blitz

COLLABORATORS			
	TITLE : blitz		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 31, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

blitz

1.1 blitz.doc

blitz.library

AddTokenUpdate()

AllocXtra()

DelTokenUpdate()

DetokeLine()

DetokeMem()

FindToken()

FreeBlitzLibs()

FreeTokeMem()

FreeXtra()

GetBlitzLibInfo()

GetFirstToken()

GetLineHeaderSize()

GetObjectMaximum()

GetObjectName()

GetVersion()

LoadBlitzLibs()

LoadFile()

LoadXtra()

```
            LockBlitzLibs()

            NewDetokeLine()

            NumMaximums()

            SaveXtra()

            SetObjectMaximum()

            SortTokens()

            TokeLine()

            TokeMem()

            UnLockBlitzLibs()
```

1.2 blitz.library/AddTokenUpdate()

```
                NAME
    AddTokenUpdate -- install a notification hook for command-library updates.

SYNOPSIS
    AddTokenUpdate(functionPtr);
                   A0

FUNCTION
    This command lets you automatically have a function called whenever the
    blitz command libraries are updated (i.e. calls to
            FreeBlitzLibs()
            ,

            LoadBlitzLibs()
             etc).

    Any registers may be treated as scratch registers.

INPUTS
    functionPtr - address of routine to call

RESULT
    NONE

BUGS
    A maximum of 16 notification hooks can be installed. No checking is
    performed to ensure that the limit is not exceeded.

SEE ALSO

            DelTokenUpdate()
```

1.3 blitz.library/AllocXtra()

```
                NAME
    AllocXtra -- allocate and initialise a valid BlitzXtra structure.

SYNOPSIS
    xtraFile = AllocXtra();
    D0

    struct BlitzXtra * AllocXtra(VOID);

FUNCTION
    This command allocates a new BlitzXtra structure, filling fields
    with default values.

INPUTS
    NONE

RESULT
    xtraFile - an initialized BlitzXtra structure, or NULL on failure.

SEE ALSO

            FreeXtra()
            , <libraries/blitz.h>, <libraries/blitz.i>
```

1.4 blitz.library/DelTokenUpdate()

```
                NAME
    DelTokenUpdate -- remove a previously created notification hook for
                      command-library updates.

SYNOPSIS
    DelTokenUpdate(functionPtr);
                   A0

FUNCTION
    This command removes a notification request, as created by a previous
    call to
            AddTokenUpdate()
            .

INPUTS
    functionPtr - address of routine to remove from notification list

RESULT
    NONE

SEE ALSO

            AddTokenUpdate()
```

1.5 blitz.library/DetokeLine()

```
NAME
    DetokeLine -- detokenise a line containing valid tokenised Blitz source
                  code.

SYNOPSIS
    success = DetokeLine(tokenSource,asciiDest);
    D0                   A0          A1

    BOOL = DetokeLine(char * tokenSource, char * asciiDest);

FUNCTION
    This command will convert a line containing tokenised source into its
    detokenised equivalent, placing the resulting string in the destination
    buffer.

INPUTS
    tokenSource - a pointer to a line containing tokenised source
                  (NOTE: the line should not contain any headers)

    asciiDest - a pointer to a line buffer where the detokenised line is to
                be placed

RESULT
    success - indication of success or failure

NOTES
    asciiDest should be at least 128 bytes in size. This routine will not
    create a detokenised string of more than 128 bytes.
```

1.6 blitz.library/DetokeMem()

```
                NAME
    DetokeMem -- detokenise a block of memory containing valid tokenised
                 Blitz source code.

                 ** THIS FUNCTION IS NOT YET IMPLEMENTED **

SYNOPSIS
    success = DetokeMem(tokenSource,asciiDest,length);
    D0                  A0          A1         D0

    BOOL =
             DetokeLine
             (char * tokenSource, char * asciiDest, ULONG length);

FUNCTION
    This function will detokenise a block of memory into its detokenised
    ASCII equivalent, placing the resultant data in the destination buffer.

INPUTS
    tokenSource - a pointer to a block of memory containing tokenised source
```

```
    asciiDest - a pointer to the destination ASCII buffer. This must be at
    least as big as numlines * 128 in size

    length - the size of the source block

RESULT
    success - indication of success or failre

NOTES
    ** THIS FUNCTION IS NOT YET IMPLEMENTED **
```

