BtoC_Doc

COLLABORATORS			
	TITLE : BtoC_Doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

BtoC_Doc

1.1 BtoC version 3.1

```
                           BtoC, version 3.1
                            Decrunch, v 3.0
               Copyright © 1992-1994 by Stefano Reksten
                     of 3AM - The Three Amigos !!!
```

CONTENTS OF THIS GUIDE:

1.2 disclaimer

DISCLAIMER

The author is NOT responsible for the suitability or accuracy of this documentation and/or the program(s) it describes. Any damage directly or indirectly caused by the use or misuse of this documentation and/or the program(s) it describes is the sole responsibility of the user her/him self

----

1.3 copyright

COPYRIGHT

BtoC v3.1, Copyright © 1992-1994 by Stefano Reksten. All rights reserved.

This program may be distributed non-commercially only providing that the executable, source code, documentation and copyright notices remain unchanged and are included with the distribution.
The archive must contain the following directories/files:

```
 BtoCDrawer/
   Docs/
     BtoC.guide
     BtoC.guide.info
   ExplodeStuff/
     GCC/
       Decrunch30.c
       Decrunch30.c.info
       Decrunch30_MIT
       Decrunch30_MIT.info
       Explode.c
       Explode.c.info
     SAS_C/
       Decrunch30.s
       Decrunch30.s.info
       Explode.c
       Explode.c.info
       Makefile
       Makefile.info
       SAS_C_interface
       SAS_C_interface.info
     Decrunch30.c
     Decrunch30.c.info
   BtoC31
   BtoC31.info
 BtoCdir.info
```

Of course Fred Fish is allowed to include this program in his library.
I know there are other people doing like Fred: they are allowed to distribute the archive if the former conditions are respected.

This program is SHAREWARE. If you use it send $15 to the address below ($20 if you are using the crunching/decrunching routines). Doing so, you will become a registered user. This means you've performed your moral duty, and that you risk to get a newer version of this program if and when it will be available.

Any commercial purpose should obey the following rules:
- o Be a registered user,
- o Mention BtoC in the credits somewhere,
- o Send me a copy of the finished program. ;-)

1.4 intro

What's BtoC ?

This program is written to allow the user to transform an IFF ILBM file in C (or Assembler) source code for Image structures along with their data, or in sprite data (for 16, 32 and 64 bit wide AGA sprites!), or in ANSI escape sequences to create a picture by simply typing the file to your CLI window.

```
It is also possible to save the picture's palette in a wide range of tables
like those used for the LoadRGB4/LoadRGB32/SetRB4/SetRGB32,  or even create
an ECS copperlist, or an AGA one. The program is also capable of creating a
mask of the picture (for blitter use) and to select or deselect one or more
bitplanes, and much more.
You can save in C, in Assembler or in Raw data everything you want to, just
in three global files or each picture in his "source" files.  If you prefer
to save Raw data in one global file, the program will give you the possibi-
ty to write to disk the offsets from the beginning of that source file.

-----------------------------------------------------------------------------

What does BtoC need?

o  Kickstart 2.04.

o  Disk space enough to store your files!
```

1.5 cli_usage

```
How to use BtoC ?

CLI usage:

BtoC31
DT=DataType/K,
PT=PaletteType/K,
MT=MaskType/K,
I=Image/S,
FD=ForceDepth/N,
N0=NoBpl0/S,N1=NoBpl1/S,N2=NoBpl2/S,N3=NoBpl3/S,
N4=NoBpl4/S,N5=NoBpl5/S,N6=NoBpl6/S,N7=NoBpl7/S,
O=Offsets/S,
CO=Compression/S,
DS=DataSave/K,
PS=PaletteSave/K,
MS=MaskSave/K,IS=ImageSave/K,
OS=OffsetsSave/K,
S=ShowIFF/S,
SA=SpriteAttached/S,
L=LowercaseLabels/S,
Cont=ContinueOnError/S,
C=CGlobal/K,
A=AsmGlobal/K,
R=RawGlobal,
F=File/A/M

DT=DataType/K:

 Used to specify which data you want to create. Can be one of
 the following strings:
 None, Image, Chunky, Sprite16, Sprite32, Sprite64, ANSI_New, ANSI_Old.
 (Case insensitive.) Default is None.
 NOTE:  If creating sprites, the depth must be 2 bitplanes for
        4 colors sprites or 4 bitplanes for 16 colors sprites.
        Sprites will be high as the picture they come from, but
        wide as much as requested (they will be cut automatically).
 NOTE:  If creating ANSI data, depth must be 3 bitplanes or
        less (due to console device's limitation). Data created
        with ANSI_New will be compatible only with version 36
        or upper of console.device, but it will be smaller and
        faster to type. Moreover, ANSI data will be automatically
        put in a RAW file, overriding DataSave option.
```

PT=PaletteType/K:

Used to specify which palette you want to create. Can be one of the following strings:
None, SetRGB4, LoadRGB4, SetRGB32, LoadRGB32, Copperlist, AA_Copperlist, IFF_ILBM.
(Case insensitive.) Default is None.

