

User's Manual

idea-L

Screen Editor

Introduction

MS-Windows™ Based Ver. V3.30

[MEMO]

- * **IBM-PC is a trademark of International Business Machines Corporation.**
- * **MS-DOS and MS-Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and / or other countries.**
- * **i386SX and i486SX are trademarks of Intel Corporation. in the USA.**

• **The information in this document is current as of April, 1997. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC. NEC assumes no responsibility for any errors that may appear in this document.
- NEC does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment, and anti-failure features.
- NEC semiconductor products are classified into the following three quality grades:
 - "Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor product before using it in a particular application.
 - "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
 - "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
 - "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC semiconductor products is "Standard" unless otherwise expressly specified in NEC's data sheets or data books, etc. If customers wish to use NEC semiconductor products in applications not intended by NEC, they must contact an NEC sales representative in advance to determine NEC's willingness to support a given application.

(Note)

- (1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.
- (2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)

Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH

Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.

Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.

Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH

Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.

Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.

Madrid Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH

Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.

Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.

Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.

United Square, Singapore
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.

Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.

Electron Devices Division
Guarulhos-SP Brasil
Tel: 55-11-6462-6810
Fax: 55-11-6462-6829

CONTENTS

1. INTRODUCTION	1
1.1 What is <i>idea-L</i>?	1
1.2 Operation Overview of <i>idea-L</i>	2
1.3 Before Setting Up <i>idea-L</i>	3
1.4 Setting Up <i>idea-L</i>	4
1.4.1 Backing up the setup disks.....	4
1.4.2 Installation.....	5
1.5 A Look at <i>idea-L</i> Operation	12
2. TUTORIAL	15
2.1 Idea Processor Functions	15
2.1.1 Now, let's try it!.....	15
2.1.2 An idea processor?!.....	17
2.1.3 Grouping.....	18
2.1.4 Forming a hierarchy.....	23
2.1.5 Forming another group and hierarchy.....	27
2.1.6 Abstracting.....	30
2.1.7 Editing practice.....	32
2.1.8 Saving to a file.....	33
2.2 Library Functions	35
2.2.1 Creating a function.....	36
2.2.2 Registering functions.....	46
2.2.3 Using functions.....	49
2.3 Creating a Source File	68
2.4 Using Old Programs	73
3. RESTRICTIONS	77

[MEMO]

1. INTRODUCTION

1.1 What is idea-L?

A major problem in program development for single-chip microcontrollers is the increase in the number of software development steps as the program size increases. One solution is to develop software assets (software libraries). However, developing software assets takes time and effort. In addition, enabling their reuse by other programmers demands more time and effort.

Thus, a library support tool loaded in the idea processor was developed. It is called ***idea-L*** (idea el). ***idea-L*** is equipped with functions capable of producing software assets without undue strain on everyday software development and easily applying these assets.

idea-L is an editor that runs on MS-Windows™ and requires the operating environment to be described later. ***idea-L*** is an ordinary editor with an added idea processor and a library tool function. The name ***idea-L*** is the combination of ***idea*** in idea processor and ***L*** for library tool, and embodies the expectation of "***ideal*** software asset management."

1.2 Operation Overview of *idea-L*

Figure 1-2-1 is a block diagram of the overall *idea-L* system.

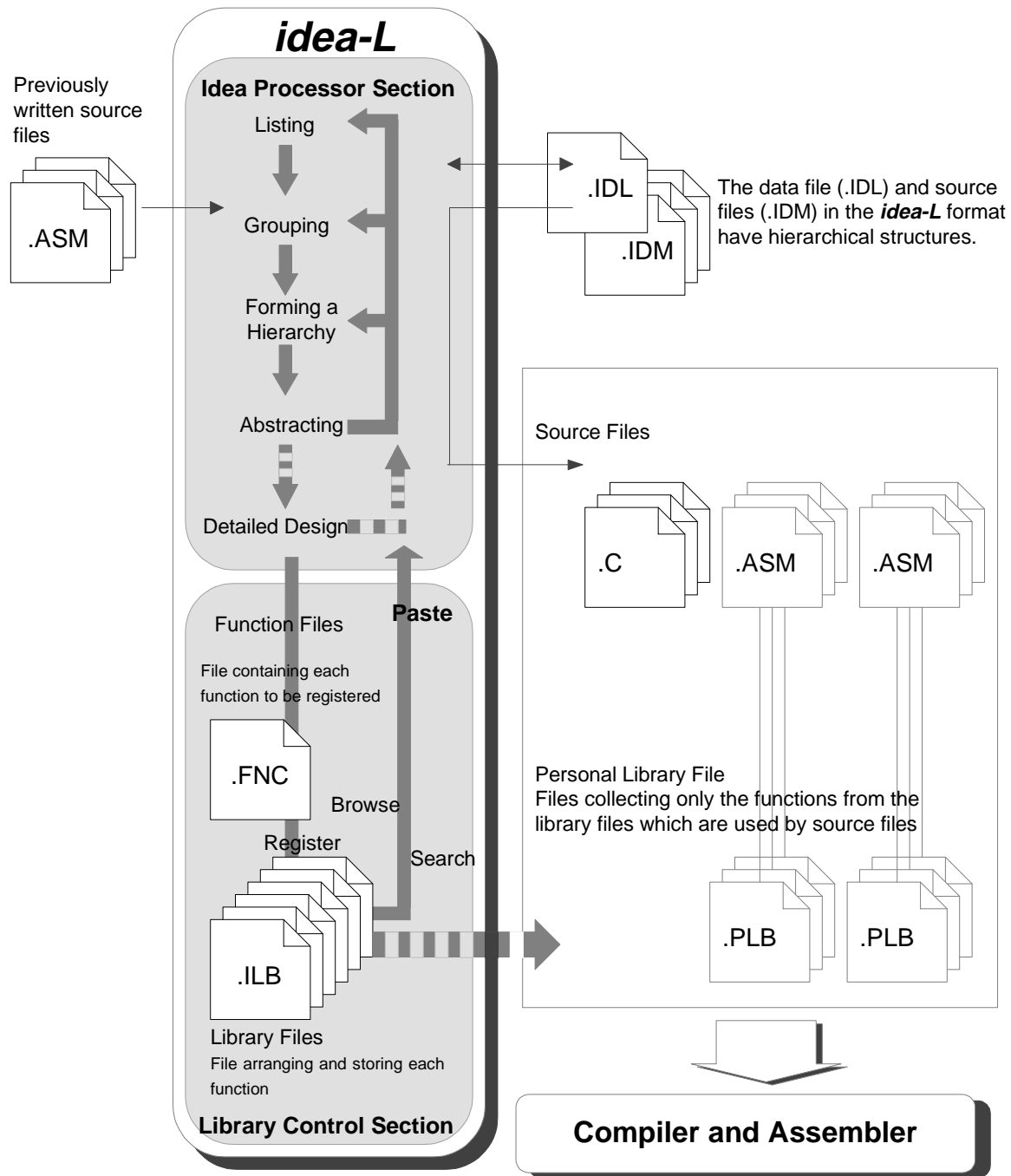


Figure 1-2-1. Block Diagram of the Overall *idea-L* System

idea-L is a software development tool for NEC single-chip microcontrollers. It is broadly divided into an idea processor section that has the functions from job management to program editing and the library control section that has the functions from function registration to application.

1.3 Before Setting Up *idea-L*

The following environment is required to set up *idea-L*.

- **Required environment**

- Hardware**

- Recommended Environment**

- NEC personal computer with an i486SX™ or newer CPU, IBM PC Series, or compatible machine

- Minimum 5.6 Mbytes of memory

- (At least 6.5 Mbytes for the high resolution mode)

- Minimum 10-Mbyte hard disk capacity

- Display with a minimum resolution of 1,024 × 768 dots

- Mouse

- Minimum Execution Environment**

- NEC personal computer with an i386SX™ or newer CPU, IBM PC Series, or compatible machine

- (Function extension processor required in PC-98XL)

- Minimum 3.6 Mbytes of memory

- (At least 4.5 Mbytes for the high resolution mode)

- Minimum 5-Mbyte hard disk capacity

- Display with a minimum resolution of 640 × 480 dots

- Mouse

- Software**

- Japanese MS-DOS™ Ver. 3.3D or later

- MS-Windows™ Ver. 3.1 or later

1.4 Setting Up *idea-L*

Before beginning the setup, check the hardware, software, and *idea-L* specified in section 1.3, "Before Setting Up *idea-L*."

1.4.1 Backing up the setup disks

Before starting the installation, to prepare for unlikely but possible problems, make backup disks of the original floppy disks in the package. After creating the backup disks, use these backup disks in the installation that follows. Store the original disks in a safe place.

1. Use three floppy disks for backup.
2. Start Windows 3.1 and start File Manager.
3. Run the "Copy Disk (C)..." command in the "Disk (D)" menu.
4. When the command is executed, the dialog box for copying floppy disks appears. It displays the "Copy Source Drive (S):" and "Copy Destination Drive (D):" items.
5. Specify the copy source and copy destination drives.
If the drive differs from the displayed drive name, click the arrow (↓) displayed to the right of the drive name and change the drive name.
6. Set system disk 1 of *idea-L* in the drive specified by "Copy Source Drive (S):". Click the "OK" button.
7. Similarly, back up system disks 2 and 3 of *idea-L*.
8. After making each copy, affix a label to the backup disk to prevent mix-ups.

1.4.2 Installation

The execution example reads *idea-L* from drive C: and installs the executable files in A:\nectools\bin, the help files in A:\nectools\hlp, the sample programs in A:\nectools\smp, and auxiliary data in A:\nectools\doc. Windows is immediately started.

1. Start the installation.
 - (1) Insert backup disk 1 in the floppy disk drive.
 - (2) Select "Run" in the "File" menu after selecting the Program Manager icon.
 - (3) Enter the following in the command line input.

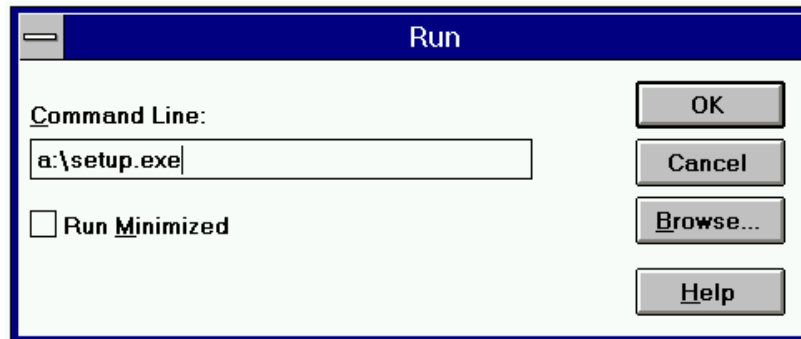


Figure 1-4-1. Specifying and Running a File Name

- (4) Click the OK button.
After initializing the setup, the installer starts.



Figure 1-4-2. Initializing the Setup

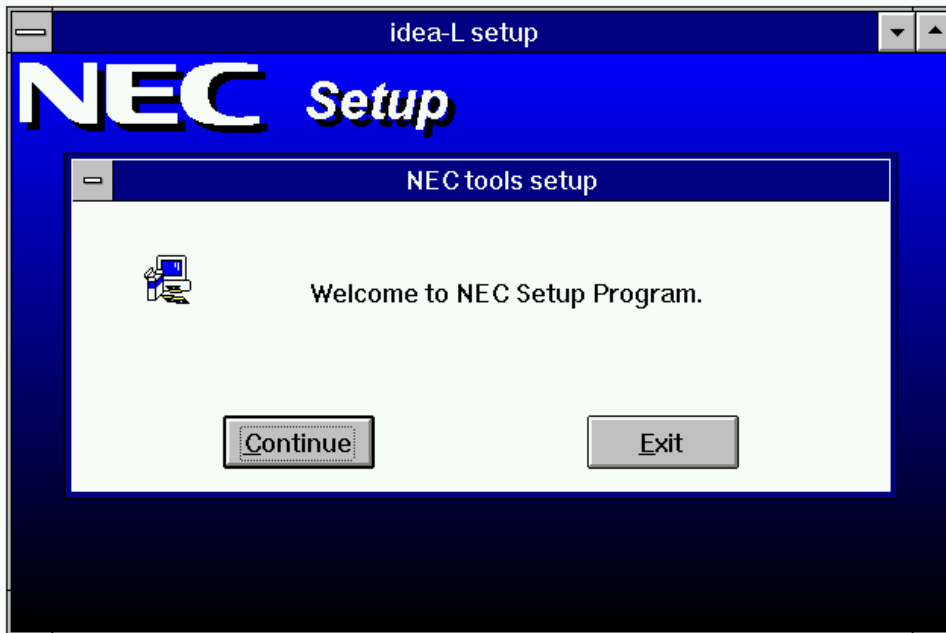


Figure 1-4-3. Welcome to NEC Setup

- (5) To continue the installation, select "Continue."
To exit the installation, select "Exit."
- 2. Verify the installation.
 - (1) If the "Continue" button was selected, the following message appears.

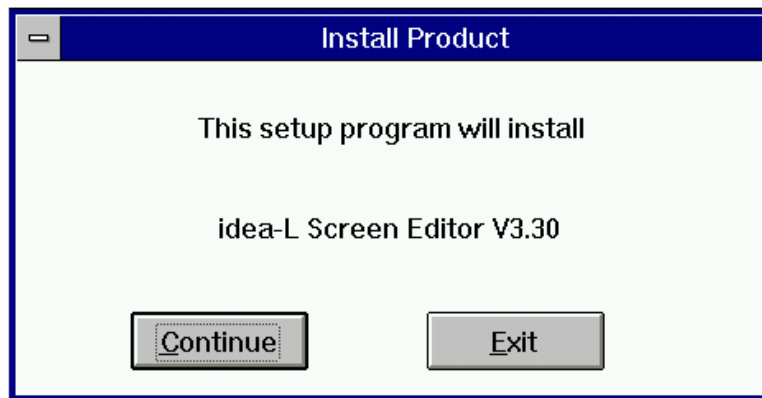


Figure 1-4-4. Verifying the Installation

- (2) After checking the installation, select "Continue."
- (3) To exit the installation, select "Exit."

3. Specify the install directories.

- (1) The dialog box for specifying the installation destinations appears.

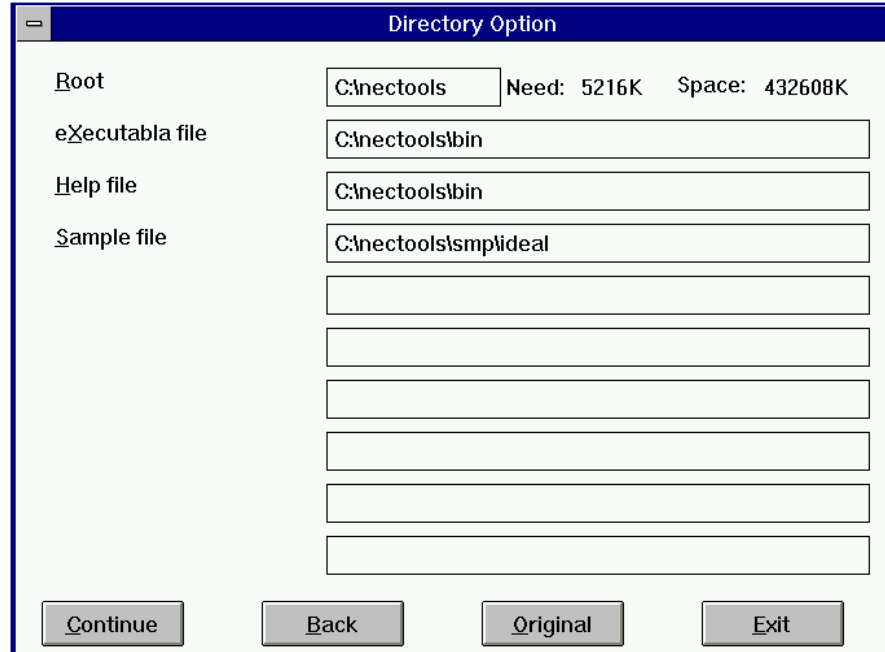


Figure 1-4-5. Specifying Installation Destinations

- (2) After entering the install directories in the text boxes, select "Continue."
(3) If "Back" is selected, the selection dialog for the installation item returns.
(4) If "Original" is selected, the specified directories indicate the default directories. The root default for the installation destination becomes \nctools of the drive on which Windows is installed. If the tools are installed after they have already been installed, that root is used. If the root is edited, the directories below it are linked and changed.
(5) If "Exit" is selected, the installation exits.
(6) Read the supplemental explanation since it is registered to an icon after the installation ends.
(7) If a specified directory is not valid, the following message appears.



Figure 1-4-6. Message for an Invalid Path

If there is insufficient disk space, an error occurs, and the following message appears.



Figure 1-4-7. Message for Insufficient Disk Space

4. Specify the group registered in the Program Manager.
 - (1) The dialog for specifying the registered group name appears.

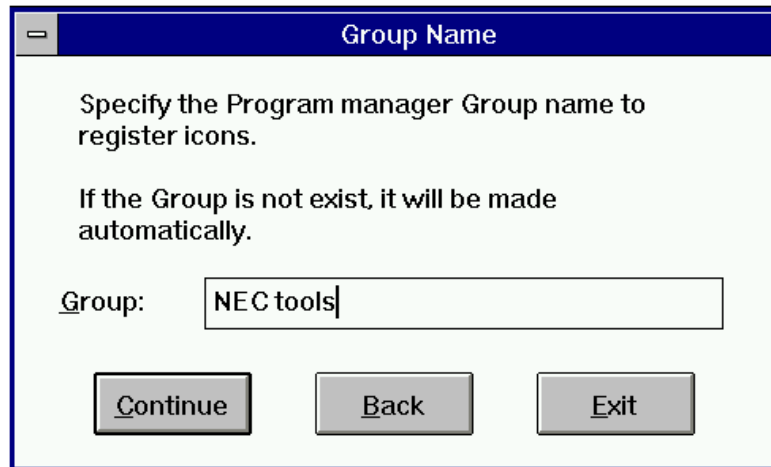


Figure 1-4-8. Specifying the Registered Group Name

- (2) After entering the group name to be registered in the text box, select "Continue." If the specified group does not exist, that group is created. If the specified group is already used and registered by the installer, that group is used.
- (3) If "Back" is selected, the dialog for specifying the installation destination returns.
- (4) If "Exit" is selected, the installation exits.

5. Start to copy the files.
 - (1) If a registered group is specified, the dialog to start copying the files appears.
 - (2) If "Continue" is selected, copying the files starts.

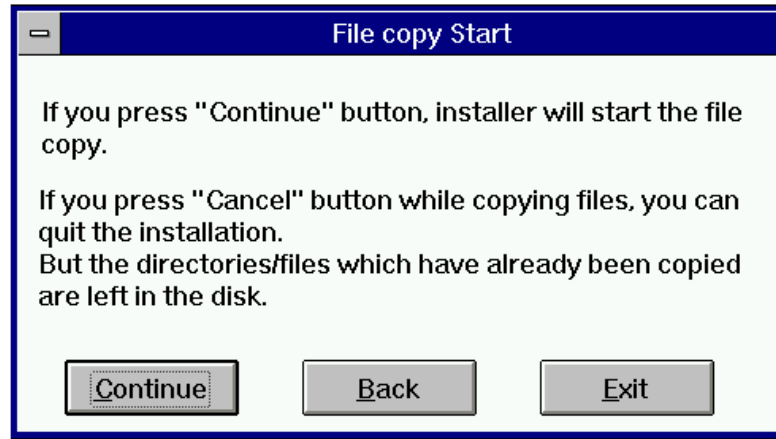


Figure 1-4-9. File Copy Start

If "Back" is selected, the dialog for specifying the registered group name returns.
If "Exit" is selected, the installation exits.

6. Copying the files starts.

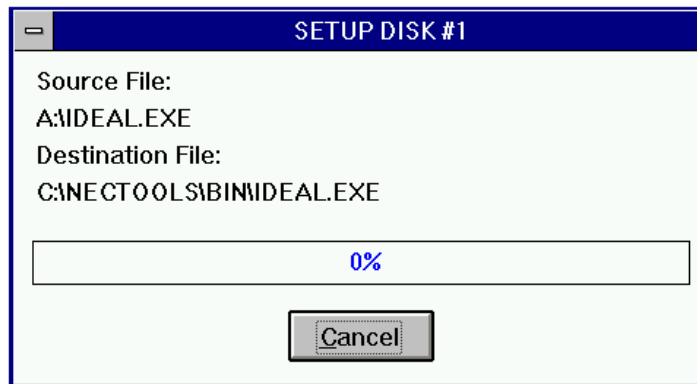


Figure 1-4-10. Dialog During File Copy

- (1) If "Cancel" is selected during a file copy, the following message appears.



Figure 1-4-11. Setup Cancel Message

If "Yes" is selected, the installation exits.

If "No" is selected, the file copy restarts.

7. Exchange the media.

- (1) When the following message appears, insert backup disk 2 in the floppy disk drive.



Figure 1-4-12. Exchange Disk Message

Similarly, when this message appears, insert backup disk 3 in the floppy disk drive.

8. The registered group and icon are created.

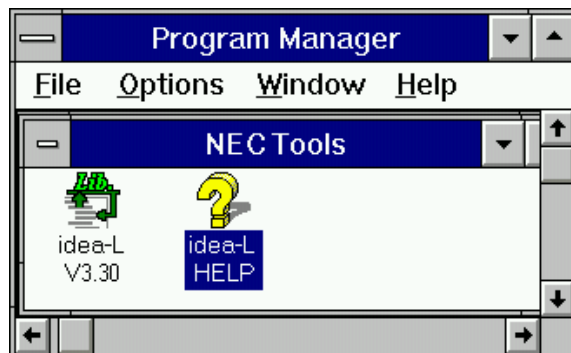


Figure 1-4-13. Creating the Registered Group

9. The installation ends.
 - (1) The following message appears.



Figure 1-4-14. Setup Completed Message

- (2) If "OK" is selected, the installation ends.
10. When the installer exits
 - (1) The following message is output.



Figure 1-4-15. Setup Stopped Message

- (2) If "Continue" is selected, the dialog where exit was selected returns.
 - (3) If "Exit" is selected, the following message appears.

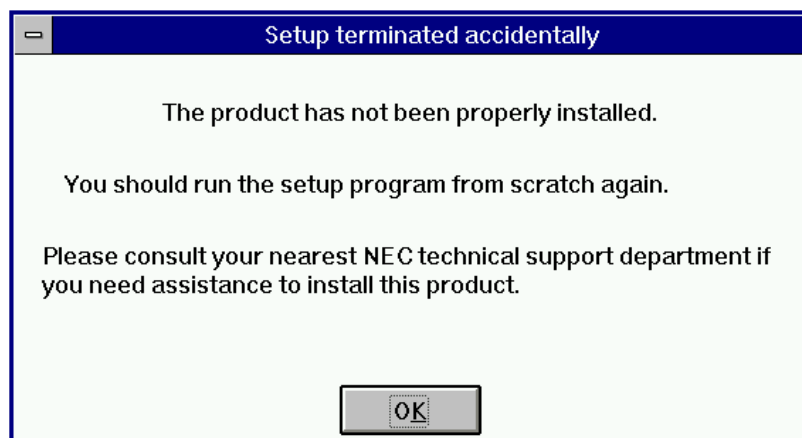


Figure 1-4-16. Setup Error Termination Message

- (4) If "OK" is selected, the installer exits.

1.5 A Look at *idea-L* Operation

First, let's start *idea-L*. Double click the "idea-L V3.1" icon in the Program Manager.

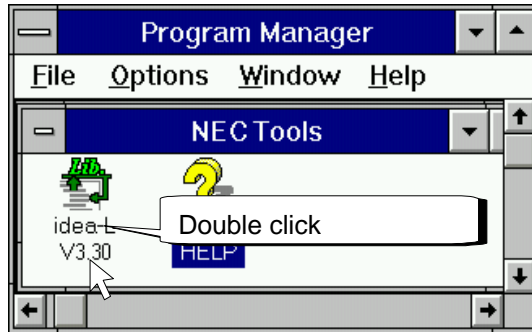


Figure 1-5-1. Start *idea-L*

As shown in Figure 1-5-2, the *idea-L* startup screen appears after the initialization screen.

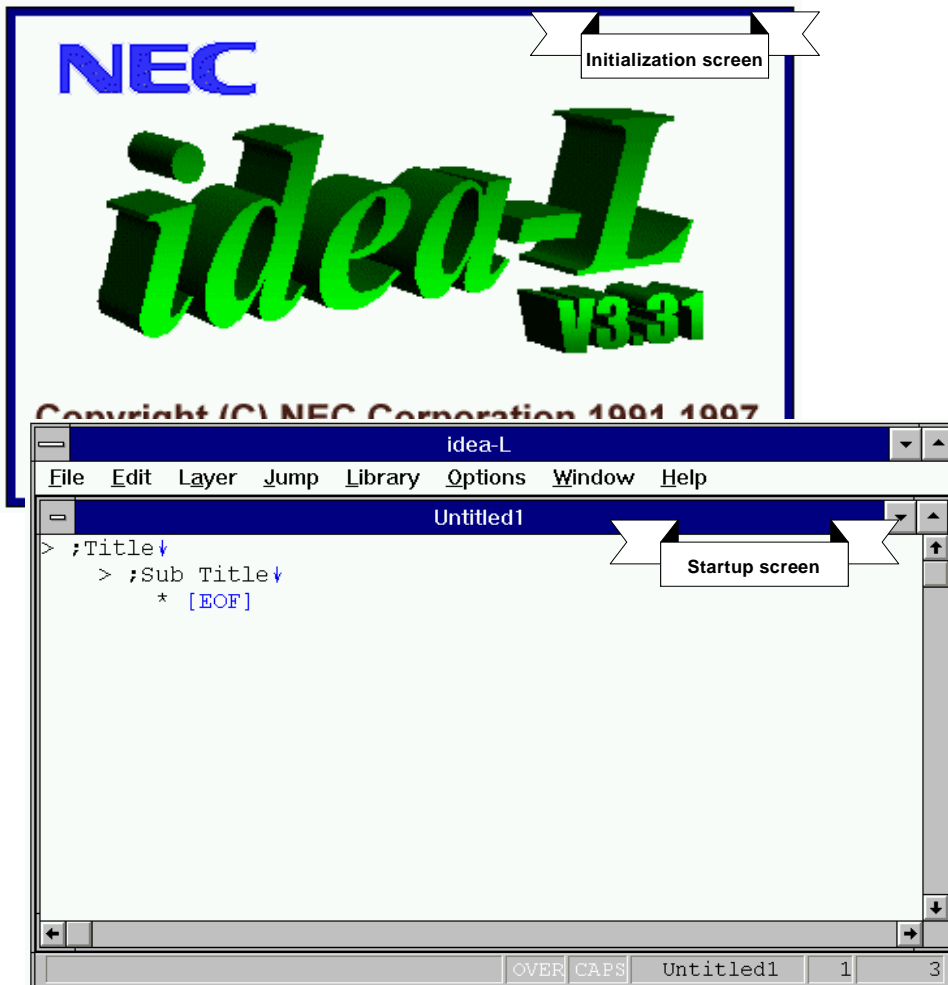


Figure 1-5-2. The *idea-L* Initialization Screen and the *idea-L* Startup Screen

The editing window for editing the source program opens.
Next, we will end *idea-L*. Click the File menu.

