APPENDIX B

BNF Grammar

This appendix provides a complete description of symbols, operators, and directives for MASM 6.1. It uses the Backus-Naur Form (BNF) for grammar notation. You can use BNF grammar to determine the exact syntax for any language component and find all available options for any MASM command.

BNF definitions consist of "nonterminals" and "terminals." Nonterminals are placeholders within a BNF definition, defined elsewhere in the BNF grammar. Terminals are endpoints in a BNF definition, consisting of MASM 6.1 keywords. In this Appendix, all nonterminals appear in italics type and all terminals appear in bold type.

BNF Conventions

The conventions use different font attributes for different items in the BNF. The symbols and formats are as follows:

Attribute	Description
nonterminal	Italic type indicates nonterminals.
RESERVED	Terminals in boldface type are literal reserved words and symbols that must be entered as shown. Characters in this context are always case insensitive.
[[ ]]	Objects enclosed in double brackets ([[ ]]) are optional. The brackets do not actually appear in the source code.
|	A vertical bar indicates a choice between the items on each side of the bar.
.8086	Underlined items indicate the default option if one is given.
default typeface	Characters in the set described or listed can be used as terminals in MASM statements.

How to Use the BNF Grammar

To illustrate the use of the BNF, Figure B.1 diagrams the definition of the **TYPEDEF** directive, starting with the nonterminal *typedefDir*.

The entries under each horizontal brace in Figure B.1 are terminals (such as **NEAR16**, **NEAR32**, **FAR16**, and **FAR32**) or nonterminals (such as *qualifier*, *qualifiedType*, *distance*, *and protoSpec*) that can be further defined. Each italicized nonterminal in the *typedefDir* definition is also an entry in the BNF. Three vertical dots indicate a branching definition for a nonterminal that, for the sake of simplicity, this figure does not illustrate.

The BNF grammar allows recursive definitions. For example, the grammar uses *qualifiedType* as a possible definition for *qualifiedType*, which is also a component of the definition for *qualifier*.

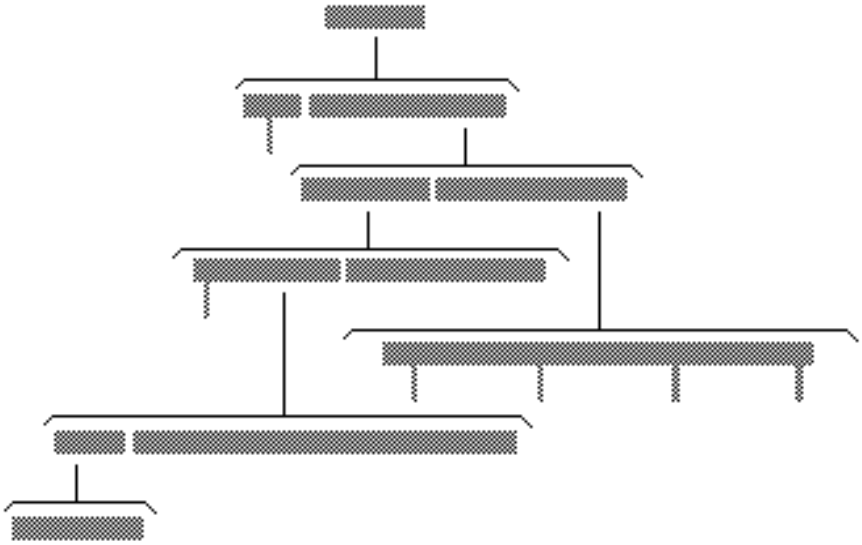

Figure B.1 BNF Definition of the TYPEDEF Directive

Nonterminal	Definition
;;	endOfLine | *comment*
=Dir	*id* = *immExpr ;;*
addOp	+ | -
aExpr	*term* | *aExpr* **&&** *term*