MT=MaskType/K:

Used to specify which mask you want to create. Can be one of the following strings:
None, All, Selected.
(Case insensitive.) Default is None.

I=Image/S:

Used to specify if an Image structure is wanted. The structure will have its fields filled with appropriate values calculated from the current picture.
(Case insensitive.) Default is image not wanted.

FD=ForceDepth/N:

Used to coherce the depth of the picture. Useful when writing ANSI data (picture must be 3 bitplanes depth), or sprite data (picture must be 2 or 4 bitplanes depth).
Can be a number from 0 to 8 (0 means always original depth).
If depth is forced to a value greater than its original value then added bitplanes will be filled with zeroes.
(Case insensitive.) Default is to keep always the picture's original depth.

N0=NoBpl0/S, ..., N7=NoBpl7/S:

Used to switch off a bitplane. It will be replaced with an empty one. (Case insensitive.) Default switches ON only all the bitplanes of the picture.

O=Offsets/S:

Used to write offsets of data saved in a Raw file. Offsets of course cannot be saved on a Raw file, but only to a C or Assembler source file. (Case insensitive.) Default is offsets not wanted.

CO=Compression/S:

```
 If this flag is set then Image data or Sprite data will be
 compresed with BYTERUN2 algorythm. (See .doc)
 (Case insensitive.) Default is data not crunched.

DS=DataSave/K, PS=PaletteSave/K, MS=MaskSave/K,IS=ImageSave/K:

 Can be one of the following strings:
 C, Asm, Raw.
 (Case insensitive.) Default is C. This parameter specifies
 whether data should be saved in a C source, in an Asm one or
 as raw data.

OS=OffsetsSave/K:

 Can be one of the following strings:
 C, Asm.
 (Case insensitive.) Default is C. This parameter specifies if
 offsets should be saved in a C source or an Asm one.

S=ShowIFF/S:

 If this flag is set picture will be shown while writing its
 data to disk. After that, the screen containing it will be
 closed.
 (Case insensitive.) Default is not to show the picture.

SA=SpriteAttached/S:

 This flag must be specified if you want to create data for
 attached sprites (16 colors sprites).
 NOTE: Depth must be 4 bitplanes (or forced to), to have
 16 colors. See also ForceDepth command.
 (Case insensitive.) Default is sprites not attached.

L=LowercaseLabels/S:

 Specifying this, all the labels that will be generated will
 be in lower case.
 (Case insensitive.) Default is to keep all the labels like
 the filename.
 NOTE: As the labels are kept EXACTLY like the filename, you
       are adviced NOT to use '.' (or '->', etc :-) in the
       picture's filename, unless you want a compiler error.

Cont=ContinueOnError/S:

 If an error is encountered while processing a picture (or
 more than one) and this flag is not set, the program will stop
 its execution, giving the control back to you. Otherwise the
 work will go on until finished.
 (Case insensitive.) Default is to stop on error.

C=CGlobal/K, A=AsmGlobal/K, R=RawGlobal:

 Used to specify the global filenames. If CGlobal is speci-
 fied, then all data that is to be written in C will be writ-
 ten in that file. The same for AsmGlobal and for RawGlobal.
 If one of them is NOT specified then data will be saved on
 a file composed by the original picture's name followed by
 '.c', '.asm' or '.raw'.
 (Case insensitive.) Default is no global file.
 E.G.: btoc30 DataType image PaletteType loadrgb32 ImageSave
       C PaletteSave Asm Image CGlobal out1 File pic1 File
       pic2 F pic3
       will convert files pic1 pic2 pic3 saving image and
       imagedata on out1.c, while saving their palette on
       pic1.asm, pic2.asm, pic3.asm.

F=File/A/M

 Used to specify the names of the files to be converted.
 If no name is specified, BtoC will open its GUI.


If there is not enough disk space for your output file on the disk, it will
be cut to previous size (or deleted if it was made from scrap).
```