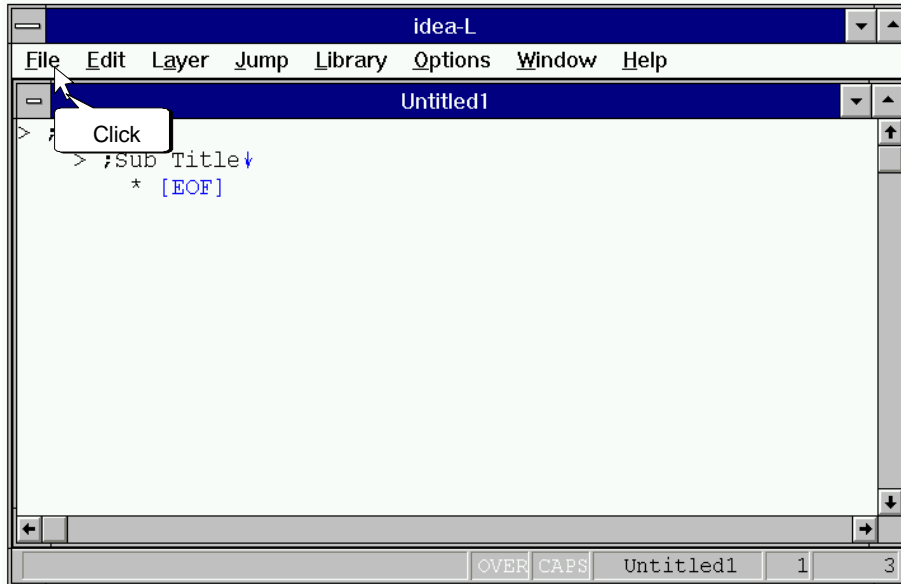


Figure 1-5-3. Click the File Menu

Since the pull-down menu for file appears, click the "Exit idea-L" command.

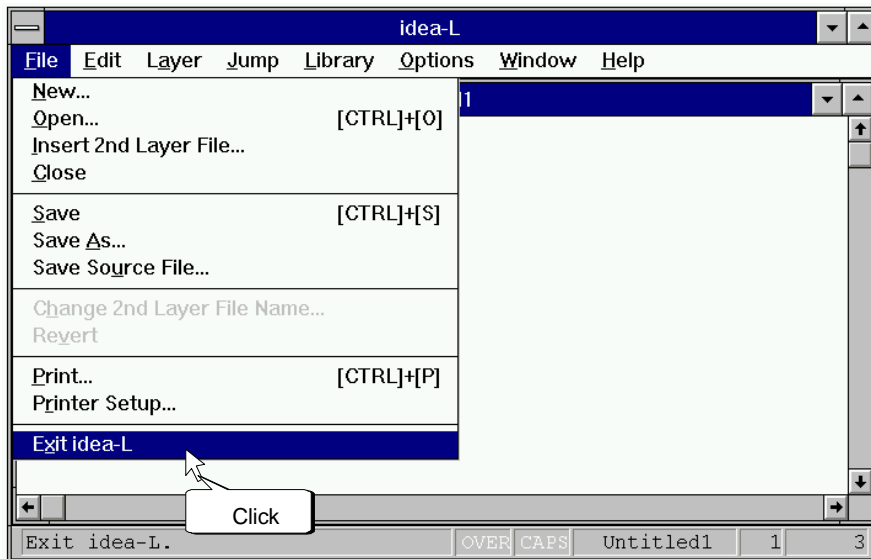


Figure 1-5-4. Exit *idea-L*

idea-L exits.

[MEMO]

2. TUTORIAL

This chapter describes the application of *idea-L* by using a sample program as an example.

Times font text in the balloons used in the figures in this chapter indicates a user action. Helvetica indicates an explanation.

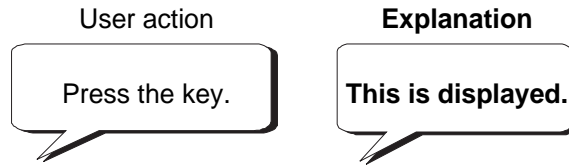


Figure 2-1. Text Characters in the Balloons

2.1 Idea Processor Functions

2.1.1 Now, let's try it!

Open the sample program. The sample program is nectools\smp\ideal\example on the drive where idea-L was installed.

To open the sample program, click on the "Open..." command in the "File" menu. The "Open" dialog box appears, so double click "example1.id1."

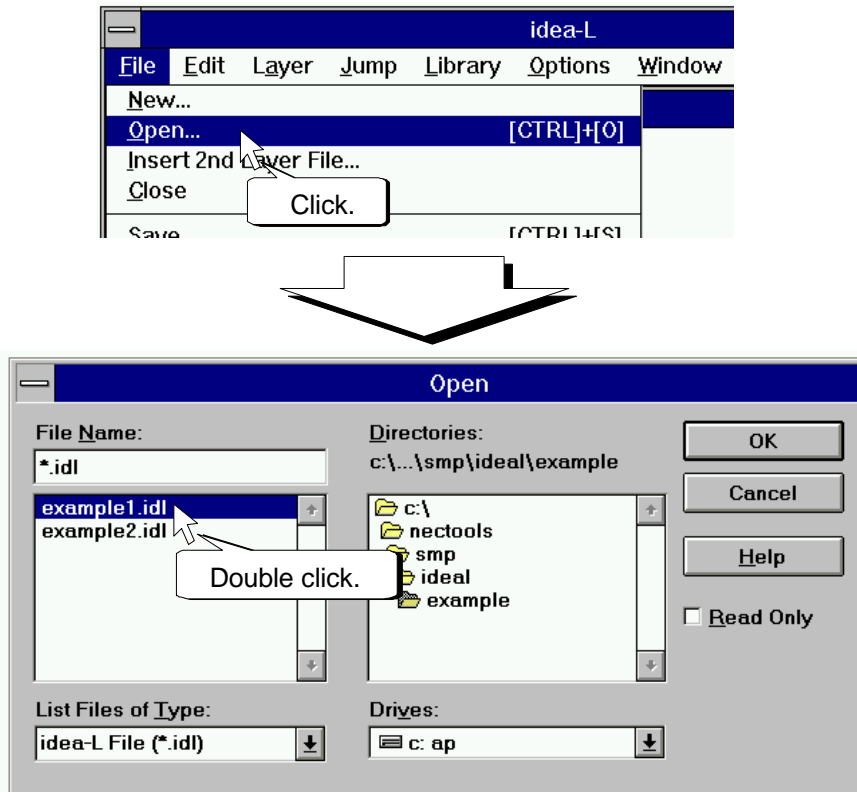


Figure 2-1-1. Open the File Menu

Open Example1. Example1 specifies the mechanism control section for a CD changer. Figure 2-1-2 shows the specification of the mechanism control section. Figure 2-1-3 shows the name of each part of the CD changer.

```

idea-L
File Edit Layer Jump Library Options Window Help
C:\NECTOOLS\SIMPIDEAL\EXAMPLE\EXAMPLE1.IDL
> ;CD Changer↓
  > ;Mechanism control section↓
    * ;When it insert a magazine, move pickup to position
    * ;When it insert a magazine, it take out the disk of
    * ;When it insert a magazine, set the pickup to the di
    * ↓
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ↓
    * ;When it proved that a disk doesn't enter a magazine
    * ;When it proved that a disk doesn't enter a magazine
    * ;When it proved that a disk doesn't enter a magazine
  
```

Figure 2-1-2. Specification of the Mechanism Control Section of a CD Changer

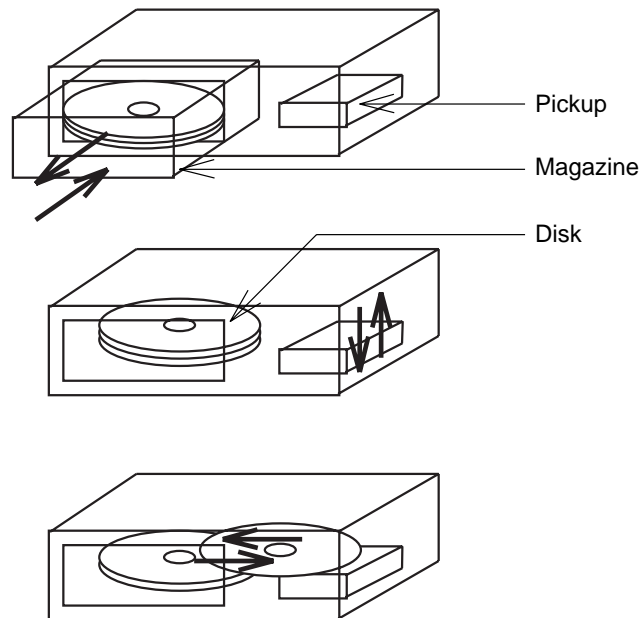


Figure 2-1-3. Internal View of the CD Changer

Normally, when writing the specification, the specification for each operating condition is assembled. However, when writing the program, each action is organized and programmed. The reason is the viewpoints of the specification writer and the programmer differ.

2.1.2 An idea processor?!

The idea processor is a tool for organizing and abstracting the ideas in a list. The cycle of the thought process apparent to people, such as listing, grouping, forming hierarchies, and abstracting ideas, is supported. Program development closely resembles the thought cycle of people in terms of organizing the specifications (idea), grouping, forming hierarchies, and abstracting.

Abstraction becomes one idea and derives broader abstractions. The contents of an abstracted program become a hierarchical (structured) module that is easy to reuse. If the interface of this module is standardized by the real-time operating system, it can be registered and managed as one independent function or task.

Based on this idea processor function, *idea-L* is not simply an editor used in the program creation process in software development, but becomes a tool that can be integrated and used from the upstream specification testing process to the program creation process.

In addition, there is a function in this idea processor to make large programs easy to read. *idea-L* omits the display of the lowest level (program) on the editing screen and displays the comments expressing the algorithm in a hierarchy. Furthermore, only the topmost level (abstracted module) can be displayed.

Generally, it is difficult to grasp the entire process flow on one screen for a long assembler program. However, *idea-L* can display only the topmost level in order to understand the process flow on one screen, and can display only the required portions in detail for editing. These functions indirectly contribute to better production efficiency in software development.

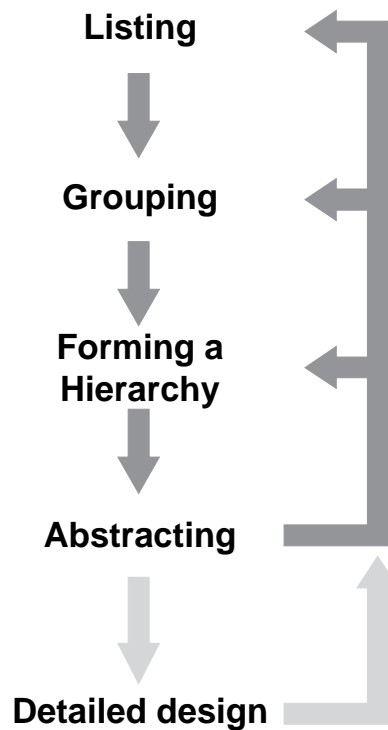


Figure 2-1-4. Idea Processor Functions




2.1.3 Grouping

Now, we will return to the explanation of example1.

Although there are various viewpoints about how to organize example1, we will first divide the actions of example1 into the pickup and the magazine.

Therefore, the lines related to the action of the pickup are collected at one location.

Line 1 is a line related to the pickup. The next line related to the pickup is line 3. Therefore, we will move line 3 between lines 1 and 2.

If the mouse cursor is moved to the first character "" in line 3, the shape of the mouse cursor becomes . Press the mouse (hold down the button) at the position where the mouse cursor became . The mouse cursor changes to  to point to the line. In this state, drag the mouse between line 1 and line 2 (move the mouse while pressing the button) and drop the mouse at that position (release the button). Line 3 is moved between lines 1 and 2.

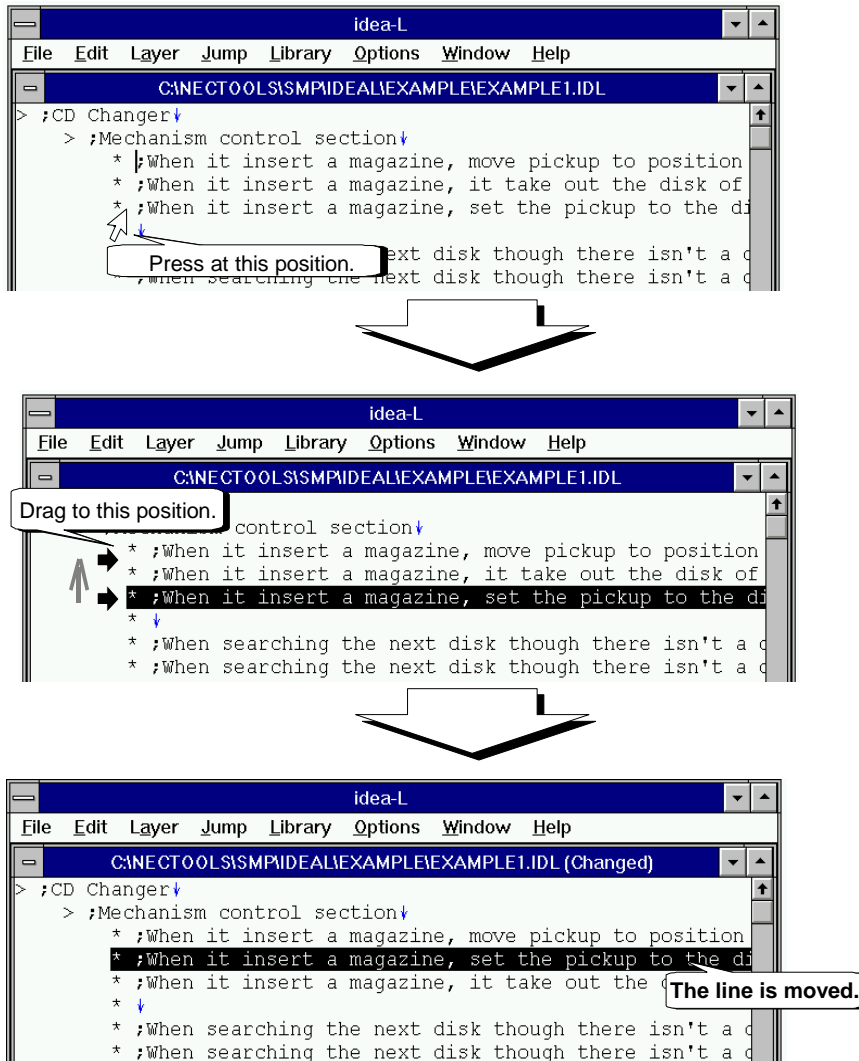


Figure 2-1-5. Moving a Line

Similarly, we will move the lines related to the pickup to one location. There is also a method to visually check and move the contents of each line. However, the method to be used here uses the string search of *idea-L* to move the lines.

First, as shown in Figure 2-1-6, use the mouse to specify the character range of "pickup" in line 2. Next, click the "Find String..." command in the "Edit" menu. This will display the "Find String" dialog box. Insert "pickup" which is the string in the specified range in the "Find What" text box. Click the "Find Next" button. The string of "pickup" is searched for in line 3 and later lines.

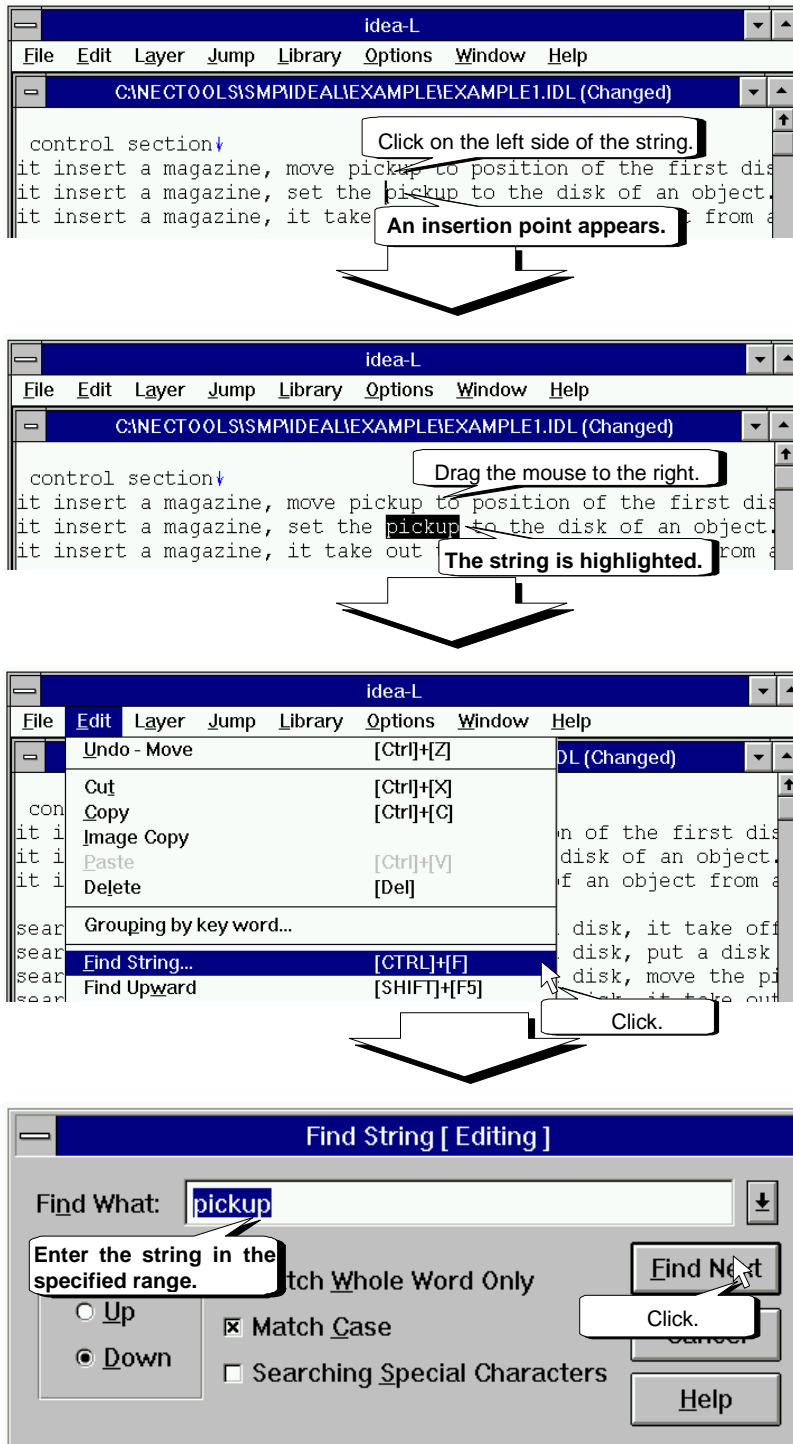


Figure 2-1-6. Searching for a String Based on a Range Specification

Since "pickup" is searched for and is highlighted, move to that line.

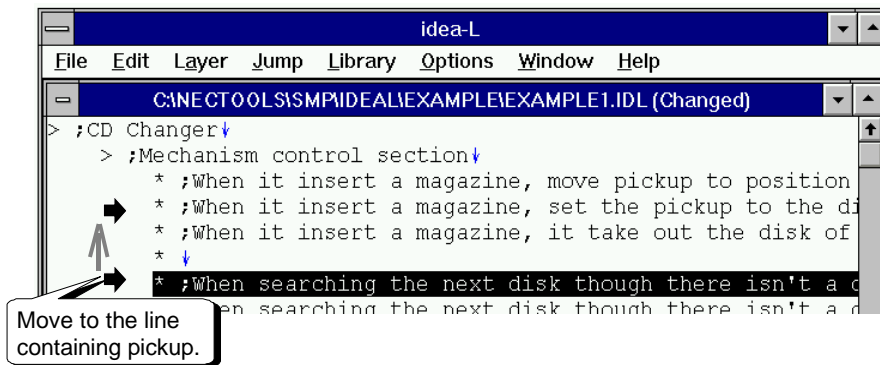
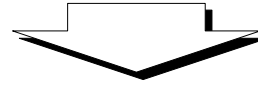
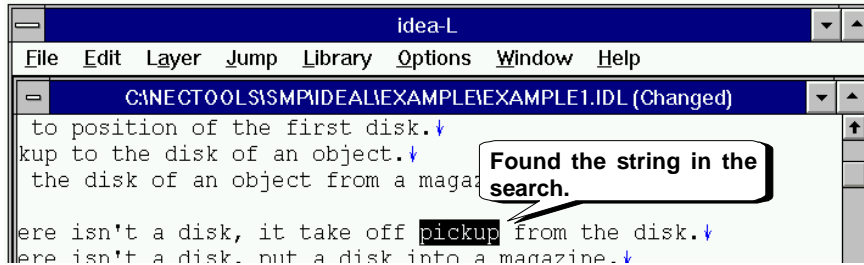
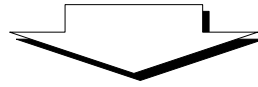


Figure 2-1-7. Successful String Search

A search string that was specified once is remembered until the range is specified again.

Click the mouse at the beginning of the line at the beginning of the search. The insertion point (cursor displayed at the position displaying the character being input) is moved to the beginning of the line.

Next, click the "Find Next" button in the "Find String [Editing]" dialog box.

This will search for the string "pickup" after the insertion point. If the string exists, that line is moved to.

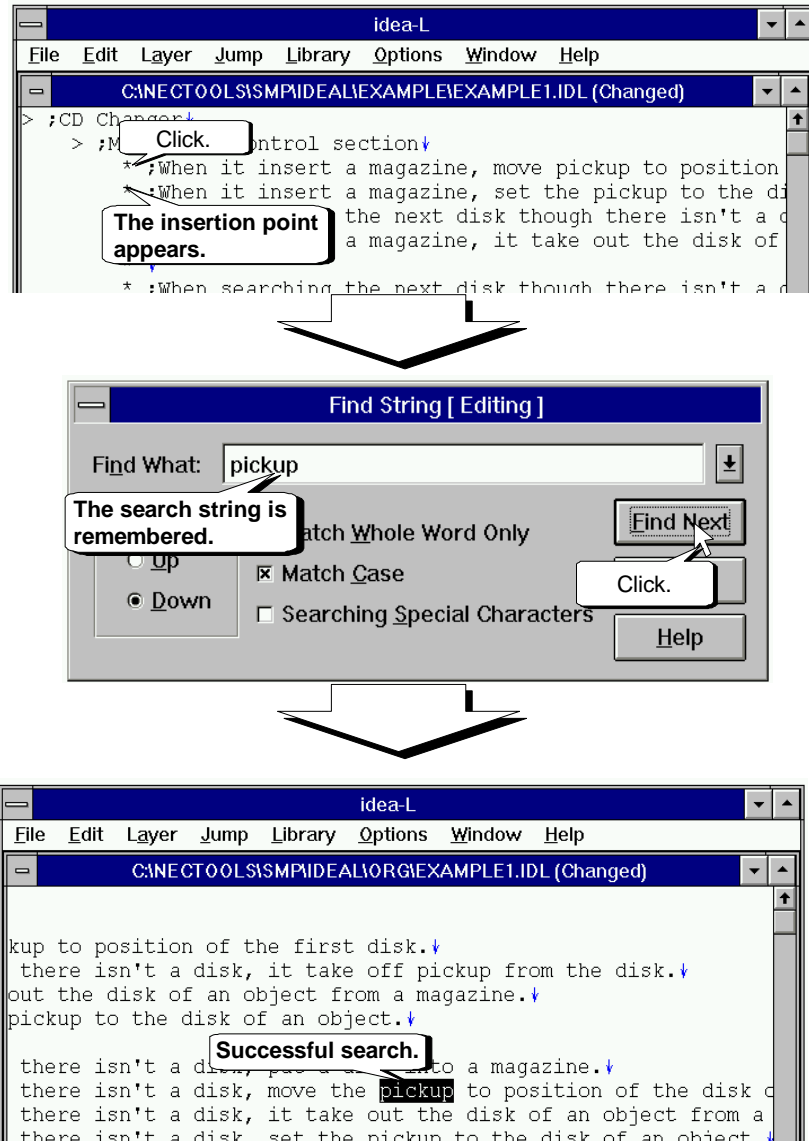


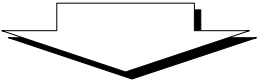
Figure 2-1-8. Repeat Search for the String

If the search is successful, that line is moved to. Similarly, move all of the lines related to the remaining pickups. If all of the lines have been moved to, click the "Cancel" button in the "Find String" dialog box. The "Find String" dialog box exits and disappears from the screen.

```

idea-L
File Edit Layer Jump Library Options Window Help
C:\NECTOOLS\SMPIDEAL\EXAMPLE\EXAMPLE1.IDL (Changed)
> ;CD Changer↓
  > ;Mechanism control section↓
    * ;When it insert a magazine, move pickup to position
    * ;When it insert a magazine, set the pickup to the di
    * ;When searching the next disk though there isn't a d
    * ;When it insert a magazine, it take out the disk of
    * ↓
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
  
```

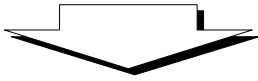
Move the pickup line.



```

idea-L
File Edit Layer Jump Library Options Window Help
C:\NECTOOLS\SMPIDEAL\EXAMPLE\EXAMPLE1.IDL (Changed)
> ;CD Changer↓
  > ;Mechanism control section↓
    * ;When it insert a magazine, move pickup to position
    * ;When it insert a magazine, set the pickup to the di
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ;When searching the next disk though there isn't a d
    * ;When it proved that a disk doesn't enter a magazine
    * ;When it proved that a disk doesn't enter a magazine
    * ;When it pushe the play key, it
    * ;When it pushe the play key, mov
    * ;When it pushe the play key, set
    * ;When it pushe the ejct key, it take off pickup from
    * ;When it pushe the ejct key, return the pickup to rd
  
```

The lines related to the pickup were collected.



Find String [Editing]

Find What: pickup

Direction: Up Down

Match Whole Word Only

Match Case

Searching Special Characters

Find Next

Cancel

Help

Click.

Figure 2-1-9. Moving All of the Lines Containing Pickup

2.1.4 Forming a hierarchy

When grouping is completed, next the hierarchy is formed. Forming a hierarchy adds a line for naming each group and indents the entire group contents.

First, click the mouse at the end of the first line in the group. The insertion point appears at the end of the first line. After pressing the "HOME CLR" key to move the insertion point to the beginning of the line, press the return key. A blank line is inserted before this line. The comment statement naming the group is entered in this blank line.

Enter "; pickup" and "; magazine" at this position.

