GenCodeC

Billault

COLLABORATORS

	TITLE : GenCodeC	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Billault	August 7, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GenCodeC

1.1 GenCodeC

```
                                ~ A C Code Generator

~              for MUIBuilder (C) Totel Eric

                      Version 2.2g

           (C) Billault Ludovic & Xavier  1995-1998
```

```
                    Note : Sorry for our bad english !!, but we think that a  ↩
                       software can't
            be distributed without english documentation.
```

1.2 Distribution

GenCodeC (c) is FREEWARE. Like any FREEWARE program, you mustn't pay more than 20FF, 3$ US, 4DM for this product.

All the files of the GenCodeC (c) archive stay under authors copyright. None modification of these files is authorized without authors permission.

There 's no warranty about this product. You can't blame us for any problem that could occurs using GenCodeC.

1.3 Requirements

- An Amiga
- OS 2.1 or under
- MUIBuilder v2.2 (c)
- MUI (c) : not included :-(
- TextField v2.0 (c) : included in MUIBuilder archive
- WriteCatalog (c) : included in archive

1.4 Installation

Installation

In order to install the new version of GenCodeC (c), you must :

- Copy the Module directory content in the MUIBuilder:Modules directory.
  Be careful, if you use the SAS/C compiler, copy the file named Hook_utility.o.sasc, if you use the gcc compiler, copy the file named Hook_utility.o.gcc. Don't forget to rename this file in Hook_utility.o.
- Copy the TextField/Gadgets directory content in the SYS:Classes/Gadgets directory (provided with MUIBuider (c)).
- If you use gcc or egcs, copy libmui.a in your lib directory and replace the file proto/muimaster.h by muimaster.h provided in GccLib directory.
- That's All !!!

Archive description

```
Module      The required files (data and code) to use GenCodeC (c),
             except TextField.

Docs        The documentations files (english and french in amaguide
            format).

GccLib      Files for gcc or egcs in order to compile without inline
            file.

MUIB        The .MUIB files used for GenCodeC (c) and WriteCatalog (c).

Source      The source files.

Safe Memory The memory manager written by Simon Bullen.
```

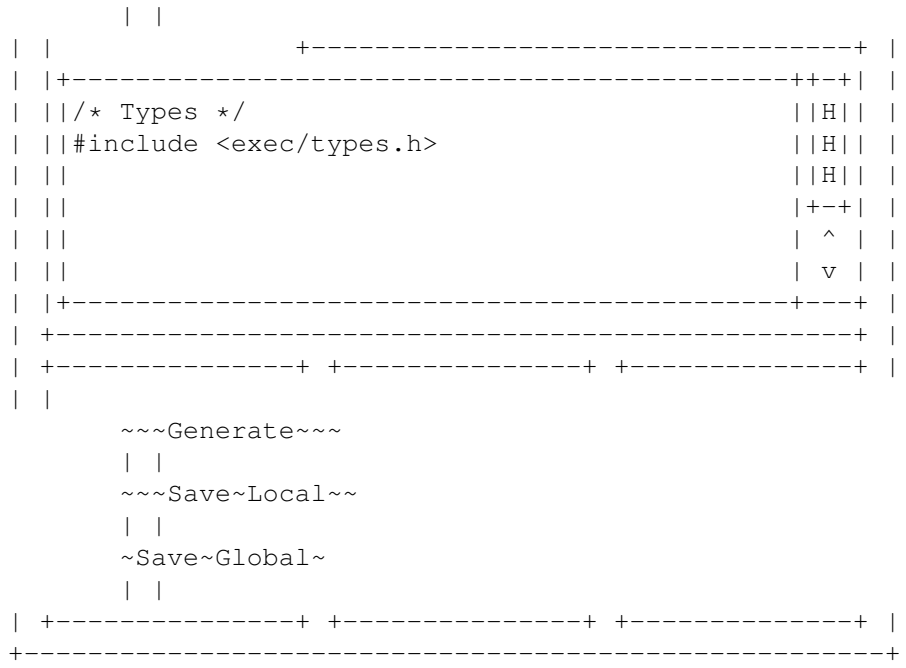
1.5 Usage of the Generator

```
                Description of the generator

 The new features of the generator are :
    - "intelligent" generation of a
        Main~File
                    - files needed for
        localization
         are now generated
    - you can defined specific
        header
         for each project


Description of the GUI

   +-------------------------------------------------------+
   | +---------------------------------------------------+ |
   | |
          ~~~~~~~~~~~~~~~~Information~Field~~~~~~~~~~~~~~~~
          | |
   | +---------------------------------------------------+ |
   |                                      +-+              |
   |
          Generate~Main~File
          |v|              |
   |                                      +-+      +-+     |
   |
          Add~new~entries~in~Catalog~Description~File
          |v|     |
   |                                                +-+     |
   | +---------------+ +--------------+ +---------------+ |
   | |
          ~~~H-Header~~~
          | |
          ~~~C-Header~~~
          | |
          ~~Main-Header~
```



