

LES MACROS APPRENTISSAGE



GMA 2 - LES MACROS - APPRENTISSAGE

INTRODUCTION

Vous pouvez tout à fait utiliser la console sans jamais vous servir de macros et sans que cela vous manque. C'est tout à fait vrai mais c'est lié à une méconnaissance du rôle des macros. Rien ne sert d'utiliser constamment des macros mais ne pas en utiliser c'est passer à côté d'un outil puissant qui peut vous rendre la vie d'opérateur plus confortable.


Pour l'exercice j'ai choisi 2 macros que j'ai mis à jour récemment pour la version 3.0.0.4 et que vous pouvez télécharger sur le forum **LC FORMATION** ainsi que 2 autres macros créées par **Andréas GLAD** que vous pouvez croiser sur le forum **MA-SHARE** et qui est pour ma part la personne qui connaît le mieux les consoles **Grand MA2**.

AVERTISSEMENT : Toutes les macros de l'exercice sont réalisables et utilisables uniquement à partir de la version 3.0.0.4 !

MACRO LAMP ON 3.0.0.4

Cette macro permet de faire un **LAMP ON** automatiquement sur une plage de groupes présents dans la palette **GROUPE**. Le LAMP ON s'effectue groupe après groupe évitant ainsi soit d'allumer l'ensemble des lampes du patch en même temps, soit de le faire manuellement en restant planter à la console.

Pour éviter de lancer accidentellement la macro, un serial code sera demandé pour exécuter la macro.



No.	Cmd	Wait	
1	SetUserVar \$Serial = "156"	0.01	Serial cod
2	SetUserVar \$PremierGroupe = ("Premier groupe a traiter :")	0.01	Defini le p
3	SetUserVar \$DernierGroupe = ("Dernier groupe a traiter ?")	0.01	Defini le d
4	AddUserVar \$DernierGroupe = 1	0.01	Ajout 1 ur
5	SetUserVar \$GoLamp = ("Serial LAMP ON ?")	0.01	Validation
6	[\$GoLamp==\$Serial] Group \$PremierGroupe	0.50	Sélectionn
7	[\$GoLamp==\$Serial] At Preset *."Lamp On"	5.00	Effectue li
8	[\$GoLamp==\$Serial] ClearAll	0.01	Vide le pr
9	AddUserVar \$PremierGroupe = 1	0.01	Incément
10	[\$DernierGroupe==\$PremierGroupe] Go Macro 1.64.13	0.01	Tous les g
11	Go Macro 1.64.6	0.01	Il reste de
12	*****	*****	*****
13	SetUserVar \$GoLamp = ""	0.01	initialise le
14	SetUserVar \$PremierGroupe = ""	0.01	Initialise r
15	SetUserVar \$DernierGroupe = ""	0.01	Initialise in

LIGNE N°1 : SetUserVar \$Serial = "156"

Cela permet de définir la variable du serial code 156. Dorénavant chaque fois que **\$Serial** sera utilisé il fera référence à 156. Vous pouvez bien sûr remplacer 156 entre les guillemets par le code de votre choix.

SetUserVar ou **SetVar** : Permet de déclarer une variable. **SetVar** sera utilisé si vous travaillez à plusieurs consoles en réseaux, la macro sera exécutable depuis n'importe quelle console. **SetUserVar** permet d'utiliser la macro uniquement sur la console où se trouve la macro même si la console est en réseau avec d'autres.

LIGNE N°2 : SetUserVar \$PremierGroupe = ("Premier groupe a traiter ?")

Une seconde variable nommée **\$PremierGroupe** est déclarée. Elle va contenir le numéro du premier groupe. Les parenthèses () vont permettre d'ouvrir un Popup afin de saisir le numéro du premier groupe.

LIGNE N°3 : SetUserVar \$DernierGroupe = ("Dernier groupe a traiter ?")

Une troisième variable nommée **\$DernierGroupe** est déclarée. Elle va contenir le numéro du dernier groupe. Un Popup grâce aux Parenthèses s'ouvrira à nouveau.

LIGNE N°4 : AddUserVar \$DernierGroupe = 1

AddUserVar permet d'étendre ou d'incrémenter une variable existante. La variable **\$DernierGroupe** est incrémentée de 1. Cela va permettre d'exécuter le LAMP OFF sur le dernier groupe avant l'initialisation des variables à la fin de la macro. Sans cela le dernier groupe ne serait jamais traité.

