English

FUJITSU Software BS2000

SPOOL V4.9A

Spool & Print Macros and Exits (Supplement)
Macros for Converting to PDF

User Guide

Valid for:

CONV2PDF V1.0B

Edition June 2017

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

Supplement to “Spool & Print Macros and Exits”

The PDFCVT, PDFSPO and PDFDIR macros offer the functionality to create PDF files from text files. The PDFTMP macro can be used to design PDF template pages with background pictures for output using the PDFCVT macro.

The macros originally assigned to the SPOOLSYS component now belong to the new component CONV2PDF. This changes the name of the parameter file accordingly.

CONV2PDF enhances the range of macros of BS2000/OSD ≥ V6.0.

The functionality of these macros is described below as a supplement to the **“Spool & Print Macros and Exits”** manual.

1 Macros for converting to PDF

This document describes the program interface for converting text files to PDF format. The interface is available only for P1 applications.

The table below lists the macros for the various programming languages (SPL is only available for internal purposes):

Name of the macros	Function	Page
PDFCVT (Assembler interface) PDFCVTC (CPP interface) PDFCVT.H (C-Include) PDFCVTY (COBOL interface) PDFCVTI (SPL interface internal)	Generates a PDF file through line-by-line conversion of text data. The layout of the PDF file is defined either by SPOOL parameters (such as FORM and LOOP) or through direct specification of the parameters.	page 8
PDFDIR (Assembler interface) PDFDIRC (CPP interface) PDFDIR.H (C-Include) PDFDIRY (COBOL interface) PDFDIRI (SPL interface internal)	Directly defines the layout parameters for the PDFCVT macro.	page 24
PDFSPO (Assembler interface) PDFSPOC (CPP interface) PDFSPO.H (C-Include) PDFSPOY (COBOL interface) PDFSPOI (SPL interface internal)	Defines the layout parameters for the PDFCVT macro in the form of SPOOL parameters (analogously to the PRINT-DOCUMENT command).	page 36
PDFTMP (Assembler interface) PDFTMPC (CPP interface) PDFTMP.H (C-Include) PDFTMPY (COBOL interface) PDFTMPI (SPL interface internal)	Defines a template for designing PDF pages. The PDFCVT macro enables predefined template pages of the template to be assigned to individual PDF pages.	page 46

These macros are contained in the runtime library SYSPRG.CONV2PDF.010.RTE. This library also contains the LLM (Linking Loader Module) PDFCVRT in order to link with the calling application. PDFCVRT must be specified explicitly when linkage takes place.

Note

The same rules must be used for the COBOL environment as for the Spool & Print macros, see the **Spool & Print - Macros and Exits** **(BS2000)** manual.

PDFCVT - Converting text data to PDF format

User group: Nonprivileged users
Programming languages: Assembler, C, CPP, COBOL
Macro type: S

The PDFCVT macro enables data in text format to be converted to PDF format line by line, thus permitting a PDF file to be generated. The layout of the PDF file can be defined either by specifying the parameters directly (see the PDFDIR macro) or using SPOOL parameters (see the PDFSPO macro). The pages of the PDF file can optionally be created in the layout of the template pages of a PDF template (for a definition see the PDFTMP macro).

Bookmarks can also be generated for the PDF file.

Format (assembly language)

Operation	Operands
PDFCVT	MF=C/D/E/L/M/I ,PREFIX=S / <name 1..1> ,MACID=PDF / <name 1..3> ,PARAM=<name 1..27> ,VARIANT=001 / <c-string 3..3> ,CALLER=*USER ,ACTION=*OPEN / *ADDTEXT / *ADDBOOKMARK / *ENDSECTION / *CLOSE / <var: enum-of action_set> ,OUTNAME=(*pointer*,*length*) *pointer*: *NONE / <var:pointer> / (reg:pointer) *length*: *STD / <integer:1..54> / <var: int:2> / (reg: int:2) ,WRMODE=*CREATE / *REPLACEONLY / *ANY / <var: enum-of write_mode_set>

Operation	Operands
PDFCVT	CCS=<u>*NONE</u> / <var:pointer> / (reg:pointer) ,TRUNCAT = <u>*YES</u> / *NO / <var: enum-of truncation_set> ,TEMPLAT = *NONE / <var:pointer> / (reg:pointer) ,FFORMAT = <u>*STD</u> / *SAM / *PAM / <var:pointer> / (reg:pointer) ,TEXT=(*pointer*,*length*) *pointer*: <u>*NONE</u> / <var:pointer> / (reg:pointer) *length*: <u>*STD</u> / <integer:1..32767> / <var: int:2> / (reg: int:2) ,BOOKMARK=(*pointer*,*length*) *pointer*: <u>*NONE</u> / <var:pointer> / (reg:pointer) *length*: <u>*STD</u> / <integer:1..54> / <var: int:2> / (reg: int:2) ,NXTSECT = (*name*,*namelength*) *name*: <u>*NONE</u> / <var:pointer> / (reg:pointer) *namelength*: <u>*STD</u> / <integer:1..8> / <var: int:2> / (reg: int:2) ,HANDLE=<var:pointer> / (reg:pointer) ,SPOPAR=<u>*NONE</u> / <var:pointer> / (reg:pointer) ,DIRPAR=<u>*NONE</u> / <var:pointer> / (reg:pointer) CCS=<u>*NONE</u> / <var:pointer> / (reg:pointer)

Description of the operands

ACTION=
Specifies which action is to be executed. Dependencies exist between the action and the specifications in the operands, see section “Dependencies of the operands” on page 20.

ACTION=*OPEN
Opens a PDF file, default value. The write mode (generate new file or overwrite file) depends on which value is specified for the WRMODE operand.

ACTION=*ADDTEXT
Adds a line to the PDF file. The line is initially saved. The PDF file is generated only when it is closed (ACTION=*CLOSE).

ACTION=*ADDBOOKMARK
Adds a bookmark to the PDF file. The bookmark is initially saved. The PDF file is generated only when it is closed (ACTION=*CLOSE).

ACTION=*ENDSECTION
Terminates the current page section of the PDF file. The current page is written to the current template section. The following applies for the subsequent text:

– If no template section is specified in the NXTSECT operand, the next text is written to the next template section. If no further template section exists, the text is lost.
– When a template section is specified in the NXTSECT operand, the next text is written to the specified template section. If the specified template section does not exist, PDF creation is terminated.

ACTION=*CLOSE
All the saved texts and bookmarks are written to the PDF file, and the file is then closed.

ACTION=<var: enum-of action_set>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions

1	*OPEN
2	*ADDTEXT
3	*ADDBOOKMARK
4	*CLOSE
5	*ENDSECTION

BOOKMARK=(*pointer,length*)
Specifies the name and the length of the bookmark.

pointer: ***NONE / <var: pointer> / (<reg: pointer>)**
Specifies the bookmark.

pointer: ***NONE**
No bookmark specified, default value. In this case the *ADDBOOKMARK value may not be specified for the ACTION operand.

pointer: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or of the field is not the actual value which is required, but the address of a storage location at which the value is stored (A(field) or specification of a register).

length: ***STD / <integer 1..32767> / <var: int: 2> / (<reg: int:2>)**
Specifies the length of the bookmark.

length: ***STD**
The bookmark is 80 characters long, default value.

length: **<integer 1..32767>**
Integer specifying the length of the bookmark.

length: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the length of the bookmark.

CALLER=
Caller of the macro

CALLER=*USER
The caller is the user (TU).

CCS=
Name of the character set which is to be used for the PDF file.

CCS=*NONE
No character set specified. By default the EDF03IRV character set is used.

CCS = <var:pointer> / (reg:pointer)
Address of a storage location at which the name of the character set is stored (A(field) or specification of a register).

Note

Only characters which are contained in the code table Windows CP1252 Partial (CCSN=WCP1252P) are used in the PDF file. These characters correspond to the character set CP1252 without characters such as ¼, ½, ¾, etc. Please note that characters from EBCDF041 such as ¼, ½, etc. cannot be displayed correctly.

The description of Windows CP1252 Partial is provided in the “**XHCS (BS2000)**” manual.

DIRPAR=
Specifies whether the parameters with the layout information are provided directly.

DIRPAR=*NONE
The parameters are not provided directly, default value. In this case a value which is not equal to *NONE must be specified for the SPOPAR operand.

DIRPAR=<var:pointer> / (reg:pointer)
Address of a parameter list which contains the layout information for the PDF file (A(field) or specification of a register). This parameter list is generated with the PDFDIR macro, see page 24.

FFORMAT=
Specifies the file format of the PDF file.

i An existing PDF file can be converted retroactively to the other file format using the SAM/PAM converter (which is called using the START-SAM-PAM-CONVERTER command, see the “BS2ZIP” manual).

FFORMAT=*STD
Uses the file format which is defined in the SYSPAR.CONV2PDF parameter file, default value. The parameter file is searched for at the following storage locations (search takes place in the specified order):

1. Caller’s user ID
2. TSOS user ID

If no parameter file is found, FFORMAT=*SAM applies.

i You will find the template for a parameter file with the file name SYSPAR.CONV2PDF.<version> under the installation ID of CONV2PDF.

FFORMAT=*SAM
The PDF file is created in SAM file format using REC-FORM=U.

FFORMAT=*PAM
The PDF file is created in PAM file format using BLOCK-CONTROL=NO.

FFORMAT=<var: enum-of wrmode_set>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*STD
2	*SAM
3	*PAM

HANDLE=<var:pointer> / (reg:pointer)
In the case of the ACTION=*OPEN call, specifies the address of a 4 byte long area which is aligned on a word boundary in which the identifier of the PDF converter is stored. This identifier must be specified for each subsequent call (ACTION=*ADDTEXT, *ADDBOOKMARK or *CLOSE).

MACID=<u>PDF</u> / ‘<name 1..3>‘
Specifies the second to fourth characters (inclusive) of the field names and equates.

MF=C / D / E / L / M / I
Type of the macro call. You will find more information in the “Executive Macros” manual.

NXTSECT=(*name*,*namelength*,*fname*,*fnamelength*)
Specifies the name of the next template section. There is no case sensitivity. The section must be defined in the PDF template (see the PDFTMP macro).

name:**<u>*NONE</u> / var:pointer> / (reg:pointer)**
Name of the next template section.

name: **<u>*NONE</u>**
No template section specified, default value. In this case the next section of the PDF template is used.

namelength: **<u>*STD</u> / <integer:1..8> / <var: int:2> / (reg: int:2)**
Length of the section name.

OUTNAME=(*pointer,length*)
Specifies the name of the PDF file and the length of the file name.

pointer: ***NONE / <var: pointer> / (<reg: pointer>)**
Name of the PDF file which is to be generated.

pointer: ***NONE**
*NONE is the default value and must be specified if a value not equal to *OPEN is specified for the ACTION operand.

pointer: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or of the field is not the actual value which is required, but the address of a storage location at which the value is stored (A(field) or specification of a register).

length: ***STD / <integer 1..54> / <var: int: 2> / (<reg: int:2>)**
Length of the file name.

length: ***STD**
The file name is 54 characters long, default value.

length: **<integer 1..54>**
Integer specifying the length of the file name.

length: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the length of the file name.

Examples

PDFCVT OUTNAME=(A(VAR1),41)

The name of the PDF file to be printed is stored in the field VAR1. 41 characters from this field are to be evaluated as the file name length.

PDFCVT OUTNAME=(A(VAR1))

The name of the PDF file to be printed is stored in the field VAR1. 54 characters from this field (default) are to be evaluated as the file name length.

PDFCVT OUTNAME=(A(VAR1),VAR2)

The name of the PDF file to be printed is stored in the field VAR1. The length of the file name is stored in the field VAR2.

PDFCVT OUTNAME=(*NONE,*STD)

Default: no PDF file specified. The entry for the default length is ignored. This specification is relevant only if a value which is not equal to *OPEN was specified for ACTION.

PARAM=‘<name 1..27>‘
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3). You will find more information in the „Executive Macros“ manual.

PREFIX=S / ‘<name 1..1>‘
Specifies the first character of field names and equates.

SPOPAR=
Specifies whether the parameters with the layout information for the PDF file are provided in the form of SPOOL parameters.

SPOPAR=*NONE
The parameters are not specified by means of SPOOL parameters, default value. In this case the DIRPAR operand must be supplied with a value which is not equal to *NONE.

SPOPAR=<var:pointer> / (reg:pointer)
Address of a parameter list which contains the layout information of the PDF file in the form of SPOOL parameters (A(field) or specification of a register). This parameter list is generated with the PDFSPO macro, see page 36.

TEMPLAT=
Specifies whether a PDF template is to be used to design the PDF pages. A template can be defined only when the PDF file is opened (ACTION=*OPEN).

TEMPLAT=*NONE
No PDF template is used.

TEMPLAT=<var:pointer> / (reg:pointer)
Address of a PDF template in which the parameters for designing a PDF page are defined (A(field) or specification of a register). The PDF template must be generated beforehand using the PDFTMP macros, see page 46.

