ETAS

DRIVING EMBEDDED EXCELLENCE

INCA-MIP V7.3

Manuel Utilisateur

www.etas.com

Copyright

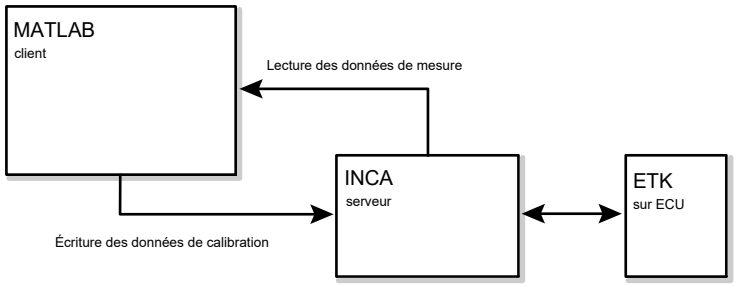
INCA-MIP V7.3 - Manuel Utilisateur R02 FR - 06.2020

Sommaire

1 Introduction

L'AddOn INCA-MIP (INCA MATLAB Integration Package) est une interface de programmation qui permet de contrôler les fonctionnalités de INCA depuis MATLAB. Dans ce cas, MATLABse comporte comme un client accédant aux ressources de INCA, qui est donc le serveur.

Le schéma suivant présente une application typique de l'AddOn INCA-MIP en utilisant INCA pour accéder à l'ETK à partir de MATLAB.



La liste suivante décrit les fonctionnalités d'INCA qui peuvent être utilisées à partir de MATLAB.

gestionnaire de pages de mémoire

Il est possible de passer d'une page mémoire à une autre et de télécharger les pages mémoire vers le calculateur.

Calibration

Toutes les variables de calibrations d'une expérimentation INCA peuvent être modifiées. Les valeurs peuvent être lues et mises à jour pour chaque élément et pour les distributions associées de points d'appui, lorsque cela est applicable.

Measuring

Toutes les variables de mesure d'une expérimentation de INCA peuvent être lues. De plus, des mesures peuvent être lancées et arrêtées à partir de MATLAB. Toutes les données de performance disponibles avec INCA sont accessibles depuis MATLAB. La capacité de traitement des données au niveau de l'interface INCA-MIP est optimisée.

Puisque INCA génère des variables de mesure et de calibration du type double, les formules de conversion pour les retraiter dans MATLAB ne sont pas nécessaires.

Les fonctions API de INCA décrites dans ce document sont appelées à partir des scripts de MATLAB (fichiers M), et peuvent être utilisés pour définir l'ensemble du flux de contrôle des expérimentations INCA.

1.1 Consignes de sécurité

AVERTISSEMENT

Les activités de calibration influencent le comportement d'un ECU et des systèmes, qui sont influencés par l'ECU. Le résultat de cette activité peut conduire à un comportement inattendu du véhicule et entraîner ainsi des situations importantes d'un point de vue de la sécurité.
Seul un personnel spécialisé et chevronné est autorisé à exécuter des tâches de calibration.

1.2 A propos de ce manuel

Les sections suivantes décrivent le principe de base de l'architecture des APIs d'INCA-MIP, les fonctions APIs disponibles ainsi que la procédure d'installation. Les opérations sous MATLAB ou sous INCA ne font pas partie de ce manuel.

Pour utiliser les APIs d'INCA-MIP, vous devrez être familier avec INCA et MATLAB. Et également avec les scripts dans MATLAB.

1.3 Conventions typographiques

Ce manuel utilise les conventions suivantes pour décrire les fonctions APIs en plus des conventions typographiques utilisées dans le Manuel Utilisateur INCA (INCA User Manual).

Le nom de la fonction API est écrit dans une police de caractère non proportionnelle. Les parenthèses sont utilisées pour les arguments des fonctions; les accolades pour les arguments d'entrée optionnels.

Exemple: La lecture d'une variable de calibration

```
value = IncaGetCalibrationValue(deviceName, cali-
brationName {, start, size} {, valueType})
```

Arguments de sortie:

`value`	la valeur actuelle de la variable de calibration; qui doit correspondre avec les types de données indiqués ci-dessous: • Scalaires: une matrice (1,1) • Courbes: une matrice (x,1) • Cartographies: une matrice (x,y) • Distribution de point d'appui: une matrice (x,1)

Arguments d'entrée:

`deviceName`	Nom du dispositif
`calibrationName`	Nom d'élément de calibration
`valueType`	Sélection de l'argument de sortie (chaîne de caractères). La fonction renvoie soit la valeur de la variable de calibration (mode par défaut), soit la distribution du point d'appui en X ou Y. Paramètres possibles : • `v`: la valeur • `x`: point d'appui en x (courbe et cartographies) • `y`: renvoie le point d'appui en Y (cartographies)

Les valeurs `v`, `x` et `y` avec `une police de caractère non proportionnelle` représentent les valeurs possibles qui peuvent être générées. Ces valeurs doivent être entourées avec "' '".

Exemple:

```
yMap = IncaGetCalibrationValue('anEtk', 'Map', 'y');
```

1.4 Glossaire INCA

La description des APIs utilise certains termes qu'un utilisateur expérimenté de INCA connaît. Ci-dessous se trouve une brève définition de ces termes.

Variable de calibration

Une variable de calibration est un élément qui peut être lu et modifié. Les variables de calibration peuvent être des scalaires, des vecteurs, des matrices, des courbes ou des cartographies. Les distributions associées de points d'appui sont également lisibles et modifiables.

Enregistrement des données

Un enregistrement est composé d'une base de temps et de toutes les valeurs de mesure d'une fréquence d'échantillonnage pour une acquisition simple. Les données d'enregistrement pour une fréquence d'échantillonnage est composé de plusieurs enregistrements qui sont générés durant la procédure d'enregistrement globale.

Module

Un module de mesure utilisé pour extraire les variables de mesure d'une grille de mesure particulière. Certains modules de mesure supporte également la calibration de variables. Par exemple, les modules SMB sont utilisables pour les mesures uniquement, alors que les ETK conviennent à la fois aux mesures et aux calibrations.

Données de mesure

Tous les enregistrements capturés d'une mesure pour différentes fréquences d'échantillonnage.

Raster de mesure

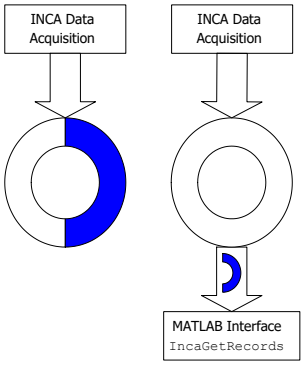
Taux d'acquisition (fréquence de mesure) utilisé pour mesurer un ou plusieurs signaux dans un groupe de signaux.

Il est possible de combiner deux rasters ou plus dans un soi-distant raster multiple. Il suffit de combiner les noms de raster au moyen d'un caractère « + », p. ex. '`10 ms+100 ms`'. L'utilisation d'un tel raster multiple crée un nouveau raster virtuel. Chaque signal peut uniquement être mesuré dans exactement un raster ou un raster multiple.

Buffer tournant

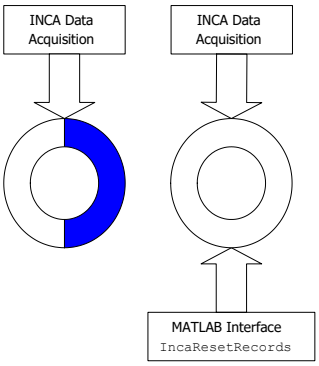
Pour assurer un transfert fiable des données de mesure d'INCA vers MATLAB, un buffer tournant dédié est utilisé pour chaque fréquence d'échantillonnage (groupe de signaux). Pendant une visualisation online d'INCA ou un enregistrement, les variables de mesure acquises sont automatiquement sauvegardées dans le buffer tournant.

La commande `IncaGetRecords` peut être utilisée pour lire les échantillons de temps et les données du buffer tournant vers MATLAB:



Le buffer tournant est limité à 1 Mo par signal et peut enregistrer les données de mesure pendant environ 8 secondes en fonction de la fréquence de mesure. Après ce temps de conservation les informations seront écrasées. Pour éviter de perdre des données la commande `IncaGetRecords` doit être exécutée régulièrement. Typiquement cela doit être fait toutes les secondes.

Avec la commande `IncaResetRecords` la base de temps et les données contenues dans le buffer tournant pour toutes les fréquences d'échantillonnage peuvent être effacées. Toutes les données déjà sauvegardées dans le buffer tournant seront perdues:



Signal

Un signal est un élément dont la valeur est mesurée dans INCA. Chaque signal est caractérisé par son type de données (booléen, entier, flottant), sa longueur (1, 2, 4 ou 8 octets) et sa formule de conversion. La conversion à partir de la valeur de mesure physique sur le niveau d'implémentation est spécifiée dans la formule de conversion.

Fréquence d'échantillonnage

Groupe de signaux; une fréquence d'échantillonnage est composée de plusieurs signaux individuels. Elle est caractérisée par sa fréquence d'acquisition (fréquence de mesure) qui est la même pour tous les signaux du groupe. Chaque groupe de signaux a un nom unique.

2 Installation de l'AddOn INCA-MIP

INCA-MIP est une extension fonctionnelle de INCA.

MATLAB utilise des appels de fonctions liés dynamiquement, appelés fichiers MEX, pour communiquer avec d'autres applications. INCA-MIP est un ensemble de fichiers MEX qui sont copiés dans les sous-répertoires associés au répertoire du programme MATLAB lors de l'installation.

INCA-MIP est disponible en deux versions. L'AddOn de base : « INCA-MIP Base API » est disponible directement après l'installation. Pour utiliser la version évoluée « INCA-MIP Extended API set » une licence sous forme de clef est requise. Une liste des APIs et des descriptions respectives peut être trouvé dans "Fonctions API" sur la page 15.

2.1 Minimum requis

Pour utiliser l'AddOn INCA-MIP, INCA doit être installé sur votre ordinateur. Pour plus d'informations sur les exigences du système INCA, consultez le guide d'installation INCA.

Si vous souhaitez développer vous-même des scripts MATLAB pour accéder à INCA, vous aurez besoin d'une licence complète de MATLAB.

INCA-MIP pour INCA V7.3 nécessite les versions de programme suivantes :

- INCA V7.3 SPx

  Note

  INCA V7.3 est requis pour l'installation de cette version INCA-MIP.

  Assurez-vous que le numéro de version INCA de l'installation INCA est compatible avec le numéro de version du module complémentaire INCA-MIP.

  Après l'installation, vous pouvez utiliser cette version INCA-MIP pour travailler avec toute version INCAV7.x ."Ouvrir INCA" sur la page 26

- Pour MATLAB 64 bits, version 2015b ou supérieure (pour une installation MATLAB intégrée).

Pour plus d'informations sur les versions de MATLAB prises en charge, contactez votre support INCA.

2.2 Istaller INCA-MIP

Avant d'installer l'AddOn il est nécessaire de connaître le type de l'installation. Les types suivants sont possible :

- **Installation en mode MATLAB intégré (MATLAB integrated installation)**
  Sélectionnez cette option si vous utilisez une version de MATLAB pour

développer des scripts MATLAB.

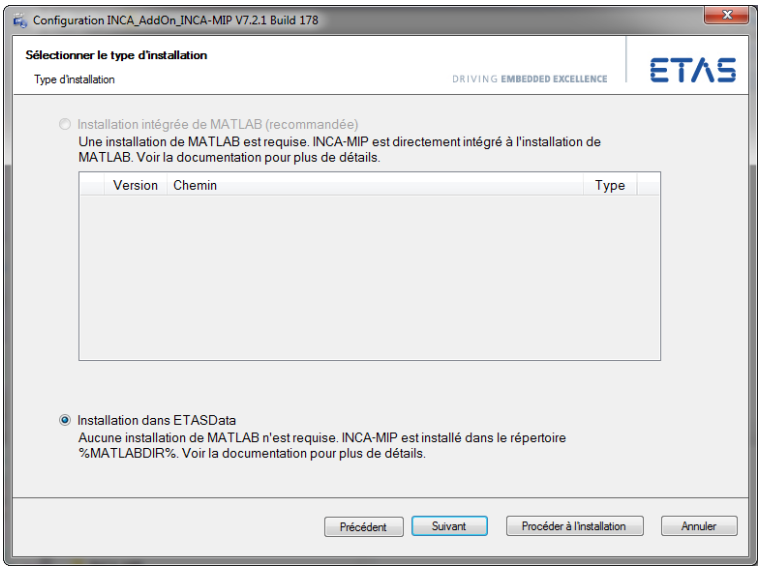
- **Installation dans ETASData**
  Sélectionnez l'option runtime si vous souhaitez uniquement lancer des scripts compilé de MATLAB ou si vous souhaitez utiliser INCA-MIP avec différentes versions de MATLAB sur votre PC. Pour plus de détails, voir ci-dessous.

