

## **E code generation**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> E code generation		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 26, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>E code generation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Introduction . . . . .	1
1.3	Le mode classique . . . . .	2
1.4	Le mode environnement . . . . .	2
1.5	Le code généré . . . . .	4
1.6	Les messages d'erreur . . . . .	5
1.7	Les exemples . . . . .	5
1.8	Bugs . . . . .	5
1.9	Historique . . . . .	5
1.10	Futur . . . . .	6
1.11	Distribution . . . . .	6
1.12	L'auteur . . . . .	6

## Chapter 1

# E code generation

### 1.1 Introduction

```
*****

                                GenCodeE (v1.3)
                        Module de génération du code E pour MUIBuilder
                                Mise-à-jour du 10 mars 1994
                                © Copyright 1993, 1994, Lionel Vintenat

*****

Introduction
Le mode classique
Le mode environnement
Le code généré
Les messages d'erreur
Les exemples
Bugs
Historique
Futur
Distribution
L'auteur
```

### 1.2 Introduction

GenCodeE a 2 modes de fonctionnement : le mode classique et le mode environnement.

Le mode classique vous sera familier si vous avez utilisé GenCodeE v1.0, qui était fourni avec MUIBuilder v1.0. En effet, ce mode classique fonctionne comme le faisait GenCodeE v1.0 (d'où son nom), mais avec la gestion de la localisation en plus.

Le mode environnement est la "grande" nouveauté de GenCodeE v1.3. En effet, dans ce mode, GenCodeE va directement modifier votre source E pour y ajouter ou mettre à jour du code MUI. Son utilisation est bien-sûr plus délicate que celle du mode classique, mais est infiniment plus agréable, vous évitant de fastueux couper/copier/coller.

---

La sélection du mode de fonctionnement de GenCodeE se fait par le checkmark "Environnement" dans la fenêtre de génération de code de MUIBuilder.

Le checkmark "Locale" indique si désirez un code localisé ou pas. Son action est la même quelque soit le mode de fonctionnement sélectionné. Par contre, les checkmarks "Déclarations" et "Code" n'agissent pas du tout de la même façon selon le mode fonctionnement. Lisez donc attentivement les paragraphes relatifs à chacun d'eux.

## 1.3 Le mode classique

Dans ce mode, GenCodeE écrit dans le fichier précisé dans le gadget chaîne "Code" le source MUI pour l'application ou l'objet sélectionné. Attention, si ce fichier existait déjà, il est écrasé sans demande de confirmation.

Si le checkmark "Déclarations" est activé, le source produit contiendra les déclarations de variables ainsi que leur valeur d'initialisation pour l'objet sélectionné (ou toutes les variables et leur valeur d'initialisation si c'est l'application entière que vous avez demandé).

Si le checkmark "Code" est activé, le source produit contiendra le code pour l'objet sélectionné (ou tout le code si c'est l'application entière que vous avez demandé).

Vous pouvez combiner ces 2 gadgets pour n'avoir que les déclarations, ou que le code, ou les deux dans le source produit.

Attention, dans ce mode classique, GenCodeE ne produit que du source pour gérer l'interface graphique que vous développez. Contrairement à GenCodeE v1.0, il ne génère pas de code annexe comme par exemple l'ouverture de la bibliothèque muimaster ou la classique boucle d'attente d'une application MUI.

C'est pourquoi j'ai joint à cette archive 2 squelettes de source E pour vous servir de base :

- Application.e pour une application MUI sans localisation
- Application+Locale.e pour une application MUI avec localisation

Pour la partie spécifique à la localisation de Application+Locale.e, je me suis très largement inspiré du code généré par FlexCat. Reportez vous donc à ce programme pour plus de détail. Par ailleurs, c'est un excellent programme qui mérite le détour (son auteur, Jochen Wiedmann, est accessible sur internet à l'adresse [wiedmann@mailserv.zdv.uni-tuebingen.de](mailto:wiedmann@mailserv.zdv.uni-tuebingen.de)).

