SIEMENS

EasyCODE

Customization

EasyCODE Version 6.xE, 06-20-1996

Table of Contents

2. COMMAND LINE OPTIONS AND PARAMETERS

Customization

This paper describes some ways of customizing EasyCODE to user requirements, which are not contained in the standard user documentation (Help, user manual).

1 Ini/Cfg Entries

The following Ini/Cfg entries cannot be modified using the EasyCODE user interface, but they, in turn, may affect application behavior considerably.

2 LockDrives

Components: All structure diagram editors
Version: V3.5 and higher versions
File: EASY-<COMP>.INI
Section: [Network]
Value: Sequence of letters indicating drives
Default: None

Example: LockDrives=CDEF

Consequences: By default, the EasyCODE network version checks network drives, but not local drives, for multiple access. This entry modifies this predefined setting in that the specified drives will also be checked for multiple access, no matter whether they are local or network drives. All available drives may be specified as well as the character '\' which controls access authorization for files addressed by \\<server>\<share\<path>\<filename> instead of logical drives.

3 UnLockDrives

Components: All structure diagram editors
Version: V3.5 and higher versions
File: EASY-<COMP>.INI
Section: [Network]
Value: Sequence of letters indicating drives
Default: None

Example: UnLockDrives=MNO

Consequences: By default, the EasyCODE network version checks network drives for multiple access. If e.g. in program linking share conflicts occur because programs not designed for network use are working on the file currently opened by EasyCODE, this situation may be modified by the UnlockDrives entry in that EasyCODE will not check the specified drives for multiple access. All available drives may be specified as well as the character '\' which controls access authorization for files addressed by \\<server>\<share\<path>\<filename> instead of logical drives.

4 SuppressSourceConvMsg

Components: DS, COB, SPX
Version: V3.5 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: SuppressSourceConvMsg=yes

Consequences: The message displayed when a source file is opened or inserted will be suppressed when this entry is set to yes, true or 1.

5 SuppressSPXConvMsg (SuppressSPConvMsg)

Components: All SE except SP, SPX, DS
Version: V5.0 and higher versions (V3.5 up to V4.0x)
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: SuppressSPXConvMsg=yes (SuppressSPConvMsg=yes)

Consequences: The message displayed when an SPX(SP) file is opened or inserted will be suppressed when this entry is set to yes, true or 1.

6 SuppressJETConvMsg

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: SuppressJETConvMsg=yes

Consequences: The message displayed when a JET file is opened or inserted will be suppressed when this entry is set to yes, true or 1.

7 SuppressETFConvMsg

Components: All structure diagram editors
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: SuppressETFConvMsg=yes

Consequences: The message displayed when an ETF file is opened or inserted will be suppressed when this entry is set to yes, true or 1.

8 FtAvailable

Components: All structure diagram editors
Version: V3.5 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: FtAvailable=yes

Consequences: File-transfer support in the structure diagram editors will be activated or deactivated. Usually, this entry is made by the SETUP program according to the option specified there. It is, however, possible to modify it later, so that file-transfer support may be activated or deactivated without installing EasyCODE again. The default entries made by the SETUP program are no for all components.

9 SourceFileFormat

Components: JET, PET
Version: ab V5.01
File: EASY-<COMP>.INI
Section: [Settings]
Value: OEM|ANSI
Default: OEM

Example: SourceFileFormat=ANSI

Consequences: Specifies whether the JOB files are to be OEM or ANSI coded. Will be evaluated when JOB files are generated, exported or imported (Generate, Open, Insert File).

10 EtfFileFormat

Components: All structure diagram editors
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: OEM|ANSI
Default: OEM

Example: EtfFileFormat=ANSI

Consequences: Specifies whether the ETF files are to be OEM or ANSI coded. Will be evaluated when ETF files are exported or imported (Open, Insert File).

11 EtfWrapSDF

Components: JET, PET
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: EtfWrapSDF=no

Consequences: Specifies whether SDF commands and statements are word wrapped according to the representation in the structure diagram and the generated job when exported. Otherwise one single line will be exported.

12 JobLog

Components: JET, PET
Version: V3.5x, V4.0x
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: JobLog=yes

Consequences: Specifies whether additional commands for calling the JOBLOG utilities are generated. Only appropriate for computing centers using these utilities. (Due to the implementation of the LOGON/LOGOFF exits in the operating system this option is dropped in EasyCODE Version 5.0 and higher).

13 DeleteWorkFiles

Components: JET, PET
Version: V3.5 and higher versions

File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: DeleteWorkFiles=yes

Consequences: Specifies whether additional command for deleting workfiles are generated if the procedure contains the SDF command ASSIGN-OUTPUT-FILE (assignment of workfiles). Only appropriate for computing centers where these commands are available. The commands are generated only in the normal end (not within the abnormal end) of the procedure.

14 Compiler

Components: COB
Version: V3.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: VISCOB|none
Default: none

Example: Compiler=VISCOB

Consequences: Specifies whether a syntax check may be carried out by the compiler during file editing or generation. Makes sense only if the Visual COBOL compiler is used.

15 Parser

Components: COB, DS, SPX
Version: V3.51 and higher versions
File: EASY-<COMP>.INI for COB and DS, configuration file (.CFG) for SPX
Section: [Settings] for COB and DS, [ParseOptions] for SPX
Value: Filename of parser DLL
Default: No such entry. In this case, EasyCODE(COB) will use the Visual COBOL parser (EASY-PAR.DLL) and EasyCODE(DS) will use the BNF parser (EASY-BNF.DLL), which are stored in the EasyCODE installation directory. There is no such default setting in EasyCODE(SPX); in this case, no parser will be called, and, if the file is neither an SPX nor an ETF file, a corresponding error message will appear ("Data in <file> do not have EasyCODE(<component>) format.").

Example: Parser=EASY-XBS.DLL

Consequences: Specifies which parser is to be used for analyzing source files.

Notes: In V5.0 and higher versions of EasyCODE(COB), a new EasyCODE-internal COBOL parser providing exended functions will be available by default. The INI entry required for this new parser (Parser=EASY-CBL.DLL) will be made automatically during the Setup procedure.
If the „Parser“ entry is not empty in EasyCODE(COB) V5.0 and higher versions, i.e. if the Visual COBOL parser is not used, the parser will be loaded when files are inserted, if the file in question is a source file. (The parsing of incomplete sources will not be supported by the Visual COBOL parser.)

16 ParserDebugFile

Components: COB, DS, SPX
Version: V3.51 and higher versions
File: EASY-<COMP>.INI for COB and DS, configuration file (.CFG) for SPX
Section: [Settings] for COB and DS, [ParseOptions] for SPX
Value: Filename of a protocol file
Default: No such entry.

Example: Parser=EASY-XBS.DLL

Consequences: This entry is used for debugging self-designed parsers. If this entry exists, EasyCODE will create a protocol file containing everything delivered by the parser to the parser interface.

