Contents for Makefile Editor Help

Makefile editor is a tool for creating and editing Makefiles with the standard PowerFRAMEVIEW Builder.

To learn how to use Help, press F1.

How to...

Commands

Others

File Menu Commands

The function of each Menu Command is explained.

New
Open
Save
Save As
Exit Makefile Editor

The Set Source File Dialog Box

Sets a Source File to be compiled by a makefile.
Set a Source File Name to Macro Definition "SRCS" in a makefile. Also set a Source File Name to Macro Definition "OBJS" by converting it to "file-name.obj".

Each option in the Set Source File Dialog Box is explained below:

Select Source Files to Add

File names
Sets a Source File Name. If the file name has blanks at the beginning and in the middle, enclose the file name in double quotation marks.

Folders
Displays the current folder. Select a folder name from the list of folders or the list of Drives.

List Files of Type
Displays the Source File Name extensions. Select from the list of Files of Type.

Makefile Information

Folder
Displays the folder name of the file selected by a Source File Name.

Source File Names
Displays a list of Source Files to be set to a makefile. A filename with an * (asterisk) at the beginning is a Main Program.

The Main Program button
Displays the Set Main Program dialog Box.
Select a Main Program for the makefile from Source File Names indicated in the list of the Source File Names.

The Add button
Adds the file name that was set in the Select Source Files to Add to the list of Source File Names of the Makefile Information.

The Delete button
Deletes a file that was selected in the list of source filenames of Makefile Information.

See Also:
Setting Source File
The Set Main Program Dialog Box
Setting Main Program

The Set Dependency File Dialog Box

Sets files dependent on source files.
Set a Dependency File Name to Macro Definition "INCS" in a makefile.

Each option in the Set Dependency File Dialog Box is explained below:

Select Dependency Files to Add

File Names
Sets a Dependency File Name. If the file name has blanks at the beginning and in the middle, enclose the file name in double quotation marks.

Folders
Displays the current folder. Select a folder name from the list of Folders or the list of Drives.

List Files of Type
Displays the Dependency File Name extension. Select from the list of Files of Type.

Makefile information

Folder
Displays the folder name of the file selected by a Dependency File Name.

Dependency File Names
Displays a list of Dependency Files to be set to a makefile.

The Add button
Adds the file name that was set in the Select Dependency Files to Add to the list of Dependency File Names of the Makefile Information.

The Delete button
Deletes a file that was selected in the list of Dependency File Names of the Makefile Information.

See Also:
Setting Dependency File

The Set Link Library Dialog box

Sets a Link Library connected by link.
Set a Link Library Name to Macro Definition "LIBS" in a makefile.

Each option in the Set Link Library Dialog Box is explained below:

Select Link Library to Add

File names
Sets a Link Library Name. If the file name has blanks at the beginning and in the middle, enclose the filename in double quotation marks.

Folders
Displays the current folder. Select a folder name from the list of Folders or the list of Drives.

List Files of Type
Displays the Link Library Name extensions. Select from the list of Files of Type.

Makefile Information

Folder
Displays the folder name of the file selected by a Link Library Name.

Link Library Name
Displays a list of Link Libraries to be set to a makefile.

The Add button
Adds the file name that was set in the Select Link Library to Add to the list of Link Library Names of the Makefile Information.

The Delete button
Deletes a file that was selected in the list of Link Library Names of Makefile Information.

See Also:
Setting Link Library

The Set Main Program Dialog Box

Sets main program for COBOL and CAPE.
Set a main program name to Macro Definition "MAINSRC" in a makefile. Also set a main program name to Macro Definition "MAINOBJ" by converting it to "filename.obj."

Each option of the Set Main Program dialog box is explained below:

Main program name
Sets the desired main program name from the list of Source File Names. If a Main Program Name has been already set, it is displayed as a default.

Folder
Displays the folder of the file selected by the Source File Name.

Source File Names
Displays the list of Source Files set by the Set Source File dialog box.

The Clear button
Clear a Main Program.

