EXTERNAL REFERENCE SPECIFICATION

FOR

CYBIL FORMATTER V1.0

60461810 01

REVISION DEFINITION SHEET

REV	DATE	DESCRIPTION
A	04/07/78	Original.
B	05/19/78	Updated to reflect comments received through the DCS review.
C	05/08/81	Updated to reflect DAP S3591 and miscelleanous corrections.
D	07/31/81	Updated to reflect comments received through the DCS review.
E	10/15/82	Updated to reflect usage on multiple machines.
1	05/23/84	Updated to reflect productization activity.

1.0 PREFACE

The CYBIL source code formatter described in this document is based on past experiences developing and using comparable formatters.

1.1 SCOPE OF DOCUMENT

This document is intended to contain information necessary to use the CYBIL formatter in an interactive or batch job.

It is assumed that the reader is familiar with the CYBIL language.

1.2 APPLICABLE DOCUMENTS

CYBIL Reference Manual (60455280)

CYBIL Language Specification (60457280)

SES User Handbook (60457250)

2.0 INTRODUCTION

The CYBIL formatter is a utility used to format CYBIL source code to maximize readability. It executes as a standalone product.

2.1 PURPOSE

The formatter provides a common tool for formatting CYBIL source code prior to compilation. CYBIL programs formatted by a common tool will be more consistent, readable and maintainable.

2.2 OVERVIEW

The formatter executes under multiple machine and operating system environments. Input and output files for the formatter are specified using appropriate operating system command parameters. Formatting may be controlled to a limited extent by the use of CYBIL pragmats.

3.0 CONVENTIONS

The formatting conventions are mostly hard coded into the formatter with some conventions being alterable by the use of pragmats.

3.1 INPUT FILE CONVENTIONS

1) Source must be normal CYBIL input files.

2) Source must be legitimate CYBIL source, but is not necessarily a compilation unit. Common decks are formattable.

3) Source file is assumed positioned and local.

4) Multiple records on source file are formatted.

5) Source line may be any length as formatter reads to end-of-line.

3.2 FORMATTED OUTPUT FILE CONVENTIONS

1) All lines will be 79 characters or less unless altered by a pragmat.

2) The character set output by the formatter will be identical to the input file character set.

3) The formatted output file is created as a local file and is not rewound before or after.

3.3 FORMATTING CONVENTIONS

1) All keywords will be converted to upper case. The predefined data types like integer, char, cell, boolean, string, array, record, and set will always be in lower case.

2) All programmer created identifiers will be converted to lower case.

3) The case of strings and comments will not be altered.
4) Comments starting at the left hand margin will not be

formatted.

5) Lines with the asterisk control character in column one are not formatted. These lines are assumed to be maintenance package control statements which will be satisfied before compilation.

6) Blank or empty lines are retained.

7) Blanks are squeezed to 1 except within strings and comments.

8) A ; (semicolon) will cause a new line to be started. If a trailing comment follows on that same line, it will remain there.

9) A space will be added before and/or after certain delimiters. Such as: [and] on attribute lists, the operators, etc.

10) Keywords that start or terminate a structured statement will start a new line.

11) Keywords that form a structured statement will cause following statements to have a margin of 2 greater than the line that contains the keyword. Keywords that terminate a structured statement will decrease this margin by 2. The maximum margin will be 40.

12) If there is no blank line before a VAR, TYPE, CONST, SECTION, or PROCEDURE declaration, or label, one will be put there. A blank line will separate a local variable declaration from the procedure statement list.

13) Each parameter declaration will start a new line when they appear within a PROCEDURE declaration. A value attribute will have a margin six spaces greater than the margin for the PROCEDURE statement. The maximum margin will be 40.

14) Identifiers defined in TYPE, VAR, CONST and SECTION declarations will each start on a new line.

15) Labels are backed up 2 columns from current margin and are alone on a line.

16) If comments need to be separated across lines, they will be separated at a blank and subsequent lines will be lined up with the original line.

17) If comments starting at the left hand margin are too long,

they are continued at the left hand margin of succeeding lines until the comment is completed.

18) The left hand column defaults to 1 unless altered by a pragmat. The starting margin is equal to the left hand column.

    NOTE: The first input line is given a margin of 1.

19) Pragmats that are processed by the formatter take effect after their appearance in the source.

20) The formatter checks column one of the input stream and if it is non-numeric or the asterisk control character, the information is assumed to be text.

21) The formatter stops execution and outputs an appropriate message upon detecting the first syntax error. Under NOS, the error flag job control register is set to 13B (forced error).

22) The THEN clause of the IF statement will normally be on the same line as the IF, space permitting.

23) The ELSE and IFEND clauses are aligned with the IF statement.

24) CASE selection specs will have the same margin as their associated CASE statement.

25) Statements that exceed the length of a line will be continued on the following line with a 6 character indentation.

26) Long strings that do not fit on a single line will be split and the CAT operator generated.

27) A label will be filled in on ending delimiters of labeled structured statements, and exit, cycle, procend, modend and funcend statements.

3.4 POTENTIAL FUTURE ENHANCEMENTS

The following list of formatting features are not done by the current formatter, but could be considered for a later version of the CYBIL formatter. The list is a composite of suggestions from CYBIL users.

