character by CHEF.

A Kent User Note describing CHEF (Ref K2.7/3) is available from the ERCC Advisory service.

15. The EMAS Subsystem Editor (EDIT)

This editor is described fully in chapter 8 of the EMAS 2900: User's Guide and in summary on the EMAS Subsystem Editor card, both of which are available from the ERCC Advisory service.

You can use EDIT with the partitioned file CONLIB.CTRLCHAR to handle any control character which may occur in your file. This file is also available on EMAS-A where it is called ERCLIB:CTRLCHAR.

This file has 66 members, each member contains one single non-printing character. Each control character is contained in two members, one with a decimal name (prefix C) and one with a HEX name (prefix X).

For example, the file named CONLIB.CTRLCHAR_C9 contains the horizontal tab character. The number after the letter C corresponds to the decimal code of the character, so the BEL character has the decimal code of 7, which will be held in a file called CONLIB.CTRLCHAR_C7.

Those member files which begin with the letter X correspond to the HEX values of the EMAS control characters. An example of such a file is the one containing the form feed character. The HEX code for this character is X0C, so the file CONLIB.CTRLCHAR_X0C would hold this character. Note that the file CONLIB.CTRLCHAR_C12 would also contain this character. You can see the names of all the members if you give the following command:

Command: **ANALYSE CONLIB.CTRLCHAR**

(1) Finding non-printing characters using EDIT

This is done by issuing the Move command with the file containing the non-printing character as its parameter. The name of the file should be enclosed in angle brackets. For example, if you wanted to search for the first occurrence of the horizontal tab character (decimal 9, X09), you would issue the following command:

Edit: **M<CONLIB.CTRLCHAR_X09>**

If this was issued when editing your copy of the example file CONLIB.EXAMPLES_CTRLCHAR1, the EDIT cursor (indicated by the ^ character) would be moved to the first occurrence of the horizontal tab character, and the following output would be printed on your terminal:

```
*T*
        ^ HELP
*B*
```

(2) Inserting non-printing characters using EDIT

The Edit command Insert is used, with the file containing the characters as its parameter. For example:

Edit: **I<CONLIB.CTRLCHAR_X0C>**

would insert a form feed character at the position of the EDIT cursor.

(3) Removing non-printing characters using EDIT

The EDIT command Remove is used in this situation again with the file containing the character as its parameter. For example:

Edit: **R<CONLIB.CTRLCHAR_X0A>**

would remove the first occurrence of the line feed character from the file. Note that the following command:

Edit: **(R<CONLIB.CTRLCHAR_X0D>)***

would remove every occurrence of the carriage return character from your file.

The EDIT commands Delete, After, Uplift and Print can also be used with one of these files as its parameter.

16. The BECCE Command

BECCE uses the same editing commands as ECCE, but prints the file on your terminal in character codes (HEX or decimal) - *not* the characters themselves. You thus edit entirely in HEX or decimal. This section shows how BECCE can be used to:

- display the contents of a file in HEX;
- search for a particular non-printing character;
- insert, delete or re-locate a non-printing character.

(1) Displaying the contents of a file in HEX

To examine the contents of the example file **TEST** you would follow these steps:

a. Type the BECCE command followed by the name of the file you want to display. For example:

Command: **BECCE TEST**

The program will prompt you first for the character you have chosen to indicate the end of each record in the file, then the length of each record. For the purpose of displaying non-printing characters, you may select the default values by pressing the return key after the two prompts. The prompts will look like this:

Separator:
Record length(20):

The above is an example of what would be displayed if you were using BECCE on EMAS-A; on EMAS 2900 the record length value would not be displayed. By default the separator byte is given the value 255 and the record length is set to 20 bytes.

b. The program will print out the separator byte value and then prompt you with the > character, to which you should reply with a P for print, followed by the number of lines you want to see (* for all of them). For example:

Separator byte = 255
> P*

This will produce the following output on your terminal:

0C 20 20 20 20 20 20 20 20 20 20 20 09 48 45 4C 50 0A
END

c. The command to terminate BECCE is *%C*, which you should type after the prompt thus:

> %C

As you can see, the HEX values of the file are displayed. If you check them with the table in Appendix I, you will find that the file contains:

- a form feed character (decimal 12, X0C);
- 11 spaces (decimal 32, X20);
- a horizontal tab character (decimal 9, X09);
- the letters H, E, L, and P (HEX codes X48, X45, X4C and X50);
- a line feed character (decimal 10, X0A).

