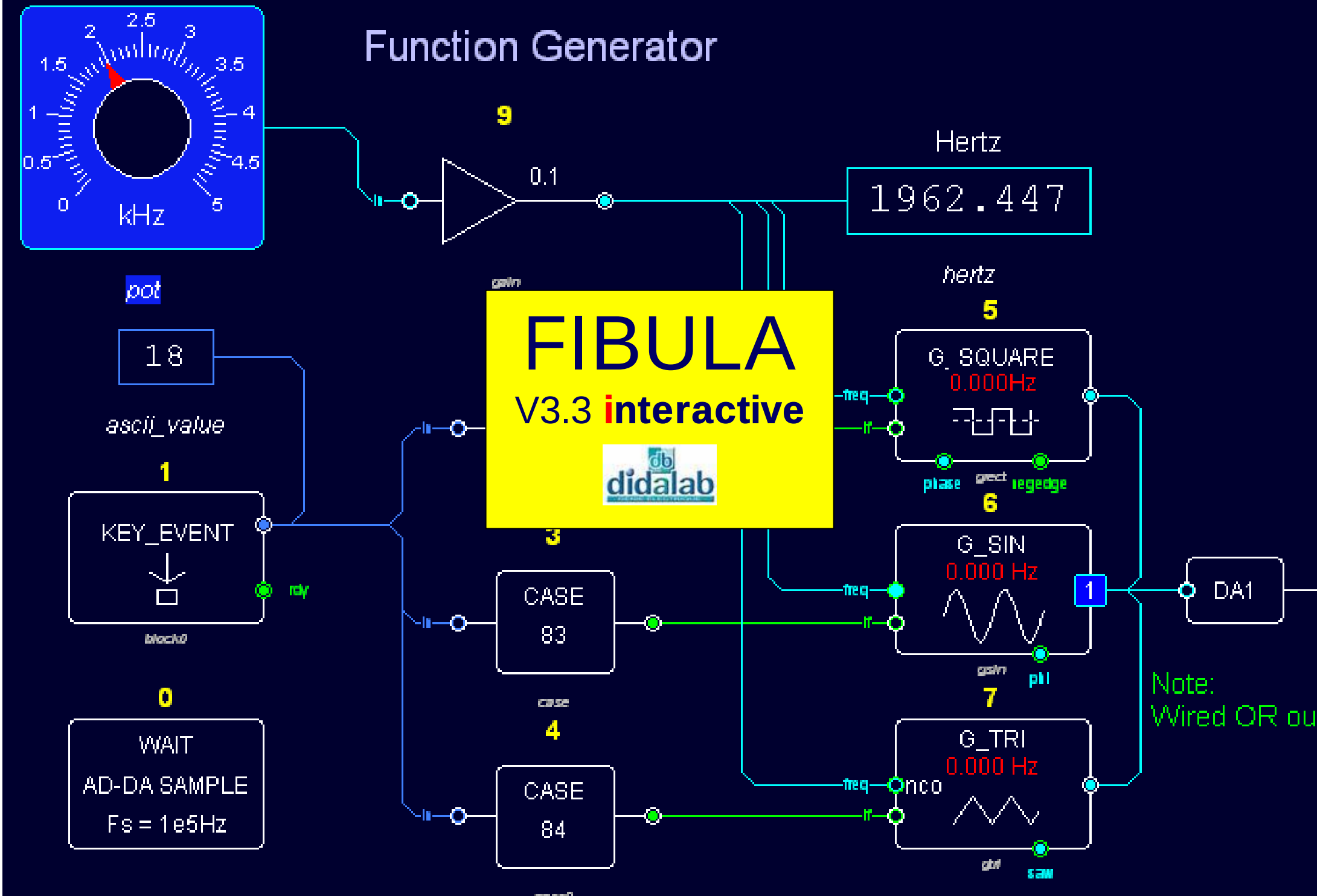

Function Generator
2.5
2
3
1.5
3.5
1
4
0.5
4.5
0
5
kHz
9
0.1
Hertz
1962.447
pot
18
ascii_value
1
KEY_EVENT
rdy
block0
0
WAIT
AD-DA SAMPLE
Fs = 1e5Hz
FIBULA
V3.3 interactive
didalab
3
CASE
83
case
4
CASE
84
hertz
5
G_SQUARE
0.000Hz
freq
trr
phase
duct
negedge
6
G_SIN
0.000 Hz
freq
trr
gain
phi
1
7
G_TRI
0.000 Hz
freq
nco
trr
ctrl
saw
DA1
Note:
Wired OR ou

FIBULA

V.3.3 **interactive**

Prise en mains

Jean-Marie Ory

Rev. V3.3 Janvier 2014

Table des Matières

NOTE IMPORTANTE V3.3

L'organisation des projets a été simplifiée, tout en restant compatible avec les versions précédentes. L'utilisateur peut se construire une librairie globale accessible depuis n'importe quel projet. (Chap 9)
Le catalogue a 3 modes: Référence, Utilisateur, ou Fusion des deux

CHAÎNES DE CARACTERES

A partir de la version 3.1, la gestion des chaînes de caractères a été modifiée par rapport aux versions précédentes.

Une chaîne est dorénavant une variable définie par un pointeur.
Cela permet de créer des tableaux de chaînes indexables,
-- voyez la démo **strsel**.

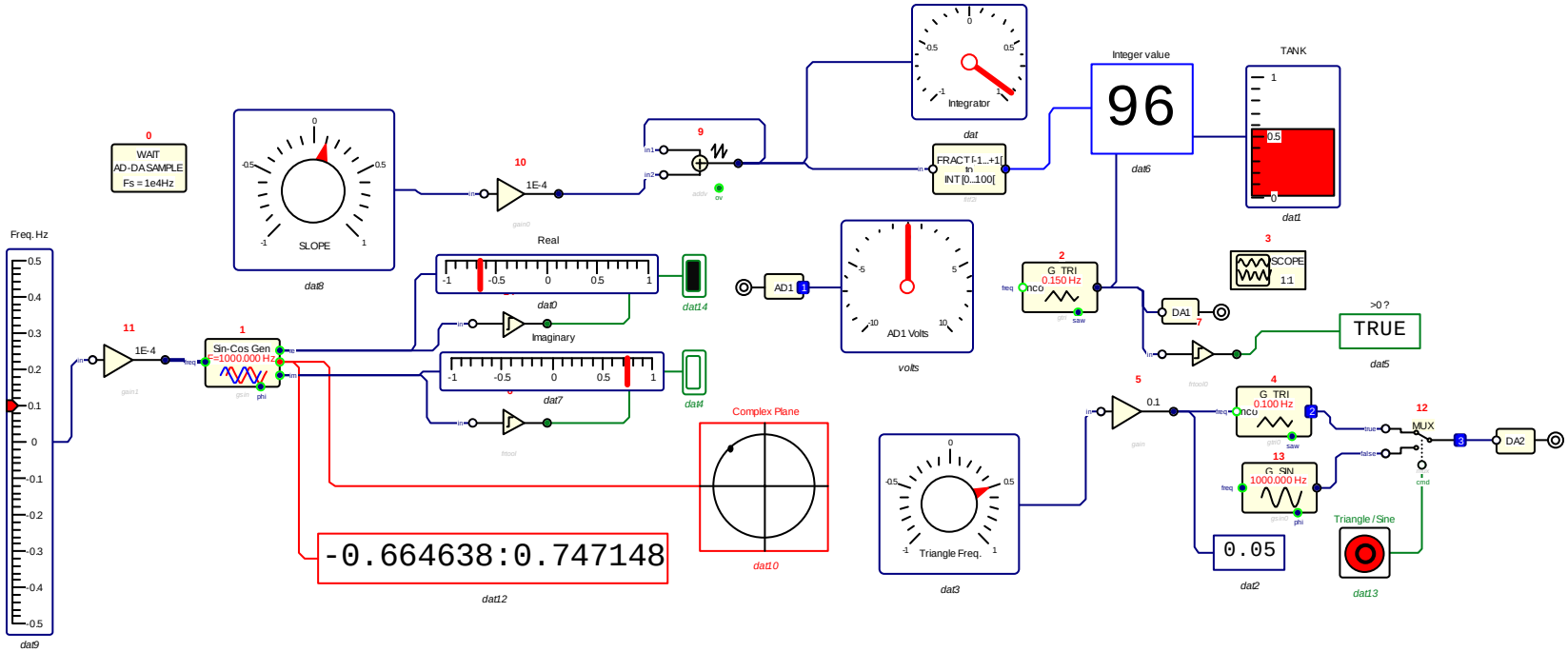
Les anciens blocs **frtohex**, **frtostr**, **inttostr**, **strnh** ne sont plus compatibles et sont remplacés par le bloc unique **makestring** (syntaxe semblable à **printf**).

Pour plus de détails, reportez-vous à 4.3.12, page 30

L'interactivité

La version interactive de FIBULA vous permet en temps réel

- De modifier n'importe quelle donnée en entrée à l'aide potentiomètres boutons et curseurs
- D'afficher la valeur instantanée de n'importe quelle variable sous forme numérique, sous forme de voltmètres, de thermomètres ou de réservoirs
- D'interagir avec l'application par des commandes au clavier ou à la souris



Avant propos

Qu’est ce que **FIBULA ?**

FIBULA signifie **F**unctional **I**nterconnected **B**locks **U**ser **LA**nguage

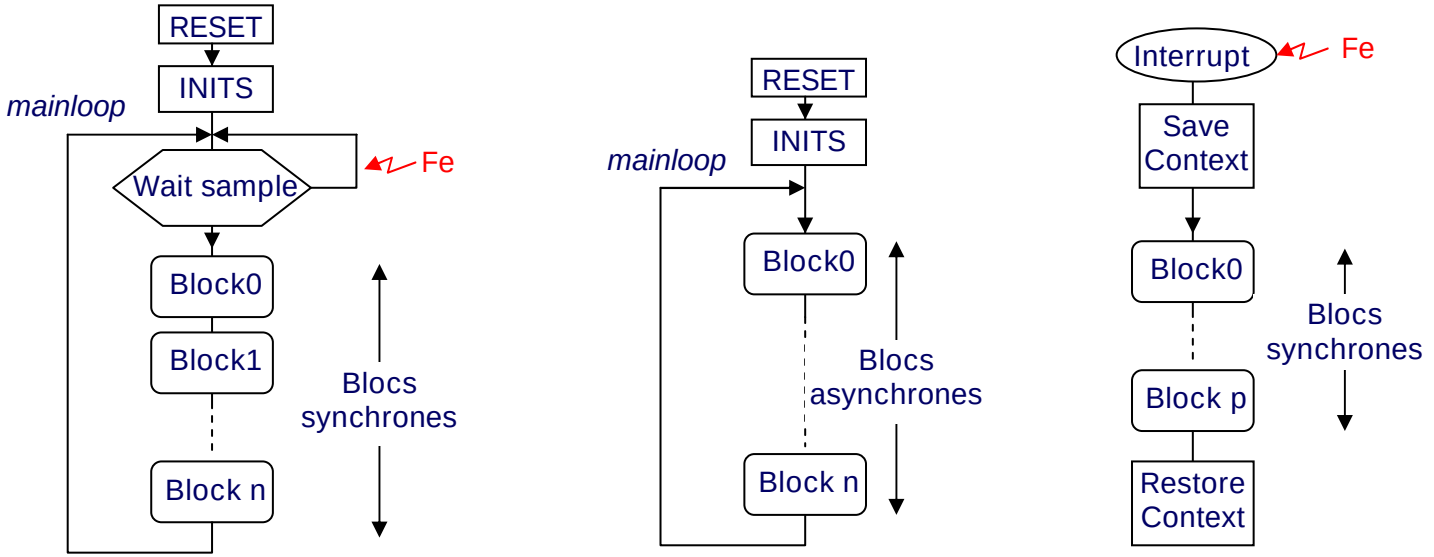
FIBULA n’est ni MATLAB SIMULINK™, ni LABVIEW™, ni encore SCILAB.
Sans doute utilisez-vous depuis longtemps ces logiciels, et ils vous sont devenus tellement familiers que vous vous attendez à ce que FIBULA se comporte de manière identique. Mais FIBULA n’est ni un outil de calcul matriciel, ni un outil de simulation, ni un générateur d’instrumentation virtuelle.

FIBULA est un environnement de développement dédié aux processeurs de signaux (Digital Signal Processors ou DSP).
A ce titre, FIBULA serait plutôt comparable à un environnement de développement intégré tel que Code Warrior ou Eclipse.
Sauf que FIBULA est graphique et interactif, et que sa prise en mains est la plus simple et rapide qui soit.

La cible de la présente version de FIBULA est le processeur Freescale 24/48 bits **DSP56720** à double cœur 200MIPs.
Elle est principalement destinée aux applications audio haut de gamme. Elle se compose de 2 cœurs DSP56300, d'une mémoire partagée de 64k mots 24 bits, d'un bus externe et de périphériques audio partagés.
Chacun des deux cœurs DSP56300 possède 3 champs mémoire à accès parallèle (92k mots de 24 bits en tout) , 8 canaux DMA, des périphériques audio, des ports SPI et JTAG, et 3 timers. La pile câblée ne comporte que 15 mots de 48 bits.

L'architecture quelque peu "non conventionnelle" de cette machine fait que les compilateurs C produisent du code assez peu efficace, car ne mettant pas en œuvre les ressources matérielles de la machine. Quelques programmes de tests nous ont démontré que le code d'un compilateur C est plus de 10 fois plus lent que du code assembleur écrit à la main.
Nous avons créé FIBULA pour programmer de manière ordonnée et efficace, à l'aide d'une librairie de blocs écrits en assembleur.

FIBULA permet principalement de créer des applications Temps Réel Isochrones et à Temps Déterministe, c'est-à-dire dans les quelles l'algorithme et les entrées/sorties sont cadencés par une fréquence rigoureusement constante appelée fréquence d'échantillonnage. Le temps d'exécution de chaque bloc y est prévisible.
La présente version de FIBULA propose deux architectures très simples: a) Boucle cadencée par l'échantillonnage b) Boucle asynchrone, et interruption à la fréquence d'échantillonnage (Fig.1).
En travaillant à Fe=100kHz, avec un cœur à 200MIPs, on peut exécuter 2000 cycles, soit 2000 instructions arithmétiques par échantillon pour chacun des cœurs. En prenant une moyenne de 40 cycles pour chaque bloc, cela nous donne 50 comme ordre de grandeur du nombre maximal de blocs de chaque cœur à cette fréquence d'échantillonnage.



a) Simple boucle rythmée par l'attente de l'instant d'échantillonnage b) Boucle asynchrone et interruption à la fréquence d'échantillonnage.

Fig.1 Les deux architectures standard d'un programme FIBULA

Dans FIBULA, vous dessinez un schéma à l'aide de blocs du catalogue. Vous les interconnectez et vous les paramétrez. L'ordre d'exécution des blocs peut être choisi indépendamment de l'ordre du flux des données qui résulte des connexions: cela permet d'élégantes astuces de programmation.

Les blocs ont une structure hiérarchique. Un bloc peut être décrit à l'aide d'un schéma incluant des sous blocs. et ainsi de suite jusqu'au niveau atomique pour lequel il n'y a plus de décomposition possible. Seuls les blocs atomiques sont écrits en assembleur. Comme (en théorie) ils sont simples et peu nombreux, la tâche consistant à les écrire est assez aisée.

Les blocs de traitement du signal utilisent majoritairement l'**arithmétique fractionnaire** signée réelle 24 ou 48 bits, ou complexe 2x24 bits. Toutes les grandeurs sont comprises dans l'intervalle [-1 .. +1[. Son intérêt vient de ce que les nombres ne peuvent pas déborder par multiplication, mais seulement par addition. Lorsque l'on veut simuler une grandeur physique **x**, on la représente en machine par le nombre fractionnaire $\mathbf{x/|x|_{max}}$. Avec 24 ou 48 bits, la précision des calculs est excellente, et permet une grande dynamique, dans les amplitudes ainsi que dans les fréquences.

Cette version 3.3 est interactive.
Elle est particulièrement efficace pour créer des modèles temps réel, des simulations, des instruments virtuels; elle est aussi très didactique.
Il est à noter que ces objets animés ne pénalisent quasiment pas l'efficacité du code DSP.
En effet, les trames de données sont transmises à l'aide d'un canal DMA à fonctionnement parallèle à l'activité des deux cœurs.

Si vous avez une suggestion, ou si vous rencontrez un quelconque problème avec FIBULA, merci de m'envoyer un courriel.
FIBULA est un produit vivant qui évolue au gré des remarques que vous nous adressez.
Si vous observez un bogue, essayez de déterminer comment faire réapparaître ce bogue et envoyez-moi également un courriel, je tenterai de le corriger dans les meilleurs délais !

Je vous souhaite de passionnantes expérimentations avec le matériel DIDALAB et le logiciel FIBULA !

Jean-Marie Ory

Courriel: **jean-marie.ory@univ-lorraine.fr**

1 Introduction

FIBULA permet de construire des programmes Temps Réel pour la famille de processeurs de signaux Freescale DSP563xx.
FIBULA tourne sous WINDOWS.
La présente version est associée à la plateforme DIDALAB ETD4100xx et ne fonctionne que avec et pour cette plateforme.

FIBULA est un environnement de développement rapide fondé sur l'interconnexion de blocs fonctionnels.
La prise en mains et l'utilisation sont extrêmement simples.

Le processus de développement d'une application se résume en 3 étapes:

1 **Saisie du schéma** (Choix de blocs du catalogue, interconnexions de ces blocs, paramétrage de ces blocs).
2 **Compilation du schéma** et chargement du code dans la maquette puis lancement du programme.
3 **Test du programme**. Si nécessaire, modifier le schéma ou les paramètres des blocs et retourner à l'étape 2.

L'étape 1 se ramène à quelques clics de souris, et à la saisie des paramètres.

L'étape 2 consiste à cliquer sur ce bouton: 

L'étape 3 se décompose en 3 niveaux hiérarchiques:
A Observation des signaux grâce aux instruments virtuels (voltmètre, oscilloscopes, analyseur de spectre, histogramme)
B Exécution des blocs fonctionnels en mode pas à pas
C Exécution instruction par instruction (Emulation JTAG ou simulateur Freescale)

En général, la simple observation des signaux sur l'oscilloscope virtuel est suffisante pour déboguer un programme. Le niveaux B et C ne sont que rarement nécessaires.

2 Installation

2.1 Installation à partir du cédérom FIBULA V3 interactive

Connectez le boîtier **Didalab ETD410010** sur une prise USB de votre PC, et mettez-le sous tension.
Si vous obtenez le message « Nouveau matériel détecté », installez le pilote fourni sur le cédérom.

Lancez le programme d'installation **Fibula_i_Install.exe** ci-contre
et suivez les directives.

Fibula_i_Install.exe

Cochez la case d'acceptation de la Licence;
Choisissez un répertoire de destination (habituellement: **C:\Program Files**)

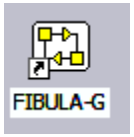
Le programme d'installation copie tous les fichiers, crée un raccourci sur le bureau.

L'option **Fichiers de référence en lecture seule** est recommandée
Annulez l'option **Inclure les programmes d'exemples** dans le cas d'un poste pour étudiant, car de nombreux TP y trouvent une réponse
A la question **Désirez-vous ouvrir Fibula lorsque vous cliquez sur un fichier *.fib ?** : répondez **OUI** (sauf si l'extension *.fib est déjà utilisée par un autre programme)

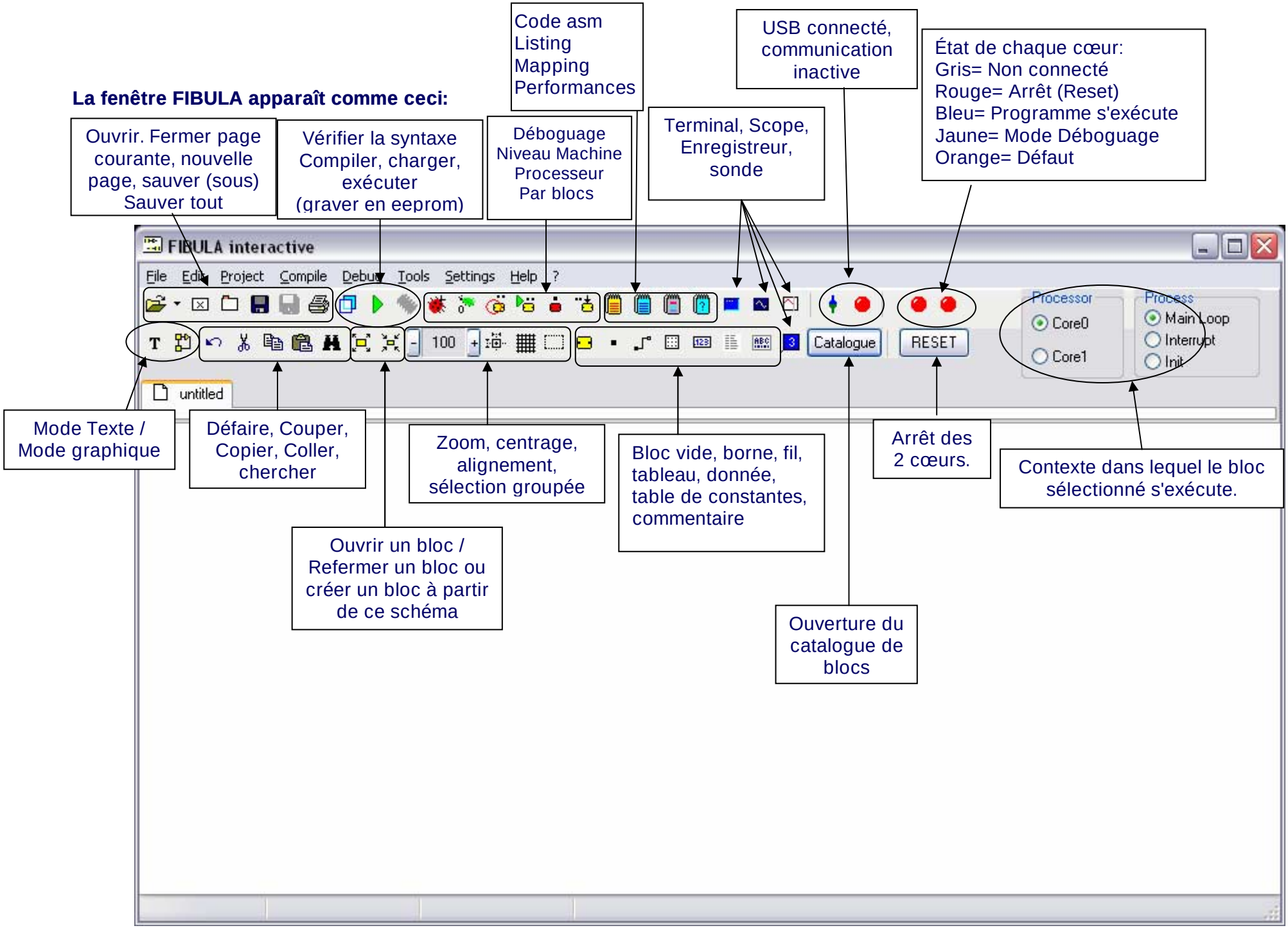
Le logiciel FIBULA est maintenant installé.

2.1.1 Lancement de FIBULA

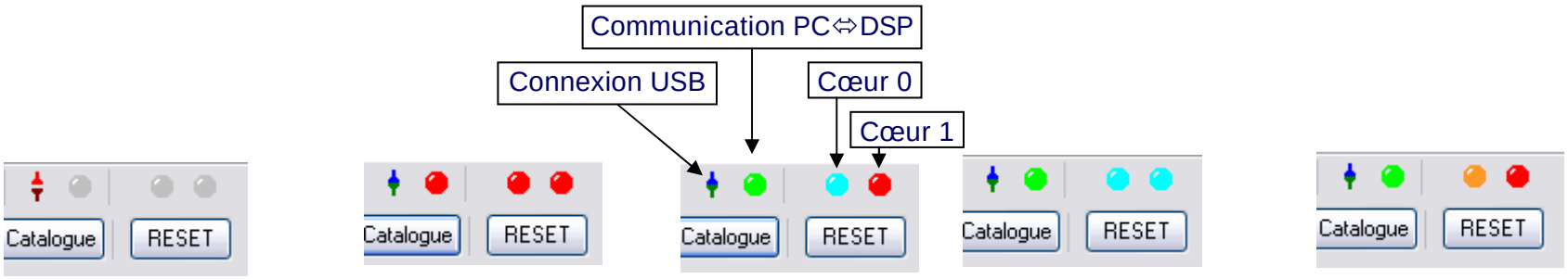
Double-cliquez sur le raccourci FIBULA



La fenêtre FIBULA apparaît comme ceci:



2.1.2 Fonctions des voyants

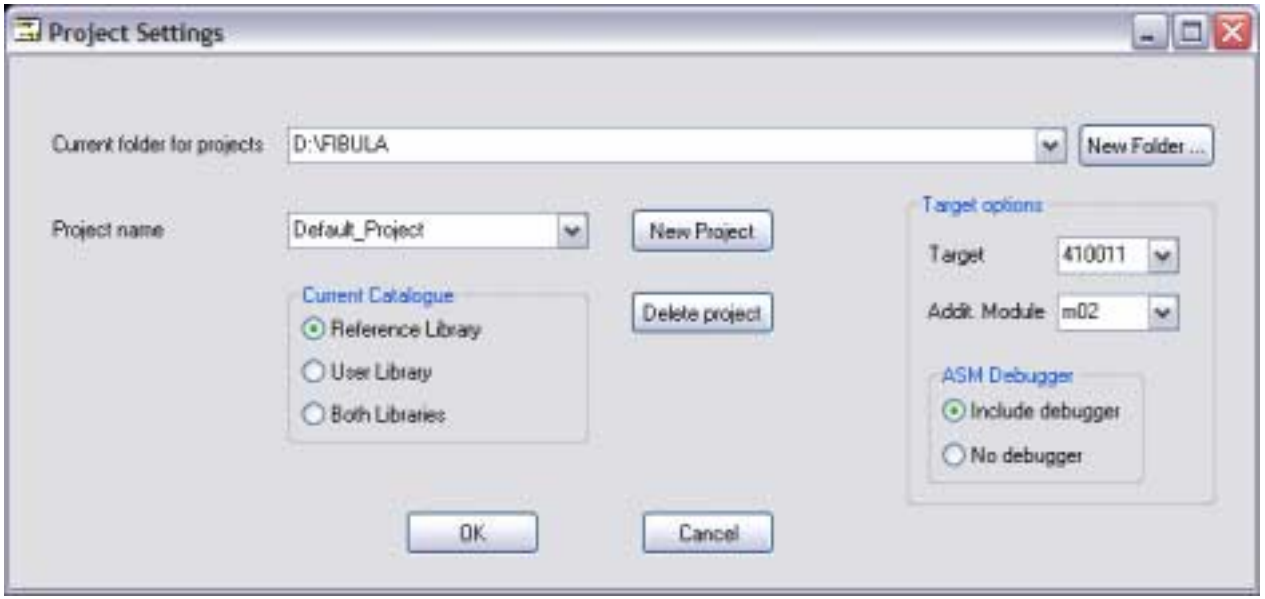


Boîtier HT ou USB non connecté **Boîtier connecté, DSP RESET** **Cœur 0 tourne** **Les cœurs 0 et 1 tournent** **Comm OK, C0 "planté", C1 Reset**

A la mise sous tension, les voyants indiquent que les deux cœurs (leds rouges) sont dans l'état RESET (attente d'un téléchargement) et que la communication PC-DSP ne se fait pas encore. (led rouge). Lorsque l'on a téléchargé un programme sur la cœur 0 la led orange passe au vert si le PC reçoit une réponse du DSP quand il l'interroge. La led bleue indique que la boucle principale (Mainloop) s'exécute pour le cœur concerné. Si un programme a été téléchargé dans un cœur, et que la boucle principale ne s'y exécute pas, la led concernée passe à l'orange. Il peut s'agir d'une attente bloquante mal gérée ou simplement d'un bug ayant provoqué l'égarement du programme.