Nonterminal	Definition
altId	*id*
arbitraryText	*charList*
asmInstruction	*mnemonic* ⟦ *exprList* ⟧
assumeDir	**ASSUME** *assumeList ;;* | **ASSUME NOTHING** *;;*
assumeList	*assumeRegister* | *assumeList* **,** *assumeRegister*
assumeReg	*register* **:** *assumeVal*
assumeRegister	*assumeSegReg* | *assumeReg*
assumeSegReg	*segmentRegister* **:** *assumeSegVal*
assumeSegVal	*frameExpr* | **NOTHING** | **ERROR**
assumeVal	*qualifiedType* | **NOTHING** | **ERROR**
bcdConst	⟦ *sign* ⟧ *decNumber*
binaryOp	== | **!=** | >= | <= | > | < | **&**
bitDef	*bitFieldId* **:** *bitFieldSize* ⟦ = *constExpr* ⟧
bitDefList	*bitDef* | *bitDefList* **,** ⟦ *;;* ⟧ *bitDef*
bitFieldId	*id*
bitFieldSize	*constExpr*
blockStatements	*directiveList* | **.CONTINUE** ⟦ **.IF** *cExpr* ⟧ | **.BREAK** ⟦ **.IF** *cExpr* ⟧
bool	**TRUE** | **FALSE**
byteRegister	AL | AH | BL | BH | CL | CH | DL | DH
cExpr	*aExpr* | *cExpr* || *aExpr*
character	Any character with ordinal in the range 0–255 except linefeed (10)
charList	*character* | *charList character*
className	*string*
commDecl	⟦ *nearfar* ⟧ ⟦ *langType* ⟧ *id* **:** *commType* ⟦ **:** *constExpr* ⟧
commDir	**COMM** *commList ;;*
comment	**;** *text ;;*

Nonterminal	Definition
commentDir	**COMMENT** *delimiter* *text* *text delimiter text ;;*
commList	*commDecl* | *commList* **,** *commDecl*
commType	*type* | *constExpr*
constant	*digits* ⟦ *radixOverride* ⟧
constExpr	*expr*
contextDir	**PUSHCONTEXT** *contextItemList ;;* | **POPCONTEXT** *contextItemList ;;*
contextItem	**ASSUMES** | **RADIX** | **LISTING** | **CPU** | **ALL**
contextItemList	*contextItem* | *contextItemList* **,** *contextItem*
controlBlock	*whileBlock* | *repeatBlock*
controlDir	*controlIf* | *controlBlock*
controlElseif	**.ELSEIF** *cExpr ;;* *directiveList* ⟦ *controlElseif* ⟧
controlIf	**.IF** *cExpr ;;* *directiveList* ⟦ *controlElseif* ⟧ ⟦ **.ELSE** *;;* *directiveList* ⟧ **.ENDIF** *;;*
coprocessor	**.8087** | **.287** | **.387** | **.NO87**
crefDir	*crefOption ;;*
crefOption	**.CREF** | **.XCREF** ⟦ *idList* ⟧ | **.NOCREF** ⟦ *idList* ⟧
cxzExpr	*expr* | **!** *expr* | *expr* == *expr* | *expr* **!=** *expr*
dataDecl	**DB** | **DW** | **DD** | **DF** | **DQ** | **DT** | *dataType* | *typeId*
dataDir	⟦ *id* ⟧ *dataItem ;;*

Nonterminal	Definition
dataItem	*dataDecl scalarInstList* | *structTag structInstList* | *typeId structInstList* | *unionTag structInstList* | *recordTag recordInstList*
dataType	**BYTE** | **SBYTE** | **WORD** | **SWORD** | **DWORD** | **SDWORD** | **FWORD** | **QWORD** | **TBYTE** | **REAL4** | **REAL8** | **REAL10**
decdigit	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
decNumber	*decdigit* | *decNumber decdigit*
delimiter	Any *character* except *whiteSpaceCharacter*
digits	*decdigit* | *digits decdigit* | *digits hexdigit*
directive	*generalDir* | *segmentDef*
directiveList	*directive* | *directiveList directive*
distance	*nearfar* | **NEAR16** | **NEAR32** | **FAR16** | **FAR32**
e01	*e01 orOp e02* | *e02*
e02	*e02* **AND** *e03* | *e03*
e03	**NOT** *e04* | *e04*
e04	*e04 relOp e05* | *e05*
e05	*e05 addOp e06* | *e06*
e06	*e06 mulOp e07* | *e06 shiftOp e07* | *e07*
e07	*e07 addOp e08* | *e08*
e08	**HIGH** *e09* | **LOW** *e09* | **HIGHWORD** *e09* | **LOWWORD** *e09* | *e09*