1.7 blitz.library/FindToken()

```
NAME
    FindToken -- retrieve the location inside the Blitz command libraries
                 of a given token.

SYNOPSIS
    tokenptr = FindToken(token);
    A3                   D0:16

    struct BlitzToken * tokenptr = FindToken(UWORD token);

FUNCTION
    This function will traverse the Blitz command libraries. If the given
    token is found, the address of the token is returned - else a standard
    ERROR token.

INPUTS
    token - a 2 byte token number

RESULT
    tokenptr - a pointer to the token (or an ERROR token)
```

1.8 blitz.library/FreeBlitzLibs()

```
                NAME
    FreeBlitzLibs -- free memory allocated to Blitz libraries

SYNOPSIS
    FreeBlitzLibs();

FUNCTION
    This function will free memory allocated by a previous call to

             LoadBlitzLibs()
                INPUTS
    NONE
```

```
RESULT
    NONE

NOTES
    This call may fail if the Blitz libraries are currently locked.
    Programs which installed a notification hook will NOT be notified of
    this change, and are removed from the notification list.

    This function is typically only called by the librarie's Expunge()
    function, and not by applications.

SEE ALSO

              LoadBlitzLibs()
```

1.9 blitz.library/FreeTokeMem()

```
                NAME
    FreeTokeMem -- free memory allocated by a previous call to
              TokeMem()
                SYNOPSIS
    FreeTokeMem(tokeMem);
                A0

FUNCTION
    This function will free any memory allocated by a call to
              TokeMem()
              .

INPUTS
    tokeMem - a pointer to a block of tokenised source, as returned by the
               call to
              TokeMem()
                RESULT
    NONE

NOTES
    It is necessary for an application to keep a track of values returned
    by
              TokeMem()
               and similar functions, since the library does not perform
    any garbage collection upon expunge.

SEE ALSO

              TokeMem()
```

1.10 blitz.library/FreeXtra()

```
                NAME
    FreeXtra -- free memory allocated by the functions
             AllocXtra()
              and

             LoadXtra()
                SYNOPSIS
    FreeXtra(xtraFile);
             A0

    void FreeXtra(struct BlitzXtra * xtraFile);

FUNCTION
    This function will free any memory allocated by a call to
             AllocXtra()
                    or
             LoadXtra()
                INPUTS
    xtraFile - a pointer to a valid BlitzXtra structure, as returned by
               either
             AllocXtra()
              or
             LoadXtra()
                RESULT
    NONE

NOTES
    It is necessary for an application to keep a track of values returned
    by
             AllocXtra()
             ,
             LoadXtra()
              and similar functions, since the library does
    not perform any garbage collection upon expunge.

SEE ALSO

             AllocXtra()
             ,
             LoadXtra()
```

1.11 blitz.library/GetBlitzLibInfo()

```
                NAME
    GetBlitzLibInfo -- fill out an applications BlitzLibComData structure with
                       relevant information

SYNOPSIS
    success = GetBlitzLibInfo(comData);
    D0                        A0

    BOOL success = GetBlitzLibInfo(struct BlitzLibComData * comData);
```

```
FUNCTION
    Following a call to
             LoadBlitzLibs()
             , the Blitz compiler requires certain
    information around the structure of the libraries. This function fills out
    a structure with this information.

INPUTS
    comData - a pointer to a previously created instance of a BlitzLibComData
              structure

RESULT
    success - an indication of success or failure

NOTES
    This routine is typically only required by the Blitz compiler

SEE ALSO

             LoadBlitzLibs()
             , <libraries/blitz.h>, <libraries/blitz.i>
```

1.12 blitz.library/GetFirstToken()

```
                NAME
    GetFirstToken -- obtain the address of the first token in the library list

SYNOPSIS
    token = GetFirstToken();
    D0

    struct BlitzToken * token = GetFirstToken(VOID);

FUNCTION
    This function returns the address of the first token in the library list,
    as created by a previous call to
             LoadBlitzLibs()
                INPUTS
    NONE

RESULT
    D0 - a pointer to the first token in memory, or NULL for failure

SEE ALSO

             LoadBlitzLibs()
             ,
             FindToken()
```