(2) Finding and highlighting non-printing characters

If you wanted to search the file for the occurrence of a particular non-printing character (line feed for example), you would use the Find command, with the HEX code of the character you are looking for as its parameter. For example:

Command: **BECCE TEST**
Separator
Record Length(20):
> F/0A/

The BECCE program in return would move the file pointer to the line feed character. To display this pointer you would have to use the P command on its own like this:

> P

This would produce the following output on your terminal:

0C 20 20 20 20 20 20 20 20 20 20 20 09 48 45 4C 50^ 0A
>

The file pointer is represented by the ^ character.

(3) Deleting and inserting non-printing characters

Suppose you wanted to change the file TEST, so that the form feed in column 1 is removed and the line feed is replaced by a new form feed. To do so you could use BECCE like this:

a. move the pointer to the top of the file by using the **Move** command;

```
> M-0
```

b. delete the form feed character from the top of the file by using the **Delete** command;

```
> D/0C/
```

c. move to the line feed character at the bottom of the file by using the **Find** command and delete it by using the **Delete** command;

```
> F/0A/D/0A/
```

d. insert a new form feed character by using the **Insert** command.

```
> I/0C/
```

e. If you wanted to show the edited contents of the file, you would do so like this:

```
> p1
 20 20 20 20 20 20 20 20 20 20 20 09 48 45 4C 50 0C^
>
```

(4) Notes on BECCE

If you wish to input text in character form rather than HEX, give the following command to BECCE:

```
> %I=C
```

The forms:

```
> %I=D
> %I=O
```

are also provided, specifying Decimal and Octal input respectively. The command

```
> %O=HC
```

causes output to be displayed in *both* HEX and Character form. You can get an on-line description of BECCE by typing HELP BECCE at command level. A short Note describing BECCE is available from the ERCC Advisory service. The BECCE program is static; new facilities added to improve ECCE are not added to BECCE.

17. The Edinburgh Compatible Context Editor (ECCE)

With ECCE you can:

- insert non-printing characters by using the Insert command;
- search for them by using the Find command;
- remove them by using the Destroy command;
- indicate where they occur in your output by using the %D command.

To start the ECCE program you would issue the ECCE command with the name of the file you want to edit as its parameter. For example:

Command: **ECCE TEST**

This command would generate the following response on your terminal:

Edit
>

the *>* character being the ECCE prompt.

(1) Inserting non-printing characters

You can insert non-printing characters through ECCE by issuing the Insert command and enclosing the HEX or CTRL codes between two ampersand (&) characters. Note that if the file you are editing is empty or the pointer is at the end, then any Insert command you issue will fail. The pointer is at the end of the file when the following message is output:

end
>

To get round this empty file problem you should insert a blank line then move back to the top of the file. The sequence of ECCE commands to do this would be:

> B
> M-0

Inserting HEX codes

If you wanted to insert the form feed character into a text file, you would use the following ECCE command:

> I&0C&

When inserting HEX codes, please note that:

- digits consist of one of 0-9, A-F, a-f;
- codes are made up of two digits;
- if only one digit is input, then a leading zero is assumed.

Inserting CTRL codes

Instead of specifying HEX codes, it is possible to specify the desired code using the up-arrow character (representing the CTRL key) followed by a letter. For example, if you knew that the form feed character was generated by holding down the CTRL key and pressing the L key, you could insert the form feed character like this:

```
> I&^L&
```

Note that using the CTRL key with the letter L when in ECCE will have no effect - the above form must be used instead.

