MELDASMAGIC is a registered trademark of Mitsubishi Electric Corporation.
Microsoft and Win32 are registered trademarks of Microsoft Corporation in the USA.
Windows is a trademark of Microsoft Corporation in the USA.
The other company names and product names are trademarks or registered trademarks of the respective companies.
Introduction

This instruction manual describes the Custom Application Interface (Custom API) Library used for developing the custom applications of MELDASMAGIC Series. The Custom API Functions used when developing the custom application are described in this manual, so please read this manual and the manuals below before programming.

Custom Application Interface Library Guide
   (Programming Section) ............................................ BNP-B2197
   (Variable Section) ..................................................... BNP-B2199

Please read the following "Precautions for Safety" to ensure safe use of the 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" and "Usable State" etc., the instruction manual issued by the maker takes precedence over this manual.

Items not described in this manual must be interpreted as "Not Possible".

This instruction manual has been written on the assumption that all options are provided. Check the specifications issued by the machine maker before starting use.

Some screens and functions may differ or may not be used depending on the NC system version.
# Contents

**Outline** .................................................................................................................. 1

**Installation of Custom API Function** ....................................................................... 2

**How to Read the Manual** ......................................................................................... 3

1. **API Function Basics** ............................................................................................. 4
   1.1 Designation of File Name ......................................................................................... 4
   1.2 Address ....................................................................................................................... 7
   1.3 Designation of Data Type .......................................................................................... 8
   1.4 Return Value of the Function ................................................................................... 9
   1.5 NC Data Access Variable ....................................................................................... 11

2. **Commands** ............................................................................................................ 12
   2.1 System Control Commands .................................................................................... 12
       melIoctl .......................................... (Execution of function) ................................... 12
   2.2 Commands Related to File Access .......................................................................... 14
       melCloseDirectory ....................... (Close directory) ............................................ 14
       melCopyFile ................................. (Copy file) ...................................................... 15
       melDeleteFile .............................. (Delete file) .................................................... 17
       melGetDiskFree .......................... (Get size of free space of drive/directory) .... 18
       melGetDriveList .......................... (Get drive list) ................................................ 19
       melOpenDirectory ...................... (Open directory) ............................................. 20
       melReadDirectory ...................... (Read file information) .................................. 22
       melRenameFile ........................... (Rename file) .................................................. 23
       melVerifyFile ............................ (Verify file) ..................................................... 24
   2.3 Commands Related to Data Access .......................................................................... 26
       melCancelModal......................... (Cancel registration of high-speed data) .............. 26
       melReadData ............................... (Read-out data) .............................................. 27
       melReadModal ............................. (High-speed read-out of data) ....................... 28
       melRegisterModal ...................... (Register high-speed read-out data) ............ 29
       melWriteData .............................. (Write data) .................................................... 30
   2.4 Commands Related to Operation ............................................................................ 31
       melActivatePLC ......................... (Start/stop PLC program).............................. 31
       melGetCurrentAlarmMsg............. (Get occurring alarm message) ..................... 32
       melGetCurrentPrgBlock ............... (Get running program) .................................. 33
       melSelectExecPrg ....................... (Operation search) ......................................... 34

3. **Appendix** .............................................................................................................. 35
   3.1 Command Name Table ............................................................................................. 35
   3.2 Error Codes ............................................................................................................... 36
Outline

This is a reference manual for the Custom Application Interface (Custom API) Library, one of the PC softwares for MELDASMAGIC Series. The Custom API Library is an interface library for the NC Card and applications developed by the user. In the Custom Application, the setting of and reference to data inside the NC Card as well as the operation of NC are possible through the API Functions described in this manual.

This library is provided in the format of DLL (Dynamic Link Library) for Windows.
Installation of Custom API Function

The Runtime Library of Custom API Library is necessary to use the Custom API function. Refer to the "MELDASMAGIC64 Custom Application Interface Library Guide (Programming Section) (BNP-B2191)" for an outline of the Runtime Library. Refer to "MELDASMAGIC64 Setup Instruction Manual (BNP-B2191)" for the installation method.

The files below, which are on the "Custom API Library" floppy disk, are necessary to create applications using the Custom API function.

• ncmcap.h          Custom API function definition file
• melncapi.h        Custom API function macro definition file
• meltype.h          Custom API data type definition file
• melerr.h           Custom API error code definition file
• melsberr.h         Custom API sub-error code definition file

Include the above mentioned header files into the source file. Refer to "MELDASMAGIC64 Custom Application Interface Library Guide (Programming Section) (BNP-B2191)" for the "Custom API Library" installation method.
How to Read the Manual

Organization

First, "1. API Function Basics" describes the basics of programming using the API Function. Next, "2. Commands" describes the commands of each command classification in alphabetical order. "3. Appendix" provides tables of all the command names and error codes listed in this manual. Indexes for "Basics", "Command classification" for command description and "Appendix" are indicated on the right-hand edge of each right page. When searching for specific items, refer to the upper corner of each page for the basic information item, command name or table name provided in each page.

How the commands are described

Each command description includes the following:
1. Simple description of the command following the command name
2. Command call method framed in bold line, with the variable name indicated in italics
3. Input, output data and details of function described, with the Item name indicated in Extra Boldface letters
4. "Reference" presenting related commands

Search index

Command name
Simple description of the function
Command call method
Input data (argument) Type and variable name: data contents
Output data (return value, others) Variable name: data contents
Detailed description of the function
Format
Related commands
Reference

---

Example command: melReadDirectory

```c
melReadDirectory(hWnd, dwDirectoryID, lpszFileInfo, lBuffSize);
```

**Input**

- **HWND hWnd**: Window handle for local window
- **DWORD dwDirectoryID**: ID of the directory of which information is to be read out (ID obtained by melOpenDirectory)
- **LPSTR lpszFileInfo**: Pointer for file information storage area
- **long lBuffSize**: Size of file information storage area (byte)

**Output**

- **dwStatus**: Size of the file information that has been read out or error code
  - 0: No file information
  - 1 or more: Byte number
  - ME_FILE_DIR_NOTOPEN: Not opened
  - ME_FILE_DIR_READ: File information read error
  - ME_FILE_DIR_DATASIZE: Data are too much to be stored in buffer secured by APP

**Note**

When an error occurs in the PC, error code ME_FILE_... becomes ME_PCFILE_....

- lpszFileInfo: File information

**Details of the function**

Reads out the file information of the directory currently opened by melOpenDirectory.

In melReadDirectory, one file information can be read per read out. By repeating melReadDirectory, the file name list of the directory designated in melOpenDirectory can be acquired.

The format of file information stored in the area indicated by lpszFileInfo is as follows:

```
file name  	 size  	 date  	 comment  
```

A TAB code is inserted between file name, size, date and comment. The NULL code is added to the end of the data.

Of the information following file name, only that designated in melOpenDirectory is stored.

For example, when "No date" is designated in melOpenDirectory, the format is as follows:

```
file name  	 size  	 comment
```

The "Comment used" designation in melOpenDirectory is invalid for files in which comments cannot be added, and the format becomes as follows:

```
file name  	 size  	 date  	 comment
```

When the file information read out exceeds lBuffSize, ME_FILE_DIR_DATASIZE is returned.

**Related commands**

- melCloseDirectory (Close directory)
- melOpenDirectory (Open directory)
1. API Function Basics

1.1 Designation of File Name

In Custom API Library, the NC Card is considered as a single drive, and various data on NC Card (machining program, tool offset etc.) are handled as files. When accessing a NC Card file with the Custom API Library, designate the file name with an absolute path as:

\texttt{drive name +":"+ directory name + file name}

\textbf{Example)} Designate the 123.PRG file in the \texttt{\PRG\ USER\} directory of the NC Card with the NC Card No. "01".

\textbf{(1) Designation of Drive Name}  
Communication in the Custom API Library is carried out with NC Card considered as a single drive. Each NC Card has a NC Card No. (established when NC Card is set up), and each drive name corresponds to NC Card No. indicated below.

\begin{tabular}{|c|c|}
\hline
\textbf{NC Card No.} & \textbf{Drive name} \\
\hline
01 & M01 \\
02 & M02 \\
03 & M03 \\
04 & M04 \\
05 & M05 \\
06 & M06 \\
07 & M07 \\
08 & M08 \\
: & : \\
: & : \\
FF & MFF \\
\hline
\end{tabular}
Designation of File Name

(2) Designation of Directory/File
The following is the directory configuration of the NC Card. Note that the directory does not accept additions or deletions.