LIGNE N°5 : SetUserVar \$GoLamp = ("Serial LAMP ON ?")

Déclaration d'une nouvelle variable qui va tester si le Serial code correspond au Serial code original. Un Popup grâce aux parenthèses s'ouvre permettant de saisir le Serial.

LIGNE N°6 : [\$GoLamp==\$Serial] Group \$PremierGroup

Cette ligne est une condition qui veut dire si la variable **\$GoLamp** (Le serial saisi) est égal à la variable **\$Serial** (le serial original) alors le premier groupe est sélectionné.

LIGNE N°7 : [\$GoLamp==\$Serial] At Preset *."Lamp On"

Cette ligne est une condition également qui veut dire si la variable **\$GoLamp** (Le serial saisi) est égal à la variable **\$Serial** (le serial original) alors le preset nommé Lamp On est activé. Veuillez noter le temps de dans la colonne **WAIT** qui permet de garder activé le preset 5s afin de permettre l'amorçage des lampes. Certaines fixtures ont besoin d'un certain temps pour déclencher la lampe. Veuillez noter également comment le nom du preset est écrit : *."Lamp On".

GMA 2 - LES MACROS - APPRENTISSAGE

MACRO LAMP ON 3.0.0.4

Pourquoi un astérisque et un point devant le nom du preset Lamp On ?

Pour pouvoir activer plusieurs presets avec le nom Lamp On. Si vous utilisez l'auto create avec diverses familles de fixtures, vous pourrez trouver dans la palette **CONTROL** plusieurs presets nommés Lamp On. Sans l'astérisque et le point, 1 seul preset (généralement le premier preset Lamp Off de la palette) serait activé.

LIGNE N°8 : [\$GoLamp==\$Serial] **ClearAll**

Comme pour les lignes n°6 et 7 si la condition est rempli, un **ClearAll** est effectué pour vider le programmeur.

LIGNE N°9 : **AddUserVar** \$PremierGroupe = 1

On Incrémente la variable **\$PremierGroupe** afin de pouvoir traiter le groupe suivant.

LIGNE N°10 : [\$DernierGroupe==\$PremierGroupe] **Go Macro** 1.64.13

Cette ligne permet de tester la condition si le dernier groupe est égale au premier groupe alors aller à la ligne n°13 de la macro. Exemple nous avons sélectionné la plage de groupe 2 à 10. Le groupe 2 vient d'être traité. La condition Premier groupe / dernier groupe n'est donc pas rempli, la macro ne saute pas à la ligne n°13 mais à la ligne suivante.

Quand le groupe n°10 sera traité, la condition sera rempli et la macro passera de la ligne n°10 à la ligne n°13.

Le chiffre **64** correspond à l'emplacement de la macro dans la palette **MACRO**. Si vous souhaitez déplacer la macro dans la palette il faudra donc modifier **64** par le numéro du nouvel emplacement. Le chiffre **13** qui suit correspond à la ligne n° **13**.

LIGNE N°11 : **Go Macro** 1.64.6

Cela permet de sauter à la ligne n°6 de la macro et donc de recommencer la procédure Lamp On pour le groupe suivant puisque la condition Premier groupe / Dernier groupe n'est pas rempli à la ligne n°10. Comme pour la ligne n°10, le chiffre **64** correspond à l'emplacement de la macro dans la palette **MACRO** et le chiffre **6** à la ligne n°6.

LIGNE N°12 :

C'est une ligne de commentaire. Elle est de couleur rouge car elle est désactivé dans la colonne **DISABLED** mais elle pourrait tout à fait être activée sans poser de problème à la macro car le caractère **#** est utilisé. Pour écrire un commentaire dans la colonne **CMD** il faut commencer par **#** et écrire entre **guillemets** ensuite.

LIGNE N°13 : **SetUserVar** \$GoLamp = ""

LIGNE N°14 : **SetUserVar** \$PremierGroupe = ""

LIGNE N°15 : **SetUserVar** \$DernierGroupe = ""

Tous les groupes ont été traités, la macro est arrivée à son terme. Toutes les variables peuvent donc être initialisée c'est à dire quelle n'auront plus aucune valeur.

Les guillemets ne doivent pas comporter d'espace entre eux.

