in

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>in | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | July 16, 2022 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# in

## 1.1 DAC interface and applications

```
Dump
    The graphical page dumper


Developer
  The developer material
```

## 1.2 Dump - the application that dumps the text-lines onto the printer

```
The application DAC-Dump

The DAC-Dump program is used to dump the contents of an archive or of a ↩
   marked text-block onto the
printer. It's currently a bit tricky (in form of "not too system ↩
   conform" but it works. The DAC-interface of
BareED has yet only limited functions - so Dump cannot offer more!

Before you use the menu "Project - Print" which will start Dump, you have to set ↩
   the desired output mode via
the System preferences editors, "Printer", "PrinterGfx" and if necessary ↩
   "PrinterPS". Dump will take your
entered settings into account when laying out the entire pages!

Since Dump and BareED are somewhat limited, you should take care that ↩
   BareED doesn't allow you to
modify the archive while a DAC application is running.

Currently, the graphical dump will be only made in monochrome (black / white) ↩
   . Since the System's printer
driver is used for dumping, the result is poor. Because of this, Dump is ↩
    an external program that can be
replaced by a third party program, which for example is connected to a commercial ↩
   printer program.
```

You have the choice to dump the whole archive or only a selected block of characters. How to mark a block
inside of BareED should be clear.

If you use a usually print-device of the OS, please firstly choose "Pica" as Print Pitch and set the
appropriate values for Page Length, Left Margin and Right Margin - since the original printer-drivers rely on
these values to compute the correct aspect-ratio - which cannot be changed by Dump.

To start dumping a page onto the printer choose "Project - Print". BareED will look then for a file labelled
Dump in its home-directory in a separate directory labelled DAC or in its home-directory only if the DAC
directory is non-existent!

Example:

Work:Tools/Editors/BareED home-directory for the program BareED

This directory contains following files and directories:
BareED
BareED.info
BareED.guide
BareED.guide.info
Button.cfg
catalogs
catcomp_files
defs
fonts
images
knobs
LastMinute
LastMinute.info
ReadMe
ReadMe.info
rexx
source
tool

If there is now also a directory DAC this one is scanned for the file Dump.
If this directory is not present, the home-directoy is scanned for the file Dump.

If this directory is not present, the home directory is scanned for the file Dump.

If the file Dump is neither to find in the DAC- and home-directory BareED will fail with the message:

Cannot load the program's code! - File not present?

If the file was found and you have still not set the right margin a requester appears where you

have to set it! Characters beyond the entered right margin will not be printed out!

If you are sure that you know the rightmost character's offset (right margin) you can enter it. If not, enter a huge value, for example 300.

A requester appears where the source width and height are displayed in pixel and the destination is displayed as dots (dots = printer pixels).

Now Dump begins to calculate each string line's length. If there is a string line that exceeds your rightmost offset, a requester appears telling so. You have now the chance to abort or to continue calculating string lengths.

In case you continue, some more requester may pop up.

When Dump has rummaged through all string lines a requester appears that tells you how wide the widest string is. Remember the rightmost character's offset for later if you have entered a huge value earlier.

Since we cannot stop here, we click "Okay".

If your set right margin would be lesser than the computed, a requester appears that tells you so. If you agree to continue, click "Okay" to dump the characters. In this case some string lines are truncated to fit onto the concerned page.

If you have entered a huge value only to compute the widest string length, you can now safely quit here.

In case you have set the right margin correct or you want some string lines to be truncated you are now at a point, where a requester shows up that tells you how many pages would be needed to dump these strings. You have another chance to quit here - or you can now start the dump.

Say, that you instruct Dump to dump the string lines onto the printer - somewhat more or less time is gone before the printer will start to work. Once more: You can't modify the archive while Dump is in progress!

The graphical dump is made in a manner so that the right margin you have entered will fit as last character onto the page. This means that basing on your defined area a complete printer-line is used to mirror the string line!

If you need at left and right of the printed area more space choose easily in ↩
the System Preferences editor
Printer new values for left and right margin. Dump will take them into account ↩
when it is going to calculate the
dimension for printout.

If you want to stop the graphic dump for a while (meanwhile you can speak to ↩
others - in case you use a
pin-writer - like me!), choose easily "Pause/Continue" in the progress- ↩
bar. Later on you can click this
button again to continue the dump.

