Section 1
Getting Started with Pascal/MT+

The Pascal/MT+ system includes a compiler, a linker, a large library of run-time subroutines, and other programming tools to help you build better programs faster. The programming tools are

- DIS8080™, a disassembler
- LIBMT+™, a software library-building utility
- a dynamic debugger

The Pascal/MT+ system runs under any of the CP/M family of operating systems on an 8080, 8085, or Z80-based computer. The compiler and linker need at least 48K bytes of memory to run. To handle larger programs, they both need more memory.

The size of a program developed with Pascal/MT+ depends on the size of the source code, and on the number of run-time subroutines it uses. Typically, the minimum size of a simple program is about 8K bytes.

Figure 1-1 illustrates the software development process using the Pascal/MT+.system.

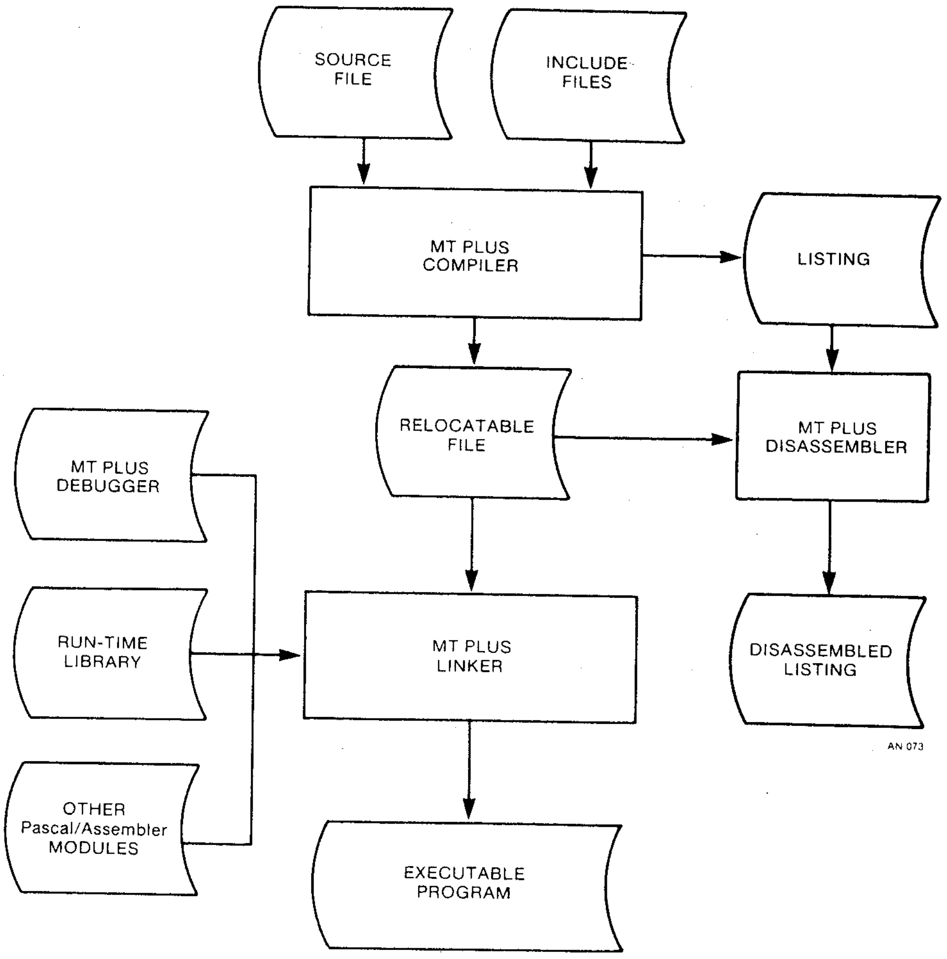

Figure 1-1. Software Development Under Pascal/MT+

1.1 Pascal/MT+ Distribution Disks

The Pascal/MT+ system is supplied on three separate disks. These disks contain a number of files of different types. Table 1-1 describes the filetypes used in the Pascal/MT+ system. Table 1-2 briefly describes the contents of each distribution disk.

