SPANEX™

Span Macros Manual

Span Software Consultants Limited

Version: 06.0

Product Number: SPOS-001

Revision: 1st March 2015

Manual Ref: SPZ-02-017

CONTENTS Page

1 Introduction 5
1.1 Changes for Release 6.0 6
1.2 Changes for Release 5.2 6
1.3 Changes for Release 5.1 6
1.4 Changes for Release 5.0 6
1.5 Changes for Release 4.6 7
1.6 Changes for Release 4.5 8

2 Span Macro Descriptions 9
2.1 Notation and Coding Conventions 9
2.2 #SPAMODE Macro - MVS/XA, ESA/390 and z/Architecture Addressing Mode Manipulation 10
2.3 #SPANCHK Macro - Data Verification Macro 13
2.4 #SPANLTG Macro - LTORG and Patch Area Generation 15
2.5 #SPANTRY Macro - Module or Subroutine Entry Point Macro 17
2.6 #SPANXIT Macro - Module or Subroutine Exit Point Macro 23
2.7 #SPBINSR Macro - Perform Binary Search of an Ordered Table 26
2.8 #SPCPTIM Macro - Execute Routine and Calculate CPU Usage 28
2.9 #SPGLBL Macro - Define SPOUTPUT/SPZPARSE Control Block 30
2.10 #SPMSG Macro - SPOUTPUT Parameter Formatting 31
2.11 #SPMVCL Macro - Simulate MVCL instruction 32
2.12 #SPTEST Macro - Check Program Environment 33
2.13 #SPXFIND Macro - Invoke SPANEX Lookup Functions 34
2.14 #SPXICB Macro - Generate SPANEX DSECTs 36
2.15 #SPXICBA Macro - Find Field in SPANEX ICB 37
2.16 #SPXMSG Macro - Internal Message Request to SPANEX 38
2.17 #SPXQ Macro - Locate SPANEX Internal Control Block 41
2.18 #SPXRSTU Macro - Request to SPANEX Utility 42
2.19 #SPXSVC Macro - Issue SPANEX SVC 47
2.20 #SPXUDDN Macro - Specify DDNAME for #SPXUMSG Message Requests 50
2.21 #SPXUMSG Macro - User Message Request to SPANEX 51
2.22 #SPZCLOS Macro - SPZQPAM Close Macro 53
2.23 #SPZCMD Macro - SPZPARSE Parameter Formatting 54
2.24 #SPZFIND Macro - Position to a Partitioned Dataset Member when using SPZQPAM 56
2.25 #SPZFLD Macro - SPOUTPUT Parameter Formatting 57
2.26 #SPZKWRD Macro - SPZPARSE Parameter Formatting 63
2.27 #SPZOPEN Macro - SPZQPAM Open Macro 67
2.28 #SPZPARS Macro - Invoke SPZPARSE Service Routine 68
2.29 #SPZPEND Macro - SPZPARSE Parameter Formatting 74
2.30 #SPZPMAP Macro - Generate SPZPARSE DSECTs 75
2.31 #SPZPOSN Macro - SPZPARSE Parameter Formatting 76
2.32 #SPZSOB Macro - SPOUTPUT Interface Macro 79
2.33 #SPZSUBP Macro - SPZPARSE Parameter Formatting 91
2.34 #SPZTITL Macro - SPOUTPUT Parameter Formatting 95

This page intentionally left blank.

1 Introduction

This manual describes all currently available Span Macros for IBM System/390 and z/Series operating systems.

These Span Macros are supported only for IBM System/390 or z/Series Assembler Language and assume the availability of one of the following IBM Assemblers or equivalent: MVS Assembler H or High-level Assembler.

Note that Span Macro names begin with the "#" symbol. This character may not appear on some 3270 keyboards. It has a hexadecimal representation of X'7B' in EBCDIC. If the symbol is not on your keyboard, use an IBM System/390 or z/OS Reference Card or manual to determine the symbol used by your keyboard for X'7B' in the EBCDIC character set.

Span Manuals in this Series	Order No
Span Macros Manual	SPZ-02
Span Service Routines Manual	SPZ-03

1.1 Changes for Release 6.0

New options of the LINKAGE parameter of #SPANTRY allow the specification of the storage location of dynamic save areas.

New CPRIGHT2 option of the #SPZFLD macro for a mixed-case copyright notice value.

New format option for SPZPARSE fields that allows format-checking of character name values (ie alphabetic first character, followed by alpha-numeric characters).

All modules can now be located above the 16-Megabyte line.

1.2 Changes for Release 5.2

The #SPZFLD macro MOD option now supports dynamic change of the field offset in the output line.

1.3 Changes for Release 5.1

The #SPAMODE macro now supports the new 64-bit addressing mode introduced with z/OS.

The new BASE=NONE option of the #SPANTRY macro allows assembler programming to be performed without module base registers. Branching within the module is performed by relative jump instructions. This feature is supported by the #SPANXIT and #SPAMODE macros, and by a limited number of IBM macros.

The new CLEAR=YES option of the #SPANXIT macro clears a dynamic save area to zeros before releasing the storage. This ensures that sensitive information cannot be retrieved in response to a subsequent request for storage.

The new SETCONT= option of the SPZPARSE #SPZKWRD macro extends the number of values supported for the SET= parameter of keyword definitions. More possible values can thus be defined for parsing keywords.

1.4 Changes for Release 5.0

SPZPARSE now supports TSO- or IDCAMS-style command syntax, as an alternative to the traditional “keyword=value” style. Command and parameter format is dynamically specified via a new parameter of the #SPZPARS macro.

The features provided in the #SPANTRY and #SPANXIT macros for MVS/ESA and OS/390 programming using the Linkage Stack may now be executed is ASC AR mode.

The #SPANTRY macro now allows dynamic save areas to be automatically set to binary zeros.

The new #SPZFIND macro is now required to be used instead of the standard FIND macro by users of the SPZQPAM service routine.

New DATE4 field type in the #SPZFLD macro supports for 4-digit year notation for Year 2000 compliance.

New SYSID field type in the #SPZFLD macro supports inclusion of the SMF System ID in program output.

New USERID field type in the #SPZFLD macro supports inclusion of the owning user ID in program output.

SPOUTPUT now supports the use of WTO tokens, to allow groups of console messages to be deleted with a single operating system DOM macro. This is implemented by means of the WTOTOKN option of the OPT= parameter of the #SPZSOB macreo, and by the WTOKEN= parameter of the #SPZSOB macro.

New version of the SOB used by SPOUTPUT eliminates all 3-byte addresses from the SPOUTPUT interface. All modules sharing a SOB must be assembled with the same version of the SPANEX macros, but SPOUTPUT supports existing modules compiled with all previous versions of the macros.

Minor editorial and technical changes have been made throughout this manual and the Span Service Routines manual.

1.5 Changes for Release 4.6

SPZSORT can now sort in-storage records up to 32767 bytes in length.

New features are provided in the #SPANTRY and #SPANXIT macros for MVS/ESA programming using the Linkage Stack.

The #SPANTRY macro now supports ESA-style module entry and exit linkage, and allows user-defined identification text to be added to a module entry point.

All invocation macros for Span Service Routine functions now support user programs running in AR mode on MVS/ESA systems.

New “Double-underline” feature is provided by SPOUTPUT formatting.

1.6 Changes for Release 4.5

The #SPAMODE macro now supports ARMODE switching for MVS/ESA systems.

The RUN=R4FSCR option of the #SPZSOB macro is now the default. Old-format full-screen output from SPOUTPUT is no longer supported for new programs. User programs assembled with Release 4.4 and earlier versions of the Span Macros can continue to use the earlier support - if these programs are assembled with Release 4.5 of the macros, the improved support will automatically be included.

The #SPZFLD macro now allows the screen high-lighting and colour attributes of fields on a 3270 screen to be modified dynamically.

SPOUTPUT now supports the BMARG= option of the #SPZSOB macro for full-screen output. This provides a “conditional end-of-screen” function to ensure that groups of output lines are kept together on the screen.

SPZDIRD now supports the reading of PDSE directories. Mixed concatenations of PDS and PDSE datasets is fully supported.

Support is now provided for versions of Span Software programs and user programs that are specific to the MVS/XA and MVS/ESA environments.

2 Span Macro Descriptions

2.1 Notation and Coding Conventions

Notation for Macros in this Manual

Square brackets, [ ], denote (1) that a macro parameter is optional: if an entire parameter with its options is enclosed in square brackets, then that parameter is optional; (2) that a range of values is permissible for a given parameter: if a series of possible values for a parameter is shown in a vertical manner, all surrounded by additional square brackets, then choose one from the values shown.

Normal parentheses, ( ), signify that parentheses should appear when the macro is coded, denoting, for example, a list of sub-parameters.

Underlining, ___, denotes default values for parameters.

Coding Conventions

Standard Assembler language coding conventions are used for all Span Macros:

- Labels must begin in column 1;
- Macro names and operands may be placed anywhere on the statement but are conventionally in columns 10 and 16 respectively;
- Continuations are indicated by a non-blank character in column 72 of the continued statement;
- Continuation statements must begin in column 16.

It is recommended that programmers using Span Macros do not define Global symbols having names beginning with "#" (Hash or Number sign). These are used extensively within the Span Macros. Additionally, the SPZPARSE macros define program labels beginning with "#".

Certain Span Macros support the "DOC=YES" parameter in addition to the parameters documented in this manual. This is to permit the generation of automated data areas manuals for some Span Program Products. Macros affected by this include: #SPXICB, #SPZFLD, #SPZPMAP, #SPZSOB, #SPZTITL.

2.2 #SPAMODE Macro - MVS/XA, ESA/390 and z/Architecture Addressing Mode Manipulation

The #SPAMODE macro is a general-purpose macro for setting and retrieving the Addressing Mode for MVS/XA, ESA/390 and z/OS systems. It supports modules running in any of 24-bit mode, 31-bit mode or 64-bit mode, and can generate re-entrant code and “portable” code for z/OS, ESA, XA and non-XA systems. The contents of Registers 1 and 15 may be destroyed by this macro.

format:

```
                            [ENTRY  ]
(name)    #SPAMODE  MODE=([BIT24  ], ...)
                            [BIT31  ]
                            [BIT64  ]
                            [INITIAL]

                    [        [SET  ]]
                    [,ARMODE=[      ]]
                    [        [RESET]]

                    [     [ONLY]]  [       [ONLY]]  [       [ONLY]]
                    [,XA=[     ]]  [,ESA=[     ]]  [,Z64=[     ]]
                    [     [TEST]]  [       [TEST]]  [       [TEST]]

                    [     [L          ]]
                    [,MF=[(E,laddr)]]
                    [     [(R,(reg))]]
```

where:

(name) - variable symbol name, max=8 bytes
- specifies the value of a label to be placed on the first executable instruction generated by the #SPAMODE macro.

MODE= - fixed keyword value
- specifies the function to be performed.
MODE=ENTRY is typically used at the entry point of a sub-module or utility routine, and specifies that the Addressing Mode is to be determined and stored for later use. If “MF=(E,...” or “MF=(R,...” is not specified, then the macro generates a word of storage in-line in which to save the Addressing Mode, and module will not be re-entrant.
MODE=BIT24 specifies that the Addressing Mode is to be changed to 24-bit mode.
MODE=BIT31 specifies that the Addressing Mode is to be changed to 31-bit mode.
MODE=BIT64 specifies that the Addressing Mode is to be changed to 64-bit mode.
MODE=INITIAL specifies that the Addressing Mode is to be changed back to the mode that was in effect when the #SPAMODE macro specifying the “MODE=ENTRY” option was executed. If the “MODE=ENTRY” form of the macro has not been previously executed, then the Addressing Mode after

executing the “MODE=INITIAL” form of the macro will be unpredictable.
Note that the mode sub-parameters can be combined; eg “MODE=(ENTRY,BIT24)” will save the current Addressing Mode and then switch to 24-bit mode.

ARMODE=
- fixed keyword value
- specifies a change to the ARMODE for ESA/390 or z/OS. Note that the MODE= parameter is ignored if the ARMODE= parameter is coded. The ESA=TEST option may be specified to ensure that modules using this option can be executed on non-ESA systems, but use of the ARMODE= parameter requires that the module be assembled on an MVS/ESA, OS/390 system or z/OS.
  ARMODE=SET is used to turn on Access Register mode for the calling program. This is a shorthand way of issuing the combination of machine instruction and assembler instruction required to switch into AR-mode.
  ARMODE=RESET is used to turn off Access Register mode for the calling program. This is a shorthand way of issuing the combination of machine instruction and assembler instruction required to switch out of AR-mode.

XA=
- fixed keyword value
- XA=TEST is used to generate “portable” code (code that can be executed without modification on MVS/XA systems and non-XA systems). The CVT mapping macro must be included in the assembly if the “XA=TEST” option is used. The default is “XA=ONLY” which generates code that will execute on MVS/XA systems only.

ESA=
- fixed keyword value
- ESA=TEST is used to generate “portable” code (code that can be executed without modification on ESA/390 systems and non-ESA systems). The CVT mapping macro must be included in the assembly if the “ESA=TEST” option is used. The “ESA=ONLY” option generates code that, if the ARMODE= option is used, will execute on MVS/ESA or OS/390 systems only.

Z64=
- fixed keyword value
- Z64=TEST is used to generate “portable” code (code that can be executed without modification on z/OS systems and non-z/OS systems). The CVT mapping macro must be included in the assembly if the “Z64=TEST” option is used. The “Z64=ONLY” option generates code using z/Architecture instructions, and will execute on z/OS systems only.

MF=

- fixed keyword value
- specifies the macro format ("List", "Execute" or "Register" form) to be generated. If this parameter is not specified, in-line coding is generated and the module will be non-re-entrant.
  <u>MF=L</u> specifies the "List" form of the macro, and generates a non-executable parameter list that is used by the "Execute" form of the macro.
  <u>MF=(E,laddr)</u> specifies the "Execute" form of the macro, with the second sub-parameter specifying the address (in label or register notation) of the "List" form parameter list to be used. Matching "MODE=ENTRY" and "MODE=INITIAL" macros must specify the same "List" form to ensure that the correct Addressing Mode is set.
  <u>MF=(R,(reg))</u> specifies the "Register" form of the macro, with the second sub-parameter specifying a register (in register notation, ie enclosed in parentheses), that is to be used to store the Addressing Mode. Register numbers from 1-15 are supported. The specified register will contain the Addressing Mode after a "MODE=ENTRY" form of the macro, and must contain the Addressing Mode before a "MODE=INITIAL" form of the macro is used.

2.3 #SPANCHK Macro - Data Verification Macro

The #SPANCHK macro provides coding to check the System Control Program that is being run, the level of the Span Product program that is being assembled, and other information that is required by the #SPANTRY macro which calls #SPANCHK internally.

format:

```
                      [(SCPVAL[,NOBASE])]
 #SPANCHK   TYPE=[SCPVAL          ]  [,SCP=scpname]
                      [PRODLEV         ]

            [,ERROR=label]  [,REG=gpr]

            [,PRODUCT=prodname]
```

where:

TYPE= - fixed keyword value
- specifies the type of #SPANCHK function to be performed.
TYPE=SCPVAL requests that in-line coding be generated to verify that the SCP under which this module is executing is compatible with the SCP for which the module was generated; the “SCP=” and “ERROR=” operands must be specified if TYPE=SCPVAL. The SCP is checked by means of TM instructions on the CVTDCB field.
TYPE=(SCPVAL,NOBASE) specifies that the TYPE=SCPVAL function be performed and that addressability to the CVT has already been established; if the NOBASE subparameter is omitted, instructions are generated to obtain addressability to the CVT - the CVT DSECT must be included in the assembly.
TYPE=PRODLEV requests that the currently defined release level of the Span Program Product to which this module belongs be returned as a 4-byte Global SETC symbol; the calling program should include a GBLC definition for the symbol &#RELNO.

SCP= - fixed keyword value
- specifies the class of Operating System under which this module is to be run. See #SPANTRY macro, SCP= operand, for permissible values. This operand must be present if TYPE=SCPVAL is specified.

ERROR= - label
- specifies the label at the beginning of the coding within the user program that handles SCP incompatibilities for this module. This operand must be present if TYPE=SCPVAL is specified.

REG= - General Purpose Register, symbolic register name
- specifies a work register to be used by TYPE=SCPVAL processing. Default is R14. This parameter is ignored if TYPE=(SCPVAL,NOBASE) is specified.

PRODUCT= - Span Program Product common name
- specifies the name of the Span Software product whose release level is required to be returned in the &#RELNO global variable. This parameter is required if TYPE=PRODLEV is specified.

