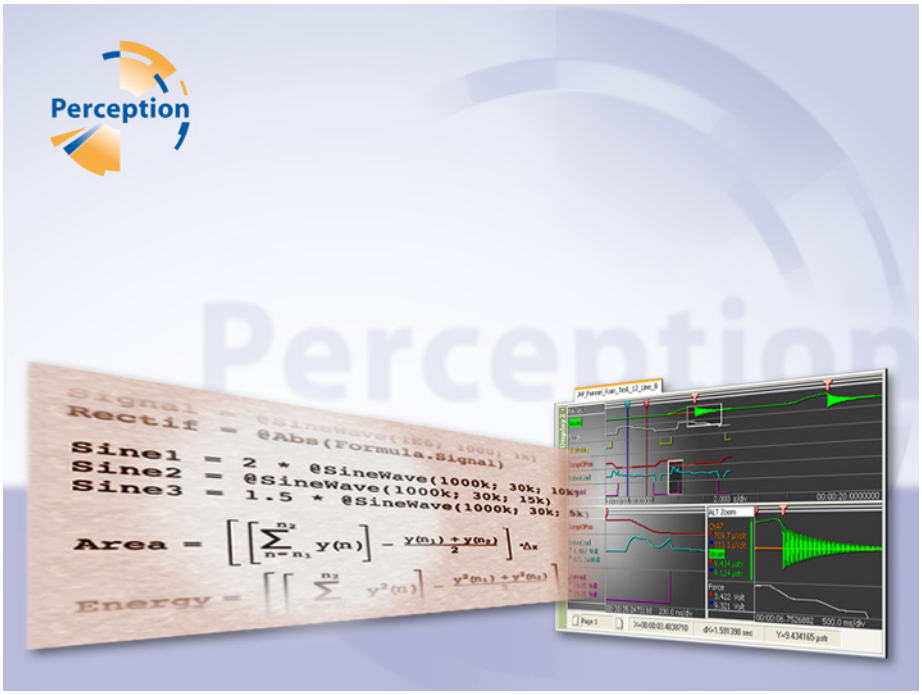
Genesis HIGH SPEED

Manuel d'utilisation

Français



Option d'analyse

Perception

HBM

I3034-2.0 fr

Version du document 2.0 - Octobre 2010

Pour Perception 6.0 ou ultérieur

Pour consulter les termes et conditions d'HBM, visiter le site www.hbm.com/terms

HBM GmbH
Im Tiefen See 45
64293 Darmstadt
Allemagne
Tél. : +49 6151 80 30
Fax : +49 6151 8039100
E-mail : info@hbm.com
www.hbm.com/highspeed

ACCORD DE LICENCE ET GARANTIE
Pour plus d'informations sur l'ACCORD DE LICENCE ET GARANTIE, veuillez vous référer à www.hbm.com/terms.

Sommaire à la page

1 Option d'analyse

1.1 Introduction

L' option d'analyse Perception permet d'effectuer des calculs à partir de données mesurées. De nombreuses fonctions intégrées, des statistiques de base aux opérations mathématiques les plus complexes, vous aident à réaliser vos analyses.

L'option d'analyse consiste essentiellement en :

- de nombreuses fonctions ;
- une base de données de formules.

La base de données de formules vous permet de créer votre propre ensemble de fonctions sans programmation ni séquençage. Il vous suffit de saisir le calcul requis et le résultat s'affiche. Les résultats des nouvelles données sont actualisés automatiquement, pas seulement dans la base de données de formules, mais aussi directement dans les affichages et les rapports. Une fois définies, les formules peuvent être enregistrées en vue d'être réutilisées.

La base de données de formules peut stocker un nombre illimité de formules, chacune ayant ses propres nom et unités. Il est possible de créer une formule à l'aide d'opérations arithmétiques portant sur des formes d'onde et des échelles et de la combiner à l'une des fonctions intégrées, à des valeurs de curseurs ou au résultat d'une autre formule. La fonction de saisie semi-automatique et le guide d'aide en ligne vous aident à exploiter les diverses options.

Pour plus de simplicité, nous utiliserons le terme « base de données de formules » pour faire référence à l'option d'analyse, à ses fonctions et à sa base de données de formules.

1.1.1 Installation de l'option d'analyse

Le logiciel Perception nécessite une clé HASP. HASP (Hardware Against Software Piracy) est un système matériel (clé matérielle) de protection contre la copie des logiciels, qui empêche toute utilisation non autorisée des applications logicielles.
Chaque clé HASP contient un numéro d'identification unique utilisé pour personnaliser l'application selon les fonctions et les options achetées. Cette clé est également utilisée pour stocker les paramètres de licence, ainsi que les données spécifiques aux applications et au client.
Si vous achetez séparément l'option d'analyse, vous recevez un « fichier de clé » personnalisé. Vous devez utiliser ce fichier pour déverrouiller les nouvelles fonctions.

Vous trouverez le numéro de série de votre clé dans **Aide ▸ À propos de Perception.**

Pour mettre à jour les informations de la clé :

1 Choisir **Aide ▸ Mettre à jour la clé...**
2 Dans la boîte de dialogue Ouvrir, rechercher le fichier de clé (*.pKey), puis cliquer sur **Ouvrir**.
3 Si tout se passe bien, le message suivant apparaît :

Figure 1.1 : Boîte de dialogue de protection contre la copie des logiciels

4 Cliquer sur **OK**.

Après l'installation, vous pouvez aller dans **Aide ▸ À propos de Perception ▸ Plus...** pour voir toutes les options installées.

Vous devez redémarrer le programme pour que les modifications soient prises en compte. L'option d'analyse est désormais disponible.

1.2 Feuille de la base de données de formules

La feuille de la base de données de formules permet de créer et de modifier des formules.

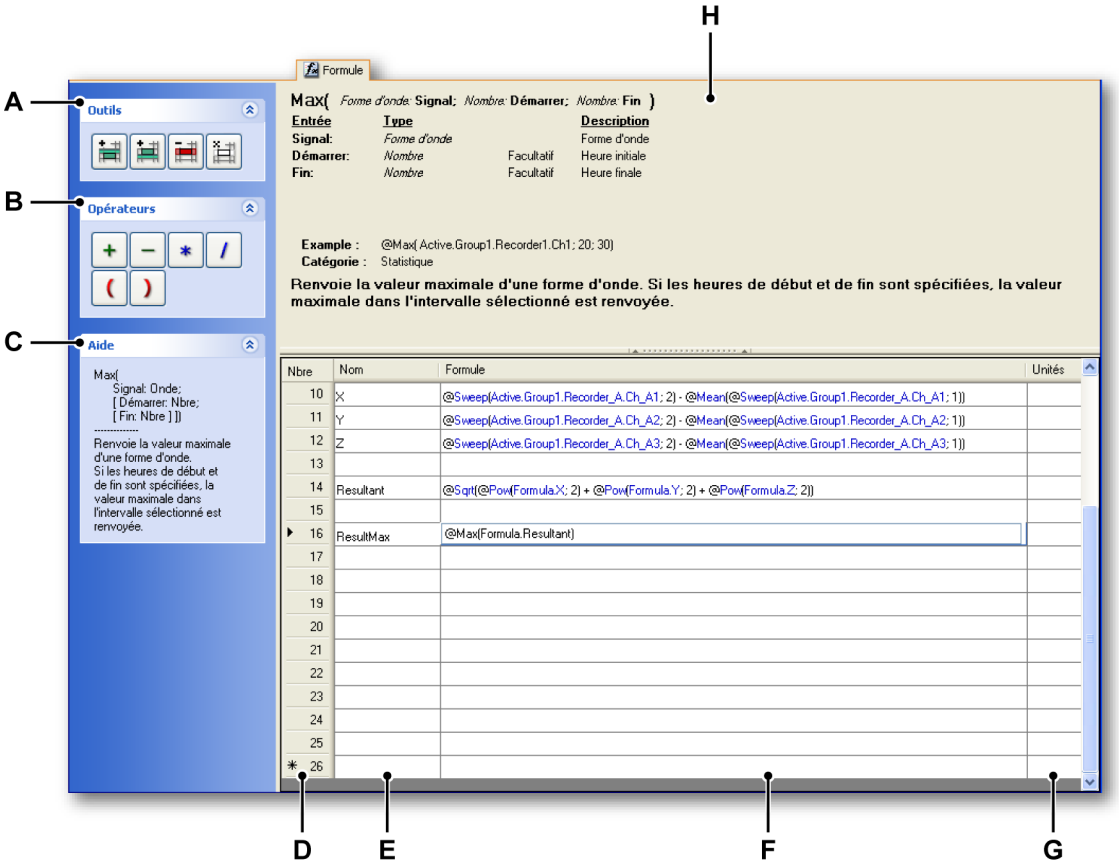


Figure 1.2 : Feuille de la base de données de formules

A Outils

B Opérateurs

C Aide par défaut

D Numéros des lignes de la base de données de formules avec pointeur de ligne

E Colonne contenant les noms des formules

F Colonne contenant les formules

G Unités des formules

H Aide facultative

A **Outils** Des outils permettent d'ajouter, de supprimer et d'effacer des lignes de formule. Ils s'appliquent à la ligne sélectionnée. Cette dernière est indiquée par le pointeur de ligne.

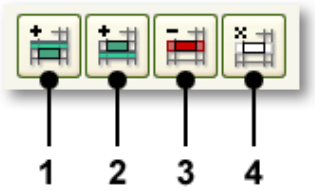


Figure 1.3 : Outils de la feuille de formules

1 Insérer une ligne au-dessus de la ligne active

2 Insérer une ligne en dessous de la ligne active

3 Supprimer la ligne

4 Effacer la ligne

B **Opérateurs** Les boutons d'opérateur permettent d'insérer les opérateurs de base.

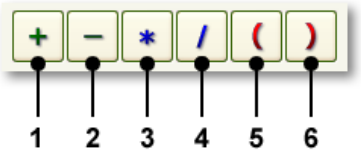


Figure 1.4 : Opérateurs de la feuille de formules

1 Ajouter

2 Soustraire

3 Multiplier

4 Diviser

5 Ouvrir la parenthèse

6 Fermer la parenthèse

C **Aide par défaut** Cette zone affiche des informations d'aide de base concernant la fonction sélectionnée.

D **Numéros de ligne** Pour plus de clarté, chaque ligne est numérotée. La ligne active est indiquée par un pointeur triangulaire en regard du numéro.

E-G **Formules** Chaque formule possède un nom, un corps et des unités. Cette zone est également appelée « éditeur de formules ».

H **Aide supplémentaire** La zone d'aide supplémentaire permet d'afficher des informations d'aide plus détaillées.

Vous pouvez afficher ou masquer cette aide supplémentaire.

Pour afficher ou masquer l'aide supplémentaire :

- Cliquer sur la poignée en haut de la zone des formules.

1.3 Définitions

La base de données de formules utilise des opérateurs et des fonctions. Les opérateurs nécessitent des variables et les fonctions, des paramètres.

1.3.1 Constantes

Les constantes sont des valeurs numériques ou des chaînes de texte saisies directement dans la formule. Les nombres saisis peuvent être en virgule flottante ou des nombres entiers. Au sein du logiciel, toutes les valeurs numériques sont des nombres en virgule flottante. Les chaînes sont saisies entre guillemets, par exemple "texte".

1.3.2 Variables

Les variables sont des éléments (formes d'onde, nombres ou chaînes de texte) provenant des sources de données. Une variable peut être sélectionnée dans les sources de données. Les formules précédemment définies peuvent également être utilisées comme variables.

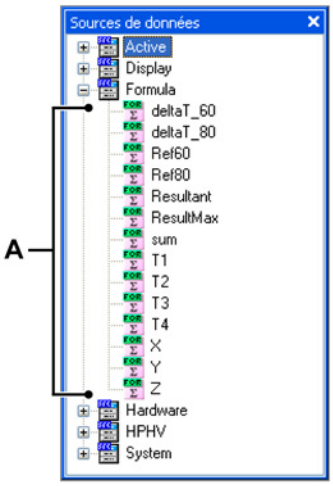


Figure 1.5 : Navigateur de sources de données avec résultats de formules

A Résultats de formules

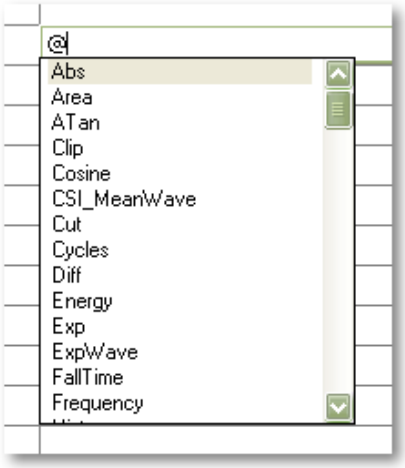
Dans le corps de la formule, une variable est référencée par son chemin complet dans les sources de données. Par exemple, dans l'illustration ci-dessus, le résultat de formule T2 est référencé de la manière suivante : **Formula.T2**

1.3.3 Fonctions

Une fonction utilise un certain nombre de paramètres (qui peuvent être des variables et/ou des constantes) pour générer un résultat. Les types des paramètres et des résultats dépendent de la fonction.

Pour saisir une fonction :

1 Cliquer une fois dans le corps de formule d'une ligne vierge pour le sélectionner, ou double-cliquer pour passer en mode d'édition.
2 Saisir le signe @.
- La liste des fonctions disponibles s'affiche.



3 Pour sélectionner une fonction dans la liste, procéder comme suit :
- Utiliser la barre de défilement pour parcourir la liste des fonctions disponibles.
- Cliquer sur la fonction voulue.
- Utiliser les touches fléchées pour sélectionner une fonction, puis appuyer sur Entrée.

La description complète de la fonction sélectionnée apparaît dans la zone Aide.

Veuillez noter que tant qu'aucun nom n'est attribué à la formule, cette dernière est considérée comme une ligne de commentaire.

1.4 Modification de la mise en page

Vous pouvez modifier la mise en page en ajoutant, supprimant et effaçant des lignes. Vous pouvez utiliser directement les outils ou cliquer avec le bouton droit de la souris pour ouvrir un menu contextuel proposant les mêmes commandes.

1.4.1 Ajout, suppression et effacement de lignes

Vous pouvez ajouter, supprimer et effacer des lignes. L'opération peut s'appliquer à une ou plusieurs lignes.

Pour sélectionner plusieurs lignes :

- Pour sélectionner des lignes consécutives :
  - Cliquer sur la première ligne et faire glisser la souris jusqu'à la dernière ligne.
  - Cliquer sur la première ligne, maintenir la touche Maj enfoncée, puis cliquer sur la dernière ligne.
- Pour sélectionner des lignes non consécutives, maintenir la touche CTRL enfoncée et cliquer sur chacune des lignes.

Pour ajouter une ou plusieurs lignes :

1. Sélectionner une ou plusieurs lignes.
   - *Le pointeur de ligne passe à la dernière ligne sélectionnée.*
2. Cliquer sur l'outil voulu :
   - **Insérer ligne(s) au-dessus** pour insérer une ou plusieurs lignes au-dessus de la ligne sélectionnée.
   - **Insérer ligne(s) en dessous** pour insérer une ou plusieurs lignes en dessous de la ligne sélectionnée.
   - *Une ou plusieurs lignes vierges sont alors insérées.*

Pour supprimer une ou plusieurs lignes :

Sélectionner une ou plusieurs lignes, puis procéder comme suit :

- Pour supprimer une ligne et son contenu, cliquer sur l'outil **Supprimer ligne(s) sélectionnée(s)**.
- Pour supprimer le contenu mais conserver la ligne, cliquer sur l'outil **Effacer ligne(s) sélectionnée(s)**.
- Dans la boîte de dialogue de confirmation qui apparaît, cliquer sur **OK**.

Pour effacer un champ :

Vous pouvez effacer le contenu d'un seul champ comme suit :

- Sélectionner le champ à effacer puis appuyer sur **Suppr.**
- Sélectionner le champ à effacer puis appuyer sur **Entrée** : le champ est alors ouvert pour être modifié et l'ensemble du texte est sélectionné. Appuyer sur **Suppr**.

1.4.2 Navigation

Vous pouvez passer d'un champ à l'autre en utilisant la touche **Tab** et les touches **fléchées**.

1.5 Création de formules

Vous pouvez facilement créer des formules et ajouter un commentaire. Diverses techniques de saisie semi-automatique vous guident pendant la définition des formules tout en limitant les saisies. Ce système réduit également les risques d'erreurs.

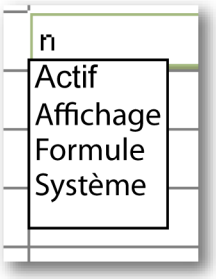
L'éditeur de formules se présente comme une base de données où chaque ligne correspond à une entrée. Trois champs modifiables permettent de saisir un commentaire et des formules.

1.5.1 Saisie d'un commentaire

Vous pouvez saisir un commentaire dans l'éditeur de formules pour fournir des précisions quant aux formules que vous créez. Ce commentaire est affiché en vert. La ligne sur laquelle il apparaît n'a pas de nom.

Pour saisir un commentaire :

1. Cliquer une fois dans le corps de formule d'une ligne vierge pour le sélectionner, ou double-cliquer pour passer en mode d'édition.
2. Commencer la saisie du texte. Une liste s'affiche après la saisie du premier caractère :



3. Ignorer cette liste. Elle disparaît une fois le premier mot terminé. Poursuivre la saisie.
4. Lorsque la saisie est terminée, appuyer sur **Entrée** ou cliquer n'importe où dans l'éditeur de formules.

Ne rien saisir dans la colonne Nom de la ligne de commentaire.

1.5.2 Saisie de formules

Les formules sont des règles créant de nouvelles formes d'onde, valeurs numériques ou chaînes à partir de sources de données, de variables et de constantes existantes. Une formule est une expression mathématique pouvant contenir des sources de données, des constantes et des fonctions. Vous pouvez utiliser des parenthèses pour modifier la priorité des opérateurs.

Les formules ont un nom, un corps et des unités facultatives. Elles apparaissent en bleu (mots-clés connus) et en noir.

Pour saisir une formule :

Procéder comme suit :

1 Saisir un nom pour la formule.
- **a** Cliquer une fois dans le champ Nom d'une ligne vierge pour le sélectionner, ou double-cliquer pour passer en mode d'édition.
- **b** Saisir un nom descriptif.

2 Passer au champ du corps de la formule comme suit :
- Appuyer sur la touche TAB.
- Appuyer sur la touche Flèche droite.

3 Saisir le corps de la formule en utilisant une ou plusieurs des techniques suivantes :
- Saisir le signe @ pour afficher la liste des fonctions disponibles et sélectionner la fonction voulue.
- Saisir un premier caractère : une liste déroulante répertoriant les racines des sources de données possibles apparaît. Sélectionner une racine puis saisir un point : une nouvelle liste déroulante répertoriant les branches de la racine sélectionnée apparaît. Sélectionner une branche puis saisir un point, etc. Répéter l'opération jusqu'à atteindre la variable ou la source de données voulue.
- Saisir directement le chemin d'accès complet de la variable/source de données.
- Utiliser des opérateurs et des parenthèses pour créer des formules plus complexes.
- Si disponibles, des informations d'aide apparaissent dans la ou les zones d'aide.

4 Une fois la saisie de la formule terminée, passer au champs Unité et saisir les unités le cas échéant.

CONSEIL

Il est également possible de faire glisser des sources de données directement dans une formule de la base de données des formules. Cela vous permet d'insérer rapidement des constantes et variables dans une fonction sans avoir à connaître le chemin complet de cette variable. Faire simplement glisser la position X d'un curseur dans la formule sans entrer le chemin complet, tel que : Display.Display1.Cursor1.XPosition.

1.6 Menu Formule

Le menu Formule répertorie les commandes relatives à la gestion des fichiers de formule. Pour la mise en page et la gestion du contenu, utiliser les outils disponibles dans la bande de tâches, à gauche de l'éditeur de formules.

Le menu Formule est un menu dynamique disponible uniquement lorsque la feuille Formule est au premier plan, c'est-à-dire affichée.

Ce menu permet d'enregistrer les formules dans un fichier distinct. En général, les réglages de la base de données de formules :

- comportent toutes les formules/fonctions spécifiées dans la feuille des formules ;
- peuvent être enregistrés dans un fichier distinct portant l'extension **.pFormulas** ;
- sont stockés automatiquement lorsqu'un environnement est enregistré dans le cadre d'un enregistrement ;
- sont chargés automatiquement avec un environnement complet ;
- peuvent être extraits/chargés depuis un environnement ou enregistrement comme réglages distincts ;
- peuvent être enregistrés dans un environnement ou enregistrement comme réglages distincts.

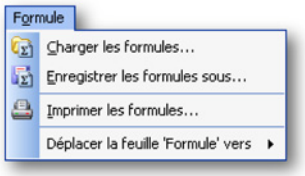


Figure 1.6 : Menu Formule

1.6.1 Chargement de formules

Vous pouvez charger des formules depuis diverses sources.

Pour charger des formules :

Pour charger des formules depuis une source externe, procéder de la manière suivante :

1 Effectuer l'une des opérations suivantes :
- Dans le **menu Formule**, cliquer sur **Charger les formules...**.
- S'il est disponible dans la **barre d'outils**, cliquer sur le bouton **Charger les formules...**  .

2 Si nécessaire, sélectionner le type de fichier dans la boîte de dialogue Charger les formules qui apparaît :
- Fichier de formule (*.pFormulas) ;
- Formules hors d'un environnement virtuel (*.pVWB) ;
- Formules hors d'une expérimentation (*.pNRF).

3 Sélectionner le fichier à utiliser.

4 Cliquer sur **Ouvrir**.

1.6.2 Enregistrement de formules

De la même manière que vous pouvez charger des formules, il est également possible d'enregistrer des formules. Vous pouvez également enregistrer dans une expérimentation ou un environnement virtuel existant. Vous remplacez ainsi les formules présentes dans ce fichier. Aucune autre donnée ne sera modifiée.

Pour enregistrer des formules :

Pour enregistrer des formules dans un fichier externe, procéder de la manière suivante :

1 Effectuer l'une des opérations suivantes :
- Dans le **menu Formule**, cliquer sur **Enregistrer les formules sous...**.
- S'il est disponible dans la **barre d'outils**, cliquer sur le bouton **Enregistrer les formules sous...** 

2 Si nécessaire, sélectionner le type de fichier dans la boîte de dialogue **Enregistrer les formules sous** qui apparaît :
- Fichier de formule (*.pFormulas) ;
- Formules hors d'un environnement virtuel (*.pVWB) ;
- Formules hors d'une expérimentation (*.pNRF).

3 Sélectionner le fichier à utiliser pour l'enregistrement ou à remplacer, ou saisir un nom pour un nouveau fichier.

4 Cliquer sur **Enregistrer**.

1.6.3 Impression de formules

Vous pouvez imprimer une copie des formules.

Pour imprimer une copie des formules :

1 Effectuer l'une des opérations suivantes :
- Dans le **menu Formule**, cliquer sur **Imprimer les formules...**.
- S'il est disponible dans la **barre d'outils**, cliquer sur le bouton **Imprimer les formules...**.

2 Dans la boîte de dialogue Imprimer qui apparaît, définir les préférences.
3 Cliquer sur **Imprimer**.

1.6.4 Déplacement de la feuille

Lorsque l'option permettant de créer plusieurs classeurs est installée, vous pouvez déplacer la feuille Formule vers un autre classeur.

Pour déplacer la feuille Formule vers un autre classeur :

1 Dans le **menu Formules**, placer le pointeur sur **Déplacer la feuille 'Formule' vers ►**
2 Dans le sous-menu qui apparaît, sélectionner un classeur.

2 Fonctions de base de données de formules

2.1 Généralités

Le présent document décrit en détail toutes les fonctions disponibles dans la base de données de formules Perception.

Vous trouverez une description de la base de données de formules en elle-même dans la section correspondante du manuel d'utilisation.

Pour de plus amples détails sur les caractéristiques, la mesure et l'analyse des impulsions, voir le chapitre :
« Mesure des impulsions et analyse » page 194.

Les formules présentes dans la base de données de formules sont définies comme suit :

sortie = formule

La sortie est disponible sous forme de variable dans le navigateur de sources de données dans la branche Formule.

La sortie peut être utilisée comme paramètre dans une autre fonction, quel que soit l'ordre physique dans la base de données.

Les formules peuvent être saisies comme sur le papier. Les règles mathématiques habituelles s'appliquent.

Exception
Aucune expression ne peut être utilisée comme paramètre d'une fonction « @ ».

Exemple :

```
          Angle      = 33
Correct   AngleRad   = System.Constants.Pi *
                       Formula.Angle / 180
          CosAngle   = @Cos(Formula.AngleRad)

Incorrect CosAngle   = @Cos(System.Constants.Pi *
                       Formula.Angle / 180)
```

Remarque *Toutes les fonctions peuvent être utilisées avec des données statiques à transitoire unique et à une seule base de temps. Elles peuvent également l'être avec des données à plusieurs bases de temps et/ou transitoires. Toutefois, les résultats risquent d'être imprévisibles en raison de la nature de la fonction utilisée. Certaines fonctions peuvent également être utilisées avec des données dynamiques (en temps réel). Le cas échéant, cela est indiqué dans la description de la fonction.*

2.2 Vue d'ensemble

Cette section fournit une vue d'ensemble de toutes les fonctions par ordre alphabétique incluant leur nom, une brève description et le numéro de page.

2.2.1 Vue d'ensemble des fonctions

Nom	Description
+ (Addition)	Additionner deux expressions : << + (Addition) >> page 33
- (Soustraction)	Soustraire deux expressions : << - (Soustraction) >> page 35
*** (Multiplication)**	Multiplier deux expressions : << * (Multiplication) >> page 37
/ (Division)	Diviser deux expressions : << / (Division) >> page 39
– (Inversion)	Inverser une expression : << – (Moins unaire) >> page 41
@Abs	Valeur absolue d'une valeur numérique ou d'une forme d'onde : << @Abs >> page 42
@And	Opérateur logique AND : << @And >> page 43
@Area	Surface sous la courbe d'une forme d'onde : << @Area >> page 44
@ATan	Calculer l'arc tangente : << @ATan >> page 46
@BlockFFT	Calculer la fréquence principale dans un bloc de données : << @BlockFFT >> page 47
@Clip	Écrêter l'amplitude d'une forme d'onde : << @Clip >> page 49
@Cos	Calculer le cosinus : << @Cos >> page 51
@CurveFitting	Ajuster une forme d'onde linéaire ou parabolique à une forme d'onde donnée : << @CurveFitting >> page 52
@Cut	Couper un segment spécifique d'une forme d'onde : << @Cut >> page 54
@Cycles	Compter le nombre de cycles dans une forme d'onde : << @Cycles >> page 56
@Diff	Différencier une forme d'onde : << @Diff >> page 58
@Energy	Calculer l'énergie sous la courbe d'une forme d'onde : << @Energy >> page 60
@EqualTo	Évaluation « égal à » : << @EqualTo >> page 62
@Exp	Exponentielle : calculer la puissance (base e) de l'entrée : << @Exp >> page 63

Nom	Description
@ExpWave	Générer une forme d'onde exponentielle : << @ExpWave >> page 64
@FallTime	Déterminer le temps de descente d'une impulsion dans une forme d'onde : << @FallTime >> page 66
@FilterButterworthLP	Filtre le signal d'entrée : << @FilterButterworthLP >> page 68
@FilterButterworthHP	Filtre le signal d'entrée : << @FilterButterworthHP >> page 70
@FilterButterworthBP	Filtre le signal d'entrée : << @FilterButterworthBP >> page 72
@FilterButterworthBS	Filtre le signal d'entrée : << @FilterButterworthBS >> page 74
@FilterBesselLP	Filtre le signal d'entrée : << @FilterBesselLP >> page 76
@FilterBesselHP	Filtre le signal d'entrée : << @FilterBesselHP >> page 78
@FilterBesselBP	Filtre le signal d'entrée : << @FilterBesselBP >> page 80
@FilterBesselBS	Filtre le signal d'entrée : << @FilterBesselBS >> page 82
@FilterChebyshevLP	Filtre le signal d'entrée : << @FilterChebyshevLP >> page 84
@FilterChebyshevHP	Filtre le signal d'entrée : << @FilterChebyshevHP >> page 87
@FilterChebyshevBP	Filtre le signal d'entrée : << @FilterChebyshevBP >> page 89
@FilterChebyshevBS	Filtre le signal d'entrée : << @FilterChebyshevBS >> page 92
@Frequency	Déterminer la fréquence d'une forme d'onde : << @Frequency >> page 95
@GreaterEqualThan	Évaluation « supérieur ou égal à » : << @GreaterEqualThan >> page 97
@GreaterThan	Évaluation « supérieur à » : << @GreaterThan >> page 98
@Histogram	Calculer l'histogramme d'amplitude : << @Histogram >> page 99
@IIF	Résultat conditionnel : << @IIF >> page 101
@Integrate	Intégrer une forme d'onde : << @Integrate >> page 103

Nom	Description
@IntLookUp	Modifier une forme d'onde en utilisant ses données comme index vers une table de conversion externe : << @IntLookUp >> page 104
@IntLookUp12	Modifier une forme d'onde en utilisant ses données comme index vers une table de conversion. Optimisée pour les données 12 bits : << @IntLookUp12 >> page 107
@Join	Concaténer deux formes d'onde ou plus : << @Join >> page 109
@Length	Renvoyer la longueur (en échantillons) d'une forme d'onde : << @Length >> page 111
@LessEqualThan	Évaluation « inférieur ou égal à » : << @LessEqualThan >> page 112
@LessThan	Évaluation « inférieur à » : << @LessThan >> page 113
@Ln	Calculer le logarithme népérien : << @Ln >> page 114
@Log	Calculer le logarithme de base 10 : << @Log >> page 116
@Max	Déterminer la valeur maximale (amplitude) d'une forme d'onde : << @Max >> page 117
@MaxNum	Déterminer la valeur maximale d'une plage de valeurs numériques : << @MaxNum >> page 119
@MaxPos	Renvoyer la position de la valeur maximale d'une forme d'onde : << @MaxPos >> page 120
@Mean	Calculer la valeur moyenne d'une forme d'onde : << @Mean >> page 122
@Min	Déterminer la valeur minimale (amplitude) d'une forme d'onde : << @Min >> page 124
@MinNum	Renvoyer la position de la valeur minimale d'une forme d'onde : << @MinNum >> page 126
@MinPos	Renvoyer la position (dans le temps) de la valeur minimale absolue : << @MinPos >> page 127
@NextHillPos	Déterminer la position de la valeur maximale locale suivante dans une forme d'onde : << @NextHillPos >> page 129
@NextLvlCross	Déterminer la position du croisement suivant d'une forme d'onde avec un niveau de signal donné : << @NextLvlCross >> page 131