2.2 Créez un projet

Créez un projet si vous utilisez FIBULA la première fois, ou si vous changez de domaine d'application
Le projet définit où vous allez sauver vos fichiers personnels.
Dans la barre de menu de FIBULA, cliquez sur **Project,** La fenêtre du projet s'ouvre:



Fenêtre du projet

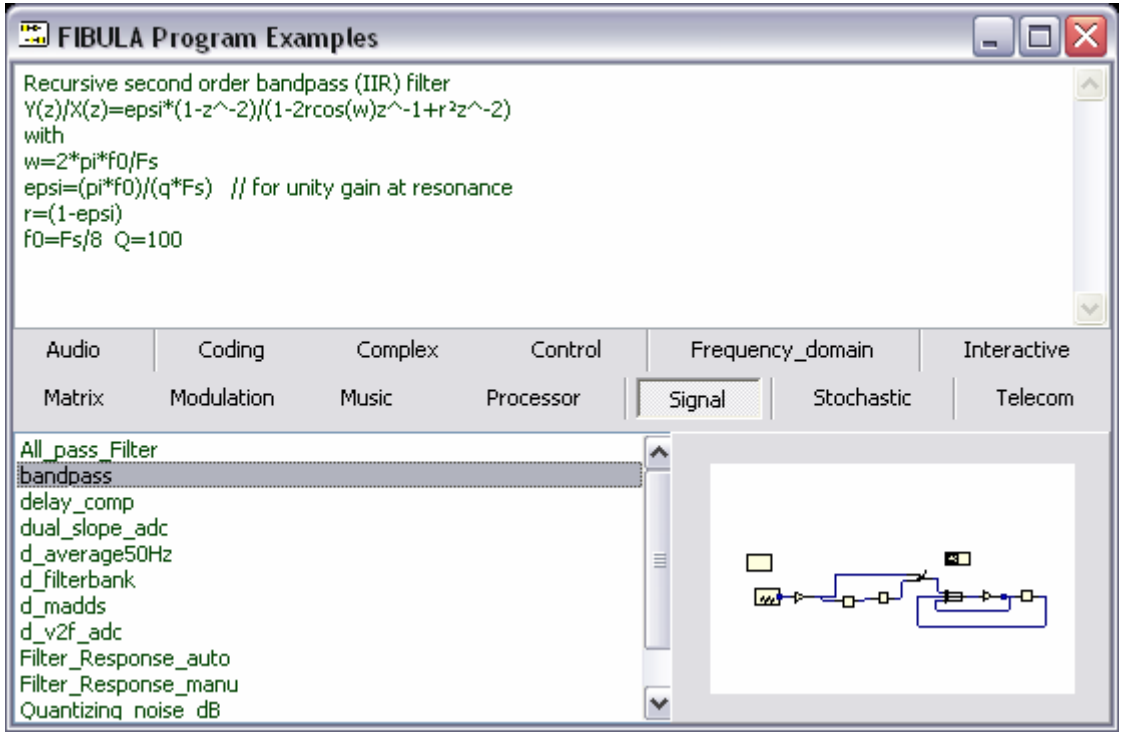
Lors d'une première utilisation, un projet par défaut **Default_Project** est créé dans le répertoire **Mes documents\Fibula**

Pour l'instant, cliquez sur **OK** pour accepter les paramètres par défaut.

Pour plus d'explications sur la gestion des projets, reportez-vous au chapitre 9.

2.3 Programmes d'exemples*

Pour découvrir quelques exemples de programmation, cliquez dans la barre de menu: **File | Example programs**
La fenêtre d'exemples apparaît



Choisissez un onglet et une application en sélectionnant un élément de la liste en bas à gauche. Sa description apparaît dans la la partie supérieure, et une miniature du schéma. Prenons comme exemple le **passe bande** de l'onglet **Signal**
Pour charger le programme choisi dans FIBULA, **double-cliquez sur la ligne sélectionnée ou sur la miniature**.

Pour compiler, charger et exécuter le code de ce programme, cliquez sur ce bouton

Pour voir ce qui se passe, ouvrez l'oscilloscope en cliquant ce bouton:

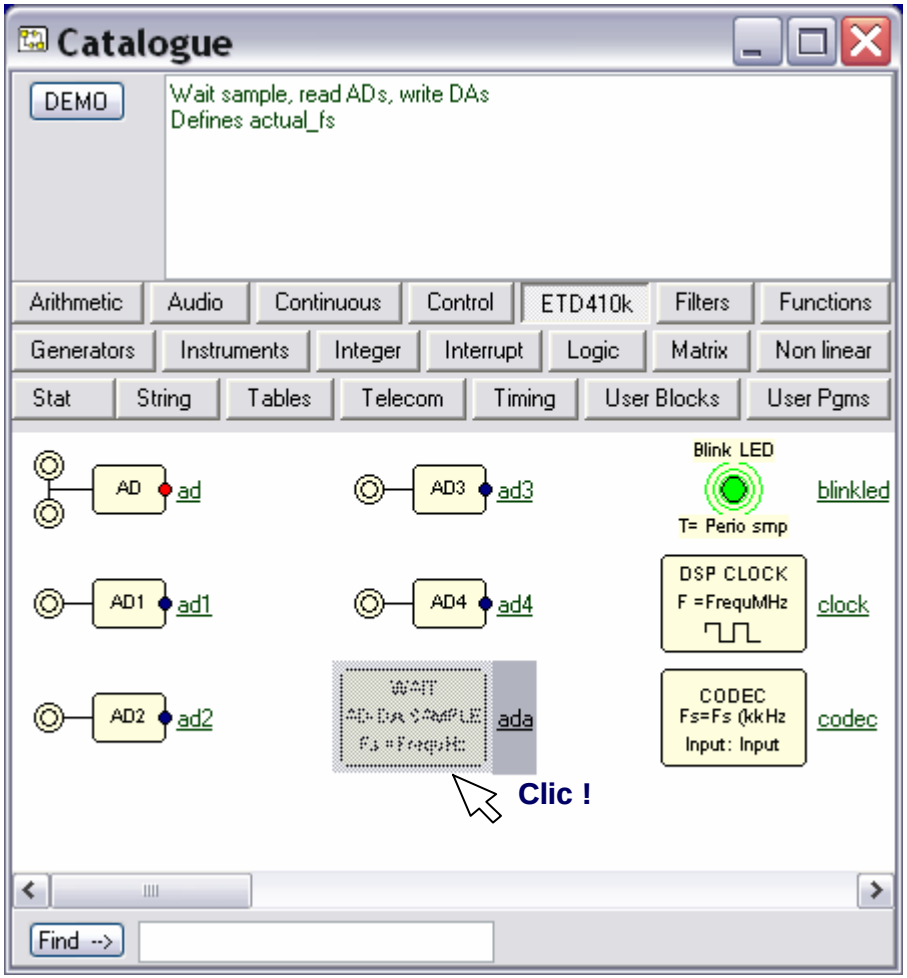
Cliquez sur le bouton START de l'oscilloscope.
Dans V3.1, le scope se synchronise automatiquement Sinon, pour stabiliser l'image, choisissez: **TRIGGER Chan level**, **Channel 1**, **Level 0**, **Up**
Dans le schéma, sélectionnez par un simple clic le générateur sinusoïdal situé à gauche. Faites varier la fréquence de la sinusoïde en **tournant la roue de la souris** jusqu'à obtenir une fréquence de 12,5kHz. (Réglage fin: touche Ctrl appuyée) Vous devriez alors observer la résonance du filtre sur le canal 2 du scope.

*** NB Les programmes d'exemples n'existent que si l'option "Inclure les programmes d'exemples" a été validée lors de l'installation.**

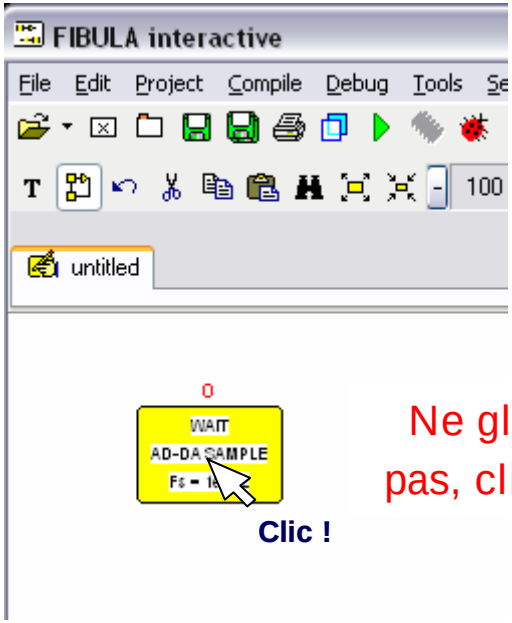
3 Créez un programme simple

3.1 Saisie d'un schéma

Cliquez sur le bouton CATALOGUE
La fenêtre catalogue s'ouvre.



Choisissez l'onglet "ETD410k" qui représente les blocs d'entrées sorties disponibles sur le boîtier DIDALAB connecté;
Sélectionnez le bloc **WAIT AD-DA SAMPLE** (**ada**) en cliquant sur son icône, puis cliquez sur la surface de travail blanche de FIBULA.
Ce bloc est maintenant instancié. Son rôle est d'échantillonner les entrées–sorties à la fréquence d'échantillonnage Fs dont la valeur par défaut est 100kHz.

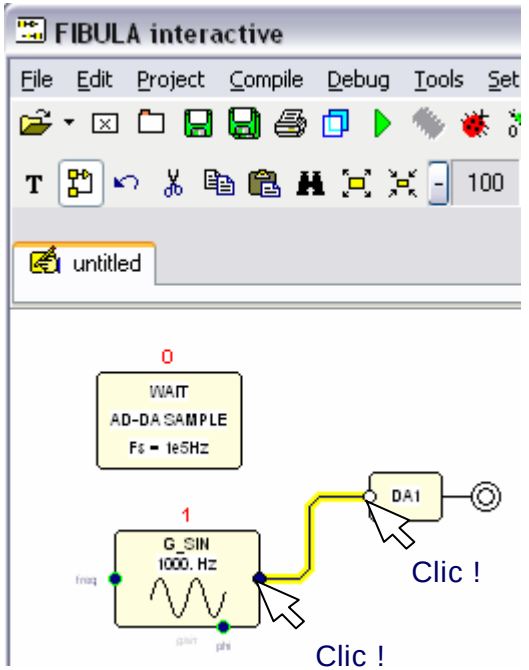


Ne glissez pas, cliquez !

Déposez un bloc **DA1** de la même manière. Sélectionnez l'onglet "Generators".
Déposez encore un générateur sinusoïdal **G_SIN**.

3.2 Créez une connexion

Cliquez sur ce bouton de la barre d'outils:



Cliquez sur la borne de droite du bloc **G_SIN** (sortie) puis cliquez sur la borne d'entrée de **DA1**. Le convertisseur Numérique Analogique **DA1** est maintenant connecté à la sortie du bloc générateur de sinusoïde **G_SIN**.

Commandes d'édition:

Pour sélectionner un objet, cliquez dessus avec le bouton gauche. Il apparaît coloré en jaune.
Pour sélectionner plusieurs objets, cliquez sur ceux-ci en maintenant la touche Majuscule enfoncée,
ou cliquez sur un espace vide et glissez la souris, bouton gauche appuyé pour former un rectangle de sélection.
Vous pouvez copier, couper ou coller les objets sélectionnés en tapant respectivement Ctrl-C, Ctrl-X, Ctrl-V.

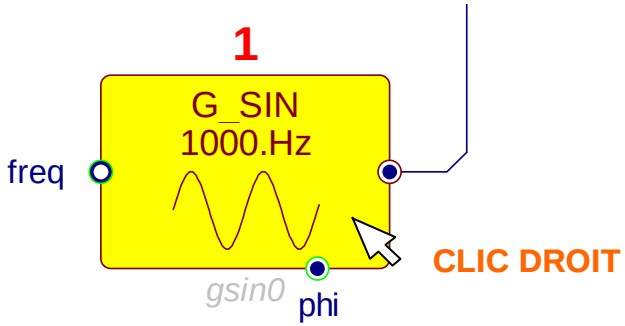
Pour déplacer les objets du schéma, cliquez dessus, puis en glissez la souris, bouton gauche appuyé.
Pour zoomer, utilisez la molette de la souris. Pour déplacer le schéma entier, cliquez (bouton droit) sur un espace vide, puis glissez la souris, bouton droit appuyé.

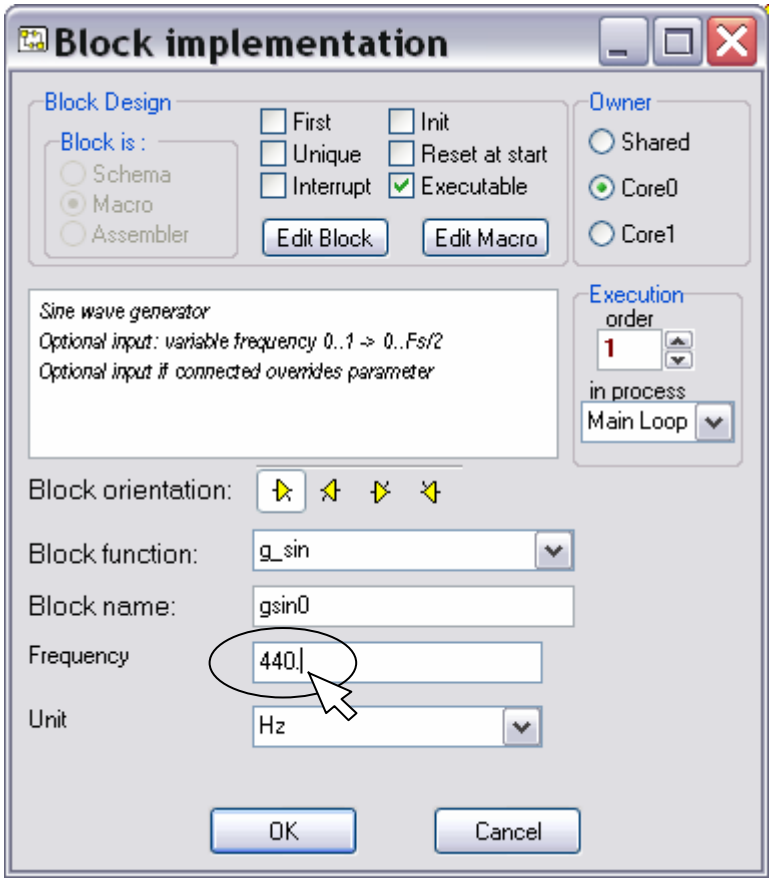
Vous pouvez supprimer un ou plusieurs objets en les sélectionnant, puis en tapant la touche DEL ou SUPPR
Attention Le bouton "Undo" (ctrl-Z) permet uniquement de récupérer un seul niveau de schéma avant la dernière suppression

3.3 Réglage des paramètres d'un bloc

Pour obtenir une sinusoïde de 440Hz au lieu de 1000Hz:

Cliquez sur le bloc **G_SIN** avec le **bouton droit**
Dans la fenêtre "Block Implementation" changez la valeur du paramètre "**frequency**"
Cliquez sur "OK"



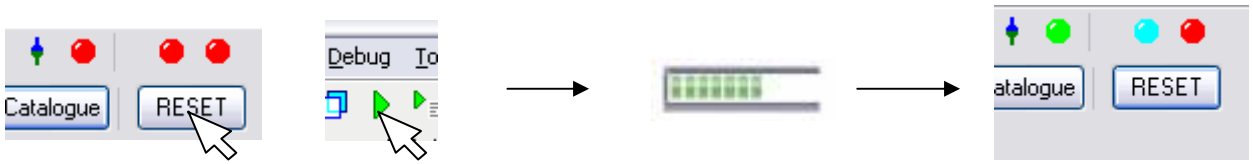


Note: Lorsque le paramètre **Unit** vaut « **Hz ou kHz ou MHz** », les fréquences sont données en absolu, cela implique la connaissance de la fréquence d'échantillonnage Fs, le schéma doit alors obligatoirement comporter le bloc ADA (ou équivalent) qui fournit cette fréquence Fs.

Lorsque le paramètre **Unit** vaut « **Fs/2** », les fréquences sont relatives à Fs/2, c'est-à-dire que 1.0 représente Fs/2 qui est la fréquence maximale autorisée pour un système échantillonné. Dans ce mode, pour obtenir 440Hz, le paramètre **Frequency** aurait alors la valeur 440/(Fs/2) = **0.0088**

4 Compilation et essai du programme

4.1 Compilation et exécution

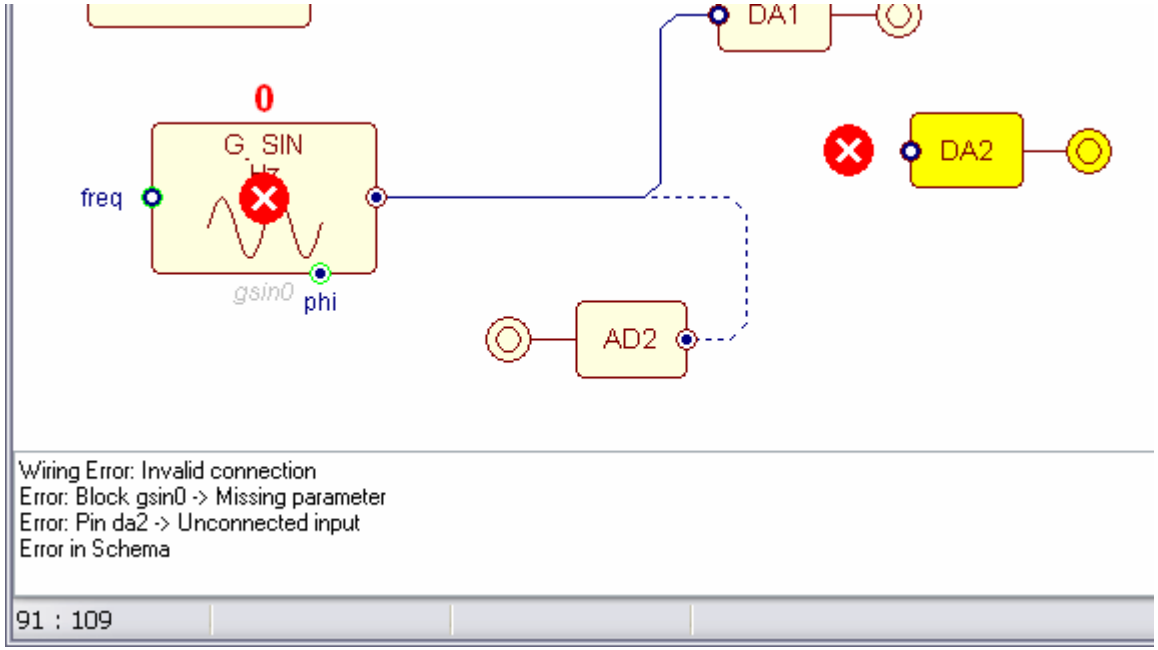


Si un programme s'exécutait déjà, arrêtez-le en cliquant sur le bouton RESET
Cliquez sur le bouton "Compile + run"

En l'absence d'erreur, après un bref instant, la led d'état du cœur 0 du DSP passe au bleu pour indiquer que le programme s'y exécute.
Si le programme a été écrit pour le cœur 1*, la led de droite passe au bleu.
Si le programme comporte des blocs du cœur 0 et du cœur 1, deux programmes sont téléchargés et les deux leds passent au bleu.
Une fois que le programme tourne, vous pouvez l'arrêter en cliquant sur le bouton **RESET**.

* Note:
Lorsque vous mettez en œuvre l'oscilloscope virtuel, la table traçante virtuelle, le Terminal virtuel, ou encore les réglages de paramètres interactifs, le canal de communication entre le DSP et votre ordinateur est géré par le cœur 0. Il faut donc dans ce cas que le cœur 0 soit activé, fut-ce au besoin par un unique bloc NOP (Control).

4.2 Erreurs dans le schéma



Avant de compiler le schéma, FIBULA procède à quelques vérifications syntaxiques et affiche les erreurs trouvées. Les erreurs les plus fréquentes sont

- Un champ laissé en blanc dans la saisie des paramètres,
- Une entrée requise non connectée
- Un court-circuit entre 2 sorties → **NB autorisé à partir de la V.3 pour permettre le OU câblé entre plusieurs blocs.**
- Une connexion redondante
- Une connexion entre types de données incompatibles (par exemple entre fractionnaire et flottant)

4.3 Testez votre application

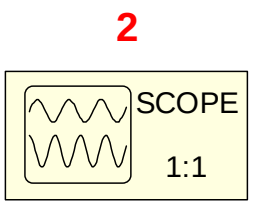
4.3.1 Test matériel direct

Connectez un oscilloscope sur la sortie **DA1** de la carte Didalab pour voir une sinusoïde 440Hz d'amplitude 2,5 ou 10 Volts selon les versions, ou connectez sur cette sortie sur un haut-parleur amplifié d'ordinateur pour entendre le LA de référence des musiciens.

4.3.2 Test avec les instruments virtuels

Si vous ne disposez pas d'un oscilloscope, vous pouvez mettre en œuvre un instrument virtuel pour observer ou analyser le signal.
L'oscilloscope virtuel **SCOPE** utilise le port SPI pour transmettre les données de la maquette vers l'ordinateur.
Le logiciel FIBULA-G affiche les courbes correspondant aux trames reçues.

4.3.3 L'oscilloscope virtuel multicanaux SCOPE



Pour voir des signaux sur l'oscilloscope virtuel, instanciez le bloc **SCOPE**, mais **ne le connectez pas.**
Attention: l'oscilloscope virtuel ne peut être instancié qu'une fois au maximum.
Pour ne pas avoir à saisir les noms de tous les signaux à visualiser, on utilise des sondes pour définir les bornes d'intérêt.
On peut placer les sondes sur toute borne d'un bloc du schéma principal ou de ses sous-schémas, avec les restrictions suivantes:

Notes:

- On ne peut pas observer à l'oscilloscope les bornes de type INTEGER et de type FLOAT du fait de leur dynamique trop étendue. On peut toutefois convertir ces signaux en fractionnaire pour les visualiser dans une fenêtre limitée.
- On ne peut pas visualiser la totalité des éléments d'un tableau (bornes matricielles carrées), mais on peut visualiser les éléments de cette matrice pris individuellement (bornes rondes portant pour nom l'adresse relative de l'élément).

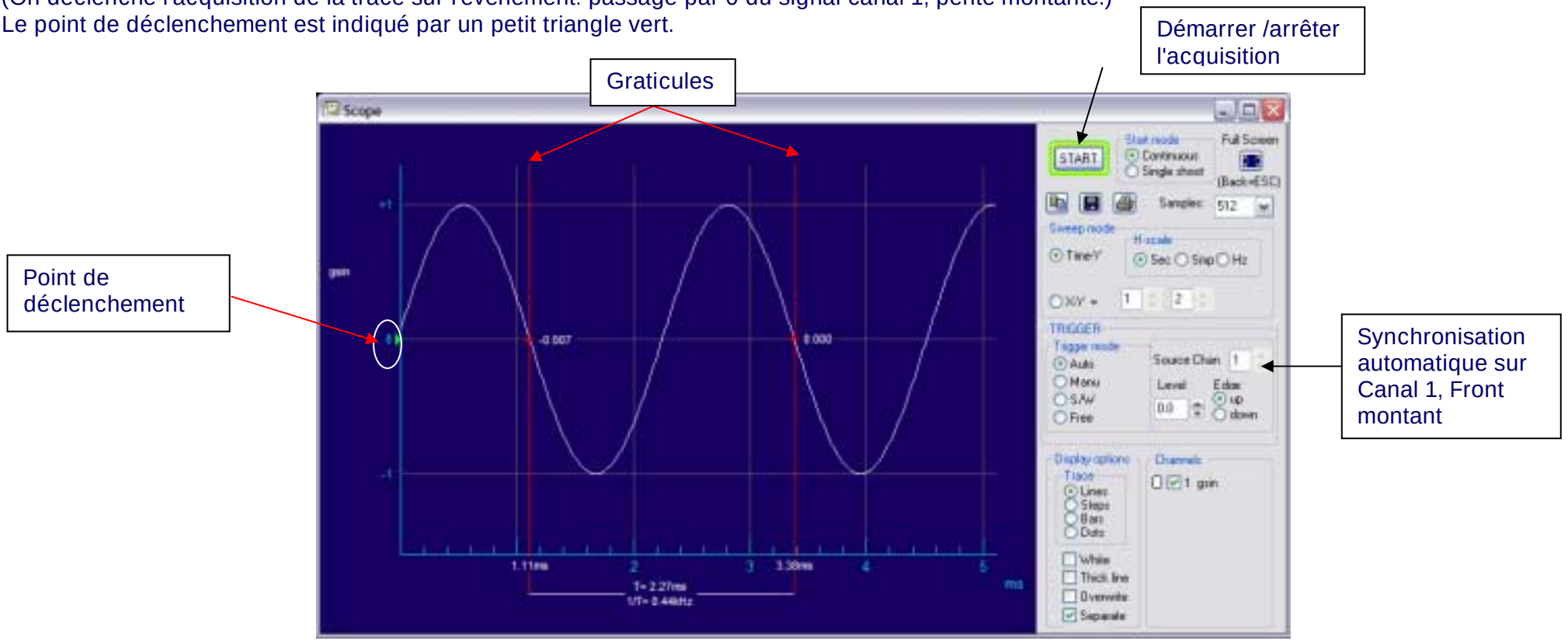
Pour installer une sonde, activez ce bouton  puis cliquez sur les différentes bornes de sorties à visualiser.

Les sondes apparaissent en bleu. Le chiffre représente le N° du canal.
A la différence de FIBULA V1, sur les Versions 2 et 3, vous pouvez poser et retirer des sondes ou encore changer leur numéro pendant que le programme s'exécute.

Pour faire apparaître le scope, soit cliquez sur ce bouton, soit double-cliquez l'objet **SCOPE** du schéma.

Pour démarrer l'acquisition, activez le bouton **START.** L'affichage temps réel est similaire à celui d'un véritable oscilloscope, c'est-à-dire qu'en l'absence de synchronisation, l'image défile sur l'écran.

Pour stabiliser l'image, dans la case **TRIGGER** sélectionnez "**Chan. Level"** avec **Channel=1 Level=0** et **Edge=Up**
(On déclenche l'acquisition de la trace sur l'événement: passage par 0 du signal canal 1, pente montante.)
Le point de déclenchement est indiqué par un petit triangle vert.



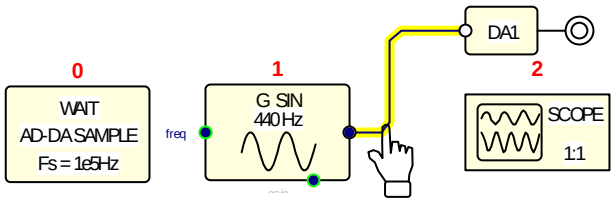
Pour faire apparaître un graticule, cliquez (**bouton gauche**) sur l'écran de l'oscilloscope, glissez à l'endroit désiré, cliquez une deuxième fois. Pour faire apparaître un deuxième graticule permettant par exemple de mesurer la période du signal, cliquez avec le **bouton droit.**

En cliquant en premier avec le bouton droit sur la trace d'un canal, on peut changer son mode de représentation:
Lines= interpolation linéaire entre les échantillons. **Steps**=échelons, **Bars**= un segment vertical pour chaque échantillon, **Dots**= un point (un pixel) par échantillon, **Color**= couleur de la trace.

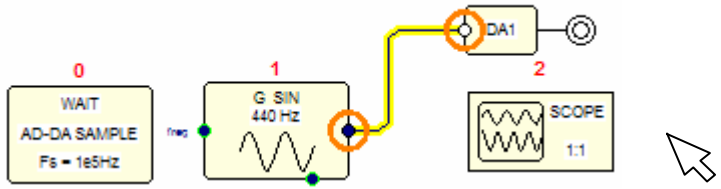
On peut visualiser des signaux fractionnaires (intervalle [-1..+1[ ), des signaux complexes (partie réelle, partie imaginaire), ou des signaux booléens (True-False).
L'échelle horizontale est graduée soit en secondes, soit en échantillons, ou bien en Hz pour les spectres.
NB: L'utilisation d'échelles Temps / Fréquence absolues implique que la fréquence d'échantillonnage ait été définie (un des blocs ADA, CODEC, ICC_WAIT ou FS_TIMER doit obligatoirement apparaître sur le schéma)

4.3.4 Oscilloscope SCOPE utilisé en mode "Pointe de Touche"

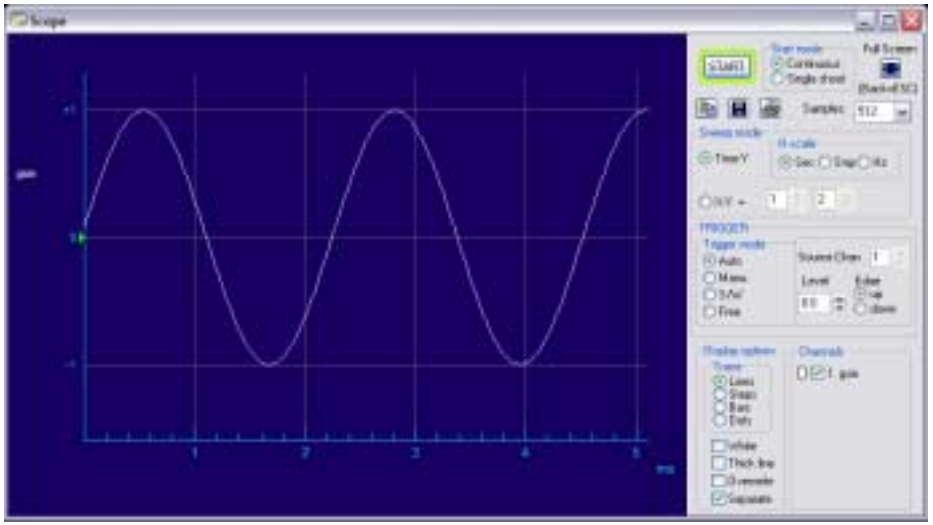
Pour comprendre rapidement pourquoi un schéma ne fonctionne pas, le mode "Pointe de Touche" de l'oscilloscope permet d'observer un signal à la fois, simplement en glissant la souris au dessus d'un fil ou au dessus d'une borne. Pour cela, le schéma doit comporter le bloc SCOPE et il faut que le programme s'exécute et que le scope soit démarré. On glisse le curseur de la souris au dessus du fil ou de la borne à observer. La borne apparaît alors sélectionnée et l'oscilloscope passe en mode monotrace pour afficher le signal de la borne.
Si l'on clique sur le fil ou la borne, cela "attache la sonde", ce qui permet de libérer le curseur de la souris pour agir sur les contrôles de l'oscilloscope. On "détache" la sonde en cliquant sur un espace vide.