TEXT=(*pointer*,*length*)
Text line which is to be added to the PDF file and the length of the line.

pointer: ***NONE / <var: pointer> / (<reg: pointer>)**
Specifies the text line.

pointer: ***NONE**
No text specified, default value. *NONE must be specified if a value which is not equal to *ADDTEXT is specified for the ACTION operand.

pointer: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or of the field is not the actual value which is required, but the address of a storage location at which the text line is stored (A(field) or specification of a register).

length: ***STD / <integer 1..32767> / <var: int: 2> / (<reg: int:2>)**
Length of the text line.

length: **<u>*STD</u>**
The text line is 80 characters long, default value.

length: **<integer 1..32767>**
Specifies an integer as the value for the length of the text line.

length: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the length of the text line.

TRUNCAT=
Specifies whether data lines which extend beyond the right margin are truncated (see also the definition of the right margin in the MARGINS operand).

TRUNCAT=<u>*YES</u>
Longer data lines are truncated, default value.

TRUNCAT=*NO
Longer data lines are wrapped. The line break takes place at the word which extends beyond the margin, a word being a string which is limited by a blank, a punctuation mark or a special character.

TRUNCAT=<var: enum-of truncation_set>
Specifies the operand value indirectly using a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the value and the required function:

1	*YES
2	*NO

VARIANT=<u>001</u> / <c-string 3..3>
Specifies the variant of the parameter list.

WRMODE=
Determines the write mode for the PDF file to be generated.

WRMODE=<u>*CREATE</u>
A new file is created, default value. If the file already exists, the operation is rejected with an error code.

WRMODE=*REPLACEONLY
The output file must already exist and is overwritten during conversion, otherwise the operation is rejected with an error code.

WRMODE=*ANY
A new output file is created. If the file already exists it is overwritten.

WRMODE=<var: enum-of wrmode_set>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

1	*CREATE
2	*REPLACEONLY
3	*ANY

Return codes

SC1	**Meaning**
0	No error
1	Parameter error
64	Correct and retry

SC2	**Meaning**
0	No error
1	Error
2	Warning

MC	**Meaning**
0	Successful processing
1	Invalid action
2	No output filename specified
3	Invalid filename length
4	Invalid write mode
5	Invalid handler
6	No layout specified (neither in DIRPAR nor in SPOPAR)
7	Both layout parameters specified (both in DIRPAR as in SPOPAR)
8	Create environment error
9	Multiple output files exists
10	Invalid output file name
11	Output file already exists
12	Output file must exist

MC	Meaning
13	Delete file failed
14	Invalid RECPART FIRST value
15	Invalid RECPART LAST value
16	Invalid RECPART range
17	Invalid LINEPP value
18	Invalid LINESPACING value
19	Invalid CCPOS value
20	Invalid PRINTER TYPE value
21	Invalid LEFT MARGIN value
22	Invalid ROTATION value
23	Invalid MEDIA value
24	Invalid PAGESIZE HEIGHT value
25	Invalid PAGESIZE WIDTH value
26	Invalid PAGESIZE internal 1
27	Invalid PAGESIZE internal 2
28	Invalid LPI value
29	Invalid FONT NAME value
30	Invalid FONT STYLE value
31	Invalid FONT SIZE value
32	Invalid TEXT parameter
33	Invalid TEXT length
34	Invalid BOOKMARK parameter
35	Invalid BOOKMARK length
37	Invalid RIGHT MARGIN value
38	Invalid TOP MARGIN value
39	Invalid BOTTOM MARGIN value
40	Work file creation failure
41	ADDTEXT internal error
42	ADDBOOKMARK internal error
43	TEXT write error
44	BOOKMARK write error
45	Work file delete error
46	Invalid call sequence

MC	Meaning
47	Version of the parameter list invalid
48	Invalid value in the TRUNCAT operand
49	Write error when ACTION=*ENDSECTION
50	Internal error when ACTION=*ENDSECTION
51	Invalid value for TEMPLAT
52	Invalid TEMPLAT length
99	Internal error
101	PDF DMS error
102	PDF open error
103	PDF file does not exist
104	PDF file exists
105	PDF internal error
106	PDF creation failed
111	Invalid recpart value
113	PDF invalid FCB type
115	PDF filename truncated
117	PDF invalid page size
118	Spool not loaded
119	Invalid CHAR parameter
121	Form and loop size inconsistency
122	Invalid FORM or LOOP
123	Bookmark is missing
208	Trace activation failed
214	PDF line truncated
220	PDF line overlapping

Dependencies of the operands

The specifications in the operands of the PDFCVT macro depend to some extent on the action which was specified in the ACTION operand. The table below shows the dependencies between the specification in ACTION and the other operands.

ACTION=	Mandatory operands	Optional operands	Ignored operands
*OPEN	OUTNAME HANDLE either SPOPAR or DIRPAR	WRMODE CCS TRUNCAT TEMPLAT	TEXT BOOKMARK
*ADDTEXT	HANDLE TEXT		OUTNAME WRMODE BOOKMARK SPOPAR DIRPAR TRUNCAT NXTSECT TEMPLAT
*ADDBOOKMARK	HANDLE BOOKMARK		OUTNAME WRMODE TEXT SPOPAR DIRPAR TRUNCAT NXTSECT TEMPLAT
*ENDSECTION	HANDLE	NXTSECT	OUTNAME WRMODE TEXT SPOPAR DIRPAR TRUNCAT TEMPLAT
*CLOSE	HANDLE		OUTNAME WRMODE TEXT BOOKMARK SPOPAR DIRPAR TRUNCAT NXTSECT TEMPLAT

Structure layout of PDFCVT (assembly language)

```
.*********************************************************************
.*  BEGIN-INTERFACE   PDFCVT
.*
.*  TITLE             (/ pdfcvt /)
.*  NAME              PDFCVT
.*  DOMAIN            CONV2PDF
.*  LANGUAGE          ASS
.*  COPYRIGHT         (C) Fujitsu Technology Solutions 2016
.*                        ALL RIGHTS RESERVED
.*  COMPILATION-SCOPE USER
.*  INTERFACE-TYPE    CALL
.*  RUN-CONTEXT       TU,
.*                    TPR
.*
.*  PURPOSE           (/ Text to PDF converter /)
.*
.*  SYNTAX            (/ Syntax Variant 1:
.*                            PDFCVT  MF = C|D|E|L|M
.*                          , PREFIX     = [S] | <name>
.*                          , MACID      = [PDF] | <name>
.*                          , PARAM      = <name 1..27>
.*                          , VARIANT    = <c-string_without_quotes 3..3> |
.*                                         default 001
.*                          , CALLER     = *USER |
.*                                         *SYSTEM |
.*                                         default *USER
.*                          , EQUATES    = [YES] | NO
.*                          , SPOPAR     = <var: pointer> |
.*                                         *NONE |
.*                                         default *NONE
.*                          , DIRPAR     = <var: pointer> |
.*                                         *NONE |
.*                                         default *NONE
.*                          , HANDLE     = <var: pointer> |
.*                                         *NONE |
.*                                         default *NONE
.*                          , TEMPLAT    = <var: pointer> |
.*                                         *NONE |
.*                                         default *NONE
.*                          , ACTION     = <var: enum-of:2 _ACTION_SET> |
.*                                         *OPEN |
.*                                         *ADDTEXT |
.*                                         *ADDBOOKMARK |
.*                                         *ENDSECTION |
.*                                         *CLOSE |
.*                                         *ADDFILE |
.*                                          *RESET |
.*                                          default *OPEN
.*                          , WRMODE      = <var: enum-of:2 _WRMODE_SET> |
.*                                          *CREATE |
.*                                          *REPLACEONLY |
.*                                          *ANY |
.*                                          default *CREATE
.*                          , TRUNCAT     = <var: enum-of:1 _TRUNCATION_SET> |
.*                                          *YES |
.*                                          *NO |
.*                                          default *YES
.*                          , FFORMAT     = <var: enum-of:1 _FILEFORMAT_SET> |
.*                                          *SAM |
.*                                          *PAM |
.*                                          *STD |
.*                                          default *STD
.*                          , CCS         = <var: char 1..8> |
.*                                          *NONE |
.*                                          *USERDEFAULT |
.*                                          default *USERDEFAULT
.*                          , OUTNAME     = list(2):
.*                                          OUTNPTR: <var: pointer> |
.*                                                   *NONE |
.*                                                   default *NONE
.*                                          OUTNLEN: <var: int 0..65535> |
.*                                                   <integer 1..54> |
.*                                                   *STD |
.*                                                   default *STD
.*                          , INNAME      = list(2):
.*                                          INNPTR: <var: pointer> |
.*                                                  *NONE |
.*                                                  default *NONE
.*                                          INNLEN: <var: int 0..65535> |
.*                                                  <integer 1..54> |
.*                                                  *STD |
.*                                                  default *STD
.*                          , NXTSECT     = list(2):
.*                                          SECNPTR: <var: pointer> |
.*                                                   *NONE |
.*                                                   default *NONE
.*                                          SECNLEN: <var: int 0..65535> |
.*                                                   <integer 1..8> |
.*                                                   *STD |
.*                                                   default *STD
.*                          , TEXT        = list(2):
.*                                          TXTPTR: <var: pointer> |
.*                                                  *NONE |
.*                                                  default *NONE
.*                                         TXTLEN: <var: int 0..65535> |
.*                                                 <integer 1..4000> |
.*                                                 *STD |
.*                                                 default *STD
.*                          , BOOKMRK    = list(2):
.*                                         BMRKPTR: <var: pointer> |
.*                                                  *NONE |
.*                                                  default *NONE
.*                                         BMRKLEN: <var: int 0..65535> |
.*                                                  <integer 1..54> |
.*                                                  *STD |
.*                                                  default *STD /)
.*
.*  REMARKS           (/ LID definition /)
.*
.**********************************************************************
```

PDFDIR - Defining the layout of the PDF file directly

User group: Nonprivileged users
Programming languages: Assembler, C, CPP, COBOL
Macro type: S

The PDFDIR macro enables you to define and save the parameters for the page layout of a PDF file in a direct form. The saved settings can be specified with the PDFCVT macro.

Format (assembly language)

Operation	Operands
PDFDIR	MF=C/D/E/L/M/I
	,PREFIX=S / <name 1..1>
	,MACID=PDF / <name 1..3>
	,PARAM=<name 1..27>
	,VARIANT=001 / <c-string 3..3>
	,RECPART=(*first*,*last*)
	first: 1 / <integer 1..32767> / <var: int:2> / (<reg: int:2>)
	last: *STD / <integer 1..32767> / <var: int:2> / (<reg: int:2>)
	,LINESP=(*spacing*,*position*)
	spacing: *SPACE_1 / *SPACE_2 / *SPACE_3 /
	*BY_ASA_CONTROL / *BY_EBCDIC_CONTROL /
	*BY_IBM_CONTROL / <var: enum-of space_set:1>
	position: 1 / <integer 1..2040> / <var: int: 2> / (<reg: int:2>)
	,PAGESZ=(*media*,*height*,*width*)
	media: *NONE / *A4 / *A4_LANDSCAPE / *A3 / *A3_LANDSCAPE /
	*A5 / *A5_LANDSCAPE / *A6 / *A6_LANDSCAPE /
	<var: enum-of pagesz_set:1>
	height: *NONE / <integer 2..2040> / <var: int:2> / (<reg: int:2>
	width: *NONE / <integer 2..2040> / <var: int:2> / (<reg: int:2>

Operation	**Operands**
PDFDIR	,MARGIN=(*left*,*right*,*top*,*bottom*) *left*: <u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>) *right*: <u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>) *top*: <u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>) *bottom*: <u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>) ,LPI=<u>6</u> / <integer 3..24> / <var: int: 2> / (<reg: int:2>) ,FONT=(*name*,*style*,*size*) *name*: <u>*COURIER</u> / *HELVETICA / *TIMES / <var: enum-of fontname_set:1> *style*: <u>*NORMAL</u> / *BOLD / *ITALIC / *BOLD_ITALIC / <var: enum-of fontstyle_set:1> *size*: <u>8</u> / <integer 1..72> / <var: int:2> / (<reg: int:2>)

Description of the operands

FONT=(*name*,*style*,*size*)
Determines the font to be used.

***name:* <u>*COURIER</u> / *HELVETICA / *TIMES / <var: enum-of fontname_set:1>**
Specifies the name of the font.

***name:* <u>*COURIER</u>**
The Courier font is used, default value.

***name:* *HELVETICA**
The Helvetica font is used.

***name:* *TIMES**
The Times font is used.

***name:* <var: enum-of fontname_set:1>**
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

0	*COURIER
1	*HELVETICA
2	*TIMES

style: ***NORMAL / *BOLD / *ITALIC / *BOLD_ITALIC / <var: enum-of fontstyle_set:1>**
Specifies the font style.

style: ***NORMAL**
The text is output in the normal font style, default value.

style: ***BOLD**
The text is output in bold.

style: ***ITALIC**
The text is output in italics.

style: ***BOLD_ITALIC**
The text is output in bold italics.

style: **<var: enum-of fontstyle_set:1>**
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

0	*NORMAL
1	*BOLD
2	*ITALIC
3	*BOLD_ITALIC

size: **8 / <integer 1..72> / <var: int:2> / (<reg: int:2>)**
Specifies the font size.

size: **8**
The font size is 8 point (pt), default value.

size: **<integer 1..72>**
Integer specifying the font size in pt.

size: **<var: int:2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the font size.