Ces sources ne sont donnés qu'à titre d'exemple. Vous pouvez donc les modifier comme bon vous semble pour les adapter à vos propres besoins.

Notez que la fonction doMethod contenue dans ces 2 fichiers m'a été donnée par Wouter van Oortmerssen (merci Wouter !) l'auteur d'Amiga E.

## 1.4 Le mode environnement

Dans ce mode, GenCodeE va chercher à modifier le fichier précisé dans le gadget chaîne "Code" , pour ajouter ou mettre à jour le source MUI de l'application ou de l'objet sélectionné.

Pour se faire, votre fichier source doit obéir à quelques règles car GenCodeE n'est pas un programme d'intelligence artificielle, et il ne peut pas

savoir ce qu'il doit changer si on ne l'aide pas un peu !

Précisément, GenCodeE cherche 2 endroits particuliers dans votre fichier source :

- là où sont placées vos déclarations de variables globales
- la procédure où doivent être placées les initialisations de variables et le code lui-même

En effet, dans le mode environnement, GenCodeE génère une procédure spécifique pour la création de chaque objet particulier ainsi que pour l'application. C'est pourquoi toutes les déclarations de variables utilisées par l'interface graphique doivent être globales, de manière à être accessibles par chacune de ces procédures.

Pour que GenCodeE repère dans votre fichier source ces variables, elles doivent être placées sur une ligne qui commence par /\*MUIB\*/. En fait, à chaque génération, GenCodeE efface toutes les lignes qui commencent par /\*MUIB\*/ et les remplace par les nouvelles. Attention, toutes ces déclarations doivent se suivre sans interruption. GenCodeE stoppera ses remplacements dès qu'il rencontre une ligne qui ne commence pas par /\*MUIB\*/.

Tout ce que vous avez donc à faire, est de placer avant la 1ère génération une ligne qui ne contient que /\*MUIB\*/ à l'endroit où vous déclarez vos variables globales. Après GenCodeE se charge de tout.

Dans le mode environnement, GenCodeE place les initialisations de variables et le code de l'interface graphique dans une procédure spécifique à chaque objet et à l'application. Pour l'application, cette procédure s'appelle `create_app()` et pour un objet de label `ObjLabel`, elle s'appelle `create_ObjLabel()`.

Donc pour générer du code, GenCodeE commence par rechercher une ligne qui commence par "PROC `create_app()`" ou "PROC `create_ObjLabel()`" selon le cas, puis il recherche ensuite le `ENDPROC` final de cette procédure. Après il efface tout le contenu de cette procédure avant d'y inclure les nouvelles initialisations de variables et le code.

Tout ce que vous avez donc à faire, est de placer avant la 1ère génération une ligne qui ne contient que "PROC `create_app()`" ou "PROC `create_ObjLabel()`" selon le cas, suivie d'une autre ligne qui ne contient que "ENDPROC". Après GenCodeE se charge de tout.

Comme l'influence des checkmarks "Déclarations" et "Code" sur le source généré est un petit peu complexe, voici un tableau qui présente clairement les choses :

		"Déclarations" ON	"Déclarations" OFF
"Code" ON	initialisations+code	seulement code	
"Code" OFF	seulement déclarations	rien !	

Prenons un exemple simple qui reflète 99% des cas. Pour votre interface graphique, vous voulez pouvoir la créer d'une traite, mais vous voulez aussi pouvoir accéder à un élément particulier (disons une liste de label `ma_liste`) pour un besoin spécifique. Vous ajoutez donc à votre source :

- une ligne qui ne contient que /\*MUIB\*/ là où vous déclarez vos variables globales
- la procédure vide `create_app()`
- la procédure vide `create_ma_liste()`

Ensuite, pour la création ou la mise à jour des variables, vous cliquez sur le bouton "Code Appli" avec le checkmark "Déclarations" activé et le checkmark

"Code" désactivé. Pour la création ou la mise à jour des initialisations et du code à l'intérieur des procédures, vous activez les 2 checkmarks "Déclarations" et "Code", puis vous cliquez sur le bouton "Code Appli" ou le bouton "Objet Code" selon le cas. Et à chaque fois, vous retrouverez votre source correctement modifiée par GenCodeE.