\PRG\ USER\ Machining program storage directory
Program number.PRG : Machining program file
(User program, sub-program, macro program)
(Each machining program is considered as a separate file)

\PRG\ FIX\ Fixed cycle program storage directory
Program number.PRG : Fixed cycle program file
(Each fixed cycle program is considered as a separate file)

\PRG\ MDI\ MDI program storage directory
MDI.PRG : MDI program file

\PRM\ Parameter file directory
ALL.PRM : Parameter file
PARAMET.BIN : Parameter file (for maintenance)
FILESYS.BIN : File system file

\LAD\ PLC program directory
USERPLC.LAD : User PLC program file

\REG\ PLC device data directory
RREG.REG : R register data file
TREG.REG : T (timer) register data file
CREG.REG : C (counter) register data file

\DAT\ Work offset, tool offset, tool life data, common variable storage directory
WORK.OFS : Work offset data file
TOOL.OFS : Tool offset data file
TOOLLIFE.TLF : Tool life management data file
COMMON.VAR : Common variable data file
Group No. (8 digits). TL1 : Tool life management data file for each tool group.
Group No. (8 digits). TL2 : Tool life management data file for each tool group.

\LOG\ History data directory (for maintenance)
ILLEGAL.ERR
For some files, there are some restrictions in copying or compare operation.

**[Files that cannot be deleted]**

\( \text{\textbackslash PRM}\)
- Parameter file directory
  - ALL.PRM : Parameter file
  - PARAMET.BIN : Parameter file (for maintenance)
  - FILESYS.BIN : File system file

**[Files whose data is initialized or becomes the default when deleted]**

\( \text{\textbackslash PRG}\text{\textbackslash MDI}\)
- MDI program storage directory
  - MDI.PRG : Changes into file with file size 0.

\( \text{\textbackslash LAD}\)
- PLC program directory
  - USERPLC.LAD : Changes into PLC program with 0 step numbers.

\( \text{\textbackslash REG}\)
- PLC device data directory
  - RREG.REG : All R register data turns to 0.
  - TREG.REG : All T(timer) register data turns to 0.
  - CREG.REG : All C(counter) register data turns to 0.

\( \text{\textbackslash DAT}\)
- Work offset, tool offset, tool life data, common variable storage directory
  - WORK.OFS : All work offset data turns to 0.
  - TOOL.OFS : All tool offset data turns to 0.
  - TOOLLIFE.TLF : All tool life management data turns to 0.
  - COMMON.VAR : All common variable data turns to 0.

**[Files that bring some errors when files are overwritten]**

\( \text{\textbackslash PRG}\text{\textbackslash USER}\)
- Machining program storage directory
  - Program number.PRG : Machining program file
    - (User program, sub-program, macro program)
    - (Each machining program is considered as a separate file)

\( \text{\textbackslash PRG}\text{\textbackslash FIX}\)
- Fixed cycle program storage directory
  - Program number.PRG : Fixed cycle program file
    - (Each fixed cycle program is considered as a separate file)

\( \text{\textbackslash PRG}\text{\textbackslash MDI}\)
- MDI program storage directory
  - MDI.PRG : MDI program file

**[Files that can be verified]**

\( \text{\textbackslash PRM}\)
- Parameter file directory
  - ALL.PRM : Parameter file

\( \text{\textbackslash DAT}\)
- Workpiece offset, tool offset, tool life data, common variable storage directory
  - WORK.OFS : Workpiece offset data file
  - TOOL.OFS : Tool offset data file
  - COMMON.VAR : Common variable data file

Files other than those above cannot be verified. An error will result if verification of a file other than one of the above is attempted.
1.2 Address

In each function, the NC Card, system and axis for the operation are designated using the address, one of arguments. An address consists of 32-bit data with the following data structure:

Address : OOOOOOOO (long data)
- Axis designation (axis number)
- System number
- (Reserved)
- NC Card designation

When in actual use, use the following macro, and command the specified logical sum (|) that is necessary. Note that there are no rules for the order of logical sum calculation.

**Address designation macro**

- Axis designation : `ADR_AXIS (n)`
- System designation : `ADR_SYSTEM (n)`
- NC Card designation : `ADR_MACHINE (n)`

**Example**
- Designation of 2nd system and 1st axis
  
  `ADR_SYSTEM (2) | ADR_AXIS (1)`

**Example**
- Designation of NC Card 2nd and 6th axis
  
  `ADR_MACHINE (2) | ADR_AXIS (6)`

The following is the relationship between NC Card number and NC Card designation.

<table>
<thead>
<tr>
<th>NC Card No.</th>
<th>NC Card designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
</tr>
<tr>
<td>07</td>
<td>07</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

What needs to be designated in each address depends on each function.
1.3 Designation of Data Type

The data type used in each function includes the following:

<table>
<thead>
<tr>
<th>Data type</th>
<th>Data area</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) 1-byte integer type (T_CHAR)</td>
<td>char</td>
</tr>
<tr>
<td>(2) 2-byte integer type (T_SHORT)</td>
<td>short</td>
</tr>
<tr>
<td>(3) 4-byte integer type (T_LONG)</td>
<td>long</td>
</tr>
<tr>
<td>(4) 8-byte real number type (TDOUBLE)</td>
<td>double</td>
</tr>
</tbody>
</table>
| (5) Character string type (T_STR)  | struct STRINGTYPE {
|                                   | long IBuffSize;  // Buffer size of lpszBuff |
|                                   | LPSTR lpszBuff;  // Buffer pointer |
| (6) Decimal integer character string type (T_DECSTR) | STRINGTYPE |
| (7) Hexadecimal character string type (T HEXSTR) | STRINGTYPE |
| (8) Binary character string type (T_BINSTR) | STRINGTYPE |
| (9) Real number character string type (T_FLOATSTR) | struct FLOATSTR {
|                                   | short nIntDataNos; // Number of integer digits |
|                                   | short nDeciDataNos; // Number of decimal digits |
|                                   | long lOption; // Option |
|                                   | FLTSTR DEC1 ZERO SUPPRESS // Decimal suppress zero |
|                                   | FLTSTR ADD PLUS SIGN // + symbol |
|                                   | long IBuffSize; // Buffer size of lpszBuff |
|                                   | LPSTR lpszBuff; // Buffer pointer |
| (10) Special type: For getting current program (T_GETPRGBLOCK). | struct GETPRGBLOCK {
| Used in the mdGetCurrentPrgBlock. | int iCurrentBlockNum; // Current block |
|                                     | (Block in the acquired data) |
|                                     | 0: Operation not in progress |
|                                     | 1: 1st block |
|                                     | 2: 2nd block |
|                                     | long lPrgDatasize; // Buffer size of lpszPrgData |
|                                     | LPSTR lpszPrgData; // Pointer for buffer storing the program |

When using this type in each function, secure a data area in the application, and specify the pointer for the secured data area and data type in the argument.
1.4 Return Value of the Function

The return value of the API Function consists of 32-bit data with the following structure:

\[
\begin{array}{cccc}
31 & 27 & 16 & 0 \\
\uparrow & \uparrow & \uparrow & \uparrow \\
(1) & (2) & (3) & \\
\end{array}
\]

(1) Error flag  
0 : no error  
1 : error  
(2) Classification code  
(3) Return value code

1) When it is not an error
When each function is not an error, error flag is set to 0. 
Classification code and return value code depend on each function.

2) When it is an error
When each function is an error, error flag is set to 1. 
The classification code and return value code have the following meaning:

- Classification code: Indicates the error type
- High-order 8-bit of return value code: Indicates the type of function
- Low-order 8-bit of return value code: Error number

Error types consists of the following:

<table>
<thead>
<tr>
<th>Classification code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Command general errors</td>
</tr>
<tr>
<td>002</td>
<td>System related errors</td>
</tr>
<tr>
<td>003</td>
<td>File access related errors</td>
</tr>
<tr>
<td>004</td>
<td>Data access related errors</td>
</tr>
<tr>
<td>005</td>
<td>Operation related errors</td>
</tr>
<tr>
<td>100</td>
<td>NC Card system related errors</td>
</tr>
<tr>
<td>101</td>
<td>PC system related errors</td>
</tr>
<tr>
<td>200</td>
<td>2-ports communication related errors</td>
</tr>
<tr>
<td>201 (Not used)</td>
<td>Serial communication related errors</td>
</tr>
<tr>
<td>202 (Not used)</td>
<td>Ethernet communication related errors</td>
</tr>
<tr>
<td>203 ~ FFE</td>
<td>Spare</td>
</tr>
<tr>
<td>FFF</td>
<td>Others</td>
</tr>
</tbody>
</table>

For details on a specific error number, refer to the error code definition file (melerr.h).

With some API functions it is possible to retrieve a more detailed error code using the GetLastError () function of the Windows API. Error codes retrieved using GetLastError () are called sub-error codes.
The sub-error code consists of 32-bit data with the following structure.

```
<table>
<thead>
<tr>
<th>29</th>
<th>26</th>
<th>16</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td></td>
</tr>
</tbody>
</table>
(1) Error flag 0: no sub-error 1: sub-error
(2) Sub-error No.
(3) Additional information
```

(1) **When it is not an error**
   When each function is not an error, error flag is set to 0.

(2) **When it is an error**
   When there is a sub-error, the error flag is set to 1.
   The sub-error No. and additional information have the following meaning:

   - Sub-error No. : Error No.
   - Additional information : Error attribute information, etc.
     (Usage method differs with each API function)

Refer to the definition file (melsberr.h) of the sub-error code for concrete details about the sub-error.
Refer to the command explanation of each API function for information on whether the API function being used supports the sub-error.
1.5 NC Data Access Variable

In the Custom API Library, various data in NC Card are accessed as variables. These variables are called NC Data Access Variables.