17 FtShowIconic

Components: All structure diagram editors, EasyFT
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: FtShowIconic=no

Consequences: This entry is evaluated for the Intrasys file transfer system WFT only and specifies whether WFT is to display the file-transfer window as an icon.

18 RetainReplaceCheckboxState

Components: All structure diagram editors
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RetainReplaceCheckboxState=no

Consequences: Usually, when you press the Ok button in the Search/Replace dialog window, the states or contents of all boxes in this dialog window will be stored until the dialow window is opened again. This entry modifies the "☐ and replace with" check box behavior in that its state will not be stored, i.e. this box will never be checked when the dialog window is opened.

19 PushDownClasses

Components: C++
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PushDownClasses=no

Consequences: Specifies whether classes (class, struct, union) are to be pushed down to a lower level automatically when a source is analyzed for the first time.

20 PushDownFunctions

Components: C, C++
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PushDownFunctions=no

Consequences: Specifies whether functions are to be pushed down to a lower level automatically when a source is analyzed for the first time.

21 PrintFileStandard

Components: All structure diagram editors, SD
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PrintFileStandard=no

Consequences: Specifies whether the file is to be printed using the default options chosen in the Print dialog window (default) when the application is started with the /print option in the command line , or whether the Print dialog window is to be opened so that the printing options can be modified. This applies also to drag&drop printing starting from the File Manager or to the printing of EasyCODE supplementary documents in a communication plan.

22 PrintReportStandard

Components: SD
Version: ab V5.01
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PrintReportStandard=no

Consequences: Specifies whether the default report options (specified in the Ini-file with the "Rep…" entries) or interactive with the Report dialog window should be taken when you start the application with the /Report option in the command line.

23 InitialIndent

Components: DS (for SPX see description of the .CFG file)
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: positive, whole number or 0
Default: 0

Example: InitialIndent=3

Consequences: Specifies the initial indent for file generation.

24 BeginComment

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '*'

Example: BeginComment='$$$'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

25 EndComment

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '*'

Example: EndComment='$$$'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated and analyzed.

26 BeginObject

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '='

Example: BeginObject='::='

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

27 EndObject

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: ''

Example: EndObject=';'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed. The BNF does not provide end object symbols, and therefore they are not supported at present. The entry is reserved for future versions.

28 Sequence

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '+'

Example: Sequence='&&'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

29 BeginIteration

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '{'

Example: BeginIteration='<<'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

30 EndIteration

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '}'

Example: EndIteration='>>'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

31 BeginOption

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '('

Example: BeginOption='<'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

32 EndOption

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: ')'

Example: EndOption='>'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

33 BeginSelection

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '['

Example: BeginSelection='??'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

34 EndSelection

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: ']'

Example: EndSelection='??'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

35 SeparateCases

Components: DS
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [SyntaxStrings]
Value: character string
Default: '|'

Example: SeparateCases='/'

Consequences: For redefining BNF symbols of syntactical relevance, will be taken into account when BNF files are generated or analyzed.

36 SuppressDoInProcedureCall

Components: xBASE parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: SuppressDoInProcedureCall=yes

Consequences: Specifies whether the DO in the procedure call of a construct is to be suppressed or added. (Should be analogous to the generation of the procedure-call construct: yes if the DO is to be added automatically during the generation process, otherwise no.)

37 SuppressLastReturnInProcedure

Components: xBASE parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: SuppressLastReturnInProcedure=no

Consequences: Specifies whether the RETURN at the end of a procedure in a structure diagram is to be suppressed or added. (Should be analogous to the generation of the procedure construct: yes if the RETURN is to be added automatically during the generation process, otherwise no.)

38 Ctl3D

Components: All structure diagram editors, SD, EasyFT
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: Ctl3D=yes

Consequences: Specifies whether the 3D versions of the dialog and message windows are to be used. (During the EasyCODE installation, the SETUP program automatically selects the 3D look for all EasyCODE components to be installed.)

39 FtDirectory

Components: All structure diagram editors, EasyFT
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: directory name
Default: none

Example: FtDirectory='c:\usr\ft'

Consequences: Specifies the directory from which the file transfer systems WFT BAM, WFT TCP/IP or FTOS will be started. (Attention: the entry is valid for these file transfer products only!) This entry overwrites all existing settings concerning the environment variables WFTDIR or FTC97. Therefore, the following strategy is used: If the INI entry FtDirectory is not empty, its contents will be interpreted as the directory name and used for starting WFT.EXE or FTD.EXE. If the entry is empty, the contents of the respective environment variables will be used. If these are empty, too, file transfer will be started without a specified path name, the EXE file must then be included in the DOS path statement. The entry FtDirectory is therefore used for starting a file transfer regardless of the PATH variable or special environment variables.

40 LockSourceOnWarnings

Components: C, C++
Version: V4.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: ask|yes|no|true|false|1|0
Default: ask

Example: LockSourceOnWarnings=no

Consequences: Specifies how to proceed with the source to be opened in case of warnings (not errors) during file analysis in the network version of the components mentioned above. If the editor to be used for displaying the source file is configurated, the user must decide whether EasyCODE or the specified editor is to open the file in read-and-write mode. The other application may then display the file in read-only mode. If the default setting (LockSourceOnWarnings=ask) is chosen, the following message will appear: "File <sourcefile> must be opened with write-protection either in EasyCODE(<COMP>) or in the editor. Write-protection in EasyCODE(<COMP>)?". Each time, the user may decide how to proceed. The INI entry LockSourceOnWarnings=yes specifies that the file will always be opened by EasyCODE in the read-and-write mode and by the editor in the read-only mode, and the message will not be displayed. The INI entry LockSourceOnWarnings=no specifies that the file will always be opened by EasyCODE in the read-only mode and by the editor in the read-and-write mode, and the message will not appear.

41 CompressBlanks

Components: COB
Version: V4.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: CompressBlanks=no

Consequences: Specifies whether a sequence of blanks is to be replaced with tabs (one tab corresponding to eight blanks) when a COBOL source is generated. The purpose of this compressing procedure is a reduction of the file size. Usually, this procedure has no further consequences, since the tabs will again be replaced with the corresponding number of blanks by the COBOL compiler or the file transfer system. The default setting is therefore 'yes'. If, however, problems should occur because of the file transfer system and/or the COBOL compiler in use, the compressing procedure may be suppressed by this INI entry.
Note: In version 5.0 and higher versions of EasyCODE(COB), this option will be provided in the user interface („Source attributes...“ command in the „Options“ menu: „Replace blanks with TABs“ button).

42 OpenFileListLength

Components: All structure diagram editors, SD
Version: V5.0
File: EASY-<COMP>.INI
Section: [RecentFileList]
Value: number between 0 and 9
Default: 4

Example: OpenFileListLength=9

Consequences: Specifies the number of files opened last which will be added to the File menu to be opened directly.