Notez qu'en activant seulement le checkmark "Déclarations" et pas le checkmark "Code", et en cliquant sur "Objet Code", vous remplacez toutes les déclarations de variables globales par celles qui ne concernent que l'objet sélectionné. Ceci est rarement ce que vous souhaitez, car ce sont toutes les variables de l'interface (et pas seulement celles relatives à un objet) qui sont intéressantes.

Un dernier conseil : faites des essais avec ce mode de fonctionnement avant de l'utiliser réellement. Comme il modifie directement votre source (et donc des données très précieuses), il est prudent de bien comprendre son mécanisme avant toute utilisation sur un de vos sources.

## 1.5 Le code généré

Comme MUI-Builder génère du code générique qui est ensuite interprété par des modules, le code E généré est identique dans son contenu au code C généré. Reportez vous donc au paragraphe sur le langage C pour tout ce qui concerne les variables générées, la notification et la localisation. La seule vraie différence concerne une constante MUI\_TRUE qui doit être définie à 1 et qui est utilisée à la place de TRUE (qui vaut -1) pour MUI, car celui-ci ne reconnaît pas toujours la valeur -1 comme vrai.

Le seul point délicat concerne les chaînes entrées dans MUI-Builder. La règle générale est de les entrer comme si vous vouliez les utiliser en C à une exception près : contrairement à ce qu'il est dit avant, le caractère " peut être entré tel quel (ce n'était gênant que pour le C). C'est le module de génération de code E qui s'occupera de la traduction. Plus exactement ce module reconnaît :

- \r qui est remplacé par \b
- \n, \t et \e laissés tels quels
- \0oo où oo est un nombre en octal : attention le \ doit être obligatoirement suivi par un 0, suivi lui-même de 2 chiffres entre 0 et 8
- \xhh où hh est un nombre hexadécimal : attention le \ doit être obligatoirement suivi par un x en minuscule, suivi lui-même de 2 symboles corrects (chiffre ou lettre entre a et f, ou A et F)
- le caractère ' est remplacé par \a

Attention, si la syntaxe de \0oo ou de \xhh n'est pas respectée, le module de génération du code E ne signalera pas d'erreur mais le résultat ne sera sûrement pas celui qui était attendu !

Si le module de génération de code E rencontre \033 ou \x1B (valeur 27 en décimal, c'est à dire le code de ESC), il le remplacera par \e. Si par contre, il rencontre un \0oo ou un \xhh de valeur décimale différente de 27, il produira une chaîne E non pas sous la forme 'chaîne', mais sous la forme [ "c", "h", "a", "i", "n", "e", 0 ]:CHAR car il n'y a pas d'équivalent en E aux \0oo et aux \xhh du C. Mais les 2 formes sont totalement équivalentes.

Enfin, le code source produit contient plein de macros identiques à celles du C. Pour utiliser ces macros, vous aurez besoin d'un préprocesseur annexe car Amiga E ne le permet pas à l'origine. C'est pour cela qu'est jointe

à cette archive une autre archive, Mac2E.lha (également disponible seule sur aminet). Vous trouverez dans cette archive tout ce qu'il faut pour exploiter les fichiers sources générés, avec en particulier un préprocesseur spécialisé dans le remplacement des macros et destiné à Amiga E. Reportez-vous donc à cette archive pour plus de détails.