1.6 Generated Code

Generated Code

The generated files depend on the options that the user chooses before generating C code.

Here, a list of the different files generated for each option :

- option 'Generate Main' <--> 'Project Name'Main.c
- option 'Localization' <--> 'Project Name'_cat.h and 'Project Name'_cat.c
- MUI Code <--> 'Project Name'GUI.h et 'Project Name'GUI.c
- Hook functions used <--> 'Hook_utility.h' et 'Hook_utility.c'
- Externals variables or functions used <--> 'Project Name'Extern.h

Then, generated files are :

-
  Main~File
  -
  MUI~Code
  -
  Files~for~localization
  -

```
Hook_utility
            _
File~'Extern.h'
```

1.7 Information Field

```
Information Field

   You can know the generated file by seeing this field.
```

1.8 Generate Main File

```
                Generate Main File

   If you want the program generate a file that contains a 'Main'
   function, check this 'chec mark'.

   In this file, you can find a loop that handles the IDCMP identifiers.
   For more information, see
          Structure~of~Main~File
          .

Note :

   Although you can always generate a Main File, it's possible that this
   file has no sense. A generation of this file is 'correct' only if you
   select 'Code', 'Environment' and 'Declarations' options in
   MUIBuilder (c).
```

1.9 Add new entries in Catalog Description File

```
                New entries in Catalog Description

   If you want add new entries in the Catalog Description file, check
   this 'check mark'. When you click on the
          Generate
           button,a GUI will
   appear and you will be able to modify the Catalog Description file.

   This 'Check Mark' appears only if you select the 'Localization' option
   in MUIBuilder (c).

   Moreover, you can modify the Catalog Description file using the

          WriteCatalog
           program. Be careful, you have to specify the same name
   for the GetStringName field in this program and in MUIBuilder (c)
   (GetString field).
```

1.10 WriteCatalog

```
WriteCatalog (c) is a little utility whose aim is generating two C
sources files in which you find all required functions for localization.



How to execute this program

  Launch WriteCatalog from WB or CLI.

  The options are (only if you launch the program from the CLI) :
     - CDN or CatalogDescriptionName : File name for the catalog
                                       description file
     - GSN or GetStringName : C function name used in order to read
                              the localized strings.
     - Reserved : It's reserved Na! (:-(((()

  You can also specify these parameters in the GUI of the program GUI. If
  you don't specify a file name for the catalog description file, a file
  requester asks you the name !


GUI description

  Put the 'GetStringName' in this field (C function called to read the
  localized strings).

  The second gadget is a TextField gadget so you can modify the catalog
  description file.

  Click on the 'Generate Files' button to generates two C files (.c and
  .h) in the same directory than the catalog description file. When you
  click on this button, the catalog description file is automatically
  saved.

  Click on the 'Save' button to save the catalog description file.

Description of the .c and .h generated files

  The .h file contains all prototypes of the next functions :
      - OpenAppCatalog in order to open the catalog
      - CloseAppCatalog in order to close the catalog
      - FunctionReadString in order to read the localized strings.

  Moreover, the .h file contains a list of all identifiers in order to
  handle localized strings (parameters for the function
  'FunctionReadString').

  The .c file contains the wole implementation of these functions and the
  declaration of the default strings.
```