3.0 CONVENTIONS
3.4 POTENTIAL FUTURE ENHANCEMENTS

- o Verifying that if the 4th character of a name is a $ sign, then the 3rd character must conform to the System Interface Standard.

- o Recognition and justification of comments within a comment block.

- o Develop a method of highlighting changes in flow of execution caused by the EXIT, CYCLE and RETURN statements.

- o Investigate readability of tabbing the ':' in a type, variable, field or parameter specification to a fixed position.

- o Enhanced interface to SCU.

- o Precede a comment block with 2 blank lines and follow it with 1 blank line. A comment block is 1 or more lines of comments.

- o Introduce a pragmat for flexible indentation.

- o Prevent movement of text to a previous line.

4.0 PRAGMATS

Pragmats appearing in the source are used to control the CYBIL formatter and will be copied to output to provide a graceful means of reformatting.

4.1 CYBIL PRAGMATS

These CYBIL pragmats control the CYBIL compiler as well as the formatter and are treated similarily.

4.1.1 ??RIGHT := N??

For the CYBIL formatter, this pragmat is used to specify maximum output line length with n >= 72 and <= 110. The default value is 79. This pragmat indicates to the CYBIL compiler the maximum length of a source line.

4.1.2 ??LEFT := N??

Used to make the CYBIL compiler and formatter ignore any foreign data outside the CYBIL syntax. For the formatter, this pragmat indicates left hand output column and starting margin. The default value is 1. The foreign data which exists in the ignored column positions will be copied to the output file.

4.2 FORMATTER PRAGMAT

This pragmat controls only the CYBIL formatter and is not processed by the CYBIL compiler.

4.2.1 SYNTAX

<formatter pragmat statement> ::= ??<formatter pragmat>??

<formatter pragmat> ::= FMT (<toggle setting list>)

<toggle setting list> ::= <toggle setting> [,<toggle setting>]

<toggle setting> ::= <toggle identifier> := <condition>

<toggle identifier> ::= FORMAT

<condition> ::= ON | OFF

4.2.2 TOGGLE IDENTIFIERS

o FORMAT

Used to control formatting of all lines. Default is FORMAT := ON.

FORMAT := ON - All source lines are formatted.

FORMAT := OFF - No source lines are formatted.

5.0 USE OF FORMATTER UNDER C170 NOS

An SES procedure interface is available for accessing the CYBIL formatter and is described in SES's User Handbook (60457250).

5.1 SAMPLE EXECUTION ON NOS

```
GET,I=SOURCE.
SES.CYBFORM,I=SOURCE,O=COMPILE
REPLACE,O=COMPILE.
```

The file COMPILE will contain the formatted program found on file SOURCE.

6.0 USE OF THE FORMATTER ON OTHER OPERATING SYSTEMS

The following control statement is used:

```
CYBFORM I=filename O=filename
```

I: Specifies the name of the file to be formatted. If I is not specified, the name of the file is I.

O: Specifies the name for the formatted file. If O is not specified, the name of the file is O.

NOTE: I and O may not use the same file name.

7.0 EXAMPLES

7.1 SOURCE PROGRAM

1)
```
       ??RIGHT := 110??
MODULE M1;
PROCEDURE A1;
VAR I: INTEGER;
PROCEND A1;
MODEND M1;
```

Formatted is:

```
??RIGHT := 110??
MODULE m1;

  PROCEDURE a1;

    VAR
      i: integer;

  PROCEND a1;
MODEND m1;
```

2)
```
       ??RIGHT := 72??
  MODULE M1;
    PROCEDURE A1;
      VAR
        I: INTEGER;
    CONST J=5;
    PROCEND ;
  MODEND ;
```

Formatted is:

```
??RIGHT := 72??
MODULE m1;

  PROCEDURE a1;

    VAR
      i: integer;

    CONST
      j=5;

  PROCEND a1;
```

```
       MODEND m1;
```

7.2 MESSAGES

Messages are written to the OUTPUT file and the dayfile. The source line that caused the error message will normally be the last one on the output file.

1) ***UNBALANCED BLOCK STRUCTURE***

   Begin and end statements for structured statements do not match.

2) ***IMPROPER PARAMETER LIST***

   Something is wrong with a procedure definition statement.

3) ***EXTRANEOUS RIGHT PARENTHESIS***

   Right and left parens do not match.

4) ***MISSING RIGHT PARENTHESIS***

   Right and left parens do not match.

5) ***IMPROPER CASE LABEL***

   Improper case statement.

6) ***IMPROPER FORMAT TOGGLE***

   Something is wrong with a format pragmat that the CYBIL formatter processes.

7) ***IMPROPER HEX CONSTANT***

   Something is wrong with a hex constant.

8) ***IDENTIFIER TOO LONG***

   Have an identifier greater than 31 characters.

9) ***IMPROPER STRING CONSTANT***

   A quote is missing at EOL causing a syntactic error.

10) ***PRAGMAT STRING TOO LONG FOR LINE***

The string in the titling pragmat does not fit on the line and cannot be continued.

11) ***ILLEGAL CONTROL CARD KEYWORD***

A keyword other than I or O was found on the control card.

Table of Contents