BNP-B2197* (ENG)

MITSUBISHI CNC

MELDAS MAGIC 64

CUSTOM APPLICATION INTERFACE LIBRARY MANUAL (PROGRAMMING SECTION)

ADVANCED AND EVER ADVANCING
MITSUBISHI ELECTRIC

MELDASMAGIC is a registered trademark of Mitsubishi Electric Corporation.
Microsoft and Windows are registered trademarks of Microsoft Corporation

Other company and product names that appear in this manual are trademarks or registered trademarks of the respective company.

Introduction

This instruction manual describes the Custom Application Interface Library (Custom API Library) used for developing the custom application of MELDASMAGIC Series.
This instruction manual describes the programming and usage of the Custom API Function/variables used when developing custom applications.
Please read this manual and the following manuals before programming.

MELDASMAGIC 64	Custom Application Interface Library Guide (Function Section)	BNP-B2198
MELDASMAGIC 64	Custom Application Interface Library Guide (Variable Section)	BNP-B2199

Please read the following "Precautions for Safety" to ensure safe use of MELDASMAGIC Series.

Precautions for Safety

Always read the specifications issued by the machine maker, this manual, related manuals and enclosed documents before starting installation, operation, programming, maintenance or inspections to ensure correct use. Thoroughly understand the basics, safety information and precautions of this numerical controller before using the unit.
The safety precautions are ranked as "DANGER", "WARNING" and "CAUTION" in this manual.

⚠ **DANGER**	When there is a great risk that the user could be subject to fatalities or serious injuries if handling is mistaken.
⚠ **WARNING**	When the user could be subject to fatalities or serious injuries if handling is mistaken.
⚠ **CAUTION**	When the user could be subject to injuries or when physical damage could occur if handling is mistaken.

Note that even if the item is ranked as " ⚠ CAUTION", incorrect handling could lead to serious results. Important information is described in all cases, so please observe the items.

⚠ **DANGER**
Not applicable in this manual.

⚠ **WARNING**
Not applicable in this manual.

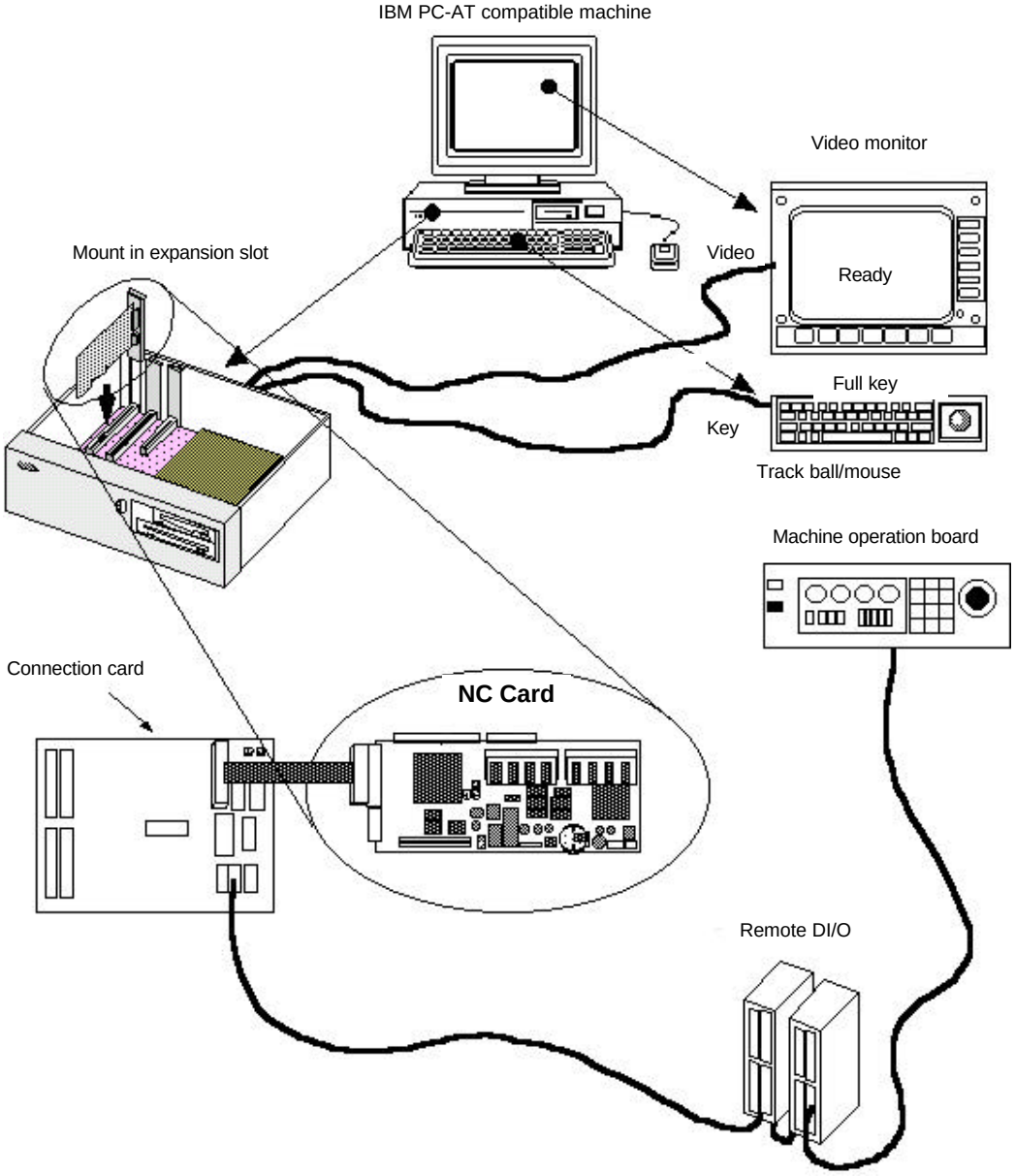
⚠ **CAUTION**
Items related to product and manual ⚠ For items described as "Restrictions" or "Usable State" in this manual, the instruction manual issued by the machine maker takes precedence over this manual. ⚠ Items not described in this manual must be interpreted as "not possible". ⚠ This manual is written on the assumption that all option functions are added. Refer to the specifications issued by the machine maker before starting use. ⚠ Some screens and functions may differ or may not be usable depending on the NC system version.

Contents

1. Outline

1.1 System configuration

1.1.1 General configuration



The applications (custom applications) for MELDASMAGIC 64 Series are developed on the IBM PC-AT compatible personal computer in which an NC Card has been mounted. All operations from programming to debugging are done on the personal computer.

1.1.2 Outline of custom release system

The following systems have been prepared for the custom release system of MELDASMAGIC 64 Series.

(1) Custom API (Application Interface) Library

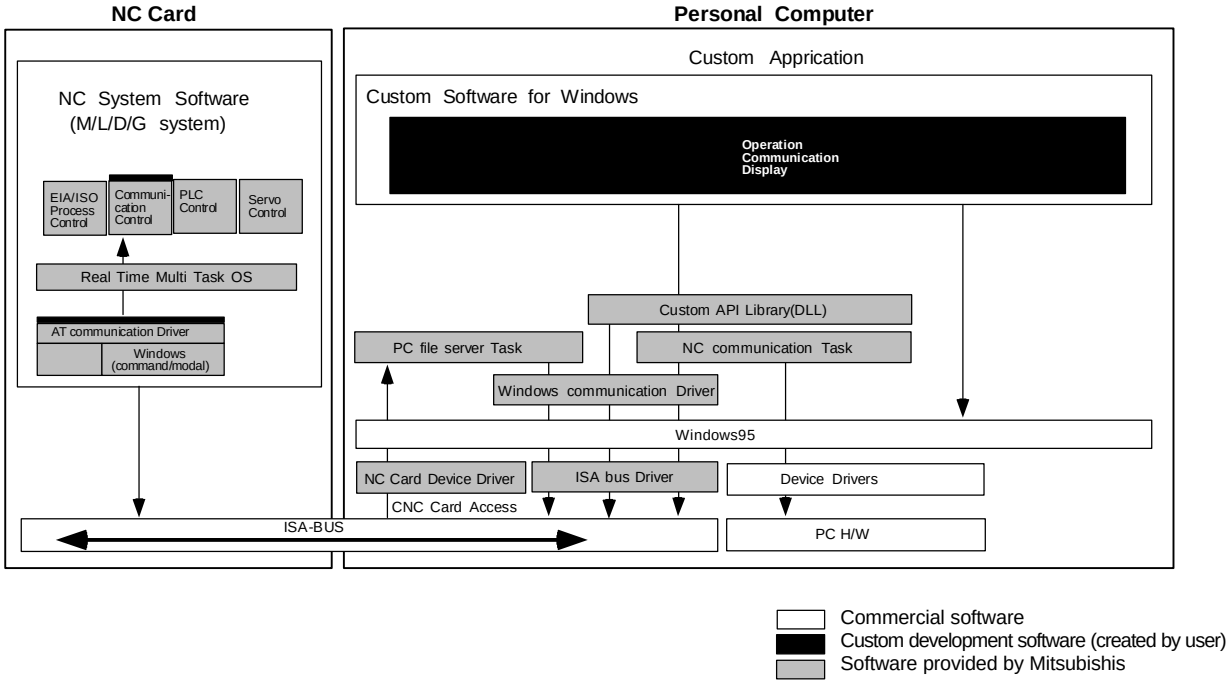
This is a library where in the functions used to access the data and files in the NC Card from the custom application are found.
The Custom API Library has several tens of functions including those listed below.

Reading of NC axis coordinate value
Reading/writing of parameters in NC Card
Getting of file information in NC Card
Transferring of files between NC Card and personal computer
Getting of NC alarm information, etc.

The Windows 95 DLL (Dynamic Link Library) format is used for the runtime library in the Custom API Library. Thus, there are no limits to the custom application development language as long as the programming language supports DLL. The user can freely select a programming language that fits the characteristics of the custom application to be developed.

1.2 Software configuration

1.2.1 Software configuration diagram



1.2.2 Software related to custom release

(1) Custom application

This is an application developed by the user. The MELDASMAGIC 64 can be accessed by calling the Custom API Library, and screen display processing can be done using the Windows function, etc.

(2) Custom API Run-Time Library

This is a group of functions used to release the NC Card internal data or access files to support the custom application.

(3) NC Card Communication Task

This is a task used to communicate with the NC system in the NC Card. This task is driven by the messages from the Custom API Function.

(4) PC File Server Task

This is a task used to supply personal computer file data to the NC system in the NC Card. The task is driven by messages from the NC system.
This task is used in the personal computer direct operation function (option function).

(5) Windows command driver

This is the driver on the personal computer side that communicates with the NC Card connected with an ISA bus.

(6) ISA Bus Driver

This is the device driver on the personal computer side used to access the device connected with an ISA bus.

(7) NC Card Driver

This is the device driver used to process various NC Card commands and messages from the NC Card.

(8) PC-AT driver

This is a driver on the NC side that communicates with the personal computer connected with an ISA bus.

(9) Command task

This is a task used to access the NC internal data and files and communicate with the personal computer.

2. Setup of Programming Environment

2.1 Before starting setup

Before starting the setup of the custom application programming environment, confirm the details of the hardware, software and Custom API Library package being used.

(1) Confirmation of required system configuration

To develop the custom application, the following hardware and software are required besides the Custom API Library.

- IBM PC/AT or compatible personal computer that satisfies the following:
  CPU that is i486SX 33MHz or more
  ISA bus
- Hard disk with 20Mbytes or more of free space
- One or more floppy disk drive compatible with the 3.5-inch, 1.44Mbyte format
- CRT compatible with Microsoft Windows 95, and having a resolution of 640 × 480 dots or more and 256 colors or more
- Memory that is 8Mbytes or larger (16MB or more is recommended)
- Mouse or compatible pointing device
- Microsoft Windows 95

When developing with Visual Basic

- Microsoft Visual Basic Professional Edition 4.0 32-bit version (Japanese version)

When developing with other programming languages

- Programming language that can be developed with Windows application, and which satisfies the following:
  Can develop Windows 32-bit application.
  Supports Windows DLL, and has a function to call DLL.

(2) Contents of Custom API Library

Before setting up the Custom API Library, confirm the contents of the Custom API Library.

- "Setup Instruction Manual (BNP-B2191)
  MELDASMAGIC MMI Operation Manual (D/M) (BNP-B2193)
  MELDASMAGIC MMI Operation Manual (L/G) (BNP-B2194)
  "Custom Application Interface Library Guide (Programming Section) (BNP-B2197)" (This manual)
- "Custom Application Interface Library Guide (Function Section) (BNP-B2198)"
- "Custom Application Interface Library Guide (Variable Section) (BNP-B2199)"
- Floppy disk
  - Custom API Library SDK1
  - Custom API Library SDK2

  The floppy disk contains the README_E.TXT (English)/README_J.TXT (Japanese). The README_E.TXT (English)/README_J.TXT (Japanese) file describes the new information not described in the manual. Confirm the contents before setting up.

2.2 Setting up of Custom API Library

(1) Installation of Custom API Run-Time Library

Install the Custom API Run-Time Library before starting development of an application using the Custom API Library. When executing the operations up to debugging with the personal computer used to develop the application, the NC Card must also be installed.
The method for installing the Custom API Run-Time Library and NC Card are described in the "MELDASMAGIC 64 Setup Instruction Manual (BNP-B2191)".

(2) Installation of Custom API Library

The method for installing the Custom API Library is described in the README_E.TXT/ README_J.TXT file. Read README_E.TXT/README_J.TXT, and install the Custom API Library.

2.3 Before using

The Custom API Library runs on Windows, and is a support kit used to develop the application to operate MELDASMAGIC 64 Series. Thus, when using the Custom API Library, the user must basically understand Windows and the programming language used for the custom application development. These are not explained in this manual. For details on Windows and the programming language, refer to each manual enclosed with the respective software.

3. Programming with Visual Basic

3.1 Before starting programming

(1) Custom API Library files to be used

The following files are used to program the custom application with Visual Basic.

NCMCAPI.BAS
MELNCAPI.BAS
MELTYPE.BAS
MELERR.BAS
MELSBERR.BAS

These files are located in the SDK32\INCLUDE\VB\ directory under the directory where the Custom API Library was installed.

1. NCMCAPI.BAS
   This is the code module that describes the declarations used to call the Custom API Functions prepared in the Custom API Library as procedures for Visual Basic.
2. MELNCAPI.BAS
   This is the code module that defines the constants used with the Custom API Functions and the procedures used for creating the function arguments.
3. MELTYPE.BAS
   This is the code module that defines the data type constants used with the Custom API Functions and defines the user defined type arrays.
4. MELERR.BAS
   This is the code module that defines the error code constants used with the Custom API Function and defines the procedures used for checking the error codes.
5. MELSBERR.BAS
   This is the code module that defines the procedure used when checking the sub-error codes and defines the sub-error code constants. These sub-error codes and definitions are detailed information of the error codes used in the Custom API Function.

(2) Premise and supplement for explanations in this manual

This explanation of the program in this manual is premised on the Visual Basic Professional Edition version 4.0 (Japanese version). Thus, the details may differ when using other versions of Visual Basic.
When using the program explained in this manual with the English version of Visual Basic, some Visual Basic function names may differ. The functions that require changes between the Japanese and English versions are as follow.

Function name in Japanese version	Function name in English version
InStrB	InStr
MidB, MidB$	Mid, Mid$
LeftB, LeftB$	Left, Left$
RightB, RightB$	Right, Right$
Len B, Len B$	Len, Len$

Important technical information for programming the custom application is described in the following sections of the Visual Basic programming guide. Please read these sections before starting programming.

"Call of procedures in DLL"
"Compatibility with other Visual Basic versions"
"Specifications and limits"

3.2 Using Custom API Library

3.2.1 How to use the Custom API Library

Incorporate the code modules for the Custom API Library into the project

When using the custom application using the Custom API Library, the following code modules for the Custom API Library will be incorporated in the project.

```
NCMCAPI.BAS
MELNCAPI.BAS
MELTYPE.BAS
MELERR.BAS
MELSBERR.BAS
```

The Visual Basic [File] menu [Add File.....] function is used to incorporate the code module.

How to call the Custom API Functions

The functions (Custom API Functions) prepared in the Custom API Library are called as Visual Basic statements or functions.
The Custom API Functions are DLL procedures. To call the DLL procedure from Visual Basic, each DLL procedure must be declared beforehand using a declare statement. For the Custom API Function the procedure is declared with the NCMCAPI.BAS of the code module for the Custom API Library, so declaring again is not required.

The Custom API Function calling method is explained using melReadData as an example.
The Custom API Functions are function procedures that return a long value for the function return value, so they are called as follows for example.

```
lRetVal = melReadData (Me.hWnd, lAddress, lSectionNum, lSubSectionNum,
                                                        lReadData, lReadType)
```

Custom API Function error check

Each Custom API Function return value has a return value or error code determined for each function. Thus, check the return value for errors before using the value. Use the RetvIsError procedure to check whether the Custom API Function return value is an error code. The RetvIsError procedure is defined in the MELERR.BAS code module for the Custom API Library. An example of the error check is shown below.

```
If RetvIsError (lRetVal) = True Then
        'The error process is described
End If
```

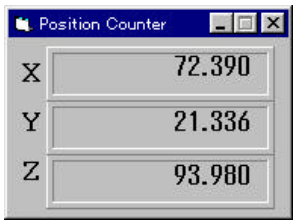
There are Custom API Functions that can retrieve a more detailed error code when an error occurs. This detailed error information is called a sub-error No. The sub-error code is retrieved using GetLastError() of Windows API. Call GetLastError() immediately after calling Custom API Function. It may not be possible to retrieve the sub-error code correctly if, directly after calling the Custom API Function, another Custom API or Windows API is called instead of GetLastError().
Refer to "Custom Application Interface Library Guide (Function section)(BNP-B2198)" for sub-error code details, usage, and the types of Custom API Functions that support the sub-error codes.

3.2.2 Creation of application with Custom API Library

3.2.2.1 Counter display application

What is a counter display application?

This counter display application (Position Counter) is a monitor used to display the NC's current position. The current positions for the No. 1 axis to No. 3 axis are displayed on the window. The No. of axes and axis names displayed are fixed. The displayed current position is updated in a 100ms cycle.



Custom API Functions to be used

melReadData

Creation of counter display application

With this application, a procedure called GetAxisPosition is created to get the current position of each axis from the NC Card. The GetAxisPosition is a function procedure used to return the current position of the axis having the axis No. designated with the argument. The current position returned by GetAxisPosition is a double type (double precision real number).

```
'Get current position of the designated axis
'The axis designation is 1 origin
Private Function GetAxisPosition (ByVal iAxisNum As Integer) As Double
    Dim dwStatus As Long                          'Variable to get return value from API function

    Dim lAddress As Long                          'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long       'Variable to designate NC Data Access Variable No.
    Dim dReadData As Double                       'Variable to store read data
    Dim lReadType As Long                         'Variable to designate requested data type

    Dim Message As String

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system, axis No. designation = 1_
    lAddress = ADR_MACHINE (1) Or ADR_SYSTEM (1) Or ADR_AXIS (iAxisNum)

    'Set NC Data Access Variable No.
    'Set current position data (Section No. = 21, Sub-section No. = 20032)
    lSectionNum = 21
    lSubSectionNum = 20032

    'Set read data type
    'Set double precision floating type (8-byte floating type)
    lReadtype = T_DOUBLE

    'Read current position data from the NC Card
    dwStatus = melReadData (Me.hWnd, lAddress, lSectionNum, lSubSectionNum, dReadData,  lReadType)

    'Check API function call for errors
    Call APIErrorCheck (dwStatus, "GetAxisPosition")

    'Return read current position
    GetAxisPosition = dReadData

  End Function
```

In GetAxisPosition, the Custom API Function melReadData is called, and the current position data for each axis is gotten from the NC Card. When calling melReadData in GetAxisPosition, address (lAddress) and NC Data Access Variable (lSectionNum, lSubSectionNum) are used to designate the current position data for each axis. The storage variable (dReadData) and requested data type (lReadType) is transferred to store the gotten current position.
The address is a long type (long integer type) data, and instructs the operation target for the Custom API Function. The Custom API Function knows which axis in which system of the NC Card to operate according to the address. The address designation is required when using most Custom API Functions. The NC Card, system and axis do not all need to be designated in all cases for the address designation. Depending on the Custom API Function, the system and axis, etc., may not need to be designated. The system and axis may need to be designated for the melReadData used here depending on the type of data to be gotten. Whether the address is required for the Custom API Function to be used and what needs to be designated is described in the "Custom Application Interface Library Guide (Function Section) (BNP-B2198)". ADR_MACHINE( ), ADR_SYSTEM( ) and ADR_AXIS( ) are used when creating the address in GetAxisPosition. These are procedures used to create the address, and are defined in the Custom API Library file MELNCAPI.BAS.

The NC Data Access Variables are Nos. assigned to the data in the NC Card. In the custom application, the data in the NC Card is read out by designating this No. and calling melReadData. The NC Data Access Variables include Section Nos. and Sub-section Nos., and one data item is designated with these two section Nos. The Section No. and Sub-section No. data is long type (long integer type) data. The details of the NC Data Access Variable section Nos. are described in the "Custom Application Interface Library Guide (Variable Section) (BNP-B2199)".

The requested data type designated the type of data to be read out by the custom application in regard to the Custom API Function. The custom application has variables of the data types to be read out (this is called the requested data type), and transfers that variable and type to the Custom API Function. The Custom API Function converts the data type originally held by the NC Card (this is called the default data type) into the requested data type, and returns it to the custom application. The types of data that can be requested to the Custom API Function are as follow.

Data types that can be requested to the Custom API Function

Data type		Type of variable prepared by custom application	
T_CHAR	1-byte integer type	Byte	Byte type
T_SHORT	2-byte integer type	Integer	Integer type
T_LONG	4-byte integer type	Long	Long integer type
T_DOUBLE	4-byte real number type	Double	Double precision real number type
T_STR	Character string type	STRINGTYPE	User defined array for character string data
T_DECSTR	Decimal integer character string type	STRINGTYPE	User defined array for character string data
T_HEXSTR	Hexadecimal character string type	STRINGTYPE	User defined array for character string data
T_BINSTR	Binary character string type	STRINGTYPE	User defined array for character string data
T_FLOATSTR	Real number character type	FLOATSTR	User defined array for real number character string data

It must be noted here that there are cases where the conversion into the requested data type may not be done correctly depending on the default data type of data to be read out. For example, the character string type (T_STR) data cannot be converted into a numerical value type such as the 4-byte integer type (T_LONG). If the 4-byte integer type (T_LONG) is converted into a 2-byte integer type (T_SHORT), the high-order 2-byte data of the 4-byte integer will be lost, and when the 4-byte real number type (T_DOUBLE) is converted into a 4-byte integer type (T_LONG), the data after the decimal point will be cut off. The combination of the default data type and requested data type is shown below.

The combination of the default data type and requested data type

	Requested data type									
Default data type	**T_BIT**	**T_CHAR**	**T_SHORT**	**T_LONG**	**T_DOUBLE**	**T_STR**	**T_DEC STR**	**T_HEXS TR**	**T_BINS TR**	**T_FLOATS TR**
T_CHAR	△	○	○	○	○	×	○	○	○	○
T_SHORT	△	△	○	○	○	×	○	○	○	○
T_LONG	△	△	△	○	○	×	○	○	○	○
T_DOUBLE	△	△	△	△	○	×	△	△	△	○
T_STR	×	×	×	×	×	○	×	×	×	×
T_DECSTR	△	△	△	△	△	○	○	○	○	○
T_HEXSTR	△	△	△	△	△	○	○	○	○	○
T_BINSTR	△	△	△	△	△	○	○	○	○	○
T_FLOATSTR	△	△	△	△	△	○	△	△	△	○

○: Conversion is possible
△: Conversion is possible, but some data may be lost
×: Conversion is not possible (An error will occur.)

List 1-1 COUNTER32.VBP project file

```
Form=Frmposit.frm
Module=Module1; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-
00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON
FILES\MICROSOFT SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Object={B16553C3-06DB-101B-85B2-0000C009BE81}#1.0#0; SPIN32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG32.OCX
ProjWinSize=200,391,243,281
ProjWinShow=0
IconForm="frmPosit"
HelpFile=""
```

```
Title="COUNTER32"
ExeName32="counter32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
```

List 1-2 FRMPOSIT.FRM Form module

```
VERSION 4.00
Begin VB.Form frmPosit
  Appearance    =      0  'Flat
  BackColor     =      &H00C0C0C0&
  BorderStyle   =      1 'Fixed (solid line)
  Caption       =      "Position Counter"
  ClientHeight  =      1728
  ClientLeft    =      3120
  ClientTop     =      1716
  ClientWidth   =      2775
  BeginProperty Font
      name        =   "System"
      charset     =   128
      weight      =   700
      size        =   13.2
      underline   =   0   'False
      italic      =   0   'False
      strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   2112
  Left            =   3072
  LinkTopic       =   "Form1"
  MaxButton       =   0   'False
  ScaleHeight     =   1728
  ScaleWidth      =   2772
  Top             =   1380
  Width           =   2868
  Begin VB.Timer timPosition
        Interval      =   100
        Left          =   2400
        Top           =   1320
  End
  Begin Threed.SSPanel Pnl3dAxis
        Height        =   540
        Index         =   0
        Left          =   372
        TabIndex      =   3
        Top           =   72
        Width         =   2268
        _Version      =   65536
        _ExtentX      =   3995
        _ExtentY      =   953
        _StockProps   =   15
        BevelInner    =   1
        Begin VB.Label lblAxisPosition
```

```
            Alignment      =   1  'Flush right
            BackStyle      =   0  'Transparent
            Caption        =   "-1234.1234"
            Height         =   390
            Index          =   0
            Left           =   75
            TabIndex       =   6
            Top            =   75
            Width          =   1965
        End
    End
    Begin Threed.SSPanel Pnl3dAxis
        Height         =   540
        Index          =   1
        Left           =   372
        TabIndex       =   4
        Top            =   600
        Width          =   2268
        _Version       =   65536
        _ExtentX       =   3995
        _ExtentY       =   953
        _StockProps    =   15
        BevelInner     =   1
        Begin VB.Label lblAxisPosition
            Alignment      =   1  'Flush right
            BackStyle      =   0  'Transparent
            Caption        =   "-1234.1234"
            Height         =   390
            Index          =   1
            Left           =   75
            TabIndex       =   7
            Top            =   75
            Width          =   1965
        End
    End
    Begin Threed.SSPanel Pnl3dAxis
        Height         =   540
        Index          =   2
        Left           =   375
        TabIndex       =   5
        Top            =   1125
        Width          =   2265
        _Version       =   65536
        _ExtentX       =   3995
        _ExtentY       =   953
        _StockProps    =   15
        BevelInner     =   1
        Begin VB.Label lblAxisPosition
            Alignment      =   1  'Flush right
            BackStyle      =   0  'Transparent
            Caption        =   "-1234.1234"
            Height         =   390
            Index          =   2
            Left           =   75
            TabIndex       =   8
            Top            =   75
            Width          =   1965
        End
    End
    Begin VB.Label lblAxisName
        Appearance     =   0  'Flat
        BackColor      =   &H80000005&
```

```
        BackStyle      =  0  'Transparent
        Caption        =  "Z"
        BeginProperty Font
            name       =  "Courier"
            charset    =  0
            weight     =  700
            size       =  15
            underline  =  0  'False
            italic     =  0  'False
            strikethrough  =  0  'False
        EndProperty
        ForeColor      =  &H80000008&
        Height         =  255
        Index          =  2
        Left           =  120
        TabIndex       =  2
        Top            =  1140
        Width          =  255
    End
    Begin VB.Label lblAxisName
        Appearance     =  0  'Flat
        BackColor      =  &H80000005&
        BackStyle      =  0  'Transparent
        Caption        =  "Y"
        BeginProperty Font
            name       =  "Courier"
            charset    =  0
            weight     =  700
            size       =  15
            underline  =  0  'False
            italic     =  0  'False
            strikethrough  =  0  'False
        EndProperty
        ForeColor      =  &H80000008&
        Height         =  255
        Index          =  1
        Left           =  120
        TabIndex       =  1
        Top            =  660
        Width          =  255
    End
    Begin VB.Label lblAxisName
        Appearance     =  0  'Flat
        BackColor      =  &H80000005&
        BackStyle      =  0  'Transparent
        Caption        =  "X"
        BeginProperty Font
            name       =  "Courier"
            charset    =  0
            weight     =  700
            size       =  15
            underline  =  0  'False
            italic     =  0  'False
            strikethrough  =  0  'False
        EndProperty
        ForeColor      =  &H80000008&
        Height         =  255
        Index          =  0
        Left           =  120
        TabIndex       =  0
        Top            =  180
        Width          =  255
    End
End
```

```
Attribute VB_Name = "frmPosit"
Attribute VB_Creatable = False
Attribute VB_Exposed = False

Option Explicit

'Get current position of the designated axis
'The axis designation is 1 origin
Private Function GetAxisPosition(ByVal iAxisNum As Integer) As Double
    Dim dwStatus As Long                        'Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC data Access Variable No.
    Dim dReadData As Double                     'Variable to store read data
    Dim lReadType As Long                       'Variable to designate requested data type

    Dim Message As String

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system, axis No. designation = 1~
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1) Or ADR_AXIS(iAxisNum)

    'Set NC Data Access Variable No.
    'Set current position data (Section No. = 21, Sub-section No. 20032)
    lSectionNum = 21
    lSubSectionNum = 20032

    'Set read data type
    'The double precision floating type (8-byte floating type) is set
    lReadType = T_DOUBLE

    'Read current position data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, dReadData, lReadType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetAxisPosition")

    'Return read current position
    GetAxisPosition = dReadData

End Function

'Timer process of Position Counter window
Private Sub timPosition_Timer()
    Dim dReadData As Double          'Variable to store read data
    Dim iAxisNum As Integer          'Loop counter

    'Get current positions for 3 axes and display on screen
    For iAxisNum = 0 To 2
        'Get current position
        dReadData = GetAxisPosition(iAxisNum + 1)

        'Display read current position on screen
        lblAxisPosition(iAxisNum).Caption = Format$(dReadData, "0.000")
    Next

End Sub
```

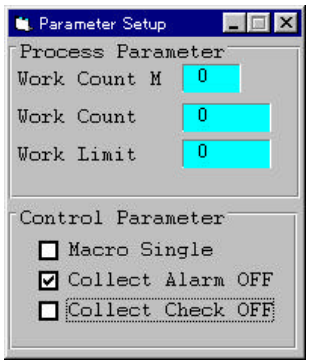
List 1-3 COMMON.BAS code module

```
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
   Dim Message As String
   If RetvIsError(dwStatus) = True Then
      'Error occurrence
      'Display error message
      Message = "Error occurred in API function call"
      Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
      Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
      MsgBox (Message)

      'Quit application
      'Stop
      End

   End If

End Sub
```

3.2.2.2 Parameter setting application

What is the parameter setting application?