Table 1-1. Pascal/MT+ System Filetypes

Filetype	Contents
BLD	Build file; input file used by LIBMT+
COM	Command file; directly executable under CP/M
CMD	Linker input command file
DOC	Document file; contains printable text in ASCII form
ERL	Relocatable object file; contains relocatable object code generated by the compiler
ERR	Error message file output by compiler
LIB	Library file; contains subroutines
MAC	Assembly language source file for RMAC
PAS	Pascal source file; contains source code in ASCII form (the compiler also accepts SRC as a source filetype)
PRN	Print file output by compiler
PSY	Intermediate symbol file used by linker
SRC	Pascal source file; contains source code in ASCII form (the compiler also accepts SRC as a source filetype)
SYP	Symbol file used by debugger
SYM	Symbol file used by SID
TXT	Text file; contains text of messages output by compiler
nnn	Hexadecimal n; used for numbering overlays

Table 1-2. **Pascal/MT+ Distribution Disks**

Disk 1

File	Content or Use
LINKMT.COM	Pascal/MT+ Linker
MTPLUS.COM	Pascal/MT+ Compiler
MTPLUS.000	Compiler Root Program
MTPLUS.001	Compiler Overlay
MTPLUS.002	Compiler Overlay
MTPLUS.003	Compiler Overlay
MTPLUS.004	Compiler Overlay
MTPLUS.005	Compiler Overlay
MTPLUS.006	Overlay used with Debugger
PASLIB.ERL	Pascal/MT+ Run-time System Module
ROVLMGR.ERL	Relocatable Overlay Manager
MOD1.SRC	Sample Program
MOD2.SRC	Sample Program
DEMOPROG.SRC	Sample Program

Disk 2

File	Content or Use
IOCHK.BLD	LIBMT+ input command file to produce IOERR.ERL
DIS8080.COM	Pascal/MT+ Disassembler
LIBMT+.COM	LIBMT+ Librarian Utility
XREF.COM	Pascal cross reference utility
AMD9511.CMD	LINK/MT+ input command file for linking AMDIO, FPRTNS, REALIO, and TRAN9511
AMD9511X.CMD	LINK/MT+ input command file for linking just AMDIO and FPRTNS

Table 1-2. (continued)

Disk 2 (continued)	
File	**Content or Use**
STRIP.CMD	LINK/MT+ input command file to produce STRIP.COM
XREF.DOC	Document file containing instructions for XREF, cross reference utility
INDEXER.DOC	Document file containing instructions for INDEXER, source file index utility
BCDREALS.ERL	BCD arithmetic module (does not include square root or transcendentals)
DEBUGGER.ERL	Debugging module that can be linked to a program
FPREALS.ERL	Software floating-point math module (contains REALIO.ERL)
FPRTNS.ERL	Hardware floating-point transcendental math module for AMD9511
FULLHEAP.ERL	Heap management and garbage collection module. PASLIB.ERL contains only USCD™-style stack/heap routines.
RANDOMIO.ERL	Random I/O file processing module
REALIO.ERL	Real arithmetic I/O module used only with AMD9511
TRAN9511.ERL	Transcendental math module for use with AMD9511
TRANCEND.ERL	Transcendental math module (for software floating-point only)
UTILMOD.ERL	Module containing KEYPRESSED, RENAME, and EXTRACT utilities
FIBDEF.LIB	File Information Block definition
APUSUB.MAC	AMD9511 routines for TRAN9511
CHN.MAC	Source for @CHN; chain routine can be altered to do bank switching in a non-CP/M environment

Table 1-2. (continued)

Disk 2 (continued)	
File	Content or Use
CWT.MAC	Source for @CWT routine
DIVMOD.MAC	Source for DIV and MOD routines that include a direct CP/M call for divide by 0 error message
OVLMGR.MAC	Overlay Manager source containing user-selectable options; unmodified version already in PASLIB.ERL
RST.MAC	Source for @RST routine
AMDIO.SRC	Module containing routines to interface with the AMD9511; must be customized for specific hardware
ATWNB.SRC	Source for @WNB routine
CALC.SRC	Sample program for testing floating-point math useful for testing AMD9511
CPMRD.SRC	Source for routine that uses @RST
GET.SRC	Source for low-level input routine
HLT.SRC	Source for a user-defined halt routine (current routine calls CP/M)
INDEXER.PAS	Source program for Pascal indexing program
IOERR.SRC	Source for a user-defined I/O error handling routine
PINI.SRC	Source for @INI initialization routine
PUT.SRC	Source for low-level output routine
RNB.SRC	Source for @RNB read next byte routine
RNC.SRC	Source for @RNC read next character routine
STRIP.SRC	Source file for utility program used with LINK/MT to eliminate unused entry points in an overlay
TRAN9511.SRC	Source for AMD9511 routines

Table 1-2. (continued)

Disk 2 (continued)	
File	**Content or Use**
UTILMOD.SRC	Source for module containing KEYPRESSED, RENAME, and EXTRACT
WNC.SRC	Source for @WNC routine
XBDOS.SRC	Source for BDOS routine that calls IOERR
XREF.SRC	Source for XREF, cross reference utility
DBUGHELP.TXT	Help file for debugger module
MTERRS.TXT	Compiler Error Message Text File

1.2 Installing Pascal/MT+

The first thing you should do when you receive your Pascal/MT+ system is make copies of both the distribution disks.