1.13 blitz.library/GetLineHeaderSize()

```
NAME
    GetLineHeaderSize -- get the size of the header prepended to Blitz source
                         code

SYNOPSIS
    size = GetLineHeaderSize();
    D0

    UWORD size = GetLineHeaderSize(VOID);

FUNCTION
    The standard Blitz line structure was fixed at 9 bytes. This consisted of
    the usual Previous and Next pointers, as well as a byte indicating how
    many characters are on the line. From v2.2 onwards, an extra control field
    was inserted to allow procedure folding. To improve backwards compatibility
    or programs it is necessary to know how large this header is. It can be
    assumed that the following fields remain static :

        NEXT            0
        PREVIOUS        4
        NUMCHARS        HEADERSIZE-1
        CHARS           HEADERSIZE

    This allows additional fields to be inserted without risking  ↩
       incompatibility
    with older software.

INPUTS
    NONE

RESULT
    size - the number of bytes which make up the line header
```

1.14 blitz.library/GetObjectMaximum()

```
                NAME
    GetObjectMaximum -- get the maximum number of instances for a given Blitz
                        object

SYNOPSIS
    maximum = GetObjectMaximum(xtraFile, libNum);
    D0                         A0        D0:16

    WORD maximum = GetObjectMaximum(struct BlitzXtra * xtraFile, UWORD libNum);

FUNCTION
    Each blitz library can have an associated 'object'. When compiling programs ↩
       ,
    it is necessary for the compiler to allocate a fixed size object buffer. To
    achieve this, a user-definable limit is set on the number of instances that
    can be created for any given object. This information is stored in the
    Blitz .xtra file. This function, when passed a valid library number and
    BlitzXtra structure can return the maximum instances for the library's
```

```
    associated object.

INPUTS
    xtraFile - a pointer to a valid BlitzXtra structure, as returned by a call
               to either
             AllocXtra()
              or
             LoadXtra()
                    libNum   - the library's identification number

RESULT
    maximum - the maximum number of instances that can be created for the
              library's associated object (-1 indicated that no object
              is associated with the given library)

SEE ALSO

             GetObjectName()
             ,
             AllocXtra()
             ,
             LoadXtra()
             , <libraries/blitz.h>,
    <libraries/blitz.i>
```

1.15 blitz.library/GetObjectName()

```
                NAME
    GetObjectName -- get the name for a library's associated object

SYNOPSIS
    name = GetObjectName(xtraFile, libNum);
    D0                   A0        D0:16

    char * name = GetObjectName(struct BlitzXtra * xtraFile, UWORD libNum);

FUNCTION
    Each blitz library can have an associated 'object'. Each object has a  ↩
       unique
    name, which is used when a programmer wishes to access a particular  ↩
       instance
    of the object, or adjust the maximum instances of the object. This function
    returns the name of the object associated with the given library.

INPUTS
    xtraFile - a pointer to a valid BlitzXtra structure, as returned by a call
               to either
             AllocXtra()
              or
             LoadXtra()
                    libNum   - the library's identification number

RESULT
    name - a pointer to the object's name (or NULL if the library does not
```

```
            have an associated object)

BUGS
    The memory used to return the string is shared. This means that
    should two applications call this function together, the result is
    unpredictable

SEE ALSO

              GetObjectMaximum()
              ,
              AllocXtra()
              ,
              LoadXtra()
              , <libraries/blitz.h>,
    <libraries/blitz.i>
```

1.16 blitz.library/GetVersion()

```
NAME
    GetVersion -- obtain a string indicating the version of the library

SYNOPSIS
    version = GetVersion();
    D0

    char * version = GetVersion(VOID);

FUNCTION
    This function simply returns the version of the library as a string.

INPUTS
    NONE

RESULT
    version - a pointer to a null terminated version string

NOTES
    This function is primarily designed to be called by SuperTED - enabling
    the version of the compiler to be adjusted.

    Do NOT modify the contents of the string - make a copy if you need to
```

1.17 blitz.library/LoadBlitzLibs()

```
                NAME
    LoadBlitzLibs -- load available Blitz libraries from disk

SYNOPSIS
    success = LoadBlitzLibs();
```

```
    D0

    BOOL success = LoadBlitzLibs(VOID);

FUNCTION
    This function will allocate memory for and read all available command
    libraries from disk. The token and library tables are created. If the
    libraries have already been loaded by a previous call, then this
    function effectively does nothing.

INPUTS
    NONE

RESULT
    success - an indication of success or failure

NOTES
    This function will fail if the libraries have been locked.

SEE ALSO

              FreeBlitzLibs()
```