Note : Dans ce cas précis de cette macro il n'est pas vital d'initialiser les variables mais c'est une bonne façon que je vous conseille de faire. Pour avoir des noms de variables qui soient significatifs vous pourrez toujours utiliser par exemple PremierGroupe, DernierGroupe etc... comme nom de variable dans d'autres macros si besoin, chose qui poserait problème si la variable n'est pas initialisée. **Sachez aussi qu'une variable garde sa valeur indéfiniment durant toute la session et est conservée si le show est sauvegardé. Les variables sont stockées non pas dans le logiciel mais dans la mémoire de l'ordinateur. (Tester sur ONPC2 à vérifier sur console).**

Faites un petit test avec la macro simple suivante :

Ligne n°1 : **SetUserVar** \$mavariab = (numero groupe)

Ligne n°2 : **Group** \$mavariab

Lancer la macro dans le popup qui s'ouvre saisir le numéro d'un groupe. Par ex 8.

Le groupe 8 va être sélectionné.

Faites un ClearAll, et dans la macro désactiver la ligne n°1

sauvegarder le show et quitter ONPC2

Relancer ONPC2 et cliquez sur la macro.

Le groupe 8 sera sélectionné alors que la variable est désactivée à la ligne n°1 et donc censé n'être pas déclarée !

Donc initialiser les variables est le bon geste à avoir.

LES GUILLEMETS :

Les guillemets permettent de traiter du texte court dans la ligne de commande, d'écrire des mots qui sont réservés à la syntaxe et ou des espaces.

Dans le cas précis de la macro, nous aurions pu ne pas mettre les guillemets aux lignes **1, 2, 3, 5** car nous avons traités que des chiffres numériques.

A contrario les guillemets doivent être bien présents dans la ligne n° **7** pour le nom du preset Lamp On qui lui est une chaîne de caractères et contient un espace.

GMA 2 - LES MACROS - APPRENTISSAGE

MACRO SOLO ON - OFF

Cette macro est intéressante dans sa conception pour l'apprentissage comme nous allons le voir.

Elle permet de simuler la fonction **SOLO** sur une sélection de fixtures avec un temps de fade et un niveau de dimmer réglable sur l'executor en mode **SELECT**.

Elle se compose de 2 macros. Une macro qui contient les variables et une autre qui exécute l'action.

MACRO DE STOCKAGE DES VARIABLES

Bien qu'il soit tout à fait possible d'avoir les variables dans une seule et même macro, il est parfois plus pratique de les séparer. Cela permet aussi de vous rendre compte que peu importe où se trouve une variable elle reste disponible dans n'importe quelle macro ou en ligne de commande tant qu'elle n'est pas initialisée.

Edit Macro 37 'SOLO Stock Var'			
No.	Cmd	Wait	
1	SetUserVar \$SoloDim ("Niveau dimmer à baisser ? : 100 = 0%")	FOLLOW	Niveau de dimmer
2	SetUserVar \$SoloFadeOut ("Temps de fade SOLO ON ?")	FOLLOW	Temps de fade
3	SetUserVar \$SoloFadeIn ("Temps de fade SOLO OFF ?")	FOLLOW	Temps de fade
4	SetUserVar \$DimmerCent = 100	FOLLOW	Condition pour
5	[\$SoloDim==\$DimmerCent] Assign Executor \$SelectedExec /autostop=0	FOLLOW	Si SoloDim est
New			

Nous n'allons pas nous attarder sur les lignes n° 1 à n° 4 de cette macro, ce sont 4 déclarations de variable **SetUserVar** avec un Popup qui s'ouvre pour les 3 premières pour la saisie de valeurs par l'opérateur.

Cette macro donc une fois lancée et les valeurs de variables demandées saisies ne servira plus jusqu'au prochain changement de valeurs de variables si il y a lieu.

LIGNE N°5 :

C'est une condition permettant de désactiver l'option **AUTOSTOP** de l'executor **SELECT** si la valeur demandée dans la variable \$SoloDim (ligne n°1) est égale à 100.

Une valeur de 100 indique que l'executor sera mis à 0%. De ce fait si l'option **AUTOSTOP** est active, l'executor sera stoppé arrivé à zéro. Pas bien !!!