The parameter setting application (Parameter Setup) is a tool for setting the NC parameters. The three machining parameters in [ProcessParameter], [WorkCountM] (M code for counting No. of workpiece machinings), [WorkCount] (No. of workpiece machinings to this point) and [WorkLimit] (Max. No. of workpiece machinings), and the three control parameters in [ControlParameter], [MacroSingle] (macro single valid), [CollectAlarmOFF] (interference version) and [CollectCheckOFF] (interference check invalid), are displayed on the window.
For each parameter in the [ProcessParameter], a numerical value is input in the text box, and the [Enter] key is pressed to write the set data in the NC Card. If each item for the [ControlParameter] parameters is clicked, the ON/OFF status of the parameter will change, and the setting value will be simultaneously written into the NC Card.
When the application is started, each parameter value will be read from the NC Card and displayed.



Custom API Function to be used

melReadData
melWriteData

Creation of parameter setting application

Getting of parameter values

In this application, the following procedure is created to get the parameter values from the NC Card. These procedures get the parameter values from the NC Card and display the gotten values. These procedures are sub-procedures and do not have a return value.

Procedures created to get parameters:

GetCollectAlarm
GetCollectCheck
GetMacroSingle
GetWorkCount
GetWorkCountM
GetWorkLimit

```
'Get interference evasion parameter value from the NC Card and display
Private Sub GetCollectAlarm()
   Dim dwStatus As Long                          ' Variable to get return value from API function

   Dim lAddress As Long                          'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long       'Variable to designate NC Data Access Variable No.
   Dim nReadData As Integer                      'Variable to store read data (short type)
   Dim lDataType As Long                 'Variable to designate requested data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*************************************************
   'Read the interference evasion parameter value
   '*************************************************
   'Set NC Data Access Variable No.
   'Designate interference evasion parameter (Section No. = 1, Sub-section No. = 193)
   lSectionNum = 1
   lSubSectionNum = 193

   'Set read data type
   'Set integer type (2-byte integer type)
   lDataType = T_SHORT

   'Read data from the NC Card
   dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "GetCollectAlarm")

   'Get only the interference evasion parameter value from the read data
   nReadData = nReadData And &H20

   'Turn check box ON/OFF according to the read parameter value
   If nReadData <> 0 Then
      'Parameter ON
      'Set check box to check state
      ChkCollectAlarm.Value = 1
   Else
      'Parameter OFF
      'Set check box to not-check state
      ChkCollectAlarm.Value = 0
   End If

End Sub
```

```
'Get interference Check invalid parameter value from the NC Card and display
Private Sub GetCollectCheck()
   Dim dwStatus As Long                  ' Variable to get return value from API function

   Dim lAddress As Long                  'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
   Dim nReadData As Integer                    'Variable to store read data (short type)
   Dim lDataType As Long                 'Variable to designate requested data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*****************************************************
   'Read the interference check invalid parameter value
   '*****************************************************
   'Designate NC Data Access Variable No.
   'Set interference check invalid parameter (Section No. = 1, Sub-section No. = 198)
   lSectionNum = 1
   lSubSectionNum = 198

   'Set read data type
   'Set integer type (2-byte integer type)
   lDataType = T_SHORT

   'Read data from the NC Card
   dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "GetCollectCheck")

   'Get only the interference check invalid parameter value from the read data
   nReadData = nReadData And &H40

   'Turn check box ON/OFF according to the read parameter value
   If nReadData <> 0 Then
      'Parameter ON
      'Set check box to check state
      ChkCollectCheck.Value = 1
   Else
      'Parameter OFF
      'Set check box to not-check state
      ChkCollectCheck.Value = 0
   End If

End Sub

'Get Check macro single valid parameter value from the NC Card and display
Private Sub GetMacroSingle()
   Dim dwStatus As Long                  ' Variable to get return value from API function

   Dim lAddress As Long                  'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
   Dim nReadData As Integer                    'Variable to store read data (short type)
   Dim lDataType As Long                 'Variable to designate requested data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*****************************************************
   'Read the macro single parameter value
   '*****************************************************
   'Designate NC Data Access Variable No.
   'Set macro single parameter (Section No. = 1, Sub-section No. = 194)
   lSectionNum = 1
   lSubSectionNum = 194
```

```
    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetMacroSingle")

    'Get only the macro single parameter value from the read data
    nReadData = nReadData And &H40

   'Turn check box ON/OFF according to the read parameter value
    If nReadData <> 0 Then
        'Parameter ON
        'Set check box to check state
        ChkMacroSingle.Value = 1
    Else
        'Parameter OFF
        'Set check box to not-check state
        ChkMacroSingle.Value = 0
    End If

End Sub

'Get No. of workpiece machinings parameter value from the NC Card and display
Private Sub GetWorkCount()
     Dim dwStatus As Long                    ' Variable to get return value from API function

    Dim lAddress As Long                     'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long      'Variable to designate NC Data Access Variable No.
    Dim nReadData As Long                            'Variable to store read data (long type)
    Dim lDataType As Long                    'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '**************************************************
    'Read No. of workpiece machinings parameter value
    '**************************************************
    'Designate NC Data Access Variable No.
    'Set No. of workpiece machinings parameter (Section No. = 10318, Sub-section No. = 2896)
    lSectionNum = 10318
    lSubSectionNum = 2896

    'Set read data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkCount")

    'Display read No. of workpiece machinings value
    txtWorkCount.Text = Str$(lReadData)

End Sub

'Get No. of workpiece machinings M parameter value from the NC Card and display
Private Sub GetWorkCountM()
    Dim dwStatus As Long                     ' Variable to get return value from API function

    Dim lAddress As Long                     'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long      'Variable to designate NC Data Access Variable No.
```

```
    Dim nReadData As Integer                    'Variable to store read data (short type)
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*************************************************
    'Read No. of workpiece machinings M parameter value
    '*************************************************
    'Designate NC Data Access Variable No.
    'Set No. of workpiece machinings M parameter (Section No. = 1, Sub-section No. = 1280)
    lSectionNum = 1
    lSubSectionNum = 1280

    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkCountM")

     'Display read No. of workpiece machinings M value
    txtWorkCountM.Text = Str$(nReadData)

End Sub

'Get max. workpiece value parameter value from the NC Card and display
Private Sub GetWorkLimit()
   Dim dwStatus As Long                         ' Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Long                       'Variable to store read data (long type)
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*****************************************
    'Read max. workpiece value parameter value
    '*****************************************
    'Designate NC Data Access Variable No.
    'Set max. workpiece value parameter (Section No. = 10318, Sub-section No. = 2898)
    lSectionNum = 10318
    lSubSectionNum = 2898

    'Set read data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG
```

```
    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkLimit")

    'Display read max. workpiece value
    txtWorkLimit.Text = Str$(lReadData)

End Sub
```

In the procedure for getting the parameter value, the Custom API Function melReadData is called and each parameter value is gotten from the NC Card. Refer to section "3.2.2.1 Counter display application" for how to use melReadData.
In each procedure, the parameter to be gotten is designated in the NC Data Access Variable. However the following items of the gotten parameters are bit parameters, so special processing is required.

Bit parameter type parameters (Only those used)

Parameter name	Section No.	Sub-section No.	Default data type	Bit
Interference evasion	1	193	T_CHAR	Bit 5
Interference check invalid	1	198	T_CHAR	Bit 6
Single block valid	1	194	T_CHAR	Bit 6

The NC Data Access Variable default data type for these parameters is T_CHAR (1-byte integer type), but the actual parameter value is one of the bits in the one byte. Thus, a process to get a specific bit from the byte gotten from the NC Card is required.
For example, the following process is executed in the GetCollectAlarm procedure.

```
'Get only the interference evasion parameter value from the read data
    nReadData = nReadData And &H40

    'Turn check box ON/OFF according to the read parameter value
    If nReadData <> 0 Then
        'Parameter ON
        'Set check box to check state
        ChkCollectAlarm.Value = 1
    Else
        'Parameter OFF
        'Set check box to not-check state
        ChkCollectAlarm.Value = 0
End If
```

Writing of parameter values

In this application, the following procedure is created to write the parameter values set with the application into the NC Card. These procedures write the parameter value transferred with an argument into the NC Card. These procedures are sub-procedures and do not have a return value.

Procedures for writing parameters

SetCollectAlarm
SetCollectCheck
SetMacroSingle
SetWorkCount
SetWorkCountM
SetWorkLimit

```
Private Sub SetCollectAlarm(ByVal SetValue As Integer)
  Dim dwStatus As Long                          ' Variable to get return value from API function

   Dim lAddress As Long                 'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
   Dim nWriteData As Integer                   'Variable to store written data (short type)
   Dim lDataType As Long                'Variable to designate write data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*********************************************
   'Write the interference evasion parameter value
   '*********************************************
   'Set NC Data Access Variable No.
   'Designate interference evasion parameter (Section No. = 1, Sub-section No. = 193)
   lSectionNum = 1
   lSubSectionNum = 193

   'Set write data type
   'Set integer type (2-byte integer type)
   lDataType = T_SHORT

   'Read current value data from the NC Card
   dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "SetCollectAlarm")
   'Mask read data
   nWriteData = nWriteData And &HDF

   'Set interference evasion parameter value
   If SetValue <> 0 Then
      'Parameter ON
      nWriteData = nWriteData Or &H20
   End If

   'Write data to NC Card
   dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
   Call APIErrorCheck(dwStatus, "SetCollectAlarm")

End Sub

Private Sub SetCollectCheck(ByVal SetValue As Integer)
   Dim dwStatus As Long                 ' Variable to get return value from API function

   Dim lAddress As Long                 'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long 'Variable to designate NC Data Access Variable No.
   Dim nWriteData As Integer              'Variable to store written data (short type)
   Dim lDataType As Long                'Variable to designate write data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*****************************************************
   'Read interference check invalid parameter value
   '*****************************************************
   'Designate NC Data Access Variable No.
   'Set interference check invalid parameter (Section No. = 1, Sub-section No. = 198)
   lSectionNum = 1
   lSubSectionNum = 198

   'Set write data type
   'Set integer type (2-byte integer type)
```

```
    lDataType = T_SHORT

     'Read current value data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetCollectCheck")

    'Mask read data
    nWriteData = nWriteData And &HBF

    'Set interference check invalid parameter value
    If SetValue <> 0 Then
        'Parameter ON
        nWriteData = nWriteData Or &H40
    End If

    'Write data to NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetCollectCheck")

End Sub

Private Sub SetMacroSingle(ByVal SetValue As Integer)
   Dim dwStatus As Long                                ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long      'Variable to designate NC Data Access Variable No.
    Dim nWriteData As Integer                           'Variable to store written data (short type)
    Dim lDataType As Long                   'Variable to designate written data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*****************************************************
    'Write macro single prameter value
    '*****************************************************
    'Designate NC Data Access Variable No.
    'Set macro single parameter (Section No. = 1, Sub-section No. = 194)
    lSectionNum = 1
    lSubSectionNum = 194

    'Set write data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read current value data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetMacroSingle")

    'Mask read data
    nWriteData = nWriteData And &HBF

    'Set macro single parameter value
    If SetValue <> 0 Then
         'Parameter ON
        nWriteData = nWriteData Or &H40
    End If

    'Write data to NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
```

```
    Call APIErrorCheck(dwStatus, "SetMacroSingle")

End Sub

Private Sub SetWorkCount(ByVal SetValue As Long)
    Dim dwStatus As Long                ' Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be written
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '**************************************************
    'Write No. of workpiece machinings parameter value
    '**************************************************
    'Set NC Data Access Variable No.
    'Set No. of workpiece machinings parameter (Section No. = 10318, Sub-section No. = 2896)
    lSectionNum = 10318
    lSubSectionNum = 2896

    'Set write data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Write data to the NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetWorkCount")

End Sub

Private Sub SetWorkCountM(ByVal SetValue As Integer)
    Dim dwStatus As Long                ' Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be written
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '**************************************************
    'Write No. of workpiece machinings M parameter value
    '**************************************************
    'Set NC Data Access Variable No.
    'Set No. of workpiece machinings M parameter (Section No. = 1, Sub-section No. = 1280)
    lSectionNum = 1
    lSubSectionNum = 1280

    'Set write data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Write data to the NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetWorkCountM")

End Sub

Private Sub SetWorkLimit(ByVal SetValue As Long)
    Dim dwStatus As Long                ' Variable to get return value from API function
```

```
  Dim lAddress As Long                    'Variable to designate address
  Dim lSectionNum, lSubSectionNum As Long      'Variable to designate NC Data Access Variable No.
  Dim lDataType As Long                   'Variable to designate requested data type

  'Set address of data to be written
  'NC Card No. = 1, system designation = No. 1 system
  lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

  '*******************************************
  'Write max. workpiece value parameter value
  '*******************************************
  'Designate NC Data Access Variable No.
  'Set max. workpiece value parameter (Section No. = 10318, Sub-section No. = 2898)
  lSectionNum = 10318
  lSubSectionNum = 2898

  'Set write data type
  'Set long integer type (4-byte integer type)
  lDataType = T_LONG

  'Write data from the NC Card
  dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)

  'Check API function call for errors
  Call APIErrorCheck(dwStatus, "SetWorkLimit")

End Sub
```

In the procedure for writing the parameter value, the Custom API Function melWriteData is called, and each parameter value is written to the NC Card. When calling melWriteData in each procedure, each parameter is designated by using address (lAddress) and NC Data Access Variable (lSectionNum, lSubSectionNum). The variable (SetValue) where the setting value is set and the write data type (lDataType) is transferred to transfer the write parameter setting value.
The details of the melWriteData argument address and NC Data Access Variable are the same as melReadData. Refer to section "3.2.2.1 Counter display application" for details on these.

The write data type has approximately the same meaning as the melReadData requested data type. The write data type is used to designate the type of data to be written in by the custom application for the custom API Function. The custom application prepares the variable of the type of data to be written (this is called the write data type), and transfers that variable and type to the Custom API Function. The Custom API Function converts the write data type to the default data type originally held by the NC Card, and then writes the data to the NC Card. The following data types can be designated for the Custom API Function. These are the same as the request data type.

Write data types that can be designated for Custom API Function

Data Type		Type of variable prepared by custom application	
T_BIT	1-bit data type	Cannot be used with Visual Basic	
T_CHAR	1-byte integer type	CHAR	Byte type
T_SHORT	2-byte integer type	Integer	Integer type
T_LONG	4-byte integer type	Long	Long integer type
T_DOUBLE	4-byte real number type	Double	Double precision real number type
T_STR	Character string type	STRINGTYPE	User defined array for character string data
T_DECSTR	Decimal integer character string type	STRINGTYPE	User defined array for character string data
T_HEXSTR	Hexadecimal character string type	STRINGTYPE	User defined array for character string data
T_BINSTR	Binary character string type	STRINGTYPE	User defined array for character string data
T_FLOATSTR	Real number character string type	FLOATSTR	User defined array for real number character string data

As with the requested data type, it must be noted here that there are cases where the conversion into the write data type may not be done correctly depending on the default data type of data to be written.

Next, the bit parameter process will be explained.
When writing a bit parameter, a special process is required as when reading the parameters.
The NC Data Access Variable default data type for the bit parameters used in this application is T_CHAR (1-byte integer type), but the actual parameter value is one of the bits in the one byte. The other bits in the one byte correspond to other parameters. Thus, when writing a value into the NC Card, a process is required so that the other bits in the one byte are not rewritten.
For example, in the SetCollectAlarm procedure, the current value is read before writing, and the write data is created based on that read value.

```
'Read current value data from NC Card
dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)
                :
                :
                :
'Mask read data
nWriteData = nWriteData And &HDF

'Set interface evasion parameter value
If SetValue <> 0 Then
   'Parameter ON
   nWriteData = nWriteData Or &H20
End If

'Write data to NC Card
dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)
                :
                :
                :
```

List 2-1 PARAMET32.VBP project file

```
Form=Frmparam.frm
Module=Module1; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard
OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT
SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Object={B16553C3-06DB-101B-85B2-0000C009BE81}#1.0#0; SPIN32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG32.OCX
ProjWinSize=81,397,243,274
ProjWinShow=0
IconForm="frmParam"
HelpFile=""
Title="PARAMETER"
ExeName32="Paramet32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
```

List 2-2 FRMPARAM.FRM form module

```
VERSION 4.00
Begin VB.Form frmParam
   Appearance      =   0  'Flat
   BackColor       =   &H00C0C0C0&
   BorderStyle     =   1  'Fixed (solid line)
   Caption         =   "Parameter Setup"
   ClientHeight    =   3156
   ClientLeft      =   1920
   ClientTop       =   1968
   ClientWidth     =   2916
   BeginProperty Font
      name           =   "System"
      charset        =   128
      weight         =   400
      size           =   13.2
      underline      =   0   'False
      italic         =   0   'False
      strikethrough  =   0   'False
   EndProperty
   ForeColor       =   &H80000008&
   Height          =   3540
   Left            =   1872
```

```
LinkTopic       =   "Form1"
LockControls    =   -1  'True
MaxButton       =   0   'False
ScaleHeight     =   3156
ScaleWidth      =   2916
Top             =   1632
Width           =   3012
Begin VB.CheckBox ChkMacroSingle
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  Caption         =   "Macro Single"
  BeginProperty Font
    name            =   "Courier"
    charset         =   0
    weight          =   400
    size            =   9.6
    underline       =   0   'False
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   315
  Left            =   300
  TabIndex        =   9
  Top             =   1950
  Width           =   2475
End
Begin VB.CheckBox ChkCollectAlarm
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  Caption         =   "Collect Alarm OFF"
  BeginProperty Font
    name            =   "Courier"
    charset         =   0
    weight          =   400
    size            =   9.6
    underline       =   0   'False
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   315
  Left            =   300
  TabIndex        =   8
  Top             =   2250
  Width           =   2475
End
Begin VB.CheckBox ChkCollectCheck
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  Caption         =   "Collect Check OFF"
  BeginProperty Font
    name            =   "Courier"
    charset         =   0
    weight          =   400
    size            =   9.6
    underline       =   0   'False
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   315
```

```
    Left          =   300
    TabIndex      =   7
    Top           =   2550
    Width         =   2475
  End
  Begin Threed.SSFrame Frame3d1
    Height        =   1590
    Left          =   0
    TabIndex      =   0
    Top           =   0
    Width         =   2895
    _Version      =   65536
    _ExtentX      =   5106
    _ExtentY      =   2805
    _StockProps   =   14
    Caption       =   "Process Parameter"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      name        =   "Courier"
      charset     =   0
      weight      =   400
      size        =   9.6
      underline   =   0   'False
      italic      =   0   'False
      strikethrough   =   0   'False
    EndProperty
    Begin Threed.SSPanel pnl3dWorkCountM
      Height      =   372
      Left        =   1680
      TabIndex    =   1
      Top         =   216
      Width       =   672
      _Version    =   65536
      _ExtentX    =   1191
      _ExtentY    =   661
      _StockProps =   15
      BevelOuter  =   0
      BevelInner  =   1
      Begin VB.TextBox txtWorkCountM
        Alignment     =   1  'Flush right
        BackColor     =   &H00FFFF00&
        BorderStyle   =   0  'None
        BeginProperty Font
          name        =   "Courier"
          charset     =   0
          weight      =   400
          size        =   9.6
          underline   =   0   'False
          italic      =   0   'False
          strikethrough   =   0   'False
        EndProperty
        Height      =   270
        Left        =   60
        TabIndex    =   11
        Text        =   "99"
        Top         =   60
        Width       =   555
      End
    End
    Begin Threed.SSPanel pnl3dWorkCount
      Height      =   372
      Left        =   1680
      TabIndex    =   3
```

```
    Top            =   600
    Width          =   972
    _Version       =   65536
    _ExtentX       =   1720
    _ExtentY       =   661
    _StockProps    =   15
    BevelOuter     =   0
    BevelInner     =   1
    RoundedCorners =   0   'False
    Begin VB.TextBox txtWorkCount
      Alignment      =   1  'Flush right
      BackColor      =   &H00FFFF00&
      BorderStyle    =   0  'None
      BeginProperty Font
        name           =   "Courier"
        charset        =   0
        weight         =   400
        size           =   9.6
        underline      =   0   'False
        italic         =   0   'False
        strikethrough  =   0   'False
      EndProperty
      Height         =   270
      Left           =   60
      TabIndex       =   12
      Text           =   "123456"
      Top            =   60
      Width          =   855
    End
  End
  Begin Threed.SSPanel pnl3dWorkLimit
    Height         =   372
    Left           =   1680
    TabIndex       =   6
    Top            =   960
    Width          =   972
    _Version       =   65536
    _ExtentX       =   1720
    _ExtentY       =   661
    _StockProps    =   15
    BevelOuter     =   0
    BevelInner     =   1
    Begin VB.TextBox txtWorkLimit
      Alignment      =   1  'Flush right
      BackColor      =   &H00FFFF00&
      BorderStyle    =   0  'None
      BeginProperty Font
        name           =   "Courier"
        charset        =   0
        weight         =   400
        size           =   9.6
        underline      =   0   'False
        italic         =   0   'False
        strikethrough  =   0   'False
      EndProperty
      Height         =   270
      Left           =   60
      TabIndex       =   13
      Text           =   "123456"
      Top            =   60
      Width          =   855
    End
  End
```

```
Begin VB.Label lblWorkCountM
  Appearance      =   0  'Flat
  BackColor       =   &H80000005&
  BackStyle       =   0  'Transparent
  Caption         =   "Work Count M"
  BeginProperty Font
    name          =   "Courier"
    charset       =   0
    weight        =   400
    size          =   9.6
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   195
  Left            =   75
  TabIndex        =   5
  Top             =   300
  Width           =   1455
End
Begin VB.Label lblWorkCount
  Appearance      =   0  'Flat
  BackColor       =   &H80000005&
  BackStyle       =   0  'Transparent
  Caption         =   "Work Count"
  BeginProperty Font
    name          =   "Courier"
    charset       =   0
    weight        =   400
    size          =   9.6
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   195
  Left            =   75
  TabIndex        =   4
  Top             =   675
  Width           =   1215
End
Begin VB.Label lblWorkLimit
  Appearance      =   0  'Flat
  BackColor       =   &H80000005&
  BackStyle       =   0  'Transparent
  Caption         =   "Work Limit"
  BeginProperty Font
    name          =   "Courier"
    charset       =   0
    weight        =   400
    size          =   9.6
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   195
  Left            =   75
  TabIndex        =   2
  Top             =   1050
  Width           =   1215
End
```

```
    End
    Begin Threed.SSFrame Frame3d2
      Height          =   1440
      Left            =   0
      TabIndex        =   10
      Top             =   1656
      Width           =   2868
      _Version        =   65536
      _ExtentX        =   5054
      _ExtentY        =   2540
      _StockProps     =   14
      Caption         =   "Control Parameter"
      BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
        name          =   "Courier"
        charset       =   0
        weight        =   400
        size          =   9.6
        underline     =   0   'False
        italic        =   0   'False
        strikethrough =   0   'False
      EndProperty
    End
End
Attribute VB_Name = "frmParam"
Attribute VB_Creatable = False
Attribute VB_Exposed = False

Option Explicit

Const KEY_ENTER = 13
Const KEY_ESC = 27

Private Sub ChkCollectAlarm_Click()
    Call SetCollectAlarm(ChkCollectAlarm.Value)
End Sub

Private Sub ChkCollectAlarm_KeyPress(KeyAscii As Integer)
    If KeyAscii = KEY_ENTER Then
        SendKeys "{TAB}"
    End If

End Sub

Private Sub ChkCollectCheck_Click()
    Call SetCollectCheck(ChkCollectCheck.Value)
End Sub

Private Sub ChkCollectCheck_KeyPress(KeyAscii As Integer)
    If KeyAscii = KEY_ENTER Then
        SendKeys "{TAB}"
    End If

End Sub

Private Sub ChkMacroSingle_Click()
    Call SetMacroSingle(ChkMacroSingle.Value)
End Sub

Private Sub ChkMacroSingle_KeyPress(KeyAscii As Integer)
    If KeyAscii = KEY_ENTER Then
        SendKeys "{TAB}"
    End If
```

```
End Sub

Private Sub Form_Load()

    'Read machining parameter
    Call GetWorkCountM      'Read No. of workpiece machinings M parameter
    Call GetWorkCount       'Read No. of workpiece machinings  parameter
    Call GetWorkLimit       'Read max. workpiece value parameter

    'Read control parameter
    Call GetMacroSingle                         'Read macro single parameter
    Call GetCollectAlarm                        'Read interference evasion parameter
    Call GetCollectCheck                        'Read interference check valid parameter

End Sub

'Get interference evasion parameter value from the NC Card and display
Private Sub GetCollectAlarm()
    Dim dwStatus As Long                'Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Integer                    'Variable to store read data (short type)
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*************************************************
    'Read the interference evasion parameter value
    '*************************************************
    'Set NC Data Access Variable No.
    'Designate interference evasion parameter (Section No. = 1, Sub-section No. = 193)
    lSectionNum = 1
    lSubSectionNum = 193

    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetCollectAlarm")

    'Get only the interference evasion parameter value from the read data
    nReadData = nReadData And &H20

    'Turn check box ON/OFF according to the read parameter value
    If nReadData <> 0 Then
        'Parameter ON
        'Set check box to check state
        ChkCollectAlarm.Value = 1
    Else
        'Parameter OFF
        'Set check box to not-check state
        ChkCollectAlarm.Value = 0
    End If

End Sub
```

```
'Get interference Check invalid parameter value from the NC Card and display
Private Sub GetCollectCheck()
    Dim dwStatus As Long                    ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Integer                    'Variable to store read data (short type)
    Dim lDataType As Long                   'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*****************************************************
    'Read the interference check invalid prameter value
    '*****************************************************
    'Designate NC Data Access Variable No.
    'Set interference check invalid parameter (Section No. = 1, Sub-section No. = 198)
    lSectionNum = 1
    lSubSectionNum = 198

    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

   'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetCollectCheck")

    'Get only the interference check invalid parameter value from the read data
    nReadData = nReadData And &H40

    'Turn check box ON/OFF according to the read parameter value
    If nReadData <> 0 Then
        'Parameter ON
        'Set check box to check state
        ChkCollectCheck.Value = 1
    Else
        'Parameter OFF
        'Set check box to not-check state
        ChkCollectCheck.Value = 0
    End If

End Sub

'Get Check macro single valid parameter value from the NC Card and display
Private Sub GetMacroSingle()
    Dim dwStatus As Long                    ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Integer                    'Variable to store read data (short type)
    Dim lDataType As Long                   'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)
```

```
    '*****************************************************
    'Read the macro single prameter value
    '*****************************************************
    'Designate NC Data Access Variable No.
    'Set macro single parameter (Section No. = 1, Sub-section No. = 194)
    lSectionNum = 1
    lSubSectionNum = 194

    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetMacroSingle")

    'Get only the macro single parameter value from the read data
    nReadData = nReadData And &H40

    'Turn check box ON/OFF according to the read parameter value
    If nReadData <> 0 Then
       'Parameter ON
       'Set check box to check state
       ChkMacroSingle.Value = 1
    Else
       'Parameter OFF
       'Set check box to not-check state
       ChkMacroSingle.Value = 0
    End If

End Sub

'Get No. of workpiece machinings parameter value from the NC Card and display
Private Sub GetWorkCount()
     Dim dwStatus As Long                  ' Variable to get return value from API function

    Dim lAddress As Long                   'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Long                       'Variable to store read data (long type)
    Dim lDataType As Long                  'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*************************************************
    'Read No. of workpiece machinings parameter value
    '*************************************************
    'Designate NC Data Access Variable No.
    'Set No. of workpiece machinings parameter (Section No. = 10318, Sub-section No. = 2896)
    lSectionNum = 10318
    lSubSectionNum = 2896

    'Set read data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lDataType)
```

```
    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkCount")

    'Display read No. of workpiece machinings value
    txtWorkCount.Text = Str$(lReadData)

End Sub

'Get No. of workpiece machinings M parameter value from the NC Card and display
Private Sub GetWorkCountM()
    Dim dwStatus As Long                    ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim nReadData As Integer                    'Variable to store read data (short type)
    Dim lDataType As Long                   'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*****************************************************
    'Read No. of workpiece machinings M value
    '*****************************************************
    'Designate NC Data Access Variable No.
    'Set No. of workpiece machinings M parameter (Section No. = 1, Sub-section No. = 1280)
    lSectionNum = 1
    lSubSectionNum = 1280

    'Set read data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkCountM")

     'Display read No. of workpiece machinings M value
    txtWorkCountM.Text = Str$(nReadData)

End Sub

'Get max. workpiece value parameter value from the NC Card and display
Private Sub GetWorkLimit()
   Dim dwStatus As Long                         ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lReadData As Long                       'Variable to store read data (long type)
    Dim lDataType As Long                   'Variable to designate requested data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*******************************************
    'Read max. workpiece value parameter value
    '*******************************************
    'Designate NC Data Access Variable No.
    'Set max. workpiece value parameter (Section No. = 10318, Sub-section No. = 2898)
```

```
    lSectionNum = 10318
    lSubSectionNum = 2898

    'Set read data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Read data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetWorkLimit")

    'Display read max. workpiece value
     txtWorkLimit.Text = Str$(lReadData)

End Sub

Private Sub SetCollectAlarm(ByVal SetValue As Integer)
   Dim dwStatus As Long                                ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long      'Variable to designate NC Data Access Variable No.
    Dim nWriteData As Integer                           'Variable to store written data (short type)
    Dim lDataType As Long                   'Variable to designate write data type

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*********************************************
    'Write the interference evasion parameter value
    '*********************************************
    'Set NC Data Access Variable No.
    'Designate interference evasion parameter (Section No. = 1, Sub-section No. = 193)
    lSectionNum = 1
    lSubSectionNum = 193

    'Set write data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read current value data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetCollectAlarm")

    'Mask read data
    nWriteData = nWriteData And &HDF

   'Set interference evasion parameter value
    If SetValue <> 0 Then
       'Parameter ON
       nWriteData = nWriteData Or &H20
    End If

    'Write data to NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

     'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetCollectAlarm")
```

```
End Sub

Private Sub SetCollectCheck(ByVal SetValue As Integer)
   Dim dwStatus As Long                    ' Variable to get return value from API function

   Dim lAddress As Long                    'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
   Dim nWriteData As Integer                   'Variable to store written data (short type)
   Dim lDataType As Long                   'Variable to designate write data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   '*****************************************************
   'Write interference check invalid prameter value
   '*****************************************************
   'Designate NC Data Access Variable No.
   'Set interference check invalid parameter (Section No. = 1, Sub-section No. = 198)
   lSectionNum = 1
   lSubSectionNum = 198

   'Set write data type
   'Set integer type (2-byte integer type)
   lDataType = T_SHORT

    'Read current value data from the NC Card
   dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "SetCollectCheck")

   'Mask read data
   nWriteData = nWriteData And &HBF

   'Set interference check invalid parameter value
   If SetValue <> 0 Then
      'Parameter ON
      nWriteData = nWriteData Or &H40
   End If

   'Write data to NC Card
   dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "SetCollectCheck")

End Sub

Private Sub SetMacroSingle(ByVal SetValue As Integer)
  Dim dwStatus As Long                         ' Variable to get return value from API function

   Dim lAddress As Long                    'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
   Dim nWriteData As Integer                   'Variable to store written data (short type)
   Dim lDataType As Long                   'Variable to designate write data type

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)
```

```
    '*****************************************************
    'Write macro single prameter value
    '*****************************************************
    'Designate NC Data Access Variable No.
    'Set macro single parameter (Section No. = 1, Sub-section No. = 194)
    lSectionNum = 1
    lSubSectionNum = 194

    'Set write data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Read current value data from the NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetMacroSingle")

    'Mask read data
    nWriteData = nWriteData And &HBF

    'Set macro single parameter value
    If SetValue <> 0 Then
        'Parameter ON
       nWriteData = nWriteData Or &H40
    End If

   'Write data to NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nWriteData, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetMacroSingle")

End Sub

Private Sub SetWorkCount(ByVal SetValue As Long)
    Dim dwStatus As Long                    ' Variable to get return value from API function

    Dim lAddress As Long                    'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lDataType As Long                   'Variable to designate requested data type

    'Set address of data to be written
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*****************************************************
    'Write No. of workpiece machinings value
    '*****************************************************
    'Set NC Data Access Variable No.
    'Set No. of workpiece machinings parameter (Section No. = 10318, Sub-section No. = 2896)
    lSectionNum = 10318
    lSubSectionNum = 2896

    'Set write data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Write data to the NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)
```

```
    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetWorkCount")

End Sub

Private Sub SetWorkCountM(ByVal SetValue As Integer)
    Dim dwStatus As Long                ' Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be written
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '**************************************************
    'Write No. of workpiece machinings M value
    '**************************************************
    'Set NC Data Access Variable No.
    'Set No. of workpiece machinings M parameter (Section No. = 1, Sub-section No. = 1280)
    lSectionNum = 1
    lSubSectionNum = 1280

    'Set write data type
    'Set integer type (2-byte integer type)
    lDataType = T_SHORT

    'Write data from the NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetWorkCountM")

End Sub

Private Sub SetWorkLimit(ByVal SetValue As Long)
    Dim dwStatus As Long                ' Variable to get return value from API function

    Dim lAddress As Long                'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lDataType As Long               'Variable to designate requested data type

    'Set address of data to be written
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    '*****************************************
    'Write max. workpiece value
    '*****************************************
    'Designate NC Data Access Variable No.
    'Set max. workpiece value parameter (Section No. = 10318, Sub-section No. = 2898)
    lSectionNum = 10318
    lSubSectionNum = 2898

    'Set write data type
    'Set long integer type (4-byte integer type)
    lDataType = T_LONG

    'Write data to the NC Card
    dwStatus = melWriteData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, SetValue, lDataType)
```

```
    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SetWorkLimit")

End Sub

Private Sub txtWorkCount_KeyPress(KeyAscii As Integer)
    Select Case KeyAscii
        Case KEY_ENTER
            Call SetWorkCount(Val(txtWorkCount.Text))
            SendKeys "{TAB}"
        Case KEY_ESC
            SendKeys Chr$(&H1A)
    End Select

End Sub

Private Sub txtWorkCountM_KeyPress(KeyAscii As Integer)
    Select Case KeyAscii
        Case KEY_ENTER
            Call SetWorkCountM(Val(txtWorkCountM.Text))
            SendKeys "{TAB}"
        Case KEY_ESC
            SendKeys Chr$(&H1A)
    End Select
End Sub

Private Sub txtWorkLimit_KeyPress(KeyAscii As Integer)
    Select Case KeyAscii
        Case KEY_ENTER
            Call SetWorkLimit(Val(txtWorkLimit.Text))
            SendKeys "{TAB}"
        Case KEY_ESC
            SendKeys Chr$(&H1A)
    End Select

End Sub
```