2.4 #SPANLTG Macro - LTORG and Patch Area Generation

The #SPANLTG macro should be used in place of the LTORG assembler statement. A LTORG statement is generated as well as a patch area for the module. The patch area is a series of S-type constants, which greatly simplifies the development of SUPERZAP fixes to code.

format:

```
                   [       [64]]  [     [NO ]]
(name)   #SPANLTG  [LENGTH=[  ]]  [,PAD=[   ]]
                   [       [nn]]  [     [YES]]

                   [,BASEREG=n]  [,LEAVE=n]
```

where:

(name) - variable symbol name, max=8 bytes
- specifies the value of a label to be placed on the generated assembler DC instruction that defines the patch area. This name appears as an 8-byte prefix to the patch area. Default is “ZAPAREA” for the first use of the #SPANLTG macro in an assembly, and a system-generated name for subsequent #SPANLTG macro instructions. If the name field defaults to “ZAPAREA”, the patch area has a 16-byte prefix of the form “ZAPAREA.csectname”.

LENGTH= - numeric value
- specifies the number of half-words of patch area to be generated in addition to the 8- or 16-byte patch area prefix. Default value is 64.

PAD= - fixed keyword value
- “PAD=YES” specifies that the patch area generated is to extend to the limit of addressability of this module, modified by the “BASEREG=” and “LEAVE=” parameters.

BASEREG= - numeric value
- specifies the number of base registers to be used to calculate the maximum addressable size of the module for the purposes of generating a patch area. This parameter is used only if the “PAD=YES” parameter is specified. The default if this parameter is not specified is the number of base registers specified on the preceding #SPANTRY macro.

LEAVE= - numeric value
- specifies the number of half-words to be left within the addressable size of this module at the end of the patch area. This parameter is used only if the “PAD=YES” parameter is specified. The default if this parameter is not specified is zero, which means that the patch area

will exactly use up the addressing range of the base registers.

2.5 #SPANTRY Macro - Module or Subroutine Entry Point Macro

The #SPANTRY macro provides Entry Point linkage for all called routines, including routines called by the Operating System. If coded at the start of an assembly, #SPANTRY must be the first executable instruction in the Control Section (and must be placed after the CSECT statement). Use of #SPANTRY should not be mixed with the use of any other Entry Point macros; the #SPANXIT macro should be used to provide matching Exit Point linkage. The first invocation of #SPANTRY provides, by default, standard (Rn) Register Equates.

format:

```
                                      [,S]
(name)  #SPANTRY  [length]  [,dname]  [  ]
                                      [,L]

                  [      {(r1,r2, ... )}]
                  [,BASE={             }]
                  [      {NONE         }]

                  [      [RENT   ]]
                  [,TYPE=[STATIC ]]  [,PRODUCT=prodname]
                  [      [NOSAVE ]]
                  [      [NOCHAIN]]

                  [       [NO ]]
                  [,CLEAR=[   ]]
                  [       [YES]]

                  [     [DOS    ]]
                  [     [DOS/VS ]]
                  [     [SVS    ]]
                  [     [VS1    ]]
                  [     [OS     ]]  [        [1]]
                  [,SCP=[OS/VS  ]]  [,MODLEV=[ ]]
                  [     [VS2R2  ]]  [        [n]]
                  [     [VS2R3  ]]
                  [     [MVS    ]]
                  [     [MVS/XA ]]
                  [     [MVS/ESA]]
                  [     [390    ]]
                  [     [OS/390 ]]
                  [     [Z/OS   ]]

                  [         [YES  ]]  [         [YES]]
                  [,EXTCALL=[NO   ]]  [,ASMTIME=[   ]]
                  [         [CSECT]]  [         [NO ]]

                  [         [    [ {RES }]  ]
                  [         [(ESA[,{24  }]) ]
                  [         [    [ {2431}]  ]
                  [,LINKAGE=[STD ]          ]
                  [         [NONE]          ]

                  [,CATCHER='eyecatcher']

                  [        [YES]]  [     [ONLY     ]]
                  [,MODUSE=[   ]]  [,EQU=[NO       ]]
                  [        [NO ]]  [     [regprefix]]
```

where:

(name) - variable symbol name, max=8 bytes for DOS
- specifies the value of a label to be placed on the first executable instruction generated by the #SPANTRY macro; the value of “name” also appears as part of the Entry Point constant. If “name” is not specified, the CSECT name appears as part of the Entry Point constant.

length - absolute value, symbolic name
- used only if TYPE=RENT, default is 72; specifies the length of the dynamic save area to be obtained upon execution of the #SPANTRY macro. If an absolute value is specified, the minimum value is 72. The length specified must include 72 bytes for the standard register save area (even if the LINKAGE=ESA option is used). Indicate the absence of this parameter with a comma if the “dname”, “L” or “S” value is to be specified.

dname - symbolic name
- used only if TYPE=RENT, no default; specifies the name of a DSECT to be based over the dynamic save area obtained upon execution of the #SPANTRY macro. The DSECT must include at the start a definition of the standard 72-byte register save area. Indicate the absence of this parameter with a comma if the “L” or “S” value is to be specified.

S - fixed character value
- specifies that the length of the work area to be obtained by the #SPANTRY macro (TYPE=RENT only) is less than 4096 bytes. This is the default and will not normally need to be specified.

L - fixed character value
- specifies that the length of the work area to be obtained by the #SPANTRY macro (TYPE=RENT only) is greater than or equal to 4096 bytes. This parameter need be specified only if a long work area is required and the work area length is not specified as a numeric value in the “length” parameter of this #SPANTRY macro.

BASE= - Register number(s), symbolic register name(s), or the keyword NONE
- specifies the base register(s) to be used for this CSECT or subroutine; default is R12. If more than one base register is required, a list of registers may be specified, enclosed in parentheses. On the second and subsequent uses of the #SPANTRY macro in an assembly, the value of BASE will be carried forward from the first occurrence of #SPANTRY, base registers apart from the first may be overridden in any invocation of #SPANTRY, but the original base

registers will still be carried forward. All registers are permitted as base registers, with the exception of R0, R1, R13, R14, R15. The specification BASE=NONE defines a module with no base registers at all, that uses relative jump instructions instead of branch instructions. This feature is for use in very special cases, to produce large assembler modules that do not use system services or standard Span Software or IBM macros. The #SPANXIT and #SPAMODE macros support the BASE=NONE feature.

TYPE= - fixed keyword value
- specifies the type of #SPANTRY expansion to be generated; default is RENT.

RENT specifies re-entrant coding with dynamic save area.
STATIC specifies an in-line save area.
NOSAVE specifies that this routine will not call a lower-level routine, no new save area will be generated.
NOCHAIN specifies that save areas are pre-chained, the save area addressed by R13 upon execution of #SPANTRY already points to a usable lower save area.

PRODUCT= - Span program product common name
- specifies the name of the Span Software program product to which this module belongs, and causes a Span Software Copyright notice to be generated. PRODUCT should be specified only once for each load module in the product; for modules which are not part of any specific product, PRODUCT=UTIL should be specified.

CLEAR= - fixed keyword value
- CLEAR=YES specifies (for a TYPE=RENT expansion only) that the dynamic save area allocated by the #SPANTRY macro is to be set to binary zeros. The cleared section of the dynamic save area begins after the 72-byte register save area that is at the start of the allocated storage. Code to perform storage clearing is generated only if the dynamic save area length is greater than 72 bytes. If this parameter is not specified, the storage is not cleared.

SCP=

- fixed keyword value
- specifies the class of Operating System for which this #SPANTRY generation should build code. SCP need be specified only on the first occurrence of the #SPANTRY macro; default is “OS/VS”.

DOS	specifies DOS/VS Release 32 onwards
DOS/VS	specifies DOS/VS Release 32 onwards
SVS	specifies SVS only
VS1	specifies OS/VS1 only
OS	specifies all versions of OS/360 and OS/VS
OS/VS	specifies OS/VS1, SVS, MVS, MVS/XA, MVS/ESA
VS2R2	specifies MVS and later systems only
VS2R3	specifies MVS and later systems only
MVS	specifies MVS and later systems only
MVS/XA	specifies MVS/XA and later systems only
MVS/ESA	specifies MVS/ESA and later systems only
390	specifies OS/390 Version 1 and later systems only
OS/390	specifies OS/390 Version 1 and later systems only
Z/OS	specifies z/OS systems only

MODLEV=

- numeric value, DOS only
- specifies the level of this module; ie MODLEV=1 specifies that this module is entered directly from DOS, any other value specifies that this module is called from another module using Span Software standard linkage macros and conventions. Default is MODLEV=1.

EXTCALL=

- fixed keyword value
- specifies, for the second or subsequent use of the #SPANTRY macro in an assembly, that the Entry Point at which the #SPANTRY macro is coded can be called from an external CSECT.
  EXTCALL=YES specifies that an external call is possible and generates coding to reload base registers for this module.
  EXTCALL=NO specifies that this Entry Point is called only from within this module.
  EXTCALL=CSECT specifies that an external call is possible from a different CSECT within the same assembly - linkage is generated as if this were the only #SPANTRY macro in this assembly.
  Default is EXTCALL=NO.

ASMTIME=
- fixed keyword value, not DOS
- specifies whether (ASMTIME=YES) or not (ASMTIME=NO) a character representation of the time and date of assembly should be included as part of the Entry Point constant for this module; default is ASMTIME=YES. This parameter is ignored except on the first occurrence of the #SPANTRY macro in an assembly.

LINKAGE=
- fixed keyword value, not DOS
- specifies the module entry and exit linkage technique to be used.
  LINKAGE=STD (the default) uses traditional MVS module linkage with register save area chaining.
  LINKAGE=ESA uses the ESA Linkage Stack instead of register save areas. The linkage stack is manipulated via BAKR, PR and EREG instructions. If this option is specified no SVCs will be issued in order to allocate storage, and the module may be invoked by an AR-mode caller. A second sub-parameter may optionally be used to specific the location of any dynamic save area - valid values for this are: LINKAGE=(ESA,RES) (default) to specify the save area will be allocated depending on the location of the caller; LINKAGE=(ESA,24) to specify storage below the 16-Megabyte line; LINKAGE=(ESA,2431) to specify storage below the line backed above the line.
  LINKAGE=NONE does not use either save areas or the linkage stack, and should be used with care.

CATCHER=
- character string
- specifies an optional “eye-catcher” literal string to be included at the start of the module for identification purposes. If the string contains blanks or special characters it should be enclosed in single quotes. The specified eye-catcher will be prefixed by two blank characters when it is included in the module.

MODUSE=
- fixed keyword value
- specifies whether (MODUSE=YES) or not (MODUSE=NO) a module/entry point use count should be maintained by the #SPANTRY macro. This option causes to be generated a fullword use count field preceded by the 3-character literal “USE” within the code of the Entry Point linkage. This count is incremented by one each time a branch is taken to this Entry Point. Note that if MODUSE=YES is used the module is not strictly reentrant. MODUSE=YES is the default except for TYPE=RENT.

EQU= - fixed keyword value or register prefix

- EQU=ONLY specifies that no entry linkage at all is to be generated; standard register equates are produced if this is the first #SPANTRY macro in the assembly.
EQU=NO specifies that no register equates at all are to be generated; if this is specified on the first occurrence of #SPANTRY, subsequent occurrences of #SPANTRY will not produce register equates either.
EQU=regprefix allows any user-defined register prefix to be generated in addition to the standard Rx equates.

2.6 #SPANXIT Macro - Module or Subroutine Exit Point Macro

The #SPANXIT macro provides exit linkage for all called routines, and is a companion to the #SPANTRY macro.

format:

```
                                          [,S]
(name)   #SPANXIT   [length]   [,dname]   [  ]
                                          [,L]

                    [,RC=returncode]

                    [      [RENT   ]]   [          (R0,     )]
                    [,TYPE=[STATIC ]]   [,PRESERV=(   ,  ... )]
                    [      [NOSAVE ]]   [          (R1,     )]
                    [      [NOCHAIN]]

                    [         [ESA ]]
                    [,LINKAGE=[STD ]]
                    [         [NONE]]

                    [       [NO ]]
                    [,CLEAR=[   ]]
                    [       [YES]]

                    [,OPT=XCTL,TARGET=modname]

                    [,PASS=parameter]
```

where:

(name)
- variable symbol name, max=8 bytes for DOS
- specifies the value of a label to be placed on the first executable instruction generated by the #SPANXIT macro.

length
- absolute value, symbolic name
- used only if TYPE=RENT specified on the #SPANXIT macro or TYPE=RENT specified or defaulted on the previous #SPANTRY macro; default is the length of the dynamic save area specified on the previous #SPANTRY macro, or 72 if none was specified. Indicate the absence of this parameter with a comma if the “dname”, “L” or “S” value is to be specified.

dname
- symbolic name
- reserved for compatibility with the #SPANTRY macro. Indicate the absence of this parameter with a comma if the “L” or “S” value is to be specified.

S
- fixed character value
- specifies that the length of the work area to be freed by the #SPANXIT macro (TYPE=RENT only) is less than 4096 bytes. This is the default and will not normally need to be specified.

L
- fixed character value
- specifies that the length of the work area to be freed by the #SPANXIT macro (TYPE=RENT only) is greater than or equal to 4096 bytes. This parameter need be specified only if a long work area was obtained by the corresponding #SPANTRY macro, or a long work area was specified on this #SPANXIT macro, and the length was not specified as a numeric value.

RC=
- absolute value, Register number, symbolic register name
- specifies the return code to be passed to the calling routine in general register 15. The maximum value that should be contained in the specified register is 4095; if this parameter is omitted, register 15 is assumed to contain the required return code and is passed to the calling routine unchanged.

TYPE=
- fixed keyword value
- specifies the type of #SPANXIT expansion to be generated - see #SPANTRY macro description. If this parameter is omitted, the coding generated will be compatible with the most recent #SPANTRY macro in the assembly.

PRESERV=
- fixed keyword value(s)
- specifies one or both of registers R0 and R1 that are to be passed back to the calling routine unchanged by the #SPANXIT macro (ie not reloaded from the save area). Register 0 is specified by the keyword “R0” and register 1 by the keyword “R1”. If both registers are to be preserved then specify the keywords as a list within brackets and separated by commas.

LINKAGE=
- fixed keyword value, not DOS
- specifies the module entry and exit linkage technique to be used.
  LINKAGE=STD (the default) uses traditional MVS module linkage with register save area chaining.
  LINKAGE=ESA uses the ESA Linkage Stack instead of register save areas. The linkage stack is manipulated via BAKR, PR and EREG instructions.
  LINKAGE=NONE does not use either save areas or the linkage stack, and should be used with care.

CLEAR=
- fixed keyword value
- CLEAR=YES specifies (for a TYPE=RENT expansion using ESA linkage only) that the dynamic save area allocated by the #SPANTRY macro is to be set to binary zeros before the storage is released. This ensures that any contents of the dynamic save area will be destroyed, and will not be visible to any subsequent routine that issues a request for storage. All the storage is cleared, including the register save

area. If this parameter is not specified, the storage is not cleared before being released to the system.

OPT= - fixed keyword value
- OPT=XCTL specifies that when the exit linkage is complete this module wishes to transfer control by means of OS SVC 7 to the load module specified in the “TARGET=” parameter. If the “TARGET=” parameter is omitted, OPT=XCTL has no effect.

TARGET= - load module name
- valid only if OPT=XCTL is specified. Specifies the name of the load module to which this module is to transfer control.

PASS= - variable name
- valid only if OPT=XCTL is specified. Specifies the name of a variable whose address is to be contained in register 1 when the module which is the target of the XCTL receives control. Note that this variable address should not be an address within the module issuing #SPANXIT with this option, as this load module may not be in storage when the target module receives control. Note also that the PASS= option is not compatible with the PRESERV= option if R1 is preserved; if both these are specified, the PASS= parameter will take precedence.

2.7 #SPBINSR Macro - Perform Binary Search of an Ordered Table

The #SPBINSR macro provides a general-purpose binary search function for ordered in-storage tables. Parameters are passed to the macro indicating the address and characteristics of the table, and the address of a search argument. A return code is issued indicating whether or not the search was successful, and, if successful, the address of the matching table entry is returned. The #SPBINSR macro may be executed in ASC AR mode on ESA-capable systems.

format:

```
(name)   #SPBINSR  TABLE=(tablereg), TABEND=(endreg)

                          [(reg) ]   [         [0      ]]
                  ,ENTLEN=[      ]   [,OFFSET=[        ]]
                          [length]   [         [offset]]

                      [(reg)      ]
                  ,ARG=[          ]  ,ARGLEN=nnn
                      [search-arg]

                  [        [(reg)]]
                  [,WKREG=[      ]]
                  [        [(R2) ]]
```

where:

(name) - variable symbol name

- specifies the value of a label to be placed on the first executable instruction generated by the #SPBINSR macro.

TABLE= - general register (0 - 12, 14)

- specifies a register that contains the address of the start of the sorted in-storage table that is to be searched.

TABEND= - general register (0 - 12, 15)

- specifies a register that contains the address of the end of the table. Note that the difference between the values of the end of the table and the start of the table must be a whole multiple of the table entry length, which is specified by the ENTLEN parameter.

ENTLEN= - general register (2 - 12), or integer value

- specifies the length of a single entry of the table.

OFFSET= - general register (2 - 12), or integer value

- specifies the offset within each table entry to the key field on which the table is sorted, and on which the binary search is to be performed. The default offset is zero.

