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

```
                                ~ Générateur~de~Code~C

~             pour MUIBuilder (C) Totel Eric

                      Version 2.2g

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

Nous !!

1.2 Distribution

GenCodeC (c) est FREEWARE. Comme tout logiciel FREEWARE, personne ne doit demander une somme d'argent supérieur au prix d'une disquette et au prix d'envoi pour le distribuer (c'est à dire pas plus de 20FF, 3$US, 4DM).

Tous les fichiers de la distribution de GenCodeC (c) restent sous copyright des auteurs. Toute modification de ceux-ci est donc subordonnée à l'autorisation des auteurs.

Ce programme est distribué sans aucune garantie de fonctionnement. Vous l'utilisez donc à vos risques et périls. Cependant, si votre A2000 se transforme en un joli A4000 (:-) prévénez nous, cela nous interesse.

1.3 Matériels et logiciels nécessaires

- Un Amiga : NON fourni :-(
- OS 2.1 ou plus
- MUIBuilder v2.2 (c)
- MUI (c) : non distribué :-(
- TextField v2.0 (c) : distribué avec MUIBuilder
- WriteCatalog (c) : distribué dans l'archive

1.4 Installation

Installation

Pour installer la nouvelle version de GenCodeC (c), il faut :

- Copier le contenu du répertoire Module dans le répertoire MUIBuilder:Modules.
  Attention, si vous utilisez SAS/C copiez Hook_utility.o.sasc, si vous utilisez gcc copiez Hook_utility.o.gcc. N'oubliez pas de renommer le fichier en Hook_utility.o.
- Copier le contenu du répertoire TextField/Gadgets dans le répertoire SYS:Classes/Gadgets (fourni avec MUIBuider (c)).