If you want to abort the dump, click the "Abandon" button in the progress-bar.

In addition, you can use a System-monitor to find the task (process) "Dump". ↩
Signal him a CTRL-C signal.
Dump will stop as soon as possible. You may also use the "BreakTask" command, ↩
which can be found within
the NDUK package.

Depending on your printer there can be a small confusion:
When a page has been completely filled with the graphic dump and a new page is ↩
required, Dump
will send an "Eject Page" command to the printer. If your printer itself ↩
has recognize that a new
page is required, it will "eject the current page" and later on it gets the ↩
same instruction from Dump
so that an empty page is ejected.
This is harmless but annoying and only occurs when the last graphic-dump-line ↩
fits exactly to the
last position of page!

## 1.3 Developer information

This page is under construction...

First of all, DAC means direct communication with an application - without using ↩
Exec-messages!

Any DAC-application runs asynchronous from BareED. BareED is not inhibited ↩
while a DAC application is
running!

Currently the DAC-interface of BareED is poor - but if there is interest in an ↩
expanded interface I will write it.

A DAC application is a standard Amiga-DOS load file (program) with no ↩
restrictions in size or hunk-layout
other than limited by the Amiga-DOS. This means also - if you use a C-compiler ↩
- that you can use small
code and data model.

A DAC application will be started from BareED! No other technique is supported ↩
or will be in the future! This

```
means that you have to know which process fired up your application. To ↩
    determine "who was that?" the
standard Workbench-start-up-message is used with additional information enclosed.

The field "Name" of the message's node points to the string "BAREED". When you ↩
    encounter this, you know
that BareED has launched your application.

In ANSI-C this would cost us these lines:

#include <workbench/startup.h>
#include <string.h>

#include <bareed_dac.h>

extern struct WBStartup *WBenchMsg;

struct PseudoMsg *PsMsg;

int main( unsigned int argc, unsigned char **argv)
{
  /* Figure out if we have been started from a CLI surround or from the desktop */
  if ( !WBenchMsg)
    return 30; // Return error, was CLI

  /* Figure out if we have been launched by Workbench or BareED */
  if ( strcmp( WBenchMsg->sm_Message.mn_Node.ln_Name, "BAREED") != NULL)
    return 20; // Return error, was Workbench

  /* Found out that we're running as DAC-application under BareED */
  .....

Since we found out that BareED launched us, we can now convert the pointer to the ↩
    WBStartup message into
a PseudoMsg pointer:

  PsMsg = (struct PseudoMsg *) WBenchMsg;

Note: The PseudoMsg is once set up by BareED (when it launched the application) ↩
    and the attributes of
  the archive (project) cannot be changed by the user while your DAC ↩
      application is still alive,
  except BlockStart and BlockEnd! But- later on by the user newly set ↩
      cursor-positions or page
  offsets, the editor window size and a bit more will not be mirrored ↩
      by the already gotten
  PseudoMsg and currently BareED does not stop the user from changing ↩
      them! Therefore
  pm_GetAttr () and pm_ChangeAttr () have been wisely (hey, I'm so clever) ↩
      implemented which are
  currently out of order (I'm so lazy).

The PseudoMsg looks like this:

struct PseudoMsg
{
  struct WBStartup   pm_Startup;      Only readable!
```

```
BPTR     pm_Lock;     Hands off!
unsigned char *pm_Name;     Hands off!
unsigned char  pm_FileName[108];      Hands off!
unsigned char  pm_Dir[256];     Hands off!
struct GfxBase  *pm_GfxBase;
struct IntuitionBase  *pm_IntuitionBase;