ARG= - general register (2 - 12), or A-constant

- specifies the address of the search argument. The length of the argument must equal the key length of the table, and is specified by the ARGLEN parameter.

ARGLEN= - general register (2 - 12), or integer value
- specifies the key length of the table and the length of the argument. The value specified must be less than or equal to the length of a table entry, and has a maximum value of 256 bytes.

WKREG= - general register (2 - 12)
- specifies a register that may be used by the macro for working. The register specified may be the same as the register used for the TABLE or TABEND parameter. If the WKREG parameter is omitted, Register 2 will be used, and it will be saved in the save area addressed by Register 13, and restored at the completion of the macro.

Return Codes

After execution of the #SPBINSR macro, register 15 will contain one of the following return codes:

0 The search was successful. Register 1 contains the address of the table entry located.

4 The search was unsuccessful. The contents of Register 1 are unpredictable.

2.8 #SPCPTIM Macro - Execute Routine and Calculate CPU Usage

The #SPCPTIM macro provides a means of simplifying the use of Span Software program product SPMV-002 (SPSMFINF) when this is used to calculate the CPU time used by a particular routine. After execution of the #SPCPTIM macro, Register 1 contains the address of a 15-byte area containing the CPU time used by the routine in question in character format, suitable for printing or displaying in a WTO message.

Note that if this macro is to be used to invoke Release 2.x of SPSMFINF on an MVS/SE or MVS/SP1 system then the SPSMFINF routine is entered in problem mode and may not be able to give completely accurate information (CPU time up to the last dispatch will be returned). For MVS/SP2 (MVS/XA) and later, accurate information is returned when SPSMFINF is entered in problem mode. See the SPSMFINF User Manual for further details. For SPANEX users the #SPXSVC macro can be used for full access to SPSMFINF features, and the SPANEX=YES operand of the #SPCPTIM macro will permit fully accurate values for CPU time to be returned on MVS/SE and MVS/SP1 systems.

This macro generates a V-constant for the SPSMFASM entry point of the SPSMFINF product unless either the INFEP= or the SPANEX=YES parameter is specified.

format:

```
(name)  #SPCPTIM  RTN=rtnname  [,LINK=YES]  [,SPANEX=YES]

                  [        [(reg) ]]
                  [,INFEP=[       ]]
                  [        [epaddr]]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPCPTIM macro.

RTN= - symbolic entry point name
- specifies the name of the entry point of the routine whose CPU time is to be measured.

LINK= - fixed keyword value
- LINK=YES specifies that the entry point name specified in to be invoked by means of a LINK (SVC 6). If LINK=YES is not specified a CALL will be generated to the specified entry point name.

SPANEX=

- fixed keyword value
- SPANEX=YES specifies that this program will be run only under SPANEX, and this macro will generate linkage to the SPANEX SVC in order to access SPSMFINF. There is no advantage in using this option under an MVS/SP2 or later system. This parameter is mutually exclusive with the INFEP= parameter. If neither the SPANEX= nor the INFEP= parameter is specified, this macro will generate a V-constant for SPSMFASM. If SPANEX=YES is specified and the program is run not under SPANEX, the #SPCPTIM macro will perform no function.

INFEP=

- SPSMFASM routine address label value, (register 2-12)
- specifies the address of the SPSMFASM entry point of the SPSMFINF product. If the label format is used, the field addressed must contain the address of SPSMFASM; if the register format is used, the register containing the address of SPSMFASM must be in the range 2 to 12. This parameter is mutually exclusive with the SPANEX= parameter. If neither the INFEP= nor the SPANEX= parameter is specified, this macro will generate a V-constant for SPSMFASM.

2.9 #SPGLBL Macro - Define SPOUTPUT/SPZPARSE Control Block

The #SPGLBL macro is assembled to define the internal control block required by the SPOUTPUT and SPZPARSE service routines. TYPE=CSECT must be specified in order to create the SPGLBL CSECT which must either be link-edited with SPOUTPUT and SPZPARSE or available as a separate load module “SPGLBL”. SPANEX user programs may obtain the address of the SPGLBL CSECT from the SPXMGLBL field of the SPANEX ICB.

format:

```
                   [      [DSECT]]  [         [SYSLOG]]
(name)   #SPGLBL   [TYPE=[       ]]  [,DDNAME=[        ]]
                   [      [CSECT]]  [         [ddname]]
```

where:

(name) - symbolic name
- specifies the value of a label to be placed on the DSECT or CSECT statement generated at the start of the control block definition. Default is SPGLBL.

TYPE= - fixed keyword value
- specifies the type of #SPGLBL generation to be performed:
TYPE=DSECT specifies that a dummy section be generated to map the SPGLBL fields.
TYPE=CSECT specifies that the SPGLBL CSECT is being generated.
TYPE=DSECT is the default if “TYPE=” is not specified.

DDNAME= - symbolic value
- specifies the DDNAME to be used for the SYSLOG dataset output from the SPOUTPUT service routine if the “LOG=YES” option is in effect. Default DDNAME is “SYSLOG”.

2.10 #SPMSG Macro - SPOUTPUT Parameter Formatting

The #SPMSG macro is used to build a "stream mode" message to be output by the SPOUTPUT service routine, and whose address can be inserted into the SOB by means of the "MSG=" parameter of the #SPZSOB macro.

format:

```
name     #SPMSG     'text'
```

where:

name - variable symbol name
- specifies the value of a label to be placed on the message definition. The name field is required.

text - quoted character string
- specifies the actual text of the message to be output.

2.11 #SPMVCL Macro - Simulate MVCL instruction

The #SPMVCL macro is a functional alternative to the MVCL instruction. Benchmark tests have shown that this macro requires approximately 50% of the CPU time used by the MVCL instruction, when used for non-trivial moves. This macro should be considered as an alternative to the MVCL instruction when large number of data moves are to be performed, or when performance is critical. Note that the contents of both parameter registers, and of both odd-numbered registers that are numerically one higher than the parameter registers, will be destroyed by the execution of this macro. The contents of the registers after the execution of this macro will not necessarily be the same as if the MVCL instruction had been used. This macro does not currently support the pad character or area filling functions of the MVCL instruction.

format:

```
name     #SPMVCL    to-register, from-register
```

where:

name - variable symbol name
- specifies the value of a label to be placed at the beginning of the executable code generated by this macro.

to-register - even-numbered general register
- specifies the even-numbered register that contains the address to which data is to be moved. The odd-numbered register that is numerically one higher than this must contain the length to be moved.

from-register - even-numbered general register
- specifies the even-numbered register that contains the address from which data is to be moved.

2.12 #SPTEST Macro - Check Program Environment

The #SPTEST macro is used to generate code to check the environment in which the program is executing (eg whether a TSO foreground or background task). Note that the code produced by this macro is not re-entrant.

format:

```
                        ([TSO    ]        )
   (name)   #SPTEST     ([       ]  ,label)
                        ([NOTTSO]         )
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPTEST macro.

TSO - fixed keyword value
- specifies that a check for running in TSO foreground is to be made. If this program is running under TSO, then a branch is taken to the address specified as “label”, the second sub-parameter of the macro operand. Note that one of “TSO” and “NOTTSO” must be specified as the first sub-parameter of a two-sub-parameter list and that the “label” value is required as the second sub-parameter.

NOTTSO - fixed keyword value
- specifies that a check for running in TSO foreground is to be made. If this program is not running under TSO, then a branch is taken to the address specified as “label”, the second sub-parameter of the macro operand. Note that one of “TSO” and “NOTTSO” must be specified as the first sub-parameter of a two-sub-parameter list and that the “label” value is required as the second sub-parameter.

2.13 #SPXFIND Macro - Invoke SPANEX Lookup Functions

The #SPXFIND macro may be used by user or SPANEX modules to perform control block look-up functions. Control blocks for the SPANEX Restart and Job Networking facilities that may be found are: JRCB (Job Restart Control Block), SRCB (Step Restart Control Block), JRCX (JRCB Extension), CLABE (Catalog Look-Aside Buffer Entry). If the requested control block is successfully found, Register 15 on return from the macro will contain zero, and Register 1 will contain the address of the control block. For further information on the use and function of the SPANEX product, see the SPANEX General Usage Manual, Span Software Manual Ref: SPX-02.

format:

```
                     [JRCB ]
(name)   #SPXFIND    [SRCB ]   [,N]   [,JOBNAME=jobname]
                     [JRCX ]
                     [CLABE]

                                      [,JRCB=jrcbaddr]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXFIND macro.

type - fixed keyword value
- the first operand of the #SPXFIND macro specifies the type of lookup to be performed:
JRCB specifies that a JRCB is to be found; the JOBNAME= parameter is required for this function.
SRCB specifies that a SRCB is to be found; the JRCB= parameter is required for this function.
JRCX specifies that a JRCB extension is to be found; the JRCB= parameter is required for this function.
CLABE specifies that a CLAB entry is to be found; the JRCB= parameter is required for this function.

N - fixed keyword value
- specifies that an Abend is not to be generated if the required control block cannot be found. If this parameter is specified, a non-zero value in Register 15 indicates that the control block cannot be located. Without this parameter, an Abend U0056 is generated for this condition.

JOBNAME= - label, (general register)
- specifies the Jobname for which a control block is to be found.

JRCB=	-	label, (general register)
	-	specifies the JRCB address for which a related control block is to be found.

2.14 #SPXICB Macro - Generate SPANEX DSECTs

The #SPXICB macro is required by user modules that are to run as SPANEX Span Product programs, by SPANEX modules, and by SPANEX Installation Exit routines, SPANEX restart user exit routines, SPANEX user submit routines or any other SPANEX exit routines. A USING statement on symbol SPXICB provides addressability as required by other SPANEX macros. In addition to the #SPXICB macro, a COPY statement must be included at the beginning of the assembly for #SPXGLOB. For further information on the use and function of the SPANEX product, see the SPANEX General Usage Manual, Span Software Manual Ref: SPX-02.

format:

```
                                [           [YES ]
    #SPXICB    [DSECT=NO]  [,SVCPARM=[NO   ]]
                                [           [ONLY]]
```

where:

DSECT= - fixed keyword value
- DSECT=NO specifies that no “DSECT” statement is to be generated at the beginning of the Internal Control Block section.

SVCPARM= - fixed keyword value
- SVCPARM=YES specifies that the SPANEX SVC parameter blocks DSECT is required.
SVCPARM=NO specifies that the SPANEX SVC parameter blocks are not to be mapped.
SVCPARM=ONLY specifies that only the SPANEX SVC parameter blocks DSECT is required, and not the Internal Control Block DSECT.
Default is SVCPARM=YES.

2.15 #SPXICBA Macro - Find Field in SPANEX ICB

The #SPXICBA macro is recommended to be used by user modules that are to access fields in the SPANEX ICB (mapped by the #SPXICB macro). The offsets to some of the fields in the ICB can change from one SPANEX release to another, and from one operating system to another. Use of the #SPXICBA macro to obtain addressability to ICB fields ensures compatibility between user programs and SPANEX. Addressability to the SPANEX ICB is required for execution of this macro. For further information on the use and function of the SPANEX product, see the SPANEX General Usage Manual, Span Software Manual Ref: SPX-02.

format:

```
(name)  #SPXICBA  fieldname  [,register]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXICBA macro.

fieldname - standard ICB field name
- specifies the name of the field within the SPANEX ICB for which addressability is required. The following is a list of the fields within the ICB for which use of the #SPXICBA macro is essential:

SPXOPENL
SPXEXTL
SPXSDCB
SPXSTAXL
SPXLINE1
SPXCPUID
SPXADYNA
SPXCALTB
SPXCLOSL
SPXATTCH
SPXSOB
SPXXAMSG
SPXKSCDT
SPXACHKX
SPXAJBST
SPXCALUS
SPXESTL
SPXUDCB
SPXUSOB
SPXXASVC
SPXSCAS1
SPXTCBOK
SPXDATA3
SPXRACFN

register - register number, register equated symbol
- specifies the general register to be used to return the address of the required field. If this parameter is omitted, Register 1 is used.

2.16 #SPXMSG Macro - Internal Message Request to SPANEX

The #SPXMSG macro is used by SPANEX modules to issue messages to all SPANEX destinations including the SPANEX Message Log, using the SPANEX message editing facilities. User programs should normally use the #SPXUMSG macro in preference to the #SPXMSG macro.

format:

```
                           [      [label     ]]
(name)  #SPXMSG   [msgid]  [,TEXT=[          ]]
                           [      [(register)]]

                                [      [YES  ]]
                  [,FILLIN=NO]  [,CNSL=[     ]]
                                [      [FORCE]]

                  [,ABTERM=YES]  [,SEND=YES]  [,PUTLINE=YES]

                  [     [YES  ]]  [      [YES ]]
                  [,LOG=[ONLY ]]  [,GLOG=[    ]]
                  [     [CLOSE]]  [      [ONLY]]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXMSG macro.

msgid - numeric SPANEX message ID
- specifies the number of the SPANEX canned message to be issued. This number is the relative position in the SPANEX message tables of the message to be issued and is also the number that appears in the "SPXnnn" message identifier of the message itself. The maximum value of "msgid" is "995" for users of the #SPXMSG macro. Note that the "msgid" and "TEXT=" operands are mutually exclusive.

TEXT= - label, (general register)
- specifies the address of the message text to be issued as a SPANEX message. The first byte of the text area contains the length of the remaining bytes of the text area (as built by the #SPMSG macro). Specify "TEXT=(0)" if register 0 is already loaded with the address of the message text area with the appropriate option flag settings in the high-order byte - this option is not permitted if the "GLOG=" parameter is specified. Note that the "msgid" and "TEXT=" operands are mutually exclusive.

FILLIN=

- fixed keyword value
- FILLIN=NO specifies that editing of the supplied text should not be performed by the SPANEX message service routines. Editing is performed if FILLIN=NO is not specified.

CNSL=

- fixed keyword value
- CNSL=YES specifies that the message should be issued as a WTO message if SPANEX is running as a batch job and has the “OPT=H” SPANEX option in effect. The message will also appear on the SPANEX Message Log.

  CNSL=FORCE specifies that the message should be issued as a WTO message if SPANEX is running as a batch job regardless of any SPANEX option specification. The message will also appear on the SPANEX Message Log.

ABTERM=

- fixed keyword value
- ABTERM=YES specifies that SPANEX should be Abended (in batch) with a code User 16 after issuing this message. The #SPXMSG macro will not return if “ABTERM=YES” is specified.

SEND=

- fixed keyword value
- SEND=YES specifies that the message should be issued to the SPANEX NOTIFY userid (if any) to be received by him (if a TSO user) when he next logs on to TSO.

PUTLINE=

- fixed keyword value
- PUTLINE=YES specifies that the message be issued as a PUTLINE to the TSO user if this program is being run as a TSO task. This option will be implemented only if SPANEX is running as a TSO Command Processor.

LOG=

- fixed keyword value
- LOG=CLOSE specifies that SPANEX has finished issuing messages. A #SPXMSG macro with the LOG=CLOSE operand will be the last #SPXMSG macro issued by SPANEX. The “msgid” and “TEXT=” operands are ignored if LOG=CLOSE is specified.

  LOG=ONLY specifies that this message is to be sent to the SPANEX Message Log only and that other message destinations are to be ignored.

  LOG=YES is the default and specifies that this message is to be sent to the SPANEX Message Log. All messages are sent to the SPANEX Message Log except when the “GLOG=ONLY” parameter is specified.

GLOG=	-	fixed keyword value
	-	GLOG=YES specifies that this message is to be sent to the SPANEX Global Log dataset (if any, supported only for SPANEX job networks and defined in the SPANEX Restart Control Module for the network) after it has been issued to any other destinations specified. GLOG=ONLY specifies that this message is to be sent to the SPANEX Global Log only. Note that it is not valid to specify the "GLOG=" parameter if "TEXT=(0)" is specified.

2.17 #SPXQ Macro - Locate SPANEX Internal Control Block

The #SPXQ macro is provided for SPANEX user programs to enable the address of the SPANEX Internal Control Block (ICB) to be obtained if this is not available from the original parameter passed by SPANEX. The #SPXQ macro also enables all programs to determine whether or not they are executing under the control of SPANEX. Programs which use the #SPXQ macro must also include the Operating System DSECT macros for the PSA (MVS only), CVT and TCB. Note that the code generated by the #SPXQ macro is Operating System dependent, and modules using it must be re-assembled before being executed under a different Operating System.

format:

```
(name)  #SPXQ     [USERCHK=YES]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXQ macro.

USERCHK= - fixed keyword value
- USERCHK=YES specifies that the user program that is issuing the #SPXQ macro is always to be run with the “OPT=C” SPANEX option because the Operating System Checkpoint/Restart system is being used. This parameter is not required for programs not using OS Checkpoint/Restart.

Return Codes

After execution of the #SPXQ macro, register 15 will contain one of the following return codes:

0 The issuing routine is executing as part of the SPANEX control task, register 1 contains the address of the SPANEX ICB.

