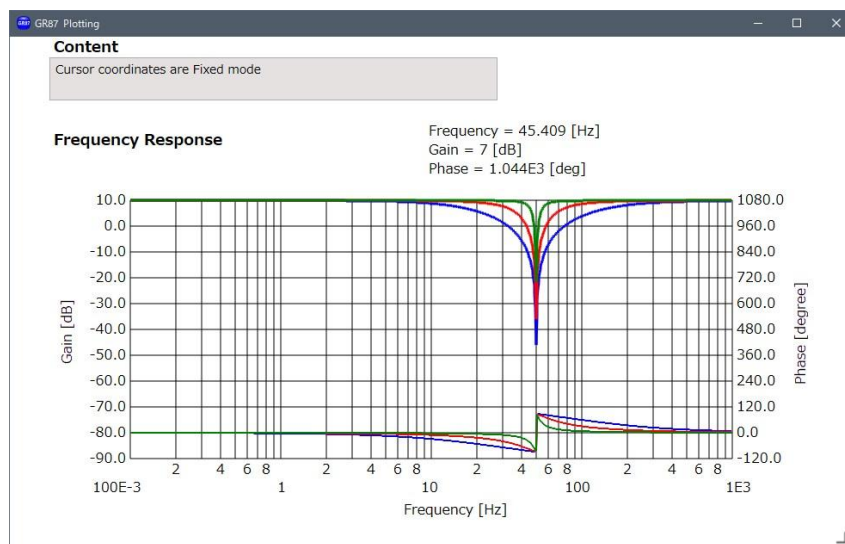
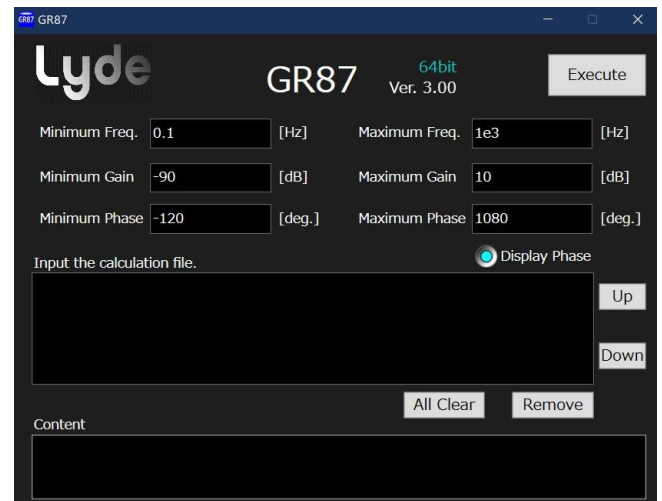
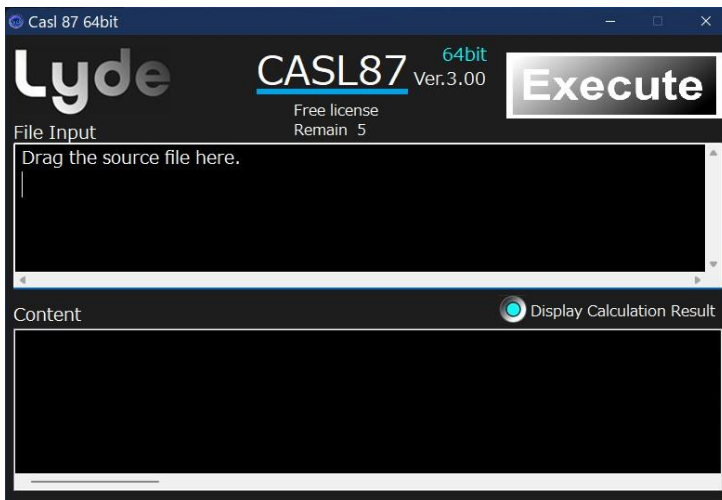


周波数特性解析用言語コンパイラー

CASL87 **ver.3.01**
GR87 **ver.3.00**

Rev. 2.0.2

使用説明書



履 歴

Revision No	日 付	内 容	担 当
Rev.1.0.0	2019.06.20	CASL87 ver.1.0 及び GR87 ver.1.0 用に Rev.1.0.0 を新規発行。	村岡如竹
Rev.1.0.0	2019.07.20	CASL87 ver.1.2 及び GR87 ver.1.0 用に Rev.1.0.0 を発行。 Include 文の拡張機能コマンドとして“Ex include”を追加。	村岡如竹
Rev.1.1.0	2019.08.05	Link 文内には回路ブロック（Circuit 文）が存在した場合のエラー処理の追加。（ページ 29）	村岡如竹
Rev.1.1.1	2019.08.11	同じ Circuit 文内で、同じノード文が重複している場合にこのエラーを追加。（ページ 29）	村岡如竹
Rev.1.1.2	2019.09.18	インストール時の Windows10 のセキュリティ画面及び、その説明を変更。（ページ 7）	村岡如竹
Rev.1.1.3	2019.09.23	Link 文における複数階層での接続指定での注意の追記（ページ 23）	村岡如竹
Rev.1.1.4	2019.09.30	Link 文における複数階層での接続指定での仕様変更（ページ 23、ページ 30）。GR87 のプロット出力ウィンドウのサイズ 800×570 に変更。	村岡如竹
Rev.1.1.5	2019.12.10	Calculation 文で入力に指定されたノードとグランドとの間に相互コンダクタンス以外の部品（R、L、C）が存在した場合にゼロ除算アボートが発生する不具合を修正。 Node 文内の相互コンダクタンスの電流方向を示すノード番号が Node 文に不一致の場合のエラーチェックの追加。（31 ページ） Node 文内の相互コンダクタンスの電流方向や差動電圧を示すノード番号が同一の場合のエラーチェックの追加。（31 ページ） Calculation 文内の Input と Output 指定が、回路ブロック内のノード番号として存在しない場合のエラーチェックを追加。（32 ページ）	村岡如竹
Rev.1.1.6	2019.12.26	CASL87 で、単一行のコメント文のエラー解析にて一部に発生する不具合の修正。 注釈の入れ子の禁止。（23 ページ）	村岡如竹
Rev.1.1.7	2020.01.10	CASL87 で Ex Include 文の実行に於いて、指定のファイル内に複数の Circuit 文がある場合、エラーとなっていたバグを修正。	村岡如竹
Rev.1.1.8	2020.01.22	CASL87 の Include 文、Ex Include 文で、相対パスでの指定を可能にした修正。（26、28 ページ）	村岡如竹
Rev.1.1.9	2020.04.15	CASL87 の計算結果が数値範囲 1E-310～1E+300 の範囲外になった場合の注意表示の追加。（35 ページ）	村岡如竹
Rev.1.1.10	2020.05.21	64 ビット版 CASL87 Ver2.00 のリリースに伴う改変事項の追加。 （4、5、6、7、8、20、29、30、31 ページ）	村岡如竹
Rev.1.1.11	2020.06.11	「64 ビット版 CASL87 では、8G バイト、32 ビット版 CASL87 では、4G バイトのメモリ（RAM）を必要とする」に変更。 （4、5、6、7、8、20、29、30、31 ページ）	村岡如竹
Rev.1.1.12	2020.07.14	相互コンダクタンスの電流方向のノード記述のエラー検出でソースコードのライン番号の不具合を修正。	村岡如竹
Rev.1.1.13	2020.09.27	Include 文や Ex Include 文での階層の深さ制限を階層ファイル数の制限に変更。（26、29 ページ） 64 ビット版の起動時のウェイト表示追加。（12 ページ） エラー文の追加（31～35 ページ）	村岡如竹
Rev.2.0.00	2022.09.27	デジタル・フィルタ用 Z 変換素子 Z^{-1} 、を追加。（23 ページ） 遅延素子 T を追加。（31 ページ） CASL87 Ver.3.00、GR87 Ver.3.00 用に Rev.2.0.0 を発行。	村岡如竹
Rev.2.0.1	2024.09.01	Windows11 を追加明記。（4 ページ） フリー版のダウンロード URL の変更。（5 ページ）	村岡如竹
Rev.2.0.2	2024.12.12	Windows11/10、64bitCPU に特化。 浮動小数点の範囲を 5E-324～1.7E+308 に拡張。 計算時間の表示機能を時間、分、秒、ミリ秒に改変。	村岡如竹

目次

■はじめに	3
■特徴	4
■使用上の注意	5
■免責事項	5
■フリー版のインストール	6
■有償版のインストール（有償版への移行）	11
■CASL87 使用方法	12
■GR87 使用方法	15
■文法	20
1. 構成	20
2. 回路ブロックの記述	21
3. Calculation 文の記述	36
4. End.宣言	37
5. Include 文	37
6. Ex Include 文	39
7. 文法や機能の注意事項	41
8. CASL87 のエラー項目	42
実行環境に関するエラー：	42
ファイルに関するエラー：	42
Circuit 文に関するエラー：	42
Circuit 文や Calculation 文に共通するエラー：	46
Calculation 文に関するエラー：	47
計算実行時のエラー：	49
9. GR87 のエラー項目	50
ファイルに関するエラー：	50
各種設定に関するエラー：	50
10. 実際の回路例	52
■本アプリケーションのお問い合わせ先	56

■はじめに

電子回路や物理学上の力学などを等価回路に置き換えて解析する手法は昔からの技術として定着しています。微分積分を必要とする計算は、「微分の素子」、「積分の素子」、つまり電気回路の各素子に置き替えると、どんなに複雑で巨大な規模のシステムでも、いとも簡単に解くことができます。特にフィードバック系の安定性の評価はボーデ線図としてオープン・ループのゲインと位相の周波数特性で解析できます。

CASL87 及びグラフ・プロッターの GR87 はこのような解析をサポートするツールとして開発されました。

CASL87 及び GR87 を使用して周波数特性を得る作業の概念を以下のように表します。

「Circuit description」はその名のとおりに「回路記述」で、回路を表すソース・コードです。任意のテキスト・エディターで表記できます。回路記述のソース・コードについては文法など後述にて説明します。回路のソース・コード（複数含む）を CASL87 にドラッグ&ドロップして入力し、実行すると、回路のソース・コードと同じ名称で、サフィックスの異なる計算結果ファイルが作成されます。

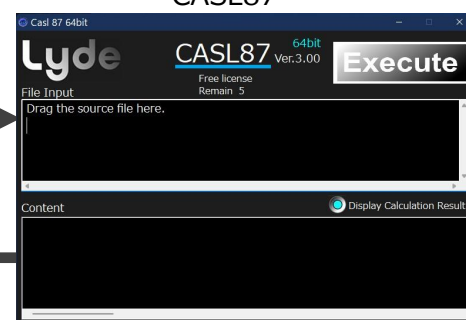
この計算結果ファイルをグラフ・プロッターの GR87 にドラッグ&ドロップして実行ボタンをクリックすると、周波数に対するゲインと位相をプロットしたグラフが表示されます。

回路ブロックを記述した
ソース・コード・ファイル

Circuit
description
Suffix = .CAS

1. Input to CASL87

CASL87



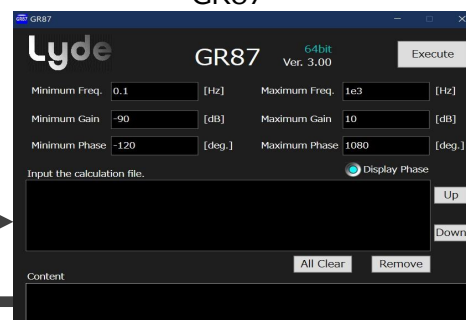
2. Create calculation result file

計算結果ファイル

Calculation
result
Suffix = .CAL

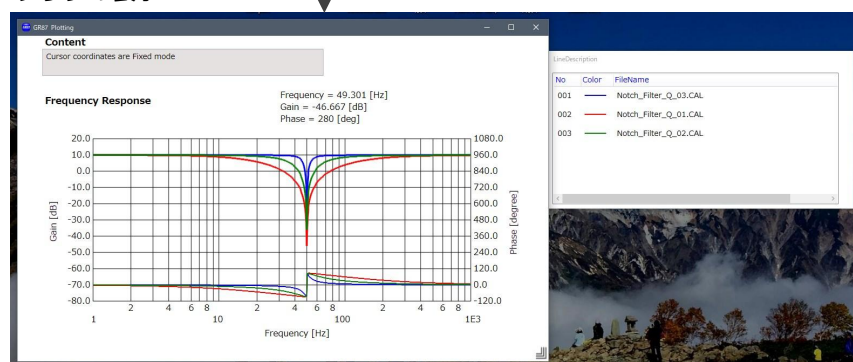
3. Input to GR87

GR87



4. Display graph characteristics

グラフの表示



CASL87、GR87 の使用の流れ（概要）

■特徴

CASL87 及び GR87 は大規模な複素マトリックスの演算を中心にして、回路の周波数特性を表示することを基本としますが、さらに以下に記述する特徴をもっています。

1. 計算専用の CASL87 とグラフ・プロッター専用の GR87 に役割を分けたことにより、周波数特性の異なる複数のソース・コード・ファイルとその複数の計算結果ファイルを同じグラフ座標に一挙に表示できるので、設計時の回路の各素子定数の比較検討が容易である。
2. CASL87 はテキスト・ファイルに記述した回路のノードを中心としたわかり易い表記で巨大な回路を計算できる専用の言語コンパイラである。CASL87 Ver.3.01 より、64 ビット版では最大ノード数 29,999、そのため、ノードに指定できる数字は 64 ビット版では 1~29,999 で順不同で指定できる（ゼロは使用禁止）。ただし、グランドは'E'や'e'を指定。
3. CASL87 でのソース・ファイルは回路をブロック化でき、階層構造にできるために、回路ブロックを他の回路に流用できる。（最深 512 層まで可能）
4. Include 文によって、他の回路のソース・コードを隠ぺいしたままライブラリーとして内包できる。内包されたソース・コードはさらに階層的に Include 文を宣言できる。（最深 512 層まで可能）
5. CASL87 は、周波数に対するゲインと位相を求めることができる。位相に関してはオフセット値を設定できる。そのため、負帰還回路のオープン・ループの位相特性を -180 度をゼロ度としてシフトし、発振などの安定性評価をできるようにしてある。
6. CASL87 及び、GR87 とともに、複数（最大 1024 ファイル）の回路を一挙に処理できるため、個別の部品の定数（パラメータ）を違えた場合の比較表示ができる。GR87 はそれぞれのファイルの特性を色別で表示できる。
7. CASL87 及び GR87 は複数のファイルを一度にドラッグ&ドロップして簡単に入力できる。それぞれ、最大 1024 の複数ファイルを一挙に扱うことができる。
8. GR87 は表示する周波数特性のウィンドウ・サイズをマウスで変更できる。
9. GR87 はカーソル・ポイントでグラフの値をピンポイントで表示できる。
10. GR87 は位相特性の表示／非表示の切り替えができる。

■使用上の注意

1. セットアップできる OS は Windows10/11(64 ビット CPU)です。さらに、.NET Framework 4.6 以上であることをご確認ください。尚、.NET Framework をインストールする場合は、以下の URL よりダウンロードしてください。

[https://msdn.microsoft.com/ja-jp/library/5a4x27ek\(v=vs.110\).aspx](https://msdn.microsoft.com/ja-jp/library/5a4x27ek(v=vs.110).aspx)

セットアップ時に各 OS やユーザーによるセキュリティの設定の違いにより、色々と確認画面が出てきますので、その都度、対応してください。

また、CASL87 Ver3.01 からは、64 ビット Windows 版のみ対応とします。インストールしようとする PC に合わせてセットアップしてください。

尚、64 ビット Windows 版の CASL87 では、8G バイト以上の実装 RAM を必要とします。

2. CASL87 のフリー版では、使用制限回数が 5 回に限られ、それ以上は有償版を購入して頂くことによりご利用できることとなります。ご所有の同じ PC に複数回、フリー版をダウンロードしても、前の使用制限回数が有効ですので残りの回数のみご利用できます。尚、GR87 は CASL87 とは異なり、フリーです。
3. CASL87 のフリー版では、LAN コントローラデバイスの MAC アドレスで管理していますので、同じセットアップ・フォルダーを他のパーソナルコンピュータにコピーしてセットアップすることはできません。他のパーソナル・コンピュータには再度フリー版のセットアップ・ソフトをダウンロードしてセットアップしてください。
4. 有償版でもフリー版同様、LAN コントローラデバイスの MAC アドレスで管理していますので、LAN コントローラデバイスの入れ替え含め、他のパーソナルコンピュータへの乗換えはできません。
5. CASL87 を使用するときは、必ず LAN (WiFi 含む) を稼働させておくことが必要です。

■免責事項

1. CASL87、GR87 の使用において、PC システムのトラブル発生には一切、対応しないことをご了承いただきます。
2. バージョンアップなど、ユーザーに告知することなく内容を変更されることがあります。

■フリー版のインストール

1. フリー版は、以下の URL より、“CASL87”、“GR87”をダウンロードできます。

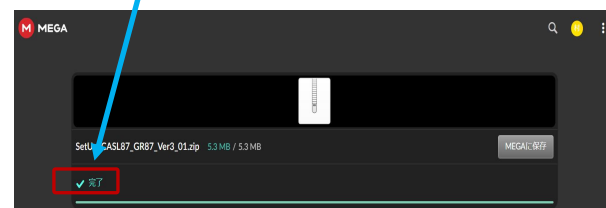
<https://sites.google.com/a/lyde-global.com/casl87/>

2. 上記の URL によるダウンロード画面では右図に示すダウンロード指示が出ますので、緑色の“ダウンロード”をクリックしてください。

クリック

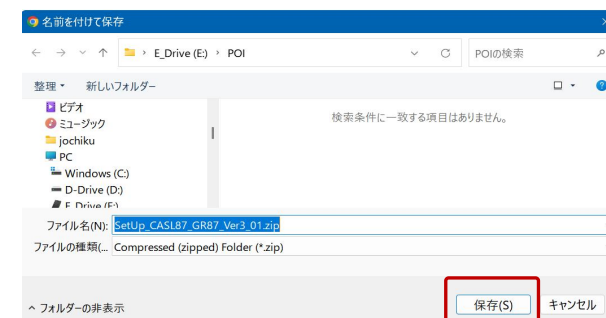


ダウンロードの完了を待つ



次に右図のようにダウンロード中の画面が出て、左下の赤枠の部分が「完了」となったら、ファイル名称“SetUp_CASL87_GR87_VerXXX.zip”の保存を促すエクスプローラーが表れますので、保存したいフォルダの場所を操作して、「保存ボタン」をクリックしてください。

尚、ダウンロード画面のデザインや操作方法は逐次、変わっていきますので、随時、指示に従ってください

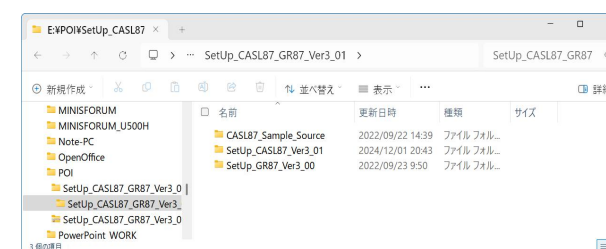


3. 保存したフォルダ先にある Zip ファイル

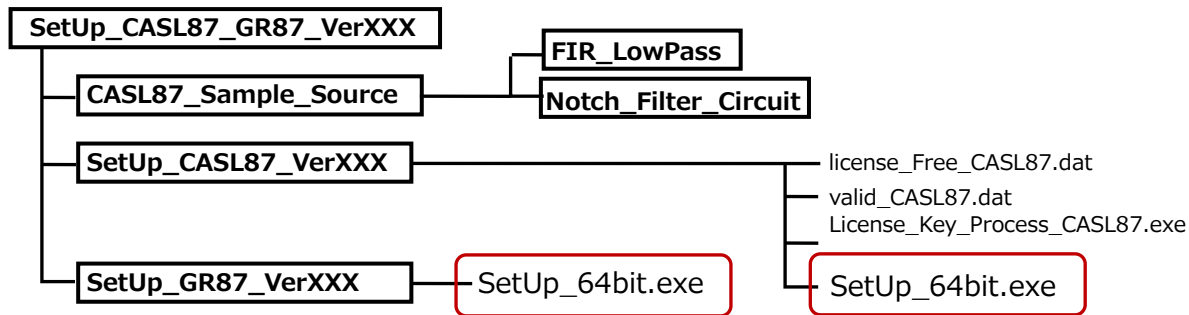
“SetUp_CASL87_GR87_VerXXX.zip”にマウスでカーソルをあて、マウスの右ボタンをクリックして「すべて展開」にカーソルを合わせてクリックしてください。



4. 右図のようにフォルダ“SetUp_CASL87_GR87_VerXXX”が表れ、そのフォルダをクリックすると、“CASL87_Sample_Source”と“SetUp_CASL87_VerXXX”及び、“SetUp_GR87_VerXXX”が展開されます。

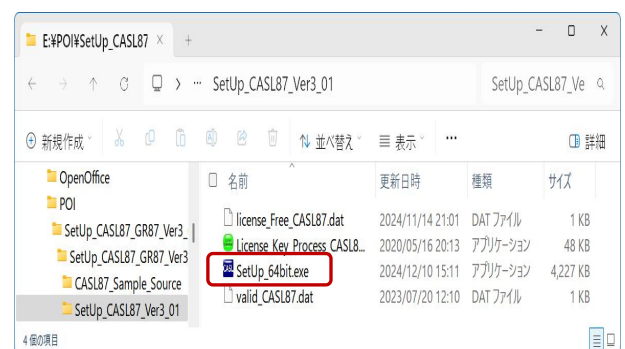


このフォルダ構造の詳細は以下の図のようになります。



5. 最初にフォルダ“SetUp_CASL87_VerXXX”の SetUp_64bit をダブルクリックして、CASL87 からインストールします。

“SetUp_64bit.exe”をダブルクリックします。



6. OS の違いで警告画面が異なりますが、Windows10 の場合、「Windows によって PC が保護されました」というセキュリティ画面が表示されます。右図に示すように“詳細情報”をクリックして進めてください。

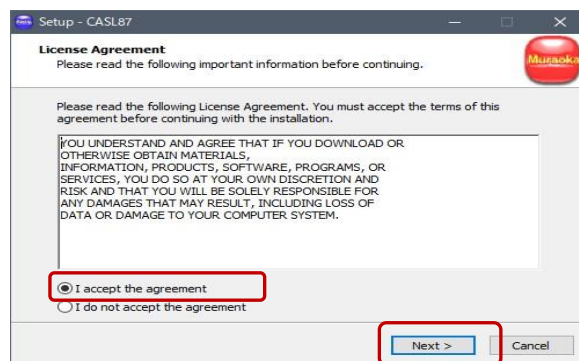
さらに右下図に示す“実行”ボタン をクリックしてください。



7. 最初に枠内に表示される免責事項に同意するなら、ラジオボタン“**I accept the agreement**”をチェックして、“**Next>**”ボタンをクリックします。

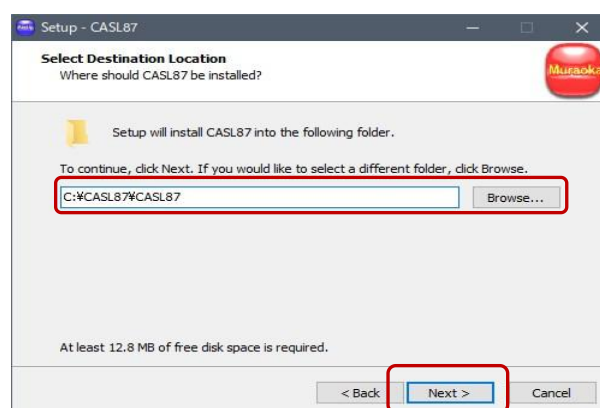
尚、英文の免責事項は以下のとおりです。

「お客様は、資料、情報、製品、ソフトウェア、プログラム、またはサービスをダウンロードもしくはその他の手段で取得する場合は、お客様ご自身の判断および責任において行っていただくこと、また、その結果として発生するデータの損失またはお客様のコンピューター・システムへの損傷などのいかなる損害もすべてお客様の責任となることを理解し、同意するものとします。」

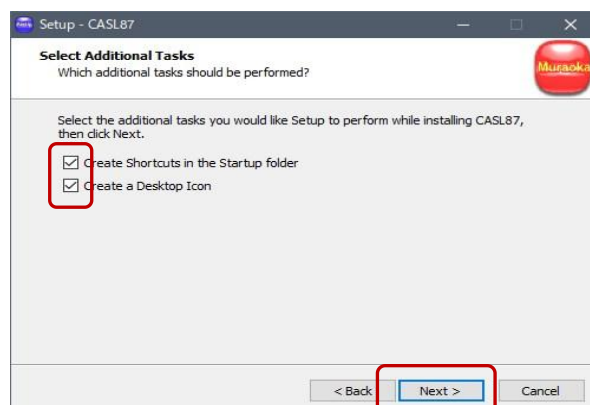


8. インストール先を確認、或いは“**Browse**”ボタンで変更して、“**Next>**”ボタンをクリックしてください。

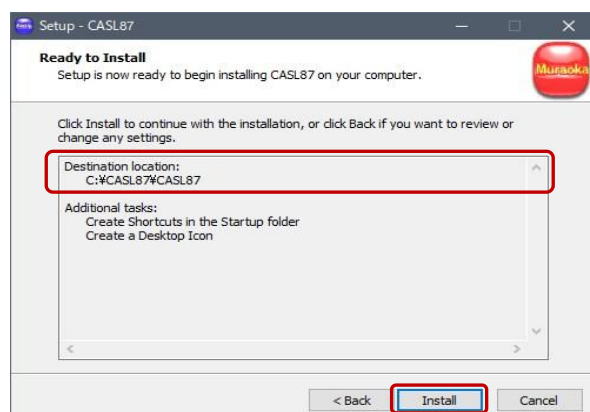
尚、CASL87 は実行時にインストール先のフォルダ内に、データ・ファイルを作成しますので、Windows のアプリケーション・ファイルのフォルダ“Program Files”や“Program Files(x86)”はセキュリティ対策でガードされているため、適しません。右に示したように表示されたインストール先か、Windows のアプリケーション・ファイルのフォルダ以外のフォルダを指定してください。右の例では“CASL87”フォルダの下に“CASL87”フォルダを作成しようとしています。フォルダ構成は自由に設定してもかまいません。



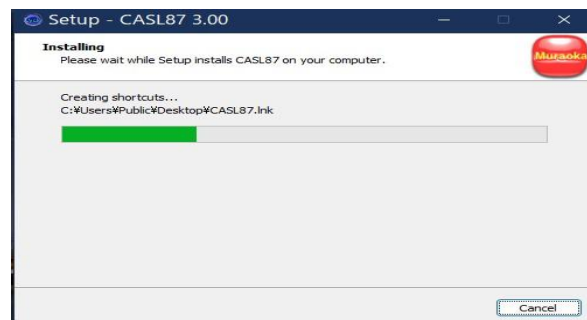
9. “**スタートアップメニューにショートカットを作成**”、及び“**デスクトップにアイコンを作成**”のラジオボタンのチェックマークを確認して、“**Next>**”ボタンをクリックします。



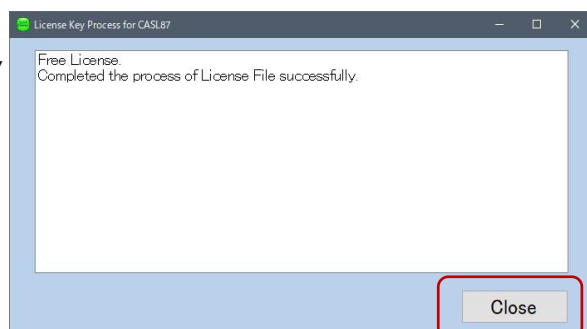
10. “**Destination location**”下のインストール先を確認して、“**Install**”ボタンをクリックします。



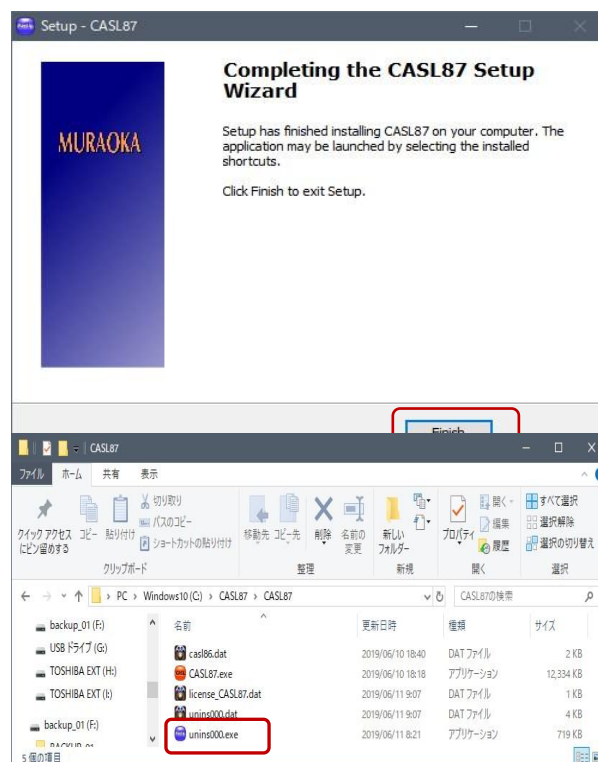
この後、次のようなウィンドウ画面が出て、なかなか先に進みませんが、5分程度お待ちください。



11. “License_Key_Process for CASL87”の画面が現れるので、“Completed the process of License File successfully”の表示を確認したら、“Close”ボタンをクリックしてください。



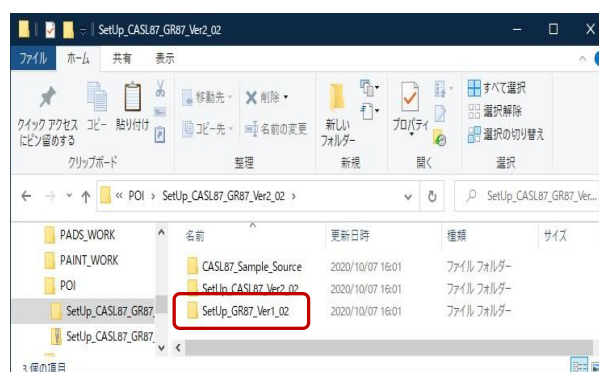
12. “Completing the CASL87 SetupWizard”の画面が現れるので、“Finish”ボタンをクリックしてインストールを完了してください。
この後、必ず、PCを再起動してください。



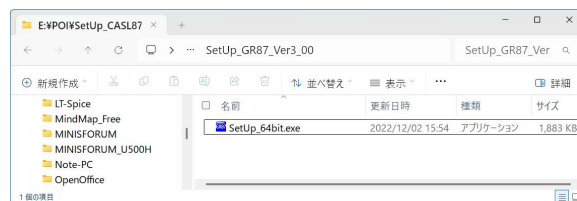
▼アンインストールに関して

アンインストールする場合は右図に示すインストール先のフォルダ内の“unins000.exe”をダブルクリックします。
後は画面の指示に従ってアンインストールしてください。

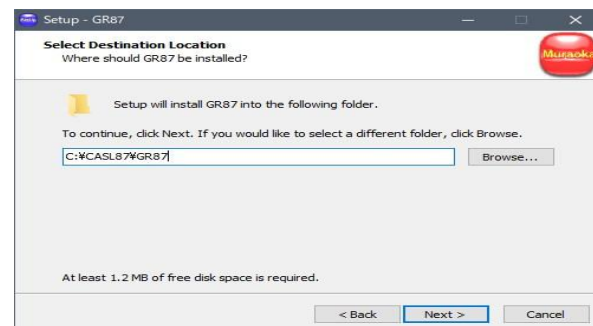
13. “GR87”のセットアップは先のダウンロードした圧縮ファイルを展開したフォルダに戻り、“SetUp_GR87_VerXXX”のフォルダを開きます。



14. “SetUp_64bit.exe”をダブル・クリックします。あとは、“CASL87”の場合と同様に進めて行きます。



15. ただし、右のようにインストール先のフォルダを確認するウィンドウが表示された場合、デフォルトでは“CASL87”フォルダの下に“GR87”フォルダが作成されます。“CASL87”同様、フォルダ構成は“C:¥”ドライブ以外の指定含め、自由に設定してもかまいません。



■有償版のインストール（有償版への移行）

最初から有償版を導入する場合でも、必ず、CASL87のフリー版のセットアップを完了してから行ってください。尚、GR87は、無償のまま使い続けることができます。

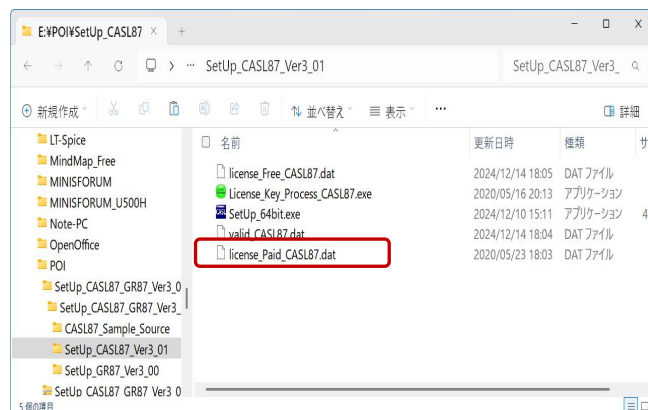
有償版のライセンスキー（license_Paid_CASL87.dat）は下記のURLのHPにアクセスし、【有効版への移行】の記述をお読みいただき購入してください。

<https://sites.google.com/a/lyde-global.com/casl87/>

価格は¥5,060（手数料+10%消費税込み）です。

尚、古いバージョンでご購入のライセンスキーをお持ちの方は新しいバージョンにも対応しますので、新規にご購入する必要はありません。

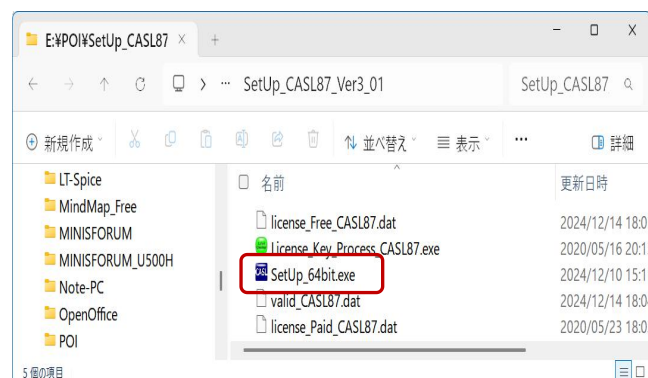
1. 得られたライセンスキー **license_Paid_CASL87.dat** を右図のようにフリー版のセットアップ用フォルダに入れます。



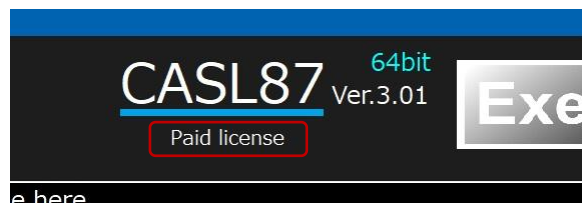
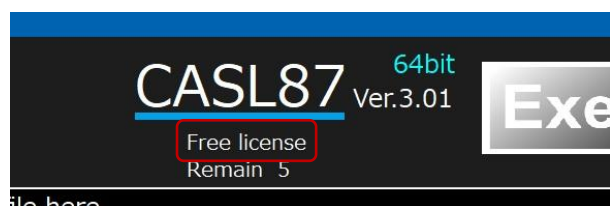
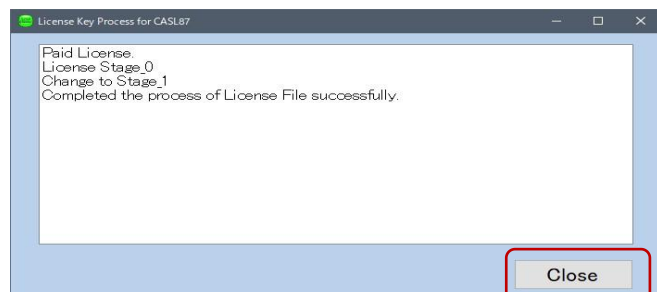
2. フリー版のセットアップと同様に“**SetUp_64bit.exe**”をマウスの左ボタンのダブルクリックして、再度、セットアップしてください。

このセットアップ時に以前のインストール先フォルダに“CASL87.exe”が残っていてもかまいませんので、そのままフリー版のセットアップと同様の作業を行ってください。

尚、ライセンス・キーのチェック時には右下のウィンドウが表示されます。




すべてのインストール終了後に、‘CASL87’を起動すると下に示すように‘Free License’が‘Paid License’に変わっていることをご確認ください。

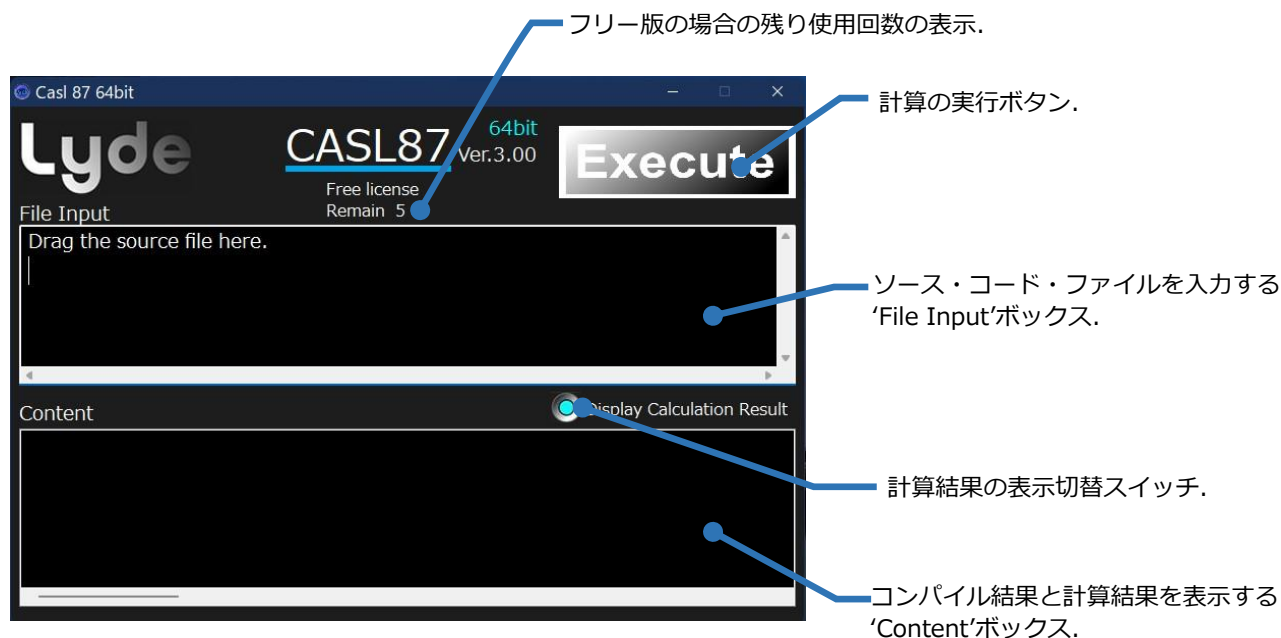


有償版のライセンスキー（license_Paid_CASL87.dat）は他のPCへのコピーは無効です。

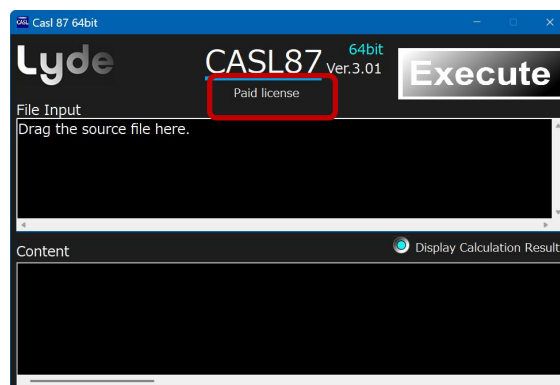
■CASL87 使用方法

1. 起動

インストール後に、Windows 上の画面にデスクトップ・アイコン  が表示されているはずですので、マウスの左ボタンでダブル・クリックしてください。下図に示す CASL87 のウィンドウが表示されます。（スタートアップメニューからも起動できます。）



尚、64 ビット版では、起動の段階で多くのメモリー空間を取得します。その為、右に示すように、'File Input'ボックスに'Please wait until the system is ready.'と表示されますので、'Drag the source file here.'と表示されるまで操作をお待ちください。



2. 操作

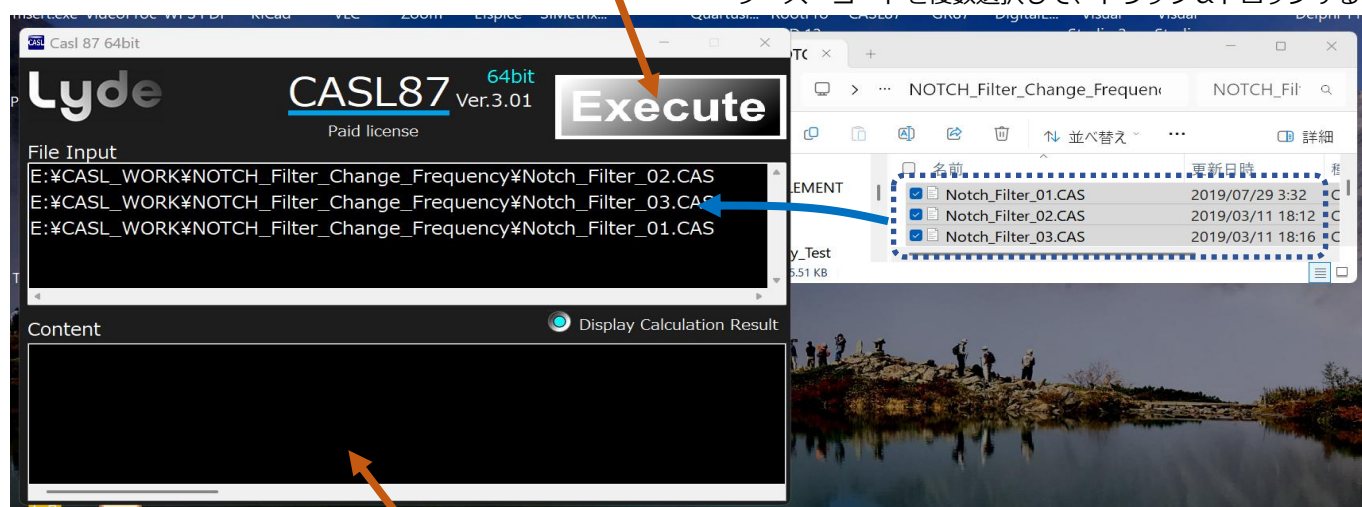
ソース・コード・ファイルを入力する'File Input'ボックス :

このボックスに CASL87 のソース・コードをフォルダ内を表示したエクスプローラからドラッグ&ドロップして入力します。複数選択したソース・コードも一挙に入力できます。最大 1024 のファイルを入力できます。入力した複数のソース・コード・ファイルはボックスの垂直水平スクロールバーで確認できます。

計算の実行ボタン :

ソース・コードの入力後、このボタンをクリックして計算を開始します。

ソース・コードを複数選択して、ドラッグ&ドロップする。

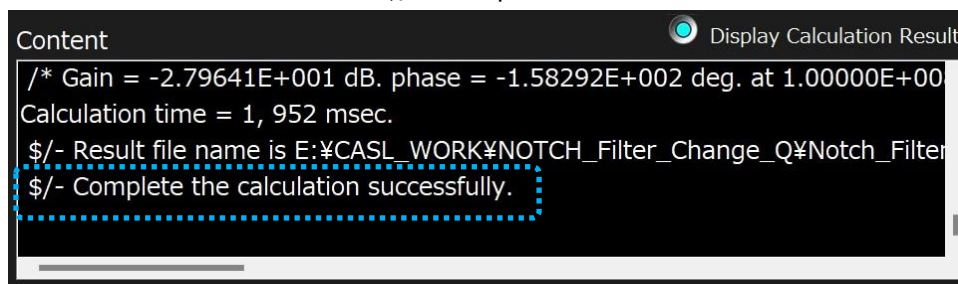


計算に成功したら '\$/- Complete the calculation successfully.' と表示される。

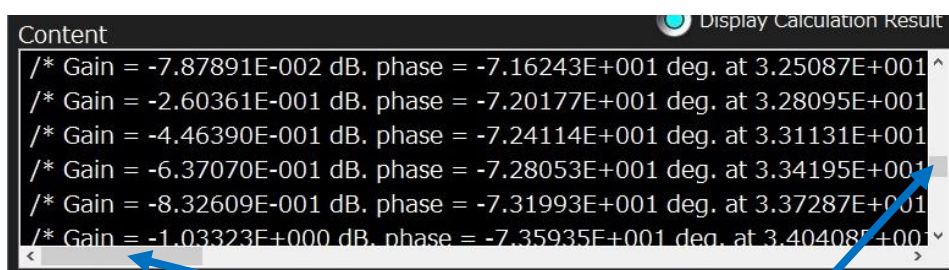
コンパイル結果と計算結果を表示する'Content'ボックス :

この'Content'ボックスはソース・コードのコンパイル結果と計算結果を表示します。ソースコードに間違いがあると、エラー個所とエラー内容を表示して処理を中止します。

成功したし場合は下の例のように、 '\$/- Complete the calculation.' と表示してストップします。



計算結果は以下のように垂直水平スクロールを使って確認できます。



水平スクロール.

垂直スクロール.

計算結果の表示切替スイッチ：

計算結果を‘Content’ボックスに逐次表示すると時間を取られてしまいます。このスイッチはトグルになっており、マウスの左ボタンのクリックで‘表示’、‘非表示’が切り替わります。‘非表示’にすると各周波数での計算結果を表示しないので、処理が速くなります。

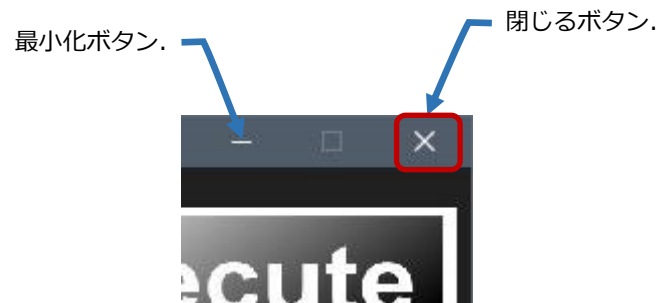
フリー版の場合の残り使用回数の表示：

フリー版では使用回数は5回に制限されています。この表示が0（ゼロ）になれば使用できません。それ以上の使用は有償版のキーを購入して頂くことになります。

尚、このカウントはコンパイルが成功した場合のみ有効であり、失敗した場合はカウントされません。また、複数のソース・コードを一括で計算する場合もカウントは1回アップするだけです。


3. 終了

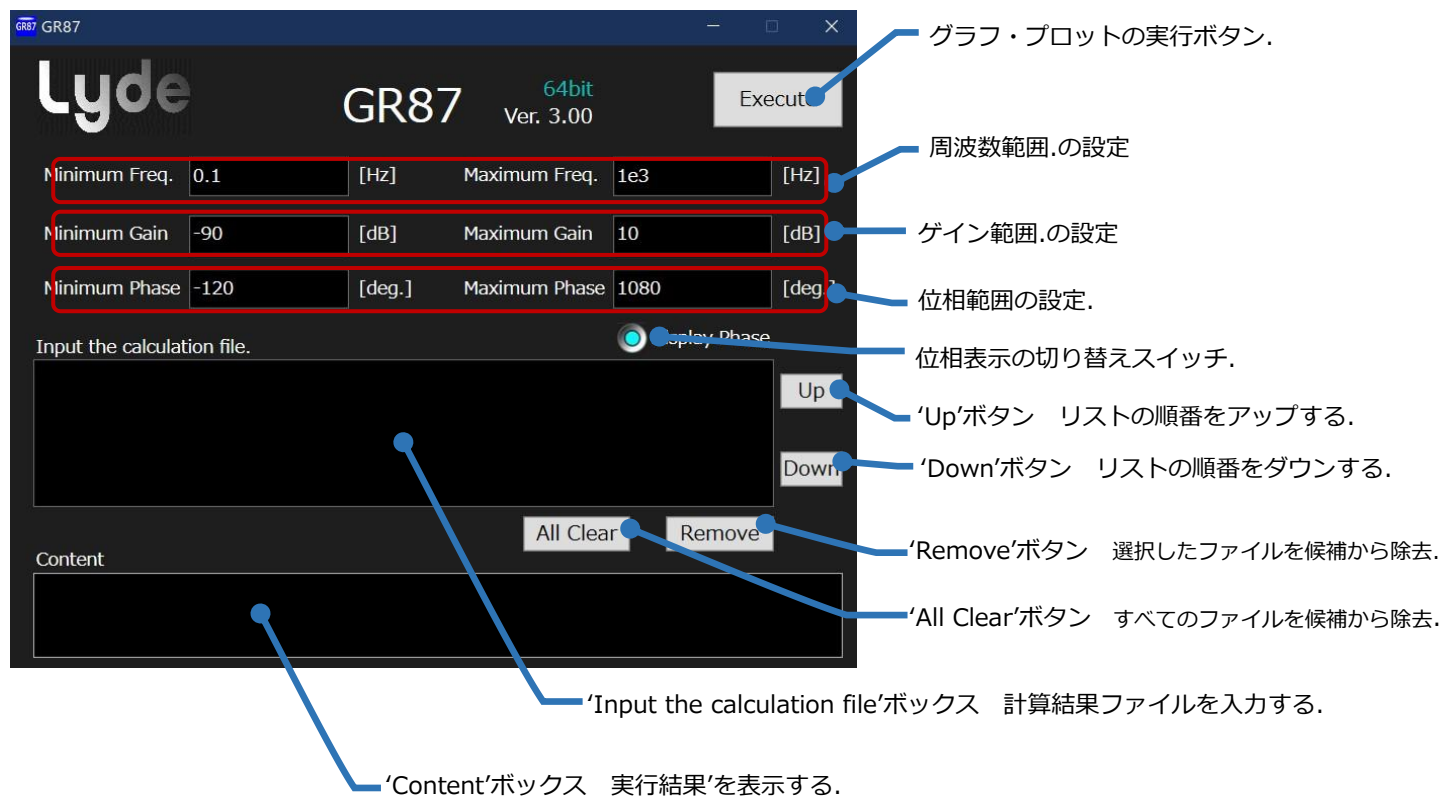
右下の図に示すように‘閉じるボタン’をクリックすると CASL87 は終了します。‘最大化ボタン’は無効であり、‘最小化ボタン’は有効です。



■ GR87 使用方法

1. 起動

インストール後に、Windows 上の画面にデスクトップ・アイコン  が表示されているはずですので、マウスの左ボタンでダブル・クリックしてください。下図に示す GR87 のウィンドウが表示されます。
(スタートアップメニューからも起動できます。)

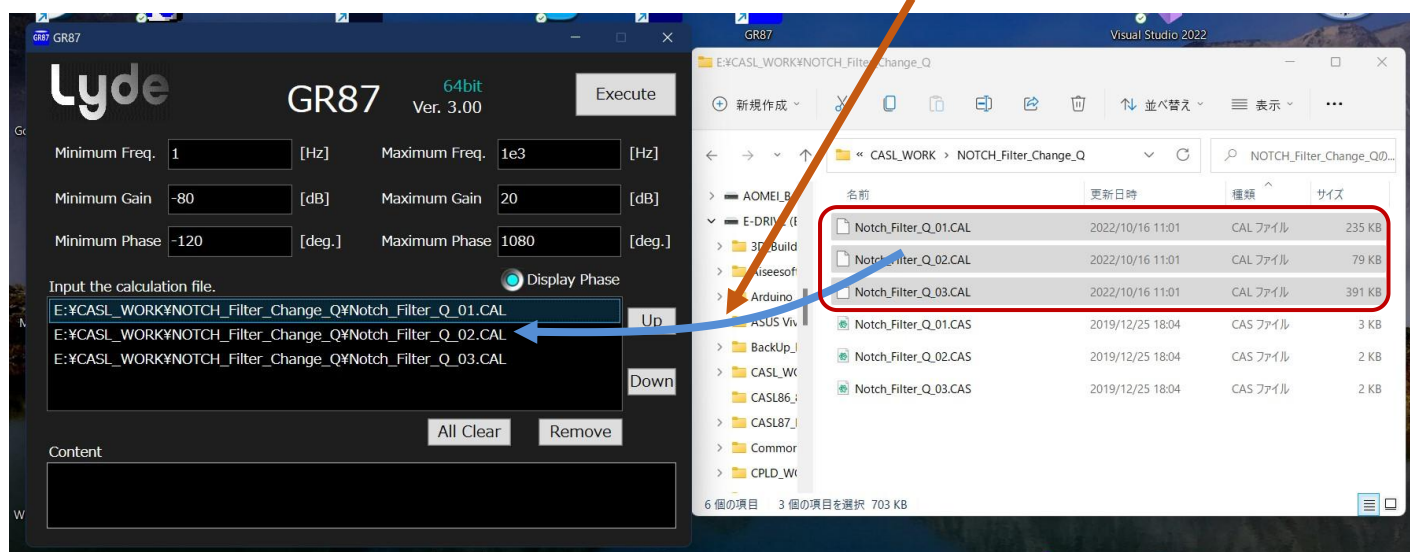


2. 操作

‘Input the lcalculation file’ボックス 計算結果ファイルを入力する：

以下に示すように、CASL87 によって出力された計算結果ファイル（サフィックス=.CAL）をフォルダ内に表示したエクスプローラからドラッグ＆ドロップでこのボックスに入力します。

計算結果ファイルを複数選択して、ドラッグ＆ドロップする。



計算結果ファイルがエクスプローラ内で複数選択された場合は一挙にこのボックスに入力できます。

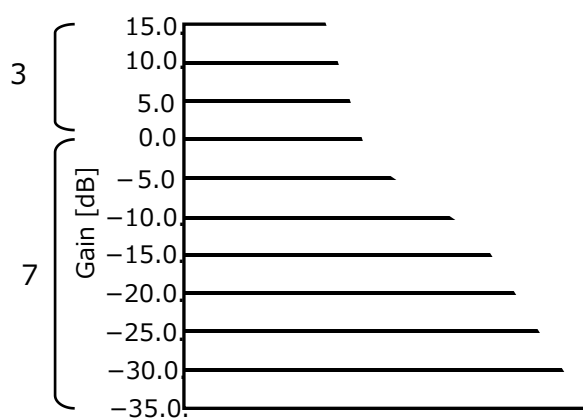
周波数範囲の設定：

表示する最小周波数'Minimum Freq.'と最大周波数'Maximum Freq.'の範囲を設定します。値は、ヘルツ[Hz]で浮動小数点表記か、指数表記で行ってください。また、値は、0.01、1、10、1e3 などのように、必ず、10 のべき乗にしてください。10 のべき乗以外は禁止です。

ゲイン範囲の設定：

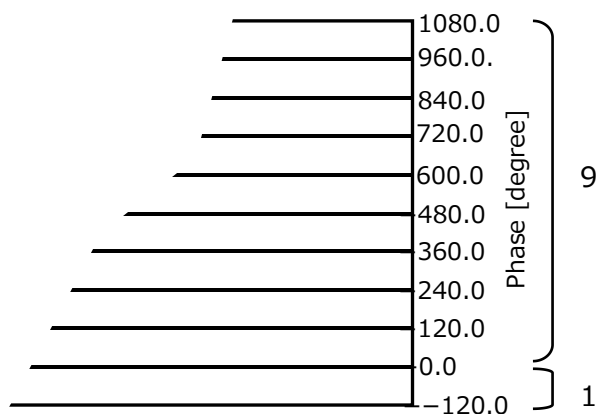
表示する最大ゲインと最小ゲインの範囲を設定します。単位はデシベル[dB]です。因みに 10 倍は 20dB です。値は、浮動小数点表記か、指数表記で行ってください。

ゲインの縦軸ステップは 10 段階に固定です。0 dB値が横軸線となることを条件とすると、以下のように 0dB を起点とする縦座標を 3 対 7 に分け、ステップ当たりのデシベルを決めます。以下の場合 5dB です。最大値は 15dB、最小値は -35dB になります。

**位相範囲の設定：**

表示する最大位相と最小位相の範囲を設定します。単位は度[degree]です。値は、浮動小数点表記か、指数表記で行ってください。

位相の縦軸ステップはゲイン同様 10 段階に固定です。0 度値が横軸線となることを条件とすると、以下のように 0 度を起点とする縦座標を 9 対 1 に分け、ステップ当たりの位相を決めます。以下の場合 120 度ですので、最大値は 1080 度、最小値は -120 度になります。



位相表示の切り替えスイッチ：

グラフ・プロット時に位相特性を表示するか否かを設定します。このブルーのトグル・スイッチをマウスの左クリックで操作します。

‘Up’ボタン リストの順番をアップする：

複数の計算結果ファイルをドラッグ&ドロップで入力した場合に、GR87は‘Input the calculation file’ボックスの最上位置から順番にグラフ番号を付けて表示します。このボックス内でマウスの左ボタンで選択した計算結果ファイルを‘Up’ボタンで順番を繰り上げていくことができます。

‘Down’ボタン リストの順番をダウンする：

上記の‘Up’ボタン同様、複数の計算結果ファイルをドラッグ&ドロップで入力した場合に、このボックス内でマウスの左ボタンで選択した計算結果ファイルを‘Down’ボタンで順番を繰り下げていくことができます。

‘Remove’ボタン 選択したファイルを候補から除去：

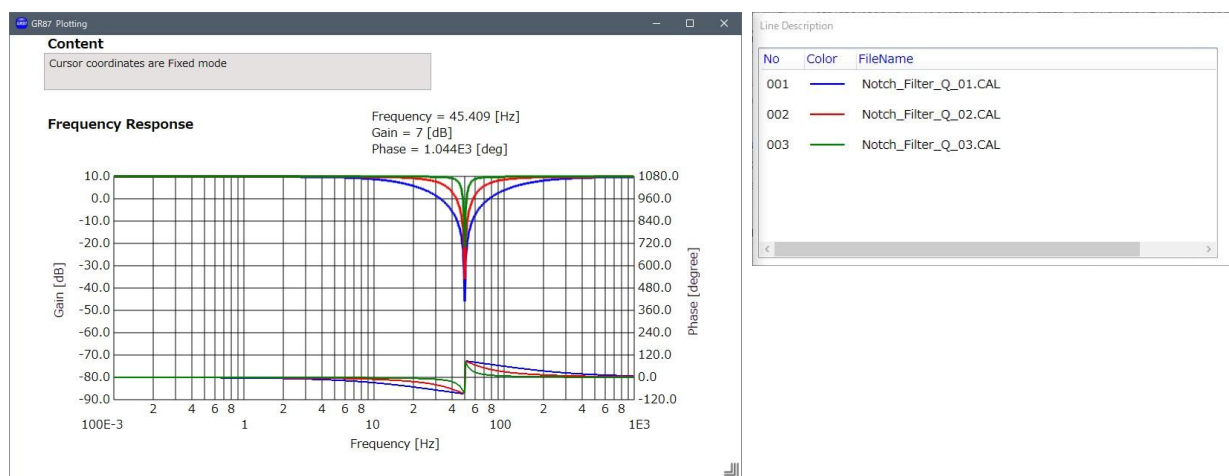
‘Input the calculation file’ボックス内でマウスの左ボタンで選択した計算結果ファイルをこの‘Remove’ボタンで候補から除去することができます。

‘All Clear’ボタン 選択したファイルを候補から除去：

‘Input the calculation file’ボックス内のすべての計算結果ファイルをこの‘All Clear’ボタンで除去することができます。

グラフ・プロットの実行ボタン：

‘Input the calculation file’ボックス内への計算結果ファイルの入力と上記の各種設定を終えたなら、この実行ボタンをクリックすると、プロットを開始して表示します。以下は表示例です。左下のウィンドウは同じ回路の素子定数を変えてQ値の違いによるノッチ・フィルターの周波数特性です。右下のウィンドウはグラフの線の色で分けた計算結果ファイルの番号がリストとして表示しています。



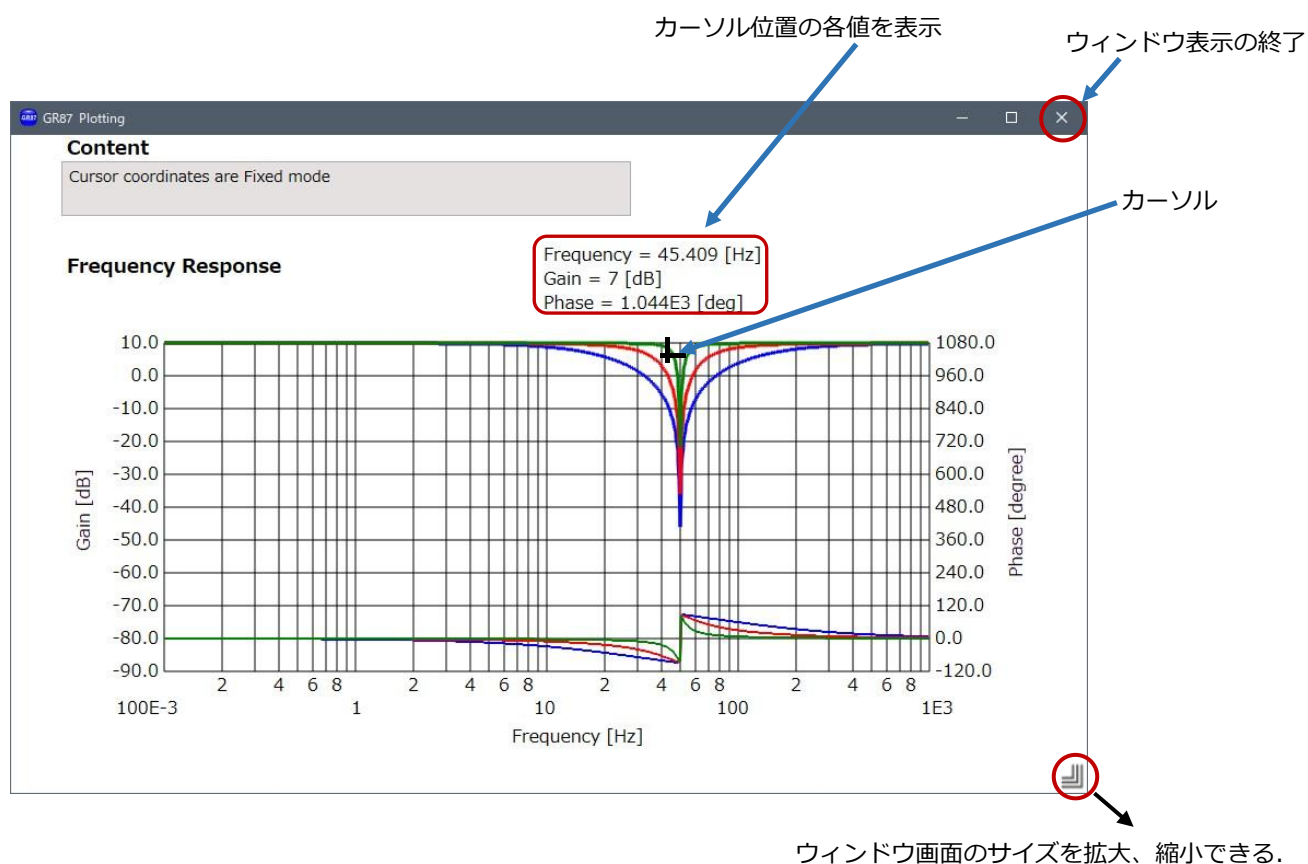
3. グラフ・プロッターの操作

グラフ・プロッターの実行ボタンをクリックすると下のような周波数特性がプロットされたウィンドウが表示されます。（計算結果ファイルのリスト・ウィンドウも同時に表示されますが、ここでは説明を省きます）

カーソル位置の各値を表示：

マウスによって+マークのカーソルをウィンドウ内で動かすと、その位置の周波数、ゲイン、位相のそれぞれの値が下図の赤枠の位置に表示されます。

例えば、カーソルをグラフ上のあるポイントに合わせ、マウスの左ボタンをクリックするとその周波数、ゲイン、位相の各値が固定表示されます。このあと、マウスの左ボタンを再度クリックすると動的表示に戻ります。



ウィンドウ画面のサイズを拡大、縮小できる：

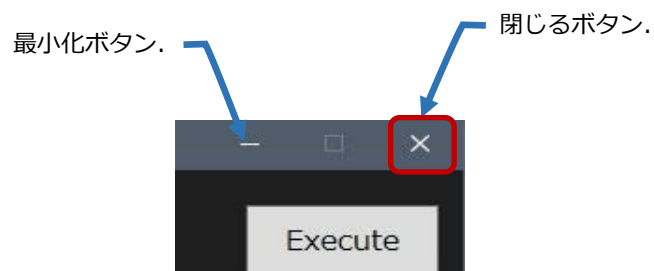
ウィンドウの右下の マークの右下にカーソルを当て、カーソルが に変化したら、マウスの左ボタンによるドラッグでウィンドウ画面のサイズを変更できます。ただし、計算結果ファイル数が増えるとサイズ変更時のスピードが遅くなります。

ウィンドウ表示の終了：

ウィンドウの右上のXマークをクリックして終了します。終了すると GR87 ウィンドウに操作が戻ります。

4. グラフ・プロッターの終了

右下の図に示すように‘閉じるボタン’をクリックすると GR87 は終了します。‘最大化ボタン’は無効であり、‘最小化ボタン’は有効です。



■文法

CASL87 用の回路記述ソース・コードの文法を説明します。ソースコード・ファイル名は

‘ソースコード名.CAS’

です。サフィックスは‘.CAS’です。

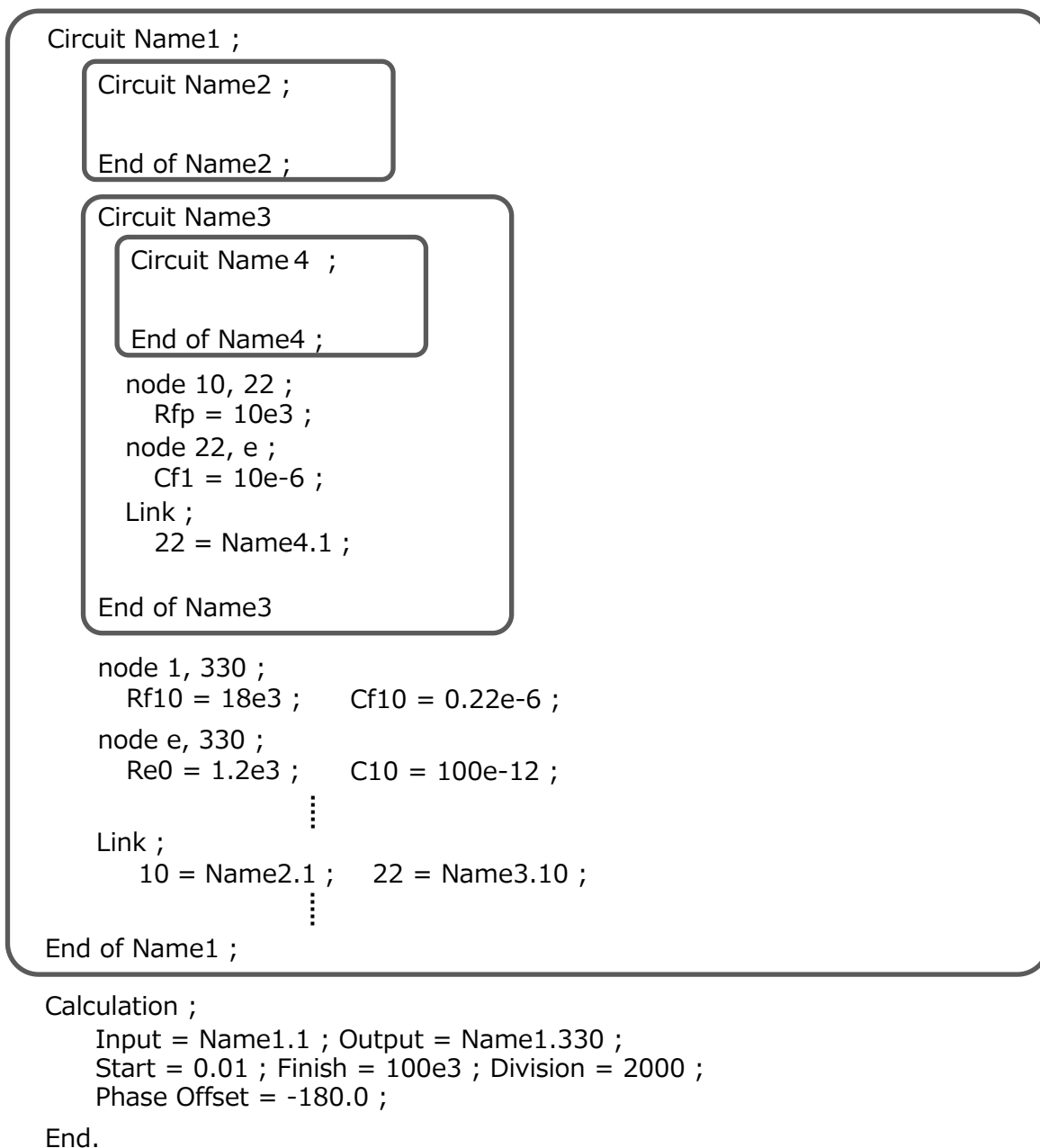
1. 構成

記述構成は以下のように回路ブロックを階層化して表現します。もちろん、回路ブロックは一層でもかまいません。全体回路の「Circuit Nmae1」及び、それを閉じる「End of Name1」の下に表記した「Calculation」で入力ノードや出力ノード、計算の周波数範囲、周波数範囲内の分解能、位相のオフセット値を設定できます。全体の終わりは「End.」で閉じます。

構成の特徴：

(ア)回路ブロック内のノード番号はローカライズされているため、異なる回路ブロック間では同じノード番号は干渉しない。そのため、回路ブロックごと、コピー＆ペーストが可能である。

(イ)回路ブロックの「Circuit ~(Name)」は最大 512 層の深さまで階層化でき、「Include」文の宣言でライブラリー化が可能である。



2. 回路ブロックの記述

回路ブロックは以下のように「Circuit ～（回路ブロック名）」で始まり、「End of ～（回路ブロック名）」で終わる構造です。この区間の中にノード番号と部品名及び、その値を記述します。

例) 右下の回路図は以下のように記述されます。

```
Circuit HighSideFilter ; // High frequency pass filter circuit.
```

```
{ High frequency pass filter.
```

```
Input node ---- 1, Output node ---- 3, Sub node(for the feedback) ---- 4 }
```

```
node 1, 2 ;
```

```
Chi1 = 0.047e-6 ;
```

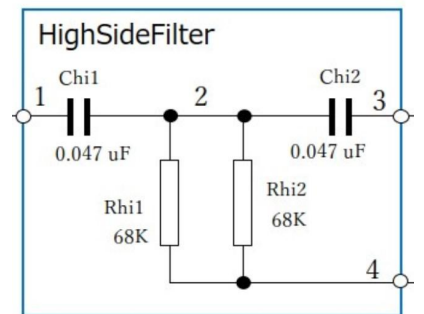
```
node 2, 4 ;
```

```
Rhi1 = 68e3 ; Rhi2 = 68e3 ;
```

```
node 2, 3 ;
```

```
Chi2 = 0.047e-6 ;
```

```
End of HighSideFilter ;
```



‘//’や‘{’、‘}’は注釈の表記です。注釈の表記については「注釈の記述」で詳細を説明します。

CASL87 では、この回路ブロックの中にさらに子回路ブロックを階層的に内包することもできます。階層の深さは最大 512 層です。

【ノードと素子定数の記述】

ノードと素子定数は以下のように記述します。

```
node 2, 4 ;
```

```
Rhi1 = 68e3 ; Rhi2 = 68e3 ;
```

上記はノード番号 2 と 4 の間に抵抗 Rhi1 (68K Ω) と Rhi2 (68K Ω) が並列に接続されていることを表します。抵抗の場合は頭文字が‘R’、キャパシターの場合は‘C’、インダクタでは、‘L’、相互コンダクタンスとしての電流源は‘G’です。さらに、Ver.300 から遅延素子が加わり、頭文字は‘T’です。

単位換算はそれぞれ、 Ω (ohm)、F (Farad)、H (Henry)、S (Siemens)、Sec (Second) で、値はすべて正の数値です。ただし、値の最後に単位表現をつけることは禁止です。

CASL87 ではこの 5 種類の部品しか扱いません。「電圧源がないではないか」と思われるかもしれませんが。電圧源は必要ありません。電圧源と等価の回路を組めばよいのです。次のページで電圧源の組み方を説明します。

ノードを表す‘node’を含め、すべての記述は大文字、小文字は問いません。ノードは、64 ビット版の CASL87 では 1～29,999 の範囲で順不同で指定できます。グランドは e (大文字の場合、E) です。グランドは他のノードと異なり、暗黙のうちにグランドどうしが共通で接続されている存在です。

部品の名称は頭文字が明確であれば、それに続く表記は無視されます。つまり、以下の例のように、抵抗として同じ部品名が並んでもかまいません。

```
node 2, 4 ;
```

```
Rhi1 = 68e3 ; Rhi1 = 68e3 ;
```

また、部品は、以下のように行替えでもかまいません。

```
node 2, 4 ;
```

```
Rhi1 = 68e3 ;
```

```
Rhi2 = 68e3 ;
```

必ず注意することは、表記の最後に区切りとして';'を付けなくてはならないことです。

CASL87 はノードを軸としてノードに集まるすべての電流の和はゼロという‘キルヒホッフの法則’に基づいて計算を行います。相互コンダクタンスとしての電流源によって様々な等価回路を表現できます。

【素子定数の説明】

〈抵抗〉

抵抗は頭文字に'R'あるいは'r'が先頭に付く名称で表し、値は整数、実数（指数表記も含む）です。単位換算は'Ω'（オーム）で正の値でなくてはなりません。

Rhi1 = 68000 ; Rhi2 = 68e3 ; // 68 KΩ

〈キャパシタ〉

キャパシタは頭文字に'C'あるいは'c'が先頭に付く名称で表し、値は整数、実数（指数表記も含む）です。単位換算は、'F'（ファラッド）で正の値でなくてはなりません。

Chi1 = 1e-6 ; Chi2 = 0.000001 ; // 1 uF

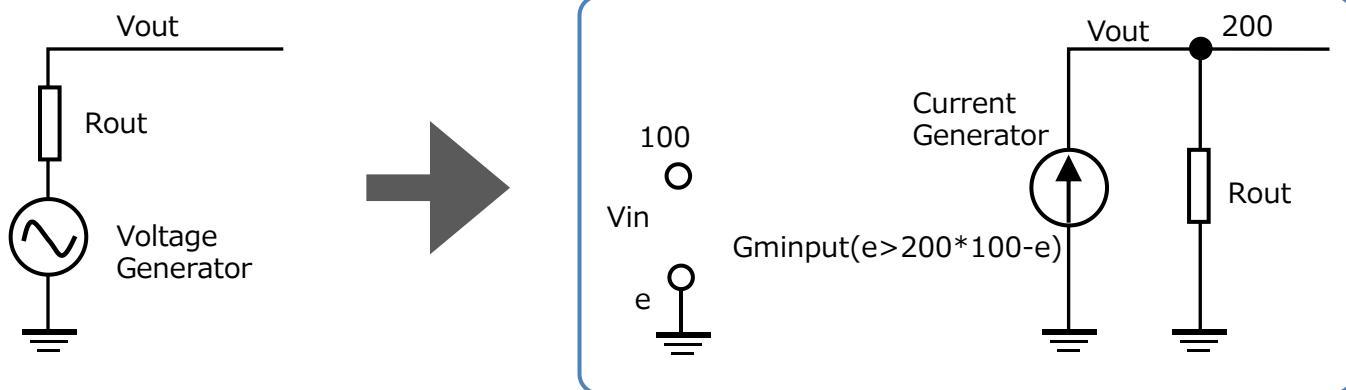
〈インダクタ〉

インダクタは頭文字に'L'あるいは'l'が先頭に付く名称で表し、値は整数、実数（指数表記も含む）です。単位換算は'H'（ヘンリー）で正の値でなくてはなりません。

Lhi1 = 1e-3 ; Lhi2 = 0.001 ; // 1 mH

〈電流源〉

CASL87 は交流信号解析のみを扱いますので直流は扱いません。また、電圧源も存在しません。電圧源は必ず電流源で構成させます。電圧（交流信号）源には必ず出力抵抗が付随します。下図では Rout がそれに当たります。



Rout が仮に 50Ω とすると、電流源（相互コンダクタンス） $G_{minput}(e>200*100-e)=0.02\text{ S}$ （ $1/R_{out}$ ）です。

相互コンダクタンスの表記は次のような法則に従ってください。また、この相互コンダクタンスの値は整数、実数（指数表記も含む）です。単位換算は‘シーメンス’または‘ひ’です。値に関しては、正、負のどちらでも適用可能です。

$G_{minput}(e>200*100-e) = 0.02 ;$

電流方向 コントロール電圧

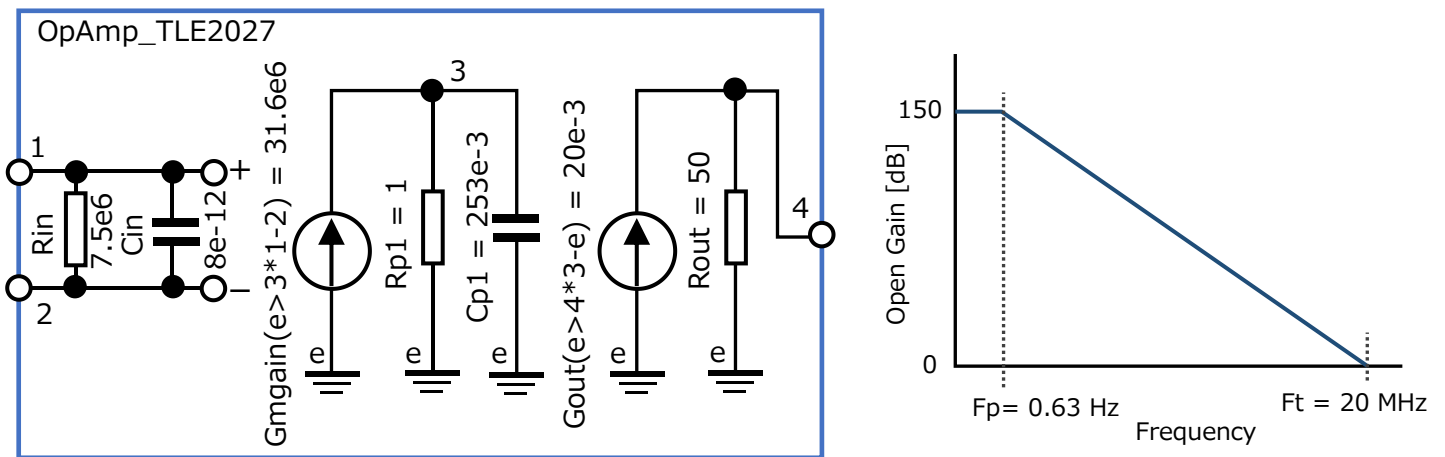
電流方向： 電流が流れる方向を示します。ここではグラウンド e からノード番号 200 に流れていることを示します。電流源が個別のノード間に存在するなら、例えば、**n1>n2** と表示します。方向を示す記号> は必ず右向きでなくてはなりません。

コントロール電圧： ノード間の電圧を表し、引き算形式で表示します。ここでは 100 - e でノード番号 100 が正側で e が負側になります。

電流方向 e>200 とコントロール電圧 100 - e の間には '*' (アスタリスク) が入ります。
この相互コンダクタンスヲ利用して、簡単なオペアンプの等価回路を表現してみます。

電流源を使ったオペアンプの等価回路の例

下図は TI 社のオペアンプ TLE2027 のオープンループ・ゲインの周波数特性です。低域周波数での最大オープンループ・ゲインは約 150dB なので $Rp1=1$ として、 $Gm_{gain}(E>3*1-2)=31.6e6$ になります。トランジション周波数 $Ft=$ 約 20MHz なので、オープンループゲイン特性中の第一ポール Fp は、 $20MHz / 31.6e6=0.63Hz$ になります。



第一ポール Fp を形成するための $Cp1$ の値は以下ようになります。

$$Cp1 = 1/(2\pi \times Fp \times Rp1) = 1/(2\pi \times 0.63 \times 1) = 253e-3 [F]$$

出力抵抗は TLE2027 の場合は約 50Ω ですので、出力段のノードの倍率を 1 として設定すると、電流源としての相互コンダクタンスは、以下ようになります。

$$Gout(E>4*3-e) = 20e-3 [S]$$

このオペアンプの等価回路は CASL87 用のソースコードとしては、以下のように記述できます。

```
Circuit OpAmp_TLE2027 ;
{ Operational amplifier TLE2027. Raw gain = 150 dB. Ft = 20 MHz.
  Input Resistor = 7.5 Mohm. Input Capacitor = 8 pF. Output Resistor = 25 ohm.
  1 ----- Input+, 2 ----- Input-, 4 ----- Output
  node 1, 2 ;
  Rin = 7.5e6 ; Cin = 8e-12 ;

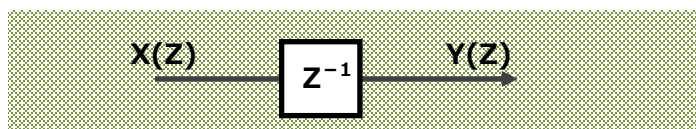
  node 3, e ;
  Gmgain(e>3*1-2) = 31.6e6 ; Rp1 = 1.0 ; Cp1 = 253e-3 ;

  node 4, e ;
  Gout(e>4*3-e) = 20e-3 ; Rout = 50.0 ;

End of OpAmp_TLE2027 ;
```

〈Z⁻¹素子〉

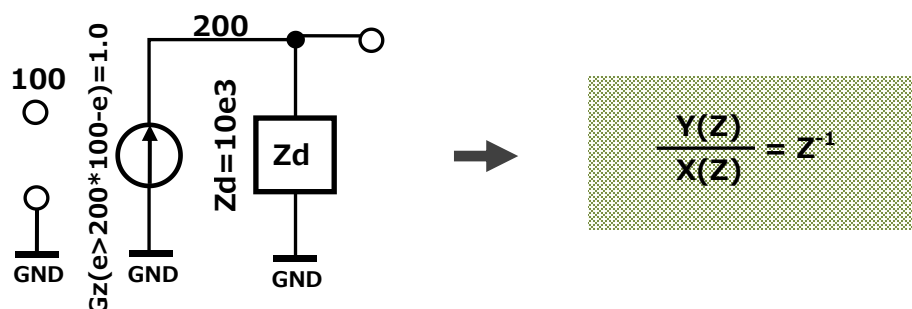
CASL87 Ver.3.00 から新規に加わった素子です。デジタル・フィルタを構成するための遅延素子 Z⁻¹ で、値はサンプリング周波数です。



この遅延素子は頭文字に'Z'あるいは'z'が先頭に付く名称で表し、値は整数、実数（指数表記も含む）です。単位換算は'Hz'（ヘルツ）で正の値でなくてはなりません。記述例は、以下の通りです。

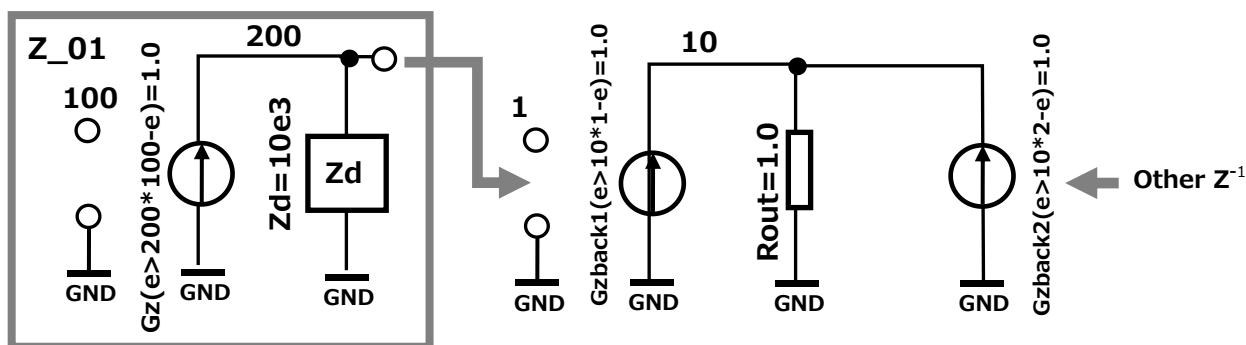
Zd = 10e3 ; // Sampling frequency = 10KHz

Z⁻¹素子を構成するには、以下に示すように必ず、電流源 G と組み合わせて、ひとつのノードのみで完結した回路でなくてはなりません。



上の図では、入力電圧であるノード 100 に比例する電流源 Gd から出力される電流によって、Zd (Z⁻¹素子) に発生する電圧がノード 200 の電圧になります。上記の回路をひとつの Z⁻¹素子の回路ブロックとして、他の回路ブロックと'Link 文'でリンクして使用することを奨めます。

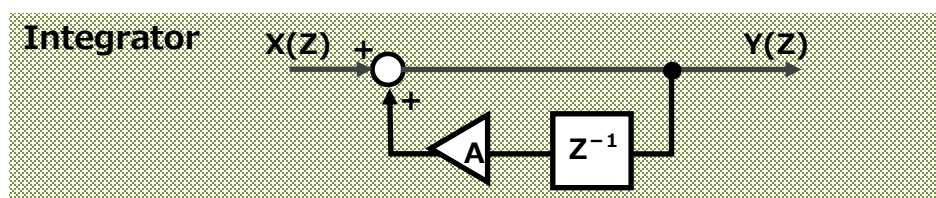
下図に示すように、Z⁻¹素子（回路ブロック名 **Z_01**）の出力は、他の回路ブロックで電流源としての出力とし、抵抗素子 1Ω を負荷とすれば、他の Z⁻¹素子の出力からくる電流源と合成できます。



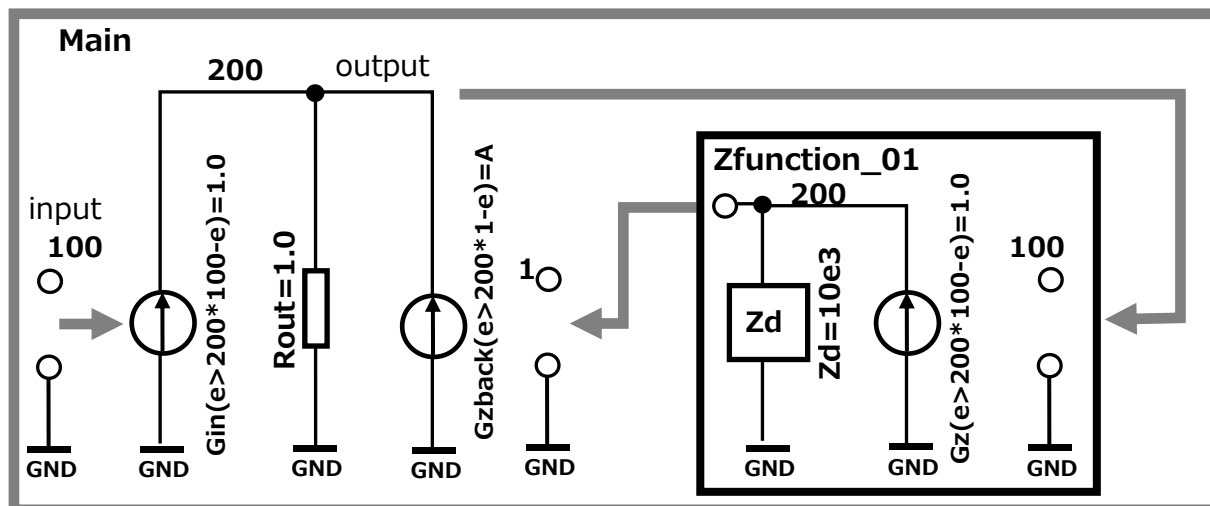
この Z⁻¹素子を使って、FIR や IIR などのデジタル・フィルタの周波数特性を CASL87 で計算できるようになります。

●デジタル・ローパスフィルタの例

Z⁻¹素子を使った 1 次ローパスフィルタの構成を下図に示します。出力 Y(Z) から Z⁻¹素子と増幅器 A を介して入力 X(Z) と加算したものが出力 Y(Z) となる構成です。典型的なフィードバック型の IIR フィルタのもっとも簡単な例です。



この図を、下の回路図にて表記します。回路ブロック **Main** に Z^{-1} 素子回路ブロック **Zfunction_01** が含まれる構造です。**Main** の入力はノード 100 で、出力はノード 200 です。このノード 200 の電圧値は **Zfunction_01** の



の入力であるノード 100 に入力されます。**Zfunction_01** の出力であるノード 200 の電圧値は **Main** のノード 1 に反映されます。**Main** のノード 200 には入力のノード 100 とノード 1 の電圧値を反映する電流源 **Gin** と **Gzback** が合流して抵抗 **Rout** ($=1\Omega$) に流れ込み、ノード 200 の出力電圧となります。尚、 Z^{-1} 素子のサンプリング周波数は、10KHz、増幅器 **A** のゲインは 0.99 とします。

上記の回路図をもとにした CASL87 のソース・コードを以下に示します。

```

/*
    Integrator with attenuation factor_01.CAS
    a program of the digital integrator with attenuation factor.
*/
Circuit Main ;
{
    100 ... Input of Integrator
    200 ... Output of Integrator.
}

Circuit Zfunction_01 ;
{
    100 ... Input of Integrator
    200 ... Output of Integrator.

    node 200, e ;
    Gz(e>200*100-e) = 1.0 ;
    Zd = 10e3 ;
}

End of Zfunction_01 ;

node 200, e ;
Gin(e>200*100-e) = 1.0 ; // Input current.
Gzback(e>200*1-e) = 0.99 ; // Feed back current. 40dB
Rout = 1.0 ;

Link ;
200 = Zfunction_01.100 ;
1 = Zfunction_01.200 ;

End of Main ;

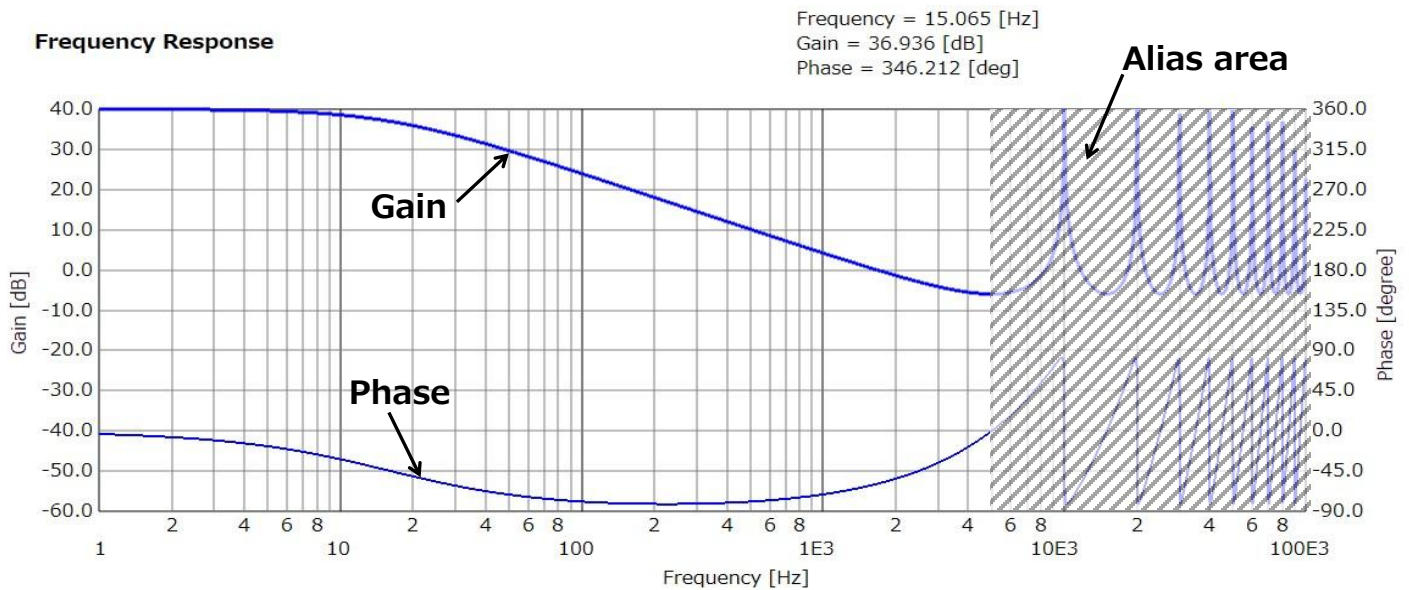
Calculation ;
Input = Main.100 ; Output = Main.200 ;
Start = 1.0 ; Finish = 100e3 ; Division = 10000 ;
Phase Offset = 0.0 ;

End.

```

← Z^{-1} を構成

このソース・コードを基に CASL87、及び GR87 でグラフしたものが下に示す周波数特性です。



Z^{-1} 素子のサンプリング周波数は 10KHz です。シャノンの定理より 5KHz 以上がエイリアス領域（同図の斜線部）になります。ローパスフィルタの最高利得 Gain を 100 倍とすると、増幅器 A の利得は以下の様に順次求められます。

$$Y(z) = X(z) + A \cdot Z^{-1} \cdot Y(z)$$

増幅率を $H(z)$ とすると、以下のようになります。

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - A \cdot Z^{-1}}$$

最大ゲインに特化すると、 $Z^{-1} = 1$ として、

$$\text{Gain} = \frac{1}{1 - A} \quad \longrightarrow \quad A = 1 - \frac{1}{\text{Gain}}$$

$$= 1 - 0.01$$

$$= 0.99 \quad (0 \leq A < 1.0)$$

となります。これより、 Z^{-1} 素子を反映する電流源の値は以下の通りです。

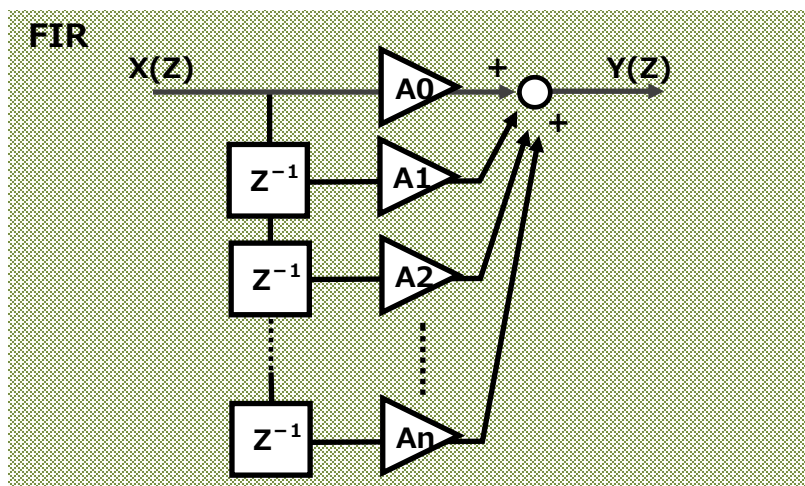
$$\text{Gzback}(e>200 \cdot 1 - e) = 0.99$$

上記の周波数特性のグラフより、レンジは 50dB（約 316 倍）幅あれば充分とみなされますので、正負（±316 → 632 ステップ）の信号としては 10 ビット（1024 ステップ）以上もあれば充分と判断されます。

●FIR によるローパスフィルタの例

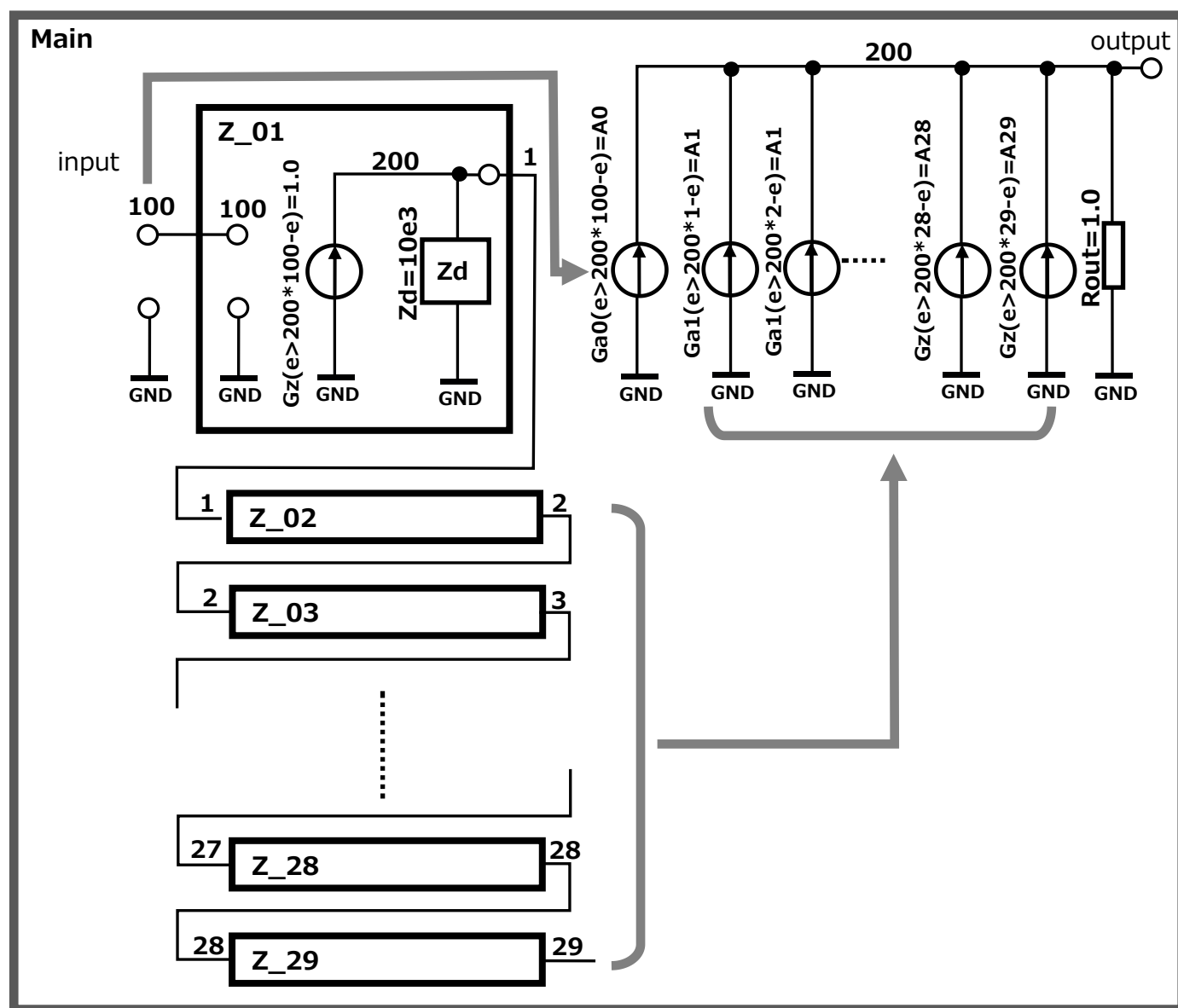
FIR によるフィルタはローパスもハイパスもまったく同じ構造でデジタル・フィルターを構築でき、IIR に比べて安定性に優れていますが、所定の性能を得るためには段数の規模が大きくなってしまいます。

ここでは、そのおおきな段数の FIR でのローパスフィルタの周波数特性をグラフ化してみましょう。FIR のタップは A0～A29 までの 30 個の例です。構成と回路図は次頁に示します。29 個の Z^{-1} 素子を直列に接続し、それぞれの出力から増幅器 A1～A29 に接続します。A0 は直接、入力に接続します。A0～A29 の出力はそれぞれ電流源 Ga0～Ga29 として合流し、負荷抵抗（1Ω）で電圧化して出力します。



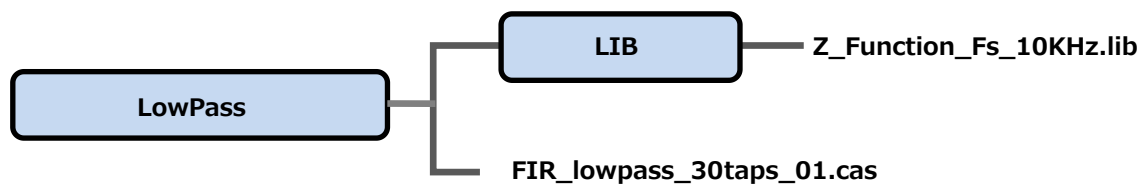
尚、FIR の係数 $A_0 \sim A_n$ の値はフリーのツール・アプリなどから求めることができます。

例：<http://digitalfilter.com/products/dfalz/jpdfalz.html>



Z^{-1} 素子である $Z_{01} \sim Z_{29}$ は、CASL87 のソースコードでは 'Ex Include' 文でライブラリーとして引き出す構造にすれば、簡単な表記になります。

Z⁻¹ 素子である Z_01～Z_29 は、以下の図のように、フォルダ'LIB'内のファイル'Z_Function_Fs_10KHz.lib'に記述されています。



上記のフォルダ構造に基づく CASL87 用の'Z_Function_Fs_10KHz.lib'の内容は以下のとおりです。

```

/*
    Z_Function_Fs_10KHz.LIB

    a program of Z function.

*/
Circuit Zfunction ;
{
    100 ... Input of Integrator
    200 ... Output of Integrator.

    node 200, e ;
    Gz(e>200*100-e) = 1.0 ;
    Zd = 10e3 ;

}
End of Zfunction ;
  
```

以下の CASL87 用ソース記述は、Z⁻¹ 素子回路を赤枠で示す外部ライブラリーとして'Ex Includ'文で引き出す構造を基本としています。

```

/*
    FIR_lowpass_30taps_01.CAS

    a program of FIR lowpass filter.

*/
//
Circuit Main ;
{
    100 ... Input of Integrator
    200 ... Output of Integrator.

}

//
Ex Include Z_01      .¥LIB¥Z_Function_Fs_10KHz.lib ; ← ライブラリーから引き出す
Ex Include Z_02      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_03      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_04      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_05      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_06      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_07      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_08      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_09      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_10      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_11      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_12      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_13      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_14      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_15      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_16      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_17      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_18      .¥LIB¥Z_Function_Fs_10KHz.lib ;
  
```

```

Ex Include Z_19      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_20      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_21      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_22      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_23      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_24      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_25      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_26      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_27      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_28      .¥LIB¥Z_Function_Fs_10KHz.lib ;
Ex Include Z_29      .¥LIB¥Z_Function_Fs_10KHz.lib ;

```

```
node 200, e ; // Adder point of current.
```

```

Ga0(e>200*100-e) = -0.000440939713601099980 ;
Ga1(e>200*1-e) = 0.003019147395237144000 ;
Ga2(e>200*2-e) = 0.002716963657404195100 ;
Ga3(e>200*3-e) = -0.005686857748294383100 ;
Ga4(e>200*4-e) = -0.005881182439938400400 ;
Ga5(e>200*5-e) = 0.010928946658392148000 ;
Ga6(e>200*6-e) = 0.011591255909648627000 ;
Ga7(e>200*7-e) = -0.019248622692254618000 ;
Ga8(e>200*8-e) = -0.021311352440475429000 ;
Ga9(e>200*9-e) = 0.033019645904294175000 ;
Ga10(e>200*10-e) = 0.039184030401879781000 ;
Ga11(e>200*11-e) = -0.059958150865893324000 ;
Ga12(e>200*12-e) = -0.082471421949337223000 ;
Ga13(e>200*13-e) = 0.149690577925755350000 ;
Ga14(e>200*14-e) = 0.446432887628742840000 ;
Ga15(e>200*15-e) = 0.446432887628742840000 ;
Ga16(e>200*16-e) = 0.149690577925755350000 ;
Ga17(e>200*17-e) = -0.082471421949337223000 ;
Ga18(e>200*18-e) = -0.059958150865893324000 ;
Ga19(e>200*19-e) = 0.039184030401879781000 ;
Ga20(e>200*20-e) = 0.033019645904294175000 ;
Ga21(e>200*21-e) = -0.021311352440475429000 ;
Ga22(e>200*22-e) = -0.019248622692254618000 ;
Ga23(e>200*23-e) = 0.011591255909648627000 ;
Ga24(e>200*24-e) = 0.010928946658392148000 ;
Ga25(e>200*25-e) = -0.005881182439938400400 ;
Ga26(e>200*26-e) = -0.005686857748294383100 ;
Ga27(e>200*27-e) = 0.002716963657404195100 ;
Ga28(e>200*28-e) = 0.003019147395237144000 ;
Ga29(e>200*29-e) = -0.000440939713601099980 ;
//
Rout = 1.0 ;

```

```
Link ;
```

```

100 = Z_01.100 ; 1 = Z_01.200 ;
1 = Z_02.100 ; 2 = Z_02.200 ;
2 = Z_03.100 ; 3 = Z_03.200 ;
3 = Z_04.100 ; 4 = Z_04.200 ;
4 = Z_05.100 ; 5 = Z_05.200 ;
5 = Z_06.100 ; 6 = Z_06.200 ;
6 = Z_07.100 ; 7 = Z_07.200 ;
7 = Z_08.100 ; 8 = Z_08.200 ;
8 = Z_09.100 ; 9 = Z_09.200 ;
9 = Z_10.100 ; 10 = Z_10.200 ;
10 = Z_11.100 ; 11 = Z_11.200 ;
11 = Z_12.100 ; 12 = Z_12.200 ;
12 = Z_13.100 ; 13 = Z_13.200 ;
13 = Z_14.100 ; 14 = Z_14.200 ;
14 = Z_15.100 ; 15 = Z_15.200 ;
15 = Z_16.100 ; 16 = Z_16.200 ;
16 = Z_17.100 ; 17 = Z_17.200 ;
17 = Z_18.100 ; 18 = Z_18.200 ;
18 = Z_19.100 ; 19 = Z_19.200 ;
19 = Z_20.100 ; 20 = Z_20.200 ;
20 = Z_21.100 ; 21 = Z_21.200 ;
21 = Z_22.100 ; 22 = Z_22.200 ;
22 = Z_23.100 ; 23 = Z_23.200 ;

```

```

23 = Z_24.100 ; 24 = Z_24.200 ;
24 = Z_25.100 ; 25 = Z_25.200 ;
25 = Z_26.100 ; 26 = Z_26.200 ;
26 = Z_27.100 ; 27 = Z_27.200 ;
27 = Z_28.100 ; 28 = Z_28.200 ;
28 = Z_29.100 ; 29 = Z_29.200 ;

```

End of Main ;

Calculation ;

```

Input = Main.100 ; Output = Main.200 ;
Start = 100 ; Finish = 10e3 ; Division = 10000 ;
Phase Offset = 0.0 ;

```

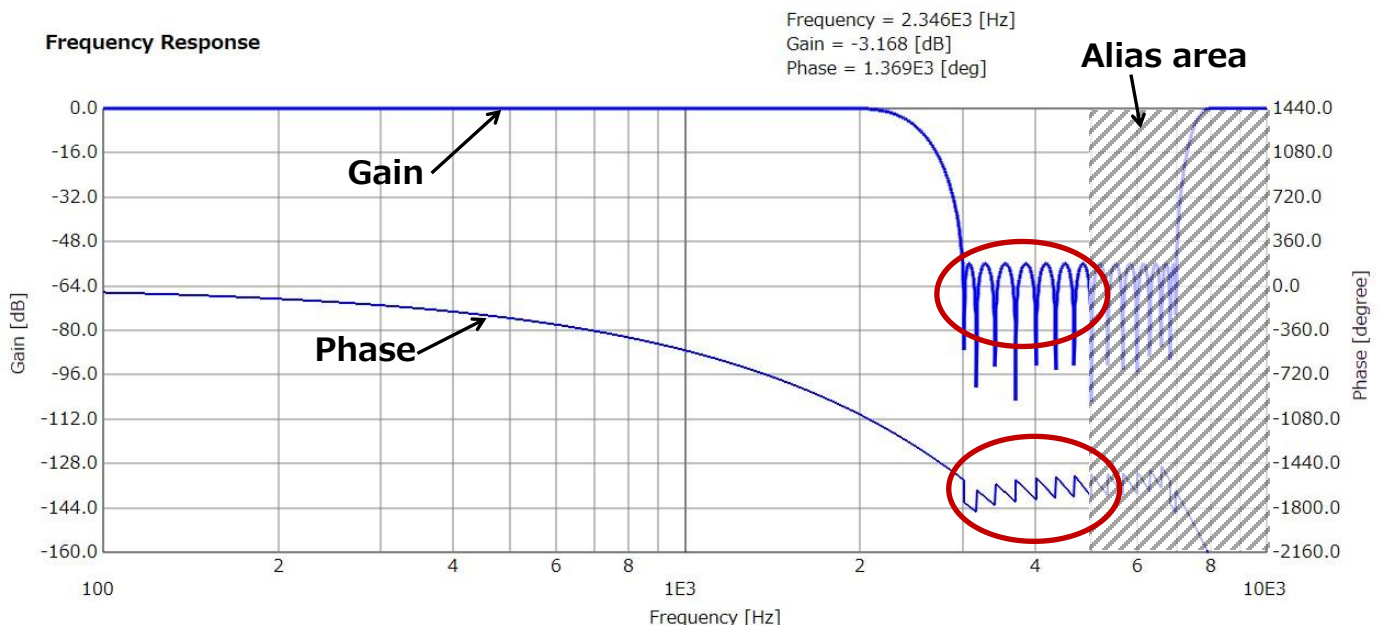
End.

※上記の CASL87 用ソースコードを含む圧縮ファイルは以下の URL より取得できます。

<https://mega.nz/file/4VIVBa5a#E84RmnBK-9RsuMpW2k3VCgMS0t8d6Q63bl2eCNBqwcY>

CASL87で上記のソースコードをコンパイル及び、GR87でグラフ化すると下の図のような結果を得ることができます。サンプリング周波数が10KHzなので、非エイリアス領域（斜線部以外）はその半分の5KHz以下になります。

デジタル・フィルタ固有のリプル（赤枠部）が発生していることが確認できます。リップルのレベルまでの信号のレンジは、約56dB（631ステップ）あるので、正負信号としてのレンジは、62dB（1262ステップ）くらいは必要になります。つまり、マグニチュード型のADやDA変換時のビット数は11ビット（2048ステップ）以上必要であることが判ります。



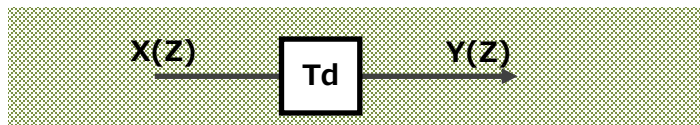
AD変換後では、例えば、10ビット分解能であっても、デジタル・フィルタ内では固定小数点、浮動小数点、倍精度型固定小数点などで、さらなる多ビット化がなされて処理されます。

ここで、注意しなくてはならないことは、CASL87は非常に高い精度（64ビットCPU版：倍精度型浮動小数点（8byte = 64bit）数値範囲：正負とも5E-324~1.7E+308）で計算しています。

つまり、ユーザーがデジタル・フィルタとして実際に使用するDSP(Digital Signal Processor)、或いは、それに類するデバイス内の計算アルゴリズムが固定小数点や単精度浮動小数点の場合では、非常に高い精度のCASL87での計算結果よりも、実物のデバイスでの精度が悪くなることです。

〈遅延素子〉

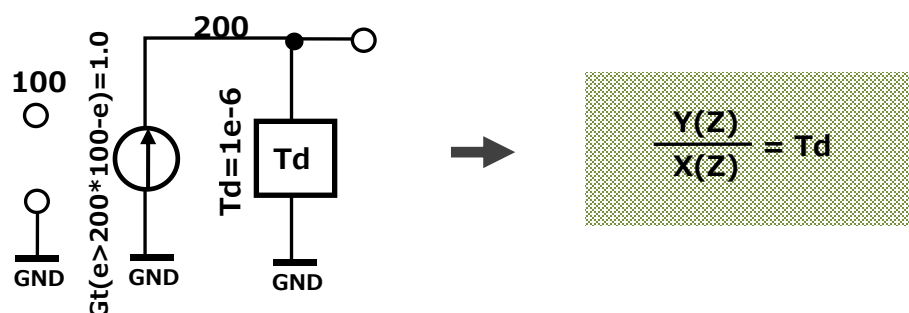
CASL87 Ver.3.00 から新規に加わった素子です。制御機器の信号伝達経路の中に人の操作による遅延や、AD/DA 変換をともなう、CPU（DSP 含む）や FPGA などでの処理時間の遅延が及ぼす影響をゲインと位相で解析できます。



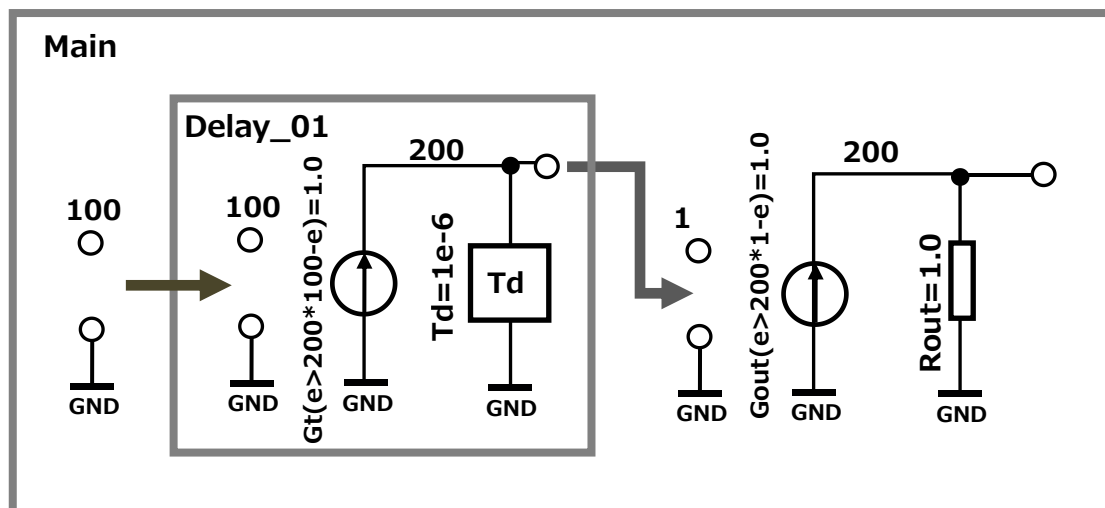
遅延素子は頭文字に'T'あるいは't'が先頭に付く名称で表し、値は整数、実数（指数表記も含む）です。単位換算は'Second'（秒）で正の値でなくてはなりません。記述例は、以下の通りです。

Tdelay1 = 1e-6 ; // 1μSec (1/1E6 Second)

以下に示すように、必ず、電流源 G と組み合わせて、ひとつのノードのみで完結した回路でなくてはなりません。



この図を、下の回路図にて表記します。回路ブロック **Main** に遅延素子回路ブロック **Delay_01** が含まれる構造です。**Main**の入力はノード 100 で、出力はノード 200 です。遅延素子回路ブロック **Delay_01**の入力ノード 100 及び出力ノード 200 は、回路ブロック **Main** にそれぞれ、ノード 100、ノード 1 に'Link 文'でリンク（結合）します。ここでは遅延時間 T_d は $1E-6$ (1μSec) です。



上記の回路図を基にした CASL87 のソースコードを次頁に示します。


```

/*
    Delay_Device_01.CAS

*/
Circuit Main ;
    Circuit Delay_01 ;
    {
        100 ... Input of Delay_01.
        200 ... Output of Delay_01.

        node 200, e ;
        Gt(e>200*100-e) = 1.0 ;
        Td = 1E-6 ; // 1uSec delay.

    }

    End of Delay_01 ;
    //
    node 200, e ;
    Gout(e>200*1-e) = 1.0 ;
    Rout = 1.0 ;

    Link ;
    100 = Delay_01.100 ; 1 = Delay_01.200 ;

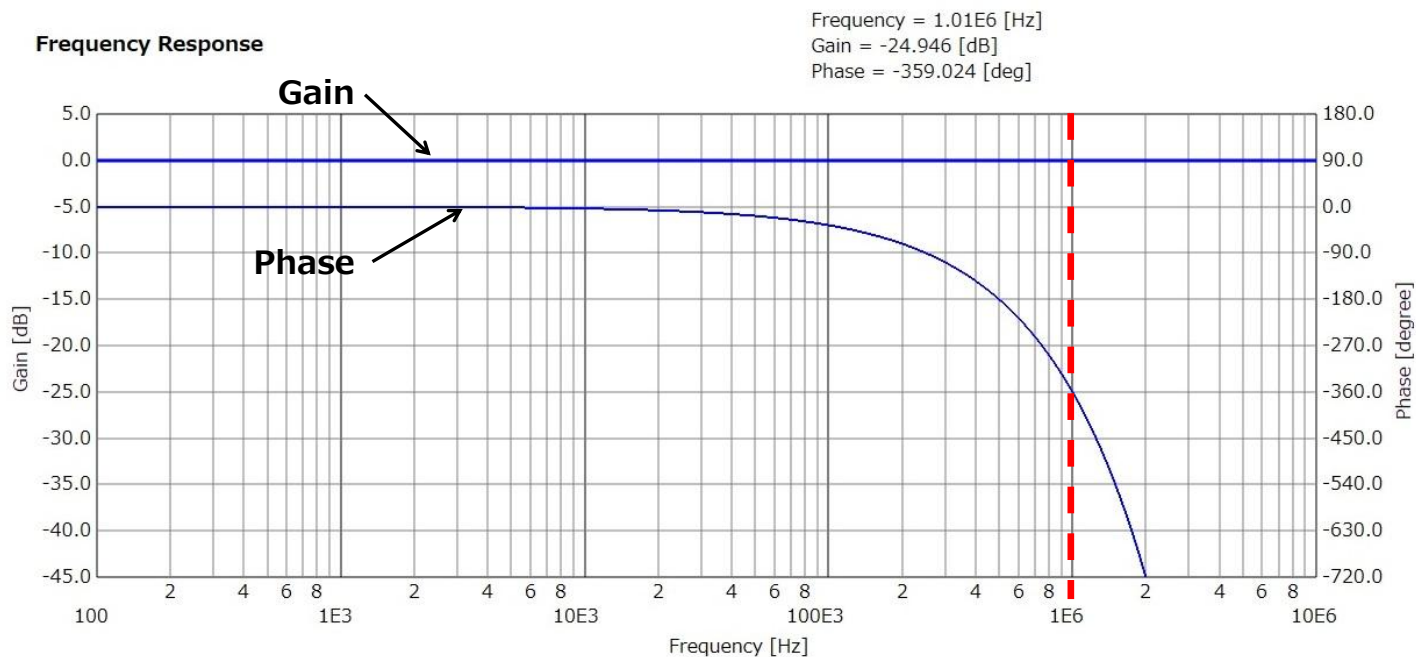
End of Main ;

Calculation ;
    Input = Main.100 ; Output = Main.200 ;
    Start = 1.0 ; Finish = 10e6 ; Division = 10000 ;
    Phase Offset = -0.0 ;

End.

```

GR87でのグラフは以下のとおりになります。遅延時間を 1 μ Sec としていたので、1MHz で 360deg.の遅れが生じているのが判ります。

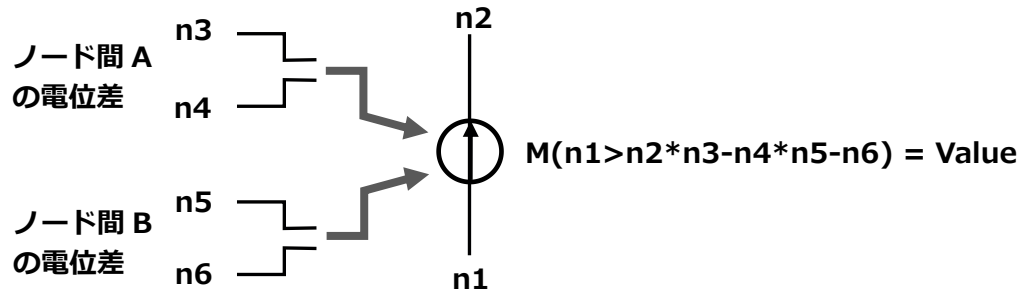


〈乗算素子〉

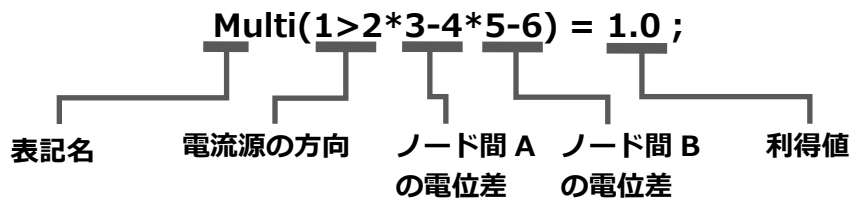
CASL87 Ver.3.01 から新規に加わった素子です。デジタル・フィルターの解析用に、2 入力の乗算機能をもつ電流源です。

概念と CASL87 でのソースコード記述は、以下のようになります。

● 概念



● CASL87 での記述



表記名： 乗算素子の表記名（文字列）の先頭文字は、必ず'**M**'、或いは'**m**'になります。

電流源の方向： 電流源のノードに対する方向。この例では、ノード 1 からノード 2 に流れます。

ノード間 A の電位差： ノード間 A の電位差を入力。

ノード間 B の電位差： ノード間 B の電位差を入力。

利得値： 電流源の利得。**電流源の出力 = A の電位差 × B の電位差 × 利得値。**

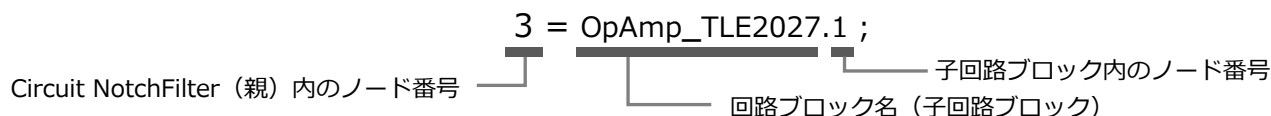
【Link 文の記述】

回路ブロックの中に子回路ブロックを内包しているときに、親回路ブロックのノード、或いは子回路ブロック間の接続を指示するのが、Link 文です。

以下の青い背景の部分に示します。

```
Circuit NotchFilter ;  
    Circuit OpAmp_TLE2027 ;  
        S  
    End of OpAmp_TLE2027 ;  
    node 100, 2 ;  
    Rin = 220 ;  
        S  
    Link ;  
        3 = OpAmp_TLE2027.1 ; 4 = OpAmp_TLE2027.2 ;  
        6 = OpAmp_TLE2027.4 ;  
        S  
    End of NotchFilter ;
```

この例では、回路ブロック「Circuit NotchFilter」内の Link 文で、「Circuit OpAmp_TLE2027」の外部向けノードであるノード番号 1, 2, 4 と、親回路ブロック「Circuit NotchFilter」のノード番号 3, 4, 6 の接続を定義付けています。



次のように左辺、右辺逆でもかまいません。

```
OpAmp_TLE2027.1 = 3 ;
```

また、次のように直接、子回路ブロックどうしの接続を記述することもできます。

```
HighSideFilter.3 = OpAmp_TLE2027.1 ;
```

ただし、階層を深くもつソース・コードで、次のような複数の深層に及ぶ階層構造を意図した接続は記述できません。

```
Hierarchy_01. Hierarchy_02.3 = 3 ;
```

Link 文でこのような複数の深層階層での接続を指定に対した場合、ユーザーは意図せずに二重の実ノード番号を発生させている可能性もありますので、'CASL87 ver.1.21'からは'CASL87'を実行すると、エラー表示されるように改変しました。

Link 文は、必ず、回路ブロック内のノード記述が終えた最後に記述してください。

【注釈の記述】

注釈は' { } 'や'/* */'で囲むことで入れられます。以下のように行を跨いでもかまいません。

```
{ Test circuit
    1----- Input. 10 ----- Output. }
/*      Circuit01.CAS
    Band stop circuit by OP amplifier.
*/
```

さらに、'//'により、行全体を注釈とすることもできます。

```
node 2, 4 ;
    //Rhi1 = 68e3 ;    {Masking}
    Rhi2 = 68e3 ; // Remark
```

尚、以下の例のような注釈のネスティング（入れ子）はできません。

```
{ /*      ABCD
    */
}
```

注釈文も含め、ソース・コード内は英数文字（ANSI）のみで記述しなくてはなりません。日本語（Shift-JIS）は許されません。特に日本語（Shift-JIS）のスペースはテキスト・エディタで見落としがちですので注意してください。日本語（Shift-JISなどはCASL87によって、エラーで表示されますが、指摘の行番号が前後する場合があります。尚、ソース・コード・ファイル名やそのパス名には漢字が使えます。

3. Calculation 文の記述

回路ブロックの記述が終えたなら、計算の仕様を示す Calculation 文です。以下の青い背景の部分に示します。

```
Circuit NotchFilter ;
Circuit OpAmp_TLE2027 ;
    S
End of OpAmp_TLE2027 ;
node 100, 2 ;
    Rin = 220 ;
    S
Link ;
    3 = OpAmp_TLE2027.1 ; 4 = OpAmp_TLE2027.2 ;
    6 = OpAmp_TLE2027.4 ;
    S
End of NotchFilter ;
Calculation ;
    Input = NotchFilter.100 ; Output = NotchFilter.200 ;
    Start = 0.1 ; Finish = 100e3 ; Division = 1000 ;
    Phase Offset = 0.0 ;
End.
```

この例では全体回路ブロック NotchFilter を計算します。以下に Calculation 文内の項目を説明します。

Calculation ; Calculation の宣言文。

Input = NotchFilter.100 ;

回路ブロック内のノード番号を指定。表記は、回路ブロック名 ■ ノード番号
入力ノードの指定。

Output = NotchFilter.200 ;

回路ブロック内のノード番号を指定。表記は、回路ブロック名 ■ ノード番号
出力ノードの指定。

Start = 0.1 ;

周波数は浮動小数点で指定する。例 0.1 100e-3。
最初の周波数の指定。

Finish = 100e3 ;

周波数は浮動小数点で指定する。例 100000.0 100e3。
最後の周波数の指定。

グラフ・プロッター GR87 では最低、最高周波数は 10 のべき乗値に固定されますので、これらの周波数も 10 のべき乗に準拠することをお勧めします。

```
Division = 1000 ;
```

分割数は整数で指定する。例 1000
Start と Finish の周波数間の分割数の指定。

```
Phase Offset = 0.0 ;
```

分割数は整数で指定する。例 0.0 -180.0
位相のオフセット値の指定。

4. End.宣言

すべての記述の最後を宣言します。表記は、End. です。最後にピリオドを付けることに注意してください。

5. Include 文

CASL87 用ソースコードとして一つの重要な機能として、Include 文があります。これは、回路ブロックをライブラリー化する場合にそれをソースコードから呼び出す機能です。

Include 文は例として次のように記述されます。

```
Include E:¥CasLibrary¥Analog¥OpAmp¥OpAmp_TLE2027.lib ;
```

ライブラリー・ファイル（相対パス也可）
Include 宣言。

上の例ではライブラリー・ファイルのサフィックスは'.lib'になっていますが、自由なサフィックスでかまいません。例えば、OpAmp¥OpAmp_TLE2027.txt や OpAmp¥OpAmp_TLE2027.CAS でもかまいません。ユーザーの自由な管理にお任せします。

次の記述例では、Circuit OpAmp_TLE2027 が内部に記述されたファイル'OpAmp_TLE2027.lib'を Include 文で呼び出しています。

```
Circuit NotchFilter ;
```

```
Include E:¥CasLibrary¥Analog¥OpAmp¥OpAmp_TLE2027.lib ;
```

```
node 100, 2 ;
```

```
Rin = 220 ;
```

⌋

```
Link ;
```

```
3 = OpAmp_TLE2027.1 ; 4 = OpAmp_TLE2027.2 ; 6 = OpAmp_TLE2027.4 ;
```

⌋

```
End of NotchFilter ;
```

```
Calculation ;
```

```
Input = NotchFilter.100 ; Output = NotchFilter.200 ;
```

```
Start = 0.1 ; Finish = 100e3 ; Division = 1000 ;
```

```
Phase Offset = 0.0 ;
```

```
End.
```


この Include 文で呼び出されるライブラリー・ファイルの内容は以下のように、先の Circuit OpAmp_TLE2027 と同じテキスト・ファイルです。つまり、呼び出し元のソース・コードの Include 文の位置にインクルードされます。呼び出し元のソース・コードの Include 文の前後で文法的に適正でなくてはなりません。

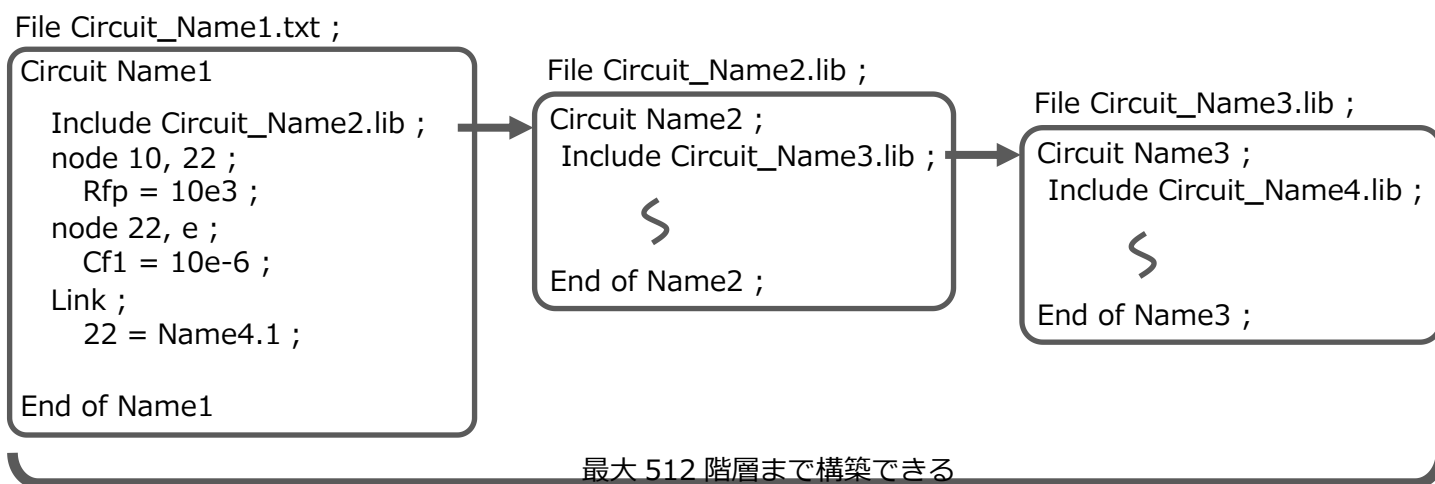
```
Circuit OpAmp_TLE2027 ;
{ Operational amplifier TLE2027. Raw gain = 150 dB. Ft = 20 MHz.
  Input Resistor = 7.5 Mohm. Output Resistor = 25 ohm.
  1 ----- Input+, 2 ----- Input-, 4 ----- Output          }
node 1, 2 ;
  Rin = 7.5e6 ;

node 3, e ;
  Gmgain(e>3*1-2) = 31.6e6 ; Rp1 = 1.0 ; Cp1 = 253e-3 ;

node 4, e ;
  Gout(e>4*3-e) = 40e-3 ; Rout = 25.0 ;

End of OpAmp_TLE2027 ;
```

CASL87 でこの Include 機能で優れている点は、ライブラリーのソースコード内にさらに Include 文を含有できることです。つまり、ライブラリーどうして階層構造を築けます。その深さの階層は無制限ですが、512 まで階層ファイルが許容されます。



Include 文を活用することで、ソース・コードが長くなることを防ぐことができ、ソース・コードのメンテナンス性が向上します。

Include 文で呼び出すライブラリー・ファイル群はユーザーが設えた特定のフォルダーにまとめて格納してください。そのフォルダーもライブラリーの種類で分けて整理することをお勧めします。

・相対パスの指定

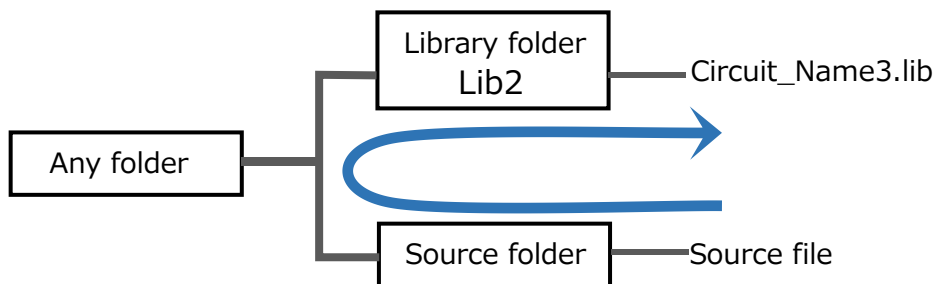
Ver.1.30 から相対パスの指定が可能になりました。

同じフォルダーにあるファイル Circuit_Name2.lib の場合：

```
Include .\Circuit_Name2.lib
```

ひとつ上のフォルダーの下の別のフォルダ¥Lib2 下のファイル Circuit_Nmae3.lib の場合：

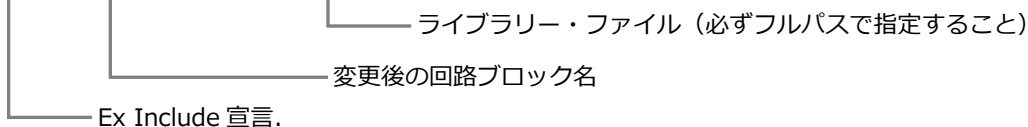
Include ..¥Lib2¥Circuit_Name3.lib



6. Ex Include 文

Include 文の拡張仕様として、“Ex Include”文があります。Include 文が一つのライブラリ・ファイルを一箇所のみしか利用できません。“Ex Include”文は、同じライブラリ・ファイルの回路ブロック名を仮想的に別の回路ブロック名として付けることができます。つまり、違う名前を付けることで同じ回路ブロックを複数のライブラリとして利用できるようにするものです。

Ex Include Rename E:¥CasLibrary¥Analog¥OpAmp¥OpAmp_TLE2027.lib ;



次の記述例では、Circuit OpAmp_TLE2027 を OpAmp_TLE2027_01 や OpAmp_TLE2027_02 として、Ex Include 文で呼び出しています。そのため、Link 文では仮想的に付けた回路ブロック名を以ってして接続していることに注意してください。

Circuit NotchFilter ;

Ex Include OpAmp_TLE2027_01 E:¥CasLibrary¥Analog¥OpAmp¥OpAmp_TLE2027.lib ;

Ex Include OpAmp_TLE2027_02 E:¥CasLibrary¥Analog¥OpAmp¥OpAmp_TLE2027.lib ;

node 100, 2 ;

Rin = 220 ;

Σ

Link ;

3 = OpAmp_TLE2027_01.1 ; 4 = OpAmp_TLE2027_01.2 ; 7 = OpAmp_TLE2027_01.4 ;

5 = OpAmp_TLE2027_02.1 ; 6 = OpAmp_TLE2027_02.2 ; 6 = OpAmp_TLE2027_02.4 ;

Σ

End of NotchFilter ;

Calculation ;

Input = NotchFilter.100 ; Output = NotchFilter.200 ;

Start = 0.1 ; Finish = 100e3 ; Division = 1000 ;

Phase Offset = 0.0 ;

End.

上の例ではライブラリー・ファイルのサフィックスは'.lib'になっていますが、自由なサフィックスでかまいません。例えば、OpAmp¥OpAmp_TLE2027.txt や OpAmp¥OpAmp_TLE2027.CAS でもかまいません。ユーザーの自由な管理にお任せします。Ex Include 文の階層制限数は、Include 文に準じます。

・相対パスの指定

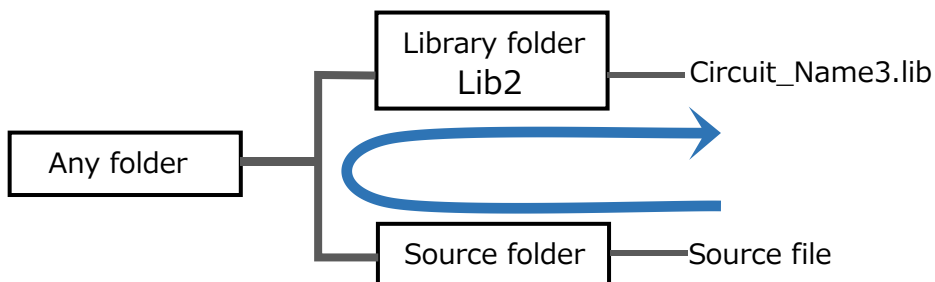
Ver.1.30 から相対パスの指定が可能になりました。

同じフォルダーにあるファイル Circuit_Name2.lib の場合：

```
Ex Include  Circuit_OpAmp1  .¥Circuit_Name2.lib
```

ひとつ上のフォルダーの下別のフォルダ¥Lib2 下のファイル Circuit_Nmae3.lib の場合：

```
IEx nclude  Circuit_OpAmp1  ..¥Lib2¥Circuit_Name3.lib
```



7. 文法や機能の注意事項

【全般】

1. 注釈内も含めて日本語（Shift-JIS 文字など）の記述は許されません。英数文字（ANSI 文字）のみの記述をお願いします。特に Shift-JIS での空白文字の誤記には気を付けてください。
2. Circuit（回路ブロック名）や Link 文、及び Calculation 文の構成順序は以下のように並んでいなくてはなりません。

```
Circuit ~ ;  
    Circuit ~ ;  
        node x, y ;  
    End of ~ ;  
    node x, y ;  
    Link ;  
End of ~ ;  
Calculation ;  
End.
```

3. ソース・コード全体の行数の制限は、最大 65,536 行です。単なる改行のみの行も含まれることに注意してください。
4. CASL87、GR87 共に一度に扱えるファイル数は 1,024 までです。

【回路ブロック】

1. 回路ブロック（Circuit 文）数の制限は、最高 10,000 です。この数は内包する階層の回路ブロック数も含まれます。
2. 回路ブロックの階層構造上の層の深さの制限はありませんが、階層ファイル数の制限は 512 です。
3. 回路ブロック名の先頭に 'Link' や 'Calculation' などの宣言文などと同じ名称を盛り込んではいけません。
4. ノード番号は、グラウンド e（大文字 E 含む）以外、32 ビット版の CASL87 では 1～6,999、64 ビット版の CASL87 では 1～29,999 の範囲で使用できます。ノード番号の付け方は順不同でかまいません。ただし、同じ回路ブロック内では、異なるノードではノード番号が重複しないことを注意してください。
5. ノード番号は整数表記のみ受け付けます。
6. 部品素子の抵抗 R やインダクタ L の値には 0（ゼロ）を代入しないでください。
7. Link 文は必ず、回路ブロック内で宣言してください。

【Calculation 文】

1. Start 及び、Finish の周波数値は自由に記述できますが、できるだけ 10 のべき乗値にしてください。
2. Division 値は整数でなくてはなりません。
3. CASL87 の位相特性は 90 度の整数倍毎に位相角の推移を判断しています。変化の激しい位相特性をスムーズに描くには Division 値は 1000 以上のできるだけ大きな値にしてください。（ただし、10,000 以下）

8. CASL87 のエラー項目

CASL87 では、エラーのない正常な終了では次のように表示されます。

\$/- Complete the calculation successfully.

以下に‘Content’ボックスに表示されるエラーの内容を説明します。

実行環境に関するエラー：

Error : not 64bit windows system.

【説明】

32 ビットの Windows 環境下で CASL87 の 64 ビット版を起動した場合にこのエラーが表示されます。32 ビットの Windows では、32 ビット版の CASL87 をセットアップしてください。尚、64 ビットの Windows では、32 ビット版の CASL87 は起動できます。ただし、32 ビットの Windows10 システムでは 64 ビット版の CASL87 はインストールそのものができません。

ファイルに関するエラー：

Error : File has not been selected.

ソース・コード・ファイルが選択（入力）されていません。

Error : Not Exist file name.

file = フルパスのファイル名

存在しないファイル名を入力しています。

Circuit 文に関するエラー：

Error : Missing semicolon ";". Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文内で、ノードや部品の定数及び、Link 文を記述する文の区切りとして“;”の記述がない場合にこのエラーが表示されます。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : The value of the part is zero. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文内で、抵抗 R とインダクタ L の値は 0（ゼロ）は許されません。影響を少なくする値にするなら、極めて小さな値、例えば、1e-3（0.001）とか低い値にしてください。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error: Number of the node is Zero or Minus. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

ノード文のノード番号に 0（ゼロ）やマイナス値が記述された場合にこのエラーが表示されます。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error: Error : Multiple characters including "E" are described. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

ノード文のノード番号のグラントを表す'E'が単一の文字でない場合にこのエラーが表示されます。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Unpermissive node number in Link statement. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文内で不適当なノード番号が記述された場合にこのエラーが表示されます。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Unpermissive linking statement in Link mode. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文内で不適当な記述があった場合にこのエラーが表示されます。例えば、Link 文内で‘node 10, 2 ;’などが記述された場合です。

Error : There are multiple Link statements in the same circuit hierarchy. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

同じ回路ブロック階層で 2 重の Link 文がある場合にこのエラーが表示されます。‘Rewrite the source file named〜’で指定されたソース・コード・ファイルの修正をしてください。

Error: Circuit block with two or more layers cannot be described in the Link statement. Line : <行番号>
<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文内のノードの接続の左辺、或いは右辺で 2 重以上の階層表示の回路ブロックの表現がある場合にこのエラーが表示されます。例えば次のような例です。

Hierarchy_01. Hierarchy_02.3 = 4 ;

‘Rewrite the source file named〜’で指定されたソース・コード・ファイルの修正をしてください。

Error : The absolute value of the number of nodes has exceeded the maximum value(29999).

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

ソース・コードによるノード数の絶対値が最大値（29999）を超えました。64 ビット版の CASL87 ではこの 29999 ですが、32 ビット版の CASL87 では、6999 と表示されます。

‘Rewrite the source file named〜’で指定されたソース・コード・ファイルの修正をしてください。

Error : Unnecessary circuit block exists. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文内に存在しない回路ブロックが宣言されている場合にこのエラーが表示されます。‘Rewrite the source file named〜’で指定されたソース・コード・ファイルの修正をしてください。

Error: Multiple nodes. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

同じ Circuit 文（回路ブロック名）内で、同じノード文（ノード番号の組み合わせが同じ）が重複している場合にこのエラーが表示されます。‘Rewrite the source file named〜’で指定されたソース・コード・ファイルの修正をしてください。

Error : The description of the current direction of mutual conductance is incorrect. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、電流方向がそのノード文のノード番号と一致しない場合にこのエラーが表示されます。‘Rewrite the source file named～」で指定されたソース・コード・ファイルの修正をしてください。

Error : Current direction of mutual conductance and node number of differential voltage are the same.

Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、電流方向のノード番号や差動電圧のノード番号が同じ番号になってしまっている場合にこのエラーが表示されます。‘Rewrite the source file named～」で指定されたソース・コード・ファイルの修正をしてください。

Error : Unpermissive expression of current source. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、")"がないなどの不当な記述がある場合にこのエラーが表示されます。‘Rewrite the source file named～」で指定されたソース・コード・ファイルの修正をしてください。

Error : There is a node number other than the specified number. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、ノード番号に指定した表記以外の、例えば、整数で記述しなくてはならないのに浮動小数点の記述がある場合にこのエラーが表示されます。‘Rewrite the source file named～」で指定されたソース・コード・ファイルの修正をしてください。

Error : "(" or ")" does not exist. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、"("や")"がない場合にこのエラーが表示されます。‘Rewrite the source file named～」で指定されたソース・コード・ファイルの修正をしてください。

Error : ">", "*" or "-" dose not exist. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、">"や"*"、"-"がない場合にこのエラーが表示されます。'Rewrite the source file named～'で指定されたソース・コード・ファイルの修正をしてください。

Error : The position of ">", "*", or "-" is wrong. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文（回路ブロック名）内の、ノード文下の相互コンダクタンスの記述において、"（"、">"や"*"、"-"、"）"の位置が不順な場合にこのエラーが表示されます。'Rewrite the source file named～'で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent block. Line : <行番号>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Circuit 文内でないところにノードの記述がある場合にこのエラーが表示されます。'Rewrite the source file named～'で指定されたソース・コード・ファイルの修正をしてください。

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で入力ノードの設定がありません。'Rewrite the source file named～'で指定されたソース・コード・ファイルの修正をしてください。

Circuit 文や Calculation 文に共通するエラー :

Error : Can not convert to numbers. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

不適当な数値が記述された場合にこのエラーが表示されます。'Rewrite the source file named～'で指定されたソース・コード・ファイルの修正をしてください。

Error : Have a forbidden string at the beginning. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文や Calculation 文内の項目の先頭部に禁じられた文字があります。回路ブロックの名称などの先頭に Link や Calculation が付いた文字列は禁止です。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Can not write a circuit statement immediately after a link statement. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Link 文の直後に Circuit 文が記述は禁じられています。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Calculation 文に関するエラー :

Error: Node specified as input node does not exist. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で指定した入力ノードの設定に対応したノード番号がすべての Circuit 文内に存在しません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Node specified as output node does not exist. Line : Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で指定した出力ノードの設定に対応したノード番号がすべての Circuit 文内に存在しません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent input node number setting in calculation statement.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で入力ノードの設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent output node number setting in calculation statement.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で出力ノードの設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent start frequency number setting in calculation statement.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内での周波数の開始値の設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent finish frequency number setting in calculation statement.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内での周波数の終了値の設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent division setting of frequency range in calculation statement.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内での周波数範囲の分割値の設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error : Non existent phase offset setting in calculation statement. Line : <行番号>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内での位相のオフセット値の設定がありません。‘Rewrite the source file named～’で指定されたソース・コード・ファイルの修正をしてください。

Error: Incorrect number of the division. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Division 値は整数でなくてはなりません。浮動小数点などを記述した場合にこのエラーが表示されます。

Error : Duplicate description in calculation statement. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で各設定が重複しています。‘Rewrite the source file named～’指定されたソース・コード・ファイルのエラー個所の行を修正をしてください。

Error : Unpermissive defining of Input node. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で Input のノード番号が不適当な場合にこのエラーが表示されます。‘Rewrite the source file named～’指定されたソース・コード・ファイルのエラー個所の行を修正をしてください。

Error : Unpermissive Instruction in Calculation mode. Line : <行番号>

<ソース・コード・ファイルのエラー個所の行を表示>

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

Calculation 文内で項目が不適当な場合にこのエラーが表示されます。‘Rewrite the source file named～’指定されたソース・コード・ファイルのエラー個所の行を修正をしてください。

計算実行時のエラー：

Error : An error occurs in matrix calculation.

Rewrite the source file named <ソース・コード・ファイルのフルパス名>

【説明】

マトリックスの要素に異常がある場合にこのエラーが表示されます。素子のないノードが存在する場合などにこのエラーが発生します。素子が存在しないノードの記述や、相互コンダクタンスのコントロール用入力ノードがどのノードにも接続されていないなどがないか、ソースコードの見直しを行ってください。

Caution : Calculation result is out of range from 1E-310 to 1E+300..

【説明】

計算途中で、計算結果の数字が CASL87 の扱える数値 1E-310～1E+300 の範囲外になった場合にこの注意が発生します。大抵は、Calculation 文内で、Start や Finish 文の周波数範囲に於けるゲインや位相の結果の値が大きすぎたり、小さすぎたりして、数値 1E-310～1E+300 の範囲外になったりした場合に発生しますので、これに影響を与えるソースコードの Start や Finish 文の周波数値の見直しを行ってください。

ただし、この数値 1E-310～1E+300 の範囲外になった事態になった場合は、CASL87 は動作を中断しますが、その時点の直前の周波数までは計算結果ファイルを出力します。

尚、CASL87 が複数のソースコードを処理している場合は、この注意に該当するソースコードの計算処理は中断し、次の順番のソースコードの計算処理に移行します。

9. GR87 のエラー項目

GR87 では、エラーのない正常な終了では‘Content’ボックスには何も表示されず、グラフ・プロットのウィンドウの‘Content’ボックスに‘Completely successful.’と表示され、グラフを正常に表示するだけです。

以下に GR87 の‘Content’ボックスに表示されるエラーの内容を説明します。

ファイルに関するエラー：

Error: None file in the input list.

【説明】

入力リストにファイルがない場合にこのエラーが表示されます。

Error: Can not find the file.

<計算結果ファイルのフルパス名>

【説明】

実在しない計算結果ファイルを‘Input the lcalculation file’ボックスに入力した状態でグラフ・プロットの‘実行ボタン’をクリックした場合にこのエラーが表示されます。もっとも、計算結果ファイルはドラッグ&ドロップで入力されますので、実行前にそのファイルを失った場合に発生するエラーです。

各種設定に関するエラー：

Error : Minimum frequency value is not a power of 10.

【説明】

周波数範囲の設定において、最低周波数の値が 10 のべき乗になっていません。最低周波数、最高周波数は 10 のべき乗値で設定しなくてはなりません。

Error : Maximum frequency value is not a power of 10.

【説明】

周波数範囲の設定において、最高周波数の値が 10 のべき乗になっていません。最低周波数、最高周波数は 10 のべき乗値で設定しなくてはなりません。

Error : Minimum frequency input value is abnormal.

【説明】

周波数範囲の設定において、最低周波数の値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

Error : Maximum frequency input value is abnormal

【説明】

周波数範囲の設定において、最高周波数の値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

Error : Minimum gain input value is abnormal.

【説明】

ゲイン範囲の設定において、最小ゲインの値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

Error : Maximum gain input value is abnormal.

【説明】

ゲイン範囲の設定において、最大ゲインの値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

Error : Minimum phase input value is abnormal.

【説明】

位相範囲の設定において、最小位相の値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

Error : Maximum phase input value is abnormal.

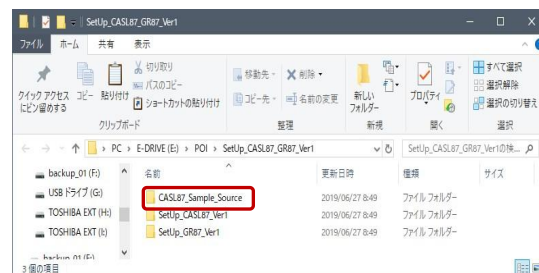
【説明】

位相範囲の設定において、最大位相の値に不適当な文字があります。正確な浮動小数点や指数表記で入力してください。

10. 実際の回路例

実際の回路例として、ノッチ・フィルタ回路を下図に示します。

この回路のソース・コードは先にダウンロードした右図に示す赤枠のセットアップ用フォルダ“SetUp_CASL87_GR87_VerXXX”内のフォルダ “CASL87_Sample_Source”のさらに下のフォルダ “Notch_Filter_Circuit”内にあります。



このフォルダには以下に示すように3つの回路のソース・コードがあります。

Notch_Filter_Q_01.CAS、 Notch_Filter_Q_02.CAS、 Notch_Filter_Q_03.CAS

これらのソース・コード・ファイルを任意のフォルダにコピーしてください。

この回路は負帰還の全体のゲインを変えずにCR フィルタ部への帰還比率を操作することで、Q 値を変化させます。この回路の特徴はフィードバック系の抵抗をオペアンプが十分ドライブできる程度に値を低くすることで、入力側のCR フィルタに影響を極力抑えることができ、結果的にオペアンプの段数を一段で済ませていることです。

負帰還の全体のゲインを変えずに Q 値を変化させる条件は以下の通りです。

$$\text{Gain} = (\text{Rfb3} + \text{Rfb1} + \text{Rfb2}) / (\text{Rfb1} + \text{Rfb2})$$

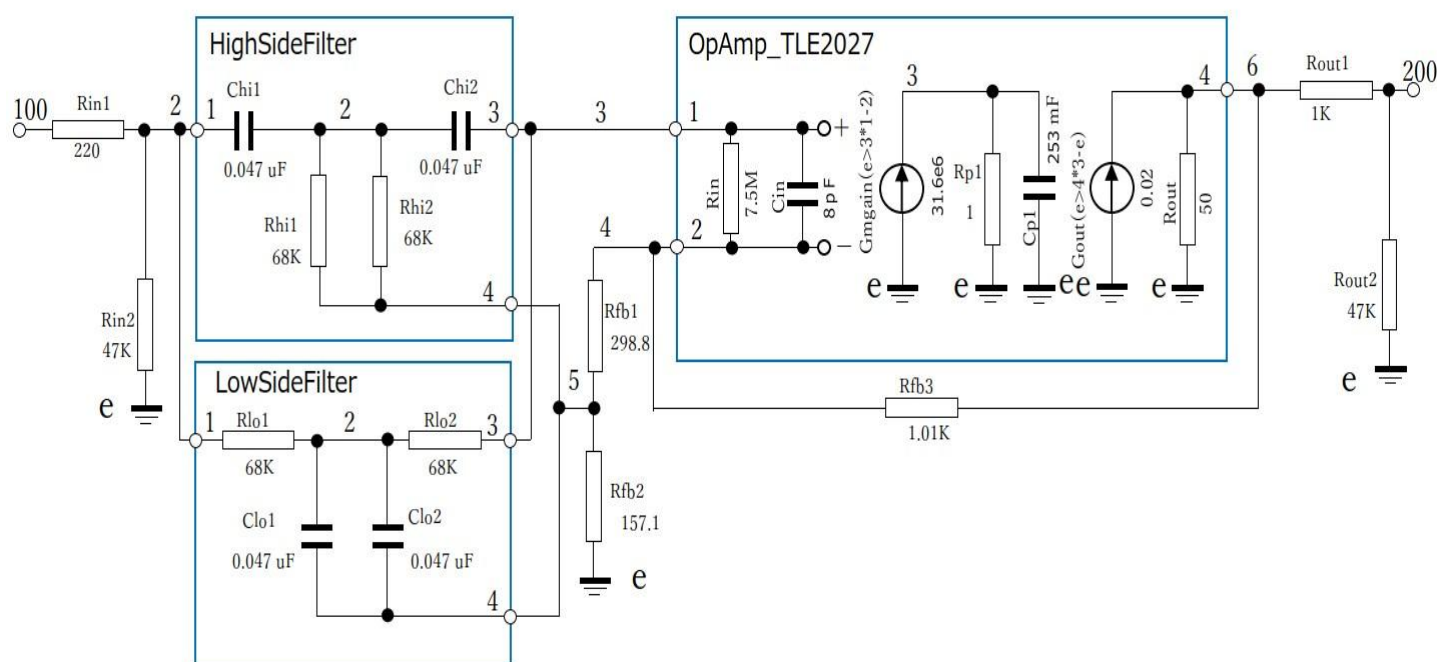
Rfb1 と Rfb2 の和は一定

ゲインを変えない条件は具体的には Rfb1 + Rfb2 の絶対を変えずに Rfb1、Rfb2 の比を変えて設定します。つまり、Rfb1 と Rfb2 の和、455.9 Ωの範囲内で Rfb1、Rfb2 のそれぞれの値を設定して周波数特性をシミュレートします。

尚、Rfb1、Rfb2 を変えるにはユーザーの任意のエディターで記述してください。ただし、文字コードは ANSI のみとしてください。日本語（Shift-JIS）コードなどは使用できないことに注意してください。

ノッチ・フィルタの回路：

NotchFilter



この回路のCASL87用ソースコードの、node 4, 5 のRfb1、node 5, e のRfb2を次のように設定してあります。

Notch_Filter_Q_01.Cas ----- Rfb1 = 298.8 ; Rfb2 = 157.1 ;
 Notch_Filter_Q_02.Cas ----- Rfb1 = 100 ; Rfb2 = 355.9 ;
 Notch_Filter_Q_03.Cas ----- Rfb1 = 22 ; Rfb2 = 433.9 ;
 以下にソース・コードの代表として Notch_Filter_Q_01.Cas の例を以下に示します。

```

/*
                                Notch_Filter_Q_01.CAS
                                a program of new notch filter circuit.

*/
Circuit NotchFilter ;
{ Main circuit.
  Input node ---- 100, OutPut node ---- 200 }
  Circuit HighSideFilter ;
  { High frequency pass filter.
    Input node ---- 1, Output node ---- 3, Sub node(for the feedback) ---- 4 }
    node 1, 2 ;
    Chi1 = 0.047e-6 ;
    node 2, 4 ;
    Rhi1 = 68e3 ; Rhi2 = 68e3 ;
    node 2, 3 ;
    Chi2 = 0.047e-6 ;
  End of HighSideFilter ;

  Circuit LowSideFilter ;
  { Low frequency pass filter.
    Input node ---- 1, Output node ---- 3, Sub node(for the feedback) ---- 4 }
    node 1, 2 ;
    Rlo1 = 68e3 ;
    node 2, 4 ;
    Clo1 = 0.047e-6 ; Clo2 = 0.047e-6 ;
    node 2, 3 ;
    Rlo2 = 68e3 ;
  End of LowSideFilter ;

  Circuit OpAmp_TLE_2027 ;
  { Operational amplifier TLE2027. Raw gain = 150 dB. Ft = 20 MHz.
    Input Resistor = 7.5 Mohm. Input Capacitor = 8 pF. Output Resistor = 25 ohm.
    1 ----- Input+, 2 ----- Input-, 4 ----- Output
    node 1, 2 ;
    Rin = 7.5e6 ; Cin = 8e-12 ;
    node 3, e ;
    Gmgain(e>3*1-2) = 31.6e6 ; Rp1 = 1.0 ; Cp1 = 253e-3 ;
  }

```

```

node 4, e ;
    Gout(e>4*3-e) = 20e-3 ; Rout = 50 ;
End of OpAmp_TLE_2027 ;

```

```

{ Circuit NotchFilter }

```

```

node 100, 2 ;
    Rinl = 220 ;
node 2, e ;
    Rin2 = 47e3 ;
node 4, 5 ;
    Rfb1 = 298.8 ;
node 5, e ;
    Rfb2 = 157.1 ;
node 4, 6 ;
    Rfb3 = 1.01e3 ;
node 6, 200 ;
    Rout1 = 1e3 ;
node 200, e ;
    Rout2 = 47e3 ;

```

これらの抵抗の和 (Rfb1 + Rfb2) を一定にした状態で Rfb1 と Rfb2 の値の比を変える。

```

Link ;
2 = HighSideFilter.1 ; 2 = LowSideFilter.1 ;
3 = HighSideFilter.3 ; 3 = LowSideFilter.3 ; 3 = OpAmp_TLE_2027.1 ;
4 = OpAmp_TLE_2027.2 ;
5 = HighSideFilter.4 ; 5 = LowSideFilter.4 ;
6 = OpAmp_TLE_2027.4 ;

```

```

End of NotchFilter ;

```

```

Calculation ;

```

```

Input = NotchFilter.100 ; Output = NotchFilter.200 ;
Start = 0.1 ; Finish = 1e3 ; Division = 1000 ;
Phase Offset = 0.0 ;

```


```

End.

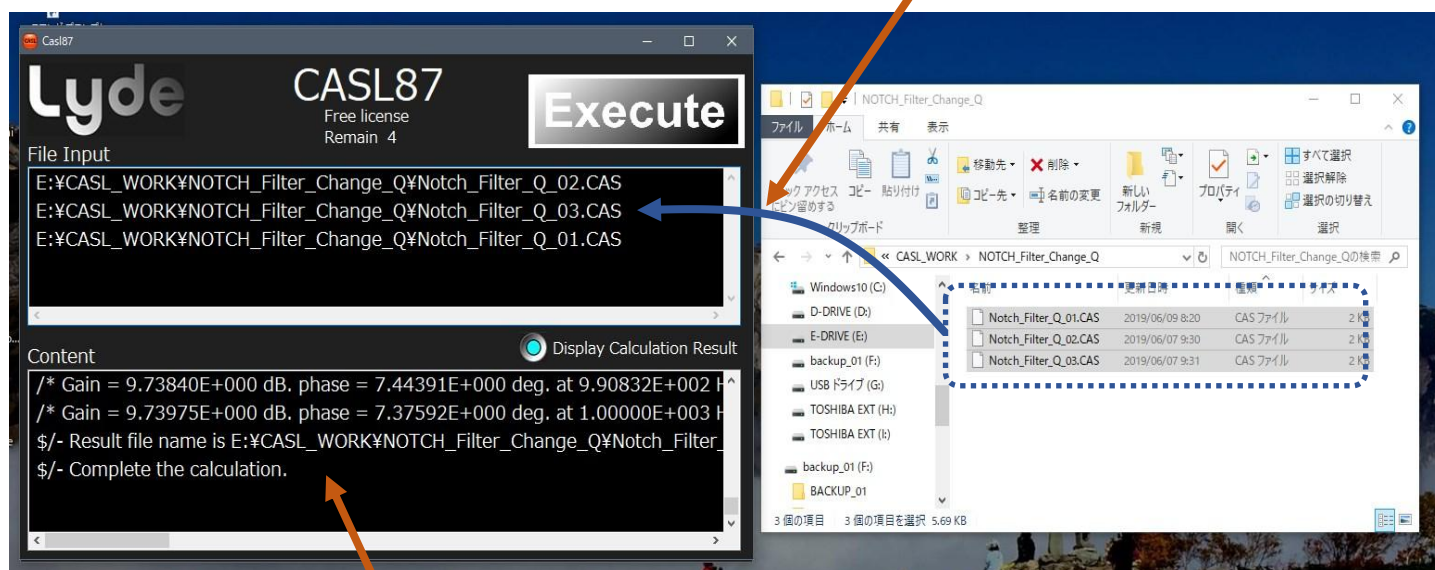
```

ソース・コード・ファイルを準備できたら、次に CASL87 を起動します。

上記の Notch_Filter_Q_01.Cas、その内容の Rfb1、Rfb2 の値を変更してある Notch_Filter_Q_02.Cas 及び、Notch_Filter_Q_03.Cas をまとめて選択します。

次ページに示すように、選択された複数のソース・コード・ファイルをマウスによって CASL87 ウィンドウ内の 'File Input' ボックスにドラッグ＆ドロップします。次に 'Execute' ボタン  を押して 3 つのソース・コードを一挙に計算をスタートします。

ソース・コードを複数選択して、ドラッグ＆ドロップする。



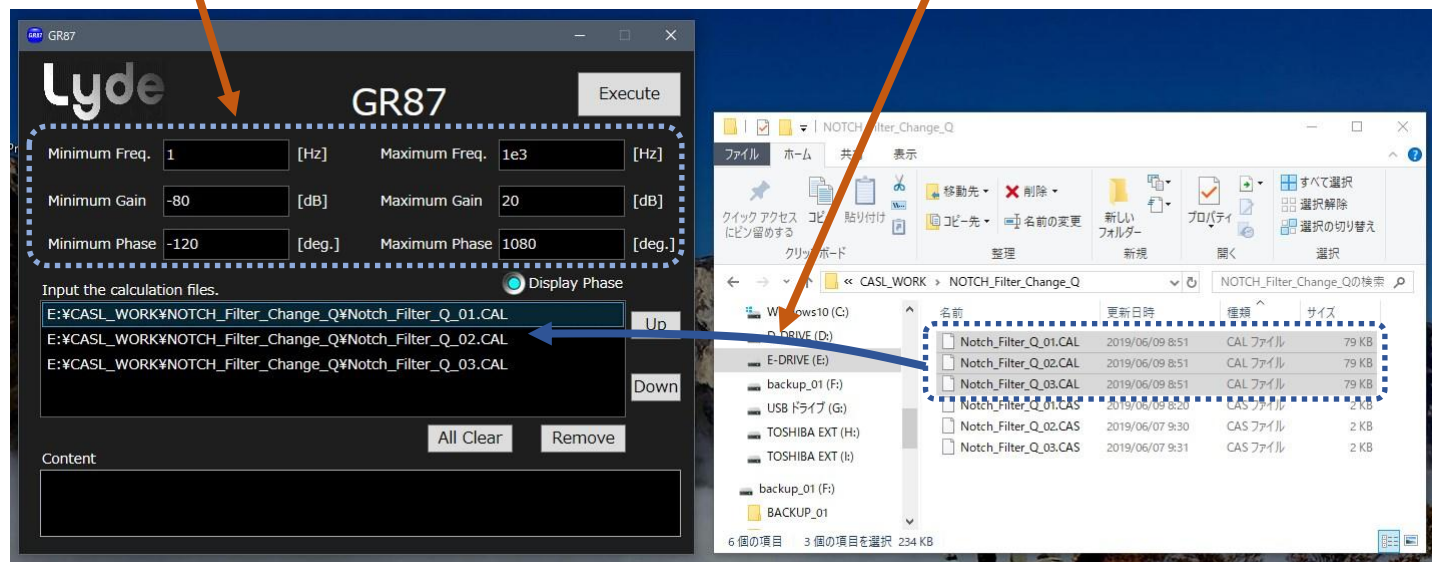
計算に成功したら '\$/- Complete the calculation.' と表示される。


エラーがなければ、'Content'ボックスに結果が表示されます。最後の行に '\$/- Complete the calculation.' と出力されたら計算が成功です。計算結果ファイルは、ソース・コードと同じフォルダ内にサフィックス '.CAL' のファイルとして出力されます。

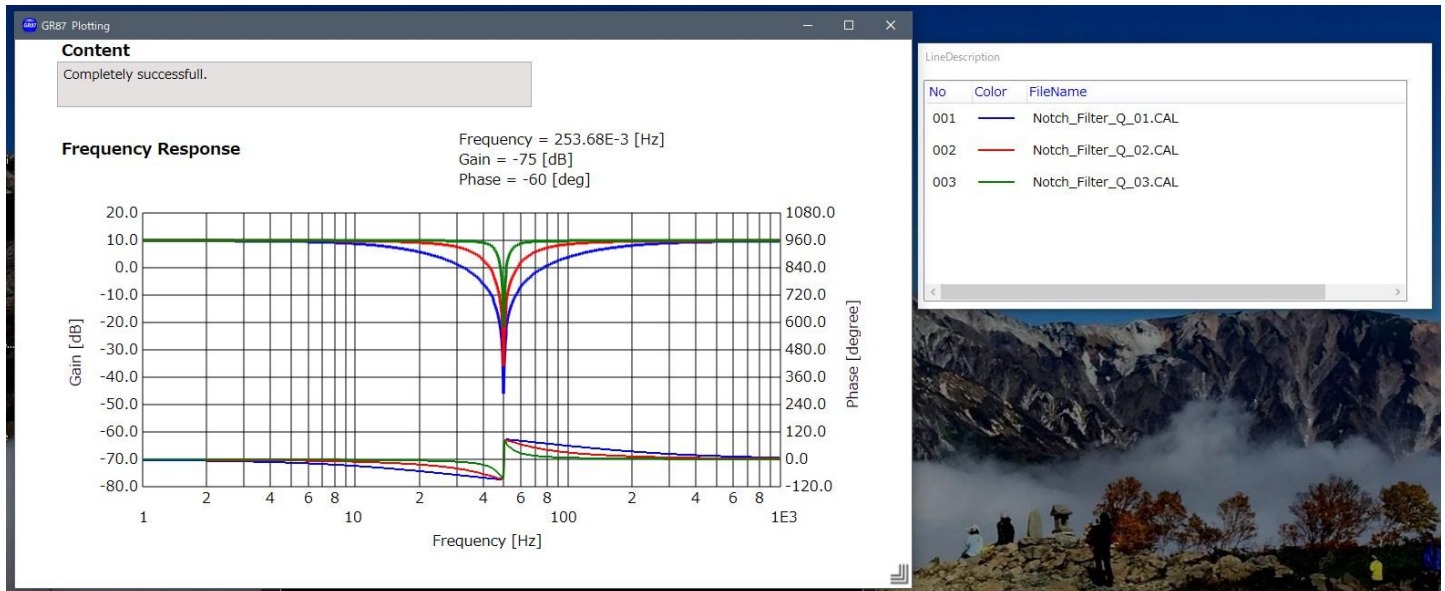
同じくインストールしたグラフ・プロッターの GR87 を起動します。計算結果ファイルを以下のようにまとめて選択し、'Input the calculation files.'ボックスにドラッグ＆ドロップします。必要なら、'Input the calculation files.'ボックス内の順序を 'Up'、'Down' ボタンでグラフへの表示順序を調整します。さらに出力グラフの最低、最高周波数やゲイン、位相の座標軸設定を行います。

出力グラフの座標軸設定を行う。

計算結果ファイルを複数選択して、ドラッグ＆ドロップする。



次に 'Execute' ボタン  を押して、グラフ・プロットをスタートさせます。次ページの図に示すようにグラフ・プロッターとグラフのリスト (色区分) のそれぞれのウィンドウ画面が表示されます。



CASL87 と GR87 は、このように複数のソース・コード・ファイルを扱えることで、回路の部品の定数を簡単に比較検討できる利点があります。

グラフ・プロットのウィンドウの右下にマウスの右クリックのドラッグでこのウィンドウの大きさを変えることができます。また、周波数、ゲイン、位相のプロット線の値を知りたい場合は、カーソルをその位置にもっていき、ウィンドウ内にそれらの値が表示されます。その位置でマウスの左ボタンをクリックすると表示がその値で固定されます。キャプチャー・アプリケーションを利用してグラフをレポートなどに利用する場合に便利です。再度クリックすると値の固定が解かれます。

■ 本アプリケーションのお問い合わせ先

お問い合わせについては、下記メール・アドレスへのメールにてお願いします。

e-mail: info@lyde-global.com