Mode "sonde volante"

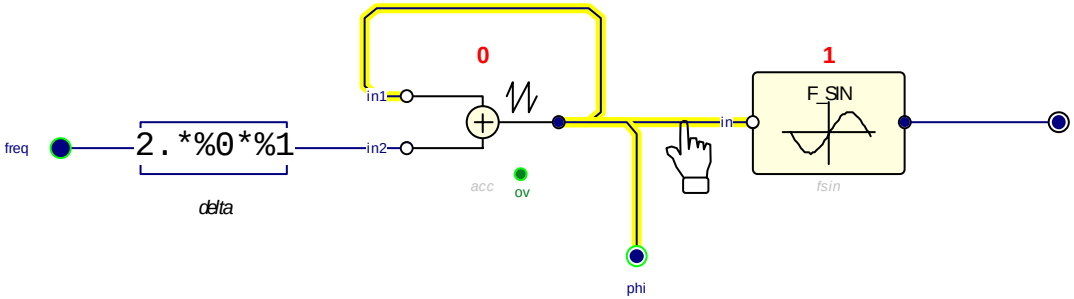


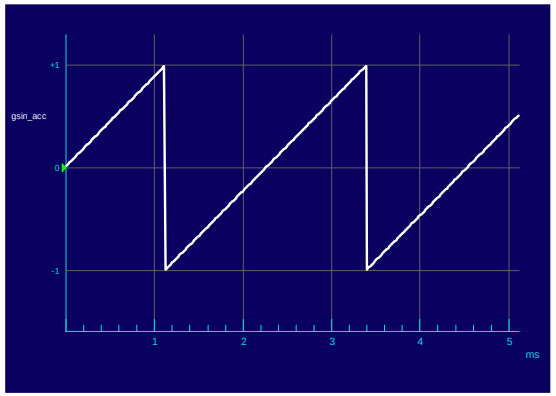
Mode "sonde attachée"



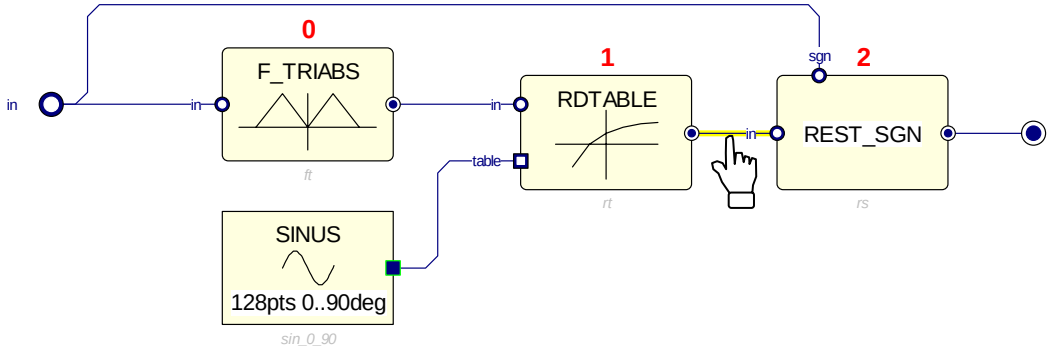
4.3.5 Exploration des signaux internes d'un bloc

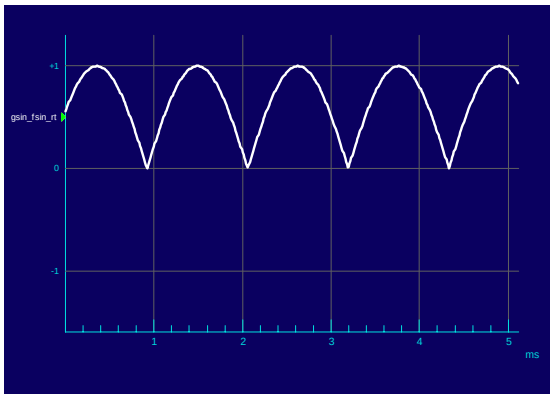
Pendant le programme s'exécute, vous pouvez ouvrir un bloc en double-cliquant dessus et utiliser le mode "pointe de touche" pour en observer les signaux internes. Par exemple, double-cliquez sur le bloc G_SIN: son schéma comporte un générateur de phase et une fonction Sinus:





Si vous double-cliquez maintenant sur le bloc F_SIN vous pourrez voir comment est réalisée la fonction Sinus à l'aide d'une table interpolée d'un quart de période:

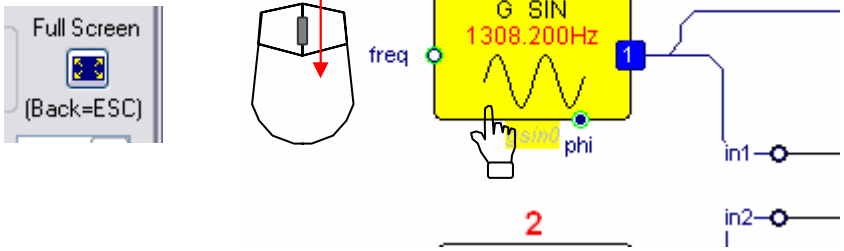




4.3.6 Réglage interactif de paramètres en cours d'exécution

Certains blocs permettent le réglage d'un paramètre en cours d'exécution. Ce sont ceux pour lesquels le paramètre se ramène à une seule variable en mémoire que l'on peut modifier sous interruption. Pour régler la fréquence de la sinusoïde en cours d'exécution, sélectionnez le bloc **G_SIN**, puis actionnez la molette de la souris. Vous verrez alors la période se modifier comme ci-dessous. (Cochez la case **Overwrite** de l'oscilloscope pour obtenir cette image. Les fréquences varient en raison géométrique).
Lorsqu'on clique avec le bouton droit, ou lorsque l'on recompile le programme, le paramètre réglable reprend sa valeur d'origine.

Mode Plein Ecran
Si vous choisissez le mode plein écran, l'écran restera toujours au premier plan. (on sort de ce mode avec la touche Échap)



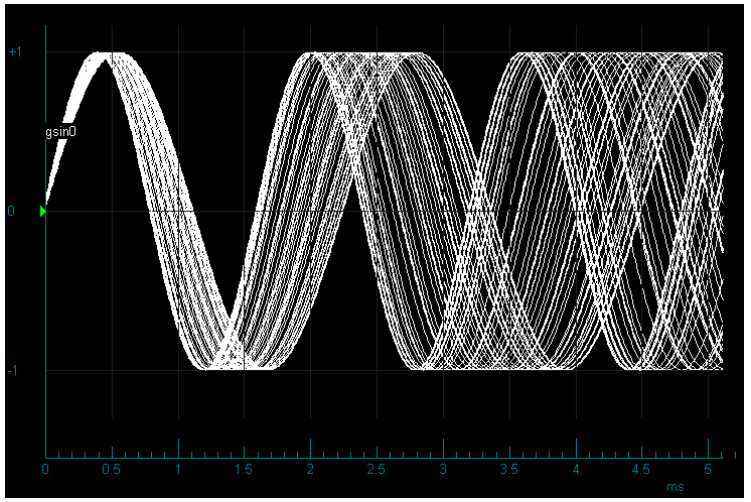
Les réglages de blocs par molette sont en progression géométrique

Grossier: Molette avec touche **SHIFT** = **+/- 1dB** par cran (r=1,2589..)

Normal: Molette seule = **+/- 0,1dB** par cran (r=1,02329..)

Fin: Molette avec touche **CTRL** =**+/- 0,01dB** par cran (r=1,002305)

Musical: Molette avec touche **ALT** = **+ ou- 1/2 ton** par cran (r=1,05946)



Réglage de la fréquence d'un générateur sinusoïdal en cours d'exécution

4.3.7 Test dans le domaine fréquentiel

4.3.7.1 De la nécessité des interruptions

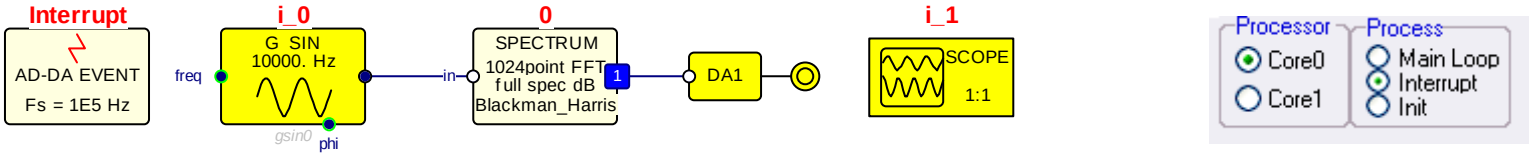
A partir de FIBULA V2.x, nous avons décidé de rendre les schémas comportant un analyseur de spectre plus explicites.
L'algorithme d'analyse spectrale (FFT + calcul de puissance moyenne en dB) est relativement long.
Par exemple pour un spectre sur 1024 points avec fenêtre de Blackman Harris et affichage de densité spectrale de puissance moyenne en dB, cela prend autour de 100 000 cycles machine. En supposant l'horloge réglée à 200MHz (maximum pour le DSP56720), le calcul dure de l'ordre de **500µs**.

Dans la plupart des cas, la fréquence d'échantillonnage sera de 100kHz; l'algorithme FFT dure donc 50 échantillons, ce qui correspond avec 1024 points à un taux de recouvrement de 95% entre deux algorithmes successifs. Cela est plus que confortable.

Le programme d'analyse spectrale s'exécute donc dans la boucle principale, alors que tous les autres blocs, ainsi que les flux d'E/S de l'analyseur s'exécutent sous interruption. Cette interruption, cadencée à la fréquence Fs est installée par le bloc **irq_ada** "AD-DA Event".
Pour plus de précisions sur l'analyseur de spectre, reportez-vous à l'Annexe 3 à la fin de ce document.

4.3.7.2 Spectre du générateur sinusoïdal

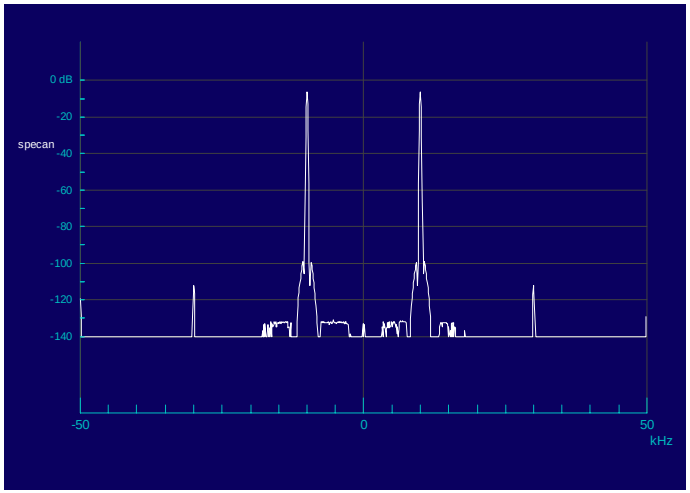
Pour tester le générateur sinusoïdal dans le domaine fréquentiel, créez une nouvelle page à l'aide de ce bouton: 
Créez le schéma ci-dessous. L'analyseur de spectre est dans **Instruments**.
Sélectionnez Mainloop pour l'analyseur de spectre et Interrupt pour tous les autres blocs.
Pour obtenir le spectre ci-dessous, choisissez 1024 points, le mode **Full**, **dB**, une fenêtre de **Blackman-Harris**, **dc.**
Sur l'oscilloscope, sélectionnez
Samples: **1024**
Trigger: **S/W synchro**
H-Scale: **Hz**
Vous pouvez faire varier la fréquence du générateur sinusoïdal en cours d'exécution (3.2.5) et observer le déplacement lent des 2 raies fondamentales, et le déplacement rapide ainsi que le repliement des petites raies harmoniques. La ligne de base à -120dB représente le niveau de bruit de quantification et de calcul.



Les blocs Temps réel (en jaune) sont exécutés dans l'interruption cadencée à la fréquence Fs.

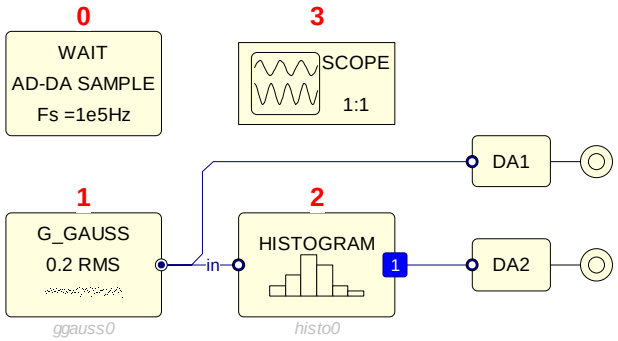
L'analyseur de spectre s'exécute de manière asynchrone dans la boucle principale.

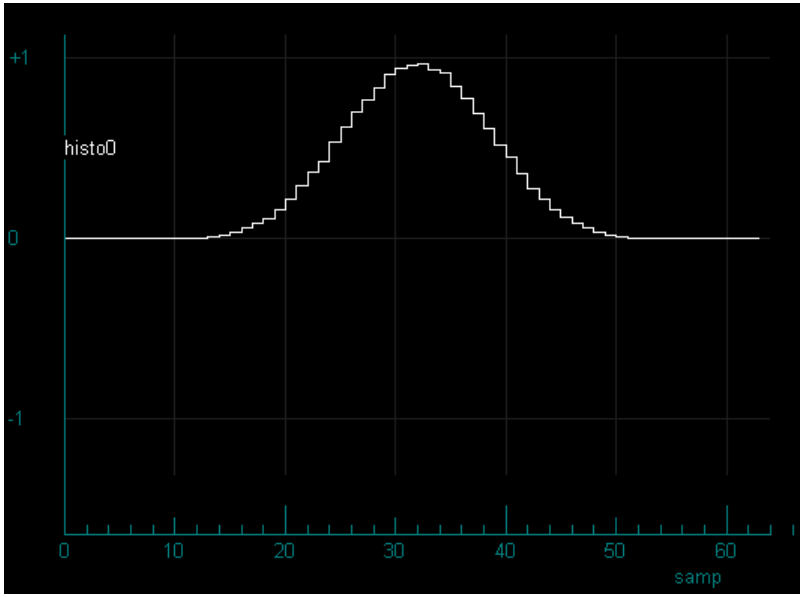
Spectre obtenu avec G_SIN à 10kHz
1024 points
Mode dB, Fenêtre de Blackman & Harris →



4.3.8 Test dans le domaine statistique

L'outil d'analyse statistique est l'Histogramme. Il donne une estimation de la densité de probabilité d'un signal aléatoire sur la plage [-1..+1[, lorsque son paramètre **gain** vaut 1.0





Pour tester l'histogramme, créez le schéma ci-dessus sur une nouvelle page, prenez un générateur de bruit gaussien d'écart type (=valeur efficace) 0.2, et réalisez l'histogramme sur **64 classes** avec un gain de **0.5** pour éviter la saturation.
Sur l'oscilloscope, choisissez:
Samples=**64**
H-Scale: **Smp** pour représenter les 64 classes entre -1 et +1
Trigger: **S/W synchro** pour stabiliser l'image
Mode de représentation (clic droit sur l'image): **Steps** ou **Bars**
Éventuellement, redimensionnez l'oscilloscope pour obtenir une image à la taille désirée.

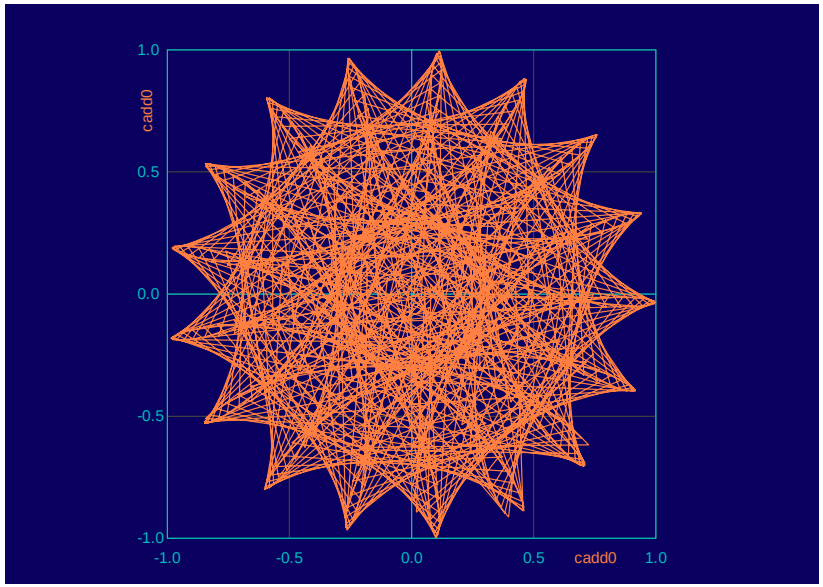
4.3.9 Options d'affichage du Scope

Lines	Courbe obtenue par interpolation linéaire entre points
Steps	Échelons (style histogramme)
Bars	Segments verticaux
Dots	Points
White	Fond blanc
Thick line	Lignes d'épaisseur 3 pixels (plus visibles en vidéo projection)
Overwrite	Rémanence des traces permettant leur superposition
Separate	Courbes séparées, un espace pour chaque courbe, ou toutes les courbes occupent la totalité de l'écran

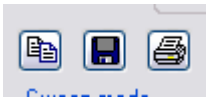
4.3.10 Mode X-Y

Les deux champs définissent le N° de la Trace X et celui de la trace Y
Lorsque l'on désire visualiser la **trajectoire d'une variable complexe**, les deux champs ont le **même numéro** de trace (exemple ci-contre).

Démo du bloc CADD (Complex Add)
Trajectoire de la variable complexe du canal 3



4.3.11 Récupération des données du Scope



A l'aide de ces trois boutons du scope, vous pouvez

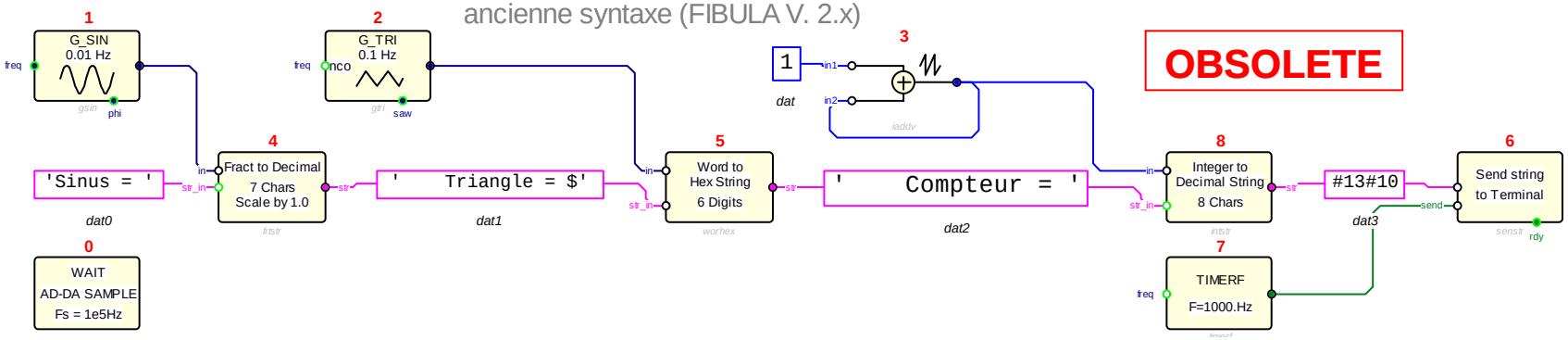
- Copier l'image du scope en format vectoriel
- Sauver les données dans un fichier ASCII pouvant être lu par Matlab ou Excel
- Imprimer l'image

Plus de détails au chapitre 11.

4.3.12 Sorties Texte

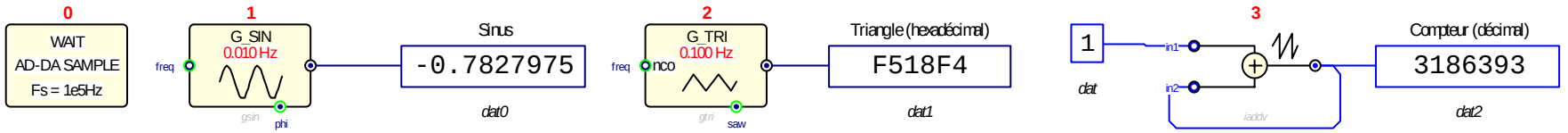
Dans les versions précédentes de FIBULA, l'obtention d'une chaîne de caractères comportant des champs de données variables se faisait en concaténant des segments de chaînes constants avec des blocs de conversion pour les parties variables comme dans le schéma ci-dessous:

Création de chaînes comportant des champs variables
ancienne syntaxe (FIBULA V. 2.x)

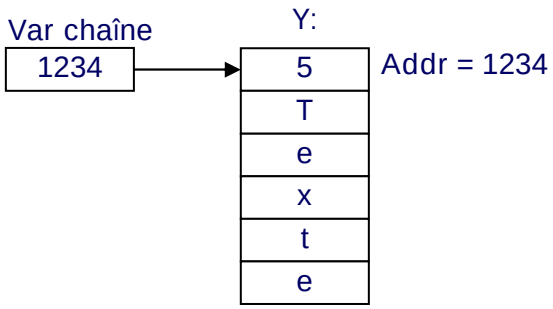


Cette technique est certes pédagogique … mais extrêmement lourde !
Elle devient d'autant moins utile que l'interactivité permet un affichage directement sur le schéma comme ceci (voir chap. 5):

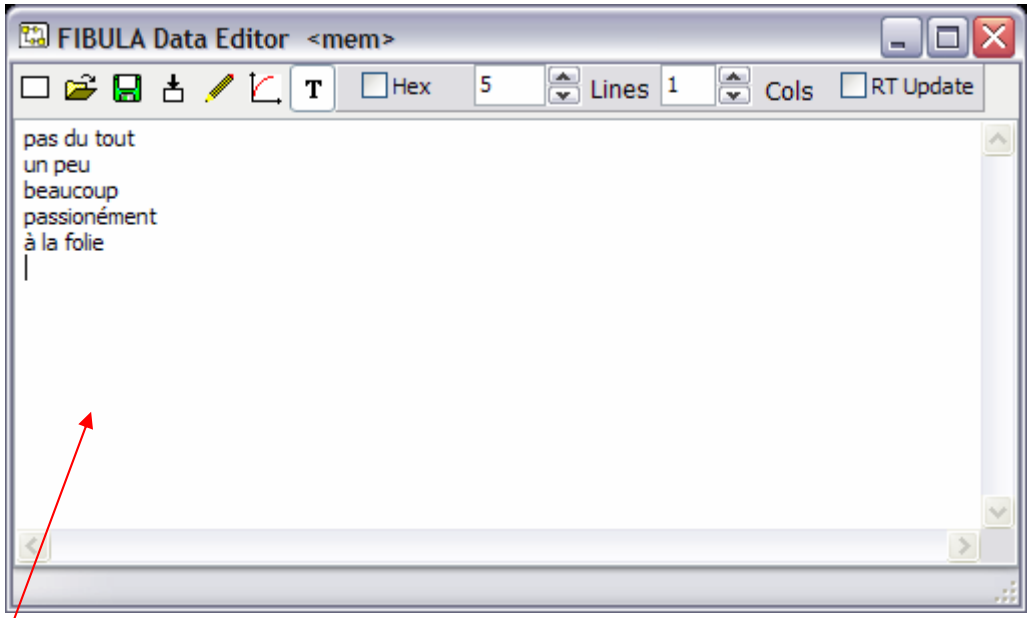
Affichage interactif à l'aide de champs de données

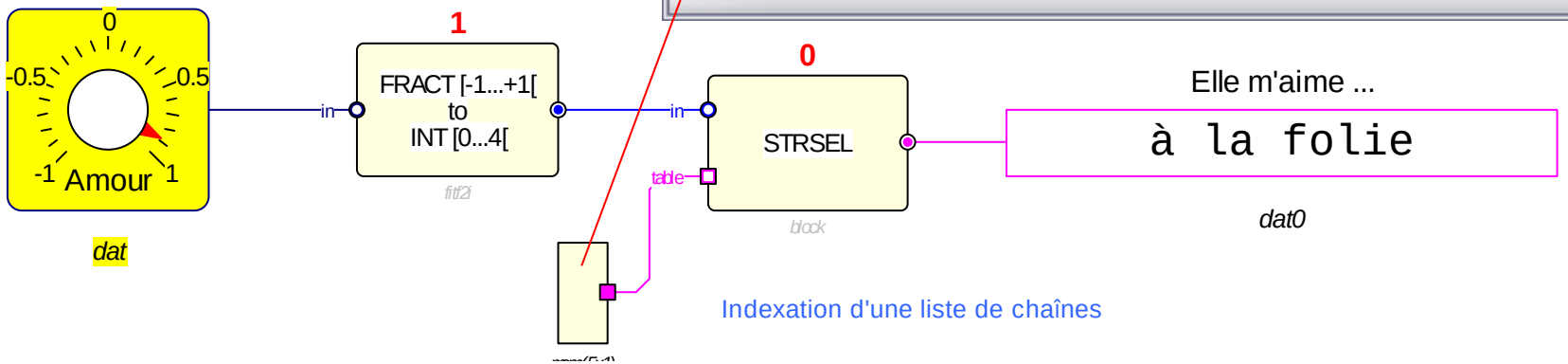


A partir de FIBULA V3.1, les variables chaînes sont des pointeurs qui pointent la structure [nombre de caractères,texte]. Les chaînes peuvent être visualisées, soit à l'aide d'un "voltmètre", soit en les envoyant vers le terminal à l'aide de **sndstring**. On peut désormais créer une liste de chaînes en utilisant l'objet Matrice et sélectionner une chaîne à l'aide de son indice dans la liste à l'aide du bloc **strsel**.



Représentation interne des chaînes à partir de V.3.1

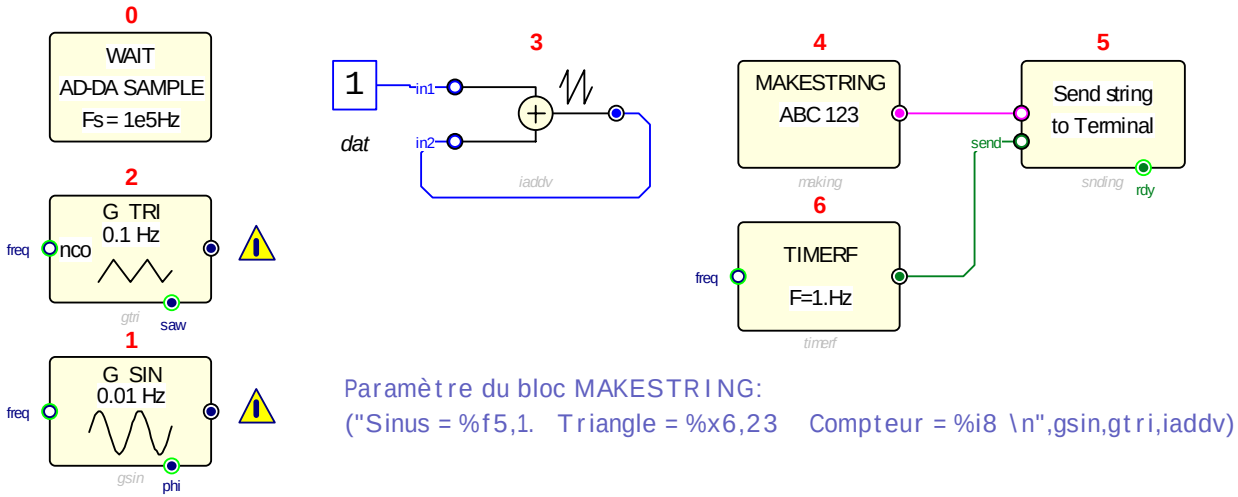


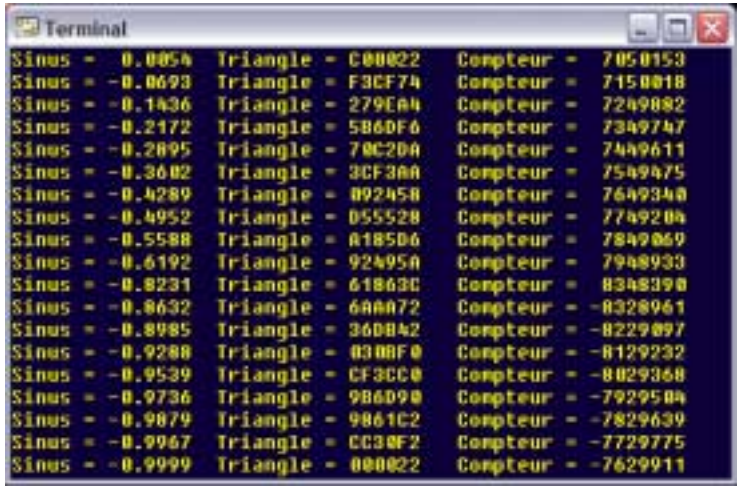


Indexation d'une liste de chaînes

Lorsque l'on désire créer des chaînes comportant des champs de données numériques variables, on utilise le bloc spécial **makestring** qui possède une syntaxe du même style que **PRINTF** en langage **C**

Chaîne avec champs de données variables (FIBULA V3.1)



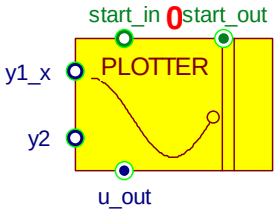


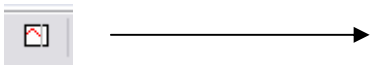
Paramètre du bloc MAKESTRING:

("Sinus = %f5,1. Triangle = %x6,23 Compteur = %i8 \n",gsin,gtri,iaddv)

Le bloc MAKESTRING ne comporte qu'un seul paramètre constitué de la chaîne au format PRINTF. Les parenthèses sont nécessaires pour éviter que Fibula ne confonde la liste de variables avec une liste de paramètres.