Pour installer INCA-MIP :

Vérifiez que INCA est installé sur votre PC et que la version d'INCA est compatible avec la version de l'AddOn INCA-MIP.

Si vous souhaitez développer vos propres scripts MATLAB pour accéder à INCA, vérifier que MATLAB est installé sur votre ordinateur et que la version de MATLAB est compatible avec celle de l'AddOn INCA-MIP.

1. Fermez tous les programmes actifs.
2. Selon la réglementation propre à votre entreprise, les fichiers d'installation sont fournis sur un lecteur réseau ou sur un DVD. En utilisant le DVD, la routine d'installation démarre automatiquement. Si ce n'est pas le cas, exécutez le fichier `Autostart.exe` file on the DVD manually. If you install the program from a network drive, also execute the `Autostart.exe`.
3. Cliquez sur **Main**.
4. Sélectionnez l'installation de INCA-MIP.
5. Suivez les instructions de la routine d'installation pour installer INCA-MIP sur votre ordinateur.
6. Dans le programme d'installation, vous êtes prié de préciser le type d'installation souhaitée :



7. Si vous souhaitez développer des scripts MATLAB avec une seule

version MATLAB installée sur votre ordinateur, séléctionnez l'option **MATLAB Integrated installation**.

ou

Sélectionnez l'option **Installation into ETASData** pour les cas suivant :

- Vous souhaitez utiliser INCA-MIP avec différentes versions de MATLAB.

  Dans ce cas vous devez ajouter le sous-répertoire d'INCA-MIP dans le répertoire toolbox MATLAB pour chaque installation MATLAB avant de pouvoir utiliser les commandes INCA-MIP. Consulter votre documentation utilisateur de MATLAB pour savoir comment ajouter des chemins d'accès supplémentaires à MATLAB.

- Vous souhaitez utiliser seulement des fichiers programme disponibles en général qui ont été créés avec MATLAB R13.MATLAB

> **Note**
>
> L'installation dans ETASData est nécessaire si vous utilisez des fichiers programme qui contiennent des instructions MATLAB pour la commande d'INCA. Dans ce cas vous n'avez pas besoin de licence MATLAB. Les fichiers exécutables doivent être mise à disposition par les développeurs avec une installation "Création et distribution de fichiers exécutables avec le compilateur MATLAB R13" sur la page 70 (see MATLAB).

8. Continuez l'installation.

Obtenir la licence pour INCA-MIP :

Si vous projetez d'utiliser la version évoluée des fonctions APIs « INCA MATLAB Extended API set », un fichier de licence sera requis.

Pour plus d'informations sur les licences, voir "Octroi des licences du logiciel" sur la page suivante.

2.3 Mettre à jour les « liens Cache » de la boîte à outils MATLAB

Après installation de l'API INCA-MIP, vous devez d'abord mettre à jour le cache pour les répertoires de la boîte à outils MATLAB au cas où ce cache a été activé lors de votre installation de MATLAB. C'est le cas avec la version MATLAB V6 ou plus si vous utilisez les paramètres par défaut ; les liens sont désactivés dans

les versions antérieures à la V6. Les liens Cache doivent être mis à jour pour que les fichiers utilisés dans l'AddOn INCA-MIP API soient enregistrés dans MATLAB.

Veuillez consulter la documentation utilisateur de MATLAB pour mettre à jour les liens Cache des répertoires de la boîte à outils MATLAB.

2.4 Désactiver les liens Caches des répertoires de boîte à outils MATLAB

Si vous travaillez avec l'AddOn INCA-MIP, il est recommandé de désactiver les « liens Cache » des répertoires de la boîte à outils MATLAB. Sinon, des dysfonctionnements peuvent survenir si l'AddOn INCA-MIP ou des scripts indépendant nouvellement intégrés se peuvent pas être trouvés.

Une autre solution que la désactivation des « liens Caches » consiste à forcer la mise à jour des liens Cache s'ils sont activés comme indiqué précédemment. Cependant pour éviter toute mauvaise manipulation nous vous recommandons de désactiver les « liens Caches » lors de l'utilisation de l'AddOn INCA-MIP.

Veuillez consulter la documentation utilisateur de MATLAB pour activer ou désactiver les liens Cache des répertoires de la boîte à outils MATLAB.

2.5 Octroi des licences du logiciel

Pour pouvoir utiliser INCA, une licence en cours de validité est nécessaire. Le fichier de licence nécessaire peut être obtenu soit auprès de votre coordinateur outils ou sur le portail self service du site Internet ETAS sous http://www.etas.com/support/licensing. Pour demander le fichier de licence, vous devez entrer le numéro d'activation, fourni par ETAS lors de votre commande.

Dans le menu Démarrer de Windows, sélectionnez

E → **ETAS** → **ETAS License Manager**.

Suivez les instructions données dans la boîte de dialogue. Pour de plus amples informations à propos, par ex. des modèles de licence ETAS et de l'emprunt d'une licence, appuyez sur **F1** dans le Gestionnaire des licences ETAS.

3 Fonctions API

L'AddOn INCA-MIP fourni une liste de fonctions pour automatiser le fonctionnement de INCA. Quelques fonctions sont disponibles dans la version de base de l'AddOn INCA-MIP, d'autres ne sont disponibles qu'en achetant la version évoluée de cet AddOn INCA-MIP.

Note

Les commandes qui ne sont disponibles qu'avec la version évolué de INCA-MIP sont protégées par une licence. Vous ne pourrez pas utiliser les fonctions de la version évoluée sans licence valide, l'exécution de ces scripts MATLAB générera une exception.
Nous vous recommandons, avant d'utiliser les commandes de la version évoluée de l'AddOn INCA-MIP, de vérifier la validité de la licence à l'aide de la commande `IncaIsLicenseValid`.

Le tableau suivant liste toutes les commandes de l'AddOn INCA-MIP qui sont disponibles avec [[[Undefined variable FM_import.Product_Version]]]. Il indique

- Si la fonction est disponible dans la version de base de l'AddOn (INCA-MIP Base Package) ou uniquement dans sa version évoluée (INCA-MIP Extended package);
- Si la fonction est utilisée pour l'initialisation, la mesure, la calibration, la gestion des pages mémoire ou s'il s'agit d'une fonction plus générale;
- L'endroit dans le document où vous pourrez trouver plus d'informations concernant la fonction.

Fonction	Base[a]	Evo.[b]	Catégorie	page
IncaAddCalibrationElement	x	x	Calibration	sur la page 56
IncaAddMeasureElement	x	x	Mesure	sur la page 35
IncaBrowseCalibrationElements		x	Calibration	sur la page 55
IncaBrowseItemsInFolder		x	Initialisation	sur la page 29
IncaBrowseMeasureElements		x	Mesure	sur la page 33
IncaClose		x	Initialisation	sur la page 26

IncaCopyPageFromTo	x	x	Gestion Page Mémoire	sur la page 66
IncaDatabaseImport		x	Initialisation	sur la page 28
IncaDownloadDifferences	x	x	Gestion Page Mémoire	sur la page 67
IncaDownloadPage	x	x	Gestion Page Mémoire	sur la page 66
IncaExecuteManualTrigger		x	Mesure	sur la page 53
IncaGetCalibrationValue	x	x	Calibration	sur la page 57
IncaGetCurrentPage		x	Gestion Page Mémoire	sur la page 65
IncaGetDatasetsForDevice		x	Calibration	sur la page 62
IncaGetDeviceProperties		x	Initialisation	sur la page 32
IncaGetDevices		x	Initialisation	sur la page 32
IncaGetHardwareStatus		x	Mesure	sur la page 49
IncaGetInstalledAddOnInfo	x	x	General	sur la page 24
IncaGetInstalledProductInfo	x	x	General	sur la page 23
IncaGetMeasureRatesForDevice		x	Mesure	sur la page 34
IncaGetProperties		x	General	sur la page 25
IncaGetRecordingMode		x	Mesure	sur la page 42
IncaGetRecordingProperties		x	Mesure	sur la page 37
IncaGetRecordingState		x	Mesure	sur la page 54
IncaGetRecords	x	x	Mesure	sur la page 46

IncaGetRecordStruct		x	Mesure	sur la page 54
IncaGetVersion	x	x	General	sur la page 25
IncaGroupDevices		x	Calibration	sur la page 64
IncaIsLicenseValid		x	General	sur la page 23
IncaMessageIds	x	x	General	sur la page 20
IncaIsPageWriteProtected	x	x	Gestion Page Mémoire	sur la page 65
IncaOpen	x	x	Initialisation	sur la page 26
IncaOpenDatabase	x	x	Initialisation	sur la page 27
IncaOpenExperiment	x	x	Initialisation	sur la page 30
IncaResetExperiment	x	x	Initialisation	sur la page 31
IncaResetRecords	x	x	Mesure	sur la page 49
IncaSetCalibrationMode		x	Calibration	sur la page 63
IncaSetCalibrationValue	x	x	Calibration	sur la page 59
IncaSetDatasetInDevice		x	Calibration	sur la page 62
IncaSetMeasureReadMode	x	x	Mesure	sur la page 45
IncaSetProjectAndDatasetInDevice		x	Initialisation	sur la page 30
IncaSetRecordingMode		x	Mesure	sur la page 43
IncaSetRecordingProperties		x	Mesure	sur la page 40
IncaSetTrigger		x	Mesure	sur la page 50

IncaShowMessages	x	x	General	sur la page 22
IncaStartMeasurement	x	x	Mesure	sur la page 37
IncaStartRecording	x	x	Mesure	sur la page 44
IncaStopMeasurement	x	x	Mesure	sur la page 37
IncaStopRecording	x	x	Mesure	sur la page 45
IncaSwitchPage	x	x	Gestion Page Mémoire	sur la page 65
IncaUploadPages		x	Gestion Page Mémoire	sur la page 67
IncaWriteToFile		x	Calibration	sur la page 64

a Fonctions supportées par la version de base de l'AddOn ("INCA-MIP Base Package")

b Fonctions supportée par la version évoluée de l'AddOn ("INCA-MIP Extended Package")

Par la suite, la description des fonctions est organisée en fonction de leurs champs d'application:

- "Fonctions Générales" sur la page 20
- "Initialisation" sur la page 26
- "Mesure et Enregistrement" sur la page 33
- "Calibration" sur la page 54
- "Gestion des pages mémoires" sur la page 64

De plus, un certain nombre de fichiers d'exemples fournis avec la version de développement de l'AddOn INCA-MIP sont décris au "Connaître l'AddOn INCA-MIP grâce à des exemples:" sur la page suivante.

Des exemples sont donnés dans "Exemples" sur la page 68.

Note

L'interface INCA-MIP fonctionne toujours à partir des paramètres globaux des Options Utilisateur de INCA. Pour plus d'information sur les Options Utilisateur de INCA veuillez consulter la documentation INCA.

3.1 Connaître l'AddOn INCA-MIP grâce à des exemples:

L'AddOn INCA-MIP est livré avec des exemples. Dans le cas où vous sélectionnez l'installation de la version évolué, ces fichiers d'exemples types sont automatiquement installés sur votre ordinateur en complémant des fichiers MEX. Les fichiers types utilise plusieurs exemples pour expliquer l'utilisation de l'AddOn INCA-MIP.

Les exemples incluent certains fichiers M qui utilisent l'AddOn INCA-MIP, ainsi qu'une base de données INCA où les éléments utilisés dans les fichiers scripts ont déjà été crées.

Les exemples sont copiés pendant l'installation dans les répertoires suivant (voir "Istaller INCA-MIP" sur la page 11):

- Pour une installation en mode Installation into MATLAB:

  Fichiers M: `%MatlabDir%\toolbox\matlab\demos`
- Pour une installation en mode Installation into ETASData:

  Base de données INCA: `%EtasDataDir%\INCA-MIPx64`

  Base de données démo INCA: `%EtasDataDir%\Database\db_matlabtest`

Pour utiliser les fichiers exemples, vous devez d'abord lancer INCA et ouvrir la base de données fournie. Il n'est pas nécessaire d'avoir du matériel.

Les fonctions des fichiers M sont décrites ci-dessous.