1.6 workbench_usage

```
Workbench usage:

Launching BtoC30 by clicking on its icon or from CLI with no args
will force BtoC30 to open its Graphic User Interface. Here you can
specify the CLI options by chosing them from the menus. To process
an image, just select the 'Process' option.

More features:

o  Started in this way, BtoC30 will search for its "BTOC:BtoC.config"
   file, so you will need not to set all the parameters every time.
   This file can be updated automatically on exit or from user's
   choice. Note that "BTOC:" should be an assignment you made before
   starting the program. (Unless using a program like ReqChange... :-)

o  You can load a new palette from another ILBM file.

o  The window is an AppWindow that will automatically process
   all the icons dropped on it. The window can be closed to get
   an AppIcon that behaves in the same manner. To get back to
   the window, double-click on the AppIcon.

o  You can specify the default coordinates for the AppIcon by
   filling the XPOS, YPOS entries in the icon's toolarray.
   -1 is NO_ICON_POSITION: Workbench will pick a reasonable
   place for the icon.

o  You can drop more than one icon on the AppIcon/AppWindow.
   These will be immediately processed. If a drawer (or a disk)
   is dragged in, all the files inside will be processed. Anyway
   note that recursive directory scanning is not done.

o  Press RETURN and the first string gadget will be activated.
```

```
    If you continue to press return to the last one, the string
    gadget will be deactivated. (Using TAB key you can run back
    and forth through the string gadgets.)
```

1.7 compression

Since its first version BtoC is able to compress data with a custom
routine, based on the well-known ByteRun1 algorhythm. To compress
data you just have to select it in the main menus (or with the
CO=Compression command from CLI). To decrunch your data I enclosed
the source in Assembler for both GCC and SAS C along with other
pieces of code. All you need is enclosed in the ExplodeStuff drawer.

```
  NAME
  Decrunch30  --  Decrunches data compressed with BYTERUN2.

  SYNOPSIS
  result = Decrunch30( dest_ptr, source_ptr )
  D0         A0    A1

  BOOL Decrunch30( UBYTE *, UBYTE * )


  FUNCTION
  Store in a0 the address of an allocated block of CHIP memory
  large enough  to contain  the  decompressed raw  image data.
  Store in a1 the address of the  compressed  raw  image data.
  If data is not compressed with BtoC 2.5 or upper, Decrunch30
  simply returns FALSE, else decrunches data and returns TRUE.
  See ExplodeStuff/SAS_C/SAS_C_interface to use it with SAS C.
  See ExplodeStuff/GCC/Decrunch30.c to use it with GCC.

    My algorythm's name was BYTERUN2 because it is just an improvement
    of ByteRun1. But while reading the datatype/pictureclass.h include
    file, I discovered there *IS* (or there *SHOULD BE* ???) a ByteRun2
    compression algorythm. So I think now I should change my algorythm's
    name, but I'm very lazy and I won't. So please remember BYTERUN2 is
    different from ByteRun2 (anyway if you know HOW ByteRun2 works, pls.
    e-mail me!)
```

1.8 remember

```
 THINGS TO REMEMBER:

 o  The *MOST IMPORTANT* thing to remember is that if you don't specify
    a destination file or path your output files will be saved in the
    same directory of the sources, *NOT* in the current working dir.

 o  Files choosen from CLI will be processed in their order, choosen with
    an ASL requester will be processed in alphabetical order, and choosen
    dragging them in the AppIcon/AppWindow, in the 'clicking order'.
```

```
o  Remember not to insert any character that can be misunderstood in
   your pictures' name, if you want to save a C or ASM source without
   having to modify them manually (e.g.: try to compile a C source
   having a colormap called 'My.pic->01' :-)
```

1.9 history

```
History :

... previous versions were removed since it's a completely NEW program
    (except for the crunching routine).

v3.0: After more than a year, the program has been totally
 rewritten for AGA compatibility, adding chunky, ANSI and Sprite
 data generation, and changing the Graphic User Interface.

v3.1: After having discovered that this program was also an
   Enforcer-hit-generator I bought a MMU and started debugging it.
   Some features added, some routines improved. Works fine with v39
   AllocBitMap, cuts to previous size files that could not be written
   due to disk-full error.
```

1.10 bugs

```
KNOWN BUGS:

First of all I must apologize for the previous 3.0 version.   It was *NOT*
tested as I wrote in the docs :-(  Now I killed my friend and,  as I got a
MMU (finally!), I could test it.  I have corrected all the bugs I've found
and tested this program with Enforcer, SegTracker, Mungwall and Eatmem.

I managed to remove all Enforcer hits I've found. Now the program seems
to work correctly; but I don't exclude there can be some bugs left. So if
you find that some data generation routine doesn't actually work the way
it should do or if you find any bug (or something weird or have any nice
idea, flames, reports or something else), write to me!
```

1.11 author

```
E-mail your messages to

   rekststef@unisi.it

Or, to send me postcards, A4000s, CD32, girls :-) ...

   Stefano Reksten c/o Naimi,
   viale Cavour, 40
   53100 Siena
   Italy
```