struct Library  *pm_GadToolsBase;
struct Library  *pm_DiskfontBase;
struct Library  *pm_AslBase;
struct Library  *pm_IconBase;
struct Library  *pm_LocaleBase;
struct Library  *pm_WorkbenchBase;
void    *pm_VisualInfo;     Only read- and useable - don't release it!
struct DrawInfo *pm_DrawInfo;     Only read- and useable - don't release it!
unsigned char *pm_RegionStart;      Start of memory block for letters
unsigned int   pm_RegionSize;     Size in bytes (multiple of 16Kb)
unsigned char *pm_TextStart;      First letter
unsigned char *pm_TextEnd;      Last letter
unsigned char *pm_BlockStart;     First letter in block
unsigned char *pm_BlockEnd;     Last letter in block
struct TextAttr *pm_FontAttr;     Font is using this attributes
struct TextFont *pm_Font;     The font itself
struct Window *pm_EdWindow;     The editor surrounding
unsigned int   pm_TabWidth;     In pixels
unsigned int   pm_TabStops;     A tap stop occurs every 'n'
unsigned int   pm_RightMargin;
unsigned char *pm_CharSpace;      Pointer to the character-spaces of the used  ↩
   font
void    (*pm_GetAttr)( struct TagItem *taglist);  Currently NULL (out of order!)
void    (*pm_ChangeAttr)( struct TagItem *taglist); Currently NULL (out of order ↩
   !)
void    (*pm_BlockInput)( void);
void    (*pm_AllowInput)( void);
void    (*pm_Tell)( STRPTR str);
unsigned int  (*pm_CaseTell)( STRPTR str);
unsigned int  (*pm_RequestNumber)( unsigned int initial, STRPTR winname, STRPTR  ↩
   hailtext,\
         STRPTR gadtext, BOOL zero);
unsigned int  (*pm_StrPixelLen)( unsigned char *start, unsigned char *end);
void    (*pm_DumpStrLine)( unsigned char *start, unsigned char *end, struct  ↩
   RastPort *rp,\
                  unsigned int x, unsigned int y);
unsigned int  (*pm_WidestStrLen)( unsigned char *text, unsigned char *stop,\
                  unsigned int (*inform_code)( unsigned int len, unsigned int  ↩
                     line),\
                  unsigned int inform);
void    (*pm_DumpStrings)( struct RastPort *rp,\
                  unsigned int (*dump_code)( unsigned int len, unsigned int  ↩
                     line),\
                  unsigned char *text, unsigned char *stop);
void        (*pm_FreeProgressBar)( struct ProgressBar *pb);
struct ProgressBar *(*pm_CreateProgressBar)( STRPTR wintitle, STRPTR hail,  ↩
   STRPTR stop, STRPTR cont,\
       STRPTR cancel);
unsigned int  (*pm_PullPBarEvent)( struct ProgressBar *pb);
void    (*pm_ChangePBarIndicator)( struct ProgressBar *pb, unsigned int percent, ↩
    STRPTR hail);
```

```
  <<< Following does not work properly yet - so don't use! >>>
  void    (*pm_TogglePBarGad)( struct ProgressBar *pb);

};


                Detailed structure description
                If  you  have  written  a DAC application and you now want to run  ↩
                   it under BareED you simply press CTRL-D
within  BareED's  editor-window  and  a  file  requester  will  appear where you  ↩
   can choose your application,
which will afterwards be loaded in and executed.

When  a  DAC application has crashed, BareED will not get back the sent message  ↩
   and therewith BareED will
not  allow  to modify the archive. Since the project is protected against  ↩
   modifications BareED isn't able to quit,
too.

There is a possibility to normalise BareED:
To do so enter at the CLI-prompt:
  1> rx "address BAREED.n; reset daccnt"
    -    where     n represents the use count
This ARexx macro line will set the intern BareED counters to zero!

NOTE:
A  DAC  application  will  be  fired  up  with  a  stack  size  of 8192 bytes.  ↩
   This should be enough for the most
programs!  You  should  hold  in  mind  that  your  task  uses  BareED  functions  ↩
   (code) so these functions are
re-entrant.
```