- `tOpen.m` – établit une connexion entre INCA et MATLAB. Cette fonction doit être utilisée au démarrage de chaque session MATLAB avant l'utilisation d'une autre fonction de l'API INCA-MIP.
- `tDummy.m` – ouvre une expérimentation INCA vide en utilisant une configuration matérielle ayant un module de test VADI. Le script crée plusieurs variables de mesure dans l'environnement INCA.
- `tEtkDummy.m` – ouvre une expérimentation INCA vide en utilisant une configuration matérielle ayant un module de test ETK. Le script crée plusieurs variables de mesure et de calibration dans l'environnement INCA. Il reçoit également les pages de travail et de référence, il lit les variables de mesure et de calibration et modifie les valeurs des variables de calibration indépendantes.
- `tGetRecords(aGroupName).m` – rassemble les données de mesure du groupe `aGroupName` pendant 20 secondes puis les transmet à MATLAB. Cette fonction est utilisable avec les exemples VADI et ETK. (pour les "fréquences d'échantillonnage," voir "Glossaire INCA" sur la page 8.)

- `tPrintDB ({aFolder{, aFileId}}).m` – Ecrit tout le contenu de la base de données commençant par le répertoire `aFolder` vers un fichier `aFileId`. Si cette fonction est utilisée sans paramètre, toute l'arborescence de la base de données est imprimée par la sortie standard.
- `tHWStatus.m` – Exemple de l'utilisation de la fonction API `IncaGetHardwareStatus`. MATLAB se connecte à une expérimentation déjà ouverte et choisit le premier élément de mesure trouvé dans le premier dispositif de mesure trouvé. Il continue une mesure de 5 minutes. S'il y a un avertissement ou une erreur pendant la mesure le cycle de mesure est abandonné puis remis en marche après un délais de 5 secondes.

3.2 Fonctions Générales

Les fonctions API générales suivantes sont disponibles:

3.2.1 Listes les messages de l'AddOn INCA-MIP

Les commandes de l'AddOn INCA-MIP peuvent retourner des erreurs.

En utilisant les blocs try/catch, une information détaillée de l'erreur est retournée.

Exemple

```
try,
      <command_1>
      ...
      <command_n>
   catch,
      [msgstr,msgid] = lasterr
      ...
   end
```

avec:

`msgstr` Une phrase de description

`msgid` L'identificateur de message. Les identificateurs de message suivant sont disponibles:

- `INCA:ParameterError`
- `INCA:ReturnParameterError`
- `INCA:WrongParameterValue`
- `INCA:WrongParameterType`
- `INCA:NaN`
- `INCA:ExecutionError`
- `INCA:ResourceError`
- `INCA:RasterFull`
- `INCA:ObjectIsWriteProtected`
- `INCA:CallSeqenceError`
- `INCA:LicenseError`
- `INCA:RecordingInProcess`
- `INCA:NotInstalled`
- `INCA:WrongVersion`

Description des identificateurs de message:

`INCA:ParameterError`	Mauvais nombre de paramètres d'entrée passé dans la fonction (Paramètres de droite)
`INCA:ReturnParameterError`	Mauvais nombre de paramètre de sortie renvoyé par la fonction (paramètre de gauche)
`INCA:WrongParameterValue`	Une des valeurs des paramètres d'entrée de la fonction est en dehors de sa plage valide ou de sa spécification
`INCA:WrongParameterType`	Un des paramètre d'entrée de la fonction n'est pas du bon type
`INCA:NaN`	Un des paramètre contient une valeur du type 'not a number' (pas un nombre)
`INCA:ExecutionError`	Durant l'exécution de la commande une erreur a eu lieu. Essayer d'exécuter la fonctionnalité à partir de INCA peut donner plus d'informations sur l'erreur. Redémarrer INCA ou l'ordinateur peut aussi aider.
`INCA:ResourceError`	Impossible d'obtenir des ressources du système d'exploitation. Redémarrer INCA ou l'ordinateur peut aider.

`INCA:RasterFull`	La liste d'acquisition est pleine pour cette fréquence d'échantillonnage lors de l'ajout d'une mesure
`INCA:ObjectIsWriteProtected`	Impossible de calibrer à cause d'une protection en écriture de l'élément
`INCA:CallSeqenceError`	Avant d'exécuter cette commande d'autres commandes doivent être exécutées en premier. Par Exemple `IncaOpenExperiment` est nécessaire avant `IncaAddMeasureElement`.
`INCA:LicenseError`	Pour exécuter la commande avec les paramètres donnés, une licence est requise.
`INCA:RecordingInProcess`	Impossible d'exécuter la commande demandée (p. ex. activer ou désactiver les signaux pour l'enregistrement avec `SetRecordingMode`) parce qu'un enregistrement est en cours.
`INCA:NotInstalled`	Impossible d'ouvrir la version INCA spécifiée avec la commande `IncaOpen` parce que la version INCA correspondante n'est pas installée.
`INCA:WrongVersion`	Impossible d'ouvrir la version INCA spécifiée avec la commande `IncaOpen` pour l'une des raisons suivantes : • INCA est déjà démarré et une commande `IncaOpen` est exécutée avec un paramètres de version différent de la version INCA déjà ouverte • Une commande `IncaOpen` est exécutée depuis INCA-MIP pour INCA Vx.y avec un paramètre de version avec version majeure != x

3.2.2 Activer l'affichage d'information durant l'exécution des scripts

Nom `IncaShowMessages`

Description Active ou désactive l'affichage d'information dans la fenêtre MATLAB pendant l'exécution du script.

Syntaxe `IncaShowMessages(VraiOuFaux)`

Arguments de sortie		
Arguments d'entrée	`VraiOuFaux`	Paramètre numérique qui est soit égal à zéro soit différent de zéro. Si le paramètre est zéro, l'affichage de l'information est désactivé, autrement l'affichage est activé (défaut).
Exemples		

3.2.3 Afficher l'état de la licence INCA-MIP (INCA-MIP Evoluée)

`Nom`	`IncaIsLicenseValid`	
`Description`	Renvoi un statut indiquant si la licence INCA-MIP est valide ou non	
`Syntaxe`	`s = IncaIsLicenseValid`	
`Arguments de sortie`	`s`	Statut de la licence • `0` : Pas de licence valide • `1` : Licence valide
`Arguments d'entrée`		
`Exemples`	`statut = IncaIsLicenseValid`	

3.2.4 Lire informations sur toutes les versions INCA installées

Nom	`IncaGetInstalledProductInfo`	
Description	Lire informations sur toutes les versions INCA installées. Cette commande peut être exécutée avant `IncaOpen`.	
Syntaxe	`info = IncaGetInstalledProductInfo`	
Arguments de sortie	`info`	information sur les versions INCA installées en tant que struct. MATLAB pour chaque installation et comprenant les entrées suivantes :
	`info.name`	le nom du produit
	`info.version`	la chaîne de version du produit
	`info.hotfixVersion`	le hotfix installé sous forme de chaîne ou une chaîne vide si aucun hotfix n'est installé
Arguments d'entrée		
Exemples	`i = IncaGetInstalledProductInfo;`	

Note

Cette commande a été introduite avec INCA-MIP V16.0.

Note

Cette commande ne fonctionne de manière fiable que si INCA V7.1 ou supérieur est installé.

3.2.5 Lire Information sur tous les add-ons installés

Nom	IncaGetInstalledAddOnInfo	
Description	Fournit des informations sur tous les add-ons installés pour un produit donné. Cette commande peut être exécutée avant `IncaOpen`.	
Syntaxe	`info = IncaGetInstalledAddOnInfo(productName, productVersion)`	
Arguments de sortie	`info`	information sur les add-ons installés en tant que struct. MATLAB pour chaque installation et comprenant les entrées suivantes :
	`info.name`	le nom de l'add-on installé
	`info.version`	la chaîne de version de l'add-on installé
Arguments d'entrée	`productName`	le nom du produit
	`productVersion`	la version du produit sous forme de chaîne. La chaîne de version complète est pertinente.
Exemples	`i = IncaGetInstalledAddOnInfo('INCA', 'V7.3.0');` `i = IncaGetInstalledAddOnInfo('INCA', 'V7.3.0 Beta 42');`	

Note

Assurez-vous de bien utiliser pour les arguments d'entrée `productName` et `productVersion` exactement le nom et la version d'un produit tel que retourné par IncaGetInstalledProductInfo.

Note

Cette commande a été introduite avec INCA-MIP V16.0.

> **Note**
>
> Cette commande ne fonctionne de manière fiable que si INCA V7.1 ou supérieur est installé.

3.2.6 Connaître la version de INCA

Nom `IncaGetVersion`

Description Cette fonction permet de connaître la version de INCA.

Syntaxe `IncaGetVersion`

Arguments de sortie

Arguments d'entrée

Exemples

3.2.7 Lire les propriétés de INCA (INCA-MIP Evoluée)

Nom `IncaGetProperties`

Description Lit les propriétés de INCA

Syntaxe `p = IncaGetProperties`

Arguments de sortie

`p` Propriétés de INCA affichées en tant que structure MATLAB, constituée des entrées suivantes:

- `p.databasePath` - Chemin d'accès de la base de données INCA ouverte. S'il n'y a pas de base de données INCA ouverte une chaîne de caractère vide est renvoyée.
- `p.dataPath` - Chemin d'accès du répertoire des données de INCA.
- `p.installationPath` - Chemin d'accès du répertoire d'installation de INCA.
- `p.tempPath` - Chemin d'accès du répertoire utilisé par les outils ETAS pour les fichiers temporaires.

Arguments d'entrée

Exemples

```
p = INCAGetProperties;
```

3.3 Initialisation

Toutes les opérations de mesure et de calibration dans INCA sont effectuées dans le cadre d'une expérimentation. Avant d'ouvrir une expérimentation, un espace de travail valide et une configuration matériel valide doivent être créés et assigné.

Pour travailler avec l'AddOn INCA-MIP, une expérimentation vide doit exister dans la base de données INCA pour laquelle un espace de travail valide et une configuration matérielle doivent d'abord être créés et attribués. Puis l'expérimentation peut être ouverte à partir de MATLAB.

Les fonctions API suivantes sont disponibles pour l'initialisation:

3.3.1 Ouvrir INCA

Nom `IncaOpen`

Description Ouvre INCA et initialise la communication entre MATLAB et INCA.

Syntaxe
```
IncaOpen
IncaOpen( version )
```

Arguments de sortie

Arguments d'entrée

`version`	Version INCA à ouvrir (facultatif). Syntaxe : `<MajorVersion>.<MinorVersion>`. INCA-MIP pour INCA x.y peut uniquement se connecter aux installations INCA de la même version majeure x.

Exemples
```
IncaOpen;
IncaOpen('7.3');
```

> **Note**
>
> L'argument optionnel 'version' a été introduit avec INCA-MIP V16.0.

3.3.2 Fermer INCA (INCA-MIP Evoluée)

Nom `IncaClose`

Description Se déconnecte par INCA et ferme INCA (optionnel) après s'être connecté avec succès à INCA avec `IncaOpen`.

Syntaxe
```
IncaClose
IncaClose( isDisconnectOnly )
```

Arguments de sortie		
Arguments d'entrée	`isDisconnectOnly`	Spécifie si MATLAB seulement déconnecte par INCA ou si en plus il ferme INCA (optionnel). Valeurs disponibles: • `0` : Déconnecte par INCA et ferme INCA (défault). • `1` : Déconnecte par INCA et laisse INCA ouvert.
Exemples	`INCAClose ;` `INCAClose(1);`	

Note

Cette commande a été introduite avec INCA-MIP V16.0.

Avec INCA-MIP V16.1, l'argument d'entrée `isDisconnectOnly` a été ajouté. Dans les versions précédentes, aucun argument d'entrée ne fût disponible, et `IncaClose` non seulement ait déconnecté par INCA mais encore ait fermé INCA.

3.3.3 Ouvrir une base de données

Nom	`IncaOpenDatabase`	
Description	Cette fonction ouvre une base de données dans le répertoire spécifié.	
Syntaxe	`IncaOpenDatabase({pathName})`	
Arguments de sortie		
Arguments d'entrée	`pathName`	Le répertoire dans lequel la base de données est ouverte et enregistrée. Si aucun répertoire n'est spécifié, la base de données actuelle est ouverte.
Exemples	`IncaOpenDatabase; % ouvrir la base de données actuelle` `IncaOpenDatabase('c:\etasdata\madatabase');`	

3.3.4 Importer une base de données (INCA-MIP Evoluée)

Nom `IncaDatabaseImport`

Description Cette fonction importe le fichier export d'une base de données (*.exp) dans INCA. Les éléments de la base de données déjà existant seront toujours écrasés.