MACID=PDF / ‘<name 1..3>‘
Specifies the second to fourth characters (inclusive) of the field names and equates.

MF=C / D / E / L / M / I
Type of the macro call. You will find more information in the „Executive Macros“ manual.

LINESP=(*spacing*, *position*)
Specifies the number of line feeds, how the control character is interpreted and the position of the control character.

spacing: **<u>*SPACE_1</u> / *SPACE_2 / *SPACE_3 / *BY_ASA_CONTROL / *BY_EBCDIC_CONTROL / *BY_IBM_CONTROL / *NO / <var: enum-of space_set:1>**
Specifies the number of line feeds and how the control character is interpreted.

spacing: **<u>*SPACE_1</u>**
The text is output with 1-line spacing.

spacing: ***SPACE_2**
The text is output with 2-line spacing, i.e. an empty line is inserted after the text.

spacing: ***SPACE_3**
The text is output with 3-line spacing, i.e. two empty lines are inserted after the text.

spacing: ***BY_ASA_CONTROL**
The content of the feed control character (see *position*) is interpreted as an ASA feed control character.

spacing: ***BY_EBCDIC_CONTROL**
The content of the feed control character (see *position*) is interpreted as an EBCDIC feed control character.

spacing: ***BY_IBM_CONTROL**
The content of the feed control character (see *position*) is interpreted as an IBM feed control character.

spacing: **<var: enum-of space_set:1>**
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

1	*SPACE_1
2	*SPACE_2
4	*SPACE_3
8	*BY_EBCDIC_CONTROL
16	*BY_ASA_CONTROL
32	*BY_IBM_CONTROL

position: **1 / <integer 1..2040> / <var: int: 2> / (<reg: int:2>)**
Number of the data byte which is interpreted as a feed control character. In the case of records of variable length, the fields containing the length are not regarded as part of the data, i.e. are not counted.

This parameter is ignored if one of the values *SPACE_1, *SPACE_2 or *SPACE_3 is specified for ***spacing***.

position: **1**
The first data byte of a text line is interpreted as a feed control character, default value.

position: **<integer 1..2040>**
Integer specifying the position of the feed control character.

position:**<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the position of the feed control character.

LPI=
Defines the character density of the PDF page in lines per inch.

LPI=6
The character density is 6 lines per inch, default value.

LPI=<integer 3..24>
Integer specifying the character density.

LPI=<var: int: 2> / (<reg: int:2>)
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the line density.

MARGIN=(*left*,*right*,*top*,*bottom*)
Defines the distance to the edges of the PDF page. The values are specified in mm.

left**: 20 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Distance to the left edge of the page in mm.

left**: 20**
The distance is 20 mm, default value.

left**: <integer 0..2040>**
Integer specifying the distance.

left**: <var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the left margin.

right**: 20 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Distance to the right edge of the page in mm.

right: **<u>20</u>**
The distance is 20 mm, default value.

right: **<integer 0..2040>**
Integer specifying the distance.

right: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the right margin.

top: **<u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Distance to the top edge of the page in mm.

top: **<u>20</u>**
The distance is 20 mm, default value.

top: **<integer 0..2040>**
Integer specifying the distance.

top: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the top margin.

bottom: **<u>20</u> / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Distance to the bottom edge of the page in mm.

bottom: **<u>20</u>**
The distance is 20 mm, default value.

bottom: **<integer 0..2040>**
Integer specifying the distance.

bottom: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the bottom margin.

PAGESZ=(*media*,*height*,*width*)
Specifies the size of a PDF page.

media: **<u>*NONE</u> / *A4 / *A4_LANDSCAPE / *A3 / *A3_LANDSCAPE / *A5 / *A5_LANDSCAPE / *A6 / *A6_LANDSCAPE / <var: enum-of pagesz_set:1>**
Format of the PDF page.

media: **<u>*NONE</u>**
No format is specified, default value. In this case the size of the PDF page must be defined explicitly with the ***height*** and ***width*** parameters.

***media*: *A4 / *A4_LANDSCAPE / *A3 / *A3_LANDSCAPE / *A5 / *A5_LANDSCAPE / *A6 / *A6_LANDSCAPE**
Specifies a predefined page format:

Operand value	**DIN format**	**Width x height (mm)**
*A4	DIN A4	210 x 297
*A4_LANDSCAPE	DIN A4 landscape	297 x 210
*A3	DIN A3	297 x 420
*A3_LANDSCAPE	DIN A3 landscape	420 x 297
*A5	DIN A5	148 x 210
*A5_LANDSCAPE	DIN A5 landscape	210 x 148
*A6	DIN A6	105 x 148
*A6_LANDSCAPE	DIN A6 landscape	148 x 105

***media*: <var: enum-of pagesz_set:1>**
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

0	*NONE
1	*A4
2	*A4_LANDSCAPE
3	*A3
4	*A3_LANDSCAPE
5	*A5
6	*A5_LANDSCAPE
7	*A6
8	*A6_LANDSCAPE

***height*: *NONE / <integer 2..2040> / <var: int:2> / (<reg: int:2>**
Specifies the page height. The following must be borne in mind here:
– If a predefined page format is specified in the ***media*** parameter, *NONE is mandatory here.
– If no predefined page format is specified in the ***media*** parameter, a value which is not equal to *NONE must be specified here.

***height*: *NONE**
No page height is specified, default value.

***height*: <integer 2..2040>**
Integer specifying the page height in mm.

***height*: <var: int:2> / (<reg: int:2>**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the height of the page (in mm).

***width*: *NONE / <integer 2..2040> / <var: int:2> / (<reg: int:2>**
Specifies the page width. The following must be borne in mind here:
- If a predefined page format is specified in the ***media*** parameter, *NONE is mandatory here.
- If no predefined page format is specified in the ***media*** parameter, a value which is not equal to *NONE must be specified here.

***width*: *NONE**
No page width is specified, default value.

***width*: <integer 2..2040>**
Integer specifying the page width in mm.

***width*: <var: int:2> / (<reg: int:2>**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the width of the page (in mm).

PARAM=‘<name 1..27>‘
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3). You will find more information in the „Executive Macros“ manual.

PREFIX=S / ‘<name 1..1>‘
Specifies the first character of field names and equates.

RECPART=(*first*,*last*)
Specifies whether a text line is added to the PDF file in full (default value) or only up to a particular part. This specification enables, for example, the ISAM key or control characters to be omitted in the PDF file.

i If one of the values *BY_ASA_CONTROL, *BY_EBCDIC_CONTROL or *BY_IBM_CONTROL was specified for LINESP, the feed control character is interpreted in accordance with the position specified in LINESP(..., ***position)*** and removed from the text line. The text line is generated again without feed control characters. The specifications for ***first*** and ***last*** consequently refer to the newly generated text line, see “Examples” on page 33.

Irrespective of the specification for ***first***, the feed control character is interpreted provided the value for ***first*** is less than the length of the text line. If the specified value is greater than the length of the text line, nothing from this line is output into the PDF file.

first**: 1 / <integer 1..32767> / <var: int:2> / (<reg: int:2>)**
Specifies the byte number (record column) as of which the text line is output into the PDF file. The bytes of a text line are numbered consecutively from left to right, beginning at 1; ISAM keys and control characters are components of a text line.

first**: 1**
The output starts with the first byte of a text line, default value.

first**: <integer 1..32767>**
Integer specifying the byte number (record column) as of which the text line is output.

first**: <var: int:2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as record column.

last**: *STD / <integer 1..32767> / <var: int:2> / (<reg: int:2>)**
Specifies the byte number of the last byte which is output from a text line into the PDF file.

If a text line is longer than a form definition permits, it is truncated.

last**: *STD**
Outputs the text line to the end, default value.

last**: <integer 1..32767>**
Integer specifying the last byte which is output from a text line.

last**: <var: int:2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as last byte.

VARIANT=001 / <c-string 3..3>
Specifies the variant of the parameter list.

Examples

1. The text line is c123YYYYYYYYYYYYYYYYYYYYY, c representing the feed control character at position 1.

Specification for PDFDIR	Text in the PDF file
LINESP=(*BY_IBM_CONTROL,1) ,RECPART=(1,*STD)	123YYYYYYYYYYYYYYYYYYYYY
LINESP=(*BY_IBM_CONTROL,1) ,RECPART=(4,*STD)	YYYYYYYYYYYYYYYYYYYYY

2. The text line is 123cXXXXXXX9876543210, c representing the feed control character at position 4.

Specification for PDFDIR	Text in the PDF file
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(1,*STD)	123XXXXXXX9876543210
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(4,*STD)	XXXXXXX9876543210
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(4,10)	XXXXXXX

Structure layout of PDFDIR (assembly language)

```
.*********************************************************************
.*  BEGIN-INTERFACE   PDFDIR
.*
.*  TITLE             (/ pdfdir /)
.*  NAME              PDFDIR
.*  DOMAIN            CONV2PDF
.*  LANGUAGE          ASS
.*  COPYRIGHT         (C) Fujitsu Technology Solutions 2015
.*                        ALL RIGHTS RESERVED
.*  COMPILATION-SCOPE USER
.*  INTERFACE-TYPE    CALL
.*  RUN-CONTEXT       TU
.*
.*  PURPOSE           (/ PDF conversion based on direct parameters /)
.*
.*  SYNTAX            (/ Syntax Variant 1:
.*                            PDFDIR  MF = C|D|L|M
.*                          , PREFIX     = [S] | <name>
.*                          , MACID      = [DIF] | <name>
.*                          , VARIANT    = <c-string_without_quotes 3..3> |
.*                                         default 001
.*                          , CALLER     = *USER |
.*                                         default *USER
.*                          , EQUATES    = [YES] | NO
.*                          , RECPART    = list(2):
.*                                         FIRSTCH: <var: int 0..65535> |
.*                                                  <integer 1..32767> |
.*                                                  default: 1
.*                                         LASTCH: <var: int 0..65535> |
.*                                                 <integer 1..32767> |
.*                                                 *STD |
.*                                                 default *STD
.*                          , LINESP     = list(2):
.*                                         SPACING: <var: enum-of:1
_SPACE_SET> |
.*                                                  *SPACE_1 |
.*                                                  *SPACE_2 |
.*                                                  *SPACE_3 |
.*                                                  *BY_ASA_CONTROL |
.*                                                  *BY_EBCDIC_CONTROL |
.*                                                  *BY_IBM_CONTROL |
.*                                                  *NO |
.*                                                  default *SPACE_1
.*                                         CCPOS: <var: int 0..65535> |
.*                                                <integer 1..2040> |
.*                                                *STD |
.*                                                default *STD
.*                          , PAGESZ     = list(3):
.*                                         MEDIA: <var: enum-of:1
_PAGESZ_SET> |
.*                                                *A4 |
.*                                                *A4_LANDSCAPE |
.*                                                *A3 |
.*                                                *A3_LANDSCAPE |
.*                                                *A5 |
.*                                                *A5_LANDSCAPE |
.*                                                *A6 |
.*                                                *A6_LANDSCAPE |
.*                                                *NONE |
.*                                                default *A4
.*                                         HEIGHT: <var: int 0..65535> |
.*                                                 <integer 2..2040> |
.*                                                 *NONE |
.*                                                 default *NONE
.*                                         WIDTH: <var: int 0..65535> |
.*                                                <integer 2..2040> |
.*                                                *NONE |
.*                                                default *NONE
.*                          , FONT       = list(3):
```

```
.*                                         NAME: <var: enum-of:1
_FONTNAME_SET> |
.*                                               *COURIER |
.*                                               *HELVETICA |
.*                                               *TIMES |
.*                                               default *COURIER
.*                                         STYLE: <var: enum-of:1
_FONTSTYLE_SET> |
.*                                                *NORMAL |
.*                                                *BOLD |
.*                                                *ITALIC |
.*                                                *BOLD_ITALIC |
.*                                                default *NORMAL
.*                                         SIZE: <var: int 0..65535> |
.*                                               <integer 1..72> |
.*                                               *STD |
.*                                               default: 8
.*                          , MARGIN     = list(4):
.*                                         LEFT: <var: int 0..65535> |
.*                                               <integer 0..2040> |
.*                                               default: 20
.*                                         RIGHT: <var: int 0..65535> |
.*                                                <integer 0..2040> |
.*                                                default: 20
.*                                         TOP: <var: int 0..65535> |
.*                                              <integer 0..2040> |
.*                                              default: 20
.*                                         BOTTOM: <var: int 0..65535> |
.*                                                 <integer 0..2040> |
.*                                                 default: 20
.*                          , LPI        = <var: int 0..65535> |
.*                                         <integer 3..24> |
.*                                         default: 6 /)
.*
.*  REMARKS           (/ LID definition /)
.*
.*********************************************************************
```

PDFSPO - Defining the layout of the PDF file using SPOOL parameters

User group: Nonprivileged users
Programming languages: Assembler, C, CPP, COBOL
Macro type: S

The PDFSPO macro enables you to define and save the parameters for the page layout of a PDF file in the form of SPOOL parameters, e.g. FORM, LOOP, etc. The saved settings can be specified with the PDFCVT macro.