## 1.4 Description

```
pm_Startup  =  a normal workbench start-up-message with the exception that the  ↩
   field "Name" of the
        node  structure  points to the string "BAREED" and where the field  ↩
           priority of the node
        structure holds this PseudoMessage version (currently 0 = beta)
pm_Lock to pm_Dir  are private, hands off - they are used by the pm_Startup  ↩
   structure

pm_GfxBase  = library base pointer
pm_IntuitionBase  = library base pointer
pm_GadToolsBase = library base pointer
pm_DiskfontBase = library base pointer
pm_AslBase  = library base pointer
pm_IconBase = library base pointer
pm_LocaleBase = library base pointer
pm_WorkbenchBase  = library base pointer

pm_VisualInfo =  pointer  to GadTools required info - read- and useable by you -  ↩
   but never release it!
```

```
        (never call FreeVisualInfo() on it!)
pm_DrawInfo = pointer to GadTools/Intuition draw-info structure - same rules as ↩
    for VisualInfo!
pm_Region = address storage start

pm_RegionSize = amount in bytes of storage
pm_TextStart  = address first character in archive
pm_TextEnd  = address last character in archive

pm_BlockStart = address first character of a marked block
pm_BlockEnd = last character of this block

pm_FontAttr = TextAttr structure that is currently used by the editor-window
pm_Font   = the already opened TextFont pointer

pm_EdWindow = pointer to an Intuition engaged window structure used as editor- ↩
    window

pm_TabWidth = how many pixel to move to the right to get the next tabulator offset
pm_TabStops = after how many space-characters a new tabulator offset is ↩
    reached (only valid if
        using mono-space-fonts - using proportional fonts it's a bit more ↩
            difficult due to
        alignment rules)

pm_RightMargin  = amount space-characters used to form the rightmost character ↩
    offset

pm_CharSpace  = pointer to an array of 256 bytes where each byte is ↩
    viewed as an index to the
        LATIN-1 char set and where these bytes will hold the concerned character's ↩
            width
       EXAMPLE:
       WidthOfSpaceChar = pseudomsg->pm_CharSpace[ 32];
       WidthOfMChar = pseudomsg->CharSpace[ 'M'];
       WidthOfDoubleSChar = pseudomsg->CharSpace[ (UBYTE) 'ß'];
       Note: casting the character is necessary if using characters greater ↩
           than
             index 127 (unsigned) to ignore the MSB!
         from assembler
       movea.l _pseudomsg,A0
       movea.l pm_CharSpace(A0),A0
       move.w #'M',D0
       move.b 0(A0,D0.w),D0
       move.w D0,_WidthOfMChar

pm_GetAttr()  = pointer to a function that will in the future allow to return ↩
    the current state of BareED
        and its project
        NOTE: this function must in no way be called for the current ↩
            versions of BareED
              because pm_GetAttr () is a NULL-pointer! For later versions, ↩
                  check first if this
              pointer is non-zero!
pm_ChangeAttr() = pointer to a function that will in the future allow to ↩
    change the current attributes of
        BareED
```

```
       NOTE: this function must in no way be called for the current ↩
          versions of BareED
                because pm_ChangeAttr () is a NULL-pointer! For later versions, ↩
                    check first if
                this pointer is non-zero!

pm_BlockInput() = pointer to a function: forbid any modifications through the user
pm_AllowInput() = pointer to function: allow modifications through the user
       PLEASE: Use pm_BlockInput() and pm_AllowInput() wisely. In the most ↩
            cases it
       should not be necessary to call these two function since when a DAC ↩
          application is
       running, BareED prevents the archive to be modified through ARexx and ↩
           the user,
       exception: the newly marking / demarking of text blocks!

pm_Tell()   = pointer to a function: to tell user what is going on
pm_CaseTell() = pointer to a function: to give the user the chance to say "Okay" ↩
   or "Cancel"
pm_RequestNumber()= pointer to a function: to get a number from the user

pm_StrPixelLen()  = pointer to a function:
      Get length in pixels a string line takes up where the current ↩
         attributes of the archive
      will be taken into account

    INPUTS:
      start - first character in line
      end - last character in line (normally Linefeed or zero byte)

    RESULTS:
      width - in pixels

pm_DumpStrLine()  = pointer to a function:
      Dump a series of characters to a specified raster port where the ↩
         current attributes of
      the archive will be taken into account

    WARNING: boundaries are not check - thus you have to ensure that no ↩
       pixels are
    drawn beyond the memory region (bit planes)

    INPUTS:
    Address first and last character in line to dump
      start - first character in line
      end - last character in line (normally LineFeed or zero byte) rp -
            Pointer to raster port where to visualize the characters x -
            leftmost position to start the render (normally 0)
      y - topmost position to start the render (normally 0)

    RESULTS:
      Printed line or none

    NOTES:
      The y-coordinate is corrected by this function with the TextFont- ↩