- Pour les utilisateurs de gcc/egcs, recopier le fichier libmui.a dans le répertoire lib de gcc et remplacer le fichier proto/muimaster.h par celui fourni (ces fichiers se trouvent dans GccLib).
- C'est tout !!!

Description de l'archive

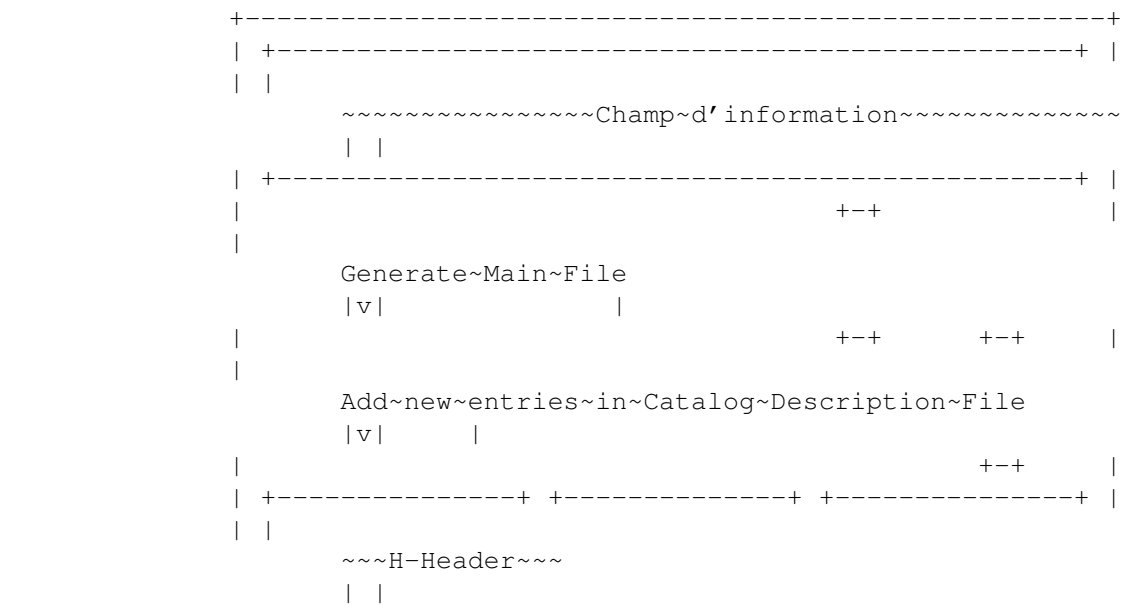
Module	Les fichiers (données et exécutables) nécessaires pour l'utilisation de GenCodeC (c) à l'exception de TextField.
GccLib	Librairie libmui.a et include muimaster.h à utiliser pour compiler sans les inline de MUI (car cela pose des pbs).
Docs	Les documentations (anglaise et française au format amigaguide).
MUIB	Les .MUIB correspondant à l'interface de GenCodeC (c) et WriteCatalog (c).
Source	Les sources.
Safe Memory	L'archive du module de gestion de mémoire écrit par Simon Bullen.

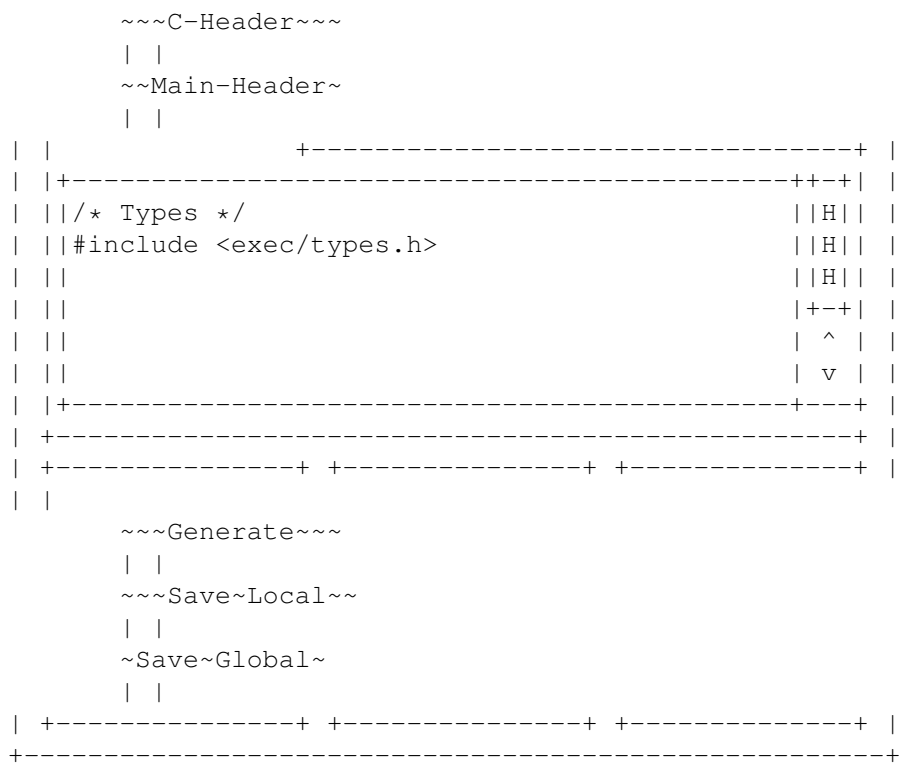
1.5 Utilisation

Description du générateur

Les principales nouveautés par rapport au générateur réalisé par Eric sont :
- génération "intelligente" d'un fichier~Main
- génération des fichiers nécessaires à la localisation
- possibilité d' entêtes spécifiques à chaque projet

Description de l'interface





1.6 Code Généré

Code généré

Les fichiers générés dépendent des options choisies dans MUIBUilder (c) et dans le générateur de code.

Voici les noms des fichiers générés en fonction des options

- option 'Generate Main' <--> 'Nom du projet'Main.c
- option 'Localisation' <--> 'Nom du projet'_cat.h et 'Nom du projet'_cat.c
- Code MUI <--> 'Nom du projet'GUI.h et 'Nom du projet'GUI.c
- Utilisation de fonctions Hook <--> 'Hook_utility.h' et 'Hook_utility.o'
- Utilisation de variables ou fonctions externes <--> 'Nom du projet'Extern.h

Les fichiers générés sont donc les suivants :

-
  Fichier~Main~File
  -
  Code~MUI