43 InsertFileListLength

Components: All structure diagram editors
Version: V5.0
File: EASY-<COMP>.INI
Section: [RecentFileList]
Value: number between 0 and 9
Default: 4

Example: InsertFileListLength=9

Consequences: Specifies the number of files inserted last which will be added to the Insert menu to be inserted directly.

44 FtWFTRecSize

Components: All structure diagram editors, EasyFT
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: positive, whole number
Default: none

Example: FtWFTRecSize=1024

Consequences: Specifies the RecordSize for the remote file, if WFT is used as your file transfer system. If this entry does not exist, WFT will implicitly use a default value. If such an entry exists, its value will be given to WFT as an additional parameter when WFT is loaded.

45 FtWFTRecForm

Components: All structure diagram editors, EasyFT
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: u|v|f
Default: none

Example: FtWFTRecForm=u

Consequences: Specifies the RecordForm for the remote file, if WFT is used as your file transfer system. If this entry does not exist, WFT will implicitly use a default value. If such an entry exists, its value will be given to WFT as an additional parameter when WFT is loaded.

46 DataPool

Components: JET, PET
Version: V2.01 (JET) resp. V5.0 (PET)and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value (JET): Name of the Datapool program
Value (PET): User-ID of the Datapool server
Default: none

Example (JET): DataPool=$A.B0X085X
Example (PET): DataPool=$A

Consequences: If the entry is specified, the option Datapool is offered when editing Resources or conditions. In PET the entry SpecialCondition must be set to yes|1|true also. Starting from version V6.0 the value of this entry is put into the corresponding dialog box when editing a construct of this type the first time. (In earlier versions the value of this entry has been used only when generating the structure diagram.)
The leading $ for the User-ID of the datapool server in PET is optional, it will not be shown in the structure diagram and not be generated.

47 RestartLogic

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: RestartLogic=yes

Consequences: Specifies whether the condition "Restart mode" and the constructs "Jump to restart" and "Restart" should be offered. The entry SpecialCondition must be set to yes|1|true also.

48 Resource

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: Resource=yes

Consequences: Specifies whether the construct "Resource" should be offered.

49 ProgramCall

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: ProgramCall=yes

Consequences: Specifies whether the construct "Program call" should be offered.

50 SpecialConditions

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: SpecialConditions=yes

Consequences: Specifies whether the conditions File(Existence/Contents), JV(Existence/Contents), Job Switches, User Switches, Datapool(Contents) and Restart Mode should be offered

51 ErrorHandling

Components: PET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: ErrorHandling=yes

Consequences: Specifies whether the constructs Action Block and EXIT-ABNORMAL should be offered and whether the constructs Procedure and Internal Procedure should have an abnormal end.

52 Tabs

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: <n>
Default: 8

Example: Tabs=4

Consequences: Specifies the configuration of the Basic parser concerning tabulator spaces. When a source is read, tabulator signs will be replaced by the number of blanks required to reach the next tabulator position. The tabulator positions are multiples of the specified value.

53 RemoveSpaces

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: 0|1|2|3
Default: 2

Example: RemoveSpaces=3

Consequences: Specifies the configuration of the Basic parser concerning the elimination of leading blanks. When a source is read, leading blanks will be eliminated from every line of a text construct according to various strategies.
0 ... Leading blanks will not be eliminated.
1 ... The minimum number of leading blanks within a construct sequence will be eliminated.
2 ... The minimum number of leading blanks within a text construct will be eliminated.
3 ... All leading blanks will be eliminated.

54 Program

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: -1

Example: Program=500

Consequences: Specifies the configuration of the Basic parser concerning the refinement of the entire structure diagram. When a source is read, programs may be pushed down according to various strategies.

Effects: When you enter the value -1, the structure diagram will not be pushed down. When you enter the value 0, the structure diagram will always be pushed down. Any other whole number higher than 0 will have the effect that the structure diagram will be pushed down only when the specified number of lines is exceeded by this area. The following refinement strategy will be applied: An attempt is made to push down the structure diagram from the “bottom“; a refinement will be considered just one line in the next level. This means: If e.g. an IF construct, its THEN branch and its ELSE branch are to be pushed down only if they contain at least 20 lines each, and each of the two branches contains e.g. 11 lines, then the individual branches will not be pushed down, but the entire IF will be pushed down. If, however, the IF contains a THEN and an ELSE branch containing 25 lines each with the same refinement specifications, the individual branches, but not the entire IF, will be pushed down.

55 BeginEnd

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: BeginEnd=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of BEGIN...END. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

56 Trap

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: Trap=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of TRAP...END TRAP. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

57 Type

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: Type=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of TYPE...END TYPE. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

58 Function

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: Function=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of FUNCTION...END FUNCTION. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

59 PrivateFunction

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: PrivateFunction=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of PRIVATE FUNCTION...END FUNCTION. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

60 Sub

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: Sub=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of SUB...END SUB. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

61 PrivateSub

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 0

Example: PrivateSub=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of PRIVATE SUB...END SUB. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

62 SingleIfWithoutElse

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0
Default: -1

Example: SingleIfWithoutElse=0

Consequences: Specifies the configuration of the Basic parser concerning the refinement of one-line IF constructs without an ELSE branch. (In one-line IF constructs, both the THEN and the ELSE branch will be in the same line as the IF, there will not be an END IF). When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

63 SingleIfWithElse

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0
Default: -1

Example: SingleIfWithElse=0

Consequences: Specifies the configuration of the Basic parser concerning the refinement of one-line IF constructs with an ELSE branch. (In one-line IF constructs, both the THEN and the ELSE branch will be in the same line as the IF, there will not be an END IF). When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

64 BlockIf

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: BlockIf=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of IF...END IF without ELSEIF branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

65 MultipleIf

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: MultipleIf=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of IF...ELSEIF...END IF (with ELSEIF branches). When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

66 Then

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: Then=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of THEN branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

67 ElseIf

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: ElseIf=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of ELSEIF branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

68 Else

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: Else=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of ELSE branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

69 SelectCase

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: SelectCase=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of SELECT CASE...END SELECT ELSE. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

70 Case

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: Case=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of CASE branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

71 CaseElse

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: CaseElse=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of CASE ELSE branches. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

72 For

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: For=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of FOR...NEXT. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

73 While

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: While=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of WHILE...WEND. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

74 DoUntil

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: DoUntil=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of DO UNTIL...LOOP. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

75 DoWhile

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: DoWhile=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of DO WHILE...LOOP. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

76 Do

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: Do=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of DO...LOOP. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

77 LoopUntil

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: LoopUntil=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of DO...LOOP UNTIL. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

78 LoopWhile

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: -1|0|<n>
Default: 20

Example: LoopWhile=50

Consequences: Specifies the configuration of the Basic parser concerning the refinement of DO...LOOP WHILE. When a source is read, these constructs may be pushed down according to various strategies.

Effects: See chapter 54

79 Ex<n>

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Exceptions]
Value: <keyword>[,<keyword>]...
Default: none

Example: Ex1=ON OVERFLOW,OVERFLOW
Ex2=ON ERROR

Consequences: Defines an Exception for a Statement.