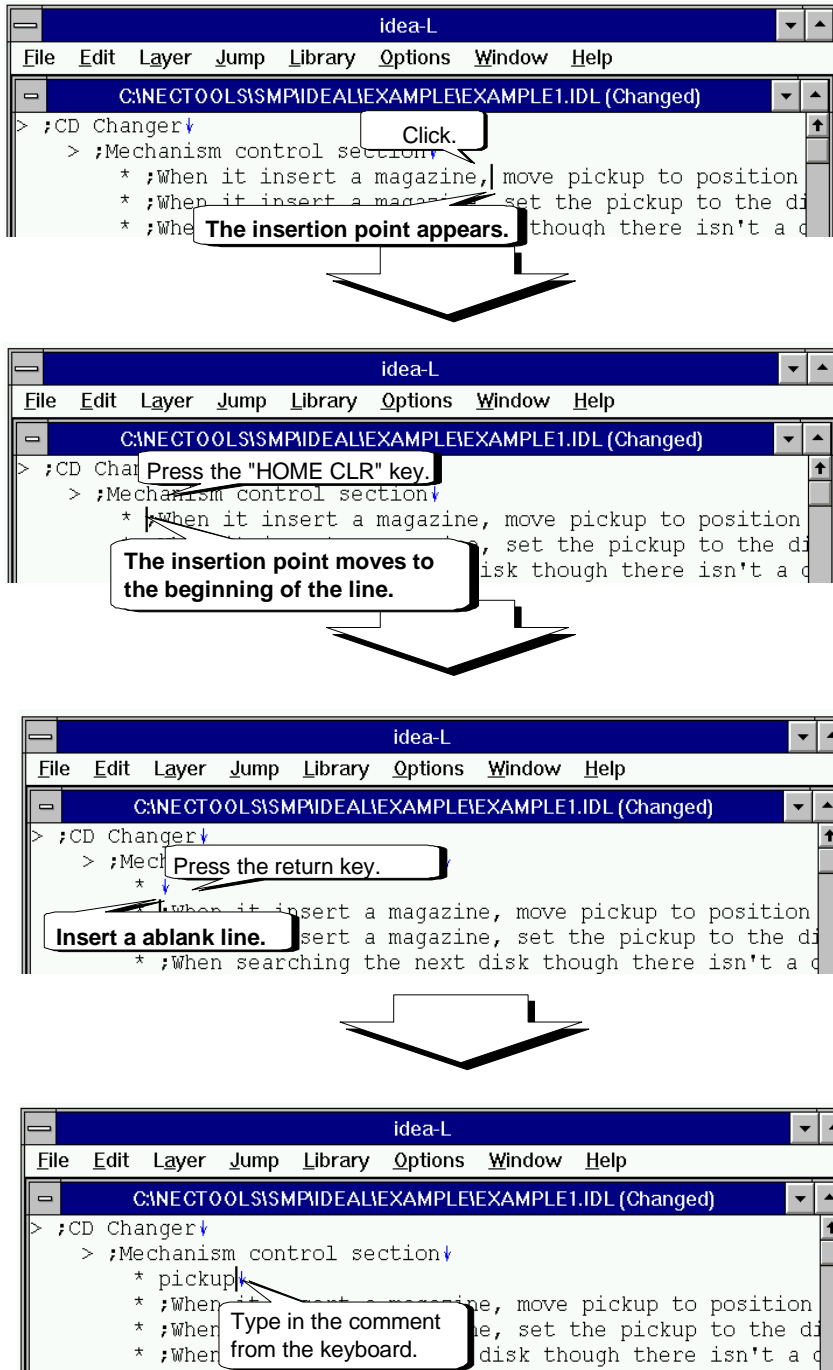



Figure 2-1-10. Adding Comment Lines

After adding the line naming the group, all of the grouped contents are indented.

First, specify the range of lines related to pickup. Press the mouse at the end of the first line and drag it down to the last line in the pickup, and then release the button. After the range is specified, move the mouse cursor to the beginning of the line of the lines specified in the range (any line will do) and press it. Since the mouse cursor changes to , drag to the right and drop.

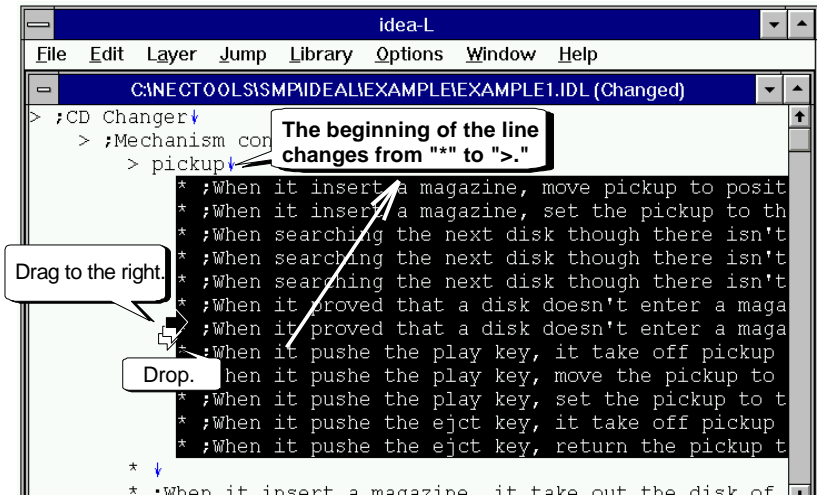
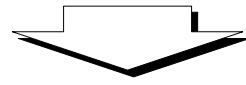
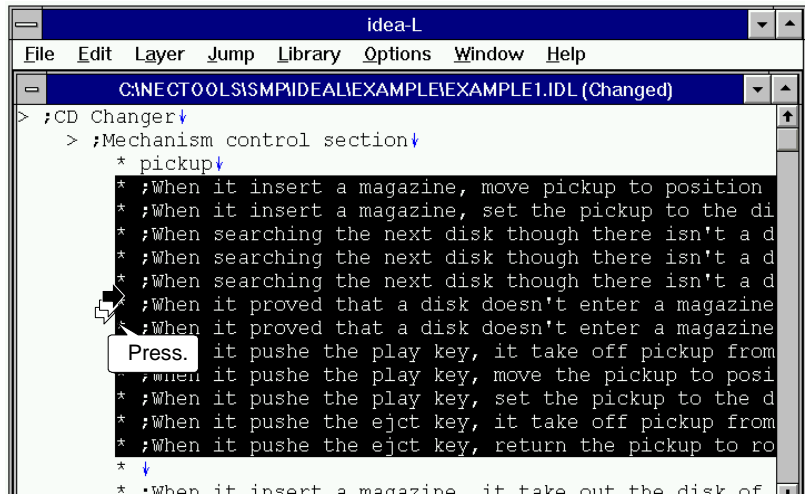
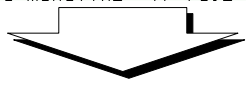
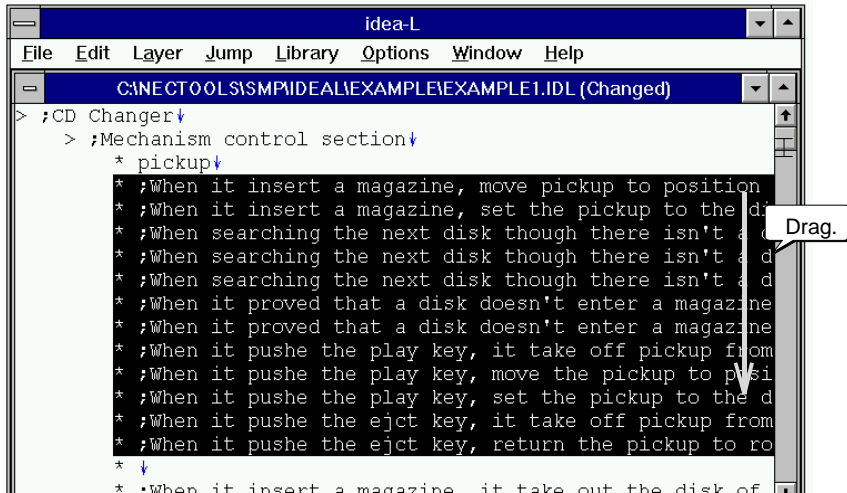


Figure 2-1-11. Selecting Multiple Lines and Moving One Level Down

The lines in the specified range are indented to the right. The character at the beginning of the line "; pickup" changed from "*" to ">." This symbol means that the contents indented in the lines of "; pickup" are one level down. Therefore, the line having a lower level is called the header. The line "; pickup" is the header for all 12 lines (including the blank lines) related to the pickup.

Similarly, the parts related to the magazine are entered in the level below "; magazine."

We will form the magazine hierarchy by only operating the keyboard without using the mouse.

Press the [SHIFT]+ [↓] keys to position the insertion point at the position shown in Figure 2-1-12 and specify the entire range of the magazine. Next, press the [SHIFT]+[CTRL]+ [→] keys. The specified range moves to the lower level.

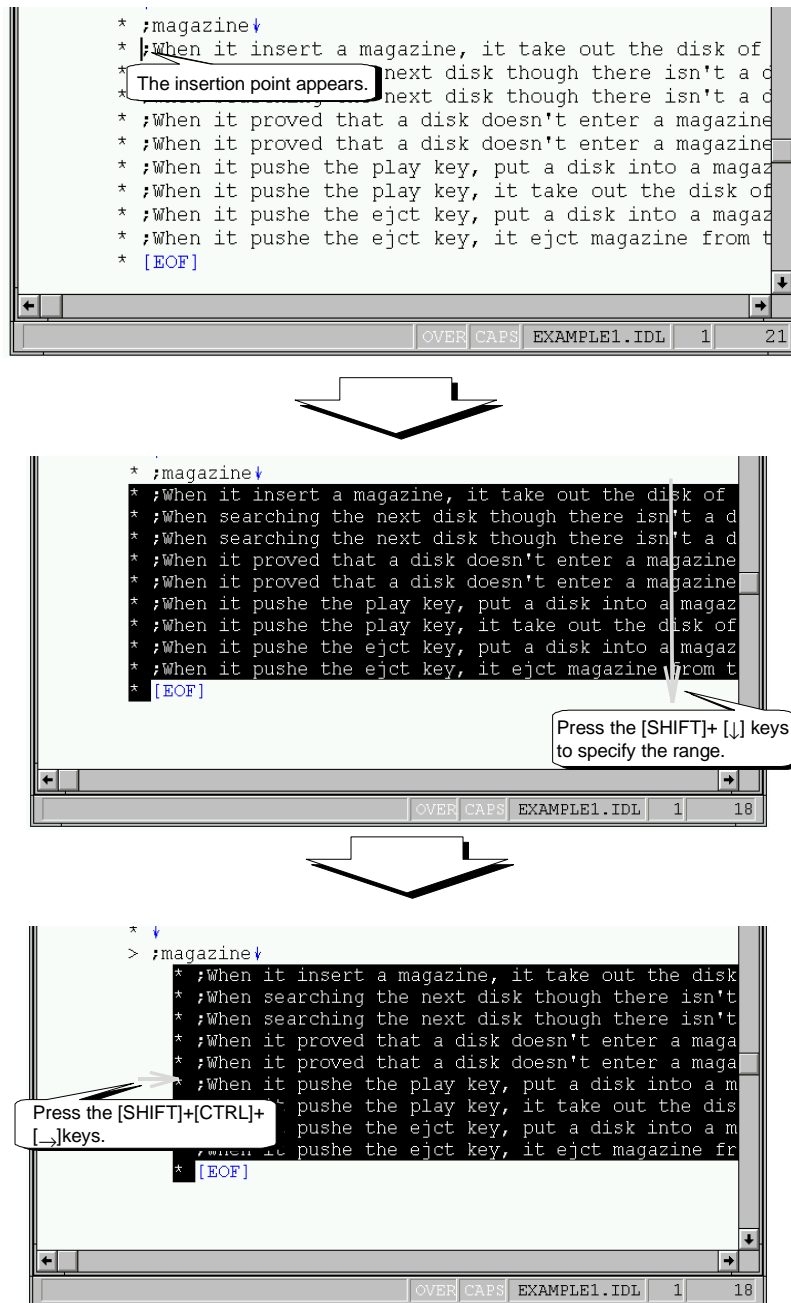


Figure 2-1-12. Completing the Hierarchy

2.1.5 Forming another group and hierarchy

Forming the groups and hierarchies is usually not finished in one try. Even in example1, if the lines for pickup are examined, detailed classifications are still possible. For example, subdivisions can be made into the following four actions.

- Move the pickup to the disk position.
- Set the pickup on the disk.
- Take the pickup off the disk.
- Return the pickup to the home position.

Therefore, the lines related to the pickup can be divided into four types. Let's regroup the lines related to the same action.

Order with the lines for moving the pickup to the disk position in one group followed by the group for setting the pickup on the disk.

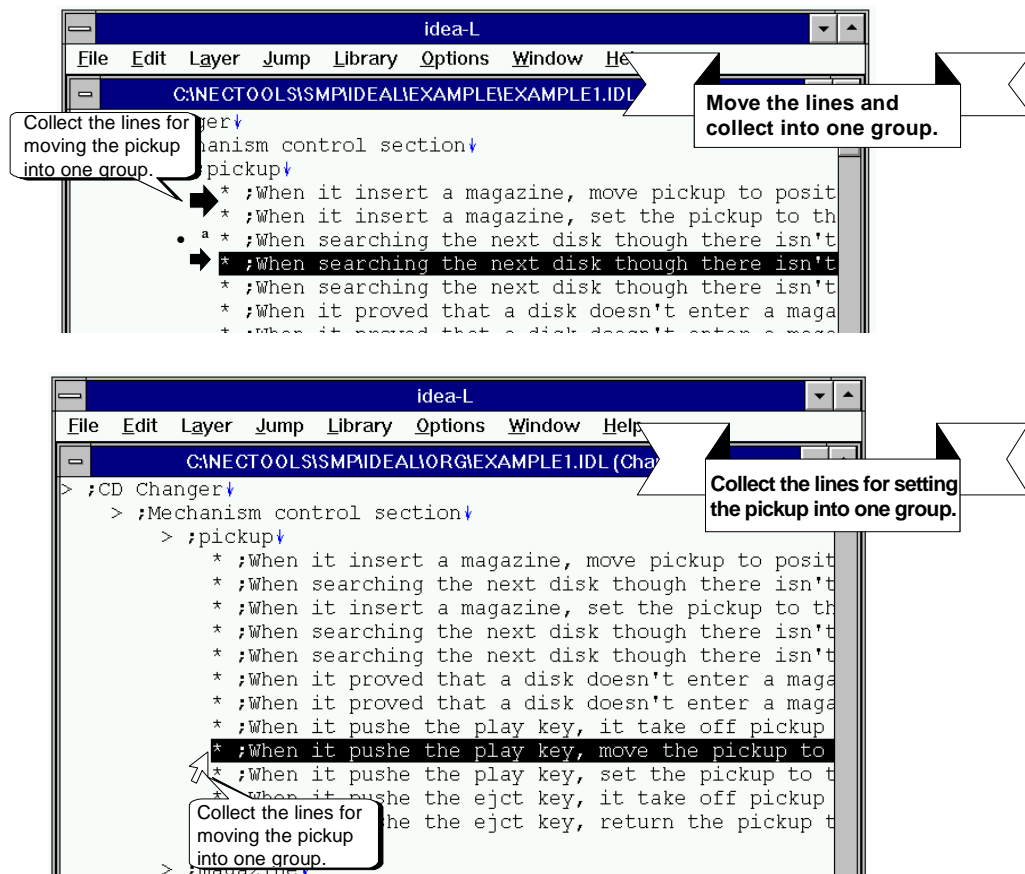



Figure 2-1-13. Collect the Lines Related to Moving and Setting into One Group

The problem of whether multiple lines can be moved may be encountered during grouping. In *idea-L*, multiple, consecutive lines can be moved.

Figure 2-1-14 is an example of moving multiple lines.

First, press the mouse at the beginning of a line in the specification range of multiple, consecutive lines to change the mouse cursor to . Drag and drop this at the top. This moves multiple lines.

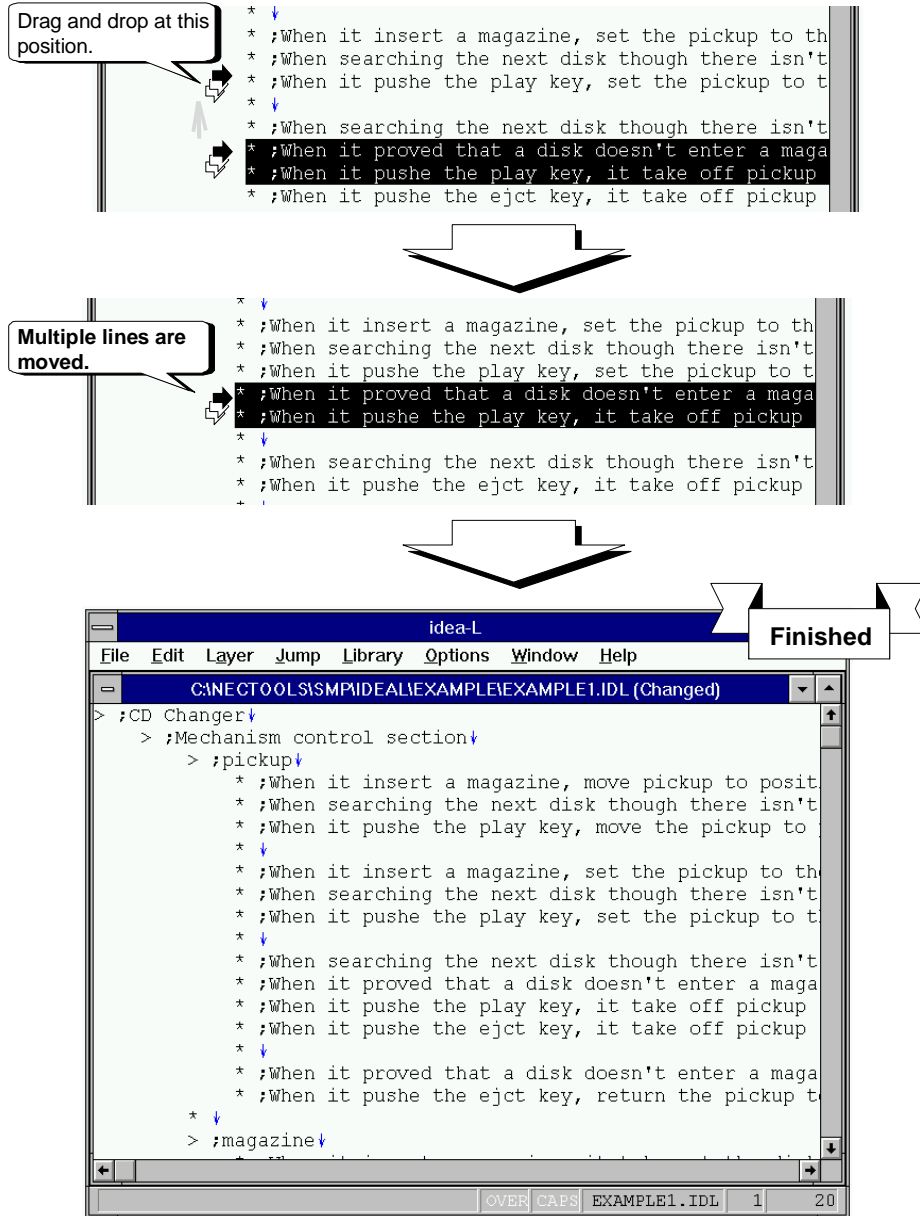


Figure 2-1-14. Moving Multiple Lines and Completing the Grouping

If the internal grouping for the pickup is finished, the following hierarchy is made in the same way. The headers become ";move," ";set," ";take off," and ";return." Form the hierarchy shown in Figure 2-1-15.

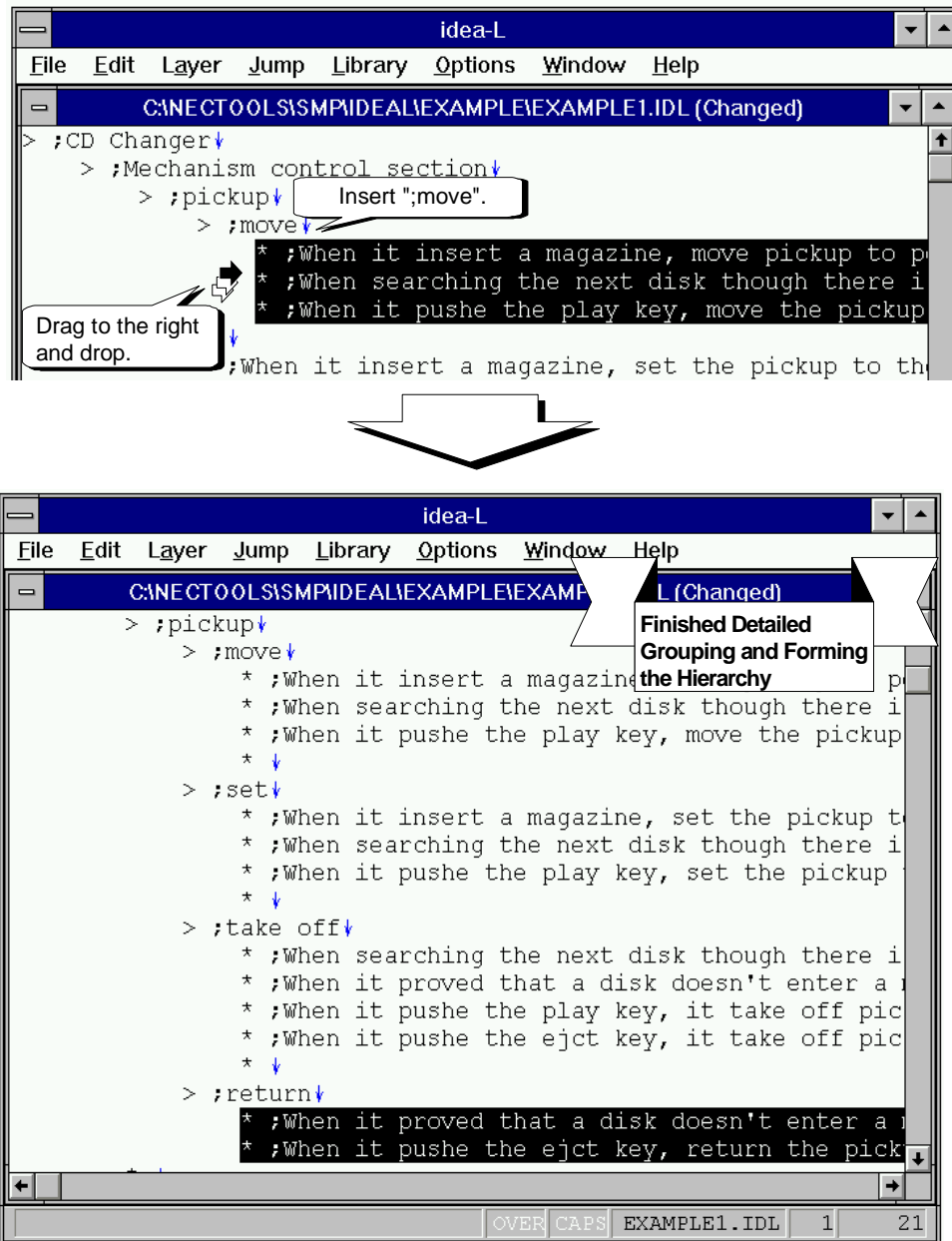


Figure 2-1-15. Completing the Hierarchy for the Pickup

2.1.6 Abstracting

The headers were formed in the hierarchy. By double clicking the mouse on a header, the lower level is closed (hidden).

By double clicking the mouse on the header with a hidden lower level, the hidden lower level opens (appears).

This closed state is called an abstraction in the idea processor.

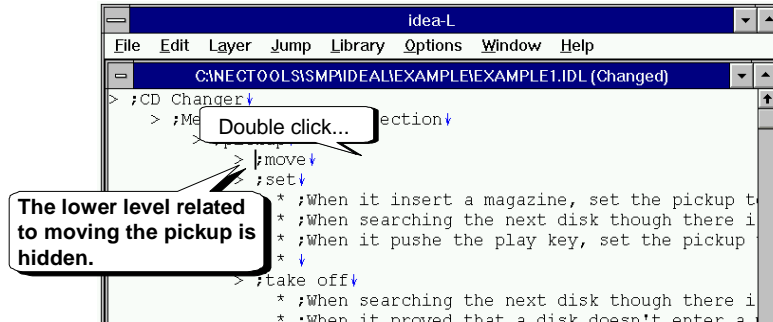


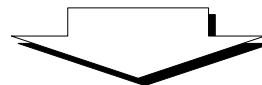
Figure 2-1-16. Closing the Lower Level

Let's only operate the keyboard and not use the mouse. Move the insertion point to the header and press the [CTRL]+[-] keys. The lower level of this header is hidden. Press the [CTRL]+[*] keys when the insertion point is placed at a header with a hidden lower level to open the hidden lower level.

Similarly, hide the lower level for the other headers.

```

> ;CD Changer↓
  > ;Mechanism control section↓
    > ;pickup↓
      > ;move↓
      > ;set↓
        * ;When it insert a magazine, set the pickup t
        * ;When searching the next disk though there i
        * ;When it pushe the play key, set the pickup
        * ↓
    > ;take off↓
      * ;When searching the next disk though there i
  
```



```

> ;CD Changer↓
  > ;Mechanism control section↓
    > ;pickup↓
      > ;move↓
      > ;set↓
      > ;take off↓
        * ;When it proved that a disk doesn't enter a
        * ;When it pushe the play key, it take off pic
        * ;When it pushe the play key, it take off pic
  
```



```

> ;CD Changer↓
  > ;Mechanism control section↓
    > ;pickup↓
      > ;move↓
      > ;set↓
      > ;take off↓
      > ;return↓
    * ↓
  > ;magazine↓
    * ;When it insert a maqazine, it take out the disk
  
```

Figure 2-1-17. Completing the Abstraction

By displaying only four lines for the pickup as shown in Figure 2-1-17, the contents are organized and abstracted.