1.18 blitz.library/LoadFile()

```
NAME
    LoadFile -- load a source code file into memory

SYNOPSIS
    filePtr = LoadFile(fileName);
    D0                 A0

    struct BlitzFile * filePtr = LoadFile(char * fileName);

FUNCTION
    This function will load a file into memory. Memory required to store the
    file is allocated by this function. This function can read ASCII files
    as well as the standard Blitz tokenised source files. Tokenisation is
    performed if required.

INPUTS
    fileName - a pointer to a null-terminated string holding the name of the
               file to load

RESULT
    filePtr - a pointer to a BlitzFile structure (or NULL for failure)
```

1.19 blitz.library/LoadXtra()

```
NAME
    LoadXtra -- load a source file's .xtra file into memory

SYNOPSIS
    xtraFile = LoadXtra(fileName);
    D0                  A0

    struct BlitzXtra * xtraFile = LoadXtra(char * fileName);

FUNCTION
    This function will allocate memory for and load into this memory a Blitz
    .xtra file.

INPUTS
    fileName - a pointer to a null-terminated string holding the name of the
               .xtra file to load

RESULT
    xtraFile - a pointer to the associated BlitzXtra structure (or NULL for
               failure)

NOTES
    No checking is made to ensure that the file passed is a valid .xtra file
```

1.20 blitz.library/LockBlitzLibs()

```
NAME
    LockBlitzLibs -- prevent access or modification to the Blitz command
                     libraries in memory

SYNOPSIS
    success = LockBlitzLibs(accessMode);
    D0                      D0

    BOOL success = LockBlitzLibs(WORD accessMode);

FUNCTION
    This function, given the appropriate access mode, will lock the Blitz
    command libraries to prevent another task from either :

    a) Accessing them
    b) Modifying them

INPUTS
    accessMode - a valid access mode. Can be one of either :

              LIBS_ACCESS_EXCLUSIVE
              LIBS_ACCESS_READ

RESULT
    success - TRUE or FALSE if the lock was successful. FALSE indicates either ↩
        :
```

```
            An EXCLUSIVE lock was present
            A READ lock was present and you requested an EXCLUSIVE lock

BUGS
    No validation is performed on the accessMode. Therefore, invalid  ↩
       accessModes
    will result in unpredictable behaviour

SEE ALSO
    UnlockBlitzLibs()
```

1.21 blitz.library/NewDetokeLine()

```
NAME
    NewDetokeLine -- an experimental command design to make line detokenisation
                     more flexible

SYNOPSIS
    success = NewDetokeLine(srcToken, destNonToken, destToken);
    D0                      A0        A1            A2

    BOOL success = NewDetokeLine(char * srcToken, char * destNonToken,
                                 char * destToken);

FUNCTION
    This function is a prototype command designed to be used by the new version
    of SuperTED, currently in development. In order to speed up the text  ↩
       displays,
    a new approach to detoking lines was developed. The tokenised source line
    is detoked into two buffers - one containing non-token source, the other
    containing token-source (the latter which should be displayed in a 'special ↩
       '
    colour).

    For example :

         NPrint "This is a Test" : MouseWait

    would be detokenised to :

                "This is a test" :                      ; non-token source
         NPrint                    MouseWait            ; token source

INPUTS
    srcToken - a pointer to the NULL terminated tokenised line of source code

    destNonToken - a pointer to the buffer to be used to store non-token source

    destToken - a pointer to the buffer to be used to store token source

RESULT
    success - an indication of the commands success. This function will return
              FALSE if the command libraries have not been loaded.
```

```
NOTES
    Since this function is in prototype stage, it is not recommended that
    application developers use this function.
```

1.22 blitz.library/NumMaximums()

```
NAME
    NumMaximums -- return the number of objects in the library list

SYNOPSIS
    nummaxs = NumMaximums();
    D0

    UINT nummaxs = NumMaximums(VOID):

FUNCTION
    Although slightly misleading by name, this function will return the total
    number of objects associated with the various Blitz command libraries.

INPUTS
    NONE

RESULT
    nummaxs - the number of objects in the library list

BUGS
    At preset (v2.2) this command ignores the Lock status of the libraries.
```