Nonterminal	Definition
e09	**OFFSET** *e10* | **SEG** *e10* | **LROFFSET** *e10* | **TYPE** *e10* | **THIS** *e10* | *e09* **PTR** *e10* | *e09* **:** *e10* | *e10*
e10	*e10* **.** *e11* | *e10* **[[** *expr* **]]** | *e11*
e11	**(** *expr* **)** | **[[** *expr* **]]** | **WIDTH** *id* | **MASK** *id* | **SIZE** *sizeArg* | **SIZEOF** *sizeArg* | **LENGTH** *id* | **LENGTHOF** *id* | *recordConst* | *string* | *constant* | *type* | *id* | **$** | *segmentRegister* | *register* | **ST** | **ST (** *expr* **)**
echoDir	**ECHO** *arbitraryText ;;* **%OUT** *arbitraryText ;;*
elseifBlock	*elseifStatement ;;* *directiveList* **[[** *elseifBlock* **]]**
elseifStatement	**ELSEIF** *constExpr* | **ELSEIFE** *constExpr* | **ELSEIFB** *textItem* | **ELSEIFNB** *textItem* | **ELSEIFDEF** *id* | **ELSEIFNDEF** *id* | **ELSEIFDIF** *textItem* **,** *textItem* | **ELSEIFDIFI** *textItem* **,** *textItem* | **ELSEIFIDN** *textItem* **,** *textItem* | **ELSEIFIDNI** *textItem* **,** *textItem* | **ELSEIF1** | **ELSEIF2**

Nonterminal	Definition
endDir	**END** ⟦ *immExpr* ⟧ *;;*
endpDir	*procId* **ENDP** *;;*
endsDir	*id* **ENDS** *;;*
equDir	*textMacroId* **EQU** *equType ;;*
equType	*immExpr* | *textLiteral*
errorDir	*errorOpt ;;*
errorOpt	**.ERR** ⟦ *textItem* ⟧ | **.ERRE** *constExpr* ⟦ *optText* ⟧ | **.ERRNZ** *constExpr* ⟦ *optText* ⟧ | **.ERRB** *textItem* ⟦ *optText* ⟧ | **.ERRNB** *textItem* ⟦ *optText* ⟧ | **.ERRDEF** *id* ⟦ *optText* ⟧ | **.ERRNDEF** *id* ⟦ *optText* ⟧ | **.ERRDIF** *textItem* , *textItem* ⟦ *optText* ⟧ | **.ERRDIFI** *textItem* , *textItem* ⟦ *optText* ⟧ | **.ERRIDN** *textItem* **,** *textItem* ⟦ *optText* ⟧ | **.ERRIDNI** *textItem* **,** *textItem* ⟦ *optText* ⟧ | **.ERR1** ⟦ *textItem* ⟧ | **.ERR2** ⟦ *textItem* ⟧
exitDir	**.EXIT** ⟦ *expr* ⟧ *;;*
exitmDir:	**EXITM** | **EXITM** *textItem*
exponent	E ⟦ *sign* ⟧ *decNumber*
expr	**SHORT** *e05* | **.TYPE** *e01* | **OPATTR** *e01* | *e01*
exprList	*expr* | *exprList* **,** *expr*
externDef	⟦ *langType* ⟧ *id* ⟦ **(** *altId* **)** ⟧ **:** *externType*
externDir	*externKey externList ;;*
externKey	**EXTRN** | **EXTERN** | **EXTERNDEF**
externList	*externDef* | *externList* **,** ⟦ *;;* ⟧ *externDef*
externType	**ABS** | *qualifiedType*
fieldAlign	*constExpr*
fieldInit	⟦ *initValue* ⟧ | *structInstance*