Syntaxe

```
IncaDatabaseImport(path)
nom = IncaDatabaseImport(path)
[nom, type] = IncaDatabaseImport(path)
```

Arguments de sortie

`nom` — Tableau de tous les chemins des éléments importés

Utilisez la fonction `deblank()` lorsque vous accéder à un tableau:

```
name2 = deblank(nom(2,:))
```

`type` — Tableau de tous les types des éléments importés

Les valeurs possibles sont:

- `Folder`: Un répertoire de base de données
- `Experiment`: Une expérimentation
- `Workspace`: Un espace de travail
- `Asap2Project`: Un projet A2L
- `MeasurementCatalog`: Un catalogue de mesure ASAP2
- `CanDB`: Une base de données CAN ASAP2

Utilisez la fonction `deblank()` lorsque vous accéder à un tableau:

```
type2 = deblank(type(2,:))
```

Arguments d'entrée

`path` — Le chemin complet du fichier *.exp à importer

Exemples

```
noms = IncaDatabaseImport('D:\
ETASData\[[[Undefined variable FM_
import.INCA_Version_Code]]]\ex-
port\Project0815.exp')
```

3.3.5 Lire les éléments de la base de données (INCA-MIP Evoluée)

Nom `IncaBrowseItemsInFolder`

Description Cette fonction permet de lister les éléments de la base de données avec un filtre de recherche

Syntaxe
```
[nom, type] = IncaBrowseItemsInFolder(pattern, folderName)
```

Arguments de sortie

`nom`	Liste des noms des éléments de la base de donneés
`type`	Liste des types des éléments de la base de données: • `Folder`: Répertoire • `Experiment`: Expérimentation • `Workspace`: Espace de travail • `Asap2Project`: Projet A2L ASAP2

Arguments d'entrée

`pattern`	Filtre de recherche à appliquer. Un '`*`' correspond à zéro, un ou plusieurs caractères supplémentaires. Un '`#`' correspond uniquement à un caractère. Tous les autres caractères doivent correspondre avec les éléments de la base de données. Il n'y a pas de différence entre les majuscules et les minuscules.
`folderName`	Répertoire de la base de données où les élément seront lus. L'arborescence des répertoire est séparée par un '`\`'. Un argument vide correspond à l'arborescence la plus haute.

Exemples
```
[n,t]=IncaBrowseItemsInFolder('*',
'DEFAULT\MyProject');
[nom,type]=IncaBrowseItemsInFolder('Prj*_##',
'');
```

3.3.6 Assigner un projet et un jeu de données à un dispositif (INCA-MIP Evoluée)

Nom `IncaSetProjectAndDatasetInDevice`

Description Cette fonction permet d'assigner un projet et un jeu de données à un dispositif dans un espace de travail précis. Cela ne peut pas être fait si une expérimentation est ouverte.

Syntaxe
```
IncaSetProjectAndDatasetInDevice(workspace,
dispositif,projet,dataset)
```

Arguments de sortie

Arguments d'entrée

`workspace`	Chemin d'accès de l'espace de travail
`dispositif`	Nom du dispositif
`projet`	Chemin d'accès du projet
`dataset`	Chemin d'accès du jeu de donnée

Exemples
```
IncaSetProjectAndDatasetInDevice('DEFAULT\-
works-
pace','ETK:1','DEFAULT\Prj0815','Ds4711\Ds4711_
3')
```

3.3.7 Ouvrir une expérimentation

Nom `IncaOpenExperiment`

Description Cette fonction permet d'ouvrir l'expérimentation spécifée. Après l'ouverture de l'expérimentation, vous pouvez utiliser l'AddOn INCA-MIP pour ajouter les variables de mesure et de calibration désirées.

Syntaxe
```
IncaOpenExperiment({closeAllViewsFlag})
```
ou
```
IncaOpenExperiment(expFolderName, expe-
rimentName, workspaceFolderName, works-
paceName {, closeAllViewsFlag})
```

Arguments de sortie

Arguments d'entrée

`expFolderName`	Répertoire dans lequel l'expérimentation est enregistrée
`experimentName`	Nom de l'expérimentation

workspaceFolderName	Répertoire dans lequel l'espace de travail est enregistré
workspaceName	Nom de l'espace de travail
closeAllViewsFlag	Ferme toutes les fenêtres des variables de mesure et de calibration dans l'expérimentation sélectionnée. Paramètre possible : • 1 : Ferme toute les fenêtres (par défaut) • 0 : Laisse les fenêtres inchangées

Exemples

```
IncaOpenExperiment('ExpFolder',
'MyExperiment', 'WorkspaceFolder',
'MyWorkspace');
```

Note

Cette fonction a changé depuis la version INCA V3.0, parce que désormais deux arguments d'entrée sont attendus pour cet espace de travail retouché.

Si l'expérimentation est déjà ouverte lors de l'appel à la fonction `IncaOpenExperiment`, les arguments d'entrée spécifiant l'environnement sont optionnels. Si l'expérimentation n'est pas encore ouverte, vous devez utiliser la fonction `IncaOpenDatabase` avant `IncaOpenExperiment`.

3.3.8 Réinitialiser une expérimentation

Nom `IncaResetExperiment`

Description Réinitialise et ferme l'expérimentation actuelle. Vous pouvez utiliser cette fonction pour retirer toutes les variables d'une expérimentation. Retirer des variables individuellement n'est actuellement pas possible.

Syntaxe `IncaResetExperiment`

Arguments de sortie

Arguments d'entrée

Exemples

Note

Si l'expérimentation a été ouverte manuellement et non une commande MATLAB, `IncaResetExperiment` libère l'expérimentation sans la fermer. Dans ce cas vous devez utiliser la command `IncaOpenExperiment` avant de pouvoir accèder de nouveau à l'expérimentation.

3.3.9 Lire les dispositifs (INCA-MIP Evoluée)

Nom `IncaGetDevices`

Description Cette fonction permet de lire tous les dispositifs de l'expérimentation

Syntaxe `[nom, type] = IncaGetDevices`

Arguments de sortie

`nom`	Liste des noms des dispositifs
`type`	Liste des types des dispositifs: • `WorkbaseDevice`: Dispositif avec jeu de données • `MeasurementDevice`: Dispositif de mesure

Arguments d'entrée

Exemples [nom,type]=IncaGetDevices;

3.3.10 Lire les propriétés du dispositif (INCA-MIP Evoluée)

Nom `IncaGetDeviceProperties`

Description Cette fonction permet de lire les propriétés d'un dispositif

Syntaxe `p = IncaGetDeviceProperties(deviceName)`

Arguments de sortie

`p`	Propriétés du dispositif sous la forme d'une structure MATLAB, avec les entrées suivantes:
`p.nom`	Nom du disposif
`p.descriptionFile`	Chemin d'accès du fichier de description du projet assigné au dispositif. Un texte vide est renvoyé s'il n'y a pas de projet assigné au dispositif.
`p.binaryFile`	Chemin d'accès du fichier de données assigné au dispositif. Un texte vide est renvoyé s'il n'y a pas de projet assigné au dispositif.
`p.projectDBPath`	Chemin d'accès dans la base de données INCA du projet assigné au dispositif. Un texte vide est renvoyé s'il n'y a pas de projet assigné au dispositif.

	`p.isWriteProtected`	• `0` : Dispositif n'ayant pas de page mémoire ou utilisant une page non protégée en écriture • `1` : Page actuelle protégée en écriture
	`p.isActive`	• `0` : Dispositif non connecté ou inactif • `1` : Dispositif connecté et actif
	`p.isWorkbaseDevice`	• `0` : Dispositif sans jeu de données • `1` : Dispositif avec jeu de données
Arguments d'entrée	`deviceName`	Nom du dispositif

Exemples

```
p = IncaGetDeviceProperties('Device');
```

3.4 Mesure et Enregistrement

Un signal ou une variable de mesure est toujours extrait/e comme faisant partie d'une fréquence d'échantillonnage de ce module de mesure uniquement. Chaque variable ne peut être affectée qu'à une fréquence d'échantillonnage. Pour configurer une expérimentation, attribuez d'abord les variables de mesure à une fréquence d'échantillonnage séparée.

> **Note**
>
> Notez que le nom des éléments, dispositifs, signaux ou fréquences d'échantillonnage sont sensibles à la casse.

3.4.1 Lire les éléments de mesure (INCA-MIP Evoluée)

Nom `IncaBrowseMeasureElements`

Description Cette fonction permet d'obtenir les éléments de mesure d'une expérimentation avec un filtre de recherche et éventuellement d'un dispositif

Syntaxe

```
[nom, type] = IncaBrowseMeasureElements (pattern, {deviceName})
[nom] = IncaBrowseMeasureElements(pattern, {deviceName})
```

Arguments de sortie	`nom`	Liste des noms des éléments de mesure

	`type`	Liste des types des éléments de mesure: • `Scalar`: Scalaire • `Array`: Vecteur • `Matrix`: Matrice
Arguments d'entrée	`pattern`	Filtre de recherche à appliquer sur les éléments de mesure. Un '*' correspond à zéro, un ou plusieurs caractères supplémentaires. Un '#' correspond uniquement à un caractère. Tous les autres caractères doivent correspondre avec les éléments de mesure. Il n'y a pas de différence entre les majuscules et les minuscules.
	deviceName	Nom du dispositif
Exemples	`[n,t]=IncaBrowseMeasureElements('ign*', 'Device');` `[nom,type]=IncaBrowseMeasureElements('*');`	

3.4.2 Lecture des fréquences d'échantillonnage (INCA-MIP Evoluée)

Nom	`IncaGetMeasureRatesForDevice`	
Description	Cette fonction permet d'obtenir toutes les fréquences d'échantillonnage d'un dispositif	
Syntaxe	`[nom] = IncaGetMeasureRatesForDevice(deviceName)`	
Arguments de sortie	`nom`	Liste des noms des fréquences d'échantillonnage
Arguments d'entrée	`deviceName`	Nom du dispositif
Exemples	`n=IncaGetMeasureRatesForDevice('Device');` `nom=IncaGetMeasureRatesForDevice('Dev');`	

3.4.3 Ajouter une variable de mesure à une fréquence d'échantillonnage

Nom `IncaAddMeasureElement`

Description Ajoute une variable de mesure à une expérimentation avec ou sans raster de mesure donné.

Syntaxe

```
IncaAddMeasureElement(deviceName, groupName,
signalName {, displayMode})
groupName = IncaAddMeasureElement( devi-
ceName, [], signalName{, displayMode})
```

Arguments de sortie

Arguments d'entrée

`deviceName`	Nom du dispositif
`groupName`	nom du raster de mesure Il est possible d'utiliser plusieurs rasters en combinant tout simplement les noms de raster au moyen d'un caractère « + », p. ex. '`10 ms+100 ms`'. L'utilisation d'un tel raster multiple crée un nouveau raster virtuel. Chaque signal peut uniquement être mesuré dans exactement un raster ou un raster multiple. Le nom de groupe peut être [] (voir remarque ci-dessous).
`signalName`	Nom du signal de mesure. Pour les scalaires, le nom est suffisant. Pour les vecteurs et matrices, l'index au format `[n]` ou `[n,m]` doit être accolé au nom. Le premier élément a l'index "zéro".
`displayMode`	Mode d'affichage de l'élément: • `1` : la variable de mesure est affichée (par défaut) • `0` : pas d'affichage

Exemples

```
IncaAddMeasureElement('MyDevice', '10ms',
'Channel01', 0);
IncaAddMeasureElement('ETK:1', '1.0ms',
'Matrix[2,1]');
group = IncaAddMeasureElement('CalcDev', [],
'MyCalcSig1');
```

Note

Si la fréquence d'échantillonnage est pleine, la variable de mesure n'est pas ajoutée à cette fréquence.

Note

Si l'argument d'entrée `groupName` est [] (c'est-à-dire vide), le groupe de signaux sera déterminé de la manière suivante :
- Si le signal fait déjà partie de l'expérimentation, c'est son nom de groupe de signaux existant qui est utilisé.
- Si le signal ne fait pas partie de l'expérimentation, un groupe de signaux disponible est utilisé de manière arbitraire. Dans le cas du dispositif Calculé (DispCalc, utilisé pour les signaux calculés) ou de CAN Monitoring, c'est le groupe de signaux défini pour ce signal qui est utilisé.

Comme le nom du groupe de signaux est requis pour `IncaGetRecords`, `IncaGetRecordStruct` ou `IncaGetRecordCount`, il est retourné en tant que paramètre de gauche en option.

Exemples:

```
groupName = IncaAddMeasureElement( 'DispCalc', [],
'MonSigCalc)
groupName = IncaAddMeasureElement( 'CAN-Monitoring:1',
[], 'nmot', 1)
```

Note

L'utilisation d'une entrée vide pour 'groupName' n'est possible que depuis INCA-MIP V16.0.