Note: you have certain responsibilities when making copies of Digital Research products. Be sure you read your licensing agreement.

Although you can use the compiler, linker, and other utilities directly from the distribution disks, it is more convenient if you copy specific files from the distribution disks to working system disks. One way to set up your Pascal/MT+ system is to use one disk for compiling and another disk for linking. You can use other disks for the programming tools, assorted source code, and examples.

This suggested configuration is just one way of setting up your disks. The important thing is that all the compiler modules are on the same disk, and all the linker modules are on one disk. For simplicity, it is a good idea to put all the related relocatable files on the same disk as the linker.

Note that the file MTPLUS.006 is only necessary when using the debugger, and that the compiler can run without the error message file MTERRS.TXT. If your compiler disk is short of space, you can eliminate these two files.

The following steps describe how to make a compiler disk and a linker disk:

1) Install both CP/M and the PIP utility on each of two blank disks. Label one disk as the compiler, and the other as the linker.

2) Put a text editor on the compiler disk.

3) Copy the following files from the distribution disks to the compiler disk:

   - MTPLUS.COM
   - MTPLUS.000 through MTPLUS.006
   - MTERRS.TXT

4) Copy the following files to the linker disk:

   - LINKMT.COM
   - all the ERL files

1.3 Compiling and Linking a Simple Program

If you have never used Pascal/MT+ before, the following step-by-step example shows you how to compile, link, and run a simple program. This example assumes that you are using a CP/M system with two disk drives, and that you are familiar with CP/M.

1) Put the compiler disk in drive A and the linker disk in drive B.

2) Using the text editor, create a file called TEST1.PAS and enter the following program. Put the file on drive B using PIP.

```
PROGRAM SIMPLE_EXAMPLE;

VAR
  I : INTEGER;

BEGIN
  WRITELN ('THIS IS JUST A TEST');
  FOR I := 1 TO 10 DO
    WRITELN (I);
  WRITELN ('ALL DONE')
END.
```

3) Now, compile the program with the following command:

   ```
   A>MTPLUS B:TEST1
   ```

   If you examine your directory, you see a file named TEST1.ERL that contains the relocatable object code generated by the compiler. If the compiler detects any errors, correct your source program and try again.

4) Now, log on to drive B, and link the program using the following command:

   ```
   B>LINKMT TEST1,PASLIB/S
   ```

   Your directory now contains a file named TEST1.COM that is directly executable under CP/M.

5) To run the program, enter the command:

   ```
   B>TEST1
   ```

Although the test program shown in the preceding steps is very simple, it demonstrates the essential steps in the development process of any program, namely editing, compiling, and linking.

If you want to write other simple programs, follow the same steps, but use your new program's filename instead of TEST1.

End of Section 1

Section 2
Compiling and Linking

This section tells how to use the compiler with its various options. It also describes how to link programs using the Pascal/MT+ linker, as well as different linkers.

2.1 Compiler Organization

The Pascal/MT+ compiler processes source files in three steps called passes or phases.

- Phase 0 checks the syntax and generates the token file.
- Phase 1 generates the symbol table.
- Phase 2 generates the relocatable object file.

The compiler creates some temporary files on the disk containing the source file, and under normal conditions it deletes those files. Make sure there is enough space on the disk, or use the T option to specify a different disk for the temporary files. See Section 2.2.3.

The compiler is segmented into overlays as shown in the following figure.