Format (assembly language)

Operation	Operands
PDFSPO	MF=C/D/E/L/M/I ,PREFIX=<u>S</u> / <name 1..1> ,MACID=<u>PDF</u> / <name 1..3> ,PARAM=<name 1..27> ,VARIANT=<u>001</u> / <c-string 3..3> ,RECPART=(*first*,*last*) *first*: <u>1</u> / <integer 1..32767> / <var: int:2> / (<reg: int:2>) *last*: <u>*STD</u> / <integer 1..32767> / <var: int:2> / (<reg: int:2>) ,LINEPP=<u>*STD</u> / <integer 1..32767> / <var: int: 2>/ (<reg: int:2>) ,LINESP=(*spacing*,*position*) *spacing*: <u>*SPACE_1</u> / *SPACE_2 / *SPACE_3 / *BY_ASA_CONTROL / *BY_EBCDIC_CONTROL / *BY_IBM_CONTROL / <var: enum-of space_set:1> *position*: <u>1</u> / <integer 1..2040> / <var: int: 2> / (<reg: int:2>) ,FORM=<u>*STD</u> / <var: char: 6> / (<reg: char:6>) / <c-string: alphanum-name 1..6> ,LOOP=<u>*STD</u> / <var: char: 3> / (<reg: char:3>) / <c-string: alphanum-name 1..3>

Operation	Operands
PDFSPO	,CHARSET=*STD / <var: char: 3> / (<reg: char:3>) / <c-string: alphanum-name 1..3> ,PRTYPE=*HP90 / *HP / *LP / <var: enum-of prtype_set:1> ,LEFTMAR=20 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>) ,ROT=*NO / *YES / <var: enum-of rotation_set:1>

Description of operands

CHARSET=
Specifies the font which is used for the PDF file.
Irrespective of the specification for CHARSET, the Courier font is always used in the PDF file. Only the WEIGHT, CHARACTER-STYLE and LINE-PER-INCH parameters are interpreted.
The combination of WEIGHT and CHARACTER-STYLE determines the font style which is used in the PDF file:

WEIGHT	Font style with CHARACTER-STYLE =*ITALICS	Font style with CHARACTER-STYLE =other values
*BOLD	*BOLD-ITALICS	*BOLD
other values	*ITALICS	*NORMAL

CHARSET=*STD
The standard font is used which is determined by the specified printer type (PRTYPE) and, if necessary, from the required format (FORM). This font can be queried with SHOW-SPOOL-FORMS.

CHARSET=<c-string: alphanum-name 1..3>
Names of the font used in the PDF file. The string must be enclosed in quotes.

CHARSET=<var: char: 3> / (<reg: char:3>)
Name of a field defined with CL or a register containing the value. A string (3 bytes in length) is stored in this field or register and interpreted as the name of the font.

FORM=
Specifies the paper (form) to be used for output (e.g. STD, STDSF1, STDWA4).
Default forms must be defined in the SPOOL parameter file for all printer types.
By means of the SHOW-SPOOL-FORMS command you can output the entries to SYSOUT.
Only forms for the printer types HP, HP90 and LP are supported. The printer type must be specified in the PRTYPE operand to ensure that the correct form is used.

FORM=*STD
Default form.

FORM='<c-string: alphanum-name 1..6>'
Name of the form with which the SPOOLOUT job is to be processed.
A loop is specified implicitly with the form specification. If the operand ROT=*YES is specified, a rotation loop is also specified implicitly for page rotation.
The assigned loop (or PAGEDEF and FORMDEF) must be contained in the printer control file $SYSSPOOL.PRFILE.
The loop which is named implicitly using the FORM operand or the rotation loop is ignored if the LOOP operand is also specified.
Without the FORM and LOOP operands the PDF file is created with the standard form entered for the printer type concerned.
A loop specified explicitly with the LOOP operand must be the same length as the loop assigned to the form used.

FORM=<var: char: 6> / (<reg: char:6>)
Name of a field defined with CL or a register containing the value. A string (6 bytes in length) is stored in this field or register and interpreted as the name of the form.

MACID=PDF / ‘<name 1..3>‘
Specifies the second to fourth characters (inclusive) of the field names and equates.

MF=C / D / E / L / M / I
Type of the macro call. You will find more information in the „Executive Macros“ manual.

LEFTMAR=
Specifies whether the output text is to be indented. The value is specified in mm.

LEFTMAR=20
The indent is 20mm, default value.

LEFTMAR=<integer 0..2040>
Integer specifying the indent in mm.

LEFTMAR=<var: int: 2> / (<reg: int:2>)
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as a left margin.

LINEPP=
Defines the number of lines which a page in the PDF file may contain.

LINEPP=<u>*STD</u>
If the operand is omitted, the number of lines per print page is calculated using the following formula:

Number of lines = P * L - N - 6

Where:
P = paper size in inches
L = line density
N = number of lines before the first vertical tab “channel 1”. "

Note

– The vertical tab "channel 1" controls the line on which printing is to start. Unless otherwise specified, 2 blank lines are set before printing starts; i.e. channel 1 (CHANNEL 01) is in the third line of the loop.

– If the value specified for the LINEPP operand is greater than the specified number of lines in the loop, the value in the loop is used.

– If LINEPP is specified together with the LINESP=*SPACE_1, *SPACE_2 or *SPACE_3 operand, the value specified for LINEPP must be at least three times that of the line feed specified for LINESP.

– The LINEPP parameter is ignored if one of the values *BY-EBCDIC-CONTROL, *BY-ASA-CONTROL or *BY-IBM-CONTROL is specified for LINESP.

LINEPP=<integer 1..32767>
Number of lines on a page.

LINEPP=<var: int: 2> / (<reg: int:2>)
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as number of lines per page.

LINESP=(*spacing*, *position*)
Specifies the number of line feeds, how the control character is interpreted and the position of the control character.

spacing: **<u>*SPACE_1</u> / *SPACE_2 / *SPACE_3 / *BY_ASA_CONTROL / *BY_EBCDIC_CONTROL / *BY_IBM_CONTROL / *NO / <var: enum-of space_set:1>**
Specifies the number of line feeds and how the control character is interpreted.

spacing: **<u>*SPACE_1</u>**
The text is output with 1-line spacing, default value.

spacing: ***SPACE_2**
The text is output with 2-line spacing.

spacing: ***SPACE_3**
The text is output with 3-line spacing.

spacing: ***BY_ASA_CONTROL**
The content of the feed control character (see ***position***) is interpreted as an ASA feed control character.

spacing: ***BY_EBCDIC_CONTROL**
The content of the feed control character (see ***position***) is interpreted as an EBCDIC feed control character.

spacing: ***BY_IBM_CONTROL**
The content of the feed control character (see ***position***) is interpreted as an IBM feed control character.

spacing: **<var: enum-of space_set:1>**
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

1	*SPACE_1
2	*SPACE_2
4	*SPACE_3
8	*BY_EBCDIC_CONTROL
16	*BY_ASA_CONTROL
32	*BY_IBM_CONTROL

position: **1 / <integer 1..2040> / <var: int: 2> / (<reg: int:2>)**
Number of the data byte which is interpreted as a feed control character. In the case of records of variable length, the fields containing the length are not regarded as part of the data, i.e. are not counted.

This parameter is ignored if *SPACE_1, *SPACE_2, *SPACE_3 or *NO is specified for ***spacing***.

position: **1**
The first data byte of a text line is interpreted as a feed control character, default value.

position: **<integer 1..2040>**
Integer specifying the position of the feed control character.

position: **<var: int: 2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as the position of the feed control character.

LOOP=
Name of the loop which is used. The loop name may not contain the characters ’$’, ’&’ and ’@’.

LOOP=*STD
The feed control character for the PDF file is implemented with the standard loop of the form used, default value.

LOOP=<c-string: alphanum-name 1..3>
Name of the loop which is to control the feed. The length of the specified loop must match the length of the standard loop of the form used.
Loops are stored in the printer control file PRFILE. If no loop is specified, the implicit specifications in the FORM operand are used.

If the FORM or LOOP operand is omitted, default values apply.

LOOP=<var: char: 3> / (<reg: char:3>)
Name of a field defined with CL or a register containing the value. An integer (3 bytes in length) is stored in this field or register and interpreted as loop name.

PARAM=‘<name 1..27>‘
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3). You will find more information in the „Executive Macros“ manual.

PREFIX=S / ‘<name 1..1>‘
Specifies the first character of field names and equates.

PRTYPE=
Specifies the printer type whose form is to be used for the layout of the PDF file.

PRTYPE=*HP90
The form for printer type HP90 is used, default value.

PRTYPE=*HP
The form for printer type HP is used.

PRTYPE=*LP
The form for printer type LP is used.

PRTYPE=<var: enum-of prtype_set:1>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

1	*HP90
2	*HP
3	*LP

RECPART=(*first*,*last*)
Specifies whether a text line is added to the PDF file in full (default value) or only up to a particular part. This specification enables, for example, the ISAM key or control characters to be omitted in the PDF file.

i If one of the values *BY_ASA_CONTROL, *BY_EBCDIC_CONTROL or *BY_IBM_CONTROL was specified for LINESP, the feed control character is interpreted in accordance with the position specified in LINESP(..., ***position)*** and removed from the text line. The text line is generated again without feed control characters. The specifications for ***first*** and ***last*** consequently refer to the newly generated text line, see “Examples” on page 33.

Irrespective of the specification for ***first***, the feed control character is interpreted provided the value for ***first*** is less than the length of the text line. If the specified value is greater than the length of the text line, nothing from this line is output into the PDF file.

***first*: 1 / <integer 1..32767> / <var: int:2> / (<reg: int:2>)**
Specifies the byte number (record column) as of which the text line is output into the PDF file. The bytes of a text line are numbered consecutively from left to right, beginning at 1; ISAM keys and control characters are components of a text line.

***first*: 1**
The output starts with the first byte of a text line, default value.

***first*: <integer 1..32767>**
Integer specifying the byte number (record column) as of which the text line of is output.

***first*: <var: int:2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as a record column.

***last*: *STD / <integer 1..32767> / <var: int:2> / (<reg: int:2>)**
Specifies the byte number of the last byte which is output from a text line into the PDF file.

If a text line is longer than a form definition permits, it is truncated.

***last*: *STD**
Outputs the text line to the end, default value.

***last*: <integer 1..32767>**
Integer specifying the last byte which is output from a text line.

***last*: <var: int:2> / (<reg: int:2>)**
Name of a field defined with FL or a register containing the value. An integer (2 bytes in length) is stored in this field or register and interpreted as last byte.

ROT=*NO / *YES / <var: enum-of rotation_set:1>
Specifies whether or not the pages are output rotated.

ROT=*NO
No page rotation, default value. The loop defined in the form is used as the loop.

ROT=*YES
The page is rotated 90 degrees clockwise. The loop defined in the form is used as the loop.

ROT=<var: enum-of rotation_set:1>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationships exist between the values and the desired functions:

0	*NO
1	*YES

VARIANT=001 / <c-string 3..3>
Specifies the variant of the parameter list.