The NC Data Access Variable indicates a single data with Section No. and Sub-section No. Depending on the variable, it may contain system and axis data. For such data, the data is specified by Section No., Sub-section No. and address designation.

For details on the NC Data Access Variables, refer to the section "Custom Application Interface Library Guide (Variable Section) BNP-B2143".
2. Commands

2.1 System Control Commands

**melIoctl**

This command executes functions such as mode and parameter setting and status information acquisition for a specified 2-port driver.

\[
dwStatus = \text{melIoctl} ( hWnd, \text{IAddress}, \text{IFunction}, \text{lpData} );
\]

**Input**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWND</td>
<td>hWnd: Window handle for local window</td>
</tr>
<tr>
<td>long</td>
<td>lAddress: Address</td>
</tr>
<tr>
<td>long</td>
<td>lFunction: Designate the following items:</td>
</tr>
<tr>
<td></td>
<td>• NC Card designation</td>
</tr>
<tr>
<td>LPVOID</td>
<td>lpData: Data</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWORD</td>
<td>dwStatus: Error code</td>
</tr>
<tr>
<td>ME_SYSFUNC_IOCTL_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_SYSFUNC_IOCTL_DATA</td>
<td>Data range illegal</td>
</tr>
<tr>
<td>ME_SYSFUNC_IOCTL_FUNCTION</td>
<td>Command illegal</td>
</tr>
<tr>
<td>ME_SYSFUNC_IOCTL_NOTOPEN</td>
<td>Device not open</td>
</tr>
</tbody>
</table>

**Details of the function**

The following is a list of and details of each function.

- Function list
  1. (1) DEV_SET_RTIMEOUT (0x001) Read timeout setting
  2. (2) DEV_SET_WTIMEOUT (0x002) Write timeout setting
  3. (3) DEV_CANCEL_RWAIT (0x003) Read wait state cancel
  4. (4) DEV_CANCEL_WWAIT (0x004) Write wait state cancel
Details of the function

(1) DEV_SET_RTIMEOUT (0x001) Read timeout setting
Sets the timeout value for reading in the message from reception cue.
Timeout does not occur when the timeout value is set to 0.
Instead, it waits until the message is read in from reception cue.
Default is 50 (5 seconds)

<table>
<thead>
<tr>
<th>Function</th>
<th>DEV_SET_RTIMEOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpData</td>
<td>Long type pointer</td>
</tr>
<tr>
<td>0</td>
<td>No timeout</td>
</tr>
<tr>
<td>1 or more</td>
<td>Timeout value (unit: 100ms)</td>
</tr>
</tbody>
</table>

(2) DEV_SET_WTIMEOUT (0x002) Write timeout setting
Sets the timeout value when transmission cue is full and the message cannot be written in.
Timeout does not occur when the timeout value is set to 0.
Instead, it waits until there is a space in the transmission cue.
Default is 50 (5 seconds).

<table>
<thead>
<tr>
<th>Function</th>
<th>DEV_SET_WTIMEOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpData</td>
<td>Long type pointer</td>
</tr>
<tr>
<td>0</td>
<td>No timeout</td>
</tr>
<tr>
<td>1 or more</td>
<td>Timeout value (unit: 100ms)</td>
</tr>
</tbody>
</table>

(3) DEV_CANCEL_RWAIT (0x003) Read wait state cancel
Cancels the read wait state. Read returns as an error. This function is insignificant when not in the read wait state. Use it to cancel a read that is in progress.

<table>
<thead>
<tr>
<th>Function</th>
<th>DEV_CANCEL_RWAIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpData</td>
<td>Not used (specify NULL)</td>
</tr>
</tbody>
</table>

(4) DEV_CANCEL_WWAIT (0x004) Write wait state cancel
Cancels the write wait state. Write returns as an error. This function is insignificant when not in the write wait state. Use it to cancel a write that is in progress.

<table>
<thead>
<tr>
<th>Function</th>
<th>DEV_CANCEL_WWAIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpData</td>
<td>Not used (specify NULL)</td>
</tr>
</tbody>
</table>
### 2.2 Commands Related to File Access

**melCloseDirectory**

This command closes a directory opened by `melOpenDirectory`.

```c
dwStatus = melCloseDirectory ( hWnd, dwDirectoryID ) ;
```

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWND hWnd</td>
<td>Window handle for local window</td>
</tr>
<tr>
<td>DWORD dwDirectoryID</td>
<td>ID of the directory to be closed (ID obtained by <code>melOpenDirectory</code>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWORD dwStatus</td>
<td>Error code</td>
</tr>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>ME_FILE_DIR_NOTOPEN</td>
<td>Directory not opened</td>
</tr>
</tbody>
</table>

**Note**: When an error occurs in the PC, error code `ME_FILE_ ...` becomes `ME_PCFILE_ .....`.

**Details of the function**: Closes the directory opened by `melOpenDirectory`.

**Reference**
- `melOpenDirectory` (Open directory)
- `melReadDirectory` (Read file information)
This command copies files.

```c
void melCopyFile(HWND hWnd, LPSRC_FILE_NAME lpszSrcFileName, LPSRC_FILE_NAME lpszDstFileName);
```

**Input**
- **HWND** hWnd: Window handle for local window
- **LPSTR** lpszSrcFileName: Copy source file name
- **LPSTR** lpszDstFileName: Copy destination file name

**Output**
- **DWORD** dwStatus: Error code
  - 0: Normal completion
  - 1: Copy source file cannot be read out
  - 2: Copy destination file cannot be written in
  - 3: File cannot be created (PC only)
  - 4: File cannot be opened (PC only)

**Details of the function**
Copies the file designated in lpszSrcFileName into the file designated in lpszDstFileName.
Designate file name with an absolute path as:
```
drive name + "" + directory name + file name
```
The file name can be omitted in lpszDstFileName. When the file name is omitted, it is copied with the same name.
lpszDstFileName should not be an existing file name.
Furthermore, the copy destination directory should already exist.

If an error occurs when copying a file, it is possible to retrieve more detailed error information. This detailed error information is called a sub-error code. The sub-error codes are retrieved using GetLastError of Windows API.
```
dwSubErrorCode = GetLastError (   ) ;
```
Using the retrieved sub-error code and the macro below, it is possible to retrieve the sub-error No. (dwSubErrorNum) and line of the file where the error occurred (dwErrorLineNum).
```
dwSubErrorNum = RetCopyFileSubError ( dwSubErrorCode ) ;
dwErrorLineNum = RetCopyFileErrorLine ( dwSubErrorCode ) ;
```
The sub-error Nos. and meanings are shown below.

**ME_FILE_COPY_READ** Sub-error No. when there is an error
- **ME_SUB_FILE_COPY_BUSY** : File output was attempted during data input/output

**ME_FILE_COPY_WRITE** Sub-error No. when there is an error
- **ME_SUB_FILE_COPY_BUSY** : File input was attempted during data input/output
- **ME_SUB_FILE_COPY_FORMAT** : Error in the format
- **ME_SUB_FILE_COPY_PARAMETERNO** : Error in the parameter No.
- **ME_SUB_FILE_COPY_TAG** : Error in the tag
- **ME_SUB_FILE_COPY_TOOLOFFSETNUM** : Error in the designation of the tool compensation No.
- **ME_SUB_FILE_COPY_TOOLOFFSETDATA** : Error in the command of the tool compensation data
- **ME_SUB_FILE_COPY_OFFSETTYPE** : Error in the designation of the offset type
- **ME_SUB_FILE_COPY_DATARANGE** : Error in the data range

This API function does not check if designated directory or file name is proper or not. Check directory or file name for such irregular operation as like; transferring between files that type or purpose differs from another (Ex. overwrite parameter file (PARAMET.BIN) on the user program file (\PRG\USER\_.PRG)), or copying a file to the directory in spite that has different purpose, etc.