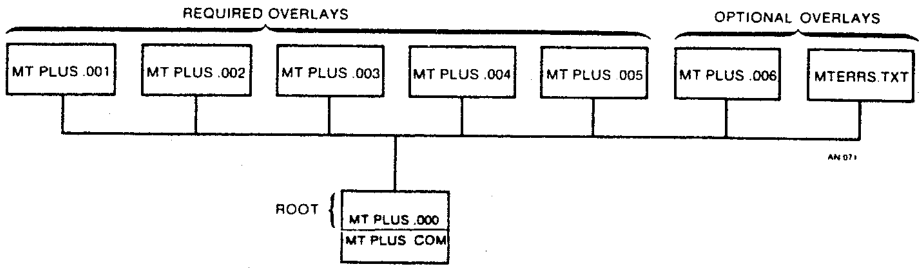

Figure 2-1. Pascal/MT+ Compiler Organization

2.2 Invoking the Compiler

You invoke the Pascal/MT+ compiler with a command line of the following form:

```
MTPLUS <filespec> {<options>}
```

where <filespec> is the source file to be compiled, and <options> is a list of optional parameters that you can use to control the compilation process.

The compiler can read the source file from any disk. The <filespec> must conform to the standard filespec format, and end with a carriage return/line-feed, and CTRL-Z. Refer to your operating system manual for a description of a Digital Research standard filespec.

If you do not specify a filetype, the compiler searches for the file with no filetype. If the compiler cannot find the file, it assumes a SRC filetype, assumes a PAS filetype. If the compiler still cannot find the file, it displays an error message.

The compiler generates a relocatable object file with the same filename as the input source program. The relocatable file has the ERL filetype.

2.2.1 Compilation Data

The Pascal/MT+ compiler periodically outputs information during Phases 0 and 1 to assure you it is running properly.

During Phase 0, the compiler outputs a + (plus sign) to the console for every 16 lines of source code it scans.

At the beginning of Phase 1, the compiler indicates the amount of available memory space. The space is shown as a decimal number of memory bytes available before generation of the the symbol table. Phase 1 also indicates available memory space following generation of the symbol table. This second indication is the amount of memory left for user symbols after the compiler symbols are loaded.

During Phase 1, the compiler also outputs a # (pound sign) to the console each time it reads a procedure or function. Symbol table overflow occurs if too little symbol table space remains for the current symbol. You can overcome this by using the $K option and breaking the program into modules. At completion, Phase 1 indicates the total number of bytes remaining in memory.

Phase 2 generates the relocatable object code. During this phase, the compiler displays the name of each procedure and function as it is read. The offset from the beginning of the module and the size of the procedure (in decimal) follow the name.

When the processing is complete, the compiler displays the following messages:

```
    Lines :     lines of source code compiled (in decimal)
    Errors:     number of errors detected
    Code  :     bytes of code generated (in decimal)
    Data  :     bytes of data reserved (in decimal)
```

2.2.2 Compiler Errors

When the compiler finds a syntax error, it displays the line containing the error. If you are using the MTERRS.TXT file, the compiler also displays an error description. If you are not using the MTERRS.TXT file, or you have a nonsyntax error, the compiler displays an error identification number.

When all processing is completed, the ERR file generated by the compiler summarizes all nonsyntactic errors.

Note: In Pascal/MT+, the compilation errors have the same sequence and meaning as in Jensen's and Wirth's Pascal User Manual and Report. Appendix A contains a complete list of the error messages, explanations, and causes.

When the compiler encounters an error, it asks if you want to continue or stop, unless you use the command line option C. (See Section 2.2.3.)

If the compiler cannot find an overlay or a procedure within an overlay, it displays messages of the following form:

```
Unable to open    <filename>  <overlay # >
Proc: "<procname>" not found ovl: <filename>  <overlay #>
```

The compiler displays the following procedure names if it cannot find an overlay name in the entry point table:

001	INITIALI
002	PHASE1
003	PH2INIT
004	BLK
005	PH2TERM
006	DBGWRITE

The number preceding the name is the group number of the overlay that contains the procedure.

Usually, you can find a missing overlay by ensuring that the name is correct, and that it is on the disk. If you cannot find it, recopy the overlay from your distribution disk. If you are sure the overlay is on the disk and you still get an error message, it means the file is corrupted.

2.2.3 Command Line Options

Compiler command line options control specific actions of the compiler such as where it writes the output files. All command line options are single letters that start with a $ or a #. Certain options require an additional parameter to specify where to send the output file or where an input file is located. If you specify more than one option, do not put any blanks between the options.

Table 2-1 describes the commmand line options. In this table, d stands for a parameter to specify a disk drive or output device. The parameters are as follows:

- X sends the output file to the console.
- P sends the output file to the printer.
- @ specifies the logged-in drive.
- Any letter from A to O specifies a specific drive.