Nom	Description
@NextValleyPos	Déterminer la position de la valeur minimale locale suivante dans une forme d'onde : << @NextValleyPos >> page 133
@Noise	Générer une forme d'onde contenant du bruit : << @Noise >> page 135
@Not	Opérateur logique NOT : << @Not >> page 136
@Or	Opérateur logique OR : << @Or >> page 137
@Period	Déterminer la période d'une forme d'onde : << @Period >> page 138
@Pow	Élévation à une puissance, base élevée à l'exposant de puissance : << @Pow >> page 140
@PrevHillPos	Déterminer la position de la valeur maximale locale précédente dans une forme d'onde : << @PrevHillPos >> page 141
@PrevLvlCross	Déterminer la position du croisement précédent d'une forme d'onde avec un niveau de signal donné : << @PrevLvlCross >> page 143
@PrevValleyPos	Déterminer la position de la valeur minimale locale précédente dans une forme d'onde : << @PrevValleyPos >> page 145
@PulseWidth	Déterminer la largeur d'une impulsion dans une forme d'onde : << @PulseWidth >> page 149
@Ramp	Générer une forme d'onde à rampe linéaire : << @Ramp >> page 151
@ReadAsciiFile	Lire les données d'une forme d'onde depuis un fichier ASCII (texte) : << @ReadAsciiFile >> page 153
@Reduce	Réduire le nombre d'échantillons d'une forme d'ondes par rééchantillonnage : << @Reduce >> page 155
@RefCheck	Contrôler une forme d'onde par rapport à une ou deux enveloppes de forme d'onde : << @RefCheck >> page 156
@RemoveGlitch	Supprimer les échantillons indésirables d'une forme d'onde : << @RemoveGlitch >> page 158
@Res2	Échantillonner une forme d'onde de sorte que sa longueur devienne une puissance de deux : << @Res2 >> page 159
@RiseTime	Déterminer le temps de montée d'une impulsion dans une forme d'onde : << @RiseTime >> page 161

Nom	Description
@RMS	Calculer la valeur moyenne quadratique d'une forme d'onde : << @RMS >> page 163
@SAEJ211Filter	Filtrer une forme d'onde selon les recommandations SAE J211 : << @SAEJ211Filter >> page 165
@Sin	Calculer le sinus : << @Sin >> page 166
@SineWave	Générer une onde sinusoïdale : << @SineWave >> page 167
@Smooth	Lisser une forme d'onde selon un nombre d'échantillons donné : << @Smooth >> page 169
@Sqrt	Calculer la racine carrée : << @Sqrt >> page 171
@SquareWave	Générer une onde carrée : << @SquareWave >> page 172
@StdDev	Calculer l'écart type d'une forme d'onde : << @StdDev >> page 173
@Sweep	Sélectionner un transitoire dans un enregistrement à plusieurs transitoires : << @Sweep >> page 175
@Tan	Calculer la tangente : << @Tan >> page 176
@TriggerTime	Renvoyer la position du trigger : << @TriggerTime >> page 177
@TriggerTimetoText	Renvoyer la position du trigger dans une chaîne formatée contenant la date et l'heure : << @TriggerTimeToText >> page 179
@TrueRMS	Calculer la valeur moyenne quadratique : << @TrueRMS >> page 182
@Value	Renvoyer la valeur de l'amplitude d'une forme d'onde à une position X donnée : << @Value >> page 184
@XDelta	Renvoyer l'intervalle d'échantillonnage d'une forme d'onde : << @XDelta >> page 185
@XDeltaHigh	Renvoyer l'intervalle d'échantillonnage maximal dans un enregistrement à plusieurs bases de temps : << @XDeltaHigh >> page 186
@XDeltaLow	Renvoyer l'intervalle d'échantillonnage minimal dans un enregistrement à plusieurs bases de temps : << @XDeltaLow >> page 187
@XFirst	Renvoyer la coordonnée X (par rapport au point de trigger) du premier échantillon d'une forme d'onde : << @XFirst >> page 188

Nom	Description
@XLast	Renvoyer la coordonnée X (par rapport au point de trigger) du dernier échantillon d'une forme d'onde : << @XLast >> page 189
@XShift	Décaler une forme d'onde dans le temps : << @XShift >> page 190
@XYArray	Créer une forme d'onde à partir d'une liste de paires de valeurs X/Y : << @XYArray >> page 191
@YArray	Créer une forme d'onde à partir d'une liste de valeurs Y : << @YArray >> page 193

3 Opérations arithmétiques

3.1 + (Addition)

Fonction
Additionne les expressions de gauche et de droite.

Syntaxe
Expression1 + *Expression2*

Paramètres

Expression1	Expression de gauche.
Expression2	Expression de droite.

Sortie
Le résultat correspond à la somme des expressions de gauche et de droite.

Description
Une expression peut être :

- une variable de forme d'onde ;
- un appel de fonction ;
- une variable numérique ;
- une valeur constante.

La priorité normale des opérateurs (multiplication et division, puis addition et soustraction) s'applique. Des parenthèses peuvent être utilisées pour modifier la priorité des opérateurs dans les expressions plus complexes.

Exemple
Voici quelques exemples d'expressions valides pour une addition :

- 2 + 3
- var1 + 4
- 5 + var2
- 1000 + @Noise(1E6; 1000)
- (Var1 + Var2) * (Var3 - Var4)

Lors de l'addition de valeurs numériques, la sortie est une valeur numérique.

Lors de l'addition d'une forme d'onde et d'une valeur numérique, la sortie est une forme d'onde. La longueur (nombre de points) de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée.

Lors de l'addition de deux formes d'onde, la sortie est une forme d'onde obtenue à partir de l'addition point par point des deux formes d'onde d'entrée. L'échelle X de la forme d'onde de sortie est identique à celle de la forme d'onde de gauche. La longueur de la forme d'onde de sortie est identique à celle de la plus courte des deux formes d'onde d'entrée. Pour obtenir un résultat significatif, les échelles X des deux formes d'onde doivent être identiques.

Voir aussi

<< * (Multiplication) >> page 37, << - (Soustraction) >> page 35, << / (Division) >> page 39

3.2 - (Soustraction)

Fonction
Soustrait les expressions de gauche et de droite.

Syntaxe
Expression1 – Expression2

Paramètres

Expression1	Expression de gauche.
Expression2	Expression de droite.

Sortie
Le résultat correspond à la différence des expressions de gauche et de droite.

Description

Une expression peut être :

- une variable de forme d'onde ;
- un appel de fonction ;
- une variable numérique ;
- une valeur constante.

La priorité normale des opérateurs (multiplication et division, puis addition et soustraction) s'applique. Des parenthèses peuvent être utilisées pour modifier la priorité des opérateurs dans les expressions plus complexes.

Exemple

Voici quelques exemples d'expressions valides pour une soustraction :

- 3 - 2
- var1 - 4
- 5 - var2
- @SineWave(1E6; 1000; 1k) - 1
- (Var1 + Var2) * (Var3 - Var4)

Lors de la soustraction de valeurs numériques, la sortie est une valeur numérique.

Lors de la soustraction d'une forme d'onde et d'une valeur numérique, la sortie est une forme d'onde. La longueur (nombre de points) de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée.

Lors de la soustraction de deux formes d'onde, la sortie est une forme d'onde obtenue à partir de la soustraction point par point des deux formes d'onde d'entrée. L'échelle X de la forme d'onde de sortie est identique à celle de la forme d'onde de gauche. La longueur de la forme d'onde de sortie est identique à celle de la plus courte des deux formes d'onde d'entrée. Pour obtenir un résultat significatif, les échelles X des deux formes d'onde doivent être identiques.

Voir aussi

<< * (Multiplication) >> page 37, << + (Addition) >> page 33, << / (Division) >> page 39

3.3 * (Multiplication)

Fonction
Multiplie les expressions de gauche et de droite.

Syntaxe
*Expression1 * Expression2*

Paramètres

Expression1	Expression de gauche.
Expression2	Expression de droite.

Sortie
Le résultat correspond au produit des expressions de gauche et de droite.

Description

Une expression peut être :

- une variable de forme d'onde ;
- un appel de fonction ;
- une variable numérique ;
- une valeur constante.

La priorité normale des opérateurs (multiplication et division, puis addition et soustraction) s'applique. Des parenthèses peuvent être utilisées pour modifier la priorité des opérateurs dans les expressions plus complexes.

Exemple

Voici quelques exemples d'expressions valides pour une multiplication :

- 2 * 3
- var1 * 4
- 5 * var2
- 10 * @SineWave(1E6; 1000; 1k)
- (Var1 - AvgVar1) * (Var2 - AvgVar2)

Lors de la multiplication de valeurs numériques, la sortie est une valeur numérique.

Lors de la multiplication d'une forme d'onde et d'une valeur numérique, la sortie est une forme d'onde mise à l'échelle. La longueur (nombre de points) de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée.

Lors de la multiplication de deux formes d'onde, la sortie est une forme d'onde obtenue à partir de la multiplication point par point des deux formes d'onde d'entrée. L'échelle X de la forme d'onde de sortie est identique à celle de la forme d'onde de gauche. La longueur de la forme d'onde de sortie est identique à celle de la plus courte des deux formes d'onde d'entrée. Pour obtenir un résultat significatif, les échelles X des deux formes d'onde doivent être identiques.

Voir aussi

<< + (Addition) >> page 33, << - (Soustraction) >> page 35, << / (Division) >> page 39

3.4 / (Division)

Fonction
Divise les expressions de gauche et de droite.

Syntaxe
Expression1 / *Expression2*

Paramètres

Expression1	Expression de gauche (dividende).
Expression2	Expression de droite (diviseur).

Sortie
Le résultat correspond au quotient des expressions de gauche et de droite.

Description
Une expression peut être :

- une variable de forme d'onde ;
- un appel de fonction ;
- une variable numérique ;
- une valeur constante.

La priorité normale des opérateurs (multiplication et division, puis addition et soustraction) s'applique. Des parenthèses peuvent être utilisées pour modifier la priorité des opérateurs dans les expressions plus complexes.

Exemple
Voici quelques exemples d'expressions valides pour une division :

- 3 / 2
- var1 / 4
- 5 / var2
- @SineWave(1E6; 1000; 1k) / 10
- (Var1 + Var2) / (Var3 - Var4)

Lors de la division de valeurs numériques, la sortie est une valeur numérique.

Lors de la division d'une forme d'onde et d'une valeur numérique, la sortie est une forme d'onde. La longueur (nombre de points) de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée.

Lors de la division de deux formes d'onde, la sortie est une forme d'onde obtenue à partir de la division point par point des deux formes d'onde d'entrée. L'échelle X de la forme d'onde de sortie est identique à celle de la forme d'onde de gauche. La longueur de la forme d'onde de sortie est identique à celle de la plus courte des deux formes d'onde d'entrée. Pour obtenir un résultat significatif, les échelles X des deux formes d'onde doivent être identiques.

Renvoie une valeur non définie (inconnue) lorsque le diviseur est une valeur numérique nulle ou lorsque le dénominateur est une forme d'onde contenant un échantillon dont la valeur est nulle.

Voir aussi

<< * (Multiplication) >> page 37, << + (Addition) >> page 33, << - (Soustraction) >> page 35

3.5 – (Moins unaire)

Fonction
Inverse le signe d'une expression.

Syntaxe
– *Expression*

Paramètres

Expression	Expression à inverser.

Sortie
Le résultat correspond à l'expression multipliée par -1 (moins un).

Description

L'expression peut être toute expression contenant :

- des variables de forme d'onde ;
- des appels de fonction ;
- des variables numériques ;
- des valeurs constantes.

La priorité normale des opérateurs (multiplication et division, puis addition et soustraction) s'applique. Des parenthèses peuvent être utilisées pour modifier la priorité des opérateurs dans les expressions plus complexes.

Exemple

Voici quelques exemples d'expressions valides utilisant le moins unaire :

- - 2
- - var1
- - @SineWave(1E6; 1000; 1k) - 1
- - ((Var1 + Var2) * (Var3 - Var4))

Lors de l'inversion d'une valeur numérique, la sortie est une valeur numérique. Lors de l'inversion d'une forme d'onde, la sortie est la forme d'onde inversée logiquement. La longueur (nombre de points) de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée.

Voir aussi
<< * (Multiplication) >> page 37, << + (Addition) >> page 33, << - (Soustraction) >> page 35, << / (Division) >> page 39

4 Guide de référence

4.1 @Abs

Fonction
Calcule la valeur **absolue** du paramètre.

Syntaxe
@Abs(Par)

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Valeur absolue de la forme d'onde ou de la valeur numérique.

Description
Calcule la valeur absolue de la forme d'onde ou de la valeur numérique d'entrée. Les valeurs positives restent telles quelles, les valeurs négatives changent de signe. Cette fonction peut être utilisée pour redresser les signaux ou imposer des résultats positifs.

Exemple
L'exemple suivant crée une onde sinusoïdale et redresse ce signal :

```
Signal  = @SineWave(1E6; 1000; 1k)
Rectif  = @Abs(Formula.Signal)
```

4.2 @And

Fonction
Évalue les paramètres d'entrée à l'aide de l'opérateur logique **AND**.

Syntaxe
@And*(Param1; …; ParamN)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation AND.
ParamN	Dernier paramètre utilisé pour l'évaluation AND. Avec N >= 2.

Sortie
La sortie est 1 ou 0.

Description
La fonction @And évalue les paramètres d'entrée à l'aide de l'opérateur logique AND. Selon cette évaluation, le résultat est 1 ou 0. Une valeur numérique non nulle correspond à un « Vrai » logique et une valeur numérique nulle à un « Faux » logique.

La table de définition de la fonction AND est la suivante :

Param1	Param2	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

Exemple
Voici une liste d'exemples avec la valeur renvoyée dans chaque cas :

```
AndExampl1 = @And(1; 1; 1)    => 1 (=true)
AndExampl2 = @And(1; 4; 10)   => 1 (=true)
AndExampl3 = @And(1; 4; 0)    => 0 (=false)
AndExampl3 = @And(0; 0; 0)    => 0 (=false)
```

Voir aussi
<< @Not >> page 136 et << @Or >> page 137

4.3 @Area

Fonction
Calcule la **surface** sous la courbe d'une forme d'onde.

Syntaxe
@Area*(Forme d'onde)*

@Area*(Forme d'onde; Début)*

@Area*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée pour laquelle la surface sous la courbe doit être calculée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Description
La surface sous la courbe est calculée à l'aide de la formule suivante :

$$\text{Area} = \left[\left[\sum_{n=n_1}^{n_2} y(n) \right] - \frac{y(n_1) + y(n_2)}{2} \right] \cdot \Delta x$$

n_1 = premier échantillon avec x ≥ *Début*

n_2 = dernier échantillon avec x ≤ *Fin*

Δx = x - différence entre deux échantillons

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres **Début** et **Fin** doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

L'intégration numérique de la courbe est réalisée en considérant que cette dernière est interpolée linéairement entre les échantillons.

Exemple
L'exemple suivant crée une onde sinusoïdale de 50 Hz et calcule la surface sous la courbe de la première demi-période du signal :

```
Signal = @SineWave(50k; 1000; 50)
Area   = @Area(Formula.Signal; 0; 10m)
```

Voir aussi
<< @Energy >> page 60 et << @Mean >> page 122

4.4 @ATan

Fonction
Calcule l'**arc tangente** du paramètre d'entrée.

Syntaxe
@ATan*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant l'arc tangente de l'entrée.

Description
La fonction arc tangente renvoie l'angle pour lequel la tangente est égale à l'argument. Cet angle est exprimé en radians. L'arc tangente est la fonction trigonométrique inverse de la tangente.

Exemple
L'exemple suivant calcule Pi en multipliant ATan(1) par quatre :

```
Pi  = 4 * @ATan(1)
```

Voir aussi
<< @Cos >> page 51, << @Sin >> page 166 et << @Tan >> page 176

4.5 @BlockFFT

Fonction
Renvoie une forme d'onde représentant la fréquence maximale détectée pour chaque **bloc** de la forme d'onde d'entrée.

Syntaxe
@BlockFFT*(Forme d'onde; Taille; Espace)*

@BlockFFT*(Forme d'onde; Taille; Espace; Début)*

@BlockFFT*(Forme d'onde; Taille; Espace; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée
Taille	Nombre : taille de bloc en millisecondes.
Espace	Nombre : espace entre les débuts de deux blocs successifs, en millisecondes.
Début	Nombre : position initiale de la fonction BlockFFT
Fin	Nombre : position finale de la fonction BlockFFT.

Sortie
Forme d'onde représentant la fréquence en fonction du temps.

Description
Cette fonction calcule la fréquence maximale de chaque bloc à l'aide d'un algorithme FFT. Le paramètre Taille définit la taille (longueur) des blocs en millisecondes. Le paramètre Espace définit l'espace entre les débuts de deux blocs successifs en millisecondes.

Le schéma ci-dessous illustre la relation entre l'espacement et les blocs.

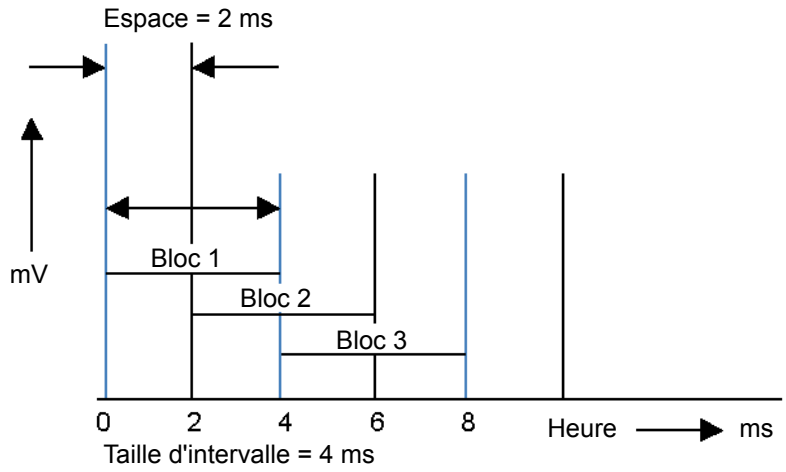


Figure 4.1 : Exemple - Relation entre l'espacement et la taille des blocs

La sortie de la fonction est une forme d'onde représentant la fréquence maximale de chaque bloc en fonction du temps. L'espacement des échantillons de cette forme d'onde de sortie est égal au paramètre Espace.

Les limites de segment (Début et Fin) sont utilisées pour sélectionner la plage de la forme d'onde dans laquelle les BlockFFT sont calculées. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Exemples
L'exemple suivant calcule la BlockFFT d'une forme d'onde composée :

```
Sine1  = 2 * @SineWave(1000k; 30k; 10k)
Sine2  = @SineWave(1000k; 30k; 15k)
Sine3  = 1.5 * @SineWave(1000k; 30k; 5k)
Signal = @Join(Formula.Sine1; Formula.Sine2;
         Formula.Sine3)
Result = @BlockFFT(Formula.Signal; 4; 2)
```

4.6 @Clip

Fonction
Écrête une forme d'onde entre les limites inférieure et supérieure définies.

Syntaxe
@Clip*(Forme d'onde; Limite inférieure; Limite supérieure)*

Paramètres

Forme d'onde	Forme d'onde dont l'amplitude doit être écrêtée.
Limite inférieure	Nombre : valeur de la limite inférieure utilisée pour l'écrêtage.
Limite supérieure	Nombre : valeur de la limite supérieure utilisée pour l'écrêtage.

Sortie
Forme d'onde contenant toutes les valeurs d'échantillon écrêtées entre les limites inférieure et supérieure.

Description
Chaque échantillon de la forme d'onde est comparé aux limites inférieure et supérieure de la plage d'écrêtage. Si la valeur de l'échantillon est comprise entre ces deux limites, elle n'est pas modifiée. Si elle est supérieure à la limite supérieure, elle est remplacée par cette dernière. Si elle est inférieure à la limite inférieure, elle est remplacée par cette dernière.

Exemple
L'exemple écrête l'onde sinusoïdale de 1,2 V entre les limites -1 et 1 pour simuler un dépassement à l'entrée.

```
Signal    = 1.2 * @SineWave(20k; 1000; 50)
InpSignal = @Clip(Formula.Signal; -1; 1)
```

L'exemple suivant détermine la durée cumulée pendant laquelle un signal d'accélération est supérieur à 150 g. La technique utilisée consiste à commencer par écrêter le signal entre 150 g et 0,001 g au-dessus de 150 g. La valeur 150 est soustraite du signal écrêté et ce dernier est augmenté d'un facteur de 1 000. Le résultat de cette technique est 0 lorsque l'accélération est inférieure à 150 g et 1 lorsqu'elle est supérieure à 150 g. La surface sous cette courbe correspond à la durée cumulée au-dessus de 150 g.

```
Accel   = 150 + @SineWave(20k; 100; 50)
```

```
Temp    = 1000 * (@Clip(Formula.Accel; 150; 150.001) -
          150)
CumTime = @Area(Formula.Temp)
```

Voir aussi

<< @Cut >> page 54

4.7 @Cos

Fonction
Calcule le **cosinus** du paramètre d'entrée.

Syntaxe
@Cos*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant le cosinus de l'entrée.

Description
La fonction trigonométrique cosinus est calculée en considérant que le paramètre d'entrée correspond à l'angle en radians. Lorsqu'un paramètre de forme d'onde est utilisé, le cosinus est calculé pour chacun des échantillons.

Exemple
L'exemple suivant calcule le cosinus de la variable « Angle » définie en degrés :

```
Angle     = 33
AngleRad  = System.Constants.Pi * Formula.Angle / 180
CosAngle  = @Cos(Formula.AngleRad)
```

Voir aussi
<< @ATan >> page 46, << @Sin >> page 166 et << @Tan >> page 176

4.8 @CurveFitting

Fonction
Renvoie la forme d'onde se rapprochant le plus d'une série de points de données du signal d'entrée ; utilise une interpolation linéaire ou une régression parabolique.

Syntaxe
@CurveFitting*(Forme d'onde)*

@CurveFitting*(Forme d'onde; Ordre)*

@CurveFitting*(Forme d'onde; Ordre; Début intervalle)*

@CurveFitting*(Forme d'onde; Ordre; Début intervalle; Fin intervalle)*

@CurveFitting*(Forme d'onde; Ordre; Début intervalle; Fin intervalle; Début)*

@CurveFitting*(Forme d'onde; Ordre; Début intervalle; Fin intervalle; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée
Ordre	Nombre : ordre de régression. 1 = linéaire, 2 = parabolique. L'ordre par défaut est une régression linéaire.
Début intervalle	Nombre : début de l'intervalle utilisé pour l'interpolation. Par défaut, le début de la forme d'onde d'entrée est utilisé.
EndIntv	Nombre : fin de l'intervalle utilisé pour l'interpolation. Par défaut, la fin de la forme d'onde d'entrée est utilisée.
Début	Nombre : heure de début de la courbe de sortie. Par défaut, l'heure de début de la forme d'onde d'entrée est utilisée.
Fin	Nombre : heure de fin de la courbe de sortie. Par défaut, l'heure de fin de la forme d'onde d'entrée est utilisée.

Sortie
Une forme d'onde correspondant à la régression linéaire ou parabolique de la forme d'onde d'entrée ou d'une partie de cette dernière.

Description
Cette fonction recherche la courbe se rapprochant le plus d'une série de points de données du signal d'entrée. Selon les paramètres, tous les points de données ou seulement un intervalle limité de points de données du signal d'entrée sont utilisés.

Si le paramètre Ordre est défini sur 1, une régression linéaire est effectuée : le signal de sortie est alors une ligne droite.

Si le paramètre Ordre est défini sur 2, une régression parabolique est effectuée : le signal de sortie est alors une parabole.

L'algorithme des moindres carrés est utilisé pour les calculs.

Exemple
Les exemples suivants produisent des adaptations linéaire et parabolique :

```
Signal       = @SineWave(8000; 8001; 5)
LineFit      = @CurveFitting(Formula.Signal; 1; 90m;
               110m)
ParabolicFit = @CurveFitting(Formula.Signal; 2; 125m;
               175m)
```

4.9 @Cut

Fonction
Coupe un segment spécifique d'une forme d'onde.

Syntaxe
@Cut*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont un segment doit être sélectionné.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
Segment de forme d'onde.

Description
Un segment spécifique d'une forme d'onde peut être sélectionné en vue d'un traitement plus poussé. Le fonction calcule le nombre d'échantillons de la sortie d'après les valeurs Début et Fin. Le premier échantillon est celui dont la coordonnée X est la plus proche du Début. Le dernier est celui dont la coordonnée X est la plus proche de la Fin.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons. Si les limites du segment se trouvent à l'extérieur de la plage X de la forme d'onde, ces valeurs sont limitées à cette plage.*

Exemple
L'exemple suivant coupe le segment d'un signal entre 100 ms et 200 ms :

```
Signal  = @SineWave(10000; 10000; 100)
Segment = @Cut(Formula.Signal; 100m; 200m)
```

Si une plage d'échantillons spécifique est requise, les fonctions d'informations peuvent être utilisées pour calculer la plage à sélectionner.

L'exemple suivant sélectionne les 1 024 premiers points du signal précédent et utilise -1E20 comme une très petite valeur :

```
XEnd      = @XBegin(Formula.Signal)
          + 1023 * @XDelta(Formula.Signal)
First1024 = @Cut(Formula.Signal; -1E20; Formula.XEnd)
```

Voir aussi

<< @Join >> page 109, << @Length >> page 111, << @XFirst >> page 188, << @XDelta >> page 185 et << @XLast >> page 189

4.10 @Cycles

Fonction
Calcule le nombre de **cycles** dans une forme d'onde ou un segment de forme d'onde.

Syntaxe
@Cycles*(Forme d'onde)*

@Cycles*(Forme d'onde; Début)*

@Cycles*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont le nombre de cycles doit être calculé.
Début	Nombre : début du segment
Fin	Nombre : fin du segment

Sortie
Nombre de cycles.

Description
La fonction @Cycles calcule le nombre d'occurrences d'une séquence d'échantillons qui se répète périodiquement (= cycle) dans la forme d'onde. Le niveau 50 % entre les valeurs maximale et minimale de l'amplitude est utilisé pour compter le nombre de franchissements de niveau et ainsi obtenir le nombre de cycles.

Les limites de segment (Début et Fin) sont utilisées pour sélectionner la plage d'échantillons dans laquelle les cycles sont calculés. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant calcule le nombre de cycles d'un signal complet et entre deux curseurs positionnés dans un affichage appelé « display ».

```
Signal  = @SineWave(10k; 10001; 50)
Cycles  = @Cycles(Formula.Signal)

Start      = Display.Display.Cursor1.XPosition
End        = Display.Display.Cursor2.XPosition
Cycles_BC  = @Cycles(Formula.Signal;
           Formula.Start; Formula.End)
```

Voir aussi

<< @Frequency >> page 95 et << @Period >> page 138

4.11 @Diff

Fonction
Différencie une forme d'onde.

Syntaxe
@Diff(Forme d'onde)

Paramètres

Forme d'onde Forme d'onde à différencier.

Sortie
Forme d'onde différentiée.

Description
La dérivée est une mesure de l'évolution d'une fonction lorsque les valeurs de ses entrées changent. Sans entrer dans le détail, une dérivée peut être vue comme l'amplitude d'une variation en un point donné. Le processus de calcul d'une dérivée est appelé différentiation. En d'autres termes, la différentiation est utilisée pour déterminer la pente d'un signal au lieu de sa valeur. La pente est obtenue en calculant la différence entre des échantillons adjacents et en divisant cette différence par l'intervalle d'échantillonnage :

$$\mathrm{Diff(n)} = \frac{y(n)-y(n-1)}{\Delta x} \quad \text{for n} = 2,...,\mathrm{N}$$

$$\mathrm{Diff}(1) = 0$$

Δx = x - différence entre deux échantillons

N = nombre d'échantillons

La différentiation met en valeur les composantes haute fréquence telles que le bruit haute fréquence et les erreurs de numérisation dues aux arrondissements. Pour une meilleure estimation de la pente, il est recommandé de lisser la forme d'onde (avant ou après la différentiation).

Exemple
L'exemple suivant crée une onde sinusoïdale affectée par du bruit et différentie ce signal. La forme d'onde obtenue est lissée pour fournir une estimation plus précise de la pente de l'onde sinusoïdale.

```
Signal  = @SineWave(10k; 1000; 50)
Differ  = @Diff(Formula.Signal)
Slope   = @Smooth(Formula.Differ; 7)
```

Voir aussi

<< @Integrate >> page 103 et << @Smooth >> page 169

4.12 @Energy

Fonction
Calcule l'**énergie** sous la courbe d'une forme d'onde.

Syntaxe
@Energy*(Forme d'onde)*

@Energy*(Forme d'onde; Début)*

@Energy*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée pour laquelle l'énergie sous la courbe doit être calculée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Description
L'énergie sous la courbe est calculée à l'aide de la formule suivante :

$$\text{Energy} = \left[\left[\sum_{n=n_1}^{n_2} y^2(n) \right] - \frac{y^2(n_1) + y^2(n_2)}{2} \right] \cdot \Delta x$$

n_1 = premier échantillon avec x ≥ *Début*

n_2 = dernier échantillon avec x ≤*Fin*

Δx = x - différence entre deux échantillons

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé. L'intégration numérique est réalisée en considérant que la courbe carrée est interpolée linéairement entre les échantillons.

Exemple
L'exemple suivant crée une onde sinusoïdale de 50 Hz et calcule l'énergie dans une période du signal :

```
Signal  = @SineWave(20k; 1000; 50)
E1      = @Energy(Formula.Signal; 0; 20m)
```

Voir aussi
<< @Area >> page 44 et << @RMS >> page 163

4.13 @EqualTo

Fonction
Cette fonction effectue une évaluation **égal à** (=) sur les deux paramètres numériques d'entrée.