```
                -
Fichiers~de~localisation
                -
Hook_utility
                -
Fichier~'Extern.h'
```

1.7 Champ d'information

Champ d'information

Ce champ indique le nom du fichier en train d'être généré par le générateur de code.

1.8 Generate Main File

Generate Main File

Ce 'Check Mark' (Case à cochée en français) permet d'indiquer que l'on veut générer un fichier correspondant au Main() en C.

Dans ce fichier, on trouve notamment la boucle de gestion des identificateurs IDCMP générés par MUIBuilder (c). Pour plus de précision sur la structure de ce fichier, se reporter à

```
Structure~du~fichier~Main
.
```

Remarque:

La génération du fichier Main peut ne pas avoir de sens suivant les options choisies dans MUIBuilder (c). En règle générale, la génération de ce fichier n'a de sens que si les options 'Code', 'Environnement', 'Déclarations' sont cochées et que l'utilisateur demande la génération de l'application.

1.9 Add new entries in Catalog Description File

New entries in Catalog Description

Ce 'Check Mark' permet d'indiquer au programme que l'on veut rajouter des entrées au fichier de description du catalogue généré par MUIBuilder (c). Après avoir appuyer sur le bouton

```
Generate
```

, une jolie IHM apparaitra permettant ainsi de modifier le fichier de description du catalogue.

Ce 'Check Mark' n'apparait seulement que lorsque l'utilisateur choisit d'avoir un code localisé (c'est-à-dire en cochant l'option

de MUIBUilder (c) correspondante).

Une autre manière de modifier le fichier de description du catalogue est l'utilisation du petit utilitaire '
WriteCatalog
'. Attention
cependant à utiliser le même nom de fonction C dans le champ GetStringName de WriteCatalog que dans le champ GetString de MUIBUilder (c).

1.10 WriteCatalog

WriteCatalog (c) est un petit utilitaire qui permet de générer deux fichiers source C (.c et .h) dans lesquels on trouve toutes les fonctions nécessaires à la gestion de la localisation.

Exécution du programme

Lancer le programme à partir du WB ou du CLI.

Les options pour le CLI sont:
- CDN ou CatalogDescriptionName : Nom du fichier de description
- GSN ou GetStringName : Nom de la fonction C utilisée pour lire les chaînes de caractères du catalogue
- Reserved : C'est privé Na! (:-(((

Vous pouvez par ailleurs rentrez ces paramètres plus tard. Par exemple, si vous ne précisez pas le nom du fichier .cd, un filerequester vous permettra de choisir le nom du fichier.

Description de l'interface

Le premier champ texte permet de rentrer le nom de la fonction C utilisée pour lire les chaînes de caractères du catalogue.

Le deuxième gadget est un champ TextField qui permet de modifier le contenu du fichier .cd.

Le bouton 'Generate Files' permet de générer les deux fichiers sources .c et .h dans le même répertoire que le fichier .cd. De plus, une sauvegarde du fichier .cd est effectuée.

Le bouton 'Save' permet de sauvegarder le fichier .cd.

Description des deux fichiers .c et .h générés

Le fichier .h contient le prototypage des fonctions suivantes:
- OpenAppCatalog qui permet d'ouvrir le catalogue
- CloseAppCatalog qui permet de fermer le catalogue
- FonctionReadString qui permet de récupérer les chaînes localisées

Le fichier .h contient de plus la liste des indentificateurs permettant de récupérer les chaînes localisées (paramètres passés à la fonction FonctionReadString)

Le fichier .c contient l'implémentation des fonctions ci-dessus et la déclaration des chaînes par défaut.

1.11 H-Header

H-Header

Ce champ est un registre TextField. Il permet de modifier ce qui sera insérer en début du fichier .h généré par le générateur correspondant au fichier de description de l'interface MUI.

L'entête affiché dans ce registre correspond :
- au fichier "H-Header"
  si le projet n'a pas d'entêtes spécifiques;
- au fichier "'Nom du projet'.H-Header"
  si le projet a des entêtes spécifiques.

Cet entête est sauvegardé dans le répertoire Module/C_Headers. Pour plus de précision sur la sauvegarde des fichiers d'entêtes, voir les deux boutons '
Save~Local
' et '
Save~Global
'

1.12 C-Header

C-Header

Ce champ est un registre TextField. Il permet de modifier ce qui sera insérer en début du fichier .c généré par le générateur correspondant au fichier de création de l'interface MUI.

L'entête affiché dans ce registre correspond :
- au fichier "C-Header"
  si le projet n'a pas d'entêtes spécifiques;
- au fichier "'Nom du projet'.C-Header"
  si le projet a des entêtes spécifiques.

Cet entête est sauvegardé dans le répertoire Module/C_Headers. Pour plus de précision sur la sauvegarde des fichiers d'entêtes, voir les deux boutons '
Save~Local
' et '
Save~Global
'.

1.13 Main-Header

Main-Header

Ce champ est un registre TextField. Il permet de modifier ce qui sera insérer en début du fichier contenant la fonction main().

L'entête affiché dans ce registre correspond :
- au fichier "Main-Header"
  si le projet n'a pas d'entêtes spécifiques;
- au fichier "'Nom du projet'.Main-Header"
  si le projet a des entêtes spécifiques.

Cet entête est sauvegardé dans le répertoire Module/C_Headers. Pour plus de précision sur la sauvegarde des fichiers d'entêtes, voir les deux boutons '
Save~Local
' et '
Save~Global
'.

1.14 Generate

Generate

Ce bouton permet de générer les différents fichiers sources C.
Après avoir générer tous les fichiers, le programme redonne la main à MUIBuilder (c).

Pour connaître les fichiers générés et leurs structures, voir

Code~Généré
.

Remarques:

Lors de la génération, le nom du fichier généré est inscrit dans le '
Champ~d'Information
'.

Si le 'Check Mark' '
Add~New~entries~in~Catalog~Description
' a été
coché, une nouvelle fenêtre apparaîtra lors de la génération pour demander à l'utilisateur de rentrer de nouvelles entrées dans le fichier de description du catalogue pour qu'elles soient intégrées dans les fichiers générés.

1.15 Structure du fichier Main

Code Main

C’est la principale nouveauté de ce générateur. En fait, GenCodeC permet maintenant de créer un fichier main se chargeant de créer votre application avec toutes les resssources nécessaires (les librairies, le catalogue si besoin est ...) et de gérer la boucle de traitement des évènements (par l’intermédiaire des identificateurs IDCMP). Cela permet de pouvoir tester directement son application sans avoir à écrire une seule ligne de code !!!.

L’autre originalité de la génération du fichier main est que lors d’une nouvelle génération les modifications que vous avez effectuées ne sont pas perdues (c’est bien utile finalement). Pour que cela soit possible, il faut respecter certaines petites contraintes concernant la modification du code de ce fichier. En fait, pour pouvoir reconnaître les modifications, le générateur inclut des commentaires à divers endroits du code. Il est donc indispensable de ne pas les effacer.

Nous donnons ci-dessous la structure d’un fichier main avec les commentaires. Cela va vous permettre de savoir ce que vous pouvez modifier et comment le générateur prendra en compte ces modifications. Les commentaires présents ici ne doivent surtout pas être effacé sinon le générateur ne pourra plus s’y retrouver. N’ayez pas trop peur, avant chaque génération du fichier main, le générateur créé un fichier Main.c.bak de façon à pouvoir récupérer vos modifications au cas où.

Le générateur ’découpe’ le fichier main en plusieurs blocs. La reconnaissance des blocs s’effectue grâce aux commentaires. Les blocs sont soit réécrits à chaque fois (typiquement l’entête), soit repris de l’ancien fichier, soit modifiés en fonction des options (typiquement la gestion de la localisation). Les blocs réécrits sont précédés du symbole ’|’, les blocs recopiés sont précédés de ’>’, et les blocs sujets à modification par ’?’.

Structure du fichier Main