Nonterminal	Definition
fieldInitList	*fieldInit* | *fieldInitList* **,** ⟦ *;;* ⟧ *fieldInit*
fileChar	*delimiter*
fileCharList	*fileChar* | *fileCharList fileChar*
fileSpec	*fileCharList* | *textLiteral*
flagName	**ZERO?** | **CARRY?** | **OVERFLOW?** | **SIGN?** | **PARITY?**
floatNumber	⟦ *sign* ⟧ *decNumber* **.** ⟦ *decNumber* ⟧ ⟦ *exponent* ⟧ | *digits* R | *digits* r
forcDir	**FORC** | **IRPC**
forDir	**FOR** | **IRP**
forParm	*id* ⟦ **:** *forParmType* ⟧
forParmType	**REQ** | = *textLiteral*
frameExpr	**SEG** *id* | DGROUP **:** *id* | *segmentRegister* **:** *id* | *id*
generalDir	*modelDir* | *segOrderDir* | *nameDir* | *includeLibDir* | *commentDir* | *groupDir* | *assumeDir* | *structDir* | *recordDir* | *typedefDir* | *externDir* | *publicDir* | *commDir* | *protoTypeDir* | *equDir* | *=Dir* | *textDir* | *contextDir* | *optionDir* | *processorDir* | *radixDir* | *titleDir* | *pageDir* | *listDir* | *crefDir* | *echoDir* | *ifDir* | *errorDir* | *includeDir* | *macroDir* | *macroCall* | *macroRepeat* | *purgeDir* | *macroWhile* | *macroFor* | *macroForc* | *aliasDir*
gpRegister	AX | EAX | BX | EBX | CX | ECX | DX | EDX | BP | EBP | SP | ESP | DI | EDI | SI | ESI
groupDir	*groupId* **GROUP** *segIdList*
groupId	*id*
hexdigit	a | b | c | d | e | f | A | B | C | D | E | F

Nonterminal	Definition
id	*alpha* | *id alpha* | *id decdigit*
idList	*id* | *idList* **,** *id*
ifDir	*ifStatement ;;* *directiveList* **[[** *elseifBlock* **]]** **[[ELSE** *;;* *directiveList* **]]** **ENDIF** *;;*
ifStatement	**IF** *constExpr* | **IFE** *constExpr* | **IFB** *textItem* | **IFNB** *textItem* | **IFDEF** *id* | **IFNDEF** *id* | **IFDIF** *textItem , textItem* | **IFDIFI** *textItem , textItem* | **IFIDN** *textItem , textItem* | **IFIDNI** *textItem , textItem* | **IF1** | **IF2**
immExpr	*expr*
includeDir	**INCLUDE** *fileSpec ;;*
includeLibDir	**INCLUDELIB** *fileSpec ;;*
initValue	*immExpr* | *string* | **?** | *constExpr* **DUP (** *scalarInstList* **)** | *floatNumber* | *bcdConst*
inSegDir	**[[** *labelDef* **]]** *inSegmentDir*
inSegDirList	*inSegDir* | *inSegDirList inSegDir*

Nonterminal	Definition
inSegmentDir	*instruction* | *dataDir* | *controlDir* | *startupDir* | *exitDir* | *offsetDir* | *labelDir* | *procDir* ⟦ *localDirList* ⟧ ⟦ *inSegDirList* ⟧ *endpDir* | *invokeDir* | *generalDir*
instrPrefix	**REP** | **REPE** | **REPZ** | **REPNE** | **REPNZ** | **LOCK**
instruction	⟦ *instrPrefix* ⟧ *asmInstruction*
invokeArg	*register* **::** *register* | *expr* | **ADDR** *expr*
invokeDir	**INVOKE** *expr* ⟦ **,** ⟦ *;;* ⟧ *invokeList* ⟧ *;;*
invokeList	*invokeArg* | *invokeList* **,** ⟦ *;;* ⟧ *invokeArg*
keyword	Any reserved word
keywordList	*keyword* | *keyword keywordList*
labelDef	*id* **:** | *id* **::** | **@@:**
labelDir	*id* **LABEL** *qualifiedType ;;*
langType	**C** | **PASCAL** | **FORTRAN** | **BASIC** | **SYSCALL** | **STDCALL**
listDir	*listOption ;;*
listOption	**.LIST** | **.NOLIST** | **.XLIST** | **.LISTALL** | **.LISTIF** | **.LFCOND** | **.NOLISTIF** | **.SFCOND** | **.TFCOND** | **.LISTMACROALL** | **.LALL** | **.NOLISTMACRO** | **.SALL** | **.LISTMACRO** | **.XALL**
localDef	**LOCAL** *idList ;;*
localDir	**LOCAL** *parmList ;;*
localDirList	*localDir* | *localDirList localDir*