More than one CTRL or HEX code can be inserted in one Insert command. For example:

```
> I&^L0C0A^I&
```

A description of the use of the CTRL key and the characters generated by it is contained in Appendix V.

(2) Searching for Non-printing characters

To find a particular pattern of non-printing characters you would use the Find command, with the HEX representation of the pattern enclosed in the ampersand character as its parameter. For example, to find the first occurrence of the horizontal tab character in the file CONLIB.EXAMPLES_CTRLCHAR1 (the contents of which are described in the Introduction), you would issue the following command:

```
> F&09&
```

This would move the file pointer to the first occurrence of the character and produce the following output on your terminal.

```
        ^ HELP
>
```

The file pointer is represented by the ^ character.

(3) Deleting non-printing characters

As with the Insert and Find commands, the Delete command takes the HEX representation of the character or characters to be removed from the file as its parameter (enclosed between ampersand characters). For example, to remove the line feed character from the example file, you would issue the following command:

```
> D&0A&
```

To remove every occurrence of the carriage return character from your file you would issue the following command:

```
> (D&0C&)*
```

(4) Controlling the output of non-printing characters

With the **%D** command you can control the output of non-printing characters on your terminal while using ECCE. The command takes a number as its parameter, thus:

%D=0	is the default value and allows non-printing characters to be sent to your terminal. What your terminal does with these characters depends on its make and how it is set up: most of them will have no effect on your terminal;
%D=1	replaces the non-printing characters with the ? character;
%D=2	replaces the non-printing characters with the ? character and also rings the bell of the terminal (if it has one).

You can save the value assigned to **%D** for future ECCE sessions by using the **%P** (profile) command.

The **%D** command can also be used in this way in the commands SHOW and RECAP.

(5) Notes on ECCE

A description of ECCE can be found by issuing the command HELP ECCE or by reading chapter 8 of the EMAS 2900: User's Guide.

18. The SUPERSNAP Command

SUPERSNAP is a file editor similar to BECCE. You could use it to:

- list all or part of a file in HEX so making it possible to pick out the non-printing characters;
- search through a file for the occurrence of non-printing characters;
- remove, alter or delete non-printing characters.

SUPERSNAP is fully described in User Note 36, which is available from the ERCC Advisory service.

I. EMAS: Internal Character Set

The table below shows the Internal Character Set used by the EMAS operating system. The first column gives the decimal code for each character, the second column gives the HEX value and the third displays the character itself.

0	00	NUL	32	20	space	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#(£)	67	43	C	99	63	c
4	04	EOT	36	24	$¤	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	Y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	-	127	7F	DEL

II. The Hexadecimal Numbering System

Because of the internal organization of computer storage, every character has to be represented by a HEX number.

The HEX system works to base 16 as distinct from the decimal system which works to base 10. This means that where decimal uses hundreds, tens and units which are powers of ten, HEX uses powers of 16 (i.e. 1, 16, 256 etc.).

The HEX system requires 16 digits and uses the numeric digits 0-9 supplemented by the letters A-F. The digits with their decimal equivalents are shown below:

Decimal	HEX
0	X00
1	X01
2	X02
3	X03
4	X04
5	X05
6	X06
7	X07
8	X08
9	X09
10	X0A
11	X0B
12	X0C
13	X0D
14	X0E
15	X0F

To convert HEX to decimal or vice versa, you could look up the values in the Internal Character Set table on the previous page.

You could also use the EMAS commands #HEX or #DEC, for example:

a. To convert the decimal number 23 to HEX, you would issue the following command:

Command: **#HEX 23**

X=00000017

Note that the leading zeros can be ignored, so decimal 23 is converted to X17 by the #HEX command.

b. To convert the HEX number X1C to decimal, you would issue the following command:

Command: **#DEC X1C**

N= 28

III. The binary representation of characters

The EMAS operating system stores characters in locations known as bytes: these bytes consist of 8 binary digits or bits. Every character in the Internal Character Set has a unique 8 bit pattern.

