MCC_HexEdit

COLLABORATORS

	TITLE : MCC_HexEdit		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MCC_HexEdit

1.1 MCC_HexEdit.doc

1.2 HexEdit/--background--

```
 NAME
HexEdit -- ... (V14)

 FUNCTION
This class provides you with an easy way to present hexadecimal
```

```
dump of a memory region to the user.  Additionally, memory
contents are visible in ASCII representation.  Editing in both
modes is possible.  HexEdit class is very flexible, with many
settable parameters and some critical methods (like byte reading
and writing) waiting to be overloaded.
```

1.3 HexEdit/MUIA_HexEdit_ActiveField

```
 NAME
MUIA_HexEdit_ActiveField, LONG [ISG] -- ... (V14)

 SPECIAL INPUTS
MUIV_HexEdit_ActiveField_HexDump
MUIV_HexEdit_ActiveField_Chars

 FUNCTION
With MUIA_HexEdit_ActiveField you can decide over which field of
the object (hexdump or character) active cursor should be
positioned.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
```

1.4 HexEdit/MUIA_HexEdit_AddressChars

```
 NAME
MUIA_HexEdit_AddressChars, LONG [I.G] -- ... (V14)

 SPECIAL INPUTS
MUIV_AddressChars_Auto

 FUNCTION
This attribute lets you define how many characters will be devoted
to representing addresses displayed by HexEdit.  For example, if
you specify a value of 4, all addresses will be displayed as 16
bit.  Default is 8, i.e.  32 bit.

Address calculation's precision is not dependant on this
attribute, it's for display purposes only.  Internally all
addresses are treated as LONGs.

 NOTES
This value ranges from 1 to 8, but please see
MUIM_HexEdit_CreateDisplayAddress().

 BUGS
No known bugs.
```

```
 SEE ALSO
MUIM_HexEdit_CreateDisplayAddress()
```

1.5 HexEdit/MUIA_HexEdit_BaseAddressOffset

```
 NAME
MUIA_HexEdit_BaseAddressOffset, LONG [ISG] -- ... (V14)

 FUNCTION
The value of this attribute is added to the current address every
time when it is about to be displayed.  For example, if you have
your data at 0x04000000 and you want the addresses to be shown as
starting from 0, set this attribute to -0x04000000.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
```

1.6 HexEdit/MUIA_HexEdit_BytesPerColumn

```
 NAME
MUIA_HexEdit_BytesPerColumn, LONG [I.G] -- ... (V14)

 FUNCTION
With MUIA_HexEdit_BytesPerColumn you can define how many bytes
should constitute one column.  For example, default value of 4
groups memory dump in a following way:

01234567 01234567 01234567 01234567

By changing it to 3, you would get:

012345 670123 456701 234567

 NOTES

 BUGS
No known bugs.

 SEE ALSO
```

1.7 HexEdit/MUIA_HexEdit_BytesPerLine

```
 NAME
MUIA_HexEdit_BytesPerLine, LONG [..G] -- ... (V14)

 FUNCTION
```

```
With this attribute you can learn how many bytes are displayed in a
single line of HexEdit object.

 BUGS
No known bugs.

 SEE ALSO
```

1.8 HexEdit/MUIA_HexEdit_ByteValue

```
 NAME
MUIA_HexEdit_ByteValue, LONG [.SG] -- ... (V14)

 FUNCTION
This attribute lets you read and modify the value of byte under
the cursor.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_NibbleValue()
```

1.9 HexEdit/MUIA_HexEdit_ColumnsPerLine

```
 NAME
MUIA_HexEdit_ColumnsPerLine, LONG [I.G] -- ... (V14)

 FUNCTION
By querying this attribute you can learn how many columns are
currently displayed in one line of HexEdit object.  By default,
the number of columns varies with the size of the object.  If,
however MUIA_HexEdit_ColumnsPerLine is set at the creation time,
the object will always use a value then specified.

 BUGS
No known bugs.

 SEE ALSO
```

1.10 HexEdit/MUIA_HexEdit_CursorAddress

```
 NAME
MUIA_HexEdit_CursorAddress, LONG [.SG] -- ... (V14)

 FUNCTION
Reading this attribute will return the cursor’s current address.
```

```
Setting it will move the cursor to a new location, possibly
scrolling and/or redrawing the display.

 NOTES
Address, as usual, is specified as byte offset from
MUIA_HexEdit_LowBound()

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_LowBound(), MUIA_HexEdit_HighBound(),
MUIA_HexEdit_CursorNibble()
```