Nonterminal	Definition
localList	*localDef* | *localList localDef*
macroArg	**%** *constExpr* | **%** *textMacroId* | **%** *macroFuncId* **(** *macroArgList* **)** | *string* | *arbitraryText* | < *arbitraryText* >
macroArgList	*macroArg* | *macroArgList* **,** *macroArg*
macroBody	**[[** *localList* **]]** *macroStmtList*
macroCall	*id macroArgList ;;* | *id* **(** *macroArgList* **)**
macroDir	*id* **MACRO [[** *macroParmList* **]]** *;;* *macroBody* **ENDM** *;;*
macroFor	*forDir forParm* **,** < *macroArgList* > *;;* *macroBody* **ENDM** *;;*
macroForc	*forcDir id* **,** *textLiteral ;;* *macroBody* **ENDM** *;;*
macroFuncId	*id*
macroId	*macroProcId* | *macroFuncId*
macroIdList	*macroId* | *macroIdList* **,** *macroId*
macroLabel	*id*
macroParm	*id* **[[:** *parmType* **]]**
macroParmList	*macroParm* | *macroParmList* **,** **[[** *;;* **]]** *macroParm*
macroProcId	*id*
macroRepeat	*repeatDir constExpr ;;* *macroBody* **ENDM** *;;*
macroStmt	*directive* | *exitmDir* | **:** *macroLabel* | **GOTO** *macroLabel*

Nonterminal	Definition
macroStmtList	*macroStmt ;;* | *macroStmtList macroStmt ;;*
macroWhile	**WHILE** *constExpr ;;* *macroBody* **ENDM** *;;*
mapType	**ALL** | **NONE** | **NOTPUBLIC**
memOption	**TINY** | **SMALL** | **MEDIUM** | **COMPACT** | **LARGE** | **HUGE** | **FLAT**
mnemonic	Instruction name
modelDir	**.MODEL** *memOption* ⟦ **,** *modelOptlist* ⟧ *;;*
modelOpt	*langType* | *stackOption*
modelOptlist	*modelOpt* | *modelOptlist* **,** *modelOpt*
module	⟦ *directiveList* ⟧ *endDir*
mulOp	***** | **/** | **MOD**
nameDir	**NAME** *id ;;*
nearfar	**NEAR** | **FAR**
nestedStruct	*structHdr* ⟦ *id* ⟧ *;;* *structBody* **ENDS** *;;*
offsetDir	*offsetDirType ;;*
offsetDirType	**EVEN** | **ORG** *immExpr* | **ALIGN** ⟦ *constExpr* ⟧
offsetType	**GROUP** | **SEGMENT** | **FLAT**
oldRecordFieldList	⟦ *constExpr* ⟧ | *oldRecordFieldList* **,** ⟦ *constExpr* ⟧
optionDir	**OPTION** *optionList ;;*

Nonterminal	Definition
optionItem	**CASEMAP :** *mapType* | **DOTNAME** | **NODOTNAME** | **EMULATOR** | **NOEMULATOR** | **EPILOGUE :** *macroId* | **EXPR16** | **EXPR32** | **LANGUAGE :** *langType* | **LJMP** | **NOLJMP** | **M510** | **NOM510** | **NOKEYWORD :** < *keywordList* > | **NOSIGNEXTEND** | **OFFSET :** *offsetType* | **OLDMACROS** | **NOOLDMACROS** | **OLDSTRUCTS** | **NOOLDSTRUCTS** | **PROC :** *oVisibility* | **PROLOGUE :** *macroId* | **READONLY** | **NOREADONLY** | **SCOPED** | **NOSCOPED** | **SEGMENT :** *segSize* | **SETIF2 :** *bool*
optionList	*optionItem* | *optionList* **,** ⟦ *;;* ⟧ *optionItem*
optText	**,** *textItem*
orOp	**OR** | **XOR**
oVisibility	**PUBLIC** | **PRIVATE** | **EXPORT**
pageDir	**PAGE** ⟦ *pageExpr* ⟧ *;;*
pageExpr	**+** | ⟦ *pageLength* ⟧ ⟦ **,** *pageWidth* ⟧
pageLength	*constExpr*
pageWidth	*constExpr*
parm	*parmId* ⟦ **:** *qualifiedType* ⟧ | *parmId* ⟦ *constExpr* ⟧ ⟦ **:** *qualifiedType* ⟧
parmId	*id*
parmList	*parm* | *parmList* **,** ⟦ *;;* ⟧ *parm*
parmType	**REQ** | **=** *textLiteral* | **VARARG**
pOptions	⟦ *distance* ⟧ ⟦ *langType* ⟧ ⟦ *oVisibility* ⟧
primary	*expr binaryOp expr* | *flagName* | *expr*