**Reference**
- melDeleteFile (Delete file)
- melRenameFile (Rename file)
- melVerifyFile (File Verify)
This command deletes a designated file.

\[
dwStatus = \text{melDeleteFile}(\text{hWnd}, \text{lpszFileName});
\]

**Input**
- **HWND hWnd**: Window handle for local window
- **LPSTR lpszFileName**: File name

**Output**
- **DWPRD dwStatus**: Error code
  - 0: Normal completion
  - ME_FILE_DEL_BUSY: Now deleting this file not permitted (program running)
  - ME_FILE_DEL_FILESYSTEM: There is something wrong in file system
  - ME_FILE_DEL_ILLEGALNAME: Format for file name illegal
  - ME_FILE_DEL_NODIR: Directory does not exist
  - ME_FILE_DEL_NODRIVE: Drive does not exist
  - ME_FILE_DEL_NOFILE: File does not exist
  - ME_FILE_DEL_NOTDELETE: Deleting this file is not permitted

**(Note)** When an error occurs in the PC, error code ME_FILE_....... becomes ME_PCFILE_......

**Details of the function**
- Deletes the file designated in lpszFileName.
- Designate the file name with an absolute path as:
  - drive name + ":" + directory name + file name

**Reference**
- melCopyFile (Copy file)
- melRenameFile (Rename file)
- melVerifyFile (File Verify)
### melGetDiskFree

This command gets size of free space of the drive or directory.

```plaintext
dwStatus = melGetDiskFree( hWnd, lpszDirectoryName ) ;
```

**Input**
- HWND hWnd: Window handle for local window
- LPSTR lpszDirectoryName: Directory name

**Output**
- DWORD dwStatus: Size of free space or error code
  - 0 or more: Size of free space (bytes)
  - ME_FILE_DISKFREE_FILESYSTEM: There is something wrong in file system
  - ME_FILE_DISKFREE_NODIR: Directory does not exist
  - ME_FILE_DISKFREE_NODRIVE: Drive does not exist

**Details of the function**
Returns the size of free space of the directory designated in lpszDirectoryName.

Byte is the unit for indicating the size of free space.

Designate the directory with an absolute path as:
- `drive name + ":" + directory name`

When the drive of PC is designated as a drive name, the directory designation is ignored and the size of free space of the drive is returned.

When an NC Card is designated as a drive name, the size of free space of the designated directory is returned. When a sub-directory exists in the designated directory, the usage amount of the sub-directory is not included in the size of free space.

**Reference**
- melCloseDirectory (Close directory)
- melGetDriveList (Get drive list)
- melOpenDirectory (Open directory)
- melReadDirectory (Read file information)
This command gets the mounted NC Card as the drive list.

\[
dwStatus = melGetDriveList ( hWnd, lpszDriveList, lBuffSize ) ;
\]

**Input**
- HWND hWnd: Window handle of local window
- LPSTR lpszDriveList: Pointer for file information storage area
- long lBuffSize: Size of file information storage area (byte)

**Output**
- DWORD dwStatus: Size of the drive list acquired or error code
  - 0: No drive list
  - 1 or more: Byte number
  - ME_FILE_DRVLIST_DATASIZE: Data are too much to be stored in buffer secured by APP
  - ME_FILE_DRVLIST_READ: Drive list read error

**(Note)** When an error occurs in the PC, error code ME_FILE_ ..... becomes ME_PCFILE_ ..... 

* lpszDriveList: Drive list

**Details of the function**
Reads out the currently connected NC Card as the drive list.
The following is the format of the drive list stored in the area indicated by lpszDriveList:
- **drive name:** CRLF **drive name:** CRLF ... **drive name:** CRLF \0
  - CR and LF codes are inserted between the drive names.
  - CRLF, NULL codes are added to the end of the data.
If the drive list that is read out is more than the lBuffSize, lBuffSize worth of data is stored in the area indicated by lpszDriveList, and ME_FILE_DRVLIST_DATASIZE is returned.
It does not read the drive list of PC.

**Reference**
- melCloseDirectory (Close directory)
- melGetDiskFree (Get size of free space)
- melOpenDirectory (Open directory)
- melReadDirectory (Read file information)
This command opens a designated directory. The file information and directory information of the directory opened can be acquired using `melReadDirectory`.

```
dwStatus = melOpenDirectory ( hWnd, lpszDirectoryName, lFileType );
```

**Input**
- **HWND hWnd**: Window handle for local window
- **LPSTR lpszDirectoryName**: Directory name
- **long lFileType**: Designate the type and form of data to be read out by `melReadDirectory`

<table>
<thead>
<tr>
<th>31</th>
<th>16</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>(1) (2) (3) (4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) 0: File 1: Directory
(2) 0: No comment 1: Comment used (only on NC Card side)
(3) 0: No date 1: Date used (only on the personal computer side)
(4) 0: No size 1: Size used

**Output**
- **DWORD dwStatus**: ID of the directory opened or error code
  - 0 or more: Directory ID (used in `melReadDirectory` and `melCloseDirectory`)
  - `ME_FILE_DIR_ALREADYOPENED`: Another directory is already opened
  - `ME_FILE_DIR_FILESYSTEM`: There is something wrong in file system
  - `ME_FILE_DIR_NODIR`: Directory does not exist
  - `ME_FILE_DIR_NODRIVE`: Drive does not exist
  - `ME_FILE_DIR_NOTOPEN`: Directory not opened

(Note) When an error occurs in the PC, error code `ME_FILE_.....` becomes `ME_PCFILE_....."
**Details of the function**

Opens the designated directory. After the directory has been opened, its file and directory information can be acquired using `melReadDirectory`. The open state is valid until `melCloseDirectory` is executed.

When the directory is successfully opened, the directory ID is obtained. The directory ID is necessary when using `melReadDirectory` and `melCloseDirectory`. The directory ID is valid until `melCloseDirectory` is executed.

Only one directory can be opened at a time. Designate the directory with an absolute path as:

```
drive name + ":" + directory name + file name
```

When the file name is omitted, all of the files in the directory become the subject of read-out. It is possible to use "?" and "*" (wild cards) in the file name so that it corresponds with some files in the directory. Make sure to add \\"\" at the end of directory path when omitting the file name.

Both the NC Card and PC can be designated as the drive name. However, when designating the directory of PC, designation of "No comment" or "Comment used" in `lFileType` is invalid.

- **Example of directory designation**

  Example 1) Parameter directory designation of NC Card No.1
  
  M01:\PRM\  
  
  In `melReadDirectory`, all of the files following M01:\PRM\ become the object of readout.

  Example 2) Program directory designation of NC Card No.2

  M02:\PRG\USER\*.PRG

  In `melReadDirectory`, files following M02: \PRG\USER\ and possessing PRG become the object of readout.

**Reference**

- `melCloseDirectory` (Close directory)
- `melReadDirectory` (Read file information)
This command reads out the file information of the directory currently opened by melOpenDirectory.

\[
dwStatus = \text{melReadDirectory}(\text{hWnd}, \text{dwDirectoryID}, \text{lpszFileInfo}, \text{IBuffSize}) ;
\]

**Input**
- **HWND hWnd**: Window handle for local window
- **DWORD dwDirectoryID**: ID of the directory of which information is to be read out (ID obtained by melOpenDirectory)
- **LPSTR lpszFileInfo**: Pointer for file information storage area
- **long lBuffSize**: Size of file information storage area (byte)

**Output**
- **DWORD dwStatus**: Size of the file information that has been read out or error code
  - 0: No file information
  - 1 or more: Byte number
  - ME_FILE_DIR_DATASIZE: Data are too much to be stored in buffer secured by APP
  - ME_FILE_DIR_NOTOPEN: Not opened
  - ME_FILE_DIR_READ: File information read error

*(Note)* When an error occurs in the PC, error code ME_FILE_...... becomes ME_PCFILE_......

* lpszFileInfo: File information

**Details of the function**
Reads out the file information of the directory currently opened by melOpenDirectory. In melReadDirectory, one file information can be read per read out. By repeating melReadDirectory, the file name list of the directory designated in melOpenDirectory can be acquired.

The format of file information stored in the area indicated by lpszFileInfo is as follows:

```
file name \t size \t date \t comment \0
```

A TAB code is inserted between file name, size, date, and comment. The NULL code is added to the end of the data.

Of the information following file name, only that designated in melOpenDirectory is stored.

For example, when "No date" is designated in melOpenDirectory, the format is as follows:

```
file name \t size \t comment \0
```

The "Comment used" designation in melOpenDirectory is invalid for files in which comments cannot be added, and the format becomes as follows:

```
file name \t size \t date \0
```

When the file information read-out exceeds IBuffSize, ME_FILE_DIR_DATASIZE is returned.

**Reference**
- melCloseDirectory (Close directory)
- melOpenDirectory (Open directory)
melRenameFile

This command changes the file name.

\[
dwStatus = \text{melRenameFile}(\text{hWnd, lpszSrcFileName, lpszDstFileName})
\]

**Input**
- HWND hWnd: Window handle for local window
- LPSTR lpszSrcFileName: Old file name
- LPSTR lpszDstFileName: New file name

**Output**
- DWORD dwStatus: Error code
  - 0: Normal completion
  - ME_FILE_REN_BUSY: Now renaming not permitted (program running)
  - ME_FILE_REN_FILEEXIST: New file name already exists
  - ME_FILE_REN_FILESYSTEM: There is something wrong in file system
  - ME_FILE_REN_ILLEGALNAME: Format for file name illegal
  - ME_FILE_REN_NODIR: Directory does not exist
  - ME_FILE_REN_NODRIVE: Drive does not exist
  - ME_FILE_REN_NOFILE: File does not exist
  - ME_FILE_REN_NOTRENAME: Renaming this file is not permitted
  - ME_FILE_REN_SAMENAME: New name and old one of file are the same

*(Note)* When an error occurs in the PC, error code ME_FILE_..... becomes ME_PCFILE_.....