1.11 HexEdit/MUIA_HexEdit_CursorNibble

```
 NAME
MUIA_HexEdit_CursorNibble, LONG [ISG] -- ... (V14)

 FUNCTION
With this attribute you can set or get the nibble over which the
cursor is currently positioned.

 RESULTS
0 for upper nibble, 1 for lower nibble.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_CursorAddress()
```

1.12 HexEdit/MUIA_HexEdit_CursorVisible

```
 NAME
MUIA_HexEdit_CursorVisible, BOOL [ISG] -- ... (V14)

 FUNCTION
With this attribute you can set and get the state of cursor’s
visibility.

 NOTES
Please do not attempt to switch off the cursor’s while in the edit
mode, as it may confuse the user.

 BUGS
No known bugs.

 SEE ALSO
```

1.13 HexEdit/MUIA_HexEdit_EditMode

```
 NAME
MUIA_HexEdit_EditMode, BOOL [ISG] -- ... (V14)

 FUNCTION
This attribute determines if HexEdit object will let the user edit
a memory space that is under its control.  If it is set to TRUE,
user can input 0-9/a-f digits while in the hexdump part of the
display and alphanumerics while in the character part.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_CursorVisible(), MUIM_HexEdit_WriteMemoryByte(),
MUIM_HexEdit_FilterChar()
```

1.14 HexEdit/MUIA_HexEdit_First

```
 NAME
MUIA_HexEdit_First, LONG [ISG] -- ... (V14)

 FUNCTION
MUIA_HexEdit_First specifies offset (counting from
MUIA_HexEdit_LowBound()) of the first byte to be displayed at the
top of HexEdit object.  This attribute is always rounded down to a
multiply of MUIA_HexEdit_BytesPerLine().

 NOTES
In certain cases this attribute does not change, despite set()ting
a value different from the current - for example when there are
not enough data lines below to fill the entire HexEdit object.  If
you want to highlight certain position, please rather use
MUIA_HexEdit_CursorAddress().

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_CursorAddress()
```

1.15 HexEdit/MUIA_HexEdit_FirstLine

```
 NAME
MUIA_HexEdit_FirstLine, LONG [.S.] -- ... (V14)

 SPECIAL INPUTS
MUIV_HexEdit_FirstLine_Up
MUIV_HexEdit_FirstLine_Down
```

```
MUIV_HexEdit_FirstLine_PageUp
MUIV_HexEdit_FirstLine_PageDown
MUIV_HexEdit_FirstLine_Top
MUIV_HexEdit_FirstLine_Bottom

 FUNCTION
Changing this attribute lets you move around your memory space.

 NOTES
Only values listed above are allowed as input.

 BUGS
No known bugs.

 SEE ALSO
```

1.16 HexEdit/MUIA_HexEdit_FullRefresh

```
 NAME
MUIA_HexEdit_FullRefresh, BOOL [.S.] -- ... (V14)

 FUNCTION
Setting this attribute to TRUE will force redraw of the entire
object area next time MUIA_HexEdit_First() is set.  Normally, only
parts that are new on display would be drawn.

 NOTES
This attribute may be helpful with some more exotic subclasses of
HexEdit.  Usually, you won’t need to touch it, though.

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_First()
```

1.17 HexEdit/MUIA_HexEdit_HighBound

```
 NAME
MUIA_HexEdit_HighBound, LONG [I.G] -- ... (V14)

 FUNCTION
With MUIA_HexEdit_HighBound you can define where in Amiga memory
space ends the area you want HexEdit to operate on.

 NOTES
This attribute MUST be specified, even if you overload
MUIM_HexEdit_ReadMemoryByte() or MUIM_HexEdit_WriteMemoryByte()!

 BUGS
No known bugs.
```

```
 SEE ALSO
MUIA_HexEdit_LowBound(), MUIM_HexEdit_ReadMemoryByte(),
MUIM_HexEdit_WriteMemoryByte()
```