<n> represents a consecutive number of the Ex entries starting with 1, which must be unique for each entry.

The first keyword reflects the entire clause, in all other keywords the optional parts are omitted.

(Since there is a variety of COBOL dialects differing in their range of language, the COBOL parser must be adjusted to the dialect you use. These adjustments are made in the [Exceptions] and [Statements] section of the EASY-COB.INI file. The Exceptions of the COBOL dialect you use must be fully specified, so that the display of the Exception construct and the recognition of area limits will function correctly.)

80 Stmt<n>

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Statements]
Value: <keyword>[,<Exception number>]...[,<End-Statement>|*]
Default: none

Example: Stmt1=ACCEPT,2,*
Stmt2=READ,6,7,*
Stmt3=TRACE
Stmt4=PURGE,*

Consequences: Defines a Statement.

<n> represents a consecutive number of the Stmt entries starting with 1, which must be unique for each entry.

The <keyword> entry defines the Statement.

The <Exception number> entries specify which Exceptions - if any - are permitted for the Statement, with the number corresponding to the consecutive number of the corresponding entry in the [Exceptions] section.

The <End-Statement> entry defines the end statement of a Statement, if existing, with the end statement having the form END-<keyword>, if the '*' value is specified.

(Since there is a variety of COBOL dialects differing in their range of language, the COBOL parser must be adjusted to the dialect you use. These adjustments are made in the [Exceptions] and [Statements] section of the EASY-COB.INI file. The Statements of the COBOL dialect you use must be fully specified, so that the structure of the source code can be correctly identified, especially if optional parts of statements are omitted. If the message “Unknown statement starting with ..." appears during source analysis, the corresponding statement is not defined in the [Statements] section.)

81 CallException

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Statements]
Value: <n>[,<n>]...
Default: none

Example: CallException=1,2

Consequences: Defines the Exceptions permitted for a Call statement.

The values specify which Exceptions are permitted for the Call statement, with the number corresponding to the consecutive number of the corresponding entry in the [Exceptions] section.

(Since there is a variety of COBOL dialects differing in their range of language, the COBOL parser must be adjusted to the dialect you use. These adjustments are made in the [Exceptions] and [Statements] section of the EASY-COB.INI file. To ensure the correct display of the Exceptions of the Call statement, the corresponding exceptions must be specified.)

82 SearchException

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Statements]
Value: <n>[,<n>]...
Default: none

Example: SearchException=6

Consequences: Defines the Exceptions permitted for a Search statement.

The values specify which Exceptions are permitted for the Search statement, with the number corresponding to the consecutive number of the corresponding entry in the [Exceptions] section.

(Since there is a variety of COBOL dialects differing in their range of language, the COBOL parser must be adjusted to the dialect you use. These adjustments are made in the [Exceptions] and [Statements] section of the EASY-COB.INI file. To ensure the correct display of the Exceptions of the Search statement, the corresponding exceptions must be specified.)

83 MoveCommentIntoProgram

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: MoveCommentIntoProgram=yes

Consequences: Specifies whether comments before IDENTIFICATION DIVISION will be considered part of the Identification Division or remain before the COBOL program construct.

84 AssignCommentToNextDivision

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: AssignCommentToNextDivision=no

Consequences: Specifies whether comments at the end of a division will be considered part of the next division (with the exception of the last division).

85 AssignCommentToNextSection

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: AssignCommentToNextSection=no

Consequences: Specifies whether comments at the end of a section will be considered part of the next section (with the exception of the last section).

86 AssignCommentToNextParagraph

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: AssignCommentToNextParagraph=no

Consequences: Specifies whether comments at the end of a paragraph will be considered part of the next paragraph (with the exception of the last paragraph).

87 SeperateCommentBeforeConstruct

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: SeperateCommentBeforeConstruct=no

Consequences: Specifies whether comments before a construct will be separated from preceding text, i.e. placed in a separate statement construct.

88 NestCommentIntoLevel

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: NestCommentIntoLevel=no

Consequences: Specifies whether comments will be pushed down with a construct.

89 AllowSentenceInAArea

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: AllowSentenceInAArea=no

Consequences: Specifies whether statements beginning in the A area are allowed or whether in this case a warning is to be issued during analysis.

90 ExpandException

Components: COB (easy-cbl.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: ExpandException=no

Consequences: Specifies whether Exception clauses will be expanded during analysis, if they are abbreviated in the source. The expanded version will be considered the first alternative of an Exception clause in the Ex<n> entry (see there).

91 LineContinuation

Components: Basic parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: Character for line continuation
Default: none

Example: LineContinuation=_

Consequences: Defines which character indicates the continuation of lines. If this entry is empty, lines will not be continued. If it is not empty, it will specify the character indicating line continuation. (Line continuations are not supported e.g. in VisualBasic or AccessBasic, while WordBasic and MSTest version 3.0 and higher versions support them and for this purpose use the characters '\' and '_', respectively.)

92 OldMouseInterface

Components: SD
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: OldMouseInterface=yes

Consequences: Specifies whether the old or new mouse interface will be used. In EasyCODE(SD) V5.0, the mouse interface was changed to be able to assign context menus to the right mouse button, as is now common use. This entry makes the old mouse interface available again; in this case, the context menus will, however, not be available.

93 NestedComments

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: NestedComments=no

Consequences: Specifies whether nested comments will be allowed in Pascal or Modula/2 sources.

94 TempJV

Components: JET
Version: ab V5.0
File: EASY-<COMP>.INI
Section: [Settings]
Value: #|@
Default: #

Example: TempJV=@

Consequences: Specifies the default value of „Temporary files/Job variables“. The defined sign is the prefix for temporary files and temporary jobvariables.

95 ParserWINAPI

Components: COB (easy-par.dll)
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: ParserWINAPI=yes

Consequences: Specifies whether the current version of the Visual COBOL parser (EASY-PAR.DLL) already supports the WINAPI extensions, i.e. whether the USING keyword is not to be added to the parameter edges of the CALL and COBOL program constructs by the Visual COBOL parser interface (yes|true|1 ... add USING, no|false|0 ... do not add USING).

Note: This INI entry comes into effect only when the Visual COBOL parser (EASY-PAR.DLL) is used, if the EasyCODE(COB) internal parser (EASY-CBL.DLL) is used, this entry is ignored. See also the notes for EasyCODE(COB) concerning the INI entry „Parser“.

96 Keyword<n>

Components: C/C++
Version: V5.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <keyword>|<keyword>()
Default: none

Example: Keyword1=‘except()’

Consequences: <keyword> defines a keyword, the optional brackets following this entry indicate whether the keyword may be followed by a bracket expression belonging to the keyword. For a single keyword only one of the two options may be used; you must not use both of them at the same time.

<n> represents a consecutive number of the keyword entries starting with 1, which must be unique for each entry.