**Details of the function**
Changes the file name designated in lpszSrcFileName to file name designated in lpszDstFileName. Designate the old file name in lpszSrcFileName with an absolute path as:

\[
\text{drive name} + ":" + \text{directory name} + \text{file name}
\]
Designate only the new file name in lpszDstFileName excluding the drive name or directory name. lpszDstFileName cannot be an existing file name.

**Reference**
- melCopyFile (Copy file)
- melDeleteFile (Delete file)
- melVerifyFile (File Verify)
This command verifies the file name.

```c
dwStatus = melVerifyFile ( hWnd, lpszSrcFileName, lpszDstFileName );
```

**Input**
- **HWND** hWnd: Window handle for local window
- **LPSTR** lpszSrcFileName: Verification source file name (NC side file)
- **LPSTR** lpszDstFileName: Verification destination file name (PC side file)

**Output**
- **DWORD** dwStatus: Error code
  - 0: Normal completion
  - ME_FILE_VERIFY_DIFFER: Difference in the verification
  - ME_FILE_VERIFY_FILESYSTEM: There is something wrong in the file system
  - ME_FILE_VERIFY_ILLEGALNAME: Format for the file name is illegal
  - ME_FILE_VERIFY_NODIR: Directory does not exist
  - ME_FILE_VERIFY_NODRIVE: Drive does not exist
  - ME_FILE_VERIFY_NOFILE: File does not exist
  - ME_FILE_VERIFY_NOTSUPPORTED: Verification function is not supported
  - ME_FILE_VERIFY_NAMELength: File name is too long
  - ME_FILE_VERIFY_OPEN: File cannot be opened

*(Note)* When an error occurs on the personal computer side, ME_FILE_... of the error code becomes ME_PCFILE_... .

**Details of the function**
- Compares the file designated lpszSrcFileName and the file designated lpszDstFileName.
- Designate the file name with an absolute path as:
  ```c
  drive name + ":" + directory name + file name
  ```
- Designate the NC side file in lpszSrcFileName, and the PC side file in lpszDstFileName.

If an error occurs when verifying a file, it is possible to retrieve more detailed error information. This detailed error information is called a sub-error code. The sub-error codes are retrieved using GetLastError of Windows API.

```c
dwSubErrorCode = GetLastError ( );
```

Using the retrieved sub-error code and the macro below, it is possible to retrieve the sub-error No. (dwSubErrorNum) and line of the file where the error occurred (dwErrorLineNum).

```c
dwSubErrorNum = RetCopyFileSubError ( dwSubErrorCode );  
dwErrorLineNum = RetCopyFileErrorLine ( dwSubErrorCode );
```

The sub-error Nos. and meanings are listed below.
- **ME_FILE_VERIFY_DIFFER** Sub-error No. when there is an error
  - ME_SUB_FILE_COPY_NONE: Difference in the file
  - ME_SUB_FILE_COPY_FORMAT: Error in the format
  - ME_SUB_FILE_COPY_PARAMETERNO: Error in the parameter No.
  - ME_SUB_FILE_COPY_TAG: Error in the tag
  - ME_SUB_FILE_COPY_TOOLOFFSETNUM: Error in the designation of the tool compensation No.
  - ME_SUB_FILE_COPY_TOOLOFFSETDATA: Error in the command of the tool compensation data
ME_SUB_FILE_COPY_OFFSETTYPE : Error in the designation of the offset type
ME_SUB_FILE_COPY_DATARANGE : Error in the data range

Reference
melCopyFile (Copy file)
melDeleteFile (Delete file)
melRenameFile (Rename file)
2.3 Commands Related to Data Access

**melCancelModal**

This command cancels registration as high-speed read-out data.

\[
\text{dwStatus} = \text{melCancelModal} \left( \text{hWnd, lAddress, lModalId} \right);
\]

**Input**
- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address
  - Designate the following items:
    - NC Card designation
- **long lModalId**: ID of the data of which registration is to be cancelled
  - (ID obtained by **melRegisterModal**)

**Output**
- **DWORD dwStatus**: Error code
  - 0: Normal completion
  - ME_DATA_MDLCANCEL_ADDR: Address illegal
  - ME_DATA_MDLCANCEL_NOTREGIST: Modal ID not registered

**Details of the function**
Cancels high-speed read-out registration carried out by **melRegisterModal**.

**Designate the data ID to be cancelled in lModalId.**

**Reference**
- **melReadModal** (High-speed read-out of data)
- **melRegisterModal** (Register high-speed read-out data)
This command reads out data.

```
dwStatus = melReadData ( hWnd, lAddress, lSectionNum, lSubSectionNum, 
                          lpReadData, lReadType ) ;
```

**Input**
- HWND hWnd: Window handle for local window
- long lAddress: Address
  Designate the following items:
  * NC Card designation
  * System designation (Necessary for some data)
  * Axis designation (Necessary for some data)
- long lSectionNum: Section No.
- long lSubSectionNum: Sub-section No.
- LPVOID lpReadData: Pointer for the area in which data is to be stored
- long lReadType: Data type

**Output**
- DWORD dwStatus: Error code
  - 0: Normal completion
  - ME_DATA_READ_ADDR: Address illegal
  - ME_DATA_READ_DATASIZE: Data are too much to be stored in buffer secured by APP
  - ME_DATA_READ_DATATYPE: Data type illegal
  - ME_DATA_READ_READ: Now reading data not permitted
  - ME_DATA_READ_SECT: Section No. illegal
  - ME_DATA_READ_SUBSECT: Sub-section No. illegal
  - ME_DATA_READ_WRITEONLY: Write only data

* lpReadData: Data that has been read out

**Details of the function**
Reads out the data designated in lAddress, lSectionNum, lSubSectionNum.
The data that is read out is converted into a type designated in lReadType, and is stored in the area indicated by lpReadData.

**Reference**
melReadModal (High-speed read-out of data)
melWriteData (Write data)
This command executes high-speed data read-out.

\[ \text{dwStatus} = \text{melReadModal} \left( \text{hWnd}, \text{lAddress}, \text{lModalId}, \text{lpReadData}, \text{lReadType} \right) ; \]

**Input**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWND</td>
<td>hWnd</td>
</tr>
<tr>
<td>long</td>
<td>lAddress</td>
</tr>
<tr>
<td>long</td>
<td>lModalId</td>
</tr>
<tr>
<td>LPVOID</td>
<td>lpReadData</td>
</tr>
</tbody>
</table>

**Designate the following items:**

- NC Card designation

**Output**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWORD</td>
<td>dwStatus</td>
</tr>
</tbody>
</table>

- Normal completion
- Address illegal
- Data are too much to be stored in buffer secured by APP
- Data type illegal
- Modal ID not registered
- Now reading data not permitted
- Write only data

*lpReadData*: Data that has been read out

**Details of the function**

Reads out the high-speed read-out data registered by `melRegisterModal` to the area indicated by `lpReadData`.

Designate the data ID obtained by `melRegisterModal` in `lModalId`.

**Reference**

- `melCancelModal` (Cancel registration of high-speed read-out data)
- `melReadData` (Read-out data)
- `melRegisterModal` (Register high-speed read-out data)
This command registers as high-speed (constant) read-out data.

\[
dwStatus = \text{melRegisterModal}(\text{hWnd}, \text{lAddress}, \text{lSectionNum}, \text{lSubSectionNum}, \text{lPriority}) ;
\]

**Input**

- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address
  - Designate the following items:
    - NC Card designation
    - System designation (Necessary for some data)
    - Axis designation (Necessary for some data)
- **long lSectionNum**: Section No.
- **long lSubSectionNum**: Sub-section No.
- **long lPriority**: Priority order
  - 1: Highest
  - 2: High
  - 3: Middle
  - 4: Low

**Output**

- **DWORD dwStatus**: ID of the data registered or error code
  - 0 or more: Data ID (used in melCancelModal and melReadModal)
  - ME_DATA_MDLREGIST_ADDR: Address illegal
  - ME_DATA_MDLREGIST_PRIORITY: Priority designation illegal
  - ME_DATA_MDLREGIST_REGIST: Number of data registration over
  - ME_DATA_MDLREGIST_SECT: Section No. illegal
  - ME_DATA_MDLREGIST_SUBSECT: Sub-section No. illegal

**Details of the function**

Registers the data designated in lAddress, lSectionNum, and lSubSectionNum as high-speed read-out data.

When registered properly, melRegisterModal returns the ID for that data. The data can be read out at a high speed by melReadModal using this data ID.

An order of priority can be assigned to the registered data through lPriority. When a high priority is given, the data is processed preferentially in the NC Card. Thus, data needing a short updating cycle should be given a high priority. However, since the order of priority order is relative to the data registered, if there are too many data items registered at a high priority, it may lead to a delayed updating cycle.

The maximum number of data that can be registered for high-speed read-out is 128. An error occurs when more than 128 data are registered.