4.3.13 L'enregistreur de signaux lents





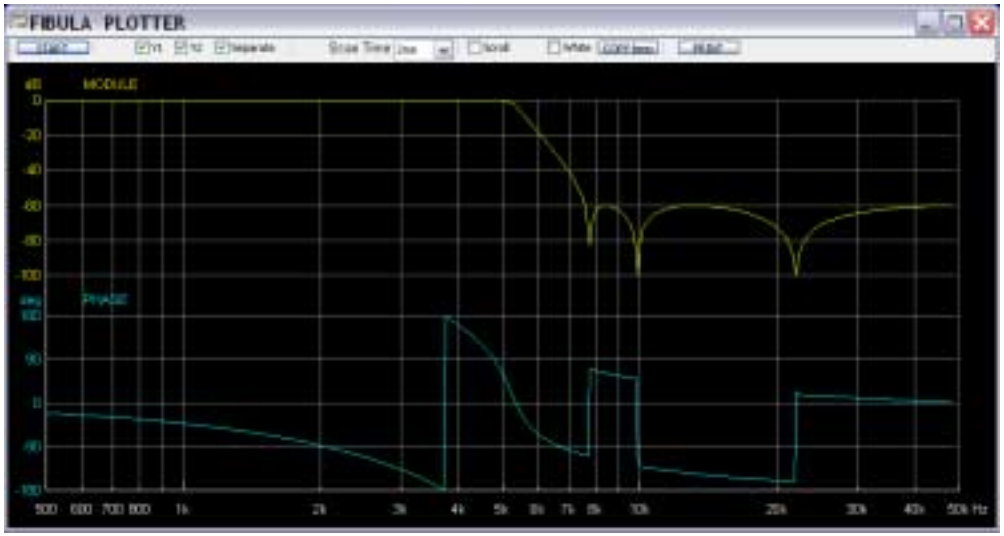


Diagramme de Bode d'un filtre passe-bas numérique du 6ème ordre.

Pour utiliser l'enregistreur de signaux lents, instanciez le bloc PLOTTER sur votre schéma et connectez aux entrées les signaux à enregistrer. Il y a 2 modes: T-Y1-Y2 ou X-Y.

Mode "scroll"
Typiquement, ce mode permet de visualiser en continu sur une image défilante un signal tel qu'un ECG ou un relevé de température.

Réponse d'un système
L'enregistreur possède une sortie **U_out** qui permet de fournir le signal stimulus d'un système dont on veut enregistrer la réponse. On peut programmer **U_out** pour une progression linéaire ou géométrique.
Typiquement ce dispositif permet d'enregistrer les diagrammes de Bode ou de Nyquist d'un système que l'on excite avec une rampe de fréquence en progression géométrique.

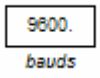
Reportez-vous aux démos du bloc PLOTTER pour voir comment utiliser le bloc plotter.

5 Les données et les objets interactifs

5.1 Les données

On crée une donnée sur le plan en activant ce bouton, puis en cliquant sur le plan, à l'endroit désiré.
Les données associent une valeur à un nom. Cette valeur peut être numérique, littérale, booléenne.

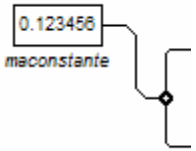
5.1.1 Donnée isolée



Une donnée non connectée est une définition globale qui sera insérée en tête d'un programme.
La donnée ci-contre produira à la compilation la ligne assembleur:

```
bauds       equ         9600.
```

5.1.2 Donnée connectées à une entrée de bloc



Une donnée connectée à l'entrée d'un bloc réserve une case mémoire initialisée par sa valeur.

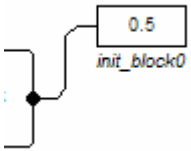
```
                org         y:
maconstante     dc          0.123456
```

La connexion réalise ensuite l'équivalence entre l'entrée du bloc et cette case mémoire

```
block0_in       equ         maconstante
```

Il en résulte que l'on peut connecter la même donnée vers plusieurs entrées
Si vous créez une constante identique à l'intérieur de plusieurs blocs différents, en cochant la case **common**, celle-ci ne sera instanciée qu'une fois.

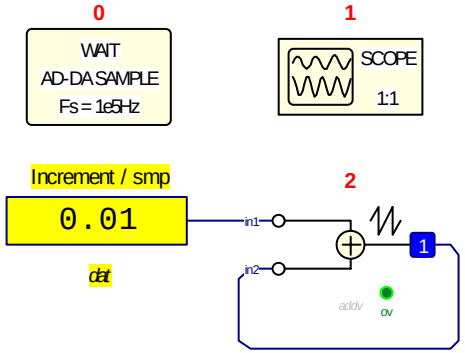
5.1.3 Donnée connectée à une sortie de bloc

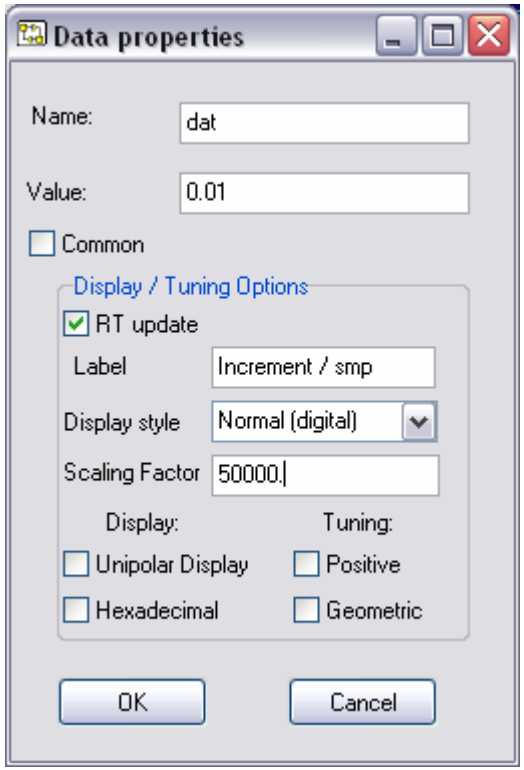


Une donnée connectée à la sortie d'un bloc fournit la **valeur initiale** de cette sortie quand celle-ci doit être différente de 0 au démarrage d'un programme.

Fonction "Voltmètre" Lorsque la **donnée est sélectionnée et que le programme s'exécute**, celle-ci affiche en temps réel la valeur fractionnaire de la borne connectée (voir 5.3)

5.2 Entrée fractionnaire réglable





Proposons-nous de réaliser un générateur de dents de scie à fréquence réglable. Son schéma se résume à un accumulateur à débordement dont l'entrée est reliée à une donnée constante.
Pour éditer les propriétés de la donnée, cliquez avec le bouton droit sur la donnée. Mettez la **valeur de référence** de cette donnée dans le champ **Value**, par exemple 0.01. Pour pouvoir régler cette donnée à l'exécution, assurez-vous que la case **RT update** est cochée (elle l'est par défaut). Dans le champ optionnel **Label** vous pouvez mettre un court descriptif de ce paramètre.
Le champ déroulant **Display style** vous permet 6 représentations graphiques différentes (voir Fig. page suivante):

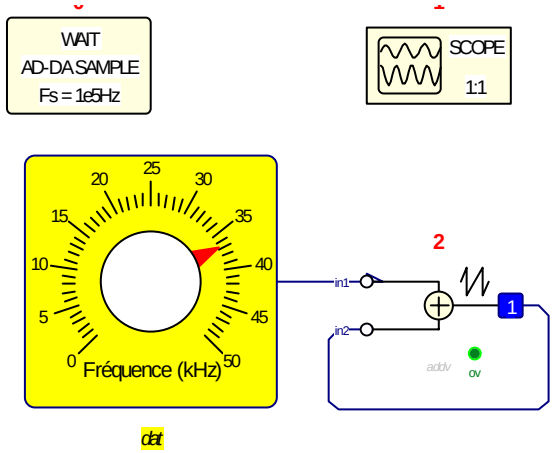
- **Normal (digital)** On obtient un affichage numérique décimal dont la précision (nombre de chiffres) est déterminée par la largeur du rectangle.
- **Circular** On obtient un bouton de réglage ou "potentiomètre" rotatif
- **Rectangular** (rectangle plus large que haut) On obtient un potentiomètre linéaire horizontal
- **Rectangular** (rectangle plus haut que large) On obtient un potentiomètre linéaire vertical
- **Bar** (rectangle plus large que haut) On obtient une barre linéaire horizontale
- **Bar** (Rectangle plus haut que large) On obtient un thermomètre ou un réservoir

Options:

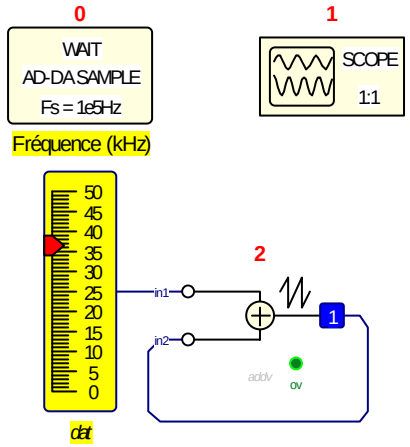
- En cochant la case **Hexadécimal,** (en mode normal), on obtient une représentation sous forme de 6 ou 12 caractères hexadécimaux selon qu'on est en simple ou double précision
- La case **Positive** contraint la donnée à rester positive ou nulle lors du réglage
- La case **Unipolar** donne lieu à des graduations uniquement positives ou uniquement négatives. Il est logique de cocher **Unipolar** dans le cas où l'on a coché **Positive.**
- Le champ **Scaling Factor** représente la valeur physique à afficher lorsque la variable machine a pour valeur **1.0**. Dans l'exemple, une valeur 1.0 produirait une dent de scie de 50kHz. On peut donc mettre **Scaling factor** = **50** et **Label** = **Fréquence (kHz)**
- La case **Geometric** permet une incrémentation / décrémentation en progression géométrique lorsqu'on utilise la roue de la souris:

Touche appuyée	Incrémentation Linéaire	Incrémentation Géométrique
Shift	+0.01	+1dB (*1,2589)
(aucune)	+0.001	+0,1dB (*1,0233)
Control	+0.0001	+0,01dB (*1,0023)
Alt	-	½ ton (*1.05946)

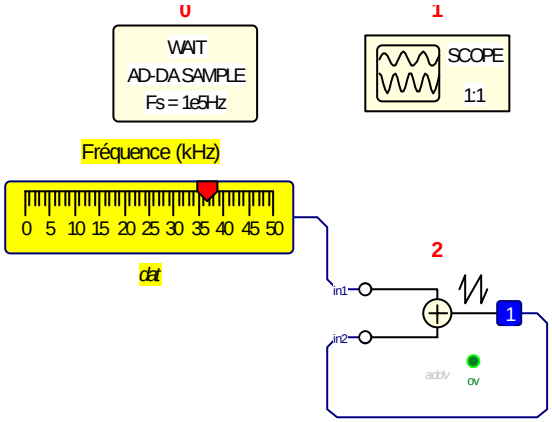
NB: Pour un réglage très grossier, on peut aussi tourner le bouton ou glisser le curseur ou encore changer le paramètre **Value**



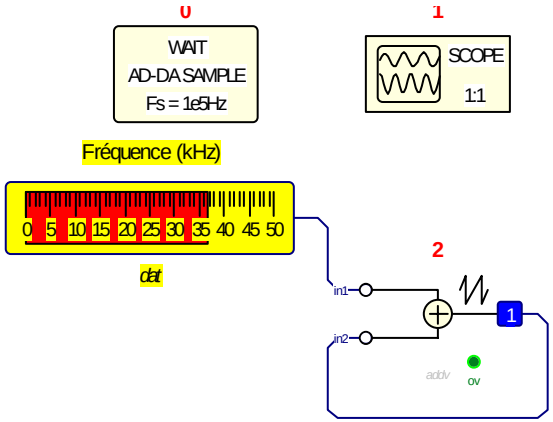
Potentiomètre rotatif,



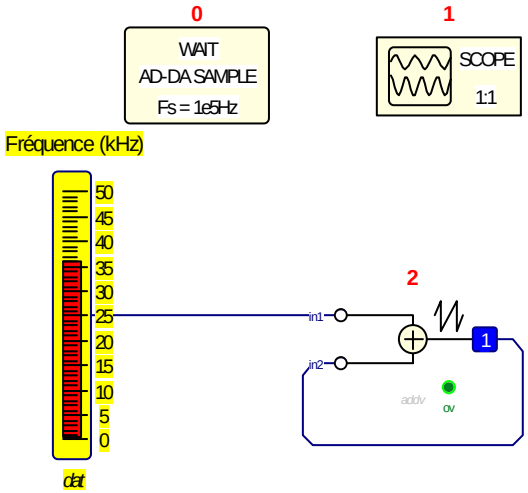
Potentiomètre linéaire vertical,



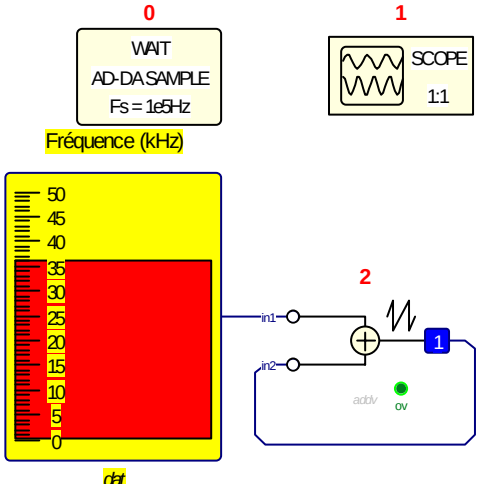
Potentiomètre linéaire horizontal



Barre horizontale



Barre verticale étroite= thermomètre



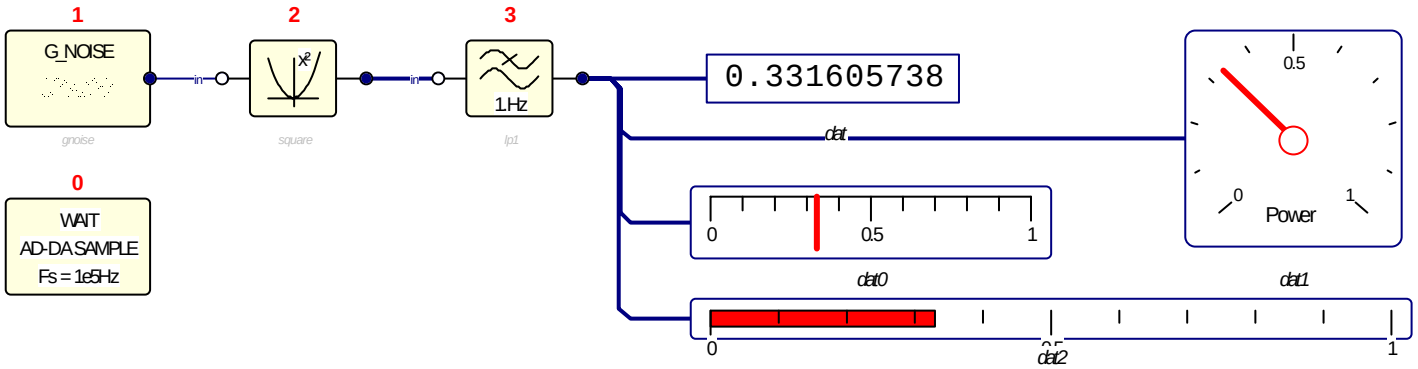
Large = Réservoir

5.3 Sortie affichée en temps réel

Nous avons vu que, lors de l'édition d'un schéma, une donnée connectée sur une sortie permet d'attribuer une valeur initiale à cette sortie. Lors de l'exécution, si la case **RT_update** est cochée, la valeur affichée sera mise à jour en temps réel avec un taux de rafraîchissement de l'ordre de 5Hz permettant de lire les valeurs.
De même que pour les entrées, les options d'affichage permettent diverses représentations:

- **Normal (digital)** On obtient un affichage numérique décimal dont la précision (nombre de chiffres) est déterminée par la largeur du rectangle (idem entrée).
- **Circular** On obtient un "voltmètre" à aiguille à cadran rond
- **Rectangular** (rectangle plus large que haut) On obtient un Vu-mètre à aiguille horizontal
- **Rectangular** (rectangle plus haut que large) On obtient un Vu-mètre à aiguille vertical
- **Bar** (rectangle plus large que haut) On obtient une barre linéaire horizontale (idem entrée)
- **Bar** (Rectangle plus haut que large) On obtient un thermomètre ou un réservoir (idem entrée)

L'exemple ci-dessous estime la puissance moyenne d'un bruit blanc uniforme dans [-1 .. +1[
Les 4 représentations décrivent la même variable



Représentations possibles d'une même valeur fractionnaire

5.4 Affichage interactif des autres types de variables

5.4.1 Type INTEGER

Les Entiers ne s'affichent qu'en mode **Normal** (nombre entier décimal ou hexadécimal)

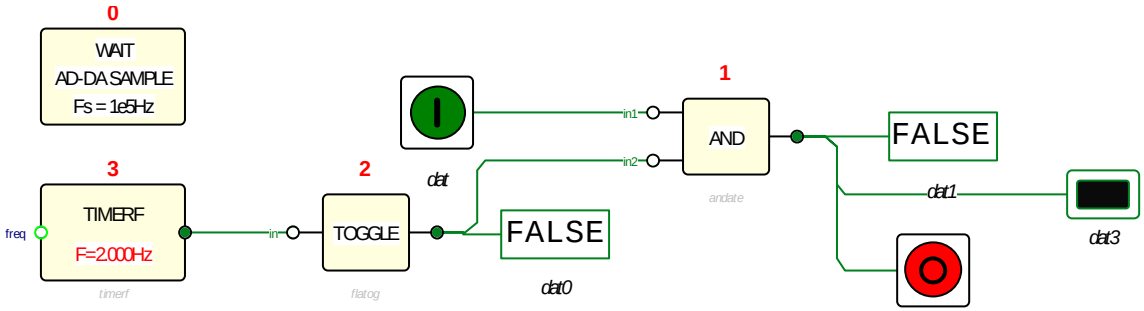
12345

5.4.2 Type BOOL

Les Booléens ont 3 représentations possibles

- **Normal**: TRUE ou FALSE (1 ou 0 acceptés en entrée)
- **Circulaire**: Bouton ON – OFF
- **Rectangulaire ou Bar** : Voyant LED

Le changement d'état d'une entrée s'obtient en tournant la roue de la souris vers l'**avant → TRUE** ou vers l'**arrière → FALSE**



Représentations des Booléens

5.4.3 Type COMPLEX

A l'édition:

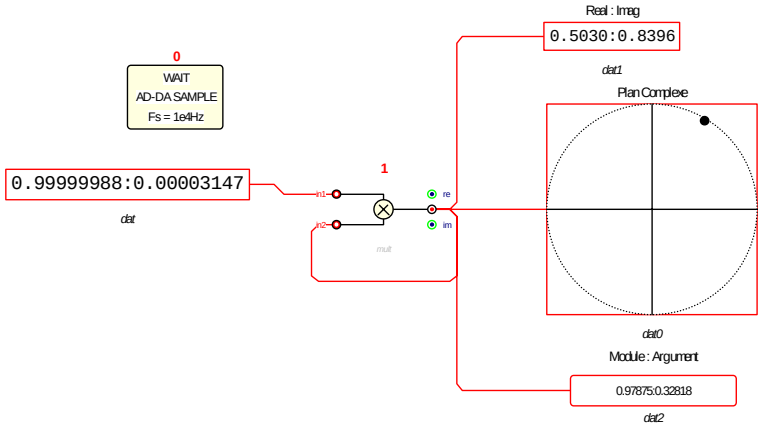
Pour éditer une valeur complexe dans la fenêtre de propriétés, on a 2 choix, cartésien ou polaire avec les syntaxes suivantes:
partie_réelle,partie_imaginaire,c ou **module,argument,p**
NB l'argument est exprimé en **demi-tours**, c'est-à-dire que la valeur 1.0 correspond à pi radians

A l'exécution

En affichage **Normal**, la donnée affiche **partie_réelle:partie_imaginaire**

En affichage **Circulaire** on représente la position du point dans le plan complexe avec son cercle unitaire

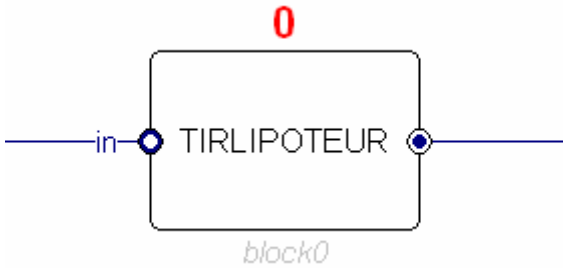
L'affichage **Rectangulaire** est identique à l'affichage normal, sauf que les valeurs affichées sont **module:argument**



Dans tous les modes, La roue de la souris incrémente l'**argument** (ou le module avec touche Alt)

6 Les autres objets graphiques

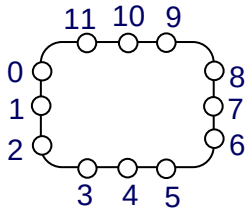
6.1 Les blocs



6.1.1 Nom de fonction, nom d'instance, numéro d'ordre

Les blocs comportent un **nom de fonction**, ici **tirlipoteur** désignant le traitement à effectuer, et un **nom d'instance** ici **block0** permettant de différencier plusieurs blocs de même fonction dans un schéma. Les blocs qui ne peuvent être instanciés qu'une fois (blocs matériels) n'ont pas de nom d'instance. Tous les symboles internes d'un bloc comportent le nom d'instance en tête et séparé par un underscore. L'ordre dans lequel s'exécute un bloc au sein d'un schéma est donné par le nombre apparaissant en rouge au dessus du bloc. On peut le modifier en éditant les propriétés du bloc.

6.1.2 Les bornes d'entrées-sorties



Les blocs possèdent jusqu'à 12 bornes d'entrée ou de sortie. Le code FIBULA associe les numéros 0 à 11 aux positions représentées ci-contre. A chaque borne est associé un type de données représenté par une couleur ci-dessous. Une borne de sortie représente une donnée en mémoire (borne ronde pleine), sauf pour le type Matrice pour lequel il représente un pointeur (borne carrée). Une borne de couleur grise n'a pas de type, cela signifie que le type dépend de la connexion sur cette borne, et donne lieu à une compilation conditionnelle. Une borne bordée de vert est à connexion optionnelle (le code obtenu est dépendant de l'existence ou non d'une connexion à cette borne).

Graphismes utilisés:

- Réel Fractionnaire (24 ou 48 bits)
- Complexe Fractionnaire (2 x 24bits)
- Entier (24 ou 48 bits)
- Booléen (1 bit)
- Chaîne de caractères (n x 8 bits)
- Flottant (48 bits)

Couleurs adoptées lorsque le fond est blanc

6.1.3 Fichier de description .FIB

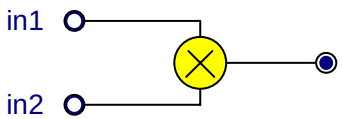
Chaque bloc possède une description sous forme d'un court fichier texte nommé **"<fonction>.fib"** où <fonction> représente le nom unique utilisé pour désigner la fonction. L'ensemble des fichiers de description des blocs de référence réside dans le répertoire BLOCKLIB. Les sous-répertoires de BLOCKLIB contiennent les blocs associés à une version matérielle du boîtier DIDALAB. Les fichiers de description créés par l'utilisateur résident dans le répertoire de travail courant de l'utilisateur. Vous trouverez en Annexe 1, le descriptif des fichiers FIB. Vous pouvez ouvrir le fichier FIB d'un bloc en cliquant sur le bouton **Edit Block**. La fonction d'un bloc peut être décrite de 3 manières: à l'aide d'un schéma FIBULA, à l'aide d'une Macro ou à l'aide d'un segment de code ASM. Lorsque le bloc est décrit par une macro, il doit exister en parallèle un fichier de macro définition "<fonction>.ASM" dans le répertoire LIB.

6.1.4 Les paramètres

La plupart des blocs possèdent des paramètres réglables dans la fenêtre d'édition des propriétés. FIBULA propose en général des valeurs par défaut de ces paramètres afin d'éviter les erreurs dues à un oubli de saisie. Voyez en Annexe 1 la syntaxe de la commande PARAMS du fichier FIB qui décrit la liste des paramètres et de leurs valeurs par défaut. Dans le cas où la fonction du bloc est décrite par une macro, les paramètres correspondent aux valeurs données aux arguments formels de la macro lors de l'extension de celle-ci. Lorsque la fonction du bloc est définie par un schéma ou du code ASM, le n^{ème} paramètre est représenté par "%n" dans le schéma ou la ligne assembleur. Lors de l'édition des propriétés du bloc, lorsque l'on donne des valeurs aux paramètres, ces valeurs se subtituent aux chaînes "%n".

6.1.5 Le graphisme

```
INPUT
in1,FRACT,0
INPUT
in2,FRACT,2
OUTPUT
,FRACT,7
MACRO
mul
DESCRIPTION
Real multiplier
                    Blanc obligatoire ici
GRAPH
mp,0                move to pin 0
ht,0.5              Horizontal line to 0.5
vt,0.75             Vertical line to 0.75
mp,7                Move to pin 7
ht,0.5              Horizontal line to 0.5
mp,2                Move to pin 2
ht,0.5              Horizontal line to 0.5
ci,0.5,0.5,0.15     Circle centre 0.5,0.5, radius 0.15
x,0.5,0.5,0.07      Sign "x" at pos 0.5,0.5 radius 0.07
```



Si le fichier .FIB de description du bloc contient la ligne "GRAPH", alors, les lignes suivantes décrivent le graphisme utilisé pour représenter le bloc à l'aide de quelques lignes d'un langage texte simple et compact. Sinon, par défaut, le bloc est représenté par un rectangle à coins arrondis, portant le nom de fonction au centre. Reportez-vous à l'Annexe 2 pour le descriptif des commandes graphiques.

Ci-contre le fichier descripteur MUL.FIB du bloc multiplieur
En caractères verts, les explications des commandes graphiques

6.2 Les listes de définitions

Lorsque l'on crée de grands schémas de simulation, il arrive fréquemment que l'on doive fournir un grand nombre de données en tête d'un programme. Dans ce cas, si l'on utilise une multitude de symboles "données" isolés, le plan devient rapidement illisible. D'autre part, il arrive fréquemment qu'une donnée soit définie à l'aide d'une autre, ce qui impose un ordre d'apparition des définitions (dépendances). D'où l'intérêt d'une liste de définitions.

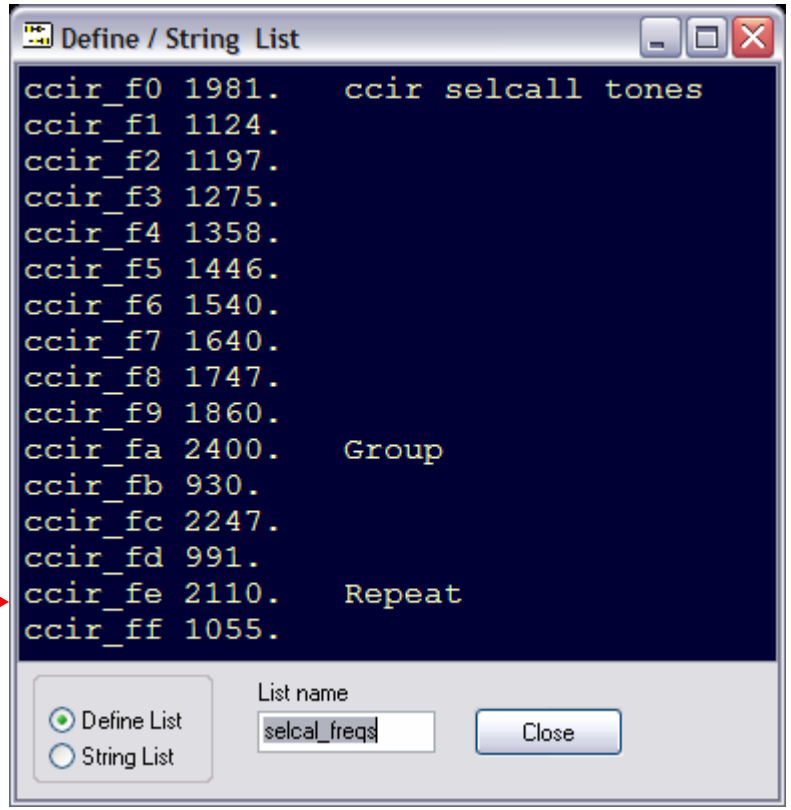
On crée une liste de définitions
en activant ce bouton, puis en cliquant sur le plan .

La liste apparaît ainsi sur le plan.
En double-cliquant sur ce symbole, on ouvre l'éditeur de liste de chaînes dans laquelle chaque ligne que l'on écrit constitue une définition. Syntaxe d'une ligne:

nom valeur commentaire

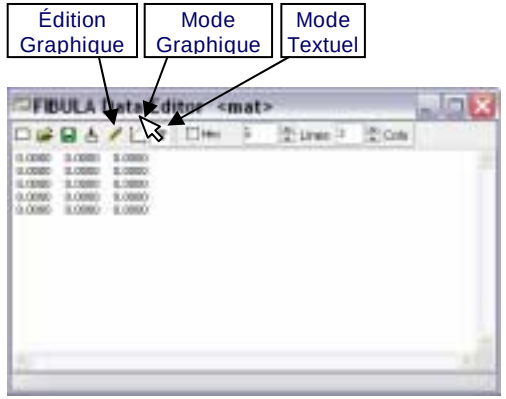
La liste de définitions s'insère dans le descriptif du schéma lorsque l'on ferme l'éditeur

Exemple:
Déclaration des fréquences de signalisation pour talkie-walkie
Les symboles ccir_f0 à ccir_ff pourront être utilisés par exemple pour remplir les cases d'un tableau



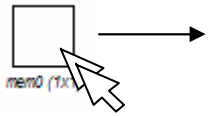
6.3 Les tableaux et matrices

On crée un tableau ou une matrice en activant ce bouton , puis en cliquant sur le plan, à l'endroit désiré.