MACRO SOLO ON - OFF

Principe de fonctionnement :

Votre executor **SELECT** a une cue envoyée qui contient votre état lumineux. Vous faite une sélection de 2, 3 fixtures dont le dimmer restera à **FULL** tandis que toutes les fixtures restantes de l'état lumineux auront le dimmer qui baisse du fait que l'executor baissera selon la valeur de la variable \$SoloDim.

Sur la photo, la macro est lancée (mode ON), l'executor a le fader qui est ramenée à la valeur \$SoloDim. Les fixtures sélectionnées conservent le dimmer à **FULL**.

Note : Dans la ligne de commande vous pouvez remarquer l'icône " P " qui indique qu'il existe des fixtures parkées.



La macro est activée à nouveau (mode OFF), l'executor est à nouveau mis à **FULL**.



GMA 2 - LES MACROS - APPRENTISSAGE

MACRO SOLO ON - OFF

C'est une macro qui s'exécute en deux clics pour fonctionner sur le principe d'un bouton **TOGGLE ON / OFF**.

CLIC 1 : La macro s'exécute jusqu'à la ligne n°5 puis s'oppe grace au **GO** (Colonne **WAIT**) de la ligne n° 5. Elle simule donc le **SOLO ON**

CLIC 2 : La macro s'exécute de la ligne n°6 à 10. Elle simule le **SOLO OFF**.

Edit Macro 38 'SOLO ON-OFF'			
No.	Cmd	Wait	
1	# "Mode SOLO ON"	FOLLOW	-----
2	Store Group 2001 /o	0.01	Store la select
3	Park PresetType "dimmer" At Group 2001	0.01	parke le dimme
4	Executor \$SelectedExec At - \$SoloDim Fade \$SoloFadeOut	0.01	Selectionne les
5	# "Mode SOLO OFF"	GO	----- Arret t
6	Assign Macro 1,38.7 /wait=\$SoloFadeIn	0.01	
7	Executor \$SelectedExec At + \$SoloDim Fade \$SoloFadeIn	2.00	fixtures restar
8	Unpark Group 2001	0.01	Deparke le gro
9	Delete Group 2001	0.01	Supprime le gr
10	ClearAll	0.01	
New			

LIGNE N°1 :

C'est une ligne de commentaire comme nous l'avons vu dans le premier exercice d'apprentissage de macro.

LIGNE N°2 : Store Group 2001 /o

Syntaxe simple, Une sélection de fixture ayant été faite avant le lancement de la macro, cette sélection sera sauvegardée dans le Groupe 2001. Le **o** après le slash (/) signifie **Overwrite**, c'est à dire si un groupe existe dans l'emplacement 2001, celui-ci sera écrasé sans qu'une fenêtre de confirmation vous le demande.

LIGNE N°3 : Park PresetType "dimmer" At Group 2001

Les dimmer du groupe 2001 sont parkés.

LIGNE N°4 : Executor \$SelectedExec At - \$SoloDim Fade \$SoloFadeOut

Le fader de l'executor SELECT baisse pour atteindre le niveau de la variable \$SoloDim avec un temps de fade défini par la variable \$SoloFadeOut.

Note : La variable **\$SelectedExec** est une variable définie par MA depuis la version 3.0.0.2 et fait référence à l'executor qui est en mode SELECT. La ligne aurait pu être écrite comme ceci " **Executor At - \$SoloDim Fade \$SoloFadeOut** " et cela aurait fonctionner. Mais c'est une bonne habitude d'utiliser désormais cette variable.

LIGNE N°5 :

C'est une ligne de commentaire simple mais qui contient un **GO** dans la colonne **WAIT**. Cela permet de stopper la macro qui attend un nouveau clic de l'opérateur.

LIGNE N°6 : Assign Macro 1.38.7 /wait = \$SoloFadeIn

Cette syntaxe permet d'assigner la valeur de la variable \$SoloFadeIn à la colonne **WAIT** de la ligne n°7 de la macro pour permettre à la macro d'attendre que le fader de l'executor soit remonter à FULL selon le temps de fade In définie dans la variable \$SoloFadeIn.

Rappel : 1. 38.7 signifie que l'emplacement de la macro est le n°38 et 7 correspond au n°de ligne.

LIGNE N°7 : Executor \$SelectedExec At + \$SoloDim Fade \$SoloFadeIn

L'executor SELECT fader est ramené à FULL selon le temps de fade défini dans la variable \$SoloFadeIn Le temps de la colonne WAIT qui a été défini par la ligne n°6 entre en vigueur pour retarder les lignes restantes 8 à 10.