1.11 H-Header

```
              H-Header

This field is a TextField register. By writting in this textfield you
can modify what it will inserted in the beginning of the .h file
corresponding to the description of MUI interface.

The header that is showed in the register group coresponds to :
  - the "H-Header" file
    if the project has no specific headers;
  - the "'Project Name'.H-Header" file
    if the project has specific headers.

The header is saved in the 'Module/C_Headers' diretory. For more
information, see the '
       Save~Local
       ' and '
       Save~Global
       ' buttons.
```

1.12 C-Header

```
              C-Header

This field is a TextField register. By writting in this textfield you
can modify what it will inserted in the beginning of the .h file
corresponding to the description of MUI interface.

The header that is showed in the register coresponds to :
  - the "C-Header" file
    if the project has no specific headers;
  - the "'Project Name'.C-Header" file
    if the project has specific headers.

The header is saved in the 'Module/C_Headers' diretory. For more
information, see the '
       Save~Local
       ' and '
       Save~Global
       ' buttons.
```

1.13 Main-Header

```
              Main-Header

This field is a TextField register. By writting in this textfield you
can modify what it will inserted in the beginning of the .h file
corresponding to the description of MUI interface.

The header that is showed in the register coresponds to :
  - the "Main-Header" file
    if the project has no specific headers;
```

```
  - the "'Project Name'.Main-Header" file
    if the project has specific headers.

The header is saved in the 'Module/C_Headers' diretory. For more
information, see the '
       Save~Local
       ' and '
       Save~Global
       ' buttons.
```

1.14 Generate

```
                Generate

Click on this button to generate C source files. After the generation,
the program gives the control to MUIBuilder (c).

If you want to know more about the generated files, see

       Generated~Code
       .

Notes :

While the generator produces files, the name of the generated file is
indicated in '
       Information~field
       '.

If the option '
       Add~New~entries~in~Catalog~Description
       ' is selected, a
new window will appear so you will be able to modify the catalog
description file so that your modifications will be included in your
code.
```

1.15 Structure of the Main file

```
                Code Main

    This is a new feature. Now, you can tell the generator to produce a
'main file' where you can find a loop that handles the IDCMP identifiers, a
function init that opens all required resources (MUI, Catalog if you want
localized code ...) and a function end that closes all resources. So you can
test your code without write anything (good !! isn't it:-)).

    Another feature is an 'intelligent' generation, that is to say the
generator doesn't erase your modifications between two generations. But be
careful, you have to respect some constraints for this (not really anoying). The
generator places some comments in the code, you don't have to erase it if you
want the generator works good.
```

We propose to explain you the structure of this 'main file', so you can know what you can or can't modify in the C file. Don't be afraid, the generator makes a backup of your old code in a file named 'old_name'.bak.

To work correctly, the generator 'cut' the 'main file' in some blocks. Some blocks are regenerated all time, some are copied from the old 'main file' and some are computing (they can change, it depends on your options). The regenerated blocks are preceded by a '|' symbol, the copied blocks are preceded by a '>' symbol and the computing blocks by a '?' symbol in the next example (not in the generated source !!!).