Table 2-1. Default Values for Compiler Command Line Options

Option	Meaning	Default
A	Automatically calls the linker at the end of compilation. This option requires a linker input command file with the same name as the input file. The linker must be named LINKMT.COM.	Compiler automatically chains.
B	Uses BCD rather than binary for real numbers.	Binary reals.
C	Continues on error; default is to pause and let user interact and asks on n each error, one at a time.	Compiler stops and asks on each error.
D	Generates debugger information in the object code and writes the PSY file to the drive specified by the R option.	No debugger information and no PSY file generated.
Ed	The MTERRS.TXT file is on disk d: where d=@,A..O.	MTERRS.TXT on default disk.
Od	MTPLUS.000, and MTPLUS.001 through MTPLUS.006 are on drive d: where d=@,A..O.	Overlays on default disk.
Pd	Puts the PRN (listing file) on disk d: where d=X,P,@, A..O.	No PRN file.
Q	Quiet; suppresses any unnecessary console messages.	Compiler outputs all messages.
Rd	Puts the ERL file on disk d: where d=@,A..O.	ERL file on default disk.

Table 2-1. (continued)

Option	Meaning	Default
Td	Puts the token file PASTEMP.TOK on disk d: where d=@,A..O.	PASTEMP.TOK on default disk.
V	Prints the name of each procedure and function when found in source code as an aid to determining error locations during Phase 0.	Procedure names not printed.
X	Generates an extended ERL file, including disassembler records.	ERL file cannot be disassembled.
@	Makes the @ character equivalent to the ^ character.	@ not equivalent to ^.

The following is an example of a Pascal/MT+ command line:

```
A>MTPLUS A:TESTPROG $RBPXA
```

This command line tells the compiler to read the source from drive A, write the ERL file to drive B, display the PRN file on the console, and call the linker automatically.

2.2.4 Source Code Options

Source code compiler options are special instructions to the compiler that you put in the program source code. A source code option is a single lower- or upper-case letter preceded by a dollar sign, embedded in a comment. The option must be the first item in the comment. Certain source code options require additional parameters.

You can put any number of options in a source program, but only one option per comment is allowed. You cannot place blanks between the dollar sign and the option letter. The compiler accepts blanks between the option letter and the parameter.

Pascal/MT+ supports twelve source code compiler options, as summarized in Table 2-2.

Table 2-2. Compiler Source Code Options

Option	Function	Default
Cn	Use RST n instructions for REAL operation.	Use CALL instructions
E +/-	Controls entry point generation.	E+
I<filespec>	Includes another source file into the input stream, for example, {$I XXX.LIB}.	
Kn	Removes built-in routines to save space in symbol table (n=0..15).	
L +/-	Controls the listing of source code.	L+
P	Enter a form-feed in the PRN file.	
Qn	Use RST n instructions for loads and stores in recursive environments.	Use CALL instructions
R +/-	Controls range checking code.	R-
S +/-	Controls recursive/static variables.	S+
T +/-	Controls strict type checking.	T-
W +/-	Generates warning messages.	W-
X +/-	Controls exception checking code.	X-
Z $nnnnH	Initialize hardware stack to nnnnH.	Contents of location 0006 at beginning of execution

The following examples show proper source code compiler options:

```
{$E+}
(*$P*)
{$I D:USERFILE.LIB}
```

Space Reduction: Real Arithmetic (Cn)

The Cn option reduces the amount of object code generated when using REAL arithemtic. The Cn option tells the compiler to change all calls to @XOP (the REAL load and store routine) into a restart instruction. This reduces all 3-byte CALL instructions to 1-byte CALL instructions.

You specify a restart instruction number in the range 0 to 7 and the compiler generates RST n instructions. Be aware that in a CP/M environment, restart numbers 0 and 7 are not available. If you have another operating system, you should consult your hardware documentation.

You must specify the Cn option in the main program so the compiler can generate code to load the restart vector and RST n instructions for any call to @XOP. You must also specify the Cn option in any modules that use real numbers so the proper RST n instructions are generated.

Entry Point Record Generation (E)

The E option generates entry point records in the relocatable file. You enable the option using a + parameter, and disable it using a - parameter. E+ is the default.

E+ makes global variables and all procedures and functions available as entry points. For example, EXTERNAL declarations in separate modules can reference global variables and all procedures and functions if the E+ option is in effect. E- suppresses the generation of entry point records, thus making all variables, procedures, and functions local.