2.1.7 Editing practice

Finally, grouping, forming a hierarchy, and abstracting for the magazine are performed in essentially the same way as for the pickup. Try to abstract the entire mechanism control section.

```

> ;magazine↓
* ;When it insert a magazine, it take out the disk
* ;When searching the next disk though there isn't
* ;When it proved that a disk doesn't enter a maga
* ;When it pushe the play key, it take out the dis
* ↓
* ;When searching the next disk though there isn't
* ;When it proved that a disk doesn't enter a maga
* ;When it pushe the play key, put a disk into a m
* ;When it pushe the ejct key, put a disk into a m
* ↓
* ;When it pushe the ejct key, it ejct magazine fr
* [EOF]

```

Figure 2-1-18. Grouping the Magazine

```

> ;magazine↓
> ;take out↓
* ;When it insert a magazine, it take out the
* ;When searching the next disk though there i
* ;When it proved that a disk doesn't enter a
* ;When it pushe the play key, it take out the
* ↓
> ;put into↓
* ;When searching the next disk though there i
* ;When it proved that a disk doesn't enter a
* ;When it pushe the play key, put a disk into
* ;When it pushe the ejct key, put a disk into
* ↓
> ;eject↓
* ;When it pushe the ejct key, it ejct magazin
* [EOF]

```

Figure 2-1-19. Forming the Magazine Hierarchy

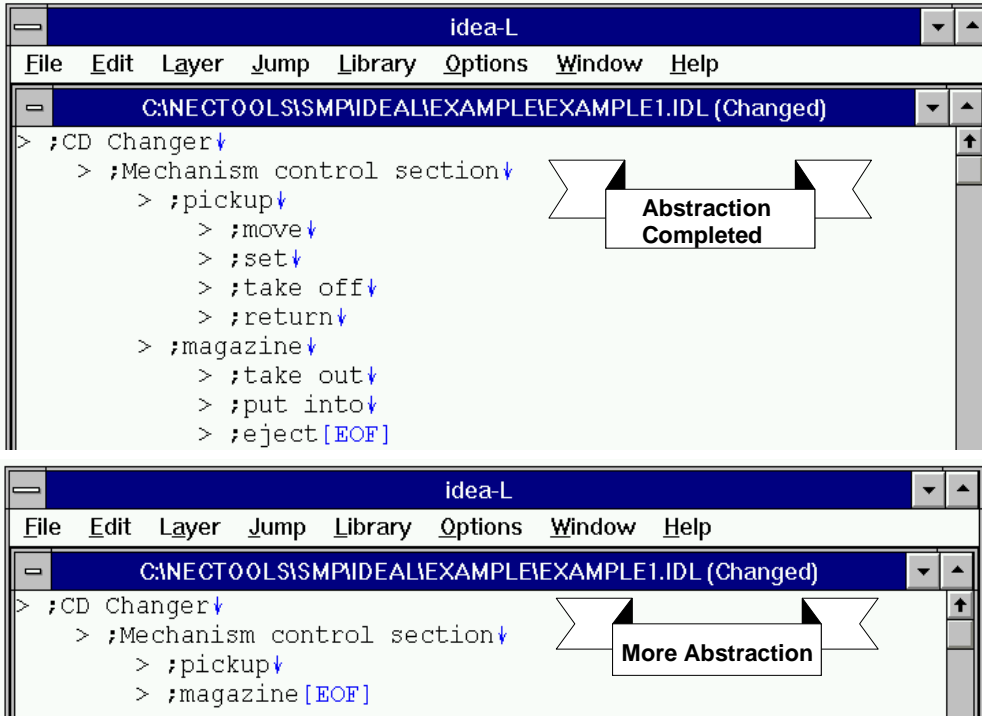


Figure 2-1-20. Completing the Abstraction

2.1.8 Saving to a file

This step saves a file that was abstracted by the idea processor.

First, click the "File" menu. When the pull-down menu appears, click the "Save" command.

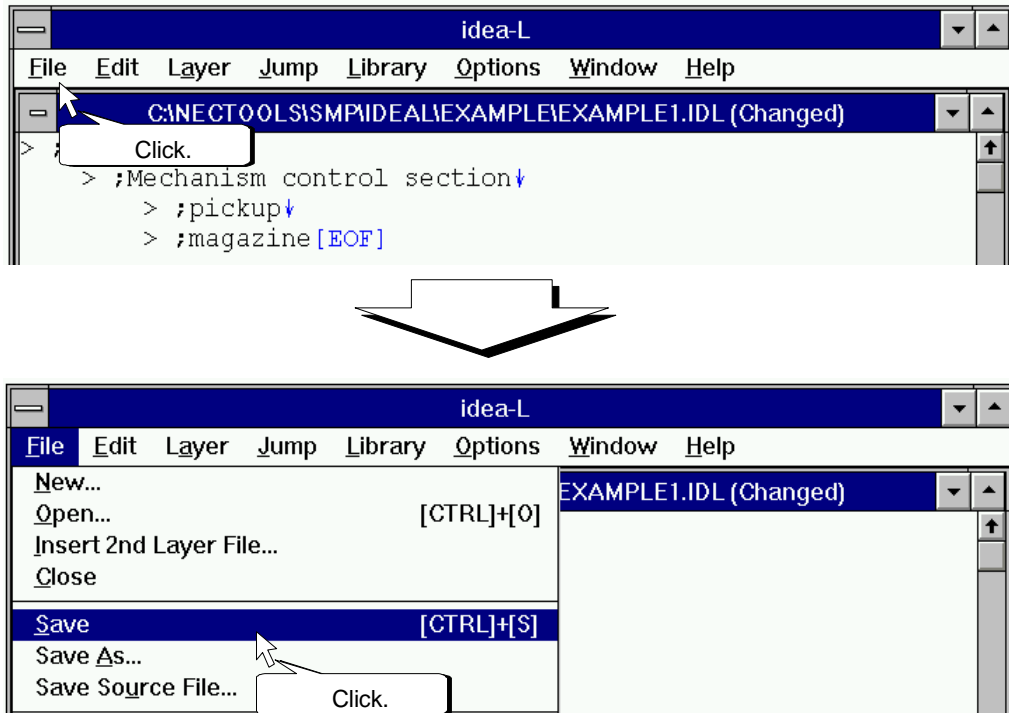


Figure 2-1-21. Click the File Menu

If you want to save under another file name, click the "Save As..." command in the pull-down menu. When the "Save As" dialog box appears, enter the file name in the text box and click the "OK" button. In this case, this example saves "example1" with the file name "ex11". Try this and see.

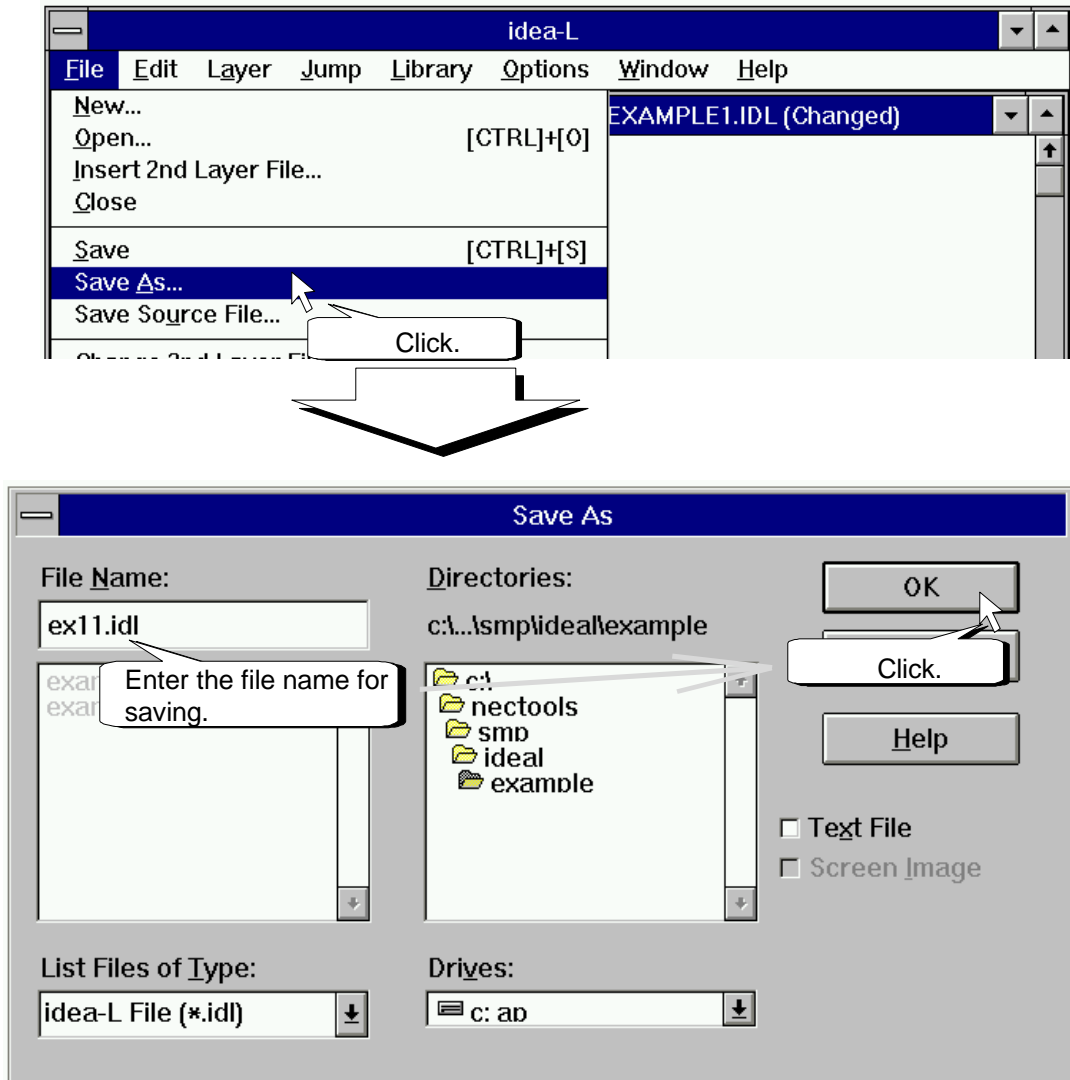


Figure 2-1-22. Saving With Another Name

This ends section 2.1, "Idea Processor Functions."

Before starting section 2.2, "Library Functions," click the "Close" command in the "File" menu to exit the window used in section 2.1, "Idea Processor Functions."

2.2 Library Functions

Typically, when a programmer uses libraries, he begins by reading a thick manual. However, this task does not demand completely remembering the library functions. Usually, the programmer only remembers the keywords (sometimes, the entry location).

If this task is performed efficiently, the programmer looks up the keyword when needed and can check unclear (or unknown) function data.

However in this task, if the programmer uses this function any number of times in software development, he becomes able to use this type of operation without referring to the manual.

Therefore, the library functions are not necessarily required tools when the functions are included. There are no problems even if functions are directly included in the source file. The library function is a tool that supports the programmer's knowledge.

Section 2.2, "Library Functions," describes the creation, registration, and use of functions by using an example (CD changer program using the 78K/0 assembler language).

2.2.1 Creating a function

A function will be created. Since functions are being written, a function will be completed as the sample program. The sample program is nectools\smp\ideal\example.

Click the "Open" command in the "File" menu. Since the "Open" dialog box appears, click the function file (*.fnc) in the "List of Files of Type" drop-down list. The list of functions in the list box appears, so double click "pick_up.fnc."

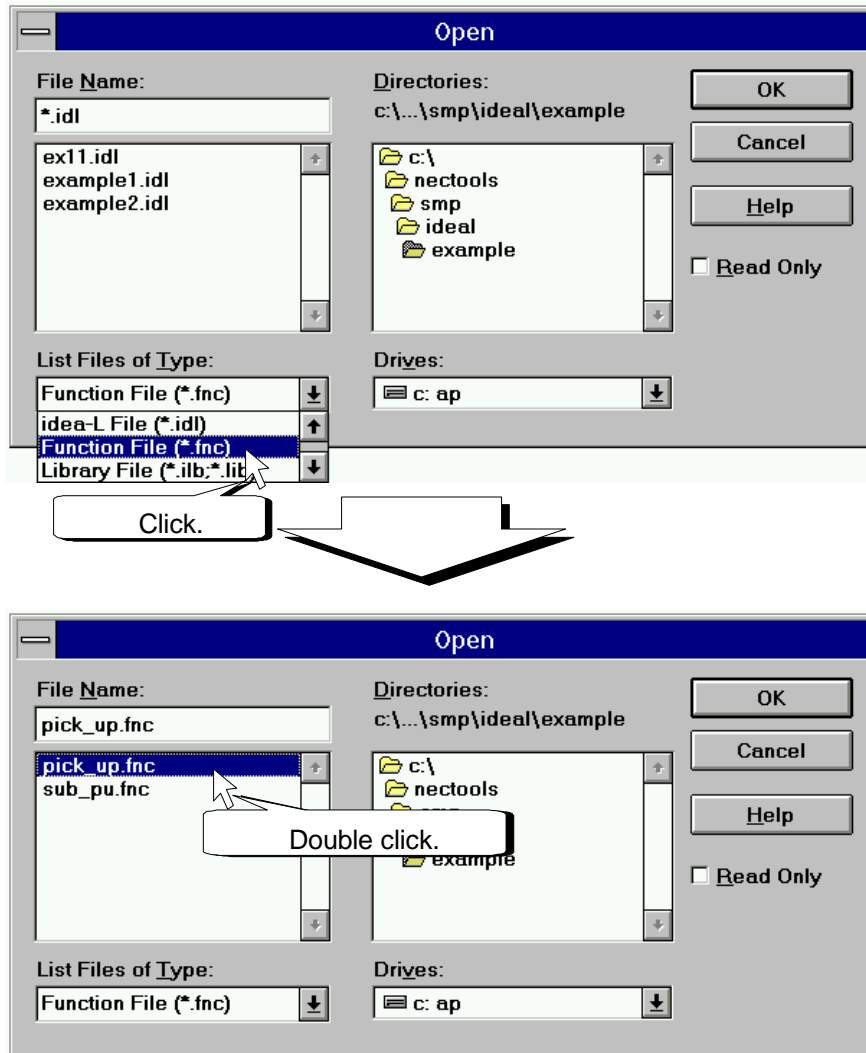


Figure 2-2-1. Open a Function File

pick_up.fnc is read. Its contents are divided into three windows and opened.

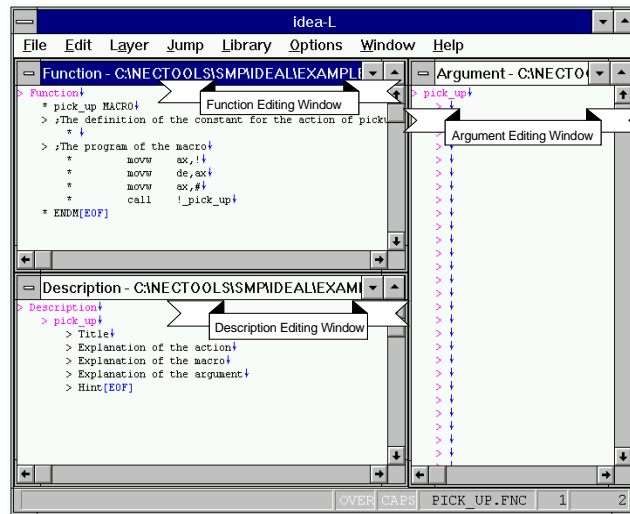


Figure 2-2-2. Open State of a Function File

The function is described before writing the function.

A library file function is described in an assembler language macro.

When the user describes the function (macro) in a program and executes the "Save Source File..." command, the source file and the personal library file are created. The personal library file is an abstraction of the function body used by the program from the library file. When writing a function, the macro body, arguments, and function description are written.

idea-L Screen

```

> ;The section of definition
* name test
* work dseg at 0fee0h
* CD_ID: ds 1
> ;Main Routine
* cseg
* start:
* pick_up _move,CD_ID ;~#
* pick_up _put,CD_ID ;~#
* pick_up _release,CD_ID ;~#CD_CHG.ILB
* br !start

```

↓ Execute the "Save Source File..." command.

Assembler file (.ASM)

```

* |~C:\NECTOOLS\SMP\IDEAL\EXAMPLE\EXAMPLE2.IDL
* ;Sub Title
* ;The section of definition
* $ INCLUDE (C:\NECTOOLS\SMP\IDEAL\EXAMPLE\EXAM001.PLB)
* name test
* work dseg at 0fee0h
* CD_ID: ds 1
* ;Main routine
* cseg
* start:

```

Personal library file (.PLB)

```

* pick_up MACRO,action,CD_number
* ;The definition of the constant
* _move set 1
* _put set 2
* _release set 3
* _remove set 4
*
* ;The program of the macro
* movw ax,!CD_number
* movw de,ax
* movw ax,#action
* call !_pick_up
* ENDM

```

Figure 2-2-3. Function Structure

The "pick_up" function is a macro that calls the pickup subroutine of some CD changer. The descriptions related to the arguments are still not in this macro. The argument relationships are described to complete the function.

The input conditions of the pickup subroutine, "_pick_up," are:

Input conditions of the "_pick_up" subroutine	
Dummy Argument	Contents
action	ax ← 1, Move the pickup. 2, Set the pickup. 3, Release the pickup. 4, Return the pickup to the home position.
CD_number	de ← CD number

First, as shown in Figure 2-2-4, describe "action" in the first dummy argument and "CD_number" in the second dummy argument.

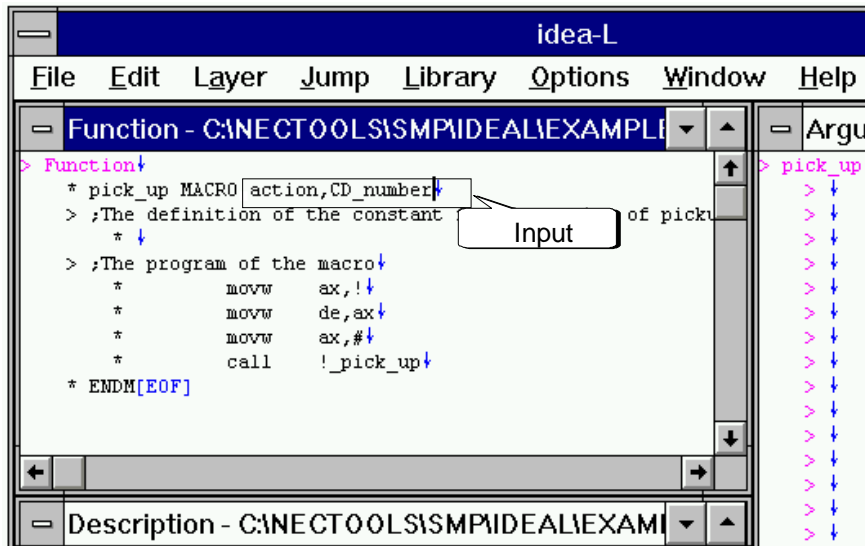


Figure 2-2-4. Setting the Dummy Arguments

Next, the side that accepts the arguments in the macro body is described. Describe as shown in Figure 2-2-5.

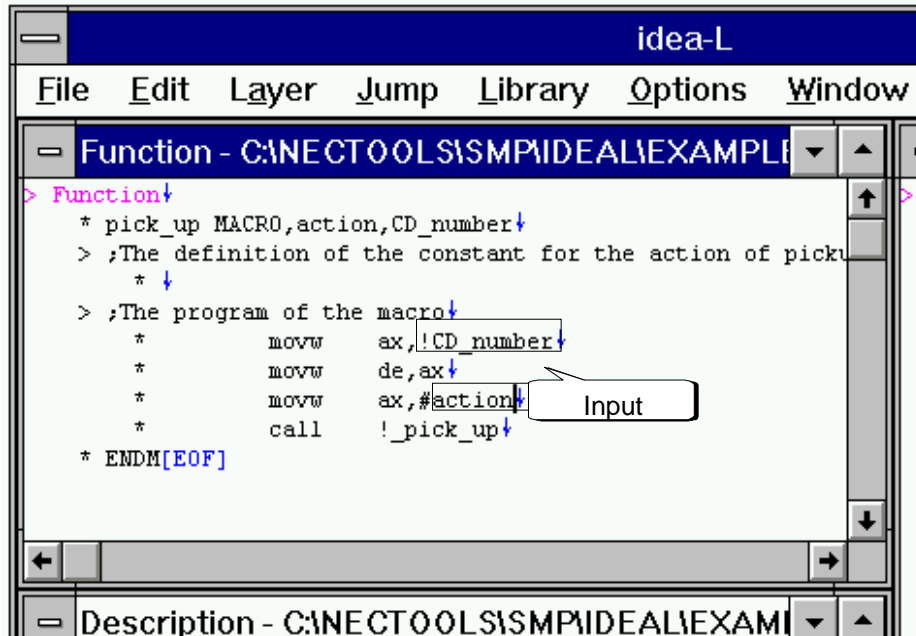


Figure 2-2-5. Argument Description

This completes the descriptions of the argument portion in the function editing window.

Next, the argument editing window is set.

While returning again to the description of the function contents, there are four types of actions in this pickup. These actions are passed as action numbers via the ax register. However, when a function is called in a program, in order to make the meaning of the number indicating the action easy to understand when viewed later and due to the increasing possibility of input errors, the action itself is described by a string constant that exactly represents it and not by directly describing a numerical value. Although this will be described in detail in section 2.2.3, "Using functions," there are menu functions to select the argument candidates of the function in the browser editing window of the library. The argument candidates are set in the argument editing window.

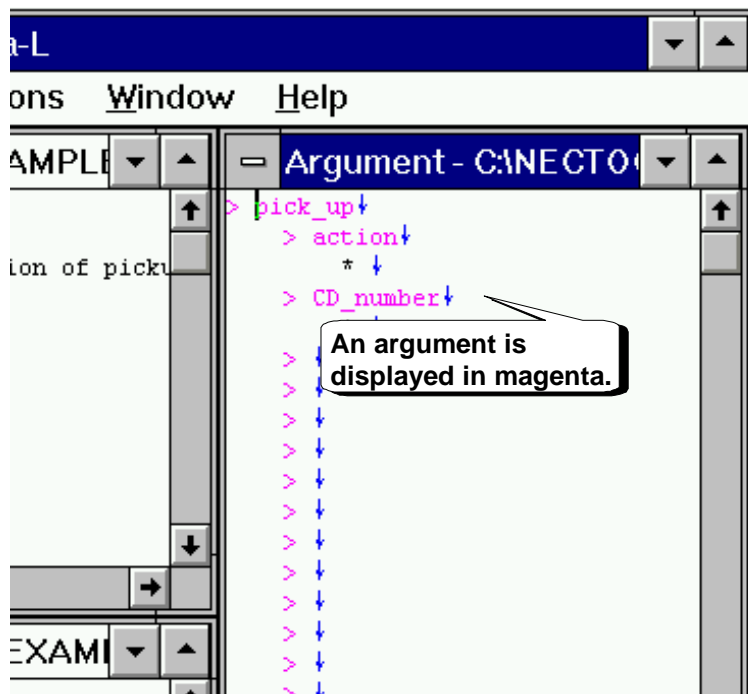


Figure 2-2-6. Click the Argument Editing Window

As shown in Figure 2-2-7, four types of actions are described in the first argument, action. Nothing is described in the second argument, CD_number.

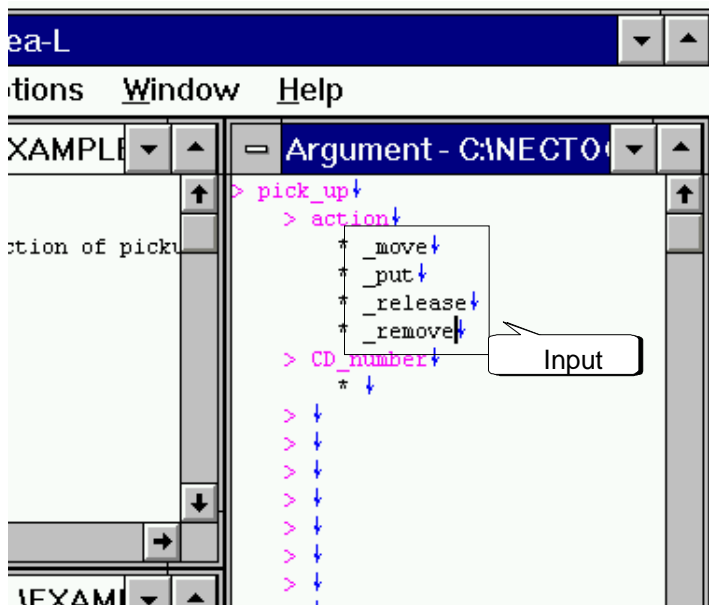


Figure 2-2-7. Description of the Argument Candidates

When the arguments of the function are set in the browser editing window, they can be selected from these four types. The next section converts a string into a numerical value. The SET pseudo-instruction converts a string into a numerical value in the assembler. Now we will use this.

Specify the range of the four lines input in the argument editing window by selecting multiple lines as in section 2.1.4, "Forming a hierarchy." After specifying, press the [CTRL]+[c] keys to copy to the clipboard. Next, click the function editing window, press the [CTRL]+[v] keys at the position in Figure 2-2-8, and paste the contents of the clipboard.

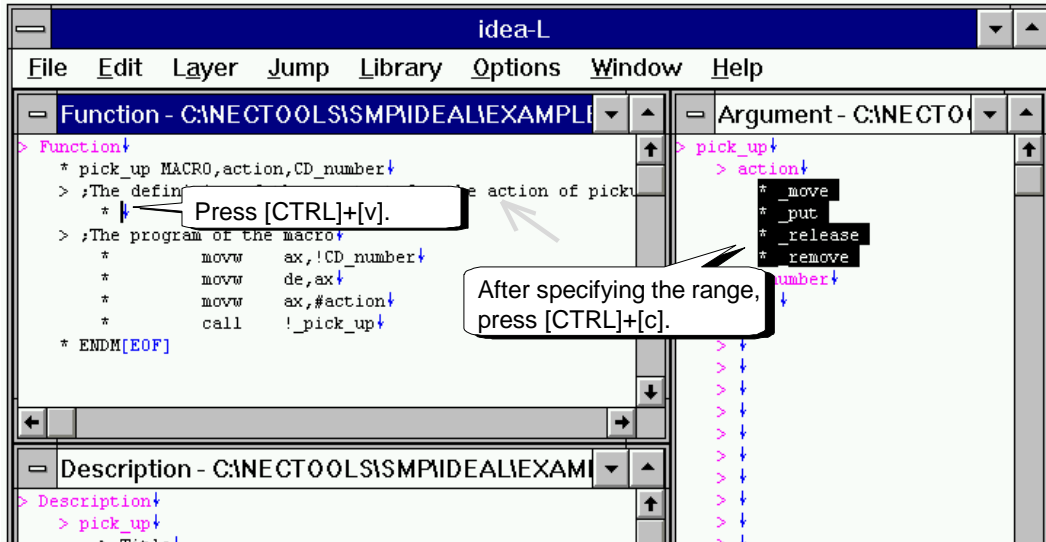


Figure 2-2-8. Copy the Argument Candidates and Paste (1)

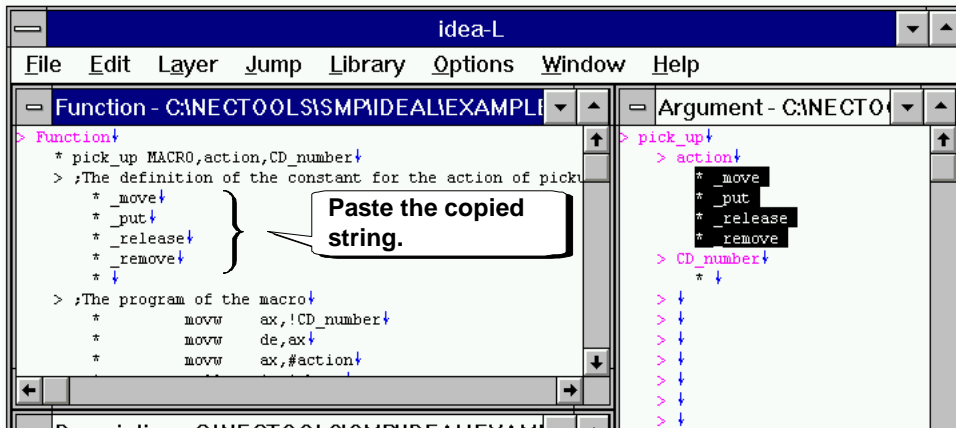


Figure 2-2-9. Copy the Argument Candidates and Paste (2)

As shown in Figure 2-2-10, pasting lines by using the SET pseudo-instruction is completed.