Examples

1. The text line is c123YYYYYYYYYYYYYYYYYYYYY, c representing the feed control character at position 1.

Specification for PDFSPO	**Text in the PDF file**
LINESP=(*BY_IBM_CONTROL,1) ,RECPART=(1,*STD)	123YYYYYYYYYYYYYYYYYYYYY
LINESP=(*BY_IBM_CONTROL,1) ,RECPART=(4,*STD)	YYYYYYYYYYYYYYYYYYYYY

2. The text line is 123cXXXXXXX9876543210, c representing the feed control character at position 4.

Specification for PDFSPO	**Text in the PDF file**
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(1,*STD)	123XXXXXXX9876543210
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(4,*STD)	XXXXXXX9876543210
LINESP=(*BY_IBM_CONTROL,4) ,RECPART=(4,10)	XXXXXXX

Structure layout of PDFSPO (assembly language)

```
.*******************************************************************
.*  BEGIN-INTERFACE   PDFSPO
.*
.*  TITLE             (/ pdfspo /)
.*  NAME              PDFSPO
.*  DOMAIN            CONV2PDF
.*  LANGUAGE          ASS
.*  COPYRIGHT         (C) Fujitsu Technology Solutions 2015
.*                        ALL RIGHTS RESERVED
.*  COMPILATION-SCOPE USER
.*  INTERFACE-TYPE    CALL
.*  RUN-CONTEXT       TU
.*
.*  PURPOSE           (/ PDF conversion based on SPOOL parameters /)
.*
.*  SYNTAX            (/ Syntax Variant 1:
.*                            PDFSPO  MF = C|D|L|M
.*                          , PREFIX     = [S] | <name>
.*                          , MACID      = [POF] | <name>
.*                          , VARIANT    = <c-string_without_quotes 3..3> |
.*                                         default 001
.*                          , CALLER     = *USER |
.*                                         default *USER
.*                          , EQUATES    = [YES] | NO
.*                          , ROT        = <var: enum-of:1 _ROTATION_SET> |
.*                                         *NO |
.*                                         *YES |
.*                                         default *NO
.*                          , PRTYPE     = <var: enum-of:1 _PRTYPE_SET> |
.*                                         *HP90 |
.*                                         *HP |
.*                                         *LP |
.*                                         default *HP90
.*                          , RECPART    = list(2):
.*                                         FIRSTCH: <var: int 0..65535> |
.*                                                  <integer 1..32767> |
.*                                                  default: 1
.*                                         LASTCH: <var: int 0..65535> |
.*                                                 <integer 1..32767> |
.*                                                 *STD |
.*                                                 default *STD
.*                          , LINESP     = list(2):
.*                                         SPACING: <var: enum-of:1
_SPACE_SET> |
.*                                                  *SPACE_1 |
.*                                                  *SPACE_2 |
.*                                                 *SPACE_3 |
.*                                                 *BY_ASA_CONTROL |
.*                                                 *BY_EBCDIC_CONTROL |
.*                                                 *BY_IBM_CONTROL |
.*                                                 default *SPACE_1
.*                                        CCPOS: <var: int 0..65535> |
.*                                               <integer 1..2040> |
.*                                               *STD |
.*                                               default *STD
.*                        , LEFTMAR    = <var: int 0..65535> |
.*                                        <integer 0..2040> |
.*                                        default: 20
.*                        , LINEPP     = <var: int 0..65535> |
.*                                        <integer 1..32767> |
.*                                        *STD |
.*                                        default *STD
.*                        , FORM       = <var: char 1..6> |
.*                                        <c-string 1..6> |
.*                                        *STD |
.*                                        default *STD
.*                        , LOOP       = <var: char 1..3> |
.*                                        <c-string 1..3> |
.*                                        *STD |
.*                                        default *STD
.*                        , CHARSET    = <var: char 1..3> |
.*                                        <c-string 1..3> |
.*                                        *STD |
.*                                        default *STD /)
.*
.*  REMARKS           (/ LID definition /)
.*
.*********************************************************************
```

PDFTMP - Defining the template for PDF pages

User group: Nonprivileged users
Programming languages: Assembler, C, CPP, COBOL
Macro type: S

The PDFTMP macro enables you to define and save the design of PDF pages in the form of templates. Here a defined PDF template page corresponds to a template section. The PDFCVT macro enables you to use a template page of a template section when creating the PDF pages in a page section.

Format (Assembler)

Operation	Operands
PDFTMP	MF=C/D/E/L/M/I ,PREFIX=S / <name 1..1> ,MACID=PDF / <name 1..3> ,PARAM=<name 1..27> ,VARIANT=001 / <c-string 3..3> ,ACTION=*OPEN / *ADDOVERLAY / *ADDSECTION / *ATTOVERLAY / <var: enum-of action_set> ,TMPNAME=(tmpnptr,tmpnlen) tmpnptr: *NONE / <var:pointer> / (reg:pointer) tmpnlen: *STD / <integer:1..8> / <var: int:2> / (reg: int:2) ,BOOKMRK=*NO / *YES / <var: enum-of bookmarkuse_set:1> ,OVERLAY = (name,namelength,fname,fnamelength) name: *NONE / <var:pointer> / (reg:pointer) namelength: *STD / <integer:1..8> / <var: int:2> / (reg: int:2) fname: *NONE / <var:pointer> / (reg:pointer) fnamelength: *STD / <integer:1..54> / <var: int:2> / (reg: int:2)

Operation	Operands
PDFTMP	,SECTION = (*secnptr*,*secnlen*)
	secnptr: *NONE / <var:pointer> / (reg:pointer)
	secnlen: *STD / <integer:1..8> / <var: int:2> / (reg: int:2)
	,OVLLOC=(*name*,*namelength*,*mediabox*,*left*,*right*,*top*,*bottom*,*halign*,*valign*)
	name: *NONE / <var:pointer> / (reg:pointer)
	namelength: *STD / <integer:1..8> / <var: int:2> / (reg: int:2)
	mediabox: *PAGE / *TEXT / *CUSTOM / <var: enum-of ovl_mediabox_set:1>
	left: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)
	right: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)
	top: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)
	bottom: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)
	halign: *LEFT / *RIGHT / *CENTER / *FIT / <var: enum-of ovl-alignement-set:1>
	valign: *TOP / *BOTTOM / *CENTER / *FIT / <var: enum-of ovl-alignement-set:1>
	,TOITEM = (*itemtyp*,*itemptr*,*itemlen*)
	itemtyp: *SECTION / <var: enum-of tmpitem_set:1>
	itemptr: *NONE / <var:pointer> / (reg:pointer)
	itemlen: *STD / <integer:1..8> / <var: int:2> / (reg: int:2)
	,HANDLE=<var:pointer> / (reg:pointer)

Operand description

ACTION=
Specifies which action is to be performed. Dependencies exist between the action and the specifications in other operands, see the section “Dependencies of the operands” on page 57.

ACTION=*OPEN
Creates a PDF template, default value. The name must be specified in the TMPNAME operand.

ACTION=*ADDOVERLAY
Defines a background picture. To enable the binary data of the picture file to be transferred to the PDF file, the background picture must be assigned to a template section using the *ATTOVERLAY action.

ACTION=*ADDSECTION
Adds a new section to the PDF template.

ACTION=*ATTOVERLAY
Positions a background picture on the PDF page and assigns the page to a template section.

ACTION=<var: enum-of action_set>
The action is not specified directly by means of an operand value; instead, it is specified indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*OPEN
2	*ADDOVERLAY
3	*ADDSECTION
4	*ATTOVERLAY

BOOKMRK=
Specifies the use of bookmarks:
– When ACTION=*OPEN, this entry specifies whether the PDF file will be created with or without bookmarks.
– When ACTION=*ADDSECTION, this entry specifies whether bookmarks will be accepted in the specified template section.

BOOKMRK=<u>*NO</u>
No bookmarks are used, default value.

BOOKMRK=*YES
Bookmarks are used. When a template section accepts bookmarks, a bookmark is set automatically at the start of this page section.

BOOKMRK=<var: enum-of bookmarkuse_set:1>
Specifies the operand value indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*NO
2	*YES

HANDLE=<var:pointer> / (reg:pointer)
In the case of the ACTION=*OPEN call, specifies the address of a 4 byte long area which is aligned on a word boundary in which the identifier of the PDF converter is stored. This identifier must be specified for each subsequent call (ACTION=*ADDOVERLAY, *ADDSECTION or *ATTOVERLAY).

MACID=PDF / ‘<name 1..3>‘
Specifies the second to fourth characters (inclusive) of the field names and equates.

MF=C / D / E / L / M / I
Type of the macro call. You will find more information in the “Executive Macros” manual.

OVERLAY = (*name,namelength,fname,fnamelength*)
Defines a background picture with overlay name and name length, plus the file name of the picture file and the length of the file name.

i Only pictures in JPG format are supported. Transfer of the pictures from the PC to BS2000 must take place in binary format.

***name*: *NONE / <var:pointer> / (reg:pointer)**
Specifies the overlay name.

***name*: *NONE**
No overlay name is specified, default value.

***name*: <var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or field is not the required value itself, but the address of a storage location at which the value is stored (A(field) or specification of a register).

***namelength*: *STD / <integer:1..8> / <var: int:2> / (reg: int:2)**
Specifies the length of the overlay name.

***namelength*: *STD**
A length of 8 characters is accepted, default value.

***namelength*: <integer 1..8>**
Specifies an integer value for the length of the overlay name.

***namelength*: <var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the length of the overlay name is stored in this field or register.

fname: ***NONE / <var:pointer> / (reg:pointer)**
Specifies the file name.

fname: ***NONE**
No file name is specified, default value.

fname: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or field is not the required value itself, but the address of a storage location at which the value is stored (A(field) or specification of a register).

fnamelength: ***STD / <integer:1..54> / <var: int:2> / (reg: int:2)**
Specifies the length of the file name.

fnamelength: ***STD**
The file name is 54 characters long, default value.

fnamelength: **<integer 1..54>**
Specifies an integer value for the length of the file name.

fnamelength: **<var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the length of the file name is stored in this field or register.

OVLLOC=(*name*,*namelength*,*mediabox*,*left*,*right*,*top*,*bottom*,*halign*,*valign*)
Defines the position of the background picture on the PDF page. The overlay name and name length, the picture frame, the distances to the frame (left, right, top, bottom) and the horizontal and vertical alignment in the frame must be specified.

name: ***NONE / <var:pointer> / (reg:pointer)**

namelength: ***STD / <integer:1..8> / <var: int:2> / (reg: int:2)**

mediabox: ***PAGE / *TEXT / *CUSTOM**
Determines the frame in which the background picture is positioned.

mediabox: ***PAGE**
The background picture is positioned within the physical PDF page (determined by the PAGEZ operand in the PDFDIR macro).

mediabox: ***TEXT**
The background picture is positioned within the text frame (determined by the MARGIN operand in the PDFDIR macro).

mediabox: ***CUSTOM**
Defines a frame by means of the distances to the margins. This frame is independent of the defined text frame.

mediabox: **<var: enum-of ovl_mediabox_set:1>**
Specifies the operand value indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*PAGE
2	*TEXT
3	*CUSTOM

left: **0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Only when media box is specified with *CUSTOM:
Distance to the left margin in mm.

left: **0**
The distance is 0 mm, default value.

left: **<integer 0..2040>**
Integer value for the distance.

left: **<var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the distance to the left margin is stored in this field or register.

right: **0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Only when media box is specified with *CUSTOM:
Distance to the right margin in mm.

right: **0**
The distance is 0 mm, default value.

right: **<integer 0..2040>**
Integer value for the distance.

right: **<var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the distance to the right margin is stored in this field or register.

***top*: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Only when media box is specified with *CUSTOM:
Distance to the top margin in mm.

***top*: 0**
The distance is 0 mm, default value.

***top*: <integer 0..2040>**
Integer value for the distance.

***top*: <var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the distance to the top margin is stored in this field or register.

***bottom*: 0 / <integer 0..2040> / <var: int: 2> / (<reg: int:2>)**
Only when media box is specified with *CUSTOM:
Distance to the bottom margin in mm.

***bottom*: 0**
The distance is 0 mm, default value.

***bottom*: <integer 0..2040>**
Integer value for the distance.

***bottom*: <var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the distance to the bottom margin is stored in this field or register.

***halign*: *LEFT / *RIGHT / *CENTER / *FIT / <var: enum-of ovl-alignement-set:1>**
Determines the horizontal alignment of the picture within the frame.

***halign*: *LEFT**
The picture is aligned with the left side of the frame.

***halign*: *RIGHT**
The picture is aligned with the right side of the frame.

***halign*: *CENTER**
The picture is centered horizontally in the frame.

***halign*: *FIT**
The picture is fitted to the width of the frame. If the relationships of scale of the frame and picture differ, the picture may be distorted.

***halign*: <var: enum-of ovl-alignement-set:1>**
Specifies the operand value indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*LEFT
2	*RIGHT
3	*CENTER
4	*FIT

***valign*: *TOP/ *BOTTOM / *CENTER / *FIT / <var: enum-of ovl-alignement-set:1>**
Determines the vertical alignment of the picture within the frame.

***valign*: *TOP**
The picture is aligned with the top, default value.

***valign*: *BOTTOM**
The picture is aligned with the bottom.

***valign*: *CENTER**
The picture is centered vertically in the frame.

***valign*: *FIT**
The picture is fitted to the height of the frame. If the relationships of scale of the frame and picture differ, the picture may be distorted.

***valign*: <var: enum-of ovl-alignement-set:1>**
Specifies the operand value indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*TOP
2	*BOTTOM
3	*CENTER
4	*FIT

PARAM=‘<name 1..27>‘
Specifies the address of the operand list (permitted only with MF formats 2 and 3). You will find more information in the “Executive Macros” manual.

PREFIX=S / ‘<name 1..1>‘
Specifies the first characters of the field names and equates.

SECTION = (*secnptr,secnlen*)
Specifies the template section together with its name and name length.

secnptr: ***NONE / <var:pointer> / (reg:pointer)**
Specifies the name of the template section.

secnptr: ***NONE**
No section name is specified, default value.

secnptr: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or field is not the required value itself, but the address of a storage location at which the value is stored (A(field) or specification of a register).

secnlen: ***STD / <integer:1..8> / <var: int:2> / (reg: int:2)**
Specifies the length of the section name.

secnlen: ***STD**
The section name is 8 characters long, default value.

secnlen: **<integer 1..8>**
Specifies an integer value for the length of the section name.

secnlen: **<var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the length of the section name is stored in this field or register.