1.18 HexEdit/MUIA_HexEdit_LowBound

```
 NAME
MUIA_HexEdit_LowBound, LONG [I.G] -- ... (V14)

 FUNCTION
With MUIA_HexEdit_LowBound you can define where in Amiga memory
space begins the area you want HexEdit to operate on.

 NOTES
This attribute MUST be specified, even if you overload
MUIM_HexEdit_ReadMemoryByte() or MUIM_HexEdit_WriteMemoryByte()!

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_HighBound(), MUIM_HexEdit_ReadMemoryByte(),
MUIM_HexEdit_WriteMemoryByte()
```

1.19 HexEdit/MUIA_HexEdit_MoveCursor

```
 NAME
MUIA_HexEdit_MoveCursor, LONG [.S.] -- ... (V14)

 SPECIAL INPUTS
MUIV_HexEdit_MoveCursor_Up
MUIV_HexEdit_MoveCursor_Down
MUIV_HexEdit_MoveCursor_Left
MUIV_HexEdit_MoveCursor_Right
MUIV_HexEdit_MoveCursor_PageUp
MUIV_HexEdit_MoveCursor_PageDown
MUIV_HexEdit_MoveCursor_Top
MUIV_HexEdit_MoveCursor_Bottom
MUIV_HexEdit_MoveCursor_WordLeft
MUIV_HexEdit_MoveCursor_WordRight
MUIV_HexEdit_MoveCursor_LineStart
MUIV_HexEdit_MoveCursor_LineEnd

 FUNCTION
With MUIA_HexEdit_MoveCursor you can control cursor movements
exactly as if you were using the keyboard.  Of course, they do not
make much sense if the cursor is actually turned off.

 NOTES
Only values listed above are allowed as input.

 BUGS
```

```
No known bugs.

 SEE ALSO
MUIA_HexEdit_CursorVisible()
```

1.20 HexEdit/MUIA_HexEdit_NibbleValue

```
 NAME
MUIA_HexEdit_NibbleValue, LONG [.SG] -- ... (V14)

 FUNCTION
This attribute lets you read and modify the value of nibble under
the cursor. The value is always passed in a lower four bits.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_ByteValue()
```

1.21 HexEdit/MUIA_HexEdit_PropObject

```
 NAME
MUIA_HexEdit_PropObject, LONG [.SG] -- ... (V14)

 FUNCTION
With MUIA_HexEdit_PropObject you can easily connect proportional
gadget with the HexEdit object.  Simply pass a pointer to prop
object as the value of this attribute and you’re done.  HexEdit
will handle all house-keeping for you. How convinient! ;)

 NOTES

 BUGS
No known bugs.

 SEE ALSO
```

1.22 HexEdit/MUIA_HexEdit_SelectMode

```
 NAME
MUIA_HexEdit_SelectMode, LONG [I.G] -- ... (V14)

 SPECIAL INPUTS
MUIV_HexEdit_SelectMode_Nibble
MUIV_HexEdit_SelectMode_Byte

 FUNCTION
```

```
This attribute defines if the cursor moving over hexadecimal data
should span entire bytes or just single nibbles.

 NOTES

 BUGS
No known bugs.

 SEE ALSO
```

1.23 HexEdit/MUIA_HexEdit_VisibleLines

```
 NAME
MUIA_HexEdit_VisibleLines, LONG [..G] -- ... (V14)

 FUNCTION
With this attribute you can learn how many text lines are
currently visible in the HexEdit object.  It may be useful for
notifications.

 BUGS
No known bugs.

 SEE ALSO
```

1.24 HexEdit/MUIM_HexEdit_CreateDisplayAddress

```
 NAME
MUIM_HexEdit_CreateDisplayAddress (V14)

 SYNOPSIS
DoMethod(obj, MUIM_HexEdit_CreateDisplayAddress, UBYTE **cp, ULONG address);

 FUNCTION
HexEdit class calls this method to generate ASCII representation
of current address.  If you want to create some fancy-looking
addresses, you can overload this method in your subclass.

 NOTES
If you're overloading this method, you HAVE TO set
MUIA_HexEdit_AddressChars() to length of the largest output of
your address creator, so that HexEdit knows how much space
allocate for it.

 EXAMPLE
This is HexEdit's implementation of
MUIM_HexEdit_CreateDisplayAddress method:

ULONG __asm _CreateDisplayAddress(REG(A0) struct IClass *cl, REG(A2) Object *obj ↩
    , REG(A1) struct MUIP_HexEdit_CreateDisplayAddress *msg)
{
  struct Data *d = INST_DATA(cl,obj);
  ULONG address;
  UBYTE i;
  UBYTE *hextable = "0123456789ABCDEF";

  address = (d->base_address + msg->address) << (HE_MAX_ADDRESS_LEN - d-> ↩
     address_chars * 4);

  for(i = 0; i < d->address_chars; i++)
  {
    *(*msg->cp)++ = hextable[address >> (HE_MAX_ADDRESS_LEN - 4)];
    address <<= 4;
  }

  return(TRUE);
}
```

```
 BUGS
No known bugs.

 SEE ALSO
MUIA_HexEdit_AddressChars()
```