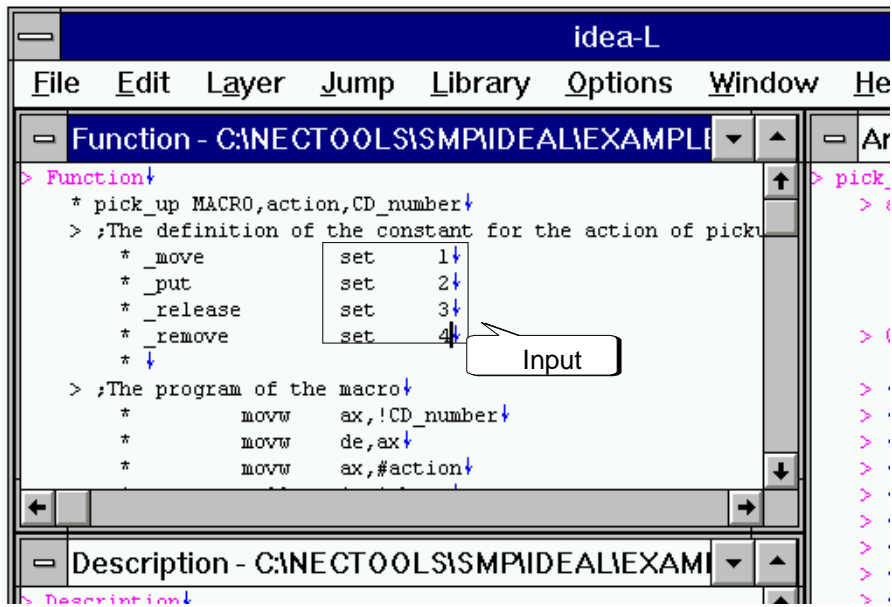


Figure 2-2-10. Assigning Numerical Values to the Strings of the Argument Candidates

The function that sets the numerical value by selecting a string was finished. (For example, when "_move" is selected in the first argument, the numerical value of 1 is set to the first dummy argument, "action.")

Next, the description of this function is written. Click in the description editing window. Double click the title line to open the lower level. Enter "call the pickup action" from the keyboard.

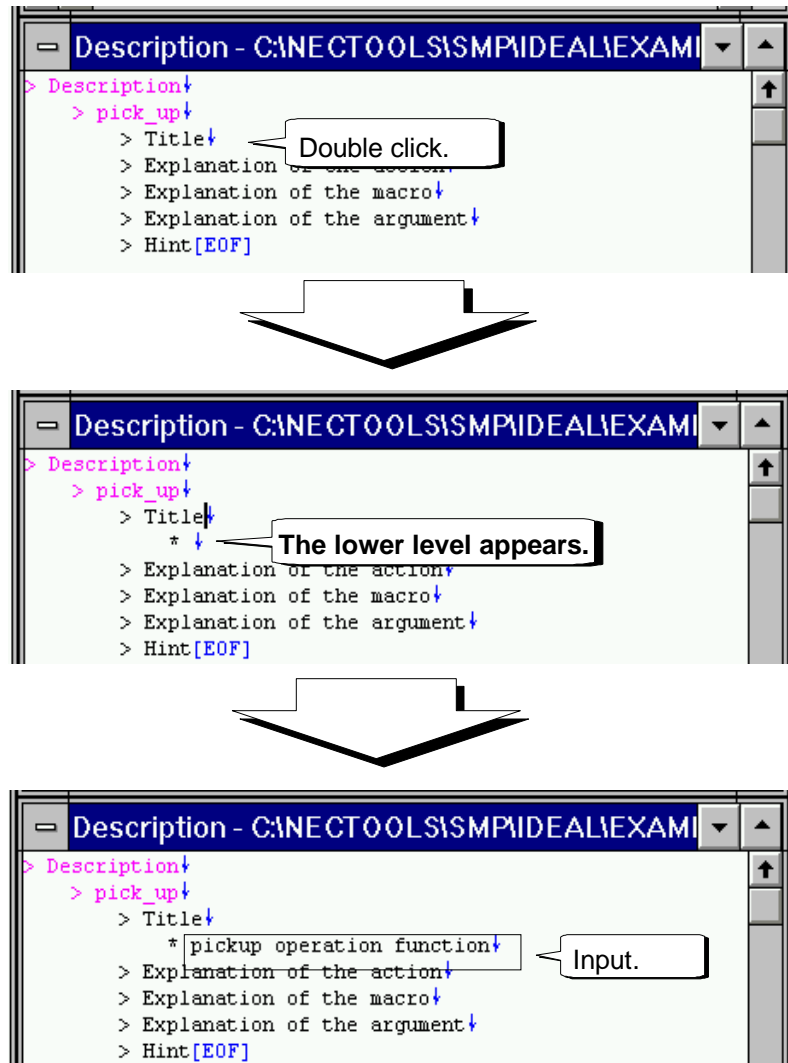


Figure 2-2-11. Enter a Comment Line in the Description Editing Window of the Function File

2.2.2 Registering functions

After a function is created, we will register it in the library.

Click the "Open" command in the "File" menu. Click "Library File (*.lib)" in the "List Files of Type" drop-down list in the "Open" dialog box. When the list of the libraries in a list box is displayed, double click the "cd_chg.lib" library file.

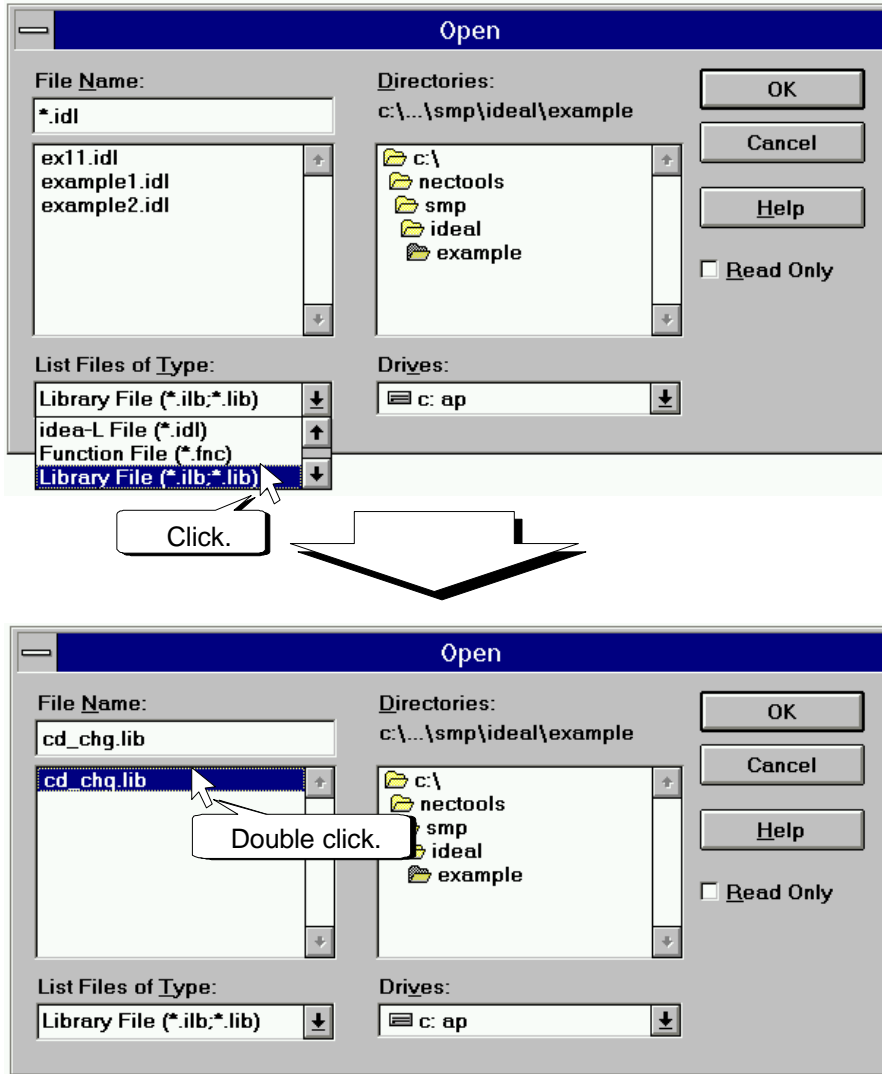


Figure 2-2-15. Open a Library File

Open the cd_chg library file. The library editing window appears. As in the idea processor, this window has the functions from grouping to abstracting.

Double click the pickup part to display the lower level. Click at the end of the line "Call the function for pickup," then the insertion point appears at the end of the line. Click the "Function Registration..." command in the "Library" menu. Since the dialog box for function registration appears, click "pick_up.fnc" in the "Function File" list box and then click the "OK" button.

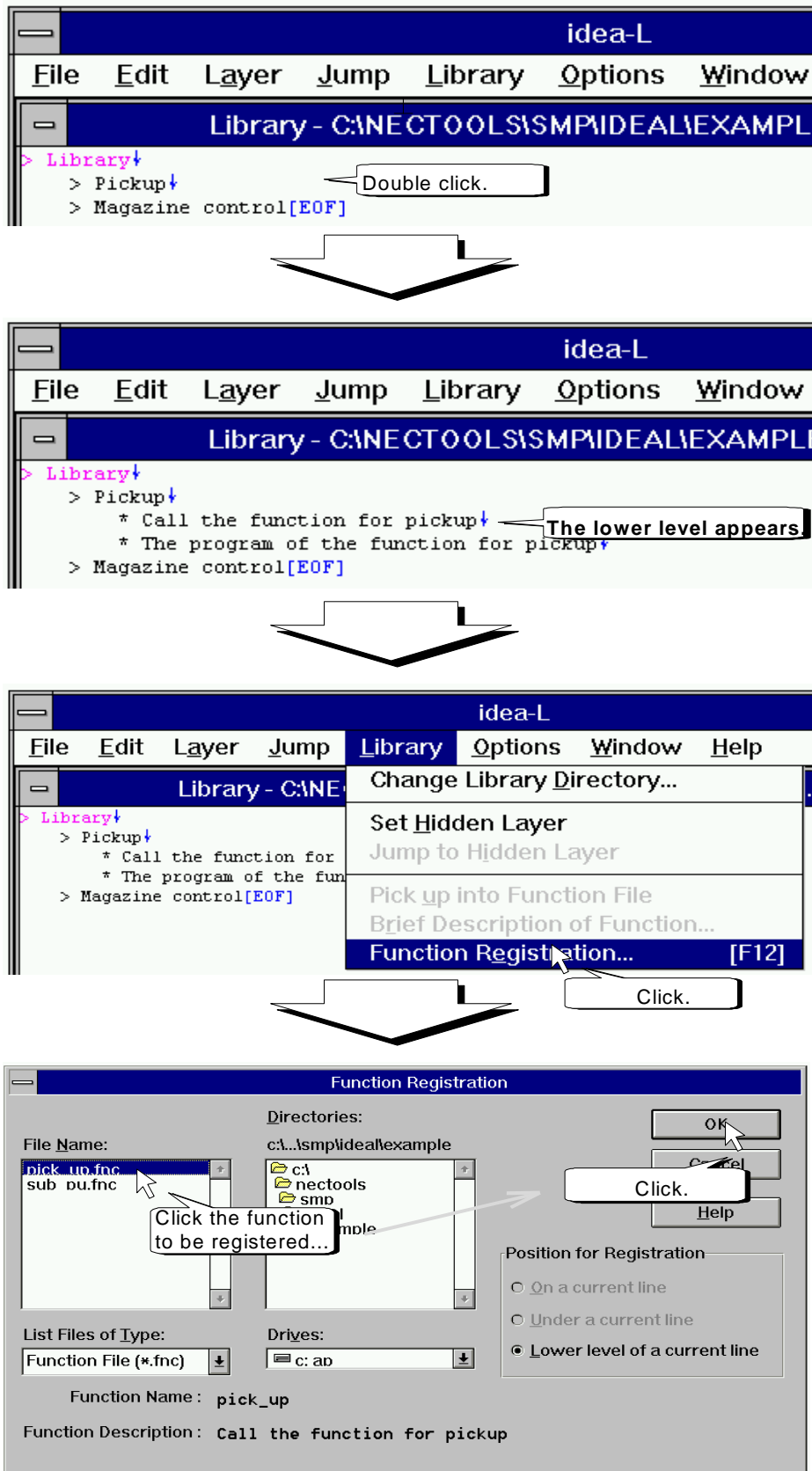


Figure 2-2-16. Registering a Function to a Library File

The specified function is registered. The level opens and the function name is displayed in magenta.

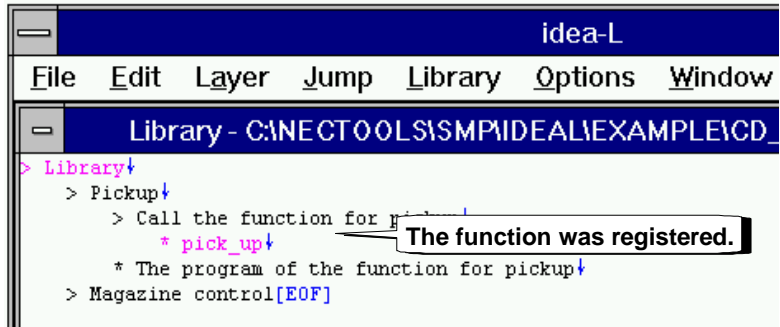


Figure 2-2-17. Completing the Function Registration

Similarly, we will register the function of the subroutine for the pickup actions.

Move the insertion point to the end of the line of the pickup function and register the sub_pu function. After the function is registered in two libraries, click the "Save" command in the "File" menu to save.

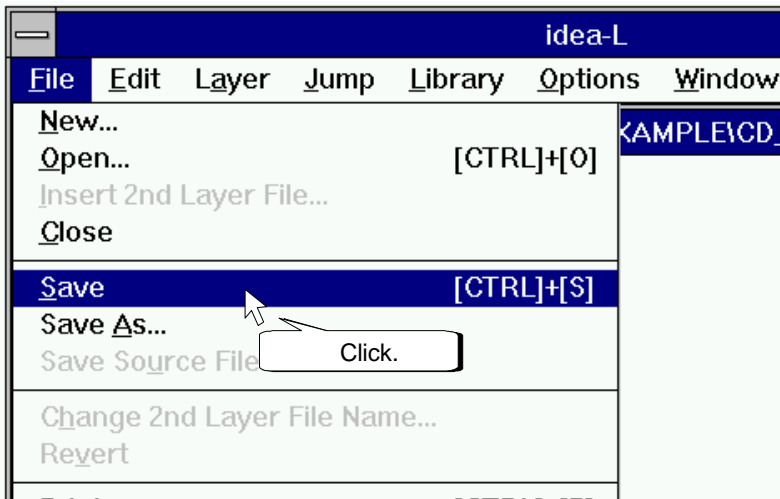
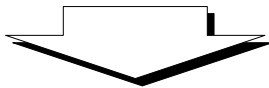
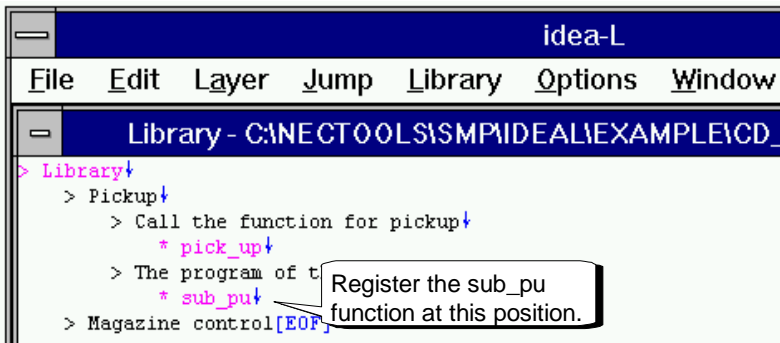


Figure 2-2-18. Finish and Save All Function Registrations

After saving the file, click the "Close" command in the "File" menu to exit this file.

2.2.3 Using functions

The registration of the function to a library file ended, and the level for using functions was entered. Before doing this, however, the library directory must be set.

In *idea-L*, the library directory can be set in two places. Each directory has its priority set to the first candidate or the second candidate to be able to use it. For example, when a personal library has priority, the personal library is set to the first candidate. A shared library is set to the second candidate.

Now, we will set library directories.

Click the "Change Library Directory..." command in the "Library" menu. Since the "Change Library Directory" dialog box appears, set a directory of "ch_chg.ilb" as the library directory.

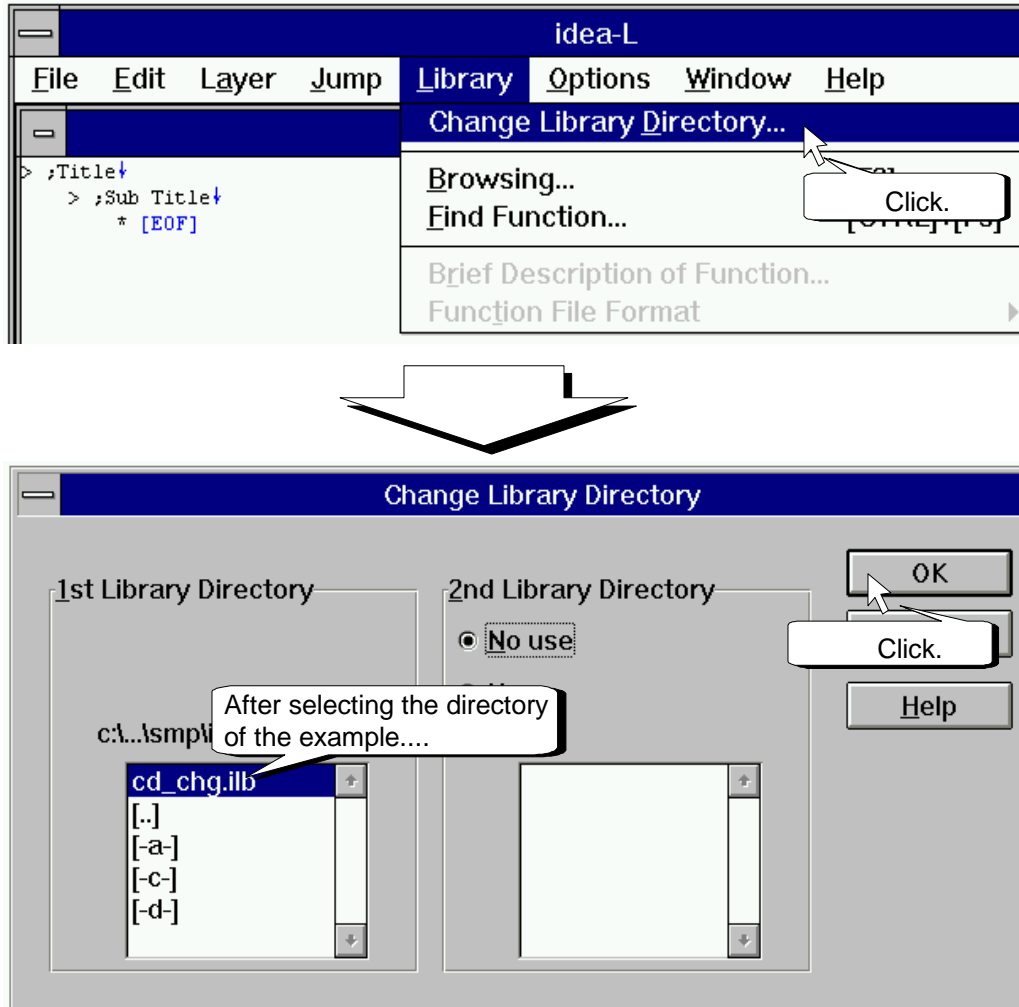


Figure 2-2-19. Setting the Library Directory

This completes the preparation.

Open "example2.idl" as a sample program that uses functions.

Click the "Open" command in the "File" menu. Double click "example2.idl" in the "Open" dialog box.

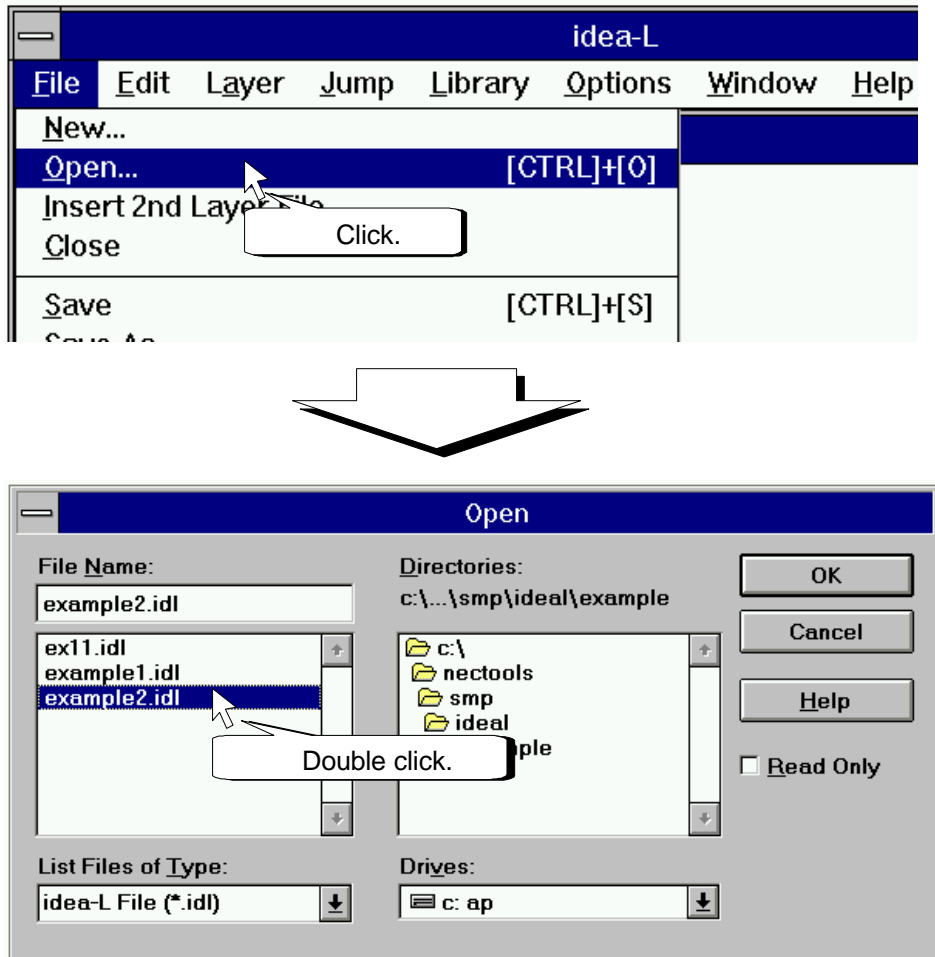


Figure 2-2-20. Open example2


```
> ;Title
> ;Sub Title
> ;The section of definition
* name test
* work dseg at 0fee0h
* CD_ID: ds 1
> ;Main routine
* cseg
* start:
*
* br !start
>
* magazine _getCD,CD_NUM ;~#CD_CHG.LIB
> ;Sub routine
*
> ;End
* end[EOF]
```

Figure 2-2-21. Contents of example2

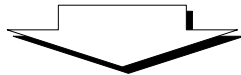
The four ways to use a function are

- (a) Use a known function
- (b) Use while examining the arguments
- (c) Search for the target function from the library
- (d) Examine the function data

(a) Use a known function

If the name, contents, and arguments of the function to be used are known, the function is directly described without using a library function. Since this is clearly a function, append ";~#" to the end of the function.

```
> ;Title
> ;Sub Title
> ;The section of definition
* name test
* work dseg at 0fee0h
* CD_ID: ds 1
> ;Main Routine
* cseg
* start:
*
* br !start
>
* magazine _getCD,CD_NUM ;~#CD_CHG.LIB
> ;Sub routine
*
> ;End
* end[EOF]
```



```
> ;Title
> ;Sub Title
> ;The section of definition
* name test
* work dseg at 0fee0h
* CD_ID: ds 1
> ;Main routine
* cseg
* start:
*
* pick_up_move,CD_ID ;~#
* pick_up_put,CD_ID ;~#
*
* br !start
>
* magazine _getCD,CD_NUM ;~#CD_CHG.LIB
> ;Sub routine
*
> ;End
* end[EOF]
```

Figure 2-2-22. Using a Known Function

(b) Use while examining the arguments

Sometimes the function name is recalled, but the order and contents of the functions are unclear. In this example, the target function is searched for by function name, detailed information about its arguments is verified, and the arguments are completed.

In the next example, it is assumed that the function name of "pick_up" is recalled, but its arguments are not known. We will use the function while investigating its arguments. First, enter the function name as shown in Figure 2-2-23. Next, use the mouse to click the first input character "*" in the input lines to specify the range of lines.

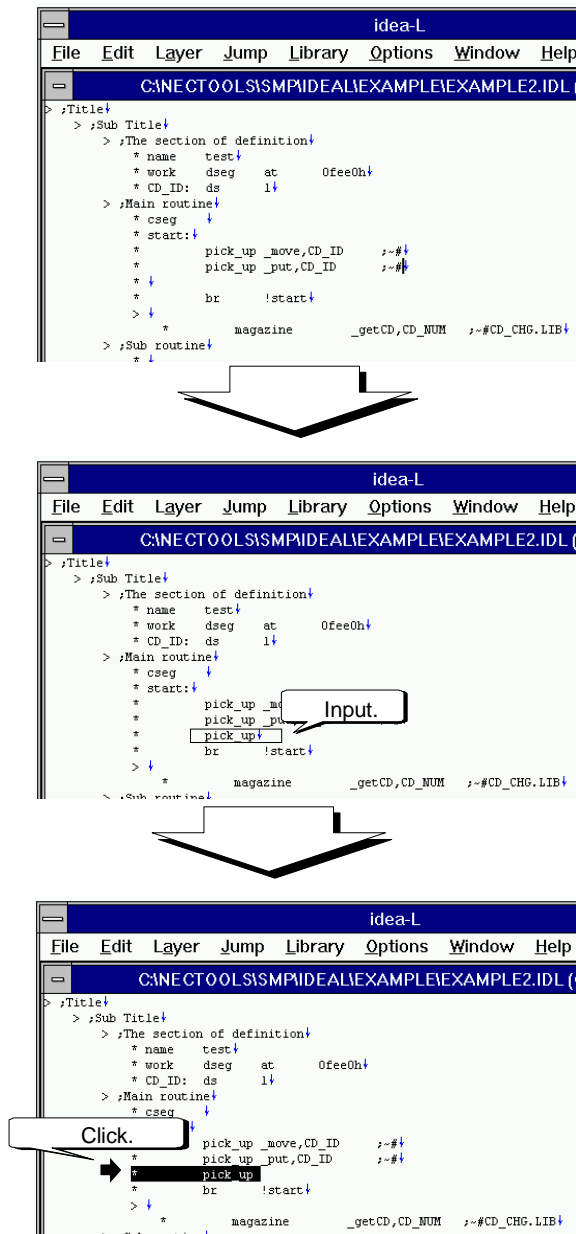


Figure 2-2-23. Range Specification of a Function with Unclear Arguments

Click the "Find Function..." command in the "Library" menu to display the "Find Function" dialog box. The function name in the range specified in Figure 2-2-23 is entered in the "Find What" text box. If you do not know which directory the library file is in, select the "All Files" check box as in Figure 2-2-24 and search.