If you make these entries, the corresponding keywords (including all following comments) will be displayed in statement constructs, so that language extensions which otherwise might be misinterpreted will be displayed correctly (e.g. Structured Exception Handling, MFC extensions, etc.).

97 GenEndProgram

Components: COB
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: GenEndProgram=no

Consequences: Specifies whether the keyword "END PROGRAM" should be generated at the end of a COBOL-program. Some COBOL85 compilers do not support this keyword so you can switch off ist generation with this entry. In this case you can only use one COBOL Program construct in your structure diagram.

98 SaveAfterGen

Components: DS, COB, SPX
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: SaveAfterGen=no

Consequences: Specifies the status of the "incl save" checkbox in the "Generate as" dialog window after starting the the application

99 SaveAfterGenAll

Components: DS, COB, SPX, JET, PET
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: SaveAfterGenAll=no

Consequences: Specifies whether a file should be saved after generation with the "Generate all" command. To save is useful if the line numbers created during generation should be saved in the internal file format. On the other side you must not save the file, if the internal file should not be younger than the generated file.

100 PrtType

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: 1|2
Default: 1

Example: PrtType=2

Consequences: Specifies the default value for "Type" in the "Print" dialog window (1…Communication Plan, 2…Additional documents). This entry is will be read when the application is started.

101 PrtArea

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value (<V6.0): 1|3|4
Value (>=V6.0) 1|3|4|6
Default: 4

Example: PrtArea=1

Consequences: Specifies the default value for "Area" in the "Print" dialog window (1 … Current segment/level, 3 … Selected area, 4 … Entire structure diagram/communication plan, 6 ... Structure list). This entry is will be read when the application is started.

102 PrtLowerSegments

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PrtLowerSegments=no

Consequences: Specifies the default value for "Including lower segments/levels" in the "Print" dialog window. This entry is will be read when the application is started.

103 PrtMaxDepth

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: 0|<n>
Default: keiner

Example: PrtMaxDepth=3

Consequences: Specifies the default value for "Maximum segment/level depth " in the "Print" dialog window. This entry is will be read when the application is started.

104 PrtPageHeader

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PrtPageHeader=no

Consequences: Specifies the default value for "Page header" in the "Print" dialog window. This entry is will be read when the application is started.

105 PrtStartPageNum

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <n>
Default: 1

Example: PrtStartPageNum=27

Consequences: Specifies the default value for "Page numbers start at" in the "Print" dialog window. This entry is will be read when the application is started.

106 PrtFitIntoPage

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: PrtFitIntoPage=yes

Consequences: Specifies the default value for "Fit into page" in the "Print" dialog window. This entry is will be read when the application is started.

107 PrtMinFontSize

Components: All structure diagram editors, SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <n>
Default: 6

Example: PrtMinFontSize=1

Consequences: Specifies the default value for "Minimal font size" in the "Print" dialog window. This entry is will be read when the application is started.

108 PrtPreviewList

Components: All structure diagram editors
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: 0|1|2
Default: 0

Example: PrtPreviewList=2

Consequences: Specifies the default value for "Segments" in the Preview of the "Print" dialog window. This entry is will be read when the application is started.

109 RepPaths

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RepPaths=no

Consequences: Specifies the default value for "With paths" in the "Report" dialog window. This entry is used when the application is started with the "Report" option in the command line.

110 RepComments

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RepComments=no

Consequences: Specifies the default value for "With comments" in the "Report" dialog window. This entry is used when the application is started with the "Report" option in the command line.

111 RepDocs

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RepDocs=no

Consequences: Specifies the default value for "With supplementary documents" in the "Report" dialog window. This entry is used when the application is started with the "Report" option in the command line.

112 RepVarRefList

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RepVarRefList=no

Consequences: Specifies the default value for "Variable refinements" in the "Report" dialog window. This entry is used when the application is started with the "Report" option in the command line.

113 RepAlphaList

Components: SD
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RepAlphaList=no

Consequences: Specifies the default value for "Alphabetic lists" in the "Report" dialog window. This entry is used when the application is started with the "Report" option in the command line.

114 CtrlZ

Components: COB (easy-cbl.dll)
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: Err|Eof|EofAtEnd
Default: EofAtEnd

Example: CtrlZ=Err

Consequences: Specifies the behaviour of the parser when the sign ^Z (0x1A) occurs in the source file.
CtrlZ=Err means that an error message is created when ^Z occurs in the source file.
CtrlZ=Eof means that the first ^Z in the file stands for end-of-file. If this ^Z sign is not the end of the file, the source code may not be correct.
CtrlZ=EofAtEnd means that the ^Z sign is interpreted as end-of-file when it is the last sign in the file and therefore no error message is created. If the sign occurs within the file an error message is created.

115 ECComment (SPX parser)

Components: ABAP parser, Basic parser (SPX)
Version: V5.11 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: "<text>"
Default: "Easy"

Example: ECComment="######"

Consequences: Specifies the layout of EasyCODE comments. This entry is meaningful during analysis of a source file. The layout of the EasyCODE comments in the source can be specified for the parser using this entry. When generating a source the EasyCODE comments are created by the corresponding GenStrings. Remember that the leading comment symbol, the following blank as the terminating symbol for the type the comment and the preceding blank can not be modified by this entry.Only the text in between is modified. Those parts that can not be modified are defined by the parser and have to be generated by the GenStrings accordingly. The text specified by this entry must also match with those defined in the GenStrings.
The following characters (with the described meaning) may be used as the terminating symbol for the type of the EasyCODE comment:
„V“ (with succeeding EasyCODE(SPX) version, modification date and short information) means that these informationen is not interpreted by the parser. Furthermore not automatic segmentation is made. The segmentation is created by the corresponding EasyCODE comment.
„-“ seperates two statement constructs.
„(“ indicates the start of a segment. The segment header follows this symbol on the same line. Succeeding „(“-comments are interpreted as one segment start with a multiline segment header.
„)“ indicates the end of a segment.
„:“ indicates the start of a block.
„;“ indicates the end of a block.
„!“ (ABAP) or „?“ (Basic) indicate the start of the body of a segment or block.

116 ECComment

Components: ASM, COL, C, C++, DS
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: "<text>"
Default (ASM): "---------- EASYCODE(ASM)"
Default (COL): "---------- EASYCODE(COL)"
Default (C/C++):"EasyCODE"
Default (DS): "EasyCODE(DS)"

Example: ECComment="######"

Consequences: Specifies the look of the EasyCODE comments. This entry is used for analysis as well as for the generation of the file.

117 AltECComment

Components: ASM, COL, C, C++, DS
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: "<text>"
Default (ASM): "---------- EasyCODE(ASM)"
Default (COL): "---------- EasyCODE(COL)"
Default (C/C++):"EasyCODE"
Default (DS): "EasyCODE(DS)"

Example: ECComment="###"

Consequences: Specifies an alternative EasyCODE-comment an. This entry is used only for analysis of a file to have compatibility to previous EasyCODE formats.