Syntaxe
@EqualTo*(Param1; Param2)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation
Param2	Nombre : second paramètre utilisé pour l'évaluation

Sortie
La sortie est 1 ou 0

Description
La fonction EqualTo effectue une évaluation « égal à » sur les paramètres d'entrée. Si Param 1 = Param 2, la valeur renvoyée est 1 (vrai), sinon la valeur renvoyée est 0 (faux).

La fonction EqualTo est généralement utilisée en conjonction avec la fonction IIF.

Exemple
L'exemple suivant compare les paramètres d'entrée et fournit un résultat dépendant de l'issue de cette évaluation :

```
EqualToExampl1  = @EqualTo(5; 5)  => 1 (true)
EqualToExampl2  = @EqualTo(12;    => 0 (false)
                  10)
IIFExample      = @IIF(Formula.EqualToExampl2; "TRUE";
                  "FALSE")
```

Voir aussi
<< @IIF >> page 101, << @GreaterEqualThan >> page 97, << @GreaterThan >> page 98, << @LessEqualThan >> page 112 et << @LessThan >> page 113

4.14 @Exp

Fonction
Fonction exponentielle ; cette opération mathématique, notée e^n, nécessite deux paramètres : la base « e » (2,7...) et l'exposant « n ».

Syntaxe
@Exp*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant l'élévation à la puissance (base e) de l'entrée.

Description
La fonction exponentielle calcule la puissance (base e) de l'entrée. Lorsqu'un paramètre de forme d'onde est utilisé, la fonction exponentielle est calculée pour chacun des échantillons. Il s'agit de la fonction inverse de @ln.

Exemple
La fonction exponentielle peut également être calculée à l'aide de la fonction @Pow. Les formules suivantes sont équivalentes (utilisent une variable fictive de la base de données de formules nommée Input) :

```
Result = @Exp(Formula.Input)
Result = @Pow(System.Constants.e; Formula.Input)
```

L'exemple suivant présente une alternative à la variable système System.Constants.e :

```
Euler = @Exp(1)
```

Voir aussi
<< @ExpWave >> page 64, << @Ln >> page 114 et << @Pow >> page 140

4.15 @ExpWave

Fonction
Génère une forme d'onde contenant une fonction **exponentielle**.

Syntaxe
@ExpWave*(Taux; Comptage; Alpha; Bêta)*

@ExpWave*(Taux; Comptage; Alpha; Bêta; X0)*

Paramètres

Taux	Nombre : fréquence d'échantillonnage
Comptage	Nombre d'échantillons
Alpha	Nombre : multiplicateur d'amplitude
Bêta	Nombre : multiplicateur d'exposant
X0	Nombre : modificateur de multiplicateur d'exposant

Sortie
La sortie est une forme d'onde contenant un signal croissant de façon exponentielle.

Description

Cette fonction génère une forme d'onde à l'aide de la formule suivante :

$$f\,(x) = \alpha \cdot e^{(\beta \cdot (x - x_0))}$$

La fréquence d'échantillonnage, le nombre d'échantillons, le taux d'accroissement et le facteur de multiplication peuvent être définis. La longueur de la forme d'onde générée est limitée à 1 Giga-échantillons (1 000 000 000).

Le logarithme népérien ln(x) est la fonction inverse de la fonction exponentielle.

La possibilité de générer une fonction d'onde exponentielle peut être exploitée pour synthétiser diverses formes d'onde. Les données simulées peuvent être utilisées comme entrées pour d'autres fonctions d'analyse.

Exemple
L'exemple suivant crée une forme d'onde exponentielle.

```
Signal = @ExpWave(1; 100; 2; 100m)
```

Voir aussi

<< @Ln >> page 114, << @Pulse >> page 147, << @SineWave >> page 167 et << @SquareWave >> page 172

4.16 @FallTime

Fonction
Détermine le **temps de descente** d'une impulsion dans une forme d'onde.

Syntaxe
@FallTime*(Forme d'onde)*

@FallTime*(Forme d'onde; Début)*

@FallTime*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée contenant le front descendant d'une impulsion.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique. Cette fonction s'applique « pendant l'enregistrement » lorsque les paramètres Début et Fin sont définis. Le résultat est calculé une fois les données entre le Début et la Fin disponibles.

Description
Le temps de descente correspond à la différence de temps entre le point distal (passage à 90 % de l'amplitude) et le point proximal (passage à 10 % de l'amplitude) du premier front descendant d'une impulsion dans la forme d'onde (ou dans un segment de forme d'onde).

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Exemple
L'exemple suivant crée une impulsion et calcule le temps de descente de 90 % à 10 % du front descendant. Le résultat est 40 ms :

```
Sig1     = @Ramp(1k; 100; 0; 0)
Sig2     = @Ramp(1k; 51; 0; 10)
Sig3     = @Ramp(1k; 100; 10; 10)
```

```
Sig4     = @Ramp(1k; 51; 10; 0)
Signal   = @Join(Formula.Sig1; Formula.Sig2;
           Formula.Sig3; Formula.Sig4; Formula.Sig1)
Falltime = @FallTime(Formula.Signal)
```

Voir aussi

<< @PulseWidth >> page 149 et << @RiseTime >> page 161

4.17 @FilterButterworthLP

Fonction
Filtre le signal d'entrée avec un filtre Butterworth passe-bas d'IIR de forme directe.

Syntaxe
@FilterButterworthLP*(Signal ; Ordre ; Fc)*

@FilterButterworthLP*(Signal ; Ordre ; Fc ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Sans phase	Numéro : méthode de filtrage
	0 Le filtre ne fonctionne pas sans phase (par défaut)
	1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Butterworth passe-bas d'IIR de forme directe.

La fréquence de coupure est la fréquence à laquelle la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterButterworthLP(Formula.Signal; 2; 200)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

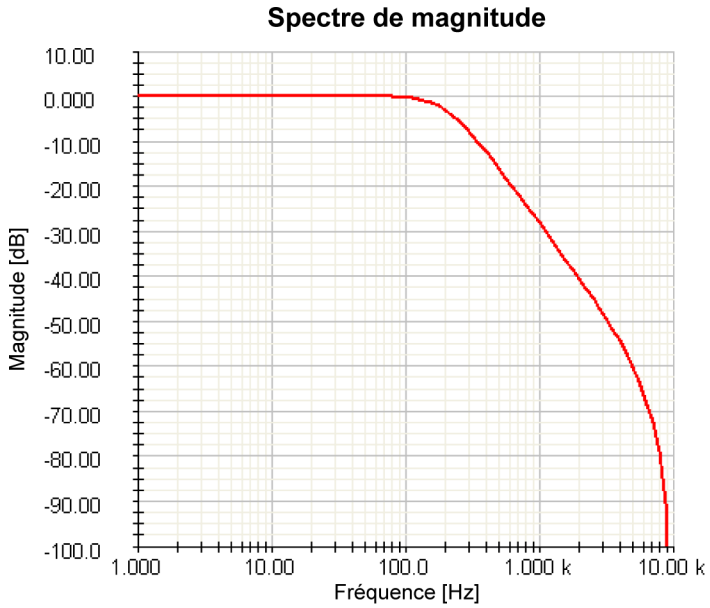


Figure 4.2 : Spectre de magnitude - FilterButterworthLP

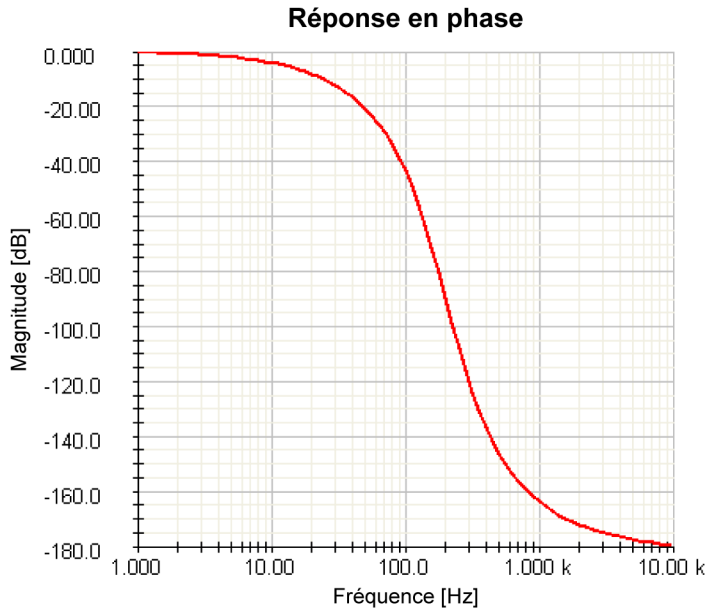


Figure 4.3 : Réponse en phase - FilterButterworthLP

Voir aussi

<< @FilterButterworthBS >> page 74, << @FilterButterworthBP >> page 72 et << @FilterButterworthHP >> page 70

4.18 @FilterButterworthHP

Fonction
Filtre le signal d'entrée avec un filtre Butterworth passe-haut d'IIR de forme directe.

Syntaxe
@FilterButterworthHP*(Signal ; Ordre ; Fc)*

@FilterButterworthHP*(Signal ; Ordre ; Fc ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Sans phase	Numéro : méthode de filtrage
	0 Le filtre ne fonctionne pas sans phase (par défaut)
	1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Butterworth passe-haut d'IIR de forme directe.

La fréquence de coupure est la fréquence à laquelle la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterButterworthHP(Formula.Signal; 2; 200)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

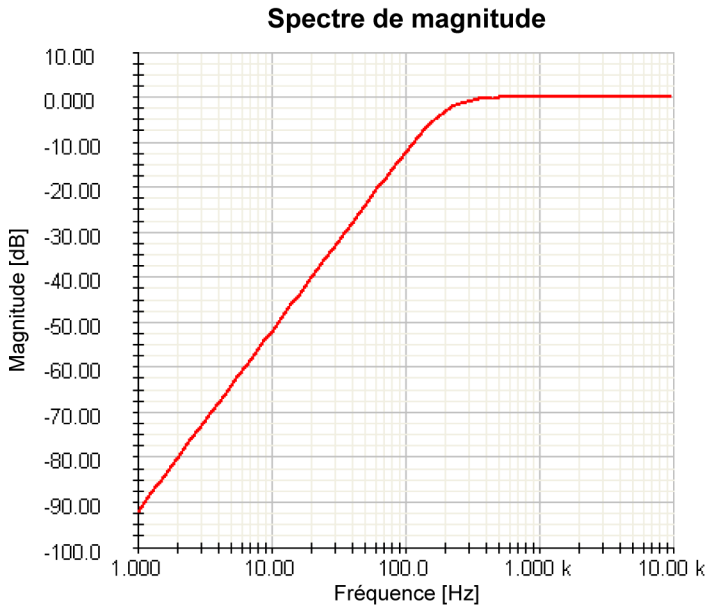


Figure 4.4 : Spectre de magnitude - FilterButterworthHP

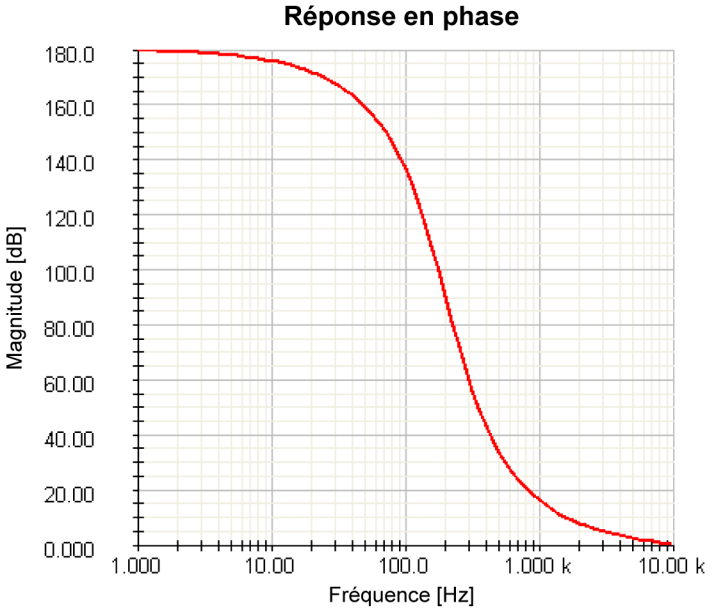


Figure 4.5 : Réponse en phase - FilterButterworthHP

Voir aussi

<< @FilterButterworthBS >> page 74, << @FilterButterworthBP >> page 72 et << @FilterButterworthLP >> page 68

4.19 @FilterButterworthBP

Fonction
Filtre le signal d'entrée avec un filtre Butterworth passe-bande d'IIR de forme directe.

Syntaxe
@FilterButterworthBP*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterButterworthBP*(Signal ; Ordre ; Fc basse ; Fc haute ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Butterworth passe-bande d'IIR de forme directe.

Les fréquences de coupure sont les fréquences auxquelles la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterButterworthBP(Formula.Signal; 2; 200;
           1000)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

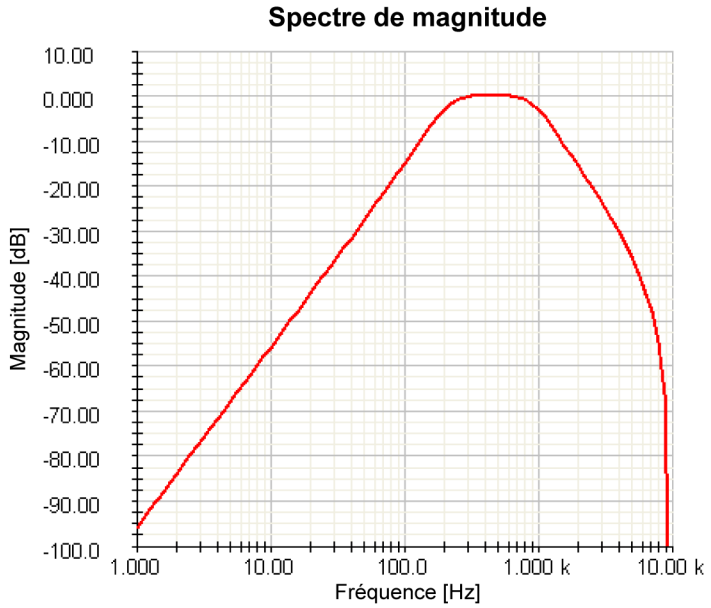


Figure 4.6 : Spectre de magnitude - FilterButterworthBP

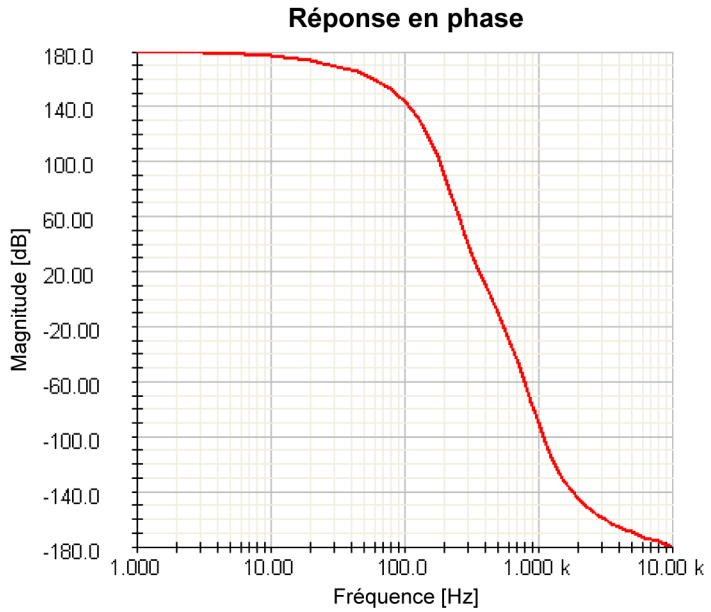


Figure 4.7 : Réponse en phase - FilterButterworthBP

Voir aussi

<< @FilterButterworthBS >> page 74, << @FilterButterworthHP >> page 70 et << @FilterButterworthLP >> page 68

4.20 @FilterButterworthBS

Fonction
Filtre le signal d'entrée avec un filtre Butterworth bloqueur de bande d'IIR de forme directe.

Syntaxe
@FilterButterworthBS*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterButterworthBS*(Signal ; Ordre ; Fc basse ; Fc haute ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Sans phase	Numéro : méthode de filtrage
	0 Le filtre ne fonctionne pas sans phase (par défaut)
	1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Butterworth bloqueur de bande d'IIR de forme directe.

Les fréquences de coupure sont les fréquences auxquelles la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterButterworthBS(Formula.Signal; 2; 200;
           1000)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

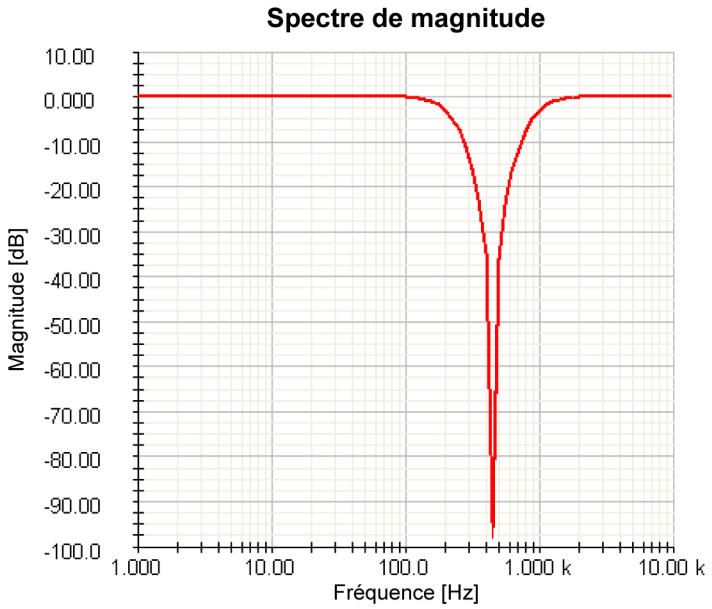


Figure 4.8 : Spectre de magnitude - FilterButterworthBS

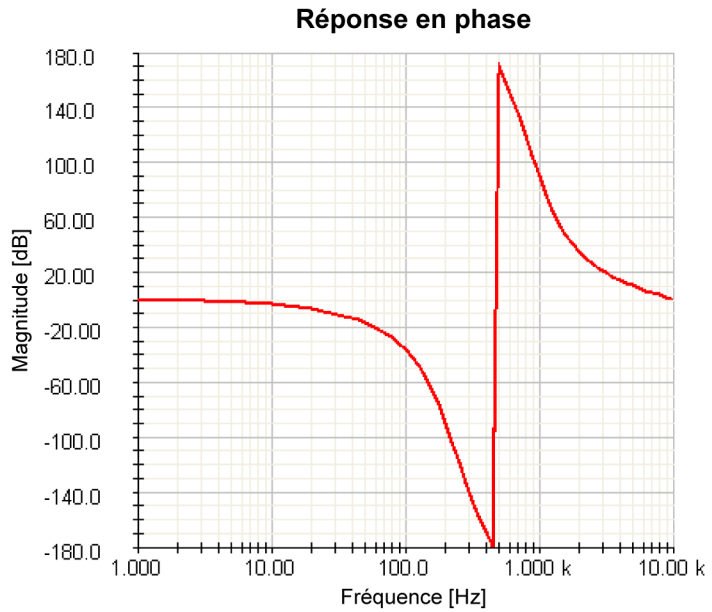


Figure 4.9 : Réponse en phase - FilterButterworthBS

Voir aussi

<< @FilterButterworthBP >> page 72, << @FilterButterworthHP >> page 70 et << @FilterButterworthLP >> page 68

4.21 @FilterBesselLP

Fonction
Filtre le signal d'entrée avec un filtre Bessel passe-bas d'IIR de forme directe.

Syntaxe
@FilterBesselLP*(Signal ; Ordre ; Fc)*

@FilterBesselLP*(Signal ; Ordre ; Fc ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Sans phase	Numéro : méthode de filtrage
	0 Le filtre ne fonctionne pas sans phase (par défaut)
	1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Bessel passe-bas d'IIR de forme directe.

La fréquence de coupure est la fréquence à laquelle la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterBesselLP(Formula.Signal; 2; 200)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

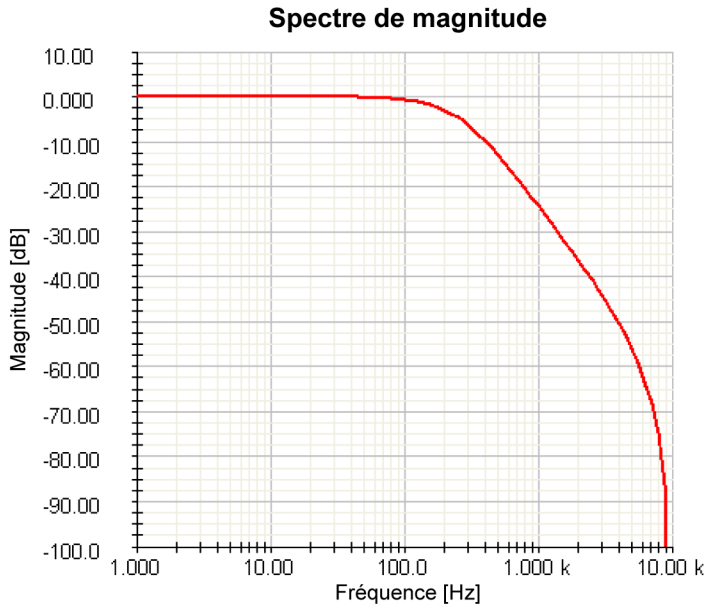


Figure 4.10 : Spectre de magnitude - FilterBesselLP

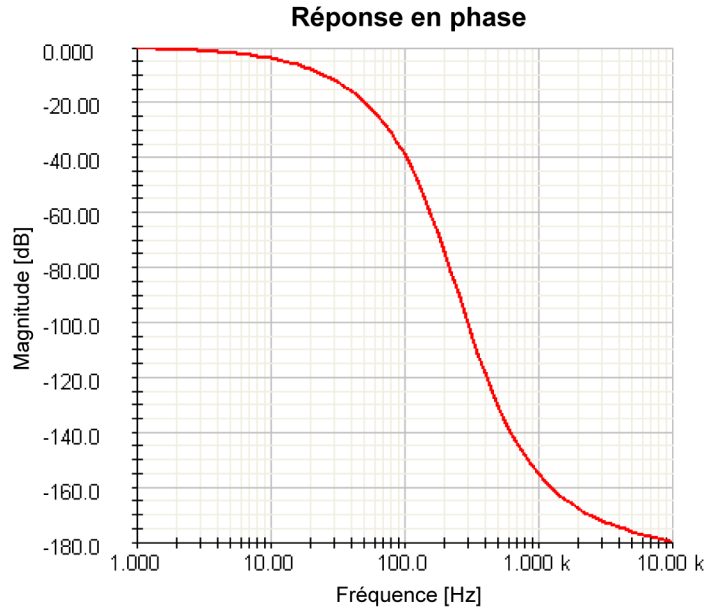


Figure 4.11 : Réponse en phase - FilterBesselLP

Voir aussi

<< @FilterBesselBP >> page 80, << @FilterBesselBS >> page 82 et << @FilterBesselHP >> page 78

4.22 @FilterBesselHP

Fonction
Filtre le signal d'entrée avec un filtre Bessel passe-haut d'IIR de forme directe.

Syntaxe
@FilterBesselHP*(Signal ; Ordre ; Fc)*

@FilterBesselHP*(Signal ; Ordre ; Fc ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Sans phase	Numéro : méthode de filtrage
	0 Le filtre ne fonctionne pas sans phase (par défaut)
	1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Bessel passe-haut d'IIR de forme directe.

La fréquence de coupure est la fréquence à laquelle la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterBesselHP(Formula.Signal; 2; 200)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

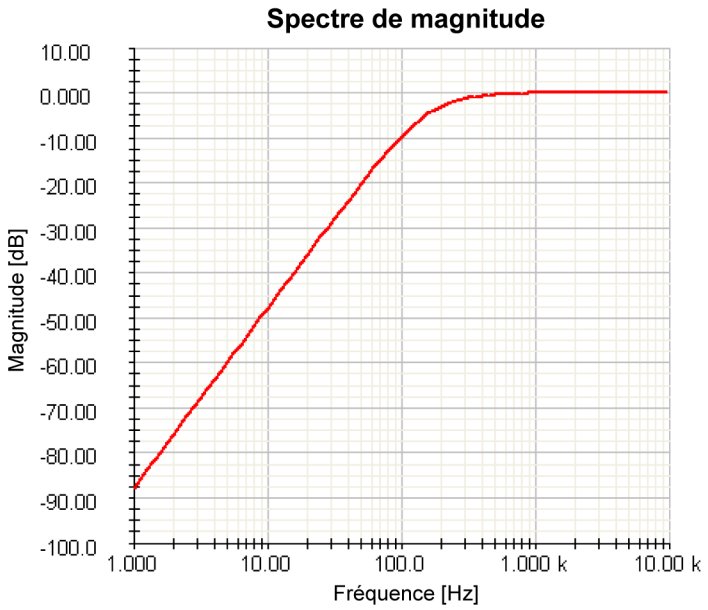


Figure 4.12 : Spectre de magnitude - FilterBesselHP

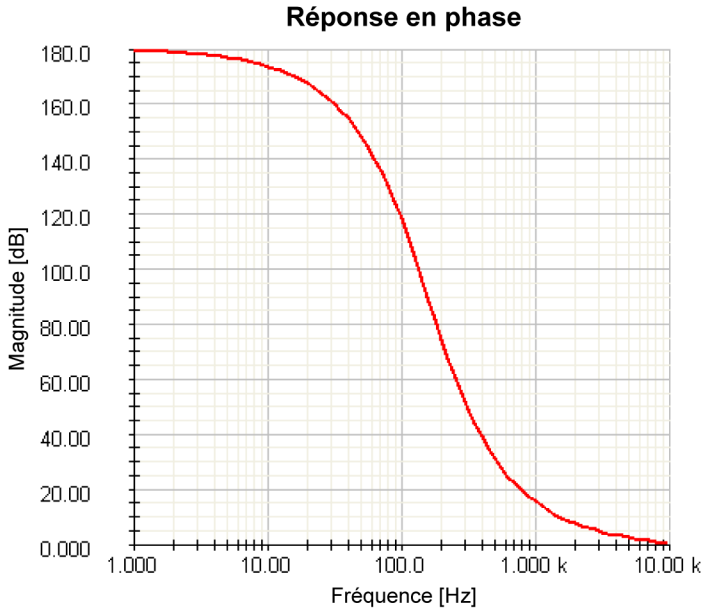


Figure 4.13 : Réponse en phase - FilterBesselHP

Voir aussi

<< @FilterBesselBP >> page 80, << @FilterBesselBS >> page 82 et << @FilterBesselLP >> page 76

4.23 @FilterBesselBP

Fonction
Filtre le signal d'entrée avec un filtre Bessel passe-bande d'IIR de forme directe.

Syntaxe
@FilterBesselBP*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterBesselBP*(Signal ; Ordre ; Fc basse ; Fc haute ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Bessel passe-bande d'IIR de forme directe.

Les fréquences de coupure sont les fréquences auxquelles la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterBesselBP(Formula.Signal; 2; 200;
           1000)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

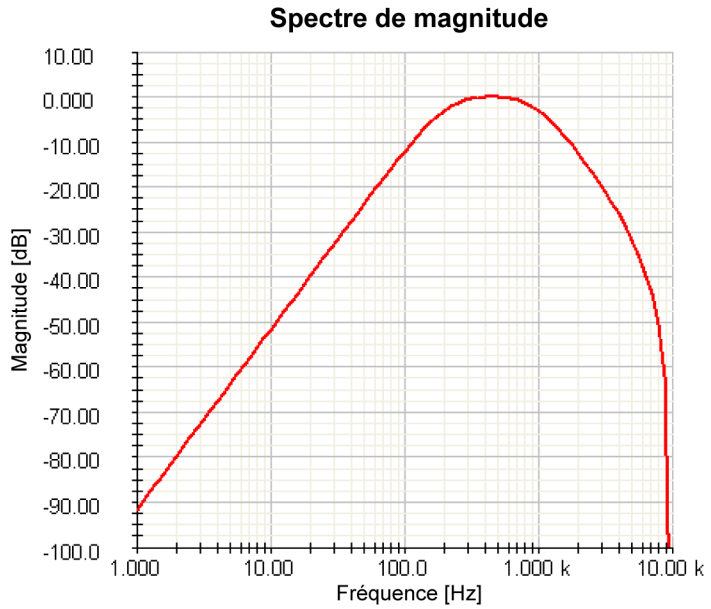


Figure 4.14 : Spectre de magnitude - FilterBesselBP

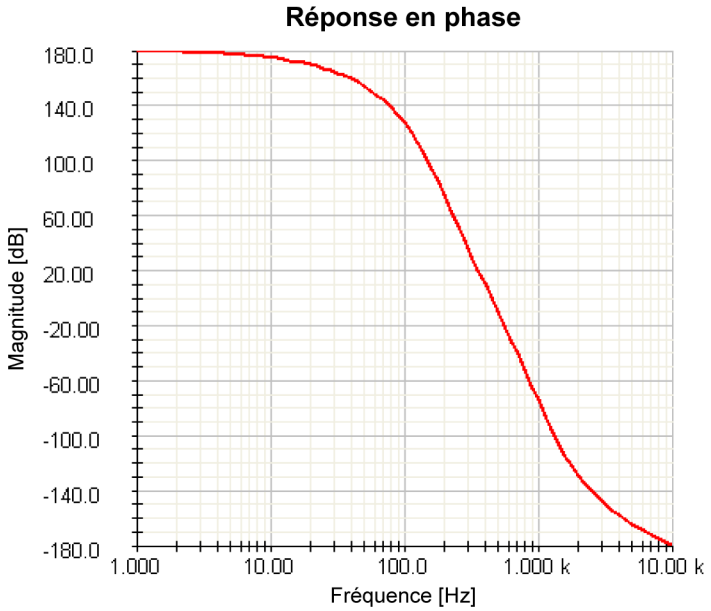


Figure 4.15 : Réponse en phase - FilterBesselBP

Voir aussi

<< @FilterBesselBS >> page 82, << @FilterBesselHP >> page 78 et << @FilterBesselLP >> page 76

4.24 @FilterBesselBS

Fonction
Filtre le signal d'entrée avec un filtre Bessel bloqueur de bande d'IIR de forme directe.