         tf_Baseline
      value to ensure that the text is right rendered.
```

```
      This function uses the Graphic-library functions Text() and Move()

pm_WidestStrLen() = pointer to a function:
      Get  the  widest  string (in pixels) and inform caller when out of his set  ↩
         range where the
      current attributes of the archive will be taken into account

     INPUTS:
      start - address of the character you like to start with
      stop  - at   this  character  (address)  WidestStrLen()  will  stop  -  if   ↩
         not
           encountered already archive's end
      inform_code - routine  which  is  invoked when a line length exeeds your set
           'inform' range
           Return  TRUE  if  you  want  to continue computing line length or
           FALSE to stop
      inform  - widest string width in pixel you allow without to be informed

     RESULTS:
      widest string length in pixels


     NOTES:
      inform_code() may be zero, then your CallBack routine is not called
      Your inform_code() is called with two stack parameters:
        1)  length in pixels
        2)  actual line number, which is counted on by "start"
      Your  inform_code()  does  not  need  to restore its base-register a4,  ↩
         BareED
      has  already  done this - but all other non-scratch registers must be  ↩
         restored
      on exit

pm_DumpStrings()  = pointer to a function:
      Dump  a series of string lines to a raster port where the current attributes ↩
          of the archive
      will be taken into account

    INPUTS:
      rp  - raster port - where to render into
      dump_code - your  function  that  dump  this  raster  port  e.  g.  to the  ↩
         printer
           RETRUN TRUE to continue with the next line or FALSE to stop
           PARAMETERS you'll get from DumpStrings:
          length - pixel length of this string
          line  -  line  number of the actual line, counted from one
          to endless
      start - address of the character you like to start with
      stop  - at  this  character  (address)  DumpStrings()  will  stop  -  if   ↩
         not
           encountered already archive's end

    RESULTS:
      rp  - Raster port with visualized and laid out string

    NOTES:
```

```
      Before you call DumpStrings() or DumpStrLine() you should use ↩
         the
      appropriate draw mode and pens.
      The length in pixels you'll get from DumpString() is the original string ↩
         length -
      perhaps it has been truncated to let this string fit into your raster port.
      After each dump you should clear the contents of your raster port, e. g. ↩
         using
      ClearEOF().
      You may use standard Bitmaps or foreign if OS 3.0 is at least available.
      This routine calls DumpStrLine().
      Your dump_code() does not need to restore its base-register a4, BareED
      has already done this - but all other non-scratch registers must be ↩
         restored
      on exit

pm_FreeProgressBar  = pointer to a function
         Free an earlier obtained ProgressBar inclusive the resources used by it
    INPUTS:
      pb - returned pointer from pm_CreateProgressBar() that points ↩
          to a
      ProgressBar structure which only has one useable item: pb_Window - which
      points to the window used by this progress-bar

    RESULTS:
      none

pm_CreateProgressBar = pointer to a function
         Create a window with a progress bar in it. Not more!

    INPUTS:
      wintitle - String shown as title of the window (this title must be supplied ↩
         !)
      hail - String shown above of progress bar [(optional parameter)] When
           used this string must not contain any format arguments! (valid is
           e. g.: "Completed to")
      stop - String shown in left gadget [(optional parameter)] (e.g.: "Pause")
           --- due to a bug in pm_TogglePBarGad() this string should be
           set up as follow: "Pause/Continue"
      cont - String shown in left gadget as alternative text [(optional)] due ↩
         to
           a bug in pm_TogglePBarGad() this text will never be displayed
           - so there is no necessity to supply it
      cancel - String shown in right gadget [(optional)]
           When you can only do "Pause/Continue" you must use this
           entry instead of "Stop"

     RESULTS:
      pb - pointer to a ProgressBar structure or zero if something went wrong

pm_PullPBarEvent  = pointer to a function
       Let this function do the necessary things to parse and interpret ↩
          messages sent by
       Intuition/GadTools

    INPUTS:
      pb - pointer to a ProgressBar structure
```

```
    RESULTS:
      ID  -0  = if this message has no meaning for you
        -1  = right  gadget  (or  centred  - if a single) has been clicked by
                user
        -2  = left gadget (stop/cont) by user clicked

pm_TogglePBarGad  = pointer to a function
      Change state (text) in left gadget from either Stop to Cont or from Cont to  ↩
         Stop.

      pm_TogglePBarGad performs only the action when all three gadgets texts are  ↩
         supplied.

    INPUTS:
      pb  - pointer to a ProgressBar structure

    RESULTS:
      none

    BUGS:
      does currently not work well
```