```
| /* Main-Header File inserted by GenCodeC */
| Entête correspondant à celle de Main-Header
| A ne pas modifier !!!. Modifier le grâce à l’interface correspondant à
|
               Main-Header
                | /* GenCodeC header end */
>
> /* Include generated by GenCodeC */
> Typiquement les includes correspondant à ’Nom de projet’GUI.h
>
| /* Declarations for libraries (inserted by GenCodeC) */
? Typiquement la MUI.library et Locale.library s’il y a localisation
>
| /* 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)
> {
>   Création de l'application
>
>   Gestion de la boucle des évènements
>   while (running)
>   {
>      switch (DoMethod(App->App,MUIM_Application_Input,&signal))
>      {
>         case MUIV_Application_ReturnID_Quit:
>                 running = FALSE;
>                 break;
>      }
>
>        Gestion des évènements définis dans MUIBuilder (c)
|        /* Insert your code between the "case" statement and
|          comment "end of case..." */
?            case ID_TRUC:
>                  Mettez ce que vous voulez
?              /* end of case ID_TRUC */
?            case ID_TRUC2:
>                  Mettez ce que vous voulez
?              /* end of case ID_TRUC2 * /
|         /* End computing of IDCMP */ }
>
>          Mettez ce que vous voulez
>      }
>      if (running && signal) Wait(signal);
>   }
>   DisposeApp(App);
>   end();
> }
```

Ne vous en faîtes pas, ce n'est pas si compliqué que cela,l'ensemble est assez logique. Normalement, vous pouvez modifier le code à peu près comme vous voulez tant que vous n'effacez pas les commentaires.

1.16 Code MUI

Code MUI

Le code généré par GenCodeC (c) comprend les deux fichiers .c et .h. Il n'a pas profondemment changé de la version réalisée par Eric.

Le fichier .h contient comme à l'habitude la déclaration de la structure de l'application et les fonctions servant à la création et la désallocation de l'application. Le fichier .c correspond en fait à la

description de l'interface MUI.

ATTENTION, ces deux fichiers sont complètement reécrits à chaque génération. Il est donc important d'essayer de modifier peu voire pas du tout ces fichiers puisque les modifications sont perdues d'une génération à l'autre. Nous ne pensons pas que cela vaille la peine de prendre en compte les modifications de ces deux fichiers. En effet, une modification correspond soit à un changement d'interface (on utilise alors MUIBuilder (c) pour effectuer les modifications) soit à un comportement particulier à mettre en place et dans ce cas il vaut mieux l'écrire soit dans le main soit dans un autre fichier. Cependant, si vous avez des exemples où cela s'avèreraient utile, faîtes nous le savoir.

1.17 Hook_utility

Le "mini-package" 'Hook_utility' permet d'installer facilement des fonctions Hook. Il se compose du fichier de prototypage 'Hook_utility.h' et du fichier objet 'Hook_utility.o'. Pour l'utiliser, il suffit d'indiquer à votre compilateur que vous 'linker' votre projet avec 'Hook_utility.o' (enfin, nous espèrons !!!).

ATTENTION: pour utiliser ce package, il faut compiler en mode 'FAR' (le compilateur ne doit pas se servir du registre 'a4' pour accéder aux données). Si vous ne voulez pas compiler en mode 'FAR', il faudra recompiler 'Hook_utility.o' avec les bonnes options (sources fournies).
Par exemple pour SAS/C, l'option à utiliser est DATA=FARONLY.

Pour installer une fonction Hook, il suffit d'écrire le code suivant (généré par GenCodeC (c)) :

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

La structure Hook est celle utilisée par MUI lors d'une notification, la fonction à appeler lors de cette notification est my_function et Data est un pointeur sur des données que l'utilisateur peut utiliser à son gré.
ATTENTION, n'oubliez pas d'allouer vous-même la structure Hook, InstallHook se charge juste d'initialiser la structure comme il le faut.

Détails Techniques

Le prototype de 'my_function' est en fait du type 'proto_c_function' défini dans le fichier 'Hook_utility.h'. La fonction 'my_function' est donc du type :