See Also:
Setting Main program

Setting Source File

Sets a Source File to be compiled.
Set Main Program of COBOL and CAPE.
Set a Source File Name to Macro Definition "SRCS" in a makefile. Also set a Source File Name to Macro Definition "OBJS" by converting it to "filename.obj."
Set a Main Program Name to Macro Definition "MAINSRC" in a makefile. Also set a Source File Name to Macro Definition "MAINOBJ" by converting it to "filename.obj."

To add a Source File to the Makefile Information

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Set a file name to be added to the file name in the Select Source Files to Add, and chose the Add button.

To delete a Source File from the Makefile Information

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Select the file name to be deleted from the list of Source File Names in the Makefile Information, and chose the Delete button.

To set to the Main Program from a Source File

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Add a Source File to the Makefile Information.
4 Chose the Main Program button.
5 The Set Main program dialog box appears.
6 Set a Main Program Name from Source File Names.

To Clear the Main Program from a Source File

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Chose the Main Program button.
4 The Set Main Program dialog box appears.
5 Chose the Clear button.

See Also:

The Set Source File Dialog Box
The Set Main Program Dialog Box
Setting Main Program

Setting Dependency File

Set files depending on the source file.
Set a Dependency File Name to Macro Definition "INCS" in a makefile.

To add a Dependency File to the Makefile Information

1 Chose the Set Dependency File button from the main window.
2 The Set Dependency File dialog box appears.
3 Set a file name to be added to the file names in the Select Dependency Files to Add, and chose the Add button.

To delete a Dependency File from the Makefile Information

1 Chose the Set Dependency File button from the main window.
2 The Set Dependency File dialog box appears.
3 Select the filename to be deleted from the list of Dependency File Names in Makefile Information, and chose the Delete button.

See Also:

The Set Dependency File Dialog Box

Setting Link Library

Selects a Link Library to be linked using a link command.
Set a Link Library Name to Macro Definition "LIBS" in a makefile.

To add a Link Library to the Makefile Information

1 Chose the Set Link Library button from the main window.
2 The Set Link Library dialog box appears.
3 Sets a filename to be added to the file names in the Select Link Libraries to Add, and chose the Add button.

To delete a Link Library from the Makefile Information

1 Chose the Set Link Library button from the main window.
2 The Set Link Library dialog box appears.
3 Select a file name to be deleted from the list of Link Library Names in the Makefile Information, and chose the Delete button.

See Also:
The Set Link Library Dialog Box

Setting Main Program

Sets a Main Program File.
Set a Main Program Name to Macro Definition "MAINSRC" in a makefile. Also set a Main Program Name to Macro Definition "MAINOBJ" by converting it to "filename.obj."

To set to a Main Program

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Add a Source File to the Makefile Information.
4 Chose the Main Program button.
5 The Set Main Program dialog box appears.
6 Set a Main Program Name from Source File Names.

To clear the Main Program

1 Chose the Set Source File button from the main window.
2 The Set Source File dialog box appears.
3 Chose the Main Program button.
4 The Set Main Program dialog box appears.
5 Chose the Clear button.

See Also:
The Set Main Program Dialog Box

Create New Makefile

Creates a new makefile based on a Template Makefile.

To create a new makefile
1 Select the New command from the File menu. If the makefile has already been edited, it can be saved.
2 The message, "Do you want to specify the Template Makefile to be referred to ?" appears.

Choosing the Yes button displays the Select Template Makefile dialog box. Set a Template Makefile, and choose the OK button.
Choosing the No button displays the contents of the Default Template Makefile as a default.

See Also:
New Command on the File Menu

Open Makefile

Opens an existing makefile or a Template Makefile.

To open a makefile

1 Select the Open command from the File menu. If the makefile has already been edited, the makefile can be saved.
2 The Open dialog box appears.
3 Set a makefile or a Template Makefile.

Note:

- Makefiles created by the COBOL Workbench Makefile Editor cannot be opened.

See Also:

Open Command on the File Menu

Save Makefile

Saves a makefile currently being edited.