118 AlignTextLines

Components: COB (easy-cbl.dll)
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: AlignTextLines=yes

Consequences: Specifies whether the lines in a text construct should be aligned or left as they are in the source file.

119 WrapComments

Components: PET
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: WrapComments=yes

Consequences: Specifies whether comments should be overrun according to the actual indent and the maximum line length for SDF commands/statements when a JOB file is generated.

120 CriticalPrograms

Components: JET, PET
Version: V5.01 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <programmname>[,<programmname>,...]
Default: keiner

Example: CriticalPrograms=PERCON

Consequences: Specifies programs that have critical statements concerning the line make-up because white spaces are created when the following line is indented. This white spaces may cause programm errors when the file is generated. Statements for programs that are specified with this entry will not be indented in folllowing lines.

121 InLineComment

Components: COB (easy-cbl.dll)
Version: V5.11 and higher versions
File: EASY-<COMP>.INI
Section: [ParseOptions]
Value: "<text>"
Default: none

Example: InLineComment ="*>"

Consequences: Specifies whether in-line comments are supported and specifies the character string indicating these comments.

122 SdfDoorsDll

Components: JET, PET
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <full path name>
Default: none

Example: SdfDoorsDll =C:\ SNICOMSV\SDF.DLL

Consequences: Specifies the full path name of the SDF-Doors DLL. If this entry contains a valid path name of an SDF-Doors DLL, the SDF-Doors cooperation is enabled. That means that the editor for SDF commands and statements contained in SDF-Doors is used. Furthermore the SDF-Doors specific menu items (e.g. BS2000 file access) and functionality (e.g. on-line syntax file) are enabled. If this entry is empty or invalid, the SDF editor contained in EasyCODE(JET/PET) is used, the SDF-Doors specific menu items and functionality are not available.

123 PrtSaveSpace

Components: All structure diagram editors
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: PrtSaveSpace=no

Consequences: Specifies the default value for "Save space" in the "Print" dialog window. This entry is will be read when the application is started.

124 PushDownVar

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownVar=yes

Consequences: Specifies whether variable declarations should be pushed down.

125 PushDownConst

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownConst=yes

Consequences: Specifies whether constants declarations should be pushed down.

126 PushDownType

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownType=yes

Consequences: Specifies whether type declarations should be pushed down.

127 PushDownProcedureBody

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownProcedureBody=yes

Consequences: Specifies whether the body of procedure should be pushed down.

128 PushDownFunctionBody

Components: Pascal parser, Modula/2 parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownFunctionBody=yes

Consequences: Specifies whether the body of a function should be pushed down.

129 PushDownInterface

Components: Pascal parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownInterface=yes

Consequences: Specifies whether interfaces should be pushed down.

130 PushDownInitialization

Components: Pascal parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownInitialization=yes

Consequences: Specifies whether initializations should be pushed down.

131 PushDownImplementation

Components: Pascal parser (SPX)
Version: V4.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: PushDownImplementation=yes

Consequences: Specifies whether implementations should be pushed down.

132 OnlineSFCheck

Components: JET, PET
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: OnlineSFCheck=no

Consequences: Specifies whether all SDF commands/statements should be checked when generating if the online syntax file is used, or just those that have been modified. Checking all commands/statements is necessary if changes in the content of the online syntax file or changes of privileges of the BS2000 connection are possible (default). To increase performance checking of commands/statements may be restricted to those commands/statements that have changed. For security reasons this is not recommended (errors resulting from changes in the syntax file or the privileges may not be recognized).

133 EncloseResource

Components: PET
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: EncloseResource=yes

Consequences: Specifies whether the constructs resulting from the construct „Resource“ should be enclosed by a BEGIN-BLOCK/END-BLOCK construct.

134 EncloseProgramCall

Components: PET
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: EncloseProgramCall=yes

Consequences: Specifies whether the constructs resulting from the construct „Program Call“ should be enclosed by a BEGIN-BLOCK/END-BLOCK construct.

135 CaseAsFrames

Components: Modula/2 parser (SPX)
Version: V5.1 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: CaseAsFrames=yes

Consequences: Specifies whether the statement CASE should be converted into nested Frame consructs in the structure diagram. Otherwise the statement CASE is converted into the construct CASE. (By default the „natural“ conversion to a CASE construct is used. In Modula/2 this conversion may be considered inappropriate for the specific reason that some Modula/2 compilers make a difference between an empty and a missing ELSE branch.)

136 HelpFile<n>

Components: All structure diagram editors
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [HelpFileList]
Value: [<alias for help file>,]<full path name of help file>
Default: none

Example: HelpFile1=MFC Reference,c:\msvc151\help\mfc.hlp

Consequences: Defines a help file for the functionality „Help on selected text“.

<n> represents a consecutive number of the HelpFile entries starting with 1, which must be unique for each entry.

The first optional part defines the alias for the help file to be used in the corresponding pop-up menu. The second mandatory part defines the full path name of the help file. Both parts are seperated by a comma. If the first part is missing, the second one will be used for the pop-up menu too.

137 EditRedimX

Components: All structure diagram editors
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <n>
Default: 8

Example: EditRedimX=1

Consequences: When direct editing in the structure diagram is used this entry defines the amount for resizing the edit control in x-direction if a line becomes to long to be fully displayed in the current edit control. Possible values: 1 to 65535, units: average width of a character.

This entry may be set to 1 on high-performance PCs to make the resizing more smooth. On slow machines a larger value should be used to prevent interruption of editing due to the slow redimensioning and repainting.

138 EditRedimY

Components: All structure diagram editors
Version: V6.0 and higher versions
File: EASY-<COMP>.INI
Section: [Settings]
Value: <n>
Default: 1

Example: EditRedimY=1

Consequences: When direct editing in the structure diagram is used this entry defines the amount for resizing the edit control in y-direction if the number of lines becomes to large to be fully displayed in the current edit control. Possible values: 1 to 65535, units: line height.

This entry may be set to 1 on high-performance PCs to make the resizing more smooth. On slow machines a larger value should be used to prevent interruption of editing due to the slow redimensioning and repainting.

139 InlineAssembler

Components: C/C++
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: "<keyword>"
Default: "asm"

Example: InlineAssembler=""

Consequences: Defines the character string that is used to recognize Inline-Assembler blocks in a C/C++ source. If this entry is present but empty the recognition of Inline-Assembler blocks is disabled completly. The keyword „asm“ (used in MS VC++) is treated as normal text in this case. If this entry is not empty it is used instead of the default character string for the recognition of Inline-Assembler blocks.

140 PrintMono

Components: All structure diagram editors
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: PrintMono=yes

Consequences: Specifies whether the structure diagram should be printed in black and white even if there are other colors specified. This option is used to get a readable printout if there are color combinations defined for the screen for optical reasons, that would produce a non readable printout.

141 Tab2Space

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: <n>
Default: 8

Example: Tab2Space=4

Consequences: Specifies the configuration of the PL1/SPL4 parser concerning tabulator spaces. When a source is read, tabulator signs will be replaced by the number of blanks required to reach the next tabulator position. The tabulator positions are multiples of the specified value.