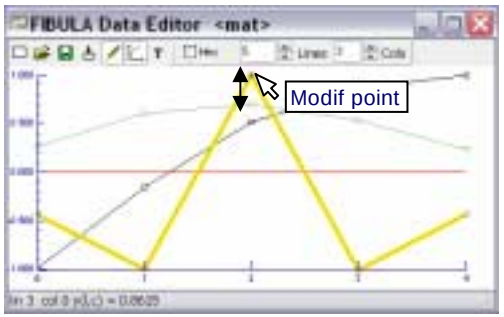
Par défaut, le type des données est FRACT.
Vous pouvez choisir BOOL*, INTEGER, ou COMPLEX, ou STRING en cliquant avec le bouton droit pour définir les propriétés de la matrice.

Double-cliquez sur l'objet matrice pour saisir ses valeurs.



Dans l'éditeur de données, réglez le nombre de lignes et de colonnes.
Vous pouvez maintenant remplacer les zéros par des nombres
Vous pouvez saisir / observer ces nombres en hexadécimal 24 bits soit 6 chiffres

*** Note: Les matrices booléennes utilisent en mémoire 1 mot de 24 bits par ligne. Le nombre maximal de colonnes est donc 24.**



Pour le type FRACT uniquement vous pouvez visualiser, saisir ou modifier les valeurs **graphiquement**. En mode graphique, chaque colonne de la matrice représente un signal interpolé de (lignes) points dans lequel le n° de ligne représente l'abscisse.
Saisissez et déplacez verticalement les petits carrés qui représentent les points pour modifier les valeurs correspondantes.
Les modifications sont **interactives**: Lorsque le programme s'exécute, la valeur modifiée à la souris est répercutée en temps réel.

Lorsque toutes les valeurs sont saisies, mettez à jour le tableau ou matrice de votre programme à l'aide de ce bouton:
Les valeurs du tableau seront mémorisées dans le fichier FIB de votre programme
Vous pouvez zoomer / déplacer l'image à l'aide de la molette de la souris:

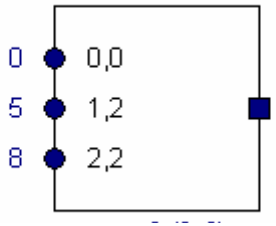
- Pointeur sur l'axe des ordonnées ⇔ zoom vertical, échelle auto adaptative
- Pointeur sur l'axe des abscisses ⇔ zoom horizontal
- Pointeur à droite ⇔ déplacement horizontal
- Pointeur en haut ⇔ déplacement vertical

Le symbole graphique d'un tableau est un rectangle sans connexions dont les dimensions relatives évoquent le rapport entre lignes et colonnes. Ainsi, une matrice carrée (lignes = colonnes) apparaît carrée.
Un tableau ou matrice représente un espace physique en mémoire, il peut être variable ou constant.

6.3.1 Propriétés du tableau

Cliquez avec le bouton droit pour éditer les propriétés du tableau.
Le tableau peut être attribué à l'un ou l'autre cœur en exclusivité, ou résider en mémoire partagée.
Il peut appartenir aux champs mémoire X:, Y:, L:, ou P: (à noter: si le tableau est partagé, le champ ne peut être que X: ou Y:).
Les options Modulo et Reverse Carry forcent l'adresse de base à respecter les contraintes de l'adressage cyclique et de l'adressage binaire en miroir.

6.3.2 Connexions à un tableau ou matrice:



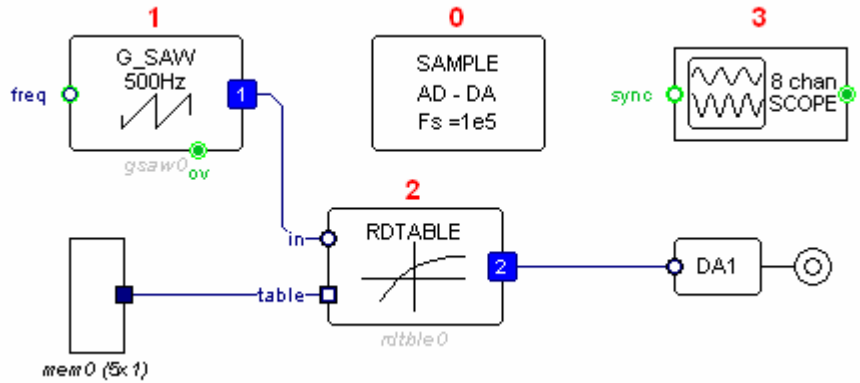
Matrice fractionnaire 3x3 avec borne d'accès global et 3 bornes d'accès aux éléments particuliers 0,0 1,2 2,2

Pour accéder au tableau, on doit y ajouter des connexions. Une connexion avec l'attribut ARRAY constitue un pointeur sur le tableau global (borne carrée).
Une connexion sans l'attribut ARRAY permet d'accéder à un élément particulier du tableau.
Le nom de la borne est alors un nombre décimal qui représente la position physique de l'élément, soit la valeur:

(ligne-1) x nb_colonnes + colonne

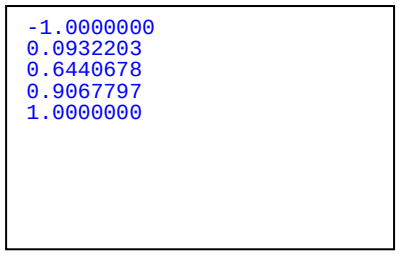
Voyez les démos des blocs COPY (onglet Arithmetic) et du bloc MATMUL (onglet Matrix) pour voir comment utiliser les matrices.

6.3.3 Lecture de tableau à une colonne

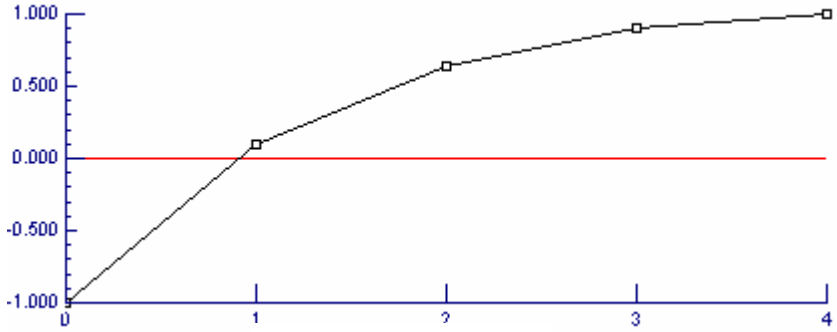


Lorsque l'on désire créer une fonction arbitraire, le plus simple est d'interpoler une table de points. Pour créer une table sur le plan, déposez l'objet matrice, et éditez ses propriétés:
Data type = FRACT
Lines = 5
Columns = 1
Initialized data
Puis cliquez sur Edit data.
Dans une nouvelle page, vous obtenez une colonne de 5 valeurs, toutes égales à 0. Passez en mode édition de courbe bouton →

Double-cliquez sur la courbe pour faire apparaître les points. Vous pouvez glisser les points verticalement à l'aide de la souris pour obtenir la courbe désirée. Vous pouvez voir les valeurs numériques à l'aide du bouton "T" (mode texte). Sauvez la courbe dans un fichier avec l'extension ".dat" dans votre répertoire de travail qui contient aussi votre programme sauvé. Terminez la saisie des propriétés du tableau en fournissant le nom du fichier .dat

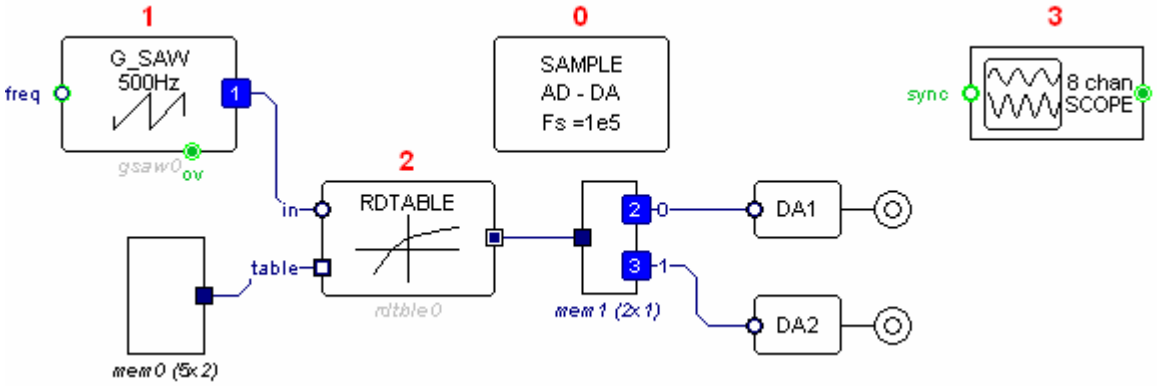


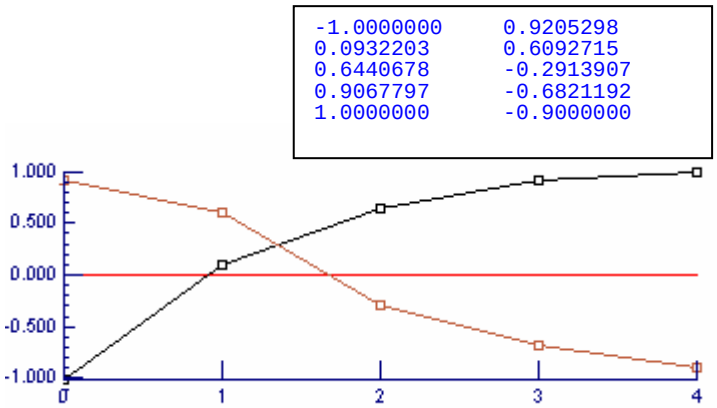
Mode texte



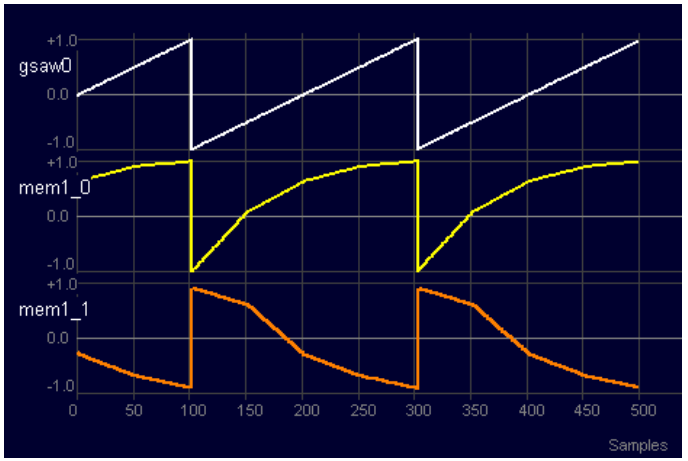
Mode graphique

6.3.4 Lecture de tableau à N colonnes





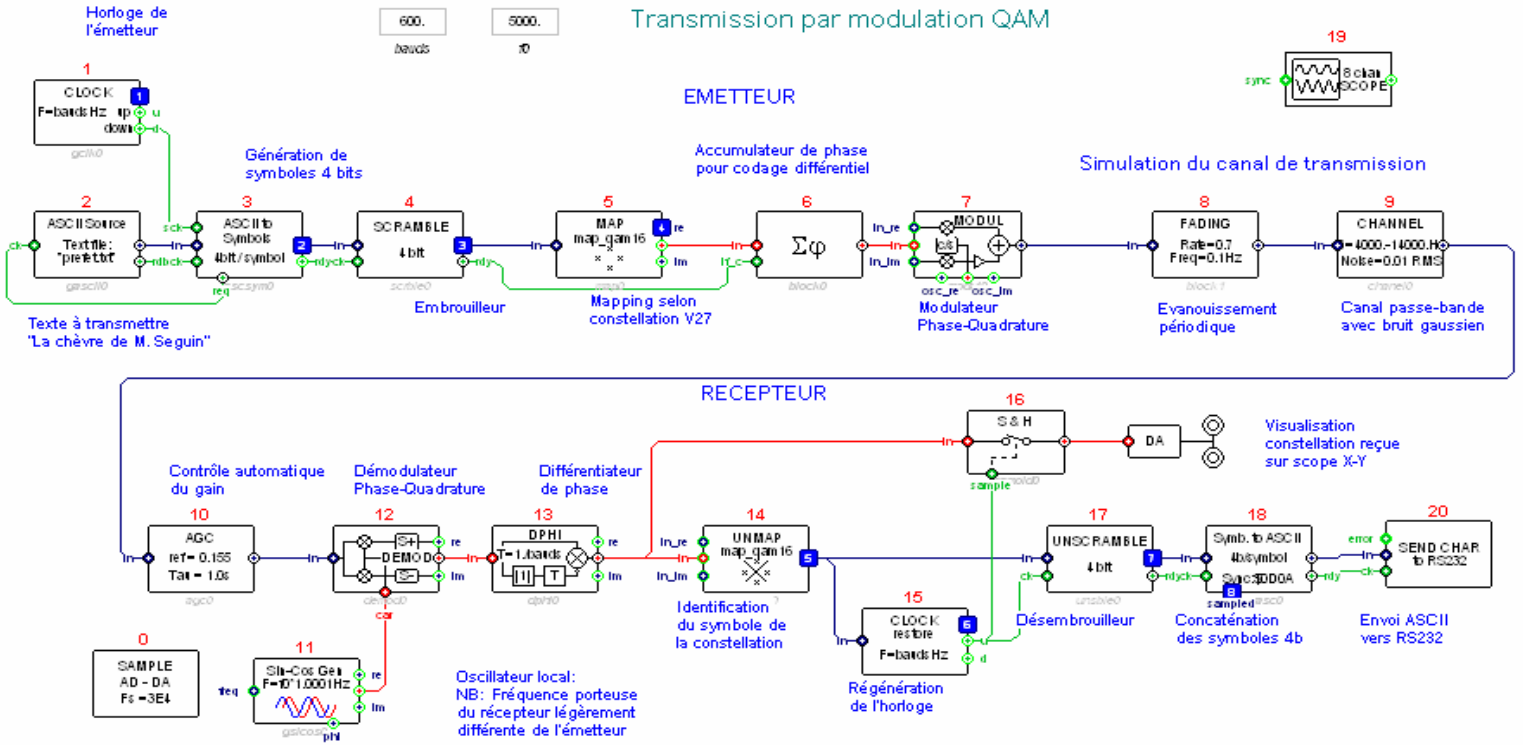
Procédez de la même manière que précédemment, mais en éditant une table de données à N colonnes. Dans ce cas, éditez la borne de sortie de RDTABLE pour lui donner l'attribut ARRAY. Connectez cette borne à un vecteur de dimension N avec des bornes.

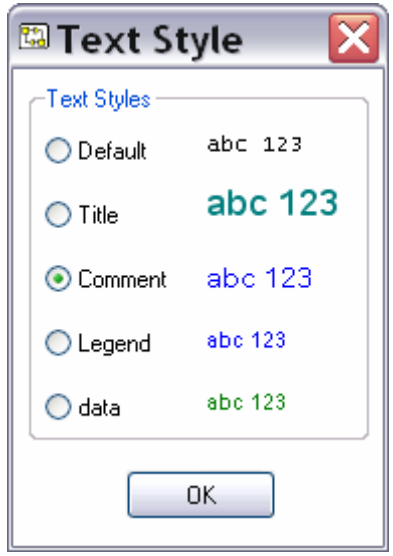


6.4 Les commentaires

L'expérience montre combien les commentaires sont utiles dans un programme !
Dans un schéma, on a la liberté de placer le commentaire près des objets concernés.
Pour déposer un commentaire sur le schéma, activez ce bouton: 

Puis cliquez à l'endroit désiré et saisissez votre texte. Cliquez en dehors de la fenêtre de saisie lorsque vous avez fini. Ensuite cliquez à droite sur le texte obtenu pour choisir son style (ci-contre).





Les styles sont programmables dans le menu Settings | Fonts de Fibula.

7 Contrôle de l'exécution, Synchronisation, Interruptions

7.1 Les variables booléennes

Les variables booléennes (bornes et connexions vertes) ne peuvent prendre que 2 états TRUE et FALSE. Elles n'occupent qu'un seul bit en mémoire. Elles sont utilisées pour la synchronisation, le contrôle de l'exécution, ou pour des fonctions logiques.

Propriété MESSAGE

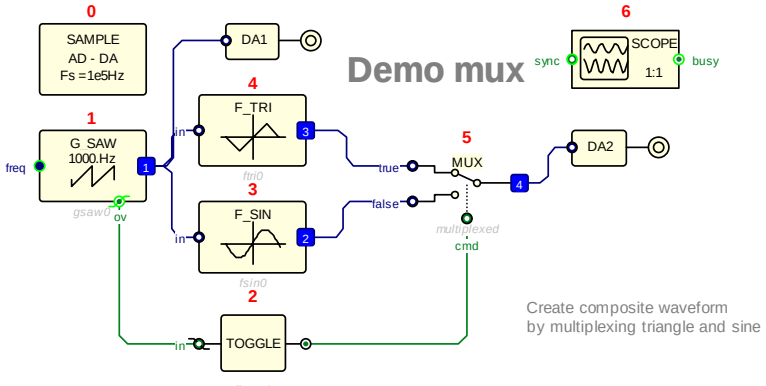
Les bornes booléennes ayant la propriété MESSAGE sont dédiées aux fonctions de **synchronisation**. Si la case "MESSAGE" est cochée dans les propriétés de la borne booléenne, alors celle-ci apparaît avec un front montant (sortie) ou descendant (entrée).

Bornes ayant la propriété MESSAGE

Cela signifie que la sortie ne réalise **que** mise à l'état TRUE (=1) lorsqu'un événement interne survient, alors que l'entrée qui y est connectée réalise, lors de sa lecture, la remise à l'état FALSE (=0).
Cette remise à 0 se fait en une seule instruction insécable de type Test-And-Clear.
Une sortie Message peut être connectée à plusieurs entrées booléennes, dont une au moins a la propriété Message. Ce dispositif est économique en temps machine et permet au bloc émetteur d'avoir un accusé de réception du message. Il permet également de mettre en œuvre des sémaphores MUTEX dans une application multi-cœur.

Les sorties booléennes "normales" sont mises à jour (TRUE ou FALSE) à chaque échantillon;
Les entrées booléennes "normales" ne font que lire l'état sans le modifier.

Exemple1:



La démo du bloc MUX (onglet Control) utilise la sortie booléenne optionnelle **ov** du bloc G_SAW avec la propriété MESSAGE pour faire basculer un flip-flop à chaque discontinuité de la dent de scie. L'entrée du flip-flop remet à 0 la variable booléenne gsaw0_ov. En revanche, la sortie booléenne du flip-flop est de type "normal", car son état statique détermine la position du commutateur MUX.

7.2 Les bornes de contrôle

On peut rajouter aux blocs exécutables des **bornes de contrôle** qui agissent sur l'exécution du bloc. Celles-ci n'existent pas par défaut et sont à **rajouter à la main** à l'aide de ce bouton:  lors de l'édition du schéma.

Borne:	*Type*	*E/S*	*Usage*
if	BOOL	entrée	exécution du bloc si TRUE
reset	BOOL	entrée	provoque la réinitialisation des variables internes du bloc (certains blocs seulement)
loop	INTEGER	entrée	exécution du bloc N fois
done	BOOL	sortie	indique que le bloc s'est exécuté (associé à **if**)

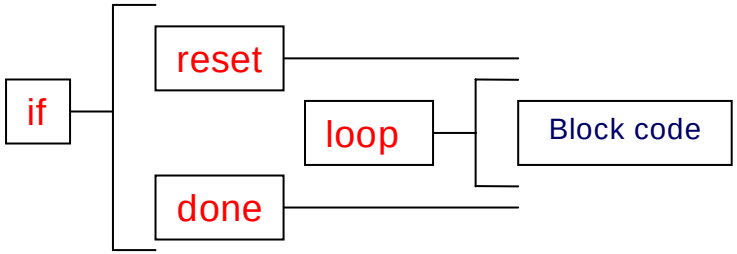
Portée et priorité des bornes de contrôle:

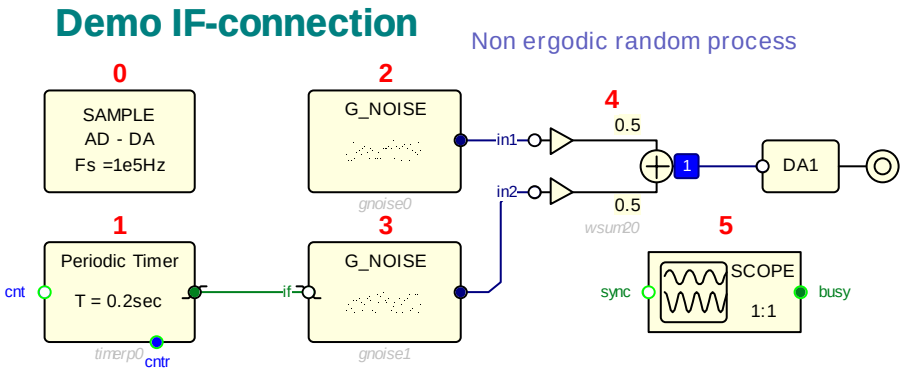
if s'applique à l'ensemble

loop encadre le code du bloc

reset est exécuté avant loop et bloc code

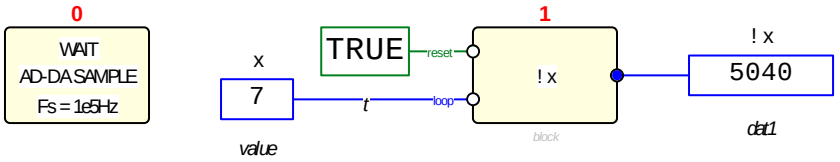
done est validé après bloc code et endloop





L'exemple ci-contre montre comment réaliser un processus pseudo aléatoire non ergodique. Le générateur pseudo aléatoire uniforme **g_noise** y est utilisé deux fois. Le bloc 1 est exécuté à chaque échantillon, tandis que le bloc 3 ne s'exécute qu'une fois chaque 0,2s grâce à une **entrée de contrôle if** connectée à un timer.
On obtient ainsi un signal dont la moyenne à court terme est différente de la moyenne à long terme.
Si on considère chaque période de 0,2s comme une épreuve du processus, la moyenne statistique (=0) est différente de la moyenne temporelle qui est une valeur aléatoire entre -0,5 et +0,5.

Exemple de l'utilisation de **reset** et **loop**



Fonction Factorielle(x)

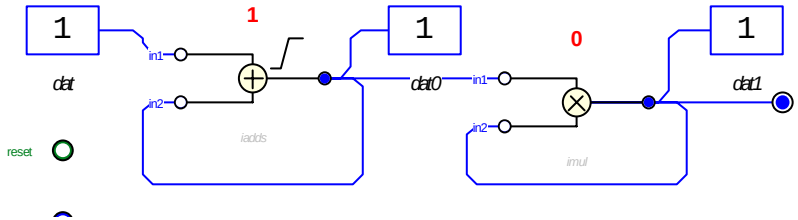


Schéma interne

Pour obtenir la fonction **factorielle(x)**, on initialise le bloc en appliquant la constante booléenne TRUE sur l'entrée de contrôle **reset**, (cela provoque l'initialisation des sorties de l'additionneur et du multiplieur à 1), puis on exécute x fois le schéma interne du bloc à l'aide de la borne de contrôle **loop.**

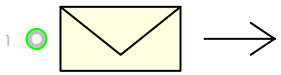
NB Le temps d'exécution de ce bloc pour x=7 n'est que de 130 cycles, à comparer avec la version récursive écrite en C dont le code compilé pour core DSP56300 prend 7336 cycles !

7.3 La synchronisation des cœurs

La machine DSP56720 comporte 2 cœurs CPU 56300, chacun possédant en propre 8kw de ram P, 36kw de ram X et 48kw de ram Y
A cela s'ajoutent 64 kw de ram partagée qui apparaît dans les 3 champs et est donc accessible depuis chacun des 2 cœurs.
(Taille des mots Word=24 bits, DWord=48 bits)

L'onglet **Interrupt** du catalogue comporte des fonctions de **communication inter cœurs**. Il y a 3 niveaux de priorité:

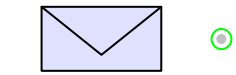
1
Post Message



to other core

iccost

0
Read Message

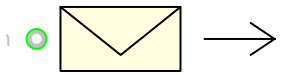


from other core

iccead

Émission asynchrone, réception asynchrone
Le facteur dépose une lettre, vous allez regarder de temps en temps dans votre boîte aux lettres

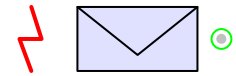
2
Send IRQ



to other core

iccirq

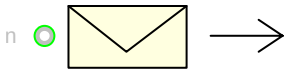
Interrupt
IRQ on Message



from other core

Interruption masquable en réception
Le facteur dépose une lettre, puis sonne pour vous prévenir
Si vous n'êtes pas au téléphone, vous allez chercher la lettre.

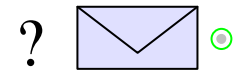
3
Send IRQ



to other core

iccirq0

1
Wait for IRQ

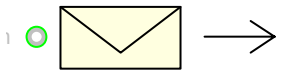


from other core

iccirq1

Requête d'interruption masquée, boucle d'attente du drapeau
Vous attendez la lettre du facteur, vous ne le laissez pas sonner

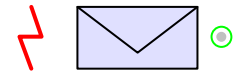
0
Send NMI



to other core

iccnmi

Interrupt
NMI on Message



from other core

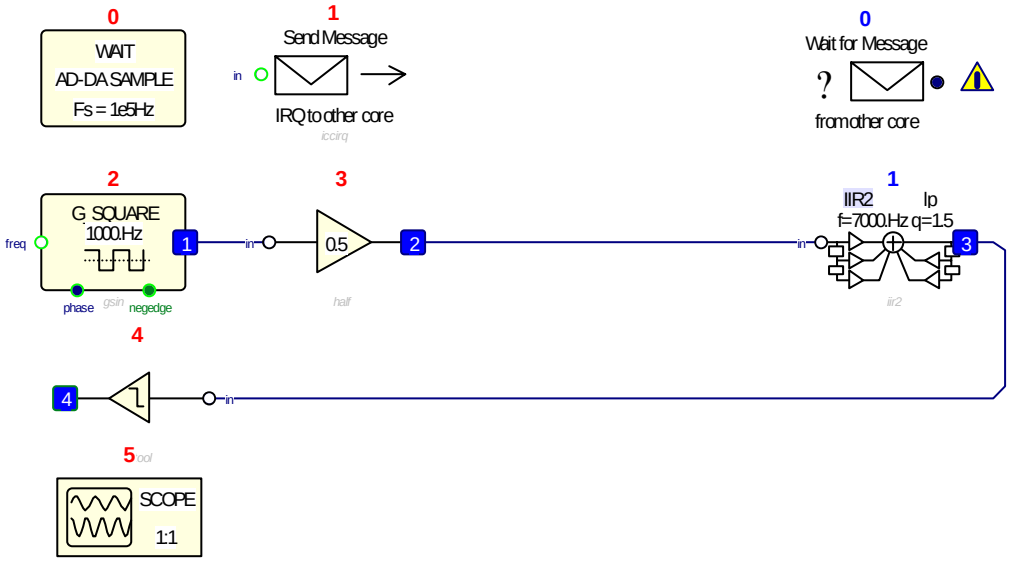
Interruption non masquable
Le facteur vous apporte un recommandé, il sonne,
vous devez signer immédiatement

Une application peut gérer les cœurs de différentes manières:

a) Le cœur 0 tourne seul, le cœur 1 est au repos (cas de la majorité des démos)
b) Une application tourne sur cœur 0, une autre, indépendante sur le cœur 1 (exemple: émission et réception d'un MODEM)
c) Les deux cœurs sont synchrones et partagent la même fréquence d'échantillonnage
d) Les 2 cœurs échangent des signaux de fréquences d'échantillonnages différentes (par exemple 48kHz et 41kHz)
e) Le cœur 1 est esclave du cœur 0 (ou vis versa)

Les problèmes de synchronisation ne concernent que les cas c, d, e

7.3.1 Cas de 2 cœurs ayant la même fréquence d'échantillonnage:



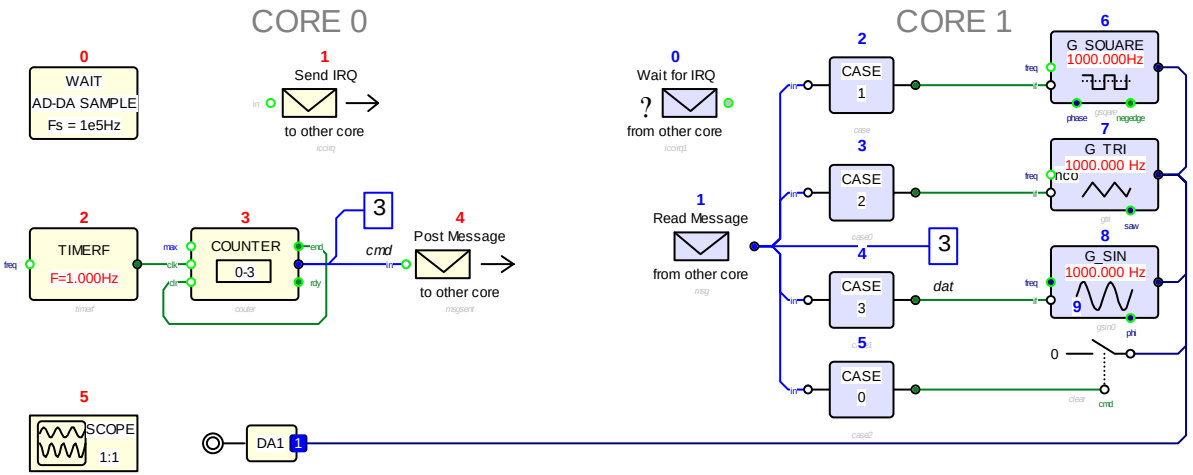
Boucle d'attente Mainloop dans chaque cœur
Le bloc **icc_snd_irq** en position 1 du cœur 0 envoie un message à l'autre cœur ce qui provoque l'interruption qui débloque la boucle d'attente du cœur 1 placée en position 0
Les instants d'échantillonnage sont quasi simultanés entre les deux cœurs. On peut donc passer des signaux d'un cœur à l'autre sans provoquer d'artéfact. Une connexion entre 2 blocs appartenant à des cœurs différents génère une variable dans la ram partagée.