Include Files (I)

I<filespec> tells the compiler to include a specified file for compilation in the input stream of the original program. The compiler supports only one level of file inclusion, so you cannot nest include files.

The filespec must contain the drive specification, filename, and filetype in standard format. If you omit the filetype, the compiler looks for a file with the type of the main file. The file must end with a carriage return/line-feed, and CTRL-Z. If you omit the drive specification, the compiler looks on the default drive.

Symbol Table Space Reduction (Kn)

Predefined identifiers normally take about 6K bytes of symbol table space. The K option removes unreferenced built-in routine definitions from the symbol table to make more room for user symbols.

The K option uses an integer parameter ranging from 0 to 15. Each integer corresponds to different groups of routines as defined in Table 2-3. Enter all K options before the words PROGRAM or MODULE in the source code. Use as many K options as required, but place only one integer parameter after each letter K. Note that any reference in a program to the removed symbols generates an undefined identifier error message.

Table 2-3. $K Option Values

Group	Routines Removed
0	ROUND, TRUNC, EXP, LN, ARCTAN, SQRT, COS, SIN
1	COPY, INSERT, POS, DELETE, LENGTH, CONCAT
2	GNB, WNB, CLOSEDEL, OPENX, BLOCKREAD, BLOCKWRITE
3	CLOSE, OPEN, PURGE, CHAIN, CREATE
4	WRD, HI, LO, SWAP, ADDR, SIZEOF, INLINE, EXIT, PACK, UNPACK
5	IORESULT, PAGE, NEW, DISPOSE
6	SUCC, PRED, EOF, EOLN
7	TSTBIT, CLRBIT, SETBIT, SHR, SHL
8	RESET, REWRITE, GET, PUT, ASSIGN, MOVELEFT, MOVERIGHT, FILLCHAR
9	READ, READLN
10	WRITE, WRITELN
11	unused
12	MEMAVAIL, MAXAVAIL
13	SEEKREAD, SEEKWRITE
14	RIM85, SIM85, WAIT
15	READHEX, WRITEHEX

Listing Controls (L,P)

The L option controls the listing that the compiler generates during Phase 0. You enable the L option with the + parameter and disable it with the - parameter.

The P option starts a new page by placing a form-feed character in the PRN file.

Space Reduction: Recursion (Qn)

The Qn option operates in a manner analagous to the Cn option. That is, you specify a restart instruction number in the range 0 to 7, and the compiler generates RST n instructions for every call to @DYN.

You must specify the Qn option in the main program so the compiler can generate code to load the restart vector and RST n instructions for any call to @DYN. You must also specify the Cn option in any modules that use recursion so the proper RST n instructions are generated.

Run-time Range Checking (R)

The R option controls the generation of run-time code that performs range checking for array subscripts and storage into subrange variables. You enable the R option with the + parameter and disable it with the - parameter. Refer to Section 4.6.1 for information on range checking.

Recursion and Stack Frame Allocation (S)

The S option controls the stack frame allocation of procedure and function parameters and local variables. The + parameter causes recursion. The default parameter is -, and causes nonrecursion. Pascal/MT+ statically allocates global variables in programs and modules. You must enable the S option before the reserved words PROGRAM and MODULE. You cannot disable the S option within a separately compiled unit. You can link modules that use the S+ option with those that do not.

Strict Type and Portability Checking (T,W)

The T option controls the strict type checking/nonportable warning facility. The W option controls the display of warning messages pertaining to the T option. You enable both options with the + parameter and disable them with the - parameter. The default value for both options is -.

When the T option is enabled, the compiler performs only weak type checking. If the T and W options are enabled, and the compiler detects a nonportable feature, the compiler displays error message 500. String operations cause error 500 when the two options are enabled, because the STRING data type is not standard.

The T and W options check for compatibility with the ISO Pascal standard. They do not check for all features listed in the Pascal/MT+ Language Reference Manual, because certain features are implementation-dependent and others are software routines.

Run-time Exception Checking (X)

In the current release of Pascal/MT+, the X option remains in effect. Normally, the X option controls exception checking. Exception checking covers integer and real zero division, string overflow, real number overflow, and underflow. Refer to Section 4.6 for information on run-time error handling.

Setting the Stack Pointer (Z)

The Z option initializes the stack pointer to nnnnH in non-CP/M environments. In a CP/M environment, the compiler initializes the hardware stack by loading the stack pointer register with the contents of absolute location 0006H. Using the Z option suppresses this initialization.