Note

Le nombre total de signaux qui peuvent être ajoutés est spécifique à l'appareil ainsi qu'au protocole. Le nombre de signaux est limité par la quantité de mémoire tampon libre allouée par le processus du serveur cible. La taille totale de la mémoire tampon dépend de la fréquence d'échantillonnage utilisée.

Exemple:
Un signal avec une fréquence d'échantillonnage de 0,1 ms nécessite > 3 Mo de données. Par conséquent, le nombre total de signaux qui peuvent être ajoutés se situe entre 400 et 600 signaux. Des fréquences d'échantillonnage plus lentes permettent d'ajouter plus de signaux.

3.4.4 Démarrer la mesure

Nom `IncaStartMeasurement`

Description Démarre la mesure sous INCA

Syntaxe `IncaStartMeasurement`

Arguments de sortie

Arguments d'entrée

Exemples

3.4.5 Arrêter la mesure

Nom `IncaStopMeasurement`

Description Cette fonction arrête la mesure en court et l'enregistrement (s'il y a lieu) sous INCA

Syntaxe `IncaStopMeasurement{(mdfFileName)}`

Arguments de sortie

Arguments d'entrée

`mdfFileName`	Nom du fichier MDF dans lequel les données enregistrées sont sauvegardées si l'enregistrement s'arrête en même temps que la mesure en cours. Il faut toujours spécifier en entier le chemin d'accès au fichier (ex.'`c:\mydata\store1.dat`').

Exemples IncaStopMeasurement('c:\mydata\store1.dat')

> **Note**
>
> Pour éviter de perdre des données du fait des mesures en cours, il faut s'assurer d'arrêter l'enregistrement en cours avec la commande `IncaStopMeasurement(mdfFileName)` si le volume de données est élevé. Ensuite il faut utiliser la commande `IncaGetRecords` pour transférer à MATLAB les données restantes.

3.4.6 Lire les propriétés de l'enregistrement (INCA-MIP Evoluée)

Nom `IncaGetRecordingProperties`

Description Lit les propriétés du fichier de sortie primaire de l'enregistreur par défaut ainsi que l'extension de fichier pour le format d'enregistrement primaire sélectionné.

Syntaxe `properties = IncaGetRecordingProperties`

Arguments de sortie	`properties`	Propriétés de l'enregistrement sous la forme d'une structure MATLAB, avec les entrées suivantes:
	`properties.fileName`	Le nom du fichier d'enregistrement y compris le format du fichier de sortie primaire
	`properties.directory`	Répertoire du fichier d'enregistrement
	`properties.fileFormat`	Format du fichier d'enregistrement; les informations textes suivantes sont possibles: • `ETASBinary` • `#DiademATF` • `ETASAscii` • `ETASGroupAscii` • `ETASMATLABMFILE` • `ETASGroupMatlabM` • `FamosRecord` • `ETASMDF` • `ETASMDF4`
	`properties.autoIncrement`	Incrément automatique du nom du fichier d'enregistrement • `0` : Pas d'incrémentation automatique • `1` : Incrémentation automatique
	`properties.comment`	Commentaire de l'entête du fichier d'enregistrement. Il ne doit pas excédé 1024 caractères moins les caractère utilisés pour les commentaire par défauts.
	`properties.defaultComment`	Commentaire par défaut généré par INCA de l'entête du fichier d'enregistrement
	`properties.company`	Nom de la société dans l'entête du fichier d'enregistrement

`properties.project`	Nom du projet dans l'entête du fichier d'enregistrement
`properties.user`	Nom de l'utilisateur dans l'entête du fichier d'enregistrement
`properties.vehicle`	Nom du véhicule dans l'entête du fichier d'enregistrement

Arguments d'entrée

Exemples

```
properties = IncaGetRecordingProperties;
```

Note

Avec INCA-MIP V16.0, les identificateurs de format pour l'argument de sortie `properties.fileFormat` ont été modifiés:

Avant INCA-MIP V16.0:

- ETAS binary data file format
- Diadem ATF measure format (write only)
- ASCII
- ASCII (Multirate/Write only)
- MATLAB M-File
- MATLAB M-File (Multirate/Write only)
- FAMOS (imc)
- MDF (Measure Data Format)
- MDF4 (Measure Data Format for VSG using 4byte floats only)

Depuis INCA-MIP V16.0:

- ETASBinary
- #DiademATF
- ETASAscii
- ETASGroupAscii
- ETASMATLABMFILE
- ETASGroupMatlabM
- FamosRecord
- ETASMDF
- ETASMDF4

3.4.7 Régler les propriétés de l'enregistrement (INCA-MIP Evoluée)

Nom `IncaSetRecordingProperties`

Description Cette fonction permet de régler les propriétés du prochain enregistrement

Syntaxe `IncaSetRecordingProperties(properties)`

Arguments de sortie

Arguments d'entrée

`properties`	Propriétés de l'enregistrement sous la forme d'une structure MATLAB, avec les entrées suivantes:
`properties.fileName`	Nom du fichier d'enregistrement
`properties.directory`	Répertoire du fichier d'enregistrement
`properties.fileFormat`	Format du fichier d'enregistrement; les informations textes suivantes sont possibles: • `ETASBinary` • `#DiademATF` • `ETASAscii` • `ETASGroupAscii` • `ETASMATLABMFILE` • `ETASGroupMatlabM` • `FamosRecord` • `ETASMDF` • `ETASMDF4`
`properties.autoIncrement`	Incrément automatique du nom du fichier d'enregistrement • `0` : Pas d'incrémentation automatique • `1` : Incrémentation automatique

`properties.comment`	Commentaire de l'entête du fichier d'enregistrement. Il ne doit pas excédé 1024 caractères moins les caractère utilisés pour les commentaire par défauts.
`properties.company`	Nom de la société dans l'entête du fichier d'enregistrement
`properties.project`	Nom du projet dans l'entête du fichier d'enregistrement
`properties.user`	Nom de l'utilisateur dans l'entête du fichier d'enregistrement
`properties.vehicle`	Nom du véhicule dans l'entête du fichier d'enregistrement

Exemples

```
properties.user = 'Michael';
properties.project = 'K70';
IncaSetRecordingProperties(properties);
```

Note

Avec INCA-MIP V16.0, les identificateurs de format pour l'argument de sortie `properties.fileFormat` ont été modifiés:

Avant INCA-MIP V16.0:

- ETAS binary data file format
- Diadem ATF measure format (write only)
- ASCII
- ASCII (Multirate/Write only)
- MATLAB M-File
- MATLAB M-File (Multirate/Write only)
- FAMOS (imc)
- MDF (Measure Data Format)
- MDF4 (Measure Data Format for VSG using 4byte floats only)

Depuis INCA-MIP V16.0:

- ETASBinary
- #DiademATF
- ETASAscii
- ETASGroupAscii
- ETASMATLABMFILE
- ETASGroupMatlabM
- FamosRecord
- ETASMDF
- ETASMDF4

Note

When setting the recording properties with `IncaSetRecordingProperties`, you should not finish a recording with `IncaStopRecording`. Instead `IncaSetTrigger` can be used. Any trigger condition can be used to stop the recording.

Examples:

Stop recording after a constant duration:

```
TIMEDURATION_SECONDS = 25;
IncaSetTrigger('none', 'none', 'none', 'none',
TIMEDURATION_SECONDS);
IncaStartRecording;
% Recording automatically stops after TIMEDURATION_
SECONDS seconds
```

Stop recording after a manual trigger condition:

```
IncaSetTrigger('none', 'manual');
IncaStartRecording;
% Do anything until the stop trigger condition is met
...
IncaExecuteManualTrigger('stop');
```

3.4.8 Mode Lecture Enregistrement (INCA-MIP Evoluée)

Nom	`IncaGetRecordingMode`
Description	Indique si un signal est enregistré ou non dans l'enregistreur par défaut.
Syntaxe	`IncaGetRecordingMode(deviceName, signalName)`
Arguments de sortie	mode d'enregistrement pour l'enregistreur par défaut : • `0` : le signal n'est pas enregistré dans l'enregistreur par défaut • `1` : le signal est enregistré dans l'enregistreur par défaut

Arguments d'entrée	`deviceName`	Nom du dispositif
	`signalName`	Nom du signal de mesure. Pour les scalaires, le nom est suffisant. Pour les vecteurs et matrices, l'index au format `[n]` ou `[n,m]` doit être accolé au nom. Le premier élément a l'index "zéro".

Exemples

```
m = IncaGetRecordingMode('ETK:1', 'hfm');
mode = IncaGetRecordingMode('DispCalc',
'MonSigCalc1');
```

Note

Cette commande a été introduite avec INCA-MIP V16.0.

Note

Avant d'utiliser `IncaSetRecordingMode`, le signal doit être ajouté avec `IncaAddMeasureElement`.

3.4.9 Set Recording Mode (INCA-MIP Evoluée)

Nom `IncaSetRecordingMode`

Description Active ou désactive l'enregistrement d'un signal dans l'enregistreur par défaut. L'enregistrement peut uniquement être désactivé pour des signaux qui sont affichés dans l'expérimentation INCA.
Avant d'exécuter cette commande, le signal doit être ajouté à l'expérimentation avec `IncaAddMeasureSignal`.

Syntaxe

```
IncaSetRecordingMode(deviceName, signalName,
recordingMode)
```

Arguments de sortie

Arguments d'entrée	`deviceName`	Nom du dispositif

`signalName`	Nom du signal de mesure. Pour les scalaires, le nom est suffisant. Pour les vecteurs et matrices, l'index au format `[n]` ou `[n,m]` doit être accolé au nom. Le premier élément a l'index "zéro".
`recordingMode`	mode d'enregistrement pour l'enregistreur par défaut : • `0` : la variable de mesure est supprimée dans l'enregistreur par défaut • `1` : le signal de mesure est ajouté à l'enregistreur par défaut

Exemples

```
IncaSetRecordingMode('ETK:1', 'hfm', 1);
IncaSetRecordingMode('DispCalc',
'MonSigCalc1', 0);
```

Note

Cette commande a été introduite avec INCA-MIP V16.0.

Note

Avant d'utiliser `IncaSetRecordingMode`, le signal doit être ajouté avec `IncaAddMeasureElement`.

3.4.10 Démarre l'enregistrement

Nom `IncaStartRecording`

Description Cette fonction permet de démarrer l'enregistrement dans INCA. Cette fonction peut être utiliser après ou à la place de la fonction `IncaStartMeasurement`.

Une fois une mesure ou un enregistrement lancé/e, les données mesurées sont également disponibles dans MATLAB.

Syntaxe `IncaStartRecording`

Arguments de sortie

Arguments d'entrée

Exemples

3.4.11 Arrêter un enregistrement

Nom `IncaStopRecording`

Description Cette fonction permet d'arrêter l'enregistrement en cours dans INCA. La mesure continue et doit être arrêtée au moyen de la fonction `IncaStopMeasurement`. Il est possible d'activer/désactiver cette fonction plusieurs fois au cours de la mesure.

Syntaxe `IncaStopRecording(mdfFileName)`

Arguments de sortie

Arguments d'entrée

`mdfFileName`	Nom du fichier MDF où les données sont enregistrées. Il faut toujours spécifier le chemin d'accès entier du fichier. (ex. `'c:\mydata\store1.dat'`).

Exemples IncaStopRecording('c:\mydata\store1.dat');

3.4.12 Régler le mode de lecture des données (Données Online / Offline)

Nom `IncaSetMeasureReadMode`

Description Cette fonction API permet de définir la source à partir de laquelle les données mesurées sont transmises à MATLAB. Les données sont, soit préparées dans INCA puis transférées à MATLAB (données offline), soit sont lues à partir du buffer circulaire (données online).

Pour certains modules, tels que la carte ES1303 et les modules de série ES6xx, aucune donnée offline n'est disponible pendant l'affichage des données mesurées. Il est donc recommandé d'utiliser les données online pendant l'affichage.

Lors de l'enregistrement des données mesurées, les données online ou offline sont utilisables. Les données online peuvent être incomplètes au moment des chargements importants, alors que les données offline sont toujours complètes lorsque les données mesurées sont enregistrées. Cependant, les données offline peuvent être transmises avec un certain délai lors d'importants chargements. Il est recommandé d'utiliser uniquement les données offline lors de l'enregistrement des données mesurées.