Nonterminal	Definition
procDir	*procId* **PROC** ⟦ *pOptions* ⟧ ⟦ < *macroArgList* > ⟧ ⟦ *usesRegs* ⟧ ⟦ *procParmList* ⟧
processor	**.8086** | **.186** | **.286** | **.286C** | **.286P** | **.386** | **.386C** | **.386P** | **.486** | **.486P**
processorDir	*processor ;;* | *coprocessor ;;*
procId	*id*
procParmList	⟦ **,** ⟦ *;;* ⟧ *parmList* ⟧ ⟦ **,** ⟦ *;;* ⟧ *parmId* **:VARARG**⟧
protoArg	⟦ *id* ⟧ **:** *qualifiedType*
protoArgList	⟦ **,** ⟦ *;;* ⟧ *protoList* ⟧ ⟦ **,** ⟦ *;;* ⟧ ⟦ *id* ⟧ **:VARARG** ⟧
protoList	*protoArg* | *protoList* **,** ⟦ *;;* ⟧ *protoArg*
protoSpec	⟦ *distance* ⟧ ⟦ *langType* ⟧ ⟦ *protoArgList* ⟧ | *typeId*
protoTypeDir	*id* **PROTO** *protoSpec*
pubDef	⟦ *langType* ⟧ *id*
publicDir	**PUBLIC** *pubList ;;*
pubList	*pubDef* | *pubList* **,** ⟦ *;;* ⟧ *pubDef*
purgeDir	**PURGE** *macroIdList*
qualifiedType	*type* | ⟦ *distance* ⟧ **PTR** ⟦ *qualifiedType* ⟧
qualifier	*qualifiedType* | **PROTO** *protoSpec*
quote	*“* | *‘*
radixDir	**.RADIX** *constExpr ;;*
radixOverride	h | o | q | t | y | H | O | Q | T | Y
recordConst	*recordTag* **{** *oldRecordFieldList* **}** | *recordTag* < *oldRecordFieldList* >
recordDir	*recordTag* **RECORD** *bitDefList ;;*
recordFieldList	⟦ *constExpr* ⟧ | *recordFieldList* **,** ⟦ *;;* ⟧ ⟦ *constExpr* ⟧

Nonterminal	Definition
recordInstance	**{** ⟦ *;;* ⟧ *recordFieldList* ⟦ *;;* ⟧ **}** | < *oldRecordFieldList* > | *constExpr* **DUP (** *recordInstance* **)**
recordInstList	*recordInstance* | *recordInstList* **,** ⟦ *;;* ⟧ *recordInstance*
recordTag	*id*
register	*specialRegister* | *gpRegister* | *byteRegister*
regList	*register* | *regList register*
relOp	**EQ** | **NE** | **LT** | **LE** | **GT** | **GE**
repeatBlock	**.REPEAT** ***;;*** *blockStatements ;;* *untilDir ;;*
repeatDir	**REPEAT** | **REPT**
scalarInstList	*initValue* | *scalarInstList* **,** ⟦ *;;* ⟧ *initValue*
segAlign	**BYTE** | **WORD** | **DWORD** | **PARA** | **PAGE**
segAttrib	**PUBLIC** | **STACK** | **COMMON** | **MEMORY** | **AT** *constExpr* | **PRIVATE**
segDir	**.CODE** ⟦ *segId* ⟧ | **.DATA** | **.DATA?** | **.CONST** | **.FARDATA** ⟦ *segId* ⟧ | **.FARDATA?** ⟦ *segId* ⟧ | **.STACK** ⟦ *constExpr* ⟧
segId	*id*
segIdList	*segId* | *segIdList* **,** *segId*
segmentDef	*segmentDir* ⟦ *inSegDirList* ⟧ *endsDir* | *simpleSegDir* ⟦ *inSegDirList* ⟧ ⟦ *endsDir* ⟧
segmentDir	*segId* **SEGMENT** ⟦ *segOptionList* ⟧ *;;*
segmentRegister	CS | DS | ES | FS | GS | SS