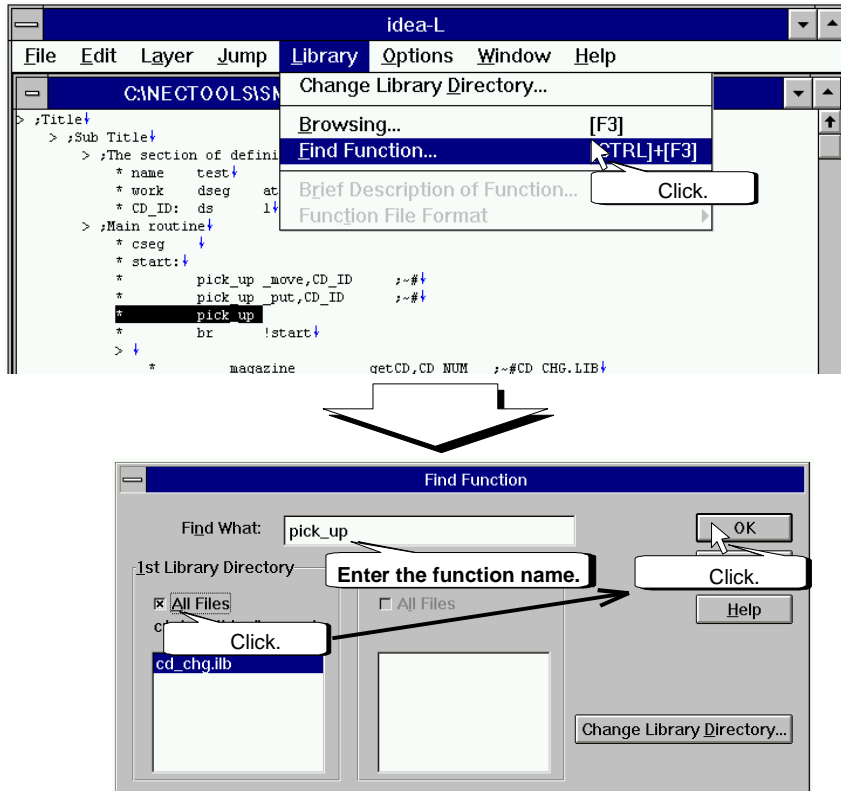


Figure 2-2-24. Function Search

If the function is found, the library browsing window appears and the information related to the target function is displayed.

The contents showing the functions registered in the library and the arguments of each function are displayed in the library browsing window. If the displayed contents are insufficient, select the "Description..." button shown in Figure 2-2-25 to display the Description window.

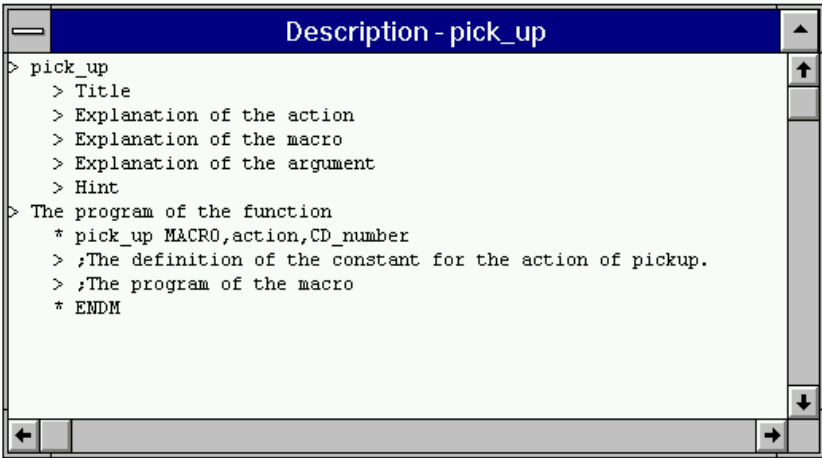
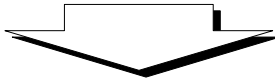
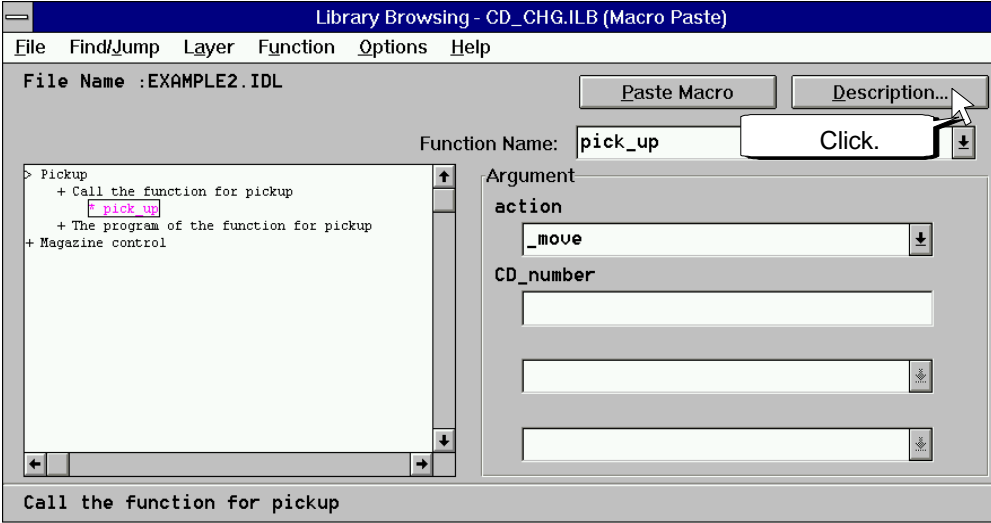


Figure 2-2-25 Checking the Function Contents

When setting this example, the contents of the arguments are not known. Therefore, we will check the contents of the arguments. Double click "Argument Description" in the Description window to display the list of dummy arguments. There is a lower level at the "action" dummy argument in the list. Double click and see what happens. In the lower level, there is a list of candidate arguments for the "action" dummy argument.

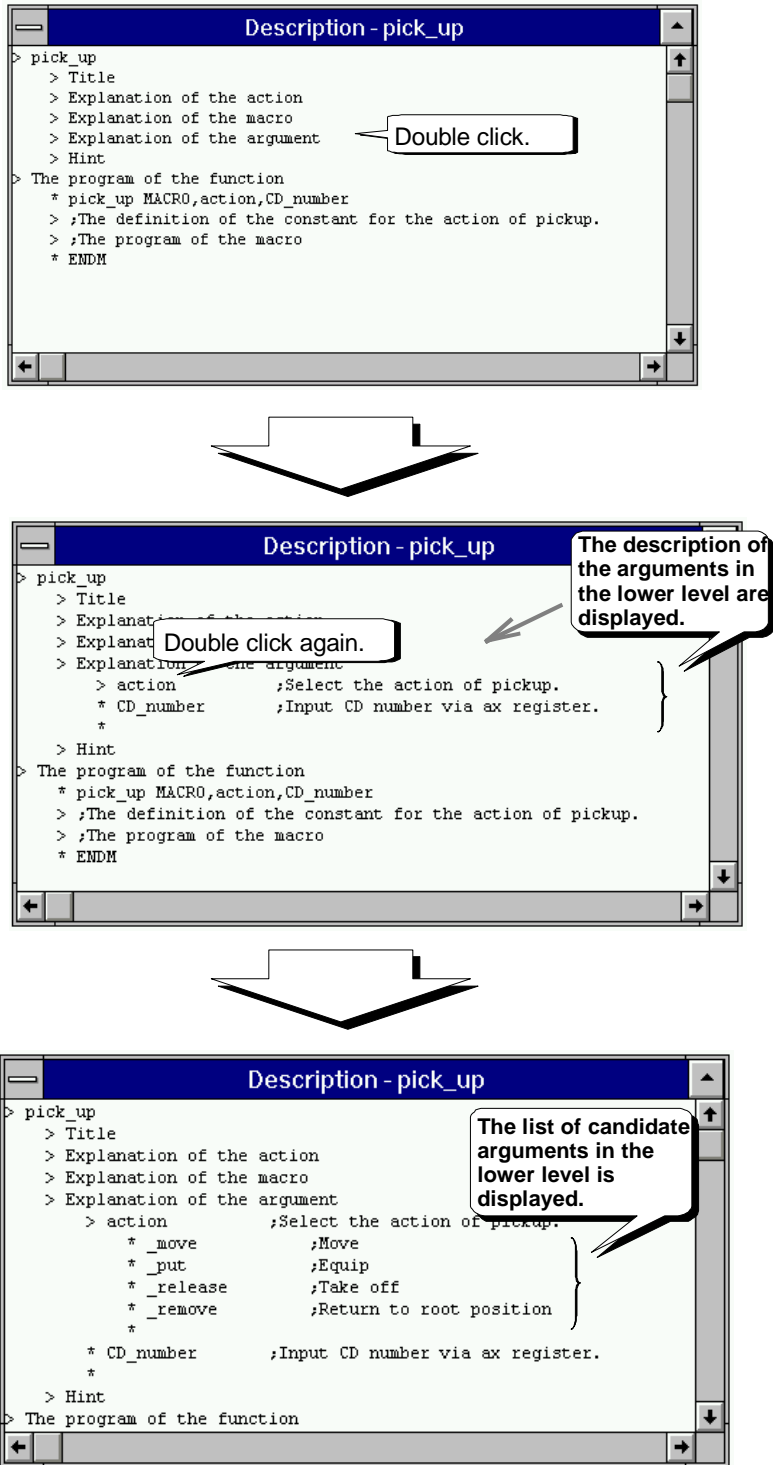


Figure 2-2-26. Checking the Function Contents

The contents can be checked in the Description window. If the target action is a temporary release operation, select "_release" from the drop-down list for the "action" dummy argument. Next, the "CD_number" dummy argument is not in the drop-down list as is the "action" dummy argument and is directly typed into the input text box. Click the "CD_number" dummy argument text box and type in the characters "CD_ID".

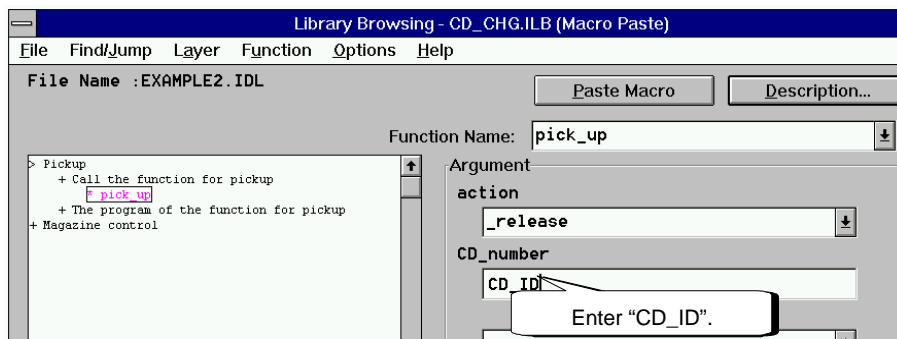
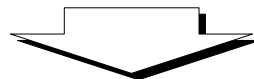
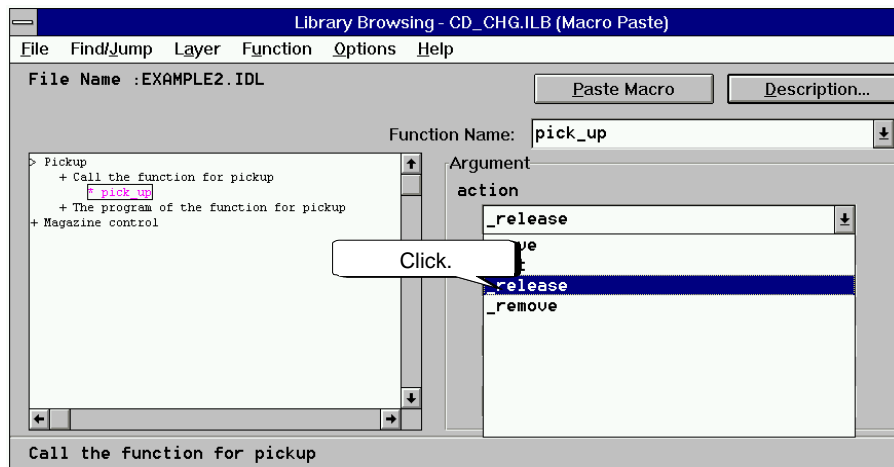
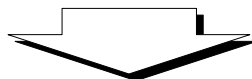
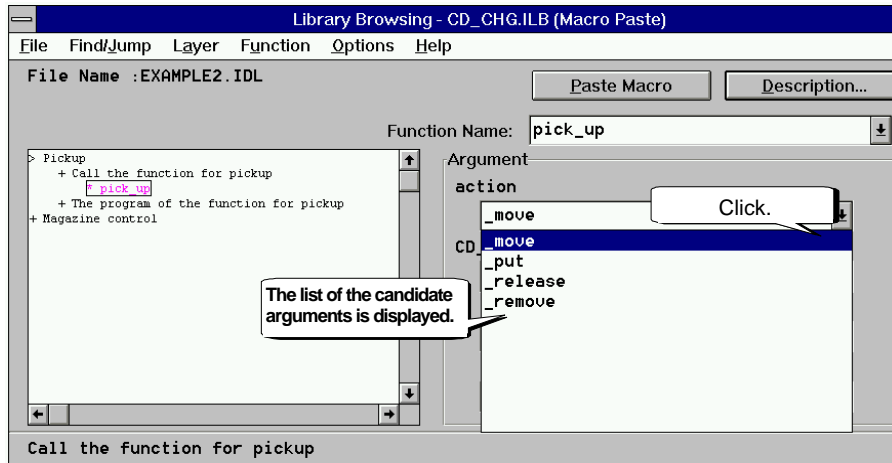


Figure 2-2-27. Setting the Function Arguments

After setting the arguments, this function is pasted in the program. To paste, click the "Paste Macro" button. The "pick_up" function is pasted into the program as in Figure 2-2-28 with the set arguments.

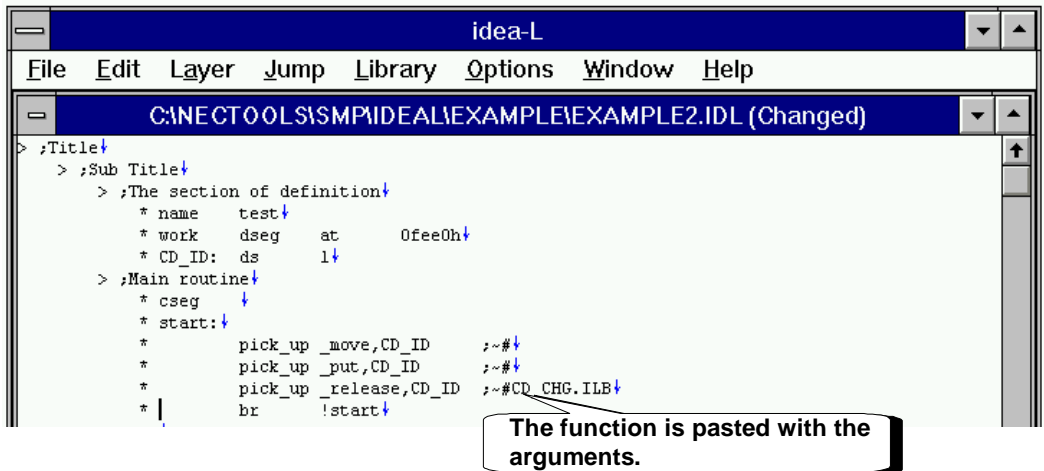
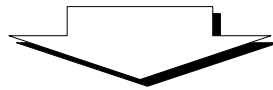
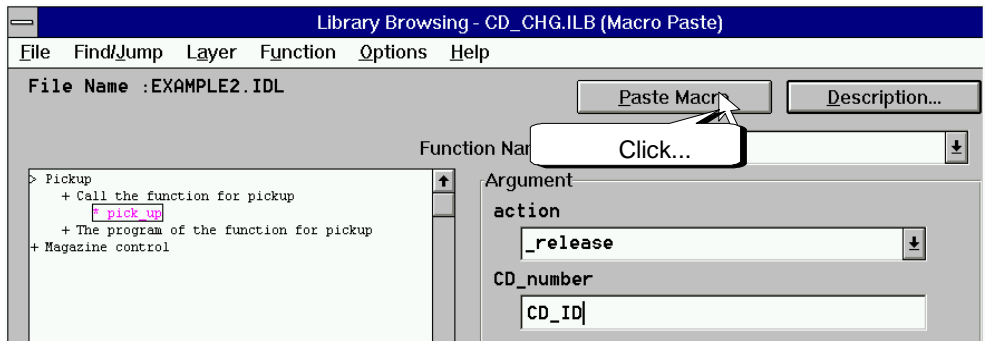


Figure 2-2-28. Pasting a Function

(c) Search for the target function in the library

Sometimes the target use is clear, but the function name is not known.

Whether this kind of function can be rapidly searched for in the library becomes a necessary condition of an easy-to-use library. In *idea-L*, in order to understand the entire library or its details, the library browsing window provided. In this example, we will use the library browsing window to paste the function suited to the objective in the program.

As a pre-condition, the objective of the example is to search for the subroutine of the "pick_up" function in the library file.

As shown in Figure 2-2-29, move the insertion point to a position in the function.

Next, click the "Browsing..." command in the "Library" menu.

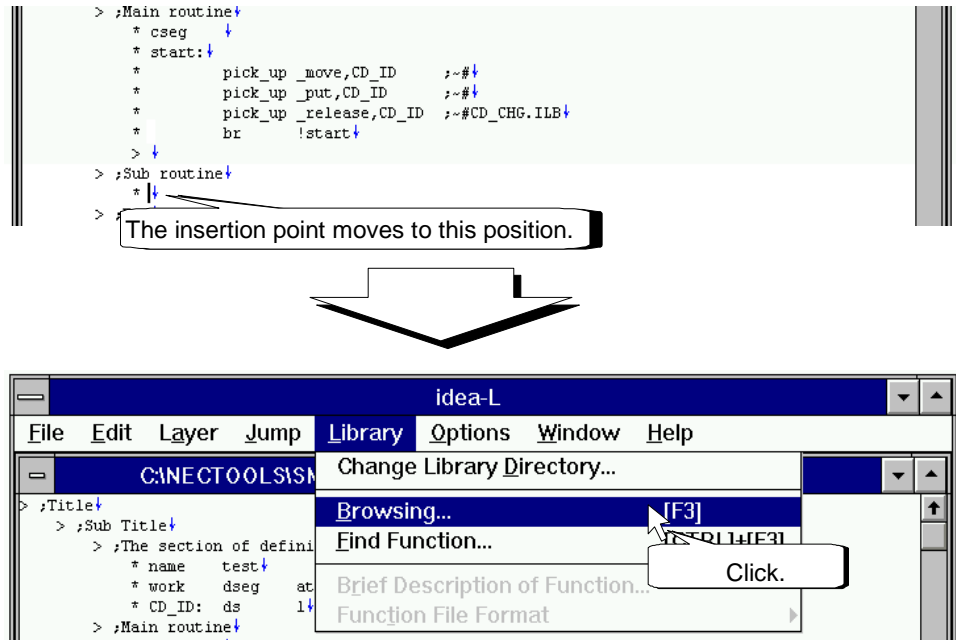


Figure 2-2-29. Specifying a Library File

The library browsing window appears.

As shown in Figure 2-2-30, search for the "sub_pu" function and click. To check the contents of this function, click the "Description..." button.

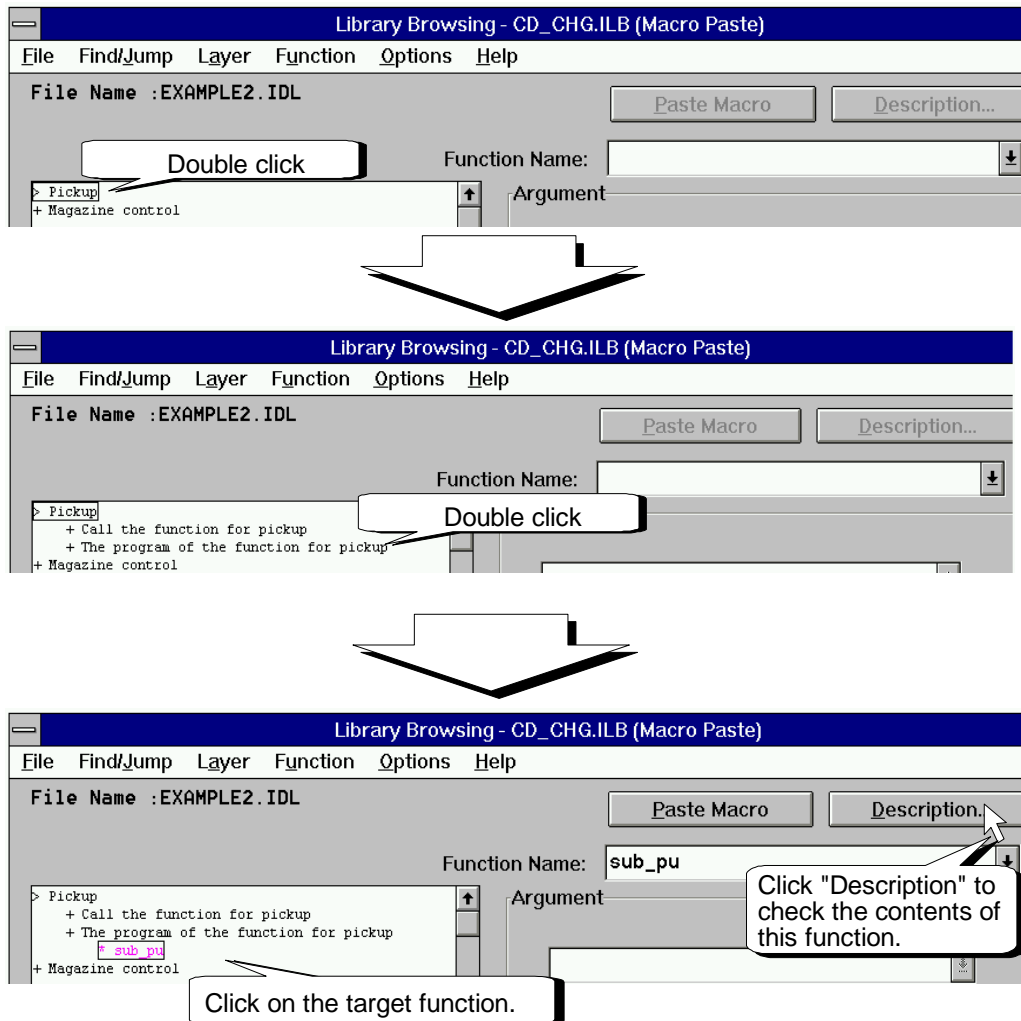


Figure 2-2-30. Searching for the Target Function

The Description window of the function appears. Since the Description window is basically the idea processor, double click a line in the lower level to display the detailed contents.

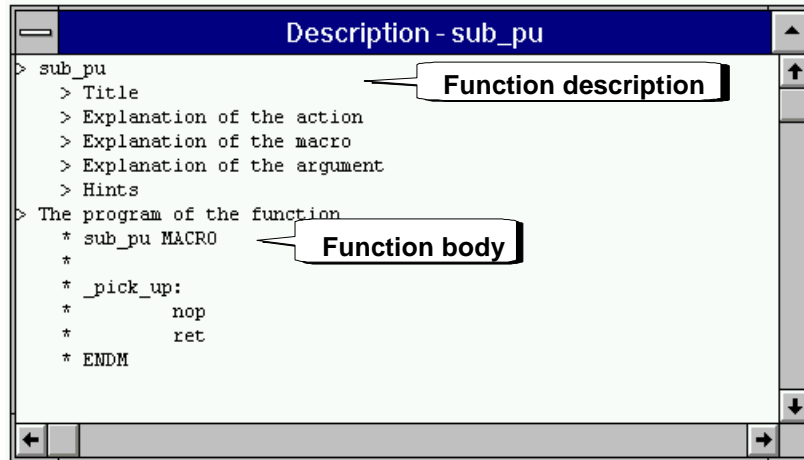


Figure 2-2-31. Function Description

Since this is the target function, it is pasted.

To paste, click the "Paste" button of the library browsing window. The target function was pasted in the program.

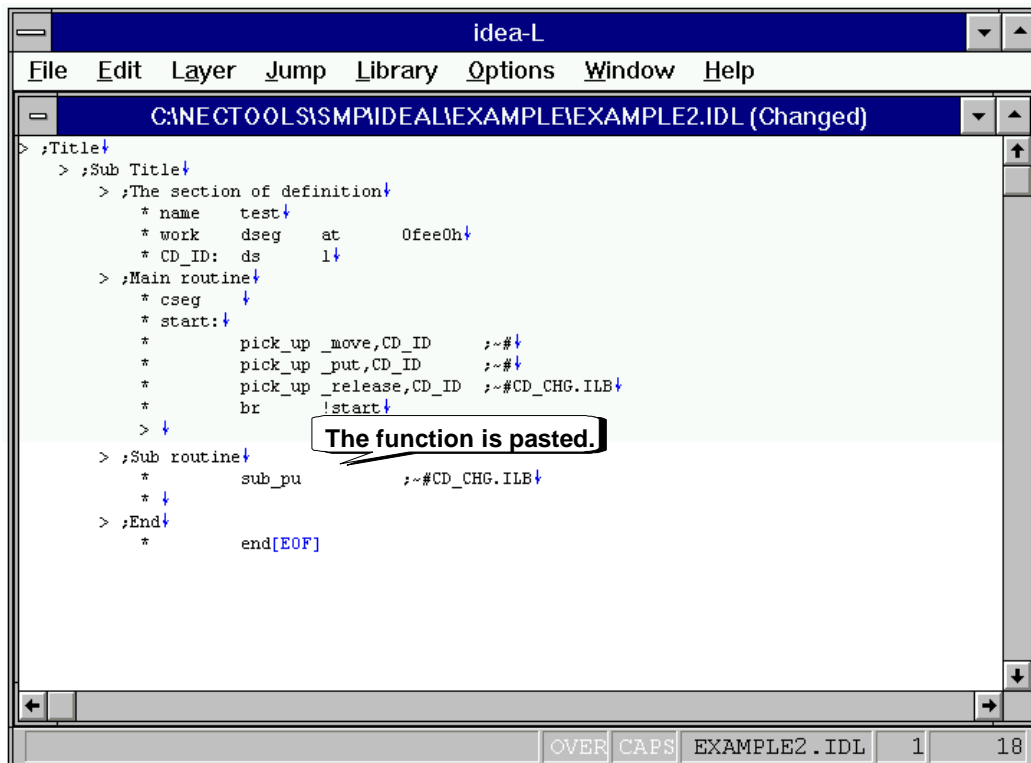


Figure 2-2-32. Pasting a Function

(d) Examine the function data

If a program written by another person will be used in your program, unknown functions are sometimes described in the program. There are no problems if the contents are easy to understand based on the name and arguments of this function and they match your objective. However, converting the arguments to the format in your program is a major problem. This example assumes this scenario.