4 The issuing routine is executing as a subtask of SPANEX, register 1 contains the address of the SPANEX ICB.

8 The issuing routine is not executing under the control of SPANEX, register 1 has no meaning.

2.18 #SPXRSTU Macro - Request to SPANEX Utility

The #SPXRSTU macro is used by SPANEX Restart and Job Networking user exit routines or user application programs, that are running with SPANEX option “OPT=M”, to access the SPANEX Utility user interface. This can be used to inquire as to the status of user program execution (as far as the Restart and Networking facilities are concerned), to set or to remove an error condition (to force or prevent a restart), and to CANCEL or SCHEDULE the execution of a job that is a member of a SPANEX job network from within another job in the same job network. This macro is also used to manipulate the “last-run” status for a multiple-execution SPANEX job, and to issue internal HOLD and POST commands for jobs within a SPANEX network. Note that many of the functions provided via the #SPXRSTU macro do not actually take effect until the normal termination of the jobstep in which they are invoked - step termination is taken as a synchronization point by SPANEX to ensure consistent and valid processing.

format:

```
(name)  #SPXRSTU  TYPE=calltype  [,JOBNAME=area]

                                 [,EVENT=event]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXRSTU macro.

TYPE= - fixed keyword value
- specifies the SPANEX Utility function name required. Valid “calltype” values:

TYPE=SETERR Force a restart next run of this job, regardless of Abend or Return Code from this application program. This option will reset a previous “TYPE=SETOK” request.

TYPE=SETOK Prevent a restart next run of this job, even if this jobstep Abends or terminates with an abnormal Return Code. This option will reset a previous “TYPE=SETERR” request.

TYPE=INQ Inquire whether or not restart is to be required due to error or “TYPE=SETERR” request so far.

TYPE=CANCEL Cancel a network job. The name of the job to be cancelled must be supplied by means of the “JOBNAME=” parameter. The cancel will be actioned only if the job in question has not yet been scheduled by SPANEX.

TYPE=SCHEDULE
Schedule a network job. The name of the job to be scheduled must be

supplied by means of the "JOBNAME=" parameter. The schedule will be actioned only if the job in question has not yet been scheduled by SPANEX, and will not take place until the termination of the user program issuing the #SPXRSTU macro with this option.

TYPE=(SCHEDULE,FORCE)

Schedule a network job that is EXCLUDEd from the network. The name of the job to be scheduled must be supplied by means of the "JOBNAME=" parameter. The schedule will be actioned only if the job in question has not yet been scheduled by SPANEX, and will not take place until the termination of the user program issuing the #SPXRSTU macro with this option.

TYPE=LASTRUN

Indicate, for a multiple-execution job, that this is the last execution of the job for this run of the network. This function can be used only from within the multiple-execution job itself. The job must be defined with the "PROCESS=MULT" option on the #SPXJOB macro in the RCM generation. This option will reset a previous "TYPE=NOTLAST" request.

TYPE=NOTLAST

Indicate, for a multiple-execution job, that this is not the last execution of the job for this run of the network. This function can be used only from within the multiple-execution job itself. The job must be defined with the "PROCESS=MULT" option on the #SPXJOB macro in the RCM generation. This option will reset a previous "TYPE=LASTRUN" request.

TYPE=HOLD

Issue internally a SPANEX Utility HOLD command for an event for another job within the same network. The name of the job to be held must be specified by means of the JOBNAME= parameter, and the event number (from 1-8) must be specified by means of the EVENT= parameter.

TYPE=POST Issue internally a SPANEX Utility POST command to signal an event complete for another job within the same network. The name of the job to be held must be specified by means of the JOBNAME= parameter, and the event number (from 1-8) must be specified by means of the EVENT= parameter.

JOBNAME= - area addr, (general register)
- specifies the address of an 8-byte area which contains the name of the job to be cancelled, scheduled, held or posted. This parameter is required for TYPEs "CANCEL", "SCHEDULE", "(SCHEDULE,FORCE)", "HOLD" and "POST".

EVENT= - event number, (general register)
- specifies the number (from 1-8) of the external event whose status is to be changed by means of a HOLD or POST function call. The event should be specified either as a literal number, or in a general register. If register notation is used, the nominated register must contain the binary event number in its low-order byte. This parameter is required for TYPEs "HOLD" and "POST".

Return Codes

After execution of the #SPXRSTU macro, register 15 will contain one of the following return codes.

TYPE=SETERR	0	always
TYPE=SETOK	0	always
TYPE=INQ	0	No error has occurred or been requested
	4	The error condition is set
TYPE=CANCEL	0	Job cancelled. Note that any post-requisite jobs of the cancelled job that may become eligible for scheduling as a result of the cancellation just performed will not be automatically scheduled by SPANEX until the end of the current jobstep.
	4	Job requested for cancel not found in the current Restart Control Module, or illegal TYPE=CANCEL macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during cancel processing.
	16	Job not cancelled because it has already been scheduled.

TYPE=SCHEDULE		
	0	Job set for scheduling. Note that the job will not be automatically scheduled by SPANEX until the end of the current jobstep.
	4	Job requested for schedule not found in the current Restart Control Module, or illegal TYPE=SCHEDULE macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during schedule processing.
	16	Job not scheduled because it has already been scheduled.
	20	Job not scheduled because it is EXCLUDEd from the network. The user program may cause the scheduling of this job if required by means of a TYPE=(SCHEDULE,FORCE) call to the #SPXRSTU macro.
TYPE=(SCHEDULE,FORCE)		
	0	Job set for scheduling. Note that the job will not be automatically scheduled by SPANEX until the end of the current jobstep.
	4	Job requested for schedule not found in the current Restart Control Module, or illegal macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during schedule processing.
	16	Job not scheduled because it was not EXCLUDEd from the network, or job has already been force-scheduled.
TYPE=LASTRUN	0	Function successfully completed. Subsequent jobs will be scheduled upon successful completion of the last step of this job, without operator action.
	4	Job not found in the current Restart Control Module, or illegal macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during processing of the LASTRUN function.
	16	Job is not defined with the PROCESS=MULT option in the RCM.

TYPE=NOTLAST	0	Function successfully completed. Subsequent jobs will not be scheduled upon successful completion of the last step of this job.
	4	Job not found in the current Restart Control Module, or illegal macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during processing of the NOTLAST function.
	16	Job is not defined with the PROCESS=MULT option in the RCM.
TYPE=HOLD	0	Function successfully completed. The requested event will be set not-complete for the specified job during step termination processing for this job.
	4	Job for which HOLD processing was requested was not found in the current Restart Control Module, or illegal macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during processing of the HOLD function.
	16	HOLD request was invalid because the Job was EXCLUDEd from this run of the network, or had already begun execution.
	20	HOLD request rejected because the specified Event Number was not in the range 1-8.
TYPE=POST	0	Function successfully completed. The requested event will be set complete for the specified job during step termination processing for this job.
	4	Job for which POST processing was requested was not found in the current Restart Control Module, or illegal macro call, eg not from a user program in OPT=M step.
	8	The current Restart Control Module does not define a SPANEX job network.
	12	Catalog error occurred during processing of the POST function.
	16	POST request was invalid because the Job was EXCLUDEd from this run of the network, or had already begun execution.
	20	POST request rejected because the specified Event Number was not in the range 1-8.

2.19 #SPXSVC Macro - Issue SPANEX SVC

The #SPXSVC macro is used by user modules that are authorized to run as SPANEX “Span Product” programs and provides access to SPANEX SVC services. Addressability to the SPANEX ICB is required. This macro is also used extensively within SPANEX modules to perform SVC functions.

format:

```
(name)  #SPXSVC   TYPE=svctype  [,BRANCH=NO]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXSVC macro.

TYPE= - fixed keyword value
- specifies the SPANEX SVC function name required. Note that all parameter registers should be loaded before issuing the #SPXSVC macro.
Valid “svctype” values:

TYPE=GETAUTH	Obtain APF authority
TYPE=RSETAUTH	Remove APF authority
TYPE=SETSWAP	Set Address Space swappable (MVS) - causes OKSWAP SYSEVENT to be issued
TYPE=NONSWAP	Set Address Space non-swappable (MVS) - causes TRANSWAP SYSEVENT to be issued
TYPE=SVC34	Issue SVC 34 for OS command
TYPE=GETCSCB	Allocate and build SPANEX CSCB
TYPE=FREECSCB	Dechain and free SPANEX CSCB
TYPE=OWNCODE	Execute caller's code in Key 0
TYPE=SDUMP1	Issue SDUMP SVC for user abend
TYPE=SDUMP2	Issue SDUMP SVC for operator command
TYPE=SDUMP3	Issue SDUMP SVC with user dump title
TYPE=SWAIT	Issue WAIT SVC on protected ECB or ECBLIST
TYPE=SIGNON	Initiate SPANEX Span Product
TYPE=CPUTIME	Enter SPSMFASM in supervisor mode (valid only if SPSMFINF product is installed, MVS only)
TYPE=DASDUCB	Perform UCB lookup

BRANCH= - fixed keyword value
- BRANCH=NO specifies that this #SPXSVC macro is to be executed by a user program task (SPANEX subtask) and therefore cannot use the branch interface to the SPANEX SVC facilities. If this operand is not specified and the #SPXSVC macro is executed by a user program, an abend of the user task will result. Default is BRANCH=YES for use by SPANEX modules and user exit routines only.

Parameter Registers and Return Codes

GETAUTH

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	Authorization has been set
	R15=4	Authorization has not been set because already authorized

RSETAUTH

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	Always

SETSWAP/NONSWAP

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=SRM Return Code from SYSEVENT	

SVC34

Input	R0	SVC 34 parameter register
	R1	SVC 34 parameter register

GETCSCB (For SPANEX Internal Use Only)

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	SPANEX CSCB built
	R15=4	SPANEX CSCB already built, STOP/MODIFY ECB reset

FREECSCB (For SPANEX Internal Use Only)

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	SPANEX CSCB freed
	R15=4	SPANEX CSCB does not exist

OWNCODE

Input	R0	No applicable information
	R1	Parameter (if any) to user code
	R15	Addr of user code entry point
Returns	R15	As set by user code

SDUMP1/SDUMP2 (For SPANEX Internal Use Only)

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	SDUMP successfully taken
	R15=4	Partial SDUMP taken
	R15=8	SDUMP not taken

SDUMP3

Input	R0	No applicable information
	R1	Addr of user dump title (first byte of title is length of remainder)
Returns	R15=0	SDUMP successfully taken
	R15=4	Partial SDUMP taken
	R15=8	SDUMP not taken

SWAIT

Input	R0	SVC WAIT parameter register
	R1	SVC WAIT parameter register

SIGNON (For SPANEX Internal Use Only)

Input	R0	No applicable information
	R1	No applicable information

CPUTIME

Input	R0	No applicable information
	R1	As required by SPSMFASM
Returns	R15	As set by SPSMFASM

DASDUCB

Input	R0	No applicable information
	R1	No applicable information
Returns	R15=0	VolSer in SPXCVOL field of ICB is not online
	R15=4	VolSer in SPXCVOL field of ICB is online
	R1	Addr of UCB for SPXCVOL volume (if R15=4)

2.20 #SPXUDDN Macro - Specify DDNAME for #SPXUMSG Message Requests

The #SPXUDDN macro is used by user modules that are designed to run as SPANEX “Span Product” programs and provides the ability to change the DDNAME used by SPANEX for user-issued messages. If the #SPXUDDN macro is not used, user messages will appear on the SPANEX Message Log. The #SPXUDDN macro must be executed before any #SPXUMSG macros or immediately preceded by a #SPXUMSG macro with the “CLOSE=YES” option.

format:

```
(name)  #SPXUDDN  DDNAME=ddname
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXUDDN macro.

DDNAME= - symbolic name, (general register)
- specifies the DDNAME (maximum 8 bytes) to be used for printed messages issued by the SPANEX user program via the #SPXUMSG macro. If the register format is used, the general register specified must contain the address of an 8-byte area which contains the required DDNAME, padded on the right with blanks if necessary.

2.21 #SPXUMSG Macro - User Message Request to SPANEX

The #SPXUMSG macro is used by user modules that are designed to run as SPANEX “Span Product” programs, and provides the ability to issue messages to all SPANEX destinations including the SPANEX Message Log, using the SPANEX message editing facilities.

format:

```
                       [      [label      ]]
(name)    #SPXUMSG     [TEXT=[            ]]
                       [      [(register)]]
                       [CLOSE=YES          ]

                       [,PRINT=ONLY] [,FILLIN=NO] [,CNSL=YES]

                       [,ABTERM=YES] [,SEND=YES] [,PUTLINE=YES]
```

where:

(name)
- variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPXUMSG macro.

TEXT=
- label, (general register)
- specifies the address of the message text to be issued as a SPANEX message. The first byte of the text area contains the length of the remaining bytes of the text area (as built by the #SPMSG macro). Specify “TEXT=(0)” if register 0 is already loaded with the address of the message text area with the appropriate option flag settings in the high-order byte.

CLOSE=
- fixed keyword value
- CLOSE=YES specifies that the user has finished issuing SPANEX messages. A #SPXUMSG macro with the CLOSE=YES operand should be the last #SPXUMSG macro issued by the user program, and should also immediately precede the #SPXUDDN macro if used to effect a change of print output DDNAME. The TEXT= operand is ignored if CLOSE=YES is specified.

PRINT=
- fixed keyword value
- PRINT=ONLY specifies that this message is to be sent only to the SPANEX Message Log (or user data set if the #SPXUDDN macro has been used to change DDNAME), and that other message destinations are to be ignored.
  PRINT=YES is the default and specifies that this message is to be sent to the SPANEX Message Log (or user data set), in addition to any other destinations.

FILLIN= - fixed keyword value
- FILLIN=NO specifies that editing of the supplied text should not be performed by the SPANEX message service routines. Editing is performed if FILLIN=NO is not specified.

CNSL= - fixed keyword value
- CNSL=YES specifies that the message should be issued as a WTO message if the user program is running as a batch job. The message will also appear on the SPANEX Message Log.

ABTERM= - fixed keyword value
- ABTERM=YES specifies that the user program should be Abended (in batch) with a code User 16 after issuing this message. The #SPXUMSG macro will not return if “ABTERM=YES” is specified.

SEND= - fixed keyword value
- SEND=YES specifies that the message should be issued to the SPANEX NOTIFY userid (if any) to be received by him (if a TSO user) when he next logs on to TSO.

PUTLINE= - fixed keyword value
- PUTLINE=YES specifies that the message be issued as a PUTLINE to the TSO user if this program is being run as a TSO task. This option will be implemented only if SPANEX is running as a TSO Command Processor.

2.22 #SPZCLOS Macro - SPZQPAM Close Macro

The #SPZCLOS macro is functionally equivalent to an Operating System "CLOSE" macro, and closes one DCB that has been in use by the Span Software SPZQPAM Queued PDS Access Method service routine.

format:

```
(name)  #SPZCLOS  dcbaddr
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPZCLOS macro.

dcbaddr - address field, (general register)
- specifies the address of the Data Control Block that is to be closed by SPZQPAM. See the Span Service Routines manual for usage information on SPZQPAM.

2.23 #SPZCMD Macro - SPZPARSE Parameter Formatting

The #SPZCMD macro is used to define a valid control statement command to the SPZPARSE Span Service Routine. It produces non-executable code that is passed as a parameter to the SPZPARSE routine by means of the #SPZPARS macro. There must be one #SPZCMD macro for each valid command, and each must be followed by #SPZPOSN and/or #SPZKWRD macros defining the parameters for that command. A #SPZCMD macro with an "ID=" parameter different from the previous ID value delimits a set of definition macros and begins a new set. See the Span Service Routines manual for usage information about the SPZPARSE routine.

format:

```
(name)     #SPZCMD    [command]   ,RTN=rtnaddr  [,ID=cmdid]

                      [,ALIAS=aliasname]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first instruction generated by the #SPZCMD macro.

command - character command name, max=8 bytes
- specifies the command to be recognized by SPZPARSE. This positional parameter is required, unless control statements with no command are to be accepted, in which case this must be the only #SPZCMD macro with no "command" parameter in this set of SPZPARSE definition macros.

RTN= - label value
- specifies the address of a routine in the user program that is to be given control when a control statement with the command specified (or with no command if the "command" positional parameter is omitted) is encountered. This routine will be entered with all registers (except 1,14,15) containing the same as before the #SPZPARS macro was issued, and with all operand parameter blocks filled in from the command operands.

ID= - single character macro-set identifier
- specifies an identifier by which a set of SPZPARSE definition macros may be identified. The value of this operand is specified in the "CMDID=" parameter of the #SPZPARS macro in order to identify the set of commands permitted.

ALIAS=

- character command name, max=8 bytes
- specifies an alias (alternative command name) of the control statement command specified by the “command” parameter. If the value specified for ALIAS is encountered in a control statement, the control statement is processed exactly as if the value specified for the “command” parameter were found. This parameter should be used to permit either alternative command names or abbreviations to be used for commands.

2.24 #SPZFIND Macro - Position to a Partitioned Dataset Member when using SPZQPAM

The #SPZFIND macro is exactly equivalent to the standard Operating System FIND macro, and is used instead of the FIND macro in user programs using the SPZQPAM Span Service Routine. Existing user programs that use the standard FIND with SPZQPAM must be changed to use #SPZFIND in order to run successfully with OS/390 Version 2.4.0 and later. The parameters of #SPZFIND are exactly equivalent to the parameters for FIND, and the appropriate Operating System manual should be referred to for definitive parameter descriptions.

format:

```
(name)     #SPZFIND  dcbaddr,

                     {nameaddr,D}
                     {          }
                     {ttrcaddr,C}
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first instruction generated by the #SPZFIND macro.

dcbaddr - DCB address
- specifies the address of the opened DCB for the Partitioned Dataset being accessed via SPZQPAM.

nameaddr,D - address of member name
- specifies the address of the member name within the Partitioned Dataset of the member to which position is to be set. The fixed character parameter “D” must be specified after the name address to indicate that a search by member name is being performed. Only one of the nameaddr and ttrcaddr parameters may be specified.

ttrcaddr,C - address of TTRC field
- specifies the address of a 4-byte field which contains a ttr (relative DASD address) in the first 3 bytes and a concatenation number in the fourth byte. This value may have been returned by an Operating System BLDL macro, or from the results of an invocation of the SPZDIRD Span Service Routine. Only one of the nameaddr and ttrcaddr parameters may be specified.