List 2-3 COMMON.BAS code module

```
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API function call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        End

    End If

End Sub
```

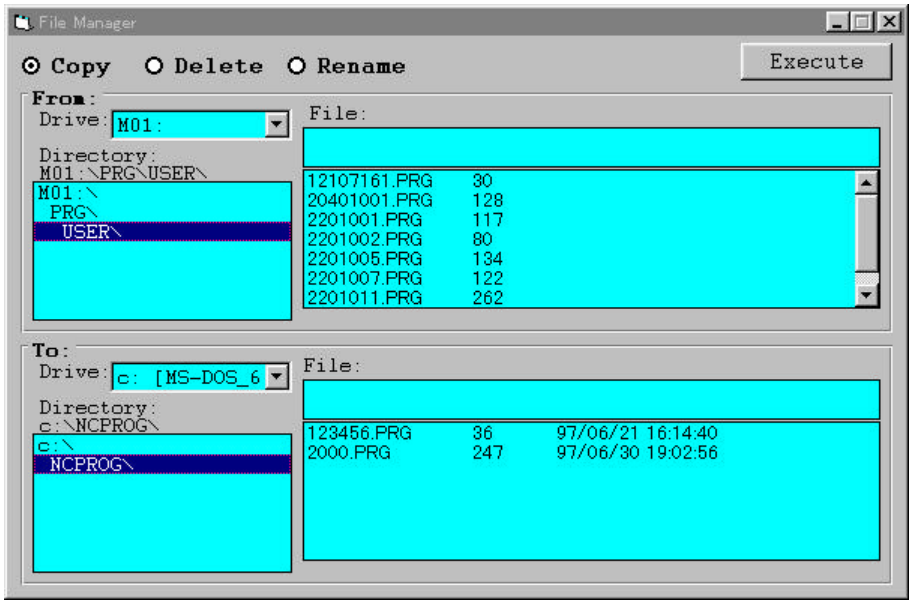
3.2.2.3 File transfer application

What is the file transfer application?

The file transfer application (File Manager) is a tool used to transfer files.
The File Manager has the following three functions

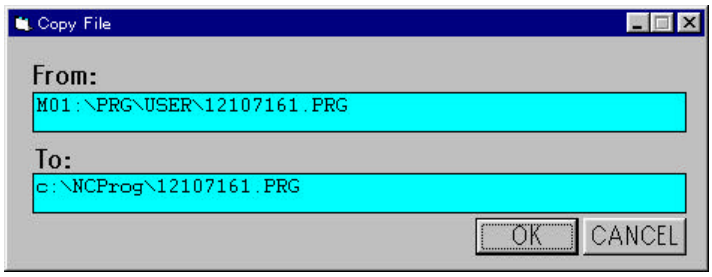
Copying of files (Copy)
Deleting of files (Delete)
Renaming of files (Rename)

The files that can be handled are the files in each drive of the personal computer and the files in the NC Card. With the file copy function, files can be copied between personal computers, between the personal computer and NC Card, and between NC Cards.



How to use the Copy function

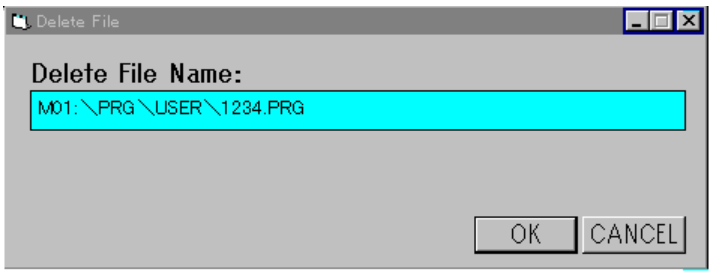
Step 1: Select [Copy] with the option button on the window.
Step 2: Select the copy source file with the [Form:] frame. To select the file, select the drive, directory and file with each [Drive:], [Directory:] and [File:] item.
Step 3: Select the copy designation file with the [To:] frame.
Step 4: Click the [Execute] button.
Step 5: A confirmation screen will appear, so confirm the operation details.



Step 6: If any corrections are required, click the text box and correct the file name, etc.
Step 7: Click [OK] button to execute the operation.
Click [CANCEL] button to cancel the operation.

How to use the Delete function

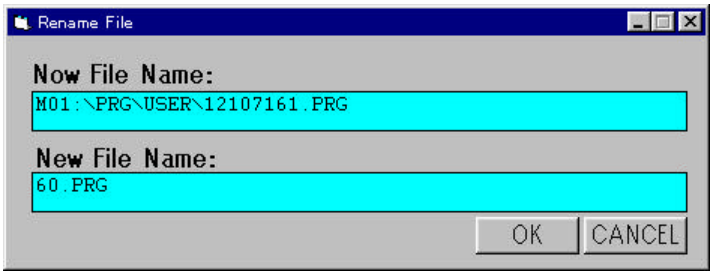
Step 1: Select [Delete] with the option button on the window.
Step 2: Select the file to be deleted . To select the file, select the drive, directory and file with each [Drive:], [Directory:] and [File:] item.
Step 3: Click the [Execute] button.
Step 4: A confirmation screen will appear, so confirm the operation details.



Step 5: If any corrections are required, click the text box and correct the file name, etc.
Step 6: Click [OK] button to execute the operation.
Click [CANCEL] button to cancel the operation.

How to use the Rename function

Step 1: Select [Rename] with the option button on the window.
Step 2: Select the file to be renamed. To select the file, select the drive, directory and file with each [Drive:], [Directory:] and [File:] item.
Step 3: Click the [Execute] button.
Step 4: A confirmation screen will appear, so confirm the operation details.



Step 5: Input the name of the file after the corrects in the [New File Name:] text box.
Step 6: If any corrections are required, click the text box and correct the file name, etc.
Step 7: Click [OK] button to execute the operation.
Click [CANCEL] button to cancel the operation.

Custom API Functions used

melGetDriveList
melOpenDirectory
melReadDirectory
melCloseDirectory
melCopyFile
melDeleteFile
melRenameFile

Creation of file transfer application

Getting of file information

With this application, the following procedure is created to get the personal computer and NC Card file information. These procedures are Function procedures that return each file information as String type (character string type) return values.

Procedures created to get file information:

GetDriveList
GetDirectoryList
GetFileList

```
'Get NC drive list
Private Function GetDriveList() As String
    Dim DriveList As String
    Dim lBuffSize As Long
    Dim dwStatus As Long

    'Secure drive list storage area
    lBuffSize = 256
    DriveList = String$(lBuffSize, 0)

    'Get drive list using API function
    dwStatus = melGetDriveList(Me.hWnd, DriveList, lBuffSize)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")

    'Return drive list
    GetDriveList = DriveList

End Function
```

In the GetDriveList, the Custom API Function melGetDriveList is called, and the list of the NC Card mounted in the personal computer is gotten as the drive list. When calling melGetData in GetDriveList, the storage variable (DriveList) and storage area size (lBuffSize) are transferred to store the gotten drive list.

It must be noted here that when transferring a variable length type such as the area for storing the Custom API Function return value, an area the size of the storage area must be secured before the Custom API Function is called. The Visual Basic String$ function is used to secure an area that is the size of the storage area.

```
'Secure drive list storage area
    lBuffSize = 256
    DriveList = String$(lBuffSize,0)
```

When melGetDriveList is called, the drive list will be stored in the variable DriveList. To identify the drive names, CR (ASCII code &h0D) and LF (ASCII code &h0A) are inserted.

```
'Get directory list
Private Function GetDirectoryList(ByVal DirectoryPath As String) As String
    Dim DirectoryList As String
    Dim DirectoryName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '****************************
    'Open directory
    '****************************
    'Set directory open method
    'Bit16 = ON(Designate directory information)
    lFileType = &H10000

    'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDirectoryList")

    'Save directory ID
    lDirectoryID = dwStatus

    '*****************************
    'Get directory list
    '*****************************
    Do
        'Secure directory name storage area
        lBuffSize = 256
        DirectoryName = String$(lBuffSize, " ")

        'Get directory name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, DirectoryName, lBuffSize)

        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of directory list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add directory name to directory list
        DirectoryList = DirectoryList + Trim$(DirectoryName) + Chr$(CR)
    Loop

    '****************************
    'Close directory
    '****************************
```

```
 'Close directory using API function
 dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

 'Check API function call for errors
 Call APIErrorCheck(dwStatus, "GetDriveList")


 'Check melReadDirectory call for errors
 Call APIErrorCheck(dwMelReadDirectoryStatus, "GetDriveList")

 '****************************
 'Return directory list
 '****************************
 GetDirectoryList = DirectoryList

End Function
```

In GetDirectoryList, the Custom API Function melOpenDirectory, melReadDirectory and melCloseDirectory are called, and the directory list that exists in the directory designated with argument DirectoryPath is gotten.

melOpenDirectory is a Custom API Function used to open the directory to be listed. When calling melOpenDirectory, the variable where the name of the directory to be opened is stored (DirectoryPath) and the variable that designates the method for opening the directory (lFileType) are transferred. The name of the directory to be opened is designated with an absolute path that includes the drive name. To open the directory, bit 16 is turned ON and the directory list retrieval is designated. If the opening of the directory succeeds, melOpenDirectory returns the directory ID. This directory ID is required to call melReadDirectory and melCloseDirectory. The directory ID is saved in a variable called lDirectoryID for the melGetDirectoryList.

```
                :
                :
    'Set directory open method
    'Bit16 = ON(Designate directory information)
    lFileType = &H10000

    'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)
                :
                :
    'Save directory ID
    lDirectoryID = dwStatus
                :
                :
```

melReadDirectory is a Custom API Function that gets one line (one directory) of the directory list at a time. When calling melReadDirectory, the storage variable (Directory Name) and storage area size (lBuffSize) are transferred to store the directory ID (lDirectoryID) and gotten directory list. melReadDirectory saves one line of the directory list in the designated variable (DirectoryName), and returns the stored No. of characters as the function return value. To get all directory lists, call melReadDirectory until the return value reaches 0.

```
                :
                :
  Do
                :
                :
         'Get directory name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, DirectoryName, lBuffSize)
                :
                :
         'Confirm end of directory list data
        If dwStatus = 0 Then
```

```
        Exit Do
    End If
            :
            :
Loop
      :
      :
```

melCloseDirectory is a Custom API Function that closes the directory opened with melOpenDirectory. When calling melCloseDirectory, the directory ID (lDirectoryID) is transferred.
Other directories cannot be opened while the directory is opened using melOpenDirectory. Thus, when a directory has been opened, always close it with melCloseDirectory. The following type of process is executed in GetDirectoryList.
Normally when API function call is executed, the APIErrorCheck procedure is called. In this procedure a check is made for errors. If an error has occurred, the application is quit. If this procedure is called while an error is occurring in the loop that gets the directory list, the application will be quit without closing the directory. Thus, if an error occurs in the loop that gets the directory list, the error code will be saved in the variable (dwMelReadDirectoryStatus), and the loop will be forcibly quit. After the loop is quit and the directory is closed, APIErrorCheck will be called, the error message will be displayed and the application will be quit.

```
'***********************************
'Get directory list
'***********************************
Do
            :
            :
    'Get directory name using API function
    dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, DirectoryName, lBuffSize)

    'Check API function call for errors
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Save status
        dwMelReadDirectoryStatus = dwStatus

        'Forcibly quit loop
        Exit Do
    End If
            :
            :
Loop

'***********************************
'Close directory
'***********************************
'Close directory using API function
dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

'Check API function call for errors
Call APIErrorCheck(dwStatus, "GetDriveList")

'Check melReadDirectory call for errors
Call APIErrorCheck(dwMelReadDirectoryStatus, "GetDriveList")
            :
            :
```

```
'Get file list
Private Function GetFileList(ByVal DirectoryPath As String) As String
    Dim FileList As String
    Dim FileName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '****************************
    'Open directory
    '****************************
    'Set directory open method
    'Bit16 = OFF(Designate file information)
    'Bit2 = ON(Designate comment available)
    'Bit1 = ON(Designate date available)
    'Bit0 = ON(Designate size available)
    lFileType = &H7

     'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Save directory ID
    lDirectoryID = dwStatus

    '*******************************
    'Get file list
    '*******************************
    Do
        'Secure file name storage area
        lBuffSize = 256
        FileName = String$(lBuffSize, " ")

        'Get file name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, FileName, lBuffSize)

        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of file list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add file name to file list
        FileList = FileList + Trim$(FileName) + Chr$(CR)
    Loop

    '****************************
    'Close directory
    '****************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)
    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Check melReadDirectory call for errors
```

```
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetFileList")

    '****************************
    'Return file list
    '****************************
    GetFileList = FileList

  End Function
```

In GetFileList, the Custom API Functions melOpenDirectory, melReadDirectory and melCloseDirectory are called, and the file list that exists in the directory designated with argument DirectoryPath is gotten.
The process details are the same as GetDirectoryList. The only difference is the method for designating the directory opening method. With GetFileList, bit 16 is turned OFF and the file information is designated to get the file list. Furthermore, bit2, bit1, and bit 0 are turned ON to get the comment, date and size information.

```
            :
            :
  'Set directory open method
    'Bit16 = OFF(Designate file information)
    'Bit2 = ON(Designate comment available)
    'Bit1 = ON(Designate date available)
    'Bit0 = ON(Designate size available)
    lFileType = &H7
            :
```

Copying of files

With this application, a Copy File procedure is created to execute copying of files. This procedure is a sub-procedure and does not have return value.

```
 'Copy file
Private Sub CopyFile(SrcFile As String, DstFile As String)
  Dim dwStatus As Long

  'Copy file using API function
  dwStatus = melCopyFile(Me.hWnd, SrcFile, DstFile)

  'Check API function call for errors
  Call APIErrorCheck(dwStatus, "CopyFile")

End Sub
```

In CopyFile, the Custom API Function melCopyFile is called and the file is copied. When calling melCopyFile, the variable where the copy source file name is stored (SrcFile) and the variable where the copy destination file name is stored (DstFile) are transferred. The copy source file name and copy destination file name are both designated with absolute paths including the drive name.

Deleting of files

With this application, a Deletefile procedure is created to execute deleting of files. This procedure is a sub-procedure and does not have return value.

```
 'Delete file
Private Sub DeleteFile(SrcFile As String)
   Dim dwStatus As Long

   'Delete file using API function
   dwStatus = melDeleteFile(Me.hWnd, SrcFile)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "DeleteFile")

End Sub
```

In DeleteFile, the Custom API Function melDeleteFile is called and the file is deleted. When calling melDeleteFile, the variable where the name of the file to be deleted is stored (SrcFile) is transferred. The file name is designated with an absolute paths including the drive name.

Renaming of file

With this application, a RenameFile procedure is created to execute renaming of files. This procedure is a sub-procedure and does not have return value.

```
 'Change file name
Private Sub RenameFile(SrcFile As String, DstFile As String)
   Dim dwStatus As Long

   'Rename file using API function
   dwStatus = melRenameFile(Me.hWnd, SrcFile, DstFile)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "RenameFile")

End Sub
```

In RenameFile, the Custom API Function melRenameFile is called and the file is renamed. When calling melRenameFile, the variable where the name of the file to be renamed is stored (SrcFile) and the variable where renamed file is to be stored (DstFile) are transferred. The rename source file name is designated with an absolute path including the drive name and the rename destination file is designated only as the file name not including the directory path.

List 3-1 FILEMAN32.VBP project file

```
Form=Frmfilem.frm
Module=filemcom; Filemcom.bas
Module=common; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Form=Frmconfi.frm
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard
OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT
SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Object={B16553C3-06DB-101B-85B2-0000C009BE81}#1.0#0; SPIN32.OCX
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
ProjWinSize=80,296,243,273
ProjWinShow=2
IconForm="frmFileManager"
HelpFile=""
ExeName32="fileman32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="IJSEC"
```

List 3-2 FRMFILEM.FRM form module

```
VERSION 4.00
Begin VB.Form frmFileManager
   Appearance      =   0  'Flat
   BackColor       =   &H00C0C0C0&
   BorderStyle     =   1  'Fixed (solid line)
   Caption         =   "File Manager"
   ClientHeight    =   5580
   ClientLeft      =   120
   ClientTop       =   1365
   ClientWidth     =   8955
   ForeColor       =   &H80000008&
   Height          =   5985
   Left            =   60
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   ScaleHeight     =   5580
   ScaleWidth      =   8955
   Top             =   1020
   Width           =   9075
```

```
Begin VB.CommandButton cmdExecute
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "Execute"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   12
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height          =   375
    Left            =   7320
    TabIndex        =   11
    Top             =   60
    Width           =   1515
  End
  Begin VB.OptionButton optCmdSelect
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "Rename"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   300
    Index           =   2
    Left            =   2775
    TabIndex        =   2
    Top             =   150
    Width           =   1275
  End
  Begin VB.OptionButton optCmdSelect
    Appearance      =   0  'Falt
    BackColor       =   &H00C0C0C0&
    Caption         =   "Delete"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   300
    Index           =   1
    Left            =   1350
    TabIndex        =   1
    Top             =   150
    Width           =   1305
  End
  Begin VB.OptionButton optCmdSelect
```

```
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "Copy"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   12
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   300
    Index           =   0
    Left            =   120
    TabIndex        =   0
    Top             =   150
    Width           =   1155
  End
  Begin Threed.SSFrame frm3dFile
    Height          =   2475
    Index           =   1
    Left            =   120
    TabIndex        =   18
    Top             =   3000
    Width           =   8715
    _Version        =   65536
    _ExtentX        =   15372
    _ExtentY        =   4366
    _StockProps     =   14
    ForeColor       =   0
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Begin VB.TextBox txtFile
      Appearance      =   0  'Flat
      BackColor       =   &H00FFFF00&
      BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
        Name            =   "MS Mincho"
        Size            =   9.75
        Charset         =   128
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
      EndProperty
      Height          =   405
      Index           =   1
      Left            =   2820
      TabIndex        =   9
      Top             =   420
      Width           =   5775
    End
    Begin VB.ComboBox cmbDrive
      Appearance      =   0  'Flat
```

```
    BackColor       =   &H00FFFF00&
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   300
    Index           =   1
    Left            =   900
    Style           =   2  'Drop down list
    TabIndex        =   7
    Top             =   240
    Width           =   1815
  End
  Begin VB.ListBox lstDirectory
    Appearance      =   0  'Flat
    BackColor       =   &H00FFFF00&
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height          =   1395
    Index           =   1
    Left            =   120
    TabIndex        =   8
    Top             =   960
    Width           =   2595
  End
  Begin VB.ListBox lstFile
    Appearance      =   0  'Flat
    BackColor       =   &H00FFFF00&
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "MS Mincho"
      Size           =   9.75
      Charset        =   128
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height          =   1395
    Index           =   1
    Left            =   2820
    Sorted          =   -1  'True
    TabIndex        =   10
    Top             =   840
    Width           =   5775
  End
  Begin VB.Label Label1
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
```

```
    Caption        =   "Drive:"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H80000008&
    Height         =   195
    Index          =   1
    Left           =   180
    TabIndex       =   22
    Top            =   240
    Width          =   735
  End
  Begin VB.Label Label2
    Appearance     =   0  'Flat
    BackColor      =   &H00C0C0C0&
    Caption        =   "Directory:"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H80000008&
    Height         =   195
    Index          =   1
    Left           =   180
    TabIndex       =   21
    Top            =   600
    Width          =   1455
  End
  Begin VB.Label Label3
    Appearance     =   0  'Flat
    BackColor      =   &H00C0C0C0&
    Caption        =   "File:"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H80000008&
    Height         =   195
    Index          =   1
    Left           =   2820
    TabIndex       =   20
    Top            =   180
    Width          =   1455
  End
  Begin VB.Label lblCurrentDirectory
    Appearance     =   0  'Flat
```

```
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
        Name            =   "Courier"
        Size            =   9.75
        Charset         =   0
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   195
      Index           =   1
      Left            =   180
      TabIndex        =   19
      Top             =   780
      Width           =   2235
    End
  End
  Begin VB.DriveListBox drvPCDrive
    Appearance      =   0  'Flat
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height          =   315
    Left            =   5160
    TabIndex        =   17
    Top             =   60
    Visible         =   0   'False
    Width           =   1095
  End
  Begin Threed.SSFrame frm3dFile
    Height          =   2475
    Index           =   0
    Left            =   120
    TabIndex        =   12
    Top             =   480
    Width           =   8715
    _Version        =   65536
    _ExtentX        =   15372
    _ExtentY        =   4366
    _StockProps     =   14
    ForeColor       =   0
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Begin VB.TextBox txtFile
      Appearance      =   0  'Flat
      BackColor       =   &H00FFFF00&
```

```
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "MS P Gothic"
      Size           =   9.75
      Charset        =   128
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   405
    Index          =   0
    Left           =   2820
    TabIndex       =   5
    Top            =   420
    Width          =   5775
  End
  Begin VB.ListBox lstFile
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font  {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "MS Gothic"
      Size           =   9.75
      Charset        =   128
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   1395
    Index          =   0
    Left           =   2820
    Sorted         =   -1  'True
    TabIndex       =   6
    Top            =   840
    Width          =   5775
  End
  Begin VB.ListBox lstDirectory
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   1395
    Index          =   0
    Left           =   120
    TabIndex       =   4
    Top            =   960
    Width          =   2595
  End
  Begin VB.ComboBox cmbDrive
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name           =   "Courier"
      Size           =   9.75
      Charset        =   0
```

```
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   300
    Index           =   0
    Left            =   900
    Style           =   2  'Drop down list
    TabIndex        =   3
    Top             =   240
    Width           =   1815
  End
  Begin VB.Label lblCurrentDirectory
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   195
    Index           =   0
    Left            =   180
    TabIndex        =   16
    Top             =   780
    Width           =   2235
  End
  Begin VB.Label Label3
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "File:"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   195
    Index           =   0
    Left            =   2880
    TabIndex        =   15
    Top             =   180
    Width           =   1455
  End
  Begin VB.Label Label2
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "Directory:"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier"
```

```
          Size           =   9.75
          Charset        =   0
          Weight         =   400
          Underline      =   0   'False
          Italic         =   0   'False
          Strikethrough  =   0   'False
        EndProperty
        ForeColor      =   &H80000008&
        Height         =   195
        Index          =   0
        Left           =   180
        TabIndex       =   14
        Top            =   600
        Width          =   1455
      End
      Begin VB.Label Label1
        Appearance     =   0  'Flat
        BackColor      =   &H80000005&
        BackStyle      =   0  'Transparent
        Caption        =   "Drive:"
        BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
          Name           =   "Courier"
          Size           =   9.75
          Charset        =   0
          Weight         =   400
          Underline      =   0   'False
          Italic         =   0   'False
          Strikethrough  =   0   'False
        EndProperty
        ForeColor      =   &H80000008&
        Height         =   195
        Index          =   0
        Left           =   180
        TabIndex       =   13
        Top            =   240
        Width          =   735
      End
    End
End
Attribute VB_Name = "frmFileManager"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

'Variable used to operate directory list
Dim DirLevel(2) As Integer

Private Sub ChangeCurrentDirectory(Index As Integer)
    Dim nLoop As Integer
    Dim DirPath As String

    If lstDirectory(Index).ListIndex <= DirLevel(Index) Then
        'Select high-order directory from current directory
        'Recreate current directory path
        For nLoop = 0 To lstDirectory(Index).ListIndex
            DirPath = DirPath + Trim(lstDirectory(Index).List(nLoop))
        Next
    Else
        'Select low-order directory from current directory
        'Add to current directory path
        DirPath = CurrentDirectory(Index)
        DirPath = DirPath + Trim(lstDirectory(Index).List(lstDirectory(Index).ListIndex))
    End If
```

```
    'Update current directory path
    CurrentDirectory(Index) = DirPath

    'Update directory list
    Call RefreshDirectoryList(Index)

    'Update file list
    Call RefreshFileList(Index)

End Sub

Private Sub cmbDrive_Click(Index As Integer)
    Dim DriveName As String

    '**********************************
    'Update current directory
    '**********************************
    DriveName = cmbDrive(Index).List(cmbDrive(Index).ListIndex)
    DriveName = Left$(DriveName, InStr(DriveName, ":"))
    CurrentDirectory(Index) = DriveName + "\"

    'Update directory list
    Call RefreshDirectoryList(Index)

End Sub

Private Sub cmdExecute_Click()

    frmConfirm.Show 1

    'Update file list
    Call RefreshFileList(0)
    Call RefreshFileList(1)


End Sub

Private Sub Form_Load()
    Dim i As Integer

    'Arrange window in center
    Me.Top = (Screen.Height / 2) - (Me.Height / 2)
    Me.Left = (Screen.Width / 2) - (Me.Width / 2)

    'Initialize drive list
    Call RefreshDriveList(0)
    Call RefreshDriveList(1)

    'Initialize directory list
    Call RefreshDirectoryList(0)
    Call RefreshDirectoryList(1)

    'Initialize file list
    Call RefreshFileList(0)
    Call RefreshFileList(1)

    'Select default command
    optCmdSelect(0).Value = True

End Sub

Private Sub FrameControl(Index As Integer, status As Integer)
```

```
    frm3dFile(Index).Enabled = status
    Label1(Index).Enabled = status
    Label2(Index).Enabled = status
    Label3(Index).Enabled = status

    lblCurrentDirectory(Index).Enabled = status
    cmbDrive(Index).Enabled = status
    lstDirectory(Index).Enabled = status
    lstFile(Index).Enabled = status

End Sub

'Get directory list
Private Function GetDirectoryList(ByVal DirectoryPath As String) As String
    Dim DirectoryList As String
    Dim DirectoryName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '****************************
    'Open directory
    '****************************
    'Set directory open method
    'Bit16 = ON(Designate directory information)
    lFileType = &H10000

    'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDirectoryList")

    'Save directory ID
    lDirectoryID = dwStatus

    '*******************************
    'Get directory list
    '*******************************
    Do
        'Secure directory name storage area
        lBuffSize = 256
        DirectoryName = String$(lBuffSize, " ")

        'Get directory name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, DirectoryName, lBuffSize)

        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of directory list data
        If dwStatus = 0 Then
            Exit Do
        End If
```

```
        'Add directory name to directory list
        DirectoryList = DirectoryList + Trim$(DirectoryName) + Chr$(CR)
    Loop

    '******************************
    'Close directory
    '******************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")


    'Check melReadDirectory call for errors
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetDriveList")

    '******************************
    'Return directory list
    '******************************
    GetDirectoryList = DirectoryList

End Function

'Get NC drive list
Private Function GetDriveList() As String
    Dim DriveList As String
    Dim lBuffSize As Long
    Dim dwStatus As Long

    'Secure drive list storage area
    lBuffSize = 256
    DriveList = String$(lBuffSize, 0)

    'Get drive list using API function
    dwStatus = melGetDriveList(Me.hWnd, DriveList, lBuffSize)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")

    'Return drive list
    GetDriveList = DriveList

End Function

'Get drive list
Private Function GetFileList(ByVal DirectoryPath As String) As String
    Dim FileList As String
    Dim FileName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long


    '******************************
    'Open directory
    '******************************
    'Set directory open method
    'Bit16 = OFF(Designate file information)
    'Bit2 = ON(Designate comment available)
```

```
    'Bit1 = ON(Designate date available)
    'Bit0 = ON(Designate size available)
    lFileType = &H7


    'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Save directory ID
    lDirectoryID = dwStatus

    '*******************************
    'Get fiel list
    '*******************************
    Do
        'Secure file name storage area
        lBuffSize = 256
        FileName = String$(lBuffSize, " ")

        'Get file name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, FileName, lBuffSize)

        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of file list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add file name to file list
        FileList = FileList + Trim$(FileName) + Chr$(CR)
    Loop

    '****************************
    'Close directory
    '****************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Check melReadDirectory call for errors
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetFileList")

    '****************************
    'Return file list
    '****************************
    GetFileList = FileList

End Function
```

```
Private Sub lstDirectory_DblClick(Index As Integer)
   'Change current directory
   Call ChangeCurrentDirectory(Index)
End Sub

Private Sub lstFile_Click(Index As Integer)
   Dim FileName As String

   'Update selected file name
   FileName = lstFile(Index).List(lstFile(Index).ListIndex)
   FileName = Left$(FileName, InStr(FileName, Chr$(TB)) - 1)
   txtFile(Index).Text = FileName

End Sub

Private Sub optCmdSelect_Click(Index As Integer)
   'Delete frame name
   frm3dFile(0).Caption = ""
   frm3dFile(1).Caption = ""

   Select Case Index
      Case CMD_COPY
         'Set frame name
         frm3dFile(0).Caption = "From:"
         frm3dFile(1).Caption = "To:"

         'Set frame setting validity state
         Call FrameControl(0, True)
         Call FrameControl(1, True)

         'Register command selection state
         CmdSelect = CMD_COPY
      Case CMD_DELETE
         'Set frame setting validity state
         Call FrameControl(0, True)
         Call FrameControl(1, False)

         'Register command selection state
         CmdSelect = CMD_DELETE
      Case CMD_RENAME
         'Set frame setting validity state
         Call FrameControl(0, True)
         Call FrameControl(1, False)

         'Register command selection state
         CmdSelect = CMD_RENAME
   End Select
End Sub

Private Sub RefreshDirectoryList(Index As Integer)
   Dim DirectoryName As String
   Dim nChrStart As Integer
   Dim nChrEnd As Integer
   Dim DriveName As String
   Dim DirName As String
   Dim DirList As String

   'Delete contents of list
   lstDirectory(Index).Clear

   '*************************************
   'Insert directory path in list
   '*************************************
```

```
    nChrStart = 1
    nChrEnd = 1
    DirLevel(Index) = 0
    Do
        nChrEnd = InStr(nChrStart, CurrentDirectory(Index), "\")
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add directory name to list
        nChrEnd = nChrEnd + 1 '"\"is included
        DirName = Mid$(CurrentDirectory(Index), nChrStart, nChrEnd - nChrStart)
        DirName = String$(DirLevel(Index), " ") + DirName
        lstDirectory(Index).AddItem DirName

        nChrStart = nChrEnd
        DirLevel(Index) = DirLevel(Index) + 1
    Loop

    '*****************************************************************
    'Insert directory list of current directory in list
    '*****************************************************************
    'Get directory list of current directory
    DirList = GetDirectoryList(CurrentDirectory(Index))

    'Add directory list to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, DirList, Chr$(CR))
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add directory name to list
        DirName = Mid$(DirList, nChrStart, nChrEnd - nChrStart - 1) + "\"
        If DirName = ".\" Or DirName = "..\" Then
        Else
            DirName = String$(DirLevel(Index), " ") + DirName
            lstDirectory(Index).AddItem DirName
        End If
        nChrStart = nChrEnd + 1
    Loop

    lstDirectory(Index).ListIndex = DirLevel(Index) - 1

    lblCurrentDirectory(Index).Caption = CurrentDirectory(Index)
End Sub

'Update drive list contents
Private Sub RefreshDriveList(Index As Integer)
    Dim nLoop As Integer
    Dim DriveList As String
    Dim nChrStart, nChrEnd As Integer
    Dim DriveName As String

    'Delete contents of list
    cmbDrive(Index).Clear

    '*******************************
    'Insert NC drive name in list
    '*******************************
```

```
    'Get NC drive name in list
    DriveList = GetDriveList()

    'Add drive list to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, DriveList, Chr$(CR))
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add drive name to list
        DriveName = Mid$(DriveList, nChrStart, nChrEnd - nChrStart)
        cmbDrive(Index).AddItem DriveName

        nChrStart = nChrEnd + 1
    Loop

    '********************************
    'Insert PC drive name list
    '********************************
    For nLoop = 0 To drvPCDrive.ListCount - 1
        cmbDrive(Index).AddItem drvPCDrive.List(nLoop)
    Next

    '********************************
    'Update current directory
    '********************************
    DriveName = cmbDrive(Index).List(0)
    DriveName = Left$(DriveName, InStr(DriveName, ":"))
    CurrentDirectory(Index) = DriveName + "\"

    '********************************
    'Select default drive
    '********************************
    cmbDrive(Index).ListIndex = 0


End Sub

Private Sub RefreshFileList(Index As Integer)
    Dim DirectoryName As String
    Dim nChrStart As Integer
    Dim nChrEnd As Integer
    Dim FileName As String
    Dim FileList As String

    'Delete contents of list
    lstFile(Index).Clear

    '*****************************************************************
    'Insert file list of current directory in list
    '*****************************************************************
    'Get file list of current directory
    FileList = GetFileList(CurrentDirectory(Index))

    'Add file list to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, FileList, Chr$(CR))
```

```
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add file name to list
        FileName = Mid$(FileList, nChrStart, nChrEnd - nChrStart - 1)
        If FileName = "." Or FileName = ".." Then
        Else
            lstFile(Index).AddItem FileName
        End If
        nChrStart = nChrEnd + 1
    Loop


    'Invalidate selected file name
    txtFile(Index).Text = ""

End Sub
```

