PL-U4101C1 PcWaveForm
Archi_1 Script 記述方法編

2012 年 9 月

Revision 1.04

お断り

記載されている会社名および製品名はその会社の所有する商標です。
記載された内容については事前のお断りなく変更させていただく場合がございます。
記載された内容は 2012 年 9 月現在のものです。ご使用にあたっては、本取扱説明書の内容を
十分お読みいただけますようお願い申し上げます。

本取扱説明書は、PDF 形式でプログラム CD の中に入っています。

株式会社 デイシー
〒198-0024
東京都青梅市新町 9-2190
電話: 0428-34-9860
メール: info@deicy.co.jp

※ Archi_1 Script とは PcWaveForm 上で記述し、任意の解析を行うことのできる機能を持った Script 言語体系の名称です。

改定履歴

発行日	Revision	内容
2011 年 10 月 15 日	1.00	Archi_1 Script 機能編から「記述の仕方」の項を分冊 初版
2011 年 12 月 20 日	1.01	誤字訂正
2012 年 7 月 25 日	1.02	16 章「バイナリファイルを生成する」を追加 17 章「演算関数を使用して数列の要素の削除/抽出/置換/連結/縫合/反転/並べ替え/生成する」を追加 18 章「演算関数を使用して数列の Index を検索する」を追加 19 章「演算関数を使用して心臓暴露解析を行う」を追加 20 章「演算関数を使用して周波数解析を行う」を追加
2012 年 8 月 30 日	1.03	19 章を 20 章に変更 20 章を 21 章に変更 19 章「演算関数を使用して計測データの基本処理を行う」を挿入追加 22 章「演算関数を使用して頻度解析/被害推定を行う」を追加
2012 年 9 月 7 日	1.04	19 章に記述追加 23 章「演算関数を使用して計測データの統計処理を行う」を追加

適用

本書(Archi_1 Script 記述方法編)は、Archi_1 Script 機能編から、Script 記述の仕方を別冊としたものです。
「Archi_1 Script 記述方法編」は、Archi_1 Script 構文の記述方法を説明します。なお、Archi_1 Script の機能操作ついては別冊「解析機能操作編」を、記述言語については別冊「Archi_1 Script 言語仕様編」、使用する演算関数については別冊「Archi_1 Script 演算関数仕様編」をご参照下さい。

本書の記述内容は、下記のプログラムバージョンに対応します。

PcWaveForm Ver.7.08 ～
PcWaveFormFANA Ver.5.38 ～
PcWaveFormWBV+ Ver.2.18 ～

PcWaveForm プログラムの取扱説明書構成は、下記の 5 部構成となっています。

- 基礎編
- 解析機能操作編
- Archi_1 Script 演算関数仕様編
- Arch_1 Script 言語仕様編
- Archi_1 Script 記述方法編(本書)

の5部構成となっています。

PcWaveForm プログラムの基本操作については基礎編を、解析機能の操作については解析機能操作編を参照下さい。

ご注意

- 本書は万全を記して作成しておりますが、万一、ご不明点や誤りなどお気づきのことがありましたらご連絡下さい。
- 本書の実行結果から生じるお客様の損害や不利益については、それが直接的、あるいは間接的を問わず一切責任を負いかねますのでご了承下さい。
- 本書は改良のため予告なしに変更する場合があります。
-
- 本書に記載された会社名や製品名は、各社の登録商標である場合がございます。

目次

1. Archi_1 記述の基本則

Archi_1 Script は PcWveForm の収録データからチャネル間演算して新規計測項目チャネルの追加や、幾つかの収録チャネルを統合したチャネルへの置き換えなどを行う「チャネル間演算機能」に Script 構文を追加し、計測データ処理用に特化した Script 言語です。従って、チャネルの扱い方や演算式の記述方法はチャネル間演算機能で記述する演算式記述と同様となります。

1. 1. 記述文字 本文の記述は原則半角小文字で記述します。但し、関数名は半角大文字となります。 文字列即値を記述する場合、ダブルコーテーションで囲んで記述し、半角/全角を問いません。 演算処理ブロック名はダブルコーテンションで囲まず半角/全角を問いません。

※ 文字列即値および演算処理ブロック名以外に全角文字を記述すると Syntax エラーとなります。全角文字で記述したスペースなどは表示上区別することが難しく、不正箇所を探せない場合があります。その場合、編集 tool の find(文字列検索)または replace(文字列置換)機能を使用して検索修正します。

1. 2. 構文構造

1. 2. 1. 演算文以外の構文構造 演算文以外の構文構造は、当該構文の機能を示すキーワードで開始され、引き数項目が続きます。キーワードとキーワード間、キーワードと項目間、項目と項目間の区切り文字はすべて半角スペースまたは改行で区切ります。なお、項目によって項目内に複数の項目内引数を記述可能な場合があります。項目内引数区切り文字はすべて半角カンマとなります。

Keyword1 Keyword2 項目1 項目2　項目3･･･

Keyword には第一キーワード(keyword1)と第二キーワード(keyword2)があり、構文種類によっては、第二キーワード(Keyword2)が存在しないものがあります。なお、引数項目の個数は構文により異なります。項目には、構文実行時に参照される項目と結果を格納する項目があります。

※ SyntaxCheck での留意点: 構文は項目区切りに改行が使用できるため、必須項目が不足した場合など文法チェックの結果表示されるエラー発生行番号が実際にエラーを含む行番号の次の行を示す場合があります。

記述例:

```
dcl menu_label “度数分布解析”
def sampl_period 1e-3 “sec”
read wave %1
write ch_column 1: &2,&3,$1,$4
```

dcl、def、read、write が Keyword1に相当しグループ分類、menu_label、sampl_period、wave、ch_column が Keyword2に相当し、個別機能を指します。また 1e-3、”sec”、%1、1:、&2,&3,$1,$4 が項目に相当します。なお、&2,&3,$1,$4 は一つの項目で、項目内に複数の項目内引数記述が可能な例で、半角カンマで区切ることにより同じ項目であることを表します。

1. 2. 2. 演算文の構文構造

演算文は、右辺に格納先チャネルを記述し、半角″ = ″で接続して演算式を記述します。 演算式は、演算子、定数、参照チャネル/収録チャネル、演算関数、括弧から構成され、インフィックス記法で記述します。 演算式中にスペースは記述できません。なお、イコール“=”の前後に半角スペースが必要です。なお、演算子を連続して記 述できません。詳細は、「2 章 演算処理を記述する」の「2.2.演算文の記述」を参照下さい。

代入先参照チャネル = 演算式 演算式中に記述する関数

の記述構造は次の様になります。 **関数名(引数,引数,引**

数･･･)

関数名は半角大文字で記述し、引数は半角括弧内に半角カンマで区切って記述します。関数の引数は即値、チャネル、演算式の何れも記述できます。

記述例:

```
$1 = 20*LGT(RRT(0.25,WAC(#1),0.01)/20e-6)
$2 “STD:” = SQR(SUM(#1-MEA(#1)^2)/LEN(#1))
```

LGT( )、RRT( )、WAC( )、SQR( )、SUM( )、MEA( )、LEN( )は演算関数、”*”、”/”、”+”は演算子、$ 1,$2,#1 はいずれもチャネルと呼称し変数を示します。

1. 3. 取り扱う定数

1. 3. 1. 数値定数(即値)

構文、または演算式内に記述し固定小数点/指数形式を問いません。

記述例:

1234.456、1.23456e+3

※ 定数記述で.01 は 0.01 と記述する必要があります。

1. 3. 2. 文字列定数(文字列即値)

構文、演算文、または関数の引数に記述し半角ダブルコーテーションで囲んで記述します

記述例:

"abcd"、"試験結果"

半角/全角を問いません。但し、文字列即値に半角ダブルコーテーション文字は記述できません。

1. 3. 3. ファイル番号

ファイル取扱関連構文の項目(引数)として記述し、先頭文字半角"%"に続き正整数>0 で記述します。

記述例:

%1、%21

ファイル番号は当該番号で定義したファイル内容の内部管理領域を指すもので、ファイル取り扱い文および演算処理ブロック呼出文の引数以外記述できません。

1. 3. 4. グラフ番号

グラフ取扱関連 Script の項目(引数)として記述し、先頭文字半角"@"に続き正整数>0 で記述します。

記述例:

@1、@3

グラフ番号は該当番号で定義したグラフ内容の管理領域を指すもので、グラフ取り扱い文および演算処理ブロック呼出文の引数以外記述できません。詳細は第 11 章「演算結果をグラフ表示する」を参照下さい。

1. 3. 5. ページ番号

結果シート取扱関連構文の項目(引数)として記述し、正整数>0 に続き半角コロン":"で記述します。

記述例:

1:、3:

結果シートの書き込みページ番号を表します。結果シートは演算結果を表示させる機能を持つ Window で、結果シートへの書き込みは実行終了直前に行います。なお、ページ番号は結果シート操作構文以外記述できません。
詳細は第 10 章「結果シートを使用する」を参照下さい。

1. 3. 6. バッファ番号

バッファ番号は、バイナリファイル生成時一時的に書き込むバッファを意味し、バイナリファイル生成関連構文の項目(引数)として記述し、半角"*"に続き正整数で記述します。

記述例:

*1、*2

バイナリファイルの取扱いは、第 16 章「バイナルファイルを生成する」を参照下さい。

1. 4. 取り扱う変数

Archi_1 Script では、変数はすべてチャネルと呼称し初期値は null となります。また、チャネルには属性が存在し、属性によって記述則が異なります。チャネルは、すべて番号で表し同じ属性で同じ番号は同一と見なします。

1. 4. 1. 数値属性参照チャネル

構文または演算文に記述し、先頭文字半角"$"に続きチャネル番号を正整数>0 で記述します。なお、チャネル番号は半角括弧内に記述しても同じ意味となります。

記述例:

$3、$(3)

1. 4. 2. 文字属性参照チャネル

構文、または結果が文字属性で戻る演算文左辺、演算式中の演算関数の文字属性引数として記述し、先頭文字半角"&"に続きチャネル番号を正整数>0 で記述します。なお、チャネル番号は半角括弧内に記述しても同じ意味となります。

記述例:

&3、&(50)

1. 4. 3. 収録チャネル

構文(参照項目)または演算式(演算文右辺)に記述し先頭文字半角"#"に続き解析対象ファイルの収録チャネル番号を記述します。

なお、チャネル番号は半角括弧内に記述しても同じ意味となります。 収録チャネルは、解析対象ファイルの記述したチャネル番号のデータを意味し、読み出し専用となり、代入操作は解析対象 ファイルを読み出した時点で自動的に行われます。解析対象ファイルが読み出しているか PcWaveForm で読み出し波形表 示 Window 上で解析範囲指定して実行されていないと値を持ちません。

記述例 :

#1、#(2)

1. 5. チャネルの構造

チャネルとは単数、または複数の要素からなる数列(配列)を意味します。

例えば収録チャネル ch1 が収録データ数 n 個とすると

ch1 ⇒ ch1(0) ch1(1) ch1(2) ch1(3) ch1(4) ch1(5)……ch1(n-1)

収録チャネル ch1 は#1 に対応し

#1 ⇒ #1(0), #1(1) #1(2) #1(3), #1(4) #1(5) ……#1(n-1)

#1 には、ch1 の先頭 Index から当該チャネルのデータ値が各要素に格納された n 個の要素を持つ数列となります。

また、チャネルは信号名、単位、属性、データ個数などが 1 セットで管理されます。

数値属性参照チャネルおよび文字属性参照チャネルは信号名、単位を付ける事ができ、データ個数は当該チャネルに値が格納された時に決定されます。

収録チャネルの信号名、単位、データ個数は解析対象ファイルを読み出した時、または PcWaveForm 上で解析範囲は指定されて実行に遷移した時に自動的に受け渡され、Archi_1 Script 記述上では格納されている内容を変更できません。

1. 5. 1. 信号名、単位の定義

参照チャネルに信号名/単位を付ける場合は参照チャネル信号名定義文、単位のみ付ける場合は参照チャネル単位定義文を用いて定義します。一旦、付けられた信号名/単位は新たに付け替えない場合は定義以降維持されます。

参照チャネル信号名定義文

def ch_name 参照チャネル番号 "信号名:単位

" def ch_name 参照チャネル番号 "信号名"

信号名は文字列即値記述以外、文字属性参照チャネルでも記述できます。

参照チャネル単位定義文

def ch_unit 参照チャネル番号 "単位"

単位名は文字列即値記述以外、予め単位名が格納された文字属性参照チャネルでも記述できます。

記述例:

```
def ch_name $2 "Aw:m/s^2"
def ch_name $3 &1          /*&1 には予め信号名又は信号名と半角コロンで連結した単位が格納されている必要があります*/
def ch_unit $3 "dB"
```

信号名、単位の定義は、代入文、演算文の左辺に記述し定義することもできます。

信号名が定義されている場合、作成編集中に、行番号にカーソルを移動すると当該行に記述されたチャネルが定義されている信号名に置き換えて表示します。

```
39 /*-----------近似線 演算----------------------------------------------------------------*/
40 $19 = ($15*$13-$10*$11)/($15*$12-$10^2)          /* 近似線傾き演算*/
41 近似線傾き = (N:サンプル数*Sxy-Sx*Sy)/(N:サンプル数*Sxx-Sx^2)ット演算*/
42 assign $21 = 5,10,100,1000,2000                  /* 近似線X生成*/
43 $22 = $19*LGT($21)+$20                          /* 近似線Y生成
```

また、結果シート表示時の列項目欄に信号名、単位を表示するほか、波形ファイル格納時に定義されている信号名、単位を波形ファイルのヘッダーに書き込まれます。

1.6. チャネルの指定法

方法と種類を下表に示します。下表構文とは先頭 Keyword で始まる構文を指し演算文を除いています。

内容	記述例	構文	演算式
チャネル全体を指す	#1,$1,&1	○	○
特定要素を指す(Index 指定)	#1(3),$1($2),&($2)	△	○
要素範囲を指す(Index 範囲指定)	#1[0,100],$2[$1+2,$3]	×	○
チャネル番号を変数で指す(間接指定)	#($1),$($2),&($3)	△	○
複数のチャネルを指す(連続指定)	#[3,4],$[10,$2],,,&[$2,$4]	○	×

1.6.1. index 指定

チャネルの特定要素を指す場合に記述します。チャネル番号に続く括弧内に指定する Index(データ番号)を記述します。Index 指定を参照チャネルで記述した場合、記述した参照チャネルが複数要素を持っていても参照される Index は 0 のみとなります。Index を示す参照チャネルの Index を指定したり間接指定、あるいはそれらを混合したりネスト構造で記述しても問題ありません。なお、構文によっては引数が複数要素を持つ参照チャネルで無ければならないものがあり、その場合は Index 指定記述できません。また、演算式の演算結果の Index を指定する事はできません。その場合は演算関数の数列切り出し関数(ERC 関数)を使用して切り出します。

記述例:

```
$1(3)              /* 構文、演算文(左辺/右辺)で使用できます。*/
$1($2)             /* 構文、演算文(左辺/右辺)で使用できます。*/
$1($2($3))         /* 構文では使用できません。演算文(左辺/右辺)で使用できます。*/
$1(($($3(1)))      /* 構文では使用できません。演算文(左辺/右辺)で使用できます。*/
$1($2+INT(SQR($3))) /* 演算式でのみ使用可能で、構文および演算文(左辺)には使用できません。*/
$1($1($3)+3)       /* 演算式でのみ使用可能で、構文および演算文(左辺)には使用できません。*/
```

※ 演算文左辺(代入先参照チャネル)を Index 指定する場合、指定した Index が代入先参照チャネルに存在している必要があります。

1.6.2. Index 範囲指定

チャネルの範囲を指定する場合、演算式のみ記述できます。[ ]には開始 Index とデータ個数を半角カンマで区切って記述します。演算関数の数列切り出し関数(ERC 関数)と同じ機能を持ちます。但し、ERC 関数の引数は開始データ番号と終了データ番号でとなり引数が異なります。

開始 Index およびデータ個数の指定を参照チャネルで記述する場合、Index 指定や間接指定、演算式でも指定できます。但し、記述した参照チャネルが複数要素を持っていても参照される index は 0 のみとなります。

記述例:

```
$1[$($3($2))+SUM($($2)),$3/2]
```

1.6.3. チャネル間接指定

チャネル番号は、それ自体数値で表現していますので、チャネル番号を括弧内に参照チャネルで記述してチャネル番号を 変数とする事ができます。この指定方法を間接指定と呼称します。

記述例:

```
$($2)          /* 構文によって記述できない場合があります。*/
$(($2($4))     /* 構文によって記述できない場合があります。*/
$($1($2)+1)    /* 演算式でのみ使用できます。*/
$(INT(SUM($2))) /* 演算式でのみ使用できます。*/
```

1.6.4. チャネル連続指定

チャネル番号を連続して指定する方法で構文に記述する項目(引数)でチャネル列と記載している項目のみ使用できます。演算文はチャネルごとの演算となりますので記述できません。 半角括弧内"[]"に先頭チャネル番号とチャネル数を半角カンマで区切って記述します。

記述例:

```
$[100,5]       /* $100,$101,$102,$103,$104 と記述する事と等価となります。*/
$[$2,$8]       /*  $2(0)が先頭チャネル番号、$8(0)がチャネル数を意味します。*/
```

※ 開始チャネル番号、チャネル数の指定に Index 指定あるいは間接指定は使用できません。

1. 7. 数値文字列変換フォーマットの指定法

数値を文字列に変換する場合や数値を表示する場合にフォーマットを指定できます。フォーマットを指定しない場合、原則 E 形式で変換されます。

1. 7. 1. E 形式(Exponential Format)

E 形式は仮数部と指数で表現され、先頭文字半角大文字"E"に続き小数点以下桁数を指定します。例えば E3 とした場合、0.0000123⇒1.23e-5、-12.345⇒-1.23e+1 と変換されます。

記述例:

```
format E3,E6
```

※ "format"は結果シートで表示するチャネル内容の表示形式を指定する構文です。記述例は本構文を使用していますが、それ以外にキーボードからの入力形式の指定やテキストファイルへの書き込み、数値属性から文字属性などへの変換などでも指定できます。

1. 7. 2. S 形式(Significant Format)

S 形式は浮動小数点形式で表現され、先頭文字半角大文字"S"に続き、符号および小数点を含まない全桁で指定します。例えば、S5 とした場合、-12.34567⇒-12.346、123.4567⇒123.46 と変換されます。

記述例:

```
format S3,S5
```

1. 7. 3. F 形式(Fractal Format)

F 形式は固定小数点で表現され、先頭文字半角大文字"F"に続き少数点以下桁数を指定します。 例えば、F2 とした場合、-123.456⇒-123.46、0.1234⇒0.12 と変換されます。

記述例:

```
format F3,F6
```

1. 8. 注釈文の記述

記述した内容の説明や記述を見易くする為の注釈文を記述できます。

注釈文は半角スラッシュ"/"と半角アスタリスク"*"の連続 2 文字で注釈文の開始を意味し、半角アスタリスク"*"と半角の連続 2 文字で注釈文の終結を意味します。注釈内容は半角/全角を問いません。 また、注釈文開始から終結まで複数行に跨っても問題ありません。その間の全ての行は注釈と見なします。

注釈文の挿入箇所は、構文の前後、または項目区切りに挿入できます。但し、記述する項目と注釈文開始"/*"文字の間、および終結"*/"文字と項目の間は半角スペース 1 文字以上必要となります。

記述例:

```
/* ------------------------------------------
                    実験データ解析
------------------------------------------*/
write ch_column 1: /* 結果書き込み ch1,ch2  */ #1,#2
/* 注釈 */ $1 = 10*LGT($3)  /*   注釈  */
```

1. 9. 実行メニューボタンラベルの宣言

構文でメニューボタンラベルを記述することができ、Archi_1 Script 実行メニューへの登録時に登録したボタンに表示します。

メニューボタンラベル宣言文

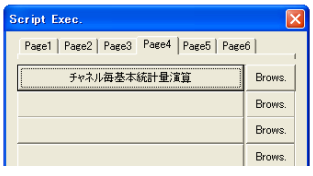
```
dcl menu_label "ラベル名" フラグ
```

本文を記述省略するとボタンラベルには Archi_1 Script ファイル名が表示されます。

フラグは 0 または 1 を記述し、1 と記述した場合、Script 実行後の実行終了通知ダイヤログを表示しません。なお、0 または記述省略した場合は 0 と見なし、Script 実行終了通知ダイヤログを表示します。

記述例:

```
dcl menu_label "チャネルごと基本統計量演算"
```



1. 10. 構文の種類と記述順

1. 10. 1. 宣言文

先頭 Keyword が **dcl** で始まる宣言文で同じ意味を持つ宣言文は唯一無二の必要があります。また、記述順は Archi_1 記述先頭に記述する必要があります。宣言文同士の記述順は問いません。言い換えれば Archi_1 Script の先頭部分は宣言文記述領域となります。

記述例:

```
dcl menu_label “script_exec”          /* Script 実行メニューボタンラベル名宣言文*/
dcl Sheet 1 {                         /* 結果シート宣言開始文*/
dcl exempt_ch $1,$2,$3                /* Index 制御禁止チャネル宣言文*/
dcl $err                              /* 実行時エラートラップ宣言文*/
dcl script_env 1,1,1,1                /* 実行時環境宣言文*/
```

1. 10. 2. 定義文

先頭 Keyword が **def** で始まる定義文で、当該定義文が参照される直前までに記述する必要があります。 一旦、定義すると新たに再定義されるまで有効となります。

記述例:

```
def sheet_tilte 0 “演算結果1”          /* 結果シート表題定義文(結果シート関連)*/
def folder_path “c:¥data”             /* カレントフォルダパス定義文(ファイル取扱関連)*/
def file_id %1 “test_wave “ wav       /* ファイル番号定義文(ファイル取扱関連)*/
def wav_sampling %1 0 1e4 “Hz” 0      /* 生成波形ファイルサンプリング定義文(波形ファイル生成関連)*/
def wav_type %1 float                 /* 生成波形ファイルデータ形式定義文(波形ファイル生成関連)*/
def wav_comment %1 1 “test”           /* 生成波形ファイルコメント定義文(波形ファイル生成関連)*/
def wav_mark %1 123 “mark1”           /* 生成波形ファイルマーク定義文(波形ファイル生成関連)*/
def wav_datetime %1 &1 &2 0           /* 生成波形ファイル開始日付時刻定義文(波形ファイル生成関連)*/
def wav_ch_series %1 $1               /* 生成波形ファイルチャネル番号定義文(波形ファイル生成関連)*/
def text_form %1 100,10               /* テキストフォーム行列数定義文(テキストファイル関連)*/
def ch_name $1 “データ個数”           /* チャネル名定義文(演算処理関連)*/
def ch_unit $1 “m/s^2”                /* チャネル単位名定義文(演算処理関連)*/
def sampl_period 1e-4 “sec”           /* サンプリング周期定義文(演算処理関連)*/
def num_samps 10000                   /* 処理データ個数定義文(演算処理関連)*/
def inherit_ch $1,.$2,$3              /* 引き継ぎチャネル定義文(サブルーチン処理関連)*/
def local_ch $1,$2,$3                 /* ローカルチャネル定義文(サブルーチン処理関連)*/
def graph_id @1 “graph1”              /* グラフ番号定義文(グラフ描画関連)*/
def graph_x_axis @1 0,0 F1 log        /* グラフ X 軸定義文(グラフ描画関連)*/
def graph_y_axis @1 0,0 F0            /* グラフ Y 軸定義文(グラフ描画関連)*/
def graph_line @1 $1,$2,$3 &1         /* グラフ描画線種線色定義文(グラフ描画関連)*/
def graph_aspect_ratio @1 2           /* グラフ描画枠縦横比定義文(グラフ描画関連)*/
def graph_ch_series @1 $1             /* グラフ表示チャネル番号定義文(グラフ描画関連)*/
def graph_line_draw @1 $1,$2,$3       /* グラフ枠内追加線描画定義文(グラフ描画関連)*/
def bin_buf *1 4096                   /* バイナリバッファ定義文(バイナリファイル関連)*/
```

1. 10. 3. 入出力文

先頭 Keyword が **read write save** で始まる構文で、 read で始まる入力文はファイルからの読み出し、あるいは内部パラメータの読み出しを行います。 write で始まる出力文はファイル出力を除く結果シートへの書き込みやテキストフォームへの書き込みなど内部出力を行います。 save で始まる出力文はファイル格納など外部機器への出力を行います。記述順は規定されませんが、本文が参照する定義文以降に記述する必要があります。

記述例:

```
read file_info &1 &2 &3 $1            /* フォルダ内ファイル情報取得文(ファイル関連)*/
read folder_path &1                   /* カレントフォルダパス取得文(ファイル関連)*/
read file_status %1 $1                /* ファイルステータス取得文(ファイル関連)*/
read wave %1                          /* 波形ファイル読出し文(波形ファイル関連)*/
read file_name &1 &2                  /* 解析対象ファイル名取得文(波形ファイル関連)*/
read num_ch $1                        /* 波形ファイルチャネル数取得文(波形ファイル関連)*/
read ch_series $1                     /* 波形ファイルチャネル番号取得文(波形ファイル関連)*/
read ch_name &1 &2 $1                 /* 波形ファイル信号名取得文(波形ファイル関連)*/
read comment $1 &1                    /* 波形ファイルコメント取得文(波形ファイル関連):/
read num_mark $1                      /* 波形ファイルマーク個数取得文(波形ファイル関連)*/
read mark_pos 1 $1                    /* 波形ファイルマーク位置取得文(波形ファイル関連)*/
read mark_memo 1 &1                   /* 波形ファイルマークメモ取得文(波形ファイル関連)*/
read header_info &1 $1 &2             /* 波形ファイルヘッダー読み出し文(波形ファイル関連)*/
```

```
save wave %1 $[1,10]                    /* 生成波形ファイル格納文(波形ファイル関連)*/
read sheet %1                           /* 結果シート読み出し文(結果シート関連)*/
write ch_column 1: $1,$2                /* 結果シートチャネル列書き込み文(結果シート関連)*/
write memo_column 1: "memo"             /* 結果シートメモ列書き込み文(結果シート関連)*/
write line_feed 1: 1                    /* 結果シート行揃え文(結果シート関連)*/
save sheet %1                           /* 結果シート格納文(結果シート関連)*/
save pos_info %1 $1 $2 $3 1             /* データ位置情報格納文(波形ファイル関連)*/
read text_form %1                       /* テキストファイル読み出し文(テキストファイル関連)*/
read num_cell %1 $1                     /* テキストフォーム行列数取得文(テキストファイル関連)*/
read cell %1 10,2 0 $1                  /* テキストファイルセル読み出し文(テキストファイル関連)*/
write cell %1 11,4 0 $1                 /* テキストフォームセル書き込み文(テキストファイル関連)*/
save text_form %1                       /* テキストフォーム格納文(テキストファイル関連)*/
plot @1 $1 $2                           /* グラフ描画文(グラフ描画関連)*/
save plot %1 @1 $1 $2                   /* グラフ格納文(グラフ描画関連)*/
read bin %1 0 1200 $1 $1 int16          /* バイナリファイル読み出し文(バイナリファイル関連)*/
read bin_struct %1 0 10<$[1,3]int16>    /* バイナリファイル構造体読み出し文(バイナリファイル関連)*/
write bin_buf *1 $1                     /* バイナリバッファ書き込み文(バイナリファイル関連)*/
save bin_buf %1 *1,*2                   /* バイナリバッファ格納文(バイナリファイル関連)*/
write progress_status &1                /* プログレスバー表示文(演算処理関連)*/
```

1. 10. 4. 演算処理ブロック制御文

先頭 Keyword が **case repeat_case index repeat_index mark repeat_mark** で始まる構文で演算処理ブロックの実行を制御します。記述順は規定されませんが、直下は制御対象となる演算処理ブロック開始文の必要があります。

記述例:

```
index 100,500                           /* データ番号実行制御文*/
mark 2 4                                /* マーク番号実行制御文*/
case $1 < $2                            /* 条件成立実行制御文*/
case ture $1                            /* 論理成立実行制御文*/
case false $1                           /* 論理不成立実行制御文*/
repeat_index $1 1000 2000               /* 繰り返しデータ番号実行制御文*/
repeat_mark $1 2 6                      /* 繰り返しマーク番号実行制御文*/
repeat_case $1 < 6                      /* 繰り返し条件成立実行制御文*/
repeat $1 0 1 10                        /* 繰り返し実行制御文*/
```

1. 10. 5. 代入文

先頭 Keyword が **assign** で始まる構文で、数列や文字列配列の生成、文字列の連結、数値文字列変換および文字列数値変換を行います。記述順は規定されません。

記述例:

```
assign $1 = 1,2,3,4,5,6,10<12>
assign &1 = $1(F0)|"Hz",&1
assign $1 = {1,2,3,4,5,6,7,8,9,10,11,12}
```

1. 10. 6. 演算文

先頭 Keyword は無く、直接演算結果代入先参照チャネルから始まります。記述順は特に規定されません。

記述例:

```
$1 = $2/200+SUM($1)
$2 = SUM($1)/LEN($1)
```

1. 10. 7. ユーザ・インターフェイス文

先頭 Keyword が **get disp** で始まる構文で、**get** で始まる構文はディスプレイにダイアログを表示し、何らかの応答を受け取ります。 **disp** で始まる構文はディスプレイにメッセージを表示したり通知ダイアログへの書き込みなどを行います。記述順は規定されません。

記述例:

```
get file_name &1 $1 "hdr"               /* ファイル選択文*/
get value $1,$2,&1                      /* 値入力文*/
get folder_select                       /* フォルダ選択文*/
get replace_ch #1,#2.#3                 /* 収録チャネル割り当て文*/
get param %1 $1                         /* 諸元表ファイル参照文*/
get reply &1 &2 &3 $1                   /* メッセージ表示応答取得文*/
disp message $1(F0)                     /* メッセージ表示文*/
```

```
get check_box_status &1 $1              /* チェックボックスステータス取得文*/
get text_page &1,&2,&3 $1               /* テキスト表示選択文*/
get radio_button_status &1 $1           /* ラジオボタンステータス取得文*/
```

1. 10. 8. 文字列操作文

先頭 Keyword が **char** で始まる構文で、演算関数を使用しない場合の文字列操作を行います。記述順は時に規定されません。なお、文字列操作文は同じ機能を持つ演算関数が用意されています。

記述例:

```
char num_element &1 $1                  /* 文字属性配列要素数数取得文*/
char length &1 $1                       /* 文字列長さ取得文*/
char extract &1 1 10 &2                 /* 文字列切り出し文*/
char find &1 "abc" 1 &2                 /* 文字列検索文*/
char recomposition &1 $1 &2             /* 文字列配列再構成文*/
char delete &1 1 4 &2                   /* 文字列削除文*/
char insert &1 "abc" 3 &2               /* 文字列挿入文*/
har replace &1 "abc" 2 "cdf" &2         /* 文字列置換文*/
```

1. 10. 9. その他の構文

先頭 Keyword が **call clr end** で始まる構文で、 **call** で始まる構文は外部演算処理ブロック(end 行以降に記述された演算処理ブロック)の読み出し、dll 形式で提供された Subroutine の呼出し、EXCEL への実行権の移譲などを行います。 **clr** で始まる構文は結果シートの初期化、実行時エラーフラグの初期化などを行います。なお、**end** は記述本文の最終行に記述します。その他の構文の記述順は規定されません。

記述例:

```
clr Sheet 1:                            /* 結果シート指定ページ初期化文*/
call sub %2 Mttfunc1 $1,$2,$3           /* dll形式サブルーチン呼び出し文*/
call proc グラフ $1,$2,$3               /* 外部演算処理ブロック呼び出し文*/
call excel "報告書"                     /* エクセル実行文*/
delete file %1                          /* ファイル削除文(ファイル関連)*/
disp value $1                           /* 通知ダイヤログ書き込み文*/
end                                     /* Script 実行終了文*/
```

2. 演算処理を記述する

2. 1. 代入文の記述 代入文は格納先チャネルに、複数の値を一度に代入する場合、文字属性チャネルに文字列即値を代入する場合、数値属性チャネ ルを文字属性チャネルに代入する場合、また逆に文字属性チャネルから数値属性チャネルに代入する場合に使用します。なお、代 入文右辺に参照チャネル或いは収録チャネルに値が存在していない、Null のチャネルは記述できません。

文法:

assign 格納先参照チャネル “信号名:単位” = 代入値列

代入値列とは参照チャネル/収録チャネルおよび即値/文字列即値を意味し演算式および演算関数は記述できません。 代入先チャネルに信号名、単位を演算文で定義する場合は、信号名と単位を半角コロン”:”で区切り全体をダブルコーテー ションで囲み左辺に記述します。なお、信号名の定義は単位の記述を行わない場合でも”=”の区切り文字“半角コロンは必要です。なお、代入文の代入値列は構文の項目に相当する為、半角カンマ”,”で区切って記述する必要があり、行を跨いで記述することはできません。テーブル定義の様に多くの値からなる代入値列を記述する場合は特殊な代入文として後述しま す。

代入値列の記述則

①格納先属性が文字列属性で代入する属性が数値属性の場合 数値属性参照チャネル或いは収録チャネルの場合、文字列変換フォーマットをチャネル番号に続き半角括弧内に記 述できます。フォーマット記述を省略すると指数形式で変換されます。

$1(F0),#1(E6)

②格納先属性が文字列属性で、文字列連結して代入する場合
文字列連結は半角”|”で接続します。

$1(F0)|”Pa “|#1(E6)|”m/s^2”

③同じ代入値を繰り返して代入する場合 半角”<”と”>”で
囲んで繰り返し数と要素値を記述します。 **10<1,2,3>、**
3<”a”>
10<1,2,3>は 1,2,3,1,2,3,1,2,3,,,と 10 回繰り返して代入され、3<”a”>は”a”,”a”,”a”と”a”が 3 回繰り返して代入されます。

記述例:

文字属性から数値属性への変換

assign &1 = “ABC=12.3e-2.123m/s^2”
assign $1 = &2

$1 には 12.3e-2 が代入されます。文字列に含まれる最初に出会う数値変換可能な地点から変換格納します。数値に変換できない場合は 0 が代入されます。

記述例:

数値属性から文字属性への変換

assign $1 = 123.4
assign &1 = $1(F2)

&1 には”123.40”が代入されます。書式を指定しない場合は指数形式で代入されます。

記述例:

assign &1 “地名:” = “Tokyo”,”Osaka”,”Nagoya “,10<”a“>

&1 は 13 個の要素を持ち、&1(0)⇒”Tokyo”、&1(1)⇒”Osaka”、&1(2)⇒”Nagoya”が代入され、&1(3)～&1(12)迄は”a”が代入されます。
要素区切りは半角カンマで記述します。10<”a“>は要素数 10 個の文字列”a”を意味します。

記述例:

assign &2 = “ch(”|$2(2)(F0)|”) Name:”|&3(2),“ch(”|$2(3)(F0)|”) Name:”|&3(3)

$2(2)は 5、$2(3)は 8、&3(2)は”ACC_X”、&3(3)は”ACC_Y”とすると、&2 は 2 個の要素を持ち、&2(0)は”ch(5) Name:ACC_X”、&2(1)は“ch(8) Name:ACC_Y”が代入されます。 文字列の連結は半角“|”で記述します。数値から文字列変換する場合にフォーマット指定しています。

記述例:

assign $1 = $1,100<1>,&4,123.4,1.23e-3

$1 は要素数 50 個、&4 は要素数 50 個とすると、代入結果の$1 は要素数 202 個を持ち、$1(0)～$1(49)は元の$1 の内

容、$1(50)～$1(149)は1となり、$1(150)～$1(199)は＆4 のそれぞれの要素の先頭から数値変換可能範囲を数値に変換した値が代入され、$1(200)は 123.4、$(201)は 1.23e-3 となります。

テーブル定義などに使用する複数の値を一度に代入する特殊な代入文

文法：

assign 格納先参照チャネル “信号名:単位” = { 値,
値,値,値
値,値,値,値
値,値,値,値}

記述する値は即値または文字列即値の何れかに限定され、その属性は格納先参照チャネルに依存します。この記述方法は例えば、テーブル定義など、多くの項目を記述する場合に、代入値列の項目内区切りに C/RL/F を許した記述方法です。

記述例：

```
assign $1 “解析テーブル:” =
{60,61,65,66,69,71,73,75,75,78,80,86,95,95,96,100,100,102,104,107,107,109,110,111,112.113
113,114,114,114,115,116,117,117,117,118,118,121,122,125,125,126,126,126,128,128,129,129
130,130,134,137,138,138,138,139,140,140,141,142,143,143,143,144,146,147,147,149,150,152
155,157,157,159,159,162,164,164,165,167,170,170,171,171,174,175,175,175,175,175,176,176
176,176,178,180,182,182,182,183,183,186,188,189,190,192,193,196,198,199,199,209,210,212
213,216,216,217,219,220,220,221,223,224,224,225,225,227,228,228,230,234,245,248,253,253
260,265,266,270,277,280,283,286,287,294,299,304,304,307,308,308,346,347,349,365,385,402
409,501,907}
```

$1 は要素数 161 個を持つ数列となり、その要素には 60～907 が順次格納されます。

2. 2. 演算文の記述 演算文は、演算結果格納先と演算式を半角スペース+半角イコール”=”+半角スペースで接続します。なお、演算式中にスペース は存在できません。代入先チャネルに信号名、単位を演算文で定義する場合は、信号名と単位を半角コロン”:”で区切り全体をダブ ルコーテーションで囲み記述します。単位の記述を行わない場合でも区切り文字“半角コロンは必要です。
格納先チャネルに Index を指定して格納する場合、既に当該 Index が存在している必要があります。また、格納先チャネルに Index を指定した場合、演算式の結果が複数要素を持つ数列でも格納される値は Index0 のみとなります。

文法：

格納先参照チャネル = 演算式 格納先参照チャネル “信号名:単位” = 演算式

2. 2. 1. 演算式の記述

演算子は加減乗除およびべき乗は”+”,”-“,”*”,”/”,”^”で記述し、演算式記法はインフィックス記法(例:1+2)となります。 但し演算子を連続して記述できません。従ってマイナス値を掛算する場合などは()で繰る必要があります。演算順序は括弧⇒べき乗算⇒乗除算⇒加減算の順序で、同じ優先度の場合、左側優先となります。演算式とは即値、参照チャネル/収録 チャネルあるいは演算関数を用いた演算式を意味し、関数の引数に関数あるいは演算式を記述することができます。また、 使用する関数により属性の異なる演算が禁止されている場合は実行時エラーとなります。 文字属性参照チャネルは演算子を用いた演算は行えません。なお、文字列即値は関数の引数としてのみ記述可能であり直 接演算式には記述できません。演算式を関数の引数と記述する場合は演算式の戻り属性が指定されている引数の属性と一 致している必要があります。 演算は数列同士の演算が基本となり、複数要素をもつチャネルの場合、オペランドが同じ要素数の場合は要素ごとに演算さ れ、オペランドの何れかが 1 個の要素の場合は複数要素を持つ数列の要素ごとに同じ値で演算され、オペランド双方何れも 複数の要素を持つ場合は何れか少ない要素数に縮退され演算されます。チャネルの要素を Index で指定する場合はチャネ ルに続き括弧()に Index を記述します。

記述例：

```
$1 “四則演算:” = ($2-$3)/($2+$3)*100
$4 “関数演算:dB” = 20*LGT($1/1e-5)
$5 = $1*(-$2)                    /* $1*-$2 の記述は演算子が連続する為,括弧で括る必要があります*/
```

※ インフィックス記法とは、オペランドとオペランドの間に演算子（オペレーション）を挟んで記述する方法を指し、括弧を用いて演算順序を規定する記法です。

原則、演算式中に値を持たない null チャネル（直前までに値が代入されていない参照チャネル或いは、存在しない収録チャネル）を記述することはできません。但し、LNK 関数（連結関数）はその限りに無く、連結元に Null の数値属性参照チャネルを記述できます。詳細は「17.3. 数列の連結」の項を参照下さい。

記述例2. 1. 参照チャネルに数列(配列)を生成する

<Archi_1 Script 記述例>

```
1          /*------ calc2.proc ---------*/
2          $1 = DAG(1000,0)            /* 要素数 1000 個、初期値 0 */
3          assign $2 = 1000<0>         /* 要素数 1000 個  初期値 0 */
4          assign &1 = 1000<" ">       /* 要素数 1000 個  初期値半角スペース */
5          assign $3 = 1,3,5,8,9,11    /* 要素数  6個  */
```

<Archi_1 Script 記述構文の説明> 先頭文字"#"で表す収録チャネルは読み出し専用で必ず複数要素を持つ数列で構成されていますが、参照チャネルは当該チャネル に代入される演算結果等の要素数により自動決定されます。数値属性参照チャネルの配列(数列)生成は波形生成関数で直流とし た引数を与えて配列生成する方法、または代入文で複数要素代入する方法の 2 種類あります。

2 行目:　波形生成関数を使用して要素値 0 の要素数 1000 個の配列(数列)を生成します。

3 行目:　代入文を使用して要素値0の要素数 1000 個の配列(数列)を生成します。

4 行目:　代入文を使用して要素値半角スペース、要素数 1000 個の文字属性配列を生成します。

※ 文字属性参照チャネルの配列生成は代入文を使用して生成する方法に限定されます。

5 行目:　代入文を使用して各要素値を半角カンマで区切って記述し要素数 6 の数列を生成します。

記述例2. 2. 異なった要素数を持つ数列同士を演算する

<Archi_1 Script 記述例>

```
1          /*------ calc3.proc ---------*/
2          assign $1 = 1,2,3,4,5,6,7,8          /* $1 は要素数 8*/
3          assign $2 = 4,6,8,10,12,14          /* $2 は要素数 6*/
4          assign $3 = 10,20,30                /* $3 は要素数 3*/
5          $4 = (($1+$2)*2+$3)/2               /* 演算結果$4 は要素数 3 に縮退*/
```

<Archi_1Srcipt 記述構文の説明>

要素数の異なる複数要素を持つ数列同士の演算は、何れか少ない要素に切り捨てられて要素ごとに演算されます。複数要素を持つ数列と単一要素を持つ値との演算は各要素に単一要素が演算され要素数は変化しません。

2 行目:　代入文で、演算文で使用する数列を要素数 8 個として$1 に生成します。

3行目:　代入文で、演算文で使用する数列を要素数 6 個として$2 に生成します。

4 行目:　代入文で、演算文で使用する数列を要素数 3 個として$3 に生成します。

5 行目:　演算文で、$1,$2,$3 を使用して要素数の異なった数列同士を演算します。

($1＋$2)の結果 ⇒ 要素数 6 で 5,8,11,14,17,20

($1+$2)*2 の結果 ⇒ 要素数 6 で 10,16,22,28,34,40

($1+$2)*2+$3)の結果 ⇒ 要素数 3 で 20,36,52

($1+$2)*2+$3)/2 の結果 ⇒ 要素数 3 で 10,18,26 となります。

記述例2. 3. 演算式に関数を使用する

<Archi_1Script 記述例>

```
1          /*------ calc4.proc ---------*/
2          $1 = ACC(DAG(1000,1))   /* 要素数 1000 個、1,2,3,....1000 の数列を生成 */
3          $2 = SQR(SUM(($1-MEA($1))^2)/LEN($1)) /* 標準偏差関数 STD(X)と等価 */
```

<Archi_1 Script 記述構文の説明>

演算関数の引数に演算関数を使用できます。

2 行目:　波形生成関数:DAG と累積関数:ACC を使用して 1～1000 の数列を生成します。

3 行目:　平均値関数:MEA、合計値関数 SUM、データ個数関数:LEN、平方根関数:SQR を使用して標準偏差を演算します。

なお、標準偏差演算は標準偏差関数を使用し STD($1)としても求めることができます。

3. 演算処理ブロックを構成する

演算式／代入文および構文などを一塊のブロックとして構成したものを演算処理ブロックと呼称します。 演算処理ブロックはプログラムロジックを構成する、プログラム内で共通に使用する、などの場合に必要となります。

3. 1. 演算処理ブロックの種類

演算処理ブロックには内部演算処理ブロックと外部演算処理ブロックがあり、記述形式は何れも同じで総称して演算処理ブロックと呼称します。

3. 1. 1. 内部演算処理ブロックとは

演算処理ブロック実行制御文の直下に記述され、直前の演算処理ブロック実行制御文の条件が成立した場合に実行される演算処理ブロックを指します。

3. 1. 2. 外部演算処理ブロックとは

プログラム終了文(end 文)以降に記述し、演算処理ブロック呼出文(call proc 文)により呼び出されて実行される演算処理ブロックを指します。

3. 2. 演算処理ブロックの記述

演算処理ブロックは演算処理ブロック開始文と演算処理ブロック終了文で囲んだ範囲が演算処理ブロックとなります。また、演算処理ブロック内に演算処理ブロックをネスト記述することもできます。

演算処理ブロック開始文
文法:

```
proc 演算処理ブロック名{
```

演算処理ブロック終了文
文法:

```
}演算処理ブロック名
```

演算処理ブロック名の記述はダブルコーテーションで囲まず、記述文字は半角/全角を問いません。
なお、演算処理ブロック開始文と演算処理ブロック終了文で一致している必要があります。

記述例:

```
proc 演算書き込み処理{
  $1 = MEA((#1+#2)/2)
  write text_form %1 2,3 0 $1
}演算書き込み処理
```

3. 3. 演算処理ブロック内で定義するチャネル

3. 3. 1. ローカルチャネル

通常記述しているチャネル番号はグローバルチャネル(共通チャネル)であり演算処理ブロックの内外に関係なく同じチャネル番号で有れば同じものとして扱われます。ローカルチャネルとは演算処理ブロック内だけで有効なチャネルを意味し、同じチャネル番号であっても別のものとして取り扱うチャネルを意味します。

ローカルチャネル定義文
文法:

```
def local_ch チャネル列
```

定義できるチャネルの属性は数値属性参照チャネル($n)、文字属性参照チャネル(&n)のみとなります。
ローカルチャネル定義されたチャネルは当該演算処理ブロック内だけで有効なチャネルとなります。
但し、結果シートでの取扱はローカルチャネルとグローバルチャネル(共通チャネル)の区別が有りませんので演算処理ブロック内から書き込むと同じチャネル番号列に表示されます。また、チャネル信号名定義文、チャネル単位定義文も同様にローカルチャネル、グローバルチャネルの区別がありませんので演算処理ブロック内でローカルチャネル定義したチャネルに信号名/単位を定義すると、その外側のチャネルも同じ信号名/単位に変わります。

記述例:

```
$1 = 100
$2 = 50
proc test{
  def local_ch $1
  $1 = 200
```

```
    disp message $1(F0),$2(F0)   /* 演算処理ブロック内*/
  }test
  disp message $1(F0),$2(F0) /* 演算処理ブロック外 */
  end
```

参照チャネル$1 は初めに 100 を代入します。演算処理ブロック演算処理ブロック内では$1 をローカルチャネル定義して 200 を代入します。演算処理ブロック内のメセージ表示文では$1 は 200、$2 は 50 と表示されます。演算処理ブロック外でのメッセージ文では$1 は 100、$2 は 50 と表示されます。つまり、ローカルチャネル定義した$1 は演算処理ブロック内の$1 と演算処理ブロック外の$1 とは同じチャネル番号であっても別のものとして取り扱われています。

3. 3. 2. 引き継ぎチャネル

引き継ぎチャネルとは演算処理ブロック呼出文から受け渡される引数を演算処理ブロック内で引き継ぐチャネルを意味します。引数が記述されていない演算処理ブロック呼出文で実行される演算処理ブロックあるいは内部演算処理ブロックでは記述できません。

引き継ぎチャネル定義文

文法:

```
def inherit_ch チャネル例
```

演算処理ブロック開始文の直下に記述します。記述するチャネル数、およびチャネル属性は演算処理ブロック呼出文で記述される引数と一致している必要があります。定義したチャネル番号が読み出し元で使用されていても演算処理ブロック内では引き継いだチャネル番号として扱われます。

※ 演算処理ブロック内でlocal定義されていないチャネルが、呼び出し元の引数として記述した場合、演算処理ブロック内で使用すると、引き継ぎチャネルと同じとして扱われる事に注意下さい。例えば、呼び出し文で$1,$2,$3 を引数として受け 渡し、演算処理ブロック内でInherit_chとして$4,$5,$6と定義した場合、$1は$4、$2は$5、$3は$6として扱われますが、同じ演算処理ブロック内で$1,$2,$3をLocal_chとして定義していない場合、$1は$4、$2は$5,$3は$6と同じチャネルとして扱われることを意味します。

記述例:

```
def inherit_ch $1,$2,#1,%1
```

記述できるチャネルの属性は数値属性参照チャネル($n)、文字属性参照チャネル(&n)、収録チャネル(#n)およびファイル 番号の 4 種類です。ページ番号はグローバル扱いしかできません。

3. 4. 外部演算処理ブロックの呼び出し

演算処理ブロック呼出文

文法:

```
call proc 演算処理ブロック名 引数 1,引数 2,引数 3....
```

外部演算処理ブロックを呼び出し実行する機能を持ちます。呼び出し先演算処理ブロックの中で引き継ぎチャネルが定義されている場合は引数の記述が必須となります。逆に引き継ぎチャネルが定義されていない演算処理ブロックの呼び出しに引数を記述することはできません。また、引数を Index 指定する場合、外部演算処理ブロックから出力/入出力パラメータとして使用できません。

記述例:

```
$10 = 100
$20 = 50
call proc 足し算 $10,$20,$30
end
proc 足し算{
  def inherit_ch $1,$2,$3
  def local_ch $4
  $4 = $1+$2
  $2 = 1224
  $3 = $4*2
}足し算
```

演算処理ブロック呼出文からの戻り$10 はそのまま 100、$20 は 1224、$30 は 300 となります。引き継ぎチャネルで定義した$1:入力パラメータ、$2:入出力パラメータ、$3:出力パラメータの役割を持ち、外部演算処理ブロック内では$1　が$10、$2 が$20、$3 が$30 として扱われます。

記述例3. 1. 引数を持つ外部演算処理ブロックを実行する

解析対象ファイルの座位 X 軸加速度、Y 軸加速度、Z 軸加速度から全身振動暴露演算を行う外部演算処理ブロックを呼び出し実行します。

<Archi_1 Script 記述例>

```
1        /*--------calc10.prc----------*/
2        dcl menu_label "立位・座位・振動暴露演算"
3        def ch_name $1 "X 軸補正加速度実効値:m/s^2"
4        def ch_name $2 "Y 軸補正加速度実効値:m/s^2"
5        def ch_name $3 "Z 軸補正加速度実効値:m/s^2"
6        def ch_name $4 "合成加速度実効値:m/s^2"
7        call proc wbv_calc 0.25,#[1,3],$[1,4]
8        $5 = (ACC(DAG(LEN(#1),1))-1)*PRD()
9        /*----グラフ描画処理----------*/
10       def graph_id @1
11       def graph_x_axis @1 0,0 F2
12       def graph_y_axis @1 0,0 F3
13       plot @1 $5 $[1,4]
14       end
15       /*----外部演算処理ブロック--------*/
16       proc wbv_calc{
17       /* in  $1: 時定数,#10:X_Acc,#11:Y_Acc,#12:Y_Acc
18           out $10:X_Awrms,$12:Y_Awrms,$13:Z_Awrms,$14:Awrms*/
19         def inherit_ch $1,#10,#11,#12,$10,$11,$12,$13
20         $10 = RRV($1,WBD(#10))
21         $11 = RRV($1,WBD(#11))
22         $12 = RRV($1,WBK(#12))
23         $13 = SQR((1.4*$10)^2+(1.4*$11)^2+$12^2)
24       }wbv_calc
```

<Archi_1 Script 記述構文の説明>

実行に先立って解析対象ファイルが読み込まれ収録チャネル1(#1)は座位 X 軸加速度、チャネル 2(#2)は座位 Y 軸加速度、チャネル 3(#3)は座位 Z 軸加速度データである事を前提としています。

7 行目： 演算処理ブロック呼出文で end 行以降に記述された演算処理ブロック wbv_calc を呼び出し実行します。
引数並びは即値で記述した時定数、加速度収録チャネル#1～#3、戻り値として各軸の補正加速度実効値と合計補正加速度実効値$1～$4 を記述します。

8 行目： グラフの X 軸スケールを描画するための経過時間を生成します。

10 行目～13 行目：グラフ関連構文で演算結果の補正加速度実効値をグラフ表示します。 グラフ描画格納関連構文に関しては後述する「11. 演算結果をグラフ表示する」項を参照して下さい。

10 行目： グラフ番号定義文でグラフ番号、グラフ表題を定義します。

11 行目: グラフ X 軸定義文で格納するグラフの X 軸スケールを定義します。

12 行目: グラフ Y 軸定義文で格納するグラフの Y 軸スケールを定義します。

13 行目： グラフ格納文でグラフを作成して格納します。

16 行目～25 行目：外部演算処理ブロックで各軸補正加速度実効値、合計補正加速度実効値を演算します･

19 行目： 引き継ぎチャネル定義文で、演算ブロック呼び出し文で記述されている引数並びを定義します。引数先頭に記述されている即値(定数)は数値属性参照チャネルで引き継ぎます。また、収録チャネルは収録チャネルで引き継ぎ、数値属性参照チャネルは数値属性参照チャネルで引き継ぎます。チャネル番号は問いません。

20 行目： X 軸補正加速度実効値を演算します。

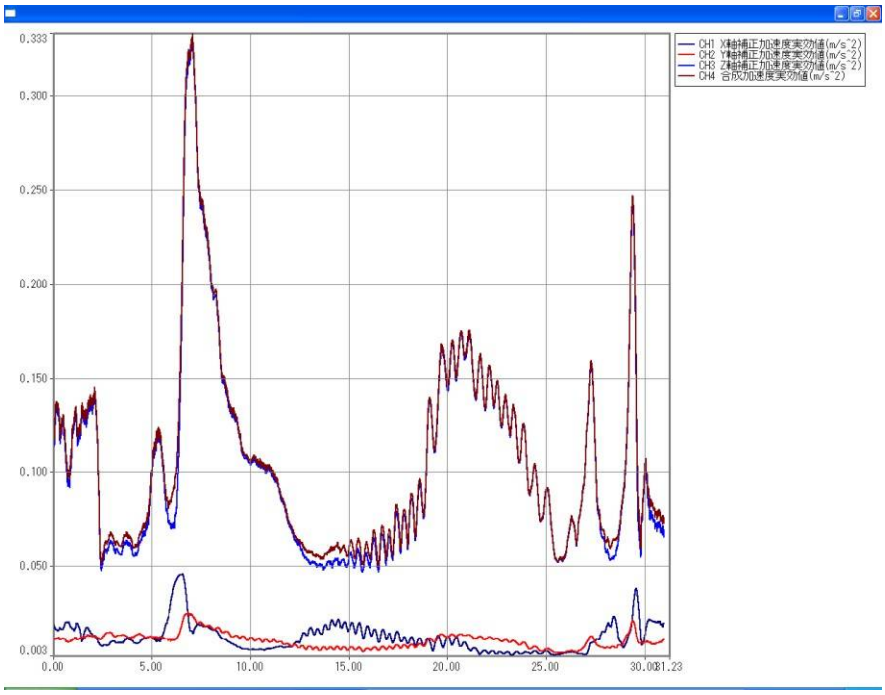
21 行目： Y 軸補正加速度実効値を演算します。

22 行目： Z 軸補正加速度実効値を演算します。

23 行目： 各軸の補正加速度実効値から合成補正加速度実効値を演算します。

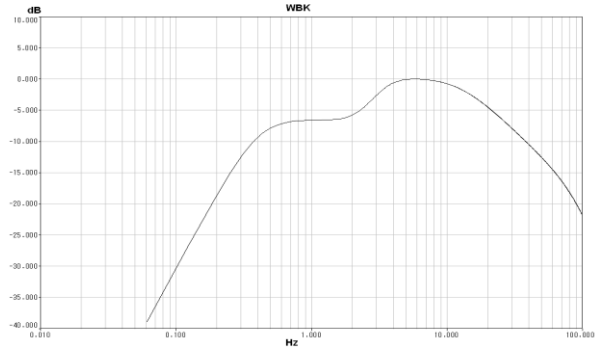
※ 補正加速度演算内容は後述する＜参考＞を参照下さい。

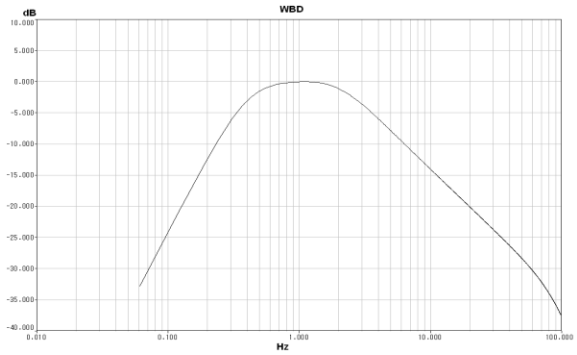
〈実行結果表示される補正加速度実効値グラフ〉



〈参考〉

全身振動暴露演算での加速度実効値演算は、実効値演算前に各軸の収録加速度データに ISO26341-1 で規定される補正フィルタ処理を行います。補正フィルタの周波数特性図を下記に示します。座位･立位の X 軸と Y 軸は同じ補正フィルタで d カーブ、Z 軸は k カーブでフィルタ処理を行います。下図左側が k カーブ、右側が d カーブです。･





Archi_1 ではそれぞれ演算関数が用意されています。D カーブが WBD(演算対象数列)、K カーブが WBK(演算対象数列)となります。実効値変換は移動実効値関数 RRV(時定数,演算対象数列)を使用して行います。時定数は通常 1 秒としますが、記述例では 0.25sec(8Hz)として演算しています。一度に補正加速度実効値を求める演算式は

X 軸/Y 軸補正加速度実効値 = RRV(時定数,WBD(X 軸あるいは Y 軸加速度)) ⇒X_Awrms、Y_Awrms
Z 軸補正加速度実効値 = RRV(時定数,WBK(Z 軸加速度)) ⇒Z_Awrms

となります。合成補正加速度実効値の演算式は

$$合成補正加速度実効値 = \sqrt{\{(1.4 \times X\_Awrms)^2 + (1.4 \times Y\_Awems)^2 + Z\_Awrms^2\}}$$

となります。

詳細は 20 章「演算関数を使用して振動曝露解析を行う」を参照下さい。

4. 演算処理ブロックを制御する

演算処理ブロックの直前に記述し、直下の演算処理ブロックを実行するか否かを指定する記述構文を演算処理ブロック制御文と呼称します。演算処理ブロック制御文には演算処理ブロック制御文が成立した時 1 回だけ、その直下に記述された演算処理ブロックの実行を行うタイプと演算処理ブロック制御文が成立している間、その直下に記述された演算処理ブロックを繰り返して実行するタイプが存在します。

※ 但し、繰り返し演算処理を制御する場合、チャネル番号を変えての処理や、複数ファイルの処理など繰り返し処理回数の少ない場合に使用します。

基本的な記述構造を示します。

```
演算処理ブロック制御文 条件
  演算処理ブロック開始文 {
          |
          |
  }演算処理ブロック終結文
```

4. 1. 与えた条件が成立した時、直下の演算処理ブロックを実行

条件実行制御文

文法:

```
case 比較値 1 比較演算子 比較値 2
```

比較演算子⇒ =,<>,>,<,>=,<= 、条件が成立した時に直下の演算処理ブロックを実行します。

記述例:

$1 が 1 に等しいか大きい場合に演算処理ブロック"calc"を実行する

```
case $1 >= 1
  proc calc{
    |
  }calc
```

論理成立実行制御文

文法:

```
case_true 論理値
```

論理値"1"の時に直下の演算処理ブロックを実行します。

記述例:

$1 が 1 の時に演算処理ブロック"calc"を実行する

```
case_true $1
  proc calc{
    |
  }calc
```

論理非成立実行制御文

文法:

```
case_false 論理値
```

論理値"0"の時に直下の演算処理ブロックを実行します。

記述例:

$1 が 0 の時に演算処理ブロック"calc"を実行する

```
case_false $1
  proc calc{
    |
  }calc
```

4. 2. 与えた条件が成立している間、直下の演算処理ブロックを繰り返し実行

繰り返し条件実行制御文

文法:

```
repeat_cace 比較値 1 比較演算子 比較理 2
```

比較演算子⇒ =,<>,>,<,>=,<= 条件が成立している間直下の演算処理ブロックを繰り返し実行します。

記述例:

$1 が$2 より小さい間演算処理ブロック"calc"を繰り返し実行する

```
repeat_case $1 < $2
  proc calc{
    |
    $1 = $1+1    /* この記述を忘れると無限 LOOP しますので注意*/
  }calc
```

記述例:

繰り返し演算制御ブロック制御文を使用して$1 の要素合計を求める場合

```
repeat_case $1 < $2
$2 = 0
$3 = LEN($1)
$4 = 0
repeat_case $2 < $3
  proc totoal{
    $4 = $4+$1($2)
    $2 = $2+1
  }total
```

※ 上記例は、repeat_case 文を使用した繰り返し処理例として記述したもので、$1 の合計値を求める例としては適切ではありません。演算対象数列の Index を直接指定する繰り返し処理は処理時間が掛り、出来るだけ演算対象数列をそのま塊として扱う必要があります。$1 の合計値を求める場合、演算関数の SUM 関数を使用して

```
$4 = SUM($1)
```

と記述し、出来るだけ Loop 処理を使用しないことが大切です。

4. 3. 演算解析範囲を index またはマーク番号で指定して、直下の演算処理ブロックを実行 当該演算処理ブロックの記述以前行、あるいはブロック内に収録チャネル(#n)の記述が存在していないか、後述する処理データ個数 定義文が記述されていない場合は実行されません。演算範囲を指定すると言う事は、直下に記述された演算処理ブロック内の複数 要素を持つチャネルは、仮想的に指定範囲が切り出され、設定された演算範囲の先頭データ番号(index)0 とし指定されたデータ個 数(要素数)に再構成されます。演算処理ブロックを終了すると元に戻ります。

4. 3. 1. Index 範囲を指定して演算処理ブロックを制御する データ番号実行制御文

文法:

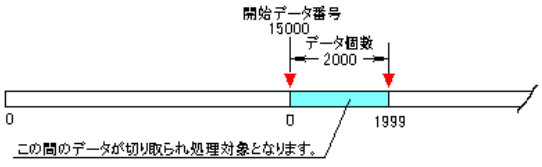
index 開始データ番号 データ個数

直下の演算処理処理ブロック内に記述されたチャネルの Index を開始データ番号からデータ個数分を切り出し実行します。また、記述する開始データ番号、データ個数の何れか一方でも複数要素を持つ数値属性チャネルで記述すると要素数分繰り返して実行します。要素数が等しい場合はペアで参照され、要素数が異なる場合は、何れか要素数の多い方で繰り返され、要素数の少ない方は複数要素であっても Index0 のみ参照されます。なお、切り出し範囲の index は 0 から振り替え再構成して実行します。

記述例:

開始データ番号:15000、データ個数:2000 で実行する

```
index 15000 2000
  proc add{
    |
  }add
```



4. 3. 2. Index 範囲を指定して直下の演算処理ブロックを繰り返して実行する 繰り返しデータ番号実行制御文

文法:

repeat_index 開始データ番号 データ個数 終了データ番号

直下の演算処理処理ブロック内に記述されたチャネルの Index を開始データ番号からデータ個数分を切り出し実行し、繰り返しによって開始データ番号はデータ個数分加算され、更新された開始データ番号が終了データ番号に等しいか大きくなるまで繰り返して実行します。従って、開始データ番号は更新されるため、必ず参照チャネルで記述する必要があります。なお、切り出し範囲の index は 0 から振り替え再構成して実行します。 記述された開始データ番号、データ個数、終了番号を参照チャネルで記述し演算処理ブロック内で変更すると、繰り返し数 や切り出し範囲が変更されます。 開始データ番号、データ個数、終了データ番号に複数要素を持つ参照チャネルで記述しても参照される Index は 0 のみとな ります。

記述例:

開始データ番号初期値:0、データ個数:500、終了データ番号:9999 で繰り返す

```
$1 = 0
repeat_index $1 500 9999
  proc sep{
    |
  }sep
```

演算処理ブロック内に記述されたチャネルのデータ個数が 999 以上存在しないと繰り返しません。

4. 3. 3. マーク番号を指定して演算処理ブロックを制御する

マーク番号とは、収録データファイルに付けられているマークを意味します。Script 記述でマーク番号を付けることはできませ ん。マーク番号の編集操作は PcWaveForm 取扱説明書 基礎編 第 4 章「虫眼鏡 Window」の 4.2.2. MARK の編集を参 照下さい。

Mark 番号実行制御文 文法:

mark 開始マーク番号 終了マーク番号

直下の演算処理処理ブロック内に記述されたチャネルの Index を開始マーク番号位置の Index から終了マーク番号位置の Index-1 までを切り出し、Index を 0 から振り替えて再構成して実行します。なお、設定されている解析対象範囲に記述されたマーク番号が存在しない場合、あるいは開始マーク番号>=終了マーク番号の場合は直下の演算処理ブロックは実行されません。

また、記述する開始マーク番号、収録マーク番号の何れか一方でも複数要素を持つ数値属性チャネルで記述すると要素数分繰り返して実行します。要素数が等しい場合はペアで参照され、要素数が異なる場合は、何れか要素数の多い方で繰り返され、要素数の少ない方は複数要素であっても Index0 のみ参照されます。

記述例:

開始マーク番号:1、終了マーク番号:4 の範囲を実行する

```
mark 1 4
  proc calc{
    |
  }calc
```

4. 3. 4. マーク番号を指定して直下の演算処理ブロックを繰り返して実行する

繰り返し MARK 番号実行制御文

文法:

repeat_mark 開始マーク番号 マーク飛び越し数 終了マーク番号

直下の演算処理処理ブロック内に記述されたチャネルの Index を開始マーク番号位置から次のマーク番号位置 index-1 までデータ個数分を切り出し実行し、開始マーク番号は飛び越しマーク数+1 され、更新された開始マーク番号が終了マーク番号に等しくなるまで繰り返して実行します。従って、開始マーク番号は更新されるため、必ず参照チャネルで記述する必要があります。なお、切り出し範囲の index は 0 から振り替え再構成して実行します。 記述された開始マーク番号、終了マーク番号、飛び越しマーク数を参照チャネルで記述し演算処理ブロック内で変更すると、 繰り返し数や切り出し範囲が変更されます。開始マーク番号、終了マーク番号、マーク飛び越し数に複数要素を持つ参照チャネルで記述しても参照される index は 0 のみとなります。

記述例:

開始データ番号初期値:2、マーク飛び越し数:1、終了マーク番号:5 で繰り返す

```
repeat_mark 2 1 5
  proc calc{
    |
  }calc
```

4. 3. 4. 処理データ個数を定義する

収録チャネルが参照されていない場合に、Index またはマーク番号を指定した演算処理ブロック制御文を有効にします。

処理データ個数定義文

文法:

```
def num_samps データ個数
```

記述例:

```
def num_samps 50000
```

定義するデータ個数は演算処理ブロック制御文で扱うデータ個数より等しいか大きなければなりません。

4. 4. Index 振り替え禁止宣言

演算範囲を index またはマーク番号で指定して、直下の演算処理ブロックを実行する場合、演算処理ブロック内に記述された複数の要素を持つ全てのチャネルの index は、制御文で設定した範囲に自動的に置き換わります。切り出し再構成を禁止するチャネルは Index 制御禁止宣言が必要です。

Index 制御禁止チャネル宣言文

文法:

```
dcl exempt_ch  チャネル列
```

本文で宣言されたチャネルは、Mark や Index で制御する演算処理ブロック内でも再構成されません。なお、宣言したチャネルは Script 記述中有効となり、解除することはできません。

記述例:

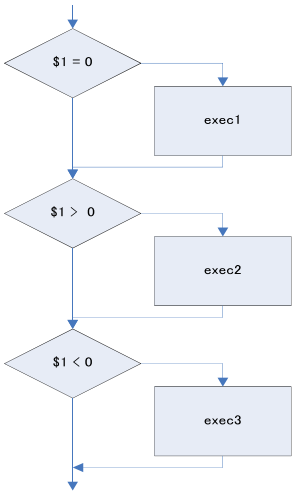
```
dcl exempt_ch $2
assign $2 = 5<2>
index 100 20
   proc calc{
     $1 = #1*SUM($2)
   }calc
```

上記記述例で、$2 を Index 制御禁止チャネル宣言文(dcl exempt_ch)で指定しない場合、演算処理ブロック内で Index 再構成を受けるチャネルは$2 と#1 となりますが、#1 の要素数は 119 以上存在するとしても、$2 の要素数は 5 個しか存在しないため、実行時エラーが発生します。

記述例4. 1. 演算処理ブロックを条件実行制御文で制御する

<Archi_1 Script 記述例>

```
1          /*---------calc8.prc-----------*/
2          $1 = MAX(#1)
3          case $1 = 0
4            proc exec1{
5              $2 = MEA(#2)
6            }exec1
7          case $1 > 0
8            proc exec2{
9              $2 = MAX(#2)
10           }exec2
11         case $1 < 0
12           proc exec3{
13             $2 = MIN(#2)
14           }exec3
```



<Archi_1 Script 記述構文の説明>

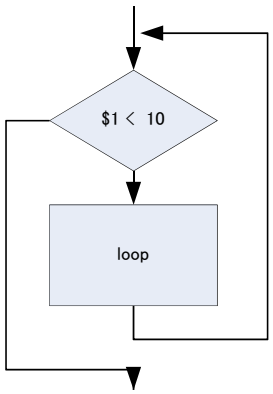
条件実行制御文で記述された条件が成立すると直下の演算処理ブロックを実行します。
3 行目:　条件実行制御文で$1 = 0 が成立すると演算処理ブロック:exec1 を実行ます。
7 行目:　条件実行制御文で$1 > 0 が成立すると演算処理ブロック:exec2 を実行します。
11 行目:　条件実行制御文で$1 < 0 が成立すると演算処理ブロック:exec3 を実行します。

演算処理ブロック制御文は演算処理ブロック内にも記述でき、多様な演算処理構造を構築できます。 但し、演算処理ブロック制御文に分岐の概念はありません。したがって、分岐処理を行う場合は背反する条件で、それぞれの演算処理 ブロックを制御します。その場合、例えば、上記例で示せば、$1 = 0 で実行される演算処理ブロック exec1 の中で$1 に例えば1を代入 すると。次の条件実行制御文$1 > 0 が成立し演算処理ブロック exec2 も実行される事を意味します。

記述例4. 2. 演算処理ブロックを繰り返し条件実行文で制御する

<Archi_1 Script 記述例>

```
1          /*---------calc9.prc-----------*/
2          $1 = 0
3          $2 = 0
4          repeat_case $1 < 10
5            proc loop{
6              $2 = $2+$1
7              $1 = $1+1     /* counter up */
8            }loop
```



<Archi_1 Script 記述構文の説明>

繰り返し条件実行制御文で記述された条件が成立している間、直下の演算処理ブロックを実行します。
4 行目:　繰り返し条件実行制御文で条件が成立している間、演算処理ブロック:loop を繰り返し実行します。
　　演算処理ブロック loop の中で条件を不成立となる様に制御しないと、無限ループとなります。
7 行目:　$1 をインクリメントして$1 が 10 となると演算処理ブロック:loop の繰り返し実行を終了します。

記述例4. 3. 演算処理ブロックをデータ番号実行制御文で制御する

収録チャネル ch2 の先頭から 100,000 点を 1000 点ごとに区切った実効値と最大値数列を生成します。

<Archi_1 Script 記述例>

```
1          /*--------- proc_ctl2.prc----------------------*/
2          dcl exempt_ch $2,$3
3          $1 = (ACC(DAG(100,1))-1)*1000   /* 開始 Index の生成*/
4          def ch_name $2 "RMS"
5          def ch_name $3 "MAX"
6          index $1 1000
```

```
7           proc calc{
8              $2 = LNK($2,EFF(#2))
9              $3 = LNK($3,MAX(#2))
10          }calc
11          /*----グラフ描画処理----------*/
12          def graph_id @1 "effective_value"
13          def graph_x_axis @1 0,0,5000 F0
14          def graph_y_axis @1 0,0 F3
15          def file_id %1 "effective_value" grp
16          save plot %1 @1 $1 $2,$3
17          end
```

<Archi_1 Script 記述構文の説明>

2 行目： Index 制御禁止チャネル宣言文にて演算処理ブロック内で使用する$2 と$3 の Index 再構成を禁止します。

3 行目： データ番号実行制御文のデータ開始 Index を生成します。生成結果は 0,1000,2000,3000･･･99000 となります。

6 行目： データ番号実行制御文で開始データ番号:2 行目で生成した$1、データ個数:1000とします。

7 行目～10 行目:データ番号実行制御文で制御される演算処理ブロックです。

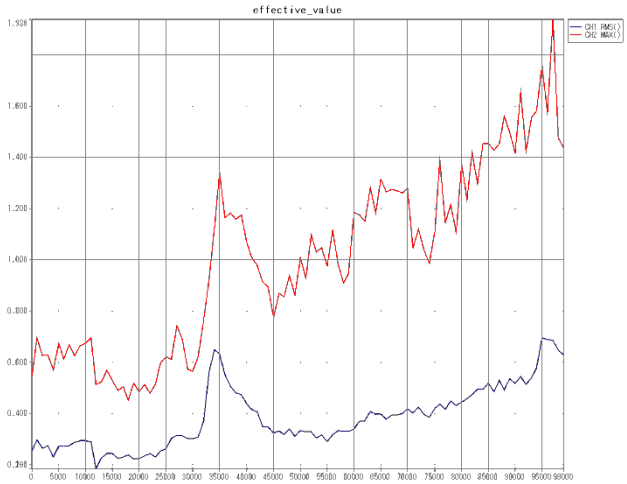
8 行目： 収録チャネル#2 の実効値を演算し$2 に連結します。

9 行目： 収録チャネル#2 の最大値を演算し$3 に連結します。

8 行目と 9 行目の演算式は、演算範囲が 0～999、1000～1999 と変更され繰り返して演算されそれぞれ$2 と$3 に連結されますので演算処理ブロック終了後は$2(0)に Index0～999 迄の実効値、$2(1)に index1000～1999 迄の実効値が格納されます。$2 と$3 を Index 制御禁止チャネル宣言した理由は、演算処理ブロック内で記述した$2,$3 は初回処理では Null、2 回目処理で要素数1、2 回目処理では要素数 2 となり複数要素を持つため Index 制御を受け、Index 制御禁止チャネル宣言を行わないと、この時点で実行時エラーが発生します。

12 行目～16 行目:演算結果の実効値と最大値のグラフを作成してグラフを格納する処理となります。

グラフ描画格納関連構文に関しては後述する第 11 章「演算結果をグラフ表示する」項を参照して下さい。 演算結果を示します。



区間最大値は赤線で表示、区間実効値は青色で表示しています。

※ 記述例4. 3. では、index 制御文の説明の都合上、実効値関数(EFF 関数)及び最大値関数(MAX 関数)のフラグ指定により区間ごとの実効値或いは最大値を求めず、区間ごとに index 制御文で切り取られた範囲をそのまま求めていますが、同じ処理を、index 文を使用しないで記述することもできます。その場合、演算処理ブロックを記述せず、直接、$2 及び$3 を以下の記述にします。

```
$2 = EFF(1000,#2)          /* index0 から 1000 個ごとに区切った実効値数列を求める*/
$3 = MAX(1000,#2)          /* index0 から 1000 個ごとに区切った最大値数列を求める*/
```

記述例4. 4. 演算処理ブロックを繰り返しデータ番号実行制御文で制御する

20000 個のデータを 500 個ごとに 0 から 500 ステップで増加する階段状波形を生成します。

<Archi_1 Script 記述例>

```
1           /*--------- proc_ctl1.prc----------*/
2           def num_samps 20000
3           assign $1 = 20000<0>
```

```
4           $2 = 0
5           repeat_index $2 500 20000
6             proc calc{
7                $1 = $1+$2
8             }calc
9           /*-----グラフ描画処理----------*/
10          def graph_id @1
11          def graph_x_axis @1 0,0,1000 F0
12          def graph_y_axis @1 0,20000 F0 5
13          def file_id %1 "step_wave" grp
14          save plot %1 @1 $1
15          end
```

<Archi_1 Script 記述構文の説明>

2 行目: 処理データ個数定義文で。収録チャネルを参照しない場合に Index 指定して制御する演算処理ブロック制御文を使用するために解析対象ファイルの Index 範囲を知らせるためです。定義するデータ個数はここで記述する Index 制御範囲以上なければなりません。

3 行目: 階段状波形格納先を数列を生成します。データ個数:20000、値は全て 0 と初期化します。

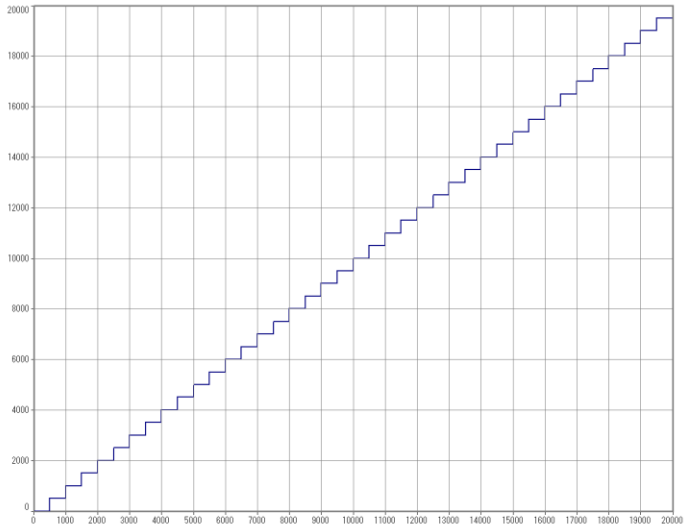
4 行目: 開始データ番号(Index)の初期値を 0 とします。

5 行目: 繰り返しデータ番号実行制御文で開始データ番号:$2、データ個数:500、終了データ番号:20000 とし ます。直下の演算処理ブロックの中に記述したチャネルの Index を、0～499、500～999、1000～ 1499,・・・・14500～19999 まで繰り返して実行します。最終的に開始データ番号 19500 に 500 を加算すると開始 Index が 20000 となり、チャネルの最終 Index は 19999 までのため、繰り返しを終了します。

7 行目: 3 行目で初期化した数列に開始データ番号を加算します。開始データ番号は繰り返しにともなって、0、500,1000、1500、・・・19500 と更新されます。したがって$1 の初期値は全て 0 ですので、最初は Index0～499 に 0 が格納され、次は Idex500～999 に 500、次は Index1000～1499 に 1000 と最後は Index19500～19999 に 19500 が格納されます。

10 行目～14 行目:演算結果の$1 をグラフに作成してグラフを格納する処理となります。 グラフ描画格納関連構文に関しては後述する第 11 章「演算結果をグラフ表示する」項を参照して下さい。

演算結果を示します。



<補足説明>

5 行目で制御される Index を別の表現をすると、演算処理ブロック calc は

1 回目:$1 の 0～,500 に ← $1[0,500]+0

2 回目:$1 の 500～,500 に ← $1[500,500]+500

3 回目:$1 の 1000～,500 に ← $1[1000,500]+1000

4 回目:$1 の 1500～,500 に ← $1[1500,500]+1500

　　|

　　|

最終回目:$1 の 19500～500 に ← $1[19500,500]+19500

$1 の初期値は要素数 20000 個で、すべて 0 で数列生成されているため、順次開始 Index 値が 0 に加算された数列が格納される事になり、演算制御処理ブロックの繰り返し実行が終了すると、$1 の要素数は元の 20000 個から変化している訳ではないため、階段状の数値が格納された事になります。

記述例4. 5. 演算処理ブロックをマーク番号実行制御文で制御する

解析対象ファイルにつけられた任意マーク番号区間の振幅値を演算表示します。

<Archi_1 Script 記述例>

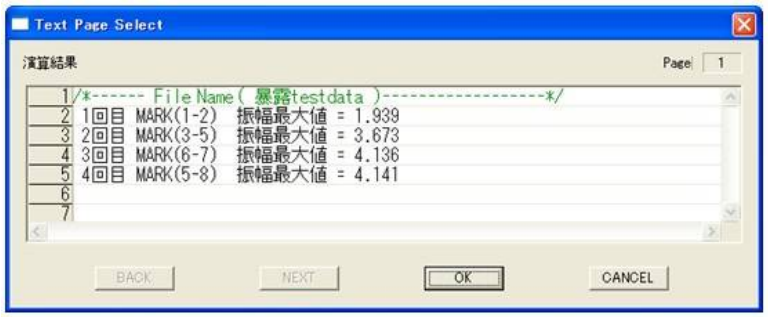
```
1       /* ---- proc_ctl5.prc----------------*/
2       dcl menu_label "マーク区間振幅値"
3       $1 "開始マーク番号:" = 1
4       $2 "終了マーク番号:" = 2
5       $4 "write_counter:" = 1
6       assign &1 = 100<" ">
7       &1 = CFNM()
8       assign &1 = "/*----- File Name( "|&1|" )----------------*/"
9       $3 = NMK()
10      repeat_case $3 >= 2
11        proc exec{
12          $5 "mark_No.:" = ACC(DAG($3,1))
13          $6 "制御フラグ:" = -1   /* $6 >= 0 正常*/
14          repeat_case $6 < 0
15            proc start_until_mark{
16              get value $5:$1(F0),$5:$2(F0) "演算区間マーク番号入力"
17              $6 = $2-$1
18              case $6 < 0
19                proc invalid_input{
20                  disp message "入力されたマーク番号が不正です"
21                }invalid_input
22            }start_until_mark
23          case $6 > 0
24            proc calc1{
25              mark $1 $2
26                proc calc2{
27                  $7 "最大振幅:" = ABS(MAX(#1)-MIN(#1))
28                  assign &1($4) = " "|$4(F0)|"回目 MARK("|$1(F0)|"-"|$2(F0)|")  振幅最大値 = "|$7(F3)
29                  $4 = $4+1
30                }calc2
31              get text_page &1 $8 "演算結果"
32            }calc1
33          case $6 = 0
34            proc cont_message{
35              get reply "終了しますか" "はい" "いいえ" $8
36              $3 = RVS($8,$3,0)
37            }cont_message
38        }exec
39      end
```

<Archi_1 Script 記述構文の説明>

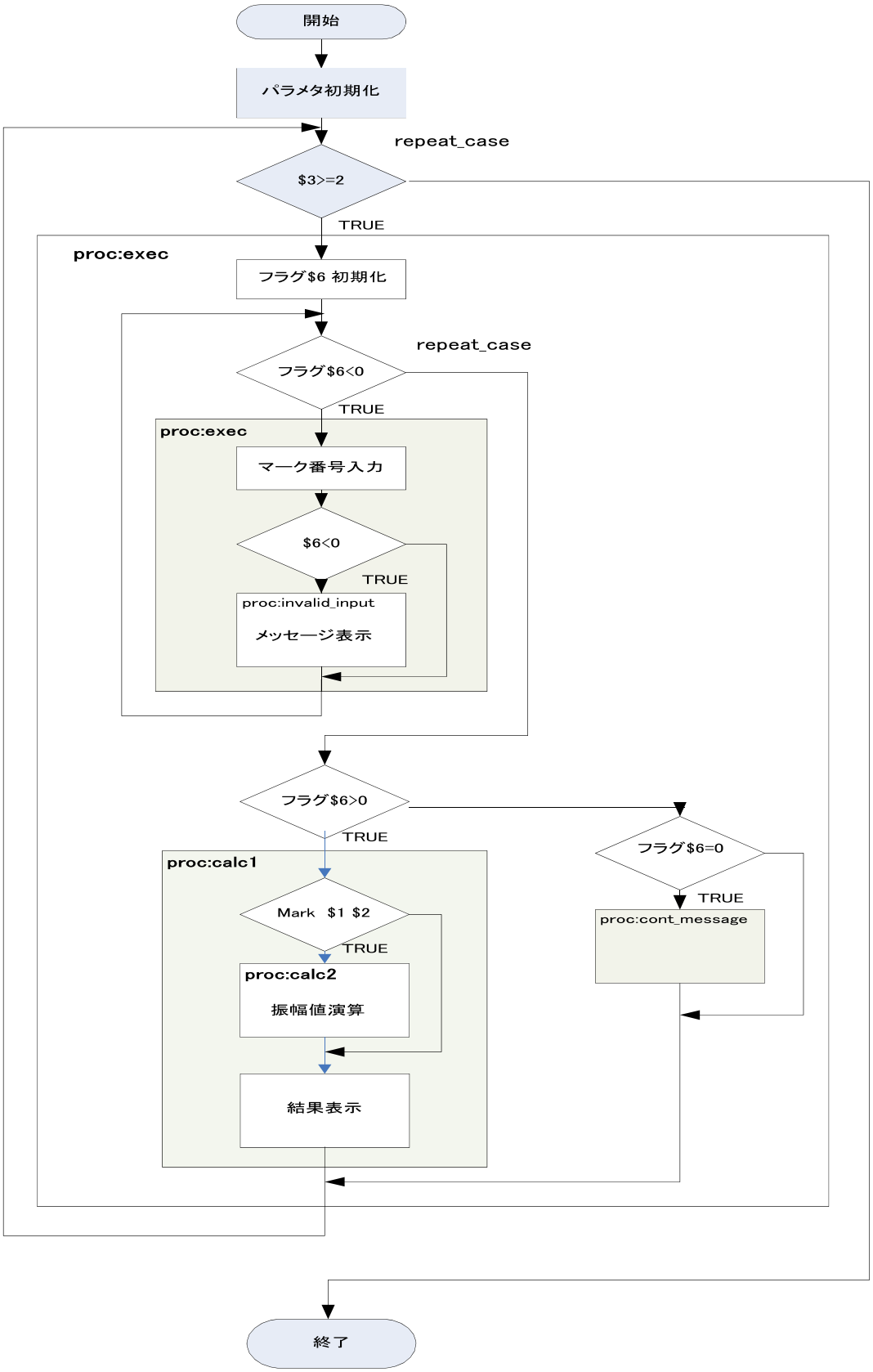
2 行目:　Script 実行メニューボタンラベルを宣言します。Script の実行に直接関係ありません。
2 行目～9 行目:パラメータの初期値、結果表示用のヘッダー行を生成します。
3 行目:　演算区間開始マーク番号の初期値で1とします。
4 行目:　演算区間終了マーク番号の初期値で 2とします。
5 行目:　結果表示用テキスト書き込み行番号の初期値で 1とします。
6 行目:　結果表示用の文字列配列で 100 行(演算 100 回分)の配列を用意します。
7 行目:　解析表示用のヘッダー行に表示する解析対象ファイル名を取得します。
8 行目:　解析結果表示用のヘッダー行を生成します。
9 行目:　解析対象ファイルに付けられているマーク個数を取得します。
10 行目:　繰り返し条件実行制御文で、マーク個数が>=2 の時、直下の演算処理ブロック"exec"を繰り返し実行します。
11 行目～38 行目が演算処理ブロック"exec"となります。
12 行目:　キーボード入力時のリストボックス表示用に取得したマーク個数からマーク番号数列を生成します。
13 行目:　演算処理ブロック"exec"に含まれる各演算処理ブロック実行制御用フラグで初期値:-1 とします。
14 行目: 繰り返し条件実行制御文でフラグ:$6 が<0 の時、直下の演算処理ブロック"start_until_mark"を繰り返し実行します。
15 行目～22 行目が演算処理ブロック"start_until_mark"となります。
16 行目: 値入力文リストボックスから選択する形式で開始マーク番号と終了マーク番号をキーボードから取得します。 記述方法は後述する「6. 解析に必要なパラメータを受け取る」項で説明します。

17 行目： 入力された開始マーク/終了番号が正しいかを演算しフラグ:$6 に格納します。
18 行目: 条件実行制御文でフラグ:$6 が<0 の時(不正入力時)、直下の演算処理ブロック"invalid_input"を実行します。
19 行目～21 行目が演算処理ブロック"invalid_input"となります。
20 行目： 入力不正メッセージを表示します。
23 行目： 条件実行制御文でフラグ:$6 が>0 の時、直下の演算処理ブロック"calc1"を実行します。
24 行目～32 行目が演算処理ブロック"calc1"となります。
25 行目: マーク番号実行制御文で入力された開始マーク番号と終了マーク番号から index を修飾して直下の演算処理ブロッ ク" calc2"を実行します。
26 行目～30 行目が演算処理ブロック"calc2"となります。
27 行目： 設定されたマーク番号間の最大振幅値を演算します。
28 行目： 結果表示用テキストに結果行を生成します。
31 行目： &1 に格納されているテキストをダイアログに表示します。
33 行目： 条件実行制御文でフラグ:$6 が=0 の時、直下の演算処理ブロック"cont_message"を実行します。
34 行目～37 行目が演算処理ブロック"cont_message"となります。

<実行結果が表示されるダイアログ>



記述例4. 5.の処理流れをフローチャートで示します。



記述例4. 6. 演算処理ブロックを繰り返しマーク番号実行制御文で制御する

解析対象ファイルに付けられたマーク番号区間ごとに最大値/最小値/平均値を演算します。

<Archi_1 Script 記述例>

```
1          /*----- proc_ctl4 ------------------*/
2          dcl sheet 1 {
3            page 1:
4              column &1,$[3,9]
5              format A,9(F3) 2
6          }sheet
7          def ch_name $3 "X 軸最大値:G"
8          def ch_name $4 "X 軸平均値:G"
9          def ch_name $5 "X 軸最小値:G"
10         def ch_name $6 "Y 軸最大値:G"
11         def ch_name $7 "Y 軸平均値:G"
12         def ch_name $8 "Y 軸最小値:G"
13         def ch_name $9 "Z 軸最大値:G"
14         def ch_name $10 "Z 軸平均値:G"
15         def ch_name $11 "Z 軸最小値:G"
16         def ch_name &1 "マーク番号"
17         $1 = NMK()          /* num of mark */
18         case $1 > 1
19           proc calc{
20             $2 = 1          /* top mark No. */
21             repeat_mark $2 0 $1
22               proc mark_proc{
23                 $12 = $2+1
24                 assign &1 = $2(F0)|" - "|$12(F0)
25                 $3 = MAX(#1)
26                 $4 = MEA(#1)
27                 $5 = MIN(#1)
28                 $6 = MAX(#2)
29                 $7 = MEA(#2)
30                 $8 = MIN(#2)
31                 $9 = MAX(#3)
32                 $10 = MEA(#3)
33                 $11 = MIN(#3)
34                 write ch_column 1: &1,$[3,9]
35               }mark_proc
36           }calc
37         end
```

<Archi_1 記述構文の説明>

2 行目～6 行目：結果を表示する結果シートを宣言します。結果シートの記述方法は第 10 章「結果結果シートを使用する」項で説明します。
17 行目： 解析対象ファイルに付けられているマーク個数を取得します。
18 行目： 条件実行制御文でマーク個数が 1 以上の場合に直下の演算処理ブロック"calc"を実行します。
20 行目： 開始マーク番号を初期値1とます。(マーク番号は1から振られます)
21 行目：繰り返しマーク番号実行制御文で開始マーク番号：$2、マーク飛び越し数：0、終了マーク番号：$1 として直下の演算処理ブロック"mark_porc"を繰り返し実行します。
23 行目： 表示する演算範囲終端マーク番号を演算します。
24 行目： 表示用にマーク番号処理範囲文字列を生成します。
25 行目： 収録チャネル ch1 の最大値を演算します。
26 行目： 収録チャネル ch1 の平均値を演算します。
27 行目： 収録チャネル ch1 の最小値を演算します。
28 行目～30 行目：収録チャネル ch2 の最大値、平均値、最小値を演算します。
31 行目～33 行目：収録チャネル ch2 の最大値、平均値、最小値を演算します。
34 行目： 演算結果を結果シートに書き込み表示します。

<Script 実行結果で表示される結果シート>

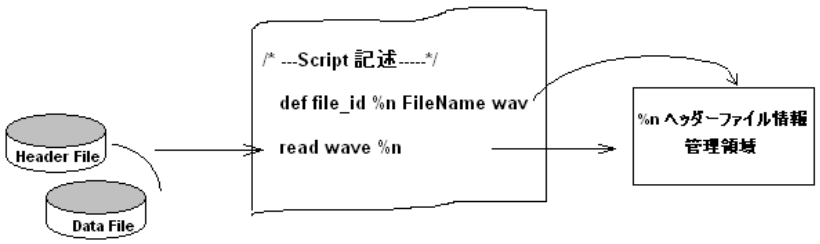
マーク番号	X軸最大値(G)	X軸平均値(G)	X軸最小値(G)	Y軸最大値(G)	Y軸平均値(G)	Y軸最小値(G)	Z軸最大値(G)	Z軸平均値(G)	Z軸最小値(G)
1 - 2	0.890	0.001	-1.049	0.696	0.001	-0.601	1.312	-0.002	-1.299
2 - 3	0.662	-0.001	-0.788	0.598	0.001	-0.700	0.874	-0.001	-1.113
3 - 4	1.776	0.001	-1.896	1.338	-0.002	-1.380	2.905	-0.002	-2.568
4 - 5	1.094	-0.001	-1.100	1.184	0.000	-0.970	2.616	0.001	-2.420
5 - 6	1.840	-0.001	-1.925	1.559	-0.000	-1.400	4.330	0.001	-3.629
6 - 7	1.835	0.000	-2.301	1.938	-0.001	-2.278	4.197	-0.000	-4.744
7 - 8	1.735	-0.001	-2.028	2.856	-0.001	-3.106	4.385	0.001	-5.043
8 - 9	3.045	-0.001	-2.859	3.239	-0.001	-2.490	5.595	0.001	-4.831
9 - 10	1.729	-0.001	-2.089	1.450	0.000	-1.323	3.763	0.001	-3.200

5. 解析対象ファイルを読み出す

PcWaveForm で解析対象ファイルを読み込み、波形表示 Window で表示し解析範囲を設定してから実行する場合は、改めてファイルを読み出す必要はありません。解析対象ファイル PcWaveForm で読み出していない場合は、Archi_1 Script 記述内で読み出す必要があります。

※ 解析対象ファイルとは,テキスト形式ヘッダーファイル(拡張子.hdr)とデータファイル(拡張子.dat)のペア(PcWaveForm 標準ファイルフォーマット)で構成された収録ファイルを指します。

解析対象ファイルを読み出す操作を模式的に示します。



5. 1. 解析対象ファイル読み出し事前処理

ファイルの読み出し或いは格納する為には、対象ファイル名をファイル番号として定義する必要があります。

ファイル番号定義文

文法:

def file_id %ファイル番号 ファイル名 属性

解析対象ファイルの場合、属性は wav となります。また、ファイル名に拡張子は付けません。 なお、ファイル番号定義文で記述する属性は取り扱うファイルにより異なります。詳細は別冊 Script 言語仕様書を参照下さ い。

5. 2. 解析対象ファイルの読み出し操作 ファイル名が既知でカレントのフォルダに格納されている場合、ファイル番号定義文と収録ファイル読み出し文を記述することで 読み出せます。

収録ファイル読み出し文

文法:

read wave %ファイル番号 読み出し開始 index,読み出し個数

読み出し開始 index と読み出し個数は、記述省略可能で、省略した場合は、ファイルの全範囲が読み出し対象となります。

記述例:

ファイル名"試験結果"の収録ファイル全データを読み出す場合

```
def file_id %1 “試験結果” wav
read wave %1
```

属性 wav で定義されたファイル管理領域は他の属性で読み出された管理領域とは異なり、Arch1_1 Script 内部では唯一無二のファイル管理領域となります。したがって、同時に複数の解析対象ファイルを読み出すことはできません。ファイル管理領域は、直前に読み出された解析対象ファイルの内容に更新されます。このことはチャネル情報や収録情報を始めとして収録チャネル割り当て文(get replace_ch 文)で使用される収録チャネル対応表も初期化されます。したがって、解析対象ファイルの連結操作など複数の解析対象ファイルを同時に扱いたい場合は、先に読み出した読み出した内容から必要な情報はデータを含め参照チャネルに代入保存してから、次の解析対象ファイルを読み出します。

5. 3. 解析対象ファイル名の選択操作

読み出す解析対象ファイル名を Script 実行時に選択して読み出したい場合は、ファイル番号定義文の前にファイル選択文を記述します。

ファイル選択文

文法:

get file_name ファイル名格納先チャネル ファイル個数格納先チャネル 拡張子

記述例:

解析対象ファイルを選択する場合、拡張子はヘッダーファイルの拡張子の"hdr"を記述します

```
get file_name &1 $1 “hdr”
def file_id %1 &1(0) wav
read wave %1
```

実行されるとファイル読み出しダイアログを表示し、選択されたファイル名が&1 に、ファイル個数が$1 に戻ります。ダイアログ

で複数ファイルを選択した場合はファイル名が複数要素として戻り、その要素数(選択ファイル数)はファイル個数に戻ります。なお、拡張子指定記述を省略すると戻りファイル名は拡張子が付加されて戻りますので、ファイル名選択後に選択されたファイルが読み出し可能か否かチェックが必要な場合があります。

5. 4. フォルダの選択操作 読み出す解析対象ファイルの格納先フォルダを選択して(カレントフォルダを切り替えて)、フォルダ内の全てのファイルを一括処理する 場合はフォルダ選択文とフォルダ内ファイル情報取得文を組み合わせて解析対象ファイル名を取得する操作を行います。

5. 4. 1. フォルダの選択

カ**レントフォルダ選択文**

文法:

get folder_select

実行するとフォルダ選択ダイアログを表示し、カレントフォルダを選択できます。フォルダ選択するとカレントフォルダが切り替わります。本文に項目(引数)はありません。

5. 4. 2. フォルダパス名の取得.

カレントフォルダ選択文を実行するとカレントフォルダパスが変更される場合がありますので、カレントフォルダ選択文実行前に 現在のカレントフォルダパスを保存する必要が在る場合に記述します。

カレントフォルダパス取得文

文法:

read folder_path **パス名格納先チャネル フラグ**

5. 4. 3. カレントフォルダの定義

カレントフォルダ選択文実行前にカレントフォルダ取得文で取得したフォルダパスに処理後戻す場合や、解析対象ファイルが 格納されているフォルダが予め判っている場合などに記述します。

フォルダパス定義文

文法:

def folder_pth **フォルダパス名**

記述例:

フォルダを指定してフォルダ内の解析対象ファイル名を取得する場合

```
def folder_path“c:¥試験データ”          /*解析対象ファイル格納先フォルダ定義*/
read file_name &1 $1 “hdr”             /* 解析対象ファイル名取得*/
def file_id %1 &1(0) wav               /* ファイル番号定義*/
read wave %1                           /* 解析対象ファイル読み出し*/
```

5. 4. 4. フォルダ内格納ファイル情報の読み出し

カレントフォルダ内に格納されているファイル名、ファイル個数などの情報を知る必要がある場合に記述します。 **フォルダ内ファイル情報取得文**

文法:

read fle_info **ファイル名 生成年月日 生成時刻 ファイル個数 拡張子**

記述例:

フォルダを選択し、フォルダ内に格納されている全ての波形ファイルを処理する場合

```
read folder_path &4                      /*カレントフォルダ選択前に現在のカレントフォルダパスを取得*/
get folder_select                        /*フォルダ選択*/
read file_info &1 &2 &3 $1 “hdr”         /*フォルダ内ファイル情報読み出し*/
$2 “file_counter:” = 0
repeat_case $2 < $1                      /*フォルダ内ファイル数分の繰り返し操作*/
  proc file_処理{
    def file_id %1 &1($2) wav            /*ファイル番号定義*/
    read wave %1                         /*解析対象ファイル読み出し*/
      |                                  /*ファイル処理 roputine記述部分*/
    $2 = $2+1                            /* file_counter_up*/
  ]file_処理
def folder_path &4                      /*カレントフォルダを元に戻す*/
```

カレントフォルダ内に格納されている指定した拡張子を持つファイルのファイル名が&1に格納され、当該フイルの生成年月日 が&2 に時刻が&3 に、存在したファイル数が $1 に戻ります。拡張子記述を省略するとフォルダ内のすべてのファイル名に拡 張子が付加されて戻ります。

5. 5. ファイルステータス確認 解析対象ファイルなどは排他的に読み出され処理されますので、ファイル名を指定して読み出す操作を行っても、既に他のプログラム が使用している(Open している)場合は実行時エラーが発生します。予め当該ファイルが読み出せるか確認する必要が在る場合などに 記述します。

ファイルステータス取得文

文法:

read file_check %ファイル番号 ステータス格納先チャネル

ファイル番号定義文で定義したファイルが読み出し可能か否かステータスを戻します。ファイルステータス取得文は同じ意味を持つ演算関数ファイルステータス取得関数:RFC(&m)を使用することができます。但し、関数の引数は拡張子を付けたファイル名となります。

記述例:

```
def file_id %1 "試験データ 20110914" wav
read file_check %1 $1
```

$1 に%1 で定義されている解析対象ファイル名"試験データ 20110914"のステータスが戻ります。

ステータス ⇒ 0 Open 可能
ステータス ⇒ 1 存在しているが現在他のプログラムが使用中で Openできない
スタータス ⇒ 2 存在していない

記述例:

5. 4. 4. で記述したフォルダ内一括処理に読み出し不可のファイルを読み飛ばす処理を追加する場合

```
read folder_path &4                          /*カレントフォルダ選択前に現在のカレントフォルダパスを取得*/
get folder_select                            /*フォルダ選択*/
read file_info &1 &2 &3 $1 "hdr"             /*フォルダ内ファイル情報読み出し*/
$2 "file_counter:" = 0
repeat_case $2 < $1                          /*フォルダ内ファイル数分の繰り返し操作*/
  proc file_処理{
    def file_id %1 &1($2) wav                /*ファイル番号定義*/
    read file_check $3
    case $3 = 0                              /*読み出し可能の場合*/
      proc 処理可能{
        read wave %1                         /*解析対象ファイル読み出し*/
          |                                  /*ファイル処理 roputine記述部分*/
    }処理可能
   case $3 <> 0                              /*読み出し不可の場合*/
    proc 処理不可{
      disp message &1($2)|" 存在していないか、他のプログラムが使用中"
    }処理不可
    $2 = $2+1                                /* file_counter_up*/
  ]file_処理
def folder_path &4                           /*カレントフォルダを元に戻す*/
```

記述例5. 1. ファイル選択ダイアログから解析対象ファイルを読み出す

<Archi_1 Script 記述例>

```
1        /* ---- read wave1.prc -------*/
2        def ch_name &1 "file_name"
3        def ch_name $1 "num_file"
4        get file_name &1 $1 "hdr"
5        def file_id %1 &1 wav
6        read wave %1
```

<Archi_1 Script 記述構文の説明>

4 行目： ファイル選択文で実行されると"ファイルを開く"ダイアログが表示されます。このダイアログで選択したファイル名が&1 に戻ります。

5 行目： ファイル番号定義文で選択されたファイル名を属性 wav(解析対象データファイル)として定義します。

6 行目： 波形ファイル読み出し文で読み出します。ファイル名を選択せずに直接読み出す場合は 5 行目の＆1の代わりにファイル 名を記述して読み出せます。なお、6 行目の解析対象ファイル読み出しでは、収録データの全範囲を Script 内に読み込みます。解析対象ファイルの全範囲を読み出すのでなく、読み出し開始 Index および読み出しデータ個数を指定して読み出す場合は、read wave 文の引数に読み出し開始 Index と読み出しデータ個数を半角カンマで区切って記述します。なお、"ファイルを開く"ダイアログでは、同時に複数のファイルを選択できますが、ここでは、選択された最初のファイルのみ読み出しています。また、"ファイルを開く"ダイアログでフォルダを変更することができ、変更された場合、カレントフォルダも変更されます。

記述例5. 2. ファイルが読み出し可能か確認して読み出す

<Archi_1 Script 記述例>

```
1        /*----- read_wave2.prc ---------------*/
2        def ch_name &1 "file_name"
3        def ch_name $1 "num_file"
4        def ch_name $2 "file_status"
5        $2 = 1                              /* ファイルステータスの初期化*/
6        repeat_case $2 <> 0                   /* ファイルステータスのチェック*/
7          proc wav_file_read{                /* open 可能なファイル選択処理ブロック*/
8            get file_name &1 $1 "hdr"        /* ファイル選択ダイアログの表示  */
9            def file_id %1 &1 wav            /* ファイル番号の定義  */
10           read file_check %1 $2            /* ファイルステータス Read */
11           case $2 <> 0                     /* ファイルステータスのチェック*/
12             proc cannt_open{                /* open 不可の時の処理ブロック*/
13               disp message &1|"他で使用中、別のファイルを選択して下さい"
14             }cannt_open
15         }wav_file_read
16       read wave %1                              /* 解析ファイルの読み出し*/
```

<Archi_1 Script 記述構文の説明>

10 行目： ファイルステータス取得文で設定されたファイルのステータスを取得します。ステータス 0 が OPEN 可能、0 でない場合はファイルが存在しないか、他で使用中を意味します。他で使用中は例えば、PcWaveForm 上で既に読み出している場合などが考えられます。上記例 OPEN可能で無い場合に読み出しファイル設定を繰り返します。

記述例5. 3. ファイル選択ダイアログで一度に選択して順次処理する

ファイル選択文は一度に複数のファイルを指定できます。複数のファイルを選択した場合、ファイル選択文の戻りファイル名は複数要素となります。

<Archi_1 Script 記述例>

```
1          /*----- read_wave3.prc ---------------*/
2          def ch_name &1 "File_name"
3          def ch_name $1 "num_file"
4          def ch_name $3 "file_counter"
5          $1 = 0
6          repeat_case $1 = 0
7            proc file_name_read{              /* ファイル名を選択する演算処理ブロック*/
8              get file_name &1 $1 "hdr"
9            }file_name_read
10         $3 = 0
11         repeat_case $3 < $1
12           proc 処理{                        /* 選択されたファイルを読出し処理する演算ブロック*/
13             def file_id %1 &1($3) wav
14             read wave %1
15             disp message &1($3)|" 処理"   /* -- 演算処理を記述します---*/
16             $3 = $3+1                          /*  ファイルカウンタ UP */
17           }処理
18         end
```

<Archi_1 Script 記述構文の説明>

8 行目： ファイル選択文で表示される”ファイルを開く“ダイアログで複数のファイルを選択します。
11 行目： 直下に記述された演算処理ブロック”処理”を選択されたファイル個数分繰り返し処理を制御します。
13 行目： ファイル定義文で選択されたファイルを順次属性 wav で定義します。
14 行目： 波形ファイル読み出し文で指定された解析対象ファイルを読み出します。

<補足説明>

記述例はファイルの解析処理部分の記述を省略しています。ファイルごとの処理は 15 行目の disp message 文の替わりに処理を記述します。

記述例5. 4. フォルダ選択してファイル情報を表示し選択された複数ファイルを順次処理する

<Archi_1 Script 記述例>

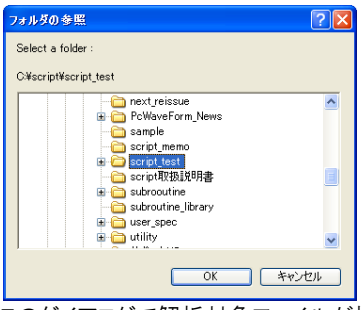
```
1          /*----- read wave4.proc ---------------*/
2          def ch_name &1 "file_name"
3          def ch_name &2 "file_date"
4          def ch_name &3 "file_time"
5          def ch_name &4 "temp1"
6          def ch_name $1 "num_file"
7          def ch_name $2 "check_status"
8          def ch_name $3 "file_counter"
10         $1 = 0                              /*  ファイル数初期化  */
11         repeat_case $1 = 0
12           proc folder_select{
13             get folder_select                 /* フォルダ選択  */
14             read file_info &1 &2 &3 $1 "hdr" /*  ファイル情報取得 */
15           }folder_select
16
17         assign &4 = &1|" "|&2|" "|&3
18         $2 = DAG($1,1)                       /* check box status 初期化 */
19         $1 = 0                               /*  ファイル数初期化 */
20         repeat_case $1 = 0
21           proc file_select{
22             get check_box_status &4 $2 "解析ファイルの選択"
23             $1 = SUM($2)
24           }file_select
25         &1 = CREC(0,$2,&1)                   /* ファイル名を選択ファイルのみに再構成*/
26         &2 = CREC(0,$2,&2)                   /* ファイル年月日を選択ファイルのみに再構成*/
```

```
27          &4 = CREC(0,$2,&3)                  /* ファイル時刻を選択ファイルのみに再構成*/
29          $3 = 0
30          repeat_case $3 < $1
31            proc 処理{                        /* 選択されたファイルを読出し処理する演算ブロック*/
32              def file_id %1 &1($3) wav
33              read wave %1
34              disp message &1($3)|" 処理" /* -- 演算処理---*/
35              $3 = $3+1                        /* file counter UP */
36            }処理
37          end
```

<Archi_1 Script 記述構文の説明>

12 行目～15 行目：演算処理ブロック proc folder_select で選択されたフォルダに解析対象ファイルが存在しないとカレントフォルダ選択を繰り返します：

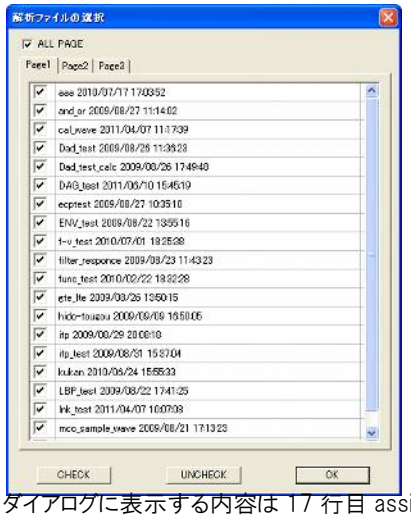
13 行目： カレントフォルダ選択文で実行されるとフォルダ選択ダイアログを表示されます。



このダイアログで解析対象ファイルが格納されているフォルダを選択します。なお、カレントフォルダは暗黙的に選択されたフォルダに設定されます。

14 行目： カレントフォルダ内ファイル情報取得文でフォルダ内の拡張子"hdr"と付けられているファイル名と生成年月日、時刻およびファイル数を取得します。

22 行目： チェックボックスステータス取得文で実行されるとチェック Box ダイアログを表示します。



ダイアログに表示する内容は 17 行目 assign &4 はチェックボックス表示用にファイル名と生成年月日時刻を連結した文字列を生成しています。また、チェック Box ステータスの初期値は 18 行目でファイル数分の要素を 1 とした数列を生成しています。このダイアログで処理するファイルを選択します。

25 行目～27 行目：選択されたファイルだけにファイル名配列、生成年月日配列および生成時刻配列を再構成処理します。

25 行目： ファイル名配列から選択されていないファイルを削除します。

26 行目： ファイル生成年月日配列から選択されていないファイルを削除します。

27 行目： ファイル生成時刻配列から選択されていないファイルを削除します。

30 行目： 直下に記述された演算処理ブロック"処理"を選択されたファイル数分繰り返す制御を行います。

32 行目： ファイル番号定義文でファイル名配列からファイル名を指定して、属性 wav として定義します。

33 行目： 波形ファイル読み出し文で定義されたファイル番号から解析対象ファイルを読み出します。

<補足説明>

ファイルごとの処理は 34 行目の disp message 文の代わりに処理を記述します。

6. 解析に必要な収録情報を取得する

解析に必要な情報を解析対象ファイルから参照する場合、PcWaveForm で解析対象ファイルを読み込み解析範囲が指定されて実行されるか、収録ファイル読み出し文で解析対象ファイルが読み込まれている必要があります。Arch_1 Script 上で解析対象ファイルを読み出す場合データを全て読み出す必要は有りません。read wave %n 0,0 とデータを読み出さず収録ヘッダー部のみ読み出した状態でも取得できます。

解析対象ファイルは、テキスト形式拡張子.hdr のヘッダーファイル（テキスト形式）と拡張子.dat のデータファイル（バイナリ形式）のペア（PcWaveForm 解析ファイル標準 Format）で構成されます。 ヘッダーファイル内容例を示します。

```
DATASET 加速度 DATA_HDR
VERSION 1
SERIES CH_1,CH_2,CH_3,CH_9
DATE 03-30-2004
TIME 12:09:05
RATE 5000
VERT_UNITS G,G,G,V
HORZ_UNITS Sec
NUM_SERIES 4
STORAGE_MODE INTERLACED
FILE_TYPE INTEGER
SLOPE 8.056395e-004,7.292616e-004,7.774538e-004,2.000000e-004
X_OFFSET 0.000000e+000
Y_OFFSET 0.000000e+000,0.000000e+000,0.000000e+000,0.000000e+000
NUM_SAMPS 156138
DATA
DEVICE HR12
FILENAME 加速度 data.dat
CLOCK INTERNAL
COMMENT1 02-16-2004 10:28:04
COMMENT2 XB132
CH1 HR-12,RANGE=0.2V,FILTER=2KHz,NAME=PCB27848(x),MIN=-20.000,MAX=20.000
CH2 HR-12,RANGE=0.2V,FILTER=2KHz,NAME=PCB27848(y),MIN=-20.000,MAX=20.000
CH3 HR-12,RANGE=0.2V,FILTER=2KHz,NAME=PCB27848(z),MIN=-20.000,MAX=20.000
CH9 HR-12,RANGE=5V,FILTER=PASS,NAME=回転パルス
M_OFFSET 0,156137,Modf.Offset,-2.626801e-002,-3.534084e-002,4.455961e-002,0.000000e+000
MARK 537,12:09:05,,1
MARK 12496,12:09:07,,2
MARK 25261,12:09:10,,3
MARK 36548,12:09:12,,3
MARK 63690,12:09:17,,4
MARK 90027,12:09:23,,6
MARK 103061,12:09:25,,7
MARK 124694,12:09:29,,8
MARK 139206,12:09:32,,9
MARK 149418,12:09:34,,0
END 156138
```

※ 赤字は、ヘッダーファイル各行の意味を示すキーワードです。

6. 1. 収録情報を読み出す 収録情報は、サンプリング周期、サンプリング単位、解析対象ファイル名、解析範囲先頭データ番号、収録開始年月日、時刻、オフセ ット秒、データ個数、収録時に付けられたコメント、などが相当します。

6. 1. 3. サンプリング周期を読み出す サンプリング周期取得関数

【PRD **関数**】を使用します。 **文法：**

PRD()　**引数なし**

記述例：

$1 **"サンプリング周期:"** = PRD()

$1 には、ヘッダーファイルの RATE 行または INTERVAL 行がサンプリング周期として格納されます。なお、解析対象ファイルが読み出されていない場合は、直前に def sampl_period 文で定義されたサンプリング周期が格納されます。

6. 1. 4. サンプリング単位を読み出す サンプリング単位取得関数

【CXUT 関数】を使用します。 **文法:**

CXUT()　引数なし

記述例:

&1 “周期単位:” = CXUT()

&1 には、ヘッダーファイルの HORZ_UNIT 行が参照されサンプリング周期単位として格納されます。

6. 1. 5. 解析対象ファイル名を読み出す

PcWaveForm の波形表示 Window で解析範囲を指定して Script を実行した場合に、Script 内で現在の解析対象ファイル名を参照する場合などに使用します。取得方法は、構文記述と演算関数の 2 種が使用できます。演算関数は収録ファイル名取得関数【CFNM 関数】を使用します。

文法:

CFNM()　引数なし

記述例:

&2 “解析対象ファイル名:” = CFNM()

&2 には、現在の解析対象ファイル名が格納されます。この関数は、波形表示 Window から解析範囲を指定して Script に遷移した時に、解析対象ファイル名を取得する時に使用します。

Script 構文では**収録ファイル名取得文**を使用します。

文法:

read file_name 格納先チャネル

記述例:

read file_name &2

6. 1. 6. 解析範囲先頭データ番号を読み出す

PcWaveForm の波形表示 Window で解析範囲を指定して読み出した場合、解析範囲の先頭を Index0 として読み出されます。Script 内で現在の解析対象範囲解析範囲の Index0 が当該収録ファイルの先頭から起算した Index で幾つかなどを参照する場合に使用します。読み出しには解析範囲開始 Index 取得関数【STA 関数】を使用します。

文法:

STA()　引数なし

記述例:

$2 “解析範囲先頭 Index:” = STA()

$2 には、現在解析範囲に指定されている先頭 Index が格納されます。

6. 1. 7. 収録開始年月日を読み出す

読み出される文字列書式は半角 10 文字で MM-DD-YYYY となります。読出しには収録開始年月日取得関数【CSDT 関数】を使用します。

文法:

CSDT() 引数なし

記述例:

&3 “収録開始年月日:” = CSDT()

&3 には、ヘッダーファイルの DATE 行が参照され収録開始年月日として格納されます。

6. 1. 8. 収録開始時刻を読み出す

読み出される文字列書式は半角 8 文字で hh:mm:ss となります。読出しには収録開始時刻取得関数【CSTM 関数】を使用します、

文法:

CSTM()　引数なし

記述例:

&4 “収録開始時刻:” = CSTM()

&4 には、ヘッダーファイルの TIME 行が参照され収録開始時刻として格納されます。

6. 1. 9. 収録開始時刻オフセット値を読み出す

収録開始 X 軸オフセット取得関数【XOF 関数】を使用します、

文法:

XOF()　引数なし

記述例:

$3 "X 軸オフセット:" = XOF()

$3 には、ヘッダーファイルの X_OFFSET 行が参照され X 軸オフセット値として格納されます解析対象ファイルの記載されている収録開始オフセット値が格納されます。単位は収録サンプリング周期となり、例えば時刻歴収録の場合は収録開始時刻にここで取得したオフセット値を減算した結果が真の開始時刻となります。

6. 1. 10. 収録データ個数を読み出す

PcWaveForm の波形表示 Window で解析範囲を指定して Script を実行した場合に、解析対象ファイルの全体でのデータ個数を参照する場合に使用します。取得方法は、収録データ数取得関数【NDT 関数】を使用します。

文法:

NDT()　引数なし

記述例:

$4 "収録データ個数:" = NDT()

$4 には、ヘッダーファイルの NUM_SAMPS 行が参照されチャネル当たりのデータ個数として格納されます。解析対象ファイルの全データを読み出している場合は、読み出された収録チャネルをデータ個数関数(例:LEN(#1))で求めた結果と等価となります。

6. 1. 11. コメントを読み出す

構文記述と演算関数の 2 種が使用できます。演算関数では、コメント取得関数【CCMT 関数】を使用します。

文法:

CCMT(コメント番号)

引数:

読み出すコメント行番号を記述します。行番号は1～3行存在します。なおコメント番号に 0 を記述した場合、すべてのコメント行の一括取得を意味し、コメント行順に格納先チャネルの Index0～2 に格納されます。

記述例:

&5 にコメント行全てを取得格納する場合

&5 "コメント:" = CCMT(0)

&5 には、ヘッダーファイルの COMMENT 行が参照され指定したコメント行の内容が格納されます。

Script 構文では、**コメント取得文**を使用します。

文法:

read comment コメント番号　格納先チャネル

記述例:

&5 にコメント行全てを取得格納する場合

read comment 0 &5

6. 1. 12. カレント設定チャネル番号を読み出す

PcWaveForm 上の波形表示 Window で解析範囲が指定されて Script が実行された時に有効な機能で、カレントチャネル番号取得関数【CCH 関数】を使用します。なお、Script 内で解析対象ファイルを読み出した場合は、無効で-1 が戻ります。**文法:**

CCH()　引数なし

記述例:

$9 "カレントチャネル番号:" = CCH()

$10 = #($9)

$9 に、波形表示 Window 上で設定されているカレントチャネル番号が格納され、$10 にはカレントの収録チャネルのデータが格納されます。

6. 2. チャネル情報を読み出す

チャネル情報は、収録チャネル数、収録チャネル番号、収録チャネルに付けられた信号名、単位、などが相当します。

6. 2. 1. 収録チャネル数を読み出す

構文記述演算関数の 2 種が使用できます。演算関数では、収録チャネル数取得関数【NCH 関数】を使用します。 **文法：**

NCH()　引数なし

記述例：

$5 “収録チャネル数:” = NCH()

$5 には、ヘッダーファイルの NUM_SERIES 行が参照され収録チャネル数として格納されます。

Script 構文では、**収録チャネル数取得文**を使用します。

文法：

read num_ch 格納先チャネル

記述例：

read num_ch $5

6. 2. 2. 収録チャネル番号を読み出す

構文記述と演算関数の 2 種が使用できます。演算関数では収録チャネル番号取得関数【CHS 関数】を使用します。

文法：

CHS()　引数なし

記述例：

$6 “収録チャネル番号:” = CHS()

$6 には、ヘッダーファイルの SERIES 行が参照され収録チャネル番号を SERIES 行の記述順に格納先チャネル$6 の要素に格納されます。例えば、SERIES 行が CH_1,CH_3,CH_4,CH_6 と 4 チャネル収録した場合は、$6(0):=1 , $6(1):=3 , $6(2):=4 , $6(3):=6 と格納されます。

Script 構文では、**収録チャネル番号取得文**を使用します。

文法：

read ch_ch_series 格納先チャネル

記述例：

read ch_series $6

6. 2. 3. チャネルに付けられている信号名を読み出す

構文記述と演算関数の 2 種が使用できます。演算関数では収録チャネル信号名取得関数【CHNM 関数】を使用します。

文法：

CHNM(チャネル番号)

引数：

チャネル番号を記述します。チャネル番号 0 は全てのチャネルの一括取得を意味します。0 を指定した場合、収録チャネル順に格納先チャネルの要素に格納されます。

記述例：

&6 = CHNM(0)

&6 には、ヘッダーファイルの CH 行が参照され CH 行の NAME=で記述されている内容が信号名として格納されます。なお、信号名が付けられていないチャネルは半角”#”に続きチャネル番号を信号名として戻します。

Scritp 構文では**収録チャネル信号名単位取得文**を使用します。

文法：

readch_name 信号名格納先チャネル 単位格納先チャネル　チャネル番号

記述例：

read ch_name &6 &7

&6 には、収録順に信号名、&7 には同じく収録順に単位が格納されます。

6. 2. 4. チャネルに付けられている単位を読み出す 収録チャネル

単位取得関数【CUNT 関数】を使用します。 **文法:**

CUNT(チャネル番号)

引数:

チャネル番号を記述します。チャネル番号 0 は全てのチャネルの一括取得を意味します。0 を指定した場合、収録チャネル順に格納先チャネルの要素に格納されます。

記述例:

```
&7 = CUNT(0)
```

&7 には、ヘッダーファイルの VERT_UNIT 行が参照されチャネルごとに単位が格納されます。なお、単位が付けられていないチャネルは半角スペースが戻ります。

記述例:

解析対象ファイルの全てのチャネルを$100 からチャネル数分のチャネルにコピーし、同じ名前、単位を定義する場合

```
$1 = CHS()                          /* 収録チャネル番号の取得*/
$2 = NCH()                          /* 収録チャネル数の取得＊・
&1 = CHNM(0)                        /* 全チャネル一括信号名の取得*/
&2 = CHUT(0)                        /* 全チャネル一括単位の取得*/
$3 = 0                              /* チャネルカウンタ初期化*/
repeat_case $3 < $2                 /* チャネル数分の繰り返し*/
  proc 信号名単位定義{
    $4 = $3+100)                    /* 格納先チャネル番号生成*/
    $($4) = #($1($3))               /* 収録チャネルのコピー*/
    def ch_name $($4) &1($3)        /* 信号名の定義*/
    def ch_unit $($4) &2($3)        /* 単位名の定義*/
    $3 = $3+1                       /* チャネルカウンタ UP*/
  }信号名単位定義
```

$100 からチャネル数分のチャネル番号に収録チャネルの値が格納され、それぞれのチャネルの収録チャネルと同じ信号名、単位が付けられます。

6. 3. マーク情報を読み出す 解析対象ファイルに付けられているマーク情報を解析制御などに使用する場合読み出す情報で、マーク個数情報、マーク位置 index 情報、マークメモ情報が相当します。マーク情報を使用する場合、必ず処理の先頭で現在の解析範囲にマーク情報が存在し ているかを確認する必要があります。マーク情報の存在を確認する手段としてマーク個数を取得します。 なお、マーク番号は、解析範囲に存在する先頭マークを 1 として振られ、絶対番号ではありません。解析範囲に存在するマーク番号は 1 から、ここで取得したマーク個数までとなります。

6. 3. 1. 解析範囲に付けられているマーク個数を読み出す

構文記述と演算関数の 2 種が使用できます。演算関数ではマーク個数取得関数【NMK 関数】を使用します。 **文法:**

NMK()　引数なし

記述例:

```
$7 = NMK()
```

＄7には、現在解析範囲にあるヘッダーファイルの MARK 行の総数が格納されます。MARK 行の総数は現在の解析範囲に付けられているマーク個数(箇所数)を意味しヘッダーファイルの MARK 行の総数とは異なります。例えば、$7 に 3 と格納された場合は、現在の解析対象範囲に MARK 行が 3 行存在(相対マーク番号 1,2,3 の 3 個)していることを示しています。

Script 構文では**マーク個数取得文**を使用します。

文法:

read num_mark 格納先チャネル

記述例:

```
read num_mark $7
```

6. 3. 2. マーク位置のデータ番号を読み出す

6.3.1 で取得したマーク個数範囲のマーク番号を指定して、マークが付けられている index(データ番号)を取得します。 取得方法は、構文記述と演算関数の 2 種が使用できます。演算関数ではマーク番号 Index 取得関数【MRK 関数】を使用し ます。

文法:

MRK(マーク番号)

引数:

マーク番号 0 は全てのマーク位置データ番号(相対 index)の一括取得を意味します。
0 を指定した場合、マーク番号順に格納先チャネルの要素に格納されます。

記述例:

$8 = MRK(0)

$8 には、ヘッダーファイルの現在指定されている解析範囲の MARK 行に記載されている Index から現在の解析範囲のファイル先頭からの Index を減算した値(相対 index)が格納されます。指定されたマーク番号が存在していない場合は-1 が戻ります。

Script 構文では**マーク番号 Index 取得文**を使用します。 **文法:**

read mark_pos マーク番号 格納先チャネル

記述例:

read mark_pos 0 $8

6. 3. 4. マークに付けられているマークメモを読み出す

マーク番号を指定し、構文記述と演算関数の 2 種が使用できます。演算関数ではマークメモ取得関数【CMMR 関数】を使用します。

文法:

CMMR(マーク番号)

引数:

マーク番号 0 は全てのマーク番号に付加されているマークメモの一括取得を意味します。
0 を指定した場合、マーク番号順に格納先チャネルの要素に格納されます。

記述例:

&8 = CMMR(0)

&8 には、ヘッダーファイルの現在指定されている解析範囲の MARK 行に記載されているマークメモが格納されます。なお、マークメモが省略されている場合は半角スペースが戻ります。

Script 構文ではマークメモ取得文を使用します。

文法:

read mark_memo マーク番号 格納先チャネル

記述例:

read mark_memo 0 &8

6. 3. 5. 絶対マーク番号を読み出す

マーク番号は指定された解析範囲に付けられている先頭マークを 1 として振られます。絶対マーク番号とは、解析範囲に付けられた先頭マーク番号 1 が収録ファイルに付けられているマークの先頭からの起算したマーク番号では幾つに相当するかを意味します。絶対マーク番号の取得は絶対マーク番号取得関数【MKS 関数】を使用します。

文法:

MKS()　引数なし

記述例:

$9 = MKS()

$9 には、ヘッダーファイルの現在指定されている解析範囲の MARK 行の先頭がヘッダーファイル全体のマーク行の何行目に当たるかを格納します。つまり、記載されている解析範囲に付けられている先頭マーク番号1が、解析対象ファイルの先頭から何番目かで表したマーク番号が格納されることになります。

6. 4. ヘッダーファイルを読み出す ヘッダーファイルに記載されてる内容を読み出したいが、関数或いは構文が用意されていない部分を読み出したい場合に使用します。 読み出す場合は Script 構文ヘッダーファイル読み出し文を使用します。

文法:

read header_info 格納先チャネル 行数 先頭文字列

記述例:

ヘッダーファイルの TIME 行を読み出す場合

read header_info &9 $10 "TIME " 先頭文字列を指定した場合は記述した文字列から始まる行のみ読み出します。読み出された文字列は記述した先頭文字 列が削除されて格納されます。また、先頭文字列を記述しない場合はヘッダーファイル全体が行ごとに格納先チャネルの要 素に格納されます。行数は読み出された行数が格納されます。
記述例では、ヘッダーファイルの TIME 行が参照され、TIME 行が"TIME 12:09:05"の場合 12:09:05 が格納されます。

記述例:

ヘッダーファイル MARK 行を読み出す場合

read header_info &1 $1 "MARK "

読み出し対象ヘッダーファイルの内容が 6 章の先頭に記載したヘッダーファイルサンプル例の場合、&1 の要素ごとに

&1(0) ⇒"537,12:09:05,,1"
&1(1) ⇒"12496,12:09:07,,2"
&1(2) ⇒"25261,12:09:10,,3"
&1(3) ⇒"36548,12:09:12,,3"
&1(4) ⇒"63690,12:09:17,,4"
&1(5) ⇒"90027,12:09:23,,6"
&1(6) ⇒"103061,12:09:25,,7"
&1(7) ⇒"124694,12:09:29,,8"
&1(8) ⇒"139206,12:09:32,,9"
&1(9) ⇒"149418,12:09:34,,0"

が格納され、$1 には存在した行数 10 が格納されます。

記述例6. 1. 解析対象ファイルのチャネル情報を取得する

解析対象ファイルの収録チャネル数および収録チャネル番号とチャネルごとの信号名および単位を取得します。 解析対象ファイルが読み込まれていない、あるいは実行前に PcWaveForm 上で読み出し解析範囲を設定して実行してない場合は読 み出すことはできません。

<Archi_1 Script 記述例>

```
1       /*--- ana_info_get1.prc ----------------------------*/
2       def ch_name &1 "ch_name"
3       def ch_name &2 "ch_unit"
4       def ch_name $1 "num_ch"
5       def ch_name $2 "ch_series"
6       $1 = NCH()      /* ヘッダーファイルの NUM_SERIES 行を参照*/
7       $2 = CHS()      /* ヘッダーファイルの SERIES 行を参照*/
8       &1 = CHNM(0)            /* ヘッダーファイルの CH 行を参照*/
9       &2 = CUNT(0)            /* ヘッダーファイルの VERT_UNITS 行を参照*/
```

<Archi_1 Script 記述構文の説明>

6 行目： 収録チャネル数取得関数:NCH()を使用し収録チャネル数を取得します。 NCH()関数はヘッダーファイルの NUM_SERIES行から収録チャネル数を読み出します。

7 行目： 収録チャネル番号取得関数:CHS()を使用し収録チャネル番号を取得します。CHS()関数はヘッダーファイルの SERIES 行から収録チャネル番号を読み出します。収録チャネル番号は、例えば収録する時に ch1,ch3,ch7,ch10 の 4 チャネル収録した場合は$2 には 1,3,7,10 が格納されます。

8 行目： 収録チャネル信号名取得関数:CHNM(0)を使用し各チャネルに付けられている信号名を取得します。CHNM()関数はヘッダーファイルの CH 行の NAME 項目から信号名を読み出します。

9 行目： 収録チャネル単位名取得関数:CUNT(0)を使用し各チャネルに付けられている単位名を取得します。CUNT()関数はヘッダーファイルの VERT_UNITS 行から単位を読み出します。

記述例6. 2. 解析対象ファイルのサンプリング情報を取得する

解析対象ファイルのサンプリング周期とその単位を取得します サンプリング周期取得関数およびサンプリング単位取得関数は、解析対象ファイルのサンプリング情報取得専用ではありません。現在 設定されているサンプリング情報です。従って、サンプリング周期定義文でサンプリング周期を変更すると、その値が戻ります。

<Archi_1 Script 記述例>

```
1       /*--- ana_info_get2.prc ----------------------------*/
2       def ch_name $3 "period"
3       def ch_name &3 "samplig_unit"
4       $3 = PRD()                      /* ヘッダーファイルの RATE 行または INTERVAL行を参照*/
5       &3 = CXUT()                     /* ヘッダーファイルの HORZ_UNITS 行を参照*/
```

<Archi_1 Script 記述構文の説明>

4 行目： サンプリング周期取得関数:PRD()を使用して収録サンプリング周期を取得します。なお、サンプリング周波数として使用する場合は 1/PRD()と逆数を演算します。サンプリング周期はヘッダーファイルの INTERVAL 行または RATE 行から取得し、RATE 行から読み出した場合はその逆数の周期に変換されています。なお、PRD()関数で取得する周期はサンプリング周期定義文で周期を変更すると変更後の周期が取得されます。

5 行目： サンプリング単位取得関数:CXUT()を使用してサンプリング周期の単位を取得します。サンプリング単位はヘッダーファイルの HORZ_UNITS 行から取得します。

記述例6. 3. 収録開始年月日/時刻、X オフセットおよびデータ個数、解析範囲先頭 index を取得する

解析対象ファイルの収録開始年月日/時刻、X オフセット値および収録データ個数、解析先頭 Index を取得します 解析対象ファイルが読み込まれていない、あるいは実行前に PcWaveForm 上で読み出し解析範囲を設定してから実行しない場合は 読み出すことはできません。

解析範囲が指定されて実行した場合、先頭 index は 0 となります、解析範囲先頭 Index とは、Index0 がファイル先頭から何番目の Index に相当するかを意味します。

<Archi_1 Script 記述例>

```
1       /*--- ana_info_get3.prc ----------------------------*/
2       def ch_name &4 "acq_start_date"
3       def ch_name &5 "acq_start_time"
```

```
4          def ch_name $4 "x_axis_offset"
5          def ch_name $5 "num_data"
6          def ch_name $6 "start_index"
7          &4 = CSDT()    /* 収録開始年月日 MM-DD-YYYY */
8          &5 = CSTM()    /* 収録開始時刻 hh:mm:ss */
9          $4 = XOF()     /* X 軸オフセット値 */
10         $5 = NDT()     /* 収録データ個数 */
11         $6 = STA()     /* 解析範囲先頭 index */
```

<Archi_1 Script 記述構文の説明>

7 行目： 解析開始年月日取得関数:CSDT()を使用して収録開始年月日を取得します。文字列形式は月日年の順(MM-DD-YYYY)の形式となります。CSDT()関数はヘッダーファイルの DATE 行から収録開始年月日を読み出します。

8 行目： 解析開始時刻取得関数:CSTM()を使用して収録開始時刻を取得します。文字列形式は hh:mm:ss となります。CSTM()関数はヘッダーファイルの TIME 行から収録開始時刻を読み出します。通常、レコーダまたは直接コンピュータで 収録した場合、内部クロックから収録開始時点のカレンダをタイムスタンプとして付加しますので仮に正秒から開始されてい ない場合でも秒以下は切り捨てられて記述されています。

9 行目： 収録開始 X 軸オフセット取得関数:XOF()を使用し、収録先頭データと開始時刻とのオフセットを取得します。XOF()関数はヘッダーファイルの X_OFFSET 行から X 軸上のオフセット値を読み出します。なお、オフセット値は解析範囲の 先頭と解析対象ファイルの先頭とのオフセットではありません。X 軸オフセットとは、プレトリガ収録等を行った時にプレトリガ 一時間を意味します。なお、収録データファイルから一部を切り取って格納された収録ファイルを解析対象ファイルとした場 合などは、元の収録データからのオフセット値が書き込まれている場合があります。

10 行目： 収録データ個数取得関数:NDT()を使用して収録ファイルのチャネル当たりのデータ個数を取得します。 NDT()関数はヘッダーファイルの NUM_SAMPS 行から収録データ個数を読み出します。解析範囲のデータ個数はデータ個数関数:LEN(#n)を使用して取得できます。解析対象範囲が収録データの全範囲の場合は、NDT()と LEN(#n)の結果は等しくなります。

11 行目: 解析範囲開始 Index 関数:STA()使用して現在解析範囲に指定されている先頭データの Index が収録ファイルのどの Index かを取得します。解析対象範囲が収録データの全範囲の場合は 0 となります。

記述例6. 4. 解析対象ファイルに付けられている Mark 情報を取得する

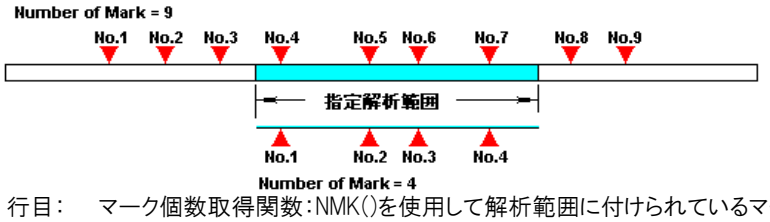
解析対象ファイルに付けられているマーク位置の Index とマークに付けられているマークメモを取得します 解析対象ファイルが読み込まれていない、あるいは実行前に PcWaveForm 上で読み出し解析範囲を設定して実行しない場合は読み 出すことはできません。また、マーク個数取得関数 NMK()を除き、マークが付けられていない場合は使用できません。 マーク番号は解析範囲に付けられている先頭マーク番号を 1 として相対的に振られます。絶対マーク番号とはマーク番号1が収録ファ イルの先頭から何番目のマーク番号かを意味します。

<Archi_1 Script 記述例>

```
1          /*--- ana_info_get6.prc ----------------------------*/
2          def ch_name &9 "mark_memo"
3          def ch_name $9 "num_mark"
4          def ch_name $10 "mark_index"
5          def ch_name $11 "abs_mark_No."
6          $9 = NMK()                  /* 解析範囲に付けられているマーク個数 */
7          case $9 > 0
8            proc get_mark_info{
9              $10 = MRK(0)            /* マーク位置 index 取得 */
10             &9 = CMMR(0)            /* マークメモ取得 */
11             $11 = MKS()             /* 絶対先頭マーク番号取得 */
12           }get_mark_info
```

<Archi_1 Script 記述構文の説明>

マークは収録中あるいは収録後 PcWaveForm 上の機能で付けられます。マーク番号はファイル先頭から付けられたマーク順に1から振られます。但し、解析範囲を指定するとマーク番号は指定された解析範囲に存在する先頭マークをマーク番号 1 として相対的に振り替えられます。



6 行目： マーク個数取得関数:NMK()を使用して解析範囲に付けられているマーク個数を取得します。

演算処理ブロック get_mark_info はマーク個数>0 の時に処理します。マーク個数が 0 の時にマーク情報を取得すると実行時エラーとなることを防止しています。

9 行目： マーク番号 index 変換関数：MRK(0)を使用し解析範囲に付けられているマーク位置の Index を取得します。

10 行目：マークメモ取得関数：CMMR(0)を使用してマークに付けられているマークメモを取得します。

11 行目：絶対マーク番号取得関数 MKS()を使用して解析範囲に付けられたマーク番号1が収録ファイルの何番目マークに当たるかの絶対マーク番号を取得します。したがって、解析範囲に付けられたマーク番号は 1～$9 までとなり、収録ファイルに付けられたマーク番号で表すと$11～$11+$9-1 となります。

7. 解析対象ファイルのチャネルデータを参照する

収録チャネルを表す場合、先頭文字半角"#"に続きチャネル番号で記述します。但し、解析対象ファイルを読み出していない場合は収録チャネル属性のチャネルは使用できません。 収録チャネルへのデータは解析対象ファイルを読み出すと、解析対象ファイルに存在している収録チャネル番号は自動的に収録デー タが格納されています。なお、PcWaveForm 上で解析範囲を指定して実行した場合は、改めて読み出し操作を行わなくても実行時に 値を持ちます。また、構文上で別の解析対象ファイルを読み出した場合は、当該ファイルに存在している収録チャネルの値は読み出さ れた解析対象ファイルの収録チャネルのデータに置き換わります。一旦、読み出した後、別の解析対象ファイルを読み出した場合、当 該解析対象ファイルに当該チャネルが存在していないと直前に格納されていたデータを失い、Null となり、そのまま参照すると実行時エ ラーとなります。

7. 1. チャネル番号を直接指定して参照する方法

収録チャネル番号が既知の場合は収録チャネルを表す先頭文字半角"#"に続き収録チャネル番号で記述します。

記述例:

```
$1 = #1
```

$1 には、収録チャネル 1 の解析範囲の全てのデータが格納されます。
※ #1 は収録チャネル 1 を意味します。

7. 2. チャネル番号を間接指定して参照する方法 解析対象ファイルのチャネル構成や収録チャネル番号が不明な場合など、構文上で直接チャネル番号を記述すると、チャネル番 号が異なっている場合は記述したチャネル番号を修正する必要があります。間接指定とは、収録チャネルを表す半角"#"に続きチャ ネル番号で表す部分を括弧で記述しその括弧内にチャネル番号を記述する方向を呼称します。例えば、#1 と表記する収録チャネ ル 1 は#(1)の様に記述しても同じ意味を持ち、#($2)と記述したら括弧内の参照チャネル$2 の値をチャネル番号とした収録チャネル を意味します。なお、チャネル番号の間接指定は収録チャネル以外でも使用できます。

記述例:

```
$1 = CHS()
$2 = #($1(0))
```

$2 には、収録先頭チャネルのデータが格納されます。

7. 3. 記述したチャネル番号と関係付けて参照する方法

収録チャネル番号が不明な場合や解析対象ファイルごとに異なっている場合など、構文中に固定的に収録チャネル番号を記述すると、チャネル構成が異なった解析対象ファイル処理する都度、チャネル番号を修正する必要が発生します。また、間接指定では、処理上で収録チャネル番号に意味を持たせている場合などでは不都合が発生する場合があります。 その場合、解析対象ファイルのチャネル構成と構文で記述した収録チャネル番号の対応表が内蔵されており、解析対象ファイルの 収録チャネル番号が記述されている構文上の収録チャネルのどの番号に対応するかを対応で関係付ける事ができます。

収録チャネル割り当て文

文法:

```
get replace_ch 収録チャネル列
```

記述例:

```
get replace_ch #1,#2,#5
```

構文中で記述した#1,#2,#5 に解析対象ファイルの収録チャネルを割り当てます。
※ 実行するとチャネル割り当てダイヤログが表示されます。ダイヤログの操作は後述する記述例 7.3 項を参照下さい。

記述例7. 1. チャネル番号を直接記述して参照する

解析対象ファイルのチャネル構成が既知で、その構成が処理対象ファイルごとに異ならない場合などは、収録チャネルを表す先頭文字半"#"に続きチャネル番号で表す収録チャネルで記述します。この場合チャネル番号は収録チャネル番号を意味します。

<Archi_1 Script 記述例>

```
1       /*--- ana_ch_set1.prc ----------------------------*/
2       $10 "ch1_平均値:" = MEA(#1)          /*  収録 ch1 の平均値を求める*/
3       $11 "ch1_最大値:" = MAX(#1)          /*  収録 ch1 の最大値を求める*/
4       $12 "ch1_最小値:" = MIN(#1)          /*  収録 ch1 の最小値を求める*/
5       $13 "ch2_平均値:" = MEA(#2)          /*  収録 ch2 の平均値を求める*/
6       $14 "ch2_最大値:" = MAX(#2)          /*  収録 ch2 の最大値を求める*/
7       $15 "ch3_最小値:" = MIN(#2)          /*  収録 ch3 の最小値を求める*/
```

<Archi_1 Script 記述構文の説明>

2 行目～7 行目：収録チャネル ch1 と ch2 のそれぞれ平均値、最大値、最小値を求めています。

記述例7. 2. チャネル番号を間接指定して参照する

構文上の収録チャネルを間接指定し、チャネル番号取得関数：CHS()を使用して収録チャネル番号を取得して、その値を使用して収録チャネルを指定します。

<Archi_1 Script 記述例>

```
1       /*--- ana_ch_set2.prc ----------------------------*/
2       def ch_name &1 "ch_name"
3       def ch_name &2 "ch_unit"
4       def ch_name $1 "num_ch"
5       def ch_name $2 "ch_series"
6       def ch_name $3 "ch_counter"
7       def ch_name $4 "new_ch_series"
8       $1 = NCH()
9       $4 = ACC(DAG($1,1))+100      /*  演算結果格納先チャネル列を生成します */
10      &3 = CHNM(0)
11      &4 = CUNT(0)
12      assign &3 = &3|"_50Hz_LPF:"|&4
13      def ch_name $($4) &3
14      $2 = CHS()
15      $3 = 0
16      repeat_case $3 < $1
17        proc LPF{
18          $($4($3)) = LPF(50,#($2($3)))   /* 収録チャネルに 50Hz ロ-パスフィルタ処理*/
19          $3 = $3+1
20        }LPF
```

<Archi_1 Script 記述構文の説明>

8 行目：　解析対象ファイルの収録チャネル数を取得します。
9 行目：　演算結果も間接指定するため、演算結果格納先チャネル番号を生成しています。
　　$4 は 100 から 101,102,103,…と$1 で示すチャネル数分の数列となります。
10 行目：解析対象ファイルの信号名を取得します。
11 行目：解析対象ファイルの単位を取得します。
12 行目：信号名に文字列"_50HzLPF:"と単位を連結します。
13 行目：演算結果格納先チャネルに収録ファイルと同じ信号名単位を定義します。
16 行目：解析対象ファイルの収録チャネルを 12 行目で$2 に取得したチャネル番号で間接指定し演算結果を 9 行目で$4 に生成した格納先チャネル番号で間接指定して格納しています。

<補足説明>

16 行目に記述した間接指定記述#($2($3))は#(チャネル番号)、チャネル番号=$2(index)、Index=$3 と分解できます。$2 は解析対象ファイルのチャネル番号並びが格納されています。$3 はチャネルカウンタとして使用しています。
例えば、$2(0) := 1、$2(1) := 3、$2(2) : = 4、$2(3) :=6 とした場合、チャネルカウンタ$3 が 1 の時、チャネル番号が 3 となり、したがって#(3) = #3 となり収録チャネル ch3 を指す事になります。

記述例7. 3. 記述した収録チャネル番号と解析対象ファイルのチャネル番号を関係付けて参照する

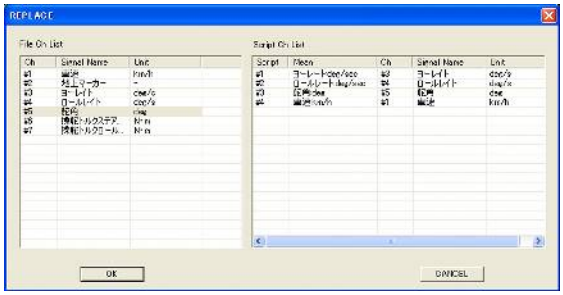
解析対象ファイルの収録チャネル構成と構文上で記述した収録チャネル番号との対応表を表示し、関係付けて参照します。

<Archi_1 Script 記述例>

```
1       /*--- ana_ch_set3.prc ----------------------------*/
2       dcl sheet 1 {
3         page 1:
4         column $1,$2,$3,$4,$5
5         format F3,F3,F3,F3,F3 2
6       }sheet
7       assign &1 = "ヨーレート:deg/sec","ロールレート:deg/sec","舵角:deg","車速:km/h"
9       def ch_name $1 "舵角ヨーレート最大相間値"
10      def ch_name $2 "舵角ヨーレート遅延 τ:sec"
11      def ch_name $3 "舵角ロールレート最大相関値"
12      def ch_name $4 "舵角ロールレート遅延 τ:dec"
13      def ch_name $5 "平均車速:km/h"
14      def ch_name $6 "temp1"
15      def ch_name $7 "temp2"
17      get replace_ch #1,#2,#3,#4 &1
19      $6 = COR(#3,#1)
20      $7 = COR(#3,#2)
21      $1 = MAX($6)
22      $2 = MXP($6)*PRD()
23      $3 = MAX($7)
24      $4 = MXP($7)*PRD()
25      $5 = MEA(#4)
26
27      write ch_column 1: $1,$2,$3,$4,$5
28      end
```

<Archi_1 Script 記述構文の説明>

17 行目: 収録チャネル割り当て文(get replace_ch)が実行されるとダイアログが表示されます。Arch1_1 Script 記述上では収録チャネル ch1:(#1)をヨーレート、ch2:(#2)をロールレート、ch3:(#3)を舵角、ch4(#4)を車速として記述していますが、記述したチャネル番号に解析対象ファイルの収録チャネル番号を割り当てます。



現在解析対象ファイルのチャネル構成が左側に表示され、右側に構文上で記述した割り当て収録チャネルが表示されます。記述したチャネル番号と解析対象ファイルのチャネル構成が異なっている場合、左側収録チャネルリストからドラッグ&ドロップして、割り当てます。

<補足説明>

割り当てられた結果、収録チャネル対応表が更新され、例えば記述上の収録チャネル#1 が割り当てた解析対象ファイルの収録チャネル ch3 と異なっていた場合、記述されている#1 を参照する場合に#1 は収録チャネル ch3 と自動的に読み変えられます。 但し、#1 が ch2 でも ch3 でもあるとする 1 対 n の関係を持たせることはできません。なお、収録チャネル対応表は解析対象ファイルに 従属し、別の解析対象ファルが読み出されると初期化されます。

8. 演算に必要なパラメータを受け取る

演算に必要なパラメータは Archi_1 Script 記述で固定的する以外に、構文で取得することができます。 なお、テキストファイルからパラメータ取得する場合は後述する第12章「テキストファイルを読み出す」項に記載します。

8. 1. キーボードから入力する

値入力文

文法:

get value 格納チャネル列 フラグ 表題

記述例:

```
$1 = 100.24
$2 = 256
get value $1(F2),$2(F0) "値の入力"
```

入力するチャネルの値は事前に初期値の設定が必要です。また、個々の入力する値にリストボックス形式での選択も可能です。戻り値は入力値または初期値となります。なお、入力する項目に書式指定しない場合は指数形式で表示されますが、入力書式は拘束されません。操作は後述する記述例 8.1.項を参照下さい。

記述例:

リストボックスからの選択入力と直接入力を混合する場合

```
assign $1 = 10,20,30,40,50
assign &1 = "Tokyo","kanagawa","Saitama","Ibaragi","Chiba","Gunma"
$2 = $1(0)
assign &2 = &1(2)
get value $1:$2(F0),&1:&2
```

予めリストボックス配列(数列)を定義し、半角コロンで区切って入力値を記述します。表示されるダイヤログの操作は後述する記述例8. 2. 項を参照下さい。

8. 2. ラジオボタンを表示して選択する

ラジオボタンスタータス取得文

文法:

get radio_button_status 表示項目名 選択番号格納先 表題

記述例:

```
assign &1 "表示項目名:" = "k  m/h","m/s","mph"
$1 = 0
get radio_button_status &1 $1 "速度単位の選択"
```

複数の項目名を表示し、その中の一個を選択します。戻り値は選択された項目番号となります。 項目番号は表示項目名の Index を意味します。表示されるダイヤログの操作は後述する記述例8. 3. 項を参照下さい。

8. 3. チェックボックスを表示して選択する

チェックボックススタータス取得文

get check_box_status 表示項目名 スタータス格納先 表題

記述例:

```
assign &1 "表示項目名:" = "平均値","最大値","最小値","最大振幅値"
assign $1 "チェックボックスステータス:" = 1,0,0,0
get check_box_status &1 $1 "解析項目の選択"
```

複数の項目名を表示させ、同時に複数選択可能です。戻り値は項目ごとの論理値数列となります。表示されるダイヤログの操作は後述する記述例8. 4. 項を参照下さい。

8. 4. 諸元表ファイルから選択する

諸元表ファイル参照文 文法:

get param %n 諸元値格納先 諸元名単位格納先

諸元値格納先は数値属性参照チャネル、文字属性参照チャネルの何れかまたは両方で記述します。数値属性参照チャネルで記述した場合は、選択された列の数値変換可能な項目のみ格納され、文字列項目は無視されます。文字属性参照チャネルで記述した場合は、選択された列の全ての項目が格納されます。又、数値属性参照チャネルと文字属性参照チャネルの両方を記述した場合は、数値属性チャネルに数値変換可能な項目が格納され、数値変換できない項目は文字属性参

照チャネルに格納されます。例えば、全ての項目が数値変換できない場合は、数値属性チャネルには何も格納されません。同様に全て数値変換可能項目の場合は文字属性参照チャネルには何も格納されません。 諸元表の諸元名、単位は諸元表の 1 列と 2 列目に記述されていますが、選択することはできず、取得したい場合は、諸元名
単位格納先に文字属性参照チャネルを半角カンマで区切って 2 チャネル記述します。先に記述した文字属性参照チャネルに諸元名、後に記述した文字属性参照チャネルに単位が格納されます。なお、諸元名単位格納先は記述省略できます。
＜諸元ファイル例＞

	MEMO	諸元表1					
	NAME	UNIT	型式1	型式2	型式3	型式4	型式5
	model		aaa	bbb	ccc	ddd	eee
	仕向け		日本	USA	AUS	EU	ASIA
	気筒		6	6	5	5	4
	排気量	cc	1800	2500	1800	1800	1500
	変速機		5AT	4AT	4AT	5AT	4AT
	演算code		1	2	2	3	4
	重量	kg	1250	1500	1300	1300	1400

記述例:

```
def file_id %1 “諸元表 3” prm
get param %1 $1 &1
```

諸元表ファイル参照文は事前にファイル番号定義文で、属性:prm として定義が必要です。 諸元表ファイル内容を表示し、列を選択します。表示されるダイヤログの操作は後述する記述例8. 5. 項を参照下さい。

8. 5. 表示した二値から選択する

メッセージ表示応答文

文法:

```
get reply  表題 ボタン 1 ボタン 2 応答格納先
```

記述例:

```
get reply “解析を続けますか?” “はい” “いいえ” $1
```

「はい」「いいえ」をラベルとした場合、戻り値は「はい」がクリックされると$1 は 0、「いいえ」がクリックされると$1 は 1 が戻ります。
表示されるダイヤログの操作は後述する記述例8. 6. 項を参照下さい。

8. 6. テキスト全体を表示して選択する

テキスト表示選択文

文法:

```
get text_page 表示テキストチャネル列 選択ページ格納先  表題
```

記述例:

```
assign &1 “page1:” = “閾値:＝12.3”,”上限値:=85.0”,”下限値:=5.5“
assign &2 “page1:” = “閾値:＝8.3”,”上限値:=73.0”,”下限値:=2.5“
assign  &3  “page1:”  =  “ 閾 値 : ＝ 15.5”,” 上 限 値 :=97.50”,” 下 限
値:=8.5“ assign &4 “page1:” = “閾値:＝20”,”上限値:=120.0”,”下限
値:=12.05“ get text_page &[1,4] $1 “解析パラメータ選択”
```

テキストが格納されているチャネルをページごとに表示し戻り値はページ番号となります。
戻りページ番号は 1 から振られます。表示されるダイヤログの操作は後述する記述例8. 7. 項を参照下さい。

記述例8. 1. キーボード入力する

<Archi_1 Script 記述例>

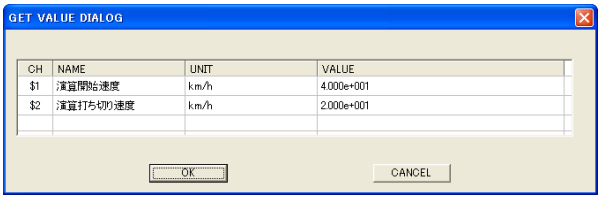
```
1        /*-------param1.prc-------------*/
2        def ch_name $1 "演算開始速度:km/h"
3        def ch_name $2 "演算打ち切り速度:km/h"
4        $1 = 40      /* 省略値設定*/
5        $2 = 20      /* 省略値設定*/
6        get value $1,$2
```

<Archi_1 Script 記述構文の説明>

6 行目： キーボードから値を入力する値入力文で、本文が実行されると、値入力ダイアログが表示されます。ダイアログの value 欄をクリックすると入力欄となり、キーボードから値を入力します。
4 行目： 入力する$2 の初期値を設定します。
5 行目： 入力する$2 の初期値を設定します。
6 行目： 値入力文で記述された$1 と$2 に値を入力します。

<補足説明>

入力値の表示形式を指定しないと指数形式となります。表示形式を指定した場合入力された戻り値もその表示形式に従います。なお、初期値は特に設定しなくても問題ありませんが、入力しないままダイアログを閉じると、null のまま戻り、以後、参照された時に実行時エラーとなりますので、設定することを推奨します。ダイアログを閉じる時に「CANCEL」ボタンをクリックすると初期表示値が戻ります。



記述例8. 2. キーボード入力にリストボックスを使用する

<Archi_1 Script 記述例>

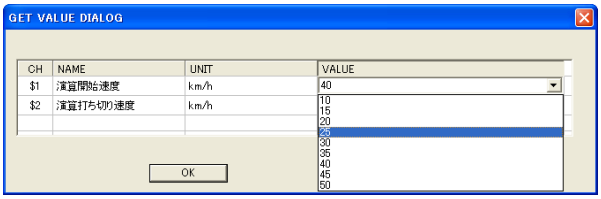
```
1        /*-------param2.prc-------------*/
2        def ch_name $1 "演算開始速度:km/h"
3        def ch_name $2 "演算打ち切り速度:km/h"
4        assign $3 = 10,15,20,25,30,35,40,45,50
5        assign $4 = 5,10,15,20,25
6        $1 = $3(6)      /* 省略値設定*/
7        $2 = $4(4)      /* 省略値設定*/
8        get value $3:$1(F0),$4:$2(F0)
```

<Archi_1 Script 記述構文の説明>

8 行目： キーボードから値を入力する値入力文で、その引数に選択する内容を半角コロンで記述します。
4 行目： 選択入力する$1 用の ListBox を生成します。
5 行目： 選択入力する$2 用の ListBox を生成します。
6 行目： 入力する$1 の初期値を代入します。
7 行目： 入力する$2 の初期値を代入します。
8 行目： 値入力文で記述された$1,$2 を対応した ListBox$3, $ 4 から、選択入力します。

<補足説明>

ダイアログの値入力欄をクリックすると ListBox 記述された項目欄は ListBox が表示され、入力する内容を選択します。表示形式は ListBox 側には記述できません。ListBox 側の表示形式は対応する入力値側に設定した表示形式が自動的に参照されます。



記述例8. 3. ラジオボタンで選択する

<Archi_1 Script 記述例>

```
1          /*------param3.prc-----------*/
2          def ch_name &1 "演算項目"
3          def ch_name $1 "選択番号"
4          assign &1 ="高低左","高低右","通り左","通り右","水準","軌間","平面性"
5          $1 = 2   /* 選択初期値設定*/
6          get radio_button_status &1 $1
```

<Archi_1 Script 記述構文の説明>

4 行目:　ラジオボタンに表示する内容を文字列配列で定義します。
5 行目:　選択初期値番号を指定します。
6 行目:　ラジオボタンを選択します。

<補足説明>

ラジオボタンステータス取得文は、幾つかの項目から唯一無二で項目を選択する場合に使用します。
6 行目のラジヲボタンステータス取得文:get radio_button_status が実行されるとダイアログが表示されます。
表示される項目は 4 行目で&1 に生成します。ダイアログ~の戻りは$1 に選択された項目番号が戻ります。なお、項目番号は 0 から振られます。(左例では 2 が戻ります)

記述例8. 4. チェックボックスで選択する

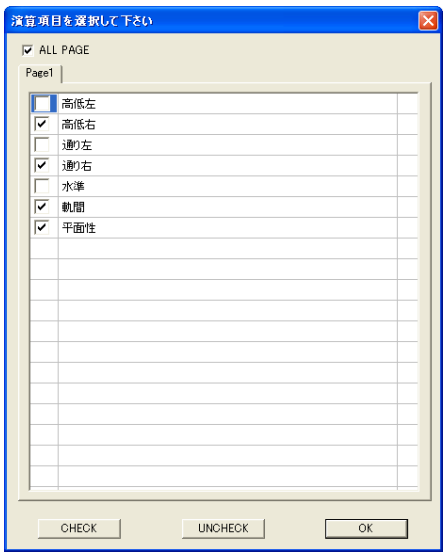
<Archi_1 Script 記述例>

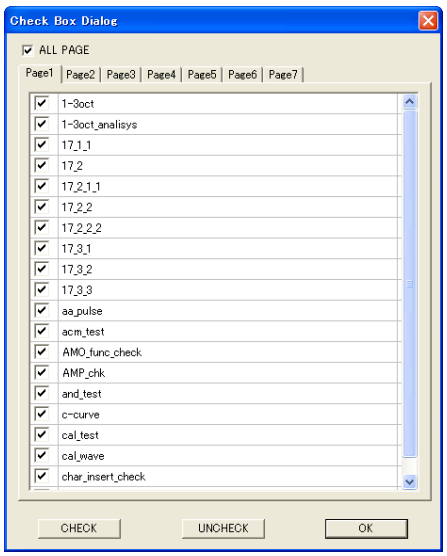
```
1          /*------param4.prc-----------*/
2          def ch_name &1 "演算項目"
3          def ch_name $1 "選択ステータス"
4          def ch_name $2 "選択項目数"
5          assign &1 ="高低左","高低右","通り左","通り右","水準","軌間","平面性"
6          $1 = DAG(ELM(&1),1)   /* 選択初期値全て選択設定*/
7          get check_box_status &1 $1 "演算項目を選択して下さい"
8          $2 = SUM($1)    /* 選択項目数取得*/
9          case $2 > 0
10           proc recon{     /* 選択した項目のみで演算項目を再構成*/
11             &1 = CREC(0,$1,&1)
12           }recon
```

<Archi_1 Script 記述構文の説明>

いくつかの項目を唯一無二でなく選択する場合に使用します。
5 行目:　選択する項目を生成します。6　行目はチェックボックスステータスの初期値を生成します。チェックボックスステータスの要素数は項目数に一致させます。
7 行目:　チェックボックスステータス取得文:get check_box_status で、本文が実行されるとダイアログが表示されます。ダイアログで項目ごとのチェックボックスを選択します。ダイアログ~の戻り値は項目に対応したチェックの場合1、アンチェックの場合 0 が $1 の項目に対応した要素に戻ります。
8 行目:　選択された項目数を取得します。
11 行目:　項目をチェックされた項目だけに縮退します。

<補足説明>
ダイヤログは 1 ページ当たり 20 項目表示できます。表示する項目がそれ以上存在する場合、自動的にページ数が増加します。 下図、左側は記述例 8.4.で表示されるダイヤログを示し、右側は表示項目が増加した場合を示します。





ダイヤログ下側、「CHECK」ボタンをクリックすると、全ての項目がチェックされ、「UNCHECK」ボタンをクリックすると、全ての項目がアンチェックされます。

記述例8. 5. 諸元表ファイルから選択する

<Archi_1 Script 記述例>

```
1        /*-------param5.prc-------------*/
2        read file_info &4 &5 &5 $1 "prm"
3        $2 = 0
4        get radio_button_status &4 $2 "使用する諸元表を選択して下さい"
5        def file_id %1 &4($2) prm
6        get param %1 &1 &2,&3
```

<Archi_1 Script 記述構文の説明>
予め作成した諸元表ファイルから列を選択してパラメータを一括して受け取る場合に使用します。
2 行目： 諸元表格納先フォルダから諸元表ファイルの拡張子”prm”だけのファイル名を取得します。
4 行目： 諸元ファイルをラジオボタンで選択します。
5 行目： 選択された諸元表ファイルのファイル番号を定義します。
6 行目： 諸元表ファイルから選択した列の項目内容を文字列として&1 に格納し、諸元名を&2 に単位を&3 に格納します。

<補足説明> 諸元表ファイルの拡張子は特に規定していません。上記例では拡張子は”.prm”あることを前提としています。例えばファイル名が諸元 表.csv の場合は、5 行目のファイル番号定義文のファイル名には拡張子を付ける必要があります。
その場合、def file_id %1 “諸元表.csv” prm と記述します。(ファイル番号定義文のファイル属性は必ず prm とします)
6 行目の諸元表ファイル参照文:get param が実行されると諸元表ダイアログが表示されます。ダイアログから取得する諸元値列をクリックして選択します。選択した列で取得不要な項目がある場合はダイアログ左側チェックボックスをアンチェックすると、アンチェックされた項目は取得しません。

なお、表示されるダイヤログから諸元表を簡易編集することができます。編集したい諸元値項目をダブルクリックすると入力欄に変わり、

キーボードから入力できます。また、選択されている諸元値項目を右クリックすると編集 tool が表示されます。編集 tool の項目は Copy、Paste、InsertRow の 3 種です。InsertRow は新規列の追加機能で、クリックすると諸元表最右側に列が追加されます。編集した結果をファイル更新する場合は、ダイヤログ下側の「SAVE」ボタンをクリックします。

<諸元表ファイルのフォーマット>
諸元表ファイルフォーマットは項目区切り半角カンマのテキスト形式です。

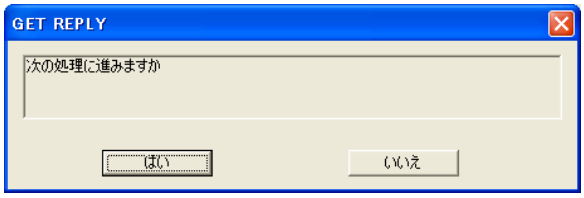
1 行目： Memo 行です。
　1 列目：”MEMO” 半角キーワード
　2 列目：任意文字列で諸元表のメモを記述します。
2 行目：ヘッダー行です。
　1 列目：”NAME” 半角キーワード
　2 列目：”UNIT” 半角キーワード
　3 列目以降：半角カンマ区切りで列ごとの諸元値名称、例えば機種名等を記述（特に付けなくても良い）します。
3 行目以降：諸元値行となります。
　1 列目：諸元値名
　2 列目：単位
　3 列目以降：各諸元値

記述例8. 6. 二値選択する

<Archi_1 Script 記述例>

```
1       /*------param6.prc-----------*/
2       def ch_name $1 "応答ステータス"
3       get reply "次の処理に進みますか" "はい" "いいえ" $1
```

<Archi_1 Script 記述構文の説明>
する/しない、yes/no などの二値選択する場合に使用します。
3 行目：　メッセージ表示応答文：get reply で、メッセージ文、左ボタンラベル、右ボタンラベルを記述します。 本文が実行されるとダイアログが表示されます。左ボタンがクリックされた場合は$1 に1が、右ボタンがクリックされた場合は 0 が戻ります。



記述例8. 7. 解析対象ファイルのヘッダーファイル全体を表示してファイルを選択する

フォルダを選択し、フォルダ内に格納されているヘッダーファイルを全て読み出してテキスト表示し、表示されたヘッダーファイルからヘッダーファイルを選択します。

<Archi_1 Script 記述例>

```
1       /*------param7.prc-----------*/
2       get folder_select
3       read file_info &1 &2 &3 $1 .hdr
4       $2 = 0
5       repeat_case $2 < $1
6         proc header_read {
7           assign &4 = &1($2)|".hdr"
8           def file_id %1 &4 chr
9           $4 = $2+100
10          read cell %1 1,1 1 &($4)
11          $2 = $2+1
12        }header_read
13        $3 = 1
14      get text_page &[100,$1] $3
15      $3 = $3-1
16      disp message &1($3)|"が選択されました。"
17      end
```

<Archi_1 Script 記述構文の説明>

2 行目： フォルダ選択します。

3 行目： 選択されたフォルダに格納されている拡張子.hdr のファイル情報を読み出します。&1 に拡張子無しファイル名が、$1 にファイル個数が格納されます。

5 行目： ファイル個数分ヘッダーファイルを読み出す演算処理ブロックの制御をします。

6 行目～12 行目はヘッダーファイル読み出し演算処理ブロックです。

7 行目： 7 行目はファイル名に拡張子を付加します。

8 行目： ファイル番号を属性 chr(く区切り文字無視)で定義します。

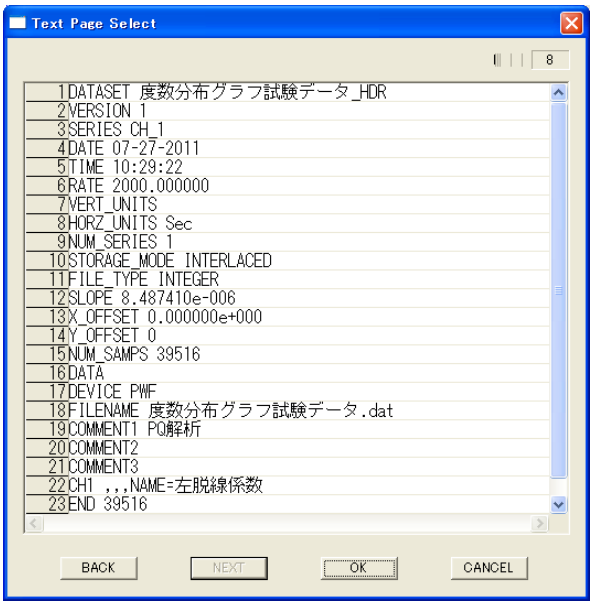
9 行目： ヘッダーファイル(テキスト)格納先チャネル番号を生成します。

10 行目: ヘッダーファイルを読み出します。属性 chr の場合 Cell は 1 個しか存在しません。したがって、読み出し開始行列番号は 1,1 となり、読み出し方向は意味を持ちません。また、格納先チャネルは間接指定したチャネルに格納されます。

11 行目: 繰り返しカウンタをインクリメントします。

13 行目: ページ選択初期値を設定します。初期値設定しない場合は、14 行目で表示されるダイアログをキャンセルした場合に$3 は null となり実行時エラーとなることを防止します。

14 行目: 読み出したヘッダーファイルをテキスト表示します。表示するチャネルは$100 からファイル個数分連続指定します。本文が実行されるとテキスト表示ダイアログが表示され、ページを選択できます。ページとは表示するチャネル列に対応し、先頭が 1 から振られ選択したページ番号が$3 に戻ります。



<ダイアログの操作> ページの選択はダイアログ下部の「NEXT」ボタンで次のページ、「BACK」で前のページを表示します。「OK」ボタンをクリックすると現在 表示されているページ番号が戻り、「CANCEL」ボタンをクリックするとページ番号は初期値のまま戻ります。つまり上記例では$3 に代入 した初期値のままとなります。なお、ダイアログのサイズ変更はダイアログ枠をドラッグしてサイズ変更できます。

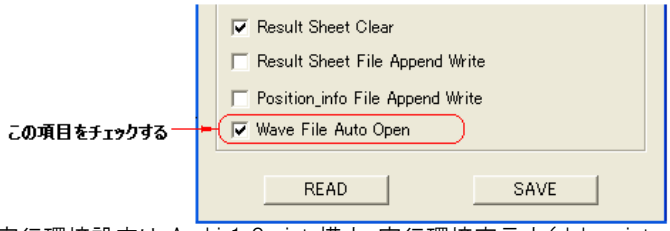
<補足説明> 上記例では解析対象ファイルのヘッダーファイルを読み出して表示していますが、ヘッダーファイルの読み出しは専用のヘッダーファイ ル読み出し文があります。参考までにヘッダーファイル読み出し文で読み出す場合を記述します。

変更部分は記述例の 6 行目～12 行目の演算処理ブロック header_read を次の様に変更します。

```
proc header_read {
   def file_id %1 &1($2) wav        ⇒ファイル情報取得文で取得したファイル名を指定し、属性を wav として定義します。
   read wave %1 0,0                 ⇒収録ファイル読み出し文で 0,0としてヘッダーファイルのみ読み出します
   $4 = $2+100
   read header_info &($4) $5        ⇒ヘッダーファイル読み出し文で読み出します。
   $2 = $2+1
}header_read
```

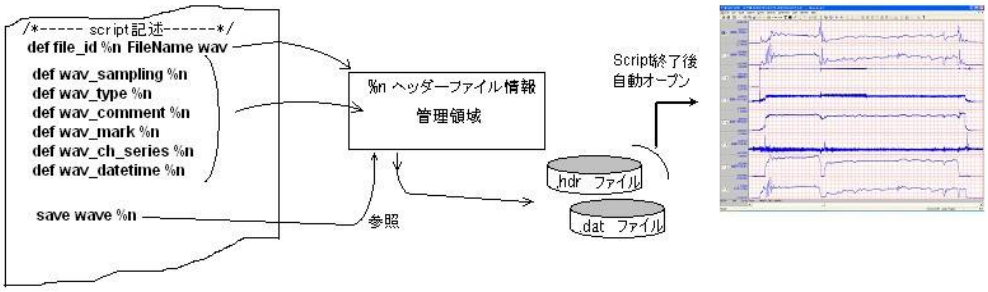
9. 演算結果波形ファイルを生成する

波形ファルとは、解析対象ファイルと同じ形式(PcWaveForm 標準フォーマット)で生成格納するファイルを意味します。演算した結果を解析対象ファイルと同じ形式で格納することができます。
格納したファイルは、Archi_1 Script 実行メニューダイアログの実行環境設定項目の Wave File Auto Open がチェックされていると、実行終了後、生成した波形ファイルを自動的に Open し波形表示します。



実行環境設定は Archi_1 Script 構文:実行環境宣言文(dcl script_env 文)でも行えます。演算結果波形ファイルを生成する操作を

模式的に示します。



9. 1. 波形ファイル生成事前処理 生成

波形ファイル番号定義文 文

法:

def file_id %ファイル番号 ファイル名 属性

属性は wav とします。ファイル番号定義文は、書き込むファイルを割り当て、ヘッダーファイルへの書き込み情報を管理します。なお、ファイル名は解析対象ファイルと同じ名前を付けることはできません。解析対象ファイル名は排他的に管理されている為です。

9. 2. 波形ファイル格納操作

ファイル格納文

文法:

save wave %ファイル番号 格納チャネル列

ファイル格納文が実行されると定義されたファイル名に格納されます。なお、同じファイル番号で同じファイル名にファイル格納文を複数実行すると上書きされます。

記述例:

収録条件を初期値のまま、最も簡単に波形ファイル生成する場合

```
def file_id %1 “test_data” wav          /* ファイル番号定義*/
save wave %1 $[1,10],#1,#5              /* 波形ファイル格納*/
```

波形ファイルには、$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,#1,#5 の 12 チャネルが格納されます。サンプリング周期は現在設定されているサンプリング周期となり、データ形式は 2byte 符号付き整数形式で、チャネル番号は記述順に 1～12 と振られます。なお、信号名及び単位は格納するチャネルに付けられている信号名および単位がそのまま付けられます。又、収録開始年月日及び収録開始時刻は、ファイル作成時の内部カレンダクロックが示す年月日、時刻が付けられます。 記述した格納チャネルの要素数が異なる場合は、最も要素数の少ないチャネルに整合され、余りは格納されません。但し、 要素数は 1 個のチャネルを含む場合は実行時 Error となります。

9. 3. 波形ファイル収録条件の設定操作

現在解析対象ファイルと異なった収録条件として格納する場合に記述します。

9. 3. 1. サンプリング周期と単位の設定 サンプリング周期定義を行わない場合は、現在解析対象ファイルと同じサンプリング周期が設定されます。解析対象ファイル と或いは現在のサンプリング周期と異なったサンプリング周期を設定して波形ファイルを生成する場合に記述します。 **生成波形ファイル・サンプリング値定義文**

文法:

def wav_sampling %ファイル番号 フラグ 周期 単位

記述例:

サンプリング周期を定義して波形ファイルを生成する場合

```
def file_id %1 “test_data” wav          /* ファイル番号定義*/
def wav_sampling %1 0 1000 “sec”        /* サンプリング周期定義*/
save wave %1 $[1,10],#1,#5              /* 波形ファイル格納*/
```

生成する波形ファイルのサンプリング周波数を 1kHz に設定します。フラグ 0 は記述する周期項目を周波数で記述、フラグ1は周期項目を周期で記述する事を意味します。なお、単位は必ず周期単位で記述します。

9. 3. 2. 格納データ形式の設定 生成する波形ファイルのデータ形式を明示的に指定する場合に使用します。本文を記述省略した場合、2byte 符号付き整 数(integer)で格納されます。

生成波形ファイル・データタイプ定義文

文法:

def wav_type %ファイル番号 データタイプ

記述例:

サンプリング周期と格納データ形式を 4byte 浮動小数点形式と定義して波形ファイルを格納する場合

```
def file_id %1 “test_data” wav          /* ファイル番号定義*/
def wav_sampling %1 0 1000 “sec”        /* サンプリング周波数定義*/
def wav_type %1 float                   /* データ形式定義*/
save wave %1 $[1,10],#1,#5              /* 波形ファイル格納*/
```

生成する波形ファイルのデータ型式を 4byte 浮動小数点形式に設定します。

※ データタイプは Keyword で、**integer、long、float、double** から選択記述します。Integer は 2byte 符号付き整数形式(int16)、long は 4byte 符号付き整数形式(int32)、flaot は 4byte 浮動小数点形式(float32)、double は 8byte 浮動小数点形式(float64)を意味します。解析対象ファイルと同じデータ形式で格納する場合は 、read header_info 文で”FILE_TYPE ”行を指定してヘッダーファイルに記述されているデータ形式を読み出して、同じデータ形式を指定します。

記述例:

解析対象ファイルと同じデータ形式を設定する場合

```
read header_info &1 $1 “FILE_TYPE “     /* &1 にデータ形式を取得する。$1 は行数を意味しダミーです*/
def file_id %1 “test_data” wav          /* ファイル番号定義*/
def wav_type %1 &1                      /* データ形式定義*/
save wave %1 $[1,10],#1,#5              /* 波形ファイル格納*/
```

9. 3. 3. コメントの書き込み

生成する波形ファイルにコメントを書き込む場合に使用します。

生成波形ファイル・コメント定義文

文法:

def wav_comment %ファイル番号 コメント番号 コメント

記述例:

サンプリング初期値で、コメント1行目を定義してデータ形式 flaot で格納する場合

```
def file_id %1 “test_data” wav          /* ファイル番号定義*/
def wav_comment %1 1 “試験データです”   /* コメント定義*/
def wav_type %1 float                   /* データ形式定義*/
save wave %1 $[1,10],#1,#5              /* 波形ファイル格納*/
```

波形ファイルのヘッダーファイルのコメント行1に“試験データです”を書き込みます。

※ コメント番号はコメント行を意味し、1～3 となります。なお、0 はコメント 1～3 行を一度に定義する場合に記述します。その場合、記述するコメント項目は文字属性参照チャネルで記述し予め要素にコメント行に対応するコメント文が格納されている必要があります。本文を記述省略した場合、ヘッダーファイルのコメント行は空欄となります。

9. 3. 4. マークの書き込み

生成する波形ファイルにマーク情報を書き込む場合に使用します。

生成波形ファイル・マーク情報定義文
文法:

```
def wav_mark %ファイル番号 マーク Index マークメモ
```

記述例:

```
def file_id %1 “test_data” wav              /* ファイル番号定義*/
def wav_type %1 float                       /* データ形式定義*/
def wav_mark %1 12345 “mark No.1”    /* Index12345 にマークメモ markNo.1 としてマークを定義*/
def wav_mark %1 13456 “mark No.2”   /* Index13456 にマークメモ markNo.2 としてマークを定義*/
save wave %1 $[1,10],#1,#5                  /* 波形ファイル格納*/
```

Index12345 にマークメモ”mark No.1”を、Index13456 にマークメモ”mark No.2”を付けます。
※ マーク情報定義文は複数行記述することができます。一度に定義する場合はマーク Index を数値属性参照チャネル、マークメモを文字属性参照チャネルで記述し、予め要素に格納して置く必要があります。なお、記述したマーク位置 Index が当該ファイルに存在しない場合実行時エラーとなります。 本文を記述省略した場合、ヘッダーファイルにマーク行は生成されません。

9. 3. 5. チャネル番号の明示的設定 生成する波形ファイルに格納するチャネルのチャネル番号を明示的に設定する場合に使用します。本文を記述省略した場 合、格納文で記述したチャネル列記述順にch1から振られます。
生成波形ファイル・チャネル番号定義文
文法:

```
def wav_ch_series %ファイル番号 チャネル番号
```

記述例:
格納するチャネル番号を明示的に定義し、その他の条件は初期値のままとする場合

```
assign $1 = 1,3,5,7,9                      /* 格納順にチャネル番号を用意する*/
def file_id %1 “test_data” wav              /* ファイル番号定義*/
def wav_ch_series %1 $1                     /* 格納順にチャネル番号を明示的に定義*/
save wave %1 $1,$2,$3,$4,$5                 /* 波形ファイル格納*/
```

格納する$1,$2,$3,$4,$5 順にチャネル番号を明示的に ch1,ch3,ch7,ch9 と設定します。
※ 生成される波形ファイルのチャネル番号はch1、ch3、ch5、ch7、ch9となります。なお、格納するチャネル番号とここで定義したチャネル番号は直接処理に関係なく、生成する波形ファイル上で振られるチャネル番号を意味しています。

9. 3. 6. 収録開始年月日時刻の設定
生成する波形ファイルのヘッダーファイルに書き込まれる開始年月日、時刻及び X 軸オフセット値を明示的に設定する場合に使用します。本文を記述省略した場合、ヘッダーファイルの DATE 行/TIME 行は、格納時の年月日時刻となり、X 軸オフセット値は 0 となります。なお、オフセット値の単位は現在設定されているサンプリング周期の単位となります。 **生成波形ファイル・開始日付時刻定義文**
文法:

```
def wav_datetime %ファイル番号  年月日 時刻 オフセット値
```

記述例:
生成年月日、時刻以外は初期値として格納する場合

```
def file_id %1 “test_data” wav              /* ファイル番号定義*/
def wav_datetime %1 “09-25-2010” “16:27:32” 1.5 /* 生成年月日、時刻、オフセットを定義*/
save wave %1 $1,$2,$3,$4,$5                 /* 波形ファイル格納*/
```

オフセット値は記述省略できます。省略した場合は 0 と見なします。

解析対象ファイルの収録開始年月日及び時刻、X 軸 Offfset 値は、収録情報として意味を持つ場合があります。解析対象ファイルと同じとして波形ファイル生成する場合は、収録開始年月日取得関数、収録開始時刻取得関数及び X 軸オフセット値取得関数を使用して解析対象ファルから取得して定義します。

記述例:
生成年月日、時刻及びオフセット値を解析対象ファイルと同じとしてその他は初期値で格納する場合

```
&1 “開始年月日:” = CSDT()
&2 “開始時刻:” = CSTM()
$1 “X_offset:” = XOF()
def file_id %1 “test_data” wav              /* ファイル番号定義*/
def wav_datetime %1 &1 &2 $1                /* /* 生成年月日、時刻、オフセットを定義*/
def wav_type %1 float                       /* データ形式定義*/
save wave %1 $[10,5]                        /* 波形ファイル格納*/
```

格納される波形データチャネルは$10～$15 となりチャネル番号は ch1、ch2、ch3、ch4、ch5 となります。

記述例9. 1. 最も記述を省略して解析結果波形ファイルを生成格納する

解析対象ファルの ch1～ch4 を別の波形ファイルとして生成格納します。

<Archi_1 Script 記述例>

```
1        /*--- wave_file_save1.prc ----------------------------*/
2        def file_id %1 "test" wav
3        save wave %1 #1,#2,#3,#4
```

<Archi_1 Script 記述構文の説明>

PcWaveForm 上で解析対象ファイルを読み出し、解析範囲を指定してから実行し、解析範囲の収録チャネル ch1～ch4 をコピーした波形ファイルをファイル名"test"でカレントフォルダに格納します。
2 行目： ファイル番号定義文で格納ファイル名："test"とし、属性 wav として定義します。
3 行目： 波形ファイル格納文でチャネル列：#1,#2,#3,#4(収録チャネル 1～4)として格納します。
※ 生成される波形ファイルのサンプリング周波数は現在のサンプリング周波数および単位となります。解析対象ファイルを読み出していない場合は生成ファイルサンプリング定義文、またはサンプリング定義文で定義が必要です。

<補足説明>

演算結果波形ファイルの生成格納に際して、記述必須構文はファイル番号定義文：def file_id と波形ファイル格納文:save wave 文の 2 種となります。
解析対象ファイル読出しに必要な構文と対比すると

解析対象ファイル読出しの場合		演算結果波形ファイル格納の場合
def file_id %n file_name wav	⇒	def file_id %n file_name wav
read wave %n	⇒	save wave %n 格納チャネル列

となります。波形ファイル読み出し文は読み出すチャネルを指定する必要はなく、収録されているすべてのチャネルを読み出しますが、波形ファイル格納文は格納するチャネルを指定する必要があります。 記述必須構文のみで格納した場合、生成されるヘッダーファイルは次の様になります。

① ファイルに格納されるデータ個数は、格納するチャネルの中で最もデータ個数の少ない数に整合されます。
② ヘッダーファイルに記述されるサンプリング周波数は、現在設定されているサンプリング周波数がそのまま記述されます。
③ ヘッダーファイルに記述される収録開始年月日/時刻はファイル格納時の年月日/時刻が記述され、また、X 軸オフセット値は 0 となります。
④ ヘッダーファイルに記述される信号名、単位は記述された格納チャネル列に付けられている信号名/単位が記述されます。
⑤ ヘッダーファイルに記述されていたマークおよびコメントは記述されません。
⑥ 格納されるデータ形式は 2byte 符号付き整数となります。なお、当該チャネルの絶対値最大を 30000 として整数変換されます。同じファイル番号に save wave を繰り返すと、直前に設定された定義文に従って上書きされ、追記処理はできません。

記述例9. 2. 収録開始年月日および時刻を解析対象ファイルと同じにして生成格納する

生成される波形ファイルのヘッダーファイルに書き込まれる収録開始年月日および時刻は定義省略すると波形ファイル格納時の年月日、時刻となります。生成格納するファイルの収録開始年月日と時刻を解析対象ファイルと同じとして生成格納します。

<Archi_1 Script 記述例>

```
1        /*--- wave_file_save2.prc ----------------------------*/
2        def ch_name &1 "収録開始年月日"
3        def ch_name &2 "収録開始時刻"
4        def ch_name $1 "X 軸オフセット値"
6        &1 = CSDT()   /* 収録開始年月日取得*/
7        &2 = CSTM()   /* 収録開始時刻取得*/
8        $1 = XOF()    /* X 軸オフセット値取得*/
10       def file_id %1 "test" wav
11       def wav_datetime %1 &1 &2 $1 /* 収録開始年月日/時刻/X 軸オフセット定義*/
12       save wave %1 #1,#2,#3,#4
```

<Archi_1 Script 記述構文の説明>

6 行目： 解析対象ファイルの収録開始年月日を取得します。
7 行目： 解析対象ファイルの収録開始時刻を取得します。
8 行目： 解析対象ファイルの X 軸オフセット値を取得します。
11 行目： 生成波形ファイル開始日付時刻定義文で収録開始年月日,時刻および X 軸オフセット値を定義します。
12 行目： 波形ファイル格納文でチャネル列：#1,#2,#3,#4(収録チャネル 1～4)として格納します。

記述例9. 3. 格納生成するチャネル番号を解析対象ファイルと同じ番号として格納する

生成される波形ファイルのヘッダーファイルに書き込まれるチャネル番号は定義省略すると 1ch から自動的に振られます。また、格納データ形式を定義省略すると 2Byte 整数形式となります。
生成格納するファイルの各チャネルを 10Hz ローパスフィルタ処理し、格納チャネル番号を収録チャネルと同じにし、格納データ形式を8byte 浮動小数点形式として生成格納します。

<Archi_1 Script 記述例>

```
1       /*--- wave_file_save3.prc ----------------------------*/
2       def ch_name $1 "収録チャネル数"
3       def ch_name $2 "収録チャネル番号"
4       def ch_name &1 "収録チャネル信号名"
5       def ch_name &2 "収録チャネル単位名"
6       def ch_name &3 "解析結果波形ファイル名"
7       def ch_name $3 "ch_counter"
8       def ch_name $4 "生成チャネル番号"
10      $1 = NCH()        /* 解析対象ファイル収録チャネル数取得*/
11      $2 = CHS()        /* 解析対象ファイル収録チャネル番号取得*/
12      &3 = CFNM()             /* 解析対象ファイル収録ファイル名取得*/
13      assign &3 = &3|"_10HzLPF" /* 解析結果ファイル名生成*/
14      $4 = 100                /*格納先チャネル番号先頭*/
15      read ch_name &1 &2      /* 解析対象ファイル収録チャネル信号名/単位取得*/
16      def ch_name $($4) &1   /*格納チャネル信号名定義*/
17      def ch_unit $($4) &2   /*格納チャネル単位名定義*/
19      $3 = 0
20      repeat_case $3 < $1
21      proc calc {
22          $($4+$3) = LPF(10,#($2($3))) /*収録チャネル 10HzLPF 処理*/
23          $3 = $3+1     /* ch_counter UP */
24      }calc
26      def file_id %1 &3 wav
27      def wav_type %1 double /* 格納データ形式を倍精度浮動小数点形式に定義*/
28      def wav_ch_series %1 $2 /* 格納チャネル番号定義*/
29      save wave %1 $[100,$1]
30      end
```

<Archi_1 Script 記述構文の説明>

10 行目： 解析対象ファイルのチャネル数を取得します。
11 行目： 解析対象ファイルのチャネル番号を取得します。
12 行目： 解析対象ファイルのファイル名を取得します。
13 行目： 生成格納するファイル名を解析対象ファイルの末尾に"_10HzLPF"を付加して生成します。
14 行目： チャネルごとの演算結果格納先チャネル番号を生成します。
15 行目： 解析対象ファイルに付けられている信号名と単位を取得します。
16 行目： 格納先チャネルに収録チャネルと同じ信号名を定義します。
17 行目： 格納先チャネルに収録チャネルと同じ単位を格定義します。
20 行目： 直下の演算処理ブロック calc を収録チャネル数分繰り返し制御します。
21 行目～24 行目：演算処理ブロックで収録チャネルに 10Hz ローパスフィルタ処理を行います。
26 行目: ファイル番号定義文でファイル名：&3(14 行目で生成したファイル名)、属性 wav として定義します。
27 行目： 生成波形ファイルデータタイプ定義文でデータ形式を倍精度浮動小数点形式に定義します。
28 行目： 生成波形ファイルチャネル番号定義文でチャネル並びを解析対象ファイルと同じに定義します。
29 行目： 波形ファイル格納文で格納チャネルを$100 から収録チャネル数分格納します。

<補足説明> 解析対象ファイルのチャネル構成が不明であることを前提として、収録チャネルを間接指定すると同時に、演算結果格納チャネルも間 接指定して処理します。22 行目の左辺$($4($3))は$4 が 14 行目で 100 から 101,102…とチャネル数分の格納チャネル番号を表す 数列が生成されており、そのIndex を$3 で示しています。つまり、$3 が 0 の時は$4(0)を指し、$4(0)は 100 ですので#($4($3))は$100 となります。また、右辺の LPF 関数の引数#($2($3))は、11 行目で$2 に解析対象ファイルのチャネル構成をCHS()関数で取得してい ますので、$3 が 0 の時は#($2(0))を意味し、$2(0)は先頭収録収録チャネル番号となりますので、例えば先頭収録チャネル番号が 1 の場合は#($2($3))は#1 を指定した事になります。したがって、$3 をチャネルカウンタとして収録チャネル分繰り返しLOOP を形成すれ ば、解析対象ファイルのすべてのチャネルを 10Hz ローパスフィルタ処理した結果を得る事になります。 例えば解析対象ファイルの収録チャネル番号を CHS()関数で取得した結果が 1,3,4 と 3 チャネル収録の場合は

```
$100 = LPF(10,#1)
```

```
$101 = LPF(10,#3)
$102 = LPF(10,#4)
```
と記述した事と等価となります。

記述例9. 4. 生成格納する波形ファイルコメントを書き込み、最大値位置にマークを付けて格納する

PcWaveForm の波形表示 Window で解析範囲に指定された各チャネルを 10HzLPF 処理後、最大値位置にマークを付け生成格納します。

<Archi_1 Script 記述例>

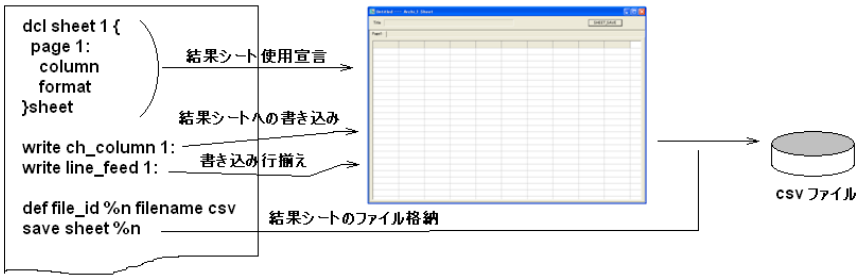
```
1   /*--- wave_file_save4.prc ----------------------------*/
2   def ch_name $1 "収録チャネル数"
3   def ch_name $2 "収録チャネル番号"
4   def ch_name &1 "収録チャネル信号名"
5   def ch_name &2 "収録チャネル単位名"
6   def ch_name &3 "解析結果波形ファイル名"
7   def ch_name &4 "マークメモ"
8   def ch_name $3 "ch_counter"
9   def ch_name $4 "生成チャネル番号"
10  def ch_name $5 "最大値マーク index"
12  $1 = NCH()        /* 解析対象ファイル収録チャネル数取得*/
13  $2 = CHS()        /* 解析対象ファイル収録チャネル番号取得*/
14  &3 = CFNM()             /* 解析対象ファイル収録ファイル名取得*/
15  assign &3 = &3|"_10HzLPF" /* 解析結果ファイル名生成*/
16  $4 = ACC(DAG($1,1))+99 /*格納先チャネル番号生成($100～)*/
17  read ch_name &1 &2     /* 解析対象ファイル収録チャネル信号名/単位取得*/
18  def ch_name $($4) &1   /*格納チャネル信号名定義*/
19  def ch_unit $($4) &2   /*格納チャネル単位名定義*/
20  assign $5 = $1<0>      /* マーク index 格納先確保*/
21  assign &4 = $1<" ">    /* マークメモ格納先確保*/
22  $3 = 0
23  repeat_case $3 < $1
24  proc calc {
25      $($4($3)) = LPF(10,#($2($3))) /*収録チャネル 10HzLPF 処理*/
26      $5($3) = MXP($($4($3)))         /* ch ごと最大値位置 Index 取得*/
27      assign &4($3) = "ch"|$2($3)(F0)|"_最大値" /* マークメモ生成*/
28      $3 = $3+1    /* ch_counter UP */
29  }calc
31  def file_id %1 &3 wav
32  def wav_comment %1 1 "mark test"
33  def wav_mark %1 $5 &4    /* マーク定義*/
34  def wav_type %1 double /* 格納データ形式を倍精度浮動小数点形式に定義*/
35  def wav_ch_series %1 $2 /* 格納チャネル番号を収録ファイルと同じに定義*/
36  save wave %1 $[100,$1]  /* 格納チャネル番号は$100 からチャネル数分格納*/
37  end
```

<Archi_1 Script 記述構文の説明>

12 行目:解析対象ファイルのチャネル数を取得します。
13 行目:解析対象ファイルのチャネル番号を取得します。
14 行目:解析対象ファイル名を取得します。
15 行目:生成格納するファイル名を解析対象ファイルの末尾に"_10HzLPF"を付加して生成します。
16 行目:チャネルごとの演算結果格納先チャネル番号を生成します。
17 行目:解析対象ファイルに付けられている信号名と単位を取得します。
18 行目:格納先チャネルに収録チャネルと同じ信号名を定義します。
19 行目:格納先チャネルに収録チャネルと同じ単位を格定義します。
20 行目:マーク Index 格納用にチャネル数分準備します。
21 行目:マークメモ格納用にチャネル数分準備します。
23 行目:直下の演算処理ブロック calc を収録チャネル数分繰り返し制御します。
26 行目:MPX 関数を使用して各チャネルの最大値位置 Index を取得します。
27 行目:マークメモを生成します。マークメモは"ch 番号_最大値"とします。
33 行目:生成波形ファイル･マーク情報定義文でマーク位置とマークメモを定義します。
31 行目:生成波形ファイル･コメント定義文でコメント行 1 行目に"mark_test"と定義します。

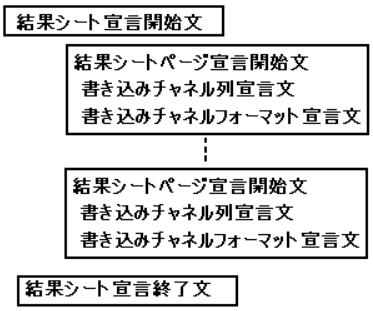
10. 結果シートを使用する

結果シート Window は、Archi_1 Script 解析した結果を数値表示する為の専用 Window です。結果シートは列ごとに書き込むチャネルを宣言し、宣言された列は記述したチャネル専用となり、他のチャネルを書き込むことはできません。従って、即値は書き込めません。即値を書き込む場合は、一旦、参照チャネルに代入してから行います。
結果シート Window への書き込み操作概要を示します。



10. 1. 結果シート使用の事前処理 結果シートを使用する場合、結果シート宣言を行う必要があります。 結果シート宣言は複数の Script 構文からなる結果シート宣言ブロックで構成されます。

結果シート宣言ブロック開始文
文法:
dcl sheet ページ数 表題 {



記述例:
dcl sheet 1 "演算結果" { 結果シート宣言ブロックの先頭に記述し、Script 中で唯一無二となります。 なお、表題は記述省略可能です。省略した場合、後述する結果シート表 題定義文により、後でも定義する事ができます。

結果シートページ宣言開始文
文法:
page ページ番号: 表題 属性

記述例:
page 1: "統計値" ページ宣言する先頭行でページ番号は1から結果シート宣言ブロック開始文で記述したページ数まで記述できます。また、結 果シートページ宣言文はページ番号昇順で記述します。属性は予約語で、現状は 1 種類しか存在しない為記述しません。ペ ージ表題は結果シート Window の tab 部分に表示される内容で記述省略可です。省略した場合、後述する結果シート表題定 義文により、後でも定義可能です。結果シートページ宣言は当該行と後述する結果書き込み列宣言文及び結果シート書き込 み列フォーマット宣言文の 3 行から構成され、ページ数分宣言できます。

結果シート書き込みチャネル列宣言文
文法:
column 書き込みチャネル列

記述例:
column $1,&1,$3,$2,&4
結果シートページ宣言開始文直下に 1 行記述し,書き込むチャネル列を記述します。チャネルの記述順に当該ページの列が占有されます。ここで宣言したチャネル以外を書き込む事は出来ません。

結果シート書き込みフォーマット列宣言文
文法:
format フォーマット列 フラグ

記述例:
format F0,A,F3,S5,A 2
結果シート書き込みチャネル列宣言文の直下に 1 行記述し、書き込みチャネル列に対応した表示形式を記述します。フラグは記述省略可能ですが、省略した場合はページの先頭列が memo 列となり、最終列に解析対象ファイル名が自動的に書き込まれます。

結果シート宣言ブロック終了宣言文
文法:
}sheet

記述例:
結果シート 1 ページ構成で、書き込みチャネル$1,$2,$3 を固定小数点下 3 桁で宣言する場合

```
dcl sheet 1 "result" {          /* 結果シートブロックの開始、結果シート表題"result"指定、ページ数 1*/
  page 1:                       /* ページ表題,ページ属性は記述省略しています*/
  column $1,$2,$3,$4,$5         /* 書き込みチャネルを記述*/ format
  F3,F3,F3 2                    /* 何れも固定小数点下 3 桁で指定*/
}sheet                          /* 結果シート宣言ブロックの終了*/
```

10. 2. 結果シートへの書き込み操作

10. 2. 1. 結果列書き込み操作 結果シートチャネル列書き込み文 文法:

write ch_column　ページ番号: 書き込みチャネル列

結果シートへの書き込みはページ番号を指定して書き込みチャネルを記述します。チャネルの記述順と書き込まれる列位置(column)とは関係なく、結果シート宣言ブロックで宣言したチャネル位置に行方向に要素数分書き込まれます。なお、文字属性参照チャネルを書き込む場合に文字列に半角カンマを含むと半角カンマを認識し隣接列に書き込みが行われますので注意が必要です。

記述例:
結果シートの 1 ページ目に$1,$2$3 の 3ch を書き込む場合

```
write ch_column 1: $1,$2,$3
```

結果シートには$1,$2,$3 の各列に数列の要素数分書き込まれます。下記表示例は$1 が要素数 3、$2 が要素数 2、$3 が要素数 4 の場合を示し、結果シート宣言文で$1～$5 が宣言されている場合を示します。

信号名(単位)行→	$1列	$2列	$3列	($4列予約)	($5列予約)		
	$1 (0)	$2 (0)	$3 (0)				
	$1 (1)	$2 (1)	$3 (1)				
	$1 (2)		$3 (2)				
			$3 (3)				

10. 2. 2. シートの行揃え操作

結果シート行揃え文
文法:

write line_feed ページ番号:　行数

結果シート行揃え文は、書き込み列ごとに書き込み行を揃えたい場合に記述します。実行されると書き込み列ごとに存在する行カウンタの最も大きな値に揃えられます。

記述例:
結果シートの書き込み行を1列開ける場合

```
write line_feed 1: 1
```

10. 2. 3. メモ列への書き込み操作

結果シートメモ列書き込み文
文法:

write memo_column ページ番号　書き込み文字列

メモ列はページごとに宣言される結果シート書き込みフォーマット列宣言文で記述されたフラグが 0 または 3 の時に生成され、メモ列への書き込みが有効となります。メモ列は 1 列目に生成されます。メモ列への書き込みは、他のチャネル列と異なり、現在最も大きな値を持つ行カウンタ+1 行に書き込まれ、すべての行カウンタがメモ列書き込み終了後の行+1 に更新されます。

記述例:

```
write memo_column 1: "memo"
```

10. 2. 4. ページ表題欄への書き込み操作 結果シート表題は結果シート宣言文で記述できますが、処理内容により後で表題を付けたい場合など使用します。 **結果シート表題定義文**

文法:

def sheet_title ページ番号 表題

※ ページ番号 0 はシート全体の表題を意味します。

記述例:

```
def sheet_title 0: “実験結果”
def sheet title 1: “結果1”
```

10. 3. 結果シートの読み出し/格納操作

10. 3. 1. 結果シートファイルの読み出し操作

結果シートファイル読み出し文 文法:

read sheet %ファイル番号

本文に先立ってファイル番号定義文で属性 sht として定義が必要です。 読み出し対象ファイルが結果シート宣言ブロック内で宣言されているページ数を越えている場合、残りのページは読み出され ません。また、読み出し対象ファイルのページ内書き込みチャネル数と、結果シート宣言ブロック内で宣言した書き込みチャ ネル数が一致しない場合、当該ページは読み出されません。

記述例:

```
def file_id %1 “sheetNo_1” sht
read sheet %1
```

10. 3. 2. 結果シートのファイルへの格納

結果シートファイル格納文 文法:

save sheet %ファイル番号

本文に先立ってファイル番号定義文で属性 sht として定義が必要です。結果シートへの書き込み設定が Append(追記)設定の場合、すでに記録されている結果シートファイルと現在のページ内チャネル数が異なるページは、すでに記録されているペ ージの内容に追記されず、書き込む内容に更新されます。格納形式は項目区切り半角カンマのテキスト形式で拡張子.csv と なります。したがってファイル上にはページ概念はなく、ページ番号を記載して格納されます。 結果シートの格納に際して追記するか否かの設定は、実行環境宣言文あるいは、Archi_1 Script 実行メニュー下段のチェック ボックス 2 項目:Result Sheet Append Write をチェックまたはアンチェックで設定することもできます。

記述例:

```
def filre_id %1 “resut_sheet” sht
save sheet %1
```

10. 4. 結果シートの初期化

既に書き込んだ結果シートを初期化する場合に使用します。

結果シート初期化文

文法:

clr sheet ページ番号

ページ番号に0を指定すると結果シート全体を初期化します。なお、実行時に既に結果シートWindow が表示されている場合に本文により初期化することができます。言い換えると、実行時に結果シート Window が存在している場合、Script 中で宣言した書き込みチャネル数と一致したページは Script 中で結果シートに書き込むと追記されることを意味しています。 なお、結果シートを実行時に初期化するか否かの設定は、Archi_1 Script 実行メニュー下段のチェックボックス 1 項目:Result Sheet Clear をチェックまたはアンチェックで設定することもできます。

記述例:

```
clr sheet 1:
```

結果シートの 1 ページ目に書き込まれている内容を削除し初期化します。

記述例10. 1. 結果シートを宣言する

結果シート宣言ブロックを作成します。

<Archi_1 Script 記述例>

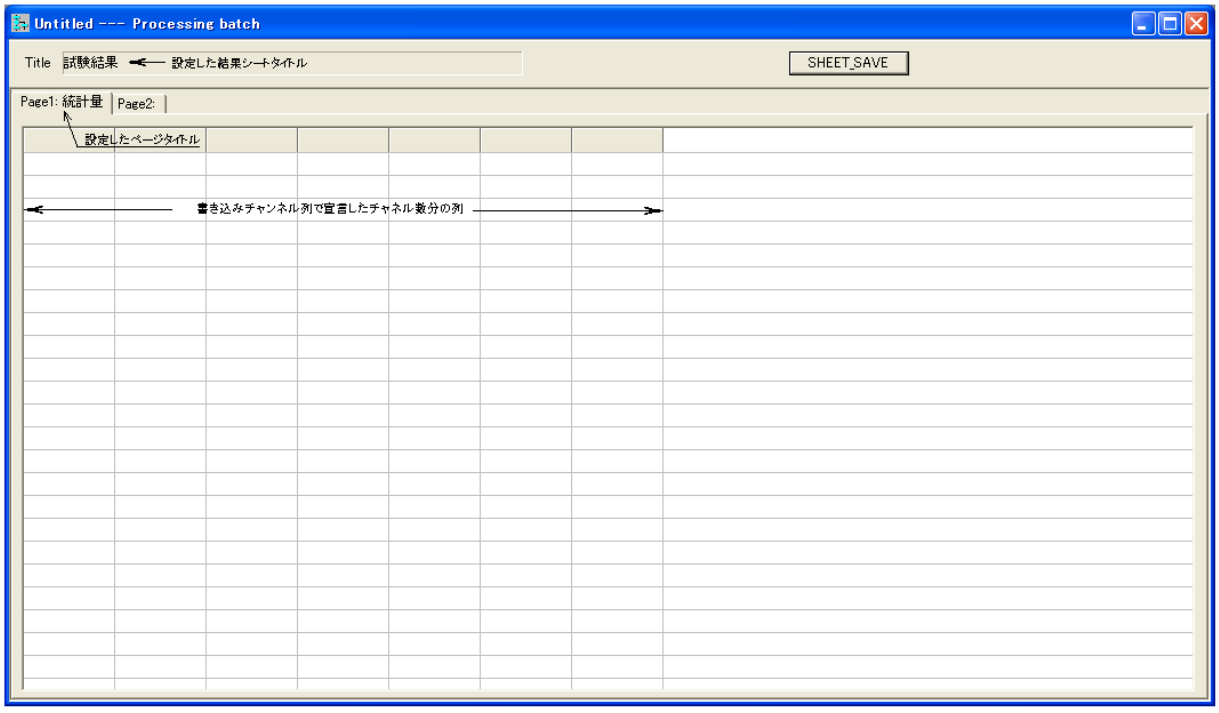
```
1          /*--- result_sheet1.prc-------------------------------------------------------------------*/
2          dcl sheet 2 "試験結果" {          /* 結果シート宣言の開始とページ数設定*/
3            page 1: "統計量"                  /* ページ宣言の開始とページ表題の設定*/
4              column $5,&6,&5,$6,$7,$8,$9     /* 列ごと書き込みチャネル宣言*/
5              format F3,A,A,F3,F3,F3,F3 2     /* 列ごと表示 Format 宣言*/
6          }sheet                              /*  結果シート宣言の終了*/5.1.2.  Archi_1記述構文の説明
```

<Archi_1 Script 記述構文の説明>

結果シートを使用する場合、Srchi_1 記述構文の先頭に結果シートの宣言を行います。結果シート宣言ブロックは結果シート宣言開始文、ページ文、書き込みチャネル列文、表示フォーマット文、および結果シート宣言終了文からなります。

2 行目：　結果シート宣言開始文で結果シートのタイトルおよび合計ページ数を宣言します。
3 行目：　ページ文でページタイトルおよび続く行が当該ページであることを宣言します。
4 行目：　書き込みチャネル列文で書き込むチャネルを列ごとに宣言します。
5 行目：　表示フォーマット文で書き込みチャネル列に対応したフォーマットを宣言します。
6 行目：　結果シート宣言終了文です。宣言が実行さ

れると次の様に結果シートが用意されます。



<補足説明>

結果シートの最上位行は、書き込まれるチャネルに付けられている信号名単位を結果シート書き込みごとに自動的に表示します。従って、書き込みごとに同じチャネル番号で信号名単位を変更した場合、最新の信号名単位が表示されます。 ページ数は宣言ブロック内でページ宣言するページ数より等しいか大きくなければなりません。

記述例10. 2. 結果シートにチャネルごとのデータ値を書き込む

結果シートに解析対象ファイルのデータ数値リストを作成します。

<Archi_1 Script 記述例>

```
1       /*--- result_sheet2.prc-----------------------------------------*/
2       dcl sheet 1 "試験結果" {              /* 結果シート宣言の開始とページ数設定*/
3         page 1: "データ数値リスト"          /* ページ宣言の開始とページ表題の設定*/
4           column $1,$2,$[100,64]            /* 列ごと書き込みチャネル宣言*/
5           format F0,F3,64(F4) 2             /* 列ごと表示 Format 宣言*/
6       }sheet                                /* 結果シート宣言の終了*/
7       /*----- 収録情報取得処理-----------------------------------------*/
8       $4 "収録チャネル数:" = NCH()          /* 収録チャネル数取得*/
9       $5 "収録チャネル番号:" = CHS()        /* 収録チャネル番号取得*/
10      &1 "収録信号名:" = CHNM(0)            /* 収録チャネル信号名取得*/
11      &2 "収録単位:" = CUNT(0)              /* 収録チャネル単位取得*/
12      assign &1 = &1|":"|&2
13      /*----- 書き込みチャネル代入処理---------------------------------*/
14      $3 "チャネルカウンタ:" = 0            /* チャネルカウンタ初期化*/
15      $6 "格納先チャネル番号:" = 100        /* 格納先チャネル番号初期化*/
16      repeat_case $3 < $4                   /* チャネルLOOP*/
17        proc data_read{
18          $($6) = #($5($3))                 /* 収録チャネル書き込みチャネル代入*/
19          def ch_name $($6) &1($3)          /* 書き込みチャネル信号名単位定義*/
20          $3 = $3+1                         /* チャネルカウンタUP*/
21          $6 = $6+1                         /* 格納先チャネル番号UP*/
22        }data_read
23      /*----- 結果シート書き込み処理-----------------------------------*/
24      $1 "データ番号:" = ACC(DAG(LEN($100),1))-1 /* データ番号生成*/
25      $2 "経過時間:sec" = SPB(0)            /* 時刻歴生成*/
26      write ch_column 1: $1,$2,$[100,$4]    /* 結果シート書き込み*/
27      end
```

<Archi_1 Script 記述構文の説明>

結果シートには、1 列目がデータ番号、2 列が時刻歴、3 列目以降が収録チャネルのデータ数値列となります。

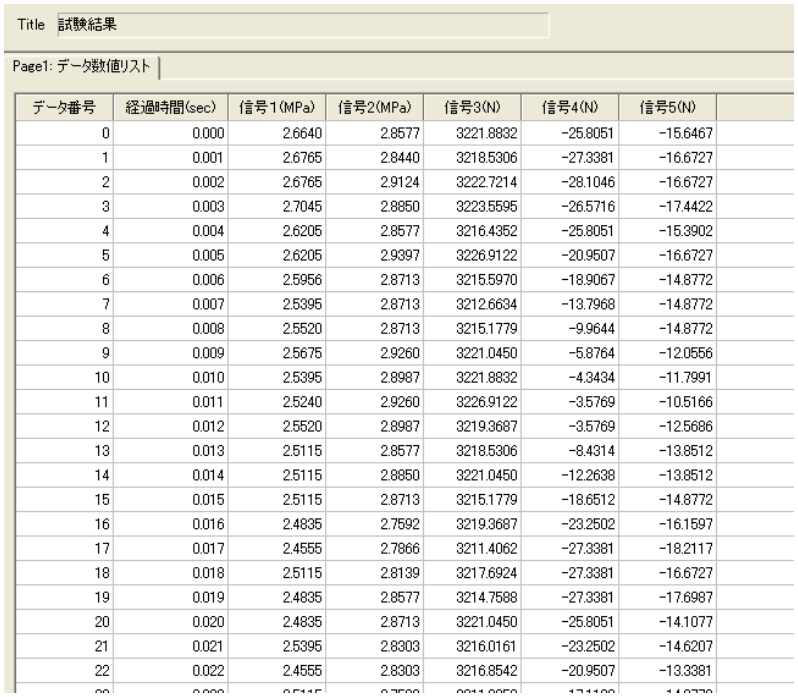
2 行目～6 行目: 結果シートを宣言します。

4 行目:　書き込みチャネルを宣言します。チャネル番号が不明な為、仮想チャネルを 100 から 64 チャネル分を用意します。

18 行目: 収録チャネルを読み出しチャネルごとに仮想チャネルに代入します。

26 行目: 結果シートに書き込みます。

<Script 実行結果:書き込まれた結果シート>



Title 試験結果

Page1: データ数値リスト

データ番号	経過時間(sec)	信号1(MPa)	信号2(MPa)	信号3(N)	信号4(N)	信号5(N)
0	0.000	2.6640	2.8577	3221.8832	-25.8051	-15.6467
1	0.001	2.6765	2.8440	3218.5306	-27.3381	-16.6727
2	0.002	2.6765	2.9124	3222.7214	-28.1046	-16.6727
3	0.003	2.7045	2.8850	3223.5595	-26.5716	-17.4422
4	0.004	2.6205	2.8577	3216.4352	-25.8051	-15.3902
5	0.005	2.6205	2.9397	3226.9122	-20.9507	-16.6727
6	0.006	2.5956	2.8713	3215.5970	-18.9067	-14.8772
7	0.007	2.5395	2.8713	3212.6634	-13.7968	-14.8772
8	0.008	2.5520	2.8713	3215.1779	-9.9644	-14.8772
9	0.009	2.5675	2.9260	3221.0450	-5.8764	-12.0556
10	0.010	2.5395	2.8987	3221.8832	-4.3434	-11.7991
11	0.011	2.5240	2.9260	3226.9122	-3.5769	-10.5166
12	0.012	2.5520	2.8987	3219.3687	-3.5769	-12.5686
13	0.013	2.5115	2.8577	3218.5306	-8.4314	-13.8512
14	0.014	2.5115	2.8850	3221.0450	-12.2638	-13.8512
15	0.015	2.5115	2.8713	3215.1779	-18.6512	-14.8772
16	0.016	2.4835	2.7592	3219.3687	-23.2502	-16.1597
17	0.017	2.4555	2.7866	3211.4062	-27.3381	-18.2117
18	0.018	2.5115	2.8139	3217.6924	-27.3381	-16.6727
19	0.019	2.4835	2.8577	3214.7588	-27.3381	-17.6987
20	0.020	2.4835	2.8713	3221.0450	-25.8051	-14.1077
21	0.021	2.5395	2.8303	3216.0161	-23.2502	-14.6207
22	0.022	2.4555	2.8303	3216.8542	-20.9507	-13.3381

記述例10. 3. フォルダ内に存在する収録データファイルリストを作成する

フォルダを選択し、フォルダ内に格納されている全ての解析対象ファイル(収録ファイル)の収録開始年月日、時刻、ファイル名、収録チャネル数、サンプリング周波数、収録チャネル番号および信号名/単位の収録ファイルリストを作成します。

<Archi_1 Script 記述例>

```
1         /*--- result_sheet3.prc ----------------------------*/
2         dcl sheet 1  {                          /* 結果シート宣言の開始とページ数設定*/
3           page 1:  "収録ファイル"               /* ページ宣言の開始とページ表題の設定*/
4             column &4,&5,&1,$4,$8,&9,$5,&6,&7   /* 列ごと書き込みチャネル宣言*/
5             format A,A,A,F0,F2,A,F0,A,A 2       /* 列ごと表示 Format 宣言*/
6         }sheet                                  /* 結果シート宣言の終了*/
7         def ch_name &1 "file_name"
8         def ch_name &2 "生成年月日"
9         def ch_name &3 "生成時刻"
10        def ch_name &4 "収録開始年月日"
11        def ch_name &5 "収録開始時刻"
12        def ch_name &6 "信号名"
13        def ch_name &7 "単位"
14        def ch_name &8 "folder_name"
15        def ch_name &9 "収録データ数"
16        def ch_name $1 "ファイル数"
17        def ch_name $4 "収録チャネル数"
18        def ch_name $5 "収録チャネル番号"
19        def ch_name $6 "file_counter"
20        def ch_name $8 "サンプリング周波数:Hz"
21        def ch_name $9 "temp"
22        $1 = 0                                  /*  ファイル数初期化  */
23        repeat_case $1 = 0
24          proc folder_select{
25            get folder_select                   /*  フォルダ選択 */ 26
              read file_info &1 &2 &3 $1 .hdr     /*  ファイル情報取得 */
27          }folder_select
28        read folder_path &8                     /*  カレントフォルダ名取得 */
29        def sheet_title 0: &8                   /* 結果シート・タイトル定義  */
30        $6 = 0                                  /* file counter  初期化*/
31        repeat_case $6 < $1
32          proc file_list {
33            def file_id %1 &1($6) wav
34            read wave %1 0,0
35            &4 = CSDT()                         /*  収録開始年月日取得 */
36            &5 = CSTM()                         /*  収録開始時刻取得 */
37            write ch_column 1: &1($6),&4,&5     /* ファイル名、開始年月日、時刻書き込み*/
38            $4 = NCH()                          /* 収録チャネル数取得*/
39            $5 = CHS()                          /* 収録チャネル番号取得*/
40            &6 = CHNM(0)                        /*  収録チャネル信号名取得*/
41            &7 = CUNT(0)                        /*  収録チャネル単位取得*/
42            write ch_column 1: $4,$5,&6,&7      /* チャネル数、番号、信号名、単位書き込み*/
43            $8 = 1/PRD()                        /*  サンプリング周波数取得 */
44            read header_info &9 $9 "NUM_SAMPS"  /*  収録データ数取得 */
45            write ch_column 1: $8,&9            /* データ数、サンプリング周波数書き込み  */
46            write line_feed 1: 1                /*  書き込み行揃え */
47            $6 = $6+1                           /* file counter UP */
48          }file_list
49        end
```

<Archi_1 Script 記述構文の説明>

23 行目～27 行目:フォルダ選択処理で、選択されたフォルダに解析対象ファイルが存在していない場合、フォル ダ選択を繰り返します。

25 行目： カレントフォルダ選択文でフォルダを選択します。

29 行目： 結果シートの表題として 28 行目で取得したフォルダパス名を定義します。

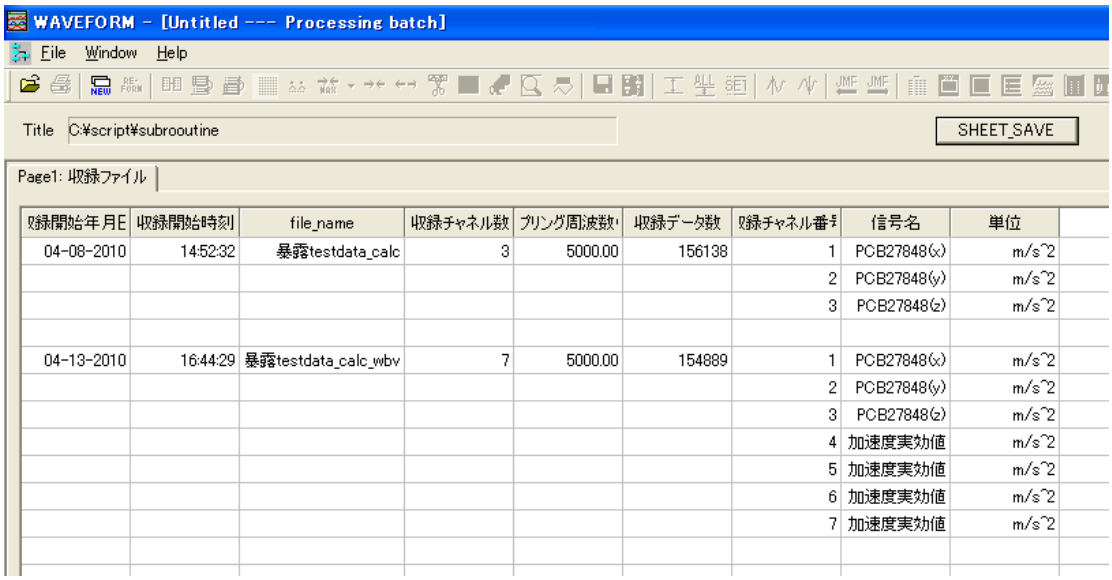
37 行目: 結果シートにファイル名、収録開始年月日、収録開始時刻を書き込みます。 書き込みチャネル列の順番は特に規定されません。書き込み順は結果シート宣言ブロックの当該ページの結果シート書き 込みチャネル列宣言文で記述した順となります。

42 行目： 結果シートにチャネル数、チャネル番号、信号名、単位を書き込みます。

45 行目: 結果シートに収録データ数、サンプリング周波数を書き込みます。 収録チャネル番号と信号名および単位は1つのファイルに複数の内容を持ちますので、当該列はチャネル数分書き込み行 を使用します。

46 行目： ファイル区切りで行を揃えるため、結果シートにラインフィード処理を行います。

<Script 実行結果の結果シート>



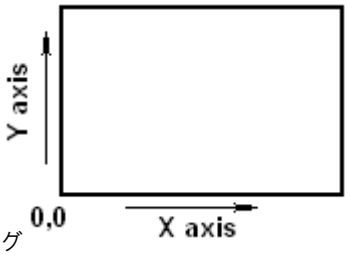
収録開始年月日	収録開始時刻	file_name	収録チャネル数	サンプリング周波数	収録データ数	収録チャネル番号	信号名	単位
04-08-2010	14:52:32	暴露testdata_calc	3	5000.00	156138	1	PCB27848(x)	m/s^2
						2	PCB27848(y)	m/s^2
						3	PCB27848(z)	m/s^2
04-13-2010	16:44:29	暴露testdata_calc_wbv	7	5000.00	154889	1	PCB27848(x)	m/s^2
						2	PCB27848(y)	m/s^2
						3	PCB27848(z)	m/s^2
						4	加速度実効値	m/s^2
						5	加速度実効値	m/s^2
						6	加速度実効値	m/s^2
						7	加速度実効値	m/s^2

11. 演算結果をグラフ表示する

グラフはグラフ描画文が実行される度にグラフダイアログとして表示されます。ダイアログの初期サイズは全画面表示となります。したがって、複数のグラフダイアログを表示した場合、作成順に重なっており、ディス プレイ上では最後に書き込んだグラフが表示されています。同時に複数のグラフダイアログを表示したい場合、グラフダイアログのリサイ ズ操作を行って下さい。

グラフ描画の概念:

現在 Archi_1 Script で実現されているグラフ描画機能は、グラフ枠左下を 0,0 とした X 軸と Y 軸を持つ 2 次元グラフとなります。例えば、Y 軸に或る 1 チャネルを指定し描画すると、X 軸は自動的に Y 軸に指定したチャネルの index が採られ、Y 軸に指定したチャネルの値が採られて描画されます。また、X 軸にあるチャネルを、Y 軸に同じ要素数の別チャネルを指定して描画すると、X 軸は X 軸に指定されたチャネルの値、Y 軸は Y 軸に指定されたチャネルの値が採られた X-Y グラフを描画します。この時、X 軸に指定したチャネルの 内容が時刻歴の場合、描画される内容は Y 軸に指定したチャネルの時刻歴波形 が描画される事になります。又、Y 軸チャネルに複数チャネルを指定すると、X 軸

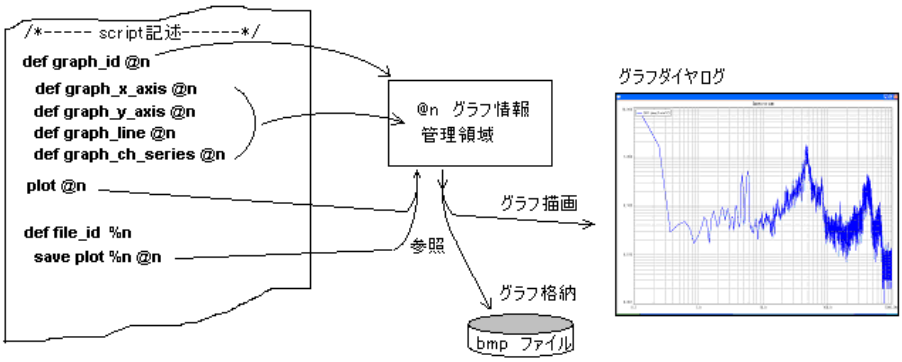


/Y 軸を共有した複数チャネルの時刻歴波形が描画されます。つまり、重要なこと は、グラフとして持っている X 軸、Y 軸は 1 本であると言う事です。

X 軸に異なった時刻歴数列を複数のチャネルで指定し、対応する Y 軸に複数のチャネルを指定すると、X 軸は X 軸に指定されたチャネル列全体の最小値から最大値が採られ、Y 軸は Y 軸に指定されたチャネル列全体の最小値から最大値が採られ、其々、記述順に X 軸設定チャネルと Y 軸設定チャネルがペアとして描画されます。従って、Y 軸或いは X 軸を分割して描画する場合は、方眼紙と考え、其々、軸の目盛位置や値を共通の仮想軸値に変換して描画します。

なお、グラフの X 軸、Y 軸の属性は Linear 尺又は log 尺を設定出来ます。

グラフ関連操作概要:



11. 1. グラフ描画の事前処理

グラフを描画する場合、描画するグラフごとにグラフ番号を定義する必要があります。ここで設定したグラフ番号ごとに管理領域が生成され、グラフ番号を指定設定グラフの描画条件などの定義を行います。

グラフ番号定義文

文法:

def graph_id @グラフ番号 表題 メモ グラフ属性

グラフ番号は先頭文字半角"@"に続き即値で記述します。表題、メモ、およびグラフ属性は記述省略できます。 グラフ番号定義文は、書き込むダイアログを割り当て、以下の定義文や描画文はここで定義したグラフ番号ごとにグラフ情報 管理領域が確保されます。なお、グラフ属性は現在の Version では X-Y しか存在していないため、記述は不要です。

11. 2. グラフを描画する

指定されグラフ番号で管理されるグラフ情報を参照してグラフ描画します。

グラフ描画文

文法:

plot @グラフ番号 X 軸チャネル列 Y 軸チャネル列

グラフ描画文が実行されると当該グラフ番号のグラフ情報管理領域を参照してグラフダイアログにグラフ描画します。チャネル列が Y チャネル列に不足した場合、X チャネル列は循環して参照されます。作成されたグラフダイアログは Script の管理から切り離されます。したがって、書き込み済みのグラフダイアログを Script から操作することはできず、また、同じグラフ番号に描画を繰り返すと、直前までに定義された描画条件で新たなグラフダイアログを作成します。

記述例:

最も記述省略してグラフを描画する場合

```
def graph_id @1 “データグラフ”          /* グラフ番号定義*/
plot $1,$2                              /* X 軸データ番号、Y 軸に$1 と$2 の 2ch をグラフ描画*/
```

記述例は描画するチャネル$1,$2 とも Y 軸データを意味し。X 軸チャネル記述を省略していますので,X 軸は index 番号となり ます。

11. 3. グラフ軸の定義

描画するグラフの X 軸 Y 軸を明示的に定義する場合に記述します。

グラフ X 軸定義文

文法:

```
def graph_x_axis @グラフ番号 グラフ枠情報  目盛情報 単位  軸属性
```

グラフ Y 軸定義文 文法:

```
def graph_y_axis @グラフ番号 グラフ枠情報  目盛情報 凡例情報  軸属性
```

グラフ枠情報は左端値(下限値),右端値(上限値),グリッド線位置の 3 個から構成され、半角カンマで区切って記述します。

X 軸のグラフ枠情報の記述文法: 左端値(必須),右端値(必須),グリッド線位置又はグリッド間隔(省略可)

Y 軸のグラフ枠情報の記述文法: 下限値(必須),上限値(必須),グリッド線位置又はグリッド間隔(省略可)

軸を AutoScale 指定する場合は、左端値,右端値(または下限値,上限値)の何れも 0 と記述します。
グリッド線位置を AutoScale 指定する場合は、記述省略します。又、グリッド線位置を数列で記述した場合は、数列の要素数分、指定した位置にグリッド線を引き、要素数1個の数列以外で記述した場合は、グリッド線間隔を意味します。

11. 3. 1. 目盛情報の定義(X 軸、Y 軸共通) 目盛情報は目盛値または目盛表示形式、表示開始グリッド線番号、表示グリッド線飛び越し数の 3 個から構成され、半角カ ンマで区切って記述します。

文法:

```
目盛値文字列または表示形式(必須),表示グリッド線番号(省略可),表示グリッド線飛び越し数(省略可)
```

目盛値文字列配列で記述する場合は、表示するグリッドごとの目盛値を示し、表示形式で記述する場合は、F 形式、S 形式、E 形式の何れかを記述します。表示グリッド線番号を数列で記述した場合は、明示的に目盛値を表示するグリッド線番号を意味し、数列以外の即値或いは単一要素で記述した場合は開始グリッド線番号を意味します。表示グリッド線飛び越し数は表示グリッド線番号を数列で明示的に指定した場合は意味を持ちません。 なお、目盛情報全体を記述省略した場合は、表示形式 E 形式で全てのグリッドに目盛値を表示します。又、目盛値文字列配 列或いは表示形式のみ記述した場合は、指定された目盛表示形式或いは目盛値を全てのグリッド線に表示します。

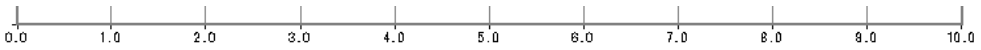
11. 3. 2. 軸属性:linear の場合のグラフ枠情報、目盛値情報の定義例 グラフ枠情報、目盛情報の記述内容により、描画される内容を示します。 なお、下記に示す表示例はグラフ描画文(plot 文)の X 軸値が 0～10 秒で描画されているものとしています。

記述例:

枠情報 AutoScale、目盛情報表示形式のみ指定

```
def graph_x_axis @1 0,0 F1
```

グラフ枠情報左端値/右端値はグラフ描画文から自動取得し、目盛情報表示形式:F1 で定義、他は記述省略した場合表示される X 軸スケール



記述例:

枠情報グリッド間隔設定、目盛情報表示形式指定

```
def graph_x_axis @1 0,0,2 F1
```

グラフ枠情報左端値/右端値はグラフ描画文から自動取得し、グリッド間隔 2、目盛情報表示形式:F1 で定義した場合表示される X 軸スケール

記述例:

枠情報 AutoScale、目盛情報表示形式、開始グリッド線番号、飛び越し数を指定

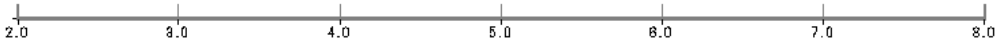
def graph_x_axis @1 0,0 F1,0,1 グラフ枠情報左端値/右端値はグラフ描画文から自動取得し、目盛情報表示形式:F1、開始グリッド線番号:0、グリッド線飛 び越し数:1で定義した場合表示される X 軸スケール

記述例:

グラフ枠情報左端値･右端値、目盛情報表示形式指定

def graph_x_axis @1 2,8 F1

グラフ枠情報左端値:2,右端値:8 としグリッド間隔 AutoScale、目盛情報表示形式:F1 で定義した場合、表示される X 軸スケール

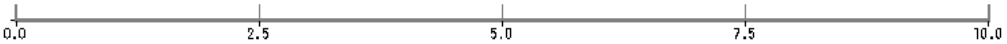


記述例:

枠情報左端値/右端値をグラフ描画文に依存、グリッド線位置、目盛情報表示形式指定

assign $1 = 0,2.5,5,7.5,10

def graph_x_axis @1 0,0,$1 F1 グラフ枠情報左端値/右端値はグラフ描画文から自動取得しグリッド線位置:$1、目盛情報表示形式 F1 で定義した場合表 示される X 軸スケール



記述例I:

グラフ枠情報左端値、右端値、グリッド線位置設定、目盛情報目盛値指定

assign $1 = 0,2.5,,3,4,5,6,7.5,10

assign &1 = $1(F1)|"S"

def graph_x_axis @1 2.5,8,$1 &1

グラフ枠情報左端値:2.5、右端値:8 グリッド線位置を 0,2.5,3,4,5,6,7.5,10 とし、目盛情報目盛値を&1 で定義した場合表示される X 軸スケール

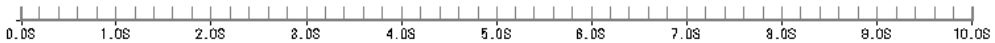
記述例:

枠情報左端値/右端値、グリッド間隔、目盛情報目盛値、表示開始グリッド線番号、飛び越し数指定

assign $2 = 0,1,2,3,4,5,6,7,8,9,10

assign &1 = $2(F1)|"S"

def graph_x_axis @1 0,10,0.2 &1,0,4 グラフ枠情報左端値:0、右端値:10、グリッド線間隔:0.2、目盛情報目盛値&1、表示開始グリッド線番号:0、グリッド線飛び 越し数 4 で定義した場合表示される X 軸スケール



記述例:

枠情報グリッド線位置設定、目盛情報目盛値、目盛表示グリッド線 index 指定

assign $3 = 1,2,3,4,5,6,7,8,9,10

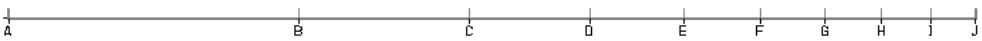
$4 = LGT($3)

assign &1 = "A","B","C","D","E","F","G","H","I","J"

$5 = ACC(DAG(10,1))-1

def graph_x_axis @1 0,1,$4 &1,$5 グラフ枠情報左端値:0、右端値:1、グリッド線位置:$4、目盛情報目盛値:&1、目盛表示グリッド線番号:$5 で定義した場 合表示される X 軸スケール。

目盛表示グリッド線番号を指定する場合、グリッド線飛び越し数は記述しません。



上記例の場合、グリッド線ごとに目盛を表示するため、グリッド線番号の替わりに表示開始グリッド番号 0 飛び越し数 0 としても同じスケールで表示されます。

11. 3. 3. 軸属性:log の場合のグラフ枠情報、目盛値情報の定義例

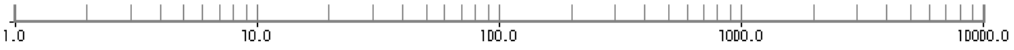
グラフ枠情報で軸属性 linear と異なる点はグリッド位置指定、あるいはグリッド線間隔指定は行えません。また、グラフのスケールはディケードで設定されるため、左端値(下限値)、右端値(上限値)を任意値で設定してもディケードの切れ目で自動的に変更されます。グリッド線はディケード内にも引かれますが、グリッド線番号は振られておらず、グリッド線番号はディケードの切れ目のみ振られます。
目盛情報は目盛値または目盛表示形式、表示開始グリッド線番号、表示グリッド線飛び越し数の 3 個から構成され、半角カンマで区切って記述します。下記に示す表示例は下記に示す表示例はグラフ描画文(plot 文)の X 軸値が 1～10000 で描画されているものとしています。

記述例:

枠情報 AutoScale、目盛情報表示形式のみ指定

```
def graph_x_axis @1 0,0 F1 log
```

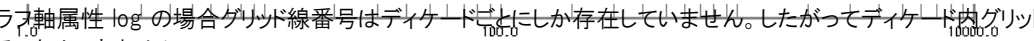
グラフ枠情報左端値/右端値はグラフ描画文から自動取得し、目盛情報表示形式:F1 で定義、他は記述省略した場合表示される X 軸スケール



記述例:

枠情報 AutoScale、目盛情報表示形式、開始グリッド線番号、飛び越し数を指定

def graph_x_axis @1 0,0 F1,0,1 log グラフ枠情報左端値/右端値はグラフ描画文から自動取得し、目盛情報表示形式:F1、開始グリッド線番号:0、グリッド線飛 び越し数:1で定義した場合表示される X 軸スケール



~~グラフ軸属性 log の場合グリッド線番号はディケードごとにしか存在していません。したがってディケード内~~グリッド線番号を指定することはできません。

記述例:

枠情報左端値、右端値を設定、目盛情報目盛値を指定

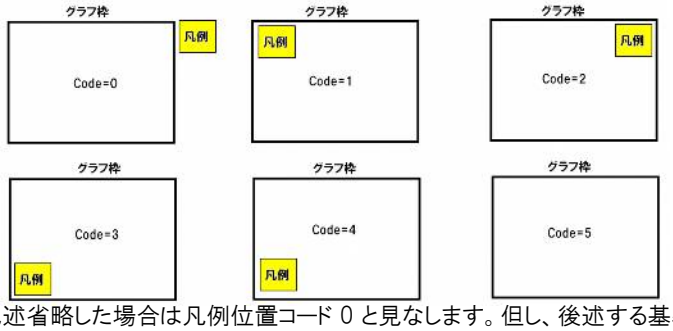
```
assign &1 = "ABC","DEF","GHI","JKL","MNO"
def graph_x_axis @1 1,8e3 &1 log
```

グラフ枠情報左端値:1,右端値:8000、目盛情報目盛値:&1 で定義した場合表示される X 軸スケール

11. 3. 4. 凡例位置情報の定義

凡例情報はグラフ Y 軸定義文で指定する凡例表示位置を意味します。 凡例とは描画されるチャネル線色、チャネル番号、信号名、単位情報を囲んでグラフダイアログに表示するものです。 凡例情報は凡例表示位置コード、表示基準軸コードを半角コロン":"で区切って記述します。

凡例位置の指定＜省略可＞
凡例位置はコードで記述します。コードは凡例表示位置により 0～5 となり、5 は表示しないを意味します。



記述省略した場合は凡例位置コード 0 と見なします。但し、後述する基準軸コードを記述する場合は省略できません。 基準

軸コードの指定＜省略可＞
基準軸とは凡例表示するチャネルをグラフ描画文の Y 軸チャネル列に記述されたチャネルを表示するのか X 軸チャネル列に記述されたチャネルを表示するのかの種別を意味します。
Y 軸チャネル列を表示する場合 ⇒ 0、X 軸チャネル列を表示する場合 ⇒ 1 記述
省略した場合は 0 と見なします。

記述例:

凡例位置をグラフ枠右外上部に表示し、基準軸を Y 軸とする場合

```
def graph_y_axis @1 0,0 F1 5,0 linear
```

11. 4. グラフ描画線種線色の定義

グラフ描画線種、線色を明示的に定義する場合に記述します。

グラフ描画線種線色定義文

文法:

```
def graph_line @グラフ番号 描画線種情報  描画線色情報
```

11. 4. 1. 描画線種の定義

描画線種はデータ描画線種、グラフ枠線種、グリッド線種の 3 個から構成され、半角カンマで区切って記述します。 線種は線種コードで記述します。コード表を下記に示します。

Code	10 の桁:線の種類	1 の桁:線の太さ
0	実線	極細
1	点線	細
2	破線	太
3	1 点鎖線	極太
4	Bar	
5	Dot	

線種 Bar は線の太さ指定は無視されます。

例えば、点線太の場合は 12 となります。なお、データ描画線を描画するチャネルごとに指定する場合は、数列で定義します。

グラフ枠線種およびグリッド線種は1つしか存在しませんので数列で定義しても参照される index は 0 のみとなります。

記述例:

描画チャネルごと、実線極細、点線極細、破線極細とし、グラフ枠線種を実線極太、グリッド線を点線極細とする場合

```
assign $1 = 0,10,20
def graph_line @1 $1,3,10
```

11. 4. 2. 描画線色の設定

描画線色情報はデータ描画線色、グラフ枠グリッド線色、目盛表示色の 3 個から構成され、半角カンマで区切って記述します。色は先頭文字半角"#"に続き HEX 文字 6 桁コードで記述します。なお、データ描画線色をチャネルごとに設定する場合は配列で定義します。

記述例:

```
assign &1 = "#800000","#0000FF","#0000FF"  /* 描画チャネル順に濃紺色、赤色、青色に設定*/
def graph_line @1 &1,"#C00000","#000000"  /* グラフ枠/グリッド線を灰色、目盛表示色を黒色に設定*/
```

色コードを記述する場合、編集機能の ColorPalet 機能を使用し、描画色を選択してコードに変換記述することができます。

11. 5. グラフ枠内追加線の描画定義

グラフ枠内に補助線を引く場合に記述します。

グラフ追加線定義文

文法:

```
def graph_line_draw @グラフ番号 追加線情報 描画線種情報  描画線色情報
```

グラフ枠内に境界線等を引く場合などに使用し、グラフ描画文の直前までに定義ます。追加線情報は X 座標、Y 座標、Pen 制御からなり半角カンマで区切って記述します。X 座標、Y 座標および Pen 制御は何れも数列で記述し、要素数が不足した数列は循環して参照されます。また、グラフ枠内追加線定義は、同一グラフ番号に複数行記述できます。複数行記述されている場合、描画に先立って連結されてますので、最初に記述する定義行の先頭ペン制御データは 0,1のいずれでも問題ありませんが、続いて定義する行のペン制御データの先頭は 0 である必要があります。描画線種情報及び描画線色情報は、グラフ描画線種線色情報と記述文法は同じです。なお、何れも記述省略可能です。

記述例:

X 軸 5.5、Y 軸 5.5 とした十字線を引く場合

```
def graph_id @1
def graph_x_axis @1 0,10,1 F1 linear
def graph_y_axis @1 0,10,1 F1 5 linear
assign $1 "pen:" = 0,1                  /* ペンアップ／ダウン制御データ生成*/
assign $2 "x_pos:" = 5.5,5.5,0,10       /* X 軸ポジションデータ生成*/
```

```
assign $3 y_pos: " = 10,0,5.5,5.5          /* Y 軸ポジションデータ生成*/
def graph_line_draw @1 $2,$3,$1            /* グラフ枠内追加線描画定義*/
```

ペン制御	X ポジション	Y ポジション	動作内容
0	5.5	10	X=5.5,Y=10 に移動
1	5.5	0	X=5,5、Y=0 にペンダウンして移動(直前位置から描画)
0	0	5.5	X=0、Y=5,5 に移動
1	10	5.5	X=10、Y=5.5 にペンダウンして移動(直前位置から描画)

11. 6. グラフ描画チャネル番号の設定

グラフ描画に際して凡例で表示するチャネル番号を明示的に定義する場合に記述します。本文が記述省略された場合凡例で表示 されるチャネル番号は、グラフ描画文の X 軸チャネル並び、または Y 軸チャネル並びでの記述順に ch1 から自動的に振られます。

グラフ表示チャネル番号定義文

文法：

def graph_ch_series @グラフ番号 チャネル番号

複数のチャネルを表示する場合はチャネル番号を数列で構成し、数列に描画順にチャネル番号を格納して記述します。

記述例:

表示チャネル番号をグラフ描画文の記述順に 3,4,7 と設定する場合

```
assign $1 = 3,4,7
def graphch_series @1 $1
```

11. 7. グラフ描画枠縦横比の設定 グラフ描画枠の初期値は接続されているディスプレイの全画面表示時の縦横比となります。グラフ描画枠の縦横比を明 示的に設定する場合に使用します。

グラフ縦横比定義文

文法:

def graph_aspect_ratio @グラフ番号 縦横比

记述例:

グラフ枠縦横比を 2 に設定する場合

```
def graph_aspect_ratio @1 2      /*横/縦比が 2*/
```

11. 8. グラフの格納

グラフ格納文

save plot %ファイル番号 @グラフ番号 X 軸チャネル列 Y 軸チャネル列

グラフを格納する場合、本文に先立ってファイル番号を属性 grp として定義が必要です。 なおファイル名に拡張子は付けません。格納されるグラフファイルは拡張子.bmp のビットマップ形式に固定されています。

記述例:

```
def file_id %1 "test_graph" grp
def graph_id @1 "試験波形"
save plot %1 @1 $1 $3,$4
```

グラフ番号@1 に書き込まれている情報を参照してファイルにグラフを描画します。なお、上記例では、グラフ表題以外は省略値を使用して X 軸に$1 を Y 軸に$3 と$4 をグラフ描画しています。

11. 9. AutoScale 目盛値の取得

グラフを描画する際に、X 軸定義文或いは Y 軸定義文で、目盛文字列を与える場合があります。その場合に目盛値数列を AutoScale で求める場合【ATS 関数】を使用します。

文法:

格納先 = ATS(最大許容目盛値数,最大値,最小値)

引数:

【最大許容目盛値数】＜必須＞

設定する最小値と最大値の間を 1,2,5 ステップで分割した場合の最大許容目盛値数を記述します。

最大許容目盛値数とは、設定した最小値と最大値を含まず、その内輪に表示する目盛値の数、言い換えればグリッド線本数を意味します。

【最小値】＜必須＞

対象範囲の最小値を記述します。

【最大値】＜必須＞ 対象範囲の最大
値を記述します。

戻り値は、設定した最小値と最大値の間のグリッド線を想定した目盛値数列となります。従って、設定した最小値及び最大値は含みません。戻り値数列の要素数は設定した許容分割数以内となり、グリッド線数を知りたい場合は改めて要素数を取得する必要があります。要素数の取得は LEN 関数を使用します。

記述例:

最小値、-0.123、最大値+2.5 の時、許容分割数 8 として目盛数列を求める場合

```
$1 = ATS(8,-0.123,2.5)
$2 = LEN($1)
```

$1 には、0,0.5,1,1.5,2 が格納され、$2 には,5 が格納されます。グラフ表示する場合は、グラフ枠最小値と最大値を数列の両端に追加する必要があります。グラフ枠最小値及び最小値は表示するチャネルの最小値、最大値を使用するか、または、数列からグリッド間隔を求め、数列の先頭値-グリッド間隔と数列の最終値+グリッド間隔を使用します。

記述例:

Y 軸定義に AutoScale 使用せず、#1 を目盛分割数 8 として AutoScale して明示的に目盛文字列を設定する場合

```
$1 = MIN(#1)                    /* 表示チャネルの最小値を取得*/
$2 = MAX(#1)                    /* 表示チャネルの最大値を取得*/
$3 = ATS(8,$1,$2)              /* 目盛数列の取得*/
$4 = DIF($3)                   /* 数列の差分演算*/
$4 = $4(1)                     /* グリッド間隔の取得*/
$3 = LNK($1,$3,$2)            /* グラフ表示目盛数列の生成*/
assign &1 = $3(F0)            /* 目盛数列の文字列変換*/
def graph_id @1               /* グラフ番号定義*/
def graph_y_axis @1 $1,$2,$4 &1.0   /*Y 軸の定義*/
def graph_x_axis @1 0,0 F3 /* X 軸の定義*/
plot @1 #1                     /* グラフ描画*/
```

上記例は def graph_y_axis @1 0,0 F3 と記述した場合と同じ意味を持ちます。

11. 9. 同一グラフ内に複数チャネルの表示方法

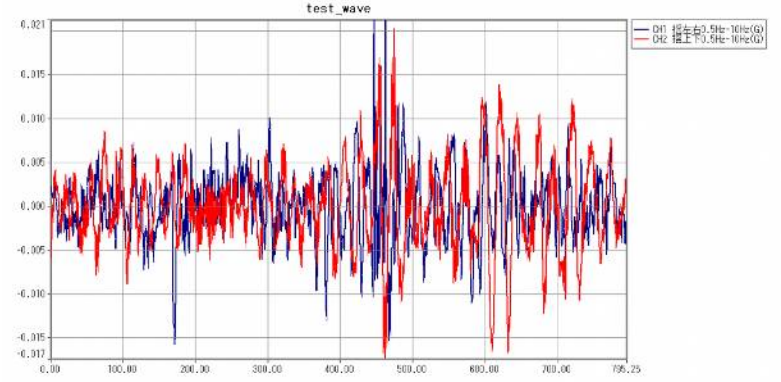
11. 9. 1. 同じ単位の複数チャネルを X 軸、Y 軸を共有して描画する方法

記述例:

表示対象チャネルの単位が同じ#3,#4 を X 軸、Y 軸を共有して AutoScale で重ね書きをする場合

```
/*--- 2ch 重ね書きグラフ( X 軸,Y 軸共有)----------------*/
$1 "時間軸:sec" = SPB(0)                /* 時間軸数列生成*/
assign $7 "チャネル表示:" = 3,4         /* #3,#4*/
def graph_id @1 "test_wave"             /* グラフ番号定義*/
def graph_x_axis @1 0,0 F2 linear       /* X 軸定義*/
def graph_y_axis @1 0,0 F3 linear      /* Y 軸定義*/
def graph_aspect_ratio @1 2             /* グラフ縦横比定義*/
plot @1 $1 #3,#4                        /* 波形データ描画*/
/*--- グラフ格納 ----------------------------------*/
def file_id %1 "波形グラフ" grp         /* ファイル番号定義*/
save plot %1 @1 $1 #3,#4                /* グラフ格納*/
```

<実行結果のグラフ>



この方式は、グラフ Y 軸定義文に任せてスケーリングする為、特に Y 軸の生成等は必要ありません。但し、表示されるチャネル波形の判別は線色、線種で行うしかなく、粗同じ様な波形の場合判別し難く、また、同じ単位で無い場合は目盛がかけ離れていない場合は Y軸の目盛を共有できない不都合が起こります。

11. 9. 2. 同じ単位の複数チャネルをグラフの Y 軸共通グリッド幅で分割して描画する方法

同じ単位のチャネルだけ集めて共通のグリッド幅で表示するグラフを作成します。

Y 軸を同じグリッド幅で異なる複数チャネルを表示する為の Y 軸生成演算手順:

① 目盛値数列を生成する為の表示する全てのチャネルの最小値と最大値を求める。

```
repeat_case チャネルカウンタ < 表示チャネル数
  proc Max_Min{
    最大値数列 = LNK(最大値数列,MAX(表示チャネル))
    最小値数列" = LNK(最小値数列,MIN(表示チャネル))
    チャネルカウンタ = チャネルカウンタ+1
  }Max_Min
最大値 = MAX(最大値数列) 最小
値 = MIN(最小値数列)
```

② ①で求めた最大値、最小値を引数として ATS 関数で目盛値数列と目盛数を求める。

```
目盛値数列 = ATS(グリッド線本数,最小値,最大値) チャネ
ル当たりグリッド線数 = LEN(目盛数列) グリッド線間隔 =
目盛値数列(1)-目盛値数列(0)
```

③ 目盛値数列を生成します。

```
目盛値数列 = LNK(最小値,目盛値数列,目盛値数列,最大値)
assign 目盛値文字列 = 目盛値数列(表示形式指定)
```

④ グラフ上段に描画するチャネルのデータをオフセットさせる修飾を行い、グラフ Y 軸仮想最大値を求める。

```
上段表示修飾データ = 上段表示チャネル+グリッド線間隔*チャネル当たりグリッド線数
仮想最大値 = MAX(上段表示修飾データ)
```

⑤ グラフ Y 軸の定義

```
def graph_y_axis @1 最小値,仮想最大値,グリッド線間隔 目盛値文字列,0
```

記述例:

グラフの Y 軸を共有して同じグリッド幅で複数チャネルを表示する外部演算処理ブロックを作成する

```
proc 波形グラフ 1{
  /*-----------------------------------------------------------------
  呼び出し方法:
  call proc 波形グラフ 1 チャネル数,チャネル番号,データ格納先先頭,グラフ表題 引
  数:
  $1 <in> チャネル数 $2 <in> チャネル番号  $3 <in> データ格納先先頭
  &1 <in> グラフ表題
  ※ 表示データは格納先先頭から表示チャネル数分連続して格納されている必要あり
  -----------------------------------------------------------------*/
  def inherit_ch $1,$2,$3,&1
  def local_ch $4,$5,$6,$7,$8,$9,$10,&2
  $4 = 0
  repeat_case $4 < $1                          /* チャネル LOOP*/
    proc Max_Min{
      $5 "MAX:" = LNK($5,MAX($($3+$4))) /* 表示チャネルごとの最大値数列を求める*/
      $6 "MIN:" = LNK($6,MIN($($3+$4)))   /* 表示チャネルごとの最小値数列を求める*/
      $4 = $4+1
    }Max_Min
  $5 = MAX($5)                                 /* 表示チャネル群の最大値を求める*/
  $6 = MIN($6)                                 /* 表示チャネル群の最小値を求める*/
  $9 = ATS(8,$6,$5)                            /* 共通グリッド目盛数列を求める*/
  $7 "基本目盛数:" = LEN($9)                   /* 目盛数列から分割数を求める*/
  assign $8 = $6,$1<$9>,$5                     /* 目盛値数列の生成*/
  assign &2 "目盛文字列:" = $8(F3)             /* 目盛文字列変換*/
  $9 = $9(1)-$9(0)                             /* 目盛数列の差分(グリッド間隔)を求める*/
  /*------ 表示チャネルデータ修飾--------------------------------*/
  $4 = 1
  repeat_case $4 < $1                           /* チャネル LOOP*/
    proc 表示データ修飾{
      $($3+$4) = $($3+$4)+($1-$4)*$7*$9   /* データのオフセット処理*/
      $10 = LNK($10,MAX($($3+$4)))
      $4 = $4+1
    }表示データ修飾
```

```
  $5 = MAX($10)                                  /* グラフ Y 軸最大値演算*/
  /*----- グラフ描画-----------------------------------------*/
  $10 "時間軸:sec" = SPB(0)                      /* 時間軸数列生成*/
  def graph_id @1 &1                             /* グラフ番号定義*/
  def graph_ch_series @1 $2                      /* 表示チャネル番号定義*/
  def graph_x_axis @1 0,0 F2 &2                  /* X 軸定義*/
  def graph_y_axis @1 $6,$5,$9 &2,0              /* Y 軸定義*/
  def graph_line @1 2 "#FF0000"
  plot @1 $10 $[$3,$1]                           /* 波形データ描画*/
  /*--- グラフ格納 ----------------------------------------*/
  def file_id %1 "波形グラフ" grp                /* ファイル番号定義*/
  save plot %1 @1 $10 $[$3,$1]                   /* グラフ格納*/
}波形グラフ 1
```

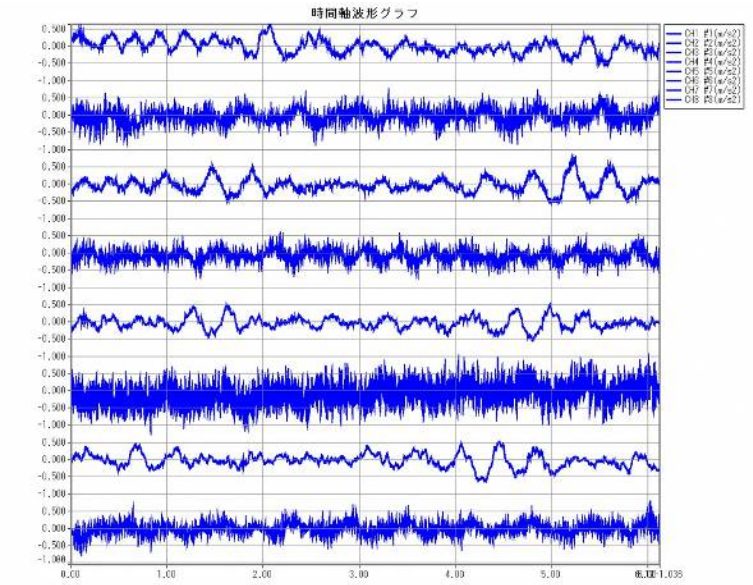
記述例:

収録データの同じ単位のチャネルを作成した外部演算処理ブロックを使用してグラフ表示する場合

```
/*-----収録チャネルから指定された単位チャネルをグラフ表示する----------------------*/
$1 "収録チャネル数:" = NCH()
$2 "収録チャネル番号:" = CHS()
&1 "収録チャネル名:" = CHNM(0)
&2 "収録チャネル単位:" = CUNT(0)
/*---- 表示するデータの単位を指定する-----------------------------------------*/
assign &3 "表示単位:" = "m/s2"
$3 "単位一致数列:" = CEQ(&3,&2)
$2 = ZSP($3,$2)                                  /* チャネル番号数列再構成*/
&1 = CREC($3,&1)                                 /* チャネル名配列再構成*/
&2 = CREC($3,&2)                                 /* 単位配列再構成*/
$1 = LEN($2)                                     /* チャネル数再取得*/
$4 "データ格納先先頭:" = 100                     /* 表示対象データ格納先先頭チャネル番号*/
$5 "ch_counter:" = 0                             /* チャネルカウンタ初期化*/
repeat_case $5 < $1
  proc 表示データ格納{
    $6 "格納先 ch:" = $4+$5
    $($6) = #($2($5))
    def ch_name $($6) &1($5)
    def ch_unit $($6) &2($5)
    $5 = $5+1
  }表示データ格納
assign &3 "グラフ表題:" = "時間軸波形グラフ"
call proc 波形グラフ 1 $1,$2,$4,&3      /* 外部演算処理ブロック呼び出し*/
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果のグラフ>



11. 9. 3. 異なる単位の複数チャネルを、グラフ枠 Y 軸を等分割して描画する方法 異なる単位の場合、グリッド間隔を共有する事はできない為、仮想 Y 軸グリッド線数列及び目盛値配列を生成し、チャネルご とに描画区画設定したセパレートグラフで表示します。それぞれチャネルごとにグリッド線間隔が異なる描画をします。

Y 軸を表示チャネル数で等分割したセパレートグラフの Y 軸生成演算手順:

① グラフ Y 軸の仮想上限を求める。
仮想 Y 軸上限 = 表示チャネル数*10
※ 例えば表示チャネル数が 16ch の場合、仮想 Y 軸は 0 から 160 迄の軸値を持つと考えます

② 表示するチャネル数分のループを構成し、チャネルごと、最大値、最小値を求め ATS 関数で目盛値数列と目盛数を求めます。(チャネル LOOP 内処理)
チャネル最大値 = MAX(表示チャネル)
チャネル最小値 = MIN(表示チャネル)
チャネル目盛値数列 = ATS(グリッド線本数,チャネル最小値,チャネル最大値)

③ チャネルごとの縮尺率とオフセットを求める。(チャネル LOOP 内処理)
チャネル縮尺率 = 10/(チャネル最大値-チャネル最小値)
チャネルオフセット = 10*(表示チャネル数-表示順位)-チャネル最小値*チャネル縮尺率

④ 換算グリッド線位置数列と表示目盛数列を生成する。(チャネル LOOP 内処理)
換算グリッド線位置 = LNK(換算グリッド線位置,REV(チャネル目盛数列*チャネル縮尺率+チャネルオフセット))
表示目盛値数列 = LNK(表示目盛値数列,REV(チャネル目盛数列))

⑤ 表示目盛数列並びを逆順にし、目盛数列を文字列に変換する。
表示目盛数列 = REV(表示目盛数列)
assign 目盛文字列 = 表示目盛数列(表示形式) 目盛表示位置 index = ACC(DAG(LEN(換算繰り度線位置),1))-1

⑥ 生成した各パラメタでグラフ Y 軸を定義する
def graph_y_axis @1 0,仮想 Y 軸上限,換算グリッド線位置 目盛値文字列,目盛表示位置 index

記述例:

グラフの Y 軸を表示チャネル数で等分割し、チャネルごとに AutoScale して複数チャネルを表示する外部演算処理ブロックを作成する

```
proc 波形グラフ 2{
  /*-------------------------------------------------------
  呼び出し方法:
  call proc 波形グラフ 2 表示チャネル数,データ格納先,表示チャネル番号,グラフ表題
  引数:
  $1 <in> 表示チャネル数   $2 <in> 表示データ格納先
  $3 <in> 表示チャネル番号 &1 <in>  グラフ表題
  ※ 表示データは格納先先頭から表示チャネル数分連続して格納されている必要あり
  -------------------------------------------------------*/
  def inherit_ch $1,$2,$3,&1
  def local_ch $[5,15],&3
  def ch_name $1 "表示チャネル数"
  def ch_name $2 "格納先先頭"
  def ch_name $3 "表示チャネル番号"
  $5 "グラフ仮想 Y 軸上限:" = $1*10                  /*グラフ仮想 Y 軸上限演算*/
  $6 "グラフ表示 ch 順位:" = ACC(DAG($1,1))          /* グラフ表示チャネル番号生成*/
  $7 "チャネル最大値:" = $6                          /* 配列準備*/
  $8 "チャネル最小値:" = $6                          /* 配列準備*/
  $9 "縮尺率:" = $6                                  /* 配列準備*/
  $10 "オフセット:" = $6                             /* 配列準備*/
  /*------- チャネル LOOP-------------------------------------------------*/
  $11 "ch_counter:" = 0
  repeat_case $11 < $1
    proc ch_loop{
      $7($11) = MAX($($11+$2))                       /* 最大値取得*/
      $8($11) = MIN($($11+$2))                       /* 最小値取得*/
      /* ----- チャネルごと最大値と最小値が等しい場合の処理----------------*/
      case $7($11) = $8($11)
        proc scale_modf{
          $7($11) = RVS(EQU($7($11),0),$7($11)*1.2,0.1)
          $8($11) = RVS(EQU($8($11),0),$8($11)*0.8,-0.1)
        }scale_modf
      /*----- チャネルごと  グラフ Y 軸スケール生成-----------------------*/
      $12 "グリッド数列:" = ATS(4,$8($11),$7($11))   /* チャネルごと AutoScleGrid 線位置*/
      $9($11) = 10/($7($11)-$8($11))                 /* Y 軸縮尺率演算*/
      $10($11) = 10*($1-$6($11))-$8($11)*$9($11) /* Y 軸オフセット量演算*/
```

```
        $($2+$11) = $($2+$11)*$9($11)+$10($11)          /* 表示データ変換*/
        $13 "換算グリッド線位置:" = LNK($13,REV($12*$9($11)+$10($11))) /*グリッド線位置換算*/
        $14 "表示目盛値:" = LNK($14,REV($12))          /* 目盛値格納*/
        $11 = $11+1                                     /* 収録チャネルカウンタ*/
      }ch_loop
    $14 = REV($14)                                      /* 目盛値逆転*/
    assign &3 "表示目盛文字列:" = $14(F2)              /* 目盛値文字列変換*/
    $13 = SRT(1,$13)                                    /* グリッド位置整列*/
    $15 "時刻歴数列:" = SPB(0)                          /* X 軸値生成*/
    $12 = MGR(ACC(DAG($1-1,10)),ACC(DAG($1-1,10))) /* ch 境界 Y 位置*/
    $16 "ch 境界 X アドレス:" = LNK(MIN($15),MAX($15))  /* ch 境界 X 位置*/
    $17 "pen 制御:" = LNK(1,0)                          /* Pen 制御*/
    $18 = ACC(DAG(LEN($13),1))-1                        /* 目盛表示グリッド番号取得*/
    def graph_id @1 &1
    def graph_ch_series @1 $3
    def graph_line @1 0 "#000000","#74B1FC"
    def graph_line_draw @1 $16,$12,$17
    def graph_y_axis @1 0,$5,$13 &3,$18 linear
    def graph_x_axis @1 0,0 F1 "sec" linear
    plot @1 $15 $[100,$1]
    def file_id %1 "波形グラフ" grp
    save plot %1 @1 $15 $[100,$1]
  }波形グラフ 2
```

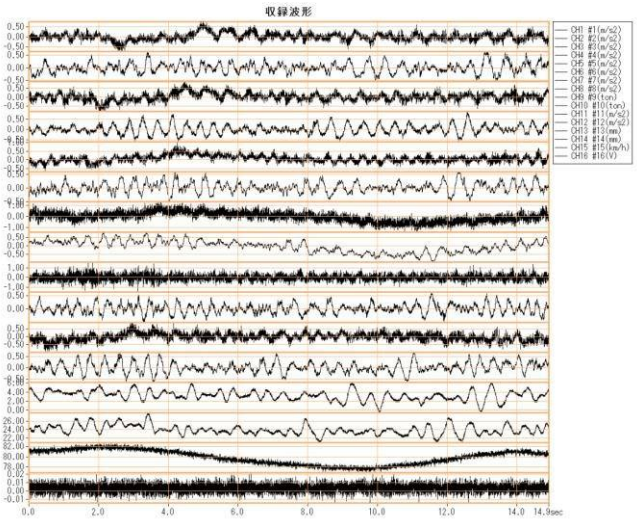
記述例:

収録データの全チャネルをセパレートグラフ形式で表示する場合

```
/*----- 収録データ全チャネルをグラフ表示する----------*/
$1 "表示チャネル数:" = NCH()            /*収録チャネル数取得*/
$2 "収録 ch 番号:" = CHS()              /* 収録チャネル番号取得*/
&1 "収録 ch 名:" = CHNM(0)              /* 収録チャネル名取得*/
&2 "収録 ch 単位:" = CUNT(0)            /* 収録チャネル単位取得*/
$3 "格納先 ch 番号:" = 100              /* 格納先チャネル番号先頭初期化*/
$4 "ch_counter:" = 0                    /* チャネル Loop カウンタ初期化*/
repeat_case $4 < $1                     /* 表示データの格納*/
  proc データ格納{
    $5 "格納先 ch 番号:" = $3+$4
    $($5) = #($2($4))
    def ch_name $($5) &1($4)
    def ch_unit $($5) &2($4)
    $4 = $4+1
  }データ格納
    assign &3 = "収録波形"
call proc 波形グラフ 2 $1,$3,$2,&3     /* 外部演算処理ブロック呼び出し*/
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:波形グラフ>



11. 9. 4. 異なる単位の複数チャネルを、グラフ枠 Y 軸を共有して描画する方法 グラフ枠を共有し、グラフ上のグリッド間隔はチャネル間で共通としたコモングラフ形式で表示します。チャネルごとに異なるグ リッド間隔値を凡例に表示します。

Y 軸を共有し表示チャネルごとに書き出し位置を割り当てたコモングラフの Y 軸生成演算手順:

① グラフ Y 軸の仮想 Y 軸と上限を求める。

仮想 Y 軸上限 = 表示チャネル数*チャネル占有グリッド本数
仮想 Y 軸数列 = ACC(DAG(チャネル占有グリッド本数*表示チャネル数+2,1))-1
仮想 Y 軸最大 = MAX(仮想 Y 軸数列)
※ 例えば表示チャネル数が 16ch の場合、仮想 Y 軸は 0 から 161 迄の軸値を持つと考えます

② 表示するチャネル数分のループを構成し、チャネルごと、最大値、最小値を求め ATS 関数で目盛値数列と目盛数を求める。(チャネル LOOP 内処理)

チャネル絶対値最大 = RVS(3,ABS(MIN($($3+$6))),ABS(MAX($($3+$6))))
チャネル目盛数列 = ATS(5,0,チャネル絶対値最大)
チャネルグリッド間隔 = ABS(チャネル目盛数列(0)-チャネル目盛数列(1))

③ 表示するチャネルごとにチャネルのゼロ位置とゼロ位置数列を求める。(チャネル LOOP 内処理)

ゼログリッド番号 = 仮想 Y 軸最大-チャネル占有グリッド本数*(チャネルカウンタ+1)+3 ゼログリッド数列 = LNK(ゼログリッド数列,ゼログリッド番号)

④ 表示データの修飾とチャネルゼロ位置表示文字列を生成する。(チャネル LOOP 内処理)

表示データ = 表示データ/チャネルグリッド間隔+ゼログリッド番号
assign ゼロ位置表示文字列 = "CH"|チャネル番号(表示形式)|" ⇒"

⑤ 生成した各パラメタでグラフ Y 軸を定義する

def graph_y_axis @1 0,仮想 Y 軸最大,チャネルグリッド間隔 ゼロ位置表示文字列,ゼログリッド数列

記述例:

グラフの Y 軸を共有したコモングラフ形式で、チャネルごとに AutoScaleしたグリッド間隔を凡例に表示して複数チャネルを表示する外部演算処理ブロックを作成する

```
proc 波形グラフ 3{
  /*-------------------------------------------------------------
  呼び出し方法:
  call proc 波形グラフ 3 表示チャネル数,表示チャネル番号,格納先先頭,グラフ表題,単位 引
  数:
  $1 <in> 表示チャネル数   $2 <in> 表示チャネル番号
  $3 <in> 格納先先頭       &1 <in> グラフ表題 &2 <in>  チャネル単位配列
  ※ 表示データは格納先先頭から表示チャネル数分連続して格納されている必要あり
  -------------------------------------------------------------*/
  def inherit_ch $1,$2,$3,&1,&2
  def local_ch $4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,&3,&4
  $14 "チャネル占有グリッド:" = 4
  $4 "仮想 Y 軸数列:" = ACC(DAG($14*$1+2,1))-1
  $5 "仮想 Y 軸最大:" = MAX($4)
  assign &3 "チャネル位置マーク:" = $1<" ">  /*配列準備*/
  $6 "チャネルカウンタ:"  = 0
  repeat_case $6 < $1                          /*  チャネル loop*/
    proc scale_conv{
      $7 "チャネル絶対値最大:" = RVS(3,ABS(MIN($($3+$6))),ABS(MAX($($3+$6))))
      $7 = RVS(EQU($7,0),$7,0.01)
      $8 "チャネル目盛数列:" = ATS($14/2,0,$7)
      $15 = LEN($8)
      case $15 = 1
        proc grid1{
          $9 "チャネルグリッド間隔:" = $8
        }grid1
      case $15 > 1
        proc grid2{
          $9 "チャネルグリッド間隔:" = ABS($8(0)-$8(1))
        }grid2
      $10 "ゼログリッド番号:" = $5-$14*($6+1)+3   /* ch zero grid Number*/
      $11 "ゼログリッド数列:" = LNK($11,$10)      /* gird_series*/
      assign &4 = $9(F2)|&2($6)|"/div"             /*  グリッド間隔表示生成*/
      $12 = $6+$3                                    /*  格納先チャネル番号生成*/
      def ch_unit $($12) &4                         /*  単位再定義*/
      $($3+$6) = $($3+$6)/$9+$10                     /*  表示データ修飾 */
      assign &3($6) = "CH"|$2($6)(F0)|" ⇒"
      $6 = $6+1                                      /*  チャネルカウンタUp*/
```

```
    }scale_conv
  $12 = SPB(0)                                    /* グラフ X 軸生成*/
  def graph_id @1 &1
  def graph_line @1 0,2,10 "#FF0000"
  def graph_x_axis @1 0,0 F2
  def graph_y_axis @1 0,$5,1 &3,$11
  plot @1 $12 $[$3,$1]
  def file_id %1 "multiCh" grp
  save plot %1 @1 $12 $[$3,$1]
}波形グラフ 3
```

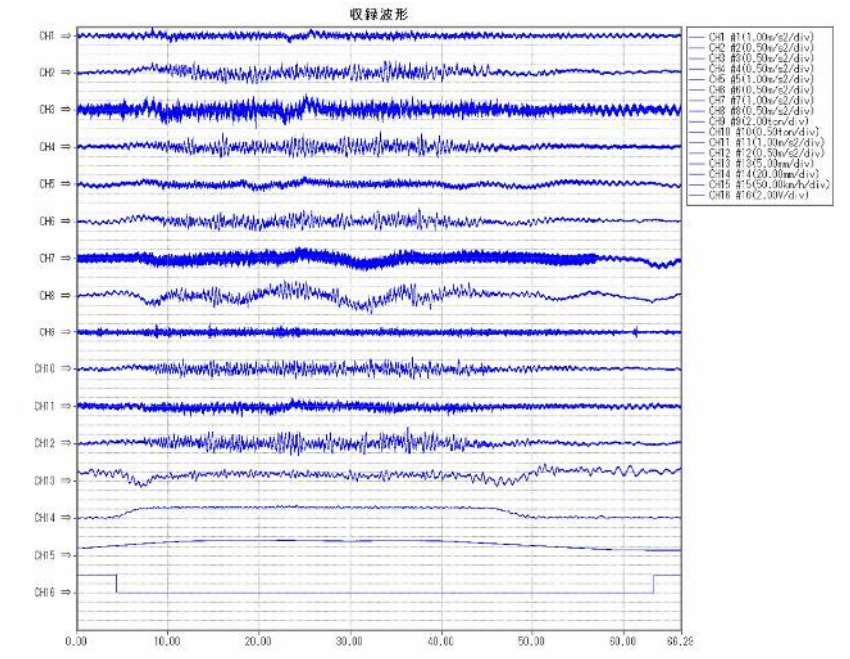
記述例:

収録データ全チャネルをコモングラフ形式で描画する場合

```
/*------------ ------------*/
$1 "収録チャネル数:" = NCH()
$2 "収録チャネル番号:" = CHS()
&1 "収録チャネル名:" = CHNM(0)
&2 "収録単位:" = CUNT(0)
$4 "格納先先頭:" = 100
$5 "ch_counter:" = 0          /*チャネルカウンタ初期化*/
repeat_case $5 < $1           /*チャネル LOOP*/
  proc 表示 ch 代入{
    $6 "格納先 ch 番号:" = $5+$4
    $($6) = #($2($5))             /*表示チャネルに代入*/
    def ch_name $($6) &1($5)      /*表示チャネルに信号名定義*/
    $5 = $5+1                       /*チャネルカウンタ Up*/
  }表示 ch 代入
assign &3 "グラフ表題:" = "収録波形"
call proc 波形グラフ 3 $1,$2,$4,&3,&2 /*外部演算処理ブロック呼び出し*/
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:波形グラフ>



記述例11. 1. 両対数グラフをリニア尺で表示するグラフを作成する

PcWaveFrom でスペクトル解析した結果格納ファイルのデータを読み出し、Y 軸は dB 変換して linear 尺とし、X 軸の周波数は、R10 系列とした linear 尺としてスペクトラムグラフを作成します。
※　R10 系列とはISO266:1997[5]で規定された周波数系列を意味します。

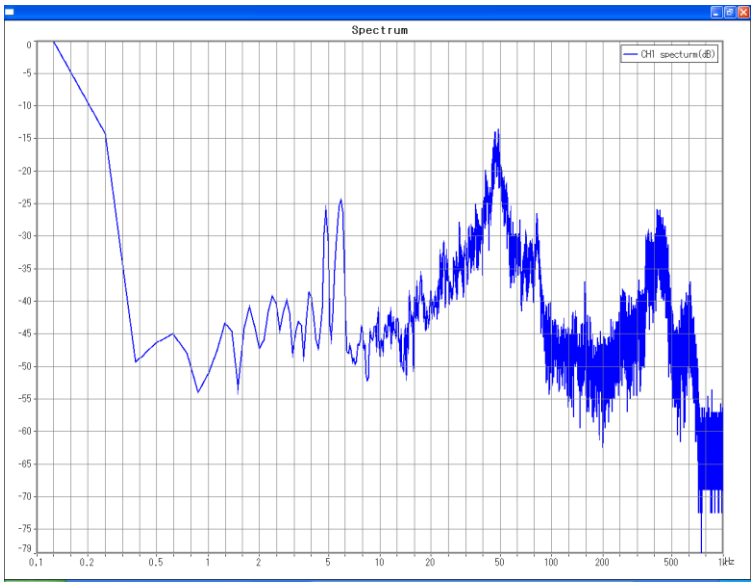
<Archi_1 Script 記述例>

```
1       /*---------graph12.prc-----------------*/
2       def ch_name $1 "freq:Hz"
3       def ch_name $2 "specturm:dB"
4       def ch_name $3 "base2index"
5       def ch_name $5 "num_index"
6       def ch_name $6 "graphX 軸"
7       def ch_name $8 "目盛 1桁目"
8       def ch_name $9 "目盛位置"
9       def ch_name $10 "目盛値"
10      def file_id %1 "fft_result" csv
11      read cell %1 8,2 1 $1,$2
12      $2 = 20*LGT($2/MAX($2))                 /* dB 変換  Max=0dB */
13      $3 = FTI($1)   /*3*LOG($1/1000)/LOG(2) /*  周波数⇒base2 Index 変換 */
14      $6 = LNK(-40,RTI($1(0),$1(LEN($1)-1))) /*RTI NG need LINK*/
15      $5 = LEN($6)-1
16      assign $8 = 0,7,3                       /* 1,2,5 step R10index*/
17      $9 = SRT(1,CMP($8,ABS($6)-INT(ABS($6)/10)*10)) /* 目盛 index 取得*/
18      $10 = RTF(PTV($9,$6))                   /* 目盛 Index 位置公称周波数取得*/
19      assign &1 = $10(F1)
20      def graph_id@%2 "Spectrum"
21      def graph_x_axis @2 $6(0),$6($5),$6 &1,$9 "Hz" linear
22      def graph_y_axis @2 0,0,5 F0,0 2 linear
23      plot @2 $3 $2
24      end
```

<Archi_1 Script 記述構文の説明>

10 行目～11 行目:PcWaveForm で FFT 解析した結果ファイルを読み出します。
10 行目:　ファイル番号定義文で読み出すファイル名を"fft_result"、属性を csv として定義します。
11 行目: テキストファイルセル読み出し文で、周波数とスペクトラムを読み出します。なお、ファイルのデータ行は 7 行目から開始されますが 7 行目は周波数 0(直流)ですので、読み出しは 8 行目から行います。
方法は第12章「テキストファイルからの読み出し」項を参照して下さい。
12 行目: 表示するスペクトラム格納チャネル$2 を dB に変換します。
13 行目: 周波数を R10 系列の小数点以下を含む R10Index に変換します。
14 行目:　R10 系列 Index を X 軸とするため、最小周波数と最大周波数を含む X 軸用 R10Index 数列を生成します。
17 行目:　目盛を表示する位置を取得します。
18 行目:　目盛表示位置を公称周波数に変換します。
19 行目:　公称周波数を表示用の文字列に変換します。
21 行目: グラフ X 軸定義文で、グラフ枠情報を左端値:$6(0)(X 軸用 R10Index 最小値)、右端値:$6($5)(X 軸用 R10Index 最大値)、グリッド線位置:$6(表示用 R10index)とし、目盛情報を目盛値:&1(公称周波数)、グリッド線目盛位置:$9(17 行目で生成した目盛位置)とし軸属性:linear として定義します。
22 行目: グラフ Y 軸定義文で、グラフ枠情報を下端値:0、上端値:0(AutoScale)、グリッド間隔:5 とし、目盛情報を目盛表示形式:F0 とし軸属性を linear として定義します。
23 行目:　グラフ描画文で X 軸チャネル列:$3(13 行目で求めた R10index)とし、Y 軸チャネル列:$2(6 行目で求めたスペクトラム dB 値)として描画します。

<実行結果グラフ>



<補足説明>

グラフの X 軸は周波数を R10Index に変換した値を使用しています。周波数から変換する R10 系列 Index の演算式を示します。

$$R10_i = 3 \cdot \log_2 \frac{f_i}{1000}$$

周波数から R10Index への変換処理は関数 FTI(X)を使用します。$3 = FTI($1)と記述した部分が相当します。つまり、グラフはここで求めた小数点以下を含む R10Index を X 値としスペクトラムを dB 変換した値を Y 値として描画します。plot %2 $3 $2 と記述した部分が相当します。

周波数の最小値と最大値を含む範囲の R10Index 数列生成は関数 RTI(n,m)を使用します。

$6 = RTI($1(0),$1(LEN($1)-1))と記述した部分が相当します。ここで、求めた数列の先頭値がグラフ X 軸左端値、最終値が右端値、数列自体がグリッド線位置を意味します。Script で用いた FFT 解析結果ファイルでは最小周波数が 0.122Hz、最大周波数は 999.878Hz ですので、数列は-40,-39,-38,. . . -3,-2,-1,0 となります。

R10Index と公称周波数との関係は、-40:0.1Hz、-39:0.125Hz、-38:0.16Hz、-37:0.2Hz、-36:0.25Hz、-35:0.315Hz、-34:0.4Hz、-33:0.5Hz、-32:0.63Hz、-31:0.8Hz、-30:1Hz、となっており、1 桁目の絶対値が

0 の時、0.1Hz、1Hz、10Hz、100Hz、1kHz、10kHz、

3 の時、0.5Hz、5Hz、50Hz、500Hz、2kHz、20kHz、

7の時、0.2Hz、2Hz、20Hz、200Hz、5kHz、50kHz、

したがって 1,2,5 ステップで目盛表示するためには、グリッド線位置数列の要素値(R10Index 値)の 1 桁目の絶対値が 0,3,7 の地点に目盛を表示すればよいことになります。$9 = SRT(1,CMP($8,ABS($6)-INT(ABS($6)/10)*10))と記述した部分が相当します。CPM(X,Y)関数は X の要素に一致した値が Y に存在する場合の Y のデータ番号を戻します。$8 の中身は 0.3.7 で構成され、$6 の 1 桁目のみの数列を対象としています。ABS($6)-INT(ABS($6)/10)*10 と記述した部分が$6 を絶対値 1 桁目だけの数列に変換処理している部分に相当します。

この結果、目盛を表示するグリッド線位置の Index(データ番号)を取得したことになります。次に取得した Index(データ番号)から PTV 関数により R10index 値を読出し、その R10index から RTF 関数を使用して公称周波数に変換します。$10 = RTF(PTV($9,$6))と記述した部分が相当します。公称周波数は定義式から求めると-2:629.9605249Hz、-1:793.700526Hz、0:1000Hz、1:1259.92105Hz、2:1587.401052Hz と異なります。この周波数を完全ベース 2 と言いますが、これを前出の様に小数点以下を無くし整理した周波数を R10 公称周波数と言います。

詳細は第 21 章「演算関数を使用して周波数解析を行う」を参照下さい。

記述例11. 2. 両対数グラフを作成する

PcWaveFromFANA で頻度解析した結果格納ファイルのデータを読み出し、S-N 線図グラフを作成します。 表示するデータは頻度数と近似曲線の 2 種類です。なお、演算項目は被害度でグラフサブタイトル行に表示します。

<Archi_1 Script 記述例>

```
1       /*---------graph14.prc-------------------*/
2       def ch_name $1 "Cell_No."
3       def ch_name $2 "Cell 中央値:MPa"
4       def ch_name $3 "count"
5       def ch_name $4 "頻度数"
6       def ch_name $5 "近似線"
7       def ch_name $6 "被害度"
8       def file_id %1 "fana_result" csv
9       read cell %1 11,1 1 $1,$2,$3
10      $4 = $3*1e10+EQU($3,0)*0.01          /* count=0 を 0.01 に置き換える*/
11      $5 = EXP(($2-78.1)/(-2.0412))        /* 近似線演算*/
12      $6 = SUM($4/$5)                      /* 被害度演算*/
13      assign &1 = "S:応力(MPa) N:頻度数 被害度  = "|$6(F5)
14      def graph_id @2 "S-N 線図" &1
15      def graph_y_axis @2 0,0 F3,0,0 2:1 log
16      def graph_x_axis @2 1e10,1e17 E1,0,0 log
17      plot @2 $4,$5 $2,$2
18      end
```

<Archi_1 Script 記述構文の説明>

8 行目～9 行目:PcWaveFormFANA で頻度解析した結果ファイルを読み出します。

8 行目:　ファイル番号定義文で読み出すファイル名を"fana_result"、属性を csvとして定義します。

9 行目:　テキストファイルセル読み出し文で、セル番号、セル中央値、頻度数を読み出します。なお、ファイルのデータ行は 11 行目から開始され、1 列目がセル番号、2 列目がセル中央値、3 列目が頻度数となります。 方法は「12. テキストファイルからの読み出し」項を参照して下さい。

10 行目: 頻度数に被害推定を行うための現実値倍率を掛け、さらにグラフの軸属性が log ため、頻度数 0 を便宜上 0.01 に置き換えます。

11 行目:　近似曲線を演算します。

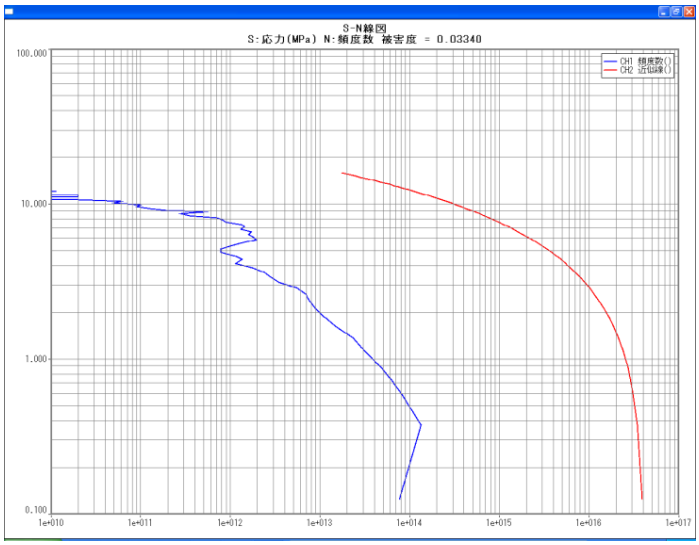
12 行目:　近似曲線から合計被害度を演算します。

15 行目: 定義するグラフ Y 軸定義文でグラフ枠情報を下端値:0、上端値:0(AutoScale)とし、目盛情報を表示形式:F3 とし、凡例位置:2:1(グラフ枠内右上、参照軸を X 軸)とし、軸属性:log として S 軸を定義します。

16 行目: グラフ X 軸定義文でグラフ枠情報を左端値:10e10、右端値:10e17 とし、目盛情報を目盛表示形式:E1 とし、軸属性:log として N 軸を定義します。

17 行目: グラフ描画文で X 軸チャネル列:$4(頻度数)、$5(近似曲線 N 値)とし、Y 軸チャネル列:$2(セル中央値)として描画とします。

<実行結果のグラフ>



12. テキストファイルから読み出す

テキスト形式で格納されているファイルから読み出し開始セル番号を指定して読み出します。読み出し対象テキストファイルの拡張子は問いません。読み出し属性は項目区切り文字半角カンマ、tab または項目区切り文字無視の 3 種類用意されており、それぞれファイル番号定義文の属性設定、csv,txt,chr に対応します。

12. 1. テキストファイル読み出し事前処理

ファイル番号定義文

文法:

def file_id %ファイル番号 ファイル名 属性

属性は項目カンマ区切りの場合は csv、tab 区切りの場合は txt、区切り文字無視の場合は chr を指定します。指定した属性とファイル拡張子が異なる場合は、記述するファイル名に拡張子を付けます。

※ ファイル番号定義文で直接ファイル名を記述せず、Script 実行時に読み出すファイルを選択する方法については、第 5 章「解析対象ファイルを読み出す」の 5.3、5.4 項を参照下さい。

12. 2. テキストファイルサイズ取得処理

読み出すファイルサイズが不明な場合に使用します。既知の場合は特に必要ありません。

ファイル列数読み出し文

文法:

read num_cell %ファイル番号 列数格納先チャネル

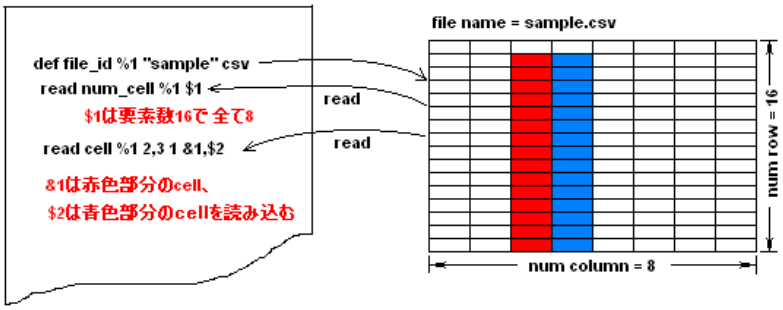
ファイル列数読み出し文での格納先チャネルの要素数は当該ファイルの行数と一致し、要素ごとに当該行の列数が格納されます。行数を取得する場合は、格納先チャネルのデータ個数関数 LEN(X)で求めます。

12. 3. テキストファイルから読み出す テキストファイルセル読み出し文

文法:

read cell %ファイル番号 行列番号 方向フラグ 格納先チャネル列 読み出しセル数

行列番号は、読み出し開始行番号と開始列番号を半角カンマで区切って記述します。方向フラグは、列方向に読み出す場合に 0、行方向に読み出す場合に1とします。格納先チャネル列は読み出し結果を格納するチャネルを意味し、複数チャネル定義する場合は、半角カンマ区切って記述するか連続指定記述則で記述します。複数チャネル記述した場合は、チャネルごとの要素読み出し方向は指定した方向フラグ(0:=列方向、1:=行方向)に従い、チャネル順は、フラグ反転となります。つまり、読み出し方向を行方向(0)とし複数チャネル記述した場合は、チャネルは開始列から順次列方向に記述したチャネル数分読み出します。



読み出しは読み出しセル数記述が省略された場合、指定開始行列番号から指定方向に Cell の存在するまで一度に読み出します(途中に cell が存在しない場合はその直前まで)。但し、格納先が数値属性参照チャネルの場合($n)は数値変換できない Cell の直前までとなります。指定した Cell の内容のみ読み出しする場合は読み出しセル数を定義します。読み出しセル数を 0 とした場合は記述省略した場合と同じ意味を持ちます。なお、読み出しセル数記述が省略されても$2(0)の様に Index を指定し明確に要素数が 1 個であることを指定した場合は指定されたセルのみ読み出します。

ファイル番号定義文で属性 chr を指定した場合、項目区切り文字無視となりますので行ごとに1個の Cell を占有し列数は必ず 1 となりまします。但し、行を指定して読み出す事はできませんので、開始行列番号は必ず 1,1 とし、方向フラグは意味を持ちません。

記述例:

行方向に読み出す場合

項目区切り文字半角カンマ","の csv ファイルの 2 行目から行方向に 1 列目を数値として$1 に格納、2 列目を文字列

として&1 に行の終端まで格納し、同じく 2 行目から行方向に 3 列目を文字列として&2 に、4 列目を数値として$2 に読み出し個数を 4 個として格納する場合

```
def file_id %1 “test_date” csv
read cell %1 2,1 1 $1,&1
read cell %1 2,3 0 &2,$2 4
```

記述例：

項目区切りを無視して行方向に読み出す場合

項目区切り半角カンマ”,”のcsvファイルから項目区切りを無視し(項目区切り文字を含めて)1 行ごとに文字列として&1 に格納する場合

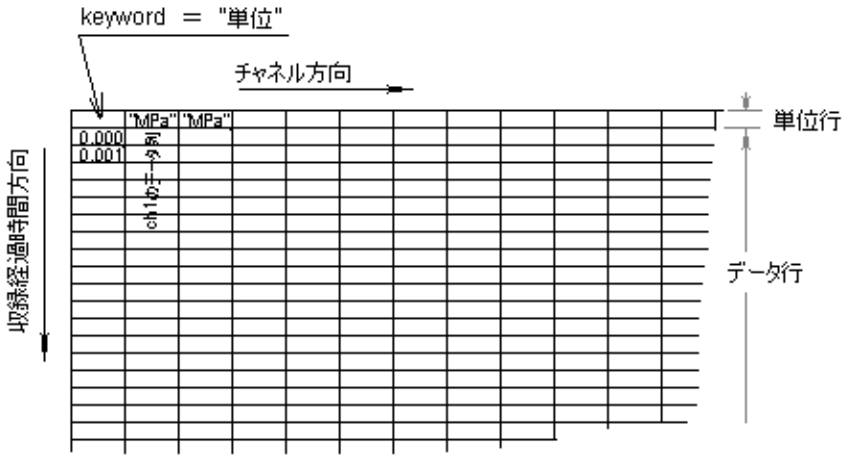
```
def file_id %1 “test.csv” chr
read cell %1 1,1 1 &1
```

※ 行を項目ごとに分離する場合、セパレータ(区切り文字)コードを指定して、文字列分離関数 CSEP(k,&m)を使用して分離する事が可能です。例えば&1 の 3 行目を半角カンマ区切りで分解する場合は、CSEP(1,&1(2))と記述します。但し、複数行を一度に分離することはできません。

記述例12. 1. テキスト形式収録ファイルを読み出し波形ファイルに変換生成する

項目区切り文字半角カンマ","の拡張子.csv のテキスト形式で収録されたファイルを読み出し、PcWaveForm 標準形式の波形ファイルに変換格します。

読み出す収録ファイルは、1 行目が単位行で 1 行 1 列目はキーワードで"単位"、2 列目以降が各チャネルの単位となります。2 行目からがデータ行となります。1 列目はサンプリング経過時刻(秒単位)列、2 列目からチャネルごとのデータが記録されています。読み出す方向は、単位行を 1 行 2 列目から列方向に文字属性チャネルに読み出し、各チャネルのデータは 2 行 2 列目から行方向に数値属性チャネルに読み出します。
なお、波形ファイル生成に当たって各チャネルのデータ数は一致しているかを確認しています。



<Archi_1 Script 記述例>

```
1        /*----- text_read1.prc----------*/
2        def ch_name &1 "unit"
3        def ch_name $1 "num_ch"
4        def ch_name $2 "filecheckflag"
5        def ch_name $3 "sampling"
6        def ch_name $4 "ch_series"
7        /*------ file assign & get size ----------*/
8        def file_id %1 "sample" csv
9        read num_cell %1 $1
10       $1 = ERC(1,LEN($1)-1,$1)
11       $2 = EQU(LEN($1),SUM(EQU($1(0),$1))) /* file format check */
12       case_true $2
13         proc file_create{
14           $1 = $1(0)-1   /* file num_ch */
15           /*----- unit row read ----------*/
16           read cell %1 1,2 0 &1
17           &1 = CDEL(-1,1,CDEL(1,1,&1))   /* 前後"文字削除*/
18           /*------sample column read ----------*/
19           read cell %1 2,1 1 $3
20           $3 = $3(1)-$3(0)                /* サンプル周期取得*/
21           /* ----- data read & wav file create ----------*/
22           read cell %1 2,2 1 $[100,$1]
23           $4 = ACC(DAG($1,1))+99         /* ch 番号生成*/
24           def ch_unit $($4) &1           /* 単位定義*/
25           def file_id %2 "sample" wav
26           def wav_sampling %2 1 $3 "sec" 0
27           save wave %2 $[100,$1]
28         }file_create
29       case_false $2
30         proc cannot_proc{
31           disp message "File データに欠損あり、変換できません!!"
32         }cannot_proc
33       end
```

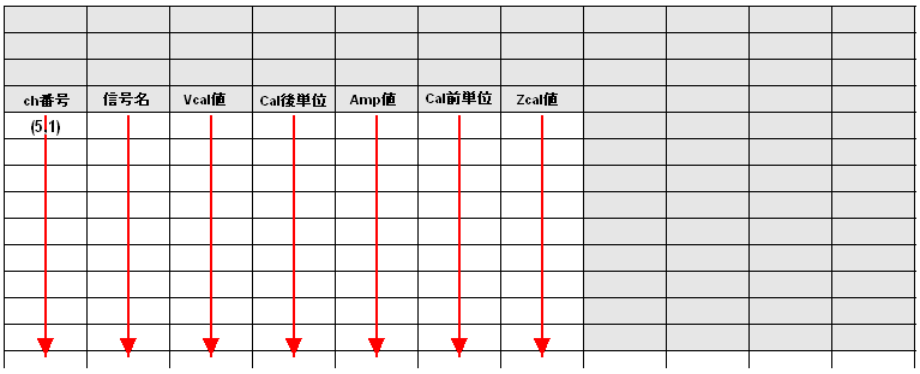
<Archi_1 Script 記述構文の説明>

8 行目:　ファイル番号定義文で読み出すファイル名を"Sample"とし属性を csvとして定義します。
9 行目:　ファイルの行ごとの列数を取得します。
10 行目: 先頭行が単位行となりますので、データのチャネル数と不整合でも良いものとするため、各行の列数確認から除外します。
11 行目: データ行の全てのチャネルでデータ欠損が存在していないかの確認演算を行います。
14 行目: 収録チャネル数を演算します。
16 行目: 単位行を読み出します。読み出し開始行列番号を 1 行 2 列目とし、読み出し方向を列方向として文字属性参照チャネル &1 に読み出します。
17 行目: 読み出した単位文字列前後に付加されているダブルコーテーション"文字を削除処理します。この処理に使用したサンプルデータが単位文字列表記にダブルコーテーションで囲んであり、PcWaveForm 波形ファイルとして生成する場合にダブルコーテーションは不要となるためです。
19 行目: 経過時間列を読み出します。読み出し開始行列番号を 2 行 1 列目とし読み出し方向を方向として数値属性参照チャネル $3 に読み出します。
20 行目: 等時間サンプルと見なし、Index1-Index0 からサンプリング周期を演算します。
22 行目: 読み出し開始行列番号を 2 行 2 列目とし、読み出し方向を行方向としてデータを読み出します。なお、データ格納先チャネルは数値属性参照チャネル$100 からチャネル数分連続指定記述則で記述します。
23 行目: チャネル$100 から収録チャネル数分のチャネル番号に読み出した単位を定義するためのチャネル番号を生成します。
24 行目: データ格納先チャネルに読み出した単位を定義します。
25 行目: 生成格納する波形ファイル番号を定義します。
26 行目: 生成する波形ファイルのサンプリング周期を定義します。
27 行目: 波形ファイルを格納します。

記述例12. 2. テキスト形式 CAL ファイルを読み出しキャリブレーションする

PcWaveForm で収録したデータを物理量に変換するための CAL ファイルを読み出し、物理量変換した波形ファイル生成します。

PcWaveForm で使用する CAL ファイルは拡張子.cal で項目区切り文字は tab のテキスト形式です。
読み出す内容は 5 行目からで項目ごとに行方向となります。パラメータ内容は、5 行 1 列目がチャネル番号列(数値属性)、2 列目が CAL 後の信号名列(文字属性)、3 列目が Vcal 値列(数値属性)、4 列目が CAL 後の単位列(文字属性)、5 列目が Vcal 値に対応した計測値(Amp 値)(数値属性)、6 列目が計測単位(Amp 単位)(文字属性)、7 列目がオフセット値列(ZeroCal)(数値属性として記録されています。



キャリブレーションは、解析対象ファイル(収録ファイル)とCAL ファイルを読み出し、解析対象ファイルのチャネル番号とCal 前単位が一致したチャネルのみ行い、CAL ファイルに存在しないチャネルや存在しても Cal 前単位が一致しないチャネルはキャリブレーションを行わずそのまま格納します。キャリブレーション演算は次式によります。

Slope = Vcal 値/Amp 値　　Offset = Zcal値 キャリブレーション該当チャネル = Slope×収録チャネル+Offset

<Archi_1 Script 記述例>

```
1     /*----- text_read2.prc----------*/
2     dcl sheet 1 {
3       page 1:
4       column   $7,&6,&5,$12,$13
5      format F0,A,A,F3,F2 2
6     }sheet
7     def ch_name &2 "cal_s_name"
8     def ch_name &3 "cal_s_unit"
9     def ch_name &4 "cal_a_unit"
10    def ch_name $3 "cal_ch_series"
11    def ch_name $4 "V_cal"
12    def ch_name $5 "Amp"
13    def ch_name $6 "Z_cal"
14    /* ------ acq file read ----------*/
15    def file_id %1 "brake_test_data" wav
16    read wave %1
17    $7 "acq_ch_series:" = CHS()      /* acq file ch_series*/
18    $8 "acq_num_ch:" = NCH()         /* acq file num_ch*/
19    &5 "acq_ch_unit:" = CUNT(0)    /* acq file ch_unit*/
20    &6 "acq_ch_name:" = CHNM(0)     /* acq file ch_name*/
21    /*------ file assign & get size ----------*/
22    def file_id %2 "brake_test.cal" txt
23    read cell %2 5,1 1 $3,&2,$4,&3,$5,&4,$6 /* Cal パラメータ読み出し*/
24    &2 = CDEC(1," ",&2)              /* Cal 後:信号名前後スペース削除*/
25    &3 = CDEC(1," ",&3)              /* Cal 後:単位前後スペース削除*/
26    &4 = CDEC(1," ",&4)              /* Cal 前:単位前後スペース削除*/
27    /*----- cal check ----------*/
28    $9 "Acq_Cal_ch:" = CMP($3,$7) /*Cal⇒Acq Ch 存在データ番号*/
29    case $9 <> -1
30      proc ch_exist{
31           $10 "Cal_Acq_ch:" = CMP($7,$3) /*Acq⇒Cal Ch 存在データ番号*/
32        $12 "Cal_Slope:" = DAG($8,1)    /*Cal Slope init*/
33        $13 "Cal_Offset:" = DAG($8,0)    /*Cal Offset init*/
34        $11 "matchindex:" = CEQ(CPTV($9,&5),CPTV($10,&4))   /* 単位一致確認*/
35        $9 = ZSP($11,$9)              /* 単位一致データ番号再構成*/
```

```
36              $10 = ZSP($11,$10)          /* 単位一致データ番号再構成*/
37              &6 = CERP($9,CPTV($10,&2),&6)           /* 信号名変更*/
38              &5 = CERP($9,CPTV($10,&3),&5)           /* 単位変更*/
39              $12 = SBV($9,PTV($10,$4)/PTV($10,$5),$12) /* Cal Slope再構成*/
40              $13 = SBV($9,PTV($10,$6),$13)   /* Cal Offset 再構成*/
41              write ch_column 1: $7,&6,&5,$12,$13
42              /*---- cal exec ----------------------------------*/
43              $1 "acq_ch_counter:" = 0
44              repeat_case $1 < $8
45                proc cal_exec{
46                  $2 "create_ch:" = $1+100
47                  def ch_name $($2) &6($1)
48                  def ch_unit $($2) &5($1)
49                  $($2) = $11($10)*#($7($1))+$13($1)
50                  $1 = $1+1
51                }cal_exec
52              def file_id %3 "brake_test_cal" wav
53              def wav_ch_series %3 $7
54              save wave %3 $[100,$8]
55            }ch_exist
56          case $9 = -1
57            proc ch_not_exist{
58              disp message "cal file に収録 ch が存在しません"
59            }ch_not_exist
60          end
```

<Archi_1 Script 記述構文の説明>

15 行目： 解析対象ファイル番号を定義します。16 行目は解析対象ファイルを読み出します。
17 行目～20 行目：解析対象ファイルのチャネル番号、収録チャネル数、解析対象ファイルに付けられていた信号名および単位を取得します。
22 行目： CAL ファイル番号を定義します。CAL ファイルは項目 tab 区切りのテキスト形式ですが拡張子は.calのため、ファイル番号定義に際してファイル名は拡張子を含んで記述し属性を txtとして定義します。
23 行目： CAL ファイルから cal パラメータを読み出します。
24 行目： 26 行目は、CAL ファイルの文字属性列はキャリブレーションプログラムでの表示都合上信号名あるいは単位の文字数に関係なく、スペース文字を付加して固定長の文字列としています。そのため、不要なスペースを除去処理します。
28 行目～40 行目：キャリブレーション事前処理で、解析対象ファイルのチャネル番号と CAL ファイルのチャネル番号が同じで Cal 前単位が一致したチャネルのみ行い、CAL ファイルに存在しないチャネルや計測単位が一致しないチャネルはキャリブレーションを行わない処理を行います。
28 行目： CAL ファイルに記載されているチャネルが解析対象ファイルに存在している Index を求めます。
29 行目： 存在していない場合、-1 が戻り値となりますのでキャリブレーションを行わない様に直下の演算処理ブロックを制御します。
31 行目： 28 行目と反対に解析対象ファイルのチャネルが CAL ファイルに存在している index を求めます。
32 行目～ 33 行目：キャリブレーションするための Slope と Offset の係数を初期化して準備します。
34 行目： チャネル番号が一致したチャネルの Cal 前単位を取得し、同じであるか否かをチェックします。
35 行目：～36 行目：Cal 前単位が一致していない Index を除去します。
37 行目： キャリブレーションするチャネルの信号名をキャル後の信号名に変更します。
38 行目： 同様にキャリブレーションするチャネルの単位を Cal 後の単位に変更します。
39 行目： キャリブレーションするチャネルの Slope 値を Vcal 値と Amp 値で変更します。
40 行目： キャリブレーションするチャネルの Offset 値を Zcal 値に変更します。
43 行目～52 行目：キャリブレーション実行処理で収録チャネル数分繰り返して各チャネルのキャリブレーションを実行します。
46 行目： 格納先チャネル番号を生成します。
47 行目： 生成した格納先チャネルに信号名を定義します。
48 行目： 同様に生成した格納先チャネルに単位を定義します。
49 行目： 収録チャネルに Slope を掛け Offsetを加算し物理量に変換します。
50 行目： 繰り返し Loop カウンタをインクリメントします。
52 行目～54 行目：キャリブレーション済みを波形ファイルとして格納します。
53 行目： 格納するチャネルを元の収録チャネルと同じチャネル番号に定義します。
54 行目： 波形ファイル格納します。

<実行の結果表示される結果シート表示内容(<キャリブレーションしたパラメータを表示)>

Page1:

acq_ch_series	acq_ch_name	acq_ch_unit	Cal_Slope	Cal_Offset
3		V	1.000	0.00
4		km/h	40.128	0.00
5		N	0.075	0.00
6		Degree	1.000	0.00
7		Degree	1.000	0.00
8		km/h	31.898	0.00
9		N	0.094	0.00
10		Degree	1.000	0.00
11		km/h	30.798	0.00
12		V	1.000	0.00

<補足説明>

ヘッダーファイルのチャネル番号/単位と CAL ファイルのチャネル番号/CAL 前単位が一致したチャネルのみキャリブレーションを行うためには、大きさの異なる配列同士の一致を確認する必要があります。

① CAL ファイルのチャネルが収録チャネル番号配列に存在しているか？

$9 = CMP($3,$7)

$3 は CAL ファイルのチャネル番号配列、$7 は収録ファイルチャネル番号配列で、CMP()関数は$3 が$7 に存在した(一致した)$7(収録チャネル番号配列)の Index を$9 に求めます。

② 収録チャネルが CAL ファイルのチャネル番号配列に存在しているか？

$10 = CMP($7,$3)

同様に$3(CAL ファイルのチャネル番号配列)の Index を$10 に求めます。 何れのチャネル番号配列のチャネル番号は唯一無二であり、昇順並びであることを前提としています。また、$9、$10 は何れも 複数要素を持ち、個数は一致します。

③ ①、②項で求めた Index 配列を使用してそれぞれの単位が一致している論理数列を求めます。

$11 = CEQ(CPTV($9,&5),CPTV($10,&4))

&5 は収録チャネルの単位配列、&4 は CAL ファイルの CAL 前単位配列で、CPTV()は Index から要素値に変換する演算関数を使用して、一致したチャネル番号の Index から配列要素となる単位を抽出し、CEQ()は文字列の比較関数を使用して一致した Index を"1"、一致しない Index を"0"とした論理数列を$11 に求めます。

④ 論理数列$11 をキーとして、$9,$10 を更に単位が一致している Index だけに縮退します。

$9 = ZSP($11,$9)
$10 = ZSP($11,$10)

ZSP()関数は$11 の論理"1"の要素だけを抽出します。ZSP($11,$9)はチャネル番号が一致した収録チャネル番号 Index 配列から更に単位も一致した収録チャネル番号 Index だけ再構成し、同様に ZSP($11,$10)はチャネル番号が一致した CAL チャネル番号 Index 配列から更に単位も一致した CAL ファイルチャネル番号 Index だけに再構成します。

⑤ 信号名配列、単位配列、Slope 値数列、Offset 数列を一致したチャネル/単位番号部分を置き換えます。

&6 = CERP($9,CPTV($10,&2),&6)
&5 = CERP($9,CPTV($10,&3),&5)
$12 = SBV($9,PTV($10,$4)/PTV($10,$5),$12)
$13 = SBV($9,PTV($10,$6),$13)

&6 は信号名配列、&2 は CAL ファイルの CAL 後信号名配列、&5 は単位配列、&3 は CAL ファイルの CAL 後単位配列で CERP()関数は文字列要素の指定 Index 位置の置き換え関数でそれぞれ更新します。$4 は CAL ファイルの VCAL 数列、$5 は CAL ファイルの VCAL 相当計測値、$12 は Slope 数列、$6 は CAL ファイルの Offset 値で、SBV()関数は指定 Index 位置の置き替え関数で、それぞれの Slope 数列および Offset 数列を更新します。

記述例12. 3. 解析対象ファイルのヘッダーファイル(拡張子.hdr)から収録情報を読み出す

解析対象ファイルのヘッダーファイルを全表示し、選択したヘッダーファイルから収録チャネル数、サンプリング周波数、収録データ数の取得した情報を結果シートに表示します。

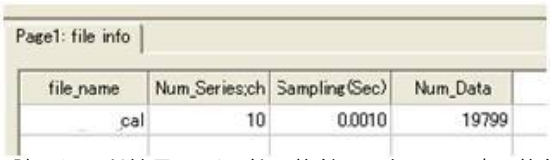
<Archi_1 Script 記述例>

```
1        /*------text_read3.prc----------*/
2        dcl sheet 1 {
3          page 1: "file info"
4            column &1,$7,$5,$6
5            format A,F0,F4,F0 2
6        }sheet
7        def ch_name &1 "file_name"
8        def ch_name $5 "Sampling"
9        def ch_name $6 "Num_Data"
10       def ch_name $7 "Num_Series:ch"
11       get folder_select   /*folder 選択*/
12       read file_info &1 &2 &3 $1 .hdr /* file 情報取得*/
13       $2 = 0
14       repeat_case $2 < $1
15         proc header_read {
16           assign &4 = &1($2)|".hdr"
17           def file_id %1 &4 chr     /* file 番号定義*/
18           $4 = $2+100
19           read cell %1 1,1 1 &($4)   /*file 読み出し*/
20           $2 = $2+1
21         }header_read
22       $3 = 1
23       get text_page &[100,$1] $3   /* file 選択*/
24       $5 = 1/CTN(CDEL(5,&(100+$3-1)(CMP(1,CFND(1,"RATE ",&(100+$3-1))))))
25       &2 = CDEL(11,&(100+$3-1)(CMP(1,CFND(1,"HORZ_UNITS ",&(100+$3-1)))))
26       def ch_unit $5 &2
27       $6 = CTN(CDEL(10,&(100+$3-1)(CMP(1,CFND(1,"NUM_SAMPS ",&(100+$3-1))))))
28       $7 = CTN(CDEL(11,&(100+$3-1)(CMP(1,CFND(1,"NUM_SERIES ",&(100+$3-1))))))
29       $3 = $3-1
30       assign &1 = &1($3)
31       write ch_column 1: &1,$5,$6,$7
32       end
```

<Archi_1 Script 記述構文の説明>

11 行目: 解析対象ファイルの格納先フォルダを選択します。
12 行目: 選択したフォルダに格納されている解析対象ファイルのヘッダーファイル情報を取得します。
13 行目～21 行目:取得したファイルを参照して格納されているすべてのヘッダーファイルを読み出します。
16行目: 12 行目で取得したファイル名には拡張子が付いていないため、ファイル番号定義に当たって拡張子を付加します。
17 行目: 読み出すファイル番号を属性 chr で定義します。
18 行目: 読み出したテキストファイルの格納先チャネルを生成します。
19 行目: テキストファイルを読み出し 18 行目で生成した文字属性参照チャネルに格納します。
23 行目: 格納した全てのテキストを表示し、ファイルを選択します。
24 行目: RATE 行を検索し、サンプリング周波数を取得し数値に変換後更に周期に変換します。
25 行目: HORZ_UNITS 行を検索し、サンプリング単位を取得します。
26 行目: 表示用にサンプリング周期チャネルにサンプリング単位を定義します。
27 行目: NUM_SAMPS 行を検索して収録データ数を取得します。
28 行目: NUM_SERIES 行を検索して収録チャネル数を取得します。
30 行目: 選択されたファイル名をファイル名配列から取得します
31 行目: 結果シートにファイル名、収録チャネル数、サンプリング周期、および収録データを書き込みます。

<実行の結果表示される結果シート>

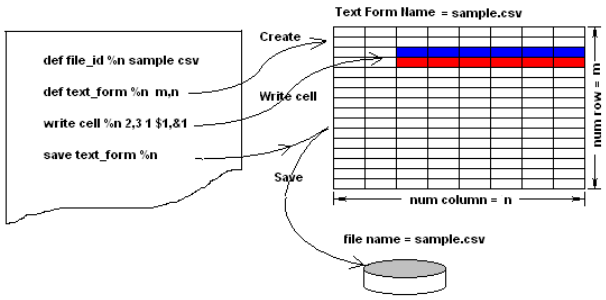


属性 chr で読み込んだ結果は、行ごとに格納チャネルの要素に格納されます。

13. テキストファイルを生成する

テキストファイルを生成する場合、テキストフォーム(仮想シート)を定義しテキストフォーム上に書き込みを行った後、そのテキストフォームを格納する方法で生成します。生成するテキストファイルの項目区切り文字は半角カンマの csv 形式または tab 区切りの txt のいずれかとなります。なお拡張子は問いません。

テキストファイル生成操作概念図



13. 1. テキストファイル生成事前処理

ファイル番号定義文

文法:

def file_id %ファイル番号 ファイル名 属性

属性は項目カンマ区切りの場合は csv、tab 区切りの場合は txtの 2 種となります。指定した属性とファイル拡張子が異なる場合は、記述するファイル名に拡張子を付けます。

13. 2. テキストフォームサイズの定義

テキストファイルを生成する場合、直接ファイルに書き込むのではなく、一旦、仮想シート(テキストフォームと呼称)を定義し、そのテキストフォームに書き込み、書き込まれたテキストフォームを格納する事で生成します。

テキストフォーム行列数定義文

文法:

def text_form %ファイル番号 行列数

テキストフォームの大きさを定義します。テキストフォームの大きさは書き込むであろう最大行最大列で定義します。つまり後述するテキストフォーム格納文はテキストフォームに書き込まれている最大行最大列に切り取られて格納されますので適当に大きくして定義します。

13. 2. テキストフォームへの書き込み操作

テキストフォーム書き込み文 文

法:

write cell %ファイル番号 行列番号 方向フラグ 書き込みチャネル列

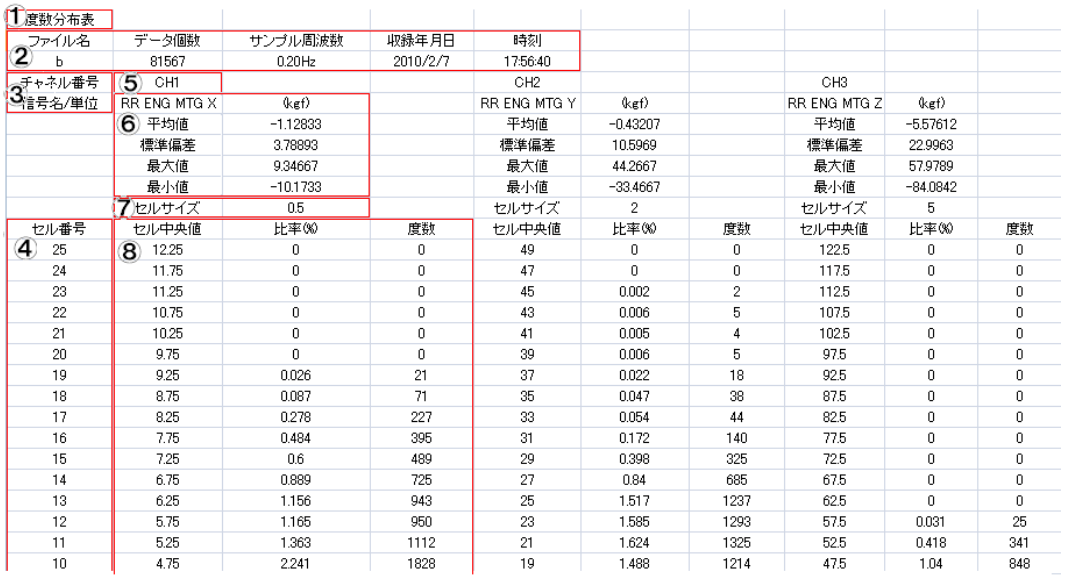
行列番号は、書き込み開始行番号と開始列番号を半角カンマで区切って記述します。方向フラグは書き込み方向設定する機能と書き込み内容を設定する複合機能を有します。なお、テキストフォーム行列定義文で定義した行列数を越えた範囲は書き込まれません。

値	機能
0	チャネルの要素を横方向(列方向)、チャネルを縦方向(行方向)にデータのみ書き込む
1	チャネルの要素を縦方向(行方向)、チャネルを横方向(列方向)にデータのみ書き込む
2	チャネルの信号名を横方向(列方向)に書き込む、信号名の無いデータ並びは空欄
3	チャネルの信号名を縦方向(行方向)に書き込む、信号名の無いデータ並びは空欄
4	チャネルの単位を横方向(列方向)に書き込む、単位の無いデータ並びは空欄
5	チャネルの単位を縦方向(行方向)に書き込む、単位の無いデータ並びは空欄
6	書き込み方向はフラグ 4 と同じですが、信号名と単位を連結して書き込む
7	書き込み方向はフラグ 5 と同じですが、信号名と単位を連結して書き込む
8	書き込み方向はフラグ 0 と同じですが、文字列チャネルに含む区切り文字を無視
9	書き込み方向はフラグ 1 と同じですが、文字列チャネルに含む区切り文字を無視

書き込み方向フラグ 2～5 は、書き込みデータ列をチャネル記述で行った時に有効で、即値記述は無視され当該チャネルに付けられている信号名および単位を書き込みます。なお、記述されたチャネルが複数要素であっても書き込みセルは 1 個となります。

記述例:

書き込むテキスト･フォームは仮想シート構造を持ちます。仮想シートは事前に書き込む cell 及び方向を決定して書き込みます。下記に示すチャネルごとの度数分布表を作成する場合、①～④はフォーム全体で一回、⑤～⑧はチャネルごとに書き込む事にします。



①度数分布表										
ファイル名	データ個数	サンプル周波数	収録年月日	時刻						
②b	81567	0.20Hz	2010/2/7	17:56:40						
③チャネル番号	⑤CH1			CH2			CH3			
信号名/単位	RR ENG MTG X	(kgf)		RR ENG MTG Y	(kgf)		RR ENG MTG Z	(kgf)		
	⑥平均値	-1.12833		平均値	-0.43207		平均値	-5.57612		
	標準偏差	3.78893		標準偏差	10.5969		標準偏差	22.9963		
	最大値	9.34667		最大値	44.2667		最大値	57.9789		
	最小値	-10.1733		最小値	-33.4667		最小値	-84.0842		
	⑦セルサイズ	0.5		セルサイズ	2		セルサイズ	5		
セル番号	セル中央値	比率(%)	度数	セル中央値	比率(%)	度数	セル中央値	比率(%)	度数	
④25	⑧12.25	0	0	49	0	0	122.5	0	0	
24	11.75	0	0	47	0	0	117.5	0	0	
23	11.25	0	0	45	0.002	2	112.5	0	0	
22	10.75	0	0	43	0.006	5	107.5	0	0	
21	10.25	0	0	41	0.005	4	102.5	0	0	
20	9.75	0	0	39	0.006	5	97.5	0	0	
19	9.25	0.026	21	37	0.022	18	92.5	0	0	
18	8.75	0.087	71	35	0.047	38	87.5	0	0	
17	8.25	0.278	227	33	0.054	44	82.5	0	0	
16	7.75	0.484	395	31	0.172	140	77.5	0	0	
15	7.25	0.6	489	29	0.398	325	72.5	0	0	
14	6.75	0.889	725	27	0.84	685	67.5	0	0	
13	6.25	1.156	943	25	1.517	1237	62.5	0	0	
12	5.75	1.165	950	23	1.585	1293	57.5	0.031	25	
11	5.25	1.363	1112	21	1.624	1325	52.5	0.418	341	
10	4.75	2.241	1828	19	1.488	1214	47.5	1.04	848	

```
write cell %1 1,1 0 "度数分布表" /*1 行 1 列目からチャネル縦方向、要素横方向に①書き込み*/
assign &1 = "ファイル名","データ個数","サンプル周波数","収録年月日","時刻"
assign &2 = &1(0),$1(F0),$2(F2)|"Hz",&2(0),&3(0)
write cell %1 2,1 0 &1,&2     /*2 行 1 列目からチャネル縦方向、要素横方向に②書き込み*/
write cell %1 4,1 1 "チャネル番号","信号名/単位" /*4 行 1 列目からチャネル横方向、要素縦方向に③書き込み*/
assign &3 = "セル番号",$10(F0)
write cell %1 11,1 1 &3       /*11 行 1 列目からチャネル横方向、要素縦方向に④書き込み*/
$20 = 2
assign &4 = "CH"|$3(0)(F0)
write cell %1 4,$20 0 &4      /*4 行$20 列目からチャネル縦方向、要素横方向に⑤書き込み*/
assign &5 = &4(0),"平均値","標準偏差","最大値","最小値"
assign &6 = &5(0),$4(F6)
write cell %1 5,$20 1 &1,&2 /*5 行$20 列目からチャネル横方向、要素縦方向に⑥書き込み*/
assign &7 = "セルサイズ",$5(F1)
write cell %1 10,$20 0 &7 /*10 行$20 列目からチャネル縦方向、要素横方向に⑦書き込み*/
assign &8 = "セル中央値",$6(F2)
assign &9 = "比率(%)",$7(F3)
assign &10 = "度数",$8(F0)
write cell %1 11,$20 1 &8,&9,&10 /*11 行$20 列目からチャネル横方向、要素縦方向に⑧書き込み*/
```

※ &1:ファイル名、$1：データ個数、$2:サンプリング、&2：収録年月日、&3:時刻、$10:セル番号が格納されていることを前提としています。

13. 3. テキストフォームの格納

テキストフォーム格納文

文法:

```
save text_form %ファイル番号
```

テキストフォームに書き込まれた内容をファイル番号定義文の属性で指定された属性で格納します。

記述例:

ファイル名"text"とし csv 形式で格納する

```
def file_id %1 "text" csv
assign $1 = 1,2,3
assign $2 = 4,5,6
assign $3 = 7,8,9
def text_form %1 7,3                    /* 7 行 3 列のテキストフォームを定義*/
write cell %1 1,1 1 $1(F0),$2(F0),$3(F0)    /* 1 行 1 列目から列方向に$1,$2,$3 を書き込む*/
write cell %1 5,1 0 $1(F0),$2(F0),$3(F0)    /* 5 行 1 列目から行方向に$1,$2,$3 を書き込む*/
save text_form %1                       /* 書き込んだテキストフォームを格納*/
```

記述例13. 1. 温度換算表テキストファイルを作成する

摂氏から華氏及びケルビン温度への換算表の scv ファイルを作成します。

<Archi_1 Script 記述例>

```
1        /*--- textwrite1.prc----------*/
2        assign &1 = "F = C×9/5-32","C = (F-32)×5/9"," K = C-273.15"
3        $1 "Celsius:" = (ACC(DAG(46,1))-1)*2-20
4        $2 "Fahrenheit:" = $1*9/5-32
5        $3 "Kelvin:" = $1+273.15
6        def file_id %1 "temperature" csv
7        def text_form %1 60,5
8        write cell %1 1,2 0 "温度換算表"
9        write cell %1 2,1 0 &1
10       write cell %1 4,1 1 "C(celsius)","F(fahrenheit)","K(kelvin)"
11       write cell %1 5,1 1 $1(F1),$2(F2),$3(F2)
12       save text_form %1
13       end
```

<Archi_1 Script 記述構文の説明>

2 行目:　換算式を文字列で生成します。
3 行目:　摂氏温度数列を-20℃～70℃範囲 2℃刻みの数列を生成します。
4 行目:　摂氏から華氏へ換算します。
5 行目:　摂氏からケルビン温度へ換算します。
6 行目～12 行目:テキストフォーム作成･書き込み･格納処理を行います。
6 行目:　ファイル番号定義文で属性 csv とします。
7 行目:　テキストフォーム定義文で、行数 60、列数 5 で定義します。
8 行目:　テキストフォームの 1 行 2 列目に表題を書き込みます。
9 行目:　テキストフォームの 1 行 2 列目に 2 行目で生成した表題を書き込みます。
10 行目: 換算表の列ごとの表題をテキストフォームの 4 行 1 列目から列方向に書き込みます。
11 行目: 3 行目～5 行目で生成した換算表をテキストフォームの 5 行 1 列目から行方向 書き込みます。
13 行目: テキストフォームを 6 行目で定義したファイル名属性で格納します。

<実行結果で生成されたテキストファイル>

エクセルで読み出した場合は次の様になります。

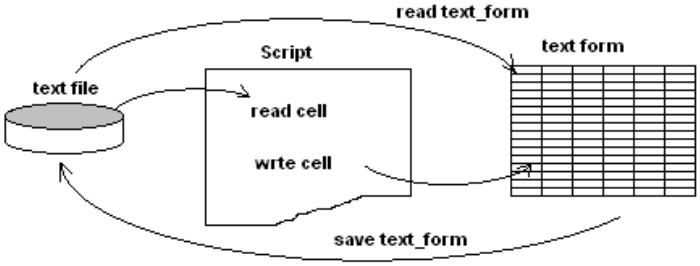
	A	B	C	D
1		温度換算表		
2	F = C×9/5-32	C = (F-32)×5/9	K = C-273.15	
3				
4	C(celsius)	F(fahrenheit)	K(kelvin)	
5	-20	-68	253.15	
6	-18	-64.4	255.15	
7	-16	-60.8	257.15	
8	-14	-57.2	259.15	
9	-12	-53.6	261.15	
10	-10	-50	263.15	
11	-8	-46.4	265.15	
12	-6	-42.8	267.15	
13	-4	-39.2	269.15	
14	-2	-35.6	271.15	
15	0	-32	273.15	
16	2	-28.4	275.15	
17	4	-24.8	277.15	
18	6	-21.2	279.15	
19	8	-17.6	281.15	
20	10	-14	283.15	
21	12	-10.4	285.15	

14. テキストファイルを読み書きする

あらかじめ作成されているテキストファイルを読み込み、追記編集して再び格納したり、他のファイルから必要部分をコピーして新たなファイルを生成したりする事ができます。

14. 1. 読み出しファイルから読み出し追記修正して更新する場合

テキストファイルからテキストフォームに直接読み出し、テキストフォームセル書き込み文で書き込みした結果を格納します。



使用する構文と概略手順:

①読み出し先ファイル番号を定義します

def file_id %ファイル番号 1 ファイル名 属性

②読み出し先ファイルの列数を取得します。(行数は戻り列数の個数)

read num_cell %ファイル番号 1 列数格納先チャネル

③テキストフォームを読み出した行列数に追加する行分大きく定義します。

def text_form %ファイル番号 1 行列数

④ファイルからテキストフォームに読み出します。

read text_form %ファイル番号 1

⑤テキストフォームに追記します。

write cell %ファイル番号 1 行列番号 方向フラグ 書き込みチャネル列

⑥テキストフォームを格納します。

save text_form %ファイル番号 1

テキストフォーム行列数定義文を記述しないでテキストフォームに読み出した場合、テキストフォームのサイズは読み出したフ ァイル番号と同じファイル番号で行列数は読み出したファイルの最大行、最大列となります。言い換えれば行の追加や列の 追加を行う事ができず、Cell 単位での変更のみ行えます。テキストフォーム行列数定義文で明示的にテキストフォームの大き さを定義した場合は、読み出したファイルは定義した大きさの範囲で書き込まれます。読み出すテキストファイルより大きくサ イズを定義する事により行の追加書き込みや列の追加書き込みが行えます。このファイル番号からテキストファイルセル読み 出し文(read cell %n)を実行すると読み出し先はファイルから直接読み出します。また、同様に同じファイル番号でテキストフォ ーム書き込み文(write cell%n)を実行するとテキストフォームに書き込まれます。但し、書き込み指定した行列番号が存在し ない場合は書き込まれません。すでに存在している行列番号に書き込みすると書き込み先 cell の内容が更新されます。なお、 テキストフォーム格納文(save text_form %n)を実行すると、読み込み先ファイルに上書き保存されます。

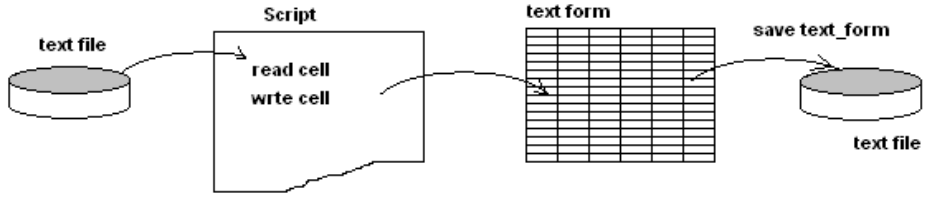
記述例:

読み出したファイルに 1 行追加する

```
def file_id %1 “ファイル名” csv          /*更新するファイル名を定義する*/
read num_cell %1 $1                      /*ファイルサイズを取得する*/
$2 “現在行数:” = LEN($1)                 /* 読み出し先ファイルの行数を取得する*/
$3 “現在列数:” = MAX($1)                 /* 読み出し先ファイルの最大列数を取得する*/
$4 “新規行数:” = $2+100                  /* 100 行追加する*/
$5 “新規列数” = $3+5                     /* 5 列追加する*/
def text_form %1 $4,$3                   /* 仮想シートを定義する*/
read text_form %1                        /* 更新するファイルを仮想シートへ読み出す*/
$4 = $2+1                                /* 新規追加行番号を生成する*/
write cell %1 $4,1 1 “abcd”              /* 行番号:最終行+1、列番号:1 に abcd を書き込む*/
save text_form %1                        /* 仮想シートを更新ファイルに上書きする*/
```

上書きされたファイルの大きさは、元の行数+1 となり、列数は変わりません。

14. 2. 読み出し先ファイルから別のテキストファイルに書き込む場合 読み出し先テキストファイル番号/ファイル名と格納先ファイル番号/ファイル名を異なった内容で定義し読み出し先テキスト ファイルからテキストファイルセル読み出し文で読み込んだ内容を格納先ファイル番号定義したテキストフォームに書き込み 格納します。



使用する構文と概略手順:

①読み出し先ファイル番号を定義します。

def file_id %ファイル番号1 ファイル名1 属性

②読み出し先ファイルから生成するファイルに書き込む内容を読み出します。

read cell %ファイル番号1 行列番号 方向フラグ 格納先チャネル列

③生成するファイル番号を定義します。

def file_id %ファイル番号2 ファイル名2 属性

④テキストフォームを定義します。

def text_form %ファイル番号2 行列数

⑤先に読み出した内容をテキストフォームに書き込みます。

write cell %ファイル番号2 行列番号 方向 書き込みチャネル列

⑥書き込まれたテキストフォームを格納します。

save text_form %ファイル番号2

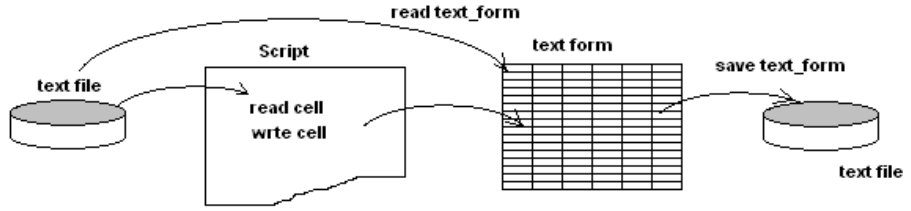
読み出し先ファイル番号/ファイル名と書き込み先ファイル番号/ファイル名は異なっている必要があります。同じファイル番号、ファイル名では実質的に 14.1.項と変わりません。 テキストフォームは自動的に生成されませんので書き込み先ファイル番号で定義します。この事により、テキストファイルセル 読み出し文は読み出し先ファイルから読み出し、テキストフォームセル書き込み文は定義したテキストフォームに対して行わ れ、テキストフォーム格納文は書き込まれたテキストフォームを格納します。

記述例:

読み出したファイルの先頭から 10 行を別のファイルにする

```
def file_id %1 “読み出し先ファイル” csv          /*読み出し先ファイル名を定義する*/
def file_id %2 “格納先ファイル” csv              /* 格納先ファイル名を定義する*/
read num_cell %1 $1                              /* 読み出しファイルの列数を取得する*/
$1 = MAX($1)                                     /* 最大列数を演算する*/
def text_form %2 10,$1                           /* テキストフォームを定義する*/
$2 = 1
repeat_case $2 <= 10
  proc read_write{
    read cell %1 $2,1 0 &1                       /* 指定行を読み出す*/
    write cell %2 $2,1 0 &1                      /* 指定行を書き込む*/
    $2 = $2+1
     }read_write
save text_form %2                                /* テキストフォームを格納する*/
```

14. 3. テキストファイルを直接テキストフォームに書き込み別のテキストファイルに書き込む場合 読み出し先ファイル番号で、テキストフォームを定義してテキストフォーム読み出し文で一括してテキストフォームに読み出し た後、格納先ファイルを別の名前で同じファイル番号で定義し、テキストフォームに書き込みを行った後、格納します。 なお、読み出し先ファイルからテキストファイルセル読み出し文で読みたい場合は、別のファイル名で定義する前に行います。



使用する構文と概略手順:

①読み出し先ファイル番号を定義します。
def file_id %ファイル番号1 ファイル名1 属性 ⇒読み出し先
②読み出し先ファイルの行列数を取得します。
read nu_cell %ファイル番号 格納先チャネル
③バッファとして使用するテキストフォームを定義します。」
def text_form %ファイル番号1 行列数
④読み出し先ファイルからテキストフォームに読み出します。
read text_form %ファイル番号
⑤テキストフォームに追記する内容を書き込みます。(必要に応じて)
write cell %ファイル番号1 行列番号 方向 書き込みチャネル列
⑥読み出し先ファイルから読み出します。(必要に応じて)
read cell %ファイル番号 1 行列番号 方向フラグ 格納先チャネル列
⑦読み出し先ファイルと同じファイル番号 1 に別ファイル名を定義します。
def file_id %ファイル番号1 ファイル名2 属性 ⇒格納先
⑧書き込まれているテキストフォームを格納します。
save text_form %ファイル番号1

※ この方法は、一旦定義したテキストフォームは同じファイル番号で別名ファイルが定義されても、ファイル番号が変わらない限り同じ内容を保持していることを意味しています。

記述例:

csv ファイルの先頭から 10 行を別の csv ファイルに格納する場合

```
def file_id %1 “読み出し先ファイル” csv      /*読み出し先ファイル名を定義する*/
read num_cell %1 $1                          /* 読み出しファイルの列数を取得する*/
$1 = MAX($1)                                 /* 最大列数を演算する*/
def text_form %1 10,$1                       /* 仮想シートを定義する*/
read text_form %1
def file_id %1 “格納先ファイル” csv          /* 格納先ファイル名を定義する*/
save text_form %1                            /* 仮想シートを格納する*/
```

記述例14. 1. 収録履歴テキストファイルを作成追記する

選択された解析対象ファイルの各チャネルの最大値、平均値、最小値、変動値、実効値、最大-最小値、中央値を演算し解析年月日、時刻を付けて履歴ファイルを生成します。既に履歴ファイルが存在している場合は、追記し履歴ファイルを更新します。

<Archi_1 Script 記述例>

```
1           /* --- text_write3.prc----------*/
2           dcl menu_label "収録履歴ファイル生成/更新"
3           def ch_name $1 "num_files"
4           def ch_name $4 "his_check_flga"
5           def ch_name &1 "file_name"
6           def ch_name &2 "now_date"
7           def ch_name &3 "now_time"
8           def ch_name &10 "current_folder"
9           /* --- 履歴ファイル定義＆存在確認 ----------*/
10          read folder_path &10
11          def file_id %1 "history" csv
12          read file_check %1 $4
13          /*--- 解析対象ファイル名 取得 ----------*/
14          get file_name &1 $1 "hdr"
15          case $1 > 0
16            proc exec1{
17              /*-履歴ファイル存在時処理 -------*/
18              case $4 = 0
19                proc history_file_read{
20                  read num_cell %1 $5                  /* 履歴ファイルサイズ取得*/
21                  $6 "行 counter:" = LEN($5)+3         /* 行カウンタ初期化*/
22                  $5 = LEN($5)+$1*100                  /* 新規行数演算*/
23                  def text_form %1 $5,12               /* text form 定義*/
24                  read text_form %1                    /* 履歴ファイル読み込み*/
25                  read cell %1 1,9 1 $18(0)
26                  $18 "total_files:" = $18+$1(0)
27                  write cell %1 1,9 1 $18(F0)
28                } history_file_read
29              /*--履歴ファイル新規生成 --------*/
30              case $4 <> 0
31                proc history_file_create{
32                  $6 = 1                               /* 行カウンタ初期化*/
33                  $5 = $1*100                          /* 新規行数演算*/
34                  def text_form %1 $5,12               /* text form 定義*/
35                  write cell %1 $6,4 1 "<試験データ収録履歴>"
36                  write cell %1 $6,9 1 $1(0)(F0)
37                  $6 = $6+3
38                }history_file_create
39              /* --- 解析ファイル処理 LOOP -----*/
40              $7 "file_counter:" = 0
41              repeat_case $7 < $1
42                proc terget_file_proc{
43                  def file_id %2 &1($7) wav
44                  read wave %2
45                  &4 "acq_date:" = CSDT()
46                  &5 "acq_time:" = CSTM()
47                  $2 "num_ch:" = NCH()
48                  $3 "ch_series:" = CHS()
49                  $8 "num_data:" = NDT()
50                  &8 "sampl_unit:" = CXUT()
51                  $9 "sampl_period:msec" = PRD()*1000
52                  &6 "signal_name:" = CHNM(0)
53                  &7 "signal_unit:" = CUNT(0)
54                  &2 "now_date:" = CNDT()
55                  &3 "now_time:" = CNTM()
56                  assign &8 = "m"|&8
57                  assign &9 = "収録年月日","時刻","ファイル名","チャネル数","収録データ数"
58                  assign &9 = &9,"周期","単位"," ","生成年月日","時刻"
59                  write cell %1 $6,1 0   &9
60                  $6 = $6+1
```

```
61                  write cell %1 $6,1 1 &4,&5,&1($7),$2(F0),$8(F0),$9(F2),&8,"  ",&2,&3
62                  $6 = $6+1
63                  write cell %1 $6,1 1 "ch No.","信号名","単位","最大値","平均値","最小値","変動値","実効値","最大-最小","中央値"
64                  $6 = $6+1
65                  write cell %1 $6,1 1 $3(F0),&6,&7 /* チャネル番号,信号名,単位  書き込み*/
66                  assign $11 "max:" = $2<0>
67                  assign $12 "mean:" = $2<0>
68                  assign $13 "min:" = $2<0>
69                  assign $14 "std:" = $2<0>
70                  assign $15 "EV:" = $2<0>
71                  assign $16 "max-min:" = $2<0>
72                  assign $17 "median:" = $2<0>
73                  /*   解析チャネル  LOOP----------*/
74                  $10 "ch_counter:" = 0
75                  repeat_case $10 < $2
76                    proc calc{
77                      $11($10) = MAX(#($3($10)))
78                      $12($10) = MEA(#($3($10)))
79                      $13($10) = MIN(#($3($10)))
80                      $14($10) = STD(#($3($10)))
81                      $15($10) = EFF(#($3($10)))
82                      $16($10) = $11($10)-$13($10)
83                      $17($10) = MED(#($3($10)))
84                      $10 = $10+1
85                    }calc
86                  write cell %1 $6,4 1 $11(F3),$12(F3),$13(F3),$14(F3),$15(F3),$16(F3),$17(F3)  /*結果書き込み*/
87                  $6 = $6+$2+2
88                  $7 = $7+1
89                }terget_file_proc
90              save text_form %1
91              def folder_path &10
92            }exec1
93          end
```

<Archi_1 Script 記述構文の説明>

10 行目～12 行目：既に履歴ファイルが存在しているかの確認を行います。
10 行目： フォルダパスを取得します。これはファイル選択文でフォルダが切替られることを想定し保存します。
11 行目： 履歴ファイルのファイル番号を定義します。
12 行目： 定義したファイル番号が既に存在しているか確認します。
14 行目: ファイル選択文で拡張子.hdr とし、履歴ファイルに登録する解析対象ファイル名を選択します。ファイル選択文が実行されるとファイル選択ダイアログが表示され、一度に複数のファイルを選択できます。
18 行目～28 行目：履歴ファイルが存在していた場合の処理を行います。
20 行目： 履歴ファイルの現在の大きさを取得します。
21 行目： テキストフォーム書き込み行カウンタに追記開始する行番号を設定します。
22 行目： 新たに追記するファイル数からテキストフォームの最大行数を演算します。
23 行目： テキストフォーム定義文で演算行数追記に使用するテキストフォームを定義します。
24 行目： 存在している履歴ファイルの内容をテキストフォームに書き込みます。
25 行目： 現在の解析結果登録ファイル数を履歴ファイルから取得します。
26 行目： 現在の登録ファイル数に追記するファイル数を加算します。
27 行目： テキストフォームに読み出されている先頭行に記載されているファイル数を更新します。
30 行目～38 行目：履歴ファイルが存在していない場合の処理を行います。
32 行目： テキストフォーム書き込み行カウンタを1行目とします。
33 行目： テキストフォームの定義用に用意する最大行を演算します。
34 行目： 新規に用意するテキストフォーム定義文です。
35 行目： テキストフォームの先頭行にファイルヘッダーを書き込みます。
36 行目： 同じくテキストフォーム先頭行に登録するファイル数を書き込みます。
37 行目： 書き込み行カウンタを更新します。
40 行目～90 行目：ファイル登録処理を行います。ファイル登録処理は選択された解析対象ファイル数分の繰り返し処理となります。
43 行目： ファイル番号定義文で選択された解析対象ファイルを定義します。
44 行目： 解析対象ファイルを読み出します。
45 行目～53 行目：解析対象ファイルから解析および履歴ファイルの登録に必要なパラメータを取得します。
45 行目： 収録開始年月日を取得します。
46 行目： 収録開始時刻を取得します。

47 行目： 収録チャネル数を取得します。
48 行目： 収録チャネル番号を取得します。
49 行目： 収録データ個数を取得します。
50 行目： 収録サンプリング周期単位を取得します。
51行目： 収録サンプリング周期を取得し表示の都合上 1000 倍します。
52 行目： 収録チャネルの信号名を取得します。
53 行目： 収録チャネルの単位を取得します。
54 行目： 履歴ファイルに登録する現在年月日を取得します。
55 行目： 同じく履歴ファイルに登録する現在時刻を取得します。
56 行目： 収録したサンプリング周期単位の先頭に”m”を付加します。
57 行目： 履歴ファイルのファイルごとのヘッダー行文字列を作成します。
59 行目： 履歴ファイルにファイルごとのヘッダー行を書き込みます。
61 行目： 履歴ファイルのファイルごとヘッダー行に直下にヘッダー取得した内容を書き込みます。
63 行目： 履歴ファイルに解析内容を表示するためのデータヘッダー行を書き込みます。
65 行目： チャネル番号列、信号名列、単位列に書き込みます。
66 行目～72 行目：解析結果を格納するための配列を準備します。
74 行目～85 行目：チャネルごとの演算結果処理を行います。
77 行目： チャネルごとに最大値を演算します。
78 行目： チャネルごとに平均値を演算します。
79 号目： チャネルごとに最小値を演算します。
80 行目： チャネルごとに標準偏差を演算します。
81 行目： チャネルごとの実効値を演算します。
82 行目： チャネルごとに最大値-最小値を演算します。
83 行目： チャネルごとに中央値を演算します。
86 行目： 演算結果をそれぞれの列に書き込みます。
87 行目： 書き込み行カウンタを次の登録ファイルブロックの先頭行番号に更新します。
90 行目： 書き込まれたテキストフォームを履歴ファイルに上書き更新します。
91 行目： カレントフォルダパス定義文で元のフォルダパスに戻します。

<生成更新された履歴ファイルをエクセルで表示>

	A	B	C	D	E	F	G	H	I	J
1			<試験データ収録履歴>						10	
2										
3	収録年月日	時刻	ファイル名	チャネル数	収録データ数	周期	単位		生成年月日	時刻
4							mSec		09-12-2011	10:17:07
5	ch No.	信号名	単位	最大値	平均値	最小値	実動値	実効値	最大-最小	中央値
6				3.045	0	-2.859	0.544	0.544	5.905	-0.063
7				3.239	0	-3.106	0.516	0.516	6.345	-0.043
8				5.595	0	-5.043	1.087	1.087	10.638	-0.126
9			V	1	0.007	0	0.085	0.085	1	0.5
10										
11										
12	収録年月日	時刻	ファイル名	チャネル数	収録データ数	周期	単位		生成年月日	時刻
13							mSec		09-12-2011	10:17:09
14	ch No.	信号名	単位	最大値	平均値	最小値	実動値	実効値	最大-最小	中央値
15			V	2.507	0.945	-0.003	1.202	1.529	2.51	0.018
16				1.798	0.953	-0.03	0.665	1.162	1.828	0.915
17				24.2	8.231	-8.6	4.415	9.34	32.8	7.8
18				44.336	43.748	42.759	0.352	43.749	1.577	43.612
19				18.432	17.897	16.768	0.266	17.899	1.664	17.68
20				2.294	1.208	-0.008	0.841	1.472	2.302	1.159
21				5240.8	1345.493	-48.006	1889.791	2319.841	5288.806	3205.902
22				92.6	64.441	51.137	5.562	64.68	21.463	68.224
23				2.368	1.202	-0.009	0.861	1.479	2.377	1.151
24			V	11.712	0.065	-0.032	0.929	0.932	11.744	0.152

記述例14. 2. 収録履歴ファイルを編集更新する

作成した履歴ファイルを読み出し、登録されているファイルの削除、同じファイルが登録されていた場合の重複登録ファイルの削除、登録されているファイルの収録年月日/時刻順への並び替えの 3 つの編集機能から選択した機能を実行し再格納します。なお、登録削除を選択した場合で全てのファイルを登録削除指定した場合は履歴ファイル自体を削除し新たにファイル名に“_old”を付加したバックアップファイルを生成します。

<Archi_1 Script 記述例>

```
1          /* --- text_write3.prc----------*/
2          dcl menu_label "収録履歴ファイル編集"
3          /*-------- 登録情報取得----------------------------------------*/
4          def ch_name $2 "行数"
5          def ch_name $3 "登録ファイル数"
6          def file_id %1 "history" csv
7          read num_cell %1 $2
8          read cell %1 1,9 0 $3
9          $2 = LEN($2)
10         assign &1 "収録年月日:" = $3<" ">
11         assign &2 "収録時刻:" = $3<" ">
12         assign &3 "ファイル名:" = $3<" ">
13         assign $1 "収録チャネル数:" = $3<0>
14         assign $6 "ファイルブロック先頭行:" = $3<0>
15         assign $7 "ファイルブロック終結行:" = $3<0>
16         $4 "行カウンタ:" = 5
17         $5 "ファイルカウンタ:" = 0
18         repeat_case $5 < $3
19           proc 登録ファイル情報取得{
20             read cell %1 $4,1 1 &1($5),&2($5),&3($5),$1($5)
21             $6($5) = $4-1
22             $7($5) = $4+$1($5)+1
23             $4 = $4+$1($5)+5
24             $5 = $5+1
25           }登録ファイル情報取得
26         /*------ 編集モード選択----------------------------------------*/
27         assign &4 "編集モード:" = "登録削除","重複登録削除","収録年月日順並び替え"
28         $8 "編集フラグ:" = 0
29         get radio_button_status &4 $8
30         /*-------編集実行開始----------------------------------------*/
31         def text_form %1 $2,12          /* text_form 定義*/
32         read cell %1 1,1 0 &11          /* 先頭行読み出し*/
33         write cell %1 1,1 0 &11         /* 先頭行書き込み*/
34         case $8 = 0 /*------------登録削除-------------------------*/
35           proc reg_del{
36             assign &5 "選択表示用:" = &1|" "|&2|" ⇒"|&3|" ("|$1(F0)|"ch)"
37             get check_box_status &5 $9          /* 削除ファイル選択*/
38             $10 = SUM($9)
39             case $10 > 0
40               proc del_exec{
41                 $10 "削除実行フラグ:" = GTE($10,1)*LTE($10,$3-1)
42                 case_true $10   /*全削除除外*/
43                   proc file_del{
44                     $9 "選択フラグ:" = NOT($9)   /* 選択フラグ反転*/
45                     $6 = ZSP($9,$6)                  /* 選択ファイルでブロック先頭行再構成*/
46                     $7 = ZSP($9,$7)                  /* 選択ファイルでブロック終端行再構成*/
47                     $3 = SUM($9)                      /* 書き込みブロック数更新*/
48                     write cell %1 1,9 0 $3(F0)   /* 書き込みブロック数書き込み*/
49                     call proc ブロック書き込み
50                   }file_del
51                 case_false $10 /*全削除*/
52                   proc back_up_save{
53                     read text_form %1
54                     delete file %1
55                     def file_id %1 "history_old" csv
56                     save text_form %1
57                   }back_up_save
58               }del_exec
```

```
59          }reg_del
60        case $8 = 1 /*-------------重複削除---------------------------------*/
61          proc del_reg_dup{
62            assign &5 = &1|&2|&3          /* 重複検査用文字列生成*/
63            $5 = 0
64            repeat_case $5 < $3           /* 重複確認Loop*/
65              proc dup_block_find{
66                assign &11 = $3<&5($5)> /* 検索対象文字列生成*/
67                $9 = CEQ(&11,&5,1)        /* 重複検索*/
68                $10 = LEN($9)
69                case $10 > 1              /* 重複検出*/
70                  proc del_block{
71                    $9 = SBV($9[1,LEN($9)-1],DAG(LEN($9)-1,0),DAG($3,1))
72                    $6 = ZSP($9,$6)       /* ブロック先頭行重複削除再構成*/
73                    $7 = ZSP($9,$7)       /* ブロック終端行重複削除再構成*/
74                    &5 = CREC($9,&5)      /* 重複検査用文字列再構成*/
75                    $3 = LEN($6)          /* 書き込みブロック数更新*/
76                  }del_block
77                $5 = $5+1                 /* loop counter up*/
78              }dup_block_find
79            write cell %1 1,9 0 $3(F0)  /* 書き込みブロック数書き込み*/
80            call proc ブロック書き込み
81          }del_reg_dup
82        case $8 = 2  /*-----------収録年月日順並び替え----------------------*/
83          proc sort_date{
84            $11 =(CTN(CEXT(1,2,&1))*31+CTN(CEXT(4,2,&1))+CTN(CEXT(7,4,&1))*360)*24*3600
85            $11 =$11+CTN(CEXT(1,2,&2))*3600+CTN(CEXT(3,4,&2))*60+CTN(CEXT(7,2,&2))
86            $9 = SRT(2,$11)               /* 収録開始年月日/時刻昇順並び Index 取得*/
87            $6 = QUE($9,$6)               /* 昇順並び Index でブロック先頭行再構成*/
88            $7 = QUE($9,$7)               /* 昇順並び Index でブロック終端行再構成*/
89            call proc ブロック書き込み
90          }sort_date
91        end
92
93        /*------- 再書き込み Subrutine-----------------------------------------*/
94        proc ブロック書き込み{
95          $4 = 4
96          $5 = 0
97          repeat_case $5 < $3                 /* ブロック書き込みLOOP*/
98            proc regist1{
99              repeat_case $6($5) <= $7($5)    /* ブロック内 LOOP*/
100               proc regist_loop{
101                 read cell %1 $6($5),1 0 &11  /* ブロック内読み出し*/
102                 write cell %1 $4,1 0 &11     /* 書き込み*/
103                 $4 = $4+1                      /* 書き込み行カウンタ更新*/
104                 $6($5) = $6($5)+1              /* 読み出し行カウンタ更新*/
105               }regist_loop
106             $4 = $4+2                        /* ブロック区切り行 counter 更新*/
107             $5 = $5+1                        /* loop counter up*/
108           }regist1
109         save text_form %1                 /* 削除編集後テキストフォーム格納*/
110       }ブロック書き込み
```

<Archi_1 Script 記述構文の説明>

6 行目： 履歴ファイルファイル番号を定義します。この Script は履歴ファイルが存在していることを前提としており、履歴ファイルの存在確認を行っていません。したがって存在していない場合、実行時エラーが発生します。

7 行目： 履歴ファイルのサイズを取得します。

8 行目： 現在登録されているファイル数を先頭行の 9 列から取得します。

10 行目～15 行目：編集に先立って各ファイル情報取得確保用に配列を準備します。

17 行目～25 行目：登録されているファイル情報取得処理を行います。処理は登録ファイル数分繰り返します。

20 行目: 各ファイル情報記録行からファイル情報を取得します。取得する情報は収録開始年月日、時刻、ファイル名および収録チャネル数の 4 項目とします。

21 行目： ファイルの登録情報の先頭行番号を格納します。

22 行目： ファイルの登録情報の終端行番号を格納します。ファイルごと登録行数は収録チャネル数に依存します。

23 行目： 読み出し行カウンタで次のファイル情報記載行に更新します。

27 行目～29 行目：行う編集機能選択処理を行います。
27 行目： 編集機能選択用ラジオボタンに表示する内容を生成します。
28 行目： ラジオボタン選択初期値$8 に設定します。
29 行目： ラジオボタンステータス取得文で実行するとラジオボタンダイアログを表示し、ラジオボタンを選択します。
31 行目～33 行目：生成するテキストフォーム初期化処理を行います。
31 行目： 使用するテキストフォームw定義します。
32 行目： 編集に関係ない履歴ファイルのヘッダー行を読み出します。
33 行目： 32 行目で読み出したファイルヘッダー行をテキストファイルに書き込みます。
34 行目～59 行目：編集機能選択で登録削除が選択された時の処理を行います。
36 行目： 登録削除するファイル選択用に表示するチェックボックスダイアログの表示内容を生成します。
37 行目：チェックボックススタータス取得文で実行するとダイアログが表示され削除するファイルを選択します。
38 行目： 削除選択された個数を演算します。
39 行目～58 行目：削除される個数が 0 個でない場合に行う処理です。
41 行目： 選択された削除指定ファイル個数が全削除か否かのフラグを演算生成します。
42 行目～50 行目：登録全削除以外の処理を行います。
44 行目：選択フラグ 1 または 0 を反転します。
45 行目： ファイルの登録情報先頭行番号配列から選択されていない先頭行番号を削除します。
46 行目： ファイルの登録情報終端行番号配列から選択されていない終端行番号を削除します。
47 行目： 登録ファイル数を更新します。
48 行目： 履歴ファイルのヘッダー行に書き込まれている登録ファイルを更新書き込みします。
49 行目： サブルーチン読み出し文で、削除しないファイル情報を履歴ファイルからテキストフォームにコピーします。
51 行目～57 行目：全削除設定された場合の処理を行います。
53 行目： 履歴ファイルから一括してテキストフォームに書き込みます。
54 行目： 履歴ファイルを削除します。
55 行目： テキストフォームと同じファイル番号で履歴ファイルバックアップするファイルを定義します。
56 行目： テキストフォームを定義したバックアップファイルに格納します。
60 行目～81 行目：編集機能選択で重複登録削除が選択された時の処理を行います。
62 行目： 重複登録検索用に収録開始年月日、時刻、ファイル名を連結してファイルごと固有文字列を生成します。
63 行目～78 行目：重複登録ファイル検索処理を行います。処理は登録ファイル数分繰り返します。
66 行目： 検索文字列を検索対象文字列配列分生成します。
67 行目： 検索文字列配列と検索対象文字列配列の要素ごと検査し、一致した Index を求めます。
68 行目： 一致した要素数を演算します。
69 行目～76 行目：重複登録が存在した時の処理を行います。
71 行目： 67 行目で求めた Index の先頭を除いた位置を 0、その他を 1 とした論理数列を演算します。
72 行目： ファイルの登録情報先頭行番号配列を 71 行目で生成した論理数列を使用して再構成します。
73 行目： ファイルの登録情報終端行番号配列を 71 行目で生成した論理数列を使用して再構成します。
74 行目： 登録ファイル数を更新します。
79 行目： 履歴ファイルのヘッダー行に書き込まれている登録ファイルを更新書き込みします。
80 行目： サブルーチン読み出し文で、削除しないファイル情報を履歴ファイルからテキストフォームにコピーします。
82 行目～83 行目：編集機能選択で収録年月日/時刻順並び替えが選択された時の処理を行います。
84 行目： 収録年月日文字列を並び替え処理用に秒単位の数値に変換します。
85 行目： 同じく 84 行目で求めた年月日の秒単位数値に時刻文字列を秒単位の数値に変換し加算します。
86 行目： 85 行目で求めた秒を昇順に並び替えたソートキー(index 並び)を演算します。
87 行目： ソートキーに従って登録情報先頭行番号配列を並び替えます。
88 行目： ソートキーに従って登録情報終端行番号配列を並び替えます。
94 行目～110 行目：ファイル登録情報をテキストフォームに書き込み、履歴ファイルに上書き格納する外部演算 処理ブロック(サブルーチン)です。
97 行目～108 行目：ファイルごとの登録情報をテキストフォームに書き込み処理を行います。この処理理ファイル数分繰り返します。
99 行目～105 行目：ファイルごとの先頭行から終端行まで 1 行ごとにファイルから読み出してテキストフォームへの書き込みを行います。この処理は先頭行番号から終端行番号まで繰り返します。先頭行とは履歴ファイルに記録されている収録ファイルごとの結果をブロックと見なした時の当該ブロックの先頭行を意味し、同様に終端行とは当該ブロックの最終行を意味します。
101 行目：履歴ファイルから 1 行読み出します。
102 行目：読み出した 1 行をテキストフォームに書き込みます。
103 行目：テキストフォーム書き込み行カウンタを更新します。
104 行目：履歴ファイル読み出し行カウンタを更新します。
109 行目：書き終えたテキストフォームを履歴ファイルに上書き格納します。

15. バイナリファイルから読み出す

バイナリ形式ファイルあるいはテキスト形式とバイナリ形式の混合ファイルから開始オフセットバイト数、読み出し数、読み出し型式を指定して読み出します。読み出し対象ファイルの拡張子は問いません。

15. 1. バイナリファイル読み出し事前処理

ファイル番号定義文

文法:

```
def file_id %ファイル番号  ファイル名 属性
```

属性は bin を指定します。また、ファイル名は必ず拡張子を付けて記述します。

※ ファイル番号定義文で直接ファイル名を記述せず、Script 実行時に読み出すファイルを選択する方法については、第 5 章「解析対象ファイルを読み出す」の 5.3、5.4 項を参照下さい。

15. 2. バイナリファイルからデータの読み出し バイナルファイルからの読み出し方法は、同じ型式を持ったデータを読み出す場合と、型式が混在する構造形式で読み出す場合の 2 種があり、一つのファイルからの読み出しに混在して使用できます。

15. 2. 1. 同じ型式が連続している場合の読み出し

バイナリファイル読み出し文

文法:

```
read bin %ファイル番号 開始オフセット 読み出し数 格納先 戻りオフセット   型式
```

開始オフセット(読み出し先頭アドレス)は必ず Byte 単位で指定します。読み出しは指定した開始オフセット位置から開始されます。読み出し数は格納先チャネルに指定した形式での読み出し個数を意味します。なお 0 を指定した場合は、特別な意味を持ち、ファイルの最終(EOF)まで読み出す意味となります。

型式はキーワードで、読み出し形式を表します。

内容	型式キーワード	読み出し数指定	格納先属性
バイト単位	byte	バイト数単位	数値属性
符号付き整数形式(2byte)	int16	データ個数単位	数値属性
符号なし整数形式(2byte)	uint16	データ個数単位	数値属性
符号付き倍精度整数形式(4byte)	int32	データ個数単位	数値属性
符号無し倍精度整数形式(4byte)	uint32	データ個数単位	数値属性
単精度浮動小数点形式(4byte)	float32	データ個数単位	数値属性
倍精度浮動小数点形式(8byte)	float64	データ個数単位	数値属性
文字列(改行または null 迄)	char	行数(項目数)注 1	文字属性

注1:文字列の格納先に、C/RL/F 文字或いは、null が付いていない場合は、byte 属性で読み出し、CVTA()関数を使用して文字列に変換します。 戻りオフセットは読み出し終了後の次のバイト位置を示します。続けて読み出す場合は、次回の読み出し開始オフセットとして 使用できます。但し、読み出しが EOF(End of File)に達した場合は-1 が戻ります。

記述例:

```
def file_id %1 “abcd.
uff” bin
read bin %1 0 10 &1 $1 char
```

ファイル名:abcd、拡張子.uff の先頭から 10 行分をテキストとして読み出します。

15. 2. 2. 型式が混在した構造が繰り返されている場合のの読み出し

バイナリファイルチャネル構造読み出し文

文法:

```
read bin_struct %ファイル番号 開始オフセット 格納先チャネル構造  戻りオフセット
```

格納先チャネル構造の記述則: 読み出し数<格納先:型式,格納先:型式,・・・格納先:型式>

読み出し数は半角”<”、”>”で囲んだレコード構成への繰り返し読み出し数を意味します。読み出し数は記述省略可で、記述省略した場合は1と見なします。また、0 は特別な意味を持ち、ファイルの最終(EOF)まで読み出す意味となります。 格納先は格納先チャネルを意味し、半角コロン”:”に続き型式のペアで記述します。

なお、格納先チャネル記述にチャネル連続指定記述則を使用できます。($[10,3]は$10,$11,$12 と同じ意味)

又、格納先チャネル構造に格納先チャネル構造を記述することができます。但し、ネストされた格納先チャネル構造の読み出し数に 0 は指定できません。

読み出し数<格納先:型式,読み出し数<格納先:型式,・・・格納先:型式>,格納先:型式>

開始オフセット(読み出し先頭アドレス)は必ず Byte 単位で指定します。読み出しは指定した開始オフセット位置から開始されます。

型式はキーワードで読み出し形式を表します。

内容	キーワード	格納先属性
バイト単位	byte 注 2	数値属性
符号付き整数形式(2byte)	int16	数値属性
符号なし整数形式(2byte)	uint16	数値属性
符号付き倍精度整数形式(4byte)	int32	数値属性
符号無し倍精度整数形式(4byte)	uint32	数値属性
単精度浮動小数点形式(4byte)	float32	数値属性
倍精度浮動小数点形式(8byte)	float64	数値属性
文字列	半角文字数	文字属性

注 2:byte 形式で読み出す場合、バイト数をキーワードに続き数値で記述します。バイト数を記述省略した場合は 1 と見なします。
例えば 2<$1:byte10,$2:int16>とレコード構成を記述した場合、
1 回目の読み出しでは、$1(0)～$1(9)に byte 単位で、$2(0)に 2byte 整数形式で格納され
2 回目の読み出しでは、$1(10)～$(19)に byte 単位で、$2(1)に2byte 整数形式で格納されます。 従って、byte 以外の型式とは読み出し結果の要素数が異なる事に留意して下さい。
なお、byte 形式で格納したチャネルからレコード単位で読み出す場合は、ERC()関数を使用して行います。
戻りオフセットは読み出し終了後の次のバイト位置を示します。続けて読み出す場合は、次回の読み出し開始オフセットとして使用できます。但し、読み出しが EOF(End of File)に達した場合は-1 が戻ります。

記述例:

```
def file_id %1 “abcd.dat” bin
read bin_struct %1 0 10<73<$[1,3]:int16,$4:byte>,$5:byte> $6
```

ファイル名:abcd、拡張子.dat のファイル先頭アドレスから、レコード構成$1～$3 を int16、$4 を byte とした 7byte を 73 回 レ属して読み出し、その後$5 に byte で読み出す 512byte を合計 10 回読み出します。読み出し結果は、$1(0)～$1(729)、$2(0)～$2(729)、$3(0)～$3(729)、$4(0)～$4(729)、$5(0)～$5(9)に格納されます。

記述例15. 1. UFF58b ファイルを PcWaveForm フォーマットに変換する

UFF58b で格納されたファイルを読み出し、PcWaveForm フォーマットに変換したファイルを生成します。なお、変換対象ファイルはバイナリ記録部分の先頭に付加されている 58 フォーマットのレコード行数は 11 行で構成されており、Record 6 の Field 1が時間応答データであり、記述されているデータ形式のバイト並びはリトルエンディアン、型式は IEEE754 で記述され、単精度浮動小数点形式あるいは倍精度浮動小数点形式であることを前提としています。
なお、フォーマット変換された生成波形ファイルの格納先は変換対象 UFF58b ファイルの格納先に拡張子違いの同じファイル名で格納します。

<Archi_1 Script 記述例>

```
dcl menu_label "UFF58 file 変換"
def ch_name &1 "header_buff"
def ch_name &2 "file_name"
def ch_name &100 "Start"
def ch_name &101 "header"
def ch_name &102 "record1"
def ch_name &103 "record2"
def ch_name &104 "record3"
def ch_name &105 "record4"
def ch_name &106 "record5"
def ch_name &107 "record6"
def ch_name &108 "record7"
def ch_name &109 "record8"
def ch_name &110 "record9"
def ch_name &111 "record10"
def ch_name &112 "record11"
def ch_name $10 "num_file"

get file_name &2 $10 "uff"
case $10 > 0
  proc conv_exec{
    assign &2 = &2|".uff"
    $9 "file_counter:" = 0
    repeat_case $9 < $10
      proc file_loop{
        /*---------- uff file loop ----------*/
        def file_id %1 &2($9) bin
        $12 "cannot_flag:" = 0
        $8 "ch_counter:" = 0
        $7 "read_position_byte:" = 0
        $1 "next_byte_position:" = 0
        repeat_case $1 <> -1
          proc read_loop{
            /*--- 58b head read & field separate ---*/
            read bin %1 $7 13 &1 $1 char   /* 58b & 58 record read */
            $8 = $8+1      /* 読み出しチャネル counter */
            assign &5 "処理通知文:" = &2($9)|" ch"|$8(F0)|" ヘッダー読み出し"
            write progress_status &5
            $3 "header_line_counter:" = 0
            repeat_case $3 < 13
              proc header_separate{
                $4 "header_record_ch_No.:" = $3+100
                &($4) = CSEP(3,&1($3))
                $3 = $3+1
              }header_separate
            case &107(0) = "1"
              proc time_domain{
                /*--------- data read ----------*/
                $5 "data_ch_No.:" = $8+200
                def ch_name $($5) &107(4) /* rec6-fld3 信号名定義*/
                def ch_unit $($5) &110(5) /* red8-fld6 単位定義*/
                assign &5 = &2($9)|" ch"|$8(F0)|" "|&107(4)|" データ読み出し"
                write progress_status &5
                $2 "num_data:" = CTN(&108(1)) /* rec7-fld2 num of data*/
                $7 = $1
```

```
56                              case &108(0) = "2"
57                                proc type_float32{
58                                  read bin %1 $7 $2 $($5) $1 float32 /* data read */
59                                }type_float32
60                              case &108(0) = "4"
61                                proc type_float64{
62                                  read bin %1 $7 $2 $($5) $1 float64 /* data read */
63                                }type_float64
64                              $7 = $1
65                              read bin %1 $7 1 &1 $1 char /* data until code read */
66                              $7 = $1
67                            }time_domain
68                          case &107(0) <> "1"
69                            proc cannot_convert{
70                              $1 = -1
71                              $12 = 1
72                            }cannot_convert
73                        }read_loop
74                      /*---------- wave file create ------------------------------*/
75                      case $12 = 0
76                        proc wave_file_create{
77                          assign &5 = &2($9)|" 波形ファイル格納処理"
78                          write progress_status &5
79                          call proc DATE_FORMAT_CONV &104   /* 年月日形式変換*/
80                          &2($9) = CDEL(-4,4,&2($9))
81                          $6 "sampling_period:" = CTN(&108(4))
82                          def file_id %2 &2($9) wav
83                          def wav_sampling %2 1 $6 &109(5) 0
84                          def wav_datetime %2 &104(0) &104(1)
85                          def wav_type %2 float
86                          save wave %2 $[201,$8]
87                        }wave_file_create
88                      $9 = $9+1       /* file_counter up */
89                    }file_loop
90                }conv_exec
91              end
92
93              proc DATE_FORMAT_CONV{
94                def inherit_ch &1
95                def local_ch &2,&3,&4,&5,&6
96                assign &3 = "JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC"
97                assign &4 ="01","02","03","04","05","06","07","08","09","10","11","12"
98                &2 = CEXT(4,3,&1(0))
99                assign &2 = 12<&2>
100               &2 = &4(CEQ(&2,&3,1))
101               &5 = CEXT(1,2,&1(0))
102               &6 = CEXT(8,2,&1(0))
103               assign &1(0) = &2|"-"|&5|"-20"|&6
104             }DATE_FORMAT_CONV
```

<Archi_1 Script 記述構文の説明>

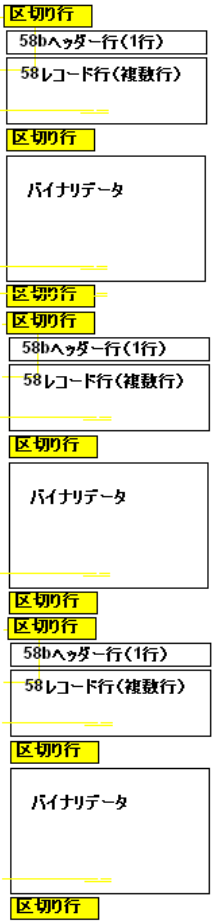
1 行目:　実行メニューボタンに表示するボタンラベル名を宣言します。
19 行目: ファイル選択文で変換する uff ファイルを指定します。同時に複数個のファイル指定が可能です。
20 行目: ファイル選択されたファイル数をチェックします。ファイルが選択されていない場合は終了します。
25 行目～89 行目:変換処理する演算処理ブロックです。
22 行目: ファイル選択文で取得したファイルは拡張子が付いていませんので拡張子文字列を付加します。
23 行目: ファイルカウンタを初期化します。
24 行目: 選択されたファイル数分処理するための変換処理の演算処理ブロックを繰り返し指定します。
25 行目～88 行目:ファイルごとに変換処理する演算処理ブロックです。
27 行目: ファイル番号定義文で変換するファイル番号を属性 bin で定義します。
28 行目: 変換処理できない場合の変換処理放棄フラグを初期化します。
29 行目: 読み出しチャネルカウンタを初期化します。
30 行目: バイナリファイルの読み出しオフセットバイト数を初期化します。
31 行目: 次に読み出すバイナリファイルのバイト位置を初期化します。
32 行目: 繰り返し条件実行制御文で、バイナリファイルの EOF 位置までを繰り返し処理制御します。

33 行目～73 行目：バイナリファイルの読み出し演算処理ブロックです。
35 行目： バイナリファイル読み出し文で 58b/58 ヘッダーをバイナリファイルから文字属性で 13 行読み出します。
36 行目： 読み出しチャネルカウンタをインクリメントします。
38 行目： 実行中に表示されるプログレスバーに現在処理内容を書き込み使用者に通知します。
39 行目： 読み出したヘッダーをフィールドに分解するための行カウンタを初期化します。
40 行目： 繰り返し条件実行制御文で、行数文、ヘッダーフィールド分解演算処理ブロックを制御します。
41 行目～45 行目：ヘッダーフィールド分解演算処理ブロックです。
42 行目： 行ごとに格納されるチャネル番号を生成します。
43 行目： 1 行ごとに項目区切りが半角スペース区切りであることを前提にフィールドに分解します。
43 行目： 行カウンタをインクリメントします。
46 行目： 58 レコード番号 6 のフィールド番号 1 の値が時間軸か否か確認します。
47 行目～67 行目：時間軸データで有った場合の読み出し演算処理ブロックです。
49 行目： データ格納先チャネル番号を生成します。
50 行目：レコード番号 6 フィールド番号 3 を信号名と見なし格納先チャネルに信号名を定義します。
51 行目：レコード番号 9 フィールド番号 6 を単位と見なし格納先チャネルに単位を定義します。
54 行目：レコード番号 7 フィールド番号 2 に記載されているデータ個数を文字列から数値に変換します。
55 行目： バイナリファイル読み出し開始バイト位置を更新します。
56 行目： レコード番号 7 フィールド番号 1 が 2：単精度浮動小数点形式かチェックします。
57 行目～59 行目：単精度浮動小数点の場合の読み出し演算処理ブロックです。
58 行目： バイナリファイル読み出し文で、バイナリファイルから属性単精度浮動小数点形式で読み出します。
60 行目：レコード番号 7 フィールド番号1が 4：倍精度浮動小数点形式かチェックします。
61 行目～63 行目：倍精度浮動小数点の場合の読み出し演算処理ブロックです。
62 行目： バイナリファイル読み出し文で、バイナリファイルから属性倍精度浮動小数点形式で読み出します。
64 行目： バイナリファイル読み出し開始バイト位置を更新します。
65 行目： バイナリファイル読み出し文でバイナリ部分の終結行を読み出します。
66 行目： バイナリファイル読み出し開始バイト位置を更新します。
67 行目： 記録されている内容が時間軸ない場合の演算処理ブロックを制御します。
69 行目～72 行目：時間軸で無い場合のフラグ処理演算処理ブロックです。
70 行目： 時間軸で無い場合、変換処理を放棄するため、次の読み出しバイト位置を-1：EOF に達したとします。
71 行目： 処理放棄処理放棄フラグを立てます。
75 行目： 変換処理放棄されずにバイナリファイル全てを読み出したか否かチェックします。
76 行目～87 行目：変換処理結果を波形ファイルとして生成する演算処理ブロックです。
79 行目： 外部演算処理ブロック読み出し文で年月日表現形式を変換します。
80 行目： 生成格納するファイルを現在のファイル名から拡張子を削除して生成します。
81 行目：サンプリング周期をレコード番号 7 フィールド番号 5 から生成します。
82 行目： ファイル番号定義文で格納先ファイルを属性 wav として定義します。
83 行目： 生成波形ファイルのサンプリリング周期および単位を定義します。
84 行目： 生成波形ファイルの収録年月日/時刻を定義します。
85 行目： 生成波形ファイルの格納データ形式を単精度浮動小数点形式として定義します。
86 行目： 生成波形ファイルを格納します。
88 行目： 次のファイル処理のため、ファイルカウンタをインクリメントします。
93 行目～104 行目：年月日表現方法変換演算処理ブロックで DD-MMM-YY から MM-DD-YYYY に変換します。
94 行目： 引き継ぎチャネル定義文で外部演算処理ブロック読み出し文から引き継ぐチャネルを定義します。
95 行目： ローカルチャネル定義文で演算処理ブロック内だけで使用するチャネルを定義します。
96 行目： 変換対象月表現文字列を定義します。
97 行目： 変換後月表現文字列を定義します。
98 行目： 変換対象年月日文字列から月表現部分を抜き出します。
99 行目： 比較するため、抜き出した月表現文字列を 12 か月分用意します。
100 行目：変換対象文字列を 96 行で定義した内容比較し、一致 Index を取得し変換後文字列に指定します。
101 行目：変換対象年月日文字列から日表記部分を抜き出します。
102 行目：変換対象文字列年月日から年表記部分を抜き出します。
103 行目：変換後の年月日文字列を組み立てます。

<参考>

UFF58b フォーマットはテキストで記述されたヘッダー部とバイナリで記述されたデータ部の混合ファイルで、それぞれのブロックの区切りはテキスト行:"-1"で区切られて記述されています。区切りはチャネルごとに存在し、複数チャネルが収録されたファイルの場合、例えば 3ch 収録の場合で説明すると、

ブロック区切り行
ch1 の 58b ヘッダー行
ch1 の 58 レコード行
　　行数は 58b ヘッダー行のフィールド 3 に記載
ブロック区切り行
ch1 のバイナリデータ
　　データバイト数は 58b ヘッダー行のフィールド 4 に記載
ブロック区切り行
ブロック区切り行
ch2 の 58b ヘッダー行、
ch2 の 58 レコード行
　　行数は 58b ヘッダー行のフィールド 3 に記載
ブロック区切り行
ch2 のバイナリデータ
　　データバイト数は 58b ヘッダー行のフィールド 4 に記載
ブロック区切り行
ブロック区切り行
ch3 の 58b ヘッダー行、
ch3 の 58 レコード行
　　行数は 58b ヘッダー行のフィールド 3 に記載
ブロック区切り行
ch3 のバイナリデータ
　　データバイト数は 58b ヘッダー行のフィールド 4 に記載
ブロック区切り行
となります。



① 区切り行の内容

Field1(I6)
-1

② 58b ヘッダー行の内容

Field1(I6)	Field2(1A)	Field3(I6)	Fileld4(I6)	Field5(I12)	Field6(I12)	Field7	Field8	Field9	Field10
58	b	バイト並び	Float 形式	58 行数	バイト数	—	—	—	—

バイト並び(Field3): 1:リトルエンディアン、2:ビッグエンディアン 但しリトルエンディアンのみサポート
Float 形式(Field4): 1:DEC VMS、2:IEEE 754、3:IBM 5/370 但し記述例では IEEE 754 のみサポート
58 行数:続く 58 レコード行数
バイト数:バイナリデータ数
"-"は参照しない

③ 58 ヘッダー(レコード)行の内容 Record
1:ID Line1 (80A1) 未参照 Record2:ID
Line2 (80A1) 未参照
Record3:ID Line3 (80A1) 9A1,1X,8A1 年月日:DD-MMM-YY、時分秒:hh:mm:ss
Record4:ID Line4 (80A1) 未参照
Record5:ID Line5 (80A1) 未参照

Record6 (2,(I5,I10), Record6 (2,(I5,I10),2(1X,10A1,I10,14)

Fileld1	Field2	Field3	Field4	Field5	Field6	Field7	Field8	Field9	Field10
Function	—	—	—	信号名	—	—	—	—	—

Function(Field1) 1:時間応答 1のみサポート

信号名(Field5) 信号名として参照
“-“は参照しない

Record7: (3I10,3E13.5)

FileId1	Field2	Field3	Field4	Field5	Field6
記録形式	データ数	X 軸刻み	X 軸最小値	サンプル周期	—

記録形式(Field1) 2:単精度 Float、4:倍精度 Float、5:複素数単精度 Float、6:複素数倍精度 Float
但し記述例では 2、4 のみサポート
データ数(Field2) データ個数(データ形式換算データ個数) 読み出しデータ個数として参照
X 軸刻み(Field3) 0:X 軸刻み不等間隔、1:X 軸刻み等間隔 記述例では 1 のみサポート
X 軸最小値(Field4) 参照しない 常に X 軸最小値を 0 と見なす
サンプル周期(Filed5) サンプリング周期として参照
“-“は参照しない

Record8:(I10,3I5,2(1X,20A1)

FileId1	Field2	Field3	Field4	Field5	Field6
データ属性	—	—	—	X 軸ラベル	X 軸単位

データ属性(Field1) データ属性コード 17:time のみサポート
X 軸ラベル: 未参照
X 軸単位: サンプリング単位として参照
“-“は参照しない Record9:

(I10,3I5,2(1X,20A1))

FileId1	Field2	Field3	Field4	Field5	Field6
—	—	—	—	データ属性	Y 軸単位

Y 軸単位(Field6) 信号単位として参照
“ー”は参照しない

Record10 および Record11 は参照しない。

記述例15. 2. WAV ファイルを PcWaveForm フォーマットに変換する

wav ファイルで格納されたファイルを読み出し、PcWaveForm フォーマットに変換したファイルを生成します。
変換対象 wav ファイルのデータ形式はリニア PCM で収録されており、チャネル数は 1ch または 2ch であることを前提としています。なお、フォーマット変換された生成波形ファイルの格納先は変換対象 wav ファイルの格納先に拡張子違いで同じファイル名で格納します。

<Archi_1 Script 記述例>

```
1           dcl menu_label "WAV file 変換"
2           dcl sheet 1 "wav file list" {
3             page 1: "RIFF header"
4               column &2,&3,$3,&4,&5,$6,$7,$8,$9,$10,$11,$12,&6,$14
5               format 2(A),F0,2(A),7(F0),A,F0 2
6           }sheet
7           /*-------------------------------------------------------*/
8           def ch_name $1 "RIFF"
9           def ch_name $2 "next_byte_position"
10          def ch_name $3 "file_size"
11          def ch_name $4 "WAVE"
12          def ch_name $5 "FMT "
13          def ch_name $6 "fmt chunk byte"
14          def ch_name $7 "format id"
15          def ch_name $8 "num_ch"
16          def ch_name $9 "sampl_rate:Hz"
17          def ch_name $10 "byte/sec"
18          def ch_name $11 "block_size"
19          def ch_name $12 "data_bit"
20          def ch_name $13 "data"
21          def ch_name $14 "num_data_byte"
22          def ch_name $15 "ch1_data"
23          def ch_name $16 "ch2_data"
24          def ch_name $17 "num_file"
25          def ch_name $18 "file_counter"
26          def ch_name &1 "file_name"
27          /*------------ 処理開始----------------------------------*/
28          get file_name &1 $17 "wav"
29          assign &2 "wav_file_name:" = &1|".wav"
30          case $17 > 0
31            proc exec{
32              $18 = 0
33              repeat_case $18 < $17
34                proc file_loop{
35                  write ch_column 1: &2($18)
36                  def file_id %1 &2($18) bin          /* wav file define */
37                  /*---------- header read ------------------------*/
38                  read bin %1 0 4 $1 $2 byte   /* RIFF */
39                  read bin %1 $2 1 $3 $2 int32 /* file size */
40                  read bin %1 $2 4 $4 $2 byte   /* WAVE */
41                  read bin %1 $2 4 $5 $2 byte   /* FMT */
42                  read bin %1 $2 1 $6 $2 uint32 /* FMT chunk byte */
43                  read bin %1 $2 1 $7 $2 int16   /* format id */
44                  read bin %1 $2 1 $8 $2 uint16   /* num of channel */
45                  read bin %1 $2 1 $9 $2 uint32 /* sampling rate */
46                  read bin %1 $2 1 $10 $2 uint32 /* byte/sec */
47                  read bin %1 $2 1 $11 $2 uint16 /* block size */
48                  read bin %1 $2 1 $12 $2 int16 /* data bit 16 or 8 */
49                  read bin %1 $2 4 $13 $2 byte   /* data */
50                  read bin %1 $2 1 $14 $2 int32   /* num of data byte */
51                  &3 "RIFF 識別子:" = CVTA($1)
52                  &4 "RIFF 種類:" = CVTA($4)
53                  &5 "fmt_chunk_id:" = CVTA($5)
54                  &6 "data_chunk_id:" = CVTA($13)
```

```
55                /* ---------- convert exec ----------*/
56                case &3 = "RIFF"
57                  proc riff{
58                    case &4 = "WAVE"
59                      proc wave_file{
60                        case $7 = 1                 /* linear PCM */
61                          proc convert_exec{
62                            write ch_column 1: &3,$3,&4,&5,$6,$7,$8,$9,$10,$11,$12,&6,$14
63                            write line_feed 1: 1
64                            /* ---------- data read ----------*/
65                            case $12 = 16
66                              proc 16bit{
67                                $14 = $14/2
68                                read bin %1 $2 $14 $15 $2 int16   /* data read */
69                              }16bit
70                            case $12 = 8
71                              proc 8bit{
72                                read bin %1 $2 $14 $15 $2 byte   /* data read */
73                              }8bit
74                            /*---------- file create ----------*/
75                            def file_id %2 &1($18) wav
76                            def wav_sampling %2 0 $9 "sec"    /* サンプリング定義*/
77                            case $8 = 1
78                              proc mono{
79                                save wave %2 $15               /*  ch1  格納 */
80                              }mono
81                            case $8 = 2
82                              proc stereo{
83                                $16 = SEP(1,1,$15)     /* ch2 分解*/
84                                $15 = SEP(0,1,$15)     /* ch1 分解*/
85                                save wave %2 $15,$16     /* ch1.ch2  格納 */
86                              }stereo
87                          } convert_exec
88                        case $7 <> 1      /* linear PCM でない*/
89                          proc not_linear_pcm{
90                            write ch_column 1: &3,&4,$7
91                            write line_feed 1: 1
92                          }not_linear_pcm
93                      }wave_file
94                    case &4 <> "WAVE"    /* RIFF 種類が wave でない*/
95                      proc not_wave{
96                        write ch_column 1: &3,&4
97                        write line_feed 1: 1
98                      }not_wave
99                  }riff
100               case &3 <> "RIFF"         /* RIFF でない*/
101                 proc not_riff{
102                   write ch_column 1: &3
103                   write line_feed 1: 1
104                 }not_riff
105               $18 = $18+1
106             }file_loop
107         }exec
108       end
```

<Archi_1 Script 記述構文の説明>

1 行目:　実行メニューボタンに表示するラベル名を宣言します。
2 行目～6 行目:変換対象 wav ファイルのヘッダーを表示する為の結果シート宣言ブロックです。RIFF の種別が WAVE でない場合やリニア PCM でない場合は、表示される内容はファイル名、RIFF 種別及びフォーマット ID のみとなります。
28 行目: ファイル選択文で変換する wav ファイルを選択します。なお、同時に複数ファイルを選択できます。
29 行目: 取得したファイル名には拡張子が付いていませんので拡張子文字列".wav"を付加します。
30 行目: 選択されたファイル数をチェックし、ファイルが選択されていない場合、実行終了します。
31 行目～107 行目:変換処理する演算処理ブロックです。
32 行目: ファイルカウンタを初期化します。
33 行目: 選択されたファイル数分、直下の演算処理ブロックを繰り返します。
34 行目～106 行目:ファイルごとに処理する演算処理ブロックです。
35 行目: 処理するファイル名を結果シートに書き込みます･
36 行目: wav ファイルのファイル番号を定義します。
38 行目～50 行目:wavファイルの RIFF ヘッダーを読み出し処理を行います。
38 行目: RIFF 識別子を byte 属性で読み出します。
39 行目: ファイルサイズを int32 属性で読み出します。
40 行目: RIFF 種別を byte 属性で読み出します。
41 行目: フォーマット chunk_id を byte 属性で読み出します。
42 行目: フォーマット chunk バイト数を uint32 属性で読み出します。
43 行目: フォーマット ID を int16 属性で読み出します。
44 行目: チャネル数を uint16 属性で読み出します。
45 行目: サンプリング周波数を uint32 属性で読み出します。
46 行目: 転送速度を uint32 属性で読み出します。
47 行目: ブロックサイズを uint16 属性で読み出します。
48 行目: データ bit 長を int16 属性で読み出します。
49 行目: データ chunk_id を byte 属性で読み出します。
50 行目: 収録データバイト数を int32 属性で読み出します。
51 行目: RIFF 識別子を ascii 文字列に変換します。
52 行目: RIFF 種別を ascii 文字列に変換します。
53 行目: フォーマット chunk_id を ascii 文字列に変換します。
54 行目: データ chunk_id を ascii 文字列に変換します。
56 行目: RIFF 識別子が"RIFF"の場合、直下の演算処理ブロックを実行します。
53 行目～99 行目:RIFF 識別子が"RIFF"の時に実行する演算処理ブロックです。
58 行目: RIFF 種別が"WAVE"の場合、直下の演算処理ブロックを実行します。
59 行目～93 行目:RIFF 識別子が"RIFF"で種別が"WAVE"の時に実行される演算処理ブロックです。
60 行目: フォーマット ID がリニア PCM の場合、直下の演算処理ブロックを実行します。
61 行目～87 行目:RIFF 識別子が"RIFF"、種別が"WAVE"でフォーマット ID がリニア PCM の時に実行される演算処理ブロックです。
62 行目: 結果シートに読み出した RIFF ヘッダーの内容を書き込みます。
65 行目: データの bit 長が 16bit の時に直下の演算処理ブロックを実行します。
66 行目～69 行目:データ bit 長が 16bit の時に実行されるデータ読み出し演算処理ブロックです。
67 行目: 収録データバイト数をワード数に変換します。
68 行目: wav ファイルから属性 int16 でデータを読み出します。
70 行目: データの bit 長が 8bit の時に直下の演算処理ブロックを実行します。
71 行目～73 行目:データ bit 長が 8bit の時に実行されるデータ読み出し演算処理ブロックです。
72 行目: wav ファイルから属性 byte でデータを読み出します。
75 行目: 生成する波形ファイル番号を定義します。
76 行目: 生成する波形ファイルのサンプリング周波数を定義します。
77 行目: チャネル数が 1ch の時、直下の演算処理ブロックを実行します。
78 行目～80 行目:1ch データ格納演算処理ブロックです。
79 行目: 波形ファイルを格納生成します。
81 行目: チャネル数は 2ch の時、直下の演算処理ブロックを実行します。
82 行目～86 行目:2ch データ格納演算処理ブロックです。
83 行目: 読み出したデータから ch2 データを分離します。
84 行目: 読み出したデータから ch1 データを分離します。
85 行目: 波形ファイルを格納生成します。
88 行目～92 行目:データ形式がリニア PCM で無い場合の結果シート書き込み処理を行います。
94 行目～98 行目:RIFF 種別が"WAVE"で無い場合の結果シート書き込み処理を行います。
100 行目～104 行目:RIFF 識別子が"RIFF"で場合場合の結果シート書き込み処理を行います。
105 行目: ファイルファイルカウンタをインクリメントします。

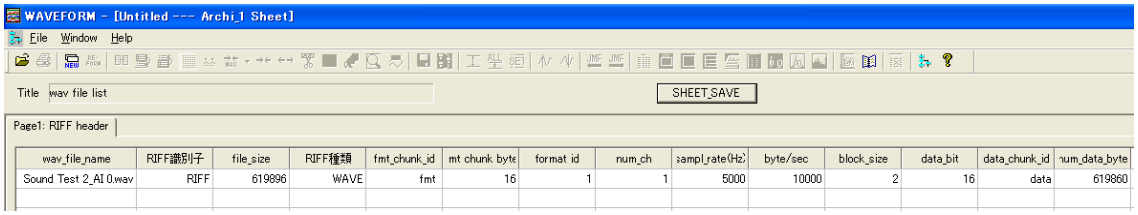
<参考>

wav ファイルフォーマットはバイナリファイルで構成され、先頭に RIFF ヘッダーが付加されています。ヘッダー部の構成を示します。

byte 数	属性	内容	備考
4byte	文字列	RIFF 識別子	“RIFF”
4byte	int32	ファイルサイズ-8	
4byte	文字列	RIFF 種別	“WAVE”
4byte	文字列	fmt chunk id	“fmt “
4byte	uint32	fmt chunk byte 数	リニア PCM の場合 16
2byte	int16	フォーマット ID	リニア PCM の場合1
2byte	uint16	チャネル数	通常 1 または 2
4byte	uint32	サンプリング周波数 Hz	小数点以下は存在しない
4byte	uint32	データ転送速度 byte/sec	
2byte	uint16	ブロックサイズ	byte/sampleXch 数
2byte	int16	データ bit 長	通常 16bit または 8bit
2byte		拡張部のサイズ(バイト数)	リニア PCM では存在しない
nbyte		拡張部	リニア PCM では存在しない
4byte	文字列	data chunk id	“data”
4byte	int32	データ部サイズ(byte 数)	

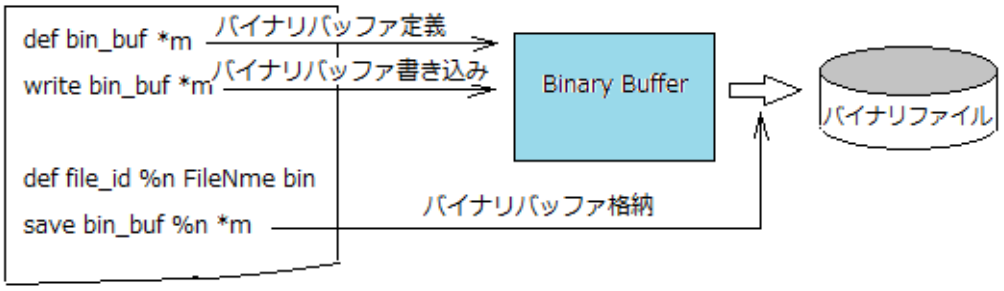
※ リニア PCM のみ変換対象としているため、拡張部サイズおよび拡張部が存在しないものとしています。
※ リニア PCM の場合、ヘッダーChunk のサイズは 44byte です。
※ wav ファイルのデータは物理量ではありません。

<変換後表示される結果シート例>



16. バイナリファイルを生成する

バイナリファイルを生成する場合、一旦、バイナリバッファ定義文で定義したバイナリバッファに型式指定して書き込み、書き込まれた バイナリバッファをファイル番号定義文で定義したファイルに格納することで生成します。



16. 1. バイナリファイル生成事前処理

ファイル番号定義文

文法:

def file_id %ファイル番号　ファイル名 属性

属性は bin を指定します。また、ファイル名は必ず拡張子を付けて記述します。

※ ファイル番号定義文で直接ファイル名を記述せず、Script 実行時に読み出すファイルを選択する方法については、第 5 章「解析対象ファイルを読み出す」の 5.3、5.4 項を参照下さい。

記述例:

```
def file_id %1 “test.dat” bin
```

16. 2. バイナリバッファの定義

バイナリファイルを生成する場合、ファイルに直接書き込むのではなく、一旦、内部に定義したバイナリバッファに書き込み、書き込 んだバイナルバッファを格納することで生成します。

バイナリバッファ定義文

文法:

def bin_buf *バッファ番号 バッファサイズ

バッファ番号は先頭文字半角”*”に続き数字で記述します。

バッファサイズは byte 単位で記述します。0 または 1 は特別な意味を持ち、0 は既に定義されているバイナリバッファを削除し、-1 は現在書き込まれているバイナリバッファの書き込み最終アドレス以降を削除し、再構成したバッファサイズを戻します。従って、-1 を記述する場合は必ず数値属性参照チャネルで記述します。なお、書き込みを行っていないバイナリバッファに-1 を記述した場合は 0 と記述したことと等価となります。

記述例:

```
def bin_buf *1 8192        /*  バッファ番号1を8192byte で定義*/
def bin_buf *2 0           /* 定義済みバッファ番号 2 を削除*/
$1 = -1
def bin_buf *3 $1          /* 定義済みバッファ番号 3 を再構成*/
```

16. 3. バイナリバッファへのデータの書き込み

バイナリバッファ書き込み文

文法:

write bin_buf *バッファ番号 開始アドレス 書き込みチャネル列 書き込み後アドレス　書き込み方式

開始アドレスはバッファ先頭アドレスを 0 とした byte 単位アドレスを意味します。 書き込みチャネルは書き込むチャネル番号と型式キーワードを半角コロン”:”で区切って記述します。 複数チャネルを記述する場合は書き込みチャネルと型式キーワードのペアを半角カンマで区切って記述します。

※ 即値、文字列即値は直接書き込みできません。一旦、参照チャネルに代入して書き込みます。

※ 書き込みチャネルの Index 指定はできません。従って、数列構成を持つ参照チャネルの特定要素のみ書き込む場合は、一旦、他の参照チャネルに Index を指定して代入し、代入された参照チャネルを書き込みチャネルとして記述します。

書き込みチャネル属性キーワード表を示します。

書き込みキーワード	内容	記述チャネル属性
byte	0～255 の数値	数値属性
int16	符号付き整数形式(2byte)	数値属性
uint16	符号なし整数形式(2byte)	数値属性
int32	符号付き倍精度整数形式(4byte)	数値属性
uint32	符号無し倍精度整数形式(4byte)	数値属性
float32	単精度浮動小数点形式(4byte)	数値属性
float64	倍精度浮動小数点形式(8byte)	数値属性
文字数※1	文字列	文字属性

文字列を書き込む場合、半角換算文字数を数値で指定します。書き込む文字列が指定した文字数に満たない場合は余りに null が書き込まれ、書き込む文字列が文字数より長い場合は指定された文字数分のみ書き込まれます。

記述例:

#1 を 2byte 符号付き整数、$2 を 8byte 浮動小数点でバイナリバッファ:1 に書き込む場合

```
def bin_buf *1 1000000
$1 "addr:" = 0
write bin_buf *1 $1 #1:int16,$2:float64 $1 0
```

書き込み後アドレスは、書き込んだ最終アドレス+1 が戻ります。

書き込み方式は 0 または 1 を記述します。0 はノンインターレス、1 はインターレスとなります。なお、記述省略した場合は 0 と見なされます。

※ 書き込み方式にインターレースを指定した場合、書き込みチャネル列の Index 順に書き込まれます。チャネル列に記述したチャネル要素数が異なる場合は、最も要素数の少ないチャネルに整合されます。残りは書き込まれません。

記述例:

バイナリバッファ*1 に文字列(10文字)、数列(4byte 浮動小数点形式)を書き込む場合

```
def bin_buf *1 4096
assign $1 = 0,1,2
assign &1 = "東京","名古屋","新大阪"
$3 = 0
write bin_buf *1 $3 &1:10,$1:float32 $3 0
```

書き込まれたバイナリバッファの内容

先頭アドレス	書き込まれる内容	バイト数	形式
0	東京	10	文字列
10	名古屋	10	文字列
20	新大阪	10	文字列
30	0	4	float32
34	1	4	float32
38	2	4	float32

戻り$3 には、42 が戻ります。

16. 4. バイナリバッファを格納する

バイナリファイル格納文

文法:

```
save bin_buf %ファイル番号 *バッファ番号列 フラグ
```

ファイル番号は予めファイル属性 bin で定義したファイル番号を記述します。 バッファ番号列は書き込むバイナリバッファを半角カンマで区切って記述します。

格納は記述されたバッファ番号順に行われます。

フラグは、追記処理するか、上書きするかのフラグで 0 または記述省略した場合は上書き、1を記述した場合は既に書き込まれているファイルの最終アドレスから追記されます。なお、新たに生成するバイナリファイルに最初に格納する場合は、0 または 1 でも 0 と同じ意味を持ちます。

記述例:

バイナリバッファ番号 1 と 2 を格納する場合

```
def file_id %1 "test.dat" bin                    /*格納先ファイル番号定義*/
def bin_buf *1 4096                              /*バイナリバッファ No.1 定義*/
def bin_buf *2 4096                              /*バイナリバッファ No.2 定義*/
/*------格納波形、信号名生成----------------------*/
def sampl_period 1e-4 "sec"
$1 "sine_wave1:" = DAG(1000,10,0,20,0)
$2 "sine_wave2:" = DAG(1000,5,0,10,0)
assign &1 = "20Hz 正弦波","V"
```

```
sssign &2 = "10Hz 正弦波","V"
$3 = LEN($1)
$4 = LEN($2)
/*----------バイナリバッファ書き込み、ファイル格納----------*/
write bin_buf *1 0 &1:16,$3:uint32,$1:float32 $5 /*バイナリバッファ No.1 に書き込み*/
write bin_buf *2 0 &2:16,$4:uint32,$2:float32 $5 /*バイナリバッファ No.2 に書き込み*/
save bin_buf %1 *1,*2     /*バイナリファイル生成*/
end
```

記述例16. 1. PcWaveForm ファイルから WAV ファイルを生成する

PcWaveForm の波形表示 Window で解析範囲を指定したカレントチャネルをリニア PCM、データ形式 16bit 整数形式で WAV 形式フォーマットに変換します。

<Archi_1 Script 記述例>

```
1       /*--- PcWaveForm to WAV Format Convert ----------------------*/
2       dcl menu_label "解析範囲 WAV変換"
3       $1 "current_CH_No.:" = CCH()                /* カレントチャネル番号取得*/
4       case $1 > 0
5         proc exec{
6           /*-----header chunk write-------------------------------*/
7           def bin_buf *1 44                        /* Header_Buffer 定義 */
8           $3 "write_addr:" = 0
9           assign &1 = "RIFF"
10          write bin_buf *1 $3 &1:4 $3              /* Chunk_ID*/
11          $4 "file_size:" = LEN(#($1))*2+44-8
12          write bin_buf *1 $3 $4:int32 $3         /* File_size*/
13          assign &1 = "WAVE"
14          write bin_buf *1 $3 &1:4 $3              /* RIFF 識別*/
15          assign &1 = "fmt "
16          write bin_buf *1 $3 &1:4 $3              /* Chunk_ID*/
17          $5 "temp1:" = 16
18          write bin_buf *1 $3 $5:uint32 $3       /* fmt_Chunk Size*/
19          $5 = 1
20          write bin_buf *1 $3 $5:int16 $3        /* Format=Linear*/
21          write bin_buf *1 $3 $5:uint16 $3       /* Num_Ch*/
22          $7 "sampling:Hz" = INT(1/PRD())
23          write bin_buf *1 $3 $7:uint32 $3       /* SampleFrwquency*/
24          $7 = $7*2
25          write bin_buf *1 $3 $7:uint32 $3       /* transfer*/
26          $5 = 2
27          write bin_buf *1 $3 $5:int16 $3        /* block_size*/
28          $5 = 16
29          write bin_buf *1 $3 $5:int16 $3        /* Data_Length*/
30          assign &1 = "data"
31          write bin_buf *1 $3 &1:4 $3             /* Data_Chunk_ID*/
32          $6 "buf_size2:" = $4-44+8
33          write bin_buf *1 $3 $6:int32 $3        /* Data_Block_Size*/
34          /*------ WAV data convert ------------------------------*/
35          def bin_buf *2 $6                        /* *2   Data_Buffer 定義*/
36          $2 "WAV_data:" = #($1)*30000/MAX(#($1))  /*データ整数変換*/
37          write bin_buf *2 0 $2:int16 $3         /* WAV Data_Write*/
38          /* -----WAV file save-----------------------------------*/
39          &2 "解析対象ファイル名:" = CFNM()      /* 解析ファイル名取得*/
40          assign &2 = &2|"_"|$1(F0)|"ch.wav"
41          def file_id %1 &2 bin                  /* 格納先ファイル番号定義*/
42          save bin_buf %1 *1,*2                   /* バッファ格納*/
43        }exec
44      case $1 < 0
45        proc error{
46          disp message "解析範囲が選択されていません!!"
47        }error
48      end
```

<Archi_1 Script 記述構文の説明>

3 行目： 解析範囲で指定されているカレントチャネル番号を取得します。
4 行目: カレントチャネルが指定されていない場合(波形表示 Window から Script を実行していない)のチェックを行います。
7 行目～34 行目:WAV ファイルのヘッダー部に必要な情報を書き込みます。
7 行目： ヘッダー部書き込み用バイナリバッファをバッファ番号1、サイズ 44byte で定義します。
8 行目： バッファ書き込み用アドレスを 0(先頭)に初期化します。
9 行目～10 行目:RIFF 識別子"RIFF"をバッファ 1 に書き込みます。
11 行目: ファイルサイズを演算します。ファイルサイズは全体サイズ-8 とします。
12 行目: ファイルサイズをバッファ 1 に書き込みます。
13 行目～14 行目:RIFF 識別子"WAVE"をバッファ 1 に書き込みます。

15 行目～16 行目:ChunkID"fmt "をバッファ1に書き込みます。
17 行目～18 行目:fmt Chunk サイズをバッファ1に書き込みます。リニア PCM の場合は 16byte とします。
19 行目～20 行目:フォーマット ID をバッファ 1 に書き込みます。リニア PCM の場合は 1 とします。
21 行目: チャネル数をバッファ 1 に書き込みます。本 Script では 1ch 再生です。
22 行目～23 行目:サンプリング周波数をバッファ 1 に書き込みます。
24 行目～25 行目:データ転送速度をバッファ 1 に書き込みます。
26 行目～27 行目:ブロックサイズをバッファ 1 に書き込みます。
28 行目～29 行目:データ bit 長をバッファ 1 に書き込みます。
30 行目～31 行目:データ ChunkID 識別子"data"をバッファ 1 に書き込みます。
32 行目～33 行目:データサイズをバッファ 1 に書き込みます。
35 行目:データ部書き込み用バイナリバッファをバッファ番号 2、サイズ データ個数×2 で定義します。
36 行目: 波形データを最大値を 30000 として 2byte 整数形式に変換します。
37 行目: 変換した 2byte 整数形式データをバッファ 2 に書き込みます。
39 行目: 格納ファイル名を生成する為、解析対象ファイルを取得します。
40 行目: 格納先ファイル名を生成します。
41 行目: 格納先ファイル番号を定義します。
42 行目: バッファ1とバッファ 2 を格納します。

<補足説明>
本 Script では解析対象ファイルのサンプリング周波数をそのまま WAV ファイルの設定に使用しています。従って、計測データを変換する場合、音として再生するとデータの周波数が低過ぎ、可聴範囲に無い場合があります。その場合、WAV ファイルに設定するサンプリング周波数を修飾して、例えば 1/2 倍で設定すると 1 オクターブ高く再生されます。

17. 演算関数を使用して数列の要素の削除/抽出/置換/連結/縫合/反転/並べ替え/生成を行う

Archi_1 で記述する収録チャネル、数値属性参照チャネルはすべて複数要素を持つ数列として扱いますので、数列から指定した要素の削除或いは抽出、要素の置換、数列同士の連結、縫合、数列の反転、並べ替え、生成などの操作は作成する Script 中で頻繁に行われます。その操作は Script 文ではなく、演算関数を使用して行います。 演算関数の引数には、即値、収録チャネル、数値属性参照チャネル、演算式が記述できます。以下説明の都合上、複数要素を持 つ数列で記述可能な引数は引数の意味+数列と記載、数列で記述しても参照される Index が 0 のみの場合は引数の意味のみで記 載しています。

17. 1. 数列要素の削除/抽出

17. 1. 1. 数列から削除する Index を指定し、指定した要素を削除する

【DEL 関数】を使用します。 **文**

法:

結果格納先 = DEL(削除する Index 数列,対象数列)

引数:

【削除する Index】＜必須＞

対象数列から削除する Index を記述します。複数の削除対象 Index を指定する場合は、数列で記述し必ず昇順並びの必要があります。又、指定する Index は対象数列に含まれている必要があります。

【対象数列】＜必須＞

演算対象数列を記述します。

記述例:

$2 の$2(2)を削除する場合

```
assign $2 = 1,2,3,4,5,6,7,8,9,10
$1 = DEL(2,$2)
```

$1 には、1,2,4,5,6,7,8,9,10 が格納されます。

記述例:

$2 の$2(0),$2(2),$2(4)の 3 個を削除する場合で DEL 関数をネストして記述する場合

```
assign $2 = 1,2,3,4,5,6,7,8,9,10
$1 = DEL(0,DEL(2,DEL(4,$2)))
```

$1 には、2,4,6,7,8,9,10 が格納されます。

※ 最深部から最浅部に向かって削除する Index が降順である必要があります。

記述例:

$2 の$2(0),$2(2),$2(4)の 3 個を削除する場合で Index を数列で指定する場合

```
assign $2 = 1,2,3,4,5,6,7,8,9,10
assign $3 = 0,2,4
$1 = DEL($3,$2)
```

$1 には、2,4,6,7,8,9,10 が格納されます。

17. 1. 2. 数列と同じ要素数の論理値数列を指定し、論理値数列の論理 0 の要素を削除する

【ZSP 関数】を使用します。 **文**

法:

結果格納先 = ZSP(論理値数列,対象数列) 引

数:

【論理値数列】＜必須＞

対象数列に対応した 0 または 1 からなる論理値数列を記述します。論理数列の要素値 0 の Index を対象数列から削除します。※ 論理値数列の要素数は対象数列と等しい必要があります。

【対象数列】＜必須＞ 演算対象

数列を記述します。

記述例:

$2 の$2(2)を削除する場合

```
assign $2 = 1,2,3,4,5,6,7,8,9,10
assign $3 = 1,1,0,1,1,1,1,1,1
$1 = ZSP($3,$2)
```

$1 には、$1⇒1,2,4,5,6,7,8,9,10 が格納されます。

記述例:

$2 の要素値が$2(n)>=3 & $2(n)<=7 の範囲を抽出する場合

```
assign $2 = 1,2,3,4,5,6,7,8,9,10
$1 = ZSP(AND(GTE($2,3),LTE($2,7)),$2)
```

$1 には、3,4,5,6,7 が格納されます。

記述例:

$2 の要素から$2(n)=0 を削除する場合(0 以外を抽出する場合)

```
assign $2 = 0,1,2,0,1,2,3,0,1,2,3
$1 = ZSP(NEQ($2,0),$2)
```

$1 には、1,2,1,2,3,1,2,3 が格納されます。

17. 1. 3. 数列の削除振幅閾値を指定し、閾値以上の振幅範囲を削除する

【LMT 関数】を使用します。

文法:

格納先 = LMT(振幅閾値,対象数列,フラグ) 引

数:

【振幅閾値】<必須>

削除対象とする振幅値を記述します。ここで記述した振幅範囲を削除します。振幅範囲とは対象数列がゼロを上昇で過り、再びゼロを上昇で過る範囲の両振幅値を意味し、振幅閾値を越えた場合の上昇で過ってから再び上昇で過る範囲を意味します。※ 閾値は複数要素を持つ数列で指定しても参照される閾値は Index0 のみとなります。

【対象数列】<必須>

演算対象数列を記述します。

【フラグ】<省略可>

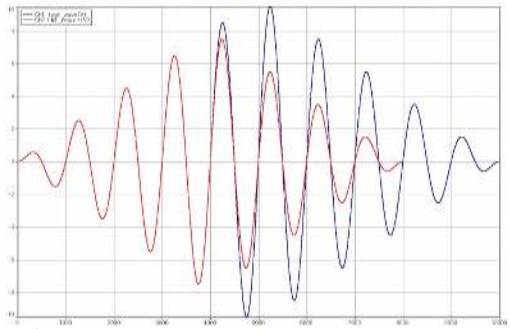
0 または1を記述します。フラグに 0 を記述するか記述省略した場合は、対象数列の振幅範囲の両振幅値が振幅閾値を越えた範囲を削除します。格納される数列は削除範囲分短くなります。フラグに 1 を記述した場合は、対象数列の振幅範囲の両振幅値が振幅閾値を越えた範囲の開始 Index と終了 Index のペアが格納されます。振幅閾値を越える範囲が複数存在する場合は、その回数×2 の index 数列となります。

記述例:

$1 に周波数 10Hz の正弦波で最大振幅 10V の試験波形を、両振幅が 15V を越える範囲を削除去する場合

```
def sampl_period 1e-4 "sec"
$1 "test_wave:V" = DAG(10000,10,0,10,0)*VRP(1,10000,5000,1)
$2 "LMT_result:V" = LMT(15,$1)
```

$2 に格納される数列をグラフ赤色、除去前の数列を紺色で示します。



※ $2 の点数は削除範囲分少なくなります。

17. 1. 4. 数列から抽出する Index を指定し、指定された Index の要素を抽出する

【PTV 関数】を使用します。

文法:

格納先 = PTV(抽出する Index 数列,対象数列)

引数:

【抽出する Index 数列】<必須>

対象数列から抽出する Index を記述します。抽出する Index が対象数列に存在していれば並び順に拘りません。

※ 格納個数は指定した Index 個数に依存します、

【対象数列】<必須> 演算対象

数列を記述します。

記述例:

$2 から$2(2)を抽出する場合
```
assign $2 = 1,2,3,4,5,6,7,8,9,10
$1 = PTV(2,$2)
```
$1 には、$1⇒3 が格納されます。

記述例:

$2 から$2(0),$2(2),$2(4)を抽出する場合
```
assign $2 = 1,2,3,4,5,6,7,8,9,10
assign $3 = 2,4,0
$1 = PTV($3,$2)
```
$1 には、$1⇒1,3,5 が格納されます。

17. 1. 5. 数列の抽出開始 Index と飛び越し数を指定し、要素を分離抽出する

【SEP 関数】を使用します。 **文**

法:

格納先 = SEP(抽出開始 Index,飛越数,対象数列) 引

数:

【抽出開始 Index】＜必須＞

対象数列から抽出開始する Index を記述します。

【飛越数】＜必須＞

飛越数とは対象数列から抽出する間隔を意味します。例えば 1 とした場合、1 個おきに抽出します。

【対象数列】＜必須＞ 演算

対象数列を記述します。

※ 格納個数は抽出開始 Index 及飛越し数と対象数列の個数に依存します。

記述例:

$2 の Index 偶数の値を$1 に、Index 奇数の値を$3 に格納する場合
```
assign $2 = 1,2,3,4,5,6,7,8,9,10
$1 = SEP(0,1,$2)
$3 = SEP(1,1,$2)
```
$1 には、1,3,5,7,9 が格納されます。
$3 には、2,4,6,8,10 が格納されます。

$2 が仮想 2 次元行列構造を持つ場合の列を抽出する時にも使用できます。

格納先 = SEP(抽出列番号-1,列数-1,行列構造数列)

記述例:

6 チャネル/データ数 1000 点のインターレス形式で$2 にデータ格納されているとし、500 点目の全てのチャネルのデータ(行列の 500 列目)を抽出する場合
```
$1 = SEP(500-1,1000-1,$2)
```
$1 には、$2 から 500 列目の 6 点が格納されます。

17. 1. 6. 数列の切り出し始端 Index と終端 Index を指定し、指定範囲の要素を抽出する

【ERC 関数】を使用します。

文法:

格納先 = ERC(切り出し始端 Index,切り出し終端 Index,対象数列)

引数:

【切り出し始端 Index】＜必須＞ 対象数列から切り出す

開始 Index を記述します。

【切り出し終端 Index】＜必須＞ 対象数列から切り出す

終端 Index を記述します。

※ 切り出す要素は始端から終端まで連続して抽出します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

記述例:

$2 の Index10 から 100 個を抽出する場合
```
$3 = 10
$1 = ERC($3,$3+99,$2)
```
$1 には、$2 の$2(10)～$2(109)の 110 個が格納されます。

ERC 関数は数列の Index 範囲を指定した記述と等価な場合があります。
Index 範囲を指定する記述は、[ ]内に始端 Index とデータ個数を半角カンマで区切って記述します。

記述例:

$2 の$2($3)から 110 個の要素値を抽出する場合で ERC 関数を使用しない場合

```
$3 = 10
$1 = $2[$3,110]
```

$1 には、$2 の Index10 から 110 個が格納されます。

前述の ERC 関数を使用した場合と同じ結果を得ます。数列の Index 範囲指定は ERC 関数、Index 範囲指定でも演算式で記述できますが、Index 範囲指定記述では対象数列を演算式で記述できません。
ERC 関数では対象数列が演算式でも記述できる点にあります。

記述例:

対象数列を演算式で記述し、ERC 関数で範囲指定して抽出する場合

```
$6 = ERC(1,LEN($4)-1,SGN(REV($4-$4($0))+$4(0)))
```

$6 には、$4 の並び順を逆転し符号反転した結果の index1 から終端まで格納されます。

ERC 関数は数列が仮想 2 次元行列構造を持つ場合の行を抽出する時にも使用できます。 格納先 = ERC(列数×(抽出行番号-1),列数×抽出行番号-1,行列構造数列)

記述例:

$2 が行列構造を持ち$4 列の構造を持つ場合に$2 の 12 行目を抽出する場合

```
$1 = ERC($4*(12-1),$4*12-1,$2)          /* ERC 関数を使用した場合*/
$1 = $2[$4*(12-1),$4]                   /* Index 範囲指定して記述した場合*/
```

$1 には、$2 の 12 行目が格納されます。
※ $2 の要素数は列数$4 で 12 行以上の行列構造であることがで前提となります。

17. 1. 7. 無効振幅を指定し波形数列の山谷位置の要素を抽出する

【PVF 関数】を使用します。 **文法:**

```
格納先 = PVF(抽出対象コード,無効振幅,対象数列)
```

引数:

【抽出対象コード】＜必須＞ 抽出対象をコードで記述します。
山値を抽出する場合:3、谷値を抽出する場合:5、山値と谷値を同時に抽出する場合:1 と記述します。
※ PVF 関数は山位置 Index 或いは谷値 Index の検索も行えます。Index の検索方法は 18 章数列の Index 検索を参照下さい。

【無効振幅】＜必須＞ 無効振幅値を記述します。無効振幅とは、対象数列のデータで構成される振幅値がここで記述された無効振幅に満たない場合は構成する山値、谷値を抽出対象としないことを意味します。なお、0 と記述した場合、無効振幅処理を行わないことを意味します。

【対象数列】＜必須＞ 演算対象
数列を記述します。

記述例:

収録データ ch1 から無効振幅値を 0 として全ての山値谷値を抽出する場合

```
$1 = PVF(1,0,#1)      /*全ての山谷値*/
$2 = PVF(3,0,#1)      /*全ての山値*/
$3 = PVF(5,0,#1)      /*全ての谷値*/
```

$1 には、ch1 の全ての山谷値が、$2 には、ch1 の全ての山値が、$3 には、ch1 の全ての谷値が格納されます。

17. 1. 8. 数列の要素に同じ値が存在した場合、削除し要素を唯一無二にする

【UNQ 関数】を使用します。 **文法:**

```
格納先 = UNQ(対象数列) 引数:
```

【対象数列】＜必須＞ 演算対象
数列を記述します。
※ 数列は昇順並びでないと結果は保証されません。

※ UNQ 関数は主にランクを求める場合に使用します。

記述例:

$2 の同値除去を行う場合

```
assign $2 = 0,2,3,3,5,6,6,7,8
$1 = UNQ($2)
```

$1 には、0,2,3,5,6,7,8 が格納されます。

17. 2. 数列要素の置換

17. 2. 1. 数列の置換開始 Index を指定し、指定した数列の内容に置換する

【REP 関数】を使用します。

文法:

格納先 = REP(置換開始 Index、置換数列、対象数列)

引数:

【置換開始 Index】＜必須＞

対象数列の置換開始 Index を記述します。

【置換数列】＜必須＞

置換数列を記述します。置換は対象数列の指定した置換開始 Index から置換数列の長さ分が置換されます。 但し、対象数列の終端を越える部分は置換されません。

記述例:

$2 の$2(2)を$3(0)に、$2(3)を$3(1)に、$2(4)を$3(2)に置換する場合

```
assign $2 = 0,1,2,3,4,5,6,7,8,9
assign $3 = 10,20,30
$1 = REP(2,$3,$2)
```

$1 には、0,1,10,20,30,5,6,7,8,9 が格納されます。

記述例:

置換数列の要素個数が対象数列より多い場合

```
assign $2 = 0,1,2,3,4,5,6,7,8,9
assign $3 = 10,20,30,40,50,60,70,80,90,100,110,12,0,130
$1 = REP(0,$3,$2)
```

$1 には、10,20,30,40,50,60,70,80,90,100 が格納されます。

17. 2. 2. 数列の置換する Index を指定し、指定した内容に置換する

【SBV 関数】を使用します。

文法:

格納先 = SBV(置換 Index 数列,置換値数列,対象数列)

引数:

【置換 Index】＜必須＞

対象数列の要素を置換する Index を記述します。置換 Index 数列の並び順は昇順である必要があります。

【置換値数列】＜必須＞

置換 Index 数列に対応した置換値数列を記述します。置換 Index 数列と置換数列の要素数は等しい必要があります。

【対象数列】＜必須＞ 演算対象

数列を記述します。

記述例:

$2 の$2(2)を$3(0)に、$2(3)を$3(1)に$2(7)を$3(2)に置換する場合

```
assign $2 = 0,1,2,3,4,5,6,7,8,9
assign $3 = 10,20,30
assign $4 = 2,3,7
$1 = SBV($4,$3,$2)
```

$1 には、0,1,10,20,4,5,6,30,9 が格納されます。

※ SBV 関数と REP 関数の違いは、REP 関数が置換開始 Index を指定し、その Index から連続して置換数列の内容に置換しますが、SBV は置換 Index 数列の要素ごとに置換数列の Index 順に参照される点が異なります。

17. 3. 数列同士の連結

17. 3. 1. 連結する数列を複数記述し、記述順に連結する

【LNK 関数】を使用します。

文法:

格納先 = LNK(対象数列 A,対象数列 B,対象数列 C…対象数列 J) 引

数:

【対象数列】＜必須＞

連結対象数列を記述します。記述出来る対象数列の数は 2～10 個の範囲となります。なお、記述数列は同じ数列を記述しても問題ありません。

記述例:

$2、$3、$4 を連結する場合

```
assign $2 = 0,1,2,3
assign $3 = 4,5,6,7
assign $4 = 8,9,10,11
$1 = LNK($2,$3,$4)
```

$1 には、$2 の終端に$3 が、$3 の終端に$4 が連結され 0,1,2,3,4,5,6,7,8,9,10,11 が格納されます。

同じ数列を繰り返して連結する場合は、Script 構文の代入文【assign 文】で記述します。

記述例:

数列$1 を 100 個連結した数列を生成する場合

```
assign $1 = 1,2,3,4,5
assign $2  100<$1>
```

$2 には、12,3,4,5 が 100 回繰り返された数列が格納されます。

記述例:

$1 に 0 から 9 までを連結した数列を作る場合

```
$2= 0
repeat_case $2 < 10
   proc 連結{
      $1 = LNK($1,$2)          /* $1は null チャネル*/
      $2 = $2+1
    }連結
```

$2には、0,1,2,3,4,5,6,7,8,9 が格納されます。$1 の初期値は直前までに値が代入されていない null チャネルを記述します。
LNK 関数は例外的に連結するチャネルに null チャネルを記述できます。

17. 3. 2. 連結する数列を複数記述し、記述順に飛越連結する

LNK 関数では、数列の記述順に連結されますが、飛越連結とは数列の記述順、Index 順に連結することを意味します。

【INL 関数】を使用します。 **文法:**

格納先 = INL(対象数列 A,対象数列 B,対象数列 C…対象数列 J) 引

数:

【対象数列】＜必須＞

連結対象数列を記述します。記述出来る対象数列の数は 2～10 個の範囲となります。記述数列は同じ要素数である必要があります。なお、同じ数列を複数記述しても問題ありません。

記述例:

$2、$3、$4 を連結する場合

```
assign $2 = 0,1,2,3
assign $3 = 4,5,6,7
assign $4 = 8,9,10,11
$1 = LNL($2,$3,$4)
```

$1 には、$2(0),$3(0),$4(0),$2(1),$3(1),＄4(1),$2(2),,,,の様に連結され 0,4,8,1,5,9,2,6,10,3,7,11 が格納されます。

17. 4. 数列同士の縫合

17. 4. 1. 2 つの数列を記述し、要素ごと、交互に縫合する

【MGR 関数】を使用します。

文法:

格納先 = MGR(対象数列 A,対象数列 B)

引数:

【対象数列】＜必須＞

縫合対象数列 A と対象数列 B を記述します。縫合は対象数列 A と B を Index 順に交互に参照して結果を戻します。対象数列 A のデータ個数と対象数列 B のデータ個数の何れか多い方の余った要素は縫合されずそのまま連結されます。

記述例:

対象数列 A と対象数列 B の要素ごとに交互に縫合する場合

```
assign $1 = 1,2,3,4,5,6,7
assign $2 = 8,9,10,11,12,13,14,15,16
$3 = MGR($1,$2)
```

$3 には、1,8,2,9,3,10,4,11,5,12,6,13,7,14,15,16 が格納されます。

17. 4. 2. 2 つの数列の要素の大小関係を比較し、何れかの要素を優先して縫合する

【MGR 関数】を使用します。17. 4. 1と同じ MRG 関数を使用しますが、フラグ記述を追加します。フラグ記述を省略した場合は 16.4.1 と同じ機能となります。

文法:

格納先 = MGR(対象数列 A,対象数列 B,フラグ)

引数:

【対象数列】＜必須＞

縫合対象数列 A と対象数列 B を記述します。

【フラグ】＜省略可＞ 縫合に際しての大小関係優先フラグを記述します。フラグに 1 を記述した場合:

対象数列 A と B の大小関係が比較され小さい方を優先して格納します。初めに A(0)<=B(0)が比較され成立すると A(0)が格納され次に A(1)<=B(0)が比較され、成立すると A(1)が格納され、その次は A(2)<=B(0)が比較され不成立の場合は B(0)が格納されます。対象数列 A、B の何れも昇順並びの場合は格納結果も昇順並びで格納されます。

フラグに-1 を記述した場合:

対象数列 A と B の大小関係が比較され大きい方を優先して格納します。初めに A(0)>=B(0)が比較され成立すると A(0)が格納され次に A(1)>=B(0)が比較され、成立すると A(1)が格納され、その次は A(2)<=B(0)が比較され不成立の場合は B(0)が格納されます。対象数列 A、B の何れも降順並びの場合は格納結果も降順並びで格納されます。

記述例:

$1 と$2 の要素の大小関係を比較し小さい方を優先して縫合する場合

```
assign $1 = 1,3,5,9,8
assign $2 = 0,2,4,10,7
$3 = MGR($1,$2,1)
```

$3 には、0,1,2,3,4,5,9,8,10,7 が格納されます。

記述例:

$4、$5,$6 の 3 個の数列を昇順並びで縫合する場合

```
assign $4 = 9,5,3
assign $5 = 0,4,6,2
assign $6 = 11,10,12,13
$3 = MGR(MRG(SRT(1,$4),SRT(1,$5),1),SRT(1,$6),1)
```

$3 には、0,2,3,4,5,6,9,10,11,12,13 が格納されます。

※ SRT 関数は 16.6 項 数列の並び替えを参照下さい。

記述例:

$1 と$2 の要素の大小関係を比較し大きい方を優先して縫合する場合

```
assign $1 = 1,3,5,9,8
assign $2 = 0,2,4,10,7
$3 = MGR($1,$2,-1)
```

$3 には、1,3,5,9,8,0,2,4,10,7 が格納されます。

記述例:

$4,$5,$6 の 3 個の数列を降順並びで縫合する場合

```
assign $4 = 9,5,3
assign $5 = 0,4,6,2
assign $6 = 11,10,12,13
$3 = MGR(MGR(SRT(-1,$4),SRT(-1,$5),-1),SRT(-1,$6),-1)
```

$3 には、13,12,11,10,9,6,5,4,3,2,0 が格納されます。

17. 4. 3. 2 つの数列の要素ごとに大小関係を比較し、大きい方または小さい方で数列を生成する

【RVS 関数】を使用します。

文法:

格納先 = RVS(フラグ,対象数列 A,対象数列 B)

引数:

【フラグ】＜必須＞

要素ごとの大小関係を記述します。記述するフラグは 2 または 3 とします。

フラグに 2 を記述した場合。要素ごとに対象数列 A(n)<=対象数列 B(n)が比較演算され、成立した場合に対象数列 A の要素が格納され、不成立の場合は対象数列 B の要素が格納されます。

フラグに 3 を記述した場合、要素ごとに対象数列 A(n)>=対象数列 B(n)が比較演算され、成立した場合に対象数列 A の要素が格納され、不成立の場合は対象数列 B の要素が格納されます。

【対象数列】＜必須＞

対象数列 A と対象数列 B を記述します。対象数列 A と対象数列 B の要素数は等しい必要があります。

記述例:

$2(n)と$3(n)を比較演算し、何れか小さい方を格納する場合

```
assign $2 = 1,3,6,4,8
assign $3 = 2,0,5,7,9
$1 = RVS(2,$2,$3)
```

$1 には、1,0,5,4,8 が格納されます。

※ $2(n)<=$3(n)が成立の時に$2(n)を、不成立の時に$3(n)を格納すると言う事は、$2(n)>$3(n)が不成立の時に$2(n) を成立の時に$3(n)を格納するという事と等価です。

記述例:

$2(n)と$3(n)を比較演算し、何れか大きい方を格納する場合

```
assign $2 = 1,3,6,4,8
assign $3 = 2,0,5,7,9
$1 = RVS(3,$2,$3)
```

$1 には、2,3,6,7,9 が格納されます。

17. 4. 4. 論理数列を指定し、論理数列の要素値 1 と 0 により数列の何れか要素で数列を生成する

【RVS 関数】を使用します。

文法:

格納先 = RVS(論理数列,対象数列 A,対象数列 B)

引数:

【論理値数列】＜必須＞

論理値数列を記述します。論理数列と対象数列 A 及び対象数列 B の要素数は等しい必要があります。 論理数列の要素ごとに、1 の時に対象数列 B の要素が格納され、0 の時に対象数列 B の要素が格納されます。

【対象数列】＜必須＞

対象数列 A と対象数列 B を記述します。

記述例:

$1(n)>=$2(n)の比較演算の結果、成立の時に$4(n)を、不成立の時に$3(n)を格納する場合

```
assign $1 = 0,3,5,9
assign $2 = 1,2,6,7
assign $3 = 0,1,2,3
assign $4 = 4,5,6,7
$5 = RVS(GTE($1,$2),$3,$4)
```

$5 には、$1>=$2 の結果が 0,1,0,1 となりますので 0,5,2,7 が格納されます。

17. 5. 数列並びの反転

17. 5. 1. 数列の並び順を反転する

【REV 関数】を使用します。

文法:

格納先 = REV(対象数列) 引数:

【対象数列】＜必須＞

対象数列を記述します。

記述例:

$2 の並び順を反転する場合
assign $2 = 0,1,2,3,4,5,6,7,8,9
$1 = REV($2)
$1 には、9,8,7,6,5,4,3,2,1,0 が格納されます。

17. 6. 数列の並び替え

17. 6. 1. 要素の並び替える順序を Index 数列で指定し、指定した Index 順に要素を並び替える

【QUE 関数】を使用します。

文法:

格納先 = QUE(並び替え Index 数列,対象数列)

引数:

【並び替え Index 数列】＜必須＞

並び替える順序を Index で指定する数列を記述します。並び替え Index 数列と対象数列の要素数は等しい必要があり、又並び替え Index 数列の要素値は唯一無二で、対象数列の全ての Index を含んでいる必要があります。

記述例:

$2 を$3 で指定した Index 並びに並び替える場合
assign $2 = 0,10,20,30,40,50,60,70,.80,90
assign $3 = 1,3,5,7,9,0,2,4,6,8
$1 = QUE($3,$2)
$1 には、10,30,50,70,90,0,20,40,60,80 が格納されます。

17. 6. 2. 数列の要素昇順或いは降順に並び替える

【SRT 関数】を使用します。

文法:

格納先 = SRT(並び替えコード,対象数列) 引数:

【並び替えコード】＜必須＞

並び替えコードを 1 とした場合は昇順、-1 とした場合は降順に並び替えます。なお、同じ値が存在した場合は出現順となります。

【対象数列】

演算対象数列を記述します。

記述例:

$2 の要素値並びを昇順に並び替える場合(フラグ=1)
assign $2 = 0,2,1,3,4,5,7,6,9,7
$1 = SRT(1,$2)
$1 には、0,1,2,3,4,5,6,7,8,9 が格納されます。

記述例:

$2 の要素値並びを降順に並び替える場合(フラグ=-1)
aasign $2 = 0,2,1,3,4,5,7,6,9,8
$1 = SRT(-1,$2)
$1 には、9,8,7,6,5,4,3,2,1,0 が格納されます。

17. 6. 3. 数列の要素昇順或いは降順に Index 並びを取得する

【SRT 関数】を使用します。17. 6. 2. と同じ SRT 関数を使用しますが、記述する並び替えコードが異なります。

文法:

格納先 = SRT(並び替えコード,対象数列) 引数:

【並び替えコード】＜必須＞

並び替えコードを 2 とした場合は昇順並びとした時の Index、-2 とした場合は降順並びとした時の Index を戻します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

記述例:

$3 の並びを$2 の昇順に並び替える場合
assign $2 = 0,2,1,3,4,5,7,6,9,8
assign $3 = 10,20,30,40,50,60,70,80,90
$1 = EQU(SRT(2,$2),$3)

$1 には、10,30,20,40,50,70,60,90,80

記述例:

$3 の降順に、$3、$4 を並び替える場合

```
assign $3 = 0,2,1,3,4,6,5
assign $4 = 1,2,3,4,5,6,7
$1 = SRT(-2,$3)
$2 = EQU($1,$3)
$5 = EQU($1,$4)
```

$1 には、$3 の降順 Index の 5,6,4,3,1,2,0 が格納されます。
$2 には、6,5,4,3,2,1,0 が格納され、$5 には、6,7,5,4,2,3,1 が格納されます。

17. 7. 数列の生成

生成する値が任意にバラバラの場合は Archi_1 Script 構文での代入文(assign 文)で行う必要があります。 但し、assign 文はそのまま、演算関数の引数として記述できません。

17. 7. 1. 数列の個数と値を指定し、数列を生成する

【DAG 関数】を使用します。 **文法:**

格納先 = DAG(生成する要素数,値)

引数:

【生成する要素数】＜必須＞ 生成する数列の長さ(要素数)を記述します。

【値】＜必須＞

要素値を記述します。要素値を数列で記述しても参照される Index は先頭のみが格納されます。

記述例:

要素数 10,000 点の要素値 0 の数列を生成する場合

```
$1 = DAG(10000,0)
```

$1 には、10000 点の要素値 0 が格納されます。

※ assign $1 = 10000<0>と記述したことと等価となります。

記述例:

要素数 1000 点の 0～999 数列を生成する場合

```
$1 = ACC(DAG(1000,1))-1
```

$1 には、0,1,2,3,…..999 が格納されます。

※ ACC( )は累算関数で、$\$1(n) = \sum_{i=0}^{n} \$1(i)$の演算を行います。

17. 7. 2. 数列の個数と振幅値、波形種別、周波数、位相角を指定し、波形数列を生成する

【DAG 関数】を使用します。17. 7. 1. と同じ DAG 関数を使用しますが、記述する引数が追加されます。

文法:

格納先 = DAG(生成する要素数,振幅値,波形種別コード,周波数、位相角) 引数:

【生成する要素数】＜必須＞ 生成する数列の長さ(要素数)を記述します。

【振幅値】＜必須＞

生成する波形の振幅値を記述します。

【波形種別コード】＜必須＞ 生成する波形種別コードを記述します。

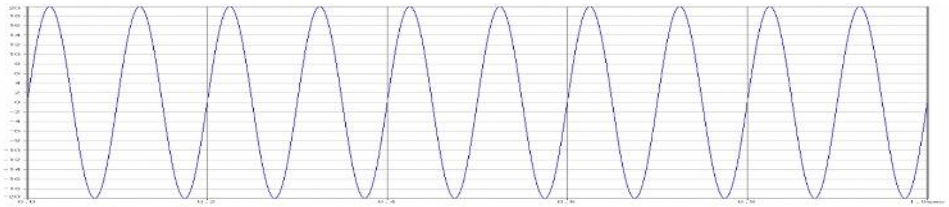
波形種別コード	生成される波形
0	正弦波
1	三角波
2	矩形波
3	鋸波

【周波数】＜必須＞

生成する波形の周波数を記述します。波形を生成する場合、事前に現在のサンプリング周波数を定義する必要があります。但し解析対象波形ファイル(PcWaveForm フォーマット)を読み込み済みの場合は既に定義されていますので 必要ありません。未定義の場合は def sampl_period 文(サンプリング周期定義文)を使用してサンプリング周期を定義します。生成される波形は定義されているサンプリング周波数(周期)によるサンプリング定理により、生成されますので、 生成する周波数がサンプリング周波数に近接した場合、波形ひずみを含みます。

【位相角】＜必須＞
位相角は初期の位相遅れを意味し、deg 単位で記述します。

記述例:
生成個数 10000、周波数 100Hz、振幅 20、位相角 0 の数列を生成する場合
```
def sampl_period 1e-4 “sec”
$1 = DAG(10000,20,0,10,0)
```
$1 に格納された、周波数 10Hz、振幅 20 の正弦波(10 波)をグラフで示します。



※ def sampl_period 文で定義したサンプリング周期は、記述している Script 文全体を唯一無二となります。従って、波形生成した後、PcWaveForm 形式の波形ファイルを読み出したり、或いは新たにサンプリング周期を定義したりした場合、直前までのサンプリング周期は自動的に更新される事に留意して下さい。

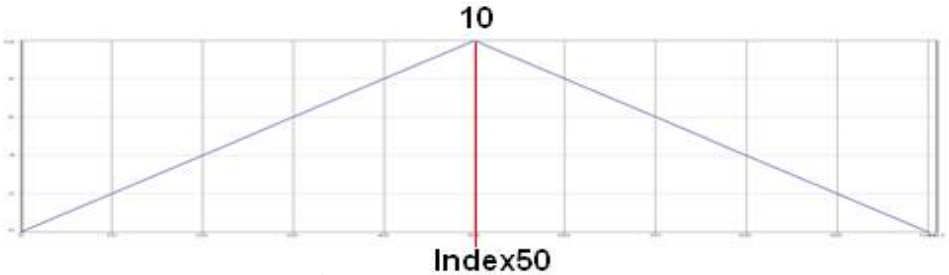
17. 7. 3. 数列の生成個数と内挿する Index 及び値を指定し、数列を生成する

【VRP 関数】を使用します。

文法:
格納先 = VRP(フラグ,生成数列の要素数,内挿する index 数列,内挿する値数列)

引数:
【フラグ】＜必須＞
フラグは、内挿する地点間を直線補間で接続する場合は 1 を記述し、直線補間しない場合は 0 を記述します。直線補間処理では、先端或いは終端に内挿していない場合は先端或いは終端に 0 が内挿されたものとして処理されます。なお、補間しない場合は内挿した地点以外は全て 0 となります。
【生成数列の要素数】＜必須＞ 生成する
数列の要素数を記述します。
【内挿する Index 数列】＜必須＞
生成する数列に値を内挿する位置を Index で指定します。記述する Index 値は生成する数列の範囲であり並び順は昇順の必要があります。
【内挿する値の数列】＜必須＞
内挿する Index 数列に対応した内挿する値数列を記述します。要素数は Index 数列と同じである必要があります。

記述例:
101 点の$1 数列の$1(50)に 10 を内挿し直線補間する場合(1 点のみ内挿)
```
$1 = VRP(1,101,50,10)
```
$1 に、格納された内容をグラフで示します。



※ フラグに 0 を記述した場合は、$1(50)に 10 が格納されそれ以外は全て 0 の数列が格納されます。

記述例:
100 点の数列に一定値 10 の数列を生成する場合(両端 2 点に内挿)
```
assign $2 = 0,99
assign $3 =10,10
$1 = VRP(1,100,$2,$3)
```
$1 には、一定値 10 とした個数 100 点の数列が格納されます。
※ 上記例は DAG(100,10)と記述したことと等価となります。また、VRP(1,100,0,99)と先端 1 点のみ記述すると、REV(ACC(DAG(100,1)-1))と記述したことと等価となります。

17. 7. 4. X/Y テーブルと X 値数列から対応する Y 値数列を生成する

【ITP 関数】を使用します。 **文法:**

格納先 = ITP(参照 X 値テーブル,参照 Y 値テーブル,参照 X 値数列)

引数:

【参照 X 値テーブル】＜必須＞

参照するテーブルの X 値数列を記述します。記述する数列は唯一無二の要素値で並びは参照 X 値テーブルと同じ昇順の必要があります。

【参照 Y 値テーブル】＜必須＞

参照するテーブルの Y 値数列を記述します。数列の要素数は参照 X 値テーブル数列と同じで、Y 値と Y 値は要素ごとに対応した値で構成されている必要があります。

【参照 X 値数列】＜必須＞

生成する Y 値数列が参照する X 値数列を記述します。記述する X 値数列は唯一無二の要素値で昇順並びの必要があります。ITP 関数は、直接対象数列から生成するのではなく、参照 X 値数列に対応した Y 値数列(戻り数列)を記述された X/Y テーブルを参照して、直線補間/補外演算により生成します。 本関数は主に定義した非線形テーブルから部分線形補間して参照する場合に使用しますが、この機能を使用して数 列の生成を行うこともできます。

記述例:

生成する数列の X 軸に-10～10 の 20 個の数列を指定し、符号逆転した数列を生成する場合

```
assign $1 = -1,0,1          /*  参照X 軸数列*/
assign $2 = 1,0,-1          /* 参照 Y 軸数列*/
$3 = ACC(DAG(21,1))-11     /* 生成する数列の X 軸数列*/
$4 = ITP($1,$2,$3)          /* Y 軸数列の生成*/
```

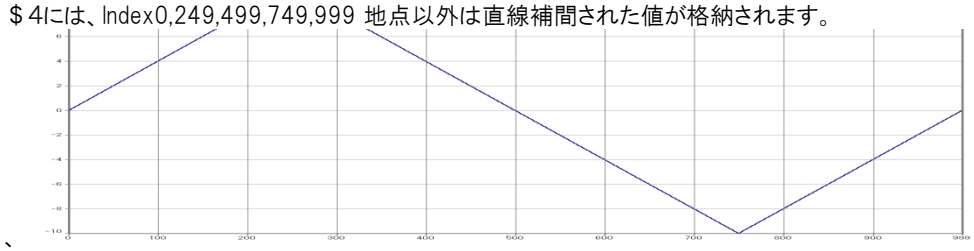
$3 は、-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10 が格納され、
$4 には、$3 に対応した 10,9,8,7,6,5,4,3,2.1,0,-1,-2,-3,-4,-5,-6,-7,-8,-9,-10 が格納されます。

記述例:

要素数 1000 個の数列に三角波 1 周期分を生成する場合

```
assign $1 = 0,249,499,749,999 /* 参照 X 軸数列*/
assign $2 = 0,10,0,-10,0           /* 参照 Y 軸数列*/
$3 = ACC(DAG(1000,1))-1         /*生成する数列の X 軸数列*/
$4 = ITP($1,$2,$3)                /* Y 軸数列の生成*/
```

＄4には、Index0,249,499,749,999 地点以外は直線補間された値が格納されます。



、

※ $1 は Index 数列、$2 は対応した要素数列、$4 は生成する Index 列となり、$4 の Index 数列に対応した要素を戻します。ITP 関数は VRP 関数と粗同様な機能となりますが、VRP 関数は生成する数列の個数を指定しますが、ITP 関数は生成する Index 列を指定する点が異なります。

17. 7. 4. カレントのサンプリング周期を参照してサンプリング周期歴数列を生成する

【SPB 関数】を使用します。

文法:

格納先 = SPB(初期値)

引数:

【初期値】＜必須＞

初期値は生成する Index0 の値を記述します。初期値の単位はサンプリングの単位で"sec"或いは"m"などで、プレトリガ収録された場合のオフセット値を記述します。

※ 本関数は内部で現在のサンプリング周期を参照し、解析対象の収録チャネル数列の個数分の初期値からサンプリング周期を増分とした数列を生成します。なお、サンプリング周期が定義されていない場合、或いは解析対象収録 チャネルが存在していない場合は実行時 Error となります。

記述例:

オフセット 2.5 秒で、1kHz サンプリングされた時刻歴数列を生成する場合
$1 = SPB(2.5)
$1 には、2.5,2.501,2.502,2.503….から収録チャネルの個数分格納されます。

記述例:
収録チャネルと同じ個数分の空数列を生成する場合
$1 = SPB(0)*0
$1 には、0 から収録チャネル数列の個数分の 0 が格納されます。
※ 上記は$1=#n*0 と同じ意味を持ちます。

17. 8. 数列の要素数の取得

17. 8. 1. 対象数列を指定して、数列の要素数を取得する

【LEN 関数】を使用します。
文法:
格納先 = LEN(対象数列)
引数:
【対象数列】＜必須＞ 演算対象
数列を記述します。
記述した対象数列が Null の場合は 0 が戻ります。

記述例:
$1 と$2 の何れか少ない要素数を取得する場合
assign $1 = 0,2,3,4,5,6,
$2 = ACC(DAG(20,1))
$3 = LEN($1+$2)
$3 には、6 が格納されます。
※ 但し、$1 及び$2 の何れも要素数は 1 個以上を前提としています。何れかが要素数が 1 個の場合は、多い方の要素数が戻ります。

18. 演算関数を使用して数列の Index を検索する

計測データ処理を記述する場合、地点を探す、区間を探す、或いは地点から或る地点前の時間演算など、演算処理上で収録波形から Index(データ番号)検索する操作が必要な場合が数多くあります。Archi_1 Script では、汎用言語と異なり、プログラムの煩雑さを避ける為に出来るだけデータを塊として扱うことを基本としています。従って、多量のデータを扱う場合にチャネルの個々の Index を指定して行う処理などはプログラムが複雑になる他に処理時間が掛ります。その為、計測波形から Index を検索する各種の演算関数が用意されています。

18. 1. 設定した閾値を過った地点 index を検索する

設定した閾値を通過した Index を検索します。

18. 1. 1. 通過方向、閾値、検索方向、検索開始 Index を指定して通過した最初の地点 index を検索する

【DTD 関数】を使用します。**文**

法:

格納先 = DTD(通過方向,閾値,検索方向,検索開始 index,対象数列)

引数:

【通過方向】＜必須＞

閾値通過方向を記述します。閾値を上昇で過る場合は 1、下降で過る場合は 0 と記述します。

【閾値】＜必須＞ 閾値

を記述します。

【検索方向】＜必須＞

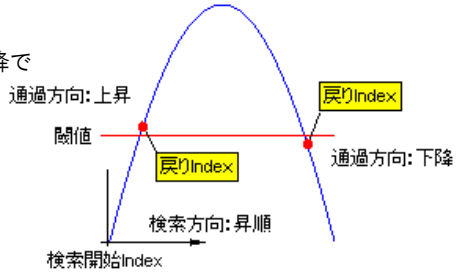
Index 昇順方向に検索する場合は1、降順方向の検索す る場合は 0 と記述します。

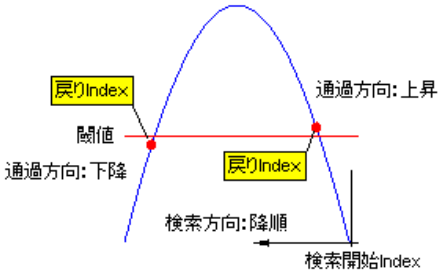
【検索開始 Index】＜必須＞

検索を開始する Index を記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。





戻り index は検索開始 Index から検索方向設定に従い、設定閾値を設定した方向で過った地点の index が戻り、閾値を過らない場合は-1 が戻ります。

記述例:

#1 の index1000 から index 昇順方向に閾値 3 を上昇で過る地点と下降で過る地点を検索する場合

```
$1 “上昇通過 Index:” = DTD(1,3,1,1000,#1)      /* 上昇通過、閾値 3、検索開始 Index1000*/
$2 “下降通過 Index:” = DTD(0,3,1,1000,#1)      /* 下降通過、閾値 3、検索開始 Index1000*/
```

18. 1. 2. 通過方向、閾値数列、検索方向数列、検索開始 Index 数列を指定して、通過した最初の地点 Index を検索する

【DTM 関数】を使用します。DTD 関数と同じ機能ですが、DTD 関数は一度に一つの組み合わせ条件しか検索できませんが DTM 関数は数列で記述された検索条件のそれぞれ要素ごとの組み合わせで成立した最初の地点 Index を数列で戻します。

文法:

格納先 = DTM(通過方向数列,閾値数列,検索方向数列,検索開始 index 数列,対象数列)

引数:

【通過方向】

閾値通過方向を記述します。閾値を上昇で過る場合は 1、下降で過る場合は 0 と記述します。

【閾値】＜必須＞ 閾値を数列

で記述します。

【検索方向】＜必須＞

Index 昇順方向に検索する場合は1、降順方向の検索する場合は 0 とした数列で記述します。

【検索開始 Index】＜必須＞

検索を開始する Index を数列で記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

引数意味は DTD 関数に同じですが、DTM 関数では通過方向以外の条件は複数要素を持つ数列で記述できます。 数列の各数列の Index 順に参照されて検索し、最初の閾値通過地点 Index を数列の要素数分検索します、 記述された閾値数列、検索方向数列、検索開始 Index 数列の要素数が同じでない場合は最も要素数の多い数列以外は index0 のみ参照されます。

戻り index の個数は検索条件数分の数列で戻ります。条件が成立しない要素は-1 が格納されます。 なお、条件は一つの組み合わせの場合は、DTD 関数と異なり、後述する 18.1.3 項の動作となります。

記述例:

```
#1の開始 Index0、順方向に閾値$1、通過方向$3 の条件で通過する最初の Index 位置を検索する場合
assign $1 “閾値:” = 1,2,3,4,1,2,3,40
assign $2 “通過方向:” = 1              /* 上昇通過設定*/
assign $3 “検索開始 Index:” = 8<0> /* 全て Index0から検索開始*/
assign $4 “検索方向:” = 8<1>          /* 全て index 昇順方向検索*/
$5 “検索結果 index:” = DTM($2,$1,$4,$3,#1)
```

$5 には設定した各数列$1～＄4の index 順の組み合わせで通過した index が格納されます、

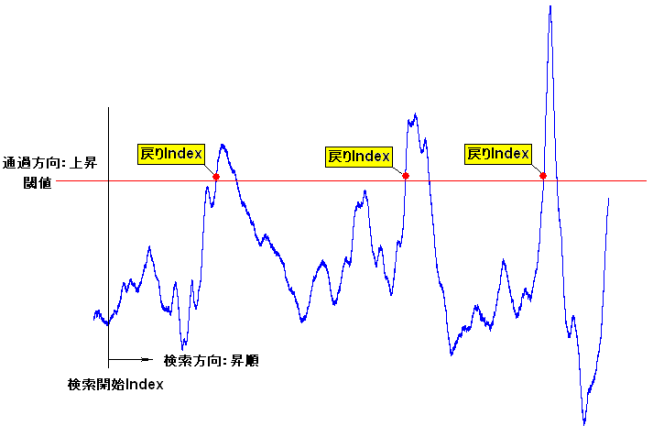
18. 1. 3. 閾値、通過方向、検索方向、検索開始 Index を指定して、通過した全ての地点 Index を検索する

【DTM 関数】を使用します。検索条件を全て単一要素で記述し、検索条件が単一の場合は、条件成立地点の全ての地点 Index を戻します。

文法:

格納先 = DTM(通過方向,閾値,検索方向,検索開始 index,対象数列) 引数:

【通過方向】＜必須＞ 閾値通過方向を記述します。
閾値を上昇で過 る場合は 1、下降で過る場合は 0 と記述します。
【閾値】＜必須＞ 閾値を記述します、
【検索方向】＜必須＞
Index 昇順方向に検索する場合は1、降順方向の検索する場合は 0 と記述します。
【検索開始 Index】＜必須＞ 検索を開始する Index を記述します。
【対象数列】＜必須＞
演算対象数列を記述します。



記述例:

```
#1 の index1000 から index 昇順方向に閾値 3 を上昇で過る地点と下降で過る地点を検索する場合
$1 = DTM(1,3,1,1000,#1)      /* 上昇通過、閾値 3、検索開始 Index1000*/
$2 = DTM(0,3,1,1000,#1)      /* 下降通過、閾値 3、検索開始 Index1000*/
```

$1 には、上昇で過った地点 Index が格納され、$2 には、下降で過った地点 index が格納されます。それぞれ過った個数を取得する場合は、LEN 関数を使用してLEN($1)、LEN($2)で求めます。

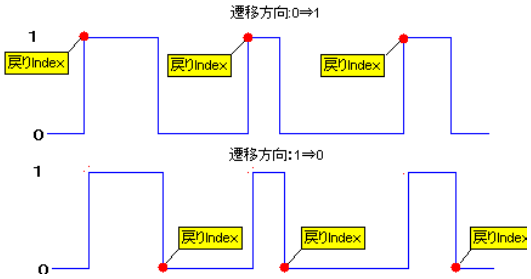
18. 1. 4. 対象波形数列を論理値数列に変換して指定し、論理遷移した全ての位置 Index を検索する

対象波形を論理値化する場合、比較関数【GT_関数、GTE 関数、LT_関数、LTE 関数】または論理値化関数【RWC 関数】を使用します。論理値から論理遷移地点 Index を求める場合は**【LST 関数】**を使用します。

文法:

格納先 = LST(論理遷移方向, 対象論理値数列) 引数:

【論理遷移方向】＜必須＞ 論理遷移方向を記述します。1を記述が
0 から 1 に遷移(上昇)した位置、0 を記述した場合は、論理が 1 から 0 に遷移(下降)した位置を検索します。
【対象論理値数列】＜必須＞ 演算対象
論理値数列を記述します。



戻り数列の要素数は検出した箇所数に従属します。検出されない場合は-1 が戻ります。
検索方向は Index 昇順に固定され、検索開始 Index は 0 に固定されています。検索方向を降順とする場合は REV 関数を使用して数列を反転しています。なお、数列を反転させて検索した場合戻り Index は逆になっていますので、先頭からの Index に換算する必要があります。又、開始 Index を設定する場合は、論理値数列から切り出しから LST 関数処理を行います。この場合も同じく Index を換算する必要があります。

記述例:

#1 の閾値 3.0 を上昇で通過する Index と下降で通過する Index 検索する場合
#1 の要素値が 3 以上を 1 にそれ以外を 0 とした論理値数列の立ち上がり地点、立下り地点を求めます。

```
$1 = LST(1,GT_(#1,3))                          /* 論理値変換、遷移 0→1 地点 Index を取得*/
$2 = LST(0,GT_(#1,3))                          /* 論理値変換、遷移 1→0 地点 Index を取得*/
$2 = RVS(GT_($1(0),$2(0)),$2,ERC(1,LEN($2)-1,$2))
$1 = RVS(GT_($1(LEN($1)-1),$2(LEN($2)-1)),$1,ERC(0,LEN($1)-2,$1))
```

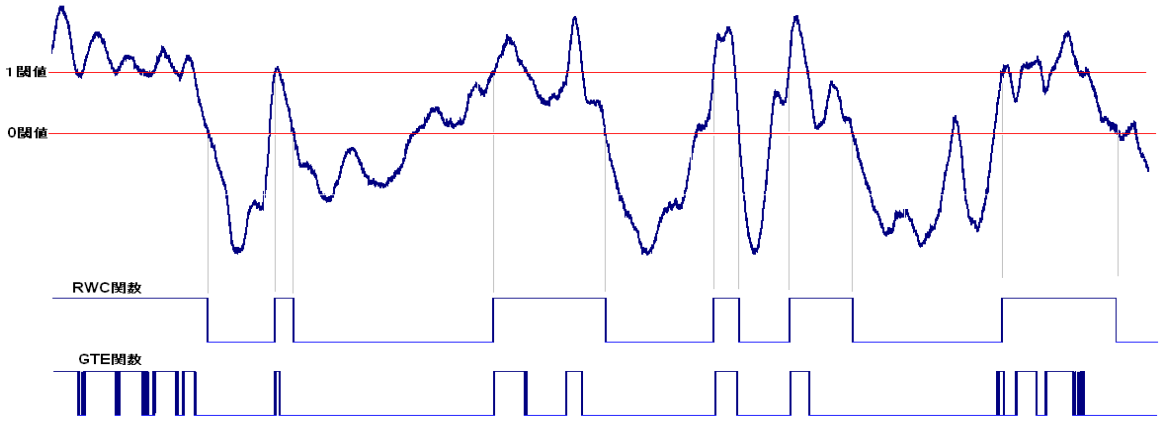
$1,$2 に遷移した Index 数列を修飾していることは、論理値数列に変換する際に使用する比較演算では遷移に関係なく設定した閾値から波形データが 1 または 0 に変換されますので、例えば波形の最初から指定した閾値を越えている場合や終端で超えたまま終わることが考えられます。その為、求めた遷移点 Index をチェックし、初めから閾値を越えている部分、及び終端部が閾値を越えている部分を削除し、解析範囲が閾値を上昇過り、下降で過るまでに修正する処理です。

記述例:

#1 の検索開始 Index1000 から閾値 3 を上昇で通過する Index と下降で通過する Index を検索する場合

```
$1 = LST(1,GT_(ERC(1000,LEN(#1)-1,#1),3))
$2 = LST(0,GT_( ERC(1000,LEN(#1)-1,#1),3))
$2 = RVS(GT_($1(0),$2(0)),$2,ERC(1,LEN($2)-1,$2))
$1 = RVS(GT_($1(LEN($1)-1),$2(LEN($2)-1)),$1,ERC(0,LEN($1)-2,$1))
```

論理値への変換に RWC 関数と GTE 関数を使用した場合の結果に相違がある事に留意下さい。



※ GTE 関数は上図の 1 閾値のみから論理値数列に変換します。

18. 1. 5. 閾値を通過後、閾値通過が指定個数持続する、全ての閾値通過位置 Index を検索する

【DTE 関数】を使用します。

文法:

格納先 = DTE(通過方向,閾値,持続個数,対象数列)

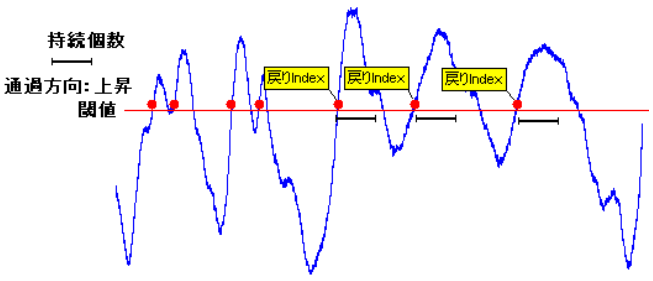
引数:

【閾値通過方向】＜必須＞
閾値通過方向を記述します。閾値を上昇で過る場合は 1、下降で過る場合は 0 と記述します。

【閾値】＜必須＞ 閾値
を記述します、

【持続個数】＜必須＞ 閾値通過後、閾値以上を保持するデータ個
数を記述します。

【対象数列】＜必須＞ 演算対象
数列を記述します。



検索方向は index 昇順に、検索開始 Index は先頭に固定されています。戻り数列の要素数は成立した箇所数に従属します。検出されない場合は-1 が戻ります。

記述例:

#1 の閾値 1.0 を上昇で通過し、100 点以上通過後閾値以上が持続する閾値通過 Index を検索する場合

```
$1 = DTE(1,1,100,#1)
```

記述例:

#1 の終端からから Index 降順に閾値 1.0 を上昇で通過し 100 点以上閾値以上が持続する閾値通過 Index を検索する場合

```
$1 = DTE(1,1,100,REV(ERC(0,2000,#1)))   /* 通過方向上昇、閾値1、通過後持続個数 100*/
```

```
$1 = REV(2000-$1)     /* 2000 は切り出し最終 index、Index を#1 のindexに修正*/
```

記述例:

$1 に上昇通過 Index、$2 に下降通過 Index が格納されているとして論理波形を生成する場合

```
$3 = LNK($1,$2)                                  /*上昇通過 Index と下降通過 Index を連結*/
$4 = LNK(DAG(LEN($1),1),DAG(LEN($1),-1)) /*上昇通過を 1、下降通過を-1 として連結*/
$5 = SRT(2,$3)                                   /*Index 並び順 Keyを取得*/
$6 = QUE($5,$3)                                  /*通過 Index を key 順に整列*/
$7 = QUE($5,$4)                                  /*値を key 順に整列*/
$4 = ACC(VRP(0,LEN(#1),$6,$7))                   /*復元論理数列の生成*/
```

$4 に論理数列が格納されます。但し、復元波形の要素数は Index 検索対象数列#1 としています。

18. 1. 6. 検索開始 Index、通過方向、閾値、通過後持続時間を設定して全ての条件成立位置 Index を検索する

【DTF 関数】を使用します。DTE 関数との違いは検索開始 Index を指定可能とした点と、持続個数ではなく時間とした点で、持続時間と持続個数の関係は持続時間は、DTE 関数の持続個数の引数に INT(持続時間/PRD())と記述することと等価となります。最も大きく異なる点は、DTF 関数の持続時間は正で記述した場合は閾値を過ってからの持続時間を意味し、負数で記述した場合は閾値通過前の持続時間を指定できる点にあります。

文法:

格納先 = DTF(検索開始 Index,閾値通過方向,閾値,持続時間,対象数列) 引数:

【検索開始 Index】＜必須＞ 検索を開始する Index を記述します。

【閾値通過方向】＜必須＞

閾値通過方向を記述します。閾値を上昇で過る場合は 1、下降で過る場合は 0 と記述します。

【閾値】＜必須＞ 閾値を記述します、

【持続時間】＜必須＞

閾値通過後或いは通過前に閾値以上を保持する時間(単位 sec)で記述します。記述する時間は正数の場合は通過後を意味し、負数の場合は通過前を意味します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

記述例:

#1 の始点から Index 昇順に閾値 1.0 を上昇で通過し、通過前 1 秒間閾値以下の値が持続する閾値通過 Index を検索する場合

```
$1 = DTF(0,1,1,-1,#1)        /* 検索開始 Index0、通過方向上昇、閾値1、持続時間通過前 1 秒*/
```

18. 1. 7. 閾値を通過後、閾値が指定個数、許容変動範囲で持続する、全ての閾値通過位置 Index を検索する

【CTD 関数】を使用します。 文法:

格納先 = CTD(閾値通過方向,閾値,検索方向,検索開始 Index 数列,持続個数,変動許容値,対象数列)

引数:

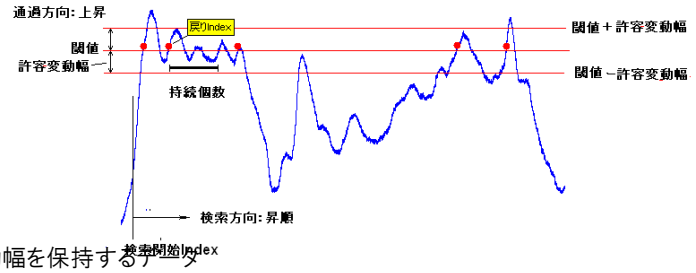
【閾値通過方向】＜必須＞

閾値通過方向を記述します。閾値を上昇で過る場合は 1、下降で過る場合は 0 と記述します。

【閾値】＜必須＞ 閾値を記述します、

【検索方向】＜必須＞

Index 昇順方向に検索する場合は1、降順方向の検索する場合は 0 と記述します。



【検索開始 Index】＜必須＞ 検索を開始する Index を記述します。

【持続個数】＜必須＞ 閾値通過後、許容変動幅を保持するデータ個数を記述します。

【変動許容幅】＜必須＞ 閾値通過後の許容変動幅を記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

戻り数列の要素数は成立した地点個数に従属します。成立地点が存在しない場合-1 が戻ります。

記述例:

#1 の検索開始 Index0 から閾値 2 を通過後、許容変動値±0.5 以内をデータ個数 200 個以上持続した Index を検索する場合

$1 = CTD(1,2,1,0,200,0.5,#1) /* 通過方向上昇、閾値 2、検索方向昇順、検索開始 Index0、持続個数 200、許容変動幅 0.5*/

18. 1. 8. 2つの対象数列から交互にそれぞれの閾値を過る位置 Index を検索する

【DTC 関数】を使用します。DTC 関数は相互に関連した 2 信号を検索対象数列として、開始検索数列から開始閾値通過 Index を検索し、終了検索数列から終了閾値通過 Index を検索します。

文法:

格納先 = DTC(開始 Index1,閾値 1,閾値 2,開始検索数列,遅延,閾値 3,終了方向,終了検索数列) 引数:

【開始 Index1】＜必須＞ 検索開始
Index を記述します。

【閾値 1】＜必須＞
初回検索開始閾値を記述します。但し、正確には開始 Index で記述された Index 位置の値にここで記述された閾値 1 が加算された値が初回開始閾値となります。なお、閾値通過方向はここで記述された閾値1の符号で決定され、正数の場合は上昇通過、負数の場合は格納通過を意味します。

【閾値2】＜必須＞
次回開始閾値閾値変化量を記述します。次回の開始閾値は終了点 Index 位置の開始検索対象数列の値に閾値 2 を加算した値となります。閾値通過方向は閾値 2 の符号で決定され、正数の場合は上昇通過、負数の場合は下降通過を意味します。

【開始検索数列】＜必須＞
開始点検索対象数列を記述します。

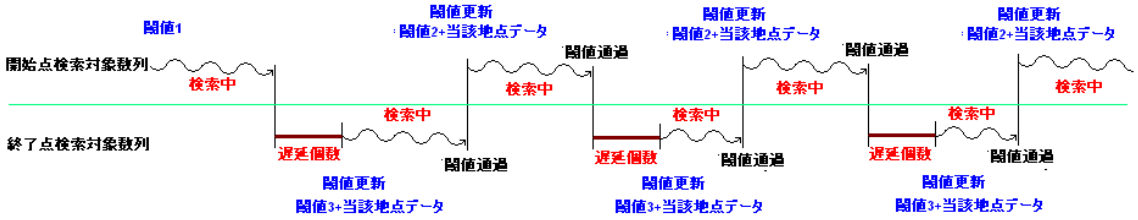
【遅延】＜必須＞ 開始閾値通過後の終了点閾値検索開始迄の遅延(不感)個数
を記述します。

【閾値 3】＜必須＞
終了点閾値変化量を記述します。終了点閾値は、ここで記述した閾値 3 に検索開始 Index の値を加算した値となります。又終了点閾値通過方向は、終了点閾値変化量の符号で決定され、閾値変化量が。正数の場合は上昇通過、負数の場合は下降通過を意味します。

【終了方向】＜必須＞
終了点検索方向を記述します。終了点検索開始方向を意味し、Index 昇順方向は 0、Index 降順方向は 1 と記述します。

【終了検索数列】＜必須＞ 終了点検索
対象数列を記述します。

検索動作を下図に示します。



最初に開始点検索数列から閾値 1 を設定方向に通過する初回開始 Index を検索します。開始点閾値通過を検出する設定されている遅延データ個数分 Index をずらし、その地点の終了検索数列の値に閾値 3 を加算した値を終了点閾値として終了点閾値通過 Index を検索します。終了点閾値通過を検出すると、その地点の開始検索数列の値に閾値 2 を加算した値を開始閾値として開始点閾値通過を検索します。開始点閾値通過を検出すると、遅延データ個数分ずらしその地点の終了検索数列の値に閾値 3 を加算した値を終了点閾値として検索します。検出するとその地点の終了点開始検索数列の値に閾値 2 を加算した値を開始閾値として検索します。その動作を繰り返します。つまり、初回の開始閾値を除き、閾値はごと回更新される事になります。
戻り数列は開始 Index,終了 Index,開始 Index,終了 Index…と格納され、その個数は検出回数に従属します。なお、検出できない場合は-1 が戻ります。

18. 2. 波形の山(Peak)谷(Valley)地点 Index を検索する

山谷位置を検索します。

18. 2. 1. 検索方向、検索開始 Index 数列を指定して検索開始位置からの最初の山または谷位置 Index を検索する

【DTP 関数】を使用します。

文法:

格納先 = DTP(検索フラグ,検索開始 Index 数列,対象数列)

引数:

【検索フラグ】＜必須＞

検索フラグは検索方向及び検索対象を指定するフラグで 0～3 を記述します。

0 の時⇒Index 降順、谷位置 Index 検索

1の時⇒Index 降順、山位置 Index 検索

2 の時⇒Index 昇順、谷位置 Index 検索

3 の時⇒Index 昇順、山位置 Index 検索

【検索開始 Index】＜必須＞

検索を開始する Index を記述します。数列で記述した場合、要素数分検索します。なお検索方向は昇順に固定されています。

【対象数列】＜必須＞ 検索対象

数列を記述します。

戻り数列は検索開始 Index 数列の要素数に等しく。指定した山或いは谷が存在しない場合は-1 が戻ります。

記述例:

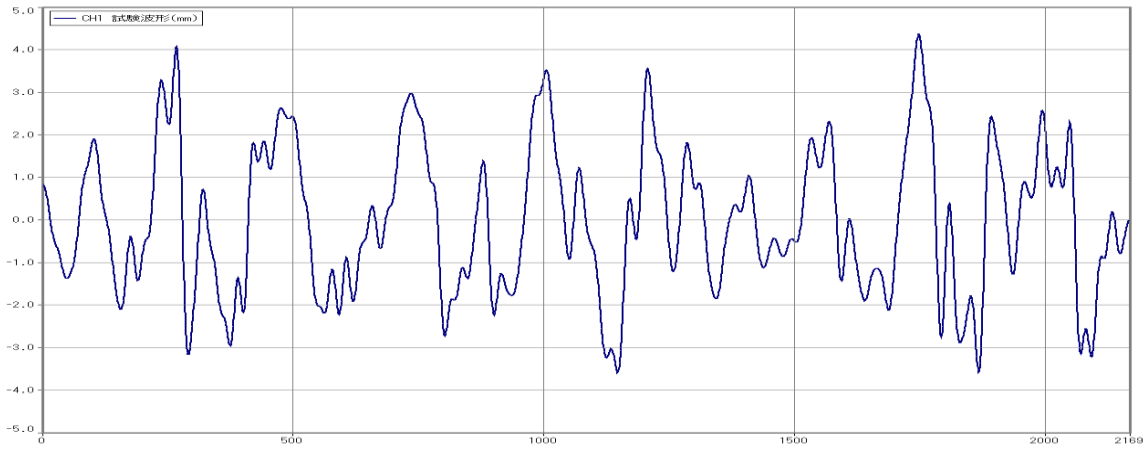
#1 の index1000 から Index 昇順に最初の山位置 Index を検索しその値を抽出する場合

```
$1 = DTP(3,1000,#1)
$2 = PTV($1,#1)
```

$1 には、山位置が格納され、$2 にはその値が格納されます。

検索開始index	$1(山index)	山値
1000	1006	3.522

記述例で使用した波形を示します。



記述例:

#1 の Index0,500,1000,1500 から Index 昇順に最初の山、谷位置 Index を検索する場合

```
assign $5 = 0,500,1000,1500
$1 = DTP(3,$5,#1)      /* 山 index */
$2 = DTP(2,$5,#1)      /* 谷 index */
$3 = PTV($1,#1)        /*  山値 */
$4 = PTV($3,#1)        /* 谷値.*/
```

$1 には、指定した開始 Index から最初の山位置 index、$2 には、最初の谷位置 Index が格納され、$3 には$1 に対応した山値、$4 には$2 対応した谷値が格納されます。

検索開始index	$1(山index)	山値	$2(谷index)	谷値
0	103	1.902	49	-1.379
500	579	-1.163	563	-2.194
1000	1006	3.522	1051	-0.927
1500	1535	1.930	1551	1.220

18. 2. 2. 検索方向、検索対象山谷種別を指定し全ての山または谷位置 Index を検索する

【PVF 関数】を使用します。PVF 関数は、DTP 関数と異なり、数列上の全ての山、谷位置 Indx をの検索以外に山値、谷値を

取得できます。又、山、谷検索に際して無効振幅を設定できる点などが異なります。但し検索方向は昇順、検索開始 Index は 0(先頭から)に固定されています。

文法:

格納先 = PVF(検索フラグ,無効振幅値,対象数列) 引

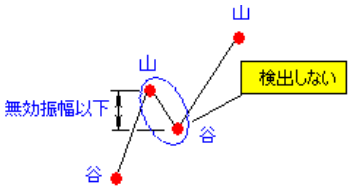
数:

【検索フラグ】＜必須＞

※ 検索フラグは検索方向、検索対象指定するフラグで-1～5 を記述します。

0 の時⇒山位置 Index、谷位置 Index 両方の同時検索
1 の時⇒山位置、谷位置両方の値の同時検索
2 の時⇒山位置 Index の検索
3 の時⇒山位置の値の検索
4 の時⇒谷位置index の検索
5 の時⇒谷位置の値の検索
-1 の時⇒山位置, 谷位置の値を直線補間した波形に変換
※ 検索フラグ-1 は山谷で構成された波形数列に変換します。

【無効振幅】＜必須＞ 無効振幅値を記述します。無効振幅は山或いは谷の抽出に際して指定 した振幅以上ないと山/谷として抽出しない値を記述します。



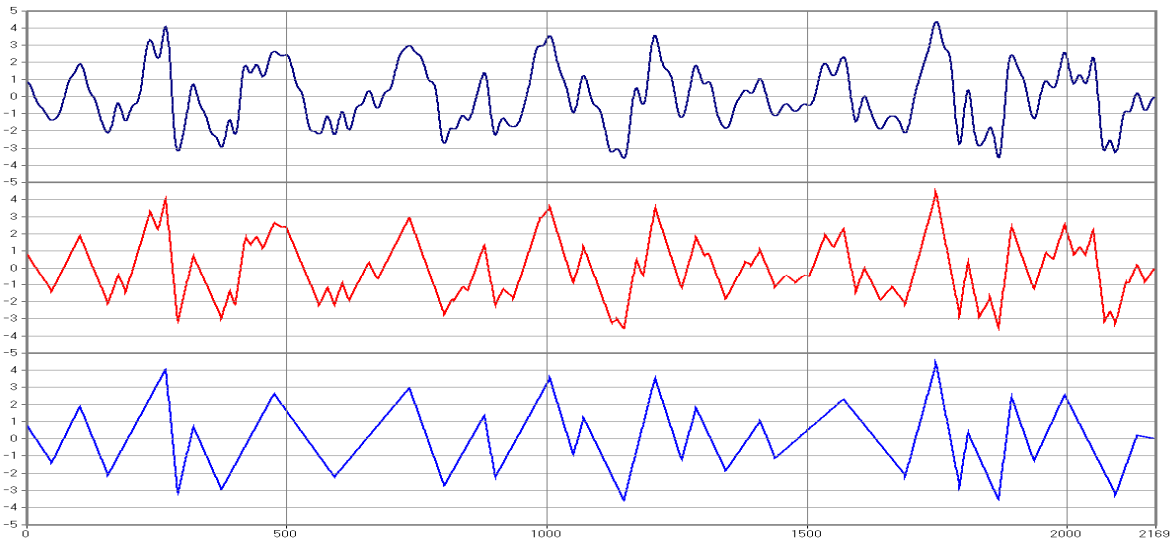
【対象数列】＜必須＞ 検索
対象数列を記述します。

記述例:

#1 の波形から無効振幅を 0 とした場合と 2 とした場合の山谷で構成された波形に変換する場合

```
$1 = PVF(-1,0,#1)        /* 無効振幅0、山谷直線補間波形*/
$2 = PVF(-1,2,#1)        /* 無効振幅2、山谷補間波形*/
```

$1 には、無効振幅 0 とした山谷で構成された波形数列が格納されます。
$2 には、無効振幅 2 とした山谷で構成された波形数列が格納されます。



※ 上段は#1 の波形、中段は無効振幅 0 とした波形、下段は無効振幅 2 とした波形を表示

記述例:

#1 の波形から無効振幅 2 として山位置 Index と谷 Index を検索してその値を抽出する場合

```
$1 = PVF(2,2,#1)         /* 無効振幅 2、山位置 Index 検索*/
$2 = PVF(4,2,#1)         /* 無効振幅 2、谷位置 Index 検索*/
$3 = PVF(3,2,#1)         /* 無効振幅 2、山値検索*/
$4 = PVF(5,2,#1)         /* 無効振幅 2、谷値検索*/
```

$1 には、山位置 Index、$2 には谷位置 Index が格納され、$3 に山 Index に対応した山値、$4 に谷 Index に対応した谷値が格納されます。

$1(山index)	山値	$2(谷index)	谷値
103	1.902	49	-1.379
268	4.078	157	-2.104
321	0.721	291	-3.159
476	2.630	375	-2.951
736	2.971	592	-2.222
880	1.373	803	-2.722
1006	3.522	901	-2.231
1071	1.219	1051	-0.927
1208	3.558	1148	-3.572
1287	1.807	1259	-1.212
1410	1.033	1344	-1.851
1570	2.313	1439	-1.120
1748	4.359	1689	-2.125
1809	0.385	1793	-2.751
1893	2.428	1868	-3.561
1995	2.569	1936	-1.277
2134	0.190	2093	-3.220

記述例:

#1 の波形から無効振幅 2 として山位置 Index 間隔を求める場合

```
$1 = PVF(2,2,#1)                /* 無効振幅 2、山位置 Index 検索*/
$1 = ERC(0,INT(LEN($1))-1,$1)   /* 山位置の数を偶数に整合*/
$2 = ERC(0,LEN($1)-2,$1))       /* 山位置の 0 から最後-1 を抽出*/
$3 = ERC(1,LEN($1)-1,$1)        /* 山位置の 1 から最後を抽出*/
$4 = ($3-$2)*PRD()              /* 山位置間隔の演算*/
```

$4 に山と山の間隔(sec)が格納されます。但し、上記例では山位置 Index が 2 個以上存在していることが条件です。

18. 3. 波形の最大値地点 Index、最小値地点 Index を検索する 最大値位置または最小値位置を検索します。

18. 3. 1. 検索範囲数列を指定して検索範囲の最大値位置 Index を検索する

【MXP 関数】を使用します。

文法:

格納先 = MXP(検索範囲数列,対象数列)

引数:

【検索範囲数列】<省略可> 検索範囲を記述します。検索範囲数列は即値或いは対象数列の要素数より少ない数列で記述した場合と対象数列 と同じ要素数で記述した場合で動作が異なります。

即値或いは要素数 1 個で記述した場合:

検索開始Indexと見なし、記述したIndexからIndex 昇順方向に終端までの範囲の最大値位置Indexを検索します。

対象数列より少ない個数で記述した場合:

記述順に検索区間と見なし当該区間内での最大値位置 Index を検索します。但し検索方向が Index 昇順に固定されている為、記述する Index は昇順並びの必要があります。例えば、0,100,500,1000 と記述すると、0～99、100～499、500～999 の 3 区間の最大値位置 Index を検索します。

対象数列の要素数と等しい場合:

論理値数列で記述されていると見なし、論理 1 の区間の最大値位置 Index を検索します。なお、検索範囲数列の記述が省略された場合は、対象数列全体を範囲として最大値位置 Index を検索します。

【対象数列】<必須> 検索対象

数列を記述します。

記述例:

#1 全体の最大値位置 Index 検索とその値を抽出する場合

```
$1 = MXP(#1)            /* 全体の最大値位置を検索*/
$2 = PTV($1,#1)         /* 検索した Index 位置の値を取得*/
```

$1 には、最大値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最大値Index)	$2(最大値)
0	2168	1748	4.359

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例:

#1 の Index1000 からの最大値位置 Index 検索とその値を抽出する場合

```
$1 = MXP(1000,#1)       /* Index1000 から終端までの最大値位置を検索*/
$2 = PTV($1,#1)         /* 検索した Index 位置の値を取得*/
```

$1 には、最大値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最大値Index)	$2(最大値)
1000	2168	1748	4.359

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例:

#1 の Index0,500,1000,1500 間の最大値位置 Index の検索とその値を抽出する場合

```
assign $3 = 0,500,1000,1500        /* 最大値位置検索数列生成*/
$1 = MXP($3,#1)                    /* 指定位置から指定位置間の最大値位置検索*/
 $2 = PTV($1,#1)                   /* 検索した Index 位置の値を取得*/
```

$1 には、最大値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最大値Index)	$2(最大値)
0	499	268	4.078
500	999	999	3.250
1000	1499	1208	3.558

※ #1 は 18.2.1 記述例で示した波形を使用しています。

※ 閾値通過から再び閾値通過するまでの最大値位置と最大値を求める場合は、上記 assign 文を DTE 関数を使用した閾値通過 Index を取得に変更する事で良い。

記述例:

#1 の閾値 2 を上昇で過ってから下降で過る区間ごとの最大値位置 Index とその値を抽出する場合

```
$1 = MXP(GT_(#1,2),#1)
$2 = PTV($1,#1)
```

$1 には、区間ごとの最大値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最大値Index)	$2(最大値)
228	277	268	4.078
466	509	476	2.630
714	763	736	2.971
975	1021	1006	3.522
1198	1222	1208	3.558
1564	1576	1570	2.313
1726	1777	1748	4.359
1888	1902	1893	2.428
1989	2002	1995	2.569
2047	2053	2050	2.263

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例:

$1 の閾値 5.2 上昇通過から 4 秒後迄間の最大値位置と最大値を求める場合

```
$2 = DTM(1,5.2,1,0,$1)       /* 上昇通過、閾値 5.2、昇順方向、検索開始 index0*/
$3 = LEN($2)                 /* 検索 Index 個数演算*/
$4 = MRG($2,$2+INT(4/PRD())) /* 検索結果 Index と 4 秒後 Index をマージ*/
assign $3 = $3<1,-1>         /* 立ち上がり点 1、立下り点-1 として数列生成*/
$5 = ACC(VRP(0,LEN($1),$4,$3))    /*復元論理数列の生成*/
$6 = MXP($5,$1)              /* 区間最大値位置 Index 検索*/
$7 = PTV($6,$1)              /* 区間最大値*/
$8 = ($7-$2)*PRD()           /* 閾値通過から最大値までの時間演算*/
```

$6 に区間ごとの最大値位置 Index、$7 に区間ごとの最大値、$8 に区間ごとに開始から最大値まで時間が格納されます。但し、閾値通過から閾値通過まで 4 秒以上離れていることを前提としています。

18. 3. 2. 検索範囲数列を指定して検索範囲の最小値位置 Index を検索する

【MNP 関数】を使用します。MXP 関数と同じ記述操作法で、最小値位置 Index を検索する事が異なります。

文法:

格納先 = MNP(検索範囲数列,対象数列)

引数:

【検索範囲数列】＜省略可＞ 検索範囲を記述します。検索範囲数列は即値或いは対象数列の要素数より少ない数列で記述した場合と対象数列 と同じ要素数で記述した場合で動作が異なります。

即値或いは要素数 1 個で記述した場合:

検索開始Indexと見なし、記述したIndexからIndex昇順方向に終端までの範囲の最小値位置Indexを検索します。

対象数列より少ない個数で記述した場合:

記述順に検索区間と見なし当該区間内での最小値位置 Index を検索します。但し検索方向が Index 昇順に固定されている為、記述する Index は昇順並びの必要があります。例えば、0,100,500,1000 と記述すると、0～99、100

～499、500～999 の 3 区間の最小値位置 Index を検索します。
対象数列の要素数と等しい場合：
論理値数列で記述されていると見なし、論理 1 の区間の最小値位置 Index を検索します。なお、検索範囲数列の記述が省略された場合は、対象数列全体を範囲として最小値位置 Index を検索します。
【対象数列】＜必須＞ 検索対象
数列を記述します。

記述例：
#1 の最小値位置 Index 検索とその値を抽出する場合
$1 = MNP(#1)
$2 = PTV($1,#1)
$1 には、最小値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最小値Index)	$2(最小値)
0	2168	1148	-3.572

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例：
#1 の Index1000 からの最小値位置 Index 検索とその値を抽出する場合
$1 = MNP(1000,#1)
$2 = PTV($1,#1)
$1 には、最小値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最小値Index)	$2(最小値)
1000	2168	1148	-3.572

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例：
#1 の Index0,500,1000,1500 間の最大値位置 Index の検索とその値を抽出する場合
assign $3 = 0,500,1000,1500
$1 = MNP($3,#1)
$2 = PTV($1,#1)
$1 には、区間ごとの最小値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最小値Index)	$2(最小値)
0	499	291	-3.159
500	999	803	-2.722
1000	1499	1148	-3.572

※ #1 は 18.2.1 記述例で示した波形を使用しています。

記述例：
#1 の閾値 2 を下降で過ってから上昇で過る区間ごとの最小値値位置 Index とその値を抽出する場合
$1 = MNP(LT_(#1,2),#1)
$2 = PTV($1,#1)
$1 には、区間ごとの最小値位置 Index が格納され、$2 には、その値が格納されます。

開始index	終了index	$1(最小値Index)	$2(最小値)
277	228	157	-2.104
509	466	291	-3.159
763	714	592	-2.222
1021	975	803	-2.722
1222	1198	1148	-3.572
1576	1564	1344	-1.851
1777	1726	1689	-2.125
1902	1888	1868	-3.561
2002	1989	1936	-1.277
2053	2047	2035	0.765

※ #1 は 18.2.1 記述例で示した波形を使用しています。

18. 3. 3. 検索閾値と有効閾値を指定して検索範囲の最大値/最小値位置 Index を検索する
【PMP 関数】を使用します。
文法：
格納先 = PMP(最大最小フラグ,検索閾値数列,有効閾値数列,対象数列) 引数：
【最大最小フラグ】＜必須＞

最大最小フラグは、最大値位置 Index を検索するか、最小値位置 index を検索するかのフラグです。
最小値位置 Index を検索する場合は 0、最大値位置を検索する場合は 1 を記述します。

【検索閾値数列】＜必須＞ 検索閾値数列と後述する有効閾値数列は、検索区間を決定する引数です。
検索閾値<=有効閾値の時：区間は検索閾値を上昇で過ってから下降で過るまでとなります。
最小/最大フラグが 1 の場合：
有効閾値を上回った中での最大値位置Indexを検索します。
最小/最大フラグが 0 の場合：
有効閾値を下回った最小値位置 Index を検索します。 検索閾値>有効閾値の時：
検索閾値を下降で過ってから上昇で過るまでとなります。
最小/最大フラグが 1 の場合： 有効閾値を上回った最大値位置 Index を検索します。
最小/最大フラグが 0 の場合： 有効閾値を下回った最小値位置を検索します。

【検索閾値数列】＜必須＞ 検索閾値数列と有効閾値数列の要素数が同じ場合：
検索対象数列の先頭から検索閾値数列と有効閾値す列の要素ごとの組み合わせで検索区間が決定され要素数分検索されます。 検索閾値数列と有効閾値数列の要素数が異なった場合：
何れも参照される Index は 0 のみとして固定され、先頭から成立した区間ごとに検索します。なお、何れも検索区間が存在しない場合は-1 が戻ります。

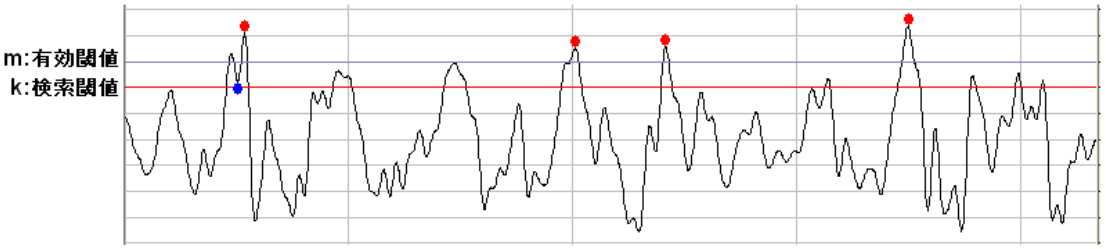
【対象数列】＜必須＞ 検索対象数列を記述します。

記述例：

#1 から検索閾値 2、有効閾値 3 を指定して、全ての範囲から最小値位置 Index と最大値位置 Index を検索し対応する値を抽出する場合

```
$1 = PMP(1,2,3,#1)
$2 = PMP(0,2,3,#1)
$3 = PTV($1,#1)
$4 = PTV($1,#1)
```

$1 には検索範囲で有効閾値を上回った最大値 Index が、$2 には有効閾値を下回った最小値 Index が格納され、$3 には $1 に対応した値が$4 に$2 に対応した値が格納されます。



赤丸は最大値位置Index　青丸は最小値位置Index

※ #1 は 18.2.1 記述例で示した波形を使用しています。

18. 4. 比較演算関数を使用して index を検索する

比較演算関数は、引数フラグに 0 を記述した場合或いは記述省略した場合、論理値数列を戻し、引数フラグに 1 を記述した場合、成立した Index を格納する機能を持ちます。比較演算関数には、【GT_関数：X > Y】、【GTE 関数：X >= Y】、【LT_関数 X < Y】、【LTE 関数：X <= Y】、【EQU 関数：X = Y】、【NEQ 関数：X <> Y】の 6 種類あります。引数はすべて同じとなりますので、GTE 関数を例として説明します。

文法：

格納先 = GTE(対象数列,比較数列,フラグ) 引数：

【対象数列】＜必須＞
比較される対象数列を記述します。X>=Y と記述した場合の X に相当します。

【比較数列】＜必須＞
比較数列を記述します。X>=Y と記述した場合の Y に相当します。

【フラグ】＜省略可＞
戻り数列が論理値は成立した地点の Index かを設定するフラグを意味します。0 または記述省略した場合は論理値が戻り、1 と記述した場合は、成立した地点の Index が戻ります。

記述例:

ch1が 1 以上で ch2 が 5 以下の状態が 5 秒以上持続した区間の開始終了 Index を求める場合

```
$1 = AND(GTE(#1,1),LTE(#2,5))   /* (ch1>=1) & (ch2<=5) */
$2 = EQU(DIF($1),1,1)                  /* 区間開始 index 取得*/
$3 = EQU(DIF($1),-1,1)                /* 区間終了 Index 取得*/
$4 = GTE(($3-$2)*PRD(),5)           /* 区間長 5 秒以上検査*/
$2 = ZSP($4,$2)                         /*  区間開始index*/
$3 = ZSP($4,$3)                         /*  区間終了index*/
```

$2 には、ch1 が 1 以上で ch2 が 5 以下の区間が 5 秒以上存在した時の区間開始 Index が、$3 には、区間終了 Index が格納されます。上記例は、設定した区間が存在していることを前提としています。

18. 5. 検索窓を設定し、波形上をスライディングさせて Index を検索する

18. 5. 1. 一定区間以上が検索窓以内の変動であった場合の開始 Index と終了 Index を求める

【SWP 関数】を使用します。 **文**

法:

格納先 = SWP(検索窓横幅,検索窓縦幅、検索窓動作閾値,対象数列)

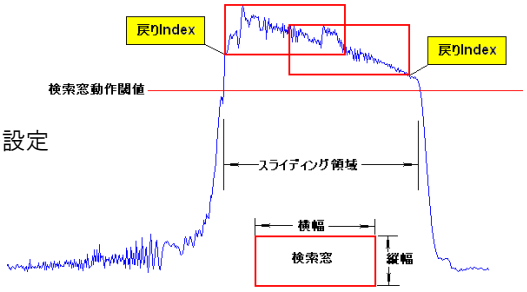
引数:

【検索窓横幅】＜必須＞
検索窓の横幅をデータ個数で記述します。

【検索窓縦幅】＜必須＞ 検索窓
の縦幅を記述します。

【検索幅動作閾値】＜必須＞ 検索窓をスライディングする領域を設定します。正数で記
述した場合は、ここで記述した閾値以上の範囲をスライディングし、負数の場合は閾値以下の範囲をスライディングします。

【対象数列】＜必須＞ 検索対象
数列を記述します。



戻り数列は、検索窓内の最大値−最小値が検索窓内に入っている時に成立し、成立した最初の Index と不成立になった検索窓最終 Index を戻します。検索窓に入った Index と検索窓から出た Index を交互並んだ数列となります。成立箇所が存在しない場合−1 が戻ります。検索窓は一定区間の変動幅を意味し、閾値以上或いは以下領域で変動幅に入っている区間を開始 Index と終了 Index を検索します。検索対象数列が全て検索窓縦幅内の変動であれば開始 Index は 0、終了 Index は終端 Index として戻り、検索対象数列のデータが検索窓横幅内で全て縦幅内を越える変動している場合は−1 が戻ります。

記述例:

$1 を検索対象数列として、動作閾値 123、検索窓横幅 2 秒、縦幅 12 として検索する場合

```
$2 = SWP(INT(2*PRD(),12,123,$1)          /* 検索窓横 2 秒、縦 12,動作閾値 123*/
```

記述例:

$1 を検索対象数列として、動作閾値を全ての範囲、,検索窓横幅 2 秒、縦幅 12 として検索する場合

```
$2 = SWP(INT(2*PRD(),12,0,$1-MIN($1))          /* 検索窓横 2 秒、縦 12,動作閾値 123*/
```

18. 5. 1. 検索窓以内に存在する Peak の Index 及び表裾の Index を求める

【PDT 関数】を使用します。 **文**

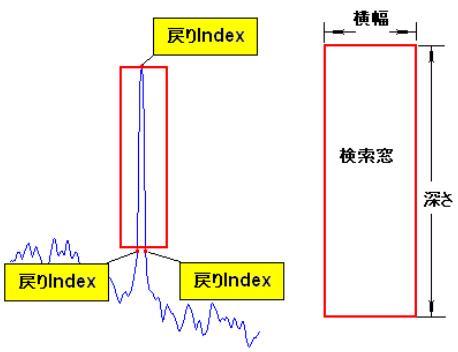
法:

格納先 = PDT(検索窓深さ、検索窓幅、対象数列,フラグ)

引数:

【検索窓深さ】＜必須＞ 検索窓の深さを記述します。深さは検索窓の縦幅と同じ意 味ですが検索対象データが検索窓を上端に位置して検索し ますので深さと呼称します。

【検索窓幅】＜必須＞ 検索窓横幅をデータ個数で記述します。検索窓は検索窓 横幅中心点が検索対象データ位置で、両側に同じデータ個 数分がとられます。従って記述するデータ個数が奇数の必 要があります。偶数で記述された場合は +1 して奇数にして 参照されます。



【対象数列】＜必須＞ 検索対象
　数列を記述します。
【フラグ】＜省略可＞
　深さを固定として取り扱うか、係数として取り扱うかのフラグを意味します。記述省略または 0 と記述した場合は深さをそのまま固定値して参照し。フラグに 1 を記述した場合は、検索対象データに記述した深さを掛けた値を検索窓深さとして参照し、検索データからの減衰比を設定したことになります。

検索窓のスライディング領域は設定された検索窓横幅が X 軸に入っている間をスライディングします。例えば検索窓横幅データ個数を 11 とした場合は、INT(11/2)=5 となり、Index5 から開始され、対象数列の終端 Index-5 までの範囲となります。検索は検索対象データ値を検索窓横幅中心の上端に位置させ、検索データより遡ったデータが検索窓底面より下回り、同様に検索データから後方に至るデータが検索窓底面より下回った場合に成立し、戻り数列は成立した時の検索対象データ位置 Index、前方に底面を下回った地点 Index、後方に底面を下回った地点 Index をペアとして、成立した箇所分戻ります。成立した箇所が存在しない場合は-1 が戻ります。

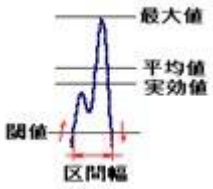
記述例：
　$1 を検索対象数列として深さ 20、幅 55 として検索する場合

```
$2 = PDT(20,55,$1)              /*検索窓深さ 20、幅 55 点*/
case $2 > 0
  proc detect{
    $3 = SEP(0,2,$2)            /* 検出ピーク位置 index*/
    $4 = SEP(1,2,$2)            /* 検出ピーク前方減衰位置 Index*/
    $5 = SEP(2,2,$2)            /* 検出ピーク後方減衰位置 Index*/
    $6 = PTV($3,$1)             /* 検出ピーク値*/
  }detect
```

$3 に検出したピーク位置 Index、$4 に前方減衰位置 Index、$5 に後方減衰位置 Index が格納されます。

記述例18. 1. 閾値を上昇で通過してから下降で通過する区間の最大値、平均値、実効値と区間幅を求める

ファイルを選択後、解析対象チャネルと閾値を指定し、区間ごとの平均値、実効値、最大値と区間開始地点、区間幅を結果シートに書き込みます。(その他項目の書き込み事項は結果シートの表示例を参照下さい) 解析は閾値を上昇で通過してから下降で通過するまでの間を区間と見なして処理します。
なお、区間検出には LST 関数を使用しています。

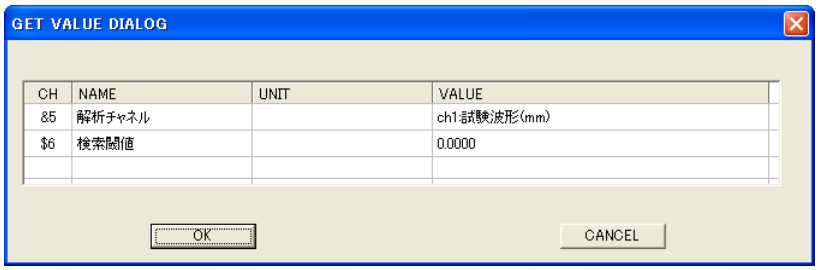


＜Archi_1 Script 記述例＞

```
1         /*--- 18. 1.proc ----*/
2         dcl menu_label "区間幅,最大値,平均値,実効値演算" 1
3         /*-------------------------------------------------------------*/
4         dcl sheet 1 {
5           page 1:
6             column &1,$20,$4,&3,$6,$2,$13,$19,$7,$8,$9,$10
7             format A,F4,F0,A,F4,F0,F4,F0,F4,F4,F4,F4 2
8         }sheet
9         /*-------------------------------------------------------------*/
10        def ch_name &1 "ファイル名"
11        def ch_name $1 "ファイル数"
12        get file_name &1 $1 "hdr"                          /* 解析対象ファイル選択*/
13        case $1 > 0
14          proc exec{
15            def file_id %1 &1 wav                          /* 収録データファイル番号定義*/
16            read wave %1                                    /* 収録データファイル読み出し*/
17            $3 "収録チャネル数:" = NCH()                    /* 収録チャネル数取得*/
18            $4 "解析 ch 番号:" = CHS()                      /* 収録チャネル番号取得*/
19            &3 "信号名:" = CHNM(0)                          /* 信号名取得*/
20            &4 "単位:" = CUNT(0)                            /* 信号単位取得*/
21            &6 "サンプリング単位:" = CXUT()                 /* サンプリング単位取得*/
22            def ch_unit $7 &6                              /* サンプリング単位定義*/
23            def ch_unit $13 &6                             /* サンプリング単位定義*/
24            $15 "サンプル周期数列:" = SPB(0)+XOF()          /* サンプリング周期数列生成*/
25            assign &2 "解析チャネル配列:" = "ch"|$4(F0)|":"|&3|"("|&4|")"
26            assign &5 "解析チャネル:" = &2(0)
27            $6 "検索閾値:" = 0                              /*閾値初期化*/
28            get value &2:&5,$6(F4)                         /* チャネル選択&閾値設定*/
29            $5 = CTN(CEXT(3,3,&5))                          /* 選択チャネル抽出*/
30            $5 "処理 ch_No.index:" = CMP(0,$5,$4)           /* チャネル Index 取得*/
31            def ch_unit $8 &4($5)                          /* 平均値に信号単位定義*/
32            def ch_unit $9 &4($5)                          /* 実効値に信号単位定義*/
33            def ch_unit $10 &4($5)                         /* 最大値に信号単位定義*/
34            $14 "対象波形論理値:" = GT_(#($4($5)),$6)       /* 対象チャネル論理化*/
35            $12 "閾値下降通過 index:" = LST(0,$14)          /* 閾値下降通過 index 検索*/
36            $14(0) = GTE(#($4($5))(0),$6)                  /* 対象チャネル論理化*/
37            $11 "閾値上昇通過 index:" = LST(1,$14)          /* 閾値上昇通過 index 検索*/
38            $12 = RVS(GT_($11(0),$12(0)),$12,ERC(1,LEN($12)-1,$12)) /* 閾値通過点整合*/
39            $11 = RVS(GT_($11(LEN($11)-1),$12(LEN($12)-1)),$11,ERC(0,LEN($11)-2,$11))
40            $18 = MGR($11,$12+1)                            /* 演算範囲数列生成*/
41            $13 "区間始点:" = PTV($11,$15)                  /* 区間開始地点抽出*/
42            $7 "区間幅:" = ($12-$11)*PRD()                  /* 区間幅演算*/
43            $17 "区間数:" = LEN($11)
44            $2 "区間始点 Index:" = $11
45            $19 "データ個数:" = $12-$11+1                   /* 区間データ個数演算*/
46            $20 "サンプル周期:" = PRD()                     /* サンプリング周期表示用*/
47            def ch_unit $20 &6                             /* サンプリング単位定義*/
48            $8 "区間平均値:" = SEP(0,1,MEA($18,#($4($5)))) /* 区間平均値演算*/
49            $9 "区間実効値:" = SEP(0,1,EFF($18,#($4($5)))) /* 区間最大値演算*/
50            $10 "区間最大値:" = SEP(0,1,MAX($18,#($4($5))))/* 区間最大値抽出*/
51            write ch_column 1: &1($5),$4($5),&3($5),$13,$6,$7,$8,$9,$10,$19,$20,$2
52            write line_feed 1: 1
53          }exec
54        end
```

＜Archi_1 Script 構文記述の説明＞

34, 35,48,49,50 行目で演算式中に記述している#($4($5))は、収録チャネルの間接指定を意味し、#(収録チャネル番号)と記述しています。収録チャネル番号は$4 数列の収録順に格納されており、$5 はその Index を示しています。28 行目の get value 文で取得する内容は、解析チャネル選択を容易にする為、チャネル番号と信号名、単位を連結した文字列を選択しており、その文字列に含むチャネル番号文字列を 29 行目で切り出し数値に変換した後、30 行目で Index に変更します。従って、#($4($5))は#(収録チャネル数列(index))と言う意味になり、解析対象数列(解析対象収録チャネル)となります。 実行するとファイル読み出しダイヤログが表示されますので、解析対象ファイルを選択します。選択を終了すると、解析対象チャネル と検索閾値設定用ダイヤログが表示されます。解析チャネルはリストボックスから選択します。



34 行目～37 行目では区間の検出は解析対象チャネルを設定された閾値で論理値数列に変換し、LST 関数により論理遷移地点 Index を検索しています。検索開始 Index のデータが閾値と同じで、検索開始 Index+1 の値が閾値より大きかった場合、34 行目の処理では、検索開始 Index+1 で 0～1 の遷移が発生するため、上昇での通過を検索する前に 36 行目の処理で検索開始 Index の値を 1 として 0～1 の遷移が発生しないようにしています。

38 行目～39 行目の演算は、0 から 1 に遷移した Index から始まり、最終 Index は 1～0 に遷移した地点で終了する様に整合します。平均値、実効値、最大値の演算は MEA 関数、FEE 関数、MAX 関数を使用して行っています。それぞれの関数に演算範囲を与える為、上昇通過 Index と下降通過 Index を、MRG 関数を使用して交互に縫合しています。40 行目がその処理に相当します。又、演算結果は閾値を下降で通過して地点から上昇で通過する区間も演算してしまう為、演算結果を SEP 関数で分離しています。48 行目～50 行目がその処理に相当します。

＜表示される結果シート＞

Page1:

ファイル名	サンプル周期(m)	解析ch番号	信号名	検索閾値	区間始点Index	区間始点(m)	データ個数	区間幅(m)	区間平均値(mm)	区間実効値(mm)	区間最大値(mm)
test_wave	0.2500	1	試験波形	0.0000	74	18.5000	56	13.7500	1.0386	1.1983	1.9016
					218	54.5000	64	15.7500	2.4905	2.7158	4.0784
					314	78.5000	17	4.0000	0.4090	0.4857	0.7213
					412	103.0000	121	30.0000	1.6805	1.8196	2.6303
					652	163.0000	14	3.2500	0.1918	0.2229	0.3231
					686	171.5000	104	25.7500	1.6799	1.9452	2.9712
					865	216.2500	26	6.2500	0.7627	0.9021	1.3731
					961	240.2500	82	20.2500	2.0872	2.3555	3.5217
					1061	265.2500	25	6.0000	0.7201	0.8255	1.2188
					1167	291.7500	13	3.0000	0.3007	0.3474	0.5040
					1191	297.7500	57	14.0000	1.9181	2.1734	3.5580
					1272	318.0000	50	12.2500	0.9734	1.0920	1.8069
					1372	343.0000	52	12.7500	0.4501	0.5468	1.0333
					1515	378.7500	71	17.5000	1.4630	1.5794	2.3130
					1609	402.2500	4	0.7500	0.0123	0.0170	0.0253
					1707	426.7500	77	19.0000	2.4739	2.7730	4.3586
					1806	451.5000	9	2.0000	0.2087	0.2660	0.3855
					1882	470.5000	44	10.7500	1.4499	1.6257	2.4284
					1949	487.2500	112	27.7500	1.2115	1.3710	2.5693
					2131	532.7500	9	2.0000	0.1191	0.1363	0.1902

19. 演算関数を使用して計測データの基本処理を行う

PcWaveForm 演算関数を使用して収録データの基本的処理を行う方法について説明します。

19. 1. フィルタ処理する

19. 1. 1. 移動平均処理によりローパスフィルタ処理する

【MAV 関数】を使用します。

文法：

格納先 = MAV(平均化個数,解析対象数列)

引数：

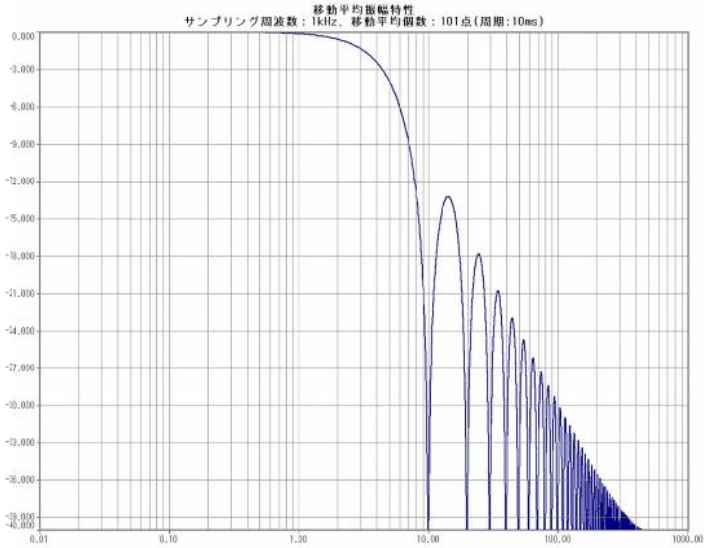
【平均化個数】＜必須＞ 移動平均するデータ個数を記述します。偶数で記述された場合、平均化個数間隔で記述されたと見なし、+1 して 参照されます。※ 平均化個数間隔とは平均化個数の間隔数です。

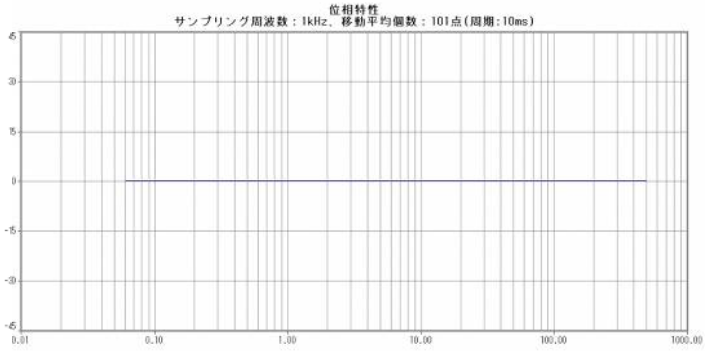
【解析対象数列】＜必須＞ 解析対象数列を記述します。

移動平均によりローパスフィルタ処理する場合は移動平均処理自体がローパスフィルタ特性を持ちますので MAV 関数の引数に平均化個数を記述することで行います。

＜移動平均処理の基本特性＞

1kHz サンプリングしたインパルスを平均化個数 101 点(平均化個数間隔 100 点)で移動平均した場合の振幅特性と位相特性を示します。





振幅特性はローパスフィルタとなり移動平均周期 10Hz(100ms)の倍数でゲイン 0 となる特性を示し、位相特性は平坦特性を示します。

f を周波数(Hz)、τ を移動平均周期(sec)として移動平均の振幅特性を式で表すと

$$\square = \frac{1}{2\pi\square\tau}\sqrt{2(1-\square\square\square 2\pi\square\tau)}$$

となります。つまり、τ =0.1sec、f=10Hz の倍数の時、√内の cos2πfτ は 1 となり、ゲイン 0 となります。

遮断周波数 f(-3dB 位置周波数)はゲインが 1/√2(-3dB)の周波数で或る事から

$$\frac{1}{\sqrt{2}} = \frac{1}{2\pi f_c \tau}\sqrt{2(1 - \cos 2\pi f_c \tau)}$$

$$\pi f_c \tau = \sqrt{1 - \cos 2\pi f_c \tau}$$

となり、三角関数公式から

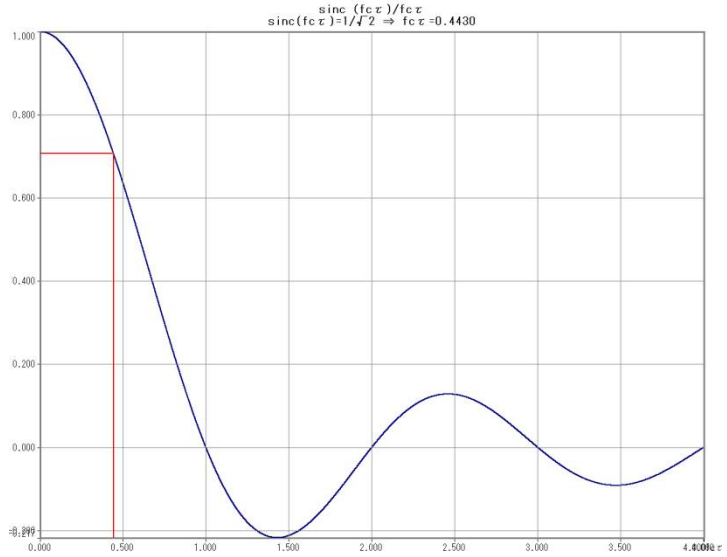
$$\sin\frac{\alpha}{2} = \pm\sqrt{\frac{1-\cos 2\alpha}{2}} \Rightarrow \sqrt{1-\cos 2\alpha} = \sqrt{2}\cdot\sin\frac{\alpha}{2}$$

$$\pi f_c \tau = \sqrt{2}\cdot\sin\pi f_c \tau$$

$$\frac{1}{\sqrt{2}} = \frac{\sin \pi f_c \tau}{\pi f_c \tau} = \mathrm{sinc}(f_c \tau)$$

となります。

sinc 関数の値が 1/√2 地点の $f_c \tau$ をグラフから求めると



$f_c \tau$ は 0.4430 となります。τ はサンプリング周期 Δt×(平均化個数間隔)となりますので 遮断周波数 f は、平均化個数間隔を M とし、サンプリング周波数を fs とすると

$$f_c = \frac{0.443}{M}\cdot\frac{1}{\Delta t} = \frac{0.443}{M}\cdot f_s$$

となります。

従って、1kHz サンプリングで移動平均個数を 101 点(平均化個数間隔 100 点)とした場合、遮断周波数は

遮断周波数=0.443/100 点×1000Hz=4.43Hz

となります。

記述例:

sinc関数の sinc(1/√2)の時の値をグラフから下 4 桁精度で求める場合

```
/*------ sinc 関数演算-----------------------------------------------------------*/
$1 "x:" = ACC(DAG(40000,0.0001))        /* sinc 関数の引数 X の数列を下 4 桁で生成*/
$2 "sinc:" = SIN(PI_()*$1)/(PI_()*$1)   /*sinc 関数の演算*/
$3 = 1/SQR(2)                           /*  検索閾値1/√2*/
$4 = PTV(DTD(0,$3,1,0,$2),$1)           /* sinc 関数の値が閾値を下降で過った地点の X 値を求める*/
/*-----  グラフ描画処理-----------------------------------------------------------*/
assign &1 = "sinc(fcτ)=1/√2 ⇒ fcτ="|$4(F4) /*グラフ副題生成*/
$6 = MIN($2)                             /* グラフ内追加線 Y 軸最小値演算*/
$3 = LNK($3,$3,MIN($2))                  /*  グラフ内追加線,Y 軸アドレス生成*/
$4 = LNK(MIN($2),$4,$4)                  /* グラフ内追加線,X 軸アドレス生成*/
def graph_id @1 "sinc(fcτ)/fcτ" &1 /* グラフ番号定義*/
def graph_x_axis @1 0,0 F3 "fcτ"       /* グラフ X 軸定義*/
def graph_y_axis @1 0,0 F3 5           /* グラフ Y 軸定義*/
```

```
plot @1 $1,$4 $2,$3                    /* グラフ描画*/
def file_id %1 "graph" grp             /* グラフ格納先ファイル番号定義*/
save plot %1 @1 $1,$4 $2,$3            /* グラフ格納*/
end
```

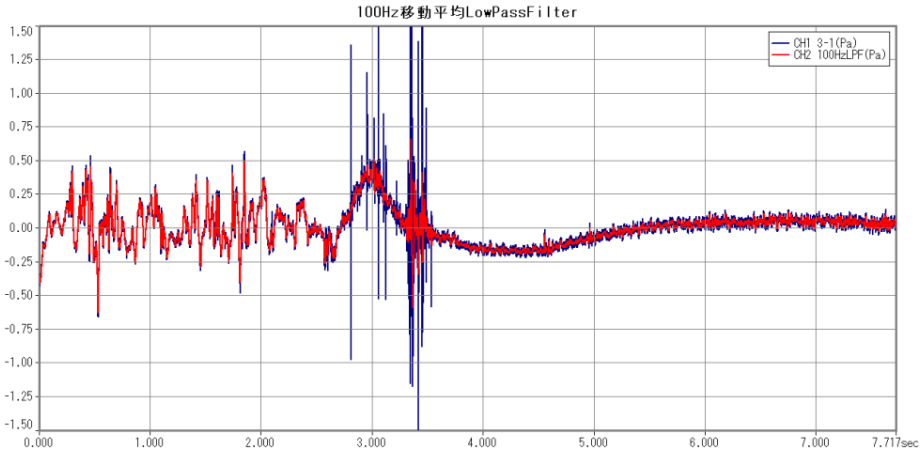
記述例:

収録したデータのカレントチャネルに遮断周波数 100Hz のローパスフィルタ処理を行う場合

```
/*------ 移動平均により 100Hz ローパスフィルタ処理----------------------------*/
$1 = CCH()                             /* カレントチャネル番号取得*/
$2 = INT(0.443/(100*PRD()))            /* 移動平均遮断周波数平均化個数間隔演算*/
&1 = CUNT($1)                          /* カレントチャネルの単位を取得*/
def ch_unit $3 &1                      /* フィルタ処理後のチャネルに単位を定義*/
$3 "100HzLPF:" = MAV($2,#($1))         /* 遮断周波数 100Hz 移動平均演算*/
/*------グラフ描画処理-----------------------------------------------------*/
$4 = SPB(0)                            /* グラフX 軸用経過時間数列生成*/
def graph_id @1 "100Hz 移動平均 LowPassFilter" /* グラフ番号定義*/
def graph_x_axis @1 0,0 F3 "sec"       /* グラフX 軸定義*/
def graph_y_axis @1 -1.5,1.5,0.25 F2 2 /* グラフ Y 軸定義*/
def graph_aspect_ratio @1 2            /* グラフ枠縦横比定義*/
plot @1 $4,$4 #($1),$3                 /* グラフ描画*/
def file_id %1 "graph" grp             /* グラフ格納先ファイル番号定義*/
save plot %1 @1 $4,$4 #($1),$3         /* グラフ格納*/
end
```

$2 の演算が遮断周波数から平均化個数間隔を求めています。

<実行結果グラフ>

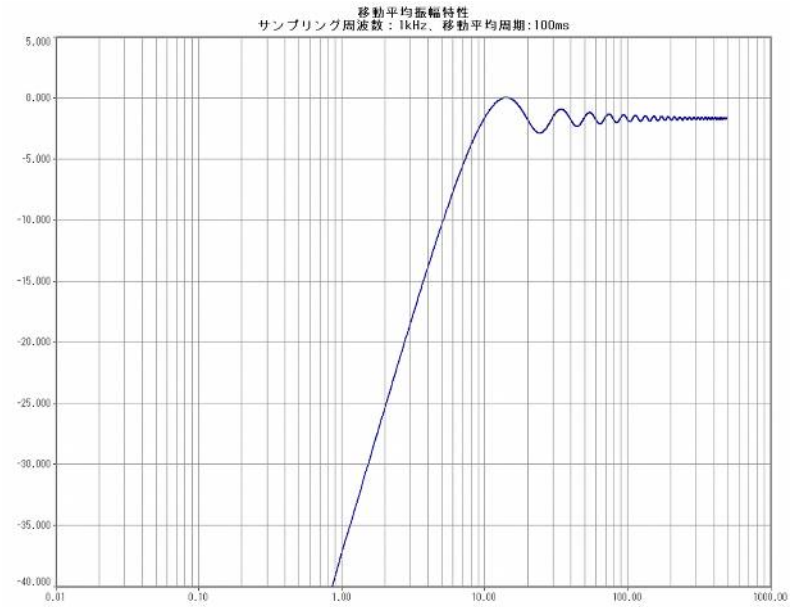


青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ローパスフィルタ通過後波形を示します。

移動平均処理に於ける過渡領域 過渡領域は先端と終端に発生します。過渡領域とは正常にフィルタが掛らない領域を意味します。 過渡領域は設定した平均化個数間隔に依存し、平均化個数間隔を M とすると、先端から M/2 迄と終端から M/2 迄が過 渡領域となります。これは移動平均後のデータ数と移動平均前のデータ個数を一致させる為の処置で、MAV 関数では、 平均化個数間隔の 1/2(片翼)に達する迄、徐々に両側に翼を広げ、設定した平均化個数間隔の片翼に達すると設定し た平均化個数間隔のまま移動平均を行い、終端に近づき残りデータ数が片翼に満たなくなると翼を畳む処理を行う為に 発生します。

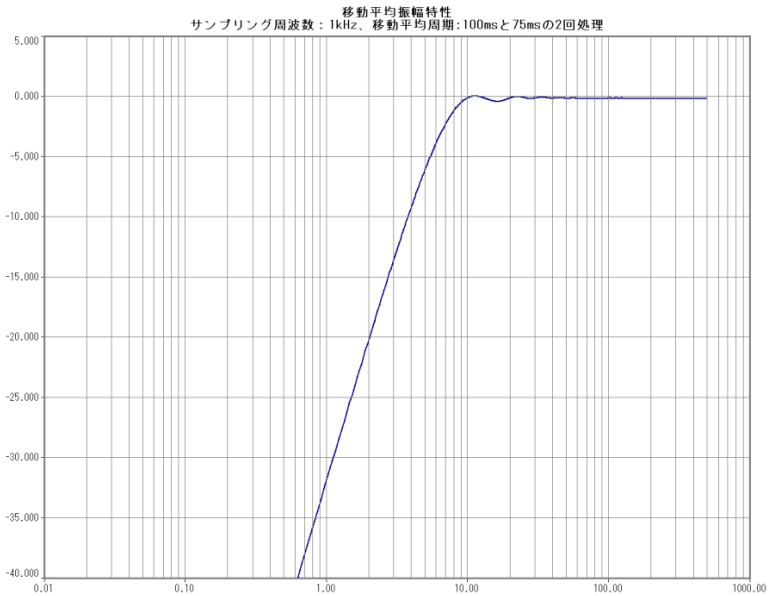
19. 1. 2. 移動平均処理によりハイパスフィルタ処理する

移動平均により簡易的にハイパスフィルタを掛ける場合、フィルタ処理前のデータから移動平均後のデータを減算する方法があります。但し、そのまま減算すると平坦部特性にうねりが発生します。サンプリング周波数 1kHz のインパルスに 101 点の移動平均処理した結果を元のデータから減算した周波数特性を示します。



基本的には、元の波形から移動平均した結果を減算した結果は、移動平均した結果グラフを上下逆さまにした特性を持ちます。(但し、上図は FFT の X 軸刻み幅 Δf が減衰 0 に当たらないこと及びグラフ表現が dB であることの考慮が必要)
なお、遮断特性及び遮断周波数はローパスフィルタと異なります。

平坦部のうねり(落ち込み)は、周波数をずらした移動平均処理を 2 回掛けることで軽減出来ます。平均化個数間隔 100 点の移動平均に更に 75 点の移動平均を行った結果を元の波形から減算した結果を示します。



記述例:

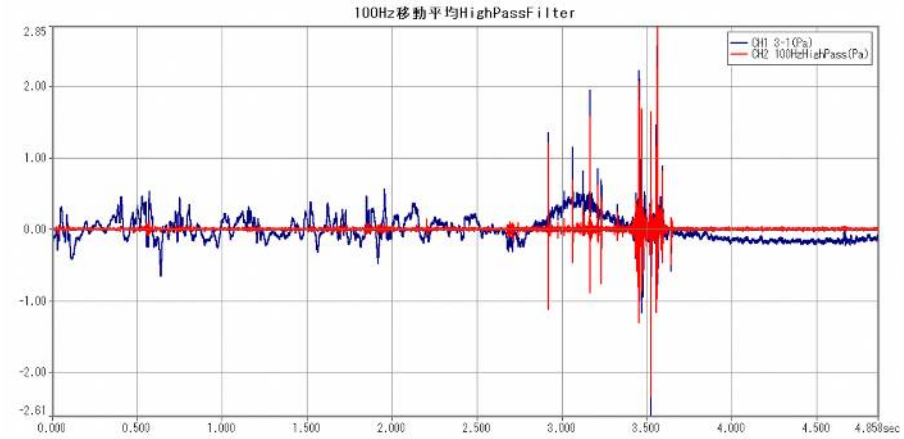
収録したデータのカレントチャネルに遮断周波数 100Hz ローパスフィルタ処理と 75Hz のローパスフィルタ処理行い元のデータから減算してハイパスフィルタ処理を行う場合

```
/*------ 移動平均 2 回処理ハイパスフィルタ処理----------------------------*/
$1 = CCH()                          /* カレントチャネル番号取得*/
$2 = INT(0.443/(100*PRD()))         /* 移動平均データ個数演算*/
&1 = CUNT($1)                       /* カレントチャネルの単位を取得*/
def ch_unit $3 &1                   /* フィルタ処理後のチャネルに単位を定義*/
$3 "2 回移動平均:" = MAV($2*3/4,MAV($2,#($1))) /* 遮断周波数 100Hz,150Hz 移動平均演算*/
$3 "HighPass:" = #($1)-$3           /* 元のデータから 2 回移動平均結果を減算*/
/*------グラフ描画処理-------------------------------------------------*/
$4 = SPB(0)                         /* グラフ X 軸用経過時間数列生成*/
def graph_id @1 "100Hz 移動平均 HighPassFilter" /* グラフ番号定義*/
def graph_x_axis @1 0,0 F3 "sec"            /* グラフ X 軸定義*/
def graph_y_axis @1 0,0 F2 2                /* グラフ Y 軸定義*/
```

```
def graph_aspect_ratio @1 2
plot @1 $4,$4 #($1),$3                          /* グラフ描画*/
def file_id %1 "graph" grp          /* グラフ格納先ファイル番号定義*/
save plot %1 @1 $4,$4 #($1),$3     /* グラフ格納*/
end
```

遮断周波数 100Hz の移動平均と、その 3/4 の 75Hz を遮断周波数とした移動平均の 2 回処理した結果を元の波形から減算しハイパスフィルタ処理した結果を示します。なお、前述通りの遮断周波数は 100Hz ではありません。

<実行結果グラフ>



青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ローパスフィルタ通過後波形を示します。

19. 1. 3. 予め設計した FIR フィルタ係数を使用してフィルタ処理する

【FIR 関数】を使用します。

文法:

結果格納先 = FIR(フィルタ係数数列,対象数列)

引数:

【フィルタ係数数列】＜必須＞

フィルタ係数数列を記述します。なお、FIR フィルタ係数生成機能は PcWaveForm に内蔵していません。他のディジタルフィルタ設計プログラムを使用して求めた係数数列を記述します。

【対象数列】＜必須＞ フィルタを掛ける数列を記述します。なお、対象数列の要素数は、フィルタ係数数列の要素数より充分に大きい必要 があります。

FIR フィルタ処理に於ける過渡領域

FIR フィルタ処理での過渡領域は先端からフィルタ係数の要素数/2、終端から同じくフィルタ係数の要素数/2 に発生します。この領域は片翼分のデータ数を満足していな為、不足分のデータを 0 として付加しフィルタ処理します。例えば、先頭データの処理は FIR フィルタの中心から前翼のデータは全て 0 で後翼データだけが処理される事になります。従って、過渡領域とは正しくフィルタ処理されない範囲を意味します。元のデータ個数とフィルタ処理後のデータ個数を一致させる為で す。又、フィルタ係数はサンプリング周波数に依存しますので、フィルタ係数は設計時と、フィルタを掛けるデータのサンプリング周波数が一致していない場合は異なったフィルタ周波数特性となり、正しくフィルタが掛らない事に留意ください。

但し、FIR フィルタは正規化周波数を使用して設計しますので(正規化周波数とはフィルタ周波数/サンプリング周波数)、例えば正規化周波数 0.005 で設計した場合、サンプリング周波数 5kHz では 25Hz を意味し、サンプリング周波数 10kHz の場合は、50Hz を意味します。

記述例:

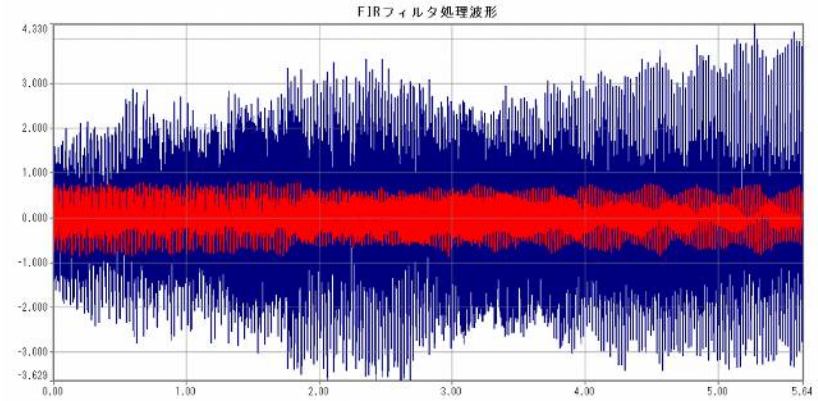
サンプリング周波数 5kHz で 25Hz～120Hz バンドパスフィルタ(係数 1023 個)で設計したバンドパスフィルタを収録データのカレントチャネルに掛ける場合

```
/*----フィルタ係数読み込み-----------------------------------------*/
def file_id %1 "25Hz-120Hz_BPF_Fs_5kHz" txt
read cell %1 1,1 1 $1                    /* フィルタ係数ファイル読み出し*/
/*----FIR フィルタ処理---------------------------------------------*/
$2 = CCH()                                 /* カレントチャネル番号取得*/
$3 = FIR($1,#($2))                        /* カレント chFIR フィルタ処理*/
/*---重ね書き時間軸波形グラフ処理----------------------------------*/
$4 = SPB(0)                              /* グラフ描画用時間軸数列生成*/
def graph_id @1 "FIR フィルタ処理波形" /*グラフ番号定義*/
def graph_x_axis @1 0,0 F2
```

```
def graph_y_axis @1 0,0 F3 5
def graph_aspect_ratio @1 2
plot @1 $4,$4 #($2),$3                    /* グラフ描画*/
def file_id %2 "graph" grp                /*グラフ格納先ファイル番号定義*/
save plot %2 @1 $4,$4 #($2),$3            /* グラフ格納*/
/*---重ね書き FFT グラフ処理----------------------------------------*/
$5 = SEP(2,1,PCX(FFT(4096,5,0,80,#($2)))) /* 対象数列 FFT 解析*/
$8 = MAX($5)                              /* 元波形の最大スペクトラム値*/
$5 = 20*LGT($5/$8)                        /*元波形の最大スペクトラム 0dB とした dB 化*/
$6 = SEP(2,1,PCX(FFT(4096,5,0,80,$3))) /* フィルタ通過後 FFT 解析*/
$6 = 20*LGT($6/$8)                        /* 元波形の最大スペクトラム 0dB とした dB 化*/
$7 = FFQ(-1,4096)                         /* FFT 周波数数列取得*/
def graph_id @2 "FFT スペクトラ解析" /* グラフ番号定義*/
def graph_x_axis @2 0,0 F2 log
def graph_y_axis @2 0,0 F1 5
plot @2 $7,$7 $5,$6                  /* 周波数グラフ描画*/
def file_id %2 "graph1" grp       /*グラフ格納先ファイル番号定義*/
save plot %2 @2 $7,$7 $5,$6       /* グラフ格納*/
end
```

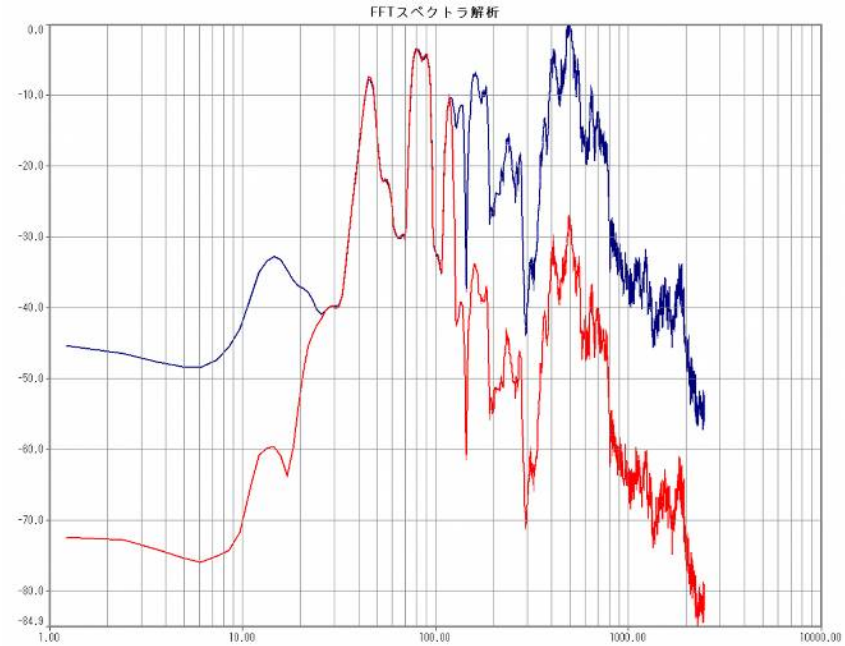
記述例では、フィルタ係数をテキストファイルから読み込んでいます。フィルタ係数は Script 中に assign 文を使用して埋め込む事も可能です。各種の遮断特性を持つフィルタを使用する場合は、ファイルの列ごとにフィルタ係数を記述し、状況に合わせて選択する方法を推奨します。FFT 関数については 21 章 21.2.「FFT 解析」の項を参照下さい。

＜バンドパスフィルタ処理結果グラフ＞



青線表示がバンドパスフィルタ通過前、赤線表示が 25Hz-120Hz バンドパスフィルタ通過後を示します。

<バンドパスフィルタ処理スペクトラム周波数結果グラフ>



青線表示がバンドパスフィルタ通過前、赤線表示が 25Hz-120Hz バンドパスフィルタ通過後を示します。

19. 1. 4. 4 次バターワースフィルタでローパスフィルタ処理する 計測データ処理では、フィルタ特性が規定されているバターワースフィルタが多用されています。演算関数では遮断特性-24dB/oct の 4 次バターワースフィルタ(IIR 形式)関数が用意されています。ローパスフィルタ処理する場合【LPF 関数】又は【LPR 関数】を使用します。

文法:

結果格納先 = LPF(遮断周波数,対象数列) = LPR(遮断周波数,対象数列)

LPF 関数とLPR 関数はフィルタを掛ける方向の違いで、LPF 関数は時間軸順方向、LPR 関数は時間軸逆方向に処理する方向の違いで特性に違いはありません。

引数:

【遮断周波数】＜必須＞

遮断周波数を記述します。遮断周波数はサンプリング周波数の 1/4 以下の必要があります。

【対象数列】

フィルタ処理する対象数列を記述します。

IIR フィルタ処理に於ける過渡領域

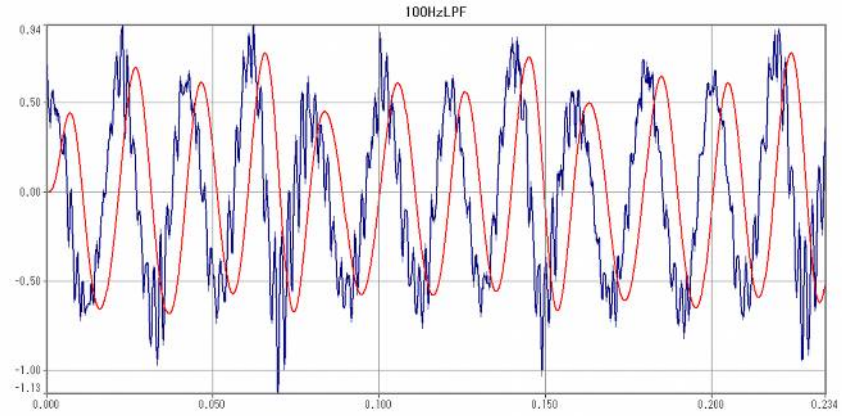
過渡領域はフィルタ処理先端に発生し、領域の大きさは設定した遮断周波数とフィルタ処理開始データに依存します。丁度、アナログフィルタ回路の電源投入時と同じ様な振る舞いとなります。なお、終端部に過渡領域は存在しません。

記述例:

収録データカレントチャネルに 100Hz ローパスフィルタを掛ける場合

```
/*----- 100Hz  4 次バターワースローパスフィルタ処理----------*/
$1 = CCH()
$2 = LPF(100,#($1))        /* 100HzLPF 処理*/
/*----- グラフ描画処理----------------------------------------*/
$3 = SPB(0)
def graph_id @1 "100HzLPF"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 2
plot @1 $3,$3 #($1),$2
def file_id %1 "graph" grp
save plot %1 @1 $3,$3 #($1),$2
end
```

<実行結果グラフ>



青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ローパスフィルタ通過後波形を示します。結果グラフから判る通り、フィルタ処理の結果は遅延を伴います。

遮断特性は-24dB 以上となりますが、位相遅延を嫌う場合は、LPF 関数で処理した結果に同じ遮断周波数の LPR 関数で時間軸逆方向にフィルタ処理することで位相遅延をキャンセルする事ができます。

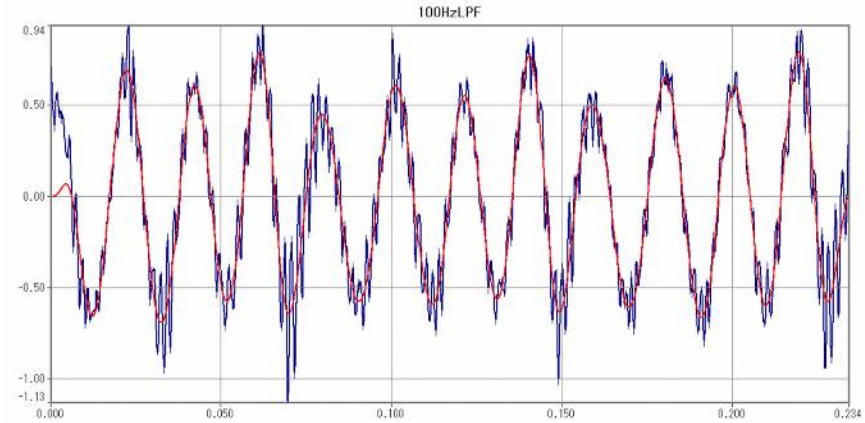
記述例:

収録データカレントチャネルに 100Hz ローパスフィルタを時間軸方向と逆方向の 2 回処理する場合

```
/*----- 100Hz 4 次バターワースローパスフィルタ 2 回処理----------*/
$1 = CCH()
$2 = LPF(100,LPR(100,#($1)))        /* 順方向と逆方向に 2 回フィルタ処理*/
/*----- グラフ描画処理----------------------------------------*/
$3 = SPB(0)
```

```
def graph_id @1 "100HzLPF"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 2
plot @1 $3,$3 #($1),$2
def file_id %1 "graph" grp
save plot %1 @1 $3,$3 #($1),$2
end
```

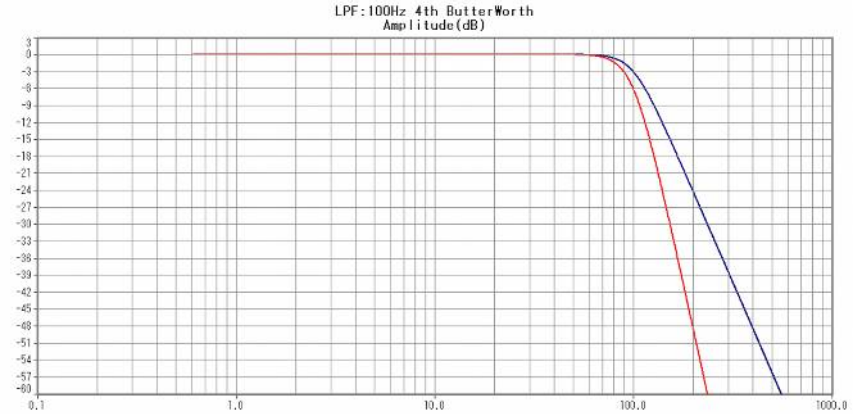
<実行結果グラフ>



青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ローパスフィルタ正方向と逆方向の 2 回通過後波形を示します。グラフから判る様に位相遅延がキャンセルされます。

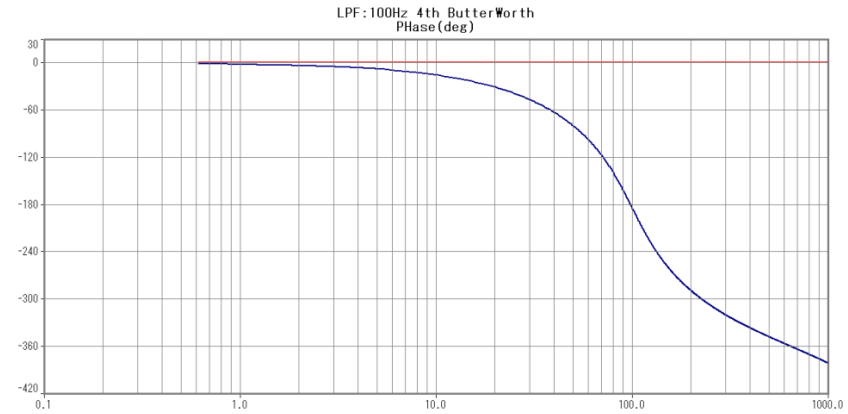
4 次バターワース 100Hz ローパスフィルタ LFP1回処理と LPF と LPR の 2 回処理の特性を示します。

<振幅特性グラフ>



青色表示が LPF 処理結果、赤色表示が LPF と LPR の 2 回処理結果を示します。
4 次バターワースローパスフィルタの振幅平坦部最大周波数は遮断周波数×0.46807 となりますので遮断周波数 100Hz の場合は、100×0.46807=46.807Hz となります。
LPF と LPR の 2 回処理した場合、遮断周波数 100Hz 地点で-6dB、減衰量は-48dB/oct となります。

<位相特性グラフ>



青色表示は LPF 処理結果、赤色表示は LPF と LPR の 2 回処理結果を示します。
2 回処理して結果、位相遅れは無くなり、位相特性はフラットになります。

19. 1. 5. 4 次バターワースフィルタでハイパスフィルタ処理する ハイパスフィルタ処理する場合【HPF 関数】又は【HPR 関数】を使用します。 **文法:**

結果格納先 = HPF(遮断周波数,対象数列) = HPR(遮断周波数,対象数列)

HPF 関数と HPR 関数はフィルタを掛ける方向の違いで、HPF 関数は時間軸順方向、HPR 関数は時間軸逆方向に処理する方向の違いで特性に違いはありません。

引数:

【遮断周波数】＜必須＞

遮断周波数を記述します。遮断周波数はサンプリング周波数の 1/4 以下の必要があります。

【対象数列】

フィルタ処理する対象数列を記述します。

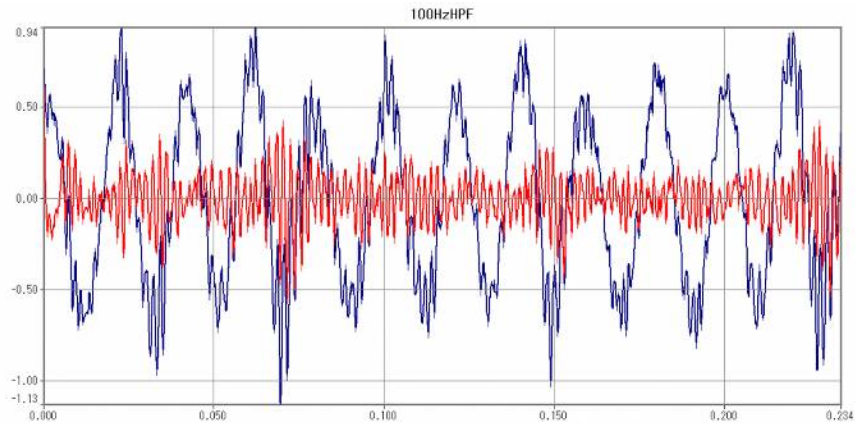
IIR フィルタ処理に於ける過渡領域 過渡領域はフィルタ処理先端に発生し、領域の大きさは設定した遮断周波数とフィルタ処理開始データに依存します。丁 度、アナログフィルタ回路の電源投入時と同じ様な振る舞いとなります。なお、終端部に過渡領域は存在しません。

記述例:

収録データカレントチャネルに 100Hz ハイパスフィルタを掛ける場合

```
/*----- 100Hz  4 次バターワースハイパスフィルタ処理----------*/
$1 = CCH()
$2 = HPF(100,#($1))        /* 100HzHPF 処理*/
/*----- グラフ描画処理----------*/
$3 = SPB(0)
def graph_id @1 "100HzHPF"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 2
plot @1 $3,$3 #($1),$2
def file_id %1 "graph" grp
save plot %1 @1 $3,$3 #($1),$2
end
```

<実行結果グラフ>



青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ハイパスフィルタ通過後波形を示します

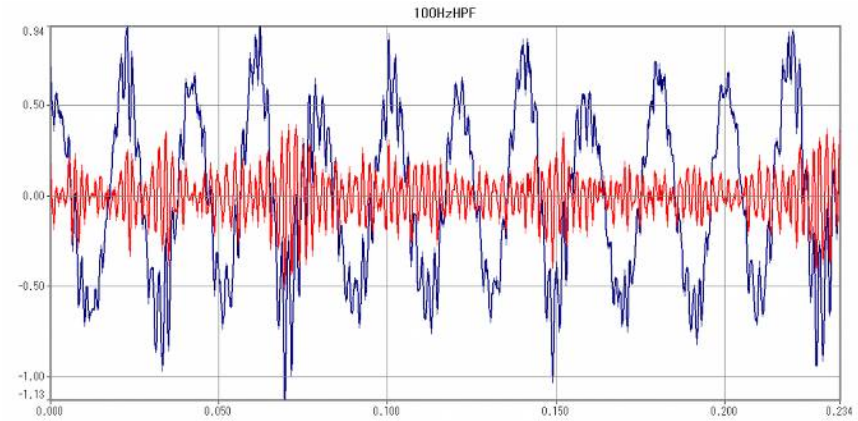
記述例:

収録データカレントチャネルに 100Hz ハイパスフィルタを時間軸方向と逆方向の 2 回処理する場合

```
/*----- 100Hz 4 次バターワースハイパスフィルタ 2 回処理----------*/
$1 = CCH()
$2 = HPF(100,HPR(100,#($1)))        /* 順方向と逆方向に 2 回フィルタ処理*/
/*----- グラフ描画処理----------*/
$3 = SPB(0)
def graph_id @1 "100HzHPF"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 2
```

```
plot @1 $3,$3 #($1),$2
def file_id %1 "graph" grp
save plot %1 @1 $3,$3 #($1),$2
end
```

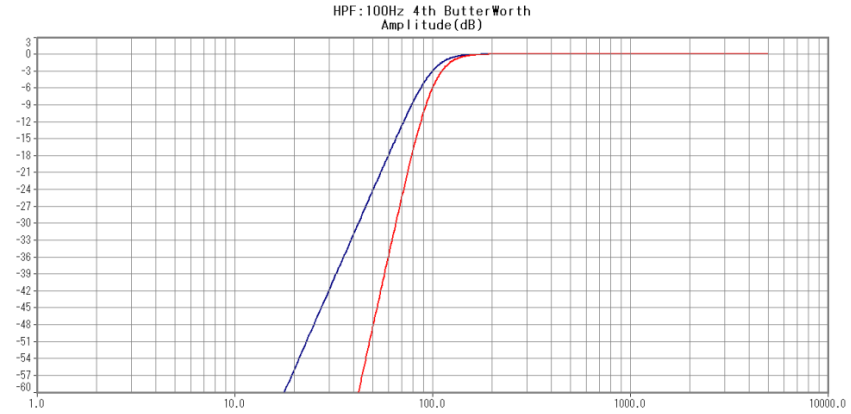
<実行結果グラフ>



青色表示がフィルタ前波形、赤線表示が遮断周波数 100Hz ハイパスフィルタ正方向逆方向の 2 回通過後波形を示します。

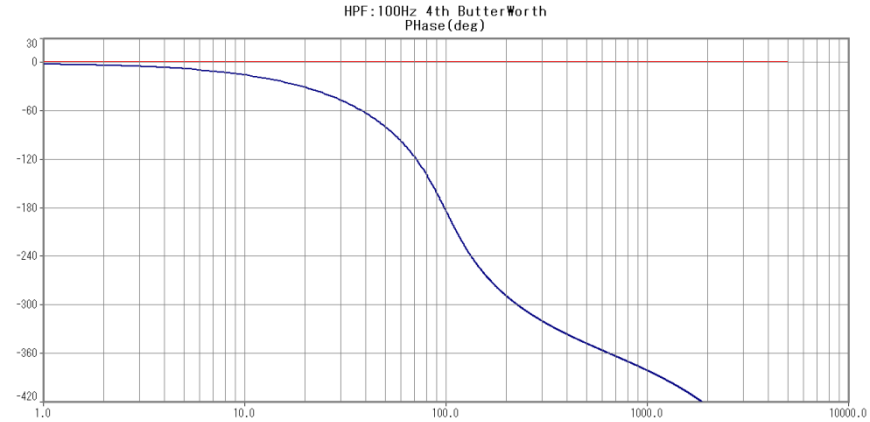
4 次バターワース 100Hz ハイパスフィルタ HFP1回処理と HPF と HPR の 2 回処理の特性を示します。

<振幅特性グラフ>



青色表示が LPF 処理結果、赤色表示が HPF と HPR の 2 回処理結果を示します。
4 次バターワースハイパスフィルタの振幅平坦部最小周波数は遮断周波数×2.13199 となりますので遮断周波数 100Hz の場合は、100×2.13199=213,2HzHz となります。
HPF と HPR の 2 回処理した場合、遮断周波数 100Hz 地点で-6dB、減衰量は-48dB/oct となります。

<位相特性グラフ>



青色表示が LPF 処理結果、赤色表示が HPF と HPR の 2 回処理結果を示します。位相特性はローパスフィルタと同じになります。

19. 2. キャリブレーション処理する

19. 2. 1. 線形キャリブレーションを行う

線形キャリブレーションとは直線方程式 Y=aX+b の傾き a とオフセット b を既知として行う方法です。主に収録データの単位変換に使用します。

記述例:

収録した ch1 の温度(℃)データを華氏単位に変換する場合

```
$1 "華氏:" = 9/5*#1-32          /* a=9/5,b=-32*/
```

19. 2. 2. 非線形キャリブレーションを行う

非線形キャリブレーションとは正負で傾きが異なる場合や、計測データに対して非線形なキャリブレーションを行う事を意味 します。非線形キャリブレーションは演算式により行う方法以外に、非線形テーブルを定義し、定義した非線形テーブルを 参照し部分線形補間処理(PeiseWiseLinear 法)により行う方法があります。部分線形補間を行う場合【ITP 関数】関数を使 用します。

文法:

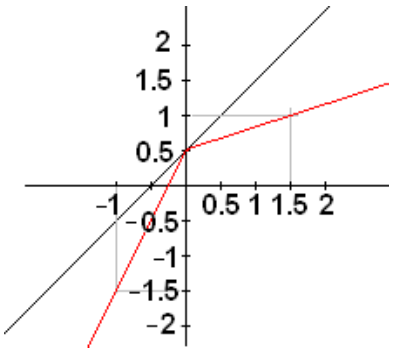
Y 値数列格納先 = ITP(非線形テーブル X 値,非線形テーブル Y 値,X 値数列)

※ ITP 関数の詳細は 17 章「17.7.4. X/Y テーブルと X 値数列から対応する Y 値数列を生成する」を参照下さい。

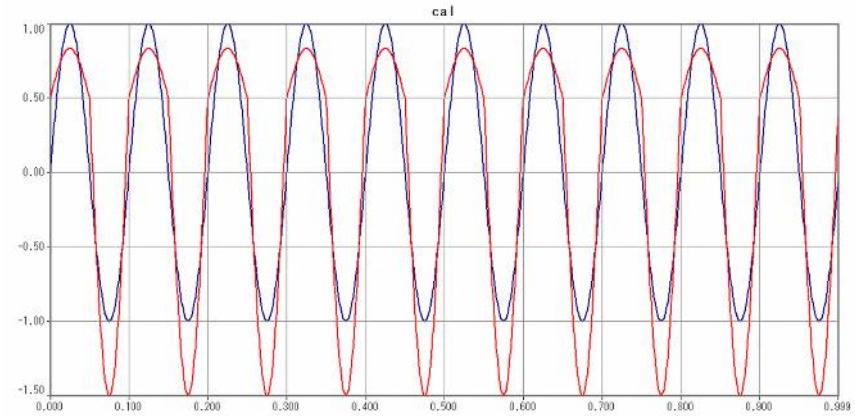
記述例:

生成した振幅 1 の正弦波に 0.5 以上の場合、傾き 1/2、0.5 以下の場合、傾き 2 としてキャリブレーションを行う場合

```
/*--- 正弦波生成----------*/
def sampl_period 1e-3 "sec" /*fs 1kHz 定義*/
$1 = DAG(1000,1,0,10,0) /* 1.0*sin(2π10)*/
/*----- テーブル定義----------*/
$2 "tbl_x:" = LNK(-1,0,1.5)
$3 "tbl_y:" = LNK(-1.5,0.5,1)
/*-----キャリブレーション----------*/
$4 = ITP($2,$3,$1)  /*部分線形補間*/
/* -----結果グラフ描画----------*/
$5 = (ACC(DAG(1000,1))-1)*1e-3
def graph_id @1 "cal"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 2
plot @1 $5,$5 $1,$4
def file_id %1 "graph" grp
save plot %1 @1 $5,$5 $1,$4
end
```



<実行結果グラフ>



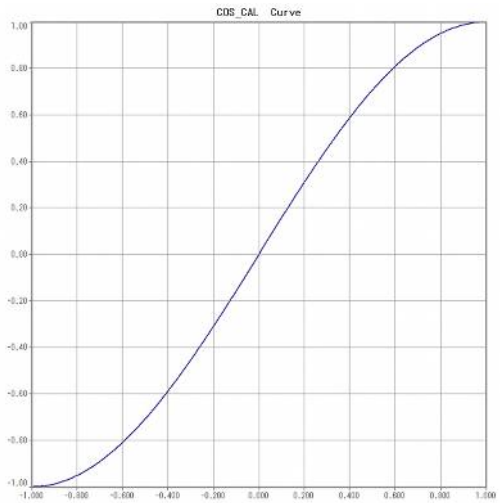
青色表示がキャリブレーション前波形、赤線表示がキャリブレーション後波形を示します。

記述例:

キャリブレーションする非線形カーブ(COS カーブ)を非線形テーブルとして定義し、収録データカレントチャネル最大値で 非線形テーブルを再構成してキャリブレーションする場合

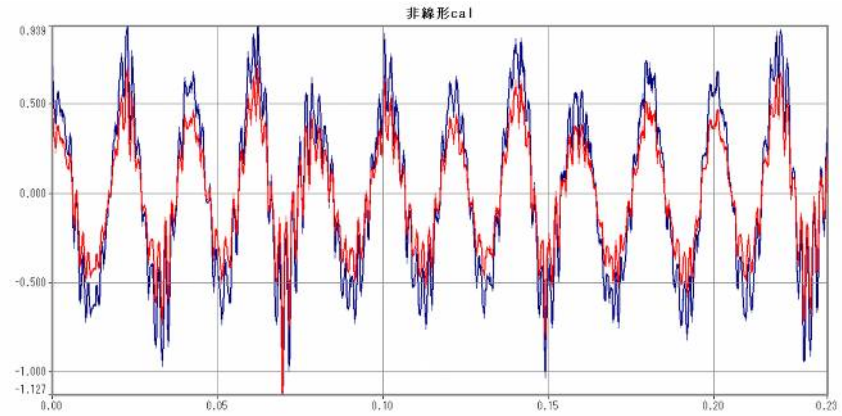
```
/* ---非線形テーブル定義----------*/
$1 = ACC(DAG(181,1))-1
$2 "TBL_X:" = REV(COS(RAD($1)))
$1 "TBL_Y:" = ($1-90)/90
```

```
/* --- 非線形キャリブレーション処理------*/
$3 "current_ch_No.:" = CCH()    /* カレント ch 番号取得*/
$4 "current_max:" = MAX(ABS(#($3))) /* 絶対値最大*/
$5 "cal:" = ITP($4*$2,$4*$1,#($3)) /*部分線形補間*/
/*-----CAL 前後波形描画処理------------*/
$6 "時間軸:" = (ACC(DAG(LEN(#($3)),1))-1)*PRD()
def graph_id @1 "非線形 cal"
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F3 5
def graph_aspect_ratio @1 2
plot @1 $6,$6 #($3),$5
def file_id %1 "grapgh" grp
save plot %1 @1 $6,$6 #($3),$5
/* ----非線形テーブルグラフ描画------*/
def graph_id @1 "COS_CAL Curve"
def graph_aspect_ratio @1 1
plot @1 $1 $2
def file_id %1 "graph" grp
save plot %1 @1 $1 $2
end
```



記述例では便宜的に絶対値最大を 1 として行っています。

<実行結果グラフ>



青色表示がキャリブレーション前波形、赤色表示が非線形キャリブレーション後を示します。

19. 2. 3. 予めファイルに格納された非線形テーブルを参照してキャリブレーションを行う 非線形キャリブレーションを行う場合、テーブルのペア個数が 30 以下の場合、キャリブレーションテーブルを Script 中に記 述せず、予め半角カンマ”,”区切りのテキスト形式、拡張子.tbl のファイルを作成することで演算式中で参照する事ができま す。テーブルファイルから参照して部分線形補間する場合は【TBL 関数】を使用します。

文法:

def file_id %ファイル番号 ファイル名 tbl
Y 軸値格納先 = X 軸値数列チャネル*TBL(テーブル名)

引数:

【テーブル名】＜必須＞ 格納されているテーブルファイル名を拡張子無しで文字属性参照チャネル又は文字列即値で記述します。なお、格 納先は Script のカレントフォルダに格納されている必要があります。

記述方法:

他の関数と異なり、使用する直前までにテーブルファイルのファイル番号定義が必要です。なお、ファイル番号定義 文のファイル属性は tbl と記述します。又、本関数の戻り値はありません演算式の掛け算に記述し、初めて機能します。
例えば、$1 をテーブル名”table”から補間して$2 に代入する場合は

def file_id %1 “table” tbl
$2 = $1*TBL(“table”) 又は $2 = $1*TBL(%1)

と記述します。なお、掛け算の順序が例えば$2 = TBL(“table”)*$1 とした記述は出来ません。

テーブルファイルフォーマット:

1 行目:PWF_TABLE ← keyword 半角大文字固定
2 行目:PAIR=10 ← PAIR=は半角大文字 KeyWord固定で、続いてテーブルペア数を記述します。
3 行目以降:X 軸値,Y 軸値 ← 半角カンマで”,”区切り、X 軸と Y 軸のペアを記述します。 なお、最大ペア数は 30 迄となり、また、必ず X 軸値昇順に記述します。

テーブル参照方式：

テーブルに掛け算される数列の値が記載されている X 軸値の範囲に入っている場合は当該区間の両隣の値が参照され補間した Y 軸値が戻り、範囲を外れている場合は最後の 2 点から補外された Y 軸値が戻ります。

記述例：

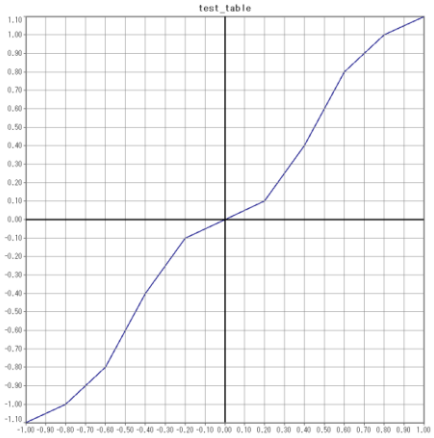
テーブルファイルの作成例を記述します。

```
/*-----Table 定義--------------------*/
assign $1 "table_X":"   = -1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1
assign $2 "table_Y:" = -1.1,-1,-0.8,-0.4,-0.1,0,0.1,0.4,0.8,1,1.1
/*----Table ファイル書き込み-------------*/
$3 "pair:" = LEN($1)
assign &1 = "PAIR="|$3(F0)
def file_id %1 "test_table.tbl" csv
def text_form %1 20,4
write cell %1 1,1 0 "PWF_TABLE"
write cell %1 2,1 0 &1
write cell %1 3,1 1 $1(F1),$2(F1)
save text_form %1
/*----- Table Graph 描画処理-------------*/
$4 = LNK(-1,1,0,0)
$5 = LNK(0,0,-1.1,1.1)
$6 = LNK(1,0,1,1)
def graph_id @1 "test_table"
def graph_x_axis @1 0,0,0.1 F2
def graph_y_axis @1 0,0,0.1 F2 5
def graph_line_draw @1 $4,$5,$6 2 "#000000"
def graph_aspect_ratio @1 1
plot @1 $1 $2
def file_id %2 "table_graph" grp
save plot %2 @1 $1 $2
end
```

※ 非線形テーブルファイルの作成はエクセル等で作成した後、拡張子.tbl で格納する方法を推奨します。

<作成したテーブルファイルの内容>

```
PWF_TABLE
PAIR=11
-1.0,-1.1
-0.8,-1.0
-0.6,-0.8
-0.4,-0.4
-0.2,-0.1
0.0,0.0
0.2,0.1
0.4,0.4
0.6,0.8
0.8,1.0
1.0,1.1
```

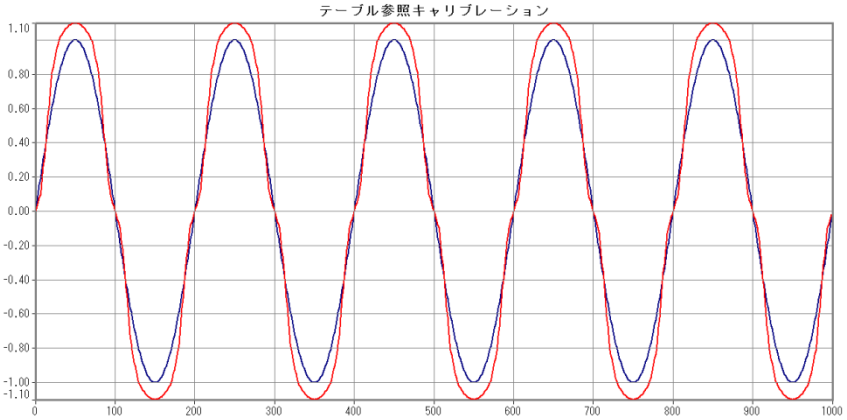


記述例：

生成した正弦波を格納されている非線形テーブルファイルでキャリブレーションを行う場合

```
/* -----正弦波生成---------------------*/
def sampl_period 1e-3 "sec"
$1 = DAG(1000,1,0,5,0)
/*-----テーブルファイル参照キャリブレーション-----*/
def file_id %1 "test_table" tbl
$2 = $1*TBL("test_table")
/*-----グラフ描画処理-------------------*/
def graph_id @1 "テーブル参照キャリブレーション"
def graph_x_axis @1 0,0 F0
def graph_y_axis @1 0,0,0.2 F2 5
def graph_aspect_ratio @1 2
plot @1 $1,$2
def file_id %1 "graph" grp
save plot %1 @1 $1,$2
end
```

<実行結果グラフ>



青色表示がキャリブレーション前波形、赤色表示がテーブル参照したキャリブレーション後を示します。

19. 3. サンプリング周波数を変更する

19. 3. 1. サンプリング周期の倍数でダウンサンプリングを行う

サンプリング周期の n 倍でのダウンサンプリングは、ダウンサンプリング周波数/2 の遮断周波数でローパスフィルタを掛けた後、SEP 関数を使用して飛越抽出を行います。

記述例:

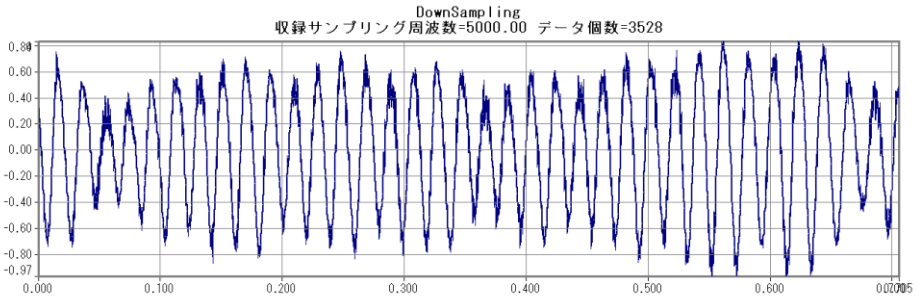
収録データカレントチャネルのサンプリング周波数を 1/4(n=4)にする場合

```
/*-------- 1/4 ダウンサンプリング----------------------------*/
$1 "カレントチャネル番号:" = CCH()
$6 "ダウンサンプリング前時刻歴:" = SPB(0)
$8 "収録サンプリング周波数:" = 1/PRD()
/*----- カレントチャネル ダウンサンプリング処理----------------*/
$2 "ダウン倍率 n:" = 4
$3 = PRD()*$2                          /* ダウンサンプリング周期*/
$4 = LPF(1/$3/2,LPF(1/$3/2,#($1))) /* ローパスフィルタ処理*/
$4 = $4/0.9977                         /* ゲイン調整*/
$5 = SEP(0,$2-1,$4)                  /* ダウンサンプリング*/
$7 "ダウンサンプリング後時刻歴:" = (ACC(DAG(LEN($5),1))-1)*$3
$9 "ダウンサンプリング周波数:" = 1/$3
/*----- ダウンサンプリング前波形グラフ描画処理----------------*/
$10 "データ個数:" = LEN(#($1))
assign &1 = "収録サンプリング周波数="|$8(F2)|" データ個数="|$10(F0)
def graph_id @1 "DownSampling" &1
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $6 #($1)
def file_id %1 "graph" grp
save plot %1 @1 $6 #($1)
/*----- ダウンサンプリング後波形グラフ描画処理----------------*/
$10 "データ個数:" = LEN($5)
assign &1 = "ダウンサンプリング周波数="|$9(F2)|" データ個数="|$10(F0)
def graph_id @1 "DownSampling" &1
plot @1 $7 $5
def file_id %1 "graph1" grp
save plot %1 @1 $7 $5
end
```

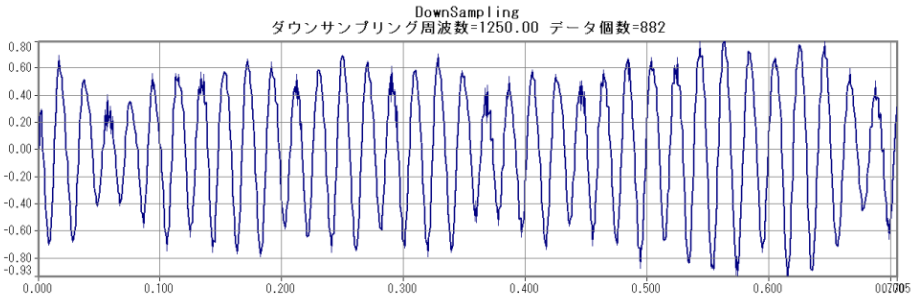
ダウン倍率 2 は上記記述例ではエラーとなります。この事は丁度、遮断周波数が 1/4 となる為です。
その場合、遮断周波数を 1/4 以下となる様に少し下げてフィルタ処理します。(下記例では 1/4.1)

修正例:`$4 = LPF(1/$3/2.05,LPF(1/$3/2.05,#($1)))`

<ダウンサンプリング前の波形>



<ダウンサンプリング後の波形>



ダウンサンプリングレートが大きくなるに従って、元々持っていた周波数成分が高域から徐々に失われて行きます。

19. 3. 2. サンプリング周期の倍数でアップサンプリングを行う

サンプリング周期の 1/n 倍でのアップサンプリングは、元のデータに 0 を内挿してデータ数をアップサンプリング周波数でサンプリングした時と同じ個数にした後、アップサンプリング周波数/2 の遮断周波数でローパスフィルタを掛けスムージングします。フィルタ通過後はゼロ内挿個数とフィルタ平坦部通過減衰係数による振幅修正が必要となります。 ゼロ内挿は数列にゼロ内挿する関数【INS 関数】を使用します。

文法:

結果格納先 = INS(ゼロ個数,対象数列)

引数:

【ゼロ個数】＜必須＞

データとデータの間に挿入するゼロ個数を記述します。例えば 4 倍にする場合は 3 個となります。

【対象数列】＜必須＞ ゼロ内挿する対象数

列を記述します。

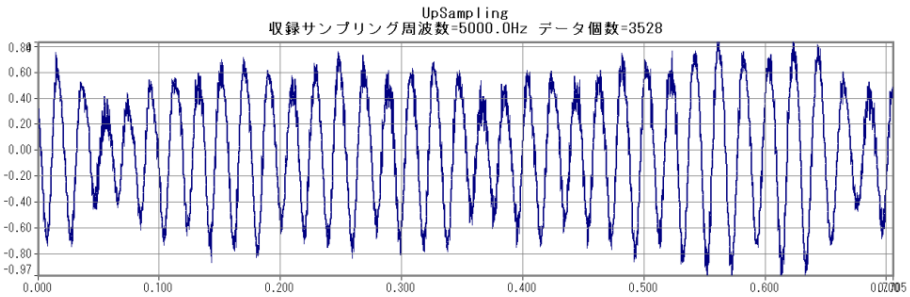
記述例:

収録データカレントチャネルのサンプリング周波数を 4 倍(n=1/4)にする場合

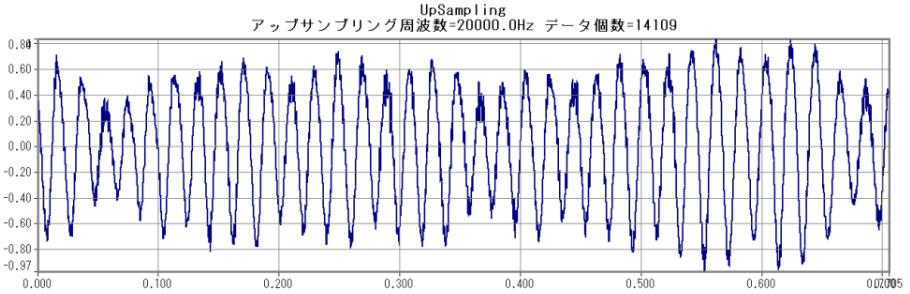
```
/*-------- ×4 アップサンプリング------------------------------*/
$1 "カレントチャネル番号:" = CCH()
$6 "アップサンプリング前時刻歴:" = SPB(0)
$8 "収録サンプリング周波数:" = 1/PRD()
/*----- カレントチャネル アップサンプリング処理---------------*/
$2 "アップレート n:" = 4
$3 = PRD()/$2                    /* アップサンプリング周期*/
$4 = INS($2-1,#($1))             /* データ間 Zero 挿入*/
$5 "遮断周波数:Hz" = $8/2/$2     /* 遮断周波数換算*/
$4 = LPF($5,LPF($5,$4))          /* ローパスフィルタ処理*/
$4 = $4*$2/0.9977                /* ゲイン調整*/
$7 "アップサンプリング後時刻歴:" = (ACC(DAG(LEN($4),1))-1)*$3
$9 "アップサンプリング周波数:" = 1/$3
/*----- アップサンプリング前波形グラフ描画処理---------------*/
$10 "データ個数:" = LEN(#($1))
assign &1 = "収録サンプリング周波数="|$8(F1)|"Hz データ個数="|$10(F0)
def graph_id @1 "UpSampling" &1
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $6 #($1)
def file_id %1 "graph" grp
save plot %1 @1 $6 #($1)
/*----- アップサンプリング後波形グラフ描画処理---------------*/
```

```
$10 "データ個数:" = LEN($4)
assign &1 = "アップサンプリング周波数="|$9(F1)|"Hz データ個数="|$10(F0)
def graph_id @1 "UpSampling" &1
plot @1 $7 $4
def file_id %1 "graph1" grp
save plot %1 @1 $7 $4
end
```

<アップサンプリング前の波形>



<アップサンプリング後の波形>



アップサンプリング後の波形はアップサンプリング前の波形と同じとなります。

19. 3. 3. サンプリング定理から任意のサンプリング周期に変更する サンプリング定理からサンプリング周波数を変更する場合は**【RSM 関数】**を使用します。 **文法:**

結果格納先 = RSM(リサンプリング周波数,対象数列)

引数:

【リサンプリング周波数】＜必須＞ 変更後のサンプリング周波数を記述します。

【対象数列】

サンプリングを変更する対象数列を記述します。

k をリサンプリング周波数、PRD()を現在のサンプリング周期、m をリサンプリング結果数列 index、n を対象数列の index とすると演算内容は次式に寄ります。

$$\Delta t_0 = PRD(\,)$$

$$\Delta t_1 = \frac{1}{k}$$

$$X_m = \sum_{n=0}^{N-1} \left\{ X_n \frac{\sin\left(\frac{\pi}{\Delta t_0}\left(m \cdot \Delta t_1 - n\Delta t_0\right)\right)}{\frac{\pi}{\Delta t_0}\left(m \cdot \Delta t_1 - n \cdot \Delta t_0\right)} \right\}$$

演算式から判る様に、リサンプリング後のデータ 1 点を求める為にごと回対象数列の全てのデータから演算する為、対象数列のデータ数及びリサンプリングのデータ数により演算時間が掛ります。又、現在のサンプリング周波数とリサンプリング周波数との比等により、生成される数列の先端部及び終端部に波形ひずみを生じる場合があります。

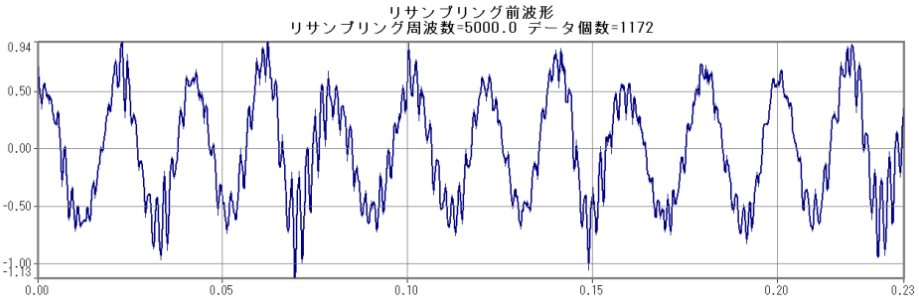
記述例:

サンプリング周波数 5kHz の収録データカレントチャネルのサンプリングを 5.12kHz にリサンプリングする場合

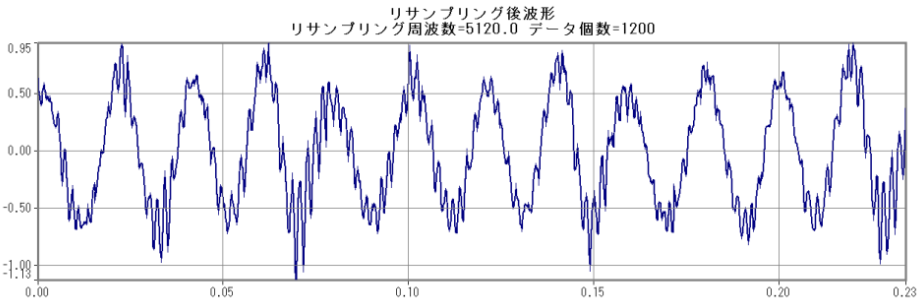
```
/*-------- リサンプリング処理----------*/
$1 = CCH()                  /*カレントチャネル番号取得*/
```

```
$2 "リンサンプリング周波数:Hz" = 5120
$3 = RSM($2,#($1))      /* リサンプリング演算*/
/*--- リサンプリング後波形グラフ描画処理--------*/
$4 = (ACC(DAG(LEN($3),1))-1)/$2 /*グラフ X 軸時刻歴生成*/
$5 = LEN($3)            /* リサンプリング後データ個数取得*/
assign &1 = "リサンプリング周波数="|$2(F1)|" データ個数="|$5(F0)
def graph_id @1 "リサンプリング後波形" &1
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $4 $3
def file_id %1 "graph" grp
save plot %1 @1 $4 $3
/*---リサンプリング前波形グラフ描画処理--------*/
$5 = LEN(#($1))         /*現在データ個数取得*/
$6 "現在サンプリング周波数:Hz" = 1/PRD() /*現在サンプリング周波数取得*/
$7 = SPB(0)             /*グラフ X 軸時刻歴生成*/
assign &1 = "リサンプリング周波数="|$6(F1)|" データ個数="|$5(F0)
def graph_id @1 "リサンプリング前波形" &1
plot @1 $7 #($1)
def file_id %1 "graph1" grp
save plot %1 @1 $7 #($1)
end
```

<リサンプリング前の波形 5kHz サンプリング>



<リサンプリング後の波形 5.12kHz サンプリング>



記述例:

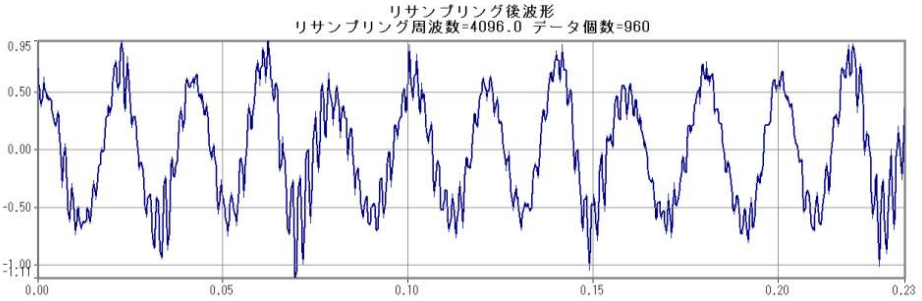
サンプリング周波数 5kHz の収録データカレントチャネルのサンプリングを 4096Hz にリサンプリングする場合

```
/*-------- リサンプリング処理----------*/
$1 = CCH()              /*カレントチャネル番号取得*/
$2 "リンサンプリング周波数:Hz" = 4096
$3 = RSM($2,#($1))      /* リサンプリング演算*/
/*--- リサンプリング後波形グラフ描画処理--------*/
$4 = (ACC(DAG(LEN($3),1))-1)/$2 /*グラフ X 軸時刻歴生成*/
$5 = LEN($3)            /* リサンプリング後データ個数取得*/
assign &1 = "リサンプリング周波数="|$2(F1)|" データ個数="|$5(F0)
def graph_id @1 "リサンプリング後波形" &1
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $4 $3
def file_id %1 "graph" grp
save plot %1 @1 $4 $3
```

```
/*---リサンプリング前波形グラフ描画処理--------*/
$5 = LEN(#($1))          /*現在データ個数取得*/
$6 "現在サンプリング周波数:Hz" = 1/PRD() /*現在サンプリング周波数取得*/
$7 = SPB(0)              /*グラフ X 軸時刻歴生成*/
assign &1 = "リサンプリング周波数="|$6(F1)|" データ個数="|$5(F0)
def graph_id @1 "リサンプリング前波形" &1
plot @1 $7 #($1)
def file_id %1 "graph1" grp
save plot %1 @1 $7 #($1)
end
```

前述の Script 記述例とリサンプリング周波数設定行のみ異なり、他は同じです。

<リサンプリング後の波形 4096Hz サンプリング>



19. 3. 4. 最小公倍数でアップサンプリングしてからダウンサンプリングし任意のサンプリング周期に変更する

最小公倍数を求めてアップサンプリングダウンサンプリングしてサンプリング周波数を変更する場合 初めに、現在のサンプリング周波数とリサンプリング周波数との最小公倍数を求めます。 最小公倍数を求める場合【LCM 関数】を使用します。

文法:

結果格納先 = LCM(有効桁数,対象数値 1,対象数値 2)

引数:

【有効桁数】＜必須＞

対象数値 1 及び 2 の有効桁数を小数点以下桁で記述します。例えば 0 とした場合、対象数値 1 及び 2 とも小数点以下を切り捨てて参照します。

【対象数値】＜必須＞ 最小公倍数を求める現在のサンプリング周波数とリサンプリング周波数を半角カンマで区切って記述します。なお、記
述順序は規定されません。

記述例:

サンプリング周波数 1000Hz、リサンプリング周波数 1024Hz の最小公倍数を求める場合

```
$1 = 1000
$2 = 1024
$3 = LCM(0,$1,$2)
```

$3 には最小公倍数 128000 が格納されます。

リサンプリングの演算手順:

① 最小公倍数 = LCM(0,サンプリング周波数,リサンプリング周波数)
② アップ倍率 = 最小公倍数/サンプリング周波数
③ ダウン倍率 = 最小公倍数/リサンプリング周波数
④ ゼロ内挿数 = アップ倍率-1
⑤ 拡張波形 = INS(ゼロ内挿個数,元の波形データ)

サンプリング周波数＜リサンプリング周波数の場合

⑥ 換算遮断周波数 = サンプリング周波数/2/アップ倍率

サンプリング周波数＞リンサンプリング周波数の場合

⑥ 換算遮断周波数 = リサンプリング周波数/2/ダウン倍率

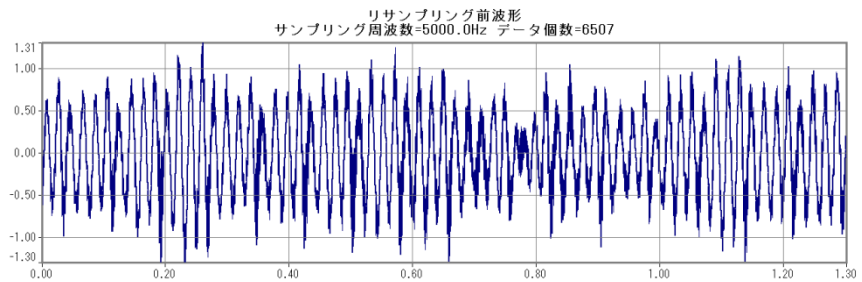
⑦ フィルタ後拡張波形 = LPF(換算遮断周波数,LPF(換算遮断周波数,拡張波形))
※ ローパスフィルタ処理を 2 回掛けて遮断特性を稼ぎます。
⑧ リサンプリング波形 = SEP(0,ダウン倍率-1,フィルタ後拡張波形)
⑨ 仕上がりリサンプリング波形 = リサンプリング波形×アップ倍率/0.9977
※ 0.9977 は 4 次バターワースフィルタ 2 回処理時の平坦部減衰係数です。

記述例:

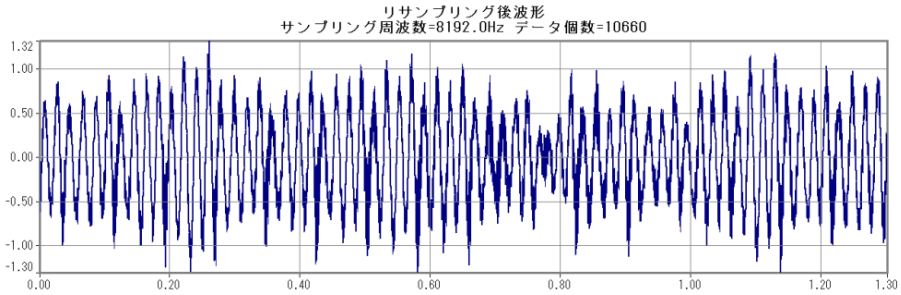
収録データカレントチャネルを設定した周波数でリサンプリングする場合

```
/*------- リサンプリング  Script-----------------------------------------*/
$1 = CCH()                                          /*カレントチャネル番号取得*/
$2 "リンサンプリング周波数:Hz" = 5120
$3 "現在のサンプリング周波数:Hz" = 1/PRD()
get value $2(F1) "リサンプリング周波数を設定して下さい"
case $2 <> $3
  proc re_samplig{
    /*------- 最小公倍数を求める-------------------------------------------*/
    $4 "LCM:" = LCM(0,$2,$3)                        /* 最小公倍数取得*/
    /*-----アップ倍率,ダウン倍率を求める---------------------------------*/
    $6 "UpRate:" = $4/$3                            /*アップ倍率演算*/
    $9 "DownRate:" = $4/$2                          /*ダウン倍率演算*/
    /*-----アップサンプリングする---------------------------------*/
    $7 "アップサンプル波形:" = INS($6-1,#($1))     /*対象波形をアップサンプリング*/
    /*-----ローパスフィルタ換算遮断周波数演算＆フィルタ処理-------------*/
    case $3 < $2          /*サンプリング周波数 ＜リサンプリング周波数*/
      proc freq_up{
        $8 "換算遮断周波数:Hz" = $3/2.05/$6
        $7 = LPF($8,LPF($8,$7))                     /*ローパスフィルタ処理*/
      }freq_up
    case $3 > $2          /*サンプリング周波数 ＞ リサンプリング周波数*/
      proc freq_down{
        $8 "換算遮断周波数:Hz" = $2/2.05/$9
        $7 = LPF($8,LPF($8,$7))                     /*ローパスフィルタ処理*/
      }freq_down
    /*---- ダウンサンプリングし振幅を元に戻す---------------------------*/
    $10 "ダウンサンプル波形:" = SEP(0,$9-1,$7)  /*ダウンサンプル波形抽出*/
    $10 = $10*$6/0.9977                                 /*振幅校正*/
    /*----- 元波形グラフ描画＆格納処理-----------------------------------*/
    $11 "グラフ X 軸データ:" = SPB(0)                /*グラフ X 軸時刻歴生成*/
    $12 "元データ個数:" = LEN(#($1))                /*リサンプリング前データ個数取得*/
    assign &1 = "サンプリング周波数="|$3(F1)|"Hz データ個数="|$12(F0)
    def graph_id @1 "リサンプリング前波形" &1
    def graph_x_axis @1 0,0 F2
    def graph_y_axis @1 0,0 F2 5
    def graph_aspect_ratio @1 3
    plot @1 $11 #($1)
    def file_id %2 "graph" grp
    save plot %2 @1 $11 #($1)
    /*---- リサンプリング波形グラフ描画＆格納処理-----------------------*/
    $11 = (ACC(DAG(LEN($10),1))-1)/$2      /*グラフ X 軸時刻歴生成*/
    $12 = LEN($10)                          /*リサンプル後データ個数取得*/
    assign &1 = "サンプリング周波数="|$2(F1)|"Hz データ個数="|$12(F0)
    def graph_id @1 "リサンプリング後波形" &1
    def graph_x_axis @1 0,0 F2
    def graph_y_axis @1 0,0 F2 5
    def graph_aspect_ratio @1 3
    plot @1 $11 $10
    def file_id %2 "graph1" grp
    save plot %2 @1 $11 $10
  }re_samplig
end
```

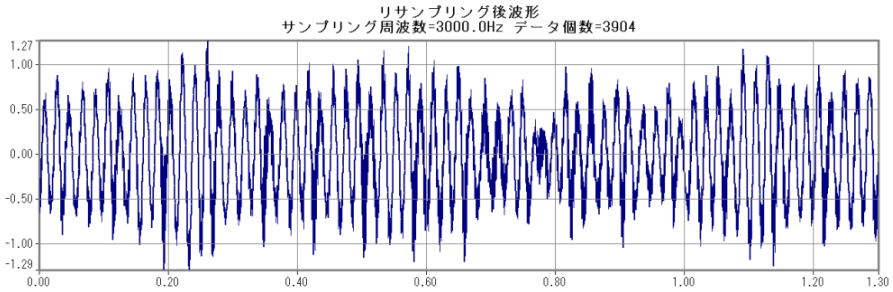
<リサンプリング前の波形 5000Hz サンプリング>



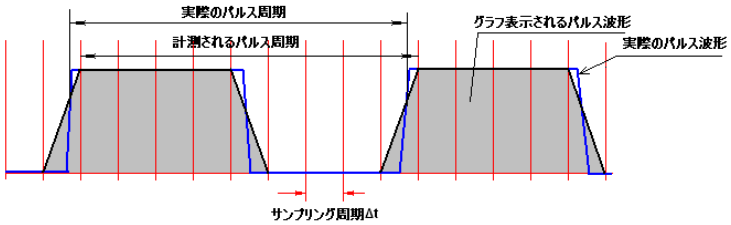
<リサンプリング後の波形① 8192Hz サンプリング>



<リサンプリング後の波形② 3000Hz サンプリング>



19. 4. 収録データをパルス列信号として処理する パルス列信号、波形信号等で入力されたデータを論理波形に変換し、パルスの個数、回転数、或いは回転角度などを求める場合、 入力されたパルス列の周期或いはパルス幅が意味を持つ事が多く、その精度は収録サンプリング周波数に依存します。 従って、**サンプリング周期<<パルス周期**で無い場合、誤差(ジッタ)が発生します。



19. 4. 1. 収録信号から論理パルス数列を生成する

エンコーダ出力信号、接点信号或いはその他の波形信号などを二値化して論理パルス数列に変換して処理する場合は【RWC 関数】を使用します。なお、エンコーダ出力信号など元々パルス列の属性を持ち、そのまま処理することもできますが電圧値が印加電圧に従属していますので、その後、論理演算等を行う為には、同じく論理パルス数列に変換してから行う必要があります。

文法:

結果格納先 = RWC(論理"1"閾値,論理"0"閾値,対象数列) 引数:

【論理"1"閾値】＜必須＞

論理"1"の閾値を記述します。論理"1"は、論理"1"閾値を上昇で越えて、論理"0"閾値を下降で過るまでを論 理"1"とします。

【論理"0"閾値】＜必須＞

論理"0"の閾値を記述します。論理"0"は、論理"0"閾値を下降で越えて、論理"1"閾値を上昇で過るまでを論 理"1"とします。

【対象数列】＜必須＞ 論理化する対象数

列を記述します。

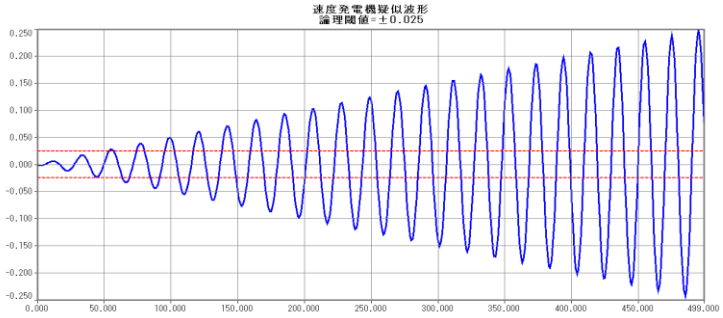
※ RWC 関数は、先頭から最初の何れかの閾値を通過しないと論理が決定されません。従って、その間は対象数列の先頭値の正負で論理が決定され、正数の場合は論理"1"区間と見なし負数の場合は"0"区間と見なし論理パルスを生成される事に注意下さい。

記述例:

収録データの ch1 を速度発電機信号として見なし論理化する場合 速度発電機信号は回転数が下がると振幅が減少し周期も長くなり、回転数が上がると振幅が大きくなり周期も短くな ります。周期はゼロを上昇で過ってから再び上昇で過る迄のデータ個数にサンプリング周期を掛けて周期に変換しま す。従って、0 を過る近傍はノイズにより正確に論理化できない場合がありますので、最低速度振幅を決定し、閾値と して設定します。記述例では±0.025 を閾値しています。

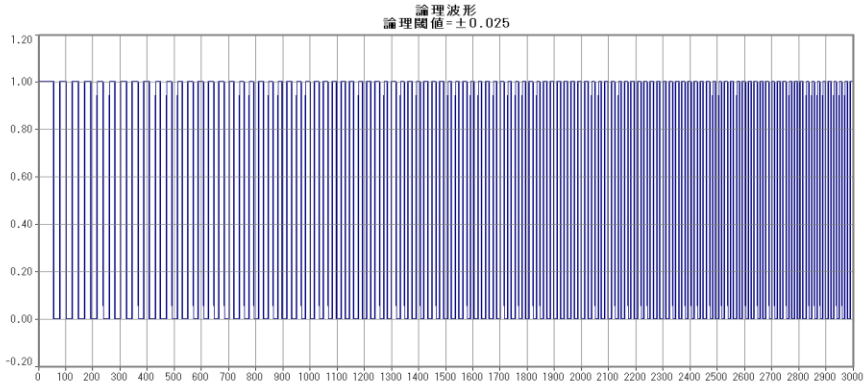
```
/* --速度発電機信号論理化処理----------------------------------------*/
$1 "論理化波形:" = RWC(0.025,-0.025,#1) /*試験波形論理値変換*/
/* ----論理波形グラフ描画処理----------------------------------------*/
def graph_id @1 "論理波形" "論理閾値=±0.025" /*グラフ番号定義*/
def graph_x_axis @1 0,0,100 F0        /*グラフ X 軸定義*/
def graph_y_axis @1 -0.2,1.2 F2 5     /*グラフ Y 軸定義*/
def graph_aspect_ratio @1 2           /*グラフ縦横比定義*/
plot @1 $1                            /*実行結果グラフ描画*/
end
```

<速度発電機疑似波形>



※ 疑義波形グラフは、説明の都合上、先頭から 500 点の範囲をグラフ化しています。

<実行結果:生成パルスグラフ>

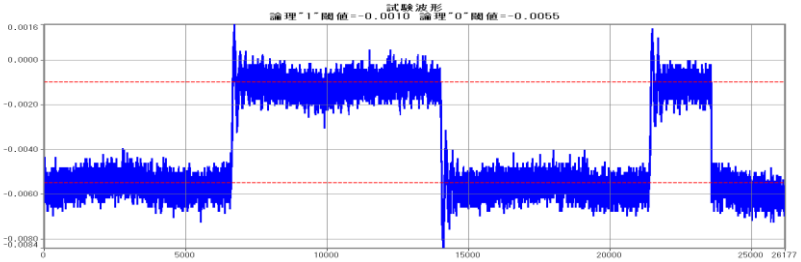


記述例:

汚れたパルス波形から論理パルス数列を求める場合 設定する論理"1"閾値は、この値を上昇で過らないと"1"に遷移させない値、同じく論理"0"閾値はこの値を下降で 過らないと"0"に遷移させない値を、変換対象波形を観察して設定します。

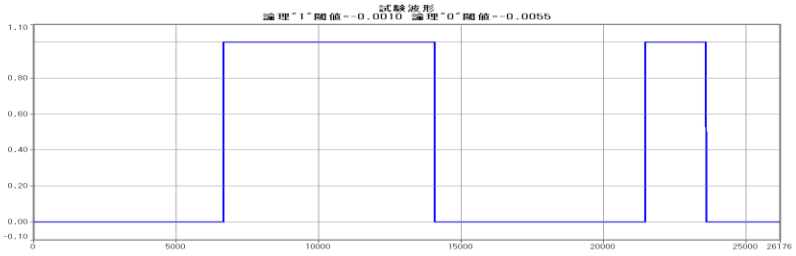
```
/* --汚れた波形の論理化処理----------------------------------*/
$1 "論理 1 閾値:" = -0.001
$2 "論理 0 閾値:" = -0.0055
$3 "論理化波形:" = RWC($1,$2,#5) /*試験波形論理変換*/
/* --- 試験波形グラフ描画処理-----------------------------*/
$7 = LNK(0,LEN(#5-1))      /*閾値線 X 軸値生成*/
$8 = LNK($1,$1)            /*論理"1"閾値線 Y 軸値生成*/
 $9 = LNK($2,$2)           /*論理"0"閾値線 Y 軸値生成*/
$11 = LNK(2,21,21)         /*グラフ描画線種生成*/
$12 = ACC(DAG(LEN(#5),1))-1 /*グラフ X 軸数列生成*/
assign &1 = "#FF0000","#0000FF","#0000FF"
assign &2 =  "論理"1"閾値="|$1(F4)|" 論理"0"閾値="|$2(F4)
def graph_id @1 "試験波形" &2 /*グラフ番号定義*/
def graph_x_axis @1 0,0 F0     /*グラフ X 軸定義*/
def graph_y_axis @1 0,0 F4 5  /*グラフ Y 軸定義*/
def graph_aspect_ratio @1 2    /*グラフ縦横比定義*/
def graph_line @1 $11 &1       /*グラフ線種線色定義*/
plot @1 $12,$7,$7 #5,$8,$9     /*試験波形グラフ描画*/
def graph_y_axis @1 -0.1,1.1 F2 /*グラフ Y 軸再定義*/
plot @1 $12 $3                 /*実行結果グラフ描画*/
end
```

<試験波形>



※ 記述例では論理"1"閾値を-0.001、論理"0"閾値を-0.0055 としています。

<実行結果:生成パルスグラフ>



19. 4. 2. パルス整形/飛越操作して論理パルス生成する パルス列に不要なパルスが混在している場合の除去を行う場合や、パルスデューティ比を 50%に整合する場合【TPC 関数】を使用します。

文法:

結果格納先 = TPC(除去パルス数,対象数列)

引数:

【除去パルス数】＜必須＞

除去パルス数を記述します。除去パルス数を 0 とした場合は、パルス除去を行わずパルスデューティ比を 50%に変換します。

【対象数列】＜必須＞

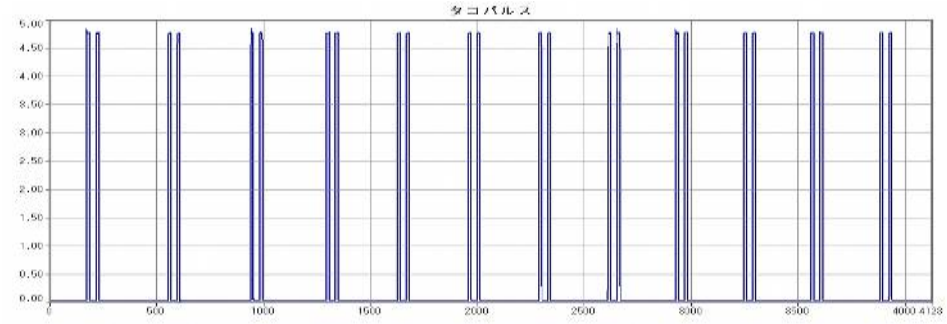
解析対象数列を記述します。

記述例:

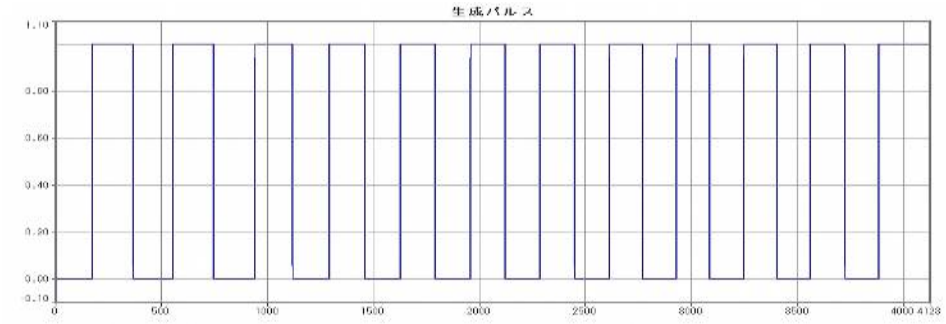
収録したタコパルスデータ(1 回転当たり近接して 2 パルス存在)から、不要パルスを除去し、デューティ比を 50%に整形した論理パルス数列を生成する場合

```
/*-- 1 回転 1 パルスに整合処理----------*/
$1 = TPC(1,#9)
/*----グラフ描画処理----------*/
def graph_id @1 "タコパルス"
def graph_x_axis @1 0,0 F1
def graph_y_axis @1 0,5 F2 5
def graph_aspect_ratio @1 2
plot @1 #9
def file_id %1 "graph" grp
save plot %1 @1 #9
def graph_id @1 "生成パルス"
def graph_y_axis @1 -0.1,1.1 F2 5
plot @1 $1
def file_id %1 "graph1" grp
save plot %1 @1 $1
end
```

<変換対象タコパルス波形>



<実行結果:生成パルスグラフ>



記述例:

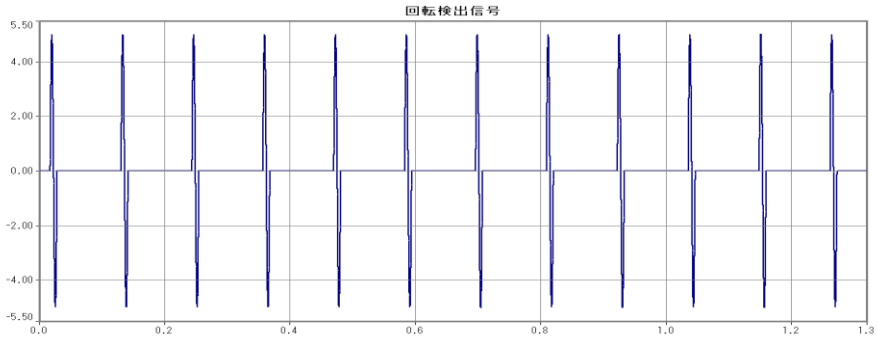
回転検出信号(微分波形)から整形した回転パルスを生成する場合

比較関数(GTE 関数)を使用し、閾値を 1V として論理波形変換した後、デューティ比を 50%に整形した論理パルス数列を求めます。

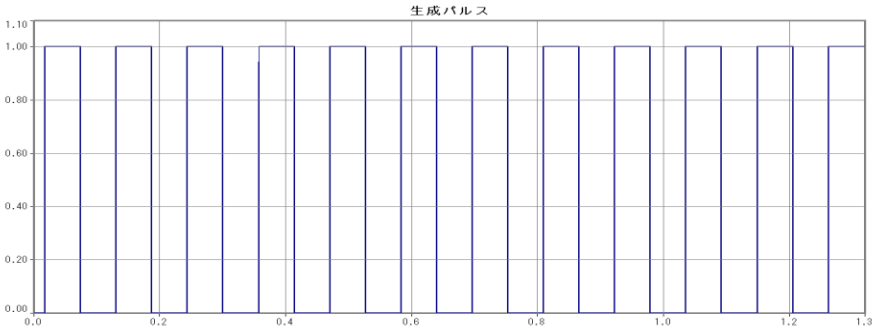
```
/*-- 微分回転信号を回転パルスに整合処理----------*/
```

```
$1 = TPC(0,GTE(#1,1))    /*回転検出信号パルス変換*/
/*----グラフ描画処理----------------------------------*/
$2 = SPB(0)
def graph_id @1 "回転検出信号"
def graph_x_axis @1 0,0 F1
def graph_y_axis @1 -5.5,5.5 F2 5
def graph_aspect_ratio @1 2
plot @1 $2 #1
def file_id %1 "graph" grp
save plot %1 @1 $2 #1
def graph_id @1 "生成パルス"
def graph_y_axis @1 0,1.1 F2 5
plot @1 $2 $1
def file_id %1 "graph1" grp
save plot %1 @1 $2 $1
end
```

<変換対象回転検出信号波形>



<実行結果:生成パルスグラフ>



19. 4. 3. パルスの累積個数を数える 速発パルスから走行距離、回転パルスから累積回転回数などは、パルスの累積個数を求め、結果にパルス間隔値(累積 回転回数の場合は 1/1 回転当たりパルス数)を掛算して求めます。パルス個数を数える場合 **【CNT 関数】**を使用します。 **文法:**

結果格納先 = CNT(対象数列)

引数:

【対象数列】＜必須＞

パルス数列を記述します。パルス振幅は 0.5 以上が必要です。なお、デューティ比は特に拘りません。

CNT 関数は、閾値 0.5 を上昇で過る数を累積しますので、特に波形整形は不要です。

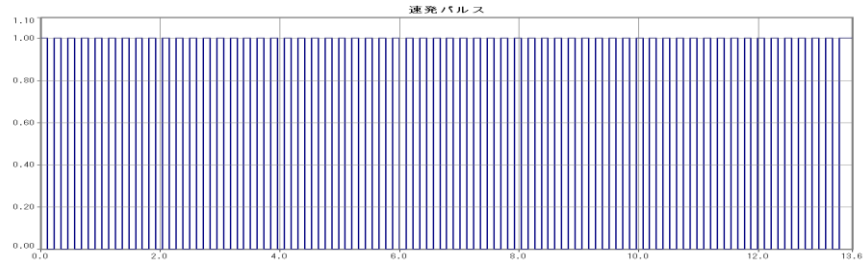
記述例:

速発パルス(間隔 2.0m)を積算して走行距離を求める場合 記述例では説明の都合上、デューティ比を 50%に整合していますが、処理には関係ありません。

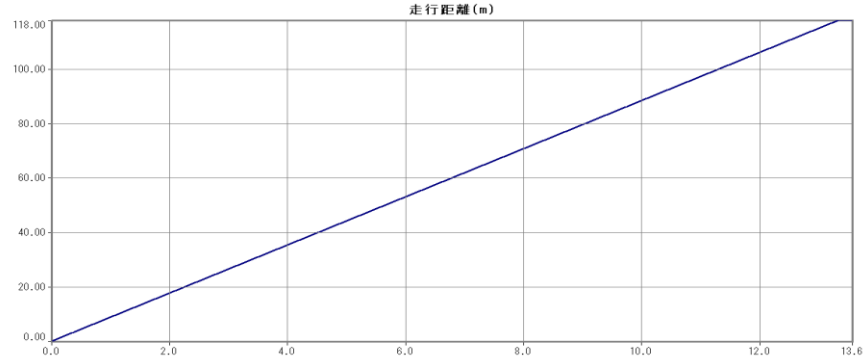
```
/*-- 速発パルスから走行距離を求める----------------------*/
$3 = TPC(0,#1)      /*  速発パルスデューティ比50%変換*.
$1 = CNT($3)*2      /*速発パルスから走行距離に変換*/
/*----グラフ描画処理----------------------------------*/
$2 "時刻歴:sec" = SPB(0)
def graph_id @1 "速発パルス"
```

```
def graph_x_axis @1 0,0 F1
def graph_y_axis @1 0,1.1 F2 5
def graph_aspect_ratio @1 2
plot @1 $2 $3
def file_id %1 "graph" grp
save plot %1 @1 $2 $3
def graph_y_axis @1 0,0 F2 5
plot @1 $2 $1
def file_id %1 "graph1" grp
save plot %1 @1 $2 $1
end
```

<速発パルス波形グラフ>



<実行結果:走行距離グラフ>



※ X 軸単位は秒(sec)、Y 軸単位はメートル(m)

19. 4. 4. 2 相エンコーダパルス列から回転角度を求める

基本的な考え方はパルス個数を累算する 19.4.3 項と同じですが、論理パルス列の差分値を累算する点が異なります。
2 相パルスエンコーダは A 相と B 相が 90° 位相差を持ってパルス出力され回転方向により AB 相間の位相差の進み遅れとなり、A 相パルスと B 相パルスを論理演算する事により回転角度を求める事ができます。

回転角度を求める演算手順:

① A 相パルス列、B 相パルス列を論理値数列に変換します。以降の演算は論理値での扱いが必須となる為です。
論理値数列への変換はエンコーダ信号の為、特に RWC 関数を使用せず簡単に比較関数を使用して振幅最大値の 1/2 を閾値として GTE 関数により論理値数列に変換します。

A 相論理パルス列 = GTE(A 相パルス列,(MAX(A 相パルス列)-MIN(A 相パルス列))/2)
B 相論理パルス列 = GTE(B 相パルス列,(MAX(B 相パルス列)-MIN(B 相パルス列))/2)

② A/B 相の論理パルス数列から DIF 関数を使用し差分を求めます。

A 相差分パルス列 = DIF(A 相論理パルス列)
B 相差分パルス列 = DIF(B 相論理パルス列)

※ 演算結果はそれぞれ、パルスの立ち上がりを 1、立下りを-1 としたデータとなります。

③ ①と②の結果を使い A/B の角度パルス数列を求めます。

A 相角度パルス列 = A 相差分パルス列*(B 相論理パルス列-0.5)*2
B 相角度パルス列 = SGN(B 相差分パルス列*(A 相論理パルス列-0.5)*2

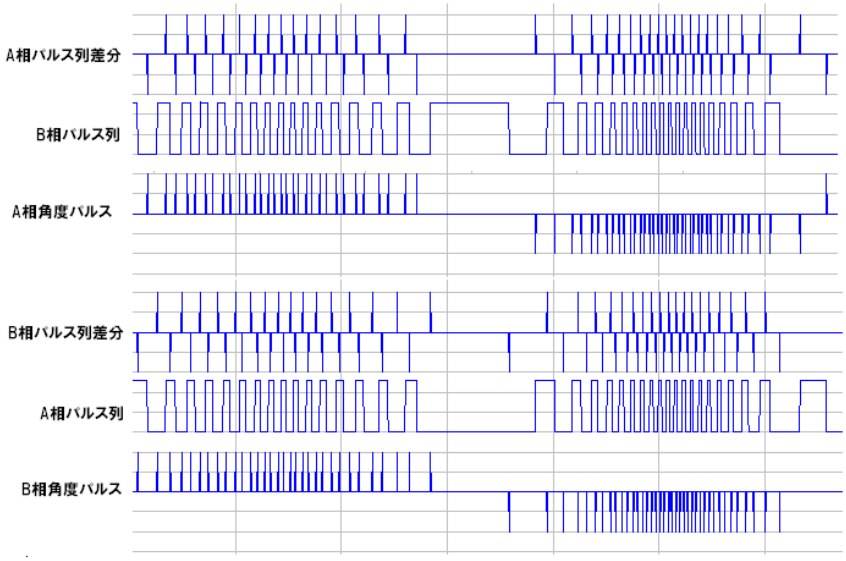
※ 演算結果は 2 逓倍された正回転方向が+1、逆回転方向が-1 となるパルス列が求まります。なお、SGN()は符号反転を行う関数です。

④ ③の結果を加算し ACC 関数により累積します。累積した結果に 1 歯当たりの角度を掛けて回転角度を求めます。

回転角度 = ACC(A 相角度パルス列+B 角度パルス列)*360/(歯数*4)

※ 1 歯当たりの角度は 360° を歯数の 4 倍で割り求めます。4 倍の理由は A/B 相角度パルスを加算する事により総合的に 4 逓倍される為です。

下図に角度パルス列演算過程を図示します。



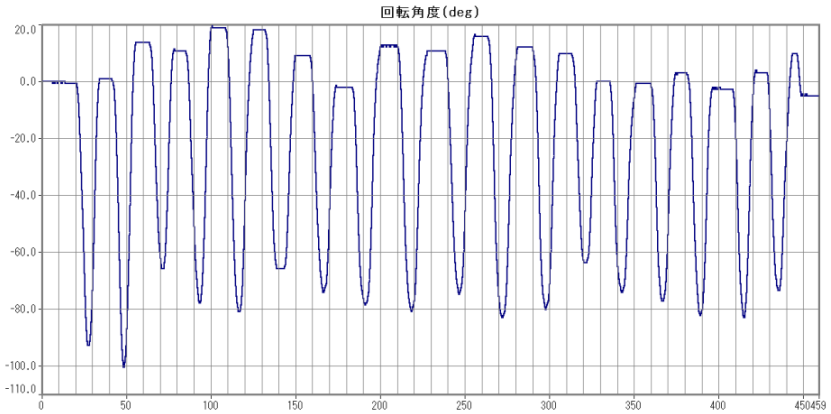
記述例:

収録データ 1ch:A 相パルス、2ch:B 相パルスの 2 相エンコーダ出力から回転角度を求める場合

```
/*-------- 2 相エンコーダ回転角度----------------*/
$8 "歯数:" = 120
proc   回転角度演算{
  def local_ch $1,$2,$3,$4,$5,$6
  $1 "A 相論理パルス:" = GTE(#1,(MAX(#1)-MIN(#1))/2)
  $2 "B 相論理パルス:" = GTE(#2,(MAX(#2)-MIN(#2))/2)
  $3 "A 相差分パルス:" = DIF($1)
  $4 "B 相差分パルス:" = DIF($2)
  $5 "A 相角度パルス:" = $3*($2-0.5)*2
  $6 "B 相角度パルス:" = SGN($4*($1-0.5)*2)
  $7 "回転角度:deg" = ACC($5+$6)*360/($8*4)
}回転角度演算
/*-------回転角度グラフ描画処理--------------------*/
$9 "経過時間:sec" = (ACC(DAG(LEN(#1),1))-1)*PRD()
def graph_id @1 "回転角度(deg)"
def graph_x_axis @1 0,00 F0
def graph_y_axis @1 0,0 F1 5
def graph_aspect_ratio @1 2
def file_id %1 "回転角度" grp
save plot %1 @1 $9 $7
end
```

※ 演算処理ブロックとして記述していますが、演算処理ブロックとして記述する必要はありません。演算処理ブロックを記述しブロック内にローカルチャネルを定義した場合、定義されたローカルチャネル番号は演算処理ブロック外では参照できません。言い換えれば、演算処理ブロック外に同じチャネル番号が記述されていても無関係となります。

<実行結果:回転角度グラフ>



19. 4. 5. パルス周期を求める

パルス周期を求める場合、パルス波形の立ち上がり地点を検索し、立ち上がり地点間のデータ個数を求めサンプリング周期を掛算します。但し、その結果の個数はパルス間隔数と一致しますのでデータ数が減少します。従って元のデータ個数と同じにする為には演算結果の周期数列からパルス数列と同じデータ個数とする処理が必要です。

パルス周期を求める演算手順:

① パルス数列を"1"と"0"からなるパルス論理数列に変換します

② 論理数列の立ち上がり地点を検索します
遷移地点数列 = LST(1,パルス論理数列)

③ 遷移地点数列を分解します
遷移地点先方数列 = ERC(0,LEN(遷移地点数列)-2,遷移地点数列)
遷移地点後方数列 = ERC(1,LEN(遷移地点数列)-1,遷移地点数列)

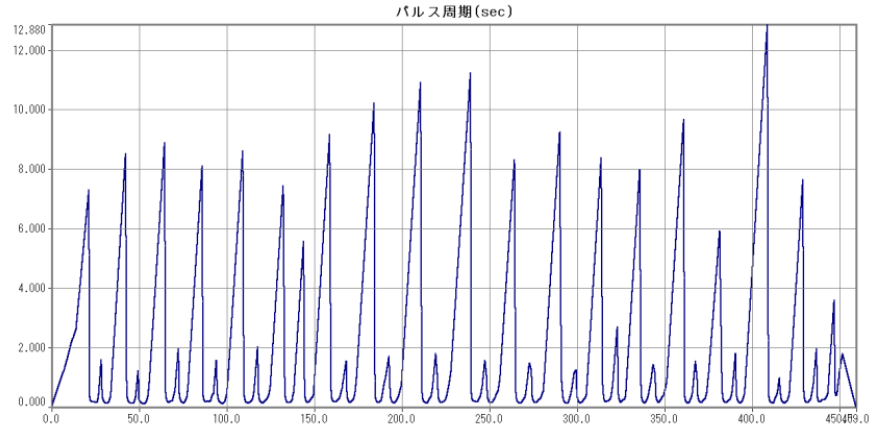
④ 遷移地点後方数列から前方数列を引き、結果にサンプリング周期を掛け周期を求めます
周期数列 = (遷移地点後方数列-遷移地点先方数列)*PRD()

⑤ パルス数列と同じ長さの数列に、遷移後方地点に周期数列を埋め込み直線補間します
周期数列 = VRP(1,LEN(パルス論理数列),遷移地点後方数列,周期数列)

記述例:前述 2 相パルスから A 相パルスの周期を求める場合

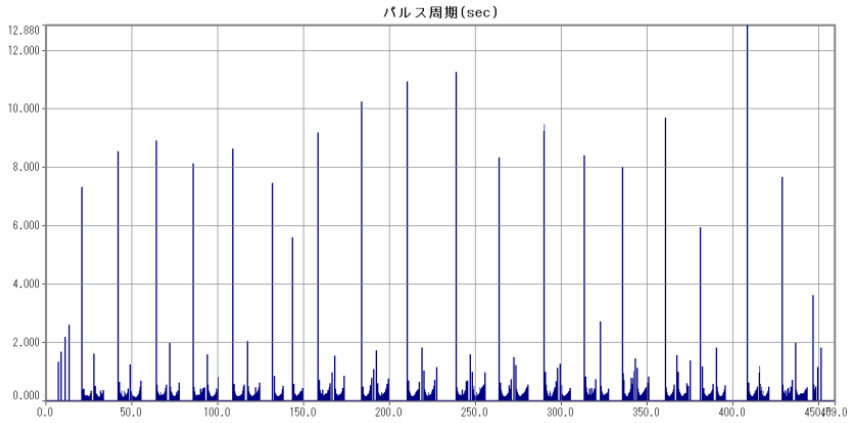
```
/*-----パルス周期演算処理----------------------------------*/
$1 "A 相論理パルス:" = GTE(#1,(MAX(#1)-MIN(#1))/2) /*論理数列変換*/
$2 "遷移地点:" = LST(1,$1)                  /*立ち上がり地点 index 検索*/
$3 "遷移先方:" = ERC(0,LEN($2)-2,$2)
$4 "遷移後方:" = ERC(1,LEN($2)-1,$2)
$5 "周期:" = ($4-$3)*PRD()
$6 "パルス周期:" = VRP(1,LEN($1),$4,$5)  /*パルス周期生成(直線補間)*/
/*------ グラフ描画&格納処理----------------------------------*/
$7 = SPB(0)
def graph_id @1 "パルス周期(sec)"
def graph_x_axis @1 0,0 F1
def graph_y_axis @1 0,0 F3 5
def graph_aspect_ratio @1 2
plot @1 $7 $6
def file_id %1 "graph" grp
save plot %1 @1 $7 $6
end
```

<実行結果:回転角パルス周期グラフ>



※ 使用したデータは回転角度パルスですので、回転を停止している状態では次のパルスが来るまで長い周期を示します。その結 果を直線補間してグラフ化しますの連続的に周期が長くなって来る様に表示されます。参考の為、補間無しグラフを示します。

<実行結果:回転角パルス周期補間処理なしグラフ>



19. 4. 6. パルス列から回転数を求める

19.4.5 項はパルス周期を求めていますが、パルス列から回転数を求めることは周期の逆数、周波数週を求める事になります。回転パルス列から回転数に変換する場合、専用の【FVC 関数】を使用します。

文法:

結果格納先 = FVC(1 回転当たりパルス数,対象数列,周期検出終止時間) 引数:

【1 回転当たりパルス数】<必須>
1 回転で何パルスかを記述します。

【対象数列】<必須> 解析対象パルス
数列を記述します。

【周期検出終止時間】<省略可> 低回転になりパルス周期が長くなった場合に、周期演算を停止する(回転数をゼロにする)周期時間を記述します。記 述省略した場合、回転数は 0 になりません。

結果格納先の単位は rpm となります。

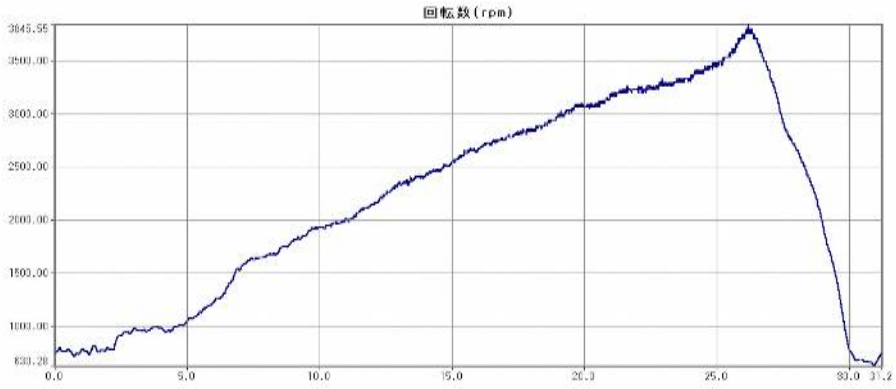
記述例:

エンジン回転数を回転パルスから求める場合

```
/*-- タコパルスからエンジン回転数を求める----------*/
$1 = TPC(1,#9)    /*演算に不要なパルスを除去*/
$3 = FVC(1,$1,1) /*タコパルスから回転数に変換*/
/*----グラフ描画処理----------------------------------*/
$2 "時刻歴:sec" = SPB(0)
def graph_id @1 "回転数(rpm)"
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
def graph_aspect_ratio @1 2
plot @1 $2 $3
def file_id %1 "graph" grp
save plot %1 @1 $2 $3
end
```

※ 不要なパルス除去処理は、記述例で使用した解析対象回転パルス固有の問題で、一般的には不要です。

<実行結果:回転数グラフ>



※ 回転数パルスはサンプリング周期Δt によりサンプリングされる為、抽出した周期にジッタを含みます。従って、演算結果の回転数にも、そのジッタ分が反映されて来ます。ディジタル方式の F-V で収録する場合は F-V 変換する段階で、高速サンプリングして周期精度を確保しますが、回転パルスから回転数を求める場合は避けられない誤差となります。

演算結果の回転数波形にスムージング処理を行う場合【SPF 関数】を使用します。

文法:

結果格納先 = SPF(対象数列)

引数:

【対象数列】＜必須＞ フィッティング処理を行う対象
数列を記述します。

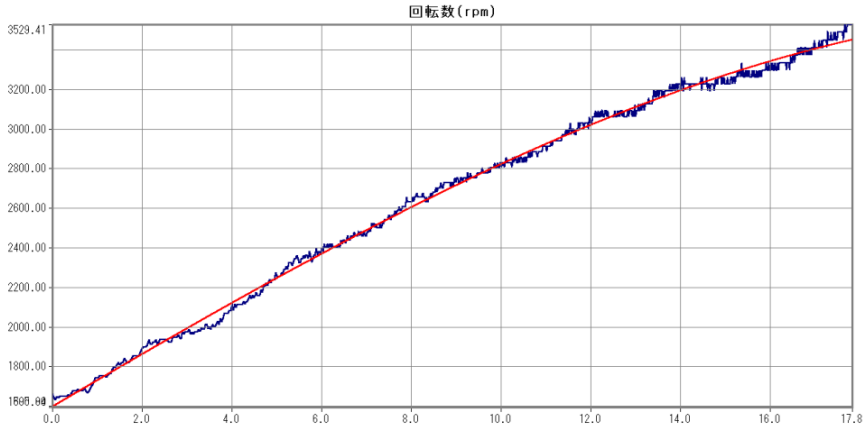
SPF 関数は最小二乗 3 次スプライン曲線へのフィッテングを行ないます。従って、対象数列が大きく変化をしている場合のフィッティング結果は現実とは離れてしまう事に注意が必要です。

記述例:

エンジン回転数を回転パルスから求め、回転上昇域をスムージング処理する場合

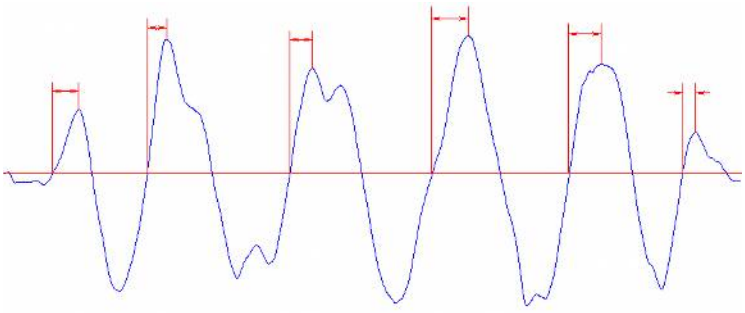
```
/*-- タコパルスからエンジン回転数を求める----------*/
$1 = TPC(1,#9)    /*演算に不要なパルスを除去*/
$3 = FVC(1,$1,1) /*タコパルスから回転数に変換*/
$4 = SPF($3)      /*スプライン曲線フィッティング*/
/*----グラフ描画処理----------------------------------*/
$2 "時刻歴:sec" = SPB(0)
def graph_id @1 "回転数(rpm)"
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
def graph_aspect_ratio @1 2
plot @1 $2 $3,$4
def file_id %1 "graph" grp
save plot %1 @1 $2 $3,$4
end
```

<実行結果:スムージング処理後回転数グラフ>



19. 5. 波形の或る地点から或る地点迄の時間を求める

19. 5. 1. 収録波形がゼロを上昇で過ってから最大値迄の時間を求める



演算手順:

① 解析波形の捉えたい最大値を含む閾値で論理化する
論理数列 = RWC(論理"1"閾値,論理"0"閾値,解析波形)
※ ここで求めた論理数列は最大値位置及び最大値を取得する為に使用します

② 論理数列立ち上がり(0⇒1 遷移)index を求める
論理遷移地点数列 = LST(1,論理数列)

③ 論理波形変換過渡領域を処理する 論理波形の先頭が論理"1"で始まる場合、次の立ち上がり地点迄の間を論理"0"にする この処理はロジックで記述します。

```
index 0 最初の立ち上がり遷移地点 /*index 制御文*/
    proc calc{
      論理数列 = DAG(論理数列の個数-1,0)
    }calc
```

※index 文で 0～最初の立ち上がり地点迄を演算範囲に設定し、その間を全て論理"0"に置き換えます

④ 論理数列"1"の範囲の最大値 index を求める
最大地点数列 = MXP(論理数列,解析波形)

⑤ 最大地点より遡りゼロを下降で過る地点を検索する
ゼロ通過地点 = DTD(下降通過,閾値,検索方向遡り,最大地点数列,解析波形)+1

⑥ 波形最大値、区間時間と傾斜を求める
最大値数列 = PTV(最大値地点数列,解析波形)
区間時間数列 = (最大値地点数列−ゼロ通過地点数列)*サンプリング周期 傾斜値数列 = 最大値数列/区間時間数列

記述例:

収録データ 4ch の波形のゼロを過ってから閾値 2 以上の最大地点迄の時間、最大値、傾斜を求める場合

```
/*---結果シート宣言処理-------------------------*/
dcl sheet 1
  { page 1:
     column $3,$4,$5,$6,$7
     format F0,F0,F4,F3,F3 2
}sheet
/*------ 区間時間解析処理----------------------------*/
$1 "論理波形:" = RWC(2,-2,#4)  /*対象波形論理化*/
$2 "論理遷移地点:" = LST(1,$1) /*論理立ち上がり位置検索*/
case_true $1(0)
/*----論理波形変換過渡領域処理-----------------------*/
proc 論理波形再構成{
  index 0 $2(0)
    proc calc{
      $1 = DAG(LEN($1)-1,0)
    }calc
}論理波形再構成
$3 "最大地点:" = MXP($1,#4)                  /*最大値位置検索*/
$4 "zero 通過地点:" = DTD(0,0,0,$2,#4)+1 /*ゼロ通過地点検索*/
$5 "時間:sec" = ($3-$4)*PRD()                /*区間時間演算*/
$7 "傾斜:" = $3/$5                           /*傾斜値演算*/
/*----- 結果シート書き込み処理----------------------*/
$6 "最大値:" = PTV($3,#4)     /*最大値位置データ取得*/
write ch_column 1: $[3,5]     /*結果シート書き込み*/
/*-----解析波形格納、波形書き込み値格納処理----------*/
def file_id %1 "wave" wav             /*波形ファイル番号定義*/
```

```
def wav_type %1 float
save wave %1 #4                    /*波形ファイル格納*/
def file_id %1 "position" pos      /*位置情報ファイル番号定義*/
save pos_info %1 $4 $3 1 5         /*位置情報ファイル格納*/
end
```

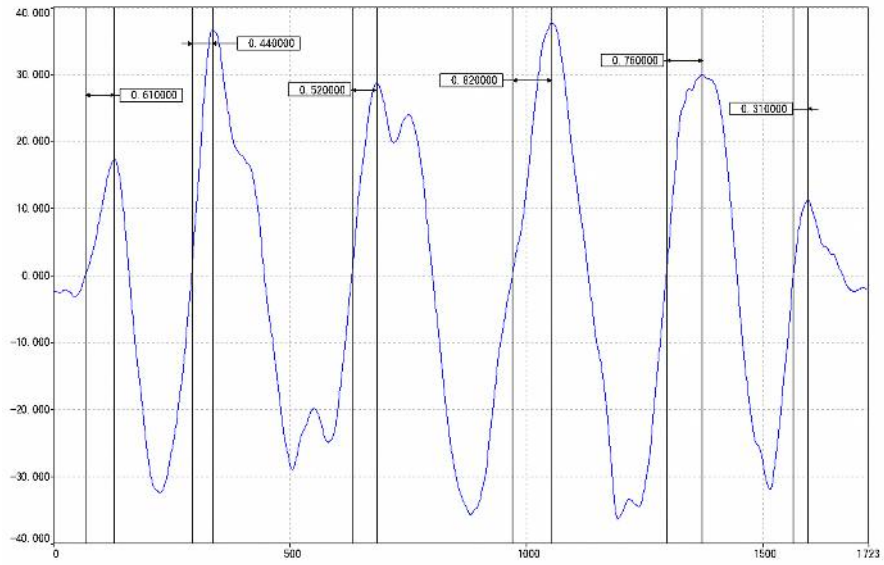
※ 記述例では論理波形変換閾値を±2 としています。又、対象波形は必ずゼロを過っている事を前提としています。

<実行結果:書き込まれた結果シート>

Page1:

最大地点	zero通過地点	時間(sec)	最大値	傾斜
193	132	0.6100	17.395	316.393
401	357	0.4400	36.655	911.364
748	696	0.5200	28.756	1438.462
1117	1035	0.8200	37.659	1362.195
1436	1360	0.7600	30.008	1889.474
1659	1628	0.3100	11.309	5351.613

<実行結果:格納された波形と区間時間値>



※ 実行後、PcWaveForm 波形表示 Window に表示された波形を「値書き込み機能」により、同じく格納された書き込み位置情報ファイルを選択して値自動書き込みを行った結果です。

19. 5. 2. 収録波形のあるチャネル間が設定閾値通過後、別のチャネルが閾値通過する迄の時間を求める

閾値通過 index を求める方法は幾つかあります。ここでは、其々対象波形を論理化数列に変換して、論理化数列同士の演算処理によって求める方法を記述します。

演算手順:

① 対象信号に其々閾値を設定し論理数列に変換し、論理波形の差分を求めます
信号1論理差分数列 = DIF(RWC(論理"1"閾値,論理"0"閾値,対象信号 1))
信号 2 論理差分数列 = DIF(RWC(論理"1"閾値,論理"0"閾値,対象信号 2))

② 2 信号の差分数列を合成し、遅延区間パルス列を生成します
遅延区間パルス列 = ACC(ASS(信号1論理差分数列,SGN(信号 2 論理差分数列)))

※ ASS 関数については後述します。

③ 遅延区間パルスの立ち上がり地点と立下り地点 index を求めます **遅延区間パルス立ち上がり地点 = LST(1,遅延区間パルス列) 遅延区間パルス立下り地点 = LST(0,遅延区間パルス列)**

④ 遅延時間を演算します
遅延時間 = (遅延区間パルス立下り地点-遅延区間パルス立ち上がり地点+1)*PRD()
※ 区間データ数に+1 する意味は、信号 2 の論理を符号反転して演算に使用した為、区間データ個数は 1 個不足する事によります。

パルスの差分結果を合成する場合【ASS 関数】を使用します。

文法:

結果格納先 = ASS(論理差分数列 1,論理差分数列 2)

引数:

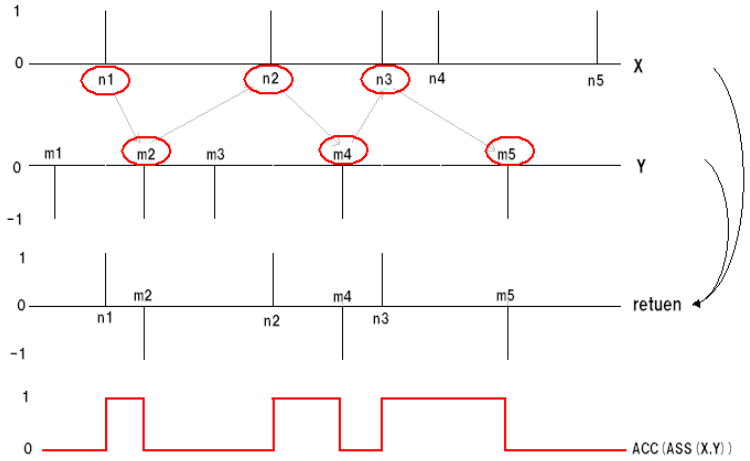
【論理差分数列 1】＜必須＞

立ち上がりを 1、立下りを-1 とした数列を記述します。関数では立ち上がり地点を参照します。

【論理差分数列 2】＜必須＞

立ち上がりを 1、立下りを-1 とした数列を記述します。関数では立下り地点を参照します。

論理差分数列1の立ち上がりを基準として、その地点から後方で直近の論理差分数列 2 の立下り地点を戻します。 従って、戻り数列を ACC 関数で累算することで立ち上がり地点が信号1．立下り地点が信号 2 の合成パルス列が生成でき ます。



記述例:

収録データ 3ch が閾値 6 を通過した地点と 4ch が閾値 6 を通過した地点の遅れ時間を求める場合

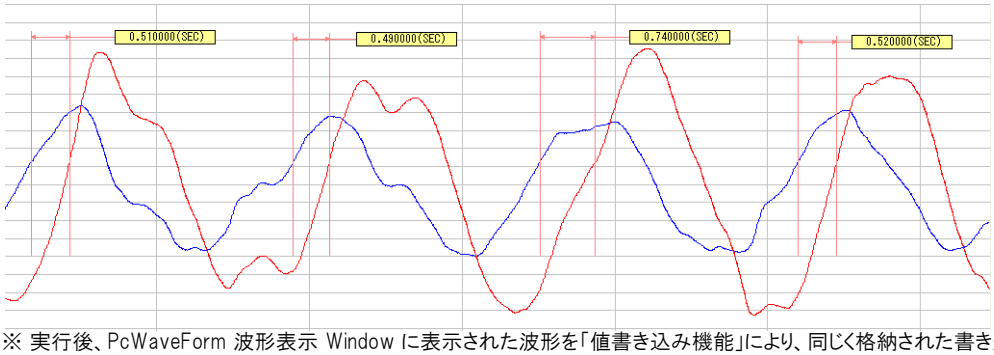
```
/*--- 結果シート使用宣言-------------------------*/
dcl sheet 1
  { page 1:
     column $7,$6
     format F3,F3 2
}sheet
/* 2 信号間遅れ時間処理-----------------------------*/
$1 = DIF(RWC(6,0,#3))          /*信号 1 論理化差分処理*/
$2 = DIF(RWC(6,0,#4))          /*信号 2 論理化差分処理*/
$3 = ACC(ASS($1,SGN($2)))      /*遅れ時間パルス生成*/
$4 = LST(1,$3)                 /*遅れ時間パルス立ち上がり地点検索*/
$5 = LST(0,$3)                 /*遅れ時間パルス立下り地点検索*/
$6 "遅れ時間:sec" = ($5-$4+1)*PRD() /*遅れ時間演算*/
$7 "信号 1 起点:sec" = $4*PRD()
/*--- 結果シート書き込み-----------------------------*/
write ch_column 1: $6,$7
/*-----解析波形格納、波形書き込み値格納処理----------*/
def file_id %1 "wave" wav      /*波形ファイル番号定義*/
def wav_type %1 float
save wave %1 #3,#4,$3              /*波形ファイル格納*/
def file_id %1 "position" pos /*位置情報ファイル番号定義*/
save pos_info %1 $4 $5 1 5     /*位置情報ファイル格納*/
def file_id %1 "wave" wav
end
```

<実行結果:書き込まれた結果シート>

Page1:

信号1起点(sec)	遅れ時間(sec)
0.360	0.510
3.790	0.490
7.020	0.740
10.400	0.520

<実行結果:格納された波形と時間遅れ値>



※ 実行後、PcWaveForm 波形表示 Window に表示された波形を「値書き込み機能」により、同じく格納された書き込み位置情報ファイルを選択して値自動書き込みを行った結果です。

19. 5. 3. 論理波形(数列)の論理"1"区間の時間を求める

論理波形の論理 0⇒1 への遷移地点と論理 1⇒0 への遷移点を其々求め、結果を引き算し、サンプリング周期を掛算することで求めます。

記述例:

収録データ・ch1 を論理数列と見なし、論理"1"区間の時間を求める場合

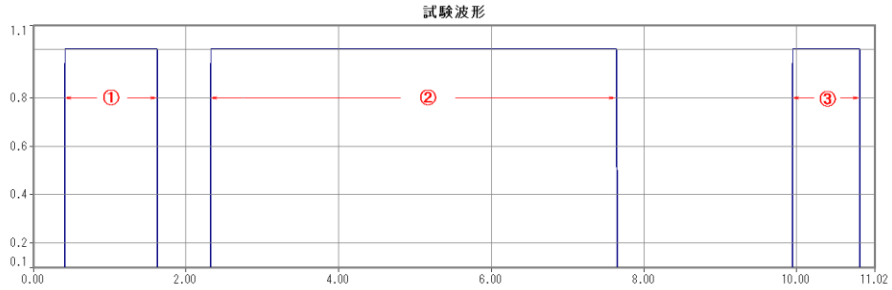
```
/*-----結果シート使用宣言--------------------*/
dcl sheet 1 {
  page 1: "論理 High 時間"
    column $5,$2,$3,$4
    format 3(F0),F4 2
}sheet
def sampl_period 1e-2
/*----- 論理波形"1"区間幅の演算---------------*/
$2 "論理 High 遷移 index:" = LST(1,#1)
$3 "論理 low 遷移 index:" = LST(0,#1)
case $2(0) > $3(0)
  proc low_index 再構成{
    $3 = ERC(1,LEN($3)-1,$3)
  }low_index 再構成
$4 "High 区間:sec" = ($3-$2)*PRD()
$5 "区間番号:" = ACC(DAG(LEN($4),1))
write ch_column 1: $[2,4]
end
```

<実行結果:結果シートに書き込まれた論理 High 時間>

Page1: 論理High時間

区間番号	理High遷移ind	理low遷移inde	High区間(sec)
1	42	163	1.2100
2	233	765	5.3200
3	995	1083	0.8800

<解析対象論理波形>



20. 演算関数を使用して振動曝露解析を行う

20. 1. 振動感覚補正加速度を求める

計測した加速度に振動感覚補正フィルタ処理して補正加速度に変換します。

20. 1. 1. 演算関数で用意されている振動感覚補正フィルタの種類と適応

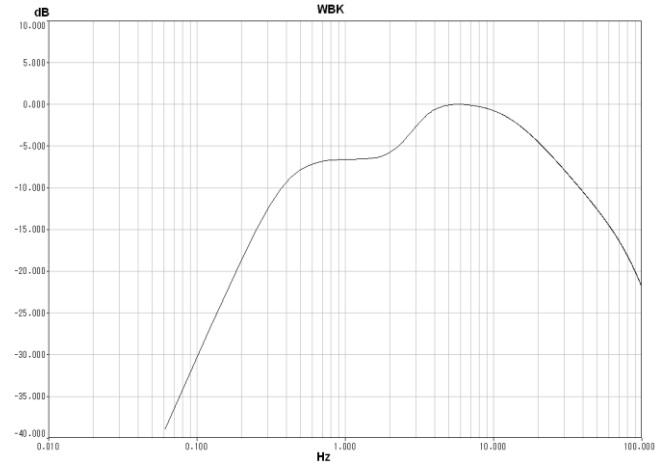
① ISO2631-1 W_k 振動感覚補正フィルタ【WBK 関数】を使用します。

文法:

結果格納先 = WBK(対象加速度数列)

適用:

座位座面 Z 軸加速度＜健康、快適性、知覚＞
座位足支持部 X 軸加速度＜快適性＞
座位足支持部 Y 軸加速度＜快適性＞
座位足支持部 Z 軸加速度＜快適性＞
立位 Z 軸加速度＜快適性、知覚＞
仰臥位 Z 軸加速度(頭部を除く)＜快適性、知覚＞

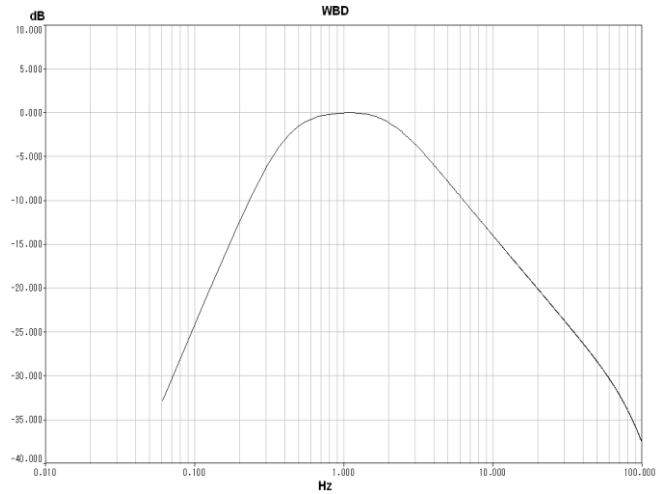


② ISO2631-1W_d 振動感覚補正フィルタ【WBD 関数】を使用します。**文法:**

結果格納先 = WBD(対象加速度数列)

適用:

座位座面 X 軸加速度＜健康、快適性、知覚＞
座位座面 Y 軸加速度＜健康、快適性、知覚＞
背もたれ Y 軸加速度＜快適性＞ 背もたれ Z 軸加速度＜快適性＞
立位、仰臥位 X 軸加速度＜快適性、知覚＞
立位、仰臥位 Y 軸加速度＜快適性、知覚＞

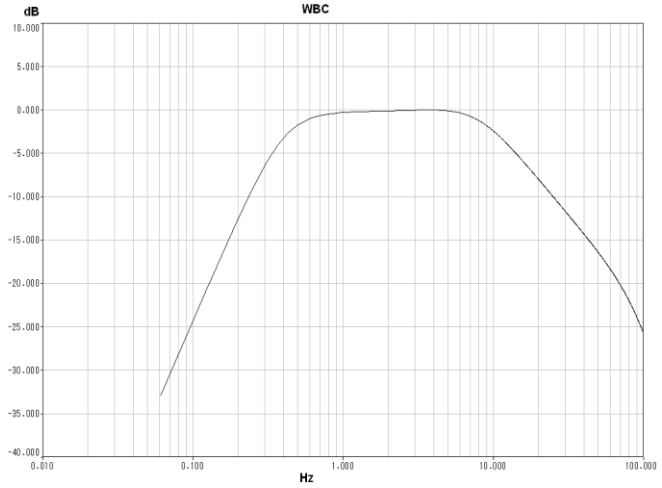


③ ISO2631-1W_c振動感覚補正フィルタ【WBC 関数】を使用します。**文法:**

結果格納先 = WBC(対象加速度数列)

適用:

背もたれ X 軸加速度＜健康、快適性、知覚＞

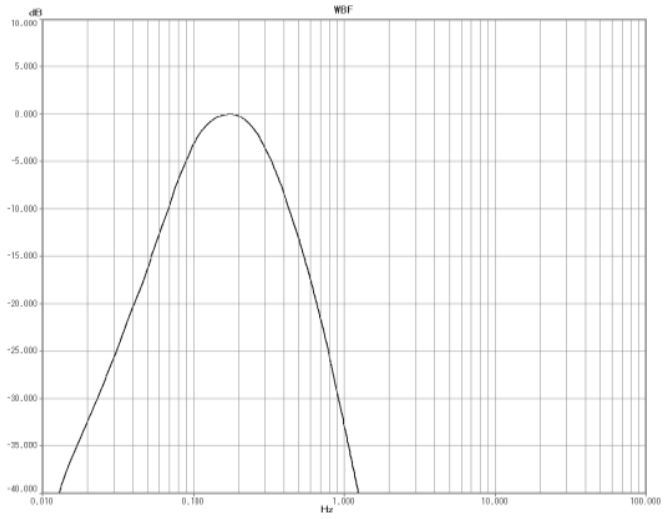


④ ISO2631-1W_f振動感覚補正フィルタ【**WBF 関数**】を使用します。**文法:**

結果格納先 = WBF(対象加速度数列) 適用:

乗り物酔い Z 軸加速度＜動揺病＞

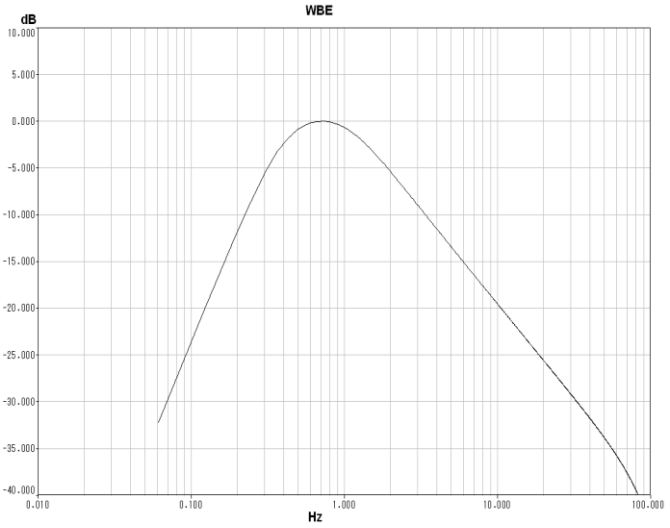
※ 船舶の場合は Z 軸を動揺病の基準軸としますが、他の乗り物の場合、動揺病と相関の或る軸を対象とします。又、その時、フィルタ特性が適合しているか検討する必要があります。



⑤ ISO2631-1W_e振動感覚補正フィルタ【**WBE 関数**】を使用します。**文法:**

結果格納先 = WBE(対象角加速度数列) 適用:

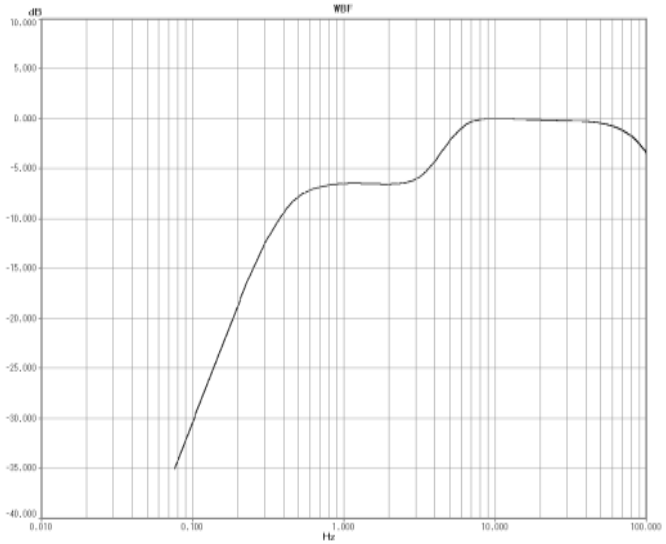
座位座面 X 軸角加速度＜快適性、知覚＞
座位座面 Y 軸角加速度＜快適性、知覚＞
座位座面 Z 軸角加速度＜快適性、知覚＞



⑥ ISO2631-1W_j振動感覚補正フィルタ【**WBJ関数**】を使用します。**文法:**

結果格納先 = WBJ(対象加速度数列) 適用:

仰臥位頭部加速度＜快適性、知覚＞



⑦ ISO2631-4W_b 振動感覚補正フィルタ 【WBB 関数】を使用します。

文法:

結果格納先 = WBB(フラグ,対象加速度数列) 引数:

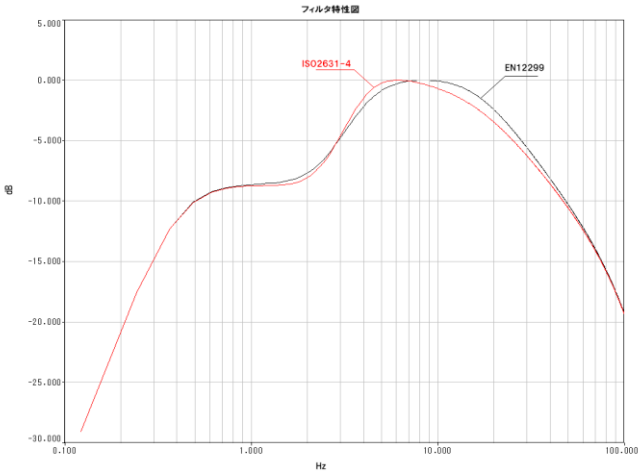
【フラグ】＜省略可＞

フラグ	フィルタ特性
0	ISO2631-4 W_b
1	EN12299

フラグ記述省略時 0 と見なします。

適用:

鉄道乗客乗員 Z 軸加速度＜快適性＞



⑧ ISO-5349-1 手腕振動感覚補正フィルタ 【HTV 関数】を使用します。

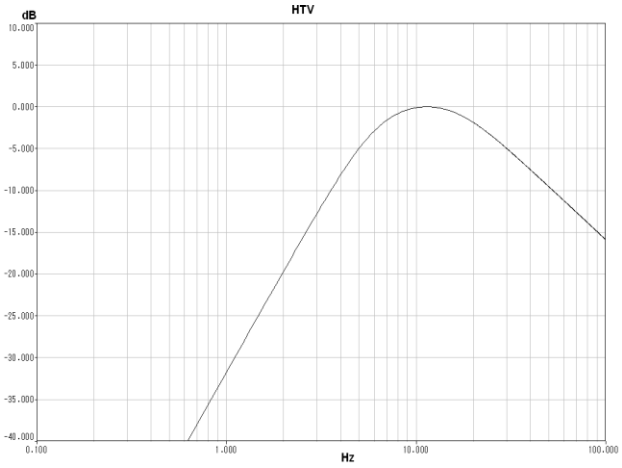
文法:

結果格納先 = HTV(対象加速度数列) 適用:

手腕 X 軸加速度＜健康＞

手腕 Y 軸加速度＜健康＞

手腕 Z 軸加速度＜健康＞



20. 1. 2. 収録加速度データから補正加速度/補正角加速度を求める

記述例:

収録を下表に示す内容で行ったものとして、補正加速度/補正角加速度を求める場合

計測部位	補正係数 k	補正フィルタ	収録チャネル	収録単位
背もたれ X 加速度	0.8	Wc (WBC 関数)	#1	m/s^2
背もたれ y 加速度	0.5	Wd (WBD 関数)	#2	m/s^2
背もたれ z 加速度	0.4	Wd (WBD 関数)	#3	m/s^2
座面 x 加速度	1(1.4)	Wd (WBD 関数)	#4	m/s^2
座面 y 加速度	1(1.4)	Wd (WBD 関数)	#5	m/s^2
座面 z 加速度	1	Wk (WBK 関数)	#6	m/s^2
座面 x 角加速度	0.63	We (WBE 関数)	#7	rad/s^2
座面 y 角加速度	0.4	We (WBE 関数)	#8	rad/s^2
座面 z 角加速度	0.2	We (WBE 関数)	#9	rad/s^2
足支持面 x 加速度	0.25	Wk (WBK 関数)	#10	m/s^2
足支持面 y 加速度	0.25	Wk (WBK 関数)	#11	m/s^2
足支持面 z 加速度	0.4	Wk (WBK 関数)	#12	m/s^2

※ kは軸合成加速度実効値を求める場合に部位ごとに必要な補正係数です。

※ 収録データのチャネル番号は記述例説明の都合上、割り当てたものです。

※ X 軸:前後方向、Y 軸:左右方向、Z 軸:上下方向を意味します。

```
$1 “背もたれ X 軸補正加速度:m/s^2” = WBC(#1)
$2 “背もたれ Y 軸補正加速度:m/s^2” = WBD(#2)
$3 “背もたれ Z 軸補正加速度:m/s^2” = WBD(#3)
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
$7 “座面 X 軸補正角加速度:rad/s^2” = WBE(1,#7)
$8 “座面 Y 軸補正角加速度:rad/s^2” = WBE(1,#8)
```

```
$9 “座面 Z 軸補正角加速度:rad/s^2” = WBE(1,#9)
$10 “足支持面 X 軸補正加速度:m/s^2” = WBK(#10)
$11 “足支持面 Y 軸補正加速度:m/s^2” = WBK(#11)
$12 “足支持面 Z 軸補正加速度:m/s^2” = WBK(#12)
```

$1～$12 に対応する補正加速度または補正角加速度が格納されます。

※ 対象数列の単位は m/s^2 が前提です。もし単位が G の場合、変換する必要があります。

重力加速度1G = 9.80665m/s^2 として変換します。

```
$4 “座面 X 軸補正加速度実効値:m/s^2” = WBD(#4*9.80665)
```

※ 信号名、単位は演算式で定義せず、def ch_name 文で定義しても良い。

```
def ch_name $4 “座面 X 軸補正加速度実効値:m/s^2”
```

20. 2. 振動感覚補正加速度、補正角加速度から実効値を求める

20. 2. 1. 移動実効値を求める

移動実効値を求める場合【RRV 関数】を使用します。

文法:

格納先 = RRV(積分時定数,対象数列,フラグ)

引数:

【積分時定数】＜必須＞

積分時定数を記述します。単位は sec で記述します。例えば 125mS とする場合は 0.125 と記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

【フラグ】＜省略可＞

フラグは、0 または 1 と記述します。又、記述省略した場合は 0 と見なします。フラグに:0 と記述した場合は戻り数列の長さは設定した積分時間分短くなります。

フラグに 1 と記述した場合は戻り数列の長さは対象数列と同じになりますが、先頭から積分周期を満たない部分は過渡領域となります。

関数内部演算式を示します。

移動実効値（フラグ省略または 0の時）

$$A_w(t) = \left\{ \frac{1}{\tau} \int_{i=t}^{t} \left[a_w(i) \right]^2 dt \right\}^{\frac{1}{2}}$$

移動実効値（フラグ省略または 1の時）

$$A_w(t) = \left\{ \frac{1}{\tau} \int_{i=t-\tau}^{t} \left[a_w(i) \right]^2 dt \right\}^{\frac{1}{2}}$$

$A_w(t)$は t 時刻における移動実効値、 $a_w(i)$:補正加速度の瞬時値、 τ ：移動平均の積分時間

記述例:

補正加速度/角加速度から積分周期 1sec として補正加速度/角加速度移動実効値を求める場合

```
/*------- 記述例①:補正加速度/角加速度を求める-*/
$1 “背もたれ X 軸補正加速度:m/s^2” = WBC(#1)
$2 “背もたれ Y 軸補正加速度:m/s^2” = WBD(#2)
$3 “背もたれ Z 軸補正加速度:m/s^2” = WBD(#3)
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
$7 “座面 X 軸補正角加速度:rad/s^2” = WBE(1,#7)
$8 “座面 Y 軸補正角加速度:rad/s^2” = WBE(1,#8)
$9 “座面 Z 軸補正角加速度:rad/s^2” = WBE(1,#9)
$10 “足支持面 X 軸補正加速度:m/s^2” = WBK(#10)
$11 “足支持面 Y 軸補正加速度:m/s^2” = WBK(#11)
$12 “足支持面 Z 軸補正加速度:m/s^2” = WBK(#12)
/*-------- 移動実効値を求める-------------------------------*/
$13 “背もたれ X 軸補正加速度実効値:m/s^2” = RRV(1,$1)
$14 “背もたれ Y 軸補正加速度実効値:m/s^2” = RRV(1,$2)
$15 “背もたれ Z 軸補正加速度実効値:m/s^2” = RRV(1,$3)
$16 “座面 X 軸補正加速度実効値:m/s^2” = RRV(1,$4)
$17 “座面 Y 軸補正加速度実効値:m/s^2” = RRV(1,$5)
$18 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,$6)
$19 “座面 X 軸補正角加速度実効値:rad/s^2” = RRV(1,$7)
$20 “座面 Y 軸補正角加速度実効値:rad/s^2” = RRV(1,$8)
```

$21 “座面 Z 軸補正角加速度実効値:rad/s^2” = RRV(1,$9)
$22 “足支持面 X 軸補正加速度実効値:m/s^2” = RRV(1,$10)
$23 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,$11)
$24 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,$12)

$13～$24 には対応した加速度実効値/角加速度実効値が格納されます。

※ $1～$12 には振動感覚補正した加速度/角加速度が格納されている事が前提です。 直接収録データから移動実効値を求める場合は、演算式に補正フィルタ処理をネストして記述します。

$13 “背もたれ X 軸補正加速度実効値:m/s^2” = RRV(1,WBD($1))

20. 2. 2. 時間重み付け移動実効値を求める

時間重み付け移動実効値を求める場合【RRT 関数】を使用します。 **文法:**

格納先 = RRT(積分時定数,対象数列,演算打ち切り値)

引数:

【積分時定数】＜必須＞ 積分

時定数を記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

【演算打ち切り値】＜必須＞

演算打ち切り値は負数或いは 1 以上は設定できません。なお、記述省略した場合は 0.001 をとります。
現在時刻 t を 0 とすると、遡り時間は ξ = τ ×Ln(演算打ち切り値)で表せ、0.001 で打ち切る場合は、
ξ = τ × Ln(0.001) = τ ×(-6.907755279)となります。例えば、実効値時定数を 1s とした場は、-6.907755279 秒遡ることを意味します。実際には、データ個数に変換する為、遡り戻りデータ個数を求める必要があり、サンプリング周期⊿tで割り整数化した値を遡り個数とします。又、この値を小さくすると演算時間が大きく掛ります。

関数内部演算式を示します。

時間重み付け移動実効値

$$Aw_t = \left\lceil \frac{1}{\tau} \int_{\xi=-\infty}^{t} aw_\xi^2 \cdot e^{-(t-\xi)/\tau} d\xi \right\rceil^{\frac{1}{2}}$$

記述例:

座面 X,Y,Z 補正加速度を積分周期 1Sec、打ち切り値を 0.01(遡り時間は約 4.6 秒)として時間重み付け移動実効値を求める場合

```
/*-------記述例①:補正加速度を求める----------------------*/
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
/* ------時間重み付け移動実効値を求める----------------------*/
$16 “座面 X 軸補正加速度実効値:m/s^2” = RRT(1,$4,0.01)
$17 “座面 Y 軸補正加速度実効値:m/s^2” = RRT(1,$5,0.01)
$18 “座面 Z 軸補正加速度実効値:m/s^2” = RRT(1,$6,0.01)
```

$16,$17,$18 には対応した時間重み付け移動加速度実効値が格納されます。

20. 2. 3. 設定した区間ごとに実効値を求める

区間ごとの実効値を求める場合【EFF 関数】を使用します。 **文法:**

格納先 = EFF(演算範囲データ個数,対象数列) 引数:

【演算範囲データ個数】＜省略可＞

演算範囲データ個数は積分区間のデータ個数を意味します。記述省略された場合は対象数列全体が演算対象となり、又、数列で記述した場合は対象数列の Index と見なし、Index 間を区間と見なして処理されます。例えば、0,1000,2000 とした場合は 0～999、1000～1999 の 2 区間の実効値を演算します。なお、数列並びは昇順の必要があります。

【対象数列】＜必須＞ 演算対象

数列を記述します。

関数内部演算式を示します。

実効値 $A_w = \left[\frac{1}{T}\int_0^T a_w^2(t)\,dt\right]^{\frac{1}{2}}$

$a_w(t)$ **：補正フィルタ通過後の加速度瞬時値（m/s^2 または rad/s^2）を意味します。** T **：計測区間（sec）**

記述例:

収録データから計測区間 5 分ごとの実効値を求める場合

```
/*--------- 計測区間ごとの実効値を求める----------------------------------------*/
$25 “背もたれ X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBC(#1))
$26 “背もたれ Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#2))
$27 “背もたれ Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#3))
$28 “座面 X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#4))
$29 “座面 Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#5))
$30 “座面 Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBK(#6))
$31 “座面 X 軸補正角加速度実効値:rad/s^2” = EFF(300/PRD(),WBE(#7))
$32 “座面 Y 軸補正角加速度実効値:rad/s^2” = EFF(300/PRD(),WBE(#8)
$33 “座面 Z 軸補正角加速度実効値:rad/s^2” = EFF(300/PRD(),WBE(#9))
$34 “足支持面 X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBK(#10))
$35 “足支持面 Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBK(#11))
$36 “足支持面 Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBK(#12))
```

$25～$36 にはには 5 分ごとの加速度実効値/角加速度実効値が格納されます。300*PRD() は 5 分(300 秒)をサンプリング周期で割り演算データ個数に変換しています。演算式中で補正加速度に変換してから区間ごとの実効値を求めています。

記述例:

収録データファイルに作業ごとに付けられたマーク間ごとに座面 X,Y,Z の実効値を求める場合

```
$64 “マーク index:” = LNK(MRK(0),LEN(#7)-1)
$65 “作業時間:sec” = (ERC(1,LEN($64)-1,$64)-ERC(0,LEN($64)-2,$64))*PRD()
&1 “マークメモ:” = CMMR(0)
$61 “座面 X 軸補正角加速度実効値:rad/s^2” = EFF($64,WBE(#7))
$62 “座面 Y 軸補正角加速度実効値:rad/s^2” = EFF($64,WBE(#8)
$63 “座面 Z 軸補正角加速度実効値:rad/s^2” = EFF($64,WBE(#9))
```

$64 には、収録ファイルに付けられている全てのマーク位置 Index が格納され、$65 には、マーク間の作業時間が格納されます。なお、収録ファイルの終端にはマークが付けられていないことを前提として最終マークを付加しています。&1 にはマークに付けられているメモが格納されます。$61～$63 には、作業区間ごとの実効値が格納されます。

MRK( )はマーク位置 Index 取得関数、LEN( )は数列要素数取得関数、LNK( )は数列連結関数、ERC( )は数列切り出し関数、CMMR( )はマークメモ取得関数です。

20. 3. 補正加速度実効値または補正角加速度実効値の軸合成と全体合成加速度実効値を求める

軸合成演算及び全体合成演算式を示します。

軸合成補正加速度 $A_{wc} = \left[(k \cdot A_{wx})^2 + (k \cdot A_{wy})^2 + (k \cdot A_{wz})^2\right]^{\frac{1}{2}}$

k：測定部位、軸ごとの補正係数、 a_{wx} **：X 軸実効値、** a_{wy} **：Y 軸実効値、** a_{wz} **：Z 軸実効値**

全体の合成値 $A_w = \left[A_{wB}^2 + A_{wS}^2 + A_{wR}^2 + A_{wL}^2\right]^{\frac{1}{2}}$

A_{wB} **：背もたれ 3 軸合成補正加速度、** A_{wS} **：座面 3 軸合成補正加速度、** A_{wR} **：座面モーメント 3 軸合成補正加速度、**

A_{wL} **：足支持部 3 軸合成補正加速度**

記述例:

快適性評価用に背もたれ、座面、座面モーメント、足支持部ごとの合成と全体合成を求める場合

```
/*---------記述例④:計測区間ごとの実効値を求める----------------------------------*/
$25 “背もたれ X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBC(#1))
$26 “背もたれ Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#2))
$27 “背もたれ Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#3))
$28 “座面 X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#4))
$29 “座面 Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBD(#5))
$30 “座面 Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),WBK(#6))
$31 “座面 X 軸補正角加速度実効値:rad/s^2” = EFF(300/PRD(),WBE(#7))
$32 “座面 Y 軸補正角加速度実効値:rad/s^2” = EFF(300/PRD(),WBE(#8)
```

```
$33 "座面 Z 軸補正角加速度実効値:rad/s^2" = EFF(300/PRD(),WBE(#9))
$34 "足支持面 X 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBK(#10))
$35 "足支持面 Y 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBK(#11))
$36 "足支持面 Z 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBK(#12))
/*----------軸合成加速度、全体合成加速を求める----------------------------------------*/
$37 "背もたれ軸合成:m/s^2" = SQR((0.8*$25)^2+(0.5*$26)^2+(0.4*$27)^2)
$38 "座面軸合成:m/s^2" = SQR($28^2+$29^2+$30^2)
$39 "座面モーメント軸合成:m/s^2" = SQR((0.63*$31)^2+(0.4*$32)^2+(0.2*$33)^2)
$40 "足支持部軸合成:m/s^2" = SQR((0.25*$34)^2+(0.25*$35)^2+(0.4*$30)^2)
$41 "全体合成実効値:m/s^2" = SQR($37^2+$38^2+$39^2+$40^2)
```

$41 には、各部位の軸合成加速度を更に全体合成した結果が格納されます。
※ 快適性評価での座面合成演算では X,Y,Z 軸の補正係数 k は1が使用されます。
※ SQR( )関数は平方根関数です。

記述例:

健康評価用に座面軸合成加速度を求める場合

```
/*---------記述例④：計測区間ごとの実効値を求める----------------------*/
$28 "座面 X 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBD(#4))
$29 "座面 Y 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBD(#5))
$30 "座面 Z 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBK(#6))
*-----座面軸合成加速度を求める----------------------------------------------*/
$42 "座面軸合成:m/S^2" = SQR((1.4*$28)^2+(1.4*$29)^2+$30^2)
```

$42 には、座面軸合成加速度が格納されます。
※ 健康評価での座面合成演算では X,Y 軸の補正係数 k は1.4、Z 軸の補正係数kは1が使用されます。

20. 4. 最大過渡振動値(MTVV:Maximum Transient Vibration Value)を求める

最大過渡振動値は振動感覚補正後の加速度を移動実効値の計測区間内の最大加速度実効値を意味します。

記述例:

座面 X,Y,Z 加速度から 5 分ごとの最大過渡振動値を求める場合

```
/*----------補正加速度を求める--------------------*/
$4 "座面 X 軸補正加速度:m/s^2" = WBD(#4)
$5 "座面 Y 軸補正加速度:m/s^2" = WBD(#5)
$6 "座面 Z 軸補正加速度:m/s^2" = WBK(#6)
/*---------移動実効値を求める--------------------*/
$16 "座面 X 軸補正加速度実効値:m/s^2" = RRV(1,$4)
$17 "座面 Y 軸補正加速度実効値:m/s^2" = RRV(1,$5)
$18 "座面 Z 軸補正加速