TMPNAME=(*tmpnptr*,*tmpnlen*)
Specifies the template name and the length of the template name.

tmpnptr: ***NONE / <var:pointer> / (reg:pointer)**
Specifies the template name.

tmpnptr: ***NONE**
No template name is specified, default value. In this case the value *OPEN may not be specified for the ACTION operand.

tmpnptr: **<var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or field is not the required value itself, but the address of a storage location at which the value is stored (A(field) or specification of a register).

tmpnlen: ***STD / <integer:1..8> / <var: int:2> / (reg: int:2)**
Specifies the length of the template name.

tmpnlen: ***STD**
The template name is 8 characters long, default value.

tmpnlen: **<integer 1..8>**
Specifies an integer value for the length of the template name.

***tmpnlen*: <var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the length of the template name is stored in this field or register.

TOITEM = (*itemtyp*,*itemptr*,*itemlen*)
Item to which the background picture will be assigned.

***itemtyp*: *SECTION / <var: enum-of tmpitem-set:1>**
Specifies the item type.

i At present only the item type *SECTION, i.e. assignment to a template section, is possible.

***itemtyp*: <var: enum-of tmpitem-set:1>**
Specifies the operand value indirectly by means of a field with constant contents (equate). An integer can be stored in the constant or the corresponding field. The following relationship exists between the values and the desired functions:

1	*SECTION

***itemptr*: *NONE / <var:pointer> / (reg:pointer)**
Specifies the name of the item.

***secnptr*: *NONE**
No item name is specified, default value.

***secnptr*: <var: pointer> / (<reg: pointer>)**
A pointer is defined, i.e. the content of the variable or field is not the required value itself, but the address of a storage location at which the value is stored (A(field) or specification of a register).

***secnlen*: *STD / <integer:1..8> / <var: int:2> / (reg: int:2)**
Specifies the length of the item name.

***secnlen*: *STD**
The item name is 8 characters long, default value.

***secnlen*: <integer 1..8>**
Specifies an integer value for the length of the item name.

***secnlen*: <var: int: 2> / (<reg: int:2>)**
Name of a field which is defined with FL or a register which contains the value. An integer (length: 2 bytes) which is interpreted as the length of the item name is stored in this field or register.

VARIANT=001 / <c-string 3..3>
Specifies the variant of the generated parameter list.

Return codes

SC1	Meaning
0	No error
1	Parameter error
64	Correct and retry

SC2	Meaning
0	No error
1	Error
2	Warning

MC	Meaning
0	Successful processing
1	Invalid action
2	Invalid identifier (HANDLE operand)
3	Invalid operand OVERLAY
4	Invalid address of the overlay name
5	Invalid overlay name length
6	Invalid picture file name address
7	Invalid picture file name length
8	Write error when ACTION=*ADDOVERLAY
9	Internal error when ACTION=*ADDOVERLAY
10	Invalid picture frame (media box)
11	Invalid value for horizontal alignment
12	Invalid value for vertical alignment
13	Invalid parameters for user-specific picture frame (media box *CUSTOM)
14	Write error when ACTION=*ATTOVERLAY
15	Internal error when ACTION=*ATTOVERLAY
16	Parameter list version invalid
17	Not used
18	Picture file not found
19	Error when reading the picture file
20	Picture is not in JPG format
21	Invalid function header

MC	**Meaning**
22	Invalid sequence
23	Invalid address for template section name
24	Invalid length for template section name
25	No PDF template specified
26	Invalid bookmark indicator
27	Name of the template section invalid
28	Template section already exists
99	Internal error

Dependencies of the operands

The specifications in the operands of the PDFTMP macro depend to some extent on the action which was specified in the ACTION operand. The table below shows the dependencies between the specification in ACTION and the other operands.

ACTION=	**Mandatory operands**	**Optional operands**	**Ignored operands**
*OPEN	TMPNAME	BOOKMRK	OVERLAY OVLLOC SECTION TOITEM
*ADDOVERLAY	HANDLE OVERLAY		OVLLOC SECTION TOITEM BOOKMRK
*ADDSECTION	HANDLE SECTION	BOOKMRK	OVERLAY OVLLOC TOITEM
*ATTOVERLAY	HANDLE OVLLOC TOITEM		OVERLAY BOOKMRK SECTION

Structure layout of PDFTMP (assembly language)

```
.**********************************************************************
.*  BEGIN-INTERFACE   PDFTMP
.*
.*  TITLE             (/ pdftmp /)
.*  NAME              PDFTMP
.*  DOMAIN            CONV2PDF
.*  LANGUAGE          ASS
.*  COPYRIGHT         (C) Fujitsu Technology Solutions 2015
.*                        ALL RIGHTS RESERVED
.*  COMPILATION-SCOPE USER
.*  INTERFACE-TYPE    CALL
.*  RUN-CONTEXT       TU
.*
.*  PURPOSE           (/ Text to PDF converter /)
.*
.*  SYNTAX            (/ Syntax Variant 1:
.*                            PDFTMP  MF = C|D|E|L|M
.*                          , PREFIX     = [S] | <name>
.*                          , MACID      = [TMP] | <name>
.*                          , PARAM      = <name 1..27>
.*                          , VARIANT    = <c-string_without_quotes 3..3> |
.*                                         default 001
.*                          , CALLER     = *USER |
.*                                         *SYSTEM |
.*                                         default *USER
.*                          , EQUATES    = [YES] | NO
.*                          , HANDLE     = <var: pointer> |
.*                                         *NONE |
.*                                         default *NONE
.*                          , ACTION     = <var: enum-of:2 _ACTION_SET> |
.*                                         *OPEN |
.*                                         *ADDOVERLAY |
.*                                         *ADDSECTION |
.*                                         *ATTOVERLAY |
.*                                         default *OPEN
.*                          , BOOKMRK    = <var: enum-of:1 _BOOKMARKUSE_SET>
|
.*                                         *YES |
.*                                         *NO |
.*                                         default *NO
.*                          , TMPNAME    = list(2):
.*                                         TMPNPTR: <var: pointer> |
.*                                                  *NONE |
.*                                                  default *NONE
.*                                         TMPNLEN: <var: int 0..65535> |
.*                                                  <integer 1..8> |
.*                                               *STD |
.*                                               default *STD
.*                          , OVERLAY    = list(4):
.*                                         OVLNPTR: <var: pointer> |
.*                                               *NONE |
.*                                               default *NONE
.*                                         OVLNLEN: <var: int 0..65535> |
.*                                               <integer 1..8> |
.*                                               *STD |
.*                                               default *STD
.*                                         OVLFPTR: <var: pointer> |
.*                                               *NONE |
.*                                               default *NONE
.*                                         OVLFLEN: <var: int 0..65535> |
.*                                               <integer 1..54> |
.*                                               *STD |
.*                                               default *STD
.*                          , OVLLOC     = list(9):
.*                                         OVLNPTR: <var: pointer> |
.*                                               *NONE |
.*                                               default *NONE
.*                                         OVLNLEN: <var: int 0..65535> |
.*                                               <integer 1..8> |
.*                                               *STD |
.*                                               default *STD
.*                                         MEDIABX: <var: enum-of:2
_MEDIABOX_SET> |
.*                                               *PAGE |
.*                                               *TEXT |
.*                                               *CUSTOM |
.*                                               default *PAGE
.*                                         LEFT: <var: int 0..65535> |
.*                                               <integer 0..2040> |
.*                                               default: 0
.*                                         RIGHT: <var: int 0..65535> |
.*                                                <integer 0..2040> |
.*                                                default: 0
.*                                         TOP: <var: int 0..65535> |
.*                                              <integer 0..2040> |
.*                                              default: 0
.*                                         BOTTOM: <var: int 0..65535> |
.*                                                 <integer 0..2040> |
.*                                                 default: 0
.*                                         HALIGN: <var: enum-of:1
_OVERLAY_ALIGN_SET> |
.*                                                 *LEFT |
.*                                                 *RIGHT |
.*                                                 *CENTER |
.*                                                   *FIT |
.*                                                   default *LEFT
.*                                           VALIGN: <var: enum-of:1
_OVERLAY_ALIGN_SET> |
.*                                                   *TOP |
.*                                                   *BOTTOM |
.*                                                   *CENTER |
.*                                                   *FIT |
.*                                                   default *TOP
.*                            , TOITEM     = list(3):
.*                                           ITEMTYP: <var: enum-of:2
_TMPITEM_SET> |
.*                                                    *SECTION |
.*                                                    default *SECTION
.*                                           ITMNPTR: <var: pointer> |
.*                                                    *NONE |
.*                                                    default *NONE
.*                                           ITMNLEN: <var: int 0..65535> |
.*                                                    <integer 1..8> |
.*                                                    *STD |
.*                                                    default *STD
.*                            , SECTION    = list(2):
.*                                           SECNPTR: <var: pointer> |
.*                                                    *NONE |
.*                                                    default *NONE
.*                                           SECNLEN: <var: int 0..65535> |
.*                                                    <integer 1..8> |
.*                                                    *STD |
.*                                                    default *STD /)
.*
.*  REMARKS            (/ LID definition /)
.*
.**********************************************************************
```

2 Examples

The sections below show the schematic structure of a program and examples for Assembler, C and Cobol. The examples must be filled out for a productive application.

2.1 Creating a program

The program interface enables the developer of P1 programs to generate PDF files directly from the developer’s own application - e.g. from logging or trace files.

The list below shows the steps which are relevant for creating a program with the associated parameters, Assembler notation being used here.

1. Optional: Define a PDF template

   When you want to design PDF pages with background pictures, you can use the PDFTMP macro to define the parameters for designing PDF template pages in a PDF template. Here a template section defines the design of a PDF template page which is used for all pages of a section of the PDF file which is to be created. Multiple template sections enable you to specify differently designed PDF pages and to assign these to individual page sections of the PDF file which is to be created.

   Proceed as follows to create a template:

   1. Create a template

      - ► Specify the *OPEN option for ACTION.
      - ► Define the template name in TMPNAME.
      - ► Optionally you can specify in BOOKMRK whether the PDF file which is created using the template will contain bookmarks.
      - ► Call the PDFTMP macro.

   2. Add a background picture to the template

      - ► Specify the ADDOVERLAY option for ACTION.
      - ► In the OVERLAY operand define an overlay name for the background picture and enter the picture file’s path name.

- ► In HANDLE enter the address of the identifier which is supplied when the template is created (OPEN call).
- ► Call the PDFTMP macro.

Repeat this step for each subsequent background picture which you wish to use in the template.

3. Define template section

   - ► Specify the *ADDSECTION option for ACTION.
   - ► Define the section name in the SECTION operand.
   - ► Optionally you use the BOOKMRK operand to specify whether you permit bookmarks in this template section when output takes place.
   - ► In HANDLE enter the address of the identifier which is supplied when the template is created (OPEN call).
   - ► Call the PDFTMP macro.

   Repeat this step for each subsequent template section.

4. Position background picture on a PDF page and assign it to a template section

   - ► Specify the *ATTOVERLAY option for ACTION.
   - ► Enter the parameters for positioning the background picture on the PDF page in the OVLLOC operand. The necessary parameters are:
     - – Overlay name with which the picture was added to the template
     - – Frame in which the picture is positioned
     - – Distances to the picture frame
     - – Vertical and horizontal alignment of the picture
   - ► In the TOITEM operand enter the item type *SECTION and the address of the template section to which the PDF page is to be assigned.
   - ► In HANDLE enter the address of the identifier which is supplied when the template is created (OPEN call).
   - ► Call the PDFTMP macro.

   Repeat this step for each subsequent PDF page. You can also position further background pictures on the PDF page of a template section.

2. Define layout parameters of the PDF file:

   You can define the layout parameters using the PDFDIR or PDFSPO macro:

   - ► To specify parameters directly, use the PDFDIR macro with the following options:
     - – Page size, margins, character density, line feed
     - – Specify whether text lines are to be output in full or in part
     - – Font (type, style, size)
   - ► To specify Spool parameters, use the PDFSPO macro with the following options:
     - – FORM, LOOP, printer type, ROT(ATION)
     - – Indent, lines per page, line feed
     - – Specify whether text lines are to be output in full or in part
     - – CHARSET (font)

3. Define properties of the PDF file:

   You define the properties of the PDF file the first time you call the PDFCVT macro:

   - ► Specify the *OPEN option for ACTION.
   - ► Via DIRPAR or SPOPAR, define which layout parameters of the PDF file are used, see Step 2. DIRPAR references the settings in the PDFDIR macro, SPOPAR the settings in PDFSPO. You may only ever specify one of the two parameters.
   - ► Use OUTNAME to define the name of the PDF file and the name length.
   - ► In HANDLE enter the address of a 4 byte long area which is aligned on a word boundary and in which the identifier of the PDF converter is stored.
   - ► Optionally you can set a write mode and font which differ from the default in WRMODE and CCS.
   - ► In TEMPLAT you can optionally use the PDF template pages of a previously defined PDF template for output, see Step 1.
   - ► Call the PDFCVT macro.

4. Specify bookmarks (optional):

   If you want to insert bookmarks in the PDF file, you must insert the first bookmark immediately after calling OPEN, i.e. before the first text line, otherwise an error will result.

   You insert a bookmark as follows using the PDFCVT macro:

   ► Specify the *BOOKMARK option for ACTION.