7.3.2 Cas où les cœurs échangent des signaux échantillonnés à des fréquences différentes et de rapport non entier

Dans ce cas on utilisera l'**ASRC** (Asynchronous Sample Rate Converter) qui est un périphérique matériel partagé par les 2 cœurs.
Ce module matériel étant conçu pour mixer diverses sources audio en provenance des ports ESAI et SPDIF, et étant par ailleurs assez complexe à programmer, la présente version de FIBULA ne contient encore aucun bloc ASRC.

7.3.3 Cas où le cœur 1 est esclave du cœur 0 (exemple)

Toutes sortes de stratégies peuvent être imaginées à l'aide des messages entre cœurs C0 et C1



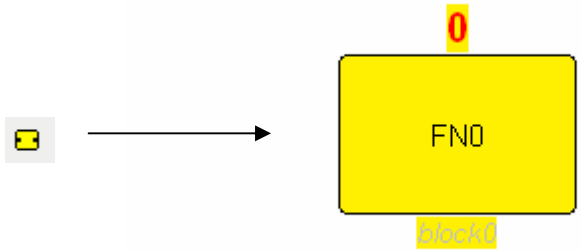
Dans l'exemple ci-contre, la fréquence d'échantillonnage de C0 est transmise à C1 par Send_IRQ → Wait_for_IRQ. La valeur du compteur dans C0 est transmise à C1 par Post_Message → Read_Message. Cette transaction est synchrone du fait que Post_Message comme Read_Message sont dans des boucles échantillonnées synchrones.

Le décodage dans C1 du message posté par C0 se fait à l'aide de conditions CASE qui activent une parmi 4 fonctions à l'aide de bornes **if**. Les 3 blocs générateurs et la mise à 0 ont leurs sorties connectées en OU câblé, ce qui est autorisé à partir de FIBULA V3
La sortie du générateur constitué par C1 est rapatriée vers le convertisseur DA1 de C0 et crée une variable partagée.

Exemple de synchronisation: générateur dans C1 esclave de C0

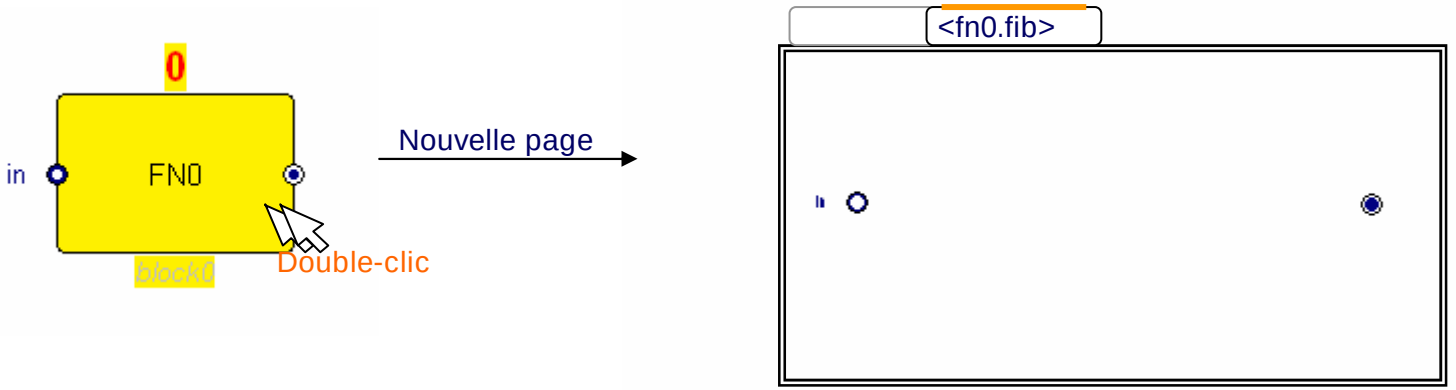
8 Création d'un nouveau bloc

8.1 Bloc à définir soi-même par un schéma



Activer le bouton "bloc vide"
Clic gauche sur le plan, à l'endroit désiré
Ajouter les bornes:

- Activer le bouton Borne
- Clic G sur le bord du bloc (12 positions possibles)
- Editer la borne en lui donnant un type et un nom
- en général, une entrée s'appelle "in"; une sortie unique n'a pas de nom (chaîne nulle)



Double cliquer sur le bloc: une nouvelle page représente le schéma interne du bloc, avec ses bornes d'entrée/sortie
Dessiner le schéma interne en ajoutant des blocs, et des connexions, dont les connexions des bornes d'entrées et de sorties.
Si le bloc doit posséder des paramètres, alors:
Dans le schéma interne, si certaines valeurs de paramètres dépendent du paramètre n du bloc parent, utiliser la notation %n

Mettre le schéma interne en mode textuel, et ajouter la commande:

PARAMS

name{,<Descr. Param1>,<nb de valeurs proposées>,<valeur1>,<valeur2>,.. <valeur k>}

où "name" indique que le bloc possède un nom et peut être instancié plusieurs fois et la partie entre { } est à écrire pour chacun des paramètres

Sauver ce schéma en lui donnant un nom différent des blocs déjà existants. Fermer la page de ce schéma.

8.2 Bloc à définir soi-même par du code assembleur*

Activer le bouton "bloc vide"
Clic gauche sur le plan, à l'endroit désiré
Ajouter les bornes:

Activer le bouton Borne
Clic G sur le bord du bloc (12 positions possibles)
Editer la borne en lui donnant un nom -- en général, une entrée s'appelle "in"; une sortie unique n'a pas de nom (chaîne nulle)

Clic D sur le bloc: la fenêtre d'édition du bloc s'ouvre.
Dans Block Design, Block is … Sélectionner **Assembler**
Activer le bouton Edit Block
Après la ligne **ASM**, insérez les lignes de du code assembleur en utilisant le symbole § pour représenter le nom du bloc. Ainsi, par exemple, l'entrée **in** du bloc sera évoquée par **§_in** dans le code ASM. Les paramètres seront appelés %0, %1, etc. …
Insérez une ligne blanche pour terminer le code ASM
S'il y a des paramètres, ajoutez la commande:

PARAMS
name{,<Descr. Param1>:,<nb de valeurs proposées>,<valeur1>,<valeur2>,.. <valeur k>}

Sauvez le fichier sous <nom de fonction>.fib

NB Technique déconseillée parce que non portable sur une autre machine cible; préférez la macro.

8.3 Bloc à définir soi-même par une macro

Activer le bouton "bloc vide"
Clic gauche sur le plan, à l'endroit désiré
Ajouter les bornes:

Activer le bouton Borne
Clic G sur le bord du bloc (12 positions possibles)
Editer la borne en lui donnant un nom -- en général, une entrée s'appelle "in"; une sortie unique n'a pas de nom (chaîne nulle)

Clic D sur le bloc: la fenêtre d'édition du bloc s'ouvre.

Dans Block Design, Block is … Sélectionner **Macro**
Activer le bouton Edit Block
S'il y a des paramètres, ajoutez la commande:

PARAMS
name{,<Descr. Param1>,<nb de valeurs proposées>,<valeur1>,<valeur2>,.. <valeur k>}

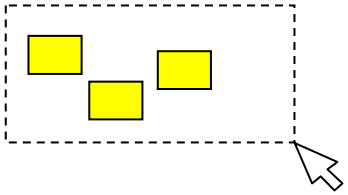
Sauvez le fichier sous <nom de fonction>.fib
Si la macro n'existe pas encore, activer le bouton **Edit Macro** éditez et sauvez la nouvelle macro.

8.4 Modification d'un bloc existant

Les blocs dont les fichiers .FIB et .ASM ne sont pas protégés en écriture peuvent être modifiés par l'utilisateur.
Toutefois, il est fortement conseillé de ne pas modifier les blocs de référence de la bibliothèque Fibula, car certains programmes pourraient alors ne plus fonctionner. **V3 Par sécurité, la sauvegarde de blocs de référence modifiés est dirigée vers les répertoires de l'utilisateur.**
IMPORTANT: Lorsque vous modifiez un bloc dans un schéma, la modification n'est prise en compte qu'après avoir rechargé le schéma. Pour cela, appuyez successivement sur les boutons mode texte puis mode schéma.

8.5 Édition Graphique à l'aide de la souris

8.5.1 Sélectionner



- 1 objet: Clic sur l'objet
- Groupe d'objets: Clic sur chaque objet, touche SHIFT appuyée ou sélection d'un rectangle (clic G + glisser)
- Tout: double clic sur un espace vide du plan

8.5.2 Déplacer

- Objets sélectionnés: Bouton G appué sur un objet sélectionné, déplacer souris
- Le plan: Bouton D appuyé sur un espace vide, déplacer souris

8.5.3 Zoomer

Utiliser la molette de la souris

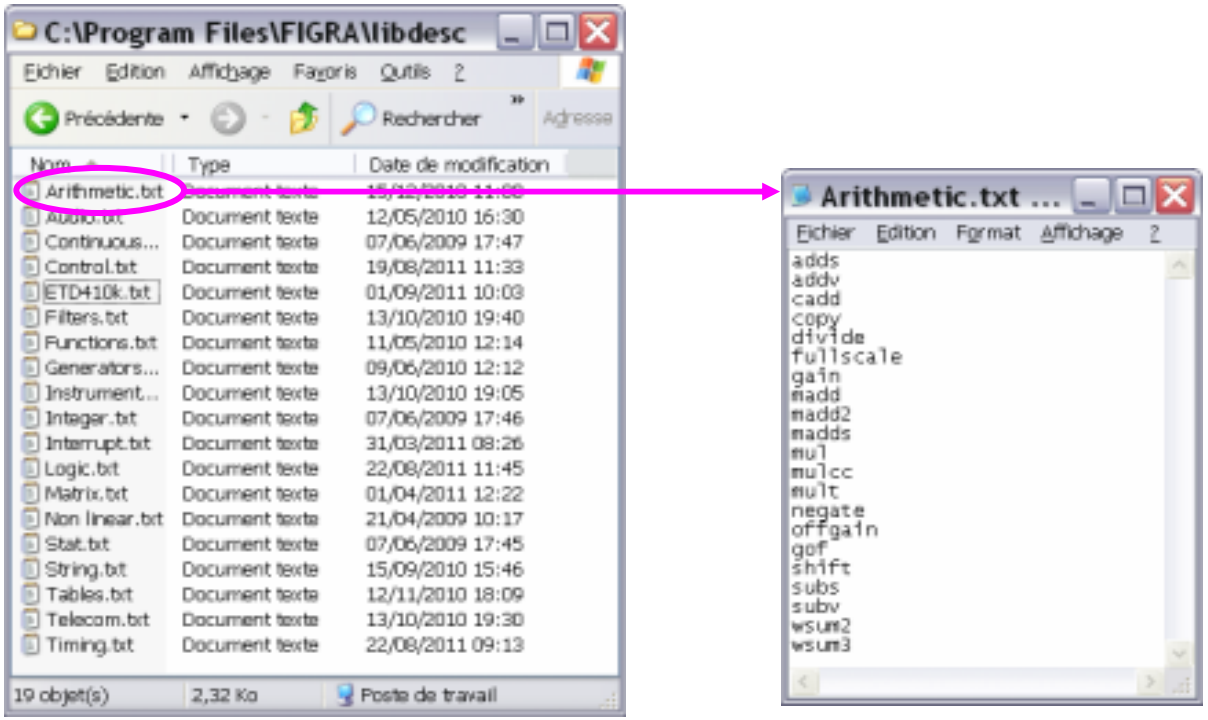
8.5.4 Redimensionner un objet (matrices, données)

Cliquer sur un des 4 angles et glisser

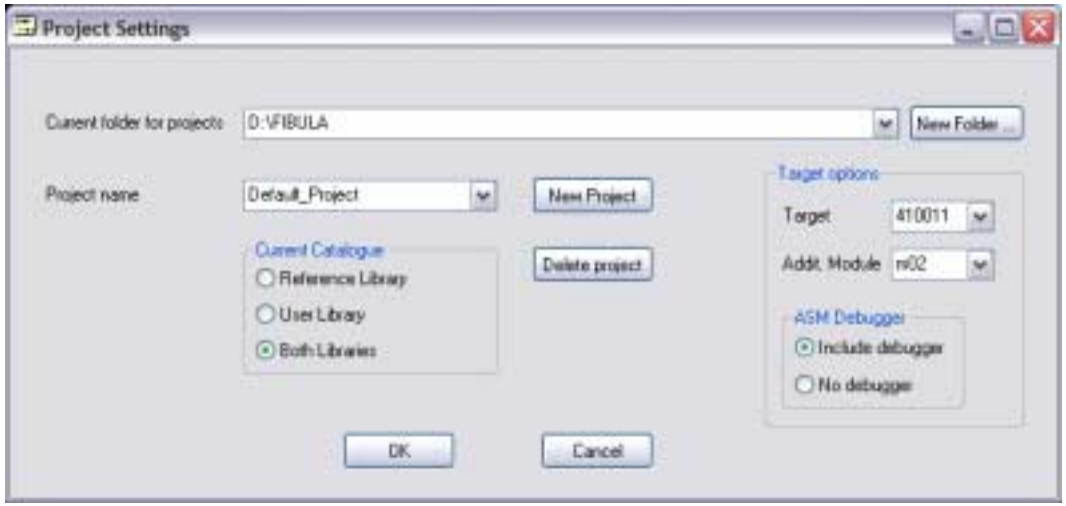
9 Catalogue et organisation des projets

9.1 Le catalogue

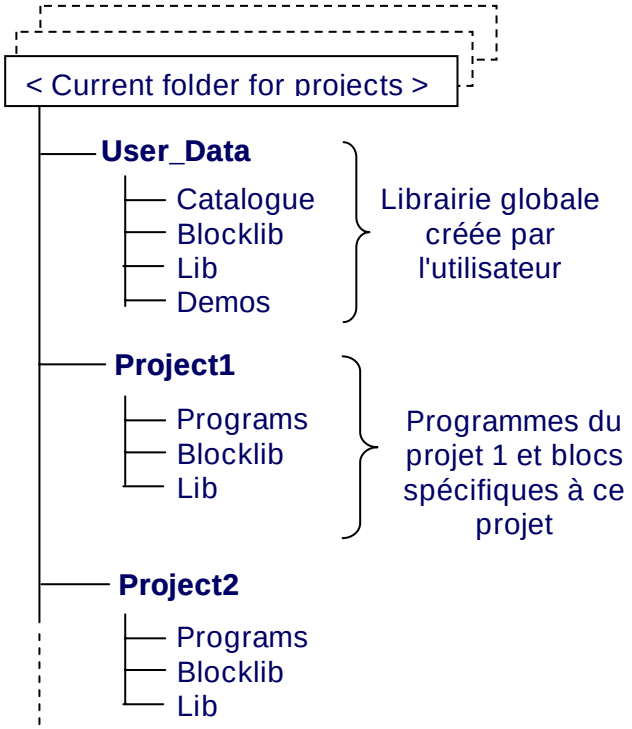
L'ensemble des blocs que l'on désire voir dans le catalogue est décrit (par défaut) par le répertoire **FIGRA \ Libdesc**. Dans ce répertoire, chaque fichier texte représente un onglet du catalogue. Le nom du fichier constitue le titre de l'onglet, et chaque ligne du fichier désigne un bloc apparaissant sous cet onglet. L'onglet **Prjct blocks** rassemble les blocs créés pour le projet en cours. L'onglet **Prjct Pgms** contient les programmes du projet en cours.
Astuces du catalogue:
Lorsqu'un bloc de la liste du fichier txt n'existe pas pour la cible en cours, son icône n'apparaît pas dans le catalogue.
Lorsque les **deux premiers éléments** de la liste sont introuvables, l'onglet correspondant à cette liste disparaît du catalogue. Cela permet de cacher les rubriques vides.



9.2 Le projet



Fenêtre du projet et organisation des fichiers de l'utilisateur



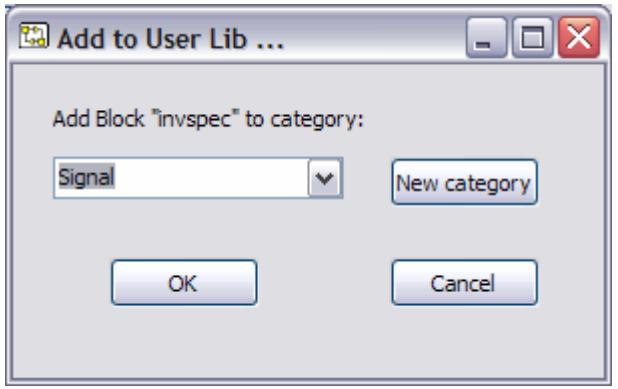
Le répertoire **Current folder for projects** contient tous les projets ainsi que la librairie créée par l'utilisateur. Son chemin par défaut est **Mes documents\FIBULA**. Si vous conservez ce chemin, cela signifie que ces données sont personnelles. Mais vous pouvez changer ce répertoire à l'aide de **New Folder** et choisir par exemple un répertoire partagé sur le réseau. En ajoutant des répertoires, vous pouvez définir plusieurs groupes de projets. Mais seul le groupe sélectionné sera visible.

Chaque projet possède un **nom** qui définit le répertoire où sont rangés 3 sous répertoires: **Programs** contient les applications et programmes de tests du projet; **Blocklib** contient les descripteurs graphiques de chaque nouveau bloc, et **Lib** contient (le cas échéant) les macros écrites en assembleur pour certains blocs de bas niveau.

Lorsque vous créez un nouveau bloc comme indiqué chapitre 8, ce bloc est dans un premier temps local à votre projet en cours. Mais vous pouvez ensuite l'ajouter à votre librairie globale **User Library**. Il sera alors accessible dans le catalogue depuis n'importe quel projet. Le répertoire **User_Data** contient la librairie globale utilisateur avec son **Catalogue**, ses descripteurs graphiques dans **Blocklib** et ses macros dans **Lib**. Pour chaque nouveau bloc, vous devez écrire une démonstration dans **Demos** (la plus simple possible) qui

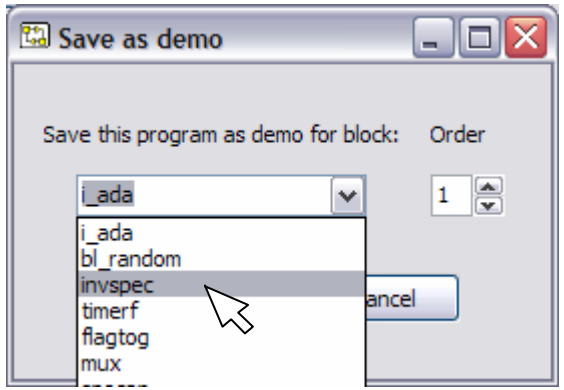
démontre la fonctionnalité du bloc. Si le bloc, en fonction de ses paramètres, possède plusieurs modes de fonctionnement, écrivez une démo pour chaque mode.
Le choix **Current Catalogue** vous permet de définir le contenu du catalogue:
Blocs de référence uniquement, blocs utilisateur uniquement, ou fusion des deux librairies. Si pour un nouveau bloc, vous avez choisi une catégorie existante de la librairie de référence (par exemple **Arithmetic**), et si vous avez choisi **Both libraries**, le nouveau bloc apparaîtra dans le catalogue comme s'il faisait partie de la référence. Si sa catégorie est nouvelle, le catalogue possèdera un nouvel onglet.

9.2.1 Ajout d'un nouveau bloc à la librairie



Pour ajouter un nouveau bloc à la librairie globale de l'utilisateur, vous devez l'avoir déjà testé et sauvé dans la librairie locale du projet en cours. Sélectionnez le bloc à sauver dans le schéma de la page courante, puis faites la commande:
File | Save selected block to user library
Choisissez une catégorie (rubrique de rangement du catalogue) ou créez en une nouvelle en effaçant la catégorie par défaut par le bouton avec **New category**
Puis faites **OK**

9.2.2 Ajout d'une nouvelle démo à la librairie



Écrivez un programme de test qui utilise le nouveau bloc.
Faites **File | Save as demo ...**
Choisissez dans la liste déroulante le bloc pour lequel vous avez écrit cette démo.
Choisissez le N° d'ordre de la démo (pas plus de 3 démos en général)
Puis faites **OK**

10 Écriture d'une macro en assembleur et déboguage du code machine

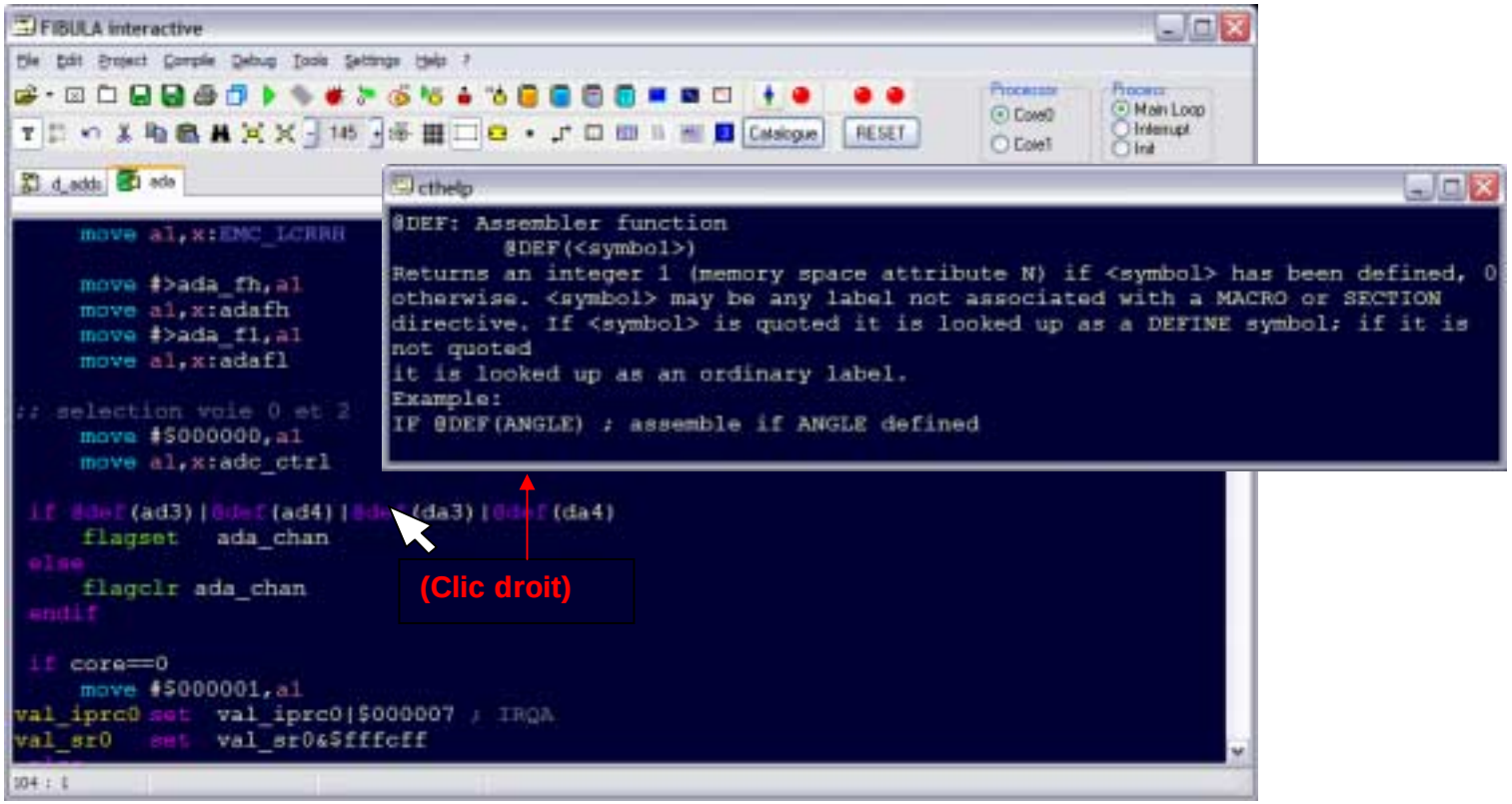
10.1 Édition de code assembleur

Avant d'écrire du code assembleur, il faut avoir bien compris le fonctionnement du coeur DSP56300:
Comportement des accumulateurs, arithmétique fractionnaire, saturation, débordement, calibrage dynamique
Comportement des pointeurs: adressage linéaire, adressage modulo, adressage en miroir
Jeu d'instructions: parallélisme, exécution conditionnelle etc…
Reportez-vous à l'ouvrage de référence **Help|Doc Freescale | DSP56300 Core**
Concernant l'assembleur: directives, macros, opérateurs et fonctions mathématiques: consultez **Help | Doc Freescale | Assembler**
Pour la programmation des périphériques et du H/W, consultez **Help| Doc Freescale|DSP56720 Ref manual**

10.2 Coloration syntaxique et aide contextuelle

Dans une page Assembleur, les couleurs ont (en fond noir) les significations suivantes:
Aqua: Instruction
Vert: Macro
Mauve: Directive ou fonctions de l'ASM
Jaune: Étiquette
Rose: Registre du coeur
Lavande: Registre de périphérique
Gris: Commentaire
Rouge: Erreur

En cliquant avec le bouton droit sur un mot coloré, une aide contextuelle surgit (popup).

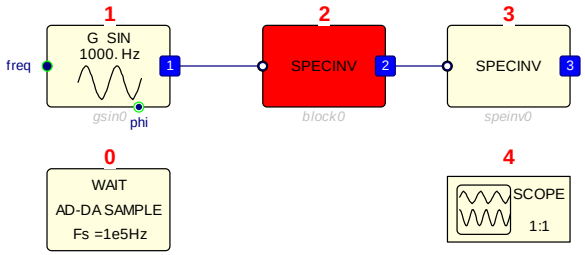


10.3 Exemple

Soit à créer un bloc fonctionnel "codeur à retournement de spectre". Le principe consiste à changer le signe d'un échantillon sur 2. Le décodage se fait avec le même bloc, avec toutefois une ambiguïté sur le signe du résultat décodé.

```
specinv  macro    name
         flagr    name\_flag
         lda      name\_in
         flagtog  name\_flag
         neg      a     ifcs
         sta      name
         u        a
         endm
```

Après avoir créé le descripteur du nouveau bloc en utilisant la procedure 6.1.3, éditez la macro.
Toute macro qui peut être instanciée plusieurs fois a pour 1er argument "**name**". Lors de l'extension de la macro, "**name**" sera remplacé par le nom d'instance du bloc. Les chaînes et toutes les variables ainsi que les étiquettes seront de la forme **name_xxxx** (concaténation du nom d'instance du bloc avec un underscore et une chaîne.
Dans le programme ci-contre, **flagr** est une macro qui crée une variable booléenne en réservant un bit de mémoire et en l'associant à un identifiant. La macro **lda** permet de charger l'accumulateur avec une variable sans connaître le champ mémoire dans lequel réside cette variable. La macro **flagtog** permute la variable booléenne et rend dans la retenue la valeur du bit avant basculement. L'instruction **neg a** change le signe de l'accumulateur. La mention **ifcs** dans le champ parallèle signifie que l'instruction ne s'exécute que si le bit de retenue (carry) vaut 1. Dans Fibula, on a l'habitude de donner le nom d'instance seul à la sortie principale du bloc, et le nom d'instance suivi de **_in** à l'entrée principale; **sta**, de même que **lda** permet de s'affranchir de la connaissance du champ de la variable **name**. Enfin la ligne **u a** indique que l'accumulateur A a été utilisé (utile pour la sauvegarde de contexte si le bloc s'exécute sous interruption).



Pour tester le nouveau bloc SPECINV, créez le schéma ci-contre.
Lancez la compilation et l'exécution du programme.

Ouvrez le débogueur de code machine en cliquant ce bouton → 
Sélectionnez le bloc N° 2 et cliquez sur le bouton point d'arrêt
Le bloc apparaît maintenant en rouge, et une ligne apparaît en rouge dans le code machine désassemblé. Cette ligne est un point d'arrêt sur la première instruction exécutable du bloc. Arrêtez et relancez l'exécution du programme à l'aide du bouton GO/HALT. La barre d'exécution en bleu va s'arrêter sur le point d'arrêt. Vous pouvez ensuite continuer en pas à pas et observer les changements dans les registres et la mémoire du DSP.

