NELIS$ - Extended List Windows Directory

The NELIS$ routine is used to list the contents of the directory on the host operating system (normally Windows) previously opened using the NEOPN$ routine.

1.   Invocation

To list the directory code:

```
CALL NELIS$ USING area de
```

where *area* is the PIC X(400) work area previously passed to the NEOPN$ routine and *de* is a block containing the returned file information:

```
01   DE
  02 DELENG      PIC 9(4) COMP      * Number of fields returned
  02 DENAME      PIC X(256)         * Filename
  02 DESIZE      PIC 9(9) COMP      * File size
  02 DEDATE      PIC DATE           * Creation date
  02 DETIME      PIC 9(9) COMP      * Creation time
  02 DETYPE      PIC 9 COMP         * File Type (see below)
```

2.   STOP Codes and Exception Conditions

The following STOP codes may be generated by NELIS$:

STOP code	Description
24001	The file name read by the NELIS$ routine exceeds the maximum length expected.

The following EXIT codes may be returned by NELIS$:

EXIT code	$$COND	Description
24001	01	An unexpected error condition has been returned by the host operating system. The error code will be returned in $$CRES.
24002	02	The end of directory has been reached.
24003	03	The data returned to the NELIS$ routine by the host operating system is invalid.

3.   Programming Notes

The NELIS$ routine must be used in conjunction with the NEOPN$ and NECLS$ routines.

The NELIS$ routine has been modelled on the traditional LIST$ routine. NELIS$ is an extended version of NLIST$.

The PIC X(400) work-area must not be used for any other routines apart from the preceding NEOPN$ call and the subsequent NECLS$ calls, until the final NECLS$ has completed. In particular, it must not be used for any nested NEOPN$ calls.

NELIS$ should be called repeatedly to return each file in the directory in turn until the End of Directory exception has been returned.

When no more files that match the wildcard spec are detected the exception from NELIS$ depends on what's already been returned. If one, or more files, have been returned from previous calls on NELIS$ then the documented End-of-Directory exception ($$COND=2) is returned by NELIS$. However, if no files match the wildcard spec, NELIS$ will return $$COND=1 with $$CRES=2 (ERROR_FILE_NOT_FOUND). For example, consider a folder that just contains:

>   C:\test\File1.jpg
>   C:\test\File2.jpg

a call of NEOPN$ with a target filename of "c:\test*.jpg" will be successful. Subsequent, calls of NELIS$ will return success, success, $$COND=2.

However, a call of NEOPN$ with a target filename of "c:\test*.xxx" (when no such files exist) will also be successful. The 1st subsequent call of NELIS$ will return an immediate $$COND=1/$$CRES=2.

Earlier versions of this document described the possible DETYPE values as follows:

>   0 = directory
>   1 = normal file
>   2 = hidden file
>   3 = system file
>   4 = hidden & system file
>   5 = not documented

The actual meanings for the returned value in DETYPE are:

0 = normal directory
1 = hidden & system directory !!! (which doesn't happen very often)
2 = normal file
3 = hidden file
4 = system file
5 = hidden & system file

Note that 2 other possible combinations:

>   Hidden directory
>   System directory

return a $$COND=3 from NELIS$ (the data returned to the NELIS$ routine by the host operating system is invalid).

Most applications are only interested in distinguishing between a folder or a file, and are not concerned with the precise nature of the file, so should be coded "defensively" as follows:

```
      CALL NELIS$ USING AREA DE
      ON NO EXCEPTION
        IF DETYPE ZERO                    * ENTRY IS FOLDER/DIRECTORY
      * folder/directory processing
        ELSE                              * ELSE A FILE (NORMAL, HIDDEN ETC.)
      * file processing
        END
      END
```

For GSM SP-33, and later, a new interface, NELI2$, has been added to supplement NELIS$. For NELI2$, the following values are returned in DETYPE:

> 0 = directory (normal)
> 1 = normal file
> 2 = hidden file
> 3 = system file
> 4 = hidden and system file
> 5 = hidden directory
> 6 = system directory
> 7 = hidden, system directory

The order in which the search returns the files, such as alphabetical order, is not guaranteed, and is dependent on the file system. If the data must be sorted, the application must do the ordering after obtaining all the results. The order in which this function returns the file names is dependent on the file system type. With the NTFS file system and CDfs file systems, the names are usually returned in alphabetical order. With FAT file systems, the names are usually returned in the order the files were written to the disk, which may or may not be in alphabetical order. However, as stated previously, these behaviours are not guaranteed.

4.   Examples
[EXAMPLES REQUIRED]

5.   Copy-Books
None.

6.   See Also
NOPEN$ Open Windows Directory
NLIST$ List Windows Directory
NLIS2$ List Windows Directory (Normalised File Type)
NCLOS$ Close Windows directory
NEOPN$ Extended Open Windows Directory
NELI2$ Extended List Windows Directory (Normalised File Type)
NECLS$ Extended Close Windows Directory
NXOPN$ Specialised Open Windows Directory
NXLIS$ Specialised List Windows Directory
NXCLS$ Specialised Close Windows Directory
OPEN$ Open Global volume
LIST$ List Global volume
CLOSE$ Close Global volume