Syntaxe `IncaSetMeasureReadMode(measureReadMode)`

Arguments de sortie		
Arguments d'entrée	`measureReadMode`	Paramètre numérique dont la valeur spécifie la source de données. Paramètres possibles : • `1` : Données offline • `0` : Données online (par défaut)
Exemples	`IncaSetMeasureReadMode(0);`	

3.4.13 Lecture des données de mesure

Nom `IncaGetRecords`

Description Transfère les données de mesure vers MATLAB. Les données de mesure de chaque groupe de signaux sont archivées dans un buffer circulaire dédié pouvant contenir jusqu'à 30 secondes de temps de mesure. Les données de mesure sont récupérées par groupes dans MATLAB. Il faudrait donc arrêter l'exécution de votre script dans MATLAB après avoir récupéré les données de mesure. Plus la quantité de données transférées est important à chaque fois, plus le transfert de données est efficace.

Cette fonction transmet un nombre spécifié d'enregistrements pour le raster de mesure spécifié.

Pour de plus amples informations sur le buffer circulaire, consultez l'entrée correspondante dans le "Glossaire INCA" sur la page 8.

Syntaxe

```
[time, data {,state}] = IncaGetRecords(deviceName, groupName, maxRecords {{,latest{, exact}})
```

Arguments de sortie	`time`	Un vecteur contenant les tampons d'horodatage des enregistrements transférés. Cette variable contient un nombre maximal de valeurs `m`, tandis que `m <= maxRecords`.
	`data`	Une matrice bidimensionnelle contenant les valeurs de données pour chaque variable de mesure dans l'ordre où elle a été ajoutée à l'expérimentation par `IncaAddMeasureElement`. Dans cette matrice, la dimension m reflète le nombre d'enregistrements transférés, tandis que n indique le nombre de raster de mesure.

	state	Paramètre en retour optionnel : • 0 : succès. Enregistrements reçus • 1: acquisition non opérationnelle. Aucun enregistrement reçu • 2: enregistrements insuffisants. Aucun enregistrement reçu. Ne peut être retourné que si exact = 1
Arguments d'entrée	`deviceName`	Nom du dispositif
	`groupName`	Nom du raster de mesure Il est possible d'utiliser plusieurs rasters en combinant tout simplement les noms de raster au moyen d'un caractère « + », p. ex. '`10 ms+100 ms`'. L'utilisation d'un tel raster multiple crée un nouveau raster virtuel. Chaque signal peut uniquement être mesuré dans exactement un raster ou un raster multiple.
	`maxRecords`	Nombre maximal d'enregistrements à transférer. Voir aussi le paramètre `exact`. Ce nombre correspond à la dimension `m` pour la variable de temps ou de données dans les valeurs retournées. Si la dimension dans les valeurs retournées atteint la valeur `maxRecords`, les enregistrements ne seront pas tous lus et un dépassement du buffer circulaire peut survenir.
	latest	Définit si les enregistrements les plus anciens ou les plus récents seront reçus
	exact	Définit de recevoir des enregistrements même si n < maxRecords sont présents dans le buffer circulaire ou définit si le buffer circulaire doit rester inchangé.

Exemples	`[t, d] = IncaGetRecords('ETK:1', '100ms', 500);`
	`data = [data; d];`
	`time = [time; t];`
	`[t, d, s] = IncaGetRecords('ETK:1', '100ms', 25, 1, 1);`
	Pour plus d'information sur l'utilisation de cet exemple de code veuillez consulter .	

Note

Les arguments d'entrée optionnels 'latest' et 'exact' et l'argument de sortie optionnel 'state' ont été introduits avec INCA-MIP V16.0.

Note

The raster used in `IncaGetRecords` directly corresponds to the raster used in `IncaAddMeasureElement`, i.e. you have to use the same raster or multi-raster. Example:

```
IncaAddMeasureElement('ETK test device:1','RASTER_
A+RASTER_B', 'N')
IncaAddMeasureElement('ETK test device:1','RASTER_
A+RASTER_B', 'n')
[t,d]= IncaGetRecords('ETK test device:1','RASTER_
A+RASTER_B',15)
```

To check the raster assignment of signals, you can use the command `IncaGetRecordStruct`. Example:

```
l=IncaGetRecordStruct('ETK test device:1', 'RASTER_
A+RASTER_B')
```

Note

Les combinaisons de paramètres suivantes s'exécutent comme suit :

- `latest = 0, exact = 0: (par défaut)`
  Retourne les enregistrements les plus anciens jusqu'à `maxRecords` depuis le buffer circulaire. Les enregistrements plus récents restent inchangés.
- `latest = 1, exact = 0:`
  Retourne les enregistrements les plus récents jusqu'à `maxRecords` depuis le buffer circulaire. Tous les enregistrements plus anciens sont délibérément rejetés.
- `latest = 0, exact = 1:`
  Retourne les `maxRecords` plus anciens enregistrements depuis le buffer circulaire. Les enregistrements plus récents restent inchangés. Si seulement n < `maxRecords` enregistrements sont disponibles dans le buffer circulaire, rien n'est reçu.
- `latest = 1, exact = 1:`
  Retourne les `maxRecords` plus récents enregistrements depuis le buffer circulaire. Tous les enregistrements plus anciens sont délibérément rejetés. Si seulement n < `maxRecords` enregistrements sont disponibles dans le buffer circulaire, rien n'est reçu.

3.4.14 Réinitialiser les buffers tournant

Nom `IncaResetRecords`

Description Cette fonction reinitialise le buffer circulaire de toutes les fréquences d'échantillonnage. Elle est utilisable même en cours de mesure pour réinitialiser tous les buffers tournant. Les buffers tournant sont réinitialisés automatiquement lors du lancement de la mesure ou de l'enregistrement. Il n'est donc pas nécessaire de spécifier explicitement l'exécution de cette commande.

Pour de plus amples informations sur le buffer circulaire, consultez l'entrée correspondante dans le "Glossaire INCA" sur la page 8.

Syntaxe `IncaResetRecords`

Arguments de sortie

Arguments d'entrée

Exemples

3.4.15 Lecture de l'état du matériel (INCA-MIP Evoluée)

Nom `IncaGetHardwareStatus`

Description Cette fonction permet de connaître l'état du matériel pendant une mesure ou un enregistrement

Syntaxe `[status, message] = IncaGetHardwareStatus`

Arguments de sortie

`status` Etat actuel du matériel
- `0` : OK
- `1` : Warning
- `2` : Erreur

`message` Si `status` renvoit la valeur `1` ou `2`, la valeur `message` renvoit un texte décrivant le warning ou l'erreur.

Si `status` renvoit la valeur `1` ou `2`, la mesure ou l'enregsitrement doivent être arrêter avant un autre appel de la fonction `IncaGetHardwareStatus`.

Voir aussi le script exemple dans le fichier `tHWStatus.m` au "Connaître l'AddOn INCA-MIP grâce à des exemples:" sur la page 19

Arguments d'entrée

Exemples

3.4.16 Définir les triggers (INCA-MIP Evoluée)

Nom `IncaSetTrigger`

Description Cette fonction permet de définir la condition de trigger avec le démarrage d'une mesure ou d'un enregistrement avec les fonctions `IncaStartMeasurement` ou `IncaStartRecording`

Syntaxe
```
IncaSetTrigger(startTrigger{, stopTrigger{, preTriggerTime{, postTriggerTime{, duration}}}})
```

Arguments de sortie

Arguments d'entrée

`startTrigger`	Trigger de début. • `manual` pour un trigger manuel • `none` s'il n'y a pas de trigger
`stopTrigger`	Trigger de fin. • `manual` pour un trigger manuel • `none` s'il n'y a pas de trigger (par défaut)
`preTriggerTime`	Durée de pré-trigger en seconde • `none` si non spécifié (par défaut)
`postTriggerTime`	Durée de post-trigger en seconde • `none` si non spécifié (par défaut)
`duration`	Durée de l'enregistrement de la mesure ou de l'enregistrement en seconde • `none` si non spécifié (par défaut) dans ce cas, la durée est infinie.

Exemples
```
IncaSetTrigger('nmot\ETK:1 > 2000', 'none', 2.0, 3.0)
IncaSetTrigger('none', 'none', 'none', 'none', 360)
```

Note

Le tableau suivant présente toutes les combinaisons de paramètres d'entrée qui sont supportées (autres combinaisons causent une exception):

Fonctionnalité du trigger	startTrigger	stopTrigger	preTriggerTime	postTriggerTime	duration
Enregistrement avec durée fixe	'none'	'none'	'none'	'none'	valeur
Enregistrement avec trigger de début manuel, temps du trigger avant le début, et trigger de fin manuel	'manual'	'manual'	valeur	'none'	'none'
Enregistrement avec trigger de début manuel et temps du trigger avant le début et après la fin	'manual'	'none'	valeur	valeur	'none'
Enregistrement avec trigger de début manuel, temps du trigger avant le début et condition du trigger de fin	'manual'	valeur	valeur	'none'	'none'
Enregistrement avec trigger de fin manuel	'none'	'manual'	'none'	'none'	'none'

Fonctionnalité du trigger	startTrigger	stopTrigger	preTriggerTime	postTriggerTime	duration
Enregistrement avec une condition du trigger de début et durée fixe	valeur	'none'	'none'	'none'	valeur
Enregistrement avec une condition du trigger de début et trigger de fin manuel	valeur	'manual'	'none'	'none'	'none'
Enregistrement avec une condition du trigger de début, temps du trigger avant le début et trigger de fin manuel	valeur	'manual'	valeur	'none'	'none'

Fonctionnalité du trigger	startTrigger	stopTrigger	preTriggerTime	postTriggerTime	duration
Enregistrement avec une condition du trigger de début et temps du trigger avant le début et après la fin	valeur	'none'	valeur	valeur	'none'
Enregistrement avec une condition du trigger de début, temps du trigger avant le début et condition du trigger de fin	valeur	valeur	valeur	'none'	'none'

3.4.17 Execution du trigger manuel (INCA-MIP Evoluée)

Nom `IncaExecuteManualTrigger`

Description Cette commande permet de déclencher le trigger manuel de début ou de fin. Cette fonction n'a d'effet que si la commande `IncaSetTrigger` a été utiliser avant avec les paramètres `startTrigger` ou `stopTrigger` règler sur `manual` (manuel).

Syntaxe

```
IncaExecuteManualTrigger(type)
```

Arguments de sortie

Arguments d'entrée

`type` Type de trigger

- `start` pour déclencher le trigger manuel de début
- `stop` pour déclencher le trigger manuel de fin

Exemples

```
IncaExecuteManualTrigger('start')
```

3.4.18 Lire l'état d'un enregistrement (INCA-MIP Evoluée)

Nom `IncaGetRecordingState`

Description Cette fonction permet d'obtenir l'état de l'enregistrement en cours.

Syntaxe `result = IncaGetRecordingState`

Arguments de sortie

result	Etats de l'enregistrement • `0` : non actif • `1` : en attente de trigger ou enregistrement en cours

Arguments d'entrée

Exemples
```
s = IncaGetRecordingState
```

3.4.19 Lire la liste des variables de mesure (INCA-MIP Evoluée)

Nom `IncaGetRecordStruct`

Description Cette fonction permet de connaître la liste des variables de mesure de la mesure ou de l'enregistrement. La liste renvoyée contient les noms des variables de mesure dans le même ordre où elles ont été appelées par la commande `IncaAddMeasureElement`.

Syntaxe
```
list = IncaGetRecordStruct(device,groupName)
```

Arguments de sortie

Arguments d'entrée

`device`	Nom du dispositif
`groupName`	Nom de la fréquence d'échantillonnage Il est possible d'utiliser plusieurs rasters en combinant tout simplement les noms de raster au moyen d'un caractère « + », p. ex. '`10 ms+100 ms`'.

Exemples
```
l = IncaGetRecordStruct('ETK:1', '10ms');
list = IncaGetRecordStruct('device1', 'Syncro');
```

3.5 Calibration

La calibration peut se faire sur des scalaires, courbes et cartographies, notamment des distributions de points d'interruption. Dans chaque expérimentation, il est possible de définir autant de variables de calibration qu'on le souhaite.

> **Note**
>
> Notez que le nom des variables de calibration est sensible à la casse.

3.5.1 Lire les éléments de calibration (INCA-MIP Evoluée)

Nom `IncaBrowseCalibrationElements`

Description Cette fonction permet d'obtenir les éléments de calibration de l'expérimentation avec un filtre de recherche et éventuellement d'un dispositif.