Syntaxe
@FilterBesselBS*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterBesselBS*(Signal ; Ordre ; Fc basse ; Fc haute ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Bessel bloqueur de bande d'IIR de forme directe.

Les fréquences de coupure sont les fréquences auxquelles la magnitude de la réponse est -3 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterBesselBS(Formula.Signal; 2; 200;
           1000)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

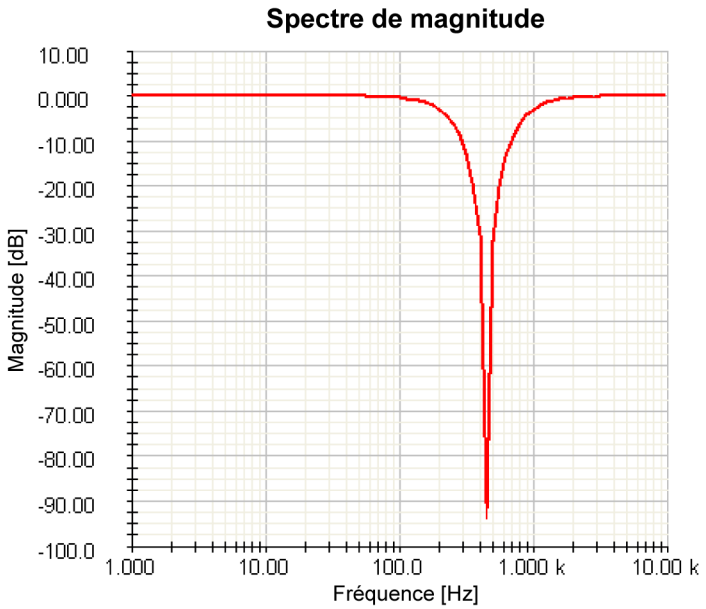


Figure 4.16 : Spectre de magnitude - FilterBesselBS

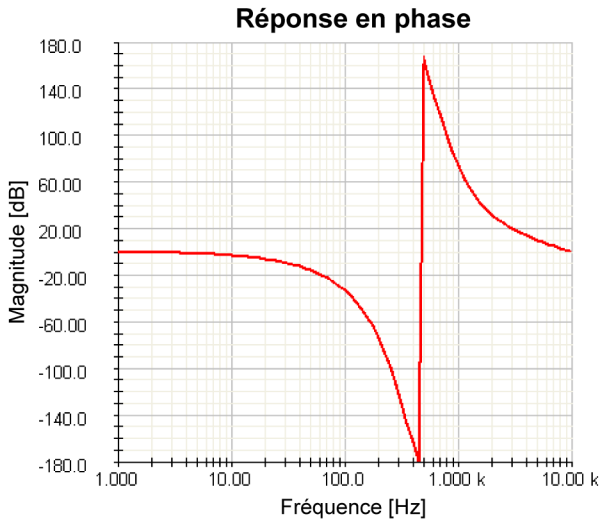


Figure 4.17 : Réponse en phase - FilterBesselBS

Voir aussi

<< @FilterBesselBP >> page 80, << @FilterBesselHP >> page 78 et << @FilterBesselLP >> page 76

4.25 @FilterChebyshevLP

Fonction
Filtre le signal d'entrée avec un filtre Chebyshev passe-bas d'IIR de forme directe.

Syntaxe
@FilterChebyshevLP*(Signal ; Ordre ; Fc)*

@FilterChebyshevLP*(Signal ; Ordre ; Fc ; Ondulations)*

@FilterChebyshevLP*(Signal ; Ordre ; Fc ; Ondulations ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Ondulations	Numéro : amplitude des ondulations bloqueuses de bande en décibels ; la valeur par défaut est 1 dB.
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Chebyshev passe-bas d'IIR de forme directe.

La fréquence de coupure n'est pas le point -3 dB défini sur les filtres Butterworth et Bessel, mais la fréquence la plus haute à laquelle la magnitude de la réponse est égale aux ondulations inférieures à la magnitude maximale spécifiées. Par exemple, si les ondulations sont définies à 2 dB et que la Fc est de 200 Hz, la magnitude à 200 Hz est de -2 dB, voir Figure 4.18.

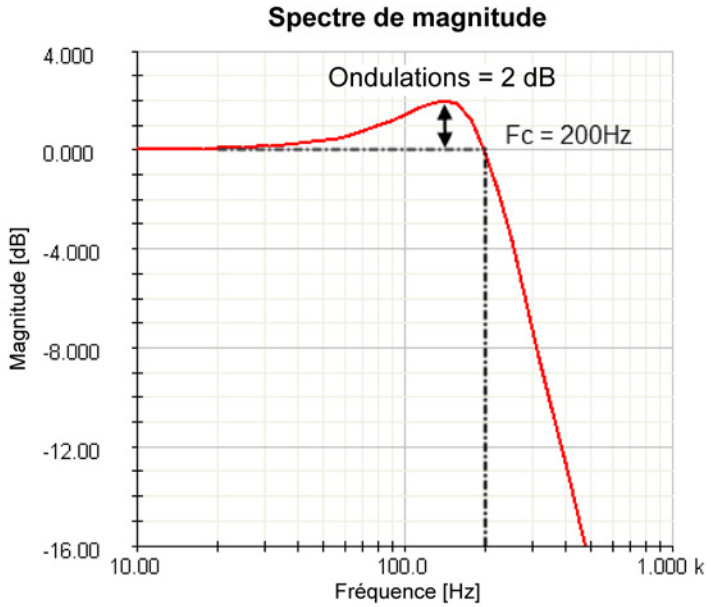


Figure 4.18 : Spectre de magnitude - FilterChebyshevLP (Ondulations)

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterChebyshevLP
           (Formula.Signal; 2; 200; 2)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

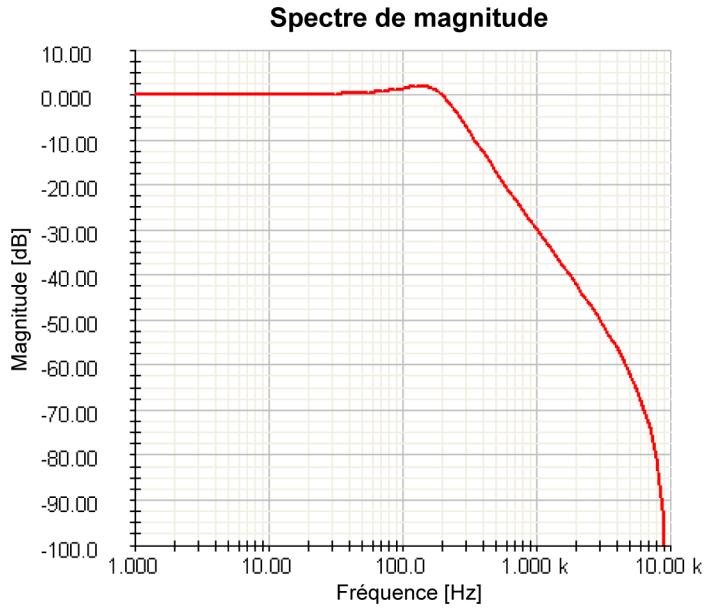


Figure 4.19 : Spectre de magnitude - FilterChebyshevLP

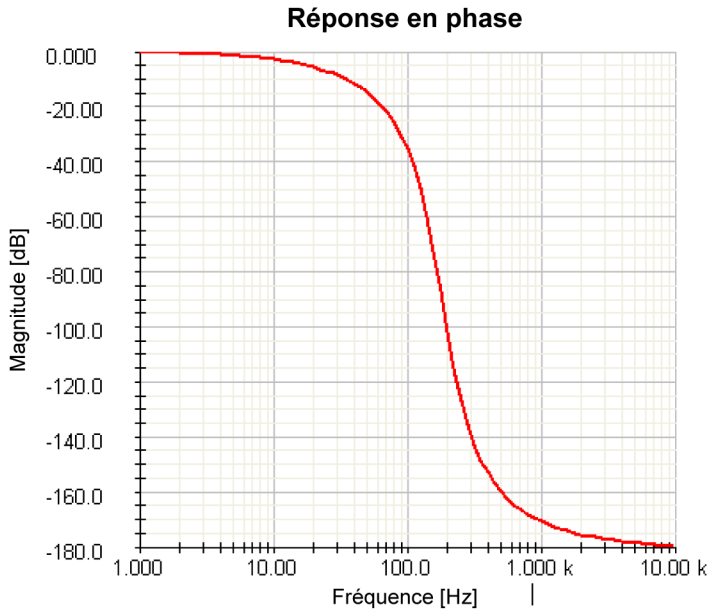


Figure 4.20 : Réponse en phase - FilterChebyshevLP

Voir aussi

<< @FilterChebyshevBP >> page 89, << @FilterChebyshevBS >> page 92 et << @FilterChebyshevHP >> page 87

4.26 @FilterChebyshevHP

Fonction
Filtre le signal d'entrée avec un filtre Chebyshev passe-haut d'IIR de forme directe.

Syntaxe
@FilterChebyshevHP*(Signal ; Ordre ; Fc)*

@FilterChebyshevHP*(Signal ; Ordre ; Fc ; Ondulations)*

@FilterChebyshevHP*(Signal ; Ordre ; Fc ; Ondulations ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc	Numéro : fréquence de coupure en Hz
Ondulations	Numéro : amplitude des ondulations bloqueuses de bande en décibels ; la valeur par défaut est 1 dB.
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Chebyshev passe-haut d'IIR de forme directe.

La fréquence de coupure n'est pas le point -3 dB défini sur les filtres Butterworth et Bessel, mais la fréquence la plus basse à laquelle la magnitude de la réponse est égale aux ondulations inférieures à la magnitude maximale spécifiées. Par exemple, si les ondulations sont définies à 2 dB et que la Fc est de 200 Hz, la magnitude à 200 Hz est de -2 dB. Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal   = @FilterChebyshevHP(Formula.Signal; 2; 200)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

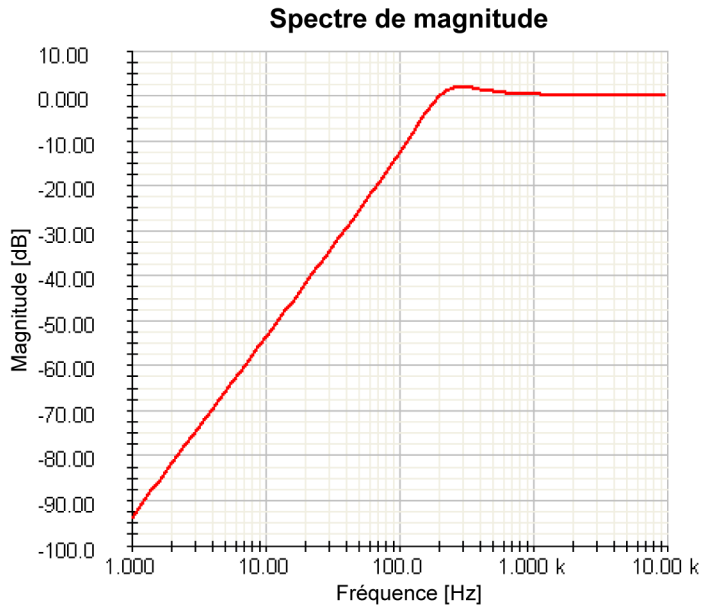


Figure 4.21 : Spectre de magnitude - FilterChebyshev

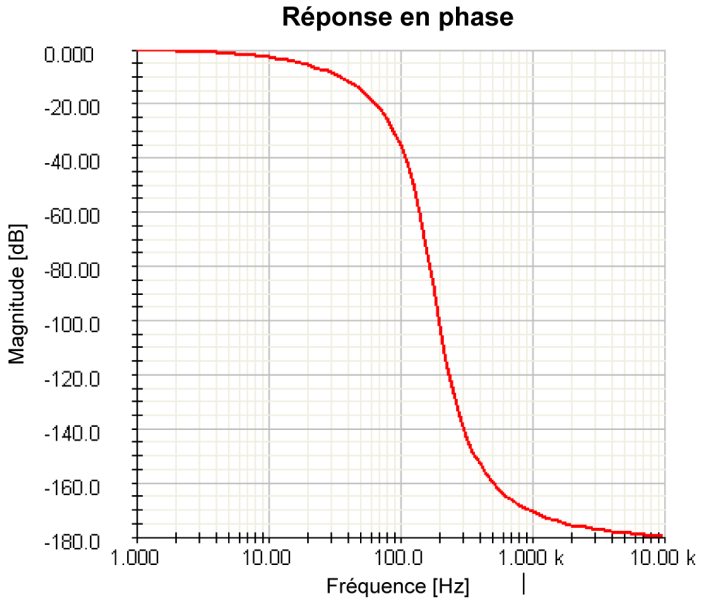


Figure 4.22 : Spectre de magnitude - FilterChebyshev

Voir aussi

<< @FilterChebyshevBP >> page 89, << @FilterChebyshevBS >> page 92 et << @FilterChebyshevLP >> page 84

4.27 @FilterChebyshevBP

Fonction
Filtre le signal d'entrée avec un filtre Chebyshev passe-bande d'IIR de forme directe.

Syntaxe
@FilterChebyshevBP*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterChebyshevBP*(Signal ; Ordre ; Fc basse ; Fc haute ; Ondulations)*

@FilterChebyshevBP*(Signal ; Ordre ; Fc basse ; Fc haute ; Ondulations ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Ondulations	Numéro : amplitude des ondulations bloqueuses de bande en décibels ; la valeur par défaut est 1 dB.
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Chebyshev passe-bande d'IIR de forme directe.
Les fréquences de coupure ne sont pas les points -3 dB définis sur les filtres Butterworth et Bessel, mais les fréquences les plus basses ou les plus élevées auxquelles la magnitude de la réponse est égale aux ondulations inférieures à la magnitude maximale spécifiées. Par exemple, si les ondulations sont définies à 2 dB et que la Fc basse est de 200 Hz, la magnitude à 200 Hz est de -2 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal    = @FilterChebyshevBP(Formula.Signal; 2; 200;
            1000; 2)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

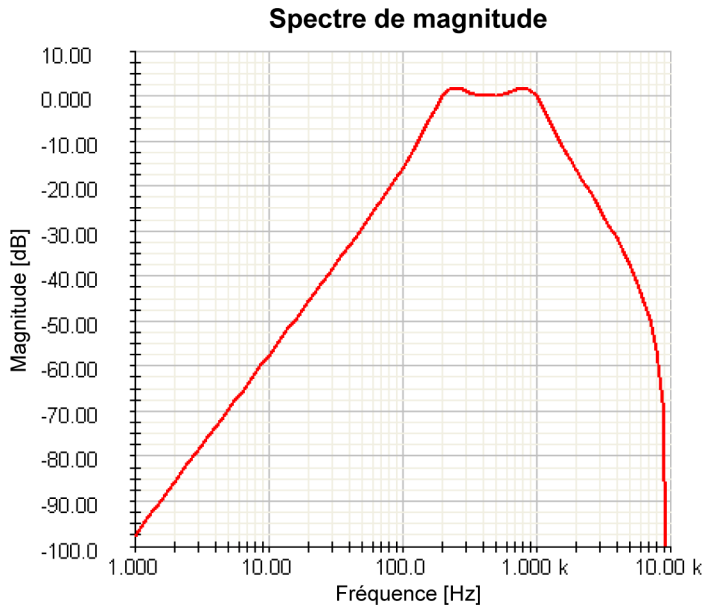


Figure 4.23 : Spectre de magnitude - FilterChebyshevBP

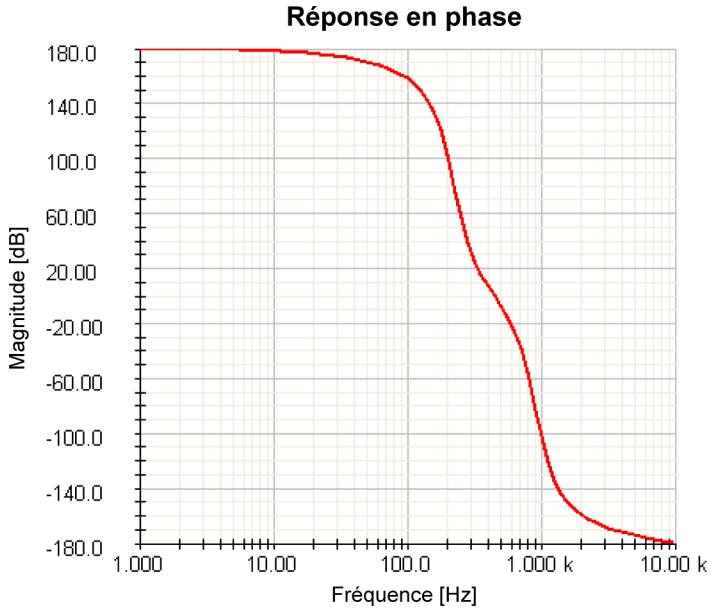


Figure 4.24 : Réponse en phase - FilterChebyshevBP

Voir aussi

<< @FilterChebyshevBS >> page 92, << @FilterChebyshevHP >> page 87 et
<< @FilterChebyshevLP >> page 84

4.28 @FilterChebyshevBS

Fonction
Filtre le signal d'entrée avec un filtre Chebyshev bloqueur de bande d'IIR de forme directe.

Syntaxe
@FilterChebyshevBS*(Signal ; Ordre ; Fc basse ; Fc haute)*

@FilterChebyshevBS*(Signal ; Ordre ; Fc basse ; Fc haute ; Ondulations)*

@FilterChebyshevBS*(Signal ; Ordre ; Fc basse ; Fc haute ; Ondulations ; Sans phase)*

Paramètres

Signal	Forme d'onde d'entrée
Ordre	Numéro : ordre de filtrage
Fc basse	Numéro : fréquence de coupure la plus basse en Hz
Fc haute	Numéro : fréquence de coupure la plus haute en Hz
Ondulations	Numéro : amplitude des ondulations bloqueuses de bande en décibels ; la valeur par défaut est 1 dB.
Sans phase	Numéro : méthode de filtrage 0 Le filtre ne fonctionne pas sans phase (par défaut) 1 Le filtre fonctionne sans phase

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction réalise un filtrage Chebyshev bloqueur de bande d'IIR de forme directe.
Les fréquences de coupure ne sont pas les points -3 dB définis sur les filtres Butterworth et Bessel, mais les fréquences les plus basses ou les plus élevées auxquelles la magnitude de la réponse est égale aux ondulations inférieures à la magnitude maximale spécifiées. Par exemple, si les ondulations sont définies à 2 dB et que la Fc basse est de 200 Hz, la magnitude à 200 Hz est de -2 dB.

Par exemple, si les ondulations sont définies à 2 dB et que la Fc est de 200 Hz, la magnitude à 200 Hz est de -2 dB.

Des informations plus précises sont fournies au chapitre **Filtres IIR** page 198.

Exemple

```
Signal    = @FilterChebyshevBS(Formula.Signal; 2; 200;
            1000)
```

Si le signal *Formula.Signal* est échantillonné à 20 kHz, la fréquence et la réponse en phase sont du type :

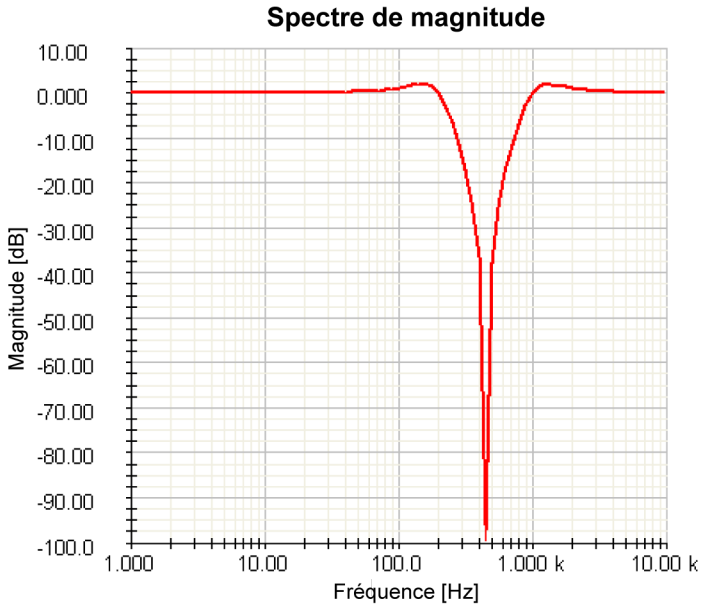


Figure 4.25 : Spectre de magnitude - FilterChebyshevBS

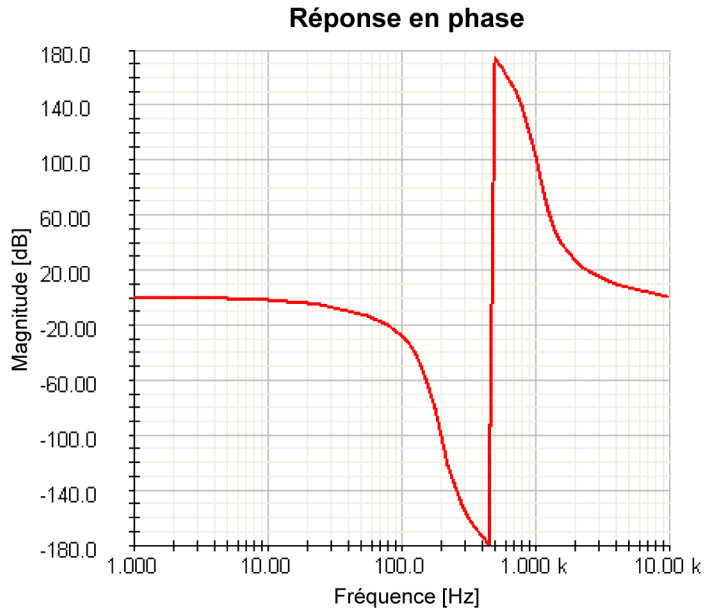


Figure 4.26 : Réponse en phase - FilterChebyshevBS

Voir aussi

<< @FilterChebyshevBP >> page 89, << @FilterChebyshevHP >> page 87 et
<< @FilterChebyshevLP >> page 84

4.29 @Frequency

Fonction
Détermine la **fréquence** d'une forme d'onde.

Syntaxe
@Frequency*(Forme d'onde)*

@Frequency*(Forme d'onde; Début)*

@Frequency*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la fréquence doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique. Cette fonction s'applique également « pendant l'enregistrement » lorsque les paramètres Début et Fin sont tous deux définis. Le résultat est calculé en temps réel à partir des données disponibles.

Description
La fréquence de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du **Début** à la **fin** de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

La fréquence est déterminée comme suit :

- Le système détermine tout d'abord les valeurs minimale et maximale de la forme d'onde. La moyenne de ces dernières est considérée comme le niveau zéro et le nombre de passages par zéro entre le Début et la Fin est calculé. Tous les passages doivent se faire dans la même direction que le premier. Les passages sont déterminés à l'aide d'une hystérésis +/-5 % autour du niveau zéro afin d'éliminer les effets du bruit.
- La période de la forme d'onde est déterminée à partir de la différence de temps entre le premier et le dernier passages par zéro dans la même direction et du nombre de passages par zéro entre ces deux points. La fréquence est l'inverse de la période.

Exemple

L'exemple suivant détermine la fréquence d'un signal de 50 Hz :

```
Signal = @SineWave(20k; 1000; 50)
Freq   = @Frequency(Formula.Signal)
```

Voir aussi

<< @Period >> page 138

4.30 @GreaterEqualThan

Fonction
Cette fonction effectue une évaluation **supérieur ou égal à** (≥) sur les deux paramètres numériques d'entrée.

Syntaxe
@GreaterEqualThan*(Param1; Param2)*

Paramètres

Param 1	Nombre : premier paramètre utilisé pour l'évaluation.
Param 2	Nombre : second paramètre utilisé pour l'évaluation.

Sortie
La sortie est 1 ou 0.

Description
La fonction GreaterEqualThan effectue une évaluation « supérieur ou égal à » sur les paramètres d'entrée.

Si Param 1≥Param 2, la valeur renvoyée est 1 (= vrai), sinon la valeur renvoyée est 0 (= faux).

La fonction **GreaterEqualThan** est généralement utilisée en conjonction avec la fonction IIF.

Exemple
L'exemple suivant compare les paramètres d'entrée et fournit un résultat dépendant de l'issue de cette évaluation :

```
GETExam1    = @GreaterEqualThan(5; 5) => 0 (false)
GETExam2    = @GreaterEqualThan(12; 10) => 1 (true)
IIFExample  = @IIF(Formula.GETExam2; "TRUE"; "FALSE")
```

Voir aussi
<< @EqualTo >> page 62, << @GreaterThan >> page 98, << @IIF >> page 101, << @LessEqualThan >> page 112 et << @LessThan >> page 113

4.31 @GreaterThan

Fonction
Cette fonction effectue une évaluation **supérieur à** (>) sur les deux paramètres numériques d'entrée.

Syntaxe
@GreaterThan*(Param1; Param2)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation.
Param2	Nombre : second paramètre utilisé pour l'évaluation.

Sortie
La sortie est 1 ou 0.

Description
La fonction GreaterThan effectue une évaluation « supérieur à » sur les paramètres d'entrée.

Si Param 1 > Param 2, la valeur renvoyée est 1 (vrai), sinon la valeur renvoyée est 0 (faux).

La fonction GreaterThan est généralement utilisée en conjonction avec la fonction IIF.

Exemple
L'exemple suivant compare les paramètres d'entrée et fournit un résultat dépendant de l'issue de cette évaluation :

```
GTExam1     = @GreaterThan(5;    => 0 (false)
              100)
GTExam2     = @GreaterThan(12;   => 1 (true)
              10)
IIFExample  = @IIF(Formula.GTExam2; "TRUE"; "FALSE")
```

Voir aussi
<< @EqualTo >> page 62, << @GreaterEqualThan >> page 97, << @IIF >> page 101, << @LessEqualThan >> page 112 et << @LessThan >> page 113

4.32 @Histogram

Fonction
Calcule l'**histogramme** d'amplitude.

Syntaxe
@Histogram*(Forme d'onde; Y bas; Y haut; Nombre)*

Paramètres

Forme d'onde	Forme d'onde d'entrée à partir de laquelle l'histogramme doit être calculé.
Y bas	Nombre : amplitude la plus faible à inclure dans l'histogramme.
Y haut	Nombre : amplitude la plus élevée à inclure dans l'histogramme.
Nombre	Nombre de divisions.

Sortie
Forme d'onde contenant l'histogramme d'amplitude.

Description
La fonction de calcul d'histogramme détermine le nombre d'occurrences des valeurs d'amplitude (valeurs d'une forme d'onde) dans chaque division d'amplitude. Les divisions d'amplitude sont définies en divisant la plage comprise entre Y bas et Y haut en autant de divisions que spécifié par le paramètre Nombre. La largeur de chaque division est (Y haut - Y bas)/Nombre. Chaque valeur de la forme d'onde est classée dans ces divisions. L'histogramme montre combien de valeurs ont été classées dans chaque division. Le nombre maximal de divisions autorisé est de 4 096. En général, on utilise des valeurs comprises entre 100 et 1 000.

Les paramètres Y bas et Y haut déterminent la plage des valeurs d'amplitude à inclure dans l'histogramme. Toute valeur hors de cette plage est ignorée. Pour que les valeurs hors plage soient incluses dans une division de l'histogramme, utiliser la fonction @Clip pour limiter les valeurs d'amplitude avant de calculer l'histogramme.

Exemple
L'exemple suivant calcule un histogramme d'amplitude entre -5,5 V et +5,5 V en utilisant 11 divisions. Cela signifie que la largeur de chaque division est de 1 V.

```
Signal = 5 * @SineWave(5000; 1000; 10)
```

```
Histo   = @Histogram(Formula.Signal; -5.5; 5.5; 11)
```

L'exemple suivant calcule un histogramme à 100 divisions entre les valeurs minimale et maximale de la forme d'onde (utilise une variable fictive de la base de données de formules nommée Signal) :

```
MinSignal = @Min(Formula.Signal)
MaxSignal = @Max(Formula.Signal)
Histo     = @Histogram(Formula.Signal;
            Formula.MinSignal;
            Formula.MaxSignal; 100)
```

Voir aussi

<< @Clip >> page 49, << @Max >> page 117 et << @Min >> page 124

4.33 @IIF

Fonction
Cette fonction effectue une évaluation Immediate If (**IIF**, Si immédiat) et renvoie une valeur si une condition est VRAIE et une autre valeur si elle est FAUSSE.

Syntaxe
@IIF*(Param; Si vrai; Si faux)*

Paramètres

Param	Forme d'onde/nombre/chaîne à évaluer.
Si vrai	Forme d'onde/nombre/chaîne renvoyé(e) si le résultat de l'évaluation de Param est vrai.
Si faux	Forme d'onde/nombre/chaîne renvoyé(e) si le résultat de l'évaluation de Param est faux.

Sortie
La sortie peut être une forme d'onde, un nombre ou une chaîne.

Description
La fonction IIF évalue le premier paramètre d'entrée ; le type d'évaluation dépend du type de ce paramètre et est décrit dans le tableau ci-dessous. Le résultat de l'évaluation est vrai ou faux. Si le résultat est vrai, le deuxième paramètre est renvoyé comme sortie de la fonction **IIF**, sinon, c'est le troisième paramètre qui est renvoyé.

Le tableau suivant décrit l'évaluation du premier paramètre :

Type de paramètre	Vrai	Faux
Nombre	Non nul	Nul
Chaîne	Non vide	Vide
Forme d'onde	Non nul	Nul

Exemple
L'exemple suivant utilise une onde sinusoïdale générée. En considérant que ce signal porte le nom Display, la fonction IIF renvoie la chaîne « Above 0.5 » (Supérieur à 0,5) si la valeur de la forme d'onde au niveau du curseur actif est supérieure à 0,5, sinon elle renvoie la chaîne « Below 0.5 » (Inférieur à 0,5). La fonction GreaterThan est utilisée pour comparer la valeur de la forme d'onde au niveau du curseur actif à la valeur 0,5. Le résultat est 1 (vrai) si la valeur est supérieure à 0,5, sinon le résultat est 0 (faux). Ce résultat est le premier paramètre de la fonction **IIF**.

```
Signal      = @SineWave(8000; 8001; 5)
GreaterHalf = @GreaterThan
              (Display.Display.ActiveCursor.YValue;
              0.5)
IIFExample  = @IIF(Formula.GreaterHalf; "Above 0.5";
              "Below 0.5")
```

Voir aussi

<< @EqualTo >> page 62, << @GreaterEqualThan >> page 97, << @GreaterThan >> page 98, << @LessEqualThan >> page 112 et << @LessThan >> page 113

4.34 @Integrate

Fonction
Intègre une forme d'onde.

