Commodities

COLLABORATORS

	TITLE : Commodities		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Commodities

1.1 Commodities V1.00

```
                    Commodities V1.00 General Information:

  * Blitz Basic II library number         : #153
  * Library size when linked to executable: 636 bytes
  * Number of commands                    : 11
  * Ressources automatically freed at end : Yes

    Commands summary:


                 NCommodityDeleteObject
                 Statement Long

                 NCommodityDisableObject
                 Statement Long

                 NCommodityEnableObject
                 Statement Long

                 NCommodityEvent
                 Function ()

                 NCommodityID
                 Function ()

                 NCommodityStandardObject
                 Function (Long,Long,Long)

                 NCreateCommodity
                 Function (Long,Long,Long,Long,Word)

                 NDisableCommodity
                 Statement

                 NEnableCommodity
                 Statement

                 NRemoveCommodity
                 Statement

                 NWaitCommodityEvent
                 Function ()
```

1.2 ncommoditydeleteobject

```
    SYNTAX
  NCommodityDeleteObject #Object

    STATEMENT
  Delete an enabled or disabled object created by NCommodityStandardObject().

  This statement don’t care if the object already is deleted.
```

1.3 ncommoditydisableobject

```
    SYNTAX
  NCommodityDisableObject #Object

    STATEMENT
  Disable an object created by NCommodityStandardObject() or an object
  enabled whith NCommodityEnableObject().

  A disabled object is kind of sleeping, it’s doing nothing until
  NCommoditEnableObject() wake it up.

  This statement don’t care if the object already is disabled.
```

1.4 ncommodityenableobject

```
    SYNTAX
  NCommodityEnableObject #Object

    STATEMENT
  Enable an object disabled by NCommodityDisableObject().

  A enabled object is processing Cxmessage and signals the commodity
  when some Cxmessage is passing the filter specified by #param2
  in NCommodityStandardObject()

  This statement don’t care if the object already is enabled.
```

1.5 ncommodityevent

```
  SYNTAX
msgtype.w = NCommodityEvent()

  FUNCTION
This function returns the messagetype of the Cxmessage, if there
is any, else the return is zero.

NCommodityEvent() wouldn't wait for somthing to happens, like
NWaitCommodityEvent() would, so this is useful when the eventloop
should go on.

msgtype
The message is either a command type which comes from the Commodities
Exchange when the user press some button. The message could also be
of event type and that's when a object receved a Cxmessage.
```

1.6 ncommodityid

```
  SYNTAX
id.w = NCommodityID()

  FUNCTION
This function return the ID of the object that received a Cxmessage.

It's the same as #param1 in NCommodityStandardObject() when the object
is created.
```

1.7 ncommoditystandardobject

```
                SYNTAX
error.w = NCommodityStandardObject(#Object,&Filter$,*InputEvent)

  FUNCTION
This function creates a object. The object is created in enabled state
and starts immediate to processing Cxmessage but only if the commodity
is enabled.

#Object
This is the objectnumber wanted and shoulden't be higher then #param1
in NCreateCommodity(), if it's then there be a crash.

IF the object is already in use the function don't care and just create
a new object whitout deleteing the old one, after that there is no
possibility to disable/enable/delete the old object.


              &Filter$
                This is a pointer to a string that describe what this object  ↩
                    want
to know about.
```

```
*InputEvent
This is a pointer to an InputEvent NewType. The real inputevent is
deleted and replaced by this new one.

If the pointer is zero the real inputevent is just deleted, no other
commodity or the system would know about it.

If the pointer is minus the inputevent would pass untouched.

error
If this is true the object coulden’t be created.
```

1.8 ncreatecommodity

```
  SYNTAX
error.w = NCreateCommodity(Objects,&Name$,&Title$,&Description$,Flag.w)

  FUNCTION
This function create the basic stuff of a commodity. Like open
commodities.library, create a messageport and create a broker.

The commodity is created in disabled state, so after some object
creation then enabel it whith NEnableCommodity.

Objects
This is the number of objects wanted plus one.

&Name$
This is a pointer to a string that describe the name of the commodity.
The name should be unique for each commodity.

&Title$
This is a pointer to a string that describe the title that shows up
in the window of Commodities Exchange when the commodity is runing.

&Description$
This is a pointer to a string that describe the description of the
commodity that shows up in the window of Commodities Exchange when the
commodity is runing.

Flag
If it’s true the commodity would use feature of show/hide a window when
the user press  show interface/hide interface  in Commodities Exchange.

error
If this is true the commodity coulden’t be created.
```

1.9 ndisablecommodity

```
  SYNTAX
NDisableCommodity
```

```
  STATEMENT
Disables the whole commodity, which is all objects included in the
commodity.

This statement don’t care if the commodity is already disabled.
```

1.10 nenablecommodity

```
  SYNTAX
NEnableCommodity

  STATEMENT
Enables the whole commodity, which is all enabled objects included in
the commodity.

This statement don’t care if the commodity is already enabled.
```

1.11 nremovecommodity

```
  SYNTAX
NRemoveCommodity

  STATEMENT
Delete all the objects and the basic stuff that NCreateCommodity()
have created.

This rutin is called when the program END’s, the programmer don’t need.
```

1.12 nwaitcommodityevent

```
  SYNTAX
msgtype.w = NWaitCommodityEvent()

  FUNCTION
This function returns the messagetype of the Cxmessage.

NWaitCommodityEvent() would wait for events to happen, unlike
NCommodityEvent(), so this is useful when to save processor time.

msgtype
The message is either of command type which comes from the Commodities
Exchange when the user press some button else the message could be
of event type and that’s when a object receved a Cxmessage.
```

1.13 filterstrings

```
                    [Class]  {[-] (Qualifier|Synonym)}  [[-] upstroke]  [highmap| ←
                       ANSICode]



                       Class

                 Qualifier|Synonym

                     upstroke

                 highmap|ANSICode
                  Some simple input description strings.
  -------------------------------------
  "rawkey upstroke a"

  "rawkey -upsroke f1"

  "timer"

  "diskremoved"

  "rawkey leftbutton f2"
```

1.14 class

```
 Class can be any one of the class strings in the table below.

     Class String
     ------------
    rawkey
    timer
    diskremoved
    diskinserted
```

1.15 qualifier|synonym

```
 Qualifier is one of the qualifier strings from the table below.
 A dash preceding the qualifier string tells the filter object not
 to care if that qualifier is present in the input event.
 Notice that there can be more than one qualifier (or none at all) in the
 input description string.

   Qualifier String
   ----------------
    lshift
    rshift
    capslock
    control
    lalt
    ralt
    lcommand
    rcommand
    numericpad
    repeat
    midbutton
    rbutton
    leftbutton
    relativemouse
```

```
Synonym is one of the synonym strings from the table below.  These
strings act as synonyms for groups of qualifiers. A dash preceding
the synonym string tells the filter object not to care if that
synonym is present in the input event.  Notice that there can be more
than one synonym (or none at all) in the input description string.

   Synonym String
   --------------
   shift        look for either shift key
   caps         look for either shift key or capslock
   alt          look for either alt key
```

1.16 upstroke

```
Upstroke is the literal string "upstroke".  If it is present alone the
filter considers only upstrokes, if it’s absent the filter considers only
downstrokes and if preceded by a dash the filter considers both upstrokes
and downstrokes.
```

1.17 highmap|ansicode

```
Highmap is one of the following strings:

   space , backspace , tab , enter , return , esc , del , help,
   up , down , right , left,
   f1 , f2 , f3 , f4 , f5 , f6 , f7 , f8 , f9 , f10.


ANSICode is a single character for example ‘a’ .
```