Structure of the main file

```
| /* Main-Header File inserted by GenCodeC */
| Header corresponding to Main-Header
| Don't modify it !!!. Modify it by the Textfield requester
|
               Main-Header
                | /* GenCodeC header end */
>
> /* Include generated by GenCodeC */
> Most often the 'Projet Name'GUI.h and some others.
>
| /* Declarations for libraries (inserted by GenCodeC) */
? MUI.library and localization if selected
>
| /* Init() function */
> void init( void )
> {
?   if (!(MUIMasterBase = OpenLibrary(MUIMASTER_NAME,MUIMASTER_VMIN)))
>   {
>      printf("Can't Open MUIMaster Library\n");
>      exit(20);
>   }
> }
| /* GenCodeC init() end */
>
| /* End() function */
> void end( void )
> {
?   CloseLibrary(MUIMasterBase);
>   exit(20);
> }
| /* GenCodeC end() end */
>
| /* Main Function inserted by GenCodeC */
> int main(int argc,char **argv)
> {
>   Creation of your application
>
>   Handling loop
>   while (running)
>   {
>      switch (DoMethod(App->App,MUIM_Application_Input,&signal))
>      {
>         case MUIV_Application_ReturnID_Quit:
```

```
>                 running = FALSE;
>                 break;
>      }
>
>        Events handling defined in MUIBuilder (c)
|        /* Insert your code between the "case" statement and
|          comment "end of case..." */
?            case ID_1:
>                  Write what you want
?              /* end of case ID_1 */
?            case ID_2:
>                  Write what you want
?              /* end of case ID_2 * /
|         /* End computing of IDCMP */ }
>
>          Write what you want
>      }
>      if (running && signal) Wait(signal);
>   }
>   DisposeApp(App);
>   end();
> }
```

Don’t be afraid, that ’s not too complex, it’s only ’logic’ :-).

1.16 MUI Code

MUI Code

The generated code produced by GenCodeC (c) contains two files .c and .h that correspond to the MUI Interface. They have not really changed from the first version of the generator (Thank you to Eric). Just some minors have changed in order to have a best code.

BE CAREFUL, these files are regenerated at each generation, so don’t modify them if you don’t want to lose your work. If you want modify your GUI, use MUIBuilder (c) (:-)), it’s so easy. However, if you have some examples where it’s necessary to modify the code, send us a mail.

1.17 Hook_utility

By using this "mini package" ’Hook_utility’ you can easily install Hook functions. It contains two files (’Hook_utility.h’ and ’Hook_utility.o’). To use it, you just have to link your code with ’Hook_utility.o’ (We hope :-)).
BE CAREFUL: You’ll have to compile your project with ’FAR’ option if you want to use this package or you’ll have to recompile Hook_utility.o.

To install a hook function, just write this (generated by GenCodeC) :

```
struct Hook my_Hook;
InstallHook(&my_Hook,my_function,Data);
```

The Hook structure is the structure used by MUI (c) when you use notification. The function that you want MUI called (when you pressed a button for instance) is my_function and Data is a pointer on data that you can use as you want.

BE CAREFUL, you have to allocate yourself the memory needed for the hook structure, InstallHook just initiliase the structure.

Technical details

The prototype for 'my_function' is defined in 'Hook_utility.h' and his named is 'proto_c_function' e.g
APTR my_function(struct Hook *a0,APTR a2,APTR a1)
The first parameter is the structure itself (this is the first parameter of InstallHook function). a2 and a1 depends on the MUI (c) notification. For more information, see MUI_Notify.doc.

The parameter named Data in InstallHook function is a field of the Hook structure (in fact, it is the h_Data field). So, you can use this pointer to passed some datas in the hook structure and you just have to use a0->h_Data in your function (my_function) to use the data.

Here, you can see an example of a code generated by GenCodeC (c):

```
struct Hook MyHook;
APTR truc(struct Hook *a0, APTR a2, APTR a1);
InstallHook(&MyHook,truc,NULL);
       /* NULL corresponds to the h_Data field */

....

DoMethod(Object->BT, /* BT is the notified object (here a button) */
  MUIM_Notifiy, MUIA_Pressed, TRUE,
  Object->App,
  2,
  MUIM_CallHook,&MyHook
  );
```

When the user will click on the BT Button, the truc function will be called by MUI (c). a0 points to myHook, a2 is Object->App and a0->h_Data is NULL.