Syntaxe

```
[nom, type] = IncaBrowseCalibrationElements (pattern, {deviceName})
nom = IncaBrowseCalibrationElements (pattern, {deviceName})
```

Arguments de sortie

`nom`	Liste des noms des éléments de calibration.
`type`	Liste de types de calibration: • `Distribution`: Axie de distribution • `OneDTable`: Courbe • `TwoDTable`: Cartographie • `Scalar`: Scalaire • `Array`: Vecteur • `Matrix`: Matrice

Arguments d'entrée

`pattern`	Filtre de recherche à appliquer sur les éléments de mesure. Un '*' correspond à zéro, un ou plusieurs caractères supplémentaires. Un '#' correspond uniquement à un caractère. Tous les autres caractères doivent correspondre avec les éléments de mesure. Il n'y a pas de différence entre les majuscules et les minuscules.
deviceName	Nom du dispositif

Exemples

```
[n,t]=IncaBrowseCalibrationElements('MAP*', 'Device');
[nom,type]=IncaBrowseCalibrationElements ('*');
```

3.5.2 Ajouter un élément de calibration

Nom `IncaAddCalibrationElement`

Description Ajoute une variable de calibration à l'expérimentation en cours. Les calibrations peuvent être effectuées avec des scalaires, des courbes caractéristiques et des cartes englobant les distributions de points d'axe associées. Dans chaque expérimentation, il est possible de définir un nombre quelconque de variables de calibration. Cette commande prend également en charge les distributions de points d'axe et les distributions de points d'axe groupés.

Syntaxe
```
IncaAddCalibrationElement(deviceName, calibrationName {, displayMode})
```

Arguments de sortie

Arguments d'entrée

`deviceName`	Nom du dispositif
`calibrationName`	Nom de la calibration
`displayMode`	Mode d'affichage de l'élément: • `2` : La variable de calibration est affichée et constamment rafraîchie (par défaut) • `1` : La variable de calibration est affichée mais non rafraichie • `0` : La variable de calibration n'est pas affichée Sélectionner `1` (affichage uniquement) peut améliorer considérablement la performance pour de grosse quantité de données.

Exemples
```
IncaAddCalibrationElement('anEtk', 'Scalar');

IncaAddCalibrationElement('anEtk', 'Curve');
IncaAddCalibrationElement('anEtk', 'Map');
```

Note

Les variables de calibration du type 'axis' et 'group axis' ont été prises en charge depuis INCA-MIP V16.0. Pour les axes groupés, aucune interpolation des courbes et des cartes dépendantes n'est exécutée.

3.5.3 Lire la valeur de calibration

Nom	`IncaGetCalibrationValue`	
Description	Cette fonction permet de lire la valeur actuelle de la variable de calibration ou des distributions de points d'interruption	
Syntaxe	`value = IncaGetCalibrationValue(deviceName, calibrationName {, start, size} {, valueType})`	
Arguments de sortie	`value`	La valeur actuelle de la calibration; elle doit correspondre aux types de données indiqués ci-dessous: • Scalaires: une matrice (1,1) • Courbes: une matrice (x,1) • Cartographies: une matrice (x,y) • Distributions de points d'interruption: une matrice (x,1)
Arguments d'entrée	`deviceName`	Nom du dispositif
	`calibrationName`	Nom d'élément de calibration

	`start`	Index de départ. Types de données supportées:
		• Pour les courbes et distributions de points d'interruption un index de départ `x` doit être spécifié.
		`x >= 1`
		• Pour les cartographies un index de départ `[x,y]` doit être spécifié.
		`x,y >= 1`

`size`	Nombre de valeur à lire. Types de données supportées:
	• Pour les courbes et distributions de points d'interruption un compteur `n` doit être spécifié.
	`n >= 1`
	• Pour les cartographies un compteur `[n,m]` doit être spécifié.
	`n,m >= 1`

`valueType`	Sélection de l'argument de sortie (texte). La fonction renvoie soit la valeur de la variable de calibration (mode par défaut), soit la distribution du point d'appui en X ou Y. Paramètres possibles : • `v`: la valeur • `x`: renvoie le point d'appui en X (courbes et cartographies) • `y`: renvoie le point d'appui en Y (cartographies)

Exemples

```
aValue = IncaGetCalibrationValue('anEtk',
'Scalar');
aCurve = IncaGetCalibrationValue('anEtk',
'Curve');
aMap = IncaGetCalibrationValue('anEtk',
'Map');
xMap = IncaGetCalibrationValue('anEtk',
'Map', 'x');
yMap = IncaGetCalibrationValue('anEtk',
'Map', 'y');
aCurveRange = IncaGetCalibrationValue
('anEtk', 'Curve', 2, 3);
aMapRange = IncaGetCalibrationValue ('anEtk',
'Map', [2,3], [3,4]);
xMapRange = IncaGetCalibrationValue ('anEtk',
'Map', 2, 3, 'x');
```

3.5.4 Modifier une valeur de calibration

Nom `IncaSetCalibrationValue`

Description Cette fonction permet d'attibuer une valeur à une variable de calibration ou à un point d'interruption (d'appui).

Syntaxe

```
IncaSetCalibrationValue( deviceName, calibrationName, value )
IncaSetCalibrationValue( deviceName, calibrationName, value, valueType )
IncaSetCalibrationValue( deviceName, calibrationName, value, start )
IncaSetCalibrationValue( deviceName, calibrationName, value, start, valueType )
result = IncaSetCalibrationValue( deviceName, calibrationName, value )
result = IncaSetCalibrationValue( deviceName, calibrationName, value, valueType )
result = IncaSetCalibrationValue( deviceName, calibrationName, value, start )
result = IncaSetCalibrationValue( deviceName, calibrationName, value, start, valueType )
```

Arguments de sortie	`result`	Résultat de la calibration (facultatif, uniquement en cas de défaillances) Si aucun bit de résultat n'est posé, c'est que la valeur de calibration a été modifiée avec succès. C'est également le cas si l'un des bits 5 à 8 est posé, ce qui fournit des informations supplémentaires. Si toutefois, l'un des bits 0 à 4 est posé, c'est que la calibration a échoué. • Bit 0 actif: Calibration non faite • Bit 1 actif: Limite logicielle basse dépassée • Bit 2 actif: Limite logicielle haute dépassée • Bit 3 actif: Limite matérielle basse dépassée • Bit 4 actif: Limite matérielle haute dépassée • Bit 5 actif: Valeur saturée à la limite logicelle basse • Bit 6 actif: Valeur saturée à la limite logicelle haute • Bit 7 actif: Valeur saturée à la limite matérielle basse • Bit 8 actif: Valeur saturée à la limite matérielle haute Il peut y avoir plusieurs raisons pour qu'une calibration ne soit pas faite. Par exemple en fonction du mode de calibration actif, si une des limite a été dépassée. Dans ce cas il y a plus d'information renvoyée par les bits 1 à 4. Une autre raisons peut être que l'élément de calibration ou la page active est protégé en écriture ou que les points d'appui x ou y ne sont pas monotones. Dans tous ces cas, seul le bit 0 est activé.
Arguments d'entrée	`deviceName`	Nom du dispositif
	`calibrationName`	Nom de l'élément de calibration

`value`	La valeur actuelle de la calibration. Elle doit correspondre aux types de données indiqués ci-dessous: • Scalaires: une matrice (1,1) • Courbes: une matrice (x,1) • Cartographies: une matrice (x,y) • Distributions de points d'interruption (d'appui) : une matrice (x,1)

`start`	Index de départ. Types de données supportées:
	• Pour les courbes et distributions de points d'interruption un index de départ `x` doit être spécifié.
	`x >= 1`
	Pour les cartographies un index de départ `[x,y]` doit être spécifié.
	`x,y >= 1`

`valueType`	Sélection de la valeur de retour (texte). La fonction renvoie soit la valeur de la variable de calibration (par défaut) soit la distribution X ou Y de points d'interruption (point d'appui). Paramètres possibles : • `v`: valeur • `x`: point d'appui en x (courbe et cartographies) • `y`: point d'appui en y (cartographie)

Exemples

```
IncaSetCalibrationValue('anEtk', 'Scalar',
aValue);
IncaSetCalibrationValue('anEtk', 'Curve',
aCurve);
IncaSetCalibrationValue('anEtk', 'Map',
aMap);
IncaSetCalibrationValue('anEtk', 'Map', xMap,
'x');
IncaSetCalibrationValue('anEtk', 'Map', yMap,
'y');
IncaSetCalibrationValue('anEtk', 'Curve',
aCurveRange, 2);
IncaSetCalibrationValue('anEtk', 'Map',
aMapRange,[2,3]);
IncaSetCalibrationValue('anEtk', 'Map',
xMapRange, 2, 'x');
```

3.5.5 Assigner un jeu de données au dispositif (INCA-MIP Evoluée)

Nom `IncaSetDatasetInDevice`

Description Cette fonction permet d'assigner un jeu de donnée à un dispositif dans une expérimentation ouverte

Syntaxe
```
IncaSetDatasetInDevice(device,dataset)
```

Arguments de sortie

Arguments d'entrée

`device`	Nom du dispositif
`dataset`	Chemin d'accès du jeu de données dans la base de données INCA

Exemples
```
IncaSetDatasetInDevice
('ETK:1','Ds4711\Ds4711_3')
```

3.5.6 Lister les jeux de données d'un dispositif (INCA-MIP Evoluée)

Nom `IncaGetDatasetsForDevice`

Description Cette fonction permet d'obtenir la liste des noms de tous les jeux de données pour un dispositif

Syntaxe
```
nom = IncaGetDatasetsForDevice(device)
[nom, properties] = IncaGetDatasetsForDevice
(device)
```

Arguments de sortie

`nom`	Liste, avec le chemin d'accès, de tous les jeux de données trouvés
`properties`	Liste des propriétés des jeux de données Les valeurs possibles sont: • '' (champ vide): Jeu de données en accès lecture/écriture • `r` : Jeu de données en lecture seule • `m` : Jeu de données maître en accès lecture/écriture • `mr` : Jeu de donnée maître en lecture seule

Arguments d'entrée

`device`	Nom du dispositif

Exemples
```
l = IncaGetDatasetsForDevice('ETK:1')
```

3.5.7 Etablir le mode de calibration (INCA-MIP Evoluée)

Nom `IncaSetCalibrationMode`

Description Cette fonction permet de fixer le mode de calibration global valide pour toutes les calibrations définis avec la fonction `IncaSetCalibrationValue`. Le mode reste valide même après avoir fermé puis réouvert l'expérimentation. En démarrant l'interface MATLAB, le mode par défaut pour les limites hautes et basses est : `rejectWeakBoundViolation`.

Syntaxe

```
IncaSetCalibrationMode(lowerLimitMode,
upperLimitMode)
```

Arguments de sortie

Arguments d'entrée

`lowerLimitMode` Le nouveau mode de calibration pour les limites basses.

`upperLimitMode` Le nouveau mode de calibration pour les limites hautes:

- `rejectWeakBoundViolation`: Rejette complétement la calibration si une limite logicielle est dépassé au moins une fois (par défaut)
- `limitToWeakBound`: Si les limites logicielles hautes et basses sont dépassées, utiliser à la place la valeur de la limite logicielle haute ou basse
- `rejectHardBoundViolation`: Ignorer les limites logicielles. Rejetter la calibration si les limites matérielles sont dépassées au moins une fois
- `limitToHardBound`: Ignorer les limites logicielles. Si les limites matérielles hautes ou basses sont dépassées, utiliser à la place la valeur de la limite matérielle haute ou basse.

Exemples

```
IncaSetCalibrationMode ('rejectHardBoundVio-
lation', 'limitToHardBound')
```

3.5.8 Groupement des dispositifs (INCA-MIP Evoluée)

Nom `IncaGroupDevices`

Description Activer ou désactiver le groupement de dispositif

Syntaxe `IncaGroupDevices(onOff)`

Arguments de sortie

Arguments d'entrée

`onOff`	• `0` : Désactiver le groupement de dispositif • `1` : Activer le groupement de dispositif

Exemples

```
IncaGroupDevices(1)
```

3.5.9 Ecrire un fichier DCM (INCA-MIP Evoluée)

Nom `IncaWriteToFile`

Description Cette fonction permet écrire un fichier DCM à partir des calibrations de l'expérimentation ouverte

Syntaxe

```
IncaWriteToFile(format,file,device, calibs
{,options})
```

Arguments de sortie

Arguments d'entrée

`format`	Identificateur du format du fichier: • '`DCM`': format DCM
`file`	Chemin d'accès complet du fichier à écrire
`device`	Dispositif des calibrations qui seront écrites
`calibs`	Liste des éléments de calibration à écrire (tableau)
`options`	Options utilisées pour écrire au format indiqué

Exemples

```
calibs = {'A0_KW', 'BRABEVI_KL', 'KFZW_GKF'};
IncaWriteToFile('DCM','C:\DCMOut1.dcm',
'device1', calibs);
IncaWriteToFile('DCM','C:\DCMOut2.dcm',
'ETK:1','A0_KW');
```

3.6 Gestion des pages mémoires

Toutes les fonctions prédemment décrites sont effectives pour la page actuellement active d'un dispositif. En principe, l'accès à la calibration est possible à partir de la page de travail uniquement. Cependant, même dans ce cas, il est

possible que le droit d'écriture dans la page de travail de l'ETK soit bloqué parce que les sommes de contrôle de la page de travail entre INCA et l'ETK ne correspondent pas.