   ► Use BOOKMARK to define the name of the bookmark and the name length.

   ► In HANDLE enter the address of the identifier which is supplied with the OPEN call, see Step 3.

   ► Call the PDFCVT macro.

5. Insert a text line in the PDF file:

   You insert a text line as follows using the PDFCVT macro:

   ► Specify the *ADDTEXT option for ACTION.

   ► In TEXT enter the address and the length of the text line.

   ► In HANDLE enter the address of the identifier which is supplied with the OPEN call, see Step 3.

   ► Call the PDFCVT macro.

6. Repeat Step 5 for all text lines which are to be inserted in the PDF file.

   Repeat Step 4 if you want to insert further bookmarks.

7. Terminate current page section in the PDF file (optional):

   If the PDF file was defined with a PDF template, the PDF template page of the first template section is used to create the current PDF page. If the template page of a different template section is to be used, you must terminate the current page section and assign the new template section.

   You terminate the section of a PDF file as follows using the PDFCVT macro:

   ► Specify the *ENDSECTION option for ACTION.

   ► In NXTSECT enter a template section which is to be used for outputting the next PDF pages (of the next section).
   If this specification is missing, the template page of the next template section will be used.

   ► In HANDLE enter the address of the identifier which is supplied with the OPEN call, see Step 3.

   ► Call the PDFCVT macro.

- ► Repeat Steps 4 through 5 for the new page section of the PDF file.

8. Generate and close the PDF file:

   The PDF file is generated only when it is closed, i.e. the text lines and bookmarks are only then written to the PDF file.

   You close the PDF file as follows using the PDFCVT macro:

   - ► Specify the *CLOSE option for ACTION.
   - ► In HANDLE enter the address of the identifier which is supplied with the OPEN call, see Step 3.
   - ► Call the PDFCVT macro.

2.2 Example for Assembler

The layout parameters are specified directly in this example (PDFDIR macro).

```
##BAL    OPSYN ##BAS
##BALR   OPSYN ##BASR
TESTASC  AMODE ANY
TESTASC  RMODE ANY
TESTASC  @ENTR TYP=M,LOCAL=DMGWA,ENV=SPLSPEC,TITLE=NO
*
         USING STAT,11
         L     11,=A(STAT)
         MVC   0(4,4),=X'00000000'
*
* DIRPL
*
         LA    1,DIRPL
         USING DIRPL,1
         LR    3,1
         MVC   DIRPL(DIRPLL),DIRPLC
         PDFDIR MF=M,                                                  C
               RECPART=(1,50),                                         C
               LINESP=(*SPACE_3,3),                                    C
               PAGESZ=(*A4_LANDSCAPE,*NONE,*NONE),                     C
               FONT=(*HELVETICA,*BOLD,8),                              C
               MARGIN=(20,40,10,30),                                   C
               LPI=8



*
         LA    2,PDFPL
         ST    2,PDFPLA
         MVC   PDFPL(PDFPLL),PDFPLC
         PDFCVT MF=M,                                                  C
               ACTION=*OPEN,                                           C
               OUTNAME=(A(FN),10),                                     C
               WRMODE=*CREATE,                                         C
               HANDLE=A(HDL),                                          C
               SPOPAR=(4),                                             C
               DIRPAR=(1)
*
         PDFCVT MF=E,PARAM=PDFPLA
*
         PDFCVT MF=M,                                                  C
               ACTION=*ADDTEXT,                                        C
               TEXT=(A(LINE1),9),                                      C
               DIRPAR=(3)
```

```
*
         PDFCVT MF=E,PARAM=PDFPLA
*
         PDFCVT MF=M,                                                  C
               ACTION=*ADDTEXT,                                        C
               TEXT=(A(LINE2),9),                                      C
               DIRPAR=(3)
*
         PDFCVT MF=E,PARAM=PDFPLA
*
         PDFCVT MF=M,                                                  C
               ACTION=*ADDBOOKMARK,                                    C
               BOOKMRK=(A(BKMRK1),13),                                 C
               DIRPAR=(3)
*
         PDFCVT MF=E,PARAM=PDFPLA
*
*
         PDFCVT MF=M,                                                  C
               ACTION=*CLOSE,                                          C
               TEXT=(*NONE,*STD),                                      C
               OUTNAME=(*NONE,*STD),                                   C
               SPOPAR=(4),                                             C
               DIRPAR=(3)
*
         PDFCVT MF=E,PARAM=PDFPLA
         @EXIT
         @END
*

DMGWA    @PAR  D=YES
         PRINT GEN
WA       DS    0F
*
PDFPLA   DS    A
PDFPL    PDFCVT MF=C
DIRPL    PDFDIR MF=C
*
DMGWA    @PAR  LEND=YES
*
STAT     DS    0F
PDFPLC   PDFCVT MF=L
PDFPLL   EQU   *-PDFPLC
*
DIRPLC   PDFDIR MF=L
DIRPLL   EQU   *-DIRPLC
*
LINE1    DC    C'LINE 0001'
```

```
LINE1L   EQU   *-LINE1
LINE2    DC    C'LINE 0002'
LINE2L   EQU   *-LINE2
BKMRK1   DC    C'BOOKMARK 0001'
BKMRK1L  EQU   *-BKMRK1
FN       DC    C'MYFILE.PDF'
FNL      EQU   *-FN
HDL      DS    A
*
         PDFCVT MF=D,PREFIX=X
*
         PDFSPO MF=D,PREFIX=X
*
         PDFDIR MF=D,PREFIX=X
*
         END
```

2.3 Example for C

The layout parameters are specified directly in this example.

```
// TESTAPI.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include "FHDR.H"
#include "PDFCVT.H"
#include "PDFDIR.H"
#include "PDFSPO.H"
#include <stdlib.h>
#include <string.h>

int main(int argc, char* argv[])
{
struct PDFCVT_pl_mdl pl;
struct PDFDIR_pl_mdl dirpl;
struct PDFSPO_pl_mdl spopl;
char fn[54];
char text1[11];
char bkmk1[11];
char line1[4096];
int j = 1;
int rc = 0;
char *pspopl = new char[1000];
memcpy(fn,"T-6000.PDF",10);

rc = 0;
memset((char*)&pl,'\0',sizeof(pl));
pl.action = PDFCVTaction_open;
pl.handle_ptr = 0;
pl.outname_ptr = fn;
pl.outname_len = 10;
pl.wrmode = PDFCVTwrmode_any;
pl.specified1.spec1_action = 1;
pl.specified1.spec1_outname = 1;
pl.specified1.spec1_wrmode = 1;
pl.specified1.spec1_spopar = 0;
pl.specified1.spec1_dirpar = 1;
pl.dirpar = &dirpl;
pl.spopar = 0;
memset ((char*)&dirpl,'\0',sizeof(dirpl));
dirpl.pagesz.media = PDFDIRpagesz_a4;
dirpl.specified1.spec1_custom = 0;
dirpl.specified1.spec1_media = 1;
dirpl.font.name = PDFDIRfn_courier;
dirpl.font.style = PDFDIRfn_normal;
dirpl.font.size = 8;
dirpl.specified1.spec1_font = 1;
dirpl.linesp.spacing = PDFDIRspace_1;
dirpl.linesp.cc_pos = 1;
dirpl.specified1.spec1_linesp = 1;
SPDFMOD(pl);
printf("open return code = %x-%x-%x\n",
pl.hdr.returncode.rc.structured_rc.subcode.subcode2,
pl.hdr.returncode.rc.structured_rc.subcode.subcode1,
pl.hdr.returncode.rc.structured_rc.mc.maincode);

if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
     printf("TEST 6000: open NOK\n");
else
{
       /* Write 1 bookmark and 100 lines */

     memcpy(bkmk1,"BOOKMARK 1",10);
     pl.action = PDFCVTaction_addb;
     pl.specified1.spec1_bkmrk = 1;
     pl.bookmark_ptr = bkmk1;
     pl.bookmark_len = 10;
     SPDFMOD(pl);
        if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
        {
          printf("TEST 6000: addbookmark NOK\n");
          rc = 1;
     }
     j = 1;
     while ((j <= 100) && (rc == 0))
     {
          pl.action = PDFCVTaction_addt;
          sprintf(text1,"LINE   %03d", j);
          pl.text_ptr = text1;
          pl.text_len = 10;
          pl.specified1.spec1_text = 1;
          SPDFMOD(pl);
               if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
               {
printf("return code = %x-%x-%x\n",
pl.hdr.returncode.rc.structured_rc.subcode.subcode2,
pl.hdr.returncode.rc.structured_rc.subcode.subcode1,
pl.hdr.returncode.rc.structured_rc.mc.maincode);
               printf("TEST 6000: addtext NOK\n");
               rc = 1;
          }
               j++;
     }
/* Write 2nd bookmark and 100 lines */

memcpy(bkmk1,"BOOKMARK 2",10);
     pl.action = PDFCVTaction_addb;
     pl.specified1.spec1_bkmrk = 1;
     pl.bookmark_ptr = bkmk1;
     pl.bookmark_len = 10;
     SPDFMOD(pl);
       if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
       {
          printf("TEST 6000: addbookmark NOK\n");
          rc = 1;
     }
     j = 1;
     while ((j <= 100) && (rc == 0))
     {
          pl.action = PDFCVTaction_addt;
          sprintf(text1,"LINE   %03d", j);
          pl.text_ptr = text1;
          pl.text_len = 10;
          pl.specified1.spec1_text = 1;
          SPDFMOD(pl);
               if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
               {
printf("return code = %x-%x-%x\n",
pl.hdr.returncode.rc.structured_rc.subcode.subcode2,
pl.hdr.returncode.rc.structured_rc.subcode.subcode1,
pl.hdr.returncode.rc.structured_rc.mc.maincode);
               printf("TEST 6000: addtext NOK\n");
               rc = 1;
          }
          j++;
     }
       /* Generate the PDF */

     pl.action = PDFCVTaction_clos;
     SPDFMOD(pl);
printf("return code = %x-%x-%x\n",
pl.hdr.returncode.rc.structured_rc.subcode.subcode2,
pl.hdr.returncode.rc.structured_rc.subcode.subcode1,
pl.hdr.returncode.rc.structured_rc.mc.maincode);
     if (pl.hdr.FHDR_RC_MAINCODE != PDFCVTok)
          printf("TEST 6000: NOK\n");
     else
          printf("TEST 6000: OK\n");
}
}
```