**Reference**

- melCancelModal (Cancel registration of high-speed read-out data)
- melReadModal (High-speed read-out of data)
This command writes in the data.

\[
dwStatus = \text{melWriteData} ( \text{hWnd}, \text{IAddress}, \text{ISectionNum}, \text{ISubSectionNum}, \\
\text{lpWriteData}, \text{IWriteType} ) ;
\]

**Input**
- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address
  - Designate the following items:
    - NC Card designation
    - System designation (Necessary for some data)
    - Axis designation (Necessary for some data)
- **long lSectionNum**: Section No.
- **long lSubSectionNum**: Sub-section No.
- **LPVOID lpWriteData**: Pointer for the data area
- **long lWriteType**: Data type

**Output**
- **DWORD dwStatus**: Error code
  - 0: Normal completion
  - ME_DATA_WRITE_ADDR: Address illegal
  - ME_DATA_WRITE_DATASIZE: Data are too much to be stored in buffer secured by APP
  - ME_DATA_WRITE_DATATYPE: Data type illegal
  - ME_DATA_WRITE_READONLY: Read only data
  - ME_DATA_WRITE_SECT: Section No. illegal
  - ME_DATA_WRITE_SUBSECT: Sub-section No. illegal
  - ME_DATA_WRITE_WRITE: Now writing data not permitted

**Details of the function**
Writes in the data designated in IAddress, ISectionNum, and ISubSectionNum.
Set the value to be written in at lpWriteData and designate the type of lpWriteData in lWriteType.

**Reference**
- melReadData (Read-out data)
2.4 Commands Related to Operation

**melActivatePLC**

Controls the startup and end of PLC program.

```
dwStatus = melActivatePLC ( hWnd, IAddress, IActivePLC ) ;
```

**Input**
- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address Designate the following items:
  - NC Card designation
- **long lActivePLC**: PLC operation mode

**Output**
- **DWORD dwStatus**: Error code
  - 0: Normal completion
  - ME_OPE_ACTPLC_ADDR: Address illegal
  - ME_OPE_ACTPLC_MODE: Mode illegal

**Details of the function**
Controls the startup and end of the PLC program in NC Card designated in lAddress.

The modes of lActivePLC are as follows:
- **M_OPE_ACTPLC_TRUE**: PLC startup
- **M_OPE_ACTPLC_FALSE**: PLC end
This command gets the currently occurring alarm message.

```c
DWORD dwStatus = melGetCurrentAlarmMsg(hWnd, lAddress, lMsgNos, lAlarmType, lpAlarmMsg, lReadType);
```

**Input**
- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address
  - Designate the following items:
    - NC Card designation
    - System Designation (all systems when 0 is set)
- **long lMsgNos**: Number of messages to be gotten
- **long lAlarmType**: Type of alarm to be gotten
- **LPVOID lpAlarmMsg**: Pointer for area in which message is to be stored
  ```c
  struct STRINGTYPE {
    long lBuffSize;  // Buffer size of lpszBuff
    LPSTR lpszBuff;  // Pointer for buffer in which message is to be stored
  };
  ```
- **long lReadType**: Data type of lpAlarmMsg (T_STR)

**Output**
- **DWORD dwStatus**: Error code
  - 0 : Normal completion
  - ME_OPE_CURRALM_ADDR : Address illegal
  - ME_OPE_CURRALM_ALMTYPE : Alarm type illegal
  - ME_OPE_CURRALM_DATAERR : There is an error in communication data between NC and PC
  - ME_OPE_CURRALM_DATASIZE : Data are too much to be stored in buffer prepared by APP
  - ME_OPE_CURRALM_DATATYPE : Data type illegal
  - ME_OPE_CURRALM_NOS : Number of getting messages illegal
  - * lpszBuff : Alarm message

**Details of the function**
Gets the alarm message of the currently occurring alarm in the NC Card designated in lAddress. Note that the alarm message is in the language designated in parameter (#1043 lang) of NC.

Designate the number of messages to be gotten within numbers 1 to 10.
Message is gotten in the order of highest priority to lowest priority.

The following is a list of alarm types.
- M_ALM_NC_ALARM : NC alarm
- M_ALM_STOP_CODE : Stop code
- M_ALM_PLC_ALARM : PLC alarm message
- M_ALM_OPE_MSG : Operator message
- M_ALM_ALL_ALARM : No alarm type classification

For the messages in lpszBuff, codes CR and LF are inserted between messages.
NULL is inserted at the end of the buffer.
melGetCurrentPrgBlock

This command gets the currently running program.

dwStatus = melGetCurrentPrgBlock ( hWnd, lAddress, lPrgBlockNos, lpPrgBlock, 
  lReadType );

Input

HWND hWnd : Window handle for local window
long lAddress : Address
Designate the following items:
  • NC Card designation
  • System designation
long lPrgBlockNos : Number of blocks to be gotten
LPVOID lpPrgBlock : Pointer for the area in which the gotten program is to be stored

struct GETPRGBLOCK {
    int iCurrentBlockNum; // Currently running block
        // (block in the gotten data)
            // 0: Not in operation
            // 1: First block
            // 2: Second block
    long lPrgDataSize; // Buffer size of lpszPrgData
    LPSTR lpszPrgData; // Pointer for the buffer in which the program is to be stored
};

long lReadType : Data type of lpPrgBlock (T_GETPRGBLOCK)

Output

DWORD dwStatus : Error code
0 : Normal completion
ME_OPE_GETPRGBLK_ADDR : Address illegal
ME_OPE_GETPRGBLK_DATAERR : There is an error in communication data between NC and PC
ME_OPE_GETPRGBLK_DATASIZE : Data are too much to be stored in buffer secured by APP
ME_OPE_GETPRGBLK_DATATYPE : Data type illegal
ME_OPE_GETPRGBLK_NOS : Number of blocks designation illegal
  * lpszPrgData : Program data

Details of the function

Reads out the currently running program block of the system designated in lAddress.
Designate the number of blocks to be gotten within numbers 1 to 10.
Codes CR and LF are inserted between the blocks of program stored in the buffer indicated by lpszPrgData. NULL is inserted at the end of the buffer.
The following are set when operation search is not carried out:
iCurrentBlockNum = 0
Buffer indicated by lpszPrgData = NULL
This command carries out operation search.

\[
dwStatus = \text{melSelectExecPrg} ( \text{hWnd}, \text{lAddress}, \text{lpSelectPrg}, \text{lDataType}, \text{lSequenceNum}, \text{lBlockNum} );
\]

**Input**

- **HWND hWnd**: Window handle for local window
- **long lAddress**: Address
  Designate the following items:
  - NC Card designation
  - System designation
- **LPVOID lpSelectPrg**: STRINGTYPE structure address
  Store the head address of the file name character string that operation searches in the STRINGTYPE structure.
- **long lDataType**: Data type of lpSelectPrg (specify T_STR)
- **long lSequenceNum**: Number of sequence to be searched
- **long lBlockNum**: Number of block to be searched

**Output**

- **DWORD dwStatus**: Error code
  - 0: Normal completion
  - ME_OPE_SELECTPRG_ADDR: Address illegal
  - ME_OPE_SELECTPRG_DATATYPE: Data type illegal
  - ME_OPE_SELECTPRG_FILESYSTEM: There is something wrong in file system
  - ME_OPE_SELECTPRG_NOPRG: Program file not found
  - ME_OPE_SELECTPRG_PGRFORMAT: Format for program file name illegal
  - ME_OPE_SELECTPRG_RUNNING: Program running
  - ME_OPE_SELECTPRG_LONGPATH: Path name is too long
  - ME_OPE_SELECTPRG_NCPCCOM: Could not communicate with the PC side file server
  - ME_OPE_SELECTPRG_TIMEOUT: Time out

**Details of the function**

Select the machining program to be executed by the NC Card designated in lAddress. Designate in lpSelectPrg, the name of program file to be searched. Designate the address of the STRINGTYPE structure in lpSelectPrg. Designate the head address of the file name character string in the lpszBuff of the STRINGTYPE structure. Only designate the file name when designating a program in the NC Card.

**Example**

Designate "2000.PRG" when designating the machining program of program No. 2000 in the NC Card.

When selecting the machining program in the NC Card, only the file in the \PRG\USER\ directory can be designated.

When designating a medium program other than the NC Card, the program name is designated by the absolute path that includes the drive name, path name, and program file name. (The personal computer direct operation function is necessary when designating a medium program other than the NC Card.)

**Example**

Drive name "C:"
Path name "\PARTPRG\MAIN\"
Program file name "1000.PRG"

In this case, designate "C:\PARTPRG\MAIN\1000.PRG"

Using lSequenceNum and lBlockNum, an operation search of the operation that designated the sequence No. and block No. is possible.