Les fonctions API suivantes peuvent être utilisées pour la gestion des pages mémoires.

3.6.1 Activer une page mémoire

Nom `IncaSwitchPage`

Description Cette fonction permet d'activer la page mémoire spécifiée

Syntaxe `IncaSwitchPage(deviceName, pageName)`

Arguments de sortie

Arguments d'entrée

`deviceName`	Nom du dispositif
`pageName`	Nom de la page: • `wp`: Page de travail • `rp`: Page de référence

Exemples

3.6.2 Get Current Page (INCA-MIP Evoluée)

Nom `IncaGetCurrentPage`

Description Cette fonction permet de connaître la page mémoire actuellement active

Syntaxe `pageName = IncaGetCurrentPage(deviceName)`

Arguments de sortie

`pageName`	Nom de la page mémoire active: • `wp`: Page de travail • `rp`: Page de référence

Arguments d'entrée

`deviceName`	Nom du dispositif

Exemples

3.6.3 Vérifier la protection en écriture

Nom `IncaIsPageWriteProtected`

Description Cette fonction permet de vérifier si la page mémoire spécifiée est protégée en écriture

Syntaxe `isRW = IncaIsPageWriteProtected(deviceName, pageName)`

Arguments de sortie	`isRw`	• `0` : la page n'est pas protégée en écriture • `not 0` : la page est protégée en écriture
Arguments d'entrée	`deviceName`	Nom du dispositif
	`pageName`	Nom de la page: • `wp`: Page de travail • `rp`: Page de référence

Exemples

3.6.4 Télécharger vers l'ECU la page mémoire

Nom `IncaDownloadPage`

Description Cette fonction permet de télécharger la page mémoire spécifiée dans le calculateur

Syntaxe `IncaDownloadPage(deviceName, pageName)`

Arguments de sortie		
Arguments d'entrée	`deviceName`	Nom du dispositif
	`pageName`	Nom de la page à télécharger • `wp`: Page de travail • `rp`: Page de référence

Exemples

3.6.5 Copier la page mémoire

Nom `IncaCopyPageFromTo`

Description Cette fonction permet de copier la page mémoire spécifiée. Actuellement il est uniquement possible de copier de la page de référence vers la page de travail; les autres combinaisons de copie entre sources et cibles ne sont pas supportées.

Syntaxe `IncaCopyPageFromTo( deviceName, sourcePageName, destinationPageName)`

Arguments de sortie		
Arguments d'entrée	`deviceName`	Nom du dispositif

`sourcePageName`	Nom de la page à copier (source): • `wp`: Page de travail • `rp`: Page de référence
`destinationPageName`	Nom de la page à écraser (cible): • `wp`: Page de travail • `rp`: Page de référence

Exemples

3.6.6 Télécharger les différences

Nom `IncaDownloadDifferences`

Description Cette fonction permet de charger les différences entre la page de travail et la page de référence dans le calculateur. Comme sur INCA, ceci n'est valable que si la page de travail et la page de référence du calculateur cible correspondent à la page de référence présente sous INCA.

Syntaxe `IncaDownloadDifferences( deviceName )`

Arguments de sortie

Arguments d'entrée	`deviceName`	Nom du dispositif

Exemples

3.6.7 Télécharger depuis l'ECU les pages mémoires (INCA-MIP Evoluée)

Nom `IncaUploadPages`

Description Cette fonction permet de récupérer sous INCA la page de référence et de travail du calculateur. Ces nouveaux jeux de données sont automatiquement assignés au dispositif.

Syntaxe `IncaUploadPages(device{,referencePage, workingPage})`

Arguments de sortie

Arguments d'entrée	`device`	Nom du dispositif

`referencePage`	Nom du jeu de données pour la page de référence récupérée. Si non précisé, INCA utilise un nom par défaut
`workingPage`	Nom du jeu de données pour la page de travail récupérée. Si non précisé, INCA utilise un nom par défaut

Exemples

```
IncaUploadPages('ETK:1');
IncaUploadPages('ETK:1', 'ref_1', 'work_1');
```

3.7 Exemples

Exemple 1

```
% Vérifier si la page de travail est protégée en
% écriture et envoyer la page si tel est le cas.
if(IncaIsPageWriteProtected (’anEtk’, ’wp’))
 IncaDownloadPage(’anEtk’, ’wp’);
end
% Passer à la page de travail
IncaSwitchPage( ’anEtk’, ’wp’);
```

Exemple 2

Dans l'exemple ci-dessous, les fonctions sont utilisées pour lire les valeurs des variables de mesure du dispostif `MyDevice` avec une fréquence d'échantillonnage de `10ms`. Pour utiliser cet exemple vous devez d'abord ouvrir une expérimentation sous INCA qui inclue un dispositif nommé `MyDevice` avec 4 voies nommée : Chan1, Chan2, Chan3, Chan4.

```
      % Sélectionner les signaux suivant
      IncaAddMeasureElement( ’MyDevice’, ’10ms’, ’Chan1’
);
      IncaAddMeasureElement( ’MyDevice’, ’10ms’, ’Chan2’
);
      IncaAddMeasureElement( ’MyDevice’, ’10ms’, ’Chan3’
);
      IncaAddMeasureElement( ’MyDevice’, ’10ms’, ’Chan4’
);
      % Maintenant lancer la mesure
      data = [];
      time = [];
      IncaShowMessages(0);
      IncaSetMeasureReadMode(0)
      IncaStartMeasurement;
```

```
      deltaT = 0;
      % Mesurer pendant 20 secondes
      while( deltaT < 20 )
            % Faire une pause pendant 0.1 secondes pour
avoir
            % plus d'un échantillon -- % économise le
temps processeur
            pause(0.1)
            % Récupérer 500 échantillons du groupe 10ms
            [ t, d ]=IncaGetRecords( 'MyDevice', '10ms',
500 );
            % Ajouter t et d au temps et aux données
            data = [data; d];
            time = [time; t];
            if( length(time) )
            % Calculer le temps de la mesure
            deltaT = time( length(time)) - time(1);
            end
      end
      IncaStopMeasurement;
      IncaShowMessages(1);
      % Tracer le résultat
      plot(time, data);
```

Cet exemple n'utilise qu'une seule fréquence d'échantillonnage. Cependant, vous pouvez utiliser plusieurs groupes et demander les données pour chaque groupe indépendamment de MATLAB.

4 Création et distribution de fichiers exécutables

Avec l'AddOn INCA-MIP, vous pouvez créer et compiler des fichiers m contenant des appels aux fonctions API de MATLAB, y compris des fonctions INCA-MIP. Le résultat est un fichier autonome qui peut être utilisé dans n'importe quel environnement sans forcément avoir le logiciel MATLAB.

La façon de procéder pour les différentes versions du compilateur MATLAB diffère dans quelques détails.

4.1 Création et distribution de fichiers exécutables avec le compilateur MATLAB R13

Pour la création de fichiers exécutables une installation de MATLAB est nécessaire. Le fichier programme ainsi créé peut cependant être utilisé avec des copies de quelques DLL MATLAB et ETAS sans qu'une installation de MATLAB ne soit présente sur le système cible.

4.1.1 Compilation des fichiers m

Compiler des fichiers m avec le compilateur MATLAB R13 :

1. Exécuter la commande suivante :

   `mcc -m <m-file-script>`

   Exemple:

   Avec la commande suivante un fichier exécutable est créé à partir du fichier `testCase1.m` :

   `mcc -m testCase1`

   Comme résultat vous obtenez le fichier `testCase1.exe`.

Veuillez consulter votre documentation MATLAB au mot clef *Compilateur MATLAB* ou *`MATLAB Compiler`* ou `mcc` pour plus d'informations.

Note

Tous les fichiers Inca*.dll utilisés par le script ainsi que le fichier incaRci2Matlab.dll doivent être copiés sur le système cible où le script compilé est exécuté (voir "Distribution des fichiers exécutables autonome" sur la page suivante).

Note

INCA ne peut être commandé une seule fois depuis MATLAB à un moment donné. La tentative d'exécuter une deuxième fois INCA depuis une deuxième instance MATLAB ou par l'intermédiaire d'un fichier exécutable est interrompue avec un message d'erreur.

4.1.2 Distribution des fichiers exécutables autonome

Les fichiers exécutables autonomes créés comme décrit ci-dessus ont besoin des bibliothèques d'exécution de MATLAB et d'ETAS. Une installation de MATLAB n'est pas nécessaire.

Distribuer les fichiers exécutables qui ont été compilés avec le compilateur MATLAB R13 :

1. Installez les bibliothèques d'exécution de MATLAB nécessitées. Consultez votre documentation MATLAB dans le chapitre *Distributing Stand-Alone Applications* pour savoir comment installer les bibliothèques d'exécution MATLAB.

   Pour installer les bibliothèques d'exécution d'ETAS, installez l'AddOn INCA-MIP et sélectionnez l'option **Installation into ETASData** dans la procédure d'installation (voir "Istaller INCA-MIP" sur la page 11).

ou

   Copiez les fichiers requis à partir de votre installation MATLAB de votre ordinateur de développement situés dans les répertoires suivant :

   ```
   %MATLABDir%\bin\win32\
   incaRci2Matlab.dll
   ```

   ```
   %MatlabDir%\toolbox\matlab\
   general\Inca*.dll
   ```

2. Copiez ces deux fichiers et le fichier exécutable dans le même répertoire.

4.2 Création et distribution de fichiers exécutables qui ont été compilés avec le compilateur MATLAB R14 ou une version supérieure

Pour la création de fichiers exécutables une installation de MATLAB est nécessaire. Le fichier exécutable ainsi créé peut cependant être utilisé sur le système cible sans qu'une installation de MATLAB ou des copies de DLL MATLAB et ETAS supplémentaires soit nécessaire.

4.2.1 Compilation des fichiers m

Compiler des fichiers m avec le compilateur MATLAB R14 :

1. Copiez tous les fichiers Inca*.dll dans le répertoire de travail actuel.
2. Exécuter la commande suivante :

   ```
   mcc -m <m-file-script> -a incaRci2Matlab.dll
   ```

   Exemple

Avec la commande suivante un fichier exécutable est créé à partir du fichier `testCase2.m` :

```
mcc -m testCase2 -a incaRci2Matlab.dll
```

Comme résultat vous obtenez le fichier `testCase2.exe`.

Le compilateur MATLAB R14 crée un conteneur avec toutes les DLL de fonction MEX et les DLL dépendantes qui sont nécessaires pour l'exécution du script MATLAB compilé. Tous les fichiers Inca*.dll utilisés par le script ainsi que le fichier incaRci2Matlab.dll doivent être contenus dans ce conteneur.

Lors de l'exécution du script compilé, les DLL ne doivent pas être présentes sur le système.

Veuillez consulter votre documentation MATLAB au mot clef *Compilateur MATLAB* ou *`MATLAB Compiler`* ou `mcc` pour plus d'informations.

Note

INCA ne peut être commandé une seule fois depuis MATLAB à un moment donné. La tentative d'exécuter une deuxième fois INCA depuis une deuxième instance MATLAB ou par l'intermédiaire d'un fichier exécutable est interrompue avec un message d'erreur.

Note

Dans MATLAB R14 SP3 (Version 7.1) ou supérieur, les DLLs de fonction INCA MEX ont l'extension *.mexw32.

4.2.2 Distribution des fichiers exécutables autonome

Pour l'appel de fichiers exécutables qui ont été compilés avec le compilateur MATLAB R14, seul le fichier exécutable lui-même est nécessaire. Une installation de MATLAB ou des copies des bibliothèques MATLAB ne sont pas nécessaires.

Distribuer les fichiers exécutables qui ont été compilés avec le compilateur MATLAB R14 :

- Copiez simplement les fichiers exécutables sur le système cible.

Vous pouvez ensuite les exécuter tout simplement ; d'autres démarches ne sont pas nécessaires.

5 Informations des Contacts

ETAS siège principal

ETAS GmbH

Borsigstraße 24	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Allemagne	Internet:	www.etas.com

ETAS filiales et support

Pour les détails sur vos services de distribution en plus que votre équipe de support et vos hotlines, regardez les pages d'internet ETAS:

ETAS filiales	Internet:	www.etas.com/en/contact.php
ETAS support:	Internet:	www.etas.com/en/hotlines.php

Index

V