```
APTR my_function(struct Hook *a0,APTR a2,APTR a1)
```

Le premier paramètre est la structure Hook elle-même (celle passée comme paramètre à InstallHook). Les deux autres paramètres dépendent de la notification faite par MUI, pour plus de précision sur ces deux pointeurs se reporter au fichier de documentation MUI (MUI_Notify.doc).

Les paramètres à passer à la fonction InstallHook sont un pointeur sur la structure Hook elle-même, la fonction C à appeler lors de la notification et enfin un pointeur sur des données (Data). Ce pointeur correspond au champ h_Data de la structure Hook, on peut donc récupérer le pointeur de ces données par la

variable a0->h_Data.

Ex de code généré par GenCodeC (c):

```
struct Hook MyHook;
APTR truc(struct Hook *a0, APTR a2, APTR a1);
InstallHook(&MyHook,truc,NULL); /* NULL correspond au champ h_Data de
                                   MyHook */

....

DoMethod(Object->BT,    /* BT est l'objet notifié (ici un bouton) */
  MUIM_Notifiy, MUIA_Pressed, TRUE,
  Object->App,
  2,
  MUIM_CallHook,&MyHook
  );
```

Lorsque l'utilisateur pressera le bouton BT, la fonction truc sera appelée. a0 pointe sur MyHook, a2 vaut Object->App et le champ a0->h_Data vaut NULL.

On peut donc récupérer par exemple le nom de l'auteur de l'application grâce à la fonction get de MUI en considérant que Object->App est le pointeur sur l'application :

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

1.18 Fichier 'Extern.h'

Fichier 'Extern.h'

Il n'y a pas grand chose à dire sur ce fichier. Il correspond à la déclaration de

```
fonctions~Hook
 et à la déclaration de variables externes
```

utilisées dans MUIBuilder (c).

1.19 Save Local

Save Local

Ce bouton permet de sauvegarder les différents fichiers d'entêtes affichés dans les 3 'Register Group' (à savoir '

```
H-Header
', '
C-Header
'
```

et '

```
Main-Header
') spécifiques au projet.
```

```
Ces fichiers seront sauvegardés dans le répertoire Module/C_Headers.
Les noms des différents fichiers seront :
  - 'H-Header'    <--> 'Nom du projet'.H-Header
  - 'C-Header'    <--> 'Nom du projet'.C-Header
  - 'Main-Header' <--> 'Nom du projet'.Main-Header
```

1.20 Save Global

```
             Save Global

Ce bouton permet de sauvegarder les différents fichiers d'entêtes
affichés dans les 3 'Register Group' (à savoir '
       H-Header
       ', '
       C-Header
       '
et '
       Main-Header
       ').
Ces fichiers seront utilisés pour tous les projets n'ayant pas de
fichiers entêtes spécifiques.