1.23 blitz.library/SaveXtra()

```
                NAME
    SaveXtra -- save a .xtra file to disk

SYNOPSIS
    success = SaveXtra(fileName, xtraFile);
    D0                 A0        A1

    BOOL success = SaveXtra(char * fileName, struct BlitzXtra * xtraFile);

FUNCTION
    This function will save a valid BlitzXtra structure to disk in the form
    of a standard Blitz .xtra file.

INPUTS
    fileName - a null-terminated string containing the name of the .xtra file
               to save

    xtraFile - a pointer to a valid BlitzXtra structure

RESULT
```

```
    success - an indication of success or failure

SEE ALSO

              AllocXtra()
              ,
              FreeXtra()
              ,
              LoadXtra()
              , <libraries/blitz.h>, <libraries/blitz.i>
```

1.24 blitz.library/SetObjectMaximum()

```
                NAME
    SetObjectMaximum -- set the maximum number of instances for a library's
                        associated object

SYNOPSIS
    SetObjectMaximum(libNum, xtraFile);
                     D0:16   A0

    void SetObjectMaximum(UINT libNum, struct BlitzXtra * xtraFile);

FUNCTION
    Similar in operation to
              GetObjectMaximum()
              , this function will set the
    limit.

INPUTS
    libNum - the library ID which the object is associated with

    xtraFile - a pointer to a valid BlitzXtra structure

RESULT
    NONE

SEE ALSO

              GetObjectMaximum()
```

1.25 blitz.library/SortTokens()

```
NAME
    SortTokens -- create a sorted list of all Blitz command tokens

SYNOPSIS
    SortTokens();
```

```
    void SortTokens(VOID);

FUNCTION
    This will create a sorted list of all Blitz comman tokens - used to improve
    the speed of the tokenisation / detokenisation commands.

INPUTS
    NONE

RESULT
    NONE

NOTES
    This function is designed to be called by SuperTED, and is not intended for
    general use by applications.
```

1.26 blitz.library/TokeLine()

```
NAME
    TokeLine() - tokenise a line of ASCII source into Blitz tokens

SYNOPSIS
    success/len = TokeLine(srcAscii, destBuffer);
    D0      D1             A0        A1

    BOOL success = TokeLine(char * srcAscii, char * destBuffer);

FUNCTION
    This function takes a standard null terminated line of ASCII text and
    attempts to produce a tokenised version of it.

INPUTS
    srcAscii - a null terminated string containing ASCII characters

    destBuffer - the destination buffer to store the tokenised source in

RESULTS
    success - an indication of whether the command was successful or not

    len - the length of the tokenised line

NOTES
    This command returns information in TWO registers, D0 and D1, and as
    such the second argument can only be accessed by assembly code. C
    programmers must calculate the length of the tokenised string themselves
    (hunt for a 0 byte). This restriction may change in the future.

SEE ALSO
    TokeLine()
```

1.27 blitz.library/TokeMem()

```
                NAME
    TokeMem - tokenise a block of text into Blitz tokens

SYNOPSIS
    tokeBlock = TokeMem(srcAscii,maxBlockSize);

    UINT = TokeMem(char * srcAscii, UINT maxBlockSize);

FUNCTION
    This command takes a block of ASCII text (with each line terminated by
    a LF, ASCII code 10) and creates a block of tokenised lines. The length
    of the block is specified in the function call.

INPUTS
    srcAscii - a pointer to a LF terminated block of ASCII text

    maxBlockSize - the size of memory to allocate to hold the tokenised text

RESULT
    tokeBlock - a pointer to the tokenised text block (or NULL for failure)

SEE ALSO

            DetokeMem()
```

1.28 blitz.library/UnLockBlitzLibs()

```
                NAME
    UnLockBlitzLibs - release a lock on the Blitz command libraries

SYNOPSIS
    UnLockBlitzLibs();

    void UnLockBlitzLibs(VOID);

FUNCTION
    Following a successful call to
            LockBlitzLibs()
            , the application must
    release the lock when access is not required. This should be done at the
    earliest opportunity to enable access for other applications.

INPUTS
    NONE

RESULT
    NONE

NOTES
    This function does not attempt to check if the application already has
    an open lock - care must be taken to ensure that calls to
            LockBlitzLibs()
                    and UnLockBlitzLibs() are paired.
```

SEE ALSO

LockBlitzLibs()