Syntaxe
@Integrate*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde à intégrer.

Sortie
Forme d'onde intégrée.

Description
L'intégration calcule le cumul de toutes les valeurs d'échantillon :

$$\begin{aligned} \mathrm{i}(1) &= \mathrm{y}(1) \\ \mathrm{i}(\mathrm{n}) &= \mathrm{i}(\mathrm{n}-1) + \mathrm{y}(\mathrm{n}) \cdot \Delta x \quad \text{for } \mathrm{n} = 2,\ldots,\mathrm{N} \end{aligned}$$

L'intégrale calcule la surface sous la courbe de la forme d'onde entre les points initial et final.

Exemple
L'exemple suivant intègre la puissance instantanée pour obtenir l'énergie en fonction du temps :

```
Signal  = @SineWave(10k; 1000; 50)
Power   = (Formula.Signal * Formula.Signal) / 600
Energy  = @Integrate(Formula.Power)
```

Voir aussi
<< @Diff >> page 58 et << @Energy >> page 60

4.35 @IntLookUp

Fonction
Utilise les données d'une forme d'onde comme index (pointeur) vers une table de conversion (CLUT). La CLUT est enregistrée dans un fichier distinct.

Syntaxe
@IntLookUp*(Forme d'onde; CLUT)*

@IntLookUp*(Forme d'onde; CLUT; Décalage index)*

Paramètres

Forme d'onde	Forme d'onde d'entrée
CLUT	Chaîne : chemin complet indiquant le nom et l'emplacement unique du fichier CLUT.
Décalage index	Nombre : décalage du pointeur index.

Sortie
Forme d'onde convertie.

Description
Cette fonction utilise les données d'une forme d'onde comme index vers une table de conversion. Ces données peuvent être des entiers sur 16 bits.

La table de conversion (CLUT) est enregistrée dans un fichier ASCII (texte).

Chaque ligne de ce fichier contient un nombre à virgule flottante. Le numéro de ligne est utilisé comme index pour la CLUT.

La première ligne est associée par défaut à l'index 0. Toutefois, si la fonction utilise le paramètre « Décalage index » facultatif, elle est associée à –Décalage index. Cela permet de travailler avec des valeurs négatives.

Lors de l'utilisation d'entiers sur 16 bits, la CLUT doit contenir 2^{16} = 65 536 points. Lorsque les données sont précédées d'un signe (positif ou négatif), utiliser un décalage de 32 768. Le premier point de la CLUT est alors associé à -32 768.

Exemple
Prenons par exemple la table de conversion suivante :

Numéro de ligne	Valeur
1	11,3
2	21,4
3	31,5
4	41,4
5	51,2
6	61,3
7	71,5
8	81,6
9	91,2
10	101,4
11	111,2

Supposons que la forme d'onde d'origine soit :

6, 2, 7, 8

La première valeur 6 est associée à la ligne 7 (index basé sur zéro). La valeur correspondante dans la CLUT est 71,5.

La forme d'onde de sortie convertie est donc :

71,5, 31,5, 81,6, 91,2

Si le paramètre « Décalage index » est utilisé et défini sur 2 :

La première valeur 6 est associée à la ligne 5 (index 6 - 2, basé sur zéro). La valeur correspondante dans la CLUT est 51,2.

La forme d'onde de sortie convertie est donc :

51,2, 11,3, 61,3, 71,5

Si la forme d'onde d'entrée contient un index supérieur à l'index maximal enregistré dans la CLUT, la dernière valeur de cette CLUT est utilisée. Si la forme d'onde d'entrée contient un index inférieur à l'index minimal enregistré dans la CLUT, la première valeur de cette CLUT est utilisée.

Pour utiliser la table de conversion externe « Lookup.asc » se trouvant dans le répertoire « C:\CalibData »:

```
Signal = @Ramp(1000; 2001; 0; 4096; 11)
Calib  = @IntLookUp
       (Formula.Signal; "C:\CalibData\Lookup.asc")
```

Voir aussi

<< @IntLookUp12 >> page 107

4.36 @IntLookUp12

Fonction
Utilise les données d'une forme d'onde comme index (pointeur) vers une table de conversion (CLUT).

Syntaxe
@IntLookUp12*(Forme d'onde; Clut)*

Paramètre

Forme d'onde	Forme d'onde à convertir.
Clut	Table de conversion

Sortie
Forme d'onde convertie.

Description
Cette fonction utilise les données d'une forme d'onde comme index (pointeur) vers une table de conversion.

Remarque *Cette fonction est optimisée pour les entiers sur 12 bits.*

La table de conversion comporte 4 096 valeurs.

Chaque valeur de la forme d'onde d'origine est utilisée comme pointeur vers la CLUT. La valeur correspondante dans cette dernière est utilisée pour générer la sortie.

La CLUT peut être créée :

- à l'aide d'un langage de programmation de haut niveau et de l'option CSI de Perception ;
- à l'aide d'une (combinaison de) fonction(s) de la base de données de formules.

Le schéma suivant présente un exemple d'utilisation de cette fonction.

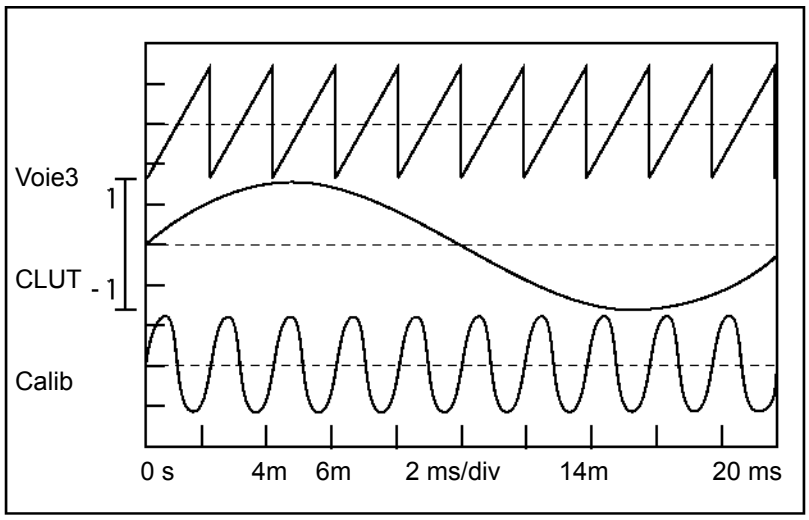


Figure 4.27 : Schéma - Fonction @IntLookUp12

- Le tracé du haut représente l'enregistrement d'origine.
- Le tracé du milieu correspond au contenu de la CLUT.
- Le tracé du bas représente les données converties.

Pour créer ce schéma, les formules suivantes sont utilisées :

```
CLUT  = @SineWave(200k; 4096; 48.8)
Org   = @Ramp(2M; 40960; 4095; 10)
Calib = @IntLookUp12(Formula.Org; Formula.CLUT)
```

Une onde sinusoïdale de 4 096 points avec une fréquence de 48,8 Hz est tout d'abord générée. Avec un taux d'échantillonnage de 200 kHz, la forme d'onde correspond à un cycle.

Le résultat intitulé « Calib » est obtenu en utilisant la forme d'onde Formula.Org générée comme index pour la CLUT.

Remarque *Les données d'origine doivent être des entiers (2 octets) compris dans la plage de -32 767 à +32 768 avec une résolution de 12 bits. Avant leur utilisation, la valeur entière est divisée par 16 pour obtenir le bon index. Les données sur 14 ou 16 bits peuvent donc être utilisées, mais cela entraîne la perte des bits de poids faible et risque de produire des résultats inattendus.*

Voir aussi
<< @IntLookUp >> page 104

4.37 @Join

Fonction
Assemble deux formes d'onde ou plus par concaténation.

Syntaxe
@Join*(Forme d'onde 1; ...; Forme d'onde N)*

Paramètres

Forme d'onde 1	Première forme d'onde.
Forme d'onde N	Dernière forme d'onde, avec N >= 2

Sortie
Forme d'onde concaténée.

Description
Cette fonction concatène deux formes d'onde ou plus pour générer une nouvelle forme d'onde. Les formes d'onde sont assemblées par concaténation de la fin d'une forme d'onde avec le début de la forme d'onde suivante. L'axe horizontal est déterminé par la première forme d'onde. La coordonnée X du premier échantillon de la forme d'onde de sortie correspond à la coordonnée X du premier échantillon de la première forme d'onde. Les échantillons de la ou des autres formes d'onde sont simplement concaténés aux échantillons de la première forme d'onde. La longueur de la forme d'onde de sortie correspond à la somme des longueurs des formes d'onde définies comme paramètres.

Lorsque les formes d'onde d'entrée présentent des bases de temps différentes, la forme d'onde de sortie contient également plusieurs bases de temps.

Exemple
Cette fonction est particulièrement utile pour générer des signaux simulés comme illustré dans l'exemple suivant :

```
Seg1   = @Ramp(1k; 100; 0; 0)
Seg2   = @Ramp(1k; 50; 0; 1)
Seg3   = @Ramp(1k; 100; 1; 1)
Signal = @Join(Formula.Seg1; Formula.Seg2;
         Formula.Seg3)
```

La fonction Join concatène le premier point de donnée valide d'une forme d'onde à la forme d'onde précédente. Cela signifie que les heures de fin et de début d'origine des deux formes d'onde concaténées peuvent être différentes. Toutefois, elles sont espacées d'un (1) intervalle d'échantillonnage dans la forme d'onde de sortie.

L'exemple suivant illustre ces propos :

```
Seg1 = @Ramp(1k; 100; 0; 0)
Seg2 = @Ramp(1k; 100; 0; 0)
```

Décalage du deuxième segment de 100 unités horizontales à l'aide de la fonction @XShift :

```
Seg3   = @XShift(Formula.Seg2; 100)
Signal = @Join(Formula.Seg1; Formula.Seg3)
```

Voir aussi
<< @Cut >> page 54

4.38 @Length

Fonction
Renvoie le **nombre d'échantillons** présents dans une forme d'onde.

Syntaxe
@Length*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde d'entrée

Sortie
La sortie est une valeur numérique.

Description
Cette fonction renvoie le nombre d'échantillons présents dans la forme d'onde d'entrée définie. Cette valeur peut être utilisée comme paramètre d'entrée pour d'autres fonctions.

Exemple
L'exemple suivant crée une onde sinusoïdale comportant 1 000 échantillons. La variable NSamples reçoit la longueur de ce signal (1 000).

```
Signal   = 25 * @SineWave(20k; 1000; 50)
NSamples = @Length(Formula.Signal)
```

Voir aussi
<< @XDelta >> page 185, << @XFirst >> page 188 et << @XLast >> page 189

4.39 @LessEqualThan

Fonction
Cette fonction effectue une évaluation **inférieur ou égal à** (≤) sur les deux paramètres numériques d'entrée.

Syntaxe
@LessEqualThan *(Param1; Param2)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation.
Param2	Nombre : second paramètre utilisé pour l'évaluation.

Sortie
La sortie est 1 ou 0.

Description
La fonction LessEqualThan effectue une évaluation « inférieur ou égal à » sur les paramètres d'entrée.

Si Param 1 ≤ Param 2, la valeur renvoyée est 1 (vrai), sinon la valeur renvoyée est 0 (faux).

La fonction LessEqualThan est généralement utilisée en conjonction avec la fonction IIF.

Exemple
L'exemple suivant compare les paramètres d'entrée et fournit un résultat dépendant de l'issue de cette évaluation :

```
LETExam1    = @LessEqualThan(5; 5)      => 1 (true)
LETExam2    = @LessEqualThan(12; 10)    => 0 (false)
IIFExample  = @IIF(Formula.LETExam2; "true"; "false")
```

Voir aussi
<< @EqualTo >> page 62, << @GreaterEqualThan >> page 97, << @GreaterThan >> page 98, << @IIF >> page 101 et << @LessThan >> page 113

4.40 @LessThan

Fonction
Cette fonction effectue une évaluation **inférieur à** (<) sur les deux paramètres numériques d'entrée.

Syntaxe
@LessThan*(Param1; Param2)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation.
Param2	Nombre : second paramètre utilisé pour l'évaluation.

Sortie
La sortie est 1 ou 0.

Description
La fonction LessThan effectue une évaluation « inférieur à » sur les paramètres d'entrée. Si Param 1 < Param 2, la valeur renvoyée est 1 (vrai), sinon la valeur renvoyée est 0 (faux).

La fonction LessThan est généralement utilisée en conjonction avec la fonction IIF.

Exemple
L'exemple suivant compare les paramètres d'entrée et fournit un résultat dépendant de l'issue de cette évaluation :

```
LessThanExampl1  = @LessThan(5;  => 1 (true)
                   100)
LessThanExampl2  = @LessThan(12; => 0 (false)
                   10)
IIFExample       = @IIF(Formula.LessThanExampl2;
                   "TRUE"; "FALSE")
```

Voir aussi
<< @EqualTo >> page 62, << @GreaterEqualThan >> page 97, << @IIF >> page 101 et << @LessEqualThan >> page 112

4.41 @Ln

Fonction
Logarithme népérien d'un nombre donné (ou logarithme de base « e »), noté ln(x) ou $\log_e(x)$: puissance à laquelle la base (e) doit être élevée pour obtenir le nombre.

Syntaxe
@Ln*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant le logarithme népérien de l'entrée.

Description
Calcule le logarithme de base e (e = 2,718...). Ce dernier est appelé logarithme népérien. Lorsqu'un paramètre de forme d'onde est utilisé, la fonction log est calculée pour chacun des échantillons.

Si un nombre est inférieur à 1E-30, la valeur est définie sur ln(1E-30) = -69,08.

Concrètement, ln(a) peut être défini comme étant la surface sous le graphique (intégrale) de 1/x de 1 à a, c'est-à-dire :

$$\mathrm{In(a)} = \int_1^a \frac{1}{x}\, dx.$$

Cette fonction est l'inverse de @Exp.

Exemple
L'exemple suivant montre que le logarithme népérien de la variable système System.Constants.e est égal à 1 :

```
One = @Ln(System.Constants.e)
```

L'exemple suivant montre comment le logarithme népérien peut être utilisé dans une forme d'onde :

```
Wave    = @Ramp(10k; 1k; 1E-5; 10)
WaveLn  = @Ln(Formula.Wave)
```

Voir aussi
<< @Exp >> page 63 et << @Log >> page 116

4.42 @Log

Fonction
Logarithme de base 10 d'un nombre donné, noté $\log_{10}(x)$: puissance à laquelle la base (10) doit être élevée pour obtenir le nombre.

Syntaxe
@Log*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant le logarithme de l'entrée.

Description
Calcule le logarithme de base 10. Lorsqu'un paramètre de forme d'onde est utilisé, la fonction log est calculée pour chacun des échantillons.

Si un nombre est inférieur à 1E-30, la valeur est définie sur log(1E-30) = -30.

Exemple
Si $10^x = 100$, alors $x = \log_{10}(100)$

```
x = @Log(100)
```

Le résultat est x = 2

Voir aussi
<< @Ln >> page 114 et << @Pow >> page 140

4.43 @Max

Fonction
Détermine la valeur **maximale** d'une forme d'onde.

Syntaxe
@Max*(Forme d'onde)*

@Max*(Forme d'onde; Début)*

@Max*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la valeur maximale doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique. Cette fonction s'applique également « pendant l'enregistrement » lorsque les paramètres Début et Fin sont tous deux définis. Le résultat est calculé en temps réel à partir des données disponibles.

Description
La valeur maximale de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant crée une onde sinusoïdale de 50 Hz. La valeur maximale des premières 20 ms (une période de 50 Hz) du signal est calculée pour déterminer l'amplitude (maximale) :

```
Signal  = 25 * @SineWave(20k; 1000; 50)
Ampl    = @Max(Formula.Signal; 0; 20m)
```

Si la position de la valeur maximale doit être déterminée, utiliser plutôt la fonction @MaxPos. La valeur maximale à cette position peut ensuite être obtenue à l'aide de la fonction @Value :

```
TMax = @MaxPos(Formula.Signal; 0; 20m)
Ampl = @Value(Formula.Signal; Formula.TMax)
```

Voir aussi
<< @MaxPos >> page 120, << @Min >> page 124 et << @MinPos >> page 127
<< @Value >> page 184

4.44 @MaxNum

Fonction
Détermine la **plus grande** de deux valeurs **numériques** ou plus.

Syntaxe
@MaxNum*(Par1; ...; ParN)*

Paramètres

Par1	Première valeur numérique.
ParN	Dernière valeur numérique, avec N >= 2.

Sortie
Maximum de toutes les valeurs numériques.

Description
Cette fonction renvoie la plus grande des valeurs numériques définies comme paramètres. Chaque paramètre doit être une valeur numérique valide. Si l'un d'eux n'est pas du bon type ou ne contient aucune valeur, la fonction ne renvoie aucune valeur.

Exemple
L'exemple suivant calcule la valeur minimale d'une forme d'onde mais écrête cette valeur si elle est inférieure à zéro. La forme d'onde est supposée être un signal réel :

```
Signal      = Active.Group1.Recorder_A.Ch_A1
MinValue    = @Min(Formula.Signal)
ClippedMin  = @MaxNum(Formula.MinValue; 0)
```

L'exemple suivant renvoie 10 :

```
Ten  = @MaxNum(-10; 3; 2.4; 10.0; -9.9)
```

Voir aussi
<< @MinNum >> page 126

4.45 @MaxPos

Fonction
Détermine la **position** de la **valeur maximale** d'une forme d'onde.

Syntaxe
@MaxPos*(Forme d'onde)*

@MaxPos*(Forme d'onde; Début)*

@MaxPos*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur maximale doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est la position X de la valeur maximale. Cette fonction s'applique également « pendant l'enregistrement » lorsque les paramètres Début et Fin sont tous deux définis. Le résultat est calculé en temps réel à partir des données disponibles.

Description
La position X de la valeur maximale de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres **Début** et **Fin** doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine le moment où la pression atteint sa valeur maximale. La pression maximale en elle-même est déterminée à l'aide de la fonction @Value pour ce point.

La pression est supposée être un signal réel :

```
Pressure   = Active.Group1.Recorder_A.Ch_A1
TimeOfMax  = @MaxPos(Formula.Pressure)
MaxPress   = @Value(Formula.Pressure;
             Formula.TimeOfMax)
```

Voir aussi
<< @Max >> page 117, << @Min >> page 124 et << @MinPos >> page 127

4.46 @Mean

Fonction
Calcule la valeur **moyenne** arithmétique d'une forme d'onde.

Syntaxe
@Mean*(Forme d'onde)*

@Mean*(Forme d'onde; Début)*

@Mean*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la valeur moyenne doit être calculée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique. Cette fonction s'applique également « pendant l'enregistrement » lorsque les paramètres Début et Fin sont tous deux définis. Le résultat est calculé en temps réel à partir des données disponibles.

Description
La valeur moyenne est calculée à l'aide de la formule suivante :

$$\text{Mean} = \frac{1}{N} \sum_{n=n_1}^{n_2} y(n) \text{ with N} = (n_2 - n_1 + 1)$$

Les limites de segment Début (n1) et Fin (n2) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant crée un signal (valeur constante de 100) affecté par un bourdonnement et du bruit. La valeur moyenne des premières 20 ms (une période de 50 Hz) du signal est calculée :

```
Signal = 100 + 0.5 * @SineWave(20k; 1000; 50) +
         @Noise(20k; 1000)
Avg    = @Mean(Formula.Signal; 0; 20m)
```

Grâce au calcul de la moyenne, Avg (Moyenne) constitue une valeur bien plus précise du signal que celle des différents échantillons de la forme d'onde.

Voir aussi
<< @RMS >> page 163 et << @StdDev >> page 173

4.47 @Min

Fonction
Détermine la valeur **minimale** d'une forme d'onde.

Syntaxe
@Min*(Forme d'onde)*

@Min*(Forme d'onde; Début)*

@Min*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la valeur minimale doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique. Cette fonction s'applique également « pendant l'enregistrement » lorsque les paramètres Début et Fin sont tous deux définis. Le résultat est calculé en temps réel à partir des données disponibles.

Description
La valeur minimale de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple

L'exemple suivant crée une onde sinusoïdale de 50 Hz au-dessus d'une valeur DC. Les valeurs maximale et minimale des premières 20 ms (une période de 50 Hz) du signal sont calculées pour déterminer l'amplitude (crête à crête) de la composante 50 Hz :

```
Signal  = 100 + 25 * @SineWave(20k; 1000; 50)
MinAmpl = @Min(Formula.Signal; 0; 20m)
```

```
MaxAmpl = @Max(Formula.Signal; 0; 20m)
Vpp     = Formula.MaxAmpl - Formula.MinAmpl
```

Si la position de la valeur minimale doit être déterminée, utiliser plutôt la fonction @MinPos. La valeur minimale à cette position peut ensuite être obtenue à l'aide de la fonction @Value :

```
TMin   = @MinPos(Signal; 0; 20m)
Minim  = @Value(Formula.Signal; Formula.TMin)
```

Voir aussi

<< @Max >> page 117, << @MaxPos >> page 120 et
<< @MinPos >> page 127

4.48 @MinNum

Fonction
Détermine la **plus petite** de deux valeurs **numériques** ou plus.

Syntaxe
@MinNum*(Par1; ...; ParN)*

Paramètres

Par1	Première valeur numérique.
ParN	Dernière valeur numérique. N >= 2.

Sortie
Minimum de toutes les valeurs numériques.

Description
Cette fonction renvoie la plus petite des valeurs numériques définies comme paramètres. Chaque paramètre doit être une valeur numérique valide. Si l'un d'eux n'est pas du bon type ou ne contient aucune valeur, la fonction ne renvoie aucune valeur.

Exemple
L'exemple suivant calcule la valeur maximale d'une forme d'onde mais écrête cette valeur si elle est supérieure à 100. La forme d'onde est supposée être un signal réel :

```
Signal      = Active.Group1.Recorder_A.Ch_A1
MaxValue    = @Max(Formula.Signal)
ClippedMax  = @MinNum(Formula.MaxValue; 100)
```

L'exemple suivant renvoie -10 :

```
MinusTen = @MinNum(45.0; 10; -3.3; 20; -10.0; 5)
```

Voir aussi
<< @MaxNum >> page 119

4.49 @MinPos

Fonction
Détermine la **position** de la **valeur minimale** d'une forme d'onde.

Syntaxe
@MinPos*(Forme d'onde)*

@MinPos*(Forme d'onde; Début)*

@MinPos*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur minimale doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est la position X de la valeur minimale.

Description
La position X de la valeur minimale de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres **Début** et **Fin** doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine le moment où la pression atteint sa valeur minimale. La pression minimale en elle-même est déterminée à l'aide de la fonction @Value pour ce point. La variable Pressure est supposée être un signal réel :

```
Pressure  = Active.Group1.Recorder_A.Ch_A1
TimeOfMin = @MinPos(Formula.Pressure)
MinPress  = @Value(Formula.Pressure; Formula.TimeOfMin)
```

Voir aussi

<< @Max >> page 117, << @MaxPos >> page 120 et << @Min >> page 124

4.50 @NextHillPos

Fonction
Cette fonction recherche la **position** de la **valeur maximale locale suivante** dans une forme d'onde.

Syntaxe
@NextHillPos*(Forme d'onde; Pos. départ; Hystérésis)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur maximale locale suivante doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer (vers l'avant).
Hystérésis	Nombre : valeur de l'hystérésis de suppression du bruit.

Sortie
La sortie est la position X de la valeur maximale locale suivante.

Description
La position X de la valeur maximale locale suivante dans la forme d'onde est déterminée.

La recherche s'effectue vers l'avant en commençant à la position X définie par le paramètre Pos. départ. Si cette dernière se trouve avant le début de la forme d'onde, ce dernier est utilisé comme position de départ de la recherche.

La valeur d'hystérésis est utilisée pour éliminer les effets du bruit dans le signal. Par exemple, si un bruit de 100 mV crête à crête est présent dans un signal, le fait de définir une valeur d'hystérésis de 200 mV permet d'éviter que l'algorithme ne détecte une petite crête de bruit comme valeur maximale locale.

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine la position d'une valeur maximale locale dans une onde sinusoïdale de 1 V avec un bruit de 10 mV en effectuant la recherche vers l'avant à partir de 500 ms et en utilisant une hystérésis de 40 mV :

```
Signal = @SineWave(10k; 10000; 50 ) + 0.01 * @Noise(10k;
         10000)
NHPos  = @NextHillPos(Formula.Signal; 500m; 40m)
```

Voir aussi

<< @NextValleyPos >> page 133, << @PrevHillPos >> page 141 et << @PrevValleyPos >> page 145

4.51 @NextLvlCross

Fonction
Détermine la position du **croisement suivant** d'une forme d'onde avec un **niveau** de signal donné.

Syntaxe
@NextLvlCross*(Forme d'onde; Pos. départ; Niveau)*

@NextLvlCross*(Forme d'onde; Pos. départ; Niveau; Pente)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position du franchissement de niveau suivant doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer.
Niveau	Nombre : niveau d'amplitude à rechercher.
Pente	Nombre : direction de la pente (-1, 0, 1)

Sortie
La sortie est la position X du franchissement de niveau suivant.

Description
La position X du croisement suivant de la forme d'onde avec le niveau défini et dans la direction spécifiée est déterminée.
La recherche s'effectue vers la droite (dans la direction positive du temps) en commençant à la position X définie par le paramètre Pos. départ. Si cette dernière se trouve avant le début de la forme d'onde, ce dernier est utilisé comme position de départ de la recherche.

Le paramètre Pente contrôle le type de franchissement de niveau recherché :

- Pente = 1 : franchissement de niveau positif (du dessous vers le dessus du niveau)
- Pente = -1 : franchissement de niveau négatif (du dessus vers le dessous du niveau)
- Pente = 0 : tout franchissement (positif ou négatif)

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine le début et la fin de la première impulsion TTL du signal TTLSignal, qui est supposé être un signal réel dans “Active.Group1.Recorder_A.Ch_A1”.

```
TTSignal   = Active.Group1.Recorder_A.Ch_A1
TimeStart  = @NextLvlCross(Formula.TTSignal; -1E20;
             2.5; 1)
TimeEnd    = @NextLvlCross(Formula.TTSignal;
             Formula.TimeStart; 2.5; -1)
```

Voir aussi
<< @PrevLvlCross >> page 143

4.52 @NextValleyPos

Fonction
Cette fonction recherche la **position** de la **valeur minimale locale suivante** dans une forme d'onde.

Syntaxe
@NextValleyPos*(Forme d'onde; Pos. départ; Hystérésis)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur minimale locale suivante doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer (vers l'avant).
Hystérésis	Nombre : valeur de l'hystérésis de suppression du bruit.

Sortie
La sortie est la position X de la valeur minimale locale suivante.

Description
La position X de la valeur minimale locale suivante dans la forme d'onde est déterminée.

La recherche s'effectue vers l'avant en commençant à la position X définie par le paramètre Pos. départ. Si cette dernière se trouve avant le début de la forme d'onde, ce dernier est utilisé comme position de départ de la recherche.

La valeur d'hystérésis est utilisée pour éliminer les effets du bruit dans le signal. Par exemple, si un bruit de 100 mV crête à crête est présent dans un signal, le fait de définir une valeur d'hystérésis de 200 mV permet d'éviter que l'algorithme ne détecte une petite crête de bruit négative comme valeur minimale locale.

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine la position d'une valeur minimale locale dans une onde sinusoïdale de 1 V avec un bruit de 10 mV en effectuant la recherche vers l'avant à partir de 500 ms et en utilisant une hystérésis de 40 mV :

```
Signal = @SineWave(10k; 10000; 50) + 0.01 * @Noise(10k;
         10000)
NVPos  = @NextValleyPos(Formula.Signal; 500m; 40m)
```

Voir aussi

<< @NextHillPos >> page 129, << @PrevHillPos >> page 141 et << @PrevValleyPos >> page 145

4.53 @Noise

Fonction
Génère une forme d'onde contenant du **bruit**.

Syntaxe
@Noise*(F échantillonnage; N échantillons)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage.
N échantillons	Nombre d'échantillons.

Sortie
Forme d'onde contenant du bruit.

Description
Cette fonction crée une forme d'onde contenant du bruit aléatoire avec une amplitude comprise entre -1 et 1. La fréquence d'échantillonnage et le nombre d'échantillons à générer doivent être définis.

La possibilité de générer du bruit peut s'avérer très utile pour étudier l'impact du bruit sur l'analyse d'une forme d'onde mesurée ou générée.

Exemple
L'exemple suivant génère une onde sinusoïdale d'une amplitude de 5 V masquée par un bruit de 10 V crête à crête.

```
Signal = 5 * @SineWave(10k; 1000; 50) + 10 *
         @Noise(10k; 1000)
```

Voir aussi
<< @Pulse >> page 147, << @Ramp >> page 151, << @SineWave >> page 167 et << @SquareWave >> page 172

4.54 @Not

Fonction
Cette fonction évalue les paramètres d'entrée à l'aide de l'opérateur logique **NOT**.

Syntaxe
@Not*(Param1)*

Paramètres

Param1	Nombre utilisé pour l'évaluation NOT.

Sortie
La sortie est 1 ou 0.

Description
La fonction @Not évalue le paramètre d'entrée à l'aide de l'opérateur logique NOT. Selon cette évaluation, le résultat est 1 ou 0. Une valeur numérique non nulle correspond à un « Vrai » logique et une valeur numérique nulle à un « Faux » logique.

La table de définition de la fonction NOT est la suivante :

Param1	Résultat
Vrai	Faux
Faux	Vrai

Exemple
Voici une liste d'exemples avec la valeur renvoyée dans chaque cas :

```
NotExampl1 = @Not(1)      => 0 (= false)
NotExampl2 = @Not(4)      => 0 (= false)
NotExampl3 = @Not(0)      => 1 (= true)
```

Voir aussi
<< @And >> page 43 et << @Or >> page 137

4.55 @Or

Fonction
Cette fonction évalue les paramètres d'entrée à l'aide de l'opérateur logique **OR**.

