General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

JPL PUBLICATION 77-24, REVISION 1, ADDENDUM 1

Software Design and Documentation Language

User's Guide for SDDL Release 4

T. M. Zepko

(NASA-CR-165025) SOFTWARE DESIGN AND DOCUMENTATION LANGUAGE: USER'S GUIDE FOR SDDL RELEASE 4 (Jet Propulsion Lab.) 13 p
HC A02/MF A01 CSCL 09B

N82-13767

Unclas
G3/61 08460

September 15, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 77-24, REVISION 1, ADDENDUM 1

Software Design and Documentation Language

User's Guide for SDDL Release 4

T. M. Zepko

September 15, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

ABSTRACT

This User's Guide is an addendum to JPL Publication 77-24, Revision 1, Software Design and Documentation Language, by Henry Kleine. It is not a complete description of SDDL, but rather a description of the changes introduced in SDDL Release 4 (the Pascal implementation). These changes include a number of new capabilities, plus some changes to make the language more consistent and easier to use. Incompatibilities with earlier versions are limited to certain of the directive statements.

INTRODUCTION

SDDL, first implemented in the SIMSCRIPT language, has been available only on computers which support SIMSCRIPT, all large machines. By the summer of 1979 the need for an SDDL processor which could be used on a wider range of computers, including minis and micros, led to a decision to build a new implementation. Pascal was chosen as the new implementation language because of its attractiveness as a high-level language and because of its immediate or anticipated availability on a wide variety of machines. It was felt that by restricting the code to standard Pascal as defined in Jensen and Wirth, Pascal User Manual and Report, portability problems would be kept to a minimum.

A new implementation of SDDL was also seen as an opportunity to correct several deficiencies and ambiguities in the language and to add a number of new capabilities. Each change introduced was evaluated with a view towards simplifying the language and making it more consistent and more easily understood. However, care was taken to maintain upward compatibility with earlier versions wherever possible.

SDDL Release 4 (January 1981) is the result of this implementation effort. Although it contains a number of changes, its only incompatibilities with earlier versions are limited to certain of the directive statements. Therefore, with only a few minor changes to these statements, any existing SDDL source file can be correctly processed by the new version.

This document describes the language changes and additional features of the new processor. It is an addendum to JPL Publication 77-24, Revision 1, Software Design and Documentation Language, and represents a preliminary step to rewriting that document.

CHANGES INTRODUCED IN SDDL RELEASE 4

(Those changes which violate strict upward compatibility are marked with an *)

1) Lower case letters are now recognized and treated as distinct from their upper case counterparts. Thus, for example, ABC, abc, AbC, and aBc are all distinct identifiers. Directives are only recognized in upper case.

2) A #KEYWORDS directive has been added to give the user a choice between several sets of built-in keywords. Note that this directive nulls out all currently defined keywords before substituting the selected set. The directive exists in four forms:

a. #KEYWORDS

#KEYWORDS STANDARD

Nulls out all currently defined keywords and establishes the standard (default) set:

	Initiator	Terminator	Escape	Substructure
Module Structures	PROGRAM	ENDPROGRAM	EXITPROGRAM	
	PROCEDURE	ENDPROCEDURE	EXITPROCEDURE	
Block Structures	FIRST			NEXT
	IF	ENDIF		ELSE ELSEIF
	SELECT	ENDSELECT		CASE
	LOOP	ENDLOOP REPEAT	EXITLOOP CYCLE	

Module Invocations	CALL DO

b. #KEYWORDS PASCAL

Nulls out all currently defined keywords, defines " as a mark character, and establishes the following:

	Initiator	Terminator	Escape	Substructure
Module Structures	PROGRAM	"ENDPROGRAM"		
	PROCEDURE	"ENDPROCEDURE"		
	FUNCTION	"ENDFUNCTION"		
	"DECLARATIONS"	"ENDDECLARATIONS"		
Block Structures	BEGIN	END		
	RECORD	END"RECORD"		
	IF	"ENDIF"		ELSE
	CASE	END"CASE"		
	WITH	"ENDWITH"		
	REPEAT	UNTIL		
	WHILE	"ENDWHILE"		
	FOR	"ENDFOR"		

Module Invocation	"CALL"

c. #KEYWORDS FORTRAN

Nulls out all currently defined keywords and establishes the following:

	Initiator	Terminator	Escape	Substructure
Module Structures	SUBROUTINE	END	RETURN	ENTRY
	PROCEDURE	ENDPROCEDURE	EXITPROCEDURE	
Block Structure	DO	CONTINUE		

Module Invocations	CALL GO

d. #KEYWORDS NONE

Nulls out all the currently defined keywords.