To save a makefile being edited

1 Select the Save command from the File menu.

To save a makefile under a new name

1 Select the Save As command from the File menu.
2 The Save As dialog box appears.
3 Set the file name.

Note:

- If a new makefile is created, the Save command of the File menu is masked.

See Also:

Save Command on the File Menu
Save As Command on the File Menu

Edit Makefile

Edits a makefile being displayed in the Makefile Editor. Set the information to the main window options of the Makefile Editor to edit a makefile.

See Also:
Main Window Options

New Command on the File Menu

Creates a new makefile.
When the New command is executed, specify a Template Makefile to be referred. If omitted, use a Default Template Makefile.
If a makefile is being edited, it can be saved.

See Also:
File

Open Command on the File Menu

Edits an existing makefile.
Display the Open dialog box, and open the specified makefile.

If a makefile is being edited when an Open command is selected, the makefile can be saved.

If the specified makefile does not exist, execute processing to create a new makefile.

Note:

- A makefile created by the COBOL Workbench Makefile Editor cannot be opened.

See Also:
File

Save Command on the File Menu

Saves a makefile currently being edited.

Notes:

- If a new makefile is created, the command is masked.
- Set a makefile name to Macro Definition "MAKEFILE" in a makefile.

See Also:
File

Save As Command on the File Menu

Saves a makefile being edited under a new name.
Display the Save As dialog box, and save the makefile using the specified makefile name.

A newly created makefile can be saved under a filename.

If the same filename as the makefile name being edited is specified, replacement can be made.

A makefile can be saved in a file other than the file being edited. In this case, the file being edited is not changed.

Note:

- Set a makefile name to Macro Definition "MAKEFILE" in a makefile.

See Also:
File

Exit Makefile Editor Command on the File Menu

Quits the Makefile Editor.

If a makefile is being edited, the file can be saved.

See Also:
File

Main window

The main window contains the following options:

Comment
Sets the information to be written as comment to a makefile.
A comment is written in the first line of a makefile.
If a comment of a makefile before update or template makefile exceeds 80 characters (en-size), it results in an error.

Target
Sets an executable file to be created by a makefile or a dynamic link library as a Target.
Enter the Target by specifying a file name only or a file name with a path.
Set the Target to Macro Definition "PROGRAM" in a makefile.
In a makefile, the processing is executed to build this Target.

Compile Information Command
Sets a command to compile a Source File.
Enter the compile command by specifying a file name only or a file name with a path.
Set the Compile Information Command to Macro Definition "COMP" in a makefile.

Note:

- The COBOL 85 compiler (COBOL.EXE) and CAPE compiler (CAPEC.EXE) terminates automatically when a command name is specified using only by a file name. When a command name is specified by full path, the compiler does not terminate automatically.

Compile Information Option
Sets a command option to compile a Source File.
Set the Compile Information Option to Macro Definition "COMPFLAGS" in a makefile.

Compile information Option Button
Display the Dialog Box to sets Command Option that compiles Source File.

Link Information Command
Sets a command to link Object Files.
Enter the link command by specifying a file name only or a file name with a path.
Set the Link Information Command to Macro Definition "LD" in a makefile.

Link Information Option
Sets a command option to link Object Files.
Set the Link Information Option to Macro Definition "LDFLAGS" in a makefile.

Link Information Option Button
Display the Dialog Box to set Command Option that links Object File.

The Set Source File button
Displays the Set Source File dialog box to set a Source File to be compiled.

The Set Dependency File button
Displays the Set Dependency File dialog box to set a Dependency File.

The Set Link Library button
Displays the Set Link Library dialog box to set a Link Library.

See Also:
The Set Source File dialog box
The Set Dependency File dialog box
The Set Link Library dialog box

Makefile

A file in which the relationship between various files in an application is defined. (e.g., Source File, Object File, Link Library, and Executable File)

Dependency File

A file that is a basis for Target creation (e.g., Source File)
Setting a Dependency File will cause build to be executed up to the Target depending on how the Dependency File is updated.
Not only the Module Definition File (DEF file) used at linking is defined. The Module Definition File must have the same folder name, and the filename of the Target and extension must be in DEF.