You should enter the option as $Z+ only once before the PROGRAM line in the main program, and not on the individual modules.

2.3 Using the Linker

LINK/MT+ is the linkage editor that reads relocatable object modules with filetype ERL and generates an executable command file with filetype COM. The linker can also generate overlay files.

You invoke LINK/MT+ with a command line of the following format:

```
        LINKMT <main module>{,<module>}{,<library>}
```

or

```
        LINKMT <new filespec>=<main module>{,<module>}{,<library>}
```

The linker writes the executable file to the same logical disk as the <main module>, unless you specify a new <filespec> using an equal sign. The <main module> and each <module> can be on any logical drive. You can specify the drive before each file in the command line.

The linker assumes a ERL filetype for the <main module> and all <modules> unless you specify a CMD filetype. See the discussion about the /F option for information about CMD files. LINK/MT+ can link a maximum of 32 files at one time.

The following examples show valid LINK/MT+ command lines:

```
A>LINKMT CALC,TRANCEND,FPREALS,PASLIB/S

A>LINKMT B:CALC=CALC,B:TRANCEND,FPREALS,PASLIB/S

A>LINKMT D:NEWPROG=B:CALC,C:TRANCEND,C:FPREALS,C:PASLIB/S/M
```

2.3.1 Linker Options

Linker options are special instructions to LINK/MT+ that you specify in the command line. You specify options as a single lower- or upper-case letter. Each option must be preceded in the command line with a slash, /. Some options require an additional parameter. LINK/MT+ supports 13 options, as summarized in Table 2-4.

Table 2-4. Linker Options

Option	Function
C	Line continuation flag. Used only in CMD linker command files.
D:nnnnH	Relocate data area to nnnnH.
E	List entry points beginning with $, ?, or @ in addition to other entry points requiring /M or /W to operate.
F	Take preceding filename as a CMD linker command file containing input filenames, one per line.
Hnnnn	Write the output as a HEX file with nnnnH as the starting location of the hex format. This option is independent of the P option. Also, if you use this option, the compiler does not generate a COM file.
L	List modules as they are being linked.
M	List all entry points in tabular form.
P:nnnn	Relocate object code to nnnnH.

Table 2-4. (continued)

Option	Function
S	Search preceding name as a library, extracting only the required routines.
W	Write a SID-compatible SYM file (written to the same disk as the COM file).
O:n	Number the overlay as n and use the previous filename as the root program symbol table. By default, the range of n is 1 to 50, but you can extend it to 1 to 256 by altering the overlay manager.
Vn:mmmm	Overlay area starting address.
X:nnnn	Overlay static variable starting address when used with overlays, or amount of overlay data area when used with root modules.

Continue Line (/C)

The C option indicates a continued line in a linker input command (CMD) file. See the discussion of the F option below.

Data Location (/D)

The D:nnnn option tells the linker to start the data area at the hexadecimal address nnnn. If you do not use the D option, the code and data are mixed in the object file. By using the D option, you can solve some memory limitation problems.

However, you should be aware that local file operations depend on the linker to zero the data area. The linker does not zero the data area when you use the D switch, so these operations cannot be guaranteed.

Linker Input Command File (/F)

Normally in a CP/M environment, you must use the SUBMIT facility for typing repetitive sequences, such as linking multiple files together. LINK/MT+ allows you to enter this data into a file and have the linker process the filenames from the file. You must specify a file with a filetype of CMD and follow this filename with a /F, for example, CFILES/F.

The linker reads input from this file and processes the filenames. Filenames can be on one line, separated by commas, or each name or group of names can be on a separate line. At the end of each line except the last, you must place a /C option. The last line must end with a carriage return or line-feed.

The input from the file is concatenated logically after the data on the left of the filename. In the command line, additional options can follow the /F, but not additional object module names.

The following example demonstrates how to use a CMD file to link the files CALC, TRANCEND, FPREALS, and PASLIB into a CMD file. Use the following command to link the files:

```
A>LINKMT CALC/F/L
```

The file CALC.CMD contains

```
A:CALC,D:TRANCEND,FPREALS,B:PASLIB/S
```

The linker searches PASLIB for the necessary modules and generates a link map.