## 1.6 Les messages d'erreur

En cas d'erreur, GenCodeE affiche un requester explicatif :

- "Out of memory !" -> GenCodeE a besoin de plus de mémoire !
- "I/O problem !" -> GenCodeE ne peut pas trouver, ou lire, ou écrire un fichier
- "Incorrect source format !" -> GenCodeE ne peut pas trouver dans votre fichier source soit /\*MUIB\*/ ou create\_app() ou create\_ObjLabel() ou ENDPROC selon le cas
- "You must generate a label for the object !" -> vous avez demandé le code d'un objet particulier en mode environnement sans générer un label pour lui

## 1.7 Les exemples

Tous les exemples fournis ont été faits en mode environnement à partir soit de Application.e pour les exemples non localisés, soit à partir de Application+Locale.e pour les exemples localisés.

Pour les exemples non localisés, il n'y a rien de spécial à dire. J'ai suivi la méthode de génération standard décrite dans le paragraphe du mode environnement. J'ai simplement ajouté quelques lignes pour l'ouverture et la fermeture des fenêtres.

Pour les exemples localisés, j'ai en plus fait un chercher/remplacer (sensible aux majuscules et aux minuscules !) du mot "application" par le nom de l'application, puis j'ai ajouté avec un copier/coller du code généré par FlexCat (encore une fois, si vous ne le connaissez pas, essayez-le de toute urgence !). Seule Click a demandé en plus de retoucher un peu le code de FlexCat pour changer 2 "\033" par "\e" dans une chaîne E.

Enfin, pour chaque exemple localisé, j'ai fourni en plus du source et de l'exécutable les fichiers xxxx.cd, xxxx.ct (version française), xxxx.catalog (version française) et xxxx\_cat.e (le code généré par FlexCat pour ce catalogue).

## 1.8 Bugs

Aucun de connu.

## 1.9 Historique

---



Version 1.0 : - 1ère version distribuée avec MUIBuilder v1.0  
Version 1.1 : - un petit bug corrigé (GenCodeE oubliait parfois d'indenter le début du code)  
- version jamais distribuée  
Version 1.2 : - ajout du support de la localisation  
- version jamais distribuée  
Version 1.3 : - ajout du mode environnement  
- 2ème version distribuée avec MUIBuilder v1.1

## 1.10 Futur

L'idéal pour un tel module de génération de code est de pouvoir utiliser la compilation séparée. Comme ceci, les mises à jour du code de l'interface se font localement à un fichier, auquel seul le module a accès. Cela est plus facile pour tout le monde.

Malheureusement, Amiga E ne supporte pas encore cette possibilité. Donc, GenCodeE évoluera sans doute beaucoup (mais pour le mieux ;-)) en parallèle avec MUIBuilder bien-sûr, mais aussi en parallèle avec l'implémentation future (soit par Amiga E lui-même, soit par un outil externe) de la compilation séparée.

## 1.11 Distribution

GenCodeE et tous les fichiers annexes contenus dans le répertoire du langage E ne peuvent en aucun cas être distribués en dehors de l'archive de MUIBuilder sans mon autorisation. De même, tous ces fichiers restent sous copyright de l'auteur, et ne peuvent être modifiés sans mon accord.

Enfin, je dégage toute responsabilité quant à l'utilisation de ce programme et les dommages qu'il pourrait causer : vous l'utilisez à vos risques et périls !

## 1.12 L'auteur

Vous pouvez me joindre par courrier :

- à mon adresse étudiante valable jusqu'en juillet 1994  
inclus :

Lionel Vintenat  
appartement 21  
11 rue François Oulié  
31500 TOULOUSE  
FRANCE

- à mon adresse familiale :

Lionel Vintenat  
3 impasse Boileau

---

Lotissement Les Termes  
87270 COUZEIX  
FRANCE

Ecrivez moi plutôt à mon adresse étudiante jusqu'en juillet 1994 car j'y suis beaucoup plus souvent qu'à mon domicile familiale.

Vous pouvez également me joindre sur INTERNET. Mon adresse e-mail est vinténat@irit.fr. Je préfère de très loin que vous me contactiez par mail que par courrier. Je répondrai toujours aux questions qui me seront posées par mail, par contre n'espérez pas de réponses à un courrier (je suis très fainéant dès qu'il s'agit de prendre un stylo...).