Main Program

The main file among files that make up a program. A source file specified here will be a program entry. Specify only one file each in the COBOL and CAPE programs.

Template Makefile

A file which contains the initial state of a new makefile.
Changes made to this file during editing are kept as a default and are used at the next creation of a makefile.

The makefile editor provides Template Makefiles for four program structures.
This file is stored in the COBOL32 folder under the folder containing PowerFRAMEVIEW.

Template Makefile for COBOL (32 bits)
SIMPLEXE.TMF: Simple Structure EXE
SIMPLDLL.TMF: Simple Structure DLL
DYNALINK.TMF: Dynamic Link Structure
DYNAPRGM.TMF: Dynamic Program Structure

Note:

- The Template Makefile in COBOL Workbench cannot be used.

Default Template Makefile

A Template Makefile that is used as a default when the Makefile Editor creates a makefile.
This file is stored as "SIMPLEXE.TMF" in COBOL32 folder under the folder containing PowerFRAMEVIEW.
Changes made to this file during editing are kept as a default and are used at the next creation of a makefile.

Makefile Format

This section describes the format of a Makefile When Build Processing is carried out using the Builder, this format must be used in describing a Makefile.

The description consists of the following items:

- Macro Description
- Dependents Relationship Line Description
- Command Line Description
- Making Rule Line Description
- Comment Line Description

Note:

- The Builder may not run normally if the Makefile contains an entry in a format other than described here.

Macro Description

The Macro Description names information. The name is replaced with the information during Build Processing. The Macro Description must precede the Dependency line. There are two types of Macro Descriptions, User-defined Macros and internal Macros.

User-defined Macro Description

Internal Macro Description

User-defined Macro Description

The User-defined Macro is a Macro Description that can be defined in a Makefile.

Define a Macro name as follows:

```
Macro Name = Expansion Character String
```

Reference a Macro name as follows:

```
$(Macro Name) or $Macro Name(if the Macro Name consists of only one character)
```

A Macro name is enclosed in "()" and prefixed with "$". "(" and ")" can be omitted only when the Macro name consists of only one character.

Macro name Description rules

- A Macro name can consist of alphanumeric characters including an underscore "_" and must begin with a letter.
- Uppercase and lowercase letters are distinguished from one another.

Expansion character string description rule

- An expansion character string is an arbitrary character string. If no character string is specified, a character string with a 0-byte length is assumed.

Example

Macro Definition and reference method

```
COMP = cobol.exe
COMPFLAGS = -M
$(COMP) $(COMPFLAGS) a.cob
```

After Macro Expand

```
cobol.exe  -M  a.cob
```

Notes:

- Neither a space nor a Tab can be described at the beginning of a line.
- Spaces at the right and left of the delimiter "=" are ignored.
- If the same Macro name is in both a Macro Definition and environment variable, the Macro Definition is regarded as valid.

See Also:

Internal Macro Description

Internal Macro Description

An Internal Macro is a Macro whose meaning is previously known to the Builder. The Internal Macros are listed below. "D" and "F" are used together with another Internal Macro.

Macro Name	Meaning
@	Represents the full name of a Target being currently processed.
*	Represents the file name after the extension has been removed from the current Target name.
?	Represents a list of files that depend on, and are older than, the current Target.
<	Represents dependents files that are older than the current Target. (These files can be used only on a Making Rule line.)
(Additional symbols)	
D	Enables taking out only the folder portion from the current Target.
F	Enables taking out only the file portion from the current Target.

Reference an Internal Macro as follows:

$Macro Name or $(Macro Name)

The Macro name is prefixed with "$". To reference an Internal Macro added with "D" or "F", it is necessary to enclose the Macro name in "()" and prefix it with "$"S.