Nonterminal	Definition
segOption	*segAlign* | *segRO* | *segAttrib* | *segSize* | *className*
segOptionList	*segOption* | *segOptionList segOption*
segOrderDir	**.ALPHA** | .**SEQ** | **.DOSSEG** | **DOSSEG**
segRO	**READONLY**
segSize	**USE16** | **USE32** | **FLAT**
shiftOp	**SHR** | **SHL**
sign	**-** | **+**
simpleExpr	**(** *cExpr* **)** | *primary*
simpleSegDir	*segDir ;;*
sizeArg	*id* | *type* | *e10*
specialChars	**:** | **.** | **[** | **]** | **(** | **)** | < | > | **{** | **}** | **+** | **-** | **/** | ***** | **&** | **%** | **!** | **'** | \ | = | **;** | **,** | **"** | *whiteSpaceCharacter* | endOfLine
specialRegister	CR0 | CR2 | CR3 | DR0 | DR1 | DR2 | DR3 | DR6 | DR7 | TR3 | TR4 | TR5 | TR6 | TR7
stackOption	**NEARSTACK** | **FARSTACK**
startupDir	**.STARTUP** ;;
stext	*stringChar* | *stext stringChar*
string	*quote* [*stext*] *quote*
stringChar	*quote quote* | Any *character* except *quote*
structBody	*structItem ;;* | *structBody structItem ;;*
structDir	*structTag structHdr* [*fieldAlign*] [**, NONUNIQUE**] *;;* *structBody* *structTag* **ENDS** *;;*
structHdr	**STRUC** | **STRUCT** | **UNION**

Nonterminal	Definition
structInstance	< ⟦ *fieldInitList* ⟧ > | **{** ⟦ *;;* ⟧ ⟦ *fieldInitList* ⟧ ⟦ *;;* ⟧ **}** | *constExpr* **DUP (** *structInstList* **)**
structInstList	*structInstance* | *structInstList* **,** ⟦ *;;* ⟧ *structInstance*
structItem	*dataDir* | *generalDir* | *offsetDir* | *nestedStruct*
structTag	*id*
term	*simpleExpr* | **!** *simpleExpr*
text	*textLiteral* | *text character* | **!** *character text* | *character* | **!** *character*
textDir	*id textMacroDir ;;*
textItem	*textLiteral* | *textMacroId* | **%** *constExpr*
textLen	*constExpr*
textList	*textItem* | *textList* **,** ⟦ *;;* ⟧ *textItem*
textLiteral	*< text >;;*
textMacroDir	**CATSTR** ⟦ *textList* ⟧ | **TEXTEQU** ⟦ *textList* ⟧ | **SIZESTR** *textItem* | **SUBSTR** *textItem* **,** *textStart* ⟦ **,** *textLen* ⟧ | **INSTR** ⟦ *textStart* **,** ⟧ *textItem* **,** *textItem*
textMacroId	*id*
textStart	*constExpr*
titleDir	*titleType arbitraryText ;;*
titleType	**TITLE** | **SUBTITLE** | **SUBTTL**
type	*structTag* | *unionTag* | *recordTag* | *distance* | *dataType* | *typeId*
typedefDir	*typeId* **TYPEDEF** *qualifier*

Nonterminal	Definition
typeId	*id*
unionTag	*id*
untilDir	**.UNTIL** *cExpr ;;* **.UNTILCXZ** [[*cxzExpr*]] *;;*
usesRegs	**USES** *regList*
whileBlock	**.WHILE** *cExpr ;;* *blockStatements ;;* **.ENDW**
whiteSpaceCharacter	ASCII 8, 9, 11–13, 26, 32