When lSequenceNum and lBlockNum are both 0, the program head is operation searched.
### 3.1 Command Name Table (alphabetical order)

<table>
<thead>
<tr>
<th>Command name</th>
<th>Function</th>
<th>Command classification</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>melActivatePLC</td>
<td>Start/stop PLC program</td>
<td>Operation</td>
<td>31</td>
</tr>
<tr>
<td>melCancelModal</td>
<td>Cancel registration of high-speed read-out data</td>
<td>Data Access</td>
<td>26</td>
</tr>
<tr>
<td>melCloseDirectory</td>
<td>Close directory</td>
<td>File Access</td>
<td>14</td>
</tr>
<tr>
<td>melCopyFile</td>
<td>Copy file</td>
<td>File Access</td>
<td>15</td>
</tr>
<tr>
<td>melDeleteFile</td>
<td>Delete file</td>
<td>File Access</td>
<td>17</td>
</tr>
<tr>
<td>melGetCurrentAlarmMsg</td>
<td>Get occurring alarm message</td>
<td>Operation</td>
<td>32</td>
</tr>
<tr>
<td>melGetCurrentPrgBlock</td>
<td>Get running program</td>
<td>Operation</td>
<td>33</td>
</tr>
<tr>
<td>melGetDiskFree</td>
<td>Get size of free space of drive/directory</td>
<td>File Access</td>
<td>18</td>
</tr>
<tr>
<td>melGetDriveList</td>
<td>Get drive list</td>
<td>File Access</td>
<td>19</td>
</tr>
<tr>
<td>melIoctl</td>
<td>Execution of function</td>
<td>System Control</td>
<td>12</td>
</tr>
<tr>
<td>melOpenDirectory</td>
<td>Open directory</td>
<td>File Access</td>
<td>20</td>
</tr>
<tr>
<td>melReadData</td>
<td>Read-out data</td>
<td>Data Access</td>
<td>27</td>
</tr>
<tr>
<td>melReadDirectory</td>
<td>Read file information</td>
<td>File Access</td>
<td>22</td>
</tr>
<tr>
<td>melReadModal</td>
<td>High-speed read-out of data</td>
<td>Data Access</td>
<td>28</td>
</tr>
<tr>
<td>melRegisterModal</td>
<td>Register high-speed read-out data</td>
<td>Data Access</td>
<td>29</td>
</tr>
<tr>
<td>melRenameFile</td>
<td>Rename file</td>
<td>File Access</td>
<td>23</td>
</tr>
<tr>
<td>melSelectExecPrg</td>
<td>Operation search</td>
<td>Operation</td>
<td>34</td>
</tr>
<tr>
<td>melVerifyFile</td>
<td>File verification</td>
<td>File Access</td>
<td>24</td>
</tr>
<tr>
<td>melWriteData</td>
<td>Write data</td>
<td>Data Access</td>
<td>30</td>
</tr>
</tbody>
</table>
### 3.2 Error Codes

The following is a list of error codes.

#### Error codes

**Command related errors**
- **ME_CMD_DIFFER** 0x80010004 Transmit command and reply command do not match
- **ME_CMD_ERRNUMGETFAIL** 0x80010005 Failed to retrieve an error No.
- **ME_CMD_NOCOMMAND** 0x80010002 No command
- **ME_CMD_NOOPTION** 0x80010001 No option
- **ME_CMD_SYNTAX** 0x80010003 Syntax error

**System related errors**
- **ME_SYSFUNC_IOCTL_ADDR** 0x80020190 Address illegal
- **ME_SYSFUNC_IOCTL_DATA** 0x80020133 Data range illegal
- **ME_SYSFUNC_IOCTL_FUNCTION** 0x80020132 Command illegal
- **ME_SYSFUNC_IOCTL_NOTOPEN** 0x80020102 Device not opened

**File access related errors**
- **ME_FILE_COPY_BUSY** 0x80030447 Now copying this file not permitted (program running)
- **ME_FILE_COPY_ENTRYOVER** 0x80030403 Number of entry over
- **ME_FILE_COPY_FILESYSTEM** 0x80030443 There is something wrong in file system
- **ME_FILE_COPY_ILLEGALNAME** 0x80030498 Format for file name illegal
- **ME_FILE_COPY_MEMORYOVER** 0x80030404 Memory capacity over
- **ME_FILE_COPY_NAMELENGTH** 0x80030448 File name is too long
- **ME_FILE_COPY_NODIR** 0x80030491 Directory does not exist
- **ME_FILE_COPY_NODRIVE** 0x8003049b Drive does not exist
- **ME_FILE_COPY_NOFILE** 0x80030442 File does not exist
- **ME_FILE_COPY_PLCRUN** 0x80030446 Now copying this file not permitted (PLC running)
- **ME_FILE_COPY_WRITE** 0x80030495 Copy source file cannot be read out
- **ME_FILE_COPY_WRITE** 0x80030495 Copy destination file cannot be written in
- **ME_PCFILE_COPY_CREATE** 0x800304a4 File cannot be created (PC)
- **ME_PCFILE_COPY_EXIST** 0x80030402 Copy destination file already exists (PC)
- **ME_PCFILE_COPY_ILLEGALNAME** 0x800304a7 Format for file name illegal (PC)
- **ME_PCFILE_COPY_NODIR** 0x800304a2 Directory does not exist (PC)
- **ME_PCFILE_COPY_NODRIVE** 0x800304a8 Drive does not exist (PC)
- **ME_PCFILE_COPY_NOFILE** 0x800304a1 File does not exist (PC)
- **ME_PCFILE_COPY_OPEN** 0x800304a3 File cannot be opened (PC)
- **ME_PCFILE_COPY_READ** 0x800304a5 Copy source file cannot be read out (PC)
- **ME_PCFILE_COPY_WRITE** 0x800304a6 Copy destination file cannot be written in (PC)