Example

How to reference an Internal Macro

1. a.obj : a.cob

    cobol.exe -M $*.cob

    After Macro Expand

    a.cob

2. e:\test\a.obj : a.cob

    cobol.exe -M $(*F).cob

    After Macro Expand

    a.cob

Notes:

- "D" and "F" are used together with an Internal Macro. They cannot be used alone. In addition, "?" cannot be used in an Internal Macro.
- The Internal Macro can be used only in a reference within a Makefile. No Internal Macro can be redefined in a Makefile.

See Also:

User-defined Macro Description

Dependents Relationship Line Description

The Dependents Relationship Line defines a source that has a Dependents Relationship with a Target.

The format of a Dependents Relationship Line is:

```
Target:Source
```

Dependents Relationship Line Description rules

- The line cannot begin with a space or Tab character.
- More than one Target or source can be specified by delimiting them by a space.
- The source name for a Dependents Relationship Line can be omitted.

Example

Dependents Relationship Definition and Command

```
a.obj : a.cob
    cobol.exe -M a.cob
```

a.obj is dependent on **a.cob**, and **a.obj** is created by a Command Line just after the Dependents Relationship Line.

Note:

File Name in a Makefile

See Also:

Source Name Inference

Command Line Description

File Name in a Makefile

If a file name described in a Makefile does not contain a pathname, the specified file is assumed to be in the same folder as the Makefile.

To specify other files in a Makefile, a pathname must be included with the file name.

Example

Describing a file name in a Makefile

```
a.obj : a.cob e:\copy\cpy.cob
    cobol.exe -I e:\copy a.cob
```

Dependents Relationships

A Dependents Relationship indicates which file is related with which file.

Command Line Description

A command necessary to generate a Target is entered on a Command Line.

The format of a Command Line is:

```
<Tab> Command Line
```

The Command Line must begin with a **Tab character** followed by a command.

Command Line Description rule

- A Command Line must be located just below a Dependents Relationship Line or Making Rule line.

Example

Command Line Description

```
a.obj : a.cob
    cobol.exe -M a.cob
```

See Also:

Making Rule Line Description

Dependents Relationship Line Description

Command Line Inference

If a Command Line is not described in a Makefile, the Making Rules in the Makefile are used to infer what command is to be executed. If a typical command is to be used in a Makefile, it can be omitted, and should be represented in the form of Making Rules.

When a Command Line is omitted, the Making Rules that can be used are inferred from the source extension and the Target extension on the Dependency line. A command registered with the inferred Making Rules is then executed. If there is no usable Making Rule in a Makefile, command execution does not occur.

Example

Making Rules and Dependents Relationship Lines where Command Lines are omitted

```
.cob.obj:
    cobol.exe -M $*.cob
a.obj : a.cob
```

```
a.obj : a.cob
    cobol.exe -M a.cob
```

See Also:

Dependents Relationship Line Description

Making Rule Line Description

Source Name Inference

If a source name is omitted, the Making Rules in the Makefile or the Making Rules the Builder initially has are used to infer from the Target name what the source name is.

The source name is decided on as follows:

First, a Making Rule having the Target extension is searched for. An extension described in a Making Rule must be registered in the Suffix List.

If the Target extension is found, a source name is decided on from the Target extension. The source name is generated by removing the Target extension from the Target name and adding the source extension in the place where the Target extension was.

If the corresponding Making Rule is not found, a source name is not inferred.

Example

Dependents Relationship Line not having a source name

```
COMP = cobol.exe
COMPFLAGS = -M
.cob.obj:
    $(COMP) $(COMPFLAGS) $*.cob
b.obj:
    After Macro Expand
b.obj : b.cob
```

Note:

When There are Multiple Source Names to be Inferred

See Also:

Dependents Relationship Line Description

Suffix List Description

When There are Multiple Source Names to be Inferred

If more than one source is to be decided on according to the Target extension, and the files are in the same folder, they are processed in the preference sequence of the Suffix List.

Example

If more than one source name is searched for

COMP = cobol.exe

COMPFLAGS = -M

.SUFFIXES:

.SUFFIXES:.obj .cob .cbl

.cob.obj:

$(COMP) $(COMPFLAGS) $*.cob

.cbl.obj:

$(COMP) $(COMPFLAGS) $*.cbl

a.obj :

In this case, **a.cob** is assumed to be a valid source name according to the Suffix List.