2.25 #SPZFLD Macro - SPOUTPUT Parameter Formatting

The #SPZFLD macro is used in message, screen format, and heading and footing definitions to describe a field of information in the output line to be generated by the SPOUTPUT Span Service Routine. A line of output is formed from one or more #SPZFLD macros in a series (with no other intervening coding), terminated by a #SPZFLD macro with the "LAST" attribute. The output line is dynamically constructed by SPOUTPUT immediately prior to performing the output operation.

format:

```
                 ([[TEXT    ]
                 ([[TIME    ]
                 ([[DATE    ]
                 ([[DATE4   ]
                 ([[PAGE    ]
                 ([[JOB     ]
(name)  #SPZFLD  ([[STEP    ] [,UL] [,LAST] [,LOW] [,BACKUP]
                 ([[PROC    ]
                 ([[CPRIGHT ]
                 ([[CPRIGHT2]
                 ([[SYSID   ]
                 ([[USERID  ]
                 ([BLANK
                 ([LINE
                 ([DLINE
                 ([MOD

                              [,colour] [,attr] [,CONTIN]

                              [ [CENTER]]
                              [,[      ]]
                              [ [RIGHT ]]

                              [,INDIRECT]  ]  )

                   [,offset]  [,'text']  [,label]

                   [,LENGTH=length]  [,DATADDR=addr]

                   [,OFFSET=offset]

                   [,NAME=macrolab]

                   [,HLIGHT=highlight]  [,COLR=colour]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the generated field definition. This permits this #SPZFLD macro to be referred to by the #SPZTITL and #SPZSOB macros for the purposes of defining titles or specifying message text areas.

The type of information to be included into the message field by this #SPZFLD macro is indicated by one of the following keywords:

TEXT - text as specified by the calling program at assembly time in the "text" parameter, or moved in at execution time by reference to the "label" label field (see below), or indirectly addressed by combination of the "DATADDR=" and "LENGTH=" operands. Length of the generated field is equal to the length of "text" specified or to the "LENGTH=" parameter.

TIME - time of day in the format HH:MM:SS. Length is always 9 for this field type.

DATE - date in the format DDMMMYY. Length is always 8 for this field type.

DATE4- date in the format DDMMMYYYY. Length is always 10 for this field type.

PAGE - current page or screen number. Length is always 6 for this field type.

JOB - current jobname. Length is always 8 for this field type.

STEP - current stepname. Length is always 8 for this field type.

PROC - current procedure stepname. Length is always 8 for this field type.

CPRIGHT Span Software Consultants Limited copyright notice. Length is always 45 for this field type.

CPRIGHT2 Span Software Consultants Limited copyright notice (mixed case). Length is always 53 for this field type.

SYSID - SMF System ID of the MVS, OS/390 or z/OS image where the program is run. Length is always 4 for this field type.

USERID user identifier of the user owning the jobstep or TSO session where the program is run. Length is always 8 for this field type.

These field definitions can be followed by one or more of the following modifiers:

UL - if the following line contains a field with the "LINE" type, this field will be underlined with a string of "-" characters. If the following line contains a field with the "DLINE" type, this field will be underlined with a string of "=" characters. The line with the "LINE" or "DLINE" type must directly follow both in assembly sequence and logical (output) sequence.

LOW - the field will appear in low intensity if output with SPOUTPUT "OPT=FULLSCR" option to a 3270-type terminal. This is the default except for screen headings, where high intensity is the default.

LAST - this is the last field on this output line.

BACKUP this field is to be positioned at the byte after the last non-blank character so far generated, plus the value specified by the "offset" parameter (see below)

colour - specifies the colour for this field if displayed on a colour VDU. SPOUTPUT currently supports 7-colour screens, and valid colour specifications are: BLUE, RED, PINK, GREEN, TURQUOIS, YELLOW, WHITE. Use of a colour attribute forces the use of the indirect format of the #SPZFLD parameter list; TEXT field types must use the DATADDR= and LENGTH= parameters to address the text contents of the field.

attr - specifies the attributes for this field if displayed on a 3270-type VDU. SPOUTPUT currently supports normal and extended high-lighting attributes, and valid attribute specifications are: HIGH (high intensity), NONDISP (non-display), PROTECT (protected), BLINK (blinking - 3270 extended high-lighting only), REVERSE (reverse video - 3270 extended high-lighting only), USCORE (underscored - 3270 extended high-lighting only). Use of a high-lighting attribute forces the use of the indirect format of the #SPZFLD parameter list; TEXT field types must use the DATADDR= and LENGTH= parameters to address the text contents of the field.

CONTIN specifies that this field is a continuation of the previous field. This option forces the use of extended #SPZFLD format generation (see the SPOUTPUT section of the Span Service Routines manual). The use of the CONTIN option is specifically for generation of 3270 full-screen output, and it is ignored for all other output destinations. Generally for full-screen output, each #SPZFLD macro definition causes a new "field" on the resulting 3270 screen, with the attendant 3270 overhead of an "attribute byte" occurring at the start of the field. Use of the CONTIN option, for 3270 screens supporting the Extended Data Stream only, allows the programmer to change the colour or attribute for succeeding characters without generating a new 3270 field; SPOUTPUT issues a 3270 "Set Attribute" order at the screen address implied by the "offset" parameter of the #SPZFLD macro. The screen attribute will be set to the combination of colour and high-lighting specified on the #SPZFLD macro; use of CONTIN with no colour or attribute specified will result in the screen being reset to the colour and attributes specified for the previous #SPZFLD macro that did not specify the CONTIN option. See the SPOUTPUT section of the Span Service Routines manual for a discussion of full-screen output.

CENTER
CENTRE specifies that the data specified by this #SPZFLD macro is to be centred within the output line. The centring is automatically adjusted according to the line width of the print dataset, and according to the model of 3270 VDU for Full-Screen mode. The position of the field within the output line can be modified by means of the "offset" parameter, which is interpreted as the number of columns to the right of the centre position to place the field. The CENTER parameter forces the use of extended #SPZFLD format

generation (see the SPOUTPUT section of the Span Service Routines manual).

RIGHT- specifies that the data specified by this #SPZFLD macro is to be positioned flush-right (against the right-hand margin) within the output line. The right-hand margin is automatically adjusted according to the line width of the print dataset, and according to the model of 3270 VDU for Full-Screen mode. The position of the field within the output line can be modified by means of the "offset" parameter, which is interpreted as the number of columns to the left of the flush-right position to place the field. The RIGHT parameter forces the use of extended #SPZFLD format generation (see the SPOUTPUT section of the Span Service Routines manual). If both the CENTER and the RIGHT parameters are specified, CENTER takes precedence and RIGHT is ignored.

INDIRECT specifies indirect addressing for text data (using the DATADDR= and LENGTH= parameters), and forces the use of extended #SPZFLD format generation (see the SPOUTPUT section of the Span Service Routines manual).

Three special type definitions are also available:
BLANK generate one complete blank output line.
LINE - generate "-" characters to underline selected fields in the previous line. These selected fields are specified with the "UL" modifier in their own #SPZFLD macros.
DLINE- generate "=" characters to double-underline selected fields in the previous line. These selected fields are specified with the "UL" modifier in their own #SPZFLD macros.
All three of the above special type definitions also imply the "LAST" modifier and should be used alone.

The further special type option "MOD" is also available. MOD must be the only type option specified, and is used to generate an executable form of #SPZFLD macro in order to permit run-time specification of indirect text field addresses (using the DATADDR=, LENGTH= and OFFSET= parameters), and full-screen attributes for high-lighting and colour (using the HLIGHT= and COLR= parameters).

offset - numeric value, *, *+n
- specifies the offset of this field from the beginning of the output line. This operand is required for all field types except "BLANK" and "LINE". "*" specifies that this field is to be placed in the character position after the last character so far generated; *+n specifies that this field is to be placed in the character position after the last character so far generated, plus the number of bytes specified by "n". For centred fields (CENTER parameter specified) the offset should be a numeric value, and specifies an offset to the right of the centre position for the field. For right-flush fields (RIGHT parameter specified) the offset should be a numeric value, and specifies an offset to the left of the right-hand margin for the placement of the field.

text - quoted character string
- specifies the actual text to be inserted into this field. If the "TEXT" field type is being generated, either this operand or the DATADDR= operand is required in order to specify the text to be inserted into the output message.

label - symbolic value
- specifies an assembler label to be used for the generated field containing the text specified by means of the "text" positional parameter. The label is given a length attribute equal to the length of the specified text. This operand is optional for a field type of "TEXT".

LENGTH= - numeric value, (general register)
- specifies the length of text being defined by this #SPZFLD macro. This length is used only if indirect addressing (DATADDR= parameter) is being used for the text data, if colour or extended highlighting is used for full-screen output, or if the assembled length of the field referenced by the DATADDR= parameter is not to be used as the length of the text. This parameter may also be used for an executable version of #SPZFLD ("MOD" type) to change the text length defined by this macro - the register format may be used in this case, with the specified register containing the required length.

DATADDR= - symbolic value, (general register)
- specifies the address of text to be included in the output message as a result of this #SPZFLD macro. This parameter is required only if indirect addressing is to be used for the text data, or if colour or extended highlighting is used for full-screen output. This parameter may also be used for an executable version of #SPZFLD ("MOD" type) to change the address of the text to be included in the output message - the register format for the address may be used in this case.

NAME= - symbolic value, (general register)
- specifies the address of a #SPZFLD macro to be dynamically modified by the executable form of #SPZFLD. This parameter is ignored if the type field of this macro is not "MOD". The NAME= parameter should specify the address of the beginning of an extended format #SPZFLD macro - a label may be placed on the start of an #SPZFLD macro by use of the "(name)" label field on the macro instruction.

HLIGHT= - one or more fixed keywords
- specifies a new option or set of options for high-lighting a field on a 3270 VDU when using the SPOUTPUT full-screen facilities. The HLIGHT= parameter is valid only if the MOD type of #SPZFLD macro is being coded, and has effect only for full-screen output. Valid high-lighting options are: LOW (resets high-brightness - high-brightness is the default for fields defined in screen headings), HIGH (high-brightness), NONDISP (non-display field), PROTECT (non-modifiable field - failure to specify PROTECT produces a modifiable field), BLINK (flashing field - 3270 extended high-lighting only), REVERSE (field shown in reverse video - 3270 extended high-lighting only), USCORE (under-scored field - 3270 extended high-lighting only). All required options must be specified together, as high-lighting options replace any previous options.

COLR= - fixed keyword value
- specifies a new colour for a field on a 3270 colour VDU when using the SPOUTPUT full-screen facilities. The COLR= parameter is valid only if the MOD type of #SPZFLD macro is being coded, and has effect only for full-screen output. Valid colour options are: BLUE (blue field), RED (red field), PINK (pink field), GREEN (green field), TURQUOIS (turquoise field), YELLOW (yellow field), WHITE (white field). The colour option specified replaces any previous colour.

2.26 #SPZKWRD Macro - SPZPARSE Parameter Formatting

The #SPZKWRD macro is used to define a valid keyword operand of a command that was defined in a previous #SPZCMD macro statement. There may be multiple #SPZKWRD macros (and #SPZPOSN macros) for one #SPZCMD macro. The #SPZKWRD macro generates non-executable code that is part of a parameter list to the SPZPARSE Span Service Routine.

format:

```
(name) #SPZKWRD parmname  [,FIELD=fieldname]

                [,LENGTH=fieldlen]

                [      [C ]]
                [      [CN]]
                [      [N ]]  [       [N]]
                [,FMT=[P ]]  [,REQD=[ ]]
                [      [B ]]  [       [Y]]
                [      [X ]]

                [,VALID=(value, ...)]

                [,SET=((value,instr,mask), ...)]

                [,SETCONT=((value,instr,mask), ...)]

                [,SETYN=mask]

                [,PRESFLD=field]  [,SETPRES=(instr,mask)]

                [        [YES]]  [         [YES]]
                [,VALUE=[NO ]]  [,ACCEPT=[   ]]
                [        [OPT]]  [         [NO ]]

                [,ALIAS=aliasname]  [,TYPE=MOD,NAME=name]
```

where:

(name) - variable symbol name

- symbol “name” is assigned to the user flag field for this operand parameter, to enable the user program to interrogate the status of this operand. The field with this label will be set to zero if this operand is not present in the command, to 4 if the operand is present, and to 8 if the operand is present but is in error. Fields at certain offsets from this label may also be interrogated in order to determine the length, value and address of the data entered for this operand - these fields are mapped by the DODDS DSECT which is generated by the #SPZPMAP macro (see the Span Service Routines manual for full details). Note that the “USING” statement must be on symbol DODUFLG. The value of this “name” parameter should be used in the “TYPE=MOD” executable form of the #SPZKWRD macro by the user program before calling SPZPARSE, in order to permit the user program to be written re-entrantly (if this is required).

parmname - character value, max=8 bytes
- specifies the literal value of the keyword parameter that this #SPZKWRD macro describes. This is the first positional parameter of the #SPZKWRD macro and is always required.

FIELD= - label value
- specifies the name of the field (if any) that is to be filled with the data input as an operand of the parameter defined by this macro (for a “keyword=value” parameter).

LENGTH= - numeric value, max=255
- specifies the length of the field that is to be filled with the data input as an operand of the parameter defined by this macro. If the “field” parameter is specified but the “LENGTH=” parameter is omitted, then the length taken will be the assembled length of the field defined by the “field” parameter. Note that if this #SPZKWRD macro describes a parameter that may have multiple sub-parameters (which are to be defined by following #SPZSUBP macros), then this length field must be the total length of all sub-parameters, including intervening commas and external parentheses; it may be helpful to set the length to the maximum (255 bytes) and to specify the lengths of individual sub-parameters on their respective #SPZSUBP macros.

FMT= - fixed keyword value
- defines the data format of the field that is to receive the parameter data.
FMT=C (the default) specifies character data.
FMT=CN specifies character name data, where the first character of the value must be alphabetic (A-Z) or one of the characters #, @ or $ (the primary currency symbol).
FMT=N specifies character numeric data.
FMT=P specifies packed decimal (numeric input data).
FMT=B specifies binary (numeric input data).
FMT=X specifies hexadecimal (character hex input data).

REQD= - fixed keyword value
- specifies whether or not this parameter is mandatory for every occurrence of the command defined by the preceding #SPZCMD macro.
REQD=N (the default) specifies that this parameter is optional.
REQD=Y specifies that this parameter is always required.

VALID=
- one or more character values
- specifies a list of values that this parameter may take. If this option is specified and the parameter as input is not one of this list of values, it will be rejected as an error.

SET=
- one or more SET parameters
- specifies an operation to be performed if this parameter as input takes one of a list of one or more values. The “SET=” option is specified as a sequence of one or more 3-part parameters, each of the form “(value,instruction,mask)”, where “value” is a literal value that the parameter may take, “instruction” is a valid Assembler operation code defining a System/390 or z/Architecture “Immediate” instruction, “mask” is the mask to be used by the immediate instruction. For example, if “SET=((ALL,OI,X'80'))” is specified, and the value “ALL” is input, then the field defined by the “field” parameter is modified by the instruction “OI field,X'80'”.
  If the “SET=” parameter is specified and the value as input is not one of the values defined, then the input will be rejected as an error. Note that if a single value is to be specified as an operand of the “SET=” parameter, it must be enclosed in two sets of parentheses as in the example above.
  Assembler language limits the total length of parameters to the “SET=” operand to 256 characters. If more “SET=” options than this are required, the “SETCONT=” parameter may be used to continue defining “SET=” options. The options specified for the “SET=” parameter and the “SETCONT=” parameter are combined to define all of the allowable values for this keyword.

SETCONT=
- one or more SET parameters
- The “SETCONT=” parameter defines a continuation of the options specified for the “SET=” parameter. See the description of the “SET=” parameter for further details.

SETYN=
- immediate instruction mask
- specifies the mask for an “Immediate” instruction to be executed if this parameter appears as an operand of the command defined on the preceding #SPZCMD macro. The “SETYN=” parameter is a special case of the “SET=” parameter described above, and causes the mask specified to be used in an “OI” instruction if the value as input is “Y” or “YES”, and to be used in a “NI” instruction if the value as input is “N” or “NO”.

PRESFLD=
- label value
- specifies the name of a field (if any) that is to be modified by the instruction specified by the “SETPRES=” parameter.

SETPRES= - SETPRES parameter
- specifies an “Immediate” instruction to be executed against the field designated by the “PRESFLD=” parameter if this operand is present in the input, and the mask to be used in this immediate instruction. There must be two sub-parameters (instruction and mask), enclosed in parentheses and separated by a comma.

VALUE= - fixed keyword value
- specifies whether the keyword defined by this #SPZKWRD macro is permitted to have an “=” sign following it, and therefore to pass a value for this keyword to the user program.
VALUE=YES (the default) specifies that this keyword must be specified as a “keyword=value” parameter.
VALUE=NO specifies that this keyword must not be specified as a “keyword=value”, the keyword must appear alone as an operand.
VALUE=OPT specifies that this keyword may or may not be specified with a value.

ACCEPT= - fixed keyword value
- specifies whether the keyword defined by this #SPZKWRD macro is currently acceptable. If “ACCEPT=NO” is specified this keyword will not be recognized until an executable #SPZKWRD macro specifying “TYPE=MOD,ACCEPT=YES” is executed.

ALIAS= - operand keyword
- specifies an alternative keyword to the main keyword that is to be treated as having the same. This facility may be used in order to accept an abbreviation of the main keyword.

TYPE= - fixed keyword value
- TYPE=MOD specifies the executable form of the #SPZKWRD macro, and will cause the generation of instructions that will dynamically modify the options on a static #SPZKWRD macro. The “NAME=” parameter is required for TYPE=MOD to specify the address of the static #SPZKWRD macro that is to be modified. The macro parameters permitted to be changed with a “TYPE=MOD” #SPZKWRD macro are “FIELD=”, “PRESFLD=”, “ACCEPT=”, “VALUE=”, “REQD=”, “FMT=”.

NAME= - label value
- specifies the name on the static #SPZKWRD macro that this #SPZKWRD macro is to modify dynamically. This parameter is meaningful only if “TYPE=MOD” is also specified, and should be accompanied by other parameters which specify the modifications to be made to the static #SPZKWRD macro.

2.27 #SPZOPEN Macro - SPZQPAM Open Macro

The #SPZOPEN macro is functionally equivalent to an Operating System "OPEN" macro, and opens one DCB for use by the Span Software SPZQPAM Queued PDS Access Method service routine.

format:

```
(name)  #SPZOPEN  dcbaddr  [,TYPE=J]
```

where:

(name) - variable symbol name
- specifies the value of a label to be placed on the first executable instruction generated by the #SPZOPEN macro.

dcbaddr - address field, (general register)
- specifies the address of the Data Control Block that is to be opened by SPZQPAM. See the Span Service Routines manual for usage information on SPZQPAM.

TYPE= - fixed character value
- TYPE=J specifies that an OPEN TYPE J is to be performed (the DCB Exit List must point to a JFCB entry in the usual way - see the appropriate Operating System Data Management manual).

2.28 #SPZPARS Macro - Invoke SPZPARSE Service Routine

The #SPZPARS macro is used to invoke the Span Software SPZPARSE Service Routine from a user program. There are “List” and “Execute” forms of the macro only, a “List” form must be defined in a non-executable area of the program, and is referred to in the “Execute” form, which will modify fields as necessary. The “Execute” form of the macro generates a “V-constant” for SPZPARSE unless the “PARSEP=” parameter is specified.

format:

```
(name)   #SPZPARS  [ERROR=erraddr]  [,EODAD=eodaddr]

                                    [       ['./']]
                   [,DATA=dataddr]  [,CCID=[     ]]
                                    [       [ccid]]

                   [       [SYSIN ]]
                   [,INDD=[ddname]]  [,LABEL=lbladdr]
                   [       [(reg) ]]

                   [,SOB=sobaddr]  [,CMDID=cmdid]

                   [,STRING=straddr]  [,STRLEN=length]

                   [,PARSEP=epaddr]  [,GLBLAD=addr]

                   [       [NONE   ]]  [         [NOPRINT]]
                   [,FLAG=[CONTROL]]  [,PRINTD=[        ]]
                   [       [DATA   ]]  [         [PRINT  ]]
                   [       [BOTH   ]]

                   [          [SPZ0]]  [         [NO ]]
                   [,MSGPREF=[xxxx]]  [,REREAD=[YES]]

                   [         [STANDARD]]
                   [,FORMAT=[AMS     ]]
                   [         [TSO     ]]

                   [       [PANV]]
                   [,LIBR=[    ]]  [,MEMBER=membername]
                   [       [LIBR]]

                      [L          ]
                  ,MF=[           ]
                      [(E,laddr)]
```

where:

(name) - variable symbol name
- symbol “name” is assigned to the first instruction generated by the “Execute” form of the macro, or to the start of the “List” form of the macro. A name field is required for the “List” form of the macro.

ERROR=

- label symbol, (register number)
- specifies the address of a routine within the user program which is to be given control if an error is detected by SPZPARSE in the input being parsed. If this parameter is not specified in either the “List” or “Execute” forms of the #SPZPARS macro, and an error is encountered, then control is returned to the instruction immediately following the “Execute” form #SPZPARS macro, with a code of 8 in register 15.

EODAD=

- label value, (register number)
- specifies the address of a routine within the user program which is to be given control if end-of-file is detected by SPZPARSE for an input control card dataset. If this parameter is not specified in either the “List” or “Execute” forms of the #SPZPARS macro, and end-of-file is encountered, then control is returned to the instruction immediately following the “Execute” form #SPZPARS macro, with a code of 12 in register 15. If SPZPARSE is invoked a further time after end-of-file has been signalled (without the “REREAD=YES” option), the end-of-file condition will be raised immediately.

DATA=

- label symbol, (register number)
- specifies the address of a routine within the user program which is to be given control if a data record is detected by SPZPARSE in the input control card dataset. If this parameter is not specified in either the “List” or “Execute” forms of the #SPZPARS macro, and a data record is encountered, then control is returned to the instruction immediately following the “Execute” form #SPZPARS macro, with a code of 4 in register 15.

CCID=

- quoted 2-character string
- specifies the two-character identifier that will appear in columns 1 and 2 to identify control records in the input. This value is only relevant when a mixture of control and data records is expected in the input (when the “DATA=” parameter is specified on either the “List” or “Execute” forms of the #SPZPARS macro). The default value is “./”. Quotes must enclose the value specified.

INDD=

- 8-character DDNAME, (register number)
- specifies the DD name to be used for the input control statement file. This DD name may be allocated to a single sequential input dataset with 80-byte logical records, to a set of concatenated sequential input datasets all with 80-byte logical records, to a partitioned input dataset of 80-byte logical records, to a CA-PANVALET dataset or to a CA-LIBRARIAN Master. For partitioned input, or for CA-PANVALET or CA-LIBRARIAN, the input member name must be specified by means of the “MEMBER=” parameter.

CA-PANVALET or CA-LIBRARIAN input must also be indicated by means of the “LIBR=” parameter. The INDD= parameter is mutually exclusive with the “STRING=” parameter, and indicates that I/O is to performed by the SPZPARSE routine to obtain input control statements. Default DDNAME if the “STRING=” parameter is omitted is “SYSIN”. If the register notation is used for this parameter (“Execute” form only), the register specified must contain the address of an 8-byte area containing the DD name required, padded on the right with blanks if necessary.

LABEL= - label symbol, (register number)

- specifies the address of an 8-byte field that is to receive the data from the “label” field of control statements. If the “LABEL=” parameter is not specified on either the “List” or “Execute” forms of the #SPZPARSE macro, then labels will not be permitted on control statements.

SOB= - label symbol, (register number)

- specifies the address of a SOB to be used for the output of print data from SPZPARSE (see elsewhere in this manual for a description of the #SPZSOB macro used to define the SOB, and see the Span Service Routines manual for a description of the SPOUTPUT service routine). If the “SOB=” parameter is not specified on either the “List” or “Execute” forms of the #SPZPARS macro, then there will be no print output from SPZPARSE. If the user program requires to intercept the print output from SPZPARSE (which consists of an echo of control statements if an input DDNAME is specified, and error messages), then the address of a user print routine may be placed in the SOB before issuing the “Execute” form of the #SPZPARS macro.

CMDID= - single-character ID

- specifies the ID character of the set of SPZPARSE definition macros to be used for this parsing operation. This value is the same as the value specified for the “ID=” parameter of the relevant #SPZCMD macro that begins the set of definition macros. If this parameter is omitted, then it is assumed that a set of definition macros is to be used where the first #SPZCMD macro had no “ID=” parameter.

STRING= - label value, (register number)

- specifies the main storage address of a parameter string to be interpreted according to the SPZPARSE rules. The length of the string should be specified with the “STRLEN=” parameter - if this parameter is not specified then the assembled length of the “STRING=” parameter is used (the “STRLEN=” parameter is required if the register form is used). This parameter overrides and replaces any “INDD=” specification.

STRLEN= - length value, (register number), max value =255
- specifies the length of the main storage string specified by the “STRING=” parameter. If the register notation is used (“Execute” form of #SPZPARS), then the register specified must contain the length of the string in its low-order byte. If this parameter is not specified then the assembled length of the “STRING=” parameter is used. The “STRLEN=” parameter is required if the register format is used for the “STRING=” parameter.

PARSEP= - label value, (register number)
- specifies the address of the entry point of the SPZPARSE service routine. If the label format is used, the field addressed must contain the address of the SPZPARSE routine. If this parameter is not specified, the macro generates a “V-constant” for SPZPARSE, and the SPZPARSE module must then be link-edited with the user program. This parameter may be specified only in the “Execute” form of the #SPZPARS macro. For a user program running under the control of SPANEX, “PARSEP=SPXMKWSC” may be specified to use SPANEX's copy of the SPZPARSE routine.

GLBLAD= - address, (register number)
- specifies the address of the Span Software SPGLBL control block CSECT. If this parameter is not specified, the “V-constant” within SPZPARSE will be used by SPZPARSE to locate SPGLBL (if SPGLBL is link-edited with SPZPARSE), or else a LOAD macro will be issued for SPGLBL. For a user program running under the control of SPANEX, “GLBLAD=SPXMGLBA” may be specified to use SPANEX's copy of the SPGLBL control block.

FLAG= - fixed keyword value
- specifies which output line types (if any) are to be flagged (high-lighted) on the output listing.
FLAG=NONE (the default) specifies that no lines are to be flagged.
FLAG=CONTROL specifies that control statements are to be flagged.
FLAG=DATA specifies that data records are to be flagged.
FLAG=BOTH specifies that both control statements and data records are to be flagged.
Flagging consists of prefixing the print line with a row of three asterisks.

PRINTD= - fixed keyword value
- specifies whether or not data records are to be printed.
PRINTD=NOPRINT (the default) specifies that data records are not to be printed.
PRINTD=PRINT specifies that data records are to be printed via the specified SOB.

MSGPREF= - four-character value
- specifies the first four characters of the message identifiers to be used on SPZPARSE error messages. SPZPARSE will add a 2-digit number and a 1-byte message type indicator to these four characters. Default is “SPZ0”. This parameter, if specified, must consist of four characters.

REREAD= - fixed keyword value
- REREAD=YES specifies that it is permissible to reread the input control statement dataset, even after the end-of-data condition has been recorded by SPZPARSE. This permits multiple uses of SPZPARSE to read from the same DD statement (which may have been re-allocated or changed to a different PDS member, for example). If the REREAD=YES option is not specified, any attempt to obtain SPZPARSE input from the same DDNAME after the end-of-data condition has been raised will produce an immediate end-of-data condition again, with no data read.

FORMAT= - fixed keyword value
- specifies the type of format user-input control statements are to take. The FORMAT= option is actioned dynamically at run-time, and it is permissible to switch between control statement formats by changing the option on successive #SPZPARS macro invocations.
FORMAT=STANDARD (the default) specifies that the control statement input is in “keyword=value” format, with parameters separated by commas, input is terminated by a blank, and continuation statements indicated by a trailing comma at the end of the previous statement.
FORMAT=AMS specifies that the control statement input is in a format similar to that used by the Access Method Services utility program (IDCAMS). Parameters are specified as keyword(value), parameters are separated by spaces, comments are bracketed by “/* . . . */”, and continuations are indicated by terminating the previous statement with a “-“ or a “+”.
FORMAT=TSO specifies that the control statement input is in a format similar to that used by the Access Method Services utility program (IDCAMS) as with the FORMAT=AMS option above. Additionally, with FORMAT=TSO any non-ambiguous abbreviation is accepted for any keyword parameter.

LIBR= - fixed keyword value
- specifies a library type for control statement input. Note that if input is to be read from a PDS member by SPZPARSE, then the member name should be specified by means of the MEMBER= parameter, and the LIBR= parameter must be omitted.
LIBR=PANV specifies that the control statement input is held as a member of a CA-PANVALET library, and that SPZPARSE is to access CA-PANVALET to obtain the input data. The CA-PANVALET member name to be used should be specified by the “MEMBER=” parameter of the #SPZPARS macro. Note that the same input member should be read until an end-of-data condition is returned by SPZPARSE. The DDNAME specified by the “INDD=” parameter is assumed to be allocated to a CA-PANVALET library.
LIBR=LIBR specifies that the control statement input is held as a member of a CA-LIBRARIAN Master, and that SPZPARSE is to access CA-LIBRARIAN to obtain the input data. The CA-LIBRARIAN member name to be used should be specified by the “MEMBER=” parameter of the #SPZPARS macro. Note that the same input member should be read until an end-of-data condition is returned by SPZPARSE. The DDNAME specified by the “INDD=” parameter is assumed to be allocated to a CA-LIBRARIAN Master.

MEMBER= - address, (register number)
- specifies the member name to be used for accessing a Partitioned Dataset, or a CA-PANVALET or CA-LIBRARIAN library. A member name must be supplied in either the List or Execute forms of the #SPZPARS macro. If this parameter is specified and the “LIBR=” parameter is not coded, then input is assumed to be from a PDS. The member name must be specified for every invocation of SPZPARSE when reading from any library type, and each input member should be read until an end-of-data condition is returned. If more than one input member is to be read in the same execution of the user program, then the “REREAD=YES” parameter must also be specified.

MF= - fixed keyword value
- specifies the macro format (“List” or “Execute” form) to be generated. This parameter is always required.
MF=L specifies the “List” form of the macro, and generates a non-executable parameter list.
MF=(E,laddr) specifies the “Execute” form of the macro, with the second sub-parameter specifying the address (in label or register notation) of the “List” form parameter list to be used.

2.29 #SPZPEND Macro - SPZPARSE Parameter Formatting

The #SPZPEND macro is used to delimit a set of SPZPARSE parameter definition macros. It must be the last SPZPARSE macro in a set. A set of SPZPARSE macros may also be delimited by a #SPZCMD macro specifying a new value for the “ID=” parameter, but the last defined set of macros must always end with the #SPZPEND macro.

format:

```
#SPZPEND
```

where:

- there are no operands

2.30 #SPZPMAP Macro - Generate SPZPARSE DSECTs

The #SPZPMAP macro is used to define the DSECTs used by the SPZPARSE Span Service Routine. The #SPZPMAP macro must be specified in any program that uses the #SPZKWRD or #SPZPOSN macros with the “TYPE=MOD” option.

format:

```
#SPZPMAP
```

where:

- there are no operands

2.31 #SPZPOSN Macro - SPZPARSE Parameter Formatting

The #SPZPOSN macro is used to define a valid positional operand of a command that was defined in a previous #SPZCMD macro statement. There may be multiple #SPZPOSN macros (and #SPZKWRD macros) for one #SPZCMD macro, but all #SPZPOSN macros should precede any #SPZKWRD macros, and any #SPZPOSN macros specifying “REQD=Y” must precede any #SPZPOSN macros defining optional positional parameters. The #SPZPOSN macro generates non-executable code that is part of a parameter list to the SPZPARSE Span Service Routine.

format:

```
(name)   #SPZPOSN  [FIELD=fieldname]  [,LENGTH=fieldlen]

                   [      [C ]]
                   [      [CN]]
                   [      [N ]]  [       [N]]
                   [,FMT=[P ]]  [,REQD=[ ]]
                   [      [B ]]  [       [Y]]
                   [      [X ]]

                   [,PRESFLD=field]  [,SETPRES=(instr,mask)]

                   [         [YES]]
                   [,ACCEPT=[   ]]
                   [         [NO ]]