LIGNE N°8 : Unpark Group 2001

Déparke les fixtures contenues dans le groupe 2001.

LIGNE N°9 : Delete Group 2001

Supprime le groupe 2001 devenu inutile.

LIGNE N°10 : ClearAll

Vide le programmeur.

AVERTISSEMENT :

La macro, lancée une première fois (**mode ON**) **DOIT TOUJOURS être lancée une seconde fois (mode OFF) pour se terminer et éviter ainsi que les fixtures sélectionnées restent parkées !!!**

GMA 2 - LES MACROS - APPRENTISSAGE

MACRO QBACKUP CUE

Crée par **Andréas GLAD** et disponible sur **MA-SHARE** cette perle de macro permet de stocker une cue ou une plage de cue de l'executor SELECT dans la palette **SEQUENCE**.
Chaque séquence à un nom spécifique composé de la date et l'heure de sauvegarde.

El permet également de travailler en relation avec la seconde macro de **Andréas GLAD** que nous verrons plus loin dans ce tutoriel.

Edit Macro 25 'QBACKUP CUE'		
No.	Cmd	
1	SetUserVar \$backup="QBackup "	FO
2	AddUserVar \$backup="\$date"	FO
3	AddUserVar \$backup=" "	FO
4	AddUserVar \$backup="\$time"	FO
5	Store Sequence \$"backup" Cue 0.001	FO
6	SetUserVar \$from=("backup de la cue :")	FO
7	SetUserVar \$to=("backup jusqu'à la cue :")	FO
8	Copy Cue \$from Thru \$to At Sequence \$"backup" Cue \$from /status=false /cueonly=false	FO
9	Copy Cue \$from At Sequence \$"backup" Cue \$from /merge /status /cueonly=false	FO
10	Call Sequence \$"backup" Cue 0.001	FO
11	Delete Sequence \$"backup" Cue 0.001 /nc	FO
New		

LIGNE 1 : SetUserVar \$backup = "QBackup "

Cette variable contient le nom qui va servir à nommer la séquence. Le nom de la séquence sera : **Qbackup**

LIGNE 2 : AddUserVar \$backup = "\$Date"

La fonction **AddUserVar** permet d'étendre la variable \$backup avec la date du jour. Le nom de la séquence sera par ex : **Qbackup 10.8.2014**

LIGNE 3 : AddUserVar \$backup = " "

La fonction **AddUserVar** permet d'étendre la variable \$backup avec un espace.

LIGNE 4 : AddUserVar \$backup = "\$Time "

La fonction **AddUserVar** permet d'étendre la variable \$backup avec l'heure de sauvegarde. Le nom de la séquence sera par ex : **Qbackup 10.8.2014 13h43m20.421s**

LIGNE 5 : Store Sequence \$"backup" cue 0.001

Une séquence vide nommée avec la variable \$backup est sauvegardée à la cue 0.001 qui servira de cue tampon.

LIGNE 6 : SetUserVar \$from = ("backup de la cue :")

Une variable nommée \$from est déclarée et ouvre un Popup permettant de définir le numéro de la première cue.

LIGNE 7 : SetUserVar \$to = ("backup jusqu'à la cue :")

Une variable nommée \$to est déclarée et ouvre un Popup permettant de définir le numéro de la dernière cue. Pour une seule cue à sauvegarder il suffira de saisir le même numéro de cue que la variable \$from de la ligne n°6 ou n'importe quel numéro supérieur pour sauvegarder une plage de cues.

LIGNE 8 : Copy Cue \$from Thru \$to At Sequence \$"backup" Cue \$from /status=false /cueonly=false

La cue ou la plage de cues définies par \$from et \$to sont copiées dans la séquence créée par la ligne n°5 à la dernière cue définie par \$from avec les options de sauvegarde Status (C'est à dire fusionner la cue avec le programmeur à false donc ici non pris en compte) et Cueonly à false également (c'est à dire en conservant les valeurs de tracking).

LIGNE 9 : Copy Cue \$from At Sequence \$"backup" Cue \$from /merge /status /cueonly=false

Une nouvelle copie de la dernière cue de la plage (\$from) est faite dans la séquence \$"backup" à la cue définie par \$from avec à nouveau les options de sauvegarde à commencer par merge. Status est cette fois ci pris en compte.