If you want to use **a.cbl** as the source name, specify **a.cbl** as the source name for the dependent Relationship Description Line, or change the preference sequence of the Suffix List.

Suffix List

The Suffix List holds extensions used for special Targets during Build Processing.

Suffix List Description

Build Processing is carried out only for extensions in the Suffix List. Extensions to be subjected to Build Processing should be included in the Suffix List. The Builder has a default Suffix List. If there is no Suffix List in the Makefile, the default Suffix List is used.

The format of a Suffix List description is:

```
.SUFFIXES : List   (List of Extensions)
```

Extensions should be described after "**.SUFFIXES:**".

The default Suffix List is as shown below.

```
.SUFFIXES : .exe .dll .obj .mcp .cob .cbl .scp
```

Suffix List Description rules

- The character string "**.SUFFIXES:**" must be written in uppercase letters.
- If more than one extension is to be registered, delimit the extensions with a space.
- When you are not going to use the default Suffix List (instead, you are going to register one in the Makefile by yourself), first describe a Suffix List with no "**list**" specified, then specify the actual Suffix List.
- If more than one extension is specified, the leftmost extension has the highest priority, the one to its right has the next highest priority, and so on. If a source name is omitted from the Dependents Relationship Line, this priority sequence becomes valid.

Example

Specifying a Suffix List for the C language

```
.SUFFIXES:
.SUFFIXES: .exe .dll .obj .mcp .cob .cbl .c .scp
```

See Also:

Source Name Inference

Making Rule

The Making Rule is used to make Dependents Relationship Lines in a Makefile perfect.

Making Rule Line Description

The Making Rule is used to make Dependents Relationship Lines in a Makefile perfect.

The format of a Making Rule Line Description is:

```
.Source Extension.Target Extension:
<Tab> Command Line
```

A Target extension is generated from a source extension. The Command Line is used at generation. An extension description and a Command Line form a Making Rule block.

Making Rule Line Description rules

- Neither a space nor Tab character can be placed between a source extension and a Target extension.
- The Making Rule line cannot begin with a space or Tab character.
- The inference of Dependents Relationship and Command Lines may or may not be carried out depending on whether the extension of the Making Rule matches the extension of the file name.

Example

Making Rule (extension and command)

```
.cob.obj:
    cobol.exe -M $*.cob
```

1. a.obj : a.cob

    After Macro Expand

    a.obj : a.cob

    cobol.exe -M a.cob

2. a.obj :

    After Macro Expand

    a.obj : a.cob

    cobol.exe -M a.cob

The **cobol.exe** command is used to generate the **.obj** extension from the **.cob** extension.

See Also:

Dependents Relationship Line Description

Source Name Inference

Command Line Inference

Comment Line Description

A comment can be described in a Makefile. This method is used to temporarily change a statement to a comment or explain the Makefile.

The format of the Comment Line is:

```
# Comment
```

"#" should be placed at the beginning of a Comment Line or at a location where a comment should begin. All entities within the line that appear after "#" is treated as a comment (except for Command Lines). The effect of "#" in a line does extend beyond the line. To specify more than one comment, "#" must be placed on each individual line.

Example

```
# User-defined Macro
COMP = cobol.exe
LD = link
```

Note:

- A comment included in a Command Line should begin in the first column. If "#" is placed in other than the first column of a Command Line, it is treated as part of the command.

Target

Entity subject to Build Processing. More than one Target can be entered in a Makefile. Usually, a Target is a file name.

Build Processing

Work to create and update an Target. An Target is created and updated from the contents of a makefile.

Simple Structure

A simple structure is a single executable program made of one or more object programs using static link. Entire calling programs and called programs (sub programs) are loaded to virtual memory when execution starts and the efficiency of the calling sub programs increases. When a Simple Structure executable file is created, all sub programs are required for linking.