Hex Output (/H)

The H:nnnn option tells the linker to generate a HEX file instead of a COM file, starting the program at the hexadecimal address nnnn. The specified address is independent of the default relocation value of 100H. This means you can relocate the program to execute at 1D00H, for example, but have the HEX file addresses start at 8000H, by using the parameters:

```
/P:1D00/H:8000
```

Load Maps (/L),(/E)

The L option tells the linker to display module code and data locations as they are linked.

When used with the M or W options, the E option tells the linker to display all routines as they are linked, including routines that begin with ? or @, which are reserved for run-time library routine names. The E option does not enable the L, M, or W option. E does not display module code and data locations if used alone.

Memory Map (/M)

The M option generates a map and sends it to the map output file. Place the M option after the last file named in the parameter list.

Program Relocation (/P)

The P:nnnn option tells the linker to start the program at the hexadecimal address nnnn. If you do not use the P option, the default address is 100H.

The linker does not generate space-filling code at the beginning of the program. The first byte of the COM file is the byte of code that belongs in the specified starting location.

The syntax of the P option is

```
/P:nnnn
```

where nnnn is a hexadecimal number in the range 0 to FFFF.

Run-time Library Search (/S)

The S option tells the linker to search the file whose name the option follows as a library and to extract only the necessary modules. The S option must follow the name of the run-time library in the linker command line. The S option extracts modules from libraries only. It does not extract procedures and functions from separately compiled modules.

The order of modules within a library is important. Each searchable library must contain routines in the correct order and be followed by /S. PASLIB and FPREALS are specially constructed for searchability. Unless otherwise indicated, the other ERL files supplied with the Pascal/MT+ system are not searchable. You cannot search user-created modules unless they are processed by LIBMT+, as described in Section 5.3.

Generate SYM File (/W)

The W option tells the linker to generate a SID-compatible SYM file. The file contains information about entry points in the program. The linker uses the SYM file when it links overlays. The V option also enables the W option.

Overlay Options

The linker uses three options to process an overlay or a root program in an overlay scheme. The O option numbers the overlay and indicates that the previous filename is the root program symbol table. The Vm option sets the address of the overlay area. The X option controls how the linker allocates data space for overlays. Section 3.2 explains these overlay options.

2.3.2 Required Relocatable Files

You must always link the run-time system PASLIB.ERL with your compiled program. In addition, you need to link other ERL files with your program if it makes use of certain features of Pascal/MT+. The following are such files:

- RANDOMIO: SEEKREAD and SEEKWRITE are resolved here.
- DEBUGGER: @NLN, @EXT, @ENT generated when the debugger option is requested. If @XOP and @WRL are undefined, see Section 5.2.

The following files contain the real-number routines:

- BCDREALS: BCD real numbers, @XOP, @RRL, and @WRL.
- FPREALS: Binary real numbers @XOP, @RRL, and @WRL.
- TRANCEND: Support for SIN, COS, ARCTAN, SQRT, LN, EXP, SQR. Use only with FPREALS.

The following files contain real number routines used with the AMD9511:

- AMDIO: Routines for interfacing with the AMD9511. You must edit and recompile these to customize for specific hardware requirements.
- FPTRNS: AMD9511 support routines.
- REALIO: Read and Write real number routines necessary only when using the AMD9511.
- TRAN9511: Transcendental routines for AMD9511 (replaces TRANCEND).

2.3.3 Linker Error Messages

Table 2-5 shows the linker error messages.

Table 2-5. Linker Error Messages

Message	Meaning
Unable to open input file: xxxxxxxx	The linker cannot find the specified input file.
Incompatible relocatable file format	The ERL file is corrupted, or it has a format that is incompatible with the format expected by LINK/MT+.
Duplicate symbol: xxxxxxx	This usually means a run-time routine or variable has the same name as a user routine or variable.
SYSMEM not found in SYM file	This means the root program symbol file is corrupt.
External offset table overflow	This means you have exceeded the 200 externals plus offset addresses that the linker allows in its offset table.
Initialization of DSEG not allowed	The linker has encountered a DB or DW instruction in the Data segment.

2.4 Using Other Linkers

When you compile your program using the X option, Pascal/MT+ generates an extended relocatable file containing disassembler records. If you do not use the X option, the ERL file might be Microsoft® compatible. However, Digital Research does not guarantee that an ERL file generated by Pascal/MT+ is compatible with other linkers such as L80.

However, using LIBMT+ to process the ERL files generated by the compiler can result in a Microsoft-compatible relocatable files (see Section 5.3).

End of Section 2