                   [,TYPE=MOD,NAME=name]
```

where:

(name) - variable symbol name
- symbol “name” is assigned to the user flag field for this operand parameter, to enable the user program to interrogate the status of this operand. The field with this label will be set to zero if this operand is not present in the command, to 4 if the operand is present, and to 8 if the operand is present but is in error. Fields at certain offsets from this label may also be interrogated in order to determine the length, value and address of the data entered for this operand - these fields are mapped by the DODDS DSECT which is generated by the #SPZPMAP macro (see the Span Service Routines manual for full details). Note that the “USING” statement must be on symbol DODUFLG. The value of this “name” parameter should be used in the “TYPE=MOD” executable form of the #SPZPOSN macro by the user program before calling SPZPARSE, in order to permit the user program to be written re-entrantly (if this is required).

FIELD= - label value
- specifies the name of the field (if any) that is to be filled with the data input for the positional parameter defined by this macro.

LENGTH=
- numeric value, max=255
- specifies the length of the field that is to be filled with the data input for the positional parameter defined by this macro. If the “field” parameter is specified but the “LENGTH=” parameter is omitted, then the length taken will be the assembled length of the field defined by the “field” parameter.

FMT=
- fixed keyword value
- defines the data format of the field that is to receive the parameter data.
  FMT=C (the default) specifies character data.
  FMT=CN specifies character name data, where the first character of the value must be alphabetic (A-Z) or one of the characters #, @ or $ (the primary currency symbol).
  FMT=N specifies character numeric data.
  FMT=P specifies packed decimal (numeric input data).
  FMT=B specifies binary (numeric input data).
  FMT=X specifies hexadecimal (character hex input data).

REQD=
- fixed keyword value
- specifies whether or not this parameter is mandatory for every occurrence of the command defined by the preceding #SPZCMD macro.
  REQD=N (the default) specifies that this is an optional positional parameter
  REQD=Y specifies that this positional parameter is always required.
  Note that it is not valid to place #SPZPOSN macros specifying “REQD=Y” after #SPZPOSN macros defining optional positional parameters (it is logically unsound to have a required positional operand after an optional positional operand). If the “TYPE=MOD” feature is used in such a way that this situation is created an execution time, then SPZPARSE will treat “required” #SPZPOSN macros defined after “optional” #SPZPOSN macros as optional, and will give no warning that it has done this.

PRESFLD=
- label value
- specifies the name of a field (if any) that is to be modified by the instruction specified by the “SETPRES=” parameter.

SETPRES=
- SETPRES parameter
- specifies an “Immediate” instruction to be executed against the field designated by the “PRESFLD=” parameter if this operand is present in the input, and the mask to be used in this immediate instruction. There must be two sub-parameters (instruction and mask), enclosed in parentheses and separated by a comma.

ACCEPT=
- fixed keyword value
- specifies whether this positional parameter is currently acceptable. If “ACCEPT=NO” is specified the positional parameter represented by this #SPZPOSN macro will not be recognized until an executable #SPZPOSN macro specifying “TYPE=MOD,ACCEPT=YES” is executed.

TYPE=
- fixed keyword value
- TYPE=MOD specifies the executable form of the #SPZPOSN macro, and will cause the generation of instructions that will dynamically modify the options on a static #SPZPOSN macro. The “NAME=” parameter is required for TYPE=MOD to specify the address of the static #SPZPOSN macro that is to be modified. The macro parameters permitted to be changed with a “TYPE=MOD” #SPZPOSN macro are “FIELD=”, “PRESFLD=”, “ACCEPT=”, “REQD=”, “FMT=”.

NAME=
- label value
- specifies the name on the static #SPZPOSN macro that this #SPZPOSN macro is to modify dynamically. This parameter is meaningful only if “TYPE=MOD” is also specified, and should be accompanied by other parameters which specify the modifications to be made to the static #SPZPOSN macro.

2.32 #SPZSOB Macro - SPOUTPUT Interface Macro

The #SPZSOB macro is used to generate the SOB, the control block used by user programs to communicate with the Span Software SPOUTPUT output Service Routine. The #SPZSOB macro can generate an in-line SOB, a dynamically-obtained and formatted SOB, a SOB DSECT, instructions to modify the settings of SOB fields, and also linkage to invoke the SPOUTPUT service routine.

format:

```
                [     [DSECT ]]
                [     [STATIC]]
                [     [DYNAM ]]
(name) #SPZSOB  [TYPE=[MOD   ]]  [,BASE=reg]
                [     [FREE  ]]
                [     [CALL  ]]
                [     [LINK  ]]

          [       [adcon]]  [         [adcon]]
          [,TITLE=[     ]]  [,SUBTITL=[     ]]
          [       [(reg) ]  [         [(reg)]]

          [        [adcon]]  [          [adcon]]
          [,HEADER=[     ]]  [,FOOTING=[     ]]
          [        [(reg)]]  [          [(reg)]]

          [     ([TSO   , ... ])]
          [,RUN=([TEST  , ... ])]
          [     ([R4FSCR, ... ])]
          [     ([NOEDS , ... ])]

          [         [60]]  [         [132]  [      [adcon]]
          [,LINECNT=[  ]]  [,PRTCOLS=[   ]  [,CPPL=[     ]]
          [         [nn]]  [         [nnn]  [      [(reg)]]

          [        [SYSPRINT]]  [     [NO ]]  [      [NO ]]
          [,DDNAME=[        ]]  [,LOG=[   ]]  [,TIME=[   ]]
          [        [ddname  ]]  [     [YES]]  [      [YES]]

          [     ([WTOR,        ])]
          [     ([PRINT,       ])]  [      ([PRINT,       ])]
          [     ([TPUT,        ])]  [      ([TPUT,        ])]
          [     ([WTO,         ])]  [      ([WTO,         ])]
          [,OPT=([WTONDEL, ... ])]  [,OPT2=([WTONDEL, ... ])]
          [     ([WTOTOKN,     ])] [       ([WTOTOKN,     ])]
          [     ([PUTLINE,     ])]  [      ([PUTLINE,     ])]
          [     ([SEND,        ])]  [      ([SEND,        ])]
          [     ([FULLSCR,     ])]  [      ([FULLSCR,     ])]
          [     ([CLOSE,       ])]  [      ([CLOSE,       ])]
          [     ([NONE,        ])]  [      ([NONE,        ])]

          [        [*    ]]  [         [YES]]
          [,USERID=[(reg)]]  [,EXTEND=[   ]] [,WTOKEN=WTOtoken]
          [        [name ]]  [         [NO ]]

          [         [YES]]  [        ([SESS,     ... ])]
          [,PRTFOLD=[   ]]  [        ([JOBNAMES, ... ])]
          [         [NO ]]  [,MSGTYP=([STATUS,   ... ])]
                            [        ([BRDCST,   ... ])]
                            [        ([HRDCPY,   ... ])]
                            [        ([NOCPY,    ... ])]
[      ([THROW,   ... ])]
[,PAGE=([NOTHROW, ... ])]  [,ROUTE=(n, ... )]
[      ([RESET,       ])]
[      ([NORESET,     ])]

                [       [0    ]]
[     [adcon]]  [       [n    ]]
[,MSG=[     ]]  [,BMARG=[+n   ]]
[     [(reg)]]  [       [-n   ]]
                [       [(reg)]]

[      [STREAM]]  [         [YES]]
[,MODE=[      ]]  [,COMMENT=[   ]]
[      [INSERT]]  [         [NO ]]

[         [adcon]]  [         [adcon]]
[,WTORECB=[     ]]  [,REPAREA=[     ]]
[         [(reg)]]  [         [(reg)]]

[        [nn   ]]
[,REPLEN=[     ]]  [,NAME=name]
[        [(reg)]]

[      [adcon]]  [      [adcon]]
[,RTN1=[     ]]  [,RTN2=[     ]]
[      [(reg)]]  [      [(reg)]]

[      [adcon]]  [      [adcon]]
[,RTN3=[     ]]  [,RTN4=[     ]]
[      [(reg)]]  [      [(reg)]]

[      [adcon]]  [      [adcon]]
[,RTN5=[     ]]  [,RTN6=[     ]]
[      [(reg)]]  [      [(reg)]]