Dynamic Link Structure

A Dynamic Link Structure is a single executable program made up of object programs from the Main Program dynamically linked to an Import Library having sub program information. Sub programs are loaded to virtual memory when they are called. Loading is done by the system using sub program information created in an executable file at dynamic linking. When an executable file with a dynamic link structure is created, the Import Library of all sub programs is required for linking.

Dynamic Program Structure

A Dynamic Program Structure is an executable program made up of only the object programs of a Main Program. The sub program information is not included in the executable program as with the Dynamic Link Structure. A calling program request to the COBOL runtime system and a loading function in the system are used to call sub programs.

Macro Definition used by the Makefile Editor

Macro name	Explanation
SUFFIXES	Extension used at Build Processing
MAKEFILE	Makefile name
EXETYPE	Values identifying the program structure of a makefile Makefile for COBOL (32 bits) 10: Simple Structure EXE 11: Simple Structure DLL 12: Dynamic Link Structure 13: Dynamic Program Structure
PROGRAM	Target
SUB_PROGRAM	Library file required when linking with the Dynamic Program Structure
MAKELIBS	Library file required when linking with the Dynamic Link Structure
COMP	Command name to compile Source Files
LD	Command name to link Object Files and Link Libraries
COMPFLAGS	Option to compile Source Files
LDFLAGS	Option to link Object Files and Link Libraries
MAINSRC	Main Program Files If this Macro is not described by Template Makefile, the Main Program button in the Set Source File Dialog Box is not displayed.
MAINOBJ	Object File created when Main Program File is compiled
SRCS	Source Files
OBJS	Object Files created when Source Files are compiled
INCS	Dependency Files
LIBS	Link Libraries

Creating a Template Makefile

- How to create a makefile for each language

For COBOL and CAPE
There are following four creation methods of executable programs of COBOL and CAPE:
Simple Structure EXE
Simple Structure DLL
Dynamic Link Structure
Dynamic Program Structure

Select a Template Makefile to be referred to by the program structure of an executable program created when a new makefile of COBOL and so on is created. The Makefile Editor provides Template Makefiles for eight program structures.

Template Makefile for COBOL (32 bits)
SIMPLEXE.TMF: Simple Structure EXE
SIMPLDLL.TMF: Simple Structure DLL
DYNALINK.TMF: Dynamic link Structure
DYNAPRGM.TMF: Dynamic Program Structure

For other languages (such as C language)
When a makefile is created using languages other than those mentioned above, create a new Template Makefile.

To create Template Makefile
1 Change the file name of Template Makefile " SIMPLEXE.TMF" provided by PowerFRAMEVIEW. Use extension ".TMF".
2 Add the extension of the Source File to the Suffixes List.
3 Describe the Making Rules.

See Also:
Dependents Relationship Line Description
Suffix List Description

View Menu Commands

The functions of each menu command are explained.

Toolbar Command on the View Menu

Specifies whether a Toolbar is displayed at the top of the window. When this field is checked, the Toolbar is displayed.
When PowerFRAMEVIEW is installed, this field is checked.

Statusbar Command on the View Menu

Specifies whether a Statusbar is displayed on the bottom of the window. When this field is checked, the Statusbar is displayed.
When PowerFRAMEVIEW is installed, this field is checked.

Option Menu Commands

The functions of each menu command are explained.

The Module Definition File

Module Definition File Command on the Option Menu

If you save Makefiles, specify whether you create a Module Definition File or not.

Module Definition File is created in these cases described below.

- In the case when a Template Makefile (SIMPLDLL.TMF) which has browsed Makefile is created or updated.

Module Definition File is created to the folder on which the final target is created. The filename's base name is the final target with its extension part removed, and the extension of the filename is DEF.

- In the case when a Makefile which has browsed Template Makefile (DYNALINK.TMF or DYNAPRGM.TMF) is created or updated.

Module Definition File is created in every source file to a folder same as the source file which was set with the Set Source File Dialog box. This Module Definition File has been made by adding extension .DEF. to a file name which is source file with its extension part removed. However, Module Definition File corresponding to Main Program file is not created.

When PowerFRAMEVIEW is installed, this field is checked.