**File delete related errors**
- **ME_FILE_DEL_BUSY** 0x80030247 Now deleting this file not permitted (program running)
- **ME_FILE_DEL_FILESIST** 0x80030243 There is something wrong in file system
- **ME_FILE_DEL_ILLEGALNAME** 0x80030298 Format for file name illegal
- **ME_FILE_DEL_NAMELENGTH** 0x80030248 File name is too long
- **ME_FILE_DEL_NODIR** 0x80030291 Directory does not exist
- **ME_FILE_DEL_NODRIVE** 0x8003029b Drive does not exist
- **ME_FILE_DEL_NOFILE** 0x80030242 File does not exist
- **ME_FILE_DEL_NOTDELETE** 0x80030201 Deleting this file is not permitted
- **ME_PCFILE_DEL_ILLEGALNAME** 0x800302a7 Format for file name illegal (PC)
- **ME_PCFILE_DEL_NODIR** 0x800302a2 Directory does not exist (PC)
- **ME_PCFILE_DEL_NODRIVE** 0x800302a8 Drive does not exist (PC)
- **ME_PCFILE_DEL_NOFILE** 0x800302a1 File does not exist (PC)
- **ME_PCFILE_DEL_NOTDELETE** 0x80030202 Deleting this file is not permitted (PC)
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME_FILE_DIR_ALREADYOPENED 0x80030101</td>
<td>Another directory is already opened</td>
</tr>
<tr>
<td>ME_FILE_DIR_DATASIZE 0x80030103</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_FILE_DIR_FILESYSTEM 0x80030143</td>
<td>There is something wrong in file system</td>
</tr>
<tr>
<td>ME_FILE_DIR_NAMELENGTH 0x80030148</td>
<td>File name is too long</td>
</tr>
<tr>
<td>ME_FILE_DIR_NODIR 0x80030191</td>
<td>Directory does not exist</td>
</tr>
<tr>
<td>ME_FILE_DIR_NODRIVE 0x8003019b</td>
<td>Drive does not exist</td>
</tr>
<tr>
<td>ME_FILE_DIR_NOTOPEN 0x80030190</td>
<td>Directory not opened</td>
</tr>
<tr>
<td>ME_FILE_DIR_READ 0x80030194</td>
<td>File information read error</td>
</tr>
<tr>
<td>ME_PCFILE_DIR_NODIR 0x800301a2</td>
<td>Directory does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_DIR_NODRIVE 0x800301a8</td>
<td>Drive does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_DIR_NOFILE 0x800301a0</td>
<td>File does not exist</td>
</tr>
<tr>
<td>ME_PCFILE_DIR_NOTOPEN 0x800301a5</td>
<td>Directory not opened (PC)</td>
</tr>
<tr>
<td>ME_FILE_DISKFREE_FILESYSTEM 0x80030643</td>
<td>There is something wrong in file system</td>
</tr>
<tr>
<td>ME_FILE_DISKFREE_NAMELENGTH 0x80030648</td>
<td>File name is too long</td>
</tr>
<tr>
<td>ME_FILE_DISKFREE_NODIR 0x80030691</td>
<td>Directory does not exist</td>
</tr>
<tr>
<td>ME_FILE_DISKFREE_NODRIVE 0x8003069b</td>
<td>Drive does not exist</td>
</tr>
<tr>
<td>ME_PCFILE_DISKFREE_NODIR 0x800306a2</td>
<td>Directory does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_DISKFREE_NODRIVE 0x800306a8</td>
<td>Drive does not exist (PC)</td>
</tr>
<tr>
<td>ME_FILE_DRVLIST_DATASIZE 0x80030701</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_FILE_DRVLIST_READ 0x80030794</td>
<td>Drive list read error</td>
</tr>
<tr>
<td>ME_FILE_REN_BUSY 0x80030347</td>
<td>Now renaming not permitted (program running)</td>
</tr>
<tr>
<td>ME_FILE_REN_FILEEXIST 0x80030301</td>
<td>New file name already exists</td>
</tr>
<tr>
<td>ME_FILE_REN_FILESYSTEM 0x80030343</td>
<td>There is something wrong in file system</td>
</tr>
<tr>
<td>ME_FILE_REN_ILLEGALNAME 0x80030398</td>
<td>Format for file name illegal</td>
</tr>
<tr>
<td>ME_FILE_REN_NAMELENGTH 0x80030348</td>
<td>File name is too long</td>
</tr>
<tr>
<td>ME_FILE_REN_NODIR 0x80030391</td>
<td>Directory does not exist</td>
</tr>
<tr>
<td>ME_FILE_REN_NODRIVE 0x8003039b</td>
<td>Drive does not exist</td>
</tr>
<tr>
<td>ME_FILE_REN_NOFILE 0x80030342</td>
<td>File does not exist</td>
</tr>
<tr>
<td>ME_FILE_REN_NOTRENAME 0x80030303</td>
<td>Renaming this file is not permitted</td>
</tr>
<tr>
<td>ME_FILE_REN_SAMENAME 0x80030305</td>
<td>New name and old one of file are the same</td>
</tr>
<tr>
<td>ME_PCFILE_REN_FILEEXIST 0x800303a7</td>
<td>Format for file name illegal (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_REN_ILLEGALNAME 0x800303a2</td>
<td>Directory does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_REN_NODIR 0x800303a8</td>
<td>Drive does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_REN_NODRIVE 0x800303a1</td>
<td>File does not exist (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_REN_NOFILE 0x80030304</td>
<td>Renaming this file is not permitted (PC)</td>
</tr>
<tr>
<td>ME_PCFILE_REN_SAMENAME 0x80030306</td>
<td>New name and old one of file are the same (PC)</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_DIFFER 0x80030805</td>
<td>Verification error</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_FILESYSTEM 0x80030843</td>
<td>There is something wrong in the file system</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_ILLEGALNAME 0x80030898</td>
<td>Format for file name illegal</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NAMELENGTH 0x80030891</td>
<td>File name is too long</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NODIR 0x8003089b</td>
<td>Directory does not exist</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NODRIVE 0x80030894</td>
<td>Drive does not exist</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NOFILE 0x80030892</td>
<td>File does not exist</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NOTSUPPORTED 0x80030849</td>
<td>Verification function is not supported</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_READ 0x80030894</td>
<td>Verification file cannot be read</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NODIR 0x8003089d</td>
<td>Directory does not exist (PC)</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NODRIVE 0x80030897</td>
<td>Drive does not exist (PC)</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_NOFILE 0x80030896</td>
<td>File does not exist (PC)</td>
</tr>
<tr>
<td>ME_FILE_VERIFY_OPEN 0x80030893</td>
<td>File cannot be opened (PC)</td>
</tr>
</tbody>
</table>
### Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME_DATA_MDLCANCEL_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLCANCEL_NOTREGIST</td>
<td>Modal ID not registered</td>
</tr>
<tr>
<td>ME_DATA_MDLREAD_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLREAD_DATASIZE</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_DATA_MDLREAD_DATATYPE</td>
<td>Data type illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLREAD_READ</td>
<td>Now reading data not permitted</td>
</tr>
<tr>
<td>ME_DATA_MDLREAD_WRITEONLY</td>
<td>Write only data</td>
</tr>
<tr>
<td>ME_DATA_MDLREGIST_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLREGIST_PRIORITY</td>
<td>Priority designation illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLREGIST_REGIST</td>
<td>Number of data registration over</td>
</tr>
<tr>
<td>ME_DATA_MDLREGIST_SECT</td>
<td>Section No. illegal</td>
</tr>
<tr>
<td>ME_DATA_MDLREGIST_SUBSECT</td>
<td>Sub-section No. illegal</td>
</tr>
<tr>
<td>ME_DATA_READ_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_DATA_READ_DATASIZE</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_DATA_READ_DATATYPE</td>
<td>Data type illegal</td>
</tr>
<tr>
<td>ME_DATA_READ_READ</td>
<td>Now reading data not permitted</td>
</tr>
<tr>
<td>ME_DATA_READ_SUBSECT</td>
<td>Sub-section No. illegal</td>
</tr>
<tr>
<td>ME_DATA_READ_WRITEONLY</td>
<td>Write only data</td>
</tr>
<tr>
<td>ME_DATA_WRITE_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_DATA_WRITE_DATASIZE</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_DATA_WRITE_DATATYPE</td>
<td>Data type illegal</td>
</tr>
<tr>
<td>ME_DATA_WRITE_READONLY</td>
<td>Read only data</td>
</tr>
<tr>
<td>ME_DATA_WRITE_SECT</td>
<td>Section No. illegal</td>
</tr>
<tr>
<td>ME_DATA_WRITE_SUBSECT</td>
<td>Sub-section No. illegal</td>
</tr>
<tr>
<td>ME_DATA_WRITE_WRITE</td>
<td>Now writing data not permitted</td>
</tr>
<tr>
<td>ME_OPE_ACTPLC_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_OPE_ACTPLC_MODE</td>
<td>Mode illegal</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_ALMTYPE</td>
<td>Alarm type illegal</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_DATAERR</td>
<td>There is an error in communication data between NC and PC</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_DATASIZE</td>
<td>Data are too much to be stored in buffer prepared the application</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_DATATYPE</td>
<td>Data type illegal</td>
</tr>
<tr>
<td>ME_OPE_CURRALM_NOS</td>
<td>Number of getting messages illegal</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_ADDR</td>
<td>Address illegal</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_DATAERR</td>
<td>There is an error in communication data between NC and PC</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_DATASIZE</td>
<td>Data are too much to be stored in buffer secured the application</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_DATATYPE</td>
<td>Data type illegal</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_NOS</td>
<td>Number of blocks designation illegal</td>
</tr>
<tr>
<td>ME_OPE_GETPRGBLK_NOSEARCH</td>
<td>Operation search not completed</td>
</tr>
</tbody>
</table>
**Error Codes**

---

**NC Card system related errors**

- `ME_SYS_MALLOC` 0x81008001 Work area cannot be secured
- `ME_SYS_SYSTEMDOWN` 0x81000102 NC Card is now system down

**PC system related errors**

- `ME_PCSYS_EXECCOMTASK` 0x81010101 Communication Task cannot be started
- `ME_PCSYS_MALLOC` 0x81018001 Work area cannot be secured
- `ME_PCSYS_MLOCK` 0x81018002 Work area illegal
- `ME_PCSYS_SEMAPHRE` 0x81018003 Exclusive control not possible
- `ME_PCSYS_SEMAPHRETIMEOUT` 0x81010114 Exclusive control time out

**2-ports memory communication related errors**

- `ME_SMEM_ALREADYOPEN` 0x82000001 Already opened
- `ME_SMEM_BADCHANNEL` 0x82000006 Channel No. illegal
- `ME_SMEM_BADFD` 0x82000007 File descriptor illegal
- `ME_SMEM_CANCELED` 0x82000016 Quitted because of cancel required
- `ME_SMEM_CANNOTOPEN` 0x82000008 Cannot open
- `ME_SMEM_CARDNOTEXIST` 0x82000004 Card not exist
- `ME_SMEM_ILLEGALSIZE` 0x82000017 Packet size illegal
- `ME_SMEM_NOTOPEN` 0x82000002 Not opened
- `ME_SMEM_TASKQUIT` 0x82000018 Quitted because of task quit
- `ME_SMEM_TIMEOUT` 0x82000014 Time out
- `ME_SMEM_UNKNOWNFUNC` 0x82000032 Command illegal

**Others**

**List of sub-error Nos.**

**File access related errors**

- `ME_SUB_FILE_COPY_NONE` 0 No sub-error: Download, etc., while operating
- `ME_SUB_FILE_COPY_BUSY` 1 File input/output attempted while data inputting/outputting
- `ME_SUB_FILE_COPY_FORMAT` 10 Error in the format
- `ME_SUB_FILE_COPY_PARAMETERNO` 11 Error in the parameter No.
- `ME_SUB_FILE_COPY_TAG` 12 Error in the tag
- `ME_SUB_FILE_COPY_TOOLOFFSETNUM` 13 Error in the designation of the tool compensation No.
- `ME_SUB_FILE_COPY_TOOLOFFSETDATA` 14 Error in the command of the tool compensation data
- `ME_SUB_FILE_COPY_OFFSETTYPE` 15 Error in the designation of the offset type
- `ME_SUB_FILE_COPY_DATARANGE` 20 Error in the data range
- `ME_SUB_FILE_COPY_IDDIFFER` 50 File with incorrect data ID was input
## Revision History

<table>
<thead>
<tr>
<th>Sub-No.</th>
<th>Date of revision</th>
<th>Revision details</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>July, 1997</td>
<td>First edition created.</td>
</tr>
</tbody>
</table>