List 3-3 FRMCONFI.FRM form module

```
VERSION 4.00
Begin VB.Form frmConfirm
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  BorderStyle     =   1  'Fixed (solid line)
  Caption         =   "ConfirmFile"
  ClientHeight    =   2292
  ClientLeft      =   492
  ClientTop       =   2208
  ClientWidth     =   6972
  ForeColor       =   &H80000008&
  Height          =   2676
  Left            =   444
  LinkTopic       =   "Form1"
  LockControls    =   -1  'True
  MaxButton       =   0   'False
  ScaleHeight     =   2292
  ScaleWidth      =   6972
  Top             =   1872
  Width           =   7068
  Begin VB.CommandButton cmdCancel
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    Caption         =   "CANCEL"
    BeginProperty Font
      name            =   "System"
      charset         =   128
      weight          =   400
      size            =   13.2
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    Height          =   375
    Left            =   5760
    TabIndex        =   1
    Top             =   1800
    Width           =   1035
  End
```

```
Begin VB.CommandButton cmdOk
  Appearance      =   0  'Flat
  BackColor       =   &H80000005&
  Caption         =   "OK"
  BeginProperty Font
    name          =   "System"
    charset       =   128
    weight        =   400
    size          =   13.2
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  EndProperty
  Height          =   375
  Left            =   4680
  TabIndex        =   0
  Top             =   1800
  Width           =   1035
End
Begin VB.TextBox txtDstFile
  Appearance      =   0  'Flat
  BackColor       =   &H00FFFF00&
BeginPropertyFont
    name          =   "Courier"
    charset       =   0
    weight        =   400
    size          =   9.6
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  Height          =   405
  Left            =   240
  TabIndex        =   2
  Top             =   540
  Width           =   6555
End
Begin VB.TextBox txtSrcFile
  Appearance      =   0  'Flat
  BackColor       =   &H00FFFF00&
BeginPropertyFont
    name          =   "Courier"
    charset       =   0
    weight        =   400
    size          =   9.6
    underline     =   0   'False
    italic        =   0   'False
    strikethrough =   0   'False
  Height          =   405
  Left            =   240
  TabIndex        =   2
  Top             =   540
  Width           =   6555
End
Begin VB.Label lblDst
  Appearance      =   0  'Flat
  BackColor       =   &H80000005&
  BackStyle       =   0  'Transparent
  Caption         =   "To:"
  BeginProperty Font
    name          =   "System"
    charset       =   128
    weight        =   700
```

```
        size          =   13.5
        underline      =   0   'False
        italic         =   0   'False
        strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   285
      Left            =   270
      TabIndex        =   5
      Top             =   1080
      Width           =   3435
    End
    Begin VB.Label lblSrc
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "From:"
      BeginProperty Font
        name          =   "System"
        charset       =   0
        weight        =   700
        size          =   9.6
        underline      =   0   'False
        italic         =   0   'False
        strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   240
      TabIndex        =   4
      Top             =   240
      Width           =   3135
    End
  End
Attribute VB_Name = "frmConfirm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdCansel_Click()
   Unload Me
End Sub

Private Sub cmdOk_Click()
   Dim SrcFileName As String
   Dim DstFileName As String

   'Create source file name
   SrcFileName = Trim$(txtSrcFile.Text)
   'Create destination file name
   DstFileName = Trim$(txtDstFile.Text)

   'Branch process with selected command
   Select Case CmdSelect
      Case CMD_COPY
         'Copy file
         Call CopyFile(SrcFileName, DstFileName)

      Case CMD_DELETE
         'Delete file
         Call DeleteFile(SrcFileName)
```

```
        Case CMD_RENAME
            'Rename file
            Call RenameFile(SrcFileName, DstFileName)

    End Select

    'Delete window
    Unload Me

End Sub

'Copy file
Private Sub CopyFile(SrcFile As String, DstFile As String)
    Dim dwStatus As Long

    'Copy file using API function
    dwStatus = melCopyFile(Me.hWnd, SrcFile, DstFile)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "CopyFile")

End Sub

'Delete file
Private Sub DeleteFile(SrcFile As String)
    Dim dwStatus As Long

    'Delete file using API function
    dwStatus = melDeleteFile(Me.hWnd, SrcFile)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "DeleteFile")


End Sub

Private Sub Form_Load()
    Dim SrcFileName As String
    Dim DstFileName As String

    'Arrange window in center
    Me.Top = (Screen.Height / 2) - (Me.Height / 2)
    Me.Left = (Screen.Width / 2) - (Me.Width / 2)

    'Branch process with selected command
    Select Case CmdSelect
        Case CMD_COPY
            'Display window title
            frmConfirm.Caption = "Copy File"

            'Display title
            lblSrc.Caption = "From:"
            lblDst.Caption = "To:"

            'Create source file name
            SrcFileName = Trim$(frmFileManager.txtFile(0))
            SrcFileName = CurrentDirectory(0) + SrcFileName
            'Create destination file name
            DstFileName = Trim$(frmFileManager.txtFile(1))
            If DstFileName = "" Then
                DstFileName = Trim$(frmFileManager.txtFile(0))
            End If
```

```
            DstFileName = CurrentDirectory(1) + DstFileName

            'Display file name
            txtSrcFile.Text = SrcFileName
            txtDstFile.Text = DstFileName

            lblSrc.Visible = True
            txtSrcFile.Visible = True
            lblDst.Visible = True
            txtDstFile.Visible = True

        Case CMD_DELETE
            'Display window title
            frmConfirm.Caption = "Delete File"

            'Display title
            lblSrc.Caption = "Delete File Name:"
            lblDst.Caption = ""

            'Create file name
            SrcFileName = Trim$(frmFileManager.txtFile(0))
            SrcFileName = CurrentDirectory(0) + SrcFileName

            'Display file name
            txtSrcFile.Text = SrcFileName
            txtDstFile.Text = ""

            lblSrc.Visible = True
            txtSrcFile.Visible = True
            lblDst.Visible = False
            txtDstFile.Visible = False

        Case CMD_RENAME
            'Display window title
            frmConfirm.Caption = "Rename File"

            'Display title
            lblSrc.Caption = "Now File Name:"
            lblDst.Caption = "New File Name:"

            'Create file name
            SrcFileName = Trim$(frmFileManager.txtFile(0))
            SrcFileName = CurrentDirectory(0) + SrcFileName

            'Display file name
            txtSrcFile.Text = SrcFileName
            txtDstFile.Text = ""

            lblSrc.Visible = True
            txtSrcFile.Visible = True
            lblDst.Visible = True
            txtDstFile.Visible = True

        End Select

    End Sub

    'Change file name
    Private Sub RenameFile(SrcFile As String, DstFile As String)
        Dim dwStatus As Long

        'Rename file using API function
```

```
    dwStatus = melRenameFile(Me.hWnd, SrcFile, DstFile)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "RenameFile")

End Sub
```

List 3-4 FILEMCOM.BAS code module

```
Atrribute VB_Name="filemcom"
Option Explicit

'Commonly used constants
Global Const CR = &HD
Global Const LF = &HA
Global Const TB = &H9

Global Const CMD_COPY = 0
Global Const CMD_DELETE = 1
Global Const CMD_RENAME = 2

'Commonly used variables
Global CurrentDirectory(2)  As String
Global CmdSelect As Integer
```

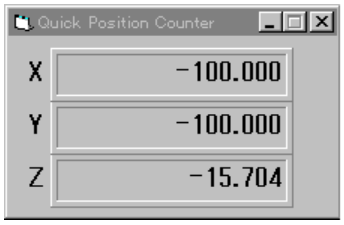
List 3-5 COMMON.BAS code module

```
Atrribute VB_Name="common"
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API function call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        End

    End If

End Sub
```

3.2.2.4 Improvement of counter display application

What is the improvement of counter display application?

In this section, the counter display application (Position Counter) created in 3.2.2.1 Counter display application is improved, and a high-speed counter display application (Quick Position Counter) is created. The high-speed counter display application executes the counter display applications' counter updating process at a high speed. The details displayed on the high-speed counter display application window are the same as for the counter display application.



Custom API Functions used

melRegisterModal
melReadModal
melCancelModal

Creation of high-speed counter display application

Changing of custom API Functions used

In the counter display application, the custom API Function melReadData is used to get the current position of each axis from the NC Card. In the high-speed counter display application, the custom API Functions melRegisterModal, melReadModal and melCancelModal for high-speed reading of the data are used to instead of melReadData.

High-speed reading of data

The three phases registration of high-speed read-out data, high-speed reading of data execution and canceling registration of high-speed read-out data are used for high-speed reading of data. Each of these phases corresponds to melRegisterModal, melReadModal and melCancelModal. In the registration of high-speed read-out data, the No. of the NC Data Access Variable for the data to be read at a high-speed is registered in the NC Card, and the registration No. (this is called the data ID) is gotten. When the NC Data Access Variable as registered for the high-speed data reading, the NC Card constantly creates a variable value, and makes preparations so that the custom application can be gotten immediately.
In the high-speed reading of data execution phase, the variable value constantly prepared by the NC Card is gotten based on the data ID.
In the canceling registration of high-speed read-out data phase, the No. of the NC Data Access Variable registered in the NC Card is deleted, and the data ID is invalidated.
In this manner, with the high-speed reading of data, the NC Card has already prepared the data value when reading the data from the custom application, so the data can be read at a high speed than when reading the data with melReadData.

Reading of data at high speeds

With this application, the following procedure is created to read the data at a high speed. Of these procedures, RegistAxisPosition and CancelAxisPosition are Sub-procedures. And ReadAxisPosition is a Function procedure that returns the current position of the axis having the axis No. designated with the argument. The current position returned by ReadAxisPosition is a Double type (double precision real number type).

Procedure created to execute high-speed reading of data

RegistAxisPosition
ReadAxisPosition
CancelAxisPosition

To read the data at a high speed, a global variable (ModalIDActive, ModalID) is created to save the data ID, and a procedure InitModalID is created to initialize this variable.

```
'Modal ID validity flag
Dim ModalIDActive(1 To 3)  As Integer

'Modal ID
Dim ModalID(1 To 3) As Long
'InitModalID is called from the form load event (Form_Load).
```

InitModalID is called from the form load event (Form_Load). The procedures RegistAxisPosition, ReadAxisPosition and CancelAxisPosition, the global variables ModalIDActive and ModalID are set and referred to directly.

```
'Current position getting variable for designated axis as modal read
'The axis designation is 1 origin
Private Sub RegistAxisPosition(ByVal iAxisNum As Integer)
   Dim dwStatus As Long                          'Variable to get return value from API function

   Dim lAddress As Long                          'Variable to designate address
   Dim lSectionNum, lSubSectionNum As Long       'Variable to designate NC data Access Variable No.
   Dim lPriority As Long                         'Variable to designate priority

   Dim Message As String

   'Nothing will occur if modal is registered
   If ModalIDActive(iAxisNum) = True Then
      Exit Sub
   End If

   'Set address of data to be read
   'NC Card No. = 1, system designation = No. 1 system, axis No. designation = 1~
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1) Or ADR_AXIS(iAxisNum)

   'Set NC Data Access Variable No.
   'Set current position data (Section No. = 21, Sub-section No. 20032)
   lSectionNum = 21
   lSubSectionNum = 20032

   'Set priority
   'Designate 1 (highest)
   lPriority = 1

   'Register current position getting variable as modal read
   dwStatus = melRegisterModal(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lPriority)

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "RegistAxisPosition")

   'Register modal ID
   ModalID(iAxisNum) = dwStatus

   'Validate modal ID valid flag
   ModalIDActive(iAxisNum) = True

End Sub
```

In RegistAxisPosition, the Custom API Function melRegisterModal is called, and the NC Data Access Variable for getting the current position data of the axis designated with the argument is registered for high-speed reading. When calling melRegisterModal, the current position data for

each axis is designated using the address (lAddress) and NC Data Access Variables (lSectionNum, lSubSectionNum). The priority of the data (lPriority) is transferred.
The argument addresses and NC Data Access Variables for melRegisterModal are the same as melReadData. Refer to section "3.2.2.1 Counter display application" for details on these.
The priority of the data refers to the priority when there are several NC Data Access Variables registered for high-speed reading. The order of the priority includes 1:highest, 2:high, 3:middle and 4:lowest. The higher the priority is, the shorter the cycle for creating the variable value in the NC Card will be. However, this order of priority is a relative priority between the registered variables, so if many variables with a high order of priority are registered, the general cycle will be longer. Thus, an appropriate priority must be assigned when registering multiple variables.
When the registration of the high-speed reading of data is successful, melRegisterModal will return the data ID. A maximum of 256 data IDs can be gotten per NC Card. However, depending on the type of variable registered, this may be lower than 256. The No. of registerable IDs will be affected by the default data type of the variable being registered.

```
Get current position for designated axis
Private Function ReadAxisPosition(iAxisNum As Integer) As Double
    Dim dwStatus As Long            'Variable to get return value from API function

    Dim lAddress As Long            'Variable to designate address
    Dim dReadData As Double         'Variable to store read data
    Dim lReadType As Long           'Variable to designate requested data type

    Dim Message As String

    'Nothing will occur if modal is not registered
    If ModalIDActive(iAxisNum) = False Then
        Exit Function
    End If

    'Set address of data to be read
    'NC Card No. = 1
    lAddress = ADR_MACHINE(1)

    'Set read data type
    'The double precision floating type (8-byte floating type)
    lReadType = T_DOUBLE

     'Read current position data from the NC Card
    dwStatus = melReadModal(Me.hWnd, lAddress, ModalID(iAxisNum), dReadData, lReadType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "ReadAxisPosition")

    'Return read current position
    ReadAxisPosition = dReadData


End Function
```

In ReadAxisPosition, the Custom API Function melReadModal is called, and the current position data of the axis designated with the argument is read. When calling melReadModal, the current position data for each axis is designated with address (lAddress) and data ID (lModalID). The storage variable (dReadData) and requested data type (lReadType) are transferred to store the gotten current position. The details of the melReadModal argument addresses and requested data types are the same as melReadData. Refer to the section "3.2.2.1 Counter display application" for details on these.

```
'Cancel modal read registration for designated axis
Private Sub CancelAxisPosition(iAxisNum As Integer)
    Dim dwStatus As Long

    Dim lAddress As Long            'Variable for designating address
    Dim Message As String
```

```
    'Nothing will occur if modal is not registered
    If ModalIDActive(iAxisNum) = False Then
        Exit Sub
    End If

    'Set address of data to be read
    'NC Card No. = 1
    lAddress = ADR_MACHINE(1)

    'Cancel modal read registration
    dwStatus = melCancelModal(Me.hWnd, lAddress, ModalID(iAxisNum))

    'Call API function for errors
    Call APIErrorCheck(dwStatus, "CancelAxisPosition")

    'Invalidate modal ID valid flag
    ModalIDActive(iAxisNum) = False

End Sub
```

In CancelAxisPosition, the Custom API Function melCancelModal is called, and the high-speed reading of the current position data for the axis designated with the argument is canceled. When melCancelModal is called, the current position data for each axis is designated with address (lAddress) and data ID (lModalID). The argument address and requested data types for melCancelModal are the same as melReadData. Refer to section "3.2.2.1 Counter display application" for details on these.

How to use procedure created for high-speed reading of data

The three procedures created to execute high-speed reading of data are used in this application, so the GetAxisPosition procedure and APIErrorCheck procedure created for the counter display application are corrected and a code is added to the Form_Unload event.

```
'Get current position for designated axis
'The axis designation is 1 origin
Private Function GetAxisPosition(ByVal iAxisNum As Integer) As Double
    Dim dReadData As Double          'Variable for storing read data

    'Register as modal read
    RegistAxisPosition (iAxisNum)

    'Read current position data from NC Card
    dReadData = ReadAxisPosition(iAxisNum)

    'Return read current position
    GetAxisPosition = dReadData

End Function
```

With GetAxisPosition, RegisterAxisPosition and ReadAxisPosition are called. Here both procedures are called unconditionally, but as a check is made for multiple registrations in each procedure, high-speed data reading registration is not done in duplicate.

```
'Form unload event process
Private Sub Form_Unload(Cancel As Integer)
    Dim iAxisNum As Integer          'Loop Counter

    'Cancel modal read registration for 3 axes
    For iAxisNum = 0 To 2
        'Cancel modal read registration
        Call CancelAxisPosition(iAxisNum + 1)
    Next

End Sub
```

With Form_Unload, CancelAxisPosition is called. As form_Unload is the event procedure executed when the window is closed, the registration of high-speed read-out data will be canceled in this event.
Form_Unload is an event executed when closing the window, but there are cases when this event is not generated. Such cases include when there is an End statement in the application. In the counter display application, the End statement is used in the APIErrorCheck procedure. Thus in this application the End statement is not used, and instead the Unload statement is used. A condition for this method is that the APIErrorCheck procedure is not called during Form_Load event. If APIErrorCheck is called during the Form_load event, and the Unload statement is to be executed, a Visual Basic error will occur.

```
'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        Unload frmPosit

    End If

End Sub
```

List 4-1 QCOUNTER32.VBP project file

```
Form=Frmposit.frm
Module=COMMON; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard
OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT
SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={0842D103-1E19-101B-9AAF-1A1626551E7C}#1.0#0; GRAPH32.OCX
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={C932BA88-4374-101B-A56C-00AA003668DC}#1.0#0; MSMASK32.OCX
Object={27395F88-0C0C-101B-A3C9-08002B2F49FB}#1.0#0; PICCLP32.OCX
Object={C1A8AF28-1257-101B-8FB0-0020AF039CA3}#1.0#0; MCI32.OCX
Object={20C62CAE-15DA-101B-B9A8-444553540000}#1.0#0; MSMAPI32.OCX
Object={A8B3B723-0B5A-101B-B22E-00AA0037B2FC}#1.0#0; GRID32.OCX
Object={BE4F3AC8-AEC9-101A-947B-00DD010F7B46}#1.0#0; MSOUTL32.OCX
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMCTL32.OCX
ProjWinSize=81,406,243,244
ProjWinShow=2
IconForm="frmPosit"
HelpFile=""
Title="QCOUNTER"
ExeName32="Qcounter32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
```

List 4-2 FRMPOSIT.FRM form module

```
VERSION 4.00
Begin VB.Form frmPosit
   Appearance      =   0  'Flat
   BackColor       =   &H00C0C0C0&
   BorderStyle     =   1  'Fixed (solid line)
   Caption         =   "Quick Position Counter"
   ClientHeight    =   1776
   ClientLeft      =   2340
   ClientTop       =   2268
   ClientWidth     =   3288
   BeginProperty Font
      name           =   "System"
      charset        =   128
      weight         =   400
      size           =   13.5
      underline      =   0   'False
```

```
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   2160
    Left            =   2292
    LinkTopic       =   "Form1"
    LockControls    =   -1  'True
    MaxButton       =   0   'False
    ScaleHeight     =   1776
    ScaleWidth      =   3288
    Top             =   1932
    Width           =   3384
    Begin VB.Timer timPosition
      Interval        =   100
      Left            =   2880
      Top             =   1320
    End
    Begin Threed.SSPanel Pnl3dAxis
      Height          =   540
      Index           =   0
      Left            =   420
      TabIndex        =   3
      Top             =   108
      Width           =   2424
      _Version        =   65536
      _ExtentX        =   4276
      _ExtentY        =   953
      _StockProps     =   15
      BevelInner      =   1
      Begin VB.Label lblAxisPosition
        Alignment       =   1  'Flush right
        Caption         =   "-1234.1234"
        BeginProperty Font
          name            =   "System"
          charset         =   128
          weight          =   700
          size            =   13.2
          underline       =   0   'False
          italic          =   0   'False
          strikethrough   =   0   'False
        EndProperty
        Height          =   330
        Index           =   0
        Left            =   105
        TabIndex        =   4
        Top             =   105
        Width           =   2220
      End
    End
    Begin Threed.SSPanel Pnl3dAxis
      Height          =   540
      Index           =   1
      Left            =   420
      TabIndex        =   5
      Top             =   636
      Width           =   2424
      _Version        =   65536
      _ExtentX        =   4276
      _ExtentY        =   953
      _StockProps     =   15
      BevelInner      =   1
```

```
    Begin VB.Label lblAxisPosition
      Alignment       =   1  'Flush right
      Caption         =   "-1234.1234"
      BeginProperty Font
        name            =   "System"
        charset         =   128
        weight          =   700
        size            =   13.2
        underline       =   0   'False
        italic          =   0   'False
        strikethrough   =   0   'False
      EndProperty
      Height          =   330
      Index           =   1
      Left            =   105
      TabIndex        =   6
      Top             =   105
      Width           =   2220
    End
  End
  Begin Threed.SSPanel Pnl3dAxis
    Height          =   540
    Index           =   2
    Left            =   420
    TabIndex        =   7
    Top             =   1152
    Width           =   2424
    _Version        =   65536
    _ExtentX        =   4276
    _ExtentY        =   953
    _StockProps     =   15
    BevelInner      =   1
    Begin VB.Label lblAxisPosition
      Alignment       =   1  'Flush right
      Caption         =   "-1234.1234"
      BeginProperty Font
        name            =   "System"
        charset         =   128
        weight          =   700
        size            =   13.2
        underline       =   0   'False
        italic          =   0   'False
        strikethrough   =   0   'False
      EndProperty
      Height          =   330
      Index           =   2
      Left            =   105
      TabIndex        =   8
      Top             =   105
      Width           =   2220
    End
  End
  Begin VB.Label lblAxisName
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "Z"
    BeginProperty Font
      name            =   "System"
      charset         =   128
      weight          =   700
      size            =   13.2
```

```
        underline      =   0   'False
        italic         =   0   'False
        strikethrough  =   0   'False
      EndProperty
      ForeColor      =   &H80000008&
      Height         =   255
      Index          =   2
      Left           =   210
      TabIndex       =   2
      Top            =   1260
      Width          =   255
    End
    Begin VB.Label lblAxisName
      Appearance     =   0  'Flat
      BackColor      =   &H80000005&
      BackStyle      =   0  'Transparent
      Caption        =   "Y"
      BeginProperty Font
        name         =   "System"
        charset      =   128
        weight       =   700
        size         =   13.2
        underline      =   0   'False
        italic         =   0   'False
        strikethrough  =   0   'False
      EndProperty
      ForeColor      =   &H80000008&
      Height         =   255
      Index          =   1
      Left           =   210
      TabIndex       =   1
      Top            =   735
      Width          =   255
    End
    Begin VB.Label lblAxisName
      Appearance     =   0  'Flat
      BackColor      =   &H80000005&
      BackStyle      =   0  'Transparent
      Caption        =   "X"
      BeginProperty Font
        name         =   "System"
        charset      =   128
        weight       =   700
        size         =   13.2
        underline      =   0   'False
        italic         =   0   'False
        strikethrough  =   0   'False
      EndProperty
      ForeColor      =   &H80000008&
      Height         =   255
      Index          =   0
      Left           =   210
      TabIndex       =   0
      Top            =   210
      Width          =   255
    End
  End
      Attribute VB_Name = "frmPosit"
      Attribute VB_Creatable = False
      Attribute VB_Exposed = False

      Option Explicit
```

```
'Modal ID valid flag
Dim ModalIDActive(1 To 3)  As Integer

'Modal ID
Dim ModalID(1 To 3) As Long


'Cancel modal read registration for designated axis
Private Sub CancelAxisPosition(iAxisNum As Integer)
   Dim dwStatus As Long

   Dim lAddress As Long                        'Variable for designating address
   Dim Message As String

   'Nothing will occur if modal is not registered
   If ModalIDActive(iAxisNum) = False Then
      Exit Sub
   End If

   'Set address of data to be read
   'NC Card No. = 1
   lAddress = ADR_MACHINE(1)

   'Cancel modal read registation
   dwStatus = melCancelModal(Me.hWnd, lAddress, ModalID(iAxisNum))

   'Call API function for errors
   Call APIErrorCheck(dwStatus, "CancelAxisPosition")

   'Invalidate modal ID valid flag
   ModalIDActive(iAxisNum) = False

End Sub

'Form load event process
Private Sub Form_Load()
   'Initialize modal ID control information
   Call InitModalID
End Sub

'Form unload event process
Private Sub Form_Unload(Cancel As Integer)
   Dim iAxisNum As Integer                     'Loop Counter

   'Cancel modal read registration for 3 axes
   For iAxisNum = 0 To 2
      'Cancel modal read registration
      Call CancelAxisPosition(iAxisNum + 1)
   Next

End Sub

'Get current position for designated axis
'The axis designation is 1 origin
Private Function GetAxisPosition(ByVal iAxisNum As Integer) As Double
   Dim dReadData As Double                     'Variable for storing read data

   'Register as modal read
   RegistAxisPosition (iAxisNum)

   'Read current position data from NC Card
   dReadData = ReadAxisPosition(iAxisNum)
```

```
    'Return read current position
    GetAxisPosition = dReadData

End Function

'Initialize modal ID control information
Private Sub InitModalID()
    Dim iAxisNum As Integer                    'Loop counter

    'Invalidate modal ID for 3 axes
    For iAxisNum = 0 To 2
        'Clear modal ID
        ModalID(iAxisNum + 1) = 0
        'Invalidate modal ID valid flag
        ModalIDActive(iAxisNum + 1) = False
    Next


End Sub

'Get current position of the designated axis
Private Function ReadAxisPosition(iAxisNum As Integer) As Double
     Dim dwStatus As Long                      'Variable to get return value from API function

    Dim lAddress As Long                       'Variable to designate address
    Dim dReadData As Double                    'Variable to store read data
    Dim lReadType As Long                      'Variable to designate requested data type

    Dim Message As String

    'Nothing will occur if modal is not registered
    If ModalIDActive(iAxisNum) = False Then
        Exit Function
    End If

    'Set address of data to be read
    'NC Card No. = 1
    lAddress = ADR_MACHINE(1)

    'Set read data type
    'The double precision floating type (8-byte floating type)
    lReadType = T_DOUBLE

     'Read current position data from the NC Card
    dwStatus = melReadModal(Me.hWnd, lAddress, ModalID(iAxisNum), dReadData, lReadType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "ReadAxisPosition")

    'Return read current position
    ReadAxisPosition = dReadData


End Function

'Register current position getting variable for designated axis as modal read
'The axis designation is 1 origin
Private Sub RegistAxisPosition(ByVal iAxisNum As Integer)
    Dim dwStatus As Long                           'Variable to get return value from API function

    Dim lAddress As Long                           'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long        'Variable to designate NC data Access Variable No.
```

```
    Dim lPriority As Long                                   'Variable to designate priority

    Dim Message As String

    'Nothing will occur if modal is registered
    If ModalIDActive(iAxisNum) = True Then
        Exit Sub
    End If

    'Set address of data to be read
    'NC Card No. = 1, system designation = No. 1 system, axis No. designation = 1~
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1) Or ADR_AXIS(iAxisNum)

    'Set NC Data Access Variable No.
    'Set current position data (Section No. = 21, Sub-section No. 20032)
    lSectionNum = 21
    lSubSectionNum = 20032

    'Set priority
    'Designate 1 (highest)
    lPriority = 1

    'Register current position getting variable as modal read
    dwStatus = melRegisterModal(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lPriority)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "RegistAxisPosition")

    'Register modal ID
    ModalID(iAxisNum) = dwStatus

    'Validate modal ID valid flag
    ModalIDActive(iAxisNum) = True

End Sub

'Timer process of Position Counter window
Private Sub timPosition_Timer()
    Dim dReadData As Double                 'Variable to store read data
    Dim iAxisNum As Integer                 'Loop counter

    'Get current positions for 3 axes and display on screen
    For iAxisNum = 0 To 2
        'Get current position
        dReadData = GetAxisPosition(iAxisNum + 1)

        'Display read current position on screen
        lblAxisPosition(iAxisNum).Caption = Format$(dReadData, "0.000")
    Next

End Sub
```

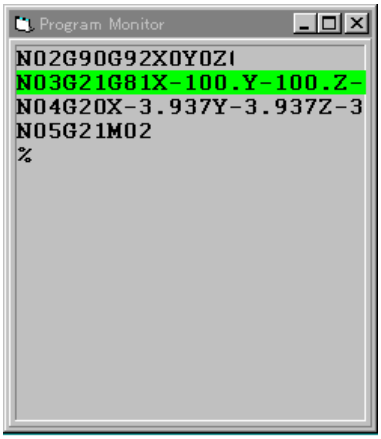
List 4-3 COMMON.BAS code module

```
Attribute VB _Name="Module1"
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
   Dim Message As String
   If RetvIsError(dwStatus) = True Then
      'Error occurrence
      'Display error message
      Message = "Error occurred in API function call"
      Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
      Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
      MsgBox (Message)

      'Quit aplication
      'Stop
      Unload frmPosit

   End If

End Sub
```

3.2.2.5 Program in operation display application

What is the program in operation display application?