142 BeepOnLines

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: <n>
Default: 0

Example: BeepOnLines=100

Consequences: A value greater than 0 means that the PL1/SPL4 parser generates a beep on all lines that are multiples of the given value. A value of 0 means that no beep is generated.

143 FirstCol

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: natürliche Zahl
Default: 2

Example: FirstCol=1

Consequences: Defines the first column in the source to be interpreted. All columns before are truncated. (This entry is necessary because PL1/SPL4 sources are column orientated.)

144 LastCol

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: natürliche Zahl
Default: 72

Example: LastCol=80

Consequences: Defines the last column in the source to be interpreted. All columns behind are truncated. (This entry is necessary because PL1/SPL4 sources are column orientated.)

145 IgnoreEntry

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: IgnoreEntry=no

Consequences: Specifies whether the Entry statement should be ignored. If the Entry statement is not ignored it will be converted to a frame construct. For the Entry statement may cause recognition and representation problems for technical reasons it is possible to ignore the statement and treat it as normal text. For security reasons this is the default behaviour.

146 LowerText

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: LowerText=yes

Consequences: Specifies whether the PL1/SPL4 parser should put large text constructs into seperate segments.

147 PROC_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: yes

Example: PROC_Level=no

Consequences: Specifies whether the PL1/SPL4 parser should put procedures and functions into seperate segments.

148 THEN_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: THEN_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put THEN branches into seperate segments.

149 ELSE_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: ELSE_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put ELSE branches into seperate segments.

150 WHEN_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: WHEN_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put WHEN branches into seperate segments.

151 WHILE_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: WHILE_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put WHILE bodies into seperate segments.

152 TO_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: TO_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put TO bodies into seperate segments.

153 UNTIL_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: UNTIL_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put UNTIL bodies into seperate segments.

154 ENTRY_Level

Components: PL1/SPL4-Parser (SPX)
Version: V6.0 and higher versions
File: Configuration file (.CFG)
Section: [ParseOptions]
Value: yes|no|true|false|1|0
Default: no

Example: ENTRY_Level=yes

Consequences: Specifies whether the PL1/SPL4 parser should put ENTRY bodies into seperate segments.

155 EmptyLineBeforeECComment

Components: C/C++
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: EmptyLineBeforeECComment =yes

Consequences: Specifies whether an empty line should be generated before specific EasyCODE comments (segments, functions, etc.) for optical reasons.

156 RemoveEmptyLines

Components: ASM
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: yes

Example: RemoveEmptyLines =no

Consequences: Specifies whether empty lines should be deleted after editing a text.

157 BrowserSupportDef

Components: C/C++, COB, SPX
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: BrowserSupportDef =yes

Consequences: Specifies whether the menu item „View-Definition“ is visible. If a Browser DLL is configured and this entry is set to „yes“, the menu item is visible. If no Browser DLL is configured and this entry is set to „yes“, the menu item is visible but grayed. If this entry is set to „no“, the menu item is not visible.

158 BrowserSupportRef

Components: C/C++, COB, SPX
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: BrowserSupportRef =yes

Consequences: Specifies whether the menu item „View-References“ is visible. If a Browser DLL is configured and this entry is set to „yes“, the menu item is visible. If no Browser DLL is configured and this entry is set to „yes“, the menu item is visible but grayed. If this entry is set to „no“, the menu item is not visible.

159 BrowserDLL

Components: C/C++, COB, SPX
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: Filename of Browser DLL
Default: None

Example: BrowserDLL =c:\easy\easy-pv.dll

Consequences: Specifies the name of the Browser DLL. If a Browser DLL is configured and the correct entry points are provided, it is possible to activate the Browser functionality (View-Definition/References).

160 AddInMenu

This entry is used for the Add-In-Interface. It is documented there.

161 AddInCmd<n>

This entry is used for the Add-In-Interface. It is documented there.

162 MouseCmd<n>

This entry is used for the Add-In-Interface. It is documented there.

163 SpecialLines

This entry is used for the Add-In-Interface. It is documented there.

164 PrintDelay

Components: All structure diagram editors, SD
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: <n>
Default: 0

Example: PrintDelay=5

Consequences: Specifies minimal duration in seconds for a printout to last from the begin to the end. If a printout is faster, the the remaining time will be consumed before the printout is finished. This delay may be necessary, if small printouts are lost (due to an operating system bug, e.g. on WindowsNT 3.50 with a local printer and a small printout).

165 PreprocessorColumn

Components: C/C++
Version: V6.0 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: <n>
Default: -1

Example: PreprocessorColumn=1

Consequences: Specifies the column for preprocessor statements. If n is greater or equal 1, preprocessor statements will be generated exactly in this column when saving the structure diagram. Otherwise (n less than 1) the preprocessor statements will be generated using the same indent as for normal code. (This entry may be necessary for some old compilers, that accept preprocessor statements only in certain columns, e.g. column 1.)

166 FtCommand

Components: All structure diagram editors, EasyFT
Version: V6.01 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: "<command name>"
Default: "DOSNCP" for FT-PCD (SNI), "DFT" for DFT (INTRA-SYS), "NCOPY" for SIAM-FT-DOS (COMPUTER KONTOR)

Example: FtCommand="ncopy"

Consequences: Specifies the filetransfer command to be used for the filetransfers FT-PCD, DFT, or SIAM-FT-DOS. (Note: This entry is only valid for these filetransfer products!) For some filetransfer products may change the names of the filetransfer commands from version to version (e.g. FT-PCD: older versions NCOPY, now DOSNCP), it is possible to make the necessary modifications via this entry.

167 JavaMode

Components: C++
Version: V6.001 and higher versions
File: EASY-<KOMP>.INI
Section: [Settings]
Value: yes|no|true|false|1|0
Default: no

Example: JavaMode=yes

Consequences: Specifies whether the component C++ should work in Java mode. The programming language Java is similar to C++ in many areas. It differs from C++ only in a few details. So Java sources can be edited using the component C++ if it is switched to Java mode with this entry. In Java mode some constructs are not present, the default filename extension is different, and some keywords are treated in a different way.

168 Command Line Options and Parameters

The following options and parameters may be entered into the command line for loading EasyCODE and will modify application behavior accordingly. The bold characters **{ } | [ ]** in the syntax description are metasigns and not part of the option or parameter (meaning: **{ | }** ... alternatives, **[ ]** optional elements), text in italics in the syntax description indicates a value to be specified. In general, all options and parameters are case-insensitive.

169 <file_name>

Components: All structure diagram editors, SD
Version: V3.1 and higher versions
Syntax: *<file_name>*

Example: easy-sp.exe myfile.sp

Consequences: The first parameter in the command line (argument not preceded by / or -) will be interpreted as the filename. The application tries to open this file immediately after its start.