Syntaxe
@Or*(Param1; …; ParamN)*

Paramètres

Param1	Nombre : premier paramètre utilisé pour l'évaluation OR.
ParamN	Nombre : dernier paramètre utilisé pour l'évaluation OR. N >= 2

Sortie
La sortie est 1 ou 0.

Description
La fonction **@Or** évalue les paramètres d'entrée à l'aide de l'opérateur logique OR. Selon cette évaluation, le résultat est 1 ou 0. Une valeur numérique non nulle correspond à un « Vrai » logique et une valeur numérique nulle à un « Faux » logique.

La table de définition de la fonction OR est la suivante :

Param1	Param2	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Exemple

Voici une liste d'exemples avec la valeur renvoyée dans chaque cas :

```
OrExampl1 = @Or(1; 1)      => 1 (= true)
OrExampl2 = @Or(1;4;10)    => 1 (= true)
OrExampl3 = @Or(1;4;0;6)   => 1 (= true)
OrExampl4 = @Or(0;0)       => 0 (= false)
```

Voir aussi
<< @And >> page 43 et << @Not >> page 136

4.56 @Period

Fonction
Détermine la **période** d'une forme d'onde. La période est l'inverse de la fréquence.

Syntaxe
@Period*(Forme d'onde)*

@Period*(Forme d'onde; Début)*

@Period*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la période doit être déterminée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Description
La période de la forme d'onde ou du segment de forme d'onde est déterminée. Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres **Début** et **Fin** doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

La période est déterminée à l'aide d'algorithmes détectant les magnitudes de la base et du sommet. La moyenne de ces magnitudes est considérée comme le niveau zéro. Le nombre de passages par zéro entre le Début et la Fin est calculé. Tous les passages doivent se faire dans la même direction que le premier. Les passages sont déterminés avec une hystérésis autour du niveau zéro afin d'éliminer les effets du bruit. Cette hystérésis correspond à +/- 5 % de la différence entre les amplitudes maximale et de base.

La période de la forme d'onde est déterminée à partir de la différence de temps entre le premier et le dernier passages par zéro dans la même direction et du nombre de passages par zéro entre ces deux points.

Exemple
L'exemple suivant détermine la période d'un signal de 50 Hz :

```
Signal = @SineWave(10k; 10k; 50) + 0.01 * @Noise(10k;
         10k)
Period = @Period(Formula.Signal)
Freq   = 1 / Formula.Period
```

Voir aussi
<< @Frequency >> page 95

4.57 @Pow

Fonction
Élévation à une puissance ; cette opération mathématique, notée a^n, nécessite deux paramètres : la base « a » et l'exposant « n ».

Syntaxe
@Pow*(Base; Exposant)*

Paramètres

Base	Forme d'onde ou valeur numérique utilisée comme base.
Exposant	Forme d'onde ou valeur numérique utilisée comme exposant.

Sortie
Forme d'onde ou valeur numérique contenant la base élevée à la puissance de l'exposant.

Description
La fonction exponentielle calcule la valeur de la base élevée à l'exposant puissance.

Si la base et l'exposant sont tous deux des valeurs numériques, le résultat est une valeur numérique.

Si la base ou l'exposant est une forme d'onde, l'élévation à la puissance est calculée pour chaque échantillon avec la valeur numérique. Le résultat est une forme d'onde.

Remarque *Il n'est pas possible d'utiliser une forme d'onde à la fois pour la base et pour l'exposant.*

Exemple
Les résultats suivants sont équivalents :

```
Input    = 2 * @SineWave(10k; 10k; 10)
Result1  = @Exp(Formula.Input)
Result2  = @Pow(System.Constants.e; Formula.Input)
```

Voir aussi
<< @Exp >> page 63

4.58 @PrevHillPos

Fonction
Cette fonction recherche la position de la **valeur maximale locale précédente** dans une forme d'onde.

Syntaxe
@PrevHillPos*(Forme d'onde; Pos. départ; Hystérésis)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur maximale locale précédente doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer (vers l'arrière).
Hystérésis	Nombre : valeur de l'hystérésis de suppression du bruit.

Sortie
La sortie est la position X de la valeur maximale locale précédente.

Description
La position X de la valeur maximale locale précédente dans la forme d'onde est déterminée.

La recherche s'effectue vers la gauche en commençant à la position X définie par le paramètre Pos. départ. Si cette position se trouve après la fin de la forme d'onde, cette dernière est utilisée comme position de départ de la recherche.

La valeur d'hystérésis est utilisée pour éliminer les effets du bruit dans le signal. Par exemple, si un bruit de 100 mV crête à crête est présent dans un signal, le fait de définir une valeur d'hystérésis de 200 mV permet d'éviter que l'algorithme ne détecte une petite crête de bruit comme valeur maximale locale.

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine la position d'une valeur maximale locale dans une onde sinusoïdale de 1 V avec un bruit de 10 mV en effectuant la recherche vers l'arrière à partir de 500 ms et en utilisant une hystérésis de 40 mV :

```
Signal  = @SineWave(10k; 10000; 50) + 0.01 * @Noise(10k;
          10000)
PHPos   = @PrevHillPos(Formula.Signal; 500m; 40m)
```

Voir aussi

<< @NextHillPos >> page 129, << @NextValleyPos >> page 133 et
<< @PrevValleyPos >> page 145

4.59 @PrevLvlCross

Fonction
Détermine la position du **croisement précédent** d'une forme d'onde avec un **niveau** de signal donné.

Syntaxe
@PrevLvlCross*(Forme d'onde; Pos. départ; Niveau)*

@PrevLvlCross*(Forme d'onde; Pos. départ; Niveau; Pente)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position du franchissement de niveau précédent doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer (vers l'arrière).
Niveau	Nombre : niveau d'amplitude à rechercher.
Pente	Nombre : direction de la pente (-1, 0, 1)

Sortie
La sortie est la position X du franchissement de niveau précédent.

Description
La position X du croisement précédent de la forme d'onde avec le niveau défini et dans la direction spécifiée est déterminée.
La recherche s'effectue vers la gauche (dans la direction négative du temps) en commençant à la position X définie par le paramètre Pos. départ.

Le paramètre Pente contrôle le type de franchissement de niveau recherché :

- Pente = 1 : franchissement de niveau positif (du dessous vers le dessus du niveau)
- Pente = -1 : franchissement de niveau négatif (du dessus vers le dessous du niveau)
- Pente = 0 : tout franchissement (positif ou négatif)

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine le début et la fin de la dernière impulsion TTL du signal TTLSignal. Ce dernier est supposé être un signal réel.

```
TTLSignal = Active.Group1.Recorder_A.Ch_A1
TimeEnd   = @PrevLvlCrossing(Formula.TTLSignal; 1E20;
            2.5; 1)
TimeStart = @PrevLvlCrossing(Formula.TTLSignal;
            Formula.TimeEnd; 2.5; -1)
```

Voir aussi
<< @NextLvlCross >> page 131

4.60 @PrevValleyPos

Fonction
Cette fonction recherche la position de la **valeur minimale locale précédente** dans une forme d'onde.

Syntaxe
@PrevValleyPos*(Forme d'onde; Pos. départ; Hystérésis)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la position de la valeur minimale locale précédente doit être déterminée.
Pos. départ	Nombre : position de départ en unités X à laquelle la recherche doit commencer (vers l'arrière).
Hystérésis	Nombre : valeur de l'hystérésis de suppression du bruit.

Sortie
La sortie est la position X de la valeur minimale locale précédente.

Description
La position X de la valeur minimale locale précédente dans la forme d'onde est déterminée.

La recherche s'effectue vers la gauche en commençant à la position X définie par le paramètre Pos. départ. Si cette position se trouve après la fin de la forme d'onde, cette dernière est utilisée comme position de départ de la recherche.

La valeur d'hystérésis est utilisée pour éliminer les effets du bruit dans le signal. Par exemple, si un bruit de 100 mV crête à crête est présent dans un signal, le fait de définir une valeur d'hystérésis de 200 mV permet d'éviter que l'algorithme ne détecte une petite crête de bruit négative comme valeur minimale locale.

Remarque *La valeur renvoyée correspond à la position sur l'axe X de la forme d'onde en termes de coordonnée X, et non en termes de numéro d'échantillon. Les unités de cette valeur numérique sont les unités X de la forme d'onde.*

Exemple
L'exemple suivant détermine la position d'une valeur minimale locale dans une onde sinusoïdale de 1 V avec un bruit de 10 mV en effectuant la recherche vers l'arrière à partir de 500 ms et en utilisant une hystérésis de 40 mV :

```
Signal  = @SineWave(10k; 10000; 50) + 0.01 * @Noise(10k;
          10000)
PVPos   = @PrevValleyPos(Formula.Signal; 500m; 40m)
```

Voir aussi

<< @NextHillPos >> page 129, << @NextValleyPos >> page 133 et << @PrevHillPos >> page 141

4.61 @Pulse

Fonction
Génère une forme d'onde contenant une seule impulsion.

Syntaxe
@Pulse*(F échantillonnage ; N échantillons ; Début impulsion)*

@Pulse*(F échantillonnage ; N échantillons ; Début impulsion ; Largeur impulsion)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage
N échantillons	Nombre d'échantillons
Début impulsion	Numéro : échantillon auquel l'impulsion débute
Largeur impulsion	Numéro : largeur d'impulsion des échantillons ; la valeur par défaut est définie sur 1.

Sortie
Forme d'onde contenant une seule impulsion.

Description
Cette fonction crée une forme d'onde d'une impulsion. La fréquence d'échantillonnage, le nombre d'échantillons générant l'impulsion unique peuvent être définis. La longueur de la forme d'onde n'est pas limitée. L'amplitude de l'onde pulsée est de 1 V.

La fonction d'impulsion peut être utilisée pour déterminer la réponse impulsionnelle et indicielle d'une fonction de filtrage.

Cette fonction peut être utilisée pour synthétiser diverses formes d'onde. Ces données peuvent être utilisées comme entrées pour d'autres fonctions d'analyse.

Exemple

```
SignalPulse     = @Pulse(80k; 80k; 20k)
ImpulseResponse = @FilterButterworthLP
                  (Formula.SignalPulse; 2; 200)
```

```
SignalStep      = @Pulse(80k; 80k; 20k; 10k)
StepResponse    = @FilterButterworthLP
                  (Formula.SignalStep; 2; 200)
```

Si vous êtes uniquement intéressé par une partie du signal, vous pouvez utiliser la fonction d'impulsion pour que toutes les valeurs non comprises dans un intervalle spécifié soient ramenées à zéro. Cet intervalle peut être défini par les paramètres *Début impulsion* et *Largeur impulsion* de la fonction d'impulsion.

```
SignalInterval  = Formula.SignalStep *
                  Active.Group1.Recorder_A.Ch_A1
```

Voir aussi

<< @Noise >> page 135, << @SineWave >> page 167, << @SquareWave >> page 172 et << @Ramp >> page 151

4.62 @PulseWidth

Fonction
Détermine la **largeur** d'une impulsion dans une forme d'onde.

Syntaxe
@PulseWidth*(Forme d'onde)*

@PulseWidth*(Forme d'onde; Début)*

@PulseWidth*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée contenant l'impulsion.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Cette fonction s'applique « pendant l'enregistrement » lorsque les paramètres Début et Fin sont définis. Le résultat est calculé une fois les données entre le Début et la Fin disponibles.

Description
La largeur d'une impulsion correspond à la différence de temps entre le premier et le deuxième passages de l'impulsion à la magnitude 50 % dans la forme d'onde (ou le segment de forme d'onde).

Les passages par un pourcentage de magnitude donné sont calculés à l'aide des magnitudes de la base et du sommet. Ces magnitudes sont déterminées en recherchant deux populations dominantes dans l'histogramme de magnitude de la forme d'onde (ou du segment de forme d'onde) et en calculant les moyennes de ces deux populations.

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé.

Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant crée une impulsion et calcule sa largeur entre 50 % et 50 %. Le résultat est 150 ms :

```
Sig1      = @Ramp(1k; 100; 0; 0)
Sig2      = @Ramp(1k; 51; 0; 10)
Sig3      = @Ramp(1k; 99; 10; 10)
Sig4      = Sig4 x
Signal    = @Join(Formula.Sig1; Formula.Sig2;
            Formula.Sig3; Formula.Sig4; Formula.Sig1)
Width     = @PulseWidth(Formula.Signal)
```

Voir aussi
<< @FallTime >> page 66 et << @RiseTime >> page 161

4.63 @Ramp

Fonction
Génère une forme d'onde contenant une ou plusieurs **rampes** linéaires.

Syntaxe
@Ramp*(F échantillonnage; N échantillons; Y début; Y fin)*

@Ramp*(F échantillonnage; N échantillons; Y début; Y fin; Comptage)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage.
N échantillons	Nombre d'échantillons.
Y début	Nombre : valeur du premier échantillon.
Y fin	Nombre : valeur du dernier échantillon.
Comptage	Nombre de rampes.

Sortie
Forme d'onde contenant une ou plusieurs rampes linéaires.

Description
Cette fonction crée une forme d'onde contenant une ou plusieurs rampes linéaires, commençant à Y début et se terminant à Y fin. La fréquence d'échantillonnage et le nombre d'échantillons à générer peuvent également être définis. La longueur de la forme d'onde est limitée à 1 Giga-échantillons.

Lorsque le paramètre facultatif Comptage est défini, le nombre de rampes indiqué est généré. L'utilisation de ce paramètre entraîne la création d'une forme d'onde en dents de scie. S'il n'est pas défini, la valeur 0 est utilisée. Une seule rampe (et non une dent de scie) est alors créée.

La possibilité de combiner une fonction Ramp à la fonction @Join et à d'autres fonctions de génération de forme d'onde permet de synthétiser diverses formes d'onde. Les données simulées peuvent être utilisées comme entrées pour d'autres fonctions d'analyse.

Exemple
L'exemple suivant génère une impulsion simulée d'une amplitude de 10 V avec un bruit de 4 V crête à crête :

```
Sig1   = @Ramp(1k; 100; 0; 0)
Sig2   = @Ramp(1k; 51; 0; 10)
Sig3   = @Ramp(1k; 100; 10; 10)
Sig4   = @Ramp(1k; 51; 10; 0)
Signal = @Join(Formula.Sig1; Formula.Sig2;
         Formula.Sig3; Formula.Sig4) + 2 * @Noise(1k;
         302)
```

Voir aussi

<< @Noise >> page 135, << @Pulse >> page 147, << @SineWave >> page 167 et << @SquareWave >> page 172

4.64 @ReadAsciiFile

Fonction
Importe les données d'une forme d'onde depuis un fichier **ASCII** (texte).

Syntaxe
@ReadAsciiFile*(Nom de fichier)*

Paramètres

Nom de fichier Chaîne : chemin complet indiquant le nom et l'emplacement unique du fichier ASCII.

Sortie
La sortie est une forme d'onde.

Description
Cette fonction importe les données d'une forme d'onde depuis un fichier ASCII. Les échantillons des données importées sont considérés comme équidistants.

Le fichier ASCII doit contenir un en-tête et une section réservée aux données. Deux types d'en-tête différents sont pris en charge : une version **courte** et une version **longue**.

En-tête court :

Ligne	Description	Exemple
1	Nombre de lignes d'en-tête	5
2	Séparateur de données (point, virgule, tabulation ou point-virgule)	;
3	Nombre de paires de données	n
4	Facteur d'échelle pour x et y (x; y)	5.0E-8;2.44E+0
5	Unités pour x et y (x; y)	s;A

En-tête long :

Ligne	Description	Exemple
1	Nombre de lignes d'en-tête	12
2	Séparateur de données (point, virgule, tabulation ou point-virgule)	;
3	Nombre de paires de données	n

Ligne	Description	Exemple
4	Date de génération des données	17.03.00
5	Heure de génération des données	23:59
6	Informations supplémentaires concernant le logiciel ayant produit les données ; cette ligne peut être vierge. Pour des raisons de compatibilité uniquement.	TDG 0.5
7	Commentaires (80 caractères max.)	Premier exemple : Test 1 ;
8	Facteur d'échelle pour x et y (x; y)	5.0E-8;2.44E+0
9	Unités pour x et y (x; y)	s;A
10	Noms des unités des échelles (x; y)	temps;courant
11	Résolution des données y en bits	12
12	Utilisation de la plage dynamique en %	80

Données :

Les données viennent après l'en-tête et commencent à la ligne 6 ou 13 selon la taille de ce dernier.

Chaque ligne de données comporte une paire x, y. Le caractère séparant les valeurs x et y est défini à la ligne 2 de l'en-tête.

Les échantillons des données importées sont considérés comme équidistants.

La première ligne de données contient la première valeur x (la plus faible).

Exemple
L'exemple suivant montre comment est lu le fichier « ReferenceCurve.asc ».

```
Signal = @ReadAsciiFile("d:\data\ReferenceCurve.asc")
```

4.65 @Reduce

Fonction
Réduit le nombre d'échantillons d'une forme d'onde par rééchantillonnage.

Syntaxe
@Reduce*(Forme d'onde; Facteur)*

Paramètres

Forme d'onde	Forme d'onde à rééchantillonner.
Facteur	Nombre : facteur de rééchantillonnage.

Sortie
Forme d'onde rééchantillonnée.

Description
Une forme d'onde est rééchantillonnée pour réduire la quantité de données. Le facteur de rééchantillonnage est arrondi à l'entier N le plus proche et doit être compris dans la plage 2, 3, ..., (longueur de la forme d'onde)/2. La forme d'onde de sortie reçoit l'échantillon 1, l'échantillon 1+N, l'échantillon 1+2N et ainsi de suite jusqu'à ce qu'il n'y ait plus d'échantillons. La longueur de la forme d'onde de sortie est N fois plus petite que celle de la forme d'onde d'entrée. L'intervalle d'échantillonnage de la forme d'onde de sortie est N fois celui de la forme d'onde d'entrée. La coordonnée X du premier échantillon reste la même.

Le rééchantillonnage peut provoquer des effets de repliement lorsque la forme d'onde contient des composantes haute fréquence. Dans ce cas, utiliser la fonction de lissage @Smooth pour appliquer un filtre passe-bas aux données avant le rééchantillonnage.

Exemple
L'exemple suivant rééchantillonne une onde sinusoïdale selon un facteur de 10, ce qui a pour effet de réduire la fréquence d'échantillonnage d'un facteur de 10. Le nombre d'échantillons passe de 1 000 à 100.

```
Sine1000  = @SineWave(10k; 1000; 50)
Sine100   = @Reduce(Formula.Sine1000; 10)
```

Voir aussi
<< @Res2 >> page 159 et << @Smooth >> page 169

4.66 @RefCheck

Fonction
Compare une forme d'onde ou un segment de forme d'onde à une ou deux formes d'onde de référence.

Syntaxe
@RefCheck*(Forme d'onde; Enveloppe supérieure)*

@RefCheck*(Forme d'onde; Enveloppe supérieure; Enveloppe inférieure)*

@RefCheck*(Forme d'onde; Enveloppe supérieure; Enveloppe inférieure; Début)*

@RefCheck*(Forme d'onde; Enveloppe supérieure; Enveloppe inférieure; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée à comparer.
Enveloppe supérieure	Forme d'onde contenant l'enveloppe supérieure.
Enveloppe inférieure	Forme d'onde contenant l'enveloppe inférieure.
Début	Nombre : début du segment à comparer.
Fin	Nombre : fin du segment à comparer.

Sortie
Nombre indiquant une réussite (entre les formes d'onde de référence) ou un échec.

Description
Cette fonction permet de comparer une forme d'onde à une ou deux formes d'onde de référence. La sortie correspond à la position X du premier croisement de la forme d'onde d'entrée avec la forme d'onde supérieure ou inférieure.

Si la forme d'onde d'entrée reste entre les formes d'onde supérieure et inférieure, le système renvoie la valeur « Pas un nombre » (Nan).

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé.

Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Exemple
L'exemple suivant crée une onde sinusoïdale et deux ondes sinusoïdales « enveloppes ». Le bruit est utilisé pour provoquer une réussite ou un échec. Une valeur de bruit plus élevée entraîne un échec.

```
Signal  = @SineWave(8000; 8001; 5)
Upper   = Formula.Signal + 0.1
Lower   = Formula.Signal - 0.1
Noise   = 0.1 * @Noise(8000; 8001)
Signal2 = Formula.Signal + Formula.Noise
Check   = @RefCheck(Formula.Signal2; Formula.Upper;
          Formula.Lower)
Result  = @IIF(Formula.Check; "NOT OK"; "OK")
```

4.67 @RemoveGlitch

Fonction
Renvoie une forme d'onde avec de nouvelles données entre des heures de début et de fin spécifiées, en utilisant une ligne droite entre ces deux points.

Syntaxe
@RemoveGlitch*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont le transitoire doit être supprimé.
Début	Nombre : début du transitoire
Fin	Nombre : fin du transitoire

Sortie
La sortie est une forme d'onde de même longueur que la forme d'onde d'origine.

Description
La forme d'onde de sortie présente une ligne droite entre le Début et la Fin. Le reste de la forme d'onde d'origine reste tel quel.

Remarque *Si cette formule est utilisée pour supprimer un transitoire pendant l'enregistrement, les paramètres Début et Fin doivent se trouver dans les limites des données disponibles. Si l'un d'eux n'a pas de valeur correspondante dans la forme d'onde, la fonction RemoveGlitch ne peut pas être calculée.*

Exemple
L'exemple suivant crée une onde sinusoïdale de 50 Hz d'une durée de 2 secondes. La fonction crée ensuite une nouvelle forme d'onde de 2 secondes dans laquelle les échantillons se trouvant dans l'intervalle de 1 à 1,5 seconde sont remplacés par une ligne droite.

```
Signal  = @SineWave(1k; 2000; 50)
Cropped = @RemoveGlitch(Formula.Signal; 1000m; 1500m)
```

4.68 @Res2

Fonction
Rééchantillonne une forme d'onde de sorte que sa longueur devienne une puissance de deux.

Syntaxe
@Res2*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde à rééchantillonner.

Sortie
Forme d'onde rééchantillonnée.

Description
Cette fonction rééchantillonne une forme d'onde de sorte que le nouveau nombre d'échantillons soit égal à la puissance de deux suivante supérieure ou égale à la longueur d'origine. Elle peut être utilisée pour calculer les spectres de n'importe quel segment d'une forme d'onde. En effet, le nombre d'échantillons dans un tel segment n'est généralement pas égal à une puissance de deux, condition requise pour un algorithme FFT. En appliquant d'abord la fonction @Res2 à ce segment, une nouvelle forme d'onde contenant le même signal mais échantillonnée à une fréquence supérieure est créée de sorte que le nombre d'échantillons soit égal à une puissance de deux. Cette forme d'onde rééchantillonnée peut alors être utilisée sans être tronquée ou remplie de zéros pour des calculs FFT.

Le facteur de rééchantillonnage n'étant généralement pas un entier, des valeurs doivent être générées entre deux échantillons de la forme d'onde. Une interpolation linéaire est utilisée à cette fin.
L'intervalle d'échantillonnage de la forme d'onde est corrigé pour correspondre au facteur de rééchantillonnage.

Exemple
L'exemple suivant génère une onde sinusoïdale de 2 500 échantillons et la rééchantillonne pour créer une forme d'onde de 4 096 échantillons. La valeur de Length (Longueur) sera de 4 096.

```
Signal     = @SineWave(10k; 2500; 100)
Resampled  = @Res2(Formula.Signal)
Length     = @Length(Formula.Resampled)
```

Voir aussi
<< @Reduce >> page 155

4.69 @RiseTime

Fonction
Détermine le **temps de montée** d'une impulsion dans une forme d'onde.

Syntaxe
@RiseTime*(Forme d'onde)*

@RiseTime*(Forme d'onde; Début)*

@RiseTime*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée contenant le front montant d'une impulsion.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Cette fonction s'applique « pendant l'enregistrement » lorsque les paramètres Début et Fin sont définis. Le résultat est calculé une fois les données entre le Début et la Fin disponibles.

Description
Le temps de montée correspond à la différence de temps entre le point proximal (passage à la magnitude 10 %) et le point distal (passage à la magnitude 90 %) du premier front montant d'une impulsion dans la forme d'onde (ou le segment de forme d'onde).

Les passages par un pourcentage de magnitude donné sont calculés à l'aide des magnitudes de la base et du sommet. Ces magnitudes sont déterminées en recherchant deux populations dominantes dans l'histogramme de magnitude de la forme d'onde (ou du segment de forme d'onde) et en calculant les moyennes de ces deux populations.

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant crée une impulsion et calcule le temps de montée de 10 % à 90 % du front montant. Le résultat est 40 ms :

```
Sig1   = @Ramp(1k; 100; 0; 0)
Sig2   = @Ramp(1k; 51; 0; 10)
Sig3   = @Ramp(1k; 100; 10; 10)
Sig4   = @Ramp(1k; 51; 10; 0)
Signal = @Join(Formula.Sig1; Formula.Sig2;
         Formula.Sig3; Formula.Sig4; Formula.Sig1)
Rise   = @RiseTime(Formula.Signal)
```

Voir aussi
<< @FallTime >> page 66 et << @PulseWidth >> page 149

4.70 @RMS

Fonction
Calcule la **valeur moyenne quadratique** d'une forme d'onde.

Syntaxe
@RMS*(Forme d'onde)*

@RMS*(Forme d'onde; Début)*

@RMS*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont la valeur moyenne quadratique doit être calculée.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Description
La valeur moyenne quadratique est calculée à l'aide de la formule suivante :

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{n=n_1}^{n_2} y^2(n)} \text{ with } \mathrm{N} = (n_2 - n_1 + 1)$$

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé. Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres Début et Fin doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple
L'exemple suivant crée une onde sinusoïdale de 50 Hz avec une amplitude de 10 V et calcule la valeur moyenne quadratique en utilisant l'ensemble de la forme d'onde :

```
Signal  = 10.0 * @SineWave(20k; 1000; 50)
RMSAmpl = @RMS(Formula.Signal)
```

Voir aussi
<< @Energy >> page 60, << @Mean >> page 122 et << @StdDev >> page 173

4.71 @SAEJ211Filter

Fonction
Filtre le signal d'entrée conformément à la norme **SAE J211**.

Syntaxe
@SAEJ211Filter*(Signal; CFC)*

Paramètres

Signal	Forme d'onde d'entrée
CFC	Numéro : classe de fréquence

Sortie
La sortie correspond à la forme d'onde filtrée.

Description
Cette fonction applique un filtre passe-bas numérique pour éliminer le bruit haute fréquence des données brutes. Il s'agit d'un filtre Butterworth passe-bas sans phase à 4 pôles. Son type est conforme à la norme SAE J211 très utilisée dans l'industrie automobile.

La relation entre la fréquence de coupure (Fc) et la classe de fréquence (CFC, Channel Frequency Class) est définie comme suit :

*Fc = CFC * 1,67.*

La fréquence de coupure doit être supérieure ou égale au quart de la fréquence d'échantillonnage.

Exemple
L'exemple suivant génère une onde sinusoïdale avec du bruit haute fréquence. Le signal est filtré à l'aide de la fonction SAEJ211Filter :

```
Signal    = 5 * @SineWave(80000; 1000; 400)
          + @Noise(80000; 1000)
Filtered  = @SAEJ211Filter(Formula.Signal; 300)
```

La fréquence de coupure est : Fc = 1.67 * 300 = 500 Hz

4.72 @Sin

Fonction
Calcule le **sinus** du paramètre d'entrée.

Syntaxe
@Sin*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant le sinus de l'entrée.

Description
La fonction trigonométrique sinus est calculée en considérant que le paramètre d'entrée correspond à l'angle en radians. Si un paramètre de forme d'onde est utilisé, le sinus est calculé pour chacun des échantillons.

Exemple
L'exemple suivant calcule le sinus de la variable « Angle » définie en degrés :

```
Angle     = 36
AngleRad  = System.Constants.Pi * Formula.Angle / 180
SinAngle  = @Sin(Formula.AngleRad)
```

Voir aussi
<< @ATan >> page 46, << @Cos >> page 51 et << @Tan >> page 176

4.73 @SineWave

Fonction
Génère une forme d'onde décrite par la fonction **sinus**.

Syntaxe
@SineWave*(F échantillonnage; N échantillons; F signal)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage.
N échantillons	Nombre d'échantillons.
F signal	Nombre : fréquence du signal.

Sortie
Forme d'onde contenant une onde sinusoïdale.

Description
Cette fonction crée une onde sinusoïdale. La fréquence d'échantillonnage et le nombre d'échantillons générant une fréquence d'onde sinusoïdale peuvent être définis. La longueur de la forme d'onde n'est pas limitée. L'amplitude de l'onde sinusoïdale est de 1 V.

Cette fonction peut être utilisée pour synthétiser diverses formes d'onde. Ces données peuvent être utilisées comme entrées pour d'autres fonctions d'analyse.

Exemple
L'exemple suivant génère un segment de forme d'onde de 100 ms contenant une onde sinusoïdale simulée de 50 Hz d'une amplitude de 10 V échantillonnée à 10 kHz avec un bruit de 4 V crête à crête :

```
Signal = 10 * @SineWave(10k; 1000; 50) + 2 *
         @Noise(10k; 1000)
```

Il est également possible d'additionner plusieurs ondes sinusoïdales pour générer des formes d'onde plus complexes :

```
Signal2  = 5 * @SineWave(10k; 1000; 50)
         + 7 * @SineWave(10k; 1000; 60)
         + 12 * @SineWave(10k; 1000; 90)
```

Voir aussi

<< @Noise >> page 135, << @Pulse >> page 147, << @Ramp >> page 151 et << @SquareWave >> page 172

4.74 @Smooth

Fonction
Lissage d'une forme d'onde à l'aide d'un filtre à moyenne mobile.

Syntaxe
@Smooth*(Forme d'onde; N)*

Paramètres

Forme d'onde	Forme d'onde à lisser.
N	Nombre d'échantillons du facteur de lissage.

Sortie
Forme d'onde lissée.

Description
La forme d'onde est lissée en calculant une moyenne non pondérée mobile à partir du nombre d'échantillons défini selon la formule suivante :

$$\mathrm{Output(n)} = \frac{1}{N} \sum_{k=n-N'}^{k=n+N'} Input\,(k)$$

Avec :

N = facteur de lissage

N' = (N-1)/2

n = nombre d'échantillons compris entre 1 et la fin de la forme d'onde

Le facteur de lissage N définit le nombre d'échantillons utilisés pour calculer la moyenne. Si N n'est pas un entier impair, il est d'abord arrondi à l'entier impair le plus proche. Si la valeur est inférieure à 3, elle est définie sur 3. Si elle est supérieure à 1 001, elle est définie sur 1 001. Plus la valeur de N est élevée, plus l'effet du lissage est prononcé.