The program in operation display application (Program Monitor) is a monitor that displays the machining program being run. A maximum of ten blocks of the machining program being run on the NC Card is displayed on the window. The block displayed and currently being executed is highlighted in green.
The size of the window can be changed.



Custom API Functions used

melGetCurrentPrgBlock

Creation of program in operation display application

In this application, a procedure called GetPrgData is created to get the machining program currently being executed from the NC Card. GetPrgData sets the No. of program blocks designated with the argument into the character string variable designated with the argument. GetPrgData returns the No. of the block currently being executed. The block No. returned by GetPrgData is the value that indicates which block of the gotten program block is executed.

```
Ger program blocks being executed
Private Function GetPrgData(lBlockNos As Long, sPrgData As String) As Long
   Dim dwStatus As Long                          ' Variable to get return value from API function

   Dim lBuffSize As Long                         'Size of program block storage area
   Dim typGetPrgBlock As GETPRGBLOCK             'Array for getting program block

   Dim lAddress As Long                          'Variable for designating address
   Dim lDataType As Long                         'Variable for designating data type

   'Secure program block storage area
   lBuffSize = 256 * lBlockNos
   sPrgData = String$(lBuffSize, 0)

   '*******************************************
   'Initialize array for getting program block
   '*******************************************
   'Set size of program block storage area
   typGetPrgBlock.lPrgDataSize = lBuffSize

  'Pointer of program block area is set using special API function for VB
   typGetPrgBlock.lpszPrgData = sPrgData

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "GetPrgData")

   '*******************************************
   'Get program block
   '*******************************************
   'Set address
   'NC Card No. = 1, system designation = No. 1 system
   lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

   'Set data type
   'Designation getting type of program being executed
   lDataType = T_GETPRGBLOCK

   'Get program block using API function
   dwStatus = melGetCurrentPrgBlock(Me.hWnd, lAddress, lBlockNos, typGetPrgBlock, lDataType)

   'lpszPrgData should be assigned explicitly.
    sPrgData = typGetPrgBlock.lpszPrgData

   'Check API function call for errors
   Call APIErrorCheck(dwStatus, "GetPrgData")

   'Return No. of block being executed
    GetPrgData = typGetPrgBlock.lCurrentBlockNum

End Function
```

With GetPrgBlock, the Custom API Function melGetCurrentPrgBlock is called, and the machining program currently being executed is gotten. When calling melGetCurrentPrgBlock in the GetPrgBlock, the address (lAddress), No. of blocks to be gotten (lBlockNos), variable to store gotten machining program (typGetPrgBlock), and the storage area type (lDataType) are transferred.
The variable that stores the gotten machining program is a user defined array called GETPRGBLOCK, T_GETPRGBLOCK is designated for the storage area type.

```
    :
    :
Dim typGetPrgBlock As GETPRGBLOCK                    'Array for getting program block
    :
    :
'Set data type
'Designate retrieval type of program being executed
lDataType = T_GETPRGBLOCK
    :
    :
```

The user definition type GETPRGBLOCK is declared as shown below in the MELTYPE.BAS code module of the Custom API Library file.

```
'.... Structure of data type for getting of program being executed ...
Type GETPRGBLOCK
   lCurrentBlockNum As Long          '/* Block being executed        */
                                       '/* (Block in gotten data)       */
                                       '/*    0:Not being run          */
                                       '/*    1:1st block              */
                                       '/*    2:2nd block              */
   lPrgDataSize As Long              '/* Buffer size of lpszPrgData     */
   lpszPrgData As Long                 '/* Buffer to store program       */
End Type
```

Before calling the Custom API Function melGetCurrentPrgBlock, the size and memory address of the buffer to store the program (sPrgData) are set in the buffer size (lPrgDataSize) and buffer memory address (lpszPrgData) of the GETPRGBLOCK elements. The area for the buffer that stores the program is secured in the application. Use the Visual Basic String $ function to secure an area that is the size of the storage area.
A problem here is now to set the memory address of the buffer (sPrgData) to store the program. Visual Basic does not use the concept of a pointer of C language so a certain variable memory address cannot be substituted in a different variable.
This problem is not limited to the Custom API Function. It can also occur when calling the WindowsAPI from Visual Basic. Windows API transfers, as an argument, the user-definition type array having a pointer to the character string in the elements. This problem is solved for Visual Basic in the following manner.
When there is a pointer to the character string in the elements of the user-definition type array that will be transferred to the DLL procedure, declare this as the variable-length character string type variable. To set an address in this pointer, substitute for the pointer variable, a character string type variable containing the character string to be transferred to the DLL procedure. When the DLL procedure is called after this, the pointer to the character string is transferred to the DLL procedure. When the called DLL procedure only refers to the character string, the other processes are not required. When the called DLL procedure writes the character string, then after the DLL is called, the character string type variable containing the character string to be transferred to the DLL procedure is substituted into the pointer variable.

An example of calling melGetCurrentPrgBlock is shown below.

```
    :
    :
Dim sPrgData As String                  'Program block storage variable
Dim typGetPrgBlock As GETPRGBLOCK       'Program block retrieve array
    :
    :
'Secure program block storage area
sPrgData=String$(256, 0)
```

```
    :
    :
'Set program block storage area pointer
typGetPrgBlock.lpszPrgData=sPrgData          'Substitute the program block storage variable in the pointer variable
    :
    :
'Retrieve the program block using the API Function
dwStatus=melGetCurrentPrgBlock(Me.hWnd, lAddress, lBlockNos, typGetPrgBlock, lDataType)
    :
    :
'Set the retrieved program block in sPrgData
sPrgData=typGetPrgBlock.lpszPrgData          'Substitute the pointer variable in the program block storage variable
    :
    :
```

With this method, the use of BSTR data type as character string type in Visual Basic can be realized. Refer to the chapter "Calling DLL Procedure" in the Visual Basic Programming Guide.

When the Custom API Function melGetCurrentPrgBlock is called, the designated No. of machining program blocks are stored in the buffer (sPrgData) for storing the program. A value indicating which block of the stored machining program that is being executed is entered in the block being executed (lCurrentBlockNum) of the GETPRGBLOCK element.

List 5-1 PRGMON32.VBP project file

```
Form=Frmprog.frm
Module=SdkCommon; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-
00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON
FILES\MICROSOFT SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={0842D103-1E19-101B-9AAF-1A1626551E7C}#1.0#0; GRAPH32.OCX
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT
VISUAL BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={B16553C3-06DB-101B-85B2-0000C009BE81}#1.0#0; SPIN32.OCX
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMCTL32.OCX
ProjWinSize=81,397,243,228
ProjWinShow=2
IconForm="frmPrgMonitor"
HelpFile=""
Title="PRGMON32"
ExeName32="Prgmon32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
```

List 5-2 FRMPROG.FRM form module

```
VERSION 4.00
Begin VB.Form frmPrgMonitor
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  Caption         =   "Program Monitor"
  ClientHeight    =   3876
  ClientLeft      =   1632
  ClientTop       =   1392
  ClientWidth     =   3612
  BeginProperty Font
    name            =   "System"
    charset         =   128
    weight          =   700
    size            =   13.2
    underline       =   0   'False
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   4260
```

```
Left          =   1584
LinkTopic     =   "Form1"
LockControls  =   -1  'True
ScaleHeight   =   3876
ScaleWidth    =   3612
Top           =   1056
Width         =   3708
Begin VB.PictureBox Picture1
  Appearance     =   0  'Flat
  BackColor      =   &H80000005&
  BeginProperty Font
    name           =   "Courier"
    charset        =   0
    weight         =   400
    size           =   9.6
    underline      =   0   'False
    italic         =   0   'False
    strikethrough  =   0   'False
  EndProperty
  ForeColor      =   &H80000008&
  Height         =   495
  Left           =   240
  ScaleHeight    =   468
  ScaleWidth     =   1128
  TabIndex       =   0
  Top            =   2940
  Visible        =   0   'False
  Width          =   1155
End
Begin VB.Timer Timer1
  Interval       =   200
  Left           =   3150
  Top            =   3360
End
Begin Threed.SSPanel Pnl3dPrgPanel
  Height         =   3795
  Left           =   0
  TabIndex       =   1
  Top            =   0
  Width          =   3615
  _Version       =   65536
  _ExtentX       =   6376
  _ExtentY       =   6694
  _StockProps    =   15
  BevelInner     =   1
  Begin VB.Label lblPrgBlock
    BackStyle      =   0  'Transparent
    BeginProperty Font
      name           =   "Courier"
      charset        =   0
      weight         =   700
      size           =   12
      underline      =   0   'False
      italic         =   0   'False
      strikethrough  =   0   'False
    EndProperty
    Height         =   675
    Left           =   120
    TabIndex       =   3
    Top            =   105
    Width          =   3075
  End
```

```
      Begin VB.Label lblPrgPoint
        BackColor       =   &H0000FF00&
        Height          =   375
        Left            =   70
        TabIndex        =   2
        Top             =   660
        Visible         =   0   'False
        Width           =   3195
      End
    End
End
Attribute VB_Name = "frmPrgMonitor"
Attribute VB_Creatable = False
Attribute VB_Exposed = False

Option Explicit

'Width of 3D panel frame
Const PANEL3D_FRAME_WIDTH = 70

'Form load event process
Private Sub Form_Load()
    Dim i As Integer

    'Arrange window in center
    Me.Top = (Screen.Height / 2) - (Me.Height / 2)
    Me.Left = (Screen.Width / 2) - (Me.Width / 2)

End Sub

'Window resize event process
Private Sub Form_Resize()
    Dim fWidth As Single

    Pnl3DPrgPanel.Top = 0
    Pnl3DPrgPanel.Left = 0

    fWidth = ScaleWidth
    If fWidth < 1 Then
        fWidth = 1
    End If
    Pnl3DPrgPanel.Width = fWidth

    fWidth = ScaleHeight
    If fWidth < 1 Then
        fWidth = 1
    End If
    Pnl3DPrgPanel.Height = fWidth

    fWidth = Pnl3DPrgPanel.Top + PANEL3D_FRAME_WIDTH
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblPrgBlock.Top = fWidth

    fWidth = Pnl3DPrgPanel.Left + PANEL3D_FRAME_WIDTH
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblPrgBlock.Left = fWidth

    fWidth = Pnl3DPrgPanel.Width - PANEL3D_FRAME_WIDTH * 2
```

```
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblPrgBlock.Width = fWidth

    fWidth = Pnl3DPrgPanel.Height - PANEL3D_FRAME_WIDTH * 2
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblPrgBlock.Height = fWidth

    lblPrgPoint.Left = lblPrgBlock.Left
    lblPrgPoint.Width = lblPrgBlock.Width

    'Adjust size of program execution position marker
    Call ResizePrgPoint
End Sub

'Obtain height of one line of text display object
Private Function GetLineHeight(lblObj As Control) As Single

    Picture1.FontBold = lblObj.FontBold
    Picture1.FontItalic = lblObj.FontItalic
    Picture1.FontSize = lblObj.FontSize
    Picture1.FontName = lblObj.FontName

    GetLineHeight = Picture1.TextHeight("A")

End Function

'Obtain the No. of displayable lines in text display object
Private Function GetLineNos(lblObj As Control) As Long
    GetLineNos = Int(lblObj.Height / GetLineHeight(lblObj))
End Function

'Get program blocks being executed
Private Function GetPrgData(lBlockNos As Long, sPrgData As String) As Long
    Dim dwStatus As Long                        'Variable to get return value from API function

    Dim lBuffSize As Long                       'Size of program block storage area
    Dim typGetPrgBlock As GETPRGBLOCK           'Array for getting program block

    Dim lAddress As Long                        'Variable for designating address
    Dim lDataType As Long                       'Variable for designating data type

    'Secure program block storage area
    lBuffSize = 256 * lBlockNos
    sPrgData = String$(lBuffSize, 0)

    '*******************************************
    'Initialize array for getting program block
    '*******************************************
    'Set size of program block storage area
    typGetPrgBlock.lPrgDataSize = lBuffSize

    'pointer of program block area is set using special API function for VB
    typGetPrgBlock.lpszPrgData = sPrgData

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetPrgData")
```

```
    '*****************************************
    'Get program block
    '*****************************************
    'Set address
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Set data type
    'Designation getting type of program being executed
    lDataType = T_GETPRGBLOCK

    'Get program block using API function
    dwStatus = melGetCurrentPrgBlock(Me.hWnd, lAddress, lBlockNos, typGetPrgBlock, lDataType)

    'lpszPrgData should be assigned explicitly.
     sPrgData = typGetPrgBlock.lpszPrgData

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetPrgData")

    'Return No. of block being executed
     GetPrgData = typGetPrgBlock.lCurrentBlockNum

End Function

'Move program execution position marker
Private Sub MovePrgPoint(lBlockNum As Long)
    Dim Offset As Single

    If lBlockNum = 0 Then
        'Erase marker
        lblPrgPoint.Visible = False

    Else
        'Obtain marker position
        Offset = GetLineHeight(lblPrgBlock) * (lBlockNum - 1)

        'Move marker
        lblPrgPoint.Top = lblPrgBlock.Top + Offset

        'Display marker
        lblPrgPoint.Visible = True

    End If
End Sub

'Adjust size of program execution position marker
Private Sub ResizePrgPoint()
    'Align to height of one line in program display area
    lblPrgPoint.Height = GetLineHeight(lblPrgBlock)
End Sub

'Timer event process
Private Sub Timer1_Timer()
    Dim lBlockNos As Long
    Dim PrgBlock As String
    Dim lCurrentBlockNum As Long

    'Obtain No. of lines displayable in display area
    lBlockNos = GetLineNos(lblPrgBlock)
```

```
    'Set range of No. of diaplayed program blocks between 1 and 10
    If lBlockNos < 1 Then
        lBlockNos = 1
    ElseIf lBlockNos > 10 Then
        lBlockNos = 10
    End If

    'Get program being executed
    lCurrentBlockNum = GetPrgData(lBlockNos, PrgBlock)

    'Display program being executed
    lblPrgBlock.Caption = PrgBlock

    'Move program execution position marker
    Call MovePrgPoint(lCurrentBlockNum)

End Sub
```

List 5-3 COMMON.BAS code module

```
Attribute VB_Name="Sdk Common"
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API function call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        End

    End If

End Sub
```

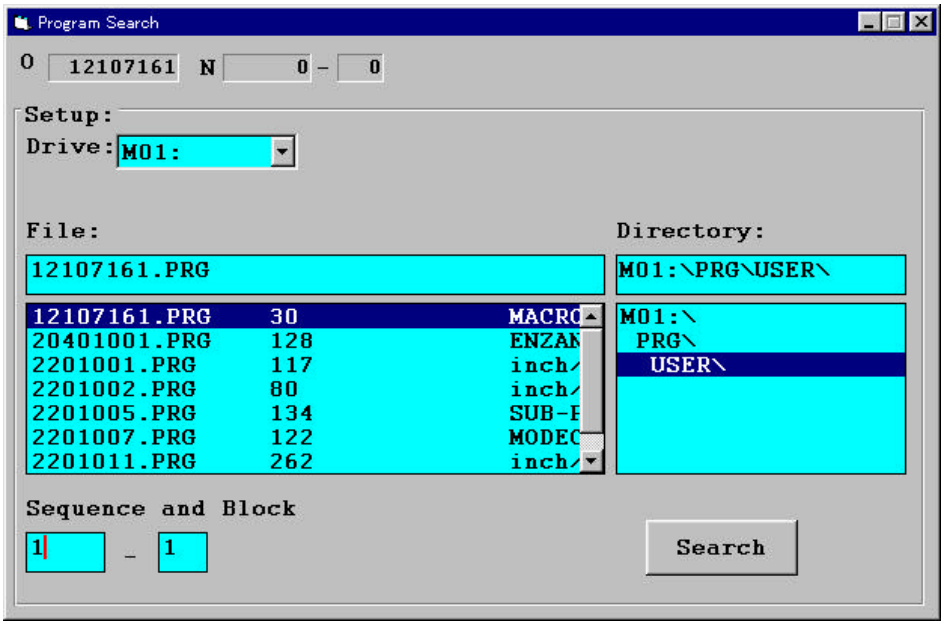
3.2.2.6 Operation search application

What is the operation search application?

The operation search application (Program Search) is a tool that searches the operation.
The No. of the program, sequence and block currently being searched will display in the upper part of the window.

Operation Process

To execute operation search, the program to be search is selected with the [Setup:] frame, and the [Search] button is clicked. To select the program to be searched, select the NC Card or personal compuoter side drive (hard disk, etc.) with the [Drive:] list box, double-click on the directory containing the machining program in the [Directory:] list box, click the No. of program to be searched from the [File:] list, and display the program No. in the text box above [File:]. (A separate option is required to seach for a file on the personal computer side.) To search a sequence No., input the sequence No. and program No. in the two text boxes of [Sequence and Block] below the [Setup:] frame.



Custom API Functions used

melGetDriveList
melOpenDirectory
melReadDirectory
melCloseDirectory
melSelectExecPrg

Creation of operation search application

Getting of file information

With this application, the following procedures are created to get the machining program file information in the NC Card. These procedures are Function procedures that return the file information as String type (character string type) return values.

Procedures created to get file information

GetDriveList
GetFileList

```
'Get NC drive list
Private Function GetDriveList() As String
    Dim DriveList As String
    Dim lBuffSize As Long
    Dim dwStatus As Long

    'Secure drive list storage area
    lBuffSize = 256
    DriveList = String$(lBuffSize, 0)

    'Get drive list using API function
    dwStatus = melGetDriveList(Me.hWnd, DriveList, lBuffSize)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")

    'Return drive list
    GetDriveList = DriveList

End Function


'Get file list
Private Function GetFileList(ByVal DirectoryPath As String) As String
    Dim FileList As String
    Dim FileName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '****************************
    'Open directory
    '****************************
    'Set directory open method
    'Bit16 = OFF(Designate file information)
    'Bit2 = ON(Designate comment available)
    'Bit1 = ON(Designate date available)
    'Bit0 = ON(Designate size available)
    lFileType = &H7

     'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Save directory ID
    lDirectoryID = dwStatus

    '*******************************
     'Get fiel list
    '*******************************
    Do
        'Secure file name storage area
        lBuffSize = 256
        FileName = String$(lBuffSize, " ")

        'Get file name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, FileName, lBuffSize)
```

```
        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of file list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add file name to file list
        FileList = FileList + Trim$(FileName) + Chr$(CR)
    Loop

    '****************************
     'Close directory
    '****************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")


    'Check melReadDirectory call for errors
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetFileList")

    '****************************
    'Return file list
    '****************************
    GetFileList = FileList

End Function
```

The Custom API Functions melGetDriveList, melOpenDirectory, melReadDirectory and melCloseDirectory are used by these procedures. The details of GetDriveList and GetFileList are approximately the same as those created in the "3.2.2.3 File transfer application", so the explanation will be omitted here.

How to execute operation search

A procedure called SearchPrg is created with this application to execute operation search. SearchPrg will search the machining program, sequence No. and block No. having the file name designated with path (only file name on NC side) with the argument. This procedure is a sub-procedure and does not have a return value.

```
'Execute operation search
Private Sub SearchPrg(FileName As String, lSequenceNum As Long, lBlockNum As Long)
    Dim dwStatus As Long                    'Variable to get return value from API function

    Dim typString As STRINGTYPE             'Array for character string data type
    Dim lAddress As Long                    'Variable for designating address
    Dim lDataType As Long                   'Variable for designating data type
```

```
    '*********************************************
    'Initialize array for character string data type
    '*********************************************
    'Set size of program block storage area
    typString.lBuffSize = Len(FileName)

    'pointer of character string data area is set using special API function for VB
    typString.lpszBuff = FileName

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SearchPrg")

    '************************
    'Execute operation search
    '************************
    'Set address
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Set data type
    'Designation character string type
    lDataType = T_STR

    'Execute operation search using API function
    dwStatus = melSelectExecPrg(Me.hWnd, lAddress, typString, lDataType, lSequenceNum, lBlockNum)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SearchPrg")

End Sub
```

In SearchPrg, the Custom API Function melSelectExecPrg is called, and the operation search is executed. When calling melSelectExecPrg in SearchPrg, the address (lAddress), variable where name of machining program to be searched is stored (typString), storage area type (lDataType), sequence No. (lSequenceNum) and block No. (lBlockNum) are transferred.
The variable that stores the name of the machining program to be searched is a character string type (T_STR) variable supplied in the Custom API Library. The Custom API Library character string type variable is a user defined type array called STRINGTYPE. T_STR is designated for the storage area type.

```
    :
    :
Dim typString As STRINGTYPE                    'Array for character string data type
    :
    :
'Set data type
'Designate character string type
lDataType = T_STR
    :
    :
```

The user definition type of STRINGTYPE is declared as shown below in the MELTYPE.BAS code module of the Custom API Library file.

```
'....Structure of character string type....
Type STRINGTYPE
    lBuffSize As Long          '/* Data area size     */
    lpszBuff As String          '/* Data area pointer  */
End Type
```

Before calling the Custom API Function melSelectExecPrg, the size and memory address of the buffer for storing the machining program name (FileName) are set in the data area size (lBuffSize)

and data area memory address(lpszBuff) of the STRINGTYPE elements. The method for setting the memory address of the buffer for storing the part program name (FileName) in the data area memory address (lpszBuff) of the STRINGTYPE element is explained in section "3.2.2.5 Program in operation display application ".

List 6-1 PRGSRCH32.VBP project file

```
Form=Frmsrch.frm
Module=COMMON; Common.bas
Module=SRCHCOM; Srchcom.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-
00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON
FILES\MICROSOFT SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={B16553C3-06DB-101B-85B2-0000C009BE81}#1.0#0; SPIN32.OCX
Object={0842D103-1E19-101B-9AAF-1A1626551E7C}#1.0#0; GRAPH32.OCX
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.0#0; COMCTL32.OCX
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMCTL32.OCX
ProjWinSize=77,776,242,230
ProjWinShow=2
IconForm="frmPrgSearch"
HelpFile=""
Title="PRGSRCH"
ExeName32="Prgsrch32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="MITSUBISHI ELECTRIC CORPORATION"
```