1.25 HexEdit/MUIM_HexEdit_FilterChar

```
 NAME
MUIM_HexEdit_FilterChar (V14)

 SYNOPSIS
DoMethod(obj, MUIM_HexEdit_FilterChar, ULONG value, UBYTE *buffer);

 FUNCTION
HexEdit class calls this method to generate ASCII representation
of character it is currently processing.  You can overload it to
e.g. show only characters you want and replace others with dots.

 EXAMPLE
This is example implementation of MUIM_HexEdit_FilterChar:

ULONG __asm _FilterChar(REG(A0) struct IClass *cl, REG(A2) Object *obj, REG(A1)  ↩
   struct MUIP_HexEdit_FilterChar *msg)
{
  if(isascii(msg->value))
    *msg->buffer = msg->value;
  else
    *msg->buffer = '.';

  return(TRUE);
}


 NOTES

 BUGS
No known bugs.
```

SEE ALSO

1.26 HexEdit/MUIM_HexEdit_ReadMemoryByte

```
 NAME
MUIM_HexEdit_ReadMemoryByte (V14)

 SYNOPSIS
DoMethod(obj, MUIM_HexEdit_ReadMemoryByte, UBYTE *value, ULONG address);

 FUNCTION
HexEdit class calls this method to read a byte from memory.  You
can overload it in your subclass, so that operation on areas which
are not present in Amiga's memory space is possible.

 NOTES
Address, as usual, is specified as byte offset from
MUIA_HexEdit_LowBound()

 EXAMPLE
This is HexEdit's implementation of MUIM_HexEdit_ReadMemoryByte
method:

ULONG __asm _ReadMemoryByte(REG(A0) struct IClass *cl, REG(A2) Object *obj, REG( ↩
   A1) struct MUIP_HexEdit_ReadMemoryByte *msg)
{
  struct Data *d = INST_DATA(cl,obj);

  *msg->value = *(UBYTE *)(d->bound_low + msg->address);

  return(TRUE);
}


 BUGS
No known bugs.

 SEE ALSO
MUIM_HexEdit_WriteMemoryByte()
```

1.27 HexEdit/MUIM_HexEdit_Redraw

```
 NAME
MUIM_HexEdit_Redraw (V14)

 SYNOPSIS
DoMethod(obj, MUIM_HexEdit_Redraw);

 FUNCTION
Redraws the entire object's contents.

 NOTES
```

BUGS
No known bugs.

SEE ALSO

1.28 HexEdit/MUIM_HexEdit_WriteMemoryByte

NAME
MUIM_HexEdit_WriteMemoryByte (V14)

SYNOPSIS
DoMethod(obj, MUIM_HexEdit_ReadMemoryByte, ULONG value, ULONG address);

FUNCTION
HexEdit class calls this method to write a byte to memory. You
can overload it in your subclass, so that operation on areas which
are not present in Amiga's memory space is possible.

NOTES
Address, as usual, is specified as byte offset from
MUIA_HexEdit_LowBound()

EXAMPLE
This is HexEdit's implementation of MUIM_HexEdit_WriteMemoryByte
method:

```
ULONG __asm _WriteMemoryByte(REG(A0) struct IClass *cl, REG(A2) Object *obj, REG ←
    (A1) struct MUIP_HexEdit_WriteMemoryByte *msg)
{
  struct Data *d = INST_DATA(cl,obj);

  *(UBYTE *)(d->bound_low + msg->address) = msg->value;

  return(TRUE);
}
```

BUGS
No known bugs.

SEE ALSO
MUIM_HexEdit_ReadMemoryByte()