The last line in the lower level of "Main" becomes the header. Double click this line. The lines in the function in the lower level were displayed. The discussion continues with the lines of this function as the part used from the other program.

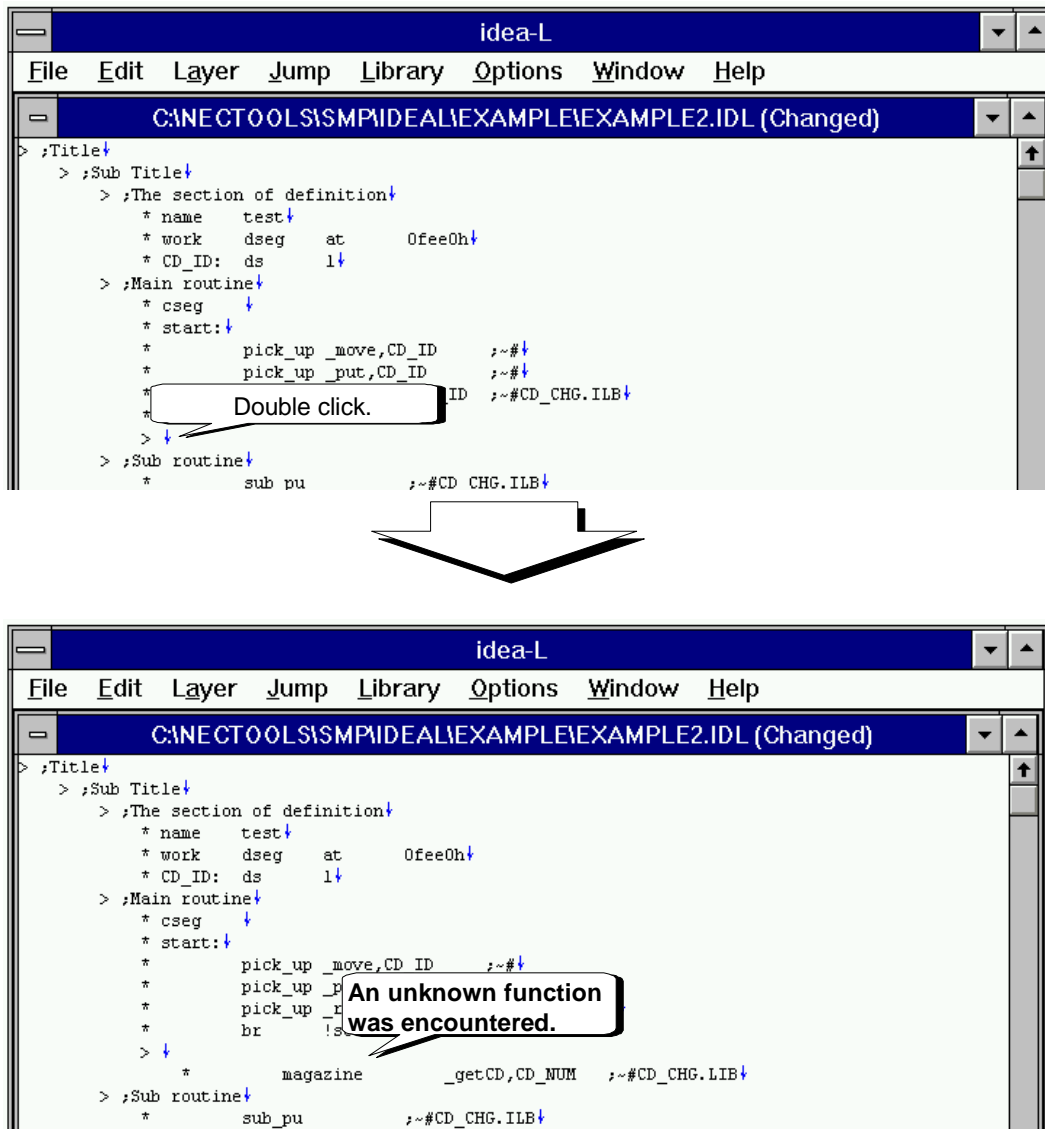


Figure 2-2-33. Encountering an Unknown Function

Now, we will examine the contents of this function. Specify this line as the range and click the "Find Function..." command in the "Library" menu.

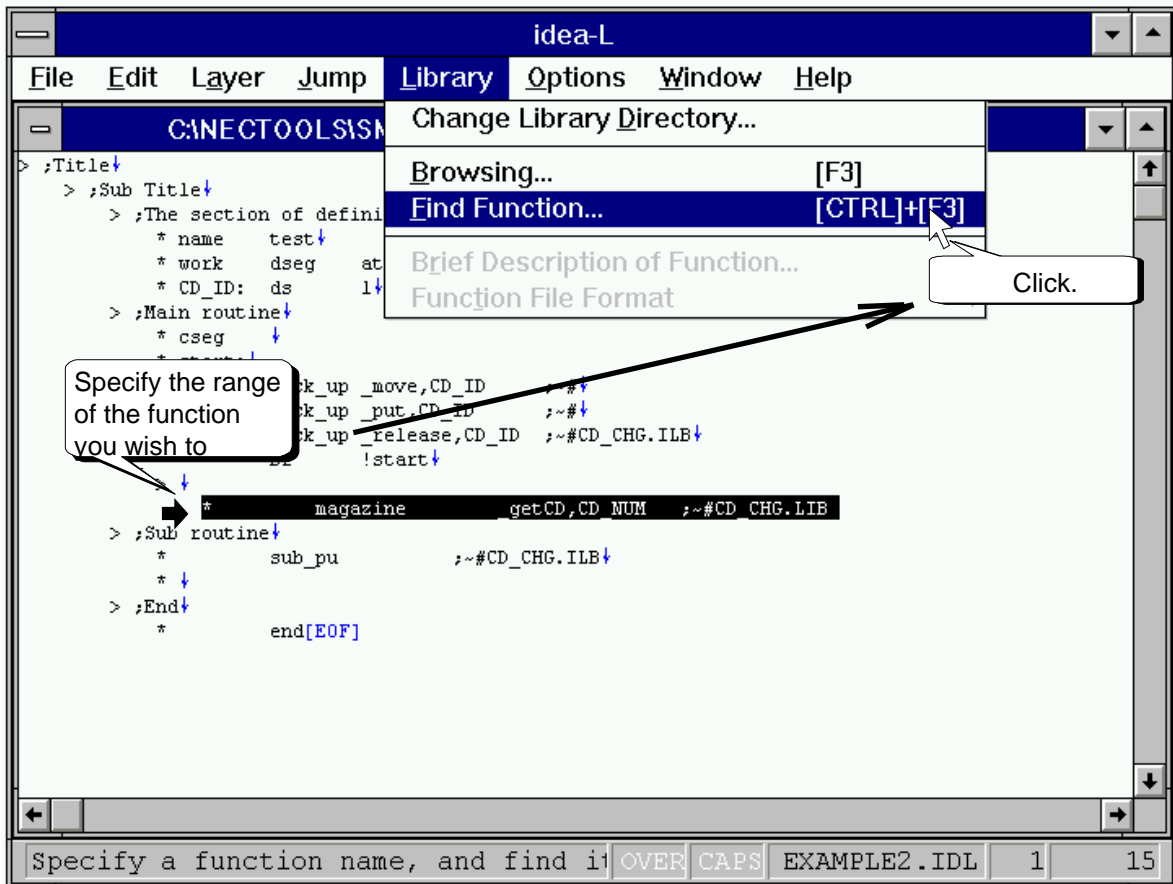


Figure 2-2-34. Examining an Unknown Function

If the function was found, the library browsing window appears. In order to learn about detailed contents, click the "Description..." button. Since this displays the Description window, check the contents of this function.

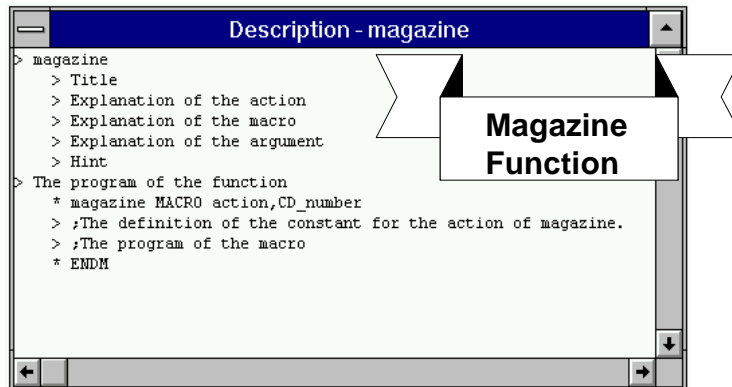
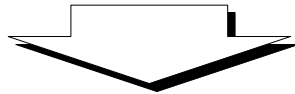
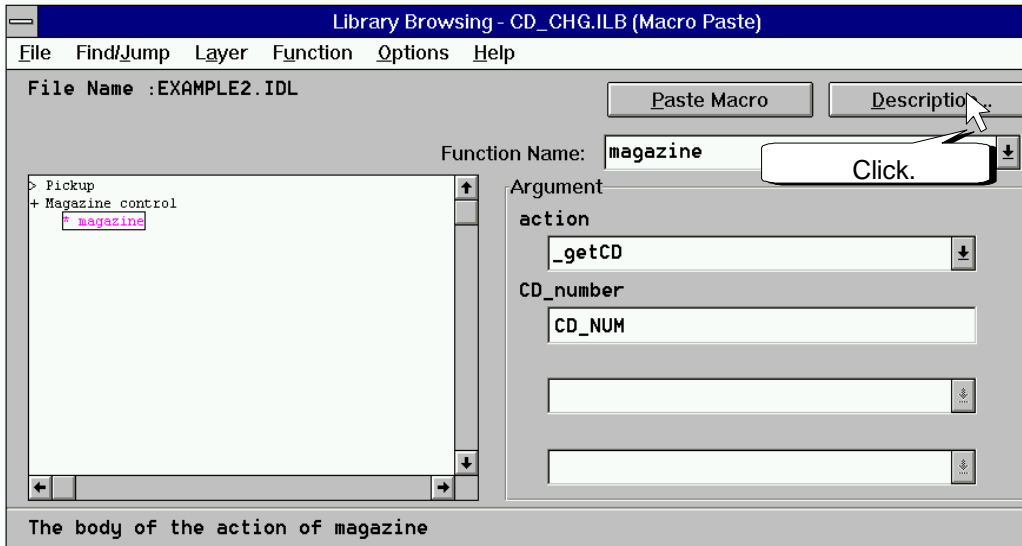
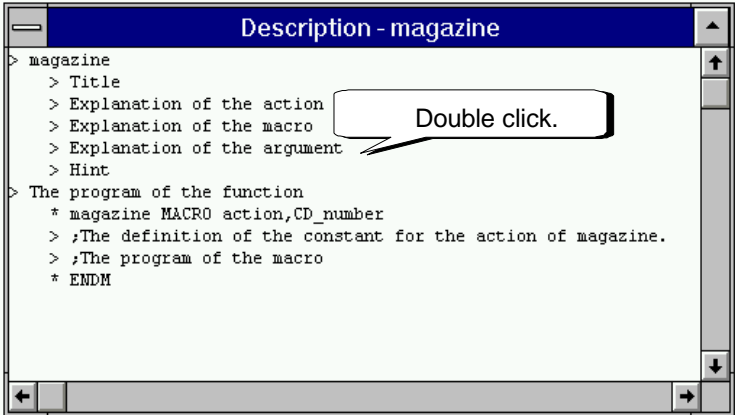


Figure 2-2-35. Learning About a Function

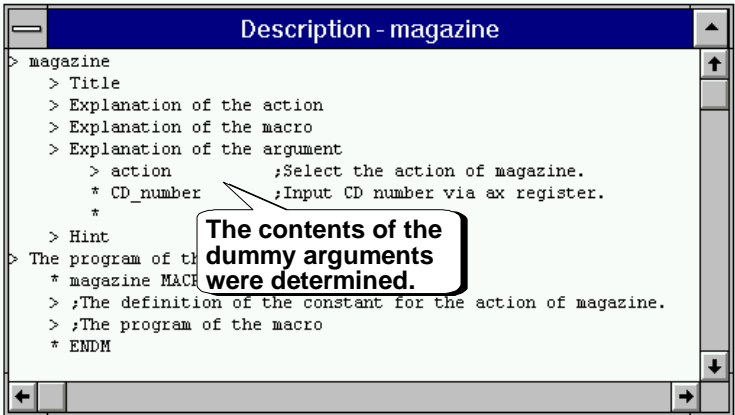
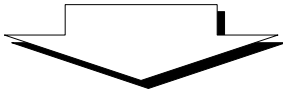
The arguments still have not been checked. So we will check the arguments. Double click the "Explanation of the argument" line to check its contents.



```

> magazine
> Title
> Explanation of the action
> Explanation of the macro
> Explanation of the argument
> Hint
> The program of the function
  * magazine MACRO action,CD_number
  > ;The definition of the constant for the action of magazine.
  > ;The program of the macro
  * ENDM

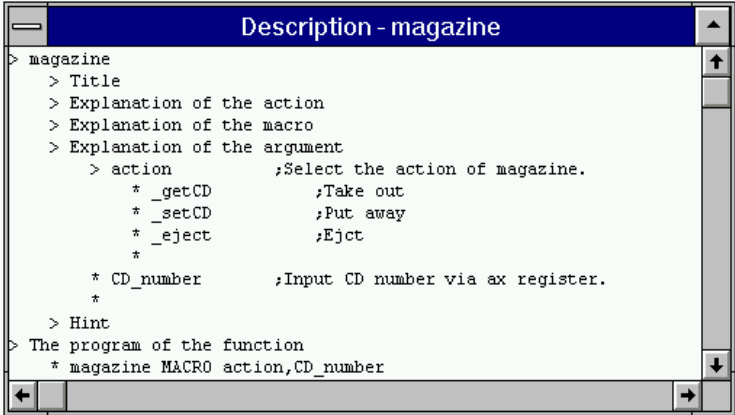
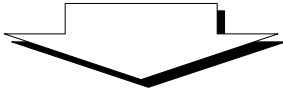
```



```

> magazine
> Title
> Explanation of the action
> Explanation of the macro
> Explanation of the argument
  > action          ;Select the action of magazine.
  * CD_number      ;Input CD number via ax register.
  *
> Hint
> The program of the function
  * magazine MACRO action,CD_number
  > ;The definition of the constant for the action of magazine.
  > ;The program of the macro
  * ENDM

```



```

> magazine
> Title
> Explanation of the action
> Explanation of the macro
> Explanation of the argument
  > action          ;Select the action of magazine.
  * _getCD         ;Take out
  * _setCD         ;Put away
  * _eject         ;Eject
  *
  * CD_number      ;Input CD number via ax register.
  *
> Hint
> The program of the function
  * magazine MACRO action,CD_number

```

Figure 2-2-36. Examining a Function

If the contents of the "CD_number" dummy argument is changed from "CD_NUM" to "CD_ID" in the argument window, we will assume it can be used in the program.

Change the contents of the "CD_number" dummy argument.

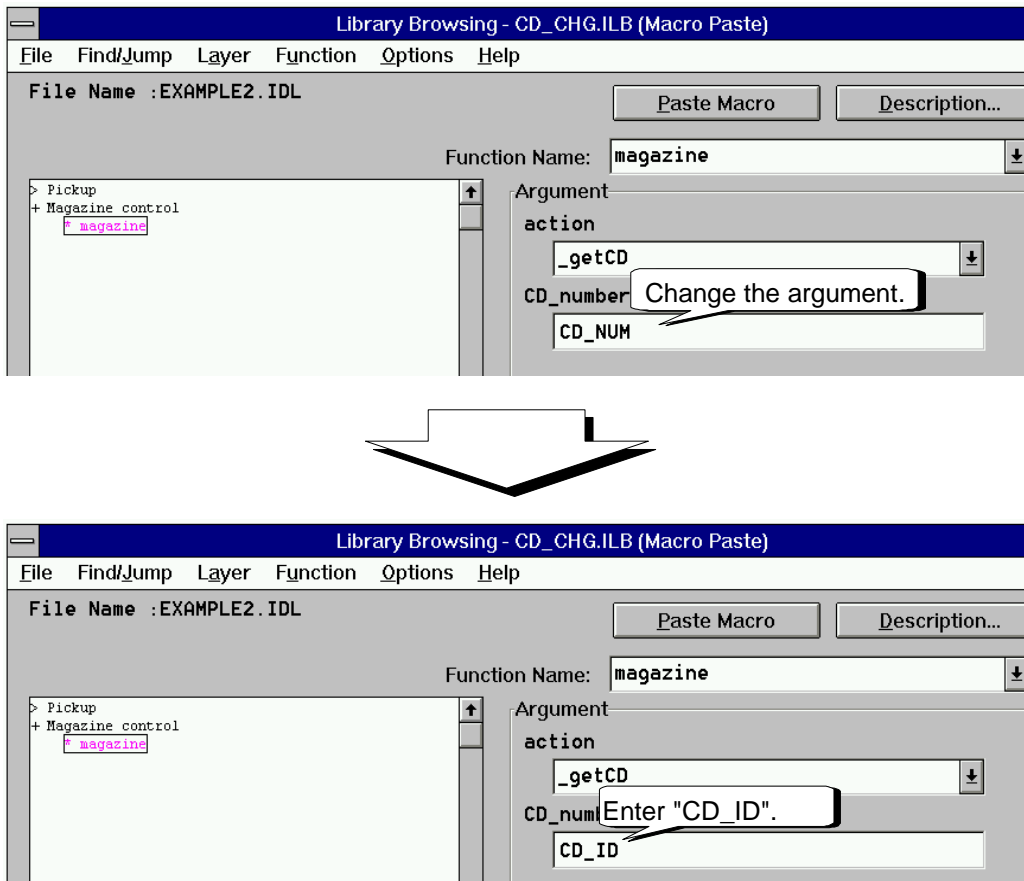


Figure 2-2-37. Change the Argument

If the change ends, finally click the "Paste Macro" button. This function is pasted in the program.

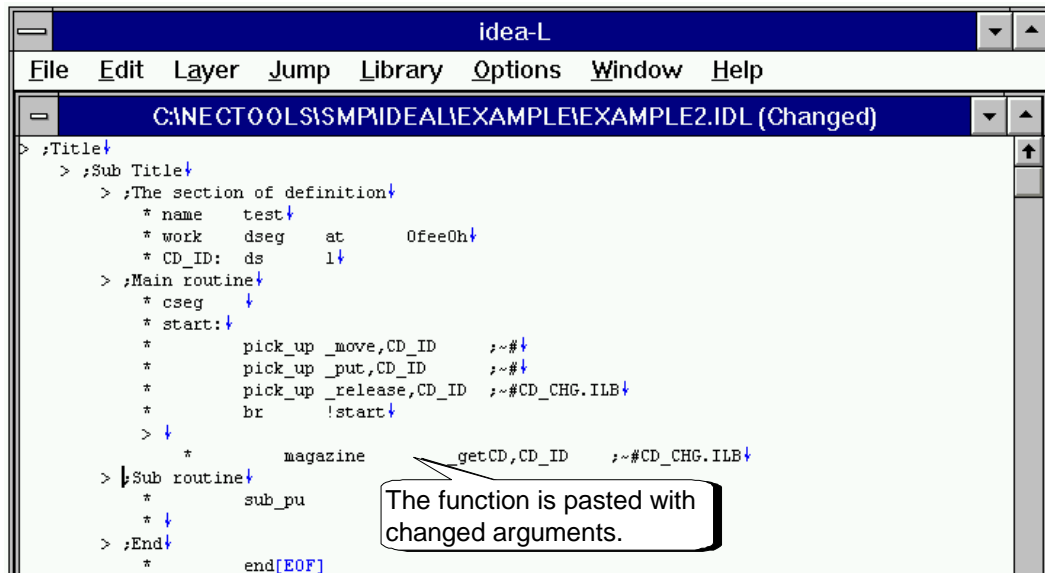
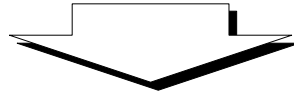
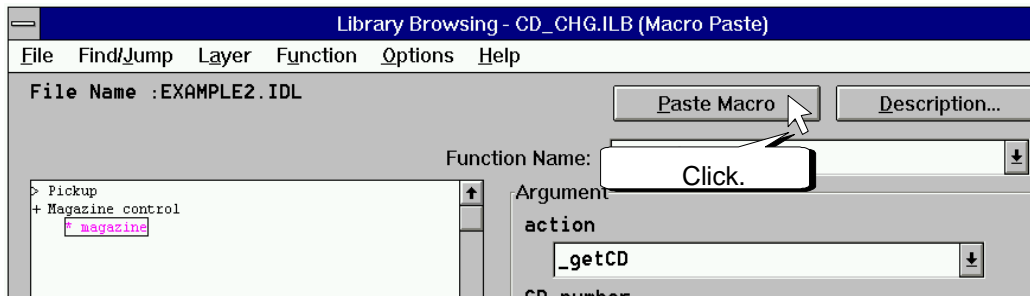


Figure 2-2-38. Paste the Function with Changed Arguments

2.3 Creating a Source File

A written program is given to the compiler and assembler to create the source file.

Now we will create a source file.

The "EXAMPLE2.IDL" window is opened on the current screen.

First, click the "Save Source File..." command in the "File" menu.

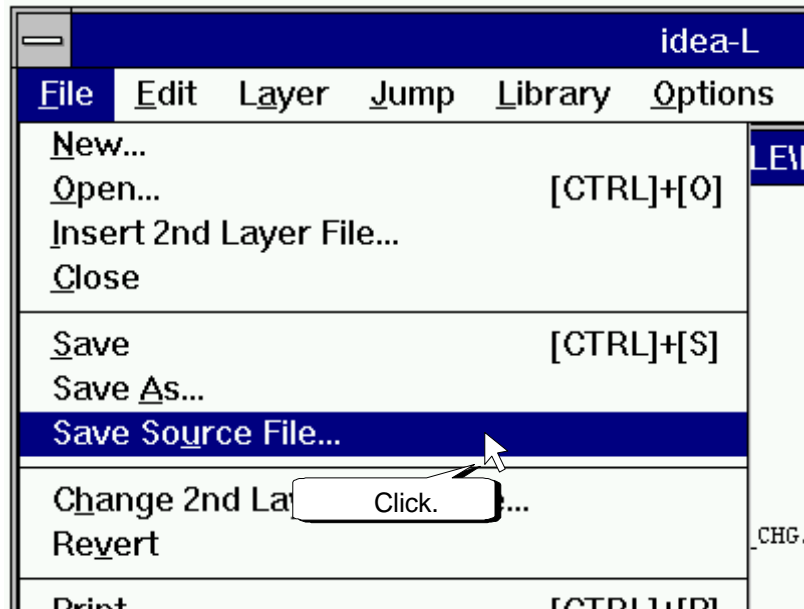


Figure 2-3-1. Creating a Source File

The "Save Source File" dialog box appears.

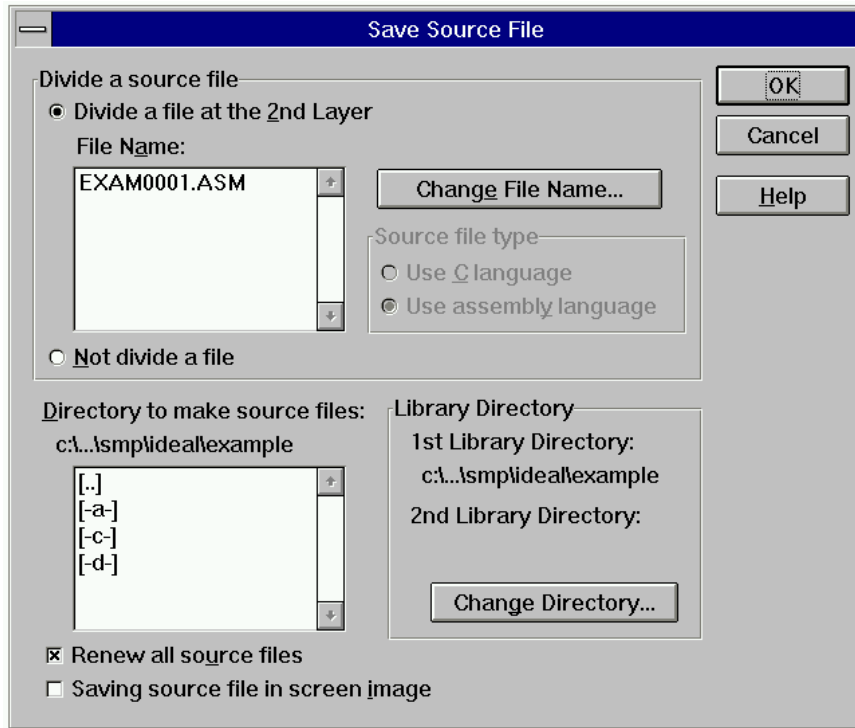


Figure 2-3-2. Save Source File Dialog Box

Click the "OK" button to create the source file. The following two files will be created.

- Source program file [EXAM0001.ASM]
Source program
- Personal library file [EXAM0001.PLB]
This file collects the actual macros (functions) used in the source program. It is included in the source file.

Figure 2-3-3 shows the contents of the two files that are created.