3) The names of the directives for defining keywords have been changed as follows:

#DEFINE	MODULE	has been changed to	#MODULE
#DEFINE	BLOCK	has been changed to	#BLOCK
#DEFINE	CALL	has been changed to	#CALL
#DEFINE	NULL	has been changed to	#NULL

(For the sake of upward compatibility, the old names are still recognized, but their continued use is discouraged.)

* 4) The syntax of these keyword definition directives has been changed to allow synonyms for any structure component including the initiator. Specifically, punctuation is required to separate the structure components (i.e., Initiator, Terminator, Escape, Substructure). Keywords not separated by punctuation are taken as synonyms for the same component.

* 5) Definition of a keyword supercedes all previous definitions of that keyword. All components of a given structure must be defined in the same directive. Specifically, this means that the old method of defining synonyms for terminators, escapes, and substructures by naming the initiator in multiple directives will no longer work. Synonyms must be defined in the manner described in Item 4.

6) The definition of terminator keywords has a new interpretation. If one or more terminators are defined for a given structure, then the first of these becomes the "default terminator", which is printed when the processor must force a structure termination (instead of ENDinitiator). If no terminator is defined for a given structure, or if the specified default terminator has been nulled or redefined, then the structure will be closed automatically when necessary, but without printing a terminator statement.

* 7) The #NULL directive applies only to keywords, not to character types. The old practice of NULLing mark and string characters will no longer work. Instead, use the directives for defining character types. (See Item 8).

* 8) The set of directives for defining character types has been expanded and modified. Characters other than blanks, letters (upper and lower case plus #), and digits are initially of type punctuation, but may be interchangeably defined by the user as type mark, string, comment, or punctuation with the directives:

```
#MARK          character list      title
#STRING        character list      title
#COMMENT       character list
#PUNCTUATION   character list
```

The character list may contain any character currently of type mark, string, comment, or punctuation. The new definition supercedes all previous definitions. Also, note that the syntax of the #MARK and #STRING directives has been changed. The list of characters now comes first, followed by the title of the cross-reference (if any) with which they are to be associated. The title begins with the first letter or digit encountered and continues to the last nonblank character of the statement. Finally, note that the #MARK directive now allows specification of at most one title. Use separate directives to associate marks with different cross-reference listings.

9) An #IDENTIFIER directive has been added to make it possible to get a cross-reference of identifiers which do not contain marks. The directive can take the forms:

a. #IDENTIFIER title

Causes all unmarked identifiers to be recorded for inclusion in the cross-reference with the given title.

b. #IDENTIFIER

Cancels the recording of unmarked identifiers. Further occurrences of those already recorded are correctly cross-referenced, but no additional unmarked identifiers are recorded.

(This directive is also recognized in its plural form: #IDENTIFIERS.)

* 10) There is no longer an untitled (Null) cross-reference listing. Strings, identifiers, and mark characters not associated with a particular cross-reference title are not cross-referenced.

11) Each token is now listed in the cross-reference listing(s) determined by the definitions in effect when the token is first encountered, even if those definitions are subsequently changed. As before, multiple directives referring to the same title result in a merged cross-reference listing.

12) When a character defined by the #COMMENT directive as a comment delimiter is found in the text, everything up to the next comment delimiter or the end of the statement is treated by the processor as a comment. Comments are printed in the output, but are otherwise ignored.

13) An #OPEN directive has been added so the user can initiate a block structure without a printed initiator statement. The directive has the forms:

a. #OPEN

Opens a new block with the current default indentation.

b. #OPEN n

Opens a new block indented by n columns.

14) The name of the #TERMINATE directive has been changed to #CLOSE in order to emphasize its complementary function to #OPEN. (For the sake of upward compatibility, the old name is still recognized, but its continued use is discouraged.) Note that as before, the optional integer in this directive specifies the number of blocks to close and is unrelated to the optional indentation amount in #OPEN. Also note that for several reasons #OPEN and #CLOSE do not have to be paired up. First, a single #CLOSE can terminate more than one block structure. Second, #CLOSE can be used to terminate blocks specified in the usual way with initiator statements. And third, since blocks specified by #OPEN have no default terminator, they can be closed automatically by the processor (See Item 6).