List 6-2 FRMSRCH.FRM form module

```
VERSION 4.00
Begin VB.Form frmPrgSearch
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  BorderStyle     =   1  'Fixed (solid line)
  Caption         =   "Program Search"
  ClientHeight    =   5760
  ClientLeft      =   1500
  ClientTop       =   3396
  ClientWidth     =   9252
  ForeColor       =   &H80000008&
  Height          =   6144
  Left            =   1452
  LinkTopic       =   "Form1"
  LockControls    =   -1  'True
  MaxButton       =   0   'False
  ScaleHeight     =   5760
  ScaleWidth      =   9252
  Top             =   3060
  Width           =   9348
  Begin VB.DriveListBox DrvLstBxDrive
    Appearance      =   0  'Flat
    Height          =   390
    Left            =   6960
    TabIndex        =   20
    Top             =   120
    Visible         =   0   'False
    Width           =   855
  End
  Begin VB.Timer Timer1
    Interval        =   500
    Left            =   5280
    Top             =   120
  End
  Begin Threed.SSPanel Pnl3DNowBlockNum
    Height          =   372
    Left            =   3240
    TabIndex        =   13
    Top             =   120
    Width           =   552
    _Version        =   65536
    _ExtentX        =   979
    _ExtentY        =   661
    _StockProps     =   15
    Caption         =   "00"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   9.6
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    BevelOuter      =   0
    BevelInner      =   1
    Alignment       =   4
  End
  Begin Threed.SSPanel Pnl3DNowSequenceNum
    Height          =   372
    Left            =   2100
    TabIndex        =   12
```

```
    Top           =   120
    Width         =   972
    _Version      =   65536
    _ExtentX      =   1720
    _ExtentY      =   661
    _StockProps   =   15
    Caption       =   "12345"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      name        =   "Courier"
      charset     =   0
      weight      =   700
      size        =   9.6
      underline   =   0   'False
      italic      =   0   'False
      strikethrough   =   0   'False
    EndProperty
    BevelOuter    =   0
    BevelInner    =   1
    Alignment     =   4
  End
  Begin Threed.SSPanel Pnl3DNowPrgNum
    Height        =   372
    Left          =   360
    TabIndex      =   11
    Top           =   120
    Width         =   1392
    _Version      =   65536
    _ExtentX      =   2461
    _ExtentY      =   661
    _StockProps   =   15
    Caption       =   "12345678"
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      name        =   "Courier"
      charset     =   0
      weight      =   700
      size        =   9.6
      underline   =   0   'False
      italic      =   0   'False
      strikethrough   =   0   'False
    EndProperty
    BevelOuter    =   0
    BevelInner    =   1
    Alignment     =   4
  End
  Begin Threed.SSFrame frm3dFile
    Height        =   5052
    Left          =   60
    TabIndex      =   3
    Top           =   600
    Width         =   9072
    _Version      =   65536
    _ExtentX      =   16007
    _ExtentY      =   8916
    _StockProps   =   14
    Caption       =   "Setup:"
    ForeColor     =   0
    BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      name        =   "Courier"
      charset     =   0
      weight      =   700
      size        =   12
      underline   =   0   'False
```

```
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  Begin VB.TextBox TxtDirectory
    Appearance      =   0  'Flat
    BackColor       =   &H00FFFF00&
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    Height          =   405
    Left            =   6000
    TabIndex        =   19
    Top             =   1560
    Width           =   2895
  End
  Begin VB.ListBox LstDirectory
    Appearance      =   0  'Flat
    BackColor       =   &H00FFFF00&
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    Height          =   1704
    Index           =   1
    Left            =   6000
    TabIndex        =   18
    Top             =   2040
    Width           =   2892
  End
  Begin VB.CommandButton cmdSearch
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "Search"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    Height          =   555
    Left            =   6300
    TabIndex        =   10
    Top             =   4200
    Width           =   1515
  End
  Begin VB.TextBox txtBlockNum
    Alignment       =   1  'Flush right
    Appearance      =   0  'Flat
```

```
    BackColor       =   &H00FFFF00&
    BeginProperty Font
      name           =   "Courier"
      charset        =   0
      weight         =   700
      size           =   12
      underline      =   0   'False
      italic         =   0   'False
      strikethrough  =   0   'False
    EndProperty
    Height         =   405
    Left           =   1440
    TabIndex       =   8
    Top            =   4320
    Width          =   495
  End
  Begin VB.TextBox txtSequenceNum
    Alignment      =   1  'Flush right
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font
      name           =   "Courier"
      charset        =   0
      weight         =   700
      size           =   12
      underline      =   0   'False
      italic         =   0   'False
      strikethrough  =   0   'False
    EndProperty
    Height         =   405
    Left           =   120
    TabIndex       =   7
    Top            =   4320
    Width          =   795
  End
  Begin VB.TextBox txtFile
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font
      name           =   "Courier"
      charset        =   0
      weight         =   700
      size           =   12
      underline      =   0   'False
      italic         =   0   'False
      strikethrough  =   0   'False
    EndProperty
    Height         =   405
    Left           =   120
    TabIndex       =   1
    Top            =   1560
    Width          =   5775
  End
  Begin VB.ListBox lstFile
    Appearance     =   0  'Flat
    BackColor      =   &H00FFFF00&
    BeginProperty Font
      name           =   "Courier"
      charset        =   0
      weight         =   700
      size           =   12
      underline      =   0   'False
```

```
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    Height          =   1704
    Left            =   120
    Sorted          =   -1  'True
    TabIndex        =   2
    Top             =   2040
    Width           =   5772
  End
  Begin VB.ComboBox cmbDrive
    Appearance      =   0  'Flat
    BackColor       =   &H00FFFF00&
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H00000000&
    Height          =   360
    Left            =   1020
    Style           =   2  'Drop down list
    TabIndex        =   0
    Top             =   360
    Width           =   1815
  End
  Begin VB.Label Label8
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "Directory:"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   6000
    TabIndex        =   17
    Top             =   1200
    Width           =   1455
  End
  Begin VB.Label Label4
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "-"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
```

```
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   1080
    TabIndex        =   9
    Top             =   4440
    Width           =   195
  End
  Begin VB.Label Label3
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "Sequence and Block"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   315
    Left            =   120
    TabIndex        =   6
    Top             =   3960
    Width           =   2730
  End
  Begin VB.Label Label2
    Appearance      =   0  'Flat
    BackColor       =   &H00C0C0C0&
    Caption         =   "File:"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
      italic        =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   120
    TabIndex        =   5
    Top             =   1200
    Width           =   1455
  End
  Begin VB.Label Label1
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "Drive:"
    BeginProperty Font
      name          =   "Courier"
      charset       =   0
      weight        =   700
      size          =   12
      underline     =   0   'False
```

```
        italic          =   0   'False
        strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   360
      Width           =   855
    End
  End
  Begin VB.Label Label7
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "-"
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   700
      size            =   12
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   3060
    TabIndex        =   16
    Top             =   180
    Width           =   195
  End
  Begin VB.Label Label6
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "N"
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   700
      size            =   12
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   1920
    TabIndex        =   15
    Top             =   180
    Width           =   195
  End
  Begin VB.Label Label5
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "O"
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   700
```

```
        size          =   12
        underline      =   0   'False
        italic         =   0   'False
        strikethrough   =   0   'False
      EndPropert
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   120
      TabIndex        =   14
      Top             =   120
      Width           =   195
    End
End
Attribute VB_Name = "frmPrgSearch"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

'Commonly used constants
Const PRGFILE_PATH = "\PRG\USER\"

'Variable used for directory list operation
Dim DirLevel(2) As Integer

'Variable used to store Drive List Index
Dim nBkDrListIndex As Integer

Dim CurrentDirectory(2)  As String

Private Sub ChangeCurrentDirectory(index As Integer)
    Dim nLoop As Integer
    Dim DirPath As String

    If LstDirectory(index).ListIndex <= DirLevel(index) Then
        'Select high-order directory from current directory
        'Recreate current directory path
        For nLoop = 0 To LstDirectory(index).ListIndex
            DirPath = DirPath + Trim(LstDirectory(index).List(nLoop))
        Next
    Else
        'Select low-order directory from current directory
        'Add to current directory path
        DirPath = CurrentDirectory(index)
        DirPath = DirPath + Trim(LstDirectory(index).List(LstDirectory(index).ListIndex))
    End If

    'Update current directory path
    CurrentDirectory(index) = DirPath

    'Update directory list
    RefreshDirectoryList Me, 1

    'Update file list
    RefreshFileList

End Sub

'Click event process of drive list
Private Sub cmbDrive_Click()
    Dim sDriveNm As String

    On Error GoTo Err_Drive
```

```
    'Get Drive
    sDriveNm = cmbDrive.List(cmbDrive.ListIndex)

    'In the case of PC , The selected drive is made as current directory.
    If InStr(1, sDriveNm, ":") = 2 Then     'PC Side drive selected
        CurrentDirectory(1) = UCase$(CurDir$(Left$(sDriveNm, 1)))
        ChDir CurrentDirectory(1)
        If Right$(CurrentDirectory(1), 1) <> "\" Then
            CurrentDirectory(1) = CurrentDirectory(1) + "\"
        End If
    Else                'NC Side drive selected
        sDriveNm = Left$(sDriveNm, InStr(sDriveNm, ":"))
        CurrentDirectory(1) = sDriveNm + PRGFILE_PATH
    End If

    'File List is Updated
    Call RefreshFileList

     RefreshDirectoryList Me, 1

    nBkDrListIndex = cmbDrive.ListIndex

Exit Sub

Err_Drive:
    If (Err = 75) Or (Err = 76) Then
        If Left(CurrentDirectory(1), 1) <> Left(CurDir$, 1) Then
            MsgBox ("The selected drive does not exist")
            cmbDrive.ListIndex = nBkDrListIndex
            Exit Sub
        Else
            CurrentDirectory(1) = CurDir$
            Resume Next
        End If

    ElseIf (Err = 71) Then
        MsgBox ("The selected drive is not ready")
        cmbDrive.ListIndex = nBkDrListIndex
        Exit Sub
    ElseIf (Err = 380) Then
        Resume Next
    Else
        Error Err
    End If

End Sub

'Click event process of search command button
Private Sub cmdSearch_Click()
    Dim FileName As String              'Variable to store file name
    Dim lSequenceNum As Long            'Variable to store sequence No.
    Dim lBlockNum As Long               'Variable to store block No.

    If (txtFile.Text = "") Then
        MsgBox ("Please select a file")
        Exit Sub
    End If

    'The file name is set
    If InStr(1, Trim(CurrentDirectory(1)), ":") = 2 Then  'PC Side drive selected
        FileName = Trim(CurrentDirectory(1)) + Trim$(txtFile.Text)
    Else                'NC Side drive selected
```

```
        FileName = Trim$(txtFile.Text)
    End If

    'Set sequence No.
    lSequenceNum = Val(txtSequenceNum.Text)

    'Set block No.
    lBlockNum = Val(txtBlockNum.Text)

    'Execute operation search
    Call SearchPrg(FileName, lSequenceNum, lBlockNum)

    MsgBox ("Program search is completed")

End Sub


'Form_load event process

Private Sub Form_Load()

    'Arrange window in center
    Me.Top = (Screen.Height / 2) - (Me.Height / 2)
    Me.Left = (Screen.Width / 2) - (Me.Width / 2)

    'Initialize drive list
    Call RefreshDriveList

    'Initialize file list
    Call RefreshFileList

     RefreshDirectoryList Me, 1

End Sub


'Get current block No.

Private Function GetBlockNum() As Integer
    Dim dwStatus As Long                          'Variable to get return value from API function


    Dim lAddress As Long                          'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long       'Variable to designate NC Data Access Variable No.
    Dim nReadData As Integer                      'Variable to store read data
    Dim lReadType As Long                         'Variable to designate requested data type

    'Set address
    'NC Card No. = 1, System designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Designate NC Data Access Variable No.
    'Designate current block No. (Section No. = 13m Sub-section No. = 15)
    lSectionNum = 13
    lSubSectionNum = 15

    'Set read data type
    'Set integer type (2-byte integer type)
    lReadType = T_SHORT

    'Read current block No. from NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, nReadData, lReadType)
```

```
    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetBlockNum")

    'Return current block No.
    GetBlockNum = nReadData

End Function

'Get NC drive list
Private Function GetDriveList() As String
    Dim DriveList As String
    Dim lBuffSize As Long
    Dim dwStatus As Long

    'Secure drive list storage area
    lBuffSize = 256
    DriveList = String$(lBuffSize, 0)

    'Get drive list using API function
    dwStatus = melGetDriveList(Me.hWnd, DriveList, lBuffSize)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")

    'Return drive list
    GetDriveList = DriveList

End Function

'Get file list

Private Function GetFileList(ByVal DirectoryPath As String) As String
    Dim FileList As String
    Dim FileName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '****************************
    'Open directory
    '****************************
    'Set directory open method
    'Bit16 = OFF(Designate file information)
    'Bit2 = ON(Designate comment available)
    'Bit1 = ON(Designate date available)
    'Bit0 = ON(Designate size available)
    lFileType = &H7

     'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetFileList")

    'Save directory ID
    lDirectoryID = dwStatus

    '******************************
     'Get file list
    '******************************
```

```
    Do
        'Secure file name storage area
        lBuffSize = 256
        FileName = String$(lBuffSize, " ")

        'Get file name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, FileName, lBuffSize)

        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of file list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add file name to file list
        FileList = FileList + Trim$(FileName) + Chr$(CR)
    Loop

    '*****************************
     'Close directory
    '*****************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")


    'Check melReadDirectory call for errors
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetFileList")

    '*****************************
    'Return file list
    '*****************************
    GetFileList = FileList

End Function

'Get currently searched program No.

Private Function GetPrgNum() As Long
     Dim dwStatus As Long                          'Variable to get return value from API function

    Dim lAddress As Long                           'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long        'Variable to designate NC Data Access Variable No.
    Dim lReadData As Long                          'Variable to store read data
    Dim lReadType As Long                          'Variable to designate requested data type

    'Set address
    'NC Card No. = 1, System designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Designate NC Data Access Variable No.
```

```
    'Designate current block No. (Section No. = 13, Sub-section No. = 240)
    lSectionNum = 13
    lSubSectionNum = 240

    'Set read data type
    'Set long integer type (4-byte long integer type)
    lReadType = T_LONG

    'Read currently searched program No. from NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lReadType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetPrgNum")

    'Returned currently searched program No.
    GetPrgNum = lReadData

End Function

'Get currently searched sequence No.
Private Function GetSequenceNum() As Long
    Dim dwStatus As Long                        'Variable to get return value from API function

    Dim lAddress As Long                        'Variable to designate address
    Dim lSectionNum, lSubSectionNum As Long     'Variable to designate NC Data Access Variable No.
    Dim lReadData As Long                       'Variable to store read data
    Dim lReadType As Long                       'Variable to designate requested data type

    'Set address
    'NC Card No. = 1, System designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Designate NC Data Access Variable No.
    'Designate current block No. (Section No. = 13, Sub-section No. = 244)
    lSectionNum = 13
    lSubSectionNum = 244

    'Set read data type
    'Set long integer type (4-byte long integer type)
    lReadType = T_LONG


    'Read current sequence No. from NC Card
    dwStatus = melReadData(Me.hWnd, lAddress, lSectionNum, lSubSectionNum, lReadData, lReadType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetSequenceNum")

    'Return current sequence  No.
    GetSequenceNum = lReadData

End Function

Private Sub LstDirectory_DblClick(Index As Integer)
    Dim sDirectoryNm As String
    Dim sDspFileType As String

    On Error GoTo ERR_NOFILE
    'Call RefreshFileList
    Call ChangeCurrentDirectory(Index)
    Call RefreshFileList
Exit Sub
```

```
ERR_NOFILE:
    If Err = 380 Then
        Resume Next
    Else
        Error Err
    End If

End Sub

Click event process of file list

Private Sub lstFile_Click()
    Dim FileName As String

    'Update selected file name
    FileName = lstFile.List(lstFile.ListIndex)
    FileName = Left$(FileName, InStr(FileName, Chr$(TB)) - 1)
    txtFile.Text = FileName

End Sub

'Display directory list
Private Sub RefreshDirectoryList(FrmValue As Form, Index As Integer)

    Dim DirectoryName As String
    Dim nChrStart As Integer
    Dim nChrEnd As Integer
    Dim DriveName As String
    Dim DirName As String
    Dim DirList As String

    'Delete contents of list
    LstDirectory(Index).Clear

    '*************************************
    'Insert directory path in list
    '*************************************
    nChrStart = 1
    nChrEnd = 1
    DirLevel(Index) = 0
    Do
        nChrEnd = InStr(nChrStart, CurrentDirectory(Index), "\") "\"
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add directory name to list
        nChrEnd = nChrEnd + 1 "\"is included
        DirName = Mid$(CurrentDirectory(Index), nChrStart, nChrEnd - nChrStart)
        DirName = String$(DirLevel(Index), " ") + DirName
        LstDirectory(Index).AddItem DirName

        nChrStart = nChrEnd
        DirLevel(Index) = DirLevel(Index) + 1
    Loop

    '*****************************************************************
    'Insert directory list of current directory in list
    '*****************************************************************
    'Get directory list of current directory
    DirList = GetDirectoryList(CurrentDirectory(Index))
```

```
    'Add directory list to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, DirList, Chr$(CR))
        If nChrEnd = 0 Then
            Exit Do
        End If

        'Add directory name to list
        DirName = Mid$(DirList, nChrStart, nChrEnd - nChrStart - 1) + "\"
        If DirName = ".\" Or DirName = "..\" Then
        Else
            DirName = String$(DirLevel(Index), " ") + DirName
            LstDirectory(Index).AddItem DirName
        End If
        nChrStart = nChrEnd + 1
    Loop

    LstDirectory(index).ListIndex = DirLevel(index) - 1

    'lblCurrentDirectory(index).Caption = CurrentDirectory(index)

End Sub

'Get directory list
Private Function GetDirectoryList(By Val DirectoryPath As String) As String
    Dim DirectoryList As string
    Dim DirectoryName As String
    Dim lBuffSize As Long
    Dim dwStatus As Long
    Dim lFileType As Long
    Dim lDirectoryID As Long
    Dim dwMelReadDirectoryStatus As Long

    '*****************************
    'Open directory
    '*****************************
    'Set directory open method
    'Bit16 = ON(Designate directory information)
    lFileType = &H10000


    'Open directory using API function
    dwStatus = melOpenDirectory(Me.hWnd, DirectoryPath, lFileType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDirectoryList")

    'Save directory ID
    lDirectoryID = dwStatus

    '******************************
    'Get directory list
    '******************************
    Do
        'Secure directory name storage area
        lBuffSize = 256
        DirectoryName = String$(lBuffSize, " ")

        'Get directory name using API function
        dwStatus = melReadDirectory(Me.hWnd, lDirectoryID, DirectoryName, lBuffSize)
```

```
        'Check API function call for errors
        If RetvIsError(dwStatus) = True Then
            'Error occurrence
            'Save status
            dwMelReadDirectoryStatus = dwStatus

            'Forcibly quit loop
            Exit Do
        End If

        'Confirm end of directory list data
        If dwStatus = 0 Then
            Exit Do
        End If

        'Add directory name to directory list
        DirectoryList = DirectoryList + Trim$(DirectoryName) + Chr$(CR)
    Loop

    '*****************************
    'Close directory
    '*****************************
    'Close directory using API function
    dwStatus = melCloseDirectory(Me.hWnd, lDirectoryID)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetDriveList")


    'Check melReadDirectory call for errors
    Call APIErrorCheck(dwMelReadDirectoryStatus, "GetDriveList")

    '*****************************
    'Return directory list
    '*****************************
    GetDirectoryList = DirectoryList

  End Function

'Update drive list contents
Private Sub RefreshDriveList()
    Dim nLoop As Integer
    Dim DriveList As String
    Dim nChrStart, nChrEnd As Integer

    Dim DriveName As String
    Dim nDriveNm As Integer
    Dim nDefaultDrvNum As Integer

'Update drive list
    DrvLstBxDrive.Refresh
    'Erase contents of list
    cmbDrive.Clear

    '********************************
    'Insert PC drive name in list
    '********************************
    For nDriveNm = 0 To (DrvLstBxDrive.ListCount - 1)
        cmbDrive.AddItem UCase$(DrvLstBxDrive.List(nDriveNm))
    Next

    nDefaultDrvNum = DrvLstBxDrive.ListCount
```

```
    '********************************
    'Insert NC drive name in list
    '********************************
    'Get NC drive list
    DriveList = GetDriveList()

    'Add drive list to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, DriveList, Chr$(CR))
        If nChrEnd = 0 Then
            Exit Do
        End If

        DriveName = Mid$(DriveList, nChrStart, nChrEnd - nChrStart)
        cmbDrive.AddItem DriveName

        nChrStart = nChrEnd + 1
    Loop

    '**********************************
    'Update current directory
    '**********************************
    DriveName = cmbDrive.List(nDefaultDrvNum)
    DriveName = Left$(DriveName, InStr(DriveName, ":"))
    CurrentDirectory(1) = DriveName + PRGFILE_PATH

    '**********************************
    'Select default drive
    '**********************************
    cmbDrive.ListIndex = nDefaultDrvNum

End Sub

'Refresh file list contents

Private Sub RefreshFileList()
    Dim DirectoryName As String
    Dim nChrStart As Integer
    Dim nChrEnd As Integer
    Dim FileName As String
    Dim FileNameBuff As String
    Dim FileList As String
    Dim nStrNm As Integer

    'Erase contents of list
    lstFile.Clear

    '*****************************************************************
    'Insert file list of current directory in the list
    '*****************************************************************
    'Get file list of current directory
    FileList = GetFileList(CurrentDirectory(1))

    'Add file to list
    nChrStart = 1
    nChrEnd = 1
    Do
        nChrEnd = InStr(nChrStart, FileList, Chr$(CR))
        If nChrEnd = 0 Then
            Exit Do
```

```
        End If

        'Add file name to list
        FileName = Mid$(FileList, nChrStart, nChrEnd - nChrStart - 1)
        If FileName = "." Or FileName = ".." Then
        Else
            nStrNm = InStr(1, FileName, Chr$(TB))
            If nStrNm <= 8 Then
                FileNameBuff = Mid$(FileName, 1, nStrNm - 1)
                FileNameBuff = FileNameBuff + Space$(8) + Mid$(FileName, nStrNm)
                lstFile.AddItem FileNameBuff
            Else
                lstFile.AddItem FileName
            End If
        End If
        nChrStart = nChrEnd + 1
    Loop

    'Invalidate selected file name
    txtFile.Text = ""

End Sub

'Execute operation search

Private Sub SearchPrg(FileName As String, lSequenceNum As Long, lBlockNum As Long)
    Dim dwStatus As Long                        'Variable to get return value from API function

    Dim typString As STRINGTYPE                 'Array for character string data type
    Dim lAddress As Long                        'Variable for designating address
    Dim lDataType As Long                       'Variable for designating data type

    '*************************************************
    'Initialize array for chracter string data type
    '*************************************************
    'Set size of program block storage area
    typString.lBuffSize = Len(FileName)

    'pointer of character string data area is set using special API function for VB
    typString.lpszBuff = FileName

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "SearchPrg")

    '************************
    'Execute operation search
    '************************
    'Set address
    'NC Card No. = 1, system designation = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Set data type
    'Designation character string type
    lDataType = T_STR

    'Execute operation search using API function
    dwStatus = melSelectExecPrg(Me.hWnd, lAddress, typString, lDataType, lSequenceNum, lBlockNum)

    'Check API function call for errors

    Call APIErrorCheck(dwStatus, "SearchPrg")
```

```
End Sub

'Timer event process
Private Sub Timer1_Timer()

    'Update program No.
    Pnl3DNowPrgNum.Caption = Str$(GetPrgNum())

    'Update sequence No.
    Pnl3DNowSequenceNum.Caption = Str$(GetSequenceNum())

    'Update block No.
    Pnl3DNowBlockNum.Caption = Str$(GetBlockNum())
End Sub
```

List 6-3 SRCHCOM.BAS code module

```
Attribute VB_Name="SRCHCOM"
Option Explicit

'Commonly used constants
Global Const CR = &HD
Global Const LF = &HA
Global Const TB = &H9

'Commonly used variables
Global CurrentDirectory  As String
```

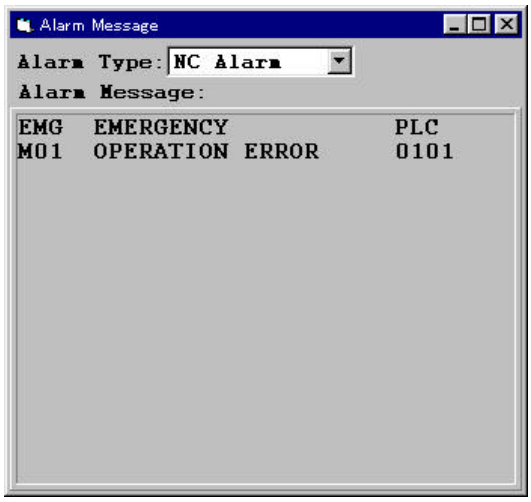
List 6-4 COMMON.BAS code module

```
Attribute VB_Name="COMMON"
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API function call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        End

    End If

End Sub
```

3.2.2.7 Alarm message display application

What is the alarm message display application?

The alarm message display application (Alarm Message) is a monitor that displays the alarm messages of the alarms occurring in the NC Card. A maximum of ten alarm messages of the alarms occurring in the NC Card will display on the screen. The type of alarm that is displayed can be selected with the [Alarm Type:] list box at the upper section of the window.
The window size can be changed.



Custom API Functions used

melGetCurrentAlarmMsg

Creation of alarm message display application

In this application, a procedure called GetAlarmMsg is created to get the alarms messages of the alarms currently occurring from the NC Card. GetAlarmMsg gets the alarm messages of the type designated in the argument, and returns the character string. The No. of messages gotten by GetAlarmMsg is the No. of messages designated in the argument.

```
'Get alarm message of currently occurring alarm
Private Function GetAlarmMsg(lMsgNos As Long, nAlarmKind As Integer) As String
   Dim dwStatus As Long                    'Variable to get return value from API function

   Dim lBuffSize As Long                   'Character string data area size
   Dim strBuff As String                   'Character string data area
   Dim typString As STRINGTYPE             'Array for character string data
   Dim lAlarmType As Long                  'Variable for designating alarm type

   Dim lAddress As Long                    'Variable for designating address
   Dim lDataType As Long                   'Variable for designating data type

   'Secure alarm message storage area
   lBuffSize = 256 * lMsgNos
```

```
    strBuff = String$(lBuffSize, 0)

    '*********************************
    'Initialize array for character string data
    '*********************************
    'Set character string data storage area size
    typString.lBuffSize = lBuffSize

     'Pointer of character string data area is set using special API function for VB
     typString.lpszBuff = strBuff

    'Check API function call for error
    Call APIErrorCheck(dwStatus, "GetAlarmMsg")

    '*********************************
    'Set alarm type
    '*********************************
    Select Case nAlarmKind
        Case 0
            'Designate alarm type
            lAlarmType = M_ALM_NC_ALARM
        Case 1
            'Designate stop code
            lAlarmType = M_ALM_STOP_CODE
        Case 2
            'Designate PLC alarm message
            lAlarmType = M_ALM_PLC_ALARM
        Case 3
            'Designate operator message
            lAlarmType = M_ALM_OPE_MSG
        Case 4
            'Designate all alarms
            lAlarmType = M_ALM_ALL_ALARM
    End Select

    '*******************************************
    'Get alarm message
    '*******************************************
    'Set address
    'NC Card No. = 1, system designation  = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Set data type
    'Designate character string data type
    lDataType = T_STR

    'Get alarm message using API function
    dwStatus = melGetCurrentAlarmMsg(Me.hWnd, lAddress, lMsgNos, lAlarmType, typString, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetAlarmMsg")

    'Return alarm message
    'lpszbuff should be assigned to strbuff exclicity

    strBuff = typString.lpszBuff

    GetAlarmMsg = strBuff

End Function
```

In GetAlarmMessage, the Custom API Function melGetCurrentAlarmMsg is called, and the alarm message of the alarm currently occurring in the NC Card is gotten. When calling melGetCurrentAlarmMsg in GetAlarmMsg, the address(lAddress) No. of messages to be gotten (lMsgNos), type of alarm to be gotten (lAlarmType), variable to store gotten message (typString) and storage area type (lDataType) are transferred.

The variable that stores the gotten message is a character string type (T_STR) variable supplied in the Custom API Library. The Custom API Library character string type variable is a user defined type array called STRINGTYPE. T_STR is designated for the storage area type. The Custom API Library character string type variable is explained in section "3.2.2.6 Operation search application".

List 7-1 ALMMSG32.VBP project file

```
Form=Frmalmsg.frm
Module=SdkCommon; Common.bas
Module=melerr; ..\..\include\vb\Melerr.bas
Module=melsberr; ..\..\include\vb\Melsberr.bas
Module=melncapi;  ..\..\include\vb\Melncapi.bas
Module=meltype; ..\..\include\vb\Meltype.bas
Module=ncmcapi.  ..\..\include\vb\Ncmcapi.bas
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL32.OCX
Object={3B7C8863-D78F-101B-B9B5-04021C009402}#1.0#0; RICHTX32.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST32.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#
Standard OLE Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#3.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT
SHARED\DC:\PROGRAM FIL#Microsoft DAO 3.0 Object Library
Object={0BA686C6-F7D3-101A-993E-0000C0EF6F5E}#1.0#0; THREED32.OCX
Reference=*\G{EF404E00-EDA6-101A-8DAF-00DD010F7EBB}#4.0#0#C:\PROGRAM FILES\MICROSOFT VISUAL
BASIC\vbext32.C:\#Microsoft Visual Basic 4.0 Development Environment
Environment
ProjWinSize=81,288,243,282
ProjWinShow=2
IconForm="frmAlarmMsg"
HelpFile=""
Title="ALARMMSG32"
ExeName32="Almmsg32.exe"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
```

List 7-2 FRMALMSG.FRM form module

```
VERSION 4.00
Begin VB.Form frmAlarmMsg
  Appearance      =   0  'Flat
  BackColor       =   &H00C0C0C0&
  Caption         =   "Alarm Message"
  ClientHeight    =   4500
  ClientLeft      =   2100
  ClientTop       =   1512
  ClientWidth     =   5100
  BeginProperty Font
    name            =   "System"
    charset         =   128
    weight          =   400
    size            =   13.2
    underline       =   0   'False
    italic          =   0   'False
    strikethrough   =   0   'False
  EndProperty
  ForeColor       =   &H80000008&
  Height          =   4884
  Icon            =   "Framalmsg.frx":0000
  Left            =   2052
  LinkTopic       =   "Form1"
  LockControls    =   -1  'True
  ScaleHeight     =   4500
  ScaleWidth      =   5100
  Top             =   1176
  Width           =   5196
  Begin VB.ComboBox cmbAlarmType
    Appearance      =   0  'Flat
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   700
      size            =   9.6
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    Height          =   300
    Left            =   1560
    Style           =   2  'Drop down list
    TabIndex        =   1
    Top             =   60
    Width           =   1875
  End
  Begin VB.PictureBox Picture1
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   400
      size            =   9.6
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   495
    Left            =   240
```

```
    ScaleHeight     =   468
    ScaleWidth      =   1128
    TabIndex        =   0
    Top             =   3480
    Visible         =   0   'False
    Width           =   1155
  End
  Begin VB.Timer Timer1
    Interval        =   500
    Left            =   4740
    Top             =   4020
  End
  Begin Threed.SSPanel Pnl3DAlarmMsgPanel
    Height          =   3840
    Left            =   0
    TabIndex        =   4
    Top             =   672
    Width           =   5112
    _Version        =   65536
    _ExtentX        =   9022
    _ExtentY        =   6773
    _StockProps     =   15
    BevelInner      =   1
    Begin VB.Label lblAlarmMsg
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      BeginProperty Font
        name            =   "MS P Gothic"
        charset         =   128
        weight          =   700
        size            =   12
        underline       =   0   'False
        italic          =   0   'False
        strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   675
      Left            =   75
      TabIndex        =   5
      Top             =   75
      Width           =   4935
    End
  End
  Begin VB.Label Label2
    Appearance      =   0  'Flat
    BackColor       =   &H80000005&
    BackStyle       =   0  'Transparent
    Caption         =   "Alarm Message:"
    BeginProperty Font
      name            =   "Courier"
      charset         =   0
      weight          =   700
      size            =   9.6
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
    EndProperty
    ForeColor       =   &H80000008&
    Height          =   255
    Left            =   75
    TabIndex        =   3
```

```
      Top            =   420
      Width          =   2115
    End
    Begin VB.Label Label1
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Alarm Type:"
      BeginProperty Font
        name          =   "Courier"
        charset       =   0
        weight        =   700
        size          =   9.6
        underline     =   0   'False
        italic        =   0   'False
        strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   60
      TabIndex        =   2
      Top             =   120
      Width           =   1575
    End
  End
  Attribute VB_Name = "frmAlarmMsg"
  Attribute VB_Creatable = False
  Attribute VB_Exposed = False

  Option Explicit

  Const PANEL3D_FRAME_WIDTH = 70

  Private Sub Form_Load()
     Dim i As Integer

     'Arrange window in center
     Me.Top = (Screen.Height / 2) - (Me.Height / 2)
     Me.Left = (Screen.Width / 2) - (Me.Width / 2)

     'Arrange object
     lblAlarmMsg.Top = PANEL3D_FRAME_WIDTH
     lblAlarmMsg.Left = PANEL3D_FRAME_WIDTH

     'Initialize list for alarm type selection
     cmbAlarmType.Clear
     cmbAlarmType.AddItem "NC Alarm"
     cmbAlarmType.AddItem "Stop Code"
     cmbAlarmType.AddItem "PLC Alarm"
     cmbAlarmType.AddItem "Ope Message"
     cmbAlarmType.AddItem "All Alarm"
     cmbAlarmType.ListIndex = 0

  End Sub

  Private Sub Form_Resize()
     Dim FormWidthMin As Single
     Dim fWidth As Single

     'Obtain lower limit of window width
     FormWidthMin = cmbAlarmType.Left + cmbAlarmType.Width + (Width - ScaleWidth)
```

```
    'Check lower limit of window width
    If Width < FormWidthMin Then
    If Not Me.WindowState=1 Then
        Width = FormWidthMin
        End If
    End If

    fWidth = ScaleWidth
    If fWidth < 1 Then
        fWidth = 1
    End If
    Pnl3DAlarmMsgPanel.Width = fWidth

    fWidth = ScaleHeight - Pnl3DAlarmMsgPanel.Top
    If fWidth < 1 Then
        fWidth = 1
    End If
    Pnl3DAlarmMsgPanel.Height = fWidth

    fWidth = Pnl3DAlarmMsgPanel.Width - PANEL3D_FRAME_WIDTH * 2
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblAlarmMsg.Width = fWidth

    fWidth = Pnl3DAlarmMsgPanel.Height - PANEL3D_FRAME_WIDTH * 2
    If fWidth < 1 Then
        fWidth = 1
    End If
    lblAlarmMsg.Height = fWidth

End Sub

'Get alarm message of currently occurring alarm
Private Function GetAlarmMsg(lMsgNos As Long, nAlarmKind As Integer) As String
    Dim dwStatus As Long                    'Variable to get return value from API function

    Dim lBuffSize As Long                   'Character string data area size
    Dim strBuff As String                   'Character string data area
    Dim typString As STRINGTYPE             'Array for character string data
    Dim lAlarmType As Long                  'Variable for designating alarm type

    Dim lAddress As Long                    'Variable for designating address
    Dim lDataType As Long                   'Variable for designating data type

    'Secure alarm message storage area
    lBuffSize = 256 * lMsgNos
    strBuff = String$(lBuffSize, 0)

    '*********************************
    'Initialize array for character string data
    '*********************************
    'Set character string data storage area size
    typString.lBuffSize = lBuffSize
    'Set pointer of character string data area using special API function for VB

    'Pointer of character string data area is set directly in Win 95
    typString.lpszBuff = strBuff

    'Check API function call for error
    Call APIErrorCheck(dwStatus, "GetAlarmMsg")
```

```
    '*********************************
    'Set alarm type
    '*********************************
    Select Case nAlarmKind
        Case 0
            'Designate alarm type
            lAlarmType = M_ALM_NC_ALARM
        Case 1
            'Designate stop code
            lAlarmType = M_ALM_STOP_CODE
        Case 2
            'Designate PLC alarm message
            lAlarmType = M_ALM_PLC_ALARM
        Case 3
            'Designate operator message
            lAlarmType = M_ALM_OPE_MSG
        Case 4
            'Designate all alarms
            lAlarmType = M_ALM_ALL_ALARM
End Select

    '*******************************************
    'Get alarm message
    '*******************************************
    'Set address
    'NC Card No. = 1, system designation  = No. 1 system
    lAddress = ADR_MACHINE(1) Or ADR_SYSTEM(1)

    'Set data type
    'Designate character string data type
    lDataType = T_STR

    'Get alarm message using API function
    dwStatus = melGetCurrentAlarmMsg(Me.hWnd, lAddress, lMsgNos, lAlarmType, typString, lDataType)

    'Check API function call for errors
    Call APIErrorCheck(dwStatus, "GetAlarmMsg")

    'Return alarm message
    'lpszbuff should be assigned to strbuff explicitly

    strBuff = typString.lpszBuff

    GetAlarmMsg = strBuff

End Function


'Obtain height of one line of text display object
Private Function GetLineHeight(lblObj As Control) As Single

    Picture1.FontBold = lblObj.FontBold
    Picture1.FontItalic = lblObj.FontItalic
    Picture1.FontSize = lblObj.FontSize
    Picture1.FontName = lblObj.FontName

    GetLineHeight = Picture1.TextHeight("A")

End Function

'Obtain No. of lines displayable in text display object
Private Function GetLineNos(lblObj As Control) As Long
```

```
        GetLineNos = Int(lblObj.Height / GetLineHeight(lblObj))
    End Function

    'Timer event process
    Private Sub Timer1_Timer()
        Dim lBlockNos As Long
        Dim nAlarmKind As Integer

        'Obtain No. of lines displayable in display area
        lBlockNos = GetLineNos(lblAlarmMsg)

        'Set range of No. of displayed program blocks between 1 and 10
        If lBlockNos < 1 Then
            lBlockNos = 1
        ElseIf lBlockNos > 10 Then
            lBlockNos = 10
        End If

        'Designate type of alarm to be gotten
        nAlarmKind = cmbAlarmType.ListIndex

        'Get alarm message and display
        lblAlarmMsg.Caption = GetAlarmMsg(lBlockNos, nAlarmKind)

    End Sub
```

List 7-3 COMMON.BAS code module

```
Attribute VB_Name="SdkCommon"
Option Explicit

'Check if API function return value is an error
'If return value is an error, display message, and
'quit application
Sub APIErrorCheck (dwStatus As Long, FunctionName As String)
    Dim Message As String
    If RetvIsError(dwStatus) = True Then
        'Error occurrence
        'Display error message
        Message = "Error occurred in API function call"
        Message = Message + Chr$(10) + "Error occurrence place is " + FunctionName + "."
        Message = Message + Chr$(10) + "Error No. is &h" + Hex$(dwStatus) + "."
        MsgBox (Message)

        'Quit application
        'Stop
        End

    End If

End Sub
```

3.2.3 Helpful information for creating custom applications

3.2.3.1 How to access T_BIT data

Each variable of the NC Data Access Variables has a default data type (data type originally held by NC Card). The following data types are included in the default data types.