Le lissage permet d'éliminer efficacement les composantes haute fréquence. Le filtrage est optimal lorsque le nombre d'échantillons est égal à un multiple entier de la période des perturbations. En raison de la nature symétrique du lissage, la position des caractéristiques propres à la forme d'onde reste la même.

La longueur de la forme d'onde de sortie est identique à celle de la forme d'onde d'entrée. Pour le lissage à proximité des bords, la fonction suppose que la forme d'onde se prolonge avec des valeurs identiques à celles du bord.

Si cela est nécessaire, un lissage triangulaire pondéré peut être obtenu en lissant deux fois la forme d'onde. Pour n'éliminer que les perturbations haute fréquence, il est souvent préférable d'appliquer plusieurs fois la fonction @Smooth avec un nombre d'échantillons réduit plutôt que de ne l'appliquer qu'une seule fois avec un nombre d'échantillons élevé.

Exemple
L'exemple suivant extrait le bruit haute fréquence d'une onde sinusoïdale de 50 Hz en soustrayant le signal lissé du signal d'origine :

```
Mains  = 220 * @SineWave(200k; 4096; 50)
       + 10 * @Noise(200k; 4096)
Smo1   = @Smooth(Formula.Mains; 55)
Noise  = Formula.Mains - Formula.Smo1
```

Voir aussi
<< @Integrate >> page 103

4.75 @Sqrt

Fonction
Calcule la **racine carrée** du paramètre d'entrée.

Syntaxe
@Sqrt*(Par)*

Paramètre

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant la racine carrée de l'entrée.

Description
Cette fonction calcule la racine carrée du paramètre d'entrée.

Si un paramètre de forme d'onde est utilisé, la racine carrée est calculée pour chacun des échantillons. Les valeurs négatives d'une forme d'onde produisent la racine carrée négative de la valeur absolue. Pour les valeurs numériques, la racine carrée d'une valeur négative n'existe pas.

Exemple
Voici des exemples de calcul de racine carrée possibles :

```
Signal   = 4 * @SineWave(10k; 1000; 50)
Result   = @Sqrt(Formula.Signal)
Five     = @Sqrt(25)
NoValue  = @Sqrt(-1)
```

Voir aussi
<< @Pow >> page 140

4.76 @SquareWave

Fonction
Génère une forme d'onde contenant une **onde carrée**.

Syntaxe
@SquareWave*(F échantillonnage; N échantillons; N fréquence)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage de la forme d'onde.
N échantillons	Nombre d'échantillons dans la forme d'onde.
N fréquence	Nombre : fréquence de l'onde carrée générée.

Sortie
Forme d'onde contenant une onde carrée.

Description
Cette fonction crée une onde carrée. La fréquence d'échantillonnage, le nombre d'échantillons à générer et la fréquence de l'onde carrée peuvent être définis. La longueur de la forme d'onde n'est pas limitée. L'amplitude de l'onde carrée est de 1 V.

La possibilité de générer une fonction d'onde carrée permet de synthétiser diverses formes d'onde. Les données simulées peuvent être utilisées comme entrées pour d'autres fonctions d'analyse.

Exemple
L'exemple suivant génère un segment de forme d'onde de 100 ms contenant une onde carrée simulée de 50 Hz d'une amplitude de 10 V échantillonnée à 10 kHz avec un bruit de 2 V crête à crête :

```
Signal  = 10 * @SquareWave(10k;1000;50)
        + 2 * @Noise(10k;1000)
```

Voir aussi
<< @Noise >> page 135, << @Pulse >> page 147, << @Ramp >> page 151 et << @SineWave >> page 167

4.77 @StdDev

Fonction
Calcule l'**écart type** d'une forme d'onde.

Syntaxe
@StdDev*(Forme d'onde)*

@StdDev*(Forme d'onde; Début)*

@StdDev*(Forme d'onde; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde d'entrée dont l'écart type doit être calculé.
Début	Nombre : début du segment.
Fin	Nombre : fin du segment.

Sortie
La sortie est une valeur numérique.

Description
L'écart type est une caractéristique statistique d'une forme d'onde. Il correspond approximativement à l'écart quadratique moyen des échantillons d'une forme d'onde par rapport à la valeur moyenne de cette dernière et constitue donc une mesure de la dispersion de ses valeurs.

L'écart type est calculé à l'aide de la formule suivante :

$$\mathrm{StdDev} = \sqrt{\frac{1}{N-1}\sum_{n=n_1}^{n_2}(y(n)-\overline{y})^2}$$

$$\text{in which: } \overline{\mathrm{y}} = \frac{1}{N}\sum_{n=n_1}^{n_2} y(n)$$

$$\text{with } \mathrm{N} = (n_2 - n_1 + 1)$$

Les limites de segment (Début et Fin) sont utilisées pour sélectionner une plage d'échantillons. Si aucune limite de segment n'est définie, l'ensemble de la forme d'onde est utilisé.

Si seul le Début est défini, le segment de forme d'onde du Début à la fin de la forme d'onde est utilisé.

Remarque *Les paramètres **Début** et **Fin** doivent être définis en utilisant l'unité de l'axe horizontal (le temps, par exemple) et non les numéros des échantillons.*

Exemple

L'exemple suivant crée une onde sinusoïdale de 50 Hz et d'une amplitude efficace de 1 au-dessus d'un composant c.c. de 100. L'écart type est déterminé à partir de ce signal :

```
Signal  = 100 + @Sqrt(2) * @SineWave(20k; 2000; 50)
StdDev  = @StdDev(Formula.Signal)
```

Voir aussi

<< @Energy >> page 60, << @Mean >> page 122 et << @RMS >> page 163

4.78 @Sweep

Fonction
Sélectionne un **transitoire** spécifique dans un enregistrement à plusieurs transitoires.

Syntaxe
@Sweep*(Signal; N transitoire)*

Paramètres

Signal	Forme d'onde : enregistrement à plusieurs transitoires.
N transitoire	Numéro (entier) du transitoire à sélectionner.

Sortie
Forme d'onde ne contenant que le transitoire sélectionné.

Description
Certains réglages matériels et logiciels des enregistreurs permettent de créer des enregistrements à plusieurs transitoires. Chaque événement de trigger valide crée un transitoire. La forme d'onde d'une voie d'un enregistrement à plusieurs transitoires contient plusieurs transitoires.

La fonction @Sweep permet de sélectionner un transitoire spécifique de la forme d'onde. Le paramètre N transitoire est une valeur numérique comprise entre 1 et le nombre de transitoires.

Remarque *La forme d'onde sélectionnée ne contient qu'un transitoire.*

Exemple
Cet exemple calcule la valeur maximale de la pression dans le deuxième transitoire de l'enregistrement.

```
PressureSweep = @Sweep
                (Active.Group1.Recorder_A.Ch_A2; 2)
MaxPressure   = @Max(Formula.PressureSweep)
```

4.79 @Tan

Fonction
Calcule la **tangente** du paramètre d'entrée.

Syntaxe
@Tan*(Par)*

Paramètres

Par	Forme d'onde ou valeur numérique d'entrée.

Sortie
Forme d'onde ou valeur numérique contenant la tangente de l'entrée.

Description
La fonction trigonométrique tangente est calculée en considérant que le paramètre d'entrée correspond à l'angle en radians. Lorsqu'un paramètre de forme d'onde est utilisé, la tangente est calculée pour chacun des échantillons.

@ATan est la fonction trigonométrique inverse de @Tan.

Exemple
L'exemple suivant calcule la tangente de la variable « Angle » définie en degrés :

```
Angle    = 72
AngleRad = System.Constants.Pi * Formula.Angle / 180
TanAngle = @Tan(Formula.AngleRad)
```

Voir aussi
<< @ATan >> page 46, << @Cos >> page 51 et << @Sin >> page 166

4.80 @TriggerTime

Fonction
Renvoie la position du **trigger** d'un transitoire sélectionné.

Syntaxe
@TriggerTime*(Forme d'onde)*

@TriggerTime*(Forme d'onde; N transitoire)*

Paramètres

Forme d'onde	Forme d'onde contenant au moins un transitoire.
N transitoire	Nombre : transitoire sélectionné (valeur par défaut = 1).

Sortie
Nombre représentant la position du trigger en unités X.

Description
Certains réglages matériels et logiciels des enregistreurs permettent de créer des enregistrements à plusieurs transitoires. Chaque événement de trigger valide crée un transitoire. La forme d'onde d'une voie d'un enregistrement à plusieurs transitoires contient les informations relatives aux positions temporelles des triggers des différents transitoires.

La fonction @TriggerTime permet d'obtenir la position en unités X du trigger correspondant au numéro de transitoire défini.

Cette fonction peut par exemple être utilisée en conjonction avec l'affichage « Relecture de transitoires ». Ce type d'affichage ne présente en effet qu'un seul transitoire, mais il peut passer d'un transitoire à un autre. La position du trigger du transitoire affiché est toujours référencée par rapport à zéro. Les valeurs de curseur sont quant à elles référencées par rapport à la position du trigger du transitoire. Lorsqu'un calcul impliquant une position de curseur doit être effectué, l'emplacement réel du curseur dans la forme d'onde d'origine doit être calculé à l'aide de la position du trigger correspondant.

Exemple
Cet exemple illustre le calcul de la valeur maximale de la tension 20 ms autour de la position du trigger du deuxième transitoire d'un enregistrement à plusieurs transitoires.

```
Signal      = Active.Group1.Recorder_A.Ch_A2
```

```
TPos       = @TriggerTime(Formula.Signal; 2)
MaxVoltage = @Max(Formula.Signal; Formula.TPos - 10m;
             Formula.TPos + 10m)
```

Voir aussi

<< @TriggerTimeToText >> page 179

4.81 @TriggerTimeToText

Fonction
Renvoie la position du **trigger** du transitoire sélectionné dans une chaîne formatée contenant la **date et l'heure**.

Syntaxe
@TriggerTimeToText*(Forme d'onde)*

@TriggerTimeToText*(Forme d'onde; N transitoire)*

@TriggerTimeToText*(Forme d'onde; N transitoire; Décimales)*

@TriggerTimeToText*(Forme d'onde; N transitoire; Décimales; Format heure)*

@TriggerTimeToText*(Forme d'onde; N transitoire; Décimales; Format heure; Format date)*

Paramètres

Forme d'onde	Forme d'onde contenant au moins un transitoire	
N transitoire	Nombre : transitoire sélectionné (valeur par défaut = 1)	
Décimales	Nombre de décimales de la partie fractionnaire des secondes, de 0 à 9. Par défaut = 3	
TimeFormat	« Relatif » (par défaut)	Durée écoulée depuis le début de l'enregistrement
	« Local »	Heure locale
	« UTC »	Temps universel coordonné (UTC)
Format date	« Aucun » (par défaut)	Aucune information de date
	« Court »	Représentation sous forme de chaîne de date courte
	« Long »	Représentation sous forme de chaîne de date longue

Sortie
La sortie est une chaîne représentant la position du trigger.

Description
Différents réglages matériels et logiciels des enregistreurs permettent de créer des enregistrements à plusieurs transitoires. Chaque événement de trigger valide crée un transitoire. La forme d'onde d'une voie d'un enregistrement à plusieurs transitoires contient les informations relatives aux positions temporelles des triggers des différents transitoires.

La fonction @TriggerTimeToText permet d'obtenir la position temporelle du trigger d'un transitoire donné sous forme de chaîne de texte.

Cette fonction peut par exemple être utilisée en conjonction avec l'affichage « Relecture de transitoires » de Perception. Ce dernier n'affiche en effet qu'un seul transitoire, mais il peut passer d'un transitoire à un autre. La position du trigger de ce transitoire est toujours référencée par rapport à zéro ; toutefois, avec la fonction TriggerTimeToText, il est possible de présenter cette position par rapport à d'autres points et dans différents formats.

Les formats de la date et de l'heure dépendent des paramètres régionaux de Windows et peuvent donc varier d'une machine à l'autre. Consulter l'aide de Windows pour plus de détails.

Remarque *Cette fonction ne s'applique qu'aux enregistrements réalisés avec la base de temps interne, ce qui signifie que la dimension de l'axe X est le temps. Les chaînes contenant la date et l'heure ne sont pas sensibles à la casse. La valeur renvoyée est une chaîne pouvant être utilisée et interprétée par d'autres programmes (tels qu'Excel) en tant que chaîne.*

Exemple
Cet exemple affiche la position temporelle du trigger du deuxième transitoire au format UTC.

```
Signal     = Active.Group1.Recorder_A.Ch_A2
TriggerUTC = @TriggerTimeToText
             (Formula.Signal; 2; 6; "UTC"; "Short")
```

Exemple de chaîne de sortie : « 3-3-2008 10:40:45.301455 »

Si seule l'heure est requise :

```
TriggerTime = @TriggerTimeToText
              (Formula.Signal; 2; 0; "Local"; "none")
```

Exemple de résultat : « 11:40:45 »

Voir aussi
<< @TriggerTime >> page 177

4.82 @TrueRMS

Fonction
Renvoie une forme d'onde représentant la **valeur moyenne quadratique** vraie pour un nombre de cycles donné.

Syntaxe
@TrueRMS*(Forme d'onde)*

@TrueRMS*(Forme d'onde; Cycles; Début; Fin)*

Paramètres

Forme d'onde	Forme d'onde utilisée pour le calcul de la valeur moyenne quadratique.
Cycles	Nombre de cycles à utiliser pour le calcul de la valeur moyenne quadratique.
Début	Nombre : début du segment
Fin	Nombre : fin du segment

Sortie
Forme d'onde contenant la valeur moyenne quadratique pour le nombre de cycles défini.

Description
Cette fonction crée une forme d'onde dont la fréquence d'échantillonnage est identique à celle de la forme d'onde d'origine. La valeur réelle de la forme d'onde créée correspond à la valeur moyenne quadratique vraie de la forme d'onde d'entrée.
La valeur par défaut est 1 cycle. Les autres valeurs sont converties en un multiple de 0,5 cycle. Cette fonction utilise le passage par zéro pour déterminer un cycle. Utiliser le paramètre Cycles pour définir le nombre de cycles (multiples de 0,5) à utiliser pour le calcul.

Exemple
L'exemple suivant crée une forme d'onde représentant la valeur moyenne quadratique vraie de la tension pour chaque cycle. La valeur moyenne quadratique moyenne est calculée pour l'ensemble du signal.

```
RMS_Signal = @TrueRMS
             (Active.Group1.Recorder_Voltage.Voltage;1)
AverageRMS = @Mean(Formula.RMS_Signal)
```

Voir aussi
<< @RMS >> page 163

4.83 @Value

Fonction
Renvoie la **valeur** de l'amplitude d'une forme d'onde à une position X donnée.

Syntaxe
@Value*(Forme d'onde; Pos X)*

Paramètres

Forme d'onde	Forme d'onde dont la valeur doit être déterminée à la position X définie.
Pos X	Nombre : position X à laquelle la valeur de la forme d'onde doit être déterminée.

Sortie
Valeur de l'amplitude de la forme d'onde à la position X définie.

Description
Cette fonction détermine la valeur de l'amplitude d'une forme d'onde à une position X donnée. Cette dernière doit être définie par une coordonnée X, et non par un numéro d'échantillon. Si la coordonnée X se trouve entre deux points d'échantillonnage, une interpolation linéaire est utilisée pour déterminer la valeur à cette position X.

Si la position X se trouve avant le premier échantillon ou après le dernier échantillon de l'ensemble de données de la forme d'onde, la fonction ne renvoie aucune valeur.

Exemple
L'exemple suivant recherche dans un signal nommé Marker (Marqueur) le franchissement vers le haut du niveau 2,5 et enregistre le temps trouvé dans la variable TimeMarker. La valeur d'un autre signal est ensuite déterminée à cette position. Les signaux sont supposés être réels.

```
Marker      = Active.Group1.Recorder_A.Ch_A1
OtherSignal = Active.Group1.Recorder_A.Ch_A2
TimeMarker  = @NextLvlCross(Formula.Marker; 0; 2.5; 1)
ValueAtMark = @Value(Formula.OtherSignal;
              Formula.TimeMarker)
```

4.84 @XDelta

Fonction
Renvoie l'intervalle d'échantillonnage (durée entre les échantillons) d'une forme d'onde.

Syntaxe
@XDelta*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde d'entrée.

Sortie
La sortie est une valeur numérique.

Description
Cette fonction renvoie l'intervalle d'échantillonnage (période) d'une forme d'onde. L'intervalle d'échantillonnage est l'inverse de la fréquence d'échantillonnage et représente la différence en unités X entre deux échantillons consécutifs.

Exemple
L'exemple suivant crée une onde sinusoïdale avec une fréquence d'échantillonnage de 1 kHz. La variable dx reçoit l'intervalle d'échantillonnage de ce signal (1 ms). La variable sf reçoit la fréquence d'échantillonnage :

```
Signal = 25 * @SineWave(1k; 1000; 50)
dx     = @XDelta(Formula.Signal)
sf     = 1.0 / @XDelta(Formula.Signal)
```

Voir aussi
<< @Length >> page 111, << @XFirst >> page 188 et << @XLast >> page 189,

4.85 @XDeltaHigh

Fonction
Détermine **le plus grand** intervalle d'échantillonnage d'une forme d'onde.

Syntaxe
@XDeltaHigh*(Forme d'onde)*

Paramètres

Forme d'onde	Forme d'onde dont le plus grand intervalle d'échantillonnage doit être déterminé.

Sortie
La sortie est une valeur numérique : la différence en unités X entre deux échantillons.

Description
Cette fonction détermine le plus grand intervalle d'échantillonnage lorsque des formes d'onde à plusieurs bases de temps sont utilisées. L'intervalle d'échantillonnage est l'inverse de la fréquence d'échantillonnage et représente la différence en unités X entre deux échantillons consécutifs.

Exemple
Cet exemple crée une forme d'onde contenant 2 fréquences d'échantillonnage différentes. Les fonctions XDelta sont utilisées pour déterminer la fréquence d'échantillonnage.

```
SignalLow  = @SineWave(10k;10k;50)
SignalHigh = @SineWave(100k;10k;50)
Signal_LHL = @Join(Formula.SignalLow;
             Formula.SignalHigh; Formula.SignalLow)
XDelta     = @XDelta(Formula.Signal_LHL)
XHigh      = @XDeltaHigh(Formula.Signal_LHL)
XLow       = @XDeltaLow(Formula.Signal_LHL)
```

Voir aussi
<< @XDelta >> page 185 et << @XDeltaLow >> page 187

4.86 @XDeltaLow

Fonction
Détermine **le plus petit** intervalle d'échantillonnage d'une forme d'onde.

Syntaxe
@XDeltaLow*(Forme d'onde)*

Paramètres

Forme d'onde	Forme d'onde dont le plus grand intervalle d'échantillonnage doit être déterminé.

Sortie
La sortie est une valeur numérique : la différence en unités X entre deux échantillons.

Description
Cette fonction détermine le plus petit intervalle d'échantillonnage lorsque des formes d'onde à plusieurs bases de temps sont utilisées. L'intervalle d'échantillonnage est l'inverse de la fréquence d'échantillonnage et représente la différence en unités X entre deux échantillons consécutifs.

Exemple
Cet exemple crée une forme d'onde contenant 2 fréquences d'échantillonnage différentes. Les fonctions XDelta sont utilisées pour déterminer la fréquence d'échantillonnage.

```
SignalLow  = @SineWave(10k;10k;50)
SignalHigh = @SineWave(100k;10k;50)
Signal_LHL = @Join(Formula.SignalLow;
             Formula.SignalHigh; Formula.SignalLow)
XDelta     = @XDelta(Formula.Signal_LHL)
XHigh      = @XDeltaHigh(Formula.Signal_LHL)
XLow       = @XDeltaLow(Formula.Signal_LHL)
```

Voir aussi
<< @XDelta >> page 185 et << @XDeltaHigh >> page 186

4.87 @XFirst

Fonction
Renvoie la **coordonnée X** du **premier** échantillon d'une forme d'onde.

Syntaxe
@XFirst*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde d'entrée.

Sortie
La sortie est une valeur numérique.

Description
La coordonnée X du premier échantillon est renvoyée. La coordonnée X est une valeur dont l'unité est celle de l'axe horizontal de la forme d'onde (généralement le temps).

Exemple
L'exemple suivant crée une onde sinusoïdale. Les signaux générés commencent toujours au temps 0. La forme d'onde est décalée horizontalement de 100 ms. La coordonnée X du premier échantillon de cette forme d'onde décalée sera donc 100 ms.

```
Signal  = @SineWave(10k; 1000; 50)
Shifted = @XShift(Formula.Signal; 100m)
Start   = @XFirst(Formula.Shifted)
```

Voir aussi
<< @Length >> page 111, << @XDelta >> page 185, << @XLast >> page 189 et
<< @XShift >> page 190

4.88 @XLast

Fonction
Renvoie la **coordonnée X** du **dernier** échantillon d'une forme d'onde.

Syntaxe
@XLast*(Forme d'onde)*

Paramètres

Forme d'onde Forme d'onde d'entrée.

Sortie
La sortie est une valeur numérique.

Description
La coordonnée X du dernier échantillon est renvoyée. La coordonnée X est une valeur dont l'unité est celle de l'axe horizontal de la forme d'onde (généralement le temps).

Exemple
L'exemple suivant crée une onde sinusoïdale. Les signaux générés commencent toujours au temps 0. Cet exemple présente deux manières de déterminer la coordonnée X du dernier échantillon de la forme d'onde :

```
Signal = @SineWave(10k; 1000; 50)
XEnd1  = @XFirst(Formula.Signal) +
         (@Length(Formula.Signal) - 1)*
         @XDelta(Formula.Signal)
XEnd2  = @XLast(Formula.Signal)
```

Voir aussi
<< @Length >> page 111, << @XDelta >> page 185, << @XFirst >> page 188 et << @XShift >> page 190

4.89 @XShift

Fonction
Décale horizontalement une forme d'onde d'une durée spécifiée.

Syntaxe
@XShift*(Forme d'onde, Décalage)*

Paramètres

Forme d'onde	Forme d'onde à décaler horizontalement.
Décalage	Nombre : valeur du décalage.

Sortie
Forme d'onde d'origine décalée dans le temps.

Description
Cette fonction ne change pas le contenu de la forme d'onde mais modifie les informations de l'échelle horizontale de façon à la décaler horizontalement de la durée (ou autre) définie. Les valeurs positives décalent la forme d'onde vers la droite, les valeurs négatives vers la gauche.

Exemple
L'exemple suivant fait en sorte que l'axe horizontal de la forme d'onde commence toujours à zéro. Le signal est supposé être réel.

```
Signal     = Active.Group1.Recorder_A.Ch_A1
Shift      = -1 * @XFirst(Formula.Signal)
CorrSignal = @XShift(Formula.Signal; Formula.Shift)
```

Voir aussi
<< @XDelta >> page 185, << @XFirst >> page 188 et << @XLast >> page 189

4.90 @XYArray

Fonction
Crée une forme d'onde à partir d'un tableau bidimensionnel.

Syntaxe
@XYArray*(X1; Y1; ...; ...; XN; YN)*

Paramètres

X1	Valeur X du premier échantillon.
Y1	Valeur Y du premier échantillon.
XN	Valeur X du dernier échantillon ; N>= 2.
YN	Valeur Y du dernier échantillon ; N>= 2.

Sortie
Forme d'onde contenant tous les points du tableau.

Description
Cette fonction crée une forme d'onde contenant tous les points définis par les paramètres X1, Y1, X2, Y2, etc. Ces points doivent être saisis dans l'ordre chronologique.

La forme d'onde de sortie contient des points de données équidistants. Cette fonction évalue les valeurs X d'entrée et détermine la fréquence d'échantillonnage la plus adaptée. Pour ce faire, la série 1-2-5 est utilisée.

Prenons par exemple X1 = 0 ms, X2 = 30 ms, X3 = 50 ms. Cela donne un intervalle d'échantillonnage de 20 ms.

Prenons par exemple X1 = 25 ms, X2 = 30 ms, X3 = 50 ms. Cela donne un intervalle d'échantillonnage de 5 ms.

Prenons par exemple X1 = 2 s, X2 = 12 s, X3 = 15 s. Cela donne un intervalle d'échantillonnage de 2 s.

Remarque *Les échantillons de sortie sont équidistants ; leur nombre peut être supérieur au nombre d'échantillons d'entrée.*

Exemple
L'exemple suivant crée une forme d'onde contenant 8 points de données, alors que seuls 5 points ont été saisis. Les trois points supplémentaires sont générés par la fonction XYArray.

```
NewWave = @XYArray(0; 0; 1; 2; 2; 1; 5; -3; 7; 0)
```

Voir aussi
<< @YArray >> page 193

4.91 @YArray

Fonction
Cette fonction crée une forme d'onde à partir d'un **tableau** unidimensionnel.

Syntaxe
@YArray*(F échantillonnage; Y1; ...; YN)*

Paramètres

F échantillonnage	Nombre : fréquence d'échantillonnage
Y1	Nombre : amplitude (valeur Y) du premier point de données.
YN	Nombre : amplitude (valeur Y) du dernier point de données. N >=2

Sortie
La sortie est une forme d'onde contenant tous les points de données du tableau.

Description
Cette fonction crée une forme d'onde contenant tous les points définis par les paramètres Y1 à YN. Le premier paramètre définit la fréquence d'échantillonnage (= 1 / période d'échantillonnage).

Exemple
L'exemple suivant crée une forme d'onde contenant 9 points de données avec une fréquence d'échantillonnage de 100 Hz.

```
Signal = @YArray(100; 0; 1; 2; 2; 1; -1; -3; 2; 0)
```

Voir aussi
<< @XYArray >> page 191

5 Mesure et analyse des impulsions

5.1 Généralités

Termes et définitions IEEE relatifs aux impulsions
Le contenu de cette section est basé sur les normes IEEE 194-1977 et 181-1977.

Pour en savoir plus, visiter le site Web de l'**IEEE Standards Association** : standards.ieee.org/

Onde, impulsion et transition

- **Onde** : modification de l'état physique d'un milieu se propageant dans ce dernier à la suite d'une ou de plusieurs perturbations.
- **Impulsion** : onde partant d'un premier état nominal pour en atteindre un second avant de revenir au premier. Dans le reste du présent document, le terme impulsion est compris dans le terme onde.
- **Transition** : partie d'une onde ou d'une impulsion avant un premier état nominal. Dans le reste du présent document, le terme transition est compris dans les termes impulsion et onde.

Le schéma suivant présente une vue d'ensemble de la forme d'onde d'une impulsion.

L'impulsion

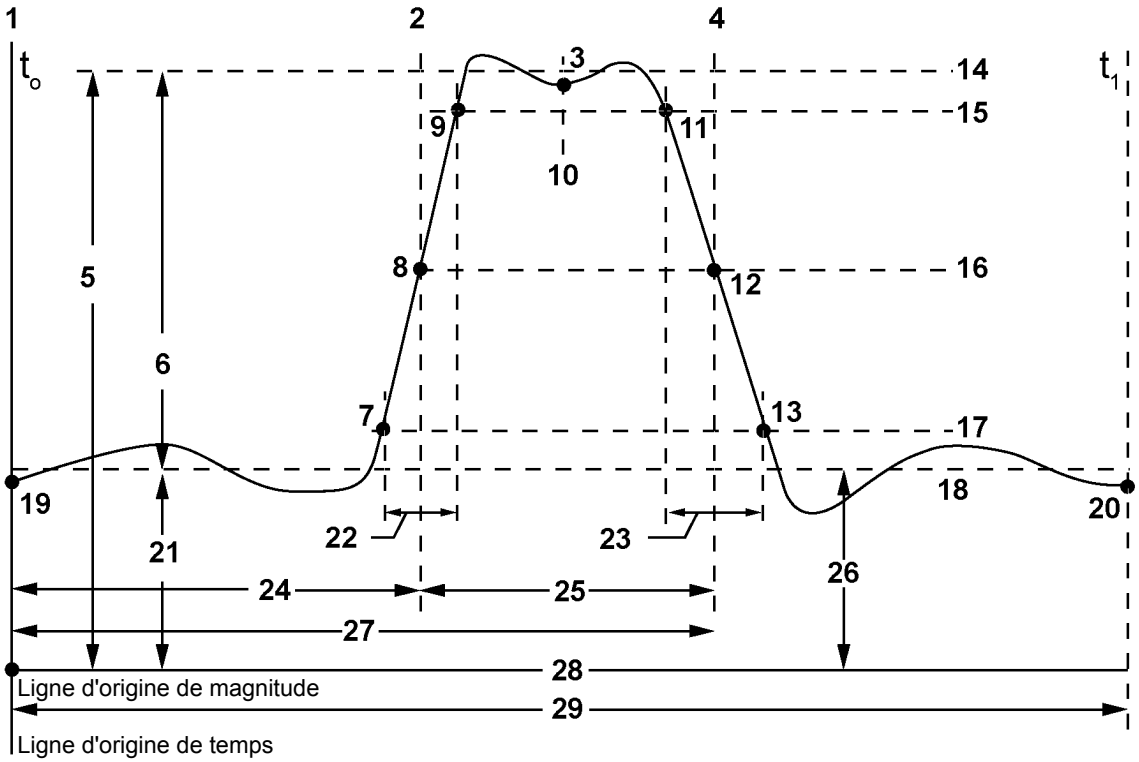


Figure 5.1 : Schéma de la forme d'onde d'une impulsion

1	Ligne d'origine de temps	Ligne dont le temps est constant et égal à zéro (sauf définition contraire) et qui passe par le premier temps de référence t_0 de l'époque d'une forme d'onde.
2	Ligne de début d'impulsion	Ligne de début d'impulsion. Ligne de référence temporelle au début de l'impulsion.
3	Centre du sommet	Point défini au sommet de la forme d'onde d'une impulsion par rapport à un temps donné ou à la magnitude. Si aucun point n'est défini, le centre du sommet est référencé par rapport au temps à l'intersection de la forme d'onde d'une impulsion et de la ligne du centre du sommet.
4	Ligne de fin d'impulsion	Ligne de fin d'impulsion. Ligne de référence temporelle à la fin de l'impulsion.
5	Magnitude du sommet	La magnitude du sommet est obtenue par le biais d'une procédure ou d'un algorithme spécifique.