2.4 Example for COBOL

The layout parameters are specified using SPOOL parameters in this example.

```
000100 IDENTIFICATION DIVISION.
000200*-----------------------*
000300*
000400 PROGRAM-ID.
000500*----------*
000600     TESTCOB.
000700/
000800 ENVIRONMENT DIVISION.
000900*--------------------*
001000*
001100 CONFIGURATION SECTION.
001200*---------------------*
001300*
001400 SPECIAL-NAMES.
001500*-------------*
001600*
001700     TERMINAL IS v-terminal,
001800     SYMBOLIC CHARACTERS
001900     COPY esmhexay. .
002000/
002100 DATA DIVISION.
002200*-------------*
002300*
002400 WORKING-STORAGE SECTION.
002500*-----------------------*
002600*
002700 01  hexa-chars             PIC X(16) VALUE "0123456789ABCDEF".
002800 01  maincode-edit.
002900     02  maincode-dec           PIC S9(9) COMP.
003000     02  FILLER                 REDEFINES maincode-dec.
003100       03  maincode-byte        PIC X(01) OCCURS 4 TIMES.
003200     02  maincode-hex           PIC X(08).
003300     02  FILLER                 REDEFINES maincode-hex.
003400       03  maincode-char2                 OCCURS 4 TIMES.
003500         04  maincode-char      PIC X(01) OCCURS 2 TIMES.
003600 01  work-fields.
003700     02  work-counters.
003800       03  i                    PIC S9(04) COMP.
003900       03  work-hw              PIC S9(04) COMP.
004000       03  FILLER               REDEFINES work-hw.
004100         04  work-hw-1          PIC X(01).
004200         04  work-hw-2          PIC X(01).
004300     02  fn                     PIC  X(54).
```

```
     02  fnl                     PIC  9(04) COMP.
     02  hdl                     PIC  9(09) COMP.
     02  line1                   PIC  X(80).
     02  lnl                     PIC  9(04) COMP.
     02  book1                   PIC  X(80).
     02  bkl                     PIC  9(04) COMP.
/
     COPY pdfcvty .
/
     COPY pdfspoy .
/
     COPY pdfdiry .
/
 PROCEDURE DIVISION.
*------------------*
*
 s-main SECTION.
*--------------*
*
 p-main.
*------*
*
     PERFORM s-test000.
     PERFORM s-test001.
     PERFORM s-test002.
     PERFORM s-test003.
     PERFORM s-test004.
*
 p-exit.
*------*
*
     STOP RUN.
/----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                *
*    Initialize PLs, and call entry                              *
*                                                                *
*----+----+----+----+----+----+----+----+----+----+----+----+----*
*
 s-test000 SECTION.
*-----------------*
*
 p-test000-strt.
*--------------*
*
     DISPLAY "TESTO OF TESTCOB"     UPON v-terminal.
     MOVE PDFCVT-I-pl               TO PDFCVT-pl.
     MOVE PDFSPO-I-pl               TO PDFSPO-pl.
     MOVE PDFDIR-I-pl               TO PDFDIR-pl.
     CALL "SCPADDR"                 USING PDFCVT-spo-par,
                                          PDFSPO-pl.
     CALL "SCPADDR"                 USING PDFCVT-dir-par,
                                          PDFDIR-pl.
*
 p-test000-call.
*--------------*
*
     CALL "SPDFMOD"                 USING PDFCVT-pl.
*
 p-test000-retc.
*--------------*
*
     IF esmfhdr-rc-nbr IN PDFCVT-pl = ZERO
     THEN
       DISPLAY "MAINCODE = X'00000000'" UPON v-terminal,
     ELSE
       MOVE esmfhdr-rc-nbr IN PDFCVT-pl TO maincode-dec,
       PERFORM s-edit-maincode,
       DISPLAY "MAINCODE = X'", maincode-hex, "'" UPON v-terminal,
     END-IF.
*
 p-test000-exit.
*--------------*
     EXIT.
/----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                *
*    Test operand SPOPAR                                         *
*                                                                *
*----+----+----+----+----+----+----+----+----+----+----+----+----*
*
 s-test001 SECTION.
*-----------------*
*
 p-test001-strt.
*--------------*
*
     DISPLAY "TEST1 OF TESTCOB"     UPON v-terminal.
     MOVE PDFCVT-I-pl               TO PDFCVT-pl.
     MOVE PDFSPO-I-pl               TO PDFSPO-pl.
     MOVE PDFDIR-I-pl               TO PDFDIR-pl.
     CALL "SCPADDR"                 USING PDFCVT-spo-par,
                                          PDFSPO-pl.
     CALL "SCPADDR"                 USING PDFCVT-dir-par,
                                          PDFDIR-pl.
*
     MOVE 2                         TO PDFSPO-first-ch.
     MOVE 85                        TO PDFSPO-last-ch.
```

```
014000     MOVE 80                        TO PDFSPO-linepp.
014100     MOVE "STD001"                  TO PDFSPO-form.
014200     MOVE "ABC"                     TO PDFSPO-loop.
014300     MOVE "101"                     TO PDFSPO-charset.
014400     SET PDFSPO-hp                  TO TRUE.
014500     MOVE 25                        TO PDFSPO-leftmarg.
014600     SET PDFSPO-rotation-yes        TO TRUE.
014700*
014800     SET PDFSPO-specified1-recpart  TO TRUE.
014900     CALL "SCPSETB1"                USING PDFSPO-specified1,
015000                                          PDFSPO-specified1-set.
015100     SET PDFSPO-specified1-linepp   TO TRUE.
015200     CALL "SCPSETB1"                USING PDFSPO-specified1,
015300                                          PDFSPO-specified1-set.
015400     SET PDFSPO-specified1-form     TO TRUE.
015500     CALL "SCPSETB1"                USING PDFSPO-specified1,
015600                                          PDFSPO-specified1-set.
015700     SET PDFSPO-specified1-loop     TO TRUE.
015800     CALL "SCPSETB1"                USING PDFSPO-specified1,
015900                                          PDFSPO-specified1-set.
016000     SET PDFSPO-specified2-charset  TO TRUE.
016100     CALL "SCPSETB1"                USING PDFSPO-specified2,
016200                                          PDFSPO-specified2-set.
016300     SET PDFSPO-specified1-prtype   TO TRUE.
016400     CALL "SCPSETB1"                USING PDFSPO-specified1,
016500                                          PDFSPO-specified1-set.
016600     SET PDFSPO-specified1-leftmar  TO TRUE.
016700     CALL "SCPSETB1"                USING PDFSPO-specified1,
016800                                          PDFSPO-specified1-set.
016900     SET PDFSPO-specified1-rot      TO TRUE.
017000     CALL "SCPSETB1"                USING PDFSPO-specified1,
017100                                          PDFSPO-specified1-set.
017200*
017300     SET PDFCVT-action-open         TO TRUE.
017400     MOVE "MYFILE.PDF"              TO fn.
017500     MOVE 10                        TO fnl.
017600     CALL "SCPADDR"                 USING PDFCVT-outname-ptr,
017700                                          fn.
017800     MOVE fnl                       TO PDFCVT-outname-len.
017900     SET PDFCVT-wrmode-create       TO TRUE.
018000     CALL "SCPADDR"                 USING PDFCVT-handle-ptr,
018100                                          hdl.
018200*
018300     SET PDFCVT-specified1-action   TO TRUE.
018400     CALL "SCPSETB1"                USING PDFCVT-specified1,
018500                                          PDFCVT-specified1-set.
018600     SET PDFCVT-specified1-outname  TO TRUE.
018700     CALL "SCPSETB1"                USING PDFCVT-specified1,
                                          PDFCVT-specified1-set.
     SET PDFCVT-specified1-wrmode   TO TRUE.
     CALL "SCPSETB1"                USING PDFCVT-specified1,
                                          PDFCVT-specified1-set.
     SET PDFCVT-specified1-handle   TO TRUE.
     CALL "SCPSETB1"                USING PDFCVT-specified1,
                                          PDFCVT-specified1-set.
     SET PDFCVT-specified1-spopar   TO TRUE.
     CALL "SCPSETB1"                USING PDFCVT-specified1,
                                          PDFCVT-specified1-set.
*
 p-test001-call.
*--------------*
*
     CALL "SPDFMOD"                 USING PDFCVT-pl.
*
 p-test001-retc.
*--------------*
*
     IF esmfhdr-rc-nbr IN PDFCVT-pl = ZERO
     THEN
       DISPLAY "MAINCODE = X'00000000'" UPON v-terminal,
     ELSE
       MOVE esmfhdr-rc-nbr IN PDFCVT-pl TO maincode-dec,
       PERFORM s-edit-maincode,
       DISPLAY "MAINCODE = X'", maincode-hex, "'" UPON v-terminal,
     END-IF.
*
 p-test001-exit.
*--------------*
     EXIT.
/----+----+----+----+----+----+----+----+----+----+----+----+----*
*                                                                *
*    Test operand BOOKMRK=(ADDR(BOOK1),SPACE(BOOK1))             *
*                                                                *
*----+----+----+----+----+----+----+----+----+----+----+----+----*
*
 s-test002 SECTION.
*-----------------*
*
 p-test002-strt.
*--------------*
*
     DISPLAY "TEST2 OF TESTCOB"     UPON v-terminal.
     SET PDFCVT-action-addbookmark  TO TRUE.
     MOVE "BOOKMARK 0001"           TO book1.
     MOVE 13                        TO bkl.
     CALL "SCPADDR"                 USING PDFCVT-bookmark-ptr,
023600                                          book1.
023700     MOVE bkl                       TO PDFCVT-bookmark-len.
023800*
023900     SET PDFCVT-specified1-action   TO TRUE.
024000     CALL "SCPSETB1"                USING PDFCVT-specified1,
024100                                          PDFCVT-specified1-set.
024200     SET PDFCVT-specified1-bkmrk    TO TRUE.
024300     CALL "SCPSETB1"                USING PDFCVT-specified1,
024400                                          PDFCVT-specified1-set.
024500*
024600 p-test002-call.
024700*--------------*
024800*
024900     CALL "SPDFMOD"                 USING PDFCVT-pl.
025000*
025100 p-test002-retc.
025200*--------------*
025300*
025400     IF esmfhdr-rc-nbr IN PDFCVT-pl = ZERO
025500     THEN
025600       DISPLAY "MAINCODE = X'00000000'" UPON v-terminal,
025700     ELSE
025800       MOVE esmfhdr-rc-nbr IN PDFCVT-pl TO maincode-dec,
025900       PERFORM s-edit-maincode,
026000       DISPLAY "MAINCODE = X'", maincode-hex, "'" UPON v-terminal,
026100     END-IF.
026200*
026300 p-test002-exit.
026400*--------------*
026500     EXIT.
026600/----+----+----+----+----+----+----+----+----+----+----+----+----*
026700*                                                                *
026800*    Test operand TEXT=(ADDR(LINE1),SPACE(LINE1))                *
026900*                                                                *
027000*----+----+----+----+----+----+----+----+----+----+----+----+----*
027100*
027200 s-test003 SECTION.
027300*-----------------*
027400*
027500 p-test003-strt.
027600*--------------*
027700*
027800     DISPLAY "TEST3 OF TESTCOB"     UPON v-terminal.
027900     SET PDFCVT-action-addtext      TO TRUE.
028000     MOVE "LINE 0001"               TO line1.
028100     MOVE 9                         TO lnl.
028200     CALL "SCPADDR"                 USING PDFCVT-text-ptr,
028300                                          line1.
```

```
028400     MOVE lnl                       TO PDFCVT-text-len.
028500*
028600     SET PDFCVT-specified1-action   TO TRUE.
028700     CALL "SCPSETB1"                USING PDFCVT-specified1,
028800                                          PDFCVT-specified1-set.
028900     SET PDFCVT-specified1-text     TO TRUE.
029000     CALL "SCPSETB1"                USING PDFCVT-specified1,
029100                                          PDFCVT-specified1-set.
029200*
029300 p-test003-call.
029400*--------------*
029500*
029600     CALL "SPDFMOD"                 USING PDFCVT-pl.
029700*
029800 p-test003-retc.
029900*--------------*
030000*
030100     IF esmfhdr-rc-nbr IN PDFCVT-pl = ZERO
030200     THEN
030300       DISPLAY "MAINCODE = X'00000000'" UPON v-terminal,
030400     ELSE
030500       MOVE esmfhdr-rc-nbr IN PDFCVT-pl TO maincode-dec,
030600       PERFORM s-edit-maincode,
030700       DISPLAY "MAINCODE = X'", maincode-hex, "'" UPON v-terminal,
030800     END-IF.
030900*
031000 p-test003-exit.
031100*--------------*
031200     EXIT.
031300/----+----+----+----+----+----+----+----+----+----+----+----+----*
031400*                                                                *
031500*    Close                                                       *
031600*                                                                *
031700*----+----+----+----+----+----+----+----+----+----+----+----+----*
031800*
031900 s-test004 SECTION.
032000*-----------------*
032100*
032200 p-test004-strt.
032300*--------------*
032400*
032500     DISPLAY "TEST4 OF TESTCOB"     UPON v-terminal.
032600     SET PDFCVT-action-close        TO TRUE.
032700*
032800     SET PDFCVT-specified1-action   TO TRUE.
032900     CALL "SCPSETB1"                USING PDFCVT-specified1,
033000                                          PDFCVT-specified1-set.
033100*
```

```
033200 p-test004-call.
033300*--------------*
033400*
033500     CALL "SPDFMOD"                  USING PDFCVT-pl.
033600*
033700 p-test004-retc.
033800*--------------*
033900*
034000     IF esmfhdr-rc-nbr IN PDFCVT-pl = ZERO
034100     THEN
034200       DISPLAY "MAINCODE = X'00000000'" UPON v-terminal,
034300     ELSE
034400       MOVE esmfhdr-rc-nbr IN PDFCVT-pl TO maincode-dec,
034500       PERFORM s-edit-maincode,
034600       DISPLAY "MAINCODE = X'", maincode-hex, "'" UPON v-terminal,
034700     END-IF.
034800*
034900 p-test004-exit.
035000*--------------*
035100     EXIT.
035200/
035300 s-edit-maincode SECTION.
035400*----------------------*
035500*
035600 p-edit-maincode-strt.
035700*--------------------*
035800*
035900     PERFORM WITH TEST AFTER          VARYING i FROM 1 BY 1
036000       UNTIL i > FUNCTION LENGTH(maincode-dec)
036100       MOVE ZERO                      TO work-hw,
036200       MOVE maincode-byte(i)          TO work-hw-2,
036300       MOVE hexa-chars(work-hw / 16 + 1: 1)
036400                                      TO maincode-char(i, 1),
036500       MOVE hexa-chars(FUNCTION MOD(work-hw, 16) + 1: 1)
036600                                      TO maincode-char(i, 2),
036700     END-PERFORM.
036800*
036900 p-edit-maincode-exit.
037000*--------------------*
037100*
037200     EXIT.
```

Related publications

You will find the manuals on the internet at *http://manuals.ts.fujitsu.com*. You can order printed copies of those manuals which are displayed with an order number.

AID (BS2000)
Advanced Interactive Debugger
Debugging of ASSEMBH Programs
User Guide

Assembler Instructions(BS2000)
Reference Manual

ASSEMBH (BS2000)
Description

BINDER (BS2000)
User Guide

BS2ZIP
Zip Archiving in BS2000
User Guide

BS2000 OSD/BC
Executive Macros
User Guide

Spool & Print - Commands (BS2000)
User Guide

Spool & Print - Macros and Exits (BS2000)
User Guide

SPOOL (BS2000)
User Guide

XHCS (BS2000)
8-Bit Code Processing in BS2000
User Guide