LIGNE 10 : Call Sequence \$"backup" cue 0.001

Rappelle la cue tampon 0.001 de la séquence \$"backup"

LIGNE 11 : Delete Sequence \$"backup" cue 0.001 /nc

Supprime la cue tampon 0.001 de la séquence \$"backup" devenue inutile. /nc veut dire No Confirm c'est à dire que la cue sera supprimée sans ouverture de fenêtre de confirmation.

GMA 2 - LES MACROS - APPRENTISSAGE

MACRO ARCHIVES QBACKUP

Crée par **Andréas GLAD** et disponible sur **MA-SHARE** également cette seconde perle de macro permet d'exporter toutes les séquences qui ont été créées avec la macro précédente **QBACKUP** et de les supprimer ensuite de la palette **SEQUENCE**.

VOICI LA MACRO ORIGINALE

Edit Macro 26 'ARCHIVES QBACKUP'			
No.	Cmd	Wait	Info
1	SetUserVar \$all="Qbackup*"	FOLLOW	namematch for all Qbackups
2	SetUserVar \$today="Qbackup "	FOLLOW	namematch for todays backu
3	AddUserVar \$today="\$date"	FOLLOW	by adding current date
4	AddUserVar \$today=" *"	FOLLOW	and wildcard for time
5	Export Sequence \$"all" - \$"today" ("Enter filename")	FOLLOW	Export all qbackups apart to
6	Delete Sequence \$"all" - \$"today"	FOLLOW	Delete exported qbackups
New			

Bizzarement elle ne fonctionne pas comme je pensais à savoir exporter toutes les séquences créées du jour avec la macro précédente **QBACKUP**.

J'ai donc fait une petite adaptation.

La macro suivante donc permet d'exporter toutes les séquences à la date du jour et de supprimer ensuite celles-ci uniquement.

Le fichier est stocké dans le dossier **IMPORT / EXPORT**.

Edit Macro 70 'Archive old Qbackups'				
No.	Cmd	Wait	Info	Disab
1	SetUserVar \$today="Qbackup "	FOLLOW	namematch for todays back	
2	AddUserVar \$today="\$date"	FOLLOW	by adding current date	
3	AddUserVar \$today=" *"	FOLLOW	and wildcard for time	
4	Export Sequence \$"today" ("Enter filename")	FOLLOW	Export all qbackups apart to	
5	Delete Sequence \$"today"	FOLLOW	Delete exported qbackups	
New				

LIGNE 1 : SetUserVar \$today = "Qbackup "

La variable \$today est déclarée et contient le nom "Qbackup "

LIGNE 2 : AddUserVar \$today = "\$Date"

La variable \$today est étendue avec AddUserVar et contient en plus du nom "Qbackup " la date du jour.

LIGNE 3 : AddUserVar \$today = " *"

La variable \$today est à nouveau étendue avec AddUserVar pour y rajouter un espace suivi de l'astérisque. Cela permet de bien sélectionner les séquences Qbackup qui contiennent en plus du nom et de la date, l'heure de création.

LIGNE 4 : Export sequence \$"today" ("Enter filename")

Exporte toutes les séquences de la variable \$today dont le nom commence par "Qbackup*" et contient la date du jour.

LIGNE 6 : Delete sequence \$"today"

Supprime toutes les séquences du jour. Cette ligne est facultative et peut être supprimées ou désactivée.

CONCLUSION

J'espère que ce tutorial vous a été utile pour comprendre la construction de macro.

Il existe également un enregistreur de macro dans **GMA2** qui enregistrera dans une macro la plupart des actions que vous faites en temps réel. Cel est une très bonne méthode d'apprentissage de création de macro.

Pour utiliser l'enregistreur, cliquez sur le bouton **MA + STORE** (qui indique **"RECORD"** en ligne de commande) puis cliquez sur une case vide de la palette **MACRO**.

Les touches **LEARN** et **MACRO** clignotent alternativement pour indiquer l'enregistrement en cours.

A partir de maintenant toutes les actions effectuées et qui sont enregistrables dans une macro, seront enregistrées en temps réel.

Pour arrêter l'enregistreur, sélectionner à nouveau **MA + STORE** et cliquer sur la case vide de la macro du départ devenue maintenant une macro.