To illustrate this method of storing characters, take the letter A. It has a HEX value of X41 and would be represented in bits as follows:

```
HEX value     =  4    1
Bit pattern   =  0100 0001
```

As you can see, the first HEX digit (4) is coded into the 4 digit binary pattern of 0100 and the second HEX digit (1) is coded into the 4 digit binary pattern of 0001. The 8 bit pattern for the letter A (X41, decimal 65) is therefore 01000001.

IV. The most significant bit

The EMAS Internal Character Set codes range from 0 to X7F (decimal 127). These minimum and maximum codes have bit patterns of 00000000 and 01111111 respectively.

You will notice that the bit pattern for the maximum character code has all the bits set except for the left hand one. This left hand bit is known as the most significant bit.

As this bit is not set for any of the characters in the EMAS Internal Character Set, it can be ignored.

However, this bit can get set from 0 to 1 for a number of reasons (see the section headed *Non-printing characters in imported files* on page 5 for more details).

If this happened on some or all of the characters in a file, they will be given values above X7F (decimal 127) and so will not be accepted as EMAS characters, but will instead be treated as non-printing ones.

To restore these characters to their correct values, you should use the FIXCODE command as described in section 9 on page 8.

V. Control Characters

Characters in the table in Appendix I with decimal values 0 to 31 are non-printing characters, but because they each have a specific function, they are also known as control characters (the DEL character, decimal 127, is also a control character).

These characters can be generated from a terminal by typing a letter while holding down the CTRL key. Holding down the CTRL key subtracts the decimal value 64 from the character key you press. This means that by typing the letter J (decimal code value 74), while holding down the CTRL key (subtracting 64) you will generate the character with the decimal code value of 10 (the line feed character).

The most common control characters and their CTRL code equivalents are shown in the table below. These CTRL sequences may vary according to the type of terminal you are using, you should check the terminal handbook for specific details.

Decimal	ISO	CTRL & Key	Function
0	NUL	CTRL @	Used to accomplish time and media fill.
1	SOH	CTRL A	Start of heading - communications control character used at beginning of sequence of characters constituting machine-sensible address or routing information.
4	EOT	CTRL D	End of transmission - communications control character used to indicate conclusion of message transmission.
7	BEL	CTRL G	Bell - character used to cause an audible alarm at a remote terminal.
8	BS	CTRL H	Backspace - Format effector causing the cursor to move one space backward on same line.
9	HT	CTRL I	Horizontal Tabulation - Format effector causing the cursor to move to the next predetermined position along the line.
10	LF	CTRL J	Line Feed - Format effector causing the cursor to advance to the next line.
11	VT	CTRL K	Vertical Tabulation - Format effector causing movement of paper to first predetermined line on next form or page.
12	FF	CTRL L	Form Feed - Format effector causing the form or paper to be advanced one form (i.e. a new page is taken).
13	CR	CTRL M	Carriage Return - Format effector which causes the cursor to move to the left hand margin.
16	DLE	CTRL P	Data Link Escape - Communications control character which will change the meaning of a limited number of continuously following characters.
24	CAN	CTRL X	Cancel - control character used to indicate that the data with which it is sent is in error or is to be disregarded.
27	ESC	CTRL [	Escape - control character used to generate an interrupt.
127	DEL	none	Delete - character used primarily for time and media fill.

All the 33 non-printing or control characters are held in a partitioned file on EMAS 2900 called CONLIB.CTRLCHAR (ERCLIB:CTRLCHAR on EMAS-A). The contents of each member of this file consists of one single non-printing character.

Some of these member files can be used with the EMAS Subsystem Editor to manipulate non-printing characters (see section 15 on page 14 for more details).

Note that these non-printing character files can also be used with ECCE, CHEF and EM. For more details you should refer to the documents which describe these editors.

The partitioned file CONLIB.EXAMPLES on EMAS 2900 (ERCLIB:EXAMPLES on EMAS-A) contains the files CTRLCHAR1 and CTRLCHAR2 which are used as examples in this Note.