Default data types of NC Data Access Variables

Default data type		Type of variable provided with custom application	
T_BIT	1-bit data type	Cannot be used with Visual Basic	
T_CHAR	1-byte integer type	Char	Byte type
T_SHORT	2-byte integer type	Integer	Integer type
T_LONG	4-byte integer type	Long	Long integer type
T_DOUBLE	4-byte real number type	Double	Double precision real number type
T_STR	Character string type	STRINGTYPE	User defined array for character string data

When reading and writing the NC Data Access Variables using melReadData and melWriteData, generally a default data type is designated for the requested data type in the custom API Function argument.
However, Visual Basic does not have the 1-bit length data type. Thus, the types corresponding to these, or in other words T_BIT, cannot be designated for the requested data type. When creating a custom application with Visual Basic, useT_CHAR or T_SHORT instead of these default data types.
If T_SHORT is designated for a default data type T_BIT variable, the Custom API Library will set the variable value in bit 0 of the 2-byte length data.
The NC Data Access Variable data type conversion is executed on all types of data regardless of it being the T_BIT. For example, the T_SHORT data can be requested as a T_LONG type or T_DOUBLE type. The data type conversion is explained in section "3.2.2.1 Counter display application".

3.2.3.2 Precautions for using variables of String type

Some Custom API Functions request the String type variable for the variable that the function sets the value. Caution is required when transferring a variable to this kind of function.
When transferring the String type variable as a variable for the Custom API Function to set the value, secure a character string storage area before calling the function. The following methods can be used to secure the area to store the character string in the String type variable.

Method 1. Declare the String type variable as a fixed length character string

```
Dim sBuff As String *256                          'Declare a 256-byte length fixed length character string
    :
    :
dwStatus = melGetDriveList(Me.hwnd,....,sBuff,.....)   'Call the Custom API Function
    :
    :
```

Method 2. Secure the area before calling

```
Dim sBuff As string                               'Declare a variable length character string
    :
    :
sBuff = String$(256,0)                            'Secure a 256-byte area
 dwStatus = melGetDriveList(Me.hwnd,....,sBuff,....)   'Call the Custom API Function
```

If the API function is called without securing the character string storage area, the windows' "Page

Breach" will occur. This problem does not apply only to the Custom API Function, but also occurs when calling a DLL function from Visual Basic.

3.2.3.3 Prohibition of Variant type for variable data type'

One of the data types used by Visual Basic is the Variant type. This Variant type is a data type applied as an initial setting when a clear variable type is not declared in Visual Basic.
If the Variant type variable is transferred to the API function as a variable for the Custom API Function to set the value in, the Windows' "Page Breach" will occur.
The Variant type does not secure an area to store the variable value when the variable is declared. Instead, the area is secured. The process to dynamically assign the area during the variable substitution is not executed in the DLL function. Thus, if a variant type variable is transferred as the variable for the Custom API Function to set the value in, the custom API Function will try to write the value in a region where the area is not secured. If this type of process is attempted, the Windows' "Page Breach" will occur.
Due to the above reasons, the Variant type variable cannot be used as the variable for the Custom API Function to set the value in.

3.2.3.4 Calling custom application from MELDASMAGIC MMI Software (MAGIC.EXE) (option)

The custom application created by the user can be registered in the command button on the menu window of the MAGIC.EXE. The custom application registered in the command button can be started by clicking the button. By using this function, the MAGIC.EXE, can be partially customized. To make a registration in the MAGIC.EXE command button, revise the initialization file (MAGICM01.INI, etc.) in the MAGIC.EXE. Refer to the "MELDASMAGIC MMI Operation Manual (For D/M) (BNP-B2193)" or "MELDASMAGIC MMI Operation Manual (For L/G) (BNP-B2194)" for details on the initialization file.

4. API Test

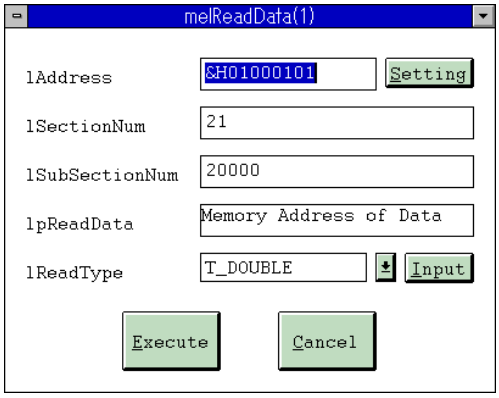
4.1 API Test Outline

The API Test is a test tool developed to confirm the operation of the Custom API Library. This tool is convenient for the user when developing original screens using the Custom API Library. After confirming the input/output and operation of the Custom API Function with this tool, the programming can efficiently advance by integrating the functions into the original screen program.

When programming using the Custom API Function, it is necessary to give the function some arguments. For example, the below arguments are for melReadData.

dwStatus=melReaddata(hWnd, lAddress, lSectionNum, lSubSectionNum, lpReadData, lReadType);

The API Test offers this kind of Custom API Function argument setting and function call in the following type of window, making operation confirmation possible in programless Custom API Functions.



This tool operates with Windows 95. The NC Card must be operating.
Use this tool with the manuals below.
"Custom Application Interface Library Guide (Function section)(BNP-B2198)"
"Custom Application Interface Library Guide (Variable section)(BNP-B2199)"

4.2 Installing the API Test

The API Test is stored on the "Custom API Library SDK1" floppy disk. The "Utility Software" must be installed to use the API Test.
The API installation procedure is shown below.

■ Install the utilities using the "Setup Instruction Manual (BNP-B2191)" as a reference.

■ Install the Custom API Library using the "2.2 Setting up of Custom API Library" as a reference.

■ Register the "API Test" in the [Start] menu.

[Procedure]

(1) Click on [Settings]-[Taskbar...] of the taskbar [Start] menu.
Operation : The "Taskbar Property" window appears.

(2) Click on the [Start Menu Settings] tab.

(3) Click on the [Add] button of the [[Start] menu] group.
Operation : The "Create Shortcut" window appears.

(4) Input the following in [Command Line:], and then click on the [Next>] button.

C:\meltools\win\bin\apitest.exe

Operation : The "Folder Selection" window appears.

(5) Select the following in [Select the Folder where the Shortcut is Saved], and then click on [Next>].

[Start Menu] - [Programs] - [MELDASMAGIC]

If the [MELDASMAGIC] folder does not exist at this time, click on the [New Folder] button, and create the [MELDASMAGIC] folder.
Operation : The "Name Designation" window appears.

(6) Input the following in [Designate the Shortcut Name], and then click on the [Finish] button.

API Test

(7) Click on the [OK] button in the "Taskbar Property" window.
Operation : The "Taskbar Property" window closes.

4.3 Starting and Ending the API Test

4.3.1 Starting the API Test

API Test will start in the following manner.

[Procedure]

(1) Click on [Programs]-[MELDASMAGIC]-[API Test] menu of the taskbar [Start] menu.
Operation : The API Test starts.

When the API test starts, the Main window will appear in the upper left of the screen.

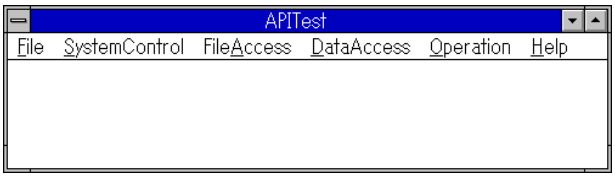


Fig. 4-1 Main window

4.3.2 Ending the API Test

(1) When the [File]-[Exit] menu is clicked on, the API Test will end.

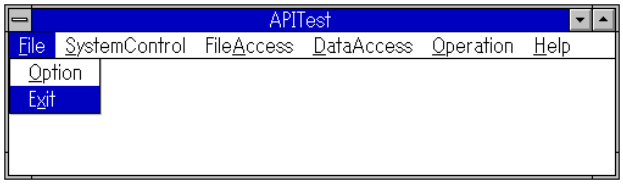


Fig. 4-2 [File]-[Exit] menu

4.4 API Test Basic Operation

4.4.1 Selecting the API Function

In the API test, one sub-window as in Fig. 4-3 corresponds to one API Function. These sub-windows are called Function windows. Functions can be called in by designating the API Function argument and clicking on [Execute]. There are 18 supporting API Functions, sorted into four groups. Select them from the pull-down menu in the Main window (Refer to Fig. 4-4). Refer to "Custom Application Interface Library Guide (Function section)(BNP-B2198)" for details on each function.

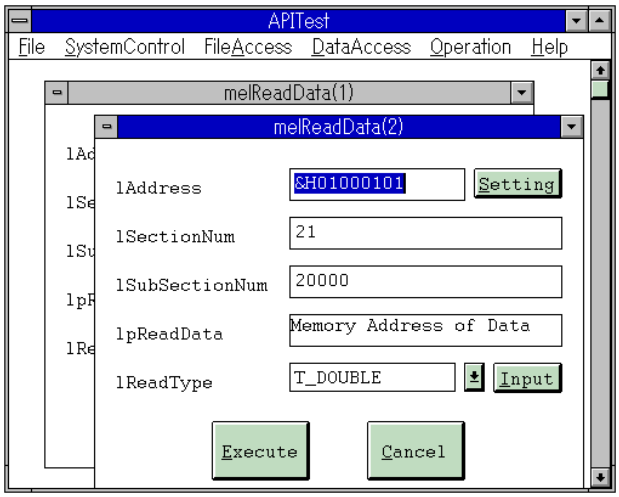


Fig. 4-3 Example of two open melReadData API Functions

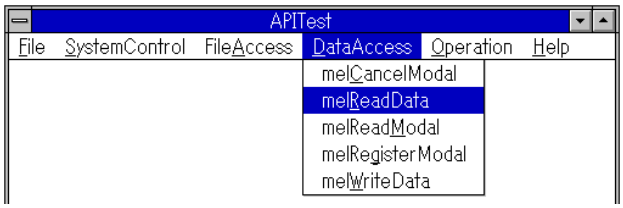


Fig. 4-4 Main window menu bar

Table 4-1 List of support functions

(1) System control command : SystemControl
melIoct1
(2) File access related commands : FileAccess
melCloseDirectory
melCopyFile
melDeleteFile
melGetDiskFree
melGetDriveList
melOpenDirectory
melReadDirectory
melRenameFile
(3) Data access related commands : DataAccess
melCancelModal
melReadData
melReadModal
melRegisterModal
melWriteData
(4) Operation related commands : Operation
melActivatePLC
melGetCurrentAlarmMsg
melGetCurrentPrgBlock
melSelectExecPrg

4.4.2 Opening multiple windows

The API Test can open multiple function windows. When two or more of the same function windows are open, a number to separate them appears in the () to the right of the API Function name.
(Refer to Fig. 4-3)

4.4.3 Starting multiple API Tests

Multiple API Tests can be opened simultaneously. During API Test start, start API Test again. Another API Test Main window will open.

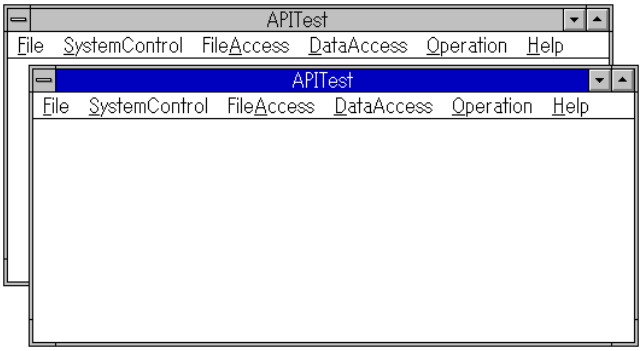


Fig. 4-5 Main windows when API Test has been started twice.

4.4.4 Setting the API Test options

The display method of the API Function output/return value can be set to decimal or hexadecimal. Click on the [File]-[Option] menu, open the Option window, and then select either Hexadecimal or Decimal from the [Output] combo box.

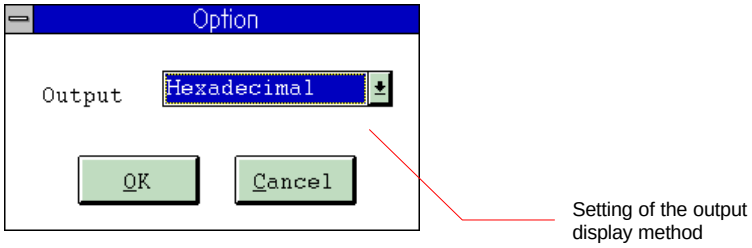


Fig. 4-6 Option window

The set option is valid from the function call executed after the option setting.

4.4.5 Version information

The Version window appears when the [Help]-[Version] menu is clicked.
The version of the API Test currently in use is displayed in this window.
The Version window closes when the [OK] button is clicked.

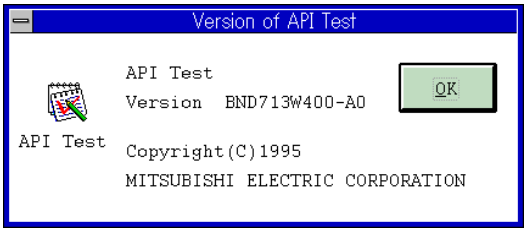
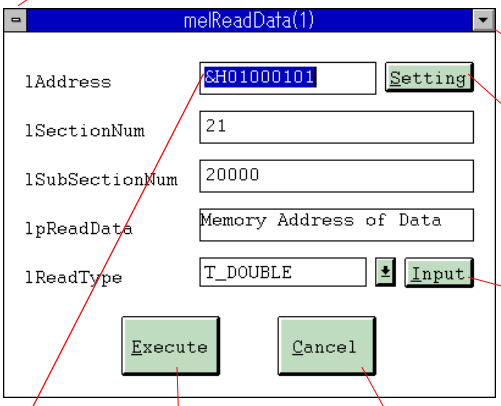


Fig. 4-7 Version window

4.5 Operation of the Function Execution window

4.5.1 Common window operations

The buttons that can be used in all API Function windows of the API Test are introduced in this chapter. A function window of an API Function, "melReadData", is shown in the figure below. Set the respective values, etc., on the right side for lAddress, lSectionNum, lSubSectionNum, lpReadData, and lReadType by inputting to the "melReadData" function.
The [Setting], [Input], etc., buttons supplement those settings or display a dialog box for detailed setting.



Control menu box
Use to move and close the window.

Minimize button
Use to display the window using icons.

[Setting] button
Use to set the address, file type, options, etc.

[Input] button
Use to input data and set the structural members.

[Cancel] button
Use to cancel the argument setting and close the window.

[Execute] button
Use to call the API Function.

Text input

Move the focus to the text box. All of the text in the text box that has the focus is selected. Input the text from the keyboard, and then press the [Enter] key.
Decimal or hexadecimal values can be input into the text box or combo box. When inputting a hexadecimal, attach "&H", &h", "0x", or "0X" in front of the value before inputting.
When inputting decimals, input the value only.

4.5.2 Displaying the return value from the function

When the API Function is executed with the [Execute] button, the Return Value window appears in the center of the screen, and the return value from the function is displayed. The example in Fig. 4-8 shows that the "&H0" value on the right side of dwStatus= is the return value from the function, and that the hexadecimal 0 was returned.

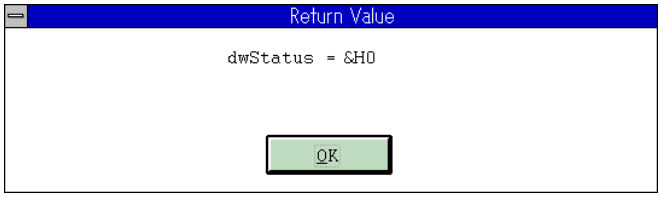


Fig. 4-8 Return Value window

When the return value is an error, the error name appears. Refer to the "Custom Application Interface Library Guide (Function section)(BNP-B2198)" for information about the error names.

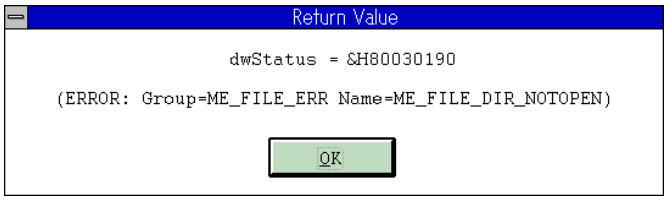


Fig. 4-9 Return Value window (error display)

When the [OK] button is clicked on, the Return Value window closes.

4.5.3 Function window initial values

The initial value of the argument transferred to the function is described in the next chapter in each function window's explanation figure. When the function window is closed and then reopened, the value of the argument transferred to each function returns to the initial value. Use the minimize button to leave the argument setting value as is and close the function window.

4.5.4 System control commands

4.5.4.1 melloctl

Calling the function

(1) Set the argument to be transferred to the melloctl function.

(2) Call the melloctl function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melloctl appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

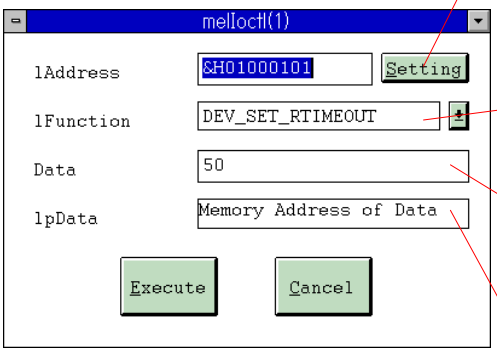


Fig. 4-10 melloctl window

Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually. Refer to "4.5.9.1 AddressSet window".

Function code
Use to select the melloctl function from the combo box.

Data
Use to set the data for each function.

Pointer
"Memory Address of Data" appears in [lpData].
This shows that the memory address of the variable where the data is stored is set in lpData argument.

4.5.5 File access related commands

4.5.5.1 melCloseDirectory

Calling the function

(1) Set the argument to be transferred to the melCloseDirectory function.

(2) Call the melCloseDirectory function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melCloseDirectory appears. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melCloseDirectory window to quit the melCloseDirectory window.

Directory ID
Use to designate the ID of the directory to be closed. The ID can be selected from the combo box or directly input.

The ID of the directory opened last with the melOpenDirectory function appears in the combo box.

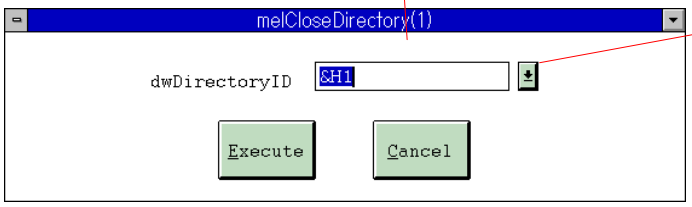


Fig. 4-11 melCloseDirectory window

4.5.5.2 melCopyFile

Calling the function

(1) Set the argument to be transferred to the melCopyFile function.

(2) Call the melCopyFile function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melCopyFile appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melCopyFile window to quit the melCopyFile window.

Copy origin file name
Use to designate the copy origin file name.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side file can be selected using the directory tree. (Key input the file name when operating NC side files)
Refer to "4.5.9.4 FileList window".

Copy destination file name
Use to designate the copy destination file name.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side file can be selected using the directory tree. (Key input the file name when operating NC side files)

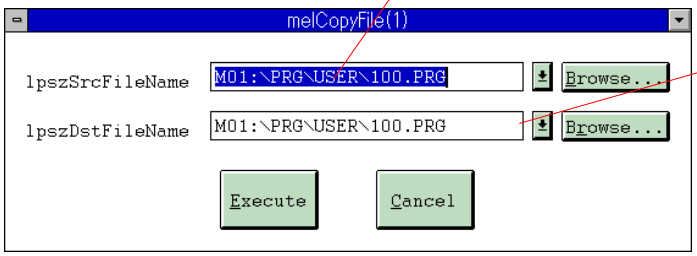


Fig. 4-12 melCopyFile window

4.5.5.3 melDeleteFile

Calling the function

(1) Set the argument to be transferred to the melDeleteFile function.

(2) Call the melDeleteFile function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melDeleteFile appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melDeleteFile window to quit the melDeleteFile window.

File name to be deleted
Use to designate the file name to be deleted.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side file can be selected using the directory tree. (Key input the file name when operating NC side files)
Refer to "4.5.9.4 FileList window".

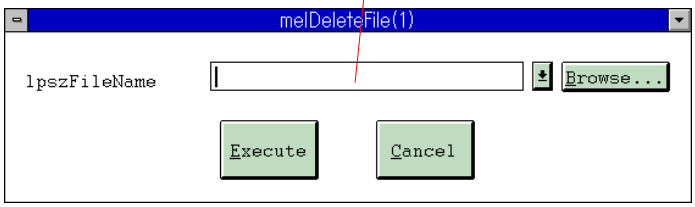


Fig. 4-13 melDeleteFile window

4.5.5.4 melGetDiskFree

Calling the function

(1) Set the argument to be transferred to the melGetDiskFree function.

(2) Call the melGetDiskFree function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melGetDiskFree appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melGetDiskFree window to quit the melGetDiskFree window.

Directory name that will retrieve the open capacity
Use to designate the directory name that will retrieve the open capacity.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side directory can be selected using the directory tree. (Key input the directory name for NC side)
Refer to "4.5.9.4 FileList window".

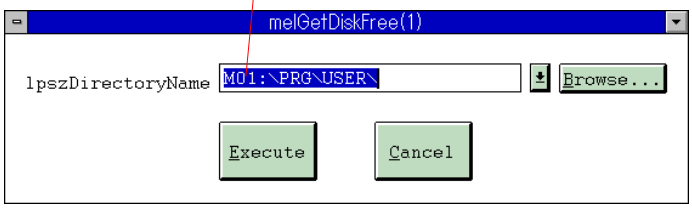


Fig. 4-14 melGetDiskFree window

4.5.5.5 melGetDriveList

Calling the function

(1) Set the argument to be transferred to the melGetDriveList function.

(2) Call the melGetDriveList function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melGetDriveList appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melGetDriveList window to quit the melGetDriveList window.

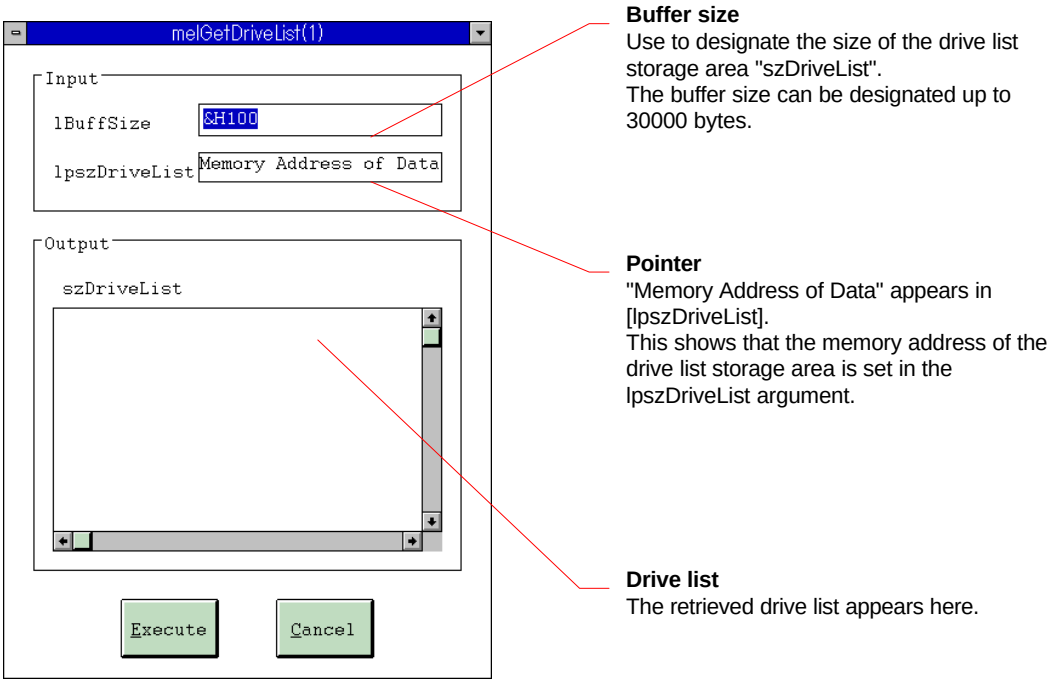


Fig. 4-15 melGetDriveList window

4.5.5.6 melOpenDirectory

Calling the function

(1) Set the argument to be transferred to the melOpenDirectory function.

(2) Call the melOpenDirectory function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melOpenDirectory appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melOpenDirectory window to quit the melOpenDirectory window.

Directory name to be opened
Use to designate the directory name to be opened.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side directory can be selected using the directory tree. (Key input the directory name for NC side)
Refer to "4.5.9.4 FileList window".

File type
Use to designate the data type and format to be read with melReadDirectory.
By using the [Setting] button, the file type can be set separately for each item.
Refer to "4.5.9.2 FileTypeSet window".

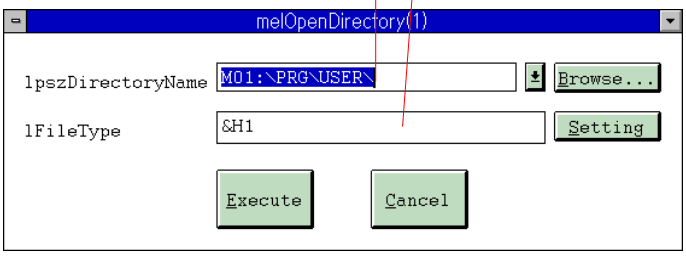


Fig. 4-16 melOpenDirectory window

4.5.5.7 melReadDirectory

Calling the function

(1) Set the argument to be transferred to the melReadDirectory function.

(2) Call the melReadDirectory function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melReadDirectory appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melReadDirectory window to quit the melReadDirectory window.

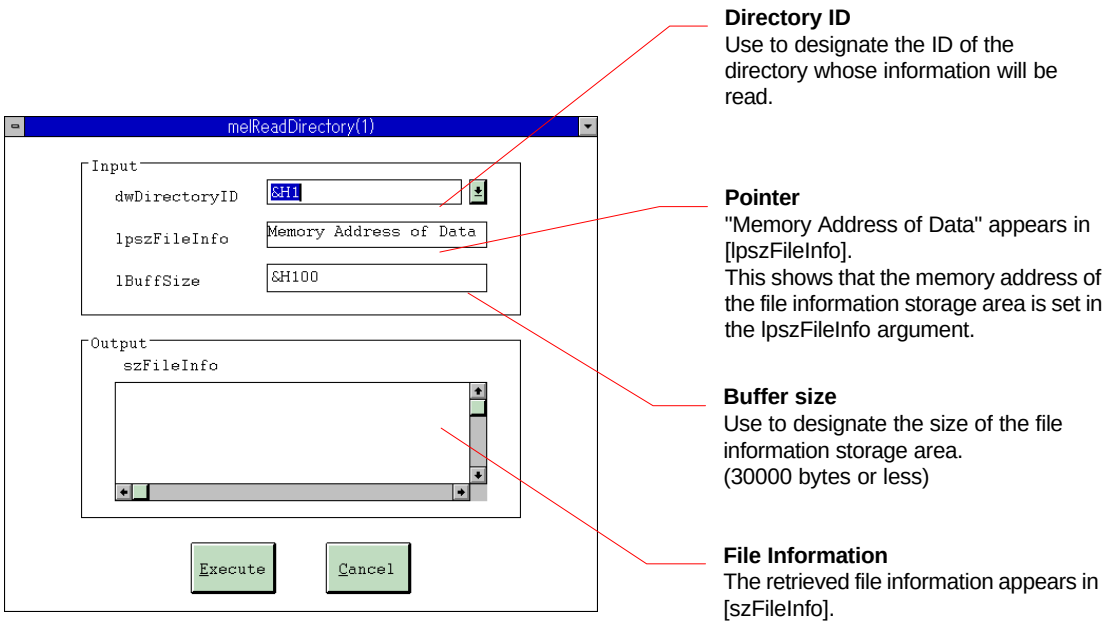


Fig. 4-17 melReadDirectory window

4.5.5.8 melRenameFile

Calling the function

(1) Set the argument to be transferred to the melRenameFile function.

(2) Call the melRenameFile function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melRenameFile appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melRenameFile window to quit the melRenameFile window.

Old file name
Use to designate the old file name.
A history of the last five file names appears in the combo box.
By clicking on the [Browse ...] button, the personal computer side file can be selected using the directory tree. (Key input the file name when operating NC side files)
Refer to "4.5.9.4 FileList window".

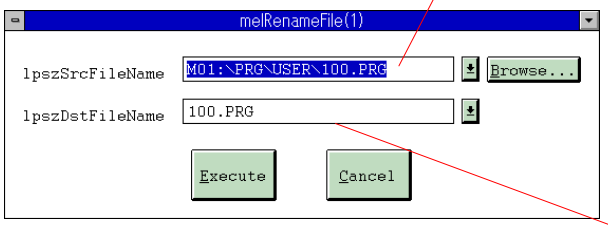


Fig. 4-18 melRenameFile window

New file name
Use to designate the new file name.
A history of the last five file names appears in the combo box.