6	Amplitude d'impulsion	Différence algébrique entre la magnitude du sommet et celle de la base.
7	Proximal (premier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne proximale.
8	Mésial (premier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne mésiale.
9	Distal (premier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne distale.
10	Ligne du centre du sommet	Ligne de référence temporelle à la moyenne des temps de début et de fin de l'impulsion.
11	Distal (dernier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne distale.
12	Mésial (dernier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne mésiale.
13	Proximal (dernier point de transition)	Point référencé par rapport à la magnitude à l'intersection d'une forme d'onde et d'une ligne proximale.
14	Ligne du sommet	Ligne de référence de magnitude à la magnitude du sommet.
15	Ligne distale	Ligne de référence de magnitude à une magnitude donnée dans la région distale de la forme d'onde d'une impulsion. Sauf définition contraire, la ligne distale se trouve à la magnitude de référence 90 %.
16	Ligne mésiale	Ligne de référence de magnitude à une magnitude donnée dans la région mésiale de la forme d'onde d'une impulsion. Sauf définition contraire, la ligne mésiale se trouve à la magnitude de référence 50 %.
17	Ligne proximale	Ligne de référence de magnitude à une magnitude donnée dans la région proximale de la forme d'onde d'une impulsion. Sauf définition contraire, la ligne proximale se trouve à la magnitude de référence 10 %.
18	Ligne de base	Ligne de référence de magnitude à la magnitude de la base.
19	Premier point de la base	Sauf définition contraire, premier point de référence de l'époque d'une impulsion.
20	Dernier point de la base	Sauf définition contraire, dernier point de référence de l'époque d'une impulsion.

21	Magnitude de la base	La magnitude de la base est obtenue par le biais d'une procédure ou d'un algorithme spécifique. Sauf définition contraire, les deux portions de la base sont incluses dans la procédure ou l'algorithme.
22	Durée du premier transit	Durée de la première transition de la forme d'onde d'une impulsion.
23	Durée du dernier transit	Durée de la dernière transition de la forme d'onde d'une impulsion.
24	Début d'impulsion	Instant défini par un point référencé par rapport à la magnitude sur la première transition de la forme d'onde d'une impulsion. Sauf définition contraire, le début de l'impulsion correspond au point mésial sur la première transition.
25	Durée d'impulsion	Durée entre le début et la fin de l'impulsion.
26	Décalage	Différence algébrique entre deux lignes de référence de magnitude définies. Sauf définition contraire, les deux lignes de référence de magnitude sont la ligne de base de la forme d'onde et la ligne d'origine de magnitude.
27	Fin d'impulsion	Instant défini par un point référencé par rapport à la magnitude sur la dernière transition de la forme d'onde d'une impulsion. Sauf définition contraire, la fin de l'impulsion correspond au point mésial sur la dernière transition.
28	Ligne d'origine de magnitude	Ligne dont la magnitude est égale à zéro (sauf définition contraire) et qui s'applique à toute l'époque de la forme d'onde.
29	Époque de forme d'onde d'impulsion	Intervalle de temps pour lequel les données de la forme d'onde sont connues ou peuvent l'être. Une époque de forme d'onde définie par des équations peut s'étendre de - l'infini à + l'infini ou, comme toutes les données de forme d'onde, d'un premier temps de référence t_0 à un second temps de référence t_1.

6 Filtres IIR

6.1 Introduction

La base de données de formules Perception contient un série de filtres IIR. Ces filtres activent une classe qui filtre la séquence d'entrée à l'aide du **filtre IIR de forme directe** défini par les coefficients de contre-réaction et de réaction positive. IIR est l'abréviation de **I**nfinite **I**mpulse **R**esponse (Réponse impulsionnelle infinie). Le rapport entre le signal de sortie et le signal d'entrée est du type de la formule suivante Figure 6.1.

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

Figure 6.1 : Infinite **I**mpulse **R**esponse (réponse impulsionnelle infinie) – sortie, entrée

où :

N = ordre de filtrage

b_i = coefficients du filtre de réaction positive, également appelés zéros

a_i = coefficients du filtre de contre-réaction, également appelés pôles

x(n) = signal d'entrée

y(n) = signal de sortie

Le schéma de principe ressemble à la Figure 6.2 :

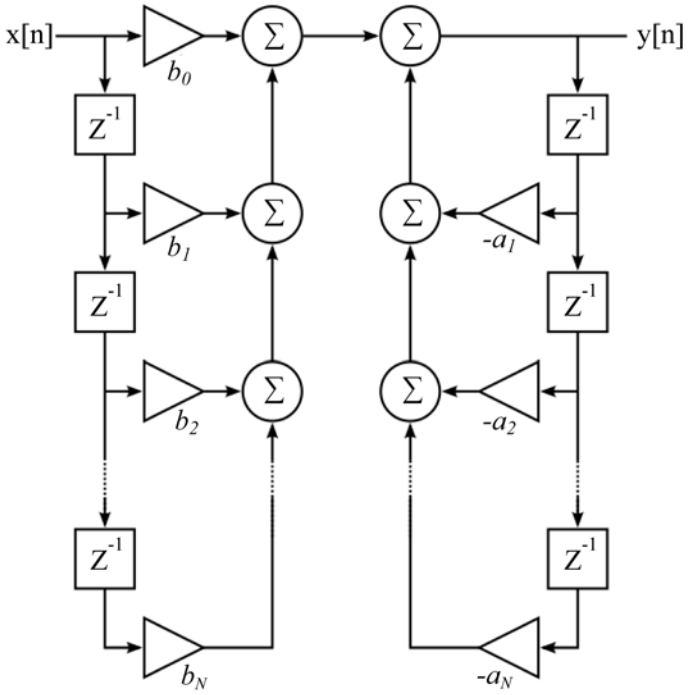


Figure 6.2 : Réponse impulsionnelle infinie – Schéma de principe

Toutes les formules de filtre commencent par **« @Filter »**. La liste complète des fonctions de filtrage est donnée ci-après.

@FilterButterworthLP
@FilterButterworthHP
@FilterButterworthBP
@FilterButterworthBS
@FilterBesselLP
@FilterBesselHP
@FilterBesselBP
@FilterBesselBS
@FilterChebyshevLP
@FilterChebyshevHP
@FilterChebyshevBP
@FilterChebyshevBS

Trois différents types de filtres sont disponibles :

- **Bessel**
- **Butterworth**
- **Chebyshev**

Pour chaque type, il existe une activation de filtre **Passe-bas** (LP), **Passe-haut** (HP), **Passe-bande** (BP) et **Bloqueur de bande** (BS).

6.1.1 Bessel

Un filtre Bessel est un type de filtre linéaire avec un temps de propagation de groupe le plus plat possible (réponse en phase la plus linéaire possible). Le filtre Bessel dispose d'une bonne réponse indicielle et de très peu de dépassements ; toutefois, la réponse en fréquence est moins bonne que celle d'un filtre Butterworth. Les filtres Bessel ont une réponse en phase quasi linéaire. Les applications du filtre Bessel sont les cas dans lesquels les relations de phase sont critiques.

Il est préférable d'utiliser ce filtre dans le domaine temporel.

Il est également recommandé d'utiliser un filtre Bessel lorsque le signal à filtrer contient des signaux non sinusoïdaux, comme des formes d'onde carrées ou du bruit par exemple, et que vous souhaitez utiliser le signal filtré à des fins de temporisation. Le filtre Bessel est alors le plus adapté car les divers éléments de fréquence du signal d'origine auront presque le même retard après avoir été filtrés.

Avantages :
Meilleure réponse indicielle, très peu de dépassement ou d'oscillation transitoire.

Inconvénients :
Vitesse d'atténuation initiale plus lente au-delà de la bande passante que le filtre Butterworth.

6.1.2 Butterworth

Le filtre Butterworth est un type de filtre de traitement de signal conçu pour avoir une réponse en fréquence la plus plate possible dans la bande passante, de sorte qu'on l'appelle également filtre de magnitude la plus plate possible. Il possède une bonne réponse en fréquence, sans ondulations ; toutefois, la réponse en phase peut être fortement non linéaire, particulièrement pour les filtres d'ordre élevé.

Il est préférable d'utiliser ce filtre dans le domaine fréquentiel.

Si le signal à filtrer est un signal sinusoïdal, Butterworth est souvent le plus adapté ; la réponse en phase non linéaire importe peu car, étant donné qu'une seule fréquence dominante du signal d'entrée est traitée, la déformation du signal de sortie sera minime.

Avantages :
- Réponse de magnitude la plus plate possible dans la bande passante.
- Bonnes performances générales.
- Meilleure réponse impulsionnelle que le filtre Chebyshev.
- Vitesse d'atténuation plus rapide que le filtre Bessel.

Inconvénients :
- dépassements et oscillations dans la réponse indicielle.

6.1.3 Chebyshev (Type I)

Les filtres Chebyshev ont un affaiblissement plus important et plus d'ondulations de bande passante (type I) que les filtres Butterworth. Les filtres Chebyshev ont la propriété de minimiser l'erreur entre la caractéristique du filtre idéalisé et du filtre réel par rapport à la plage du filtre, mais avec des ondulations de la bande passante.

Par rapport à un filtre Butterworth, un filtre Chebyshev peut obtenir une transition plus précise entre la bande passante et la bande de coupure avec un filtre d'ordre inférieur.

Il est préférable d'utiliser ce filtre dans le domaine fréquentiel.

Avantages :
- Vitesse d'atténuation plus rapide au-delà de la bande passante que le filtre Butterworth.

Inconvénients :
- Ondulations de la bande passante.
- Oscillations considérablement plus présentes dans la réponse indicielle que le filtre Butterworth.

6.1.4 Spectre de magnitude

La Figure 6.3 suivante montre les spectres de magnitude des trois différents filtres IIR passe-bas du 2ème ordre. Le filtre Bessel est celui qui a la moins bonne réponse en fréquence sur la bande de coupure. Le filtre Chebyshev a le plus fort affaiblissement, mais comporte des ondulations de la bande passante.

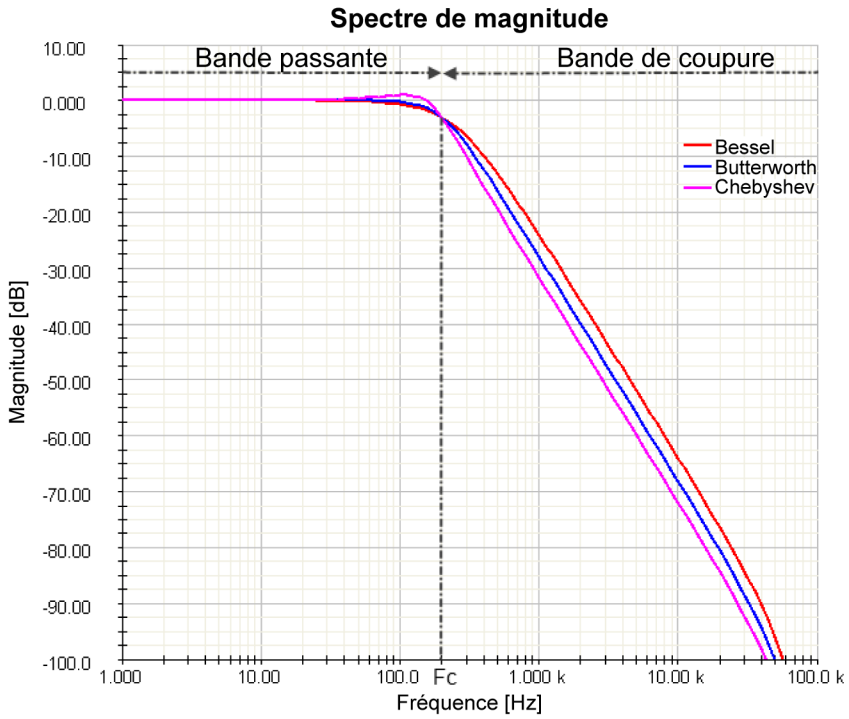


Figure 6.3 : Spectre de magnitude des filtres passe-bas

Figure 6.4 montre la bande passante des trois différents filtres.

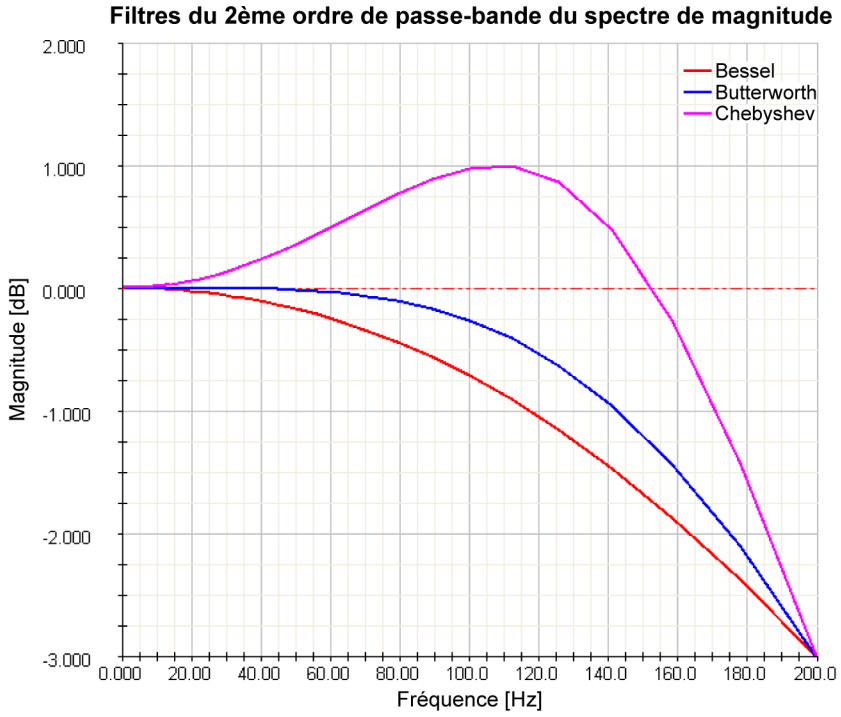


Figure 6.4 : Bande passante de spectre de magnitude des filtres

Figure 6.5 montre la bande de coupure des trois filtres.

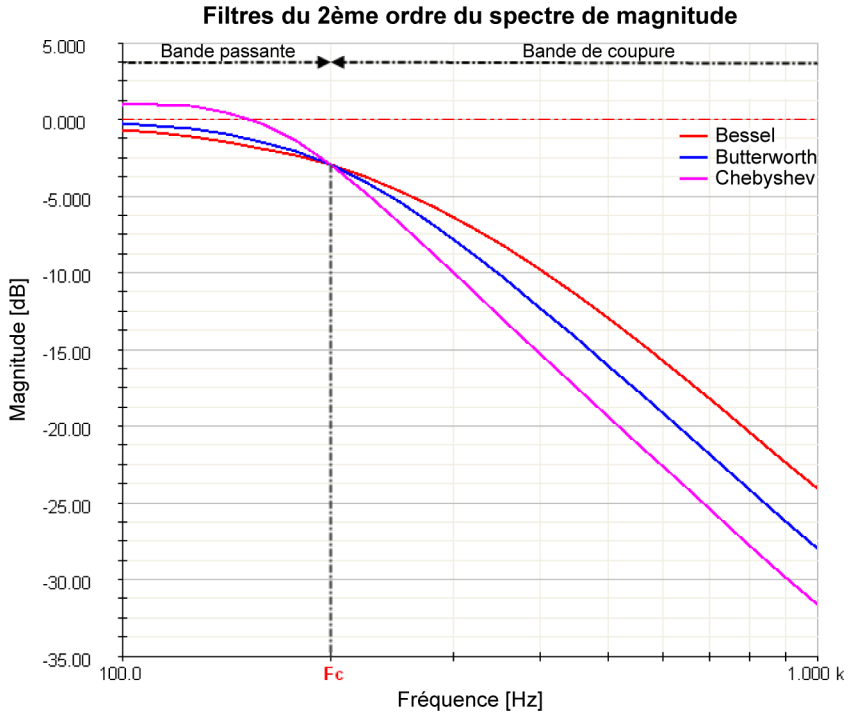


Figure 6.5 : Spectre de magnitude – bande de coupure

6.1.5 Réponse impulsionnelle

La Figure 6.6 suivante montre la réponse impulsionnelle des trois différents filtres IIR passe-bas du 2ème ordre. Le filtre Bessel offre la meilleure réponse impulsionnelle.

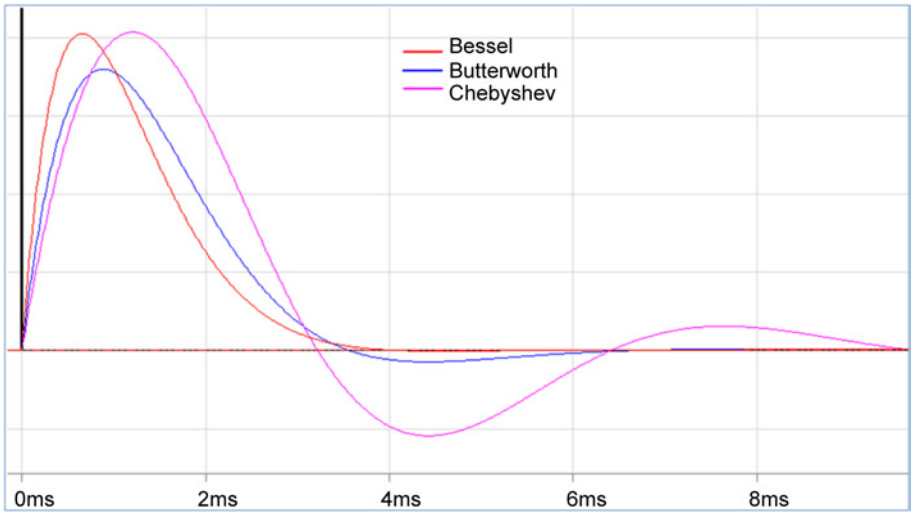


Figure 6.6 : Réponse impulsionnelle des filtres passe-bas

6.1.6 Réponse indicielle

La Figure 6.7 ci-après montre la réponse indicielle des trois différents filtres IIR passe-bas du 2ème ordre.

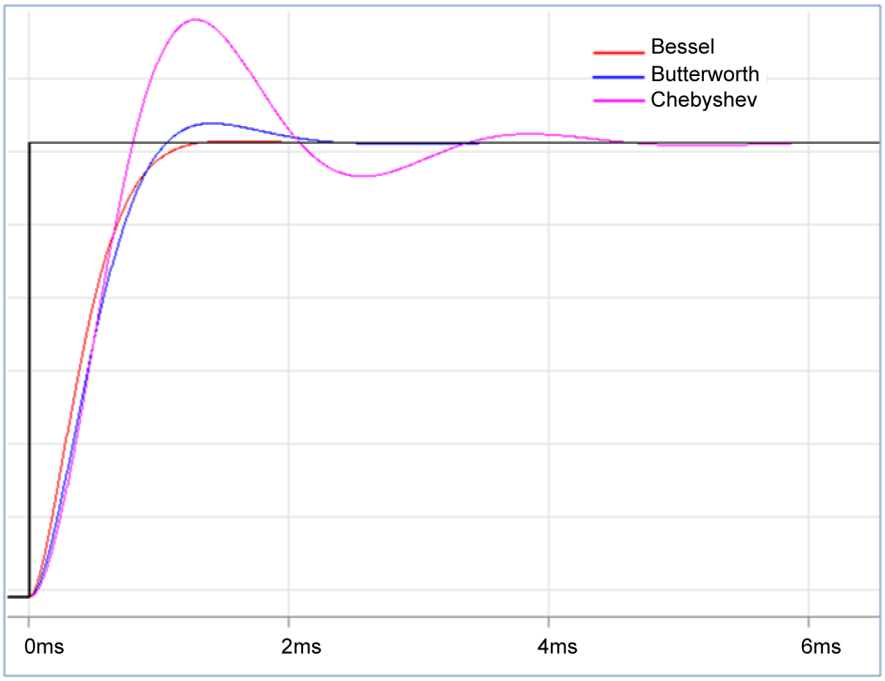


Figure 6.7 : Réponse indicielle des filtres passe-bas

Le filtre Bessel est une fois de plus celui qui apporte la meilleure réponse indicielle. Les réponses impulsionnelle et indicielle ci-dessus peuvent être reconstruites en utilisant les formules suivantes :

Num	Name	Formula	Units
1	Pulse	@Pulse(80k; 80k; 20k)	
2	ButherworthPR	@FilterButterworthLP(Formula.Pulse; 2; 200)	
3	BesselPR	@FilterBesselLP(Formula.Pulse; 2; 200)	
4	ChebyshevPR	@FilterChebyshevLP(Formula.Pulse; 2; 200; 3)	
5			
6	Step	@Pulse(80k; 80k; 20k; 10k)	
7	ButherworthSR	@FilterButterworthLP(Formula.Step; 2; 200)	
8	BesselSR	@FilterBesselLP(Formula.Step; 2; 200)	
9	ChebyshevSR	@FilterChebyshevLP(Formula.Step; 2; 200; 3)	
10			

Figure 6.8 : Réponse filtre

Les formules ci-dessus peuvent également être utilisées pour examiner les réponses indicielle et impulsionnelle d'un filtre quelconque que vous pouvez créer avec les formules de filtre actuelles de Perception.

6.1.7 Filtrage sans phase

Le filtrage sans phase est réalisé en utilisant la technique de l'ordre inversé. Cette technique filtre le signal deux fois. L'ordre du filtre utilisé est la moitié de la valeur d'ordre saisie *(Ordre/2)*. Au cours de la première étape, le signal est filtré de manière habituelle. Au cours de la deuxième étape, le résultat filtré est de nouveau filtré avec le même filtre mais avec les points de données entrant dans le filtre dans le sens inverse. Étant donné que le filtre est utilisé deux fois, l'ordre général du filtre est égal à la valeur d'ordre saisie *(Ordre)*. Si l'ordre saisi est une valeur *impaire*, l'ordre du filtre sera la première valeur paire inférieure à cette valeur impaire. Par exemple, si la valeur d'ordre saisie est 7, le filtre aura la valeur d'ordre 6. Le changement de phase du signal filtré est nul ; voir Figure 6.9.

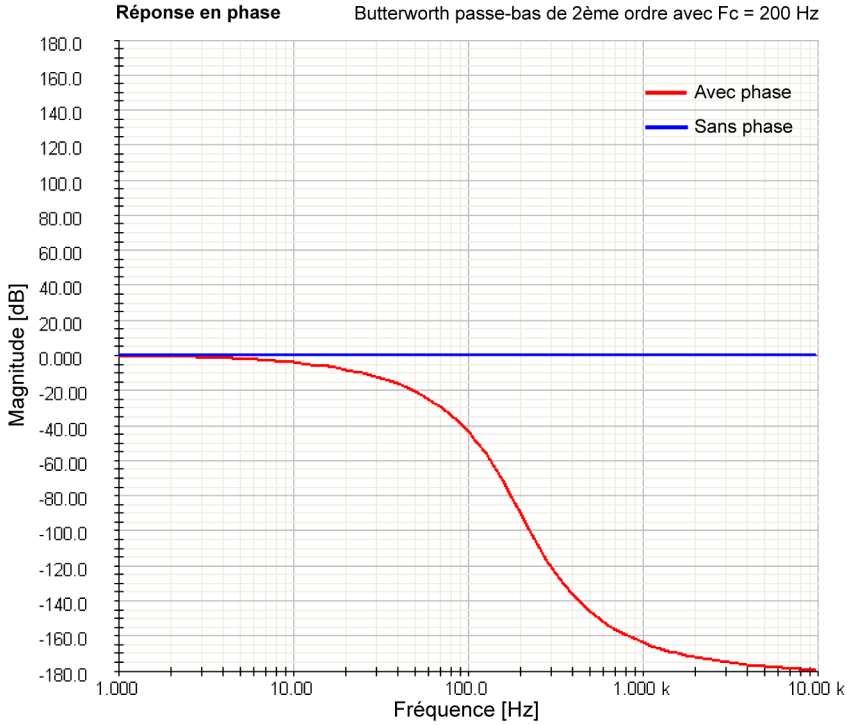


Figure 6.9 : Réponse en phase des filtres

Il existe une différence entre les spectres de magnitude des filtres *sans phase* et des *filtres avec phase*. Cette différence est due au fait que le filtre *sans phase* est le résultat d'un filtrage effectué deux fois avec un filtre à la moitié de l'ordre, l'atténuation à la fréquence de coupure n'étant alors pas de -3 dB mais de -6 dB ; voir Figure 6.10.

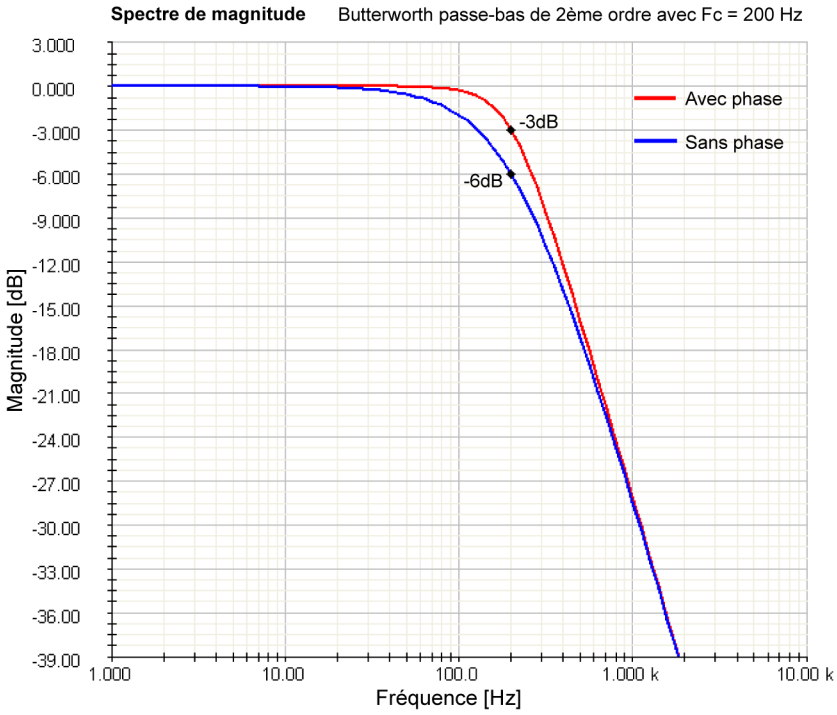


Figure 6.10 : Spectre de magnitude - Atténuation

6.1.8 Importance de la fréquence d'échantillonnage et de la fréquence de coupure

Il est impératif de connaître l'influence de la fréquence d'échantillonnage du signal discret à filtrer ainsi que la fréquence de coupure. La fréquence d'échantillonnage doit être au moins le double de la fréquence de coupure, ce qui est également connu sous le nom de fréquence de Nyquist.

La courbe de magnitude du filtre est également influencée par la fréquence d'échantillonnage. Un filtre du deuxième ordre possède un affaiblissement de 12 dB/octave ou de 40 dB/décade, même si pour les fréquences proches de la moitié de la fréquence d'échantillonnage, le filtre se comporte différemment ; voir Figure 6.11 ci-dessous. Vous devez connaître cette différence si vous voulez comparer le filtrage numérique et le filtrage analogique. Les filtres analogiques ont un affaiblissement ayant une pente constante sur la totalité de la bande de coupure.

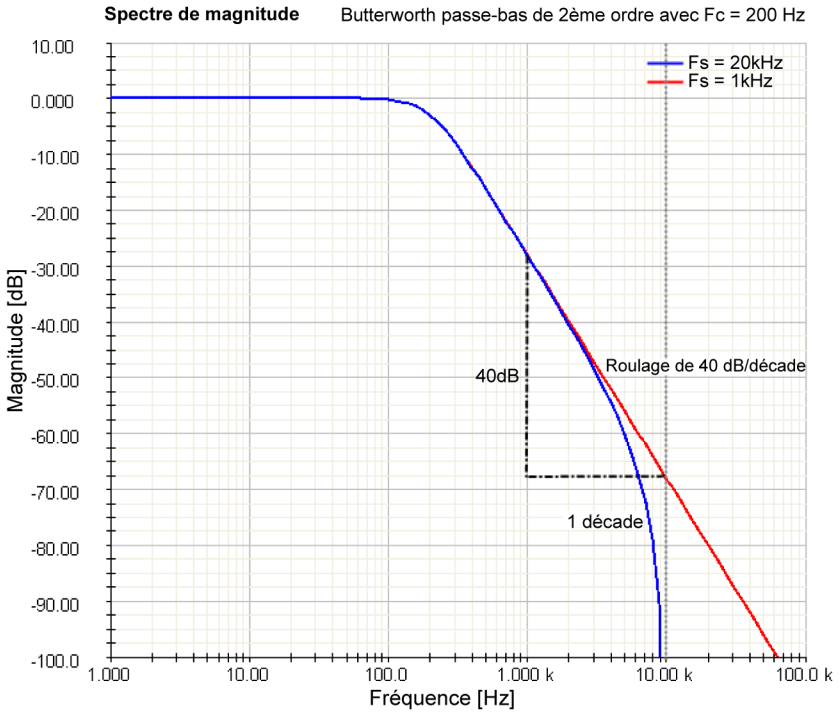


Figure 6.11 : Fréquences à la moitié de la fréquence d'échantillonnage

Index alphabétique

X

Y

Head Office
HBM
Im Tiefen See 45
64293 Darmstadt
Germany
Tel: +49 6151 8030
Email: info@hbm.com

France
HBM France SAS
46 rue du Champoreux
BP76
91542 Mennecy Cedex
Tél:+33 (0)1 69 90 63 70
Fax: +33 (0) 1 69 90 63 80
Email: info@fr.hbm.com

UK
HBM United Kingdom
1 Churchill Court, 58 Station Road
North Harrow, Middlesex, HA2 7SA
Tel: +44 (0) 208 515 6100
Email: info@uk.hbm.com

USA
HBM, Inc.
19 Bartlett Street
Marlborough, MA 01752, USA
Tel : +1 (800) 578-4260
Email: info@usa.hbm.com

PR China
HBM Sales Office
Room 2912, Jing Guang Centre
Beijing, China 100020
Tel: +86 10 6597 4006
Email: hbmchina@hbm.com.cn

measure and predict with confidence

I3034-2.0 fr