10.4 Le déboguage du code machine

Le déboguage du code machine peut se faire, soit à l'aide du simulateur logiciel Freescale, soit à l'aide du débogueur Fibula JTAG-OnCE, soit enfin à l'aide du nouveau débogueur Temps Réel ci-contre.
Le simulateur Freescale fonctionne parfaitement, mais ne permet pas facilement de tenir compte du comportement des périphériques matériels.

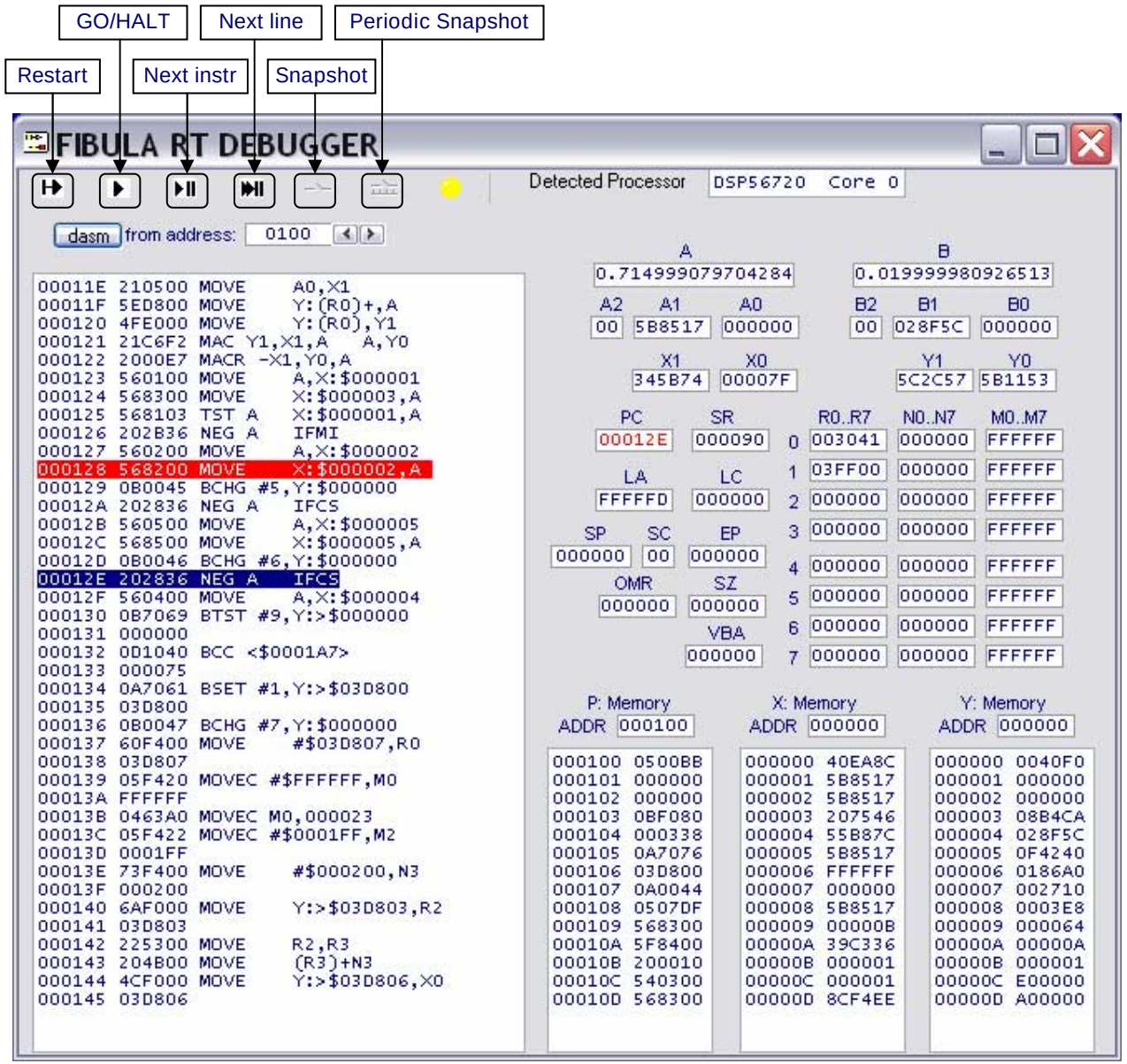
Le débogueur JTAG-OnCE est un émulateur non invasif qui ne peut fonctionner que lorsque le programme est arrêté dans le mode déboguage.
Il ne fonctionne que sur la version 002 des boîtiers Didalab.

Le **débogueur temps réel** utilise le port SPI et un canal DMA pour accéder à la mémoire et aux registres en cours de fonctionnement. Le programme doit avoir été lancé pour pouvoir le déboguer.

On peut observer l'état de la machine en cours de fonctionnement en utilisant les boutons Snapshot ou Periodic Snapshot.

On peut arrêter / relancer le programme à l'aide du bouton GO/HALT. La commande GO active les points d'arrêt, et la commande HALT les inhibe.

On dépose / supprime un point d'arrêt en double-cliquant sur une ligne du désassembleur.
Dans le cas d'instructions à 2 mots, le point d'arrêt doit être déposé sur le premier mot.



Lorsque le programme rencontre un point d'arrêt, la tâche courante est échangée avec une boucle d'attente, ce qui permet d'observer ou de modifier le contexte en écrivant des valeurs hexadécimales dans les champs des différents registres. Dans cet état arrêté, on peut continuer en pas à pas, soit ligne par ligne, soit instruction par instruction, ce qui est différent lorsqu'il y a un saut conditionnel ou un appel de sous-programme.

Les points d'arrêt sont obtenus par insertion de l'instruction TRAP. Le pas à pas installe et désinstalle des points d'arrêt provisoires.

NB:
Dans cette version Bêta du débogueur, on ne peut toutefois pas, en mode pas à pas, recouvrer l'adresse de retour d'un sous-programme ou d'une interruption, (instructions RTS ou RTI) car cette version ne gère pas une image de la pile.

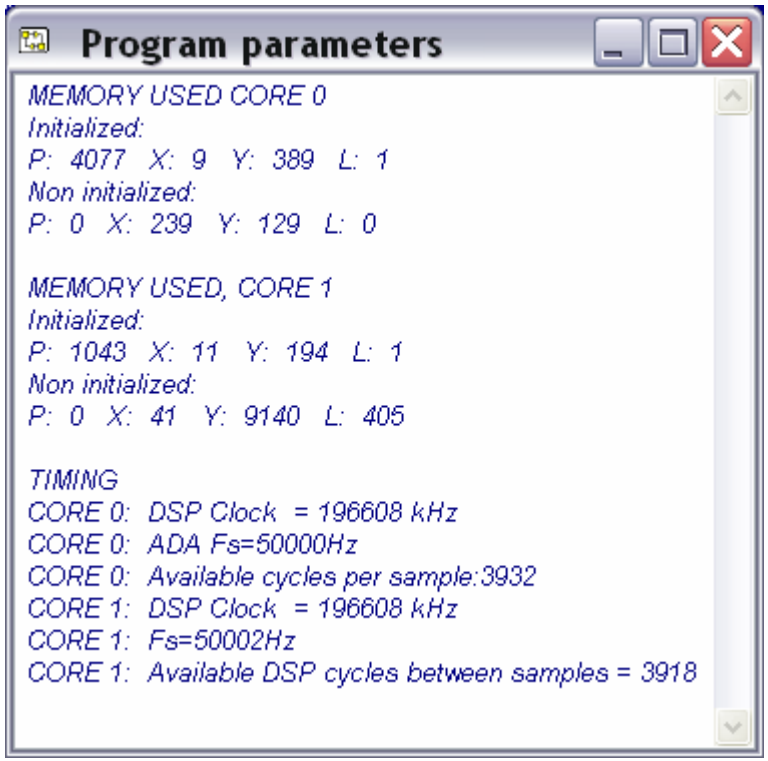
En cas de plantage du code machine (rare), le débogueur logiciel ne sera d'aucun secours si la tâche de communication temps réel est HS. Dans ce cas, le seul recours est l'utilisation du simulateur Freescale (Debug | Freescale simulator)

IMPORTANT:
Le débogueur FIBULA, et en particulier, son désassembleur ne doivent pas être utilisés à des fins d'ingénierie inverse.

10.5 Évaluation des performances

10.5.1 Consommation mémoire, et paramètres divers

En cliquant sur ce bouton, vois faites apparaître la fenêtre des paramètres et constantes du dernier programme compilé. La consommation mémoire est exprimée en nombre de mots 24 bits. Elle vous permet d'évaluer la taille minimale d'une E2PROM pour la sauvegarde de ce programme (fonctionnalité non encore implantée sur la version Didalab).

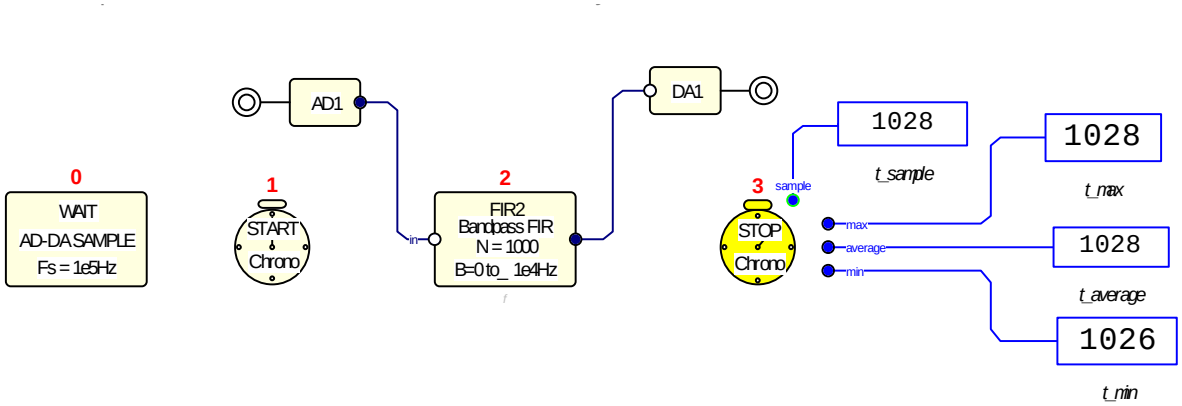


NB
Diverses autres informations sont extraites du listing.
Si vous programmez des macros en assembleur, vous pouvez également ajouter des messages dans cette fenêtre. A titre d'exemple, voici la directive Assembleur qui produit la 1ère ligne du TIMING ci-contre:

```
msg     '>>> DSP Clock  = ',@cvi(fdsp/1000.),' kHz'
```

Le triple chevron en début de message est la clef qui permet de marquer les lignes à afficher.

10.5.2 Mesure du temps d'exécution entre 2 points d'un programme



Pour mesurer le temps d'exécution entre deux points du programme, insérez les blocs Chronomètre **Tic** et **Toc** (start et stop chrono) en choisissant leurs numéros d'exécution de manière à ce qu'ils encadrent le bloc ou le groupe de blocs à tester.
Les bornes du bloc TOC ont l'attribut optionnel. On peut donc par exemple ne connecter que t_max.

Le bloc TOC peut fonctionner en **mode statistique**. Il possède 4 sorties de type INTEGER sur lesquelles on doit connecter des objets voltmètres virtuels.
Les valeurs affichées correspondent aux temps d'exécution en cycles machine, de haut en bas:

- Temps instantané,
- Temps maximal,
- Temps moyen,
- Temps minimal

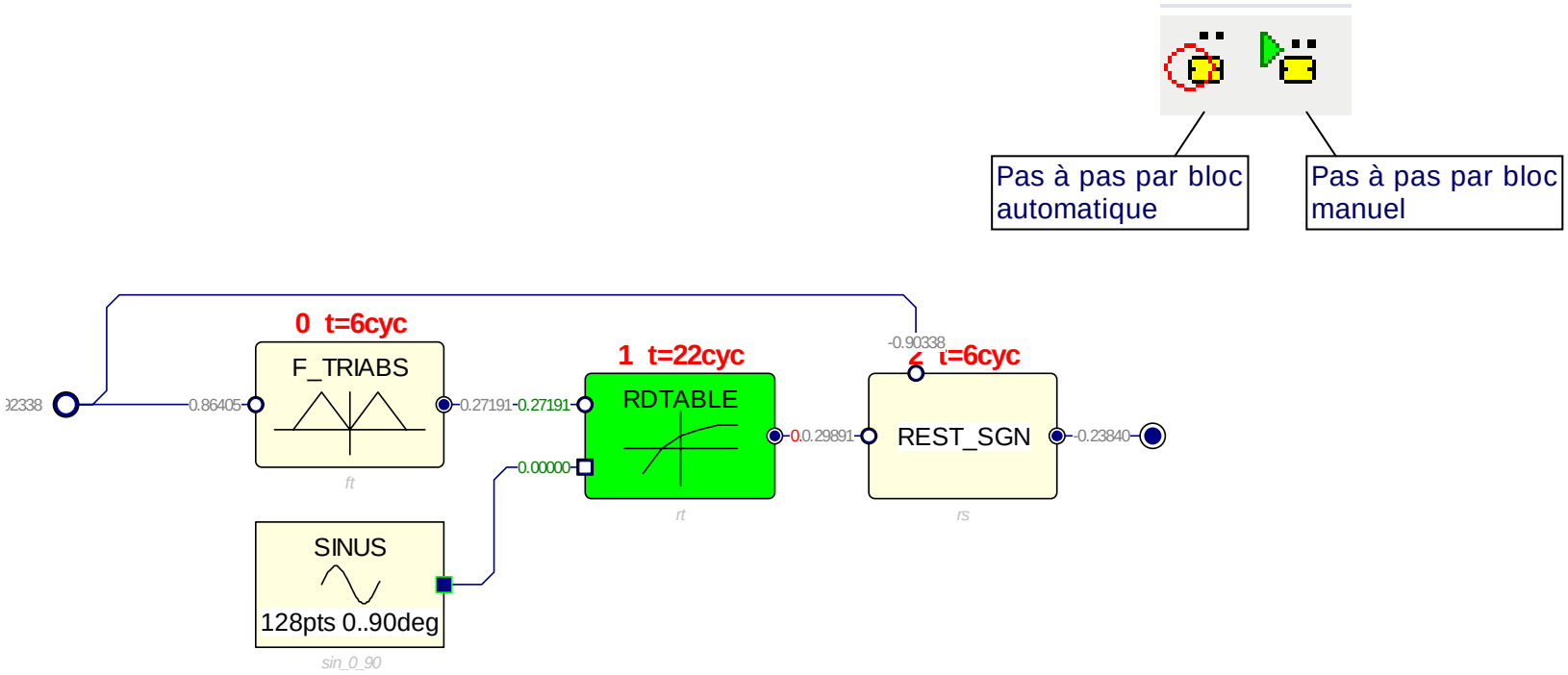
Le chronomètre utilise l'un des 3 timers dont dispose chaque processeur. Comme les timers sont alimentés par la moitié de la fréquence système, le résultat est juste **à 2 cycles près**.

NB
Lorsque le programme comporte des clauses conditionnelles, il faut programmer le bloc TOC avec un nombre d'occurrences suffisant pour que le cas le plus favorable, et le cas le moins favorable se produisent, et conduisent à une valeur minimale et une valeur maximale du temps respectivement.

10.5.3 Exécution du programme en mode Bloc par Bloc

En utilisant le pas à pas bloc par bloc, les durées d'exécution de chaque bloc s'affichent à 2 cycles près.
On peut également voir l'évolution des entrées / sorties en petits caractères.

En vert, le bloc qui va s'exécuter.
En rouge, les points d'arrêt.



11 Les outils de FIBULA

11.1 Création de documents

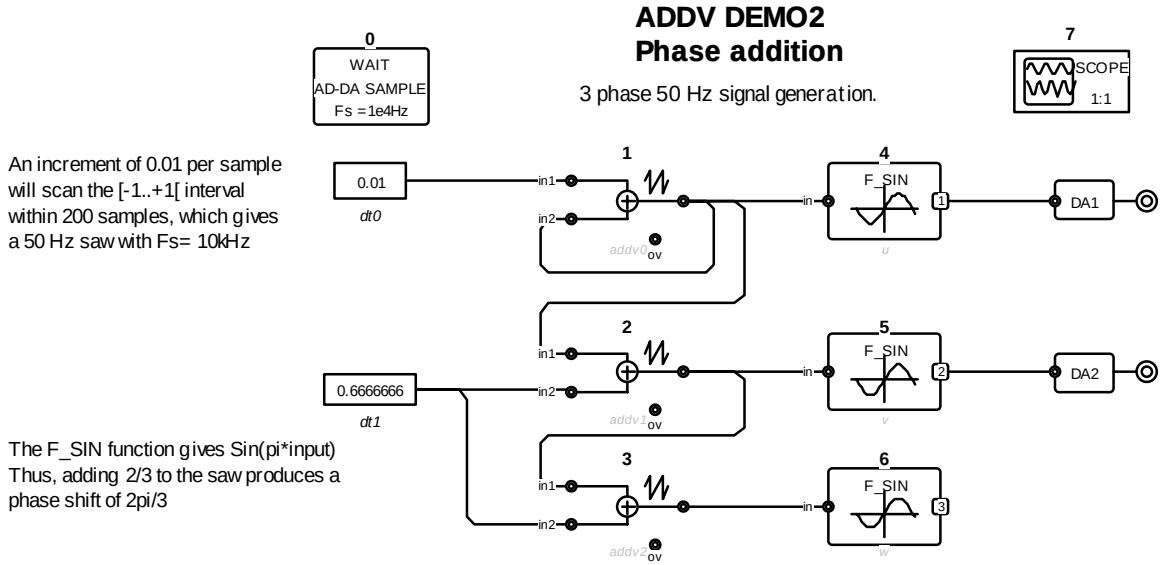
11.1.1 Copie de l'image d'un schéma

Pour copier le schéma de la page courante au format **image vectorielle** EMF (Enhanced Metafile Format), faites simplement

Ctrl-M

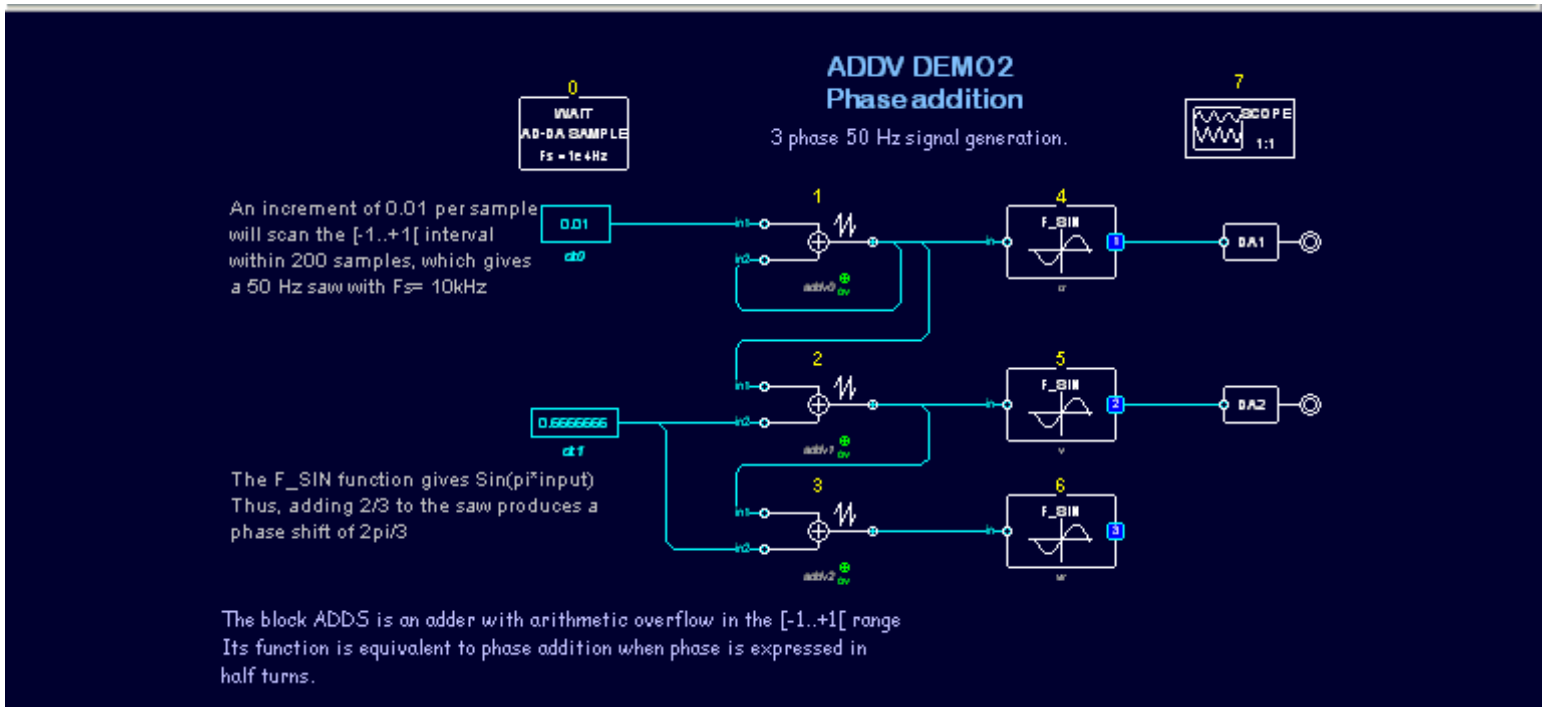
Vous pouvez ensuite coller l'image dans un document MS Word par exemple en faisant **Ctrl-V**, et vous pouvez le redimensionner à volonté

Notez que, hormis les colorations de sélection et de point d'arrêt, la copie est en noir et blanc, quelle que soit la couleur du fond



Copier - Coller en mode vectoriel avec Ctrl-M

Vous pouvez aussi copier l'image au format Bitmap (BMB) en faisant **Ctrl-B**
Dans ce cas, les couleurs sont conservées lorsque vous collez l'image dans un document. Mais le redimensionnement de l'image Bitmap s'accompagnera d'une perte de qualité.

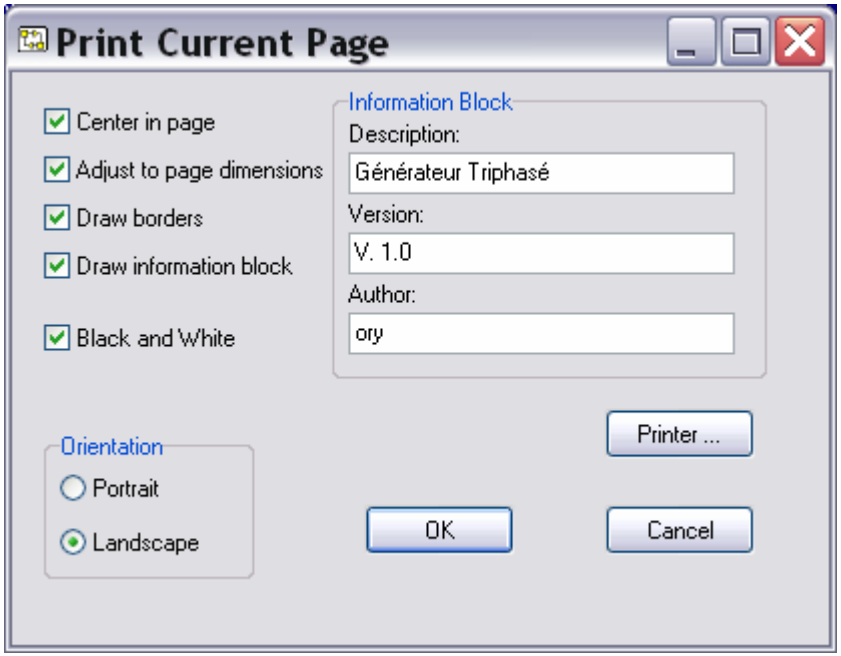


Copier - Coller en mode Bitmap avec Ctrl-B

Notez que lorsque vous faites Ctrl-C, vous copiez la description textuelle de la partie sélectionnée de la du schéma. Ctrl-V vous permettra de coller cette partie de schéma dans une page graphique quelconque de Fibula, mais pas dans un document Word.

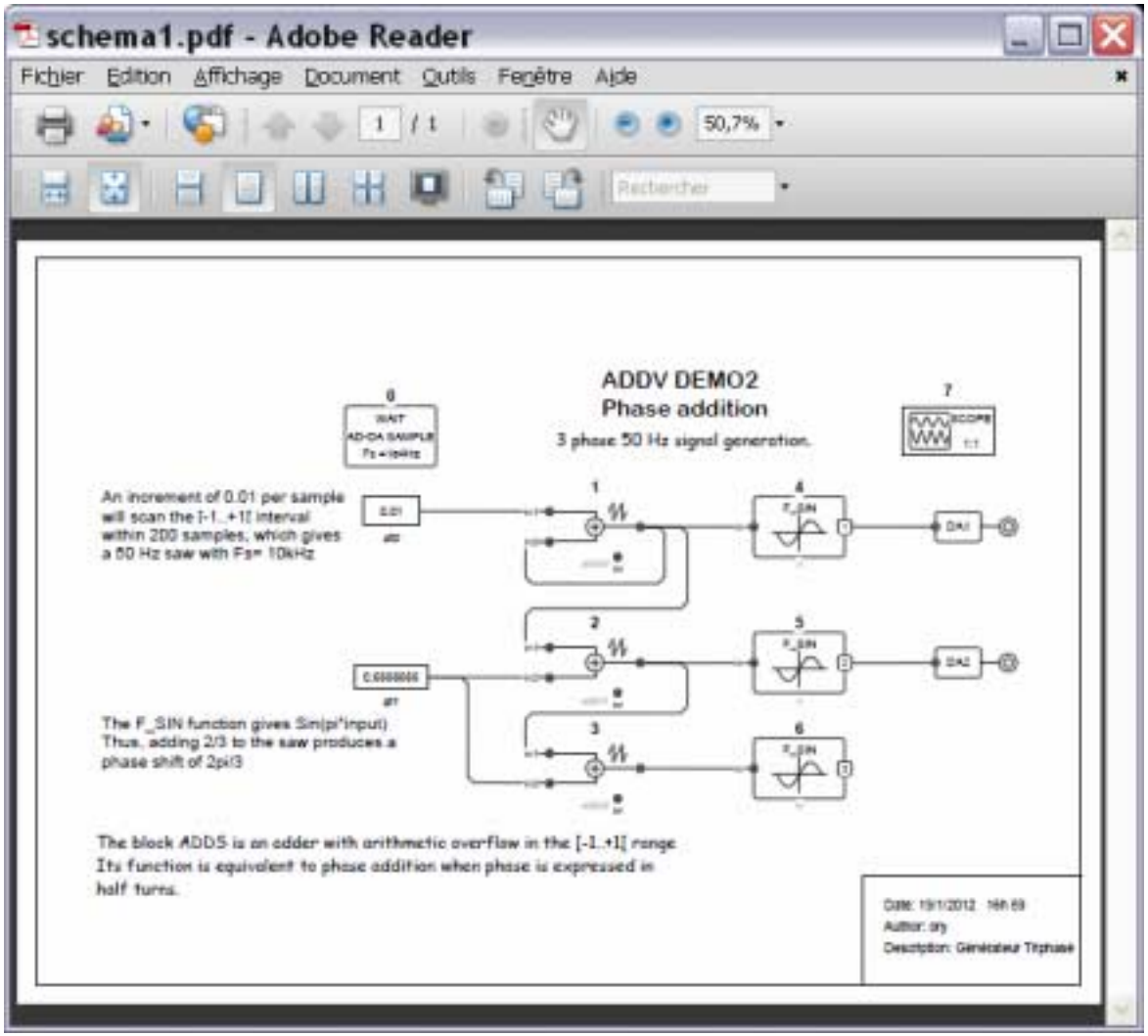
11.1.2 Impression d'un schéma

En cliquant le bouton Imprimante ou en faisant **Ctrl-P**, vous imprimez le schéma de la page courante sur l'imprimante que vous avez sélectionnée. Pour des raisons évidentes d'économie, il est fortement recommandé de laisser la case **Black and White** cochée si vous imprimez sur papier. Tous les fonds colorés apparaîtront alors en blanc.



Avec les réglages par défaut, le schéma sera imprimé de manière à remplir une page A4 en orientation Paysage.

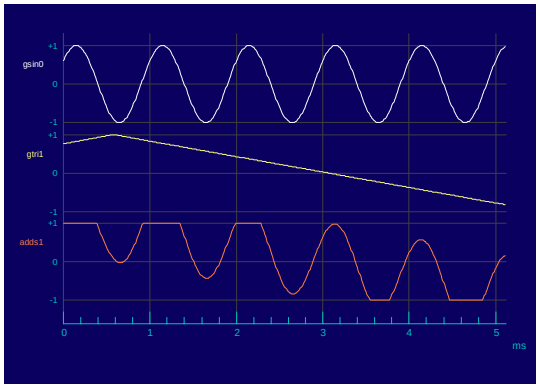
Impression d'un schéma sur l'imprimante virtuelle Adobe PDF Distiller ⟶



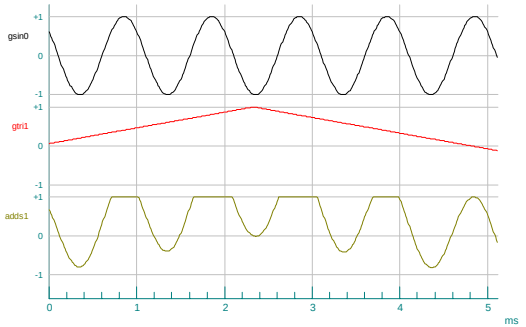
11.1.3 Copie d'un oscillogramme

Le bouton Copier de l'oscilloscope place une image vectorielle EMF en couleurs dans le presse-papier.
Comme précédemment, vous pouvez coller cette image dans un document, et la redimensionner à volonté.