         [       [adcon]]
[,ID=x]  [,OUTEP=[     ]]
         [       [(reg)]]
```

where:

(name) - variable symbol name
- symbol “name” is assigned to the first instruction generated by the TYPE=DYNAM, TYPE=MOD, and TYPE=FREE options. It is ignored for TYPE=STATIC and TYPE=DSECT.

TYPE= - fixed keyword value
- specifies the type of macro generation to be performed.
TYPE=DSECT specifies that an assembler DSECT is to be generated defining all fields in the SOB control block.
TYPE=STATIC specifies that a SOB is to be assembled in-line at this point, initialized with all default field values, and including values defined by any other operands of this macro that are specified.
TYPE=DYNAM specifies that a dynamically-obtained SOB is to be used. If the “BASE=” operand is also specified, code is generated to GETMAIN storage for a SOB, which is then initialized from default values and from any other specified macro operands. If the “BASE=” operand is not specified, the GETMAIN is

bypassed and the SOB identified by the "NAME=" operand is assumed to be the SOB area to be initialized. The "NAME=" operand is always required for TYPE=DYNAM.

TYPE=MOD specifies that an existing SOB is to be modified with the values specified for other operands of this macro.

TYPE=FREE specifies that the dynamically-obtained SOB addressed by the register specified by the "BASE=" operand is to be FREEMAINed. This must only be done if the SOB has previously been closed.

TYPE=CALL specifies that the SPOUTPUT routine is to be invoked by means of a "CALL" process. The function of TYPE=MOD may be combined with this option by specifying the other operands of the #SPZSOB macro - the SOB fields will be modified before SPOUTPUT is invoked.

TYPE=LINK specifies that the SPOUTPUT routine is to be invoked by means of a "LINK" SVC (dynamic program fetch). The function of TYPE=MOD may be combined with this option by specifying the other operands of the #SPZSOB macro - the SOB fields will be modified before SPOUTPUT is invoked.

BASE= - general register number
- For "TYPE=DYNAM" this operand is optional, and, if specified, causes a GETMAIN to be performed for a SOB area and a "USING" statement to be generated naming the register specified in this operand and the label specified in the "NAME=" operand. See "TYPE=" operand description above.
- For "TYPE=FREE" the "BASE=" operand is required, and specifies the register that the contains the address of the SOB area that is to be FREEMAINed.

TITLE= - label, (register number)
- provides the address of a list of addresses of output lines to be used as page headings for print output. Each output line is defined by a series of #SPZFLD macros. The last address in the list must have the high-order bit set to one. The #SPZTITL macro may be used to set up this address list.

SUBTITL= - label, (register number)
- provides the address of a list of addresses of output lines to be used as page sub-headings for print output. Sub-headings are printed after headings defined by the "TITLE=" operand. Each output line is defined by a series of #SPZFLD macros. The last address in the list must have the high-order bit set to one. The #SPZTITL macro may be used to set up this address list.

HEADER= - label, (register number)
- provides the address of a list of addresses of output lines to be used as screen headings for full-screen mode output to a 3270-type terminal. Each output line is defined by a series of #SPZFLD macros. The last address in the list must have the high-order bit set to one. The #SPZTITL macro may be used to set up this address list.

FOOTING= - label, (register number)
- provides the address of a list of addresses of output lines to be used as page footings for print output. Each output line is defined by a series of #SPZFLD macros. All lines required, including any blank lines immediately following the last line of text, must be specified. The last address in the list must have the high-order bit set to one. The #SPZTITL macro may be used to set up this address list.

RUN= - fixed keyword value
- specifies some optional SPOUTPUT processing features.
RUN=TSO changes the default DDNAME for print output to “TSOPRINT” from “SYSPRINT”.
RUN=TEST merely suppresses any WTO output, regardless of output options that may be specified.
RUN=R4FSCR specifies that all full-screen output issued using this SOB is to be in the new format introduced with Release 4.0 of the Span Macros and Service Routines package. This effectively treats all #SPZFLD macros as using extended #SPZFLD options, whether or not each individual macro defines any extended options. Thus, for example, each message field defined by a #SPZFLD macro will be displayed as a separate 3270 field when the screen data is output. This option is the default, and is always used for new programs producing full-screen displays using SPOUTPUT.
RUN=NOEDS specifies that the 3270-type terminal being used for full-screen displays does not support the 3270 Extended Data Stream. If any #SPZFLD macros appear in the user program specifying screen field attributes such as colour or extended high-lighting, but the program is run on an old-style 3270, this option will prevent SPOUTPUT from issuing any Extended Data Stream control characters to the terminal.

LINECNT= - decimal number
- specifies the number of lines to be printed on the print dataset before an automatic page throw is generated (the numeric value specified must include all required heading and footing lines). The default page line-count is 60.

PRTCOLS= - decimal number, maximum=254
- specifies the maximum number of print positions to be used on the print dataset. This parameter must be specified before the first call to SPOUTPUT for this DDNAME, or it will have no effect. SPOUTPUT will write the output file with RECFM of FBA, and an LRECL one greater than the value specified for PRTCOLS. The default page width is 132 columns.

CPPL= - label, (register number)
- specifies the address of the TSO Command Processor Parameter List (CPPL) as passed from the TSO Terminal Monitor Program to the Command Processor via register 1. The CPPL address is required if OPT=PUTLINE is specified; if omitted a PUTLINE will not be performed by the SPOUTPUT routine.

DDNAME= - ddname, max length=8 bytes
- specifies the DDNAME to be used for the SPOUTPUT print output file. Default is SYSPRINT unless “RUN=TSO” is specified, when the default is TSOPRINT.

LOG= - fixed keyword value
- LOG=YES specifies that all messages are to be logged on the SPOUTPUT disk log (default DDNAME for this log is “SYSLOG”, but this is defined in the SPGLBL CSECT, assembled from the #SPGLBL macro, and may be altered by the user). If the DD statement for this log is not present, the output will not be logged. Default is LOG=NO.

TIME= - fixed keyword value
- TIME=YES specifies that the current time of day is to be added to the beginning of the output message when sent to the print dataset. Default is TIME=NO. The TIME=YES option will add 10 bytes to the length of the printed message.

OPT= - fixed keyword value(s)
- specifies one or more output destinations for message data.
OPT=WTOR specifies that the output is to be sent to the operator console via a WTOR (SVC 35), formatted as indicated by the “MSGTYP=” and “ROUTE=” operands. If “RUN=TEST” is specified, the WTOR will be suppressed. The “WTORECB=”, “REPAREA=” and “REPLEN=” operands should be used to specify the other values needed for receiving the operator's reply to a WTOR.
OPT=PRINT specifies that the output is to be directed to a print-type listing, typically a SYSOUT file, whose DDname is specified via the “DDNAME=” operand; page control for this dataset is handled automatically by SPOUTPUT based upon the values specified for the

"LINECNT=", "TITLE=", "SUBTITL=" and "FOOTING=" operands. Page skipping can be made unconditional by use of the "PAGE=" operand, or conditional on the number of lines remaining on the current page ("BMARG=" operand).

OPT=TPUT specifies that the output is to be sent to a TSO user, as specified by the "USERID=" operand; or to the current TSO user in whose address space SPOUTPUT is executing, if the "USERID=" operand is omitted.

OPT=WTO specifies that the output is to be sent to the operator console via a WTO (SVC 35), formatted as indicated by the "MSGTYP=" and "ROUTE=" operands. If "RUN=TEST" is specified, the WTO will be suppressed.

OPT=WTONDEL specifies that the output is to be sent to the operator console via a WTO (as with the "OPT=WTO" option), and that Descriptor Code 2 will be set, making the message non-deletable on DIDOCS consoles and available in response to an "*I R" command on JES3 consoles. If this option is requested, register 1 on return from SPOUTPUT contains the WTO Msg ID that may be used to "DOM" the message when necessary. If "RUN=TEST" is specified, the WTO will be suppressed.

OPT=WTOTOKN specifies that the output is to be sent to the operator console via a WTO (as with the "OPT=WTO" option), and that a WTO token is to be associated with the message. The token may be specified by means of the "WTOKEN=" operand. If the message is to be non-deletable then the WTONDEL option must be specified in addition to the WTOTOKN option (ie "OPT=(WTOTOKN,WTONDEL)" should be specified). The token may then be used to "DOM" the message when necessary. If "RUN=TEST" is specified, the WTO will be suppressed.

OPT=PUTLINE specifies that the output is to be sent to the invoking TSO user (or "SYSTSPRT" dataset for the batch TSO TMP) by means of the PUTLINE TSO service routine. This has the advantage over TPUT of being effective when the TSO Terminal Monitor Program is run in batch (MVS only). However, the Command Processor Parameter List (CPPL) address must be supplied via the "CPPL=" operand, and this is normally known only to a program invoked as a TSO Command Processor.

OPT=SEND specifies that the message is to be sent to the TSO user specified by the "USERID=" operand via the SEND operator command, using the "LOGON" option of that command. This option will be effective only when SPOUTPUT is being run under the SPANEX Span Software program product, because the authority needed to issue operator commands is not otherwise available.

OPT=FULLSCR specifies that the output is to be formatted into pages ready for sending to a 3270-type VDU. Standard support within SPOUTPUT is for full-screen TSO, but, by means of the user I/O routine options, output can be sent to the screen via any TP access method. Screen headings can be supplied by means of the “HEADER=” operand. If the output is sent to a TSO terminal, but the terminal is not a VDU, this format becomes the same as with the “OPT=TPUT” option. See the Span Service Routines manual for a discussion on the use of the various SPOUTPUT options for full-screen support.

OPT=CLOSE specifies that this is the end of current output processing, and requests that SPOUTPUT DCBs are closed, storage is freed, and that the user program is preparing to terminate.

“OPT=(FULLSCR,CLOSE)” should be specified to terminate full-screen output and to force the last screen of data to be output.

OPT=NONE specifies that no output options at all are required.

Note: Specification of the “OPT=” parameter nullifies all previous output option specifications, and remains in effect until the “OPT=” operand is respecified on another executable #SPZSOB macro instruction.

OPT2=

- fixed keyword value(s)
- specifies one or more output destinations for this message only. The “OPT2=” parameter is used to effect a temporary or “one-shot” change to the output options in effect without altering the current output options. “OPT2” options replace “OPT” options for the duration of one invocation only of SPOUTPUT, thereafter the “OPT2” options are reset to zero. Values for “OPT2” are the same as for the “OPT=” parameter with the exception of the “WTOR” option which is not supported.

USERID=

- userid (max=7 bytes), (register number), *
- specifies a TSO userid, or the location of a TSO userid.

If “TYPE=STATIC” is specified, the only valid specification for the “USERID=” parameter is the name of the TSO user, and this is the actual userid to which the output is to be sent. This combination of options is of little practical value as the actual userid is unlikely to be known at assembly time.

If “TYPE=DYNAM”, “TYPE=MOD” or “TYPE=CALL” is specified, the userid specified is the address of the actual userid.

The register format specifies a register that contains the address of the actual userid to be used.

“USERID=*” specifies that the output is to be sent to the TSO user in whose address space SPOUTPUT is executing.

EXTEND= - fixed keyword value
- EXTEND=YES specifies that if the print dataset is opened by SPOUTPUT, then the “EXTEND” option of OPEN is to be used, and data is to be added to the end of existing data in the dataset (analogous to the “DISP=MOD” JCL parameter).
EXTEND=NO specifies that the print dataset should be reused, with new data being written to the beginning of the dataset. The “EXTEND=” parameter need be used only if the SOB is to be closed and then used for further output within the same jobstep, or if an existing disk or tape dataset is to be extended with the output produced.

WTOKEN= - 4-byte WTO Token value
- specifies a WTO token to be used for WTO messages issued via this SOB. The token is any 4-byte value, and should be unique within the program. If a general register is specified for this operand, the register must contain the actual token, not the address of the token. This token is used in conjunction with the “OPT=WTOTOKN” option in order to associate a token with a WTO message so that the message or a group of messages may be deleted from the operator console by means of a DOM macro. Once specified, the token will remain in the SOB until altered by means of another #SPZSOB macro with the WTOKEN= operand specified.

PRTFOLD= - fixed keyword value
- PRTFOLD=YES (the default) specifies that all output to the print dataset is to be converted to Upper Case characters by SPOUTPUT. Printed output is not converted to upper case before being passed to the SPOUTPUT Type 1 User Exit (see the RTN1= parameter).
PRTFOLD=NO specifies that print output is to be printed exactly as passed to SPOUTPUT by the user program, with no translation of characters to be performed. This function applies to both Stream and Insert mode printed output, and to all page titles, subtitles and footings.

MSGTYP= - fixed keyword value(s)
- For “MSGTYP=SESS”, “MSGTYP=JOBNAMES” and “MSGTYP=STATUS”, see the description of the “MSGTYP” operand of the operating system “WTO” macro in the appropriate systems programming manual for your operating system.
For “MSGTYP=BRDCST”, “MSGTYP=HRDCPY” and “MSGTYP=NOCPY”, see the description of the “MCSFLAG” operand of the operating system “WTO” macro in the appropriate systems programming manual for your operating system.

PAGE=

- fixed keyword value(s)
- PAGE=THROW specifies that an unconditional page throw is to be performed on the print dataset (if OPT=PRINT is specified) on the next entry to SPOUTPUT. If OPT=FULLSCR is specified, PAGE=THROW requests that the next output line is to appear on a new screen, and will cause lines accumulated for the current screen to be written out on the next entry to SPOUTPUT.

  PAGE=RESET specifies that the print page number is to be reset to one on the next page throw.

  Note: Both these options are effective only once; they are automatically reset by SPOUTPUT when they have been actioned.

  PAGE=NOTHROW specifies that a previously-set request for a page throw is to be cancelled before an output line had been issued. This will be effective only if SPOUTPUT has not been entered since the page-throw flag was set on in the SOB by means of the "PAGE=THROW" option. Note that this will also nullify a "RESET" request unless the option "PAGE=(NOTHROW,RESET)" is specified.

  PAGE=NORESET specifies that a previously-set request for a page-number reset is to be cancelled before an output line had been issued. This will be effective only if SPOUTPUT has not been entered since the page-reset flag was set on in the SOB by means of the "PAGE=RESET" option. Note that this will also nullify a page throw request unless the option "PAGE=(NORESET,THROW)" is specified.

ROUTE=

- numeric value(s), 1-16
- specifies one or more MCS Routing Codes (numbers 1 to 16, separated by commas and enclosed in parentheses) to be used for WTO and WTOR messages. See the appropriate operating system manual for the meaning of individual route codes, some of which may be assigned by the user installation.

MSG=

- label value, (register number)
- specifies the location of the message or output to be processed by SPOUTPUT. This may be either a stream-mode message (a single string of text), or an insert-mode message consisting of a series of #SPZFLD macros describing the message content field-by-field.

BMARG=

- numeric value, numeric delta value to add or subtract from current BMARG value, (register number)
- provides a "conditional-end-of-page" function for printed output and full-screen displays. BMARG= specifies that a new output print page, or screen display page, is to be started if the current line is within the specified number of lines of the bottom of the current page. The relevant field of the SOB is reset to zero on each exit from the SPOUTPUT service

routine, and so the BMARG value must be refreshed before SPOUTPUT is invoked on each call. If the register notation is used, the specified register must contain a valid packed decimal number in its low-order two bytes - this value will be stored in the SOB without validation.

MODE= - fixed keyword value
- specifies the "mode" of the input that is provided to the SPOUTPUT service routine at the address specified by the "MSG=" operand.
MODE=STREAM specifies that each line is presented as a single string of printable characters, preceded by a single byte length field, where the length value does not include its own length. This format is generated by the #SPMSG macro.
MODE=INSERT specifies that each line is presented as a series of message field definitions, each one generated by the #SPZFLD macro. Each field contains text supplied by the user program, or a code requesting SPOUTPUT to generate some standard information (eg current time, date, jobname, etc). See the description of the #SPZFLD macro in this manual, or the Span Service Routines manual, for further information.

COMMENT=- fixed keyword value
- COMMENT=NO specifies that the block of "comments" generated in the assembly listing by the #SPZSOB macro if "TYPE=DSECT", "TYPE=DYNAM" or "TYPE=STATIC" is specified, describing the format of #SPZFLD field definitions, is to be suppressed. Comments will appear if this operand is not coded.

WTORECB= - symbolic name, (register number)
- specifies the address of a standard Operating System Event Control Block (ECB) to be passed to WTOR (SVC 35), and which is to be POSTed when the operator replies to the message. This operand is referenced only for "OPT=WTOR".

REPAREA= - symbolic name, (register number)
- specifies the address of an area to be used by WTOR (SVC 35) to return to the user program the reply data entered by the operator in response to the message sent to him. This operand is referenced only for "OPT=WTOR".

REPLEN= - numeric value, (register number)
- specifies the length of the reply permitted in response to a WTOR message. If the "REPAREA=" operand is specified without the "REPLEN=" operand, a length of 8 is assumed. This operand is referenced only for "OPT=WTOR".

NAME=
- symbolic name
- specifies the assembler name field to be placed on the SOB definition for the “TYPE=DSECT”, “TYPE=DYNAM” and “TYPE=STATIC” formats of the #SPZSOB macro. The “NAME=” operand is always required for “TYPE=DYNAM”: it specifies the name to be used in a “USING” statement for the SOB area if a GETMAIN is issued to allocate the SOB; and specifies the name of the SOB area to be initialized if a GETMAIN is not generated.

RTN1=
- symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after the output line has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is called for print format output. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

RTN2=
- symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after the output line has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is called for WTO/WTOR format output. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

RTN3=
- symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after the output line has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is called for line-mode output to a user terminal. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

RTN4=
- symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after an entire screen-full of data has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is called for full-screen mode output to a user terminal. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

RTN5= - symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after the output line has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is reserved for future use. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

RTN6= - symbolic name, (register number)
- specifies the address of an optional user output exit routine. This routine will be called by SPOUTPUT after the output line has been completely formatted, and may supplement or replace the standard output technique used by SPOUTPUT. This routine is reserved for future use. See the Span Service Routines manual for a description of SPOUTPUT user output exit routines.

ID= - single character value
- specifies a single-character SOB identifier that permits multiple SOBs to be used or generated in one assembly. All #SPZSOB macros referring to a specific SOB control block must specify the same value for the "ID=" operand. Default is a null ID. This operand is not required if only one SOB is to be used.
Note: For modules that may require to include the #SPXICB macro to define SPANEX control blocks, SPANEX reserves the values "X" and "U" for the #SPZSOB "ID=" parameter.

OUTEP= - label value, (register number)
- specifies the entry point address of the SPOUTPUT routine. If the label format is used, the field addressed must contain the address of the SPOUTPUT routine. If this parameter is not specified for a #SPZSOB macro with the "TYPE=CALL" option, the macro generates an assembler "V-constant" for SPOUTPUT, which must then be link-edited with the user program.

2.33 #SPZSUBP Macro - SPZPARSE Parameter Formatting

The #SPZSUBP macro is used to define a valid sub-parameter of the parameter that was defined in a previous #SPZKWRD macro statement. There may be multiple #SPZSUBP macros for one #SPZKWRD macro. Note that the "LENGTH" parameter of the preceding #SPZKWRD macro, and the field to which it refers, must include sufficient storage for all of the expected sub-parameters, including separating commas. The #SPZSUBP macro generates non-executable code that is part of a parameter list to the SPZPARSE Span Service Routine.

format:

```
(name) #SPZSUBP [FIELD=fieldname]  [,LENGTH=fieldlen]

                [      [C ]]
                [      [CN]]
                [      [N ]]  [       [N]]
                [,FMT=[P ]]  [,REQD=[ ]]
                [      [B ]]  [       [Y]]
                [      [X ]]

                [,VALID=(value, ...)]

                [,SET=((value,instr,mask), ...)]

                [,SETYN=mask]

                [,PRESFLD=field]  [,SETPRES=(instr,mask)]

                [         [YES]]
                [,ACCEPT=[    ]]
                [         [NO ]]

                [,TYPE=MOD,NAME=name]
```

where:

(name) - variable symbol name

- symbol "name" is assigned to the user flag field for this operand sub-parameter, to enable the user program to interrogate the status of this operand. The field with this label will be set to zero if this operand is not present in the command, to 4 if the operand is present, and to 8 if the operand is present but is in error. Fields at certain offsets from this label may also be interrogated in order to determine the length, value and address of the data entered for this operand - these fields are mapped by the DODDS DSECT which is generated by the #SPZPMAP macro (see the Span Service Routines manual for full details). Note that the "USING" statement must be on symbol DODUFLG. The value of this "name" parameter should be used in the "TYPE=MOD" executable form of the #SPZSUBP macro by the user program before calling SPZPARSE, in order to permit the user program to be written re-entrantly (if this is required).

FIELD=
- label value
- specifies the name of the field (if any) that is to be filled with the data input as an operand of the sub-parameter defined by this macro.

LENGTH=
- numeric value, max=255
- specifies the length of the field that is to be filled with the data input as a sub-parameter defined by this macro. If the “field” parameter is specified but the “LENGTH=” parameter is omitted, then the length taken will be the assembled length of the field defined by the “field” parameter.

FMT=
- fixed keyword value
- defines the data format of the field that is to receive the sub-parameter data.

  FMT=C (the default) specifies character data.

  FMT=CN specifies character name data, where the first character of the value must be alphabetic (A-Z) or one of the characters #, @ or $ (the primary currency symbol).

  FMT=N specifies character numeric data.

  FMT=P specifies packed decimal (numeric input data).

  FMT=B specifies binary (numeric input data).

  FMT=X specifies hexadecimal (character hex input data).

REQD=
- fixed keyword value
- specifies whether or not this sub-parameter is mandatory for every occurrence of the command defined by the preceding #SPZCMD macro.

  REQD=N (the default) specifies that this sub-parameter is optional.

  REQD=Y specifies that this sub-parameter is always required.

VALID=
- one or more character values
- specifies a list of values that this sub-parameter may take. If this option is specified and the parameter as input is not one of this list of values, it will be rejected as an error.

SET=
- one or more SET parameters
- specifies an operation to be performed if this sub-parameter as input takes one of a list of one or more values. The “SET=” option is specified as a sequence of one or more 3-part parameters, each of the form “(value,instruction,mask)”, where “value” is a literal value that the parameter may take, “instruction” is a valid Assembler operation code defining a System/390 or z/Architecture “Immediate” instruction, “mask” is the mask to be used by the immediate instruction. For example, if “SET=((ALL,OI,X'80'))” is specified, and the value “ALL” is input for the keyword sub-parameter,

then the field defined by the “field” parameter is modified by the instruction “OI field,X'80'”.
If the “SET=” parameter is specified and the value as input is not one of the values defined, then the input will be rejected as an error. Note that if a single value is to be specified as an operand of the “SET=” parameter, it must be enclosed in two sets of parentheses as in the example above.

SETYN= - immediate instruction mask
- specifies the mask for an “Immediate” instruction to be executed if this sub-parameter appears as an operand of the command defined on the preceding #SPZCMD macro. The “SETYN=” parameter is a special case of the “SET=” parameter described above, and causes the mask specified to be used in an “OI” instruction if the value as input is “Y” or “YES”, and to be used in a “NI” instruction if the value as input is “N” or “NO”.

PRESFLD= - label value
- specifies the name of a field (if any) that is to be modified by the instruction specified by the “SETPRES=” parameter.

SETPRES= - SETPRES parameter
- specifies an “Immediate” instruction to be executed against the field designated by the “PRESFLD=” parameter if this sub-parameter is present in the input, and the mask to be used in this immediate instruction. There must be two sub-parameters (instruction and mask), enclosed in parentheses and separated by a comma.

ACCEPT= - fixed keyword value
- specifies whether the sub-parameter defined by this #SPZSUBP macro is currently acceptable. If “ACCEPT=NO” is specified this sub-parameter will not be recognized until an executable #SPZSUBP macro specifying “TYPE=MOD,ACCEPT=YES” is executed.

TYPE= - fixed keyword value
- TYPE=MOD specifies the executable form of the #SPZSUBP macro, and will cause the generation of instructions that will dynamically modify the options on a static #SPZSUBP macro. The “NAME=” parameter is required for TYPE=MOD to specify the address of the static #SPZSUBP macro that is to be modified. The macro parameters permitted to be changed with a “TYPE=MOD” #SPZSUBP macro are “FIELD=”, “PRESFLD=”, “ACCEPT=”, “VALUE=”, “REQD=”, “FMT=”.

NAME= - label value

- specifies the name on the static #SPZSUBP macro that this #SPZSUBP macro is to modify dynamically. This parameter is meaningful only if “TYPE=MOD” is also specified, and should be accompanied by other parameters which specify the modifications to be made to the static #SPZSUBP macro.

2.34 #SPZTITL Macro - SPOUTPUT Parameter Formatting

The #SPZTITL macro is used to format data specified in the “TITLE=”, “SUBTITL=”, “HEADER=” and “FOOTING=” parameters of the #SPZSOB macro, and specifies the addresses of strings of #SPZFLD macros, one string of #SPZFLD macros for each line of title or page footing.

format:

```
(name)  #SPZTITL  (line1,line2, ... )
```

where:

(name) - variable symbol name
- specifies the value of a label to be specified by the relevant “TITLE=”, “SUBTITL=”, “HEADER=” or “FOOTING=” operand of the #SPZSOB macro.

line1,etc - label symbol
- specifies the labels of strings of #SPZFLD macros that define the format of title and footing lines. Each label is the label placed on the first #SPZFLD macro of a string.

This manual is published by

Span Software Consultants Limited

Little Moss, Peacock Lane

High Legh

Knutsford

Cheshire

WA16 6PL

England

Tel: +44/0 1565 832999

Fax: +44/0 1565 830653

Email: spanex@spansoftware.com

www.spansoftware.com

to whom all comments and suggestions should be sent.