4.5.6 Data access-related commands

4.5.6.1 melCancelModal

Calling the function

(1) Set the argument to be transferred to the melCancelModal function.

(2) Call the melCancelModal function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melCancelModal appears in the Return Value window. The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melCancelModal window to quit the melCancelModal window.

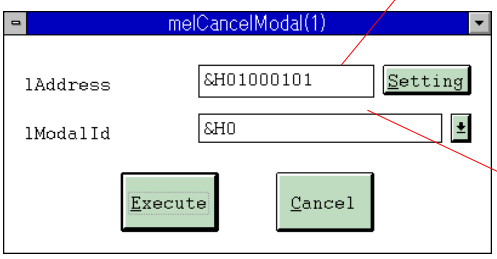


Fig. 4-19 melCancelModal window

Address
Use to designate the NC Card.
By using the [Setting] button, the address can be set separately for each item.
Refer to "4.5.9.1 AddressSet window".

Data ID
Use to designate the ID of the data that cancels the registration of the data high-speed reading.
Select the data from the combo box or input directly.
The ID of the data registered in the melRegisterModal window appears in the combo box.

4.5.6.2 melReadData

Calling the function

(1) Set the argument to be transferred to the melReadData function.

(2) Click on the [Input] button. The window of the data type designated in lReadType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.

In this case, the default values are used for the structural members.

(3) Input the data in the Input window, and set the structural members.

(4) Call the melReadData function using the [Execute] button.

(5) The Return Value window appears in the center of the screen. The return value from melReadData appears in the Return Value window.

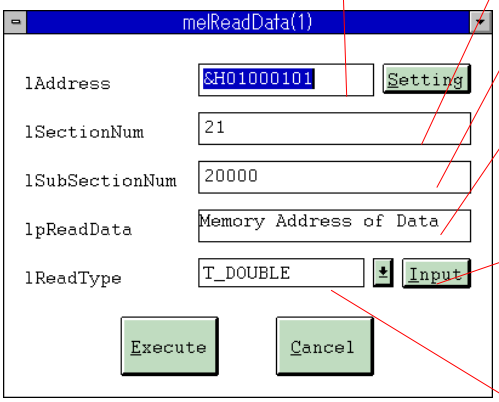
The output from melReadData appears simultaneously in [Data Area] of the Input window. The Return Value window will close when the [OK] button is clicked.

(6) Click on the [Cancel] button in the melReadData window to quit the melReadData window. The Input window also closes simultaneously.

Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Section No.
Use to designate the section No.

Sub-section No.
Use to designate the sub-section No.



Pointer
"Memory Address of Data" appears in [lpszReadData].
This shows that the memory address of the variable where the data is stored is set in the lpszReadData argument.

Data input
Use to input data and structural member settings.
Refer to "4.5.8 Input window"

Data type
Use to designate the type of variable that will store the read data.
The symbol name of the data type can be selected from the combo box.

Fig. 4-20 melReadData window

4.5.6.3 melReadModal

Calling the function

(1) Set the argument to be transferred to the melReadModal function.

(2) Click on the [Input] button. The window of the data type designated in lReadType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.

In this case, the default values are used for the structural members.

(3) Input the data in the Input window, and set the structural members.

(4) Call the melReadModal function using the [Execute] button.

(5) The Return Value window appears in the center of the screen. The return value from melReadModal appears in the Return Value window.

The output from melReadModal appears simultaneously in [Data Area] of the Input window. The Return Value window will close when the [OK] button is clicked.

(6) Click on the [Cancel] button in the melReadModal window to quit the melReadModal window. The Input window also closes simultaneously.

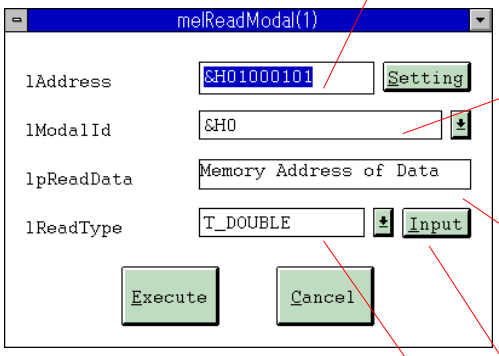


Fig. 4-21 melReadModal window

Address
Use to designate the NC Card, system, and axis.
Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Data ID
Use to designate the ID of the data to be read.
The ID of the data registered in the melRegisterModal window appears in the combo box.

Pointer
"Memory Address of Data" appears in [lpReadData].
This shows that the memory address of the variable where the data is stored is set in the lpReadData argument.

Data input
Use to input data and structural member settings.
Refer to "4.5.8 Input window"

Data type
Use to designate the type of variable that will store the read data.

4.5.6.4 melRegisterModal

Calling the function

(1) Set the argument to be transferred to the melRegisterModal function.

(2) Call the melRegisterModal function using the [Execute] button.

(3) The Return Value window appears in the center of the screen, and the return value from melRegisterModal appears.
The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melRegisterModal window to quit the melRegisterModal window.

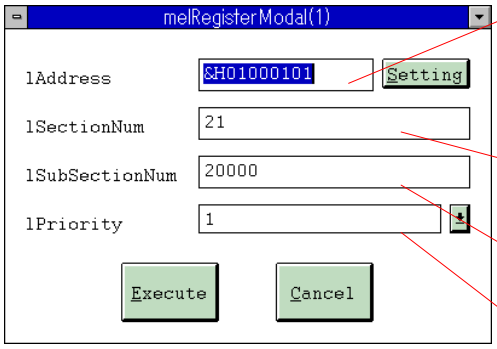


Fig. 4-22 melRegisterModal window

Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Section No.
Use to designate the section No.

Sub-section No.
Use to designate the sub-section No.

Priority order
Use to designate the priority order. The candidates for the priority order can be selected in the combo box.

4.5.6.5 melWriteData

Calling the function

(1) Set the argument to be transferred to the melWriteData function.
(2) Click on the [Input] button. The window of the data type designated in lWriteType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.
In this case, the default values are used for the structural members.
(3) Input the data in the Input window, and set the structural members.
(4) Call the melWriteData function using the [Execute] button.
(5) The Return Value window appears in the center of the screen. The return value from melWriteData appears in the Return Value window.
The Return Value window will close when the [OK] button is clicked.
(6) Click on the [Cancel] button in the melWriteData window to quit the melWriteData window. The Input window also closes simultaneously.

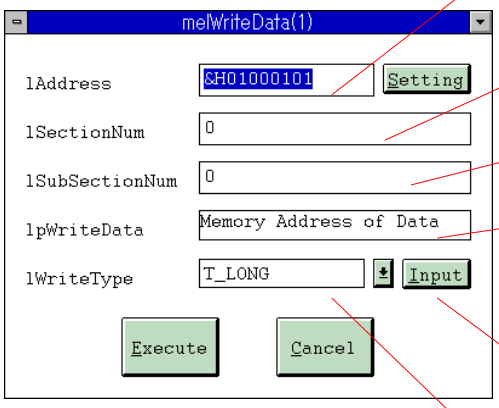


Fig. 4-23 melWriteData window

Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Section No.
Use to designate the section No.

Sub-section No.
Use to designate the sub-section No.

Pointer
"Memory Address of Data" appears in [lpWriteData].
This shows that the memory address of the variable where the data is stored is set in the lpWriteData argument.

Data input
Use to input data and structural member settings.
Refer to "4.5.8 Input window"

Data type
Use to designate the variable type that stored the written data.

4.5.7 Operation related commands

4.5.7.1 melActivatePLC

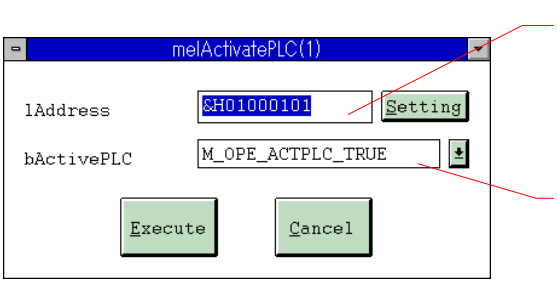
Calling the function

(1) Set the argument to be transferred to the melActivatePLC function.

(2) Call the melActivatePLC function using the [Execute] button.

(3) The Return Value window appears in the center of the screen. The return value from melActivatePLC appears in the Return Value window.
The Return Value window will close when the [OK] button is clicked.

(4) Click on the [Cancel] button in the melActivatePLC window to quit the melActivatePLC window.



Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Operation mode
Use to designate the PLC operation mode (PLC operation or stop).
The setting value can be selected from the combo box.

Fig. 4-24 melActivatePLC window

4.5.7.2 melGetCurrentAlarmMsg

Calling the function

(1) Set the argument to be transferred to the melGetCurrentAlarmMsg function.

(2) Click on the [Input] button. The window of the data type designated in lReadType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.

In this case, the default values are used for the structural members.

(3) Input the data in the Input window, and set the structural members.

(4) Call the melGetCurrentAlarmMsg function using the [Execute] button.

(5) The Return Value window appears in the center of the screen, and the return value from melGetCurrentAlarmMsg appears. The output from melGetCurrentAlarmMsg also appears simultaneously in [Data Area] of the Input window. The Return Value window will close when the [OK] button is clicked.

(6) Click on the [Cancel] button in the melGetCurrentAlarmMsg window to quit the melGetCurrentAlarmMsg window. The Input window also closes simultaneously.

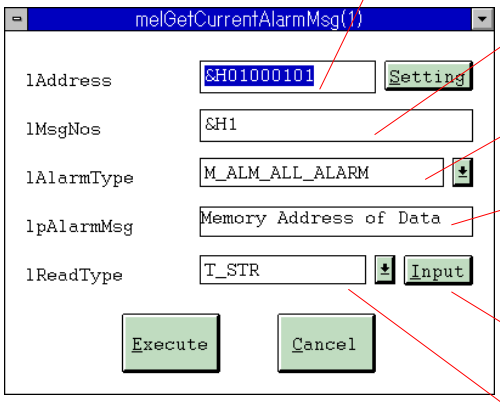


Fig. 4-25 melGetCurrentAlarmMsg window

Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually. Refer to "4.5.9.1 AddressSet window".

Number of messages
Use to designate the number of messages to be retrieved.

Alarm type
Use to designate the type of alarm to be retrieved.

Pointer
"Memory Address of Data" appears in [lpAlarmMsg].
This shows that the memory address of the variable where the message is stored is set in the lpAlarmMsg argument.

Data input
Use to input data and structural member settings.

Data type
Use to designate the variable type where the message will be stored.

4.5.7.3 melGetCurrentPrgBlock

Calling the function

(1) Set the argument to be transferred to the melGetCurrentPrgBlock function.

(2) Click on the [Input] button. The window of the data type designated in lReadType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.

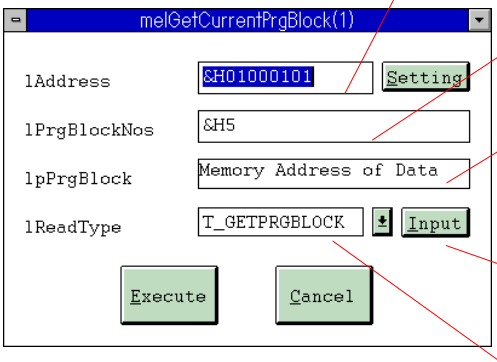
In this case, the default values are used for the structural members.

(3) Input the data in the Input window, and set the structural members.

(4) Call the melGetCurrentPrgBlock function using the [Execute] button.

(5) The Return Value window appears in the center of the screen, and the return value from melGetCurrentPrgBlock appears. The output from melGetCurrentPrgBlock also appears simultaneously in [Data Area] of the Input window. The Return Value window will close when the [OK] button is clicked.

(6) Click on the [Cancel] button in the melGetCurrentPrgBlock window to quit the melGetCurrentPrgBlock window. The Input window also closes simultaneously.



Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Number of blocks
Use to designate the number of blocks to be retrieved.

Pointer
"Memory Address of Data" appears in [lpPrgBlock]. This shows that the memory address of the variable where the program to be retrieved is stored is set in the lpPrgBlock argument.

Data input
Use to input data and structural member settings.
Refer to "4.5.8 Input window".

Data type
Use to designate the variable type where the program data will be input.

Fig. 4-26 melGetCurrentPrgBlock window

4.5.7.4 melSelectExecPrg

Calling the function

(1) Set the argument to be transferred to the melSelectExecPrg function.

(2) Click on the [Input] button. The window of the data type designated in lReadType appears in the center of the screen. When not inputting data, call the function without clicking on the [Input] button.

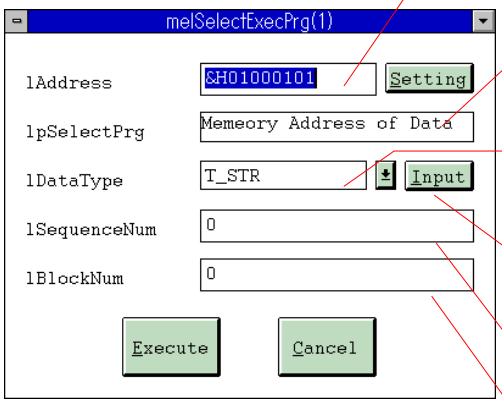
In this case, the default values are used for the structural members.

(3) Input the data in the Input window, and set the structural members.

(4) Call the melSelectExecPrg function using the [Execute] button.

(5) The Return Value window appears in the center of the screen, and the return value from melSelectExecPrg appears. The output from melSelectExecPrg also appears simultaneously in [Data Area] of the Input window. The Return Value window will close when the [OK] button is clicked.

(6) Click on the [Cancel] button in the melSelectExecPrg window to quit the melSelectExecPrg window. The Input window also closes simultaneously.



Address
Use to designate the NC Card, system, and axis. Use the [Setting] button when setting addresses individually.
Refer to "4.5.9.1 AddressSet window".

Pointer
"Memory Address of Data" appears in [lpSelectPrg].

Data type
Use to designate the variable type where the file name of the program to be operation searched will be stored.

Data input
Use to input data and structural member settings.
Refer to "4.5.8 Input window".

Sequence No.
Use to designate the sequence No. to be searched.

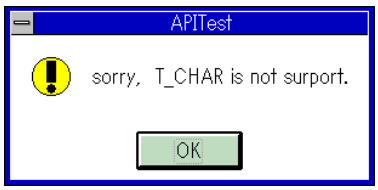
Block No.
Use to designate the block No. to be searched.

Fig. 4-27 melSelectExecPrg window

4.5.8 Input window

The structural members and data to be transferred to the API Function are set in the Input window. The Input window that opens will differ depending on the type set in the data type (lReadType, lWriteType, lDataType) API Function window.

Caution) T_CHAR data types cannot be used. If T_CHAR data type is designated, the window below will appear in the center of the screen when the [Input] button is clicked on or the API Function called.



4.5.8.1 Integer type, real number type

The Input window in Fig. 4-28 will open when T_SHORT, T_LONG, or T_DOUBLE is designated in the data type.

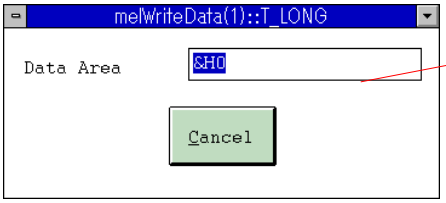
Inputting the data

(1) Set the data in [Data Area] when inputting.

(2) Return the focus to the API Function window.

(3) After API Function execution, the output from the function will appear in [Data Area].

(4) Clicking on the [Cancel] button in the API Function window quits the API Function window and closes the Input window. To only close the Input window, click on the [Cancel] button in the Input window.



Data area
Use to input the data to be transferred to the API Function.
After executing each API Function, the output from the function appears here.

Fig. 4-28 Input window of T_LONG Type opened from the melWriteData Function window

4.5.8.2 Character string type

The Input window in Fig. 4-29 will open when T_STR, T_DECSTR, T_HEXSTR, or T_BINSTR is designated in the data type.

Inputting the data

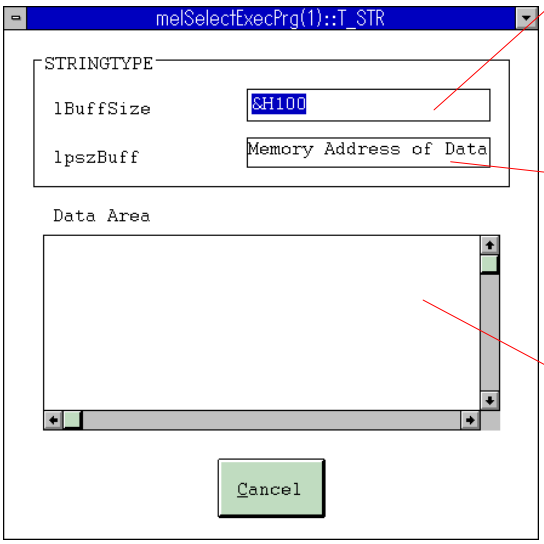
(1) Set the structural members in the setting area of the Input window.

(2) Set the data in [Data Area] when inputting.

(3) Return the focus to the API Function window.

(4) After API Function execution, the output from the function will appear in [Data Area].

(5) Clicking on the [Cancel] button in the API Function window quits the API Function window and closes the Input window. To only close the Input window, click on the [Cancel] button in the Input window.



Buffer size
Use to designate the buffer size of the data area. (30000 bytes or less)
Designate the buffer size including a 1-byte terminal character (Null).

Pointer
"Memory Address of Data" appears in [lpszBuff]. This shows that the memory address of the variable where the data is stored is set in the lpszBuff argument.

Data area
This area can be used to input data.
After each API Function execution, the output from the function will appear here.

Fig. 4-29 Input window of character string type opened from the melSelectExecPrg Function window

4.5.8.3 Real number character string type

The Input window in Fig. 4-30 will open when T_STR, T_DECSTR, T_HEXSTR, or T_BINSTR is designated in the data type.

Inputting the data

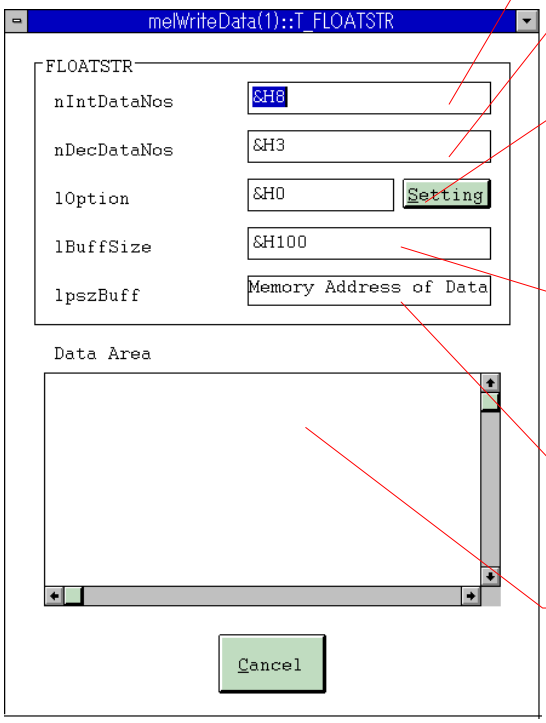
(1) Set the structural members in the setting area of the Input window.

(2) Set the data in [Data Area] when inputting.

(3) Return the focus to the API Function window.

(4) After API Function execution, the output from the function will appear in [Data Area].

(5) Clicking on the [Cancel] button in the API Function window quits the API Function window and closes the Input window. To only close the Input window, click on the [Cancel] button in the Input window.



Number of digits in the integer area
Use to set the number of digits in the integer area.

Number of digits in the decimal area
Use to set the number of digits in the decimal area

Option
Use to designate the option.
Use the [Setting] button when designating options individually. Refer to "4.5.9.3 OptionSet window".

Buffer size
Use to designate the buffer size of the data area. (30000 bytes or less)
Designate the buffer size including a 1-byte terminal character (Null).

Pointer
This shows that the memory address of the variable where the data is stored is set in the lpszBuff argument.

Data area
This area can be used to input data.
After each API Function execution, the output from the function will appear here.

Fig. 4-30 Input window of real number character string type opened from the melWriteData Function window

4.5.8.4 Special type

Inputting the data

(1) Set the structural members in the setting area of the Input window.

(2) Set the data in [Data Area] when inputting.

(3) Return the focus to the API Function window.

(4) After API Function execution, the output from the function will appear in [Data Area].

Caution) When the melGetCurrentPrgBlock function is executed, the blocks in the retrieved data appear in [iCurrentBlockNum] of the melGetCurrentPrgBlock window.

(5) Clicking on the [Cancel] button in the API Function window quits the API Function window and closes the Input window. To only close the Input window, click on the [Cancel] button in the Input window.

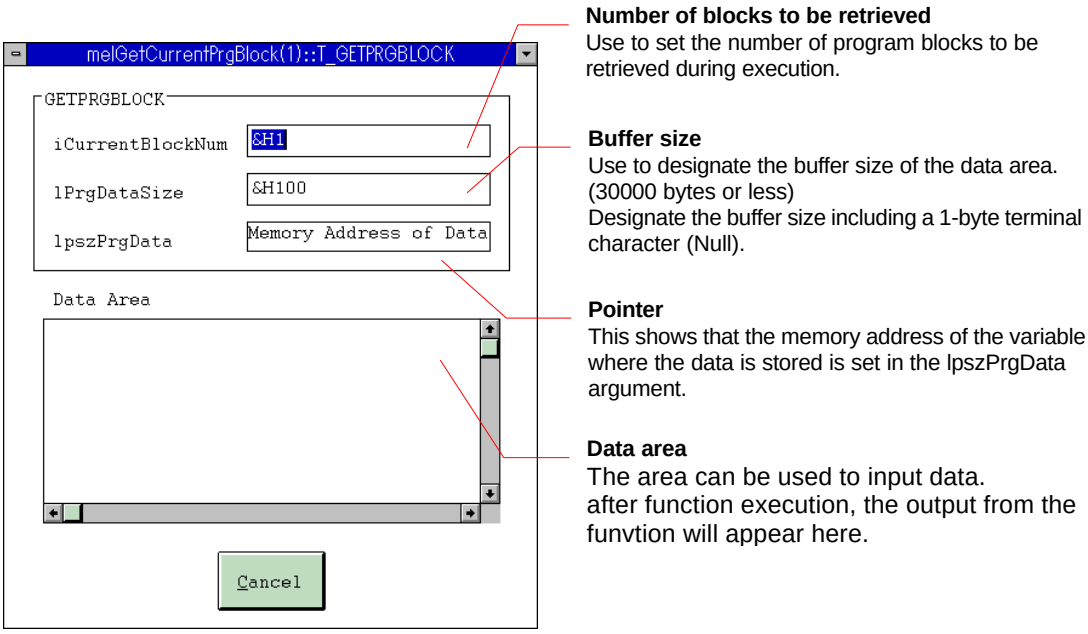


Fig. 4-31 Input window opened from the melGetCurrentPrgBlock Function window

4.5.9 Setting, Browse windows

4.5.9.1 AddressSet window

Setting the address

(1) Set NC Card, etc., items in the setting area of the AddressSet window.

(2) Click on the [OK] button. The focus returns to the API Function window, and the AddressSet window closes.

(3) The designated address appears in [lAddress] of the API Function window.

(4) Click on the [Cancel] button to cancel the settings.

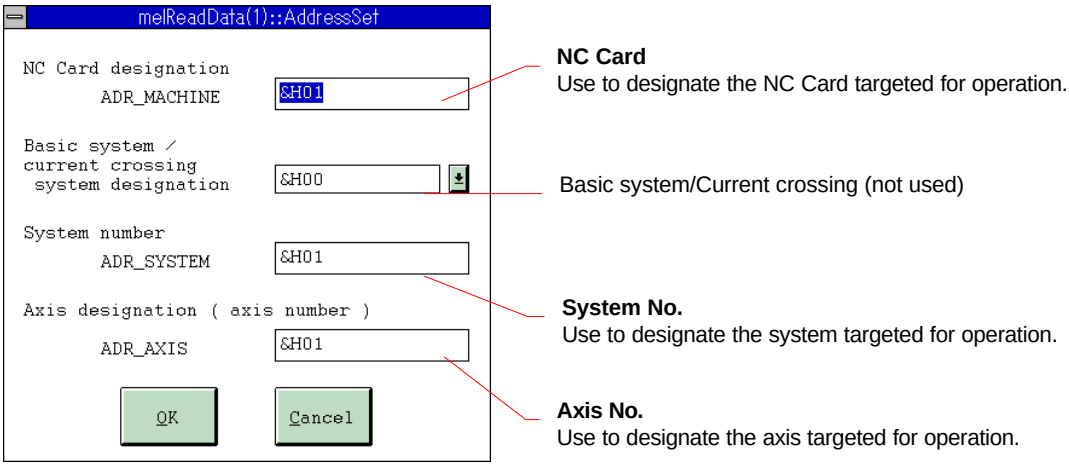


Fig. 4-32 AddressSet window

4.5.9.2 FileTypeSet window

Setting the file type

(1) Set format, etc., items in the setting area of the FileTypeSet window.

(2) Click on the [OK] button. The focus returns to the API Function window, and the FileTypeSet window closes.

(3) The designated file type appears in [lFileType] of the API Function window.

(4) Click on the [Cancel] button to cancel the settings.

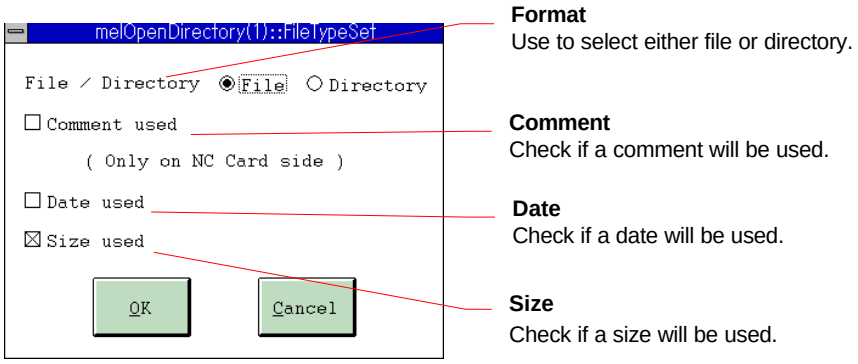


Fig. 4-33 FileTypeSet window

4.5.9.3 OptionSet window

Setting the option

(1) Set decimal zero suppression, etc., items in the setting area of the OptionSet window.

(2) Click on the [OK] button. The focus returns to the API Function window, and the OptionSet window closes.

(3) The designated option appears in [lOption] of the API Function window.

(4) Click on the [Cancel] button to cancel the settings.

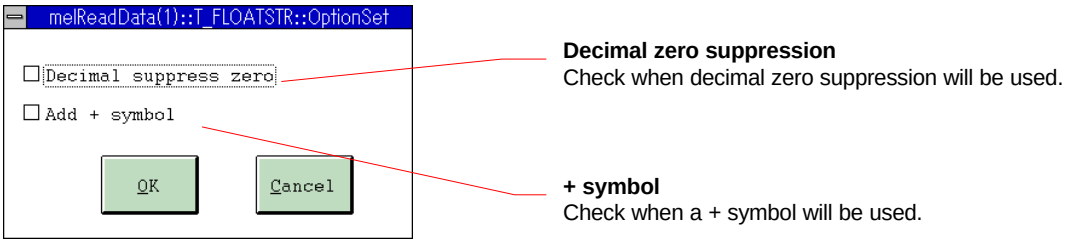


Fig. 4-34 OptionSet window

4.5.9.4 FileList window

Setting the file names

(1) Set drive, etc., items in the setting area of the FileList window.

(2) Click on the [OK] button. The focus returns to the API Function window, and the FileList window closes.

(3) The designated file name appears in file name setting area of the API Function window.

(4) Click on the [Cancel] button to cancel the settings.

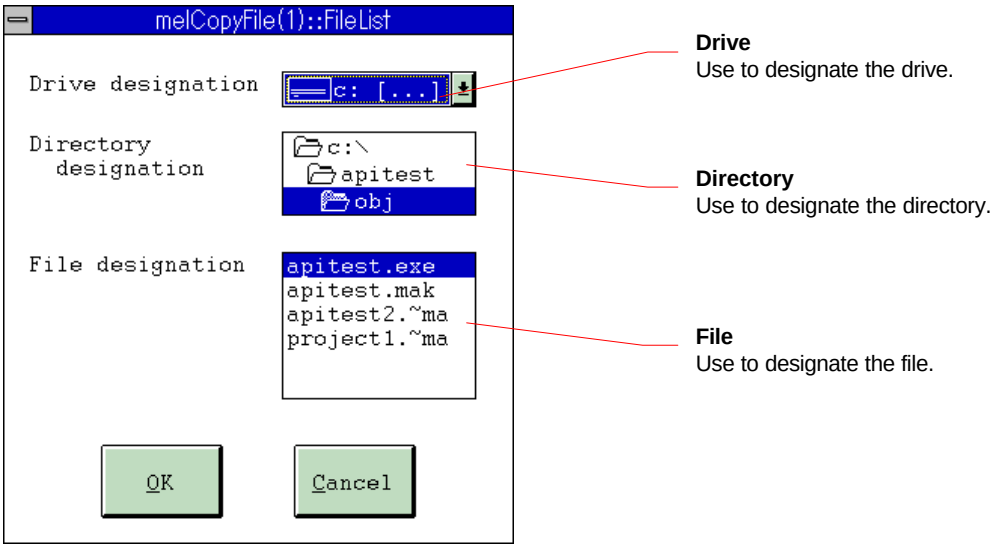


Fig. 4-35 FileList window

5. Restrictions

Appendix List of Sample Applications

Project name	Application name	Corresponding section
COUNTER32.VBP	Counter display application	3.2.2.1
PARAMET32.VBP	Parameter setting application	3.2.2.2
FILEMAN32.VBP	File transfer application	3.2.2.3
QCOUNTER32.VBP	High-speed counter display application	3.2.2.4
PRGMON32.VBP	Program in operation display application	3.2.2.5
PRGSRCH32.VBP	Operation search application	3.2.2.6
ALMMSG32.VBP	Alarm message display application	3.2.2.7

List of related documents

Setup Instruction Manual (BNP-B2191)
MELDASMAGIC MMI Operation Manual (D/M)(BNP-B2193)
MELDASMAGIC MMI Operation Manual (L/G)(BNP-B2194)
Custom Application Interface Library Guide (Programming section)(BNP-B2197)
Custom Application Interface Library Guide (Function section)(BNP-B2198)
Custom Application Interface Library Guide (Variable section)(BNP-B2199)

Index

Revision History

Sub-No.	Date of revision	Revision details
*	July, 1997	First edition created.

© 1996-1997 MITSUBISHI ELECTRIC CORPORATION
ALL RIGHTS RESERVED

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: MITSUBISHI DENKI BLD. MARUNOUCHI, TOKYO 100 TEL:03-218-3426