Pour obtenir une image sur fond blanc, cochez la case White avant de faire la copie



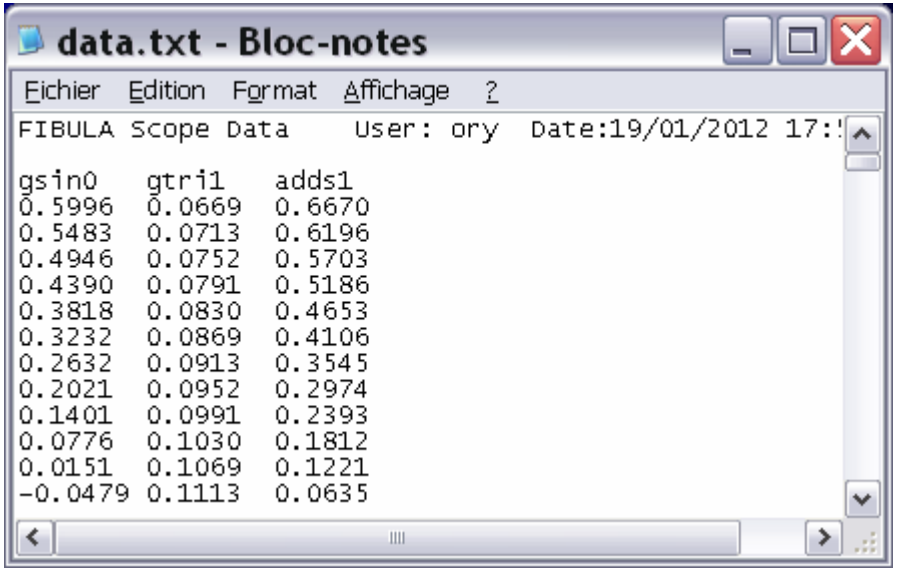
Copier - Coller d'oscillogramme (par défaut)



idem, avec case White cochée

11.1.4 Récupération des données de l'oscilloscope

Le bouton "disquette" de l'oscilloscope crée un fichier texte qui affiche pour chaque canal une colonne représentant les échantillons


data.txt - Bloc-notes

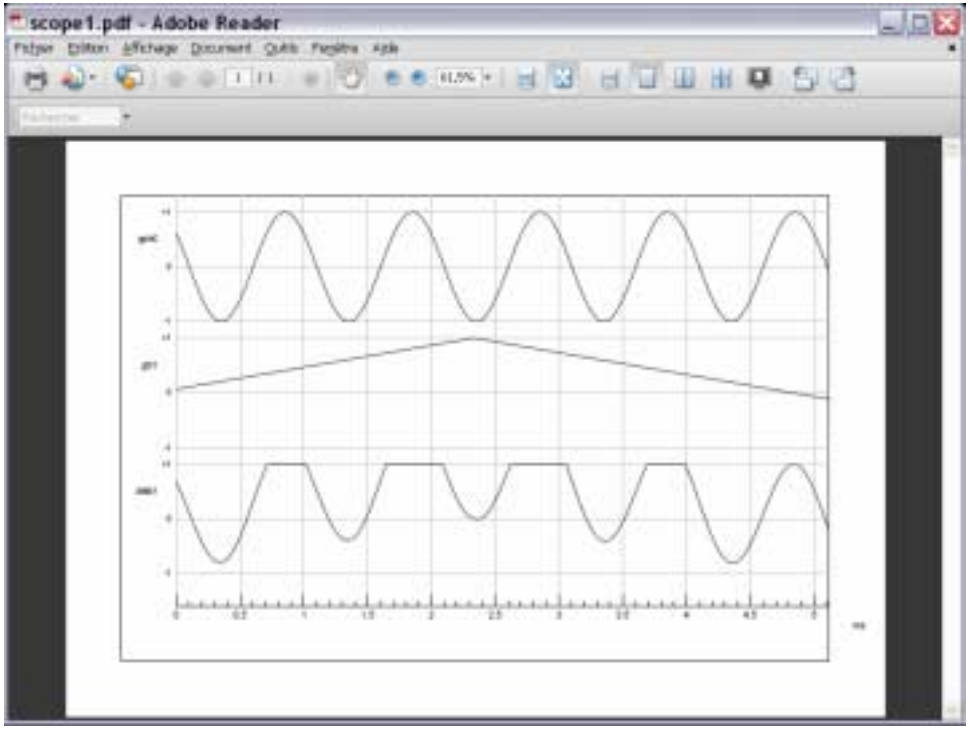
Fichier Edition Format Affichage ?

```
FIBULA Scope Data    User: ory  Date:19/01/2012 17:!

gsin0   gtri1   adds1
0.5996  0.0669  0.6670
0.5483  0.0713  0.6196
0.4946  0.0752  0.5703
0.4390  0.0791  0.5186
0.3818  0.0830  0.4653
0.3232  0.0869  0.4106
0.2632  0.0913  0.3545
0.2021  0.0952  0.2974
0.1401  0.0991  0.2393
0.0776  0.1030  0.1812
0.0151  0.1069  0.1221
-0.0479 0.1113  0.0635
```

11.1.5 Impression d'un oscillogramme

Cliquez sur le bouton "Imprimer" de l'oscilloscope, et choisissez l'orientation Paysage pour votre imprimante.

L'image remplit une page A4.

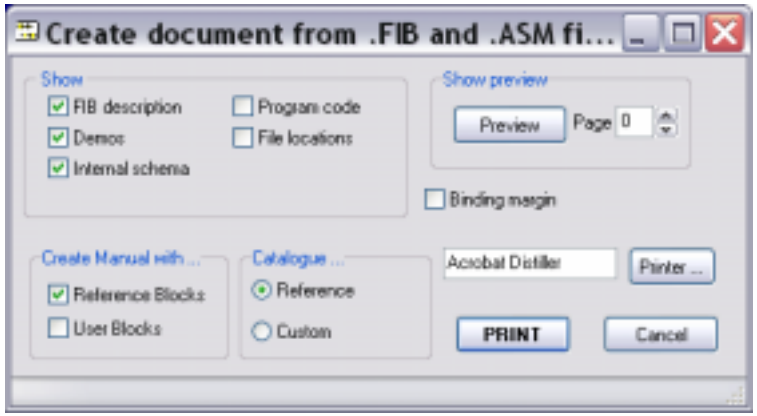


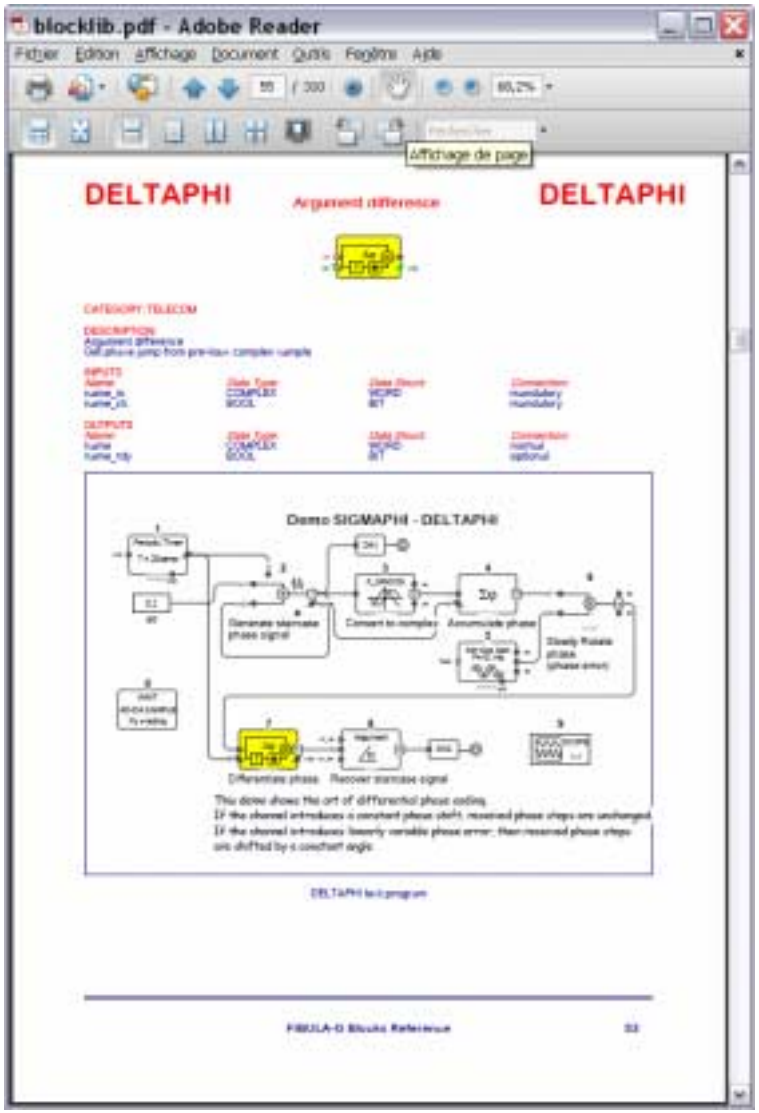
11.2 Création et impression du manuel de la librairie

Fibula dispose d'un **générateur automatique de documentation**. Au fil des mises à jour logicielles, il peut être utile de mettre à jour le document Blocklib.PDF de l'aide en ligne qui recense tous les blocs disponibles en bibliothèque.

Pour cela, faites la commande de menu **Tools | Print FIBULA Reference Manual**

Important: Réglez votre imprimante en orientation Portrait





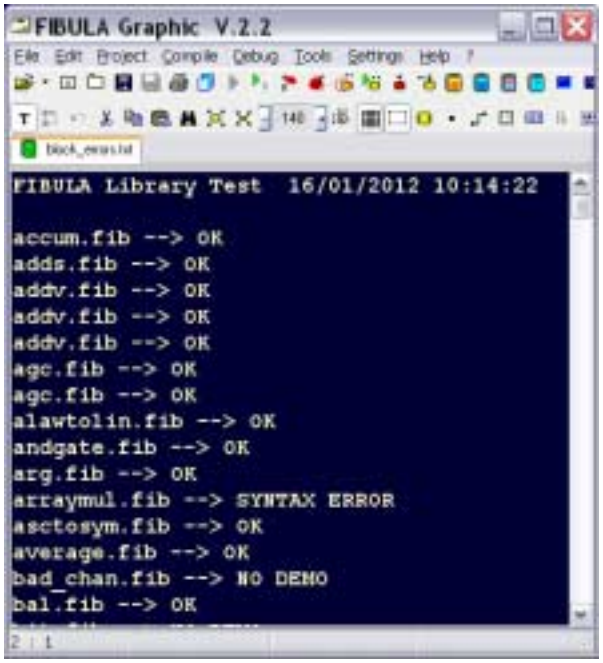
La page 53 sur 307 du manuel imprimé sur Adobe Distiller

11.3 Test de toutes les démos et rapport final

On peut tester l'ensemble des blocs de la bibliothèque avec la commande

Tools | Test/Run entire library

Pour chaque bloc, le programme recherche les différentes démos de ce bloc. Ensuite, il en vérifie les erreurs de syntaxe et les erreurs d'assemblage. Si le programme comporte un oscilloscope, celui-ci est exécuté quelques secondes, et on vérifie que des signaux apparaissent sur l'écran. Lorsque tous les blocs sont testés, on obtient un rapport comme ci-contre.



11.4 Visualisation des fichiers intermédiaires

Ce bouton réalise une compilation de la page courante et affiche le code ASM complet, le listing, ainsi que le code objet téléchargeable pour chacun des cœurs concernés. **Supprimé sur V3.x**

Ce bouton affiche le code ASM obtenu par compilation de la page courante (sans entête ni épilogue).

Ce bouton affiche le listing d'assemblage. Il permet de retrouver la cause des erreurs du code assembleur.

Ce bouton affiche le Mapping généré par l'éditeur de liens. Il permet de retrouver les adresses physiques des variables et des points d'entrée du programme.

12 ANNEXES

12.1 Format des fichiers FIB

Fichier de description d'un bloc:

```
INPUT | OPT_INPUT | OUTPUT | OPT_OUTPUT
name,datatype,pos[,posx,posy[,field[,address]]]

RESET
INIT
UNIQUE
FIRST
ISR
NONEX
DEFINE
string,value{,string,value}

SCHEMA
{schema description}
ENDSCH

MACRO
macname

ASM
        instr
...
        instr
<blanc>

PARAMS
param0,nbdefts,deft1,deft2,..,deftn{,parami,nbdefts,deft1,deft2,..,deftm}

DESCRIPTION
blabla..
..
blabla
<blanc>

GRAPH
{graphcmds}
<blanc
```

Fichier de description de schéma:

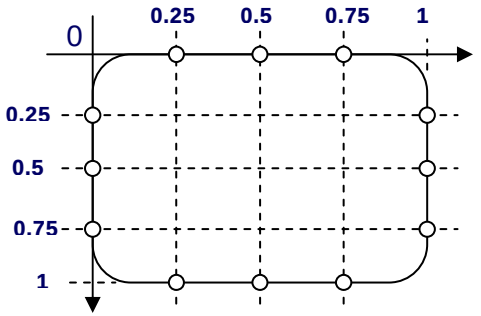
```
BLOCKD                  //
{block description}     // Only for unsaved blocks
ENDB                    //

BLOCK, core, process
function,name,blocktype
param0,param1,..paramn
posx,posy,sizx,sizy,orient

DATA
name,value
posx,posy,sizx,sizy

MEM
name,initfile,size,col,datatype,field[,modulo]
pinname,pinpos{,pinname,pinpos}
posx,posy,sizx,sizy

DEFLIST
name,posx,posy,sizx,sizy
name1   value1   comment1
name2   value2   comment2
...
<blanc>

TEXT
posx,posy,sizx,sizy,style
blablabla ..
blabala ..
<blanc>
WIRE
sblkname,sioname,siopos,dblkname,dioname,diopos
nbsegments,seg,seg,seg ..,seg

DESCRIPTION
blabla
<blanc>
```

12.2 Commandes graphiques vectorielles

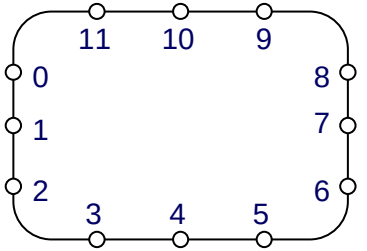
mp,pin_nr	move to pin
lp,pin_nr	line to pin
dp,pin_nr	line to pin (dotted)
mt,x,y	move to coordinates x,y
lt,x,y	line to coordinates x,y
dt,x,y	line to coordinates x,y (dotted)
ht,x	horizontal line to coordinate x
vt,y	vertical line to coordinate y
ci,x,y,r	circle centre at x,y; radius r (relative to ysize)
tr,x,y,r,o	triangle centre at x,y; radius=r; orientation=o
re,x0,y0,x1,y1	rounded rectangle defined by diagonal (x0,y0 ; x1,y1)
rec,x0,y0,x1,y1	rectangle defined by diagonal (x0,y0 ; x1,y1)
rnd,x0,y0,w,h,g|b|u	random signal centre x0,y0; width w; height h; gaussian|binary|uniform
sin,x0,y0,w,a,phi0,phi1	sine wave centred at x0,y0; width w; amplitude a; initial phase/pi; end phase/pi
sinc,x0,y0,w,a	sinc function centred at x0,y0, width w, amplitude a
sqr,x0,y0,w,h	parabolic function centred at x0,y0, width w, height h
exp,x0,y0,w,h	exponential function centred at x0,y0, width w, height h
expa,x0,y0,w,h	exp(-abs(x)) function centred at x0,y0, width w, height h
gauss,x0,y0,w,h	Gauss function centred at x0,y0, width w, height h
x,x0,y0,r	cross at centre x0,y0; radius=r;
* + - . :	same syntax as above
name	concatenate block name to current text
a,n	concatenate argument nr n to current text
af,n	concatenate 'Hz' or '*Fs/2' to current text, depending if argument n is 'abs' or 'rel'
at,n	concatenate 'sec' or 'samples' to current text, depending if argument n is 'abs' or 'rel'
t,text	concatenate text to current text
g,text	concatenate Greek characters to current text
tw,x,y	write current text centered at x,y
f,x,y	write block function centered at x,y
curv,x0,y0,dx,y1,y2...	draw curve (constant dx increment)
segs,x0,y0,x1,y1,...	draw curve (between random points)
step,x0,y0,dx,y1,y2..	same as curve but histogram style
pen,w,c	change pen properties width and color (negative width means dotted)
bcol,c	change brush color

colors are:

0:black	8:gray
1:brown	9:white
2:red	10:silver
3:orange	11:fuchsia
4:yellow	12:aqua
5:green	13:lime
6:blue	14:navy
7:purple	



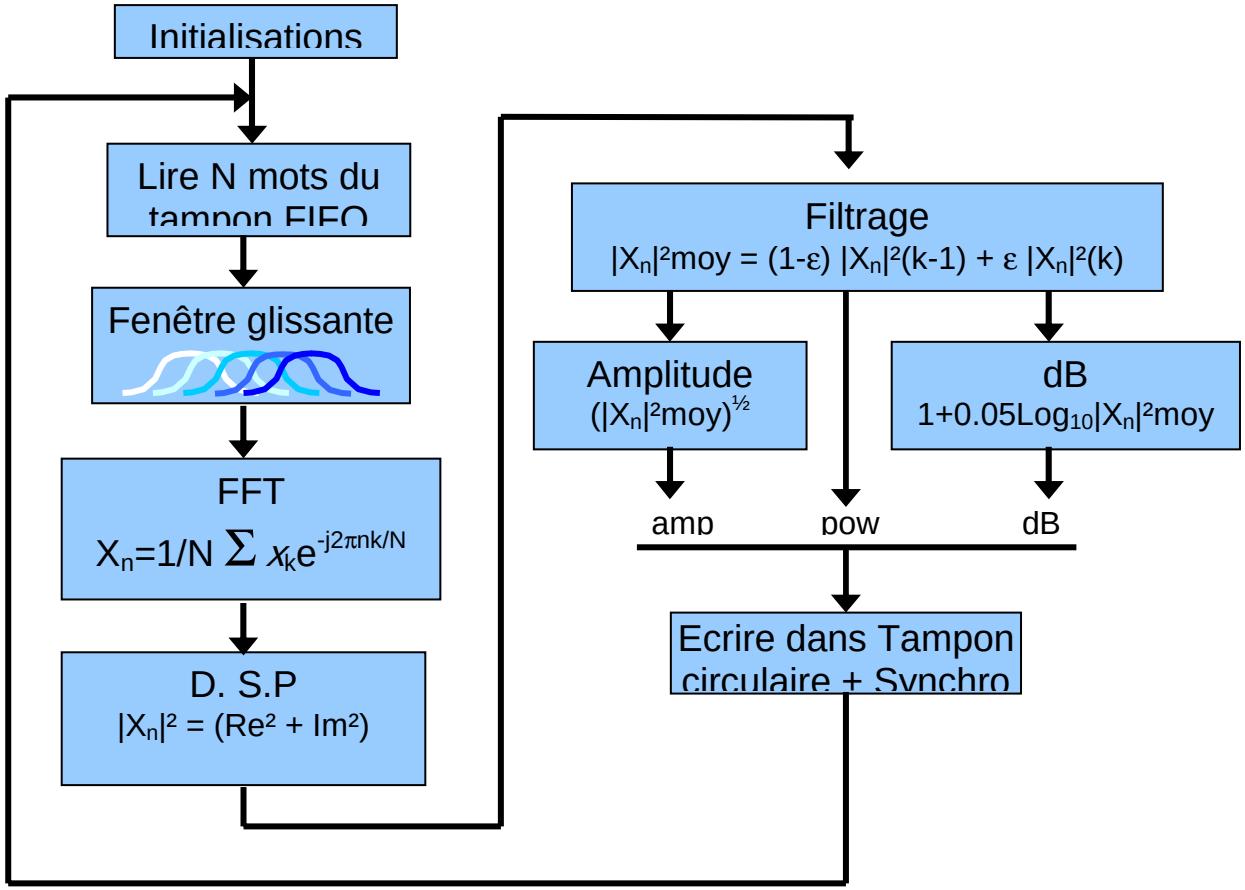
Système de coordonnées utilisées pour dessiner un bloc.

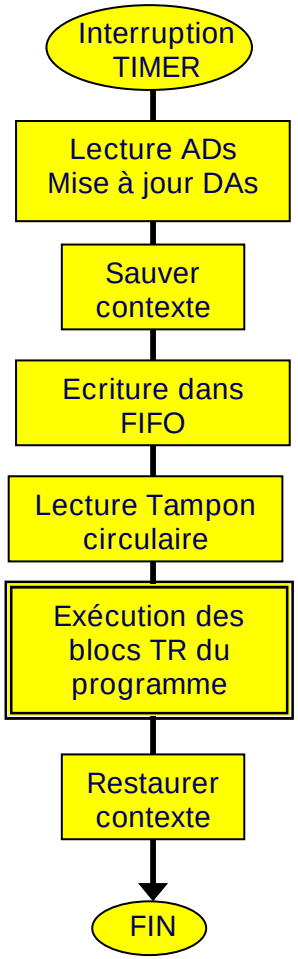


Numérotation de la position des bornes

12.3 Comment fonctionne l'analyseur de spectre SPECAN

L'analyseur de spectre SPECAN s'utilise comme n'importe quel bloc, en connectant son entrée à un signal à analyser, et sa sortie à un organe permettant la visualisation du spectre, par exemple DA1 connecté à un oscilloscope, sonde d'oscilloscope virtuel SCOPE, ou Miniscope. Le signal spectre obtenu est un balayage périodique du spectre avec des impulsions négatives pour la synchronisation de l'image.Toutefois, la compilation du bloc SPECAN donne lieu à une architecture logicielle particulière: le code de SPECAN se compose d'une boucle principale comprenant le calcul du spectre (en bleu ci-dessous) et un service d'interruption exécuté à chaque échantillon (en jaune). Tous les autres blocs du schéma sont alors exécutés durant cette interruption.





12.4 Nouveautés de la mise à jour V3.1

Oscilloscope

L'oscilloscope virtuel possède désormais un mode de déclenchement automatique. Le mode sonde volante ou pointe de touche est actif dès que le programme s'exécute (le bouton sonde n'est plus nécessaire). On peut alors attacher la sonde à un fil en cliquant dessus, ce qui permet d'utiliser les commandes du scope. L'ouverture par double-clic d'un bloc possédant un schéma interne permet de tester les signaux internes.

Gestion des chaînes de caractères

Les chaînes sont maintenant des variables définies par un pointeur. Elles sont affichables en temps réel dans un rectangle de données de la même manière que les autres types. Les chaînes comportant des champs de données variables sont obtenues à l'aide du nouveau bloc MAKESTRING qui traite une chaîne formatée à la manière de PRINTF du langage C.
Les blocs frtohex, frtostr, inttostr, string, wordtobin, wordtohex ne sont plus compatibles et sont donc supprimés du catalogue.

Prise en charge des modules Radio Logicielle (ETD410011)

Détection automatique du module connecté

12.5 Nouveautés de la Version 3.0 interactive

Interactivité

Toutes les variables de l'application peuvent être visualisées en temps réel en y connectant un objet "donnée". Celui-ci peut prendre différentes apparences ou skins (potentiomètres, voltmètres, …) Les paramètres d'entrée peuvent en outre être modifiés par action de la souris. Les éléments d'une matrice peuvent également être visualisés et modifiés pendant l'exécution.
Chronomètre TIC - TOC statistique Améliorations de l'éditeur de schémas (routage automatique des fils).

Nouvelles fonctions

Nouveaux blocs pour la synchronisation inter core
Blocs pour la synthèse musicale: séquenceur MIDI, filtre de canal, instruments de musique simples, fichiers MIDI encodés ASM

Démos, Exemples

Les démos de blocs accessibles par le Catalogue ont été simplifiées pour ne montrer que la fonctionnalité du bloc courant.

Les programmes et démos plus conséquents sont rassembles sous la rubrique Example Programs.

12.6 Nouveautés de la version 2.2

Nouveau débogueur temps réel pour le code machine.
Son utilisation est beaucoup plus simple que le simulateur Freescale. Il permet en particulier l'observation dynamique des données en cours d'exécution du programme ("periodic snapshot").

Analyseur de spectre
L'analyseur de spectre **SPECAN** a été modifié afin de ne plus être dépendant du matériel (convertisseurs AN / Codec audio).
L'utilisation d'une nouvelle version de la FFT permet maintenant de descendre un peu en dessous des -120dB en bruit de calcul.
Il possède un facteur de décimation programmable qui permet avec un filtrage passe-bas d'explorer les très basses fréquences.

Nouveaux blocs de transformées
Le bloc **FFT** a été réécrit pour que son emploi soit plus facile; le débrouillage binaire en miroir y est réalisé, l'entrée et la sortie sont distinctes. Le facteur 1/N de la transformée directe (nécessaire en arithmétique fractionnaire) y est obtenu par un calibrage dynamique.
Les nouveaux blocs **IFFT, DFT, IDFT, DCT, IDCT** (FFT inverse, TF Discrète, TFD Inverse, Transformée en Cosinus Discret directe et inverse) sont écrits dans le même esprit. Pour réaliser des applications à flux de données continu comportant les transformées ci-dessus, les nouveaux blocs **FLOWTOPAK** et **PAKTOFLOW** permettent de mettre en œuvre très simplement des FIFO Producteur Consommateur.

Interpréteur de fichiers MIDI:
Ce "premier jus" met en œuvre un synthétiseur musical qui utilise des tables de formes d'ondes et des tables d'enveloppes. Le décodage des percussions n'est pas encore réalisé. Présentement, seuls, le piano et l'orgue donnent un son satisfaisant. Afin de pouvoir réaliser des applications romables et autonomes, les fichiers MIDI font partie du code machine téléchargé. Le petit utilitaire MIDItoFIB.exe permet de convertir des fichiers MIDI en blocs de données Fibula connectables à l'interpréteur musical.

12.7 Nouveautés de la Version 2.1

Nouveaux blocs, et réécriture de certains blocs
Certains blocs précédemment écrits en assembleur (en particulier, les générateurs périodiques tels que G_SIN) sont maintenant décrits à l'aide de schémas. Notre intention est de tendre vers une description la plus graphique possible, donc moins dépendante de la cible, car seuls les blocs écrits en code textuel dépendent de la machine cible.

Unités de fréquence et de temps

Les fréquences peuvent être désormais exprimées en **Hz**, **kHz**, **MHz**, **Fs**, **Fs/2** et les durées en **s**, **ms**, **µs**, et **samp**. Chacun de ces symboles correspond à un facteur multiplicateur de la fréquence ou de la durée.

La propriété MESSAGE des bornes booléennes
Les bornes booléennes (vertes) peuvent désormais posséder la propriété MESSAGE qui permet à une entrée réceptrice d'accuser réception du message (fonction Test-And-Clear). Voir 4.5 page 31.

Editeur de données amélioré
Au type fractionnaire s'ajoute maintenant la possibilité de saisir des matrices entières, booléennes et complexes
L'éditeur de données est interactif lorsque le programme s'exécute.

Scope:
les options d'affichage fonctionnent aussi à l'arrêt de l'oscilloscope.
Les bornes **sync** et **busy** sont supprimées (car souvent confondues avec des entrées), mais les variables **scope_sync** et **scope_busy** subsistent.

Correction de plusieurs bugs de la V2.0

12.8 Nouveautés de la Version 2.0

La liaison USB2
La gestion de projet
Il est maintenant possible de définir le répertoire de travail habituel, les répertoires liés aux blocs créés par l'utilisateur, ainsi que le répertoire qui contient les fichiers descripteurs du catalogue. Cela permet de créer facilement un catalogue sur mesure en fonction des besoins de chaque enseignement.
Le choix du processeur et du processus
Les blocs peuvent être exécutés par le cœur0 ou le cœur1. La coloration du bloc indique le cœur. Habituellement, les blocs s'exécutent dans une bouche principale temporisée par l'échantillonneur. On peut maintenant faire exécuter un ensemble de blocs dans une interruption (en général déclenchée par l'apparition de données) et réserver la boucle principale à une tâche asynchrone. On peut également placer des blocs dans une tâche d'initialisation exécutée en début de programme.
Les tableaux et matrices
Les tableaux et matrices sont éditables numériquement ou graphiquement à l'aide d'un double-clic. Elles peuvent appartenir à un cœur ou être partagées. La modification du contenu est interactive, c'est-à-dire qu'elle est prise en compte en temps réel lorsque le programme s'exécute.

Scope amélioré

L'oscilloscope virtuel possède un mode plein écran « reste dessus » bien adapté aux réglages interactifs. On peut copier l'image en vectoriel, ou l'imprimer, ou encore recueillir les données dans un fichier texte. La représentation des échelles a été améliorée.

Table traçante virtuelle

L'Enregistreur ou PLOTTER permet d'afficher des phénomènes lents comme par exemple le relevé d'un diagramme de Bode par balayage de fréquence

Interactivité

Certains réglages sont pris en compte en cours de fonctionnement : Constantes numériques, valeurs dans les tableaux et matrices, fréquence ou période de certains blocs. Pour afficher en TR la valeur d'une sortie lente, il suffit de la connecter à une donnée et de sélectionner cette donnée.

Impression / Copier coller

Les schémas et les oscillogrammes sont imprimables en vectoriel. Le copier/coller, peut se faire soit au format Bitmap, soit au format vectoriel EMF

Couleurs

[1]Parce que le fond blanc fatigue la vue, on peut maintenant choisir 4 autres couleurs de fond associées à des palettes de couleurs syntaxiques adaptées

---