170 Embedding

Components: All structure diagram editors
Version: V3.5 and higher versions
Syntax: **{**-|/**}**Embedding

Example: easy-sp.exe /Embedding

Consequences: This option is reserved for OLE; with this option the OLE client application calls the OLE server application. Therefore, this option must ***not*** be specified by the user! (If the application is called with this option by the user instead of the OLE client application, the result will be unpredictable behavior!)

171 Initialize

Components: All structure diagram editors, SD
Version: V3.5 and higher versions
Syntax: **{**-|/**}**Initialize

Example: easy-sp.exe /Initialize

Consequences: When this option is used, the Registration Database and the WIN.INI file will be re-initialized by the application after its call. The same entries will be made which are also made during the setup procedure, while entries modified by the user will be overwritten. In this case, the application will not open a window, but will be closed immediately after the appropriate entries have been updated. (This option is also be used during single-user setup and workstation setup to make the appropriate entries.)

172 Print

Components: All structure diagram editors, SD
Version: V3.5 and higher versions
Syntax: **{**-|/**}**P**[**rint**]**

Example: easy-sp.exe /p abc.sp

Consequences: If this option is used, the specified file will be printed. The main window of the application will not be opened, only the dialog box for cancelling printing will appear. The application will be closed immediately after printing. The following applies to V4.0 and higher versions: By default, the standard printer settings will be used. You may, however, also make an appropriate entry into the INI file so that the Print dialog window for modifying the printer settings will be opened (see INI entry PrintFileStandard). To V3.5x, the following applies: The Print dialog window will always appear.

173 PrintStructure

Components: All structure diagram editors
Version: V6.0 and higher versions
Syntax: {-|/}P[rint]S[tructure][=<*index*>]

Example: easy-spx.exe /ps=1 abc.spx

Consequences: If this option is used, the structure list of the file given as a parameter will be printed. If an index is specified within this option, the structure list specified by this index will be printed. (The index is defined by the order of the entries in the combo box for the type of the structure list contained in the structure list window. The index starts with 0, which is also the default, if no index is given.) As with the option Print the following applies: The main window of the application will not be opened, only the dialog box for cancelling printing will appear. The application will be closed immediately after printing. By default, the standard printer settings will be used. You may, however, also make an appropriate entry into the INI file so that the Print dialog window for modifying the printer settings will be opened (see INI entry PrintFileStandard). The same effect as with this option may also be achieved using the option Print with the INI entries PrtArea and LevelListType (is created by „Save settings...“) set properly.

174 Report

Components: SD
Version: V5.01 and higher versions
Syntax: {-|/}R[eport]

Example: easy-sd.exe /r abc.sd

Consequences: If this option is used, the Report for the specified file will be printed. The main window of the application will not be opened, only the dialog box for cancelling printing will appear. The application will be closed immediately after printing.
By default, the standard report settings that can be modified in the Ini file (see entrys "Rep…) will be used. The settings can also be made in the Report dialog window that appears when the corresponding entry in the Ini file is set (see PrintReportStandard).

175 Line/Construct

Components: All structure diagram editors
Version: V4.0 and higher versions
Syntax: {-|/}L[ine]=<*line_number*>

Example: easy-c.exe /l=200 abc.c

Consequences: If this option is used, the application tries to display this line number immediately after the specified file has been started. If the line number does not exist, the preceding line number will be displayed automatically (without further inquiry with the help of a message window, because it is not possible in the initialization stage for technical reasons). If you start the application without specifying a filename or with a file that does not include line numbers, this option will be ignored. (The latter always applies to SP.)

176 Inifile

Components: All structure diagram editors, SD
Version: V4.0 and higher versions
Syntax: {-|/}I[nifile]=<*file_name*>

Example: easy-spx.exe /i=spx-pas.ini

Consequences: If this option is used, the application will not use the default INI file (EASY-<COMP>.INI stored in the Windows directory), but the file specified in this option as its INI file.

(Especially in SPX, this allows different INI files to be used for different languages or, more generally, different INI files to be used by different persons.)

177 Generate

Components: DS, SPX, COB, JET, PET
Version: V5.01 and higher versions
Syntax: **{**-|/**}G[**enerate**][**=<*dateiname*>**]**

Example: easy-spx.exe /g=abc.txt abc.spx

Consequences: This option causes the generation of the specified file. If a filename for the generated file is also specified this name will be used (addition of working directory if no path is specified). Otherwise the name for the generated file will be created from the file to be generated, the path for the source files (from the Ini file) and the file extension (also from the Ini file).

178 GenerateAll

Components: DS, SPX, COB, JET, PET
Version: V5.01 and higher versions
Syntax: **{**-|/**}G[**enerate**]a[**ll**]**

Example: easy-spx.exe /ga

Consequences: This option causes the execution of the "Generate all" command. See also the Help function.

179 Save

Components: All structure diagram editors
Version: V5.01 and higher versions
Syntax: **{**-|/**}S[**ave**][**=<*file_name*>**]**

Example: easy-spx.exe /s=abc.spx abc.etf

Consequences: This option causes the saving of the specified file. If a filename for the saved file is also specified this name will be used (addition of working directory if no path is specified). Otherwise the name for the saved file will be created from the file to be saved, the path for the internal files (resp. source files for C++, ASM, COL) (from the Ini file) and the file extension (also from the Ini file).

180 Export

Components: All structure diagram editors
Version: V5.01 and higher versions
Syntax: **{**-|/**}E[**xport**][**=<*file_name*>**]**

Example: easy-spx.exe /e=abc.etf abc.spx

Consequences: This option causes the export of the specified file. If a filename for the exported file is also specified this name will be used (addition of working directory if no path is specified). Otherwise the name for the exported file will be created from the file to be exported, the path for the ETF files (from the Ini file) and the file extension (also from the Ini file).

181 OpenFileDlg

Components: All structure diagram editors, SD
Version: V6.0 and higher versions
Syntax: {-|/}O[penFileDlg][=[<*path*>][<*filter*>]]

Example: easy-spx.exe /o=c:\work*.clp

Consequences: This option causes the „Open file“ dialog box to be shown immediately after the start of the application. If present the path and the filter given in this option are used for initializing the dialog box (directory, filter). A given filter is ignored in SD, for only *.SD is allowed as a file extension in SD.

182 ReadOnly

Components: All structure diagram editors, SD
Version: V6.0 and higher versions
Syntax: {-|/}R[ead]O[nly]

Example: easy-sp.exe /ro abc.spx

Consequences: This option causes the file given to the application via the command line to be opened in read-only mode. The option is only available in the network installation, for the single user installation does not support the read-only mode when opening files.

183 Project

Components: C/C++, COB, SPX
Version: V6.0 and higher versions
Syntax: {-|/}Pr[o]j[ect]

Example: easy-cpp.exe /prj=d:\myproj\myproj.pvx

Consequences: This option causes the project database given to the application via this command line option to be used instead of that from the INI file.

184 AddIn

This command line option is used for the Add-In-Interface. It is documented there.