So, you can get the Author of the application using a2 (Here, Object->App points to the application) :

```
char *name;
get(a2,MUIA_Application_Author,&name);
printf("%s\n",name);
```

1.18 File 'Extern.h'

File 'Extern.h'

There 's not too many things to say. In fact, it's only an interface

```
between your
               Hook~functions
               , extern variables and so on and your own code.
```

1.19 Save Local

```
                  Save Local

          Click on this button to save all the headers (
                 H-Header
                 ,
                 C-Header
                  and

                 Main-Header
                 ) in three files. These files are located in the
          'Module/c_Headers' directory and have the next names:
            - 'H-Header'    <--> 'Project Name'.H-Header
            - 'C-Header'    <--> 'Project Name'.C-Header
            - 'Main-Header' <--> 'Project Name'.Main-Header

          These files will be used for the project whose name is 'Project Name'.
```

1.20 Save Global

```
                  Save Global

          Click on this button to save all the headers (
                 H-Header
                 ,
                 C-Header
                  and

                 Main-Header
                 ) in three files. These files are located in the
          'Module/c_Headers' directory and have the next names :
            - 'H-Header'    <--> H-Header
            - 'C-Header'    <--> C-Header
            - 'Main-Header' <--> Main-Header

          These files will be used for all projects that have no specific
          headers.
```

1.21 History

```
                   Version 1.0 :  First C Code Generator written by Eric Totel.

      Version 2.0ß:  A main file is generated. So we can test your work quickly
                     (just a little compilation).
                     Generation of files in order to localize your work.
```

```
                Not distributed (???).

Version 2.1 :   New C sources for the generator.
                Some bugs fixed.
                Not distributed.

Version 2.2ß:   Integration of a GUI (generated by MUIBuilder (c) and
                GenCodeC (c) 2.1 and 2.2ß).
                Some other bugs fixed.
                Creation of
           WriteCatalog
           .

Version 2.2c:   Some bugs fixed.
                Addition of an object file of
           Hook_utility.o
            for gcc.

Version 2.2d:   Some bugs fixed.
                Modification of the header of *cat.c file.

Version 2.2e:   One bug fixed :
                    This bug appeared when GenCodeC could not copy
                    "Hook_Utility.*"
                Hook_Utility.o is now generated with NoStackExt and
                    NoStackCheck option
                An entry was added in "Main-Header" :
                    LONG __stack=8192 (in order to avoid bugs with MUI 3.7
                                       and more)

Version 2.2f:   You can now compile with SAS/C, gcc and the others
                    compilers
                Add a file "libmui.a" in order to compile without any pbs
                    with gcc and MUI 3.8 (no need to use inlines)

Version 2.2g:   Allocations are no longer 'PUBLIC'

Version 2.2h:   Add muimaster.h file in GccLib directory in order to
                    compile with gcc
```

1.22 Bugs

```
                How to report bugs

   If you find some bugs, contact us at
          our~adress
          .
```

1.23 Future

```
                Future for GenCodeC
```

```
Wait for MUIBuilder 3.0 !!!

Replace 'Textfield' by 'TextView' written by E.F.Pritchard.

If you have any ideas about an improvement of this generator, contact
us at
       our~address
       .
```

1.24 Acknowledges

Thank you for your work :

- Stephan Stuntz for MUI (c).
- Eric Totel for MUIBuilder (c) and for the first GenCodeC.
- Edd Dumbill for Heddley v1.1 (c). It's a good Guide Generator.
- Jochen Wiedmann for FlexCat (c).
- Simon Bullen for his very good 'Safe Memory' (See the 'Safe Memory' directory in the archive).
- Mark Thomas for his TextField gadget (See the 'TextField' directory in the archive).
- AND the very best of 'the AMIGA (c) Copyright 1995 Amiga Technologies'.

1.25 Authors

M. Billault Ludovic & Xavier
24, rue Mesliers
Les Noëls
41350 Vineuil

FRANCE

E-Mail: Xavier.Billault@ramses.frmug.org or
        Ludovic.Billault@tcc.thomson-csf.com