15) The name of the #WIDTH directive has been changed to #MARGIN and its capabilities extended so that both margins may now be specified. (For the sake of upward compatibility, the old name is still recognized, but its continued use is discouraged.) The directive can take the forms:

a. #MARGIN n

Sets the left margin to column 1 and the right margin to column n.

b. #MARGIN n m

#MARGIN n punctuation character m

Sets the left margin to column n and the right margin to column m.

c. #MARGIN

Restores the margins to their initial values, 1 and 80.

(This directive is also recognized in its plural form: #MARGINS.)

* 16) A #SPLIT directive has been added to allow the user to define a line split character and a character to be printed in its place. This directive is necessary if line splitting is to be used, since the feature is initially disabled. The directive can take the forms:

a. #SPLIT char

Sets the split character to char and the character to be printed in its place to blank.

b. #SPLIT char1 char2

Sets the split character to char1 and the character to be printed in its place to char2. Note that char1 and char2 may be the same or different characters and that any characters may be used in this context without affecting their usage elsewhere.

c. #SPLIT

Restores the initial condition of line splitting disabled.

17) A #LABEL directive has been added to allow specification of a "label field" at the beginning of each input line. These columns will be printed in the output but are otherwise ignored by the processor. The directive can take the forms:

a. #LABEL n

Sets the number of label columns to n (maximum of 15 columns).

b. #LABEL

Restores the number of label columns to the initial value of zero (labeling disabled).

* 18) The #SEQUENCE directive now requires two values, the first and last columns of the sequence number field. This allows greater flexibility in that the input lines can now be variable lengths. However, when sequence columns are defined, these will be the last columns read and any additional columns will be ignored. The directive now has the forms:

a. #SEQUENCE n m

#SEQUENCE n punctuation character m

Reserves columns n through m for a sequence number (maximum of 15 columns).

b. #SEQUENCE

Restores the initial condition of sequencing disabled.

* 19) The interpretation of the #SAMEPAGE directive has been changed slightly in order to make it a bit clearer and to allow nesting the ranges of multiple #SAMEPAGE directives. The directive can take the forms:

a. #SAMEPAGE n

Where n greater than 0, this means, "keep the next n modules together by suppressing the automatic page ejects between them." Note that the correct location for this directive is now before the first of the modules to be kept together instead of after this module. If a directive in this form specifies a range that lies entirely within the range of an earlier #SAMEPAGE directive, the new directive will have no effect.

b. #SAMEPAGE 0

#SAMEPAGE

Restores the initial condition of automatically ejecting a page between modules, thus cancelling the effect of any samepage range that is still open.

Thus, it is possible to bracket a group of modules to be kept together by specifying a very large range and later cancelling it. Smaller, explicit ranges nested within a larger range will no longer affect the larger range.

* 20) The input for #TEXT and #TITLE boxes must now be terminated with an explicit #END, rather than any line beginning with a # character. The number of lines which may be included in either kind of box is limited to what will fit on a single page of the output. Title boxes are always printed on a page by themselves; text boxes are printed on the current page if there is sufficient room, on the next page otherwise.

21) A special #SUPPRESS directive has been added so that certain processor functions may be disabled on a given run. This directive is unique in that it is only recognized if it appears as the first statement in the input. It has the form:

#SUPPRESS option list

The option list may specify any number of options, with or without separating punctuation. The possible options and their meanings are:

a. NONKEY - "Suppress nonkeyword (passive) statements." This results in an abbreviated source listing since nonkeyword statements are not printed, and in shorter cross-reference listings since these statements are not scanned for token occurrences. (Corresponds to the former D option.)

b. ERRORS - "Suppress error messages." This suppresses the printing of error messages and the flagging of processor supplied statements but still allows them to be counted correctly. (Corresponds to the former E option.)

c. TOC - "Suppress the Table of Contents." (Corresponds to the former T option.)

d. MODTREE - "Suppress the Module Invocation Tree." (Corresponds to the former R option.)

e. MODXREF - "Suppress the Module Cross-Reference Listing." (Corresponds to the former M option.)

f. GENXREF - "Suppress the general (user defined) cross-reference listings." This suppresses cross-references for which titles are defined in #MARK, #STRING, or #IDENTIFIER directives. Remember that no untitled cross-reference is printed anyway. (See item 10.)

In addition to the visible effects described above, most of these options will result in a reduction of memory requirements (often substantial) since no information is stored by the processor if it will not be needed later. For example, suppressing GENXREF will mean that most tokens and their lists of occurrences will not have to be stored.

22) A variation of the #SUPPRESS directive has been provided which will cause the directive itself to be bypassed. It has the form:

##SUPPRESS any text

Thus, by adding a # sign to the directive, its effect can be cancelled without physically removing it from the source file.