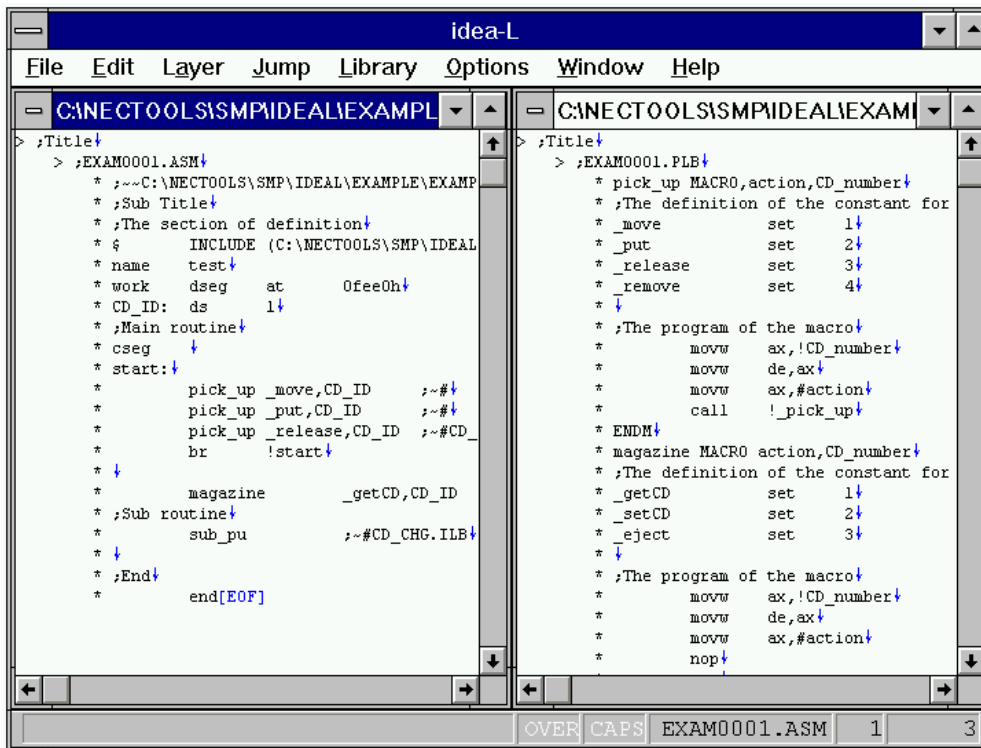


Figure 2-3-3. Created File Group

Finally, click the "Close" command in the "File" menu to exit "EXAMPLE2.IDL."

To Change the Name of a Source Program File

idea-L automatically sets the source program file name in the "Save Source File" command. If you wish to create the file with any file name, perform the following procedure.

Click the file name you wish to change in the source "File Name" list box in the "Save Source File" dialog box. After selecting either the "Use C language" or the "Use assembly language" radio button for the "Source file type," click the "Change File Name" button. This displays the "Change Source File Name" dialog box so type in the new file name in the source "File Name" text box. When finished making the input, click the "OK" button. This changes the file name of the source program.

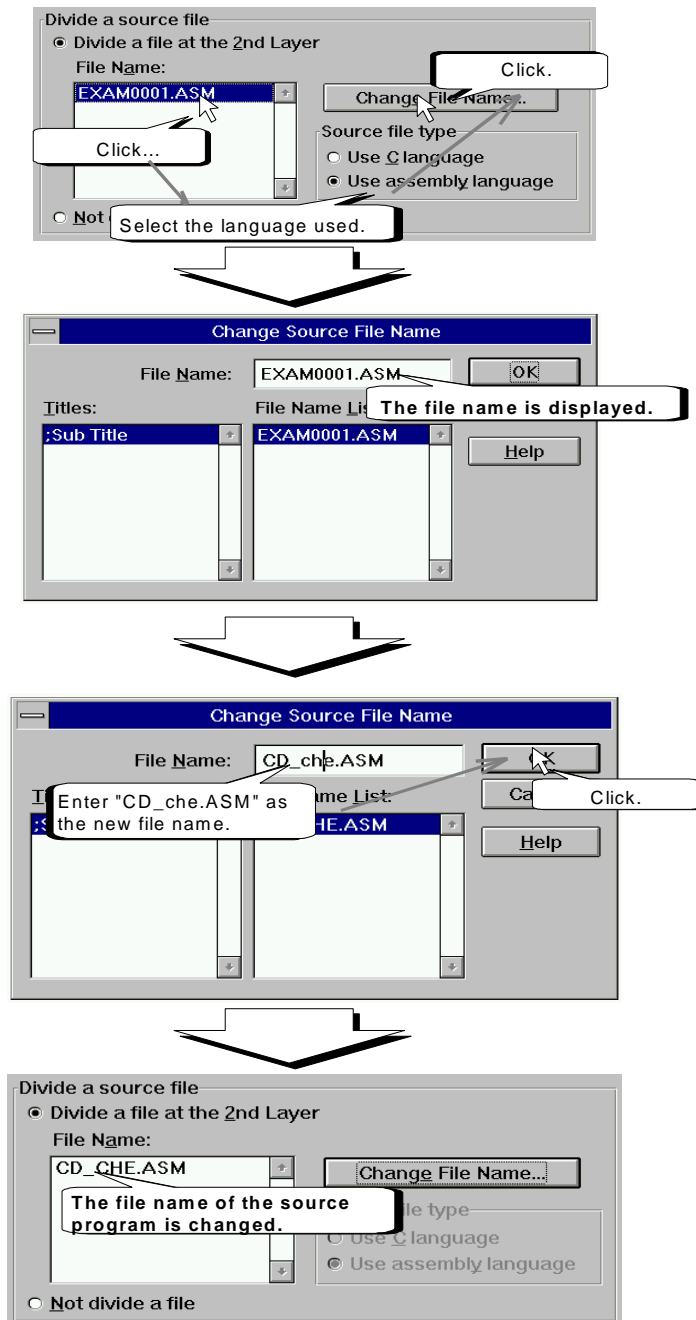


Figure 2-3-4. Changing the File Name of a Source Program

Errors During Compiling or Assembly

If a compiling or an assembly error occurs, do the following.

Use the *idea-L* tag jump function to jump to the error line and correct the error.

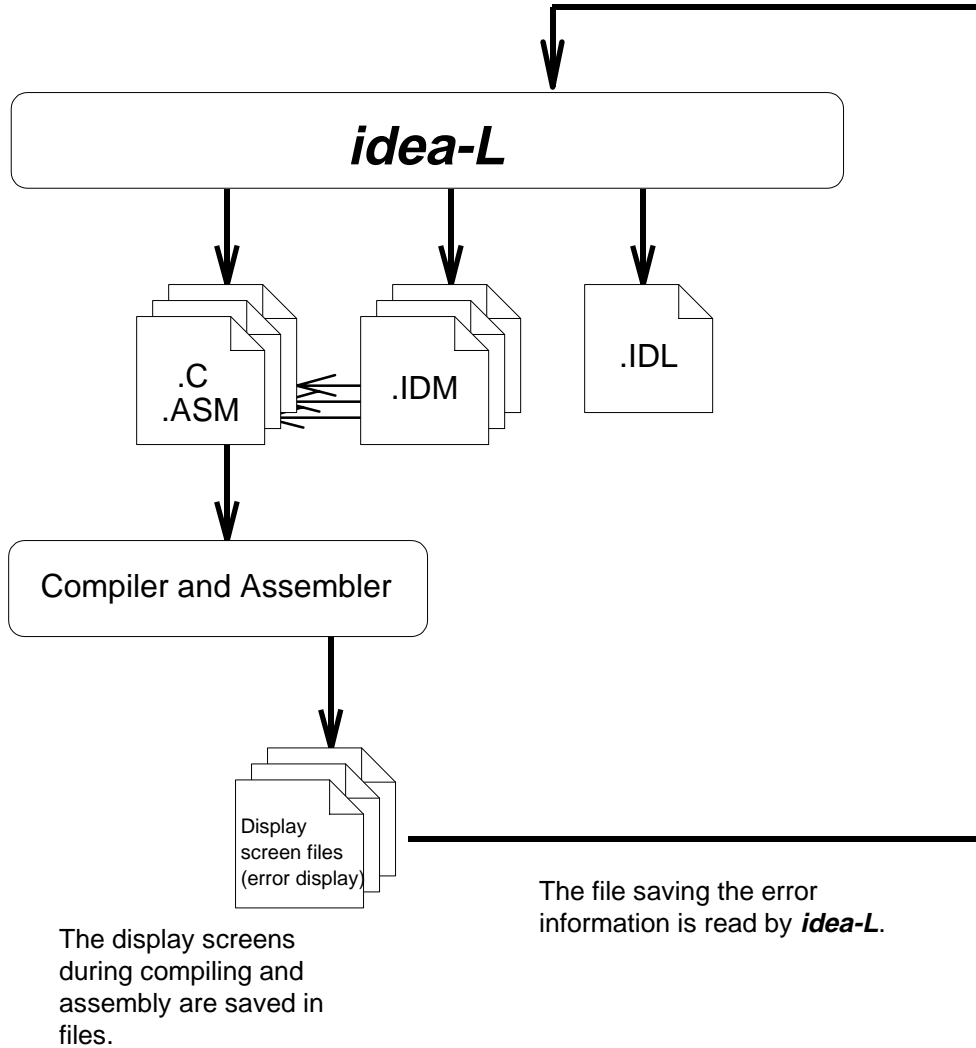


Figure 2-3-5. Process Flow for Error Correction

2.4 Using Old Programs

idea-L can read source files in a conventional text format. Now, we will open and abstract a text source program.

First, click the "Open..." command in the "File" menu. The file "Open" dialog box appears, so click the "List Files of Type" list box and select "Text (*.c, *.h, *.asm, *.txt)."

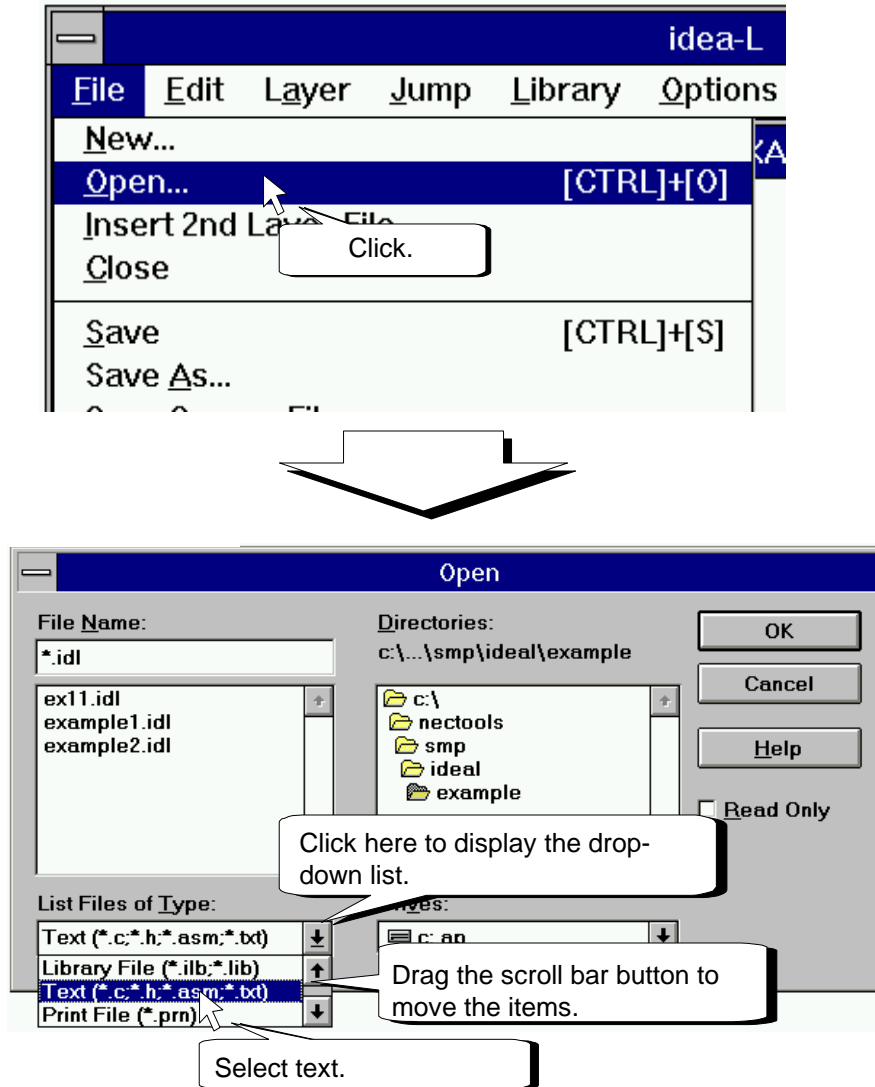


Figure 2-4-1. Selecting a Text File

Double click the file name "example3.asm."

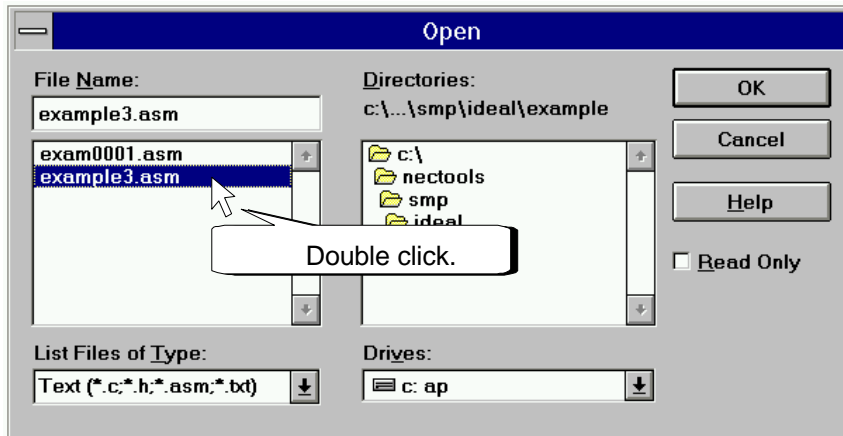


Figure 2-4-2. Open a Conventional Source Program

Open the "example3.asm" window. example3.asm is a text source program. The file name becomes the header, and the entire program is its lower level.

If the comment lines in the program are examined, their structure is "a comment line is added in the line before the program."

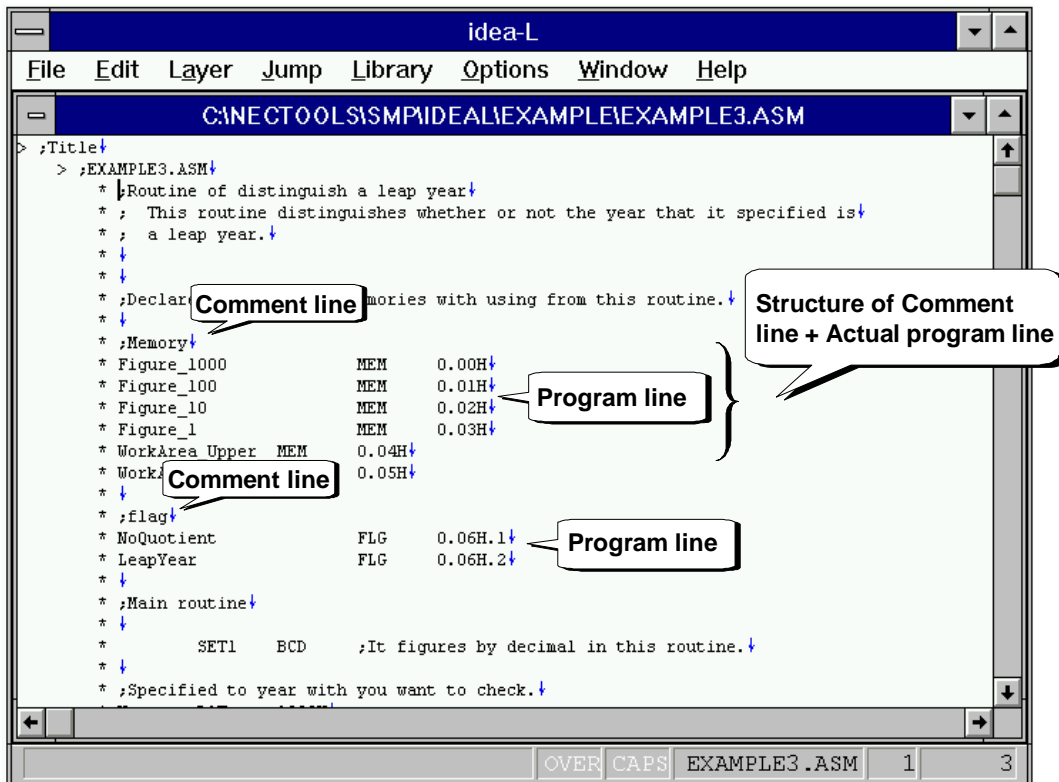


Figure 2-4-3. Conventional Source Program

We will use the *idea-L* function to form a hierarchy.

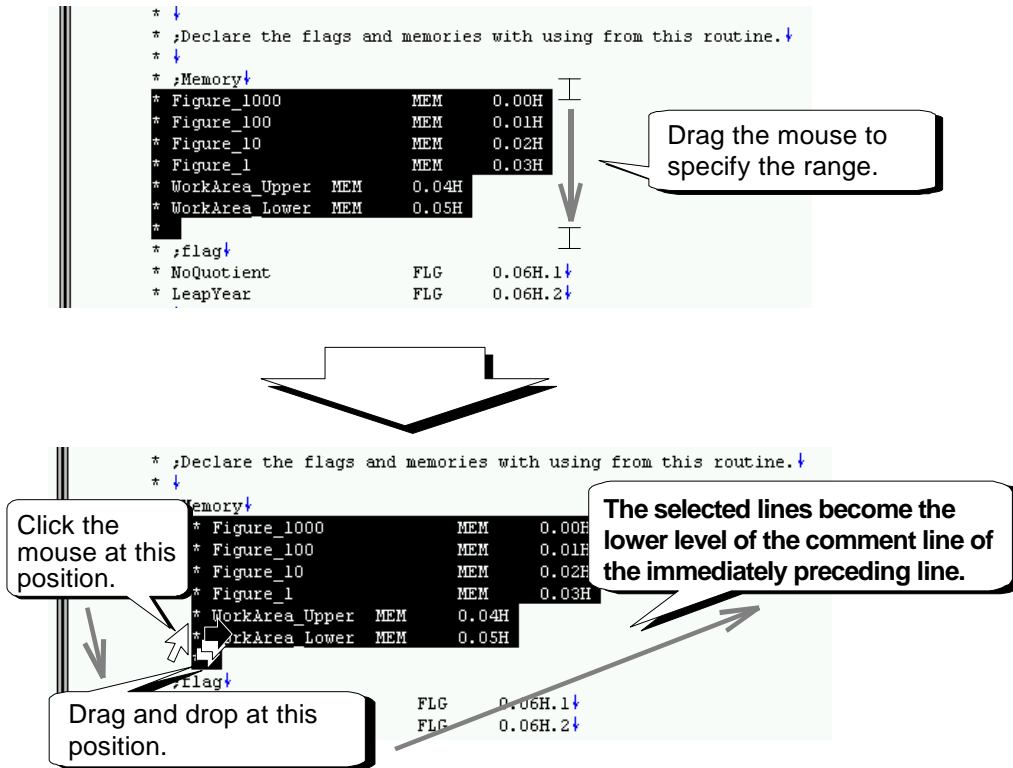


Figure 2-4-4. Make a Hierarchy of the Program Lines

Form hierarchies of all of the lines in the same way.

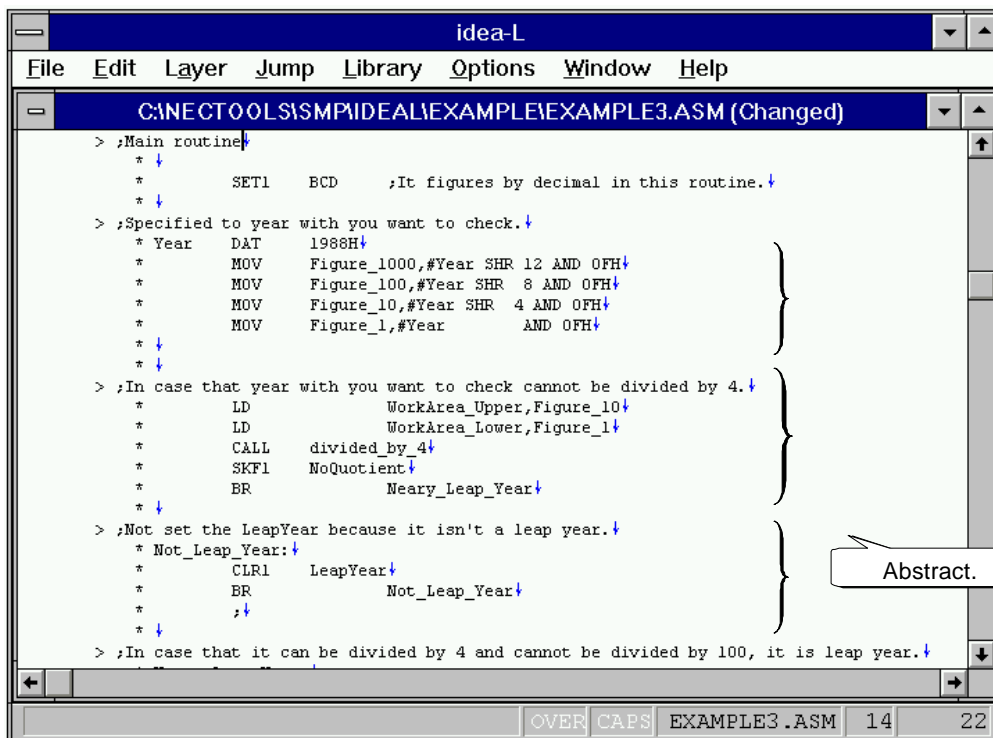


Figure 2-4-5. Completed Hierarchy of the Program Lines

Double click the header line to hide the lower level.

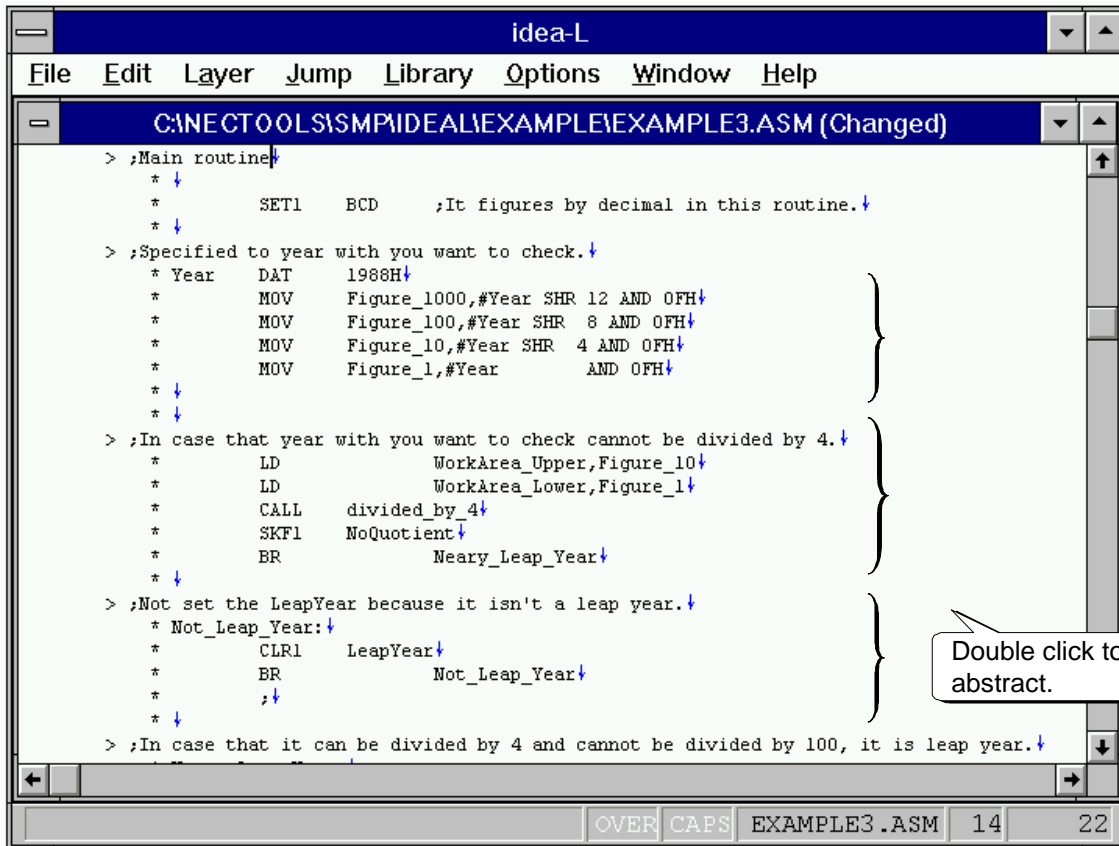


Figure 2-4-6. Abstract the Source Program

3. RESTRICTIONS

Be careful when using because the following restrictions apply in *idea-L*.

- Multiple *idea-L*s cannot be executed simultaneously.
 - If multiple *idea-L*s were started, the *idea-L* currently being started becomes active.
- The same IDL file and text file cannot be used simultaneously.
 - The file opened last is opened as read only and a message appears.
- A text file containing control codes other than new lines (0DH, 0AH) and tab (09H) cannot be opened.
- Restrictions of each editing window
 - Restrictions on the number of lines
 - Each editing window can have up to 129,024 lines.
 - Restrictions on the number of characters in one line
 - One line has up to 255 characters.
 - Restrictions on the levels in the hierarchy
 - The argument window can be created with up to three levels.
 - The library editing window can be created with up to five levels.
 - Otherwise, up to 127 levels can be created.
 - Restrictions on the number of types of functions
 - Up to 1,000 types of functions can be registered in one library file.
 - Restrictions on the number of arguments
 - Up to 32 dummy arguments can be set.
 - Restrictions on the number of argument candidates
 - Up to 32 (lines of) argument candidates for one dummy argument can be set.
 - Restrictions on function names and names of the argument candidates
 - They have a maximum of 32 characters.
- In "Save As" and "Copy the *idea-L* file," file names including the dollar sign (\$) or tilde (~) in the extension cannot be used.
 - In the directories for saving files and creating assembler files, do not use file names in which the extension combines the dollar sign (\$) and the tilde (~). *idea-L* creates temporary files when saving a file or creating an assemble file. However, since the extension of a temporary file combines the dollar sign and the tilde, the file names may clash, and a file may be overwritten or erased.

Examples) abc.\$~\$, abc.~\$~, abc.\$\$~...
- The "Options," "Environment Settings," and "Window" menus can be set to display underlines in the insertion point line. However, the underline disappears when the window switches.
- Since the file specification format of an INCLUDE statement created by *idea-L* is in parentheses "()" when using a macro library, a macro library in the *idea-L* format cannot be used in 17K, 75X, and V800 systems.

[MEMO]

Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel.

FAX

Address

Thank you for your kind support.

North America

NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Hong Kong, Philippines, Oceania

NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Asian Nations except Philippines

NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Europe

NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

Korea

NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Japan

NEC Semiconductor Technical Hotline
Fax: 044-435-9608

South America

NEC do Brasil S.A.
Fax: +55-11-6462-6829

Taiwan

NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>