Ces fichiers seront sauvegardés dans le répertoire Module/C_Headers.
Les noms des différents fichiers seront :
  - 'H-Header'    <--> H-Header
  - 'C-Header'    <--> C-Header
  - 'Main-Header' <--> Main-Header
```

1.21 Historique

```
               Version 1.0 :  Programme originel réalisé par Eric Totel.

Version 2.0ß:  Génération d'un fichier contenant la fonction Main().
                   On peut donc tester son travail directement
                   (Juste une petite compilation :-))
               Génération des fichiers traitant la localisation.
               Version non distribuée (???).

Version 2.1 :  Refonte compléte du générateur de code (source).
               Correction de divers bugs.
               Version non distribuée.

Version 2.2ß:  Couplage avec une petite IHM (entièrement générée
               par MUIBuilder (c) et GenCodeC (c) 2.1 et 2.2ß).
               Correction d'autres bugs (Ohh!!
                   on en trouve toujours :-().
               Ajout de l'outil
         WriteCatalog
          permettant de générer
               les fichiers sources C traitant la localisation à
               partir d'un fichier .cd
```

```
                (On peut même rajouter des entrées dans le .cd).

      Version 2.2c:  Correction de bugs minimes.
                     Ajout du fichier
                Hook_utility.o
                 pour gcc.

      Version 2.2d:  Correction de bugs minimes.
                     Changement des headers pour le fichier *_cat.c

      Version 2.2e:  Correction d'un bug :
                         bug si la copie de "Hook_Utility.*" s'est mal passée
                     Modification de Hook_Utility.o :
                         compilé avec l'option NoStackCheck et NoStackExtend de
                         SAS C
                     Modification du "Main-Header" :
                         Ajout de la ligne LONG __stack=8192 (pour éviter des
                         pbs avec MUI3.7 et suivant)

      Version 2.2f:  Modification des "Header" pour pouvoir compiler avec
                         SAS/C, gcc et les autres compilos
                     Ajout d'un fichier "libmui.a" pour compiler avec gcc
                         et MUI 3.8 (pas besoin de passer par les inlines)

      Version 2.2g:  Les allocations ne sont plus 'PUBLIC'

      Version 2.2h:  Ajout du fichier muimaster.h dans GccLib pour compiler
                         avec gcc.
```

1.22 Rapport de Bugs

```
      Report de Bugs

         Comment cela des bugs ???
```

Bon, allez personne n'est parfait donc nous non plus. Faîtes nous donc part des bugs ou dysfonctionnements du générateur le plus tôt possible, de façon à ce qu'ils soient vites corrigés.

1.23 Evolution future

```
                      Evolution de GenCodeC

         Attente de MUIBuilder 3.0 !!!

         Remplacement de 'Textfield' par 'TextView' de E.F.Pritchard

         Si vous avez des propositions à faire et nous pensons que vous en
         aurez, surtout n'hésitez à nous les faire parvenir à notre
                adresse
                .
```

1.24 Remerciements

Nous tenons à remercier les personnes suivantes:

- Stephan Stuntz pour avoir réalisé MUI (c).

- Eric Totel pour nous avoir concocté à tous un programme de génération d'IHM aussi souple (:-) et d'avoir dégrossi le travail pour le générateur de code C.

- Edd Dumbill grâce à qui nous avons pu facilement faire le guide que vous êtes en train de lire (réalisé avec Heddley v1.1 (c)).

- Jochen Wiedmann grâce à qui nous avons pu comprendre facilement l'utilisation de la localisation (Auteur de FlexCat (c)).

- Simon Bullen qui a écrit un super module pour la gestion de la mémoire (voir le répertoire 'Safe Memory' de l'archive).

- Mark Thomas qui a écrit TextField.

- ET bien sûr l'AMIGA (c) Copyright 1995 Amiga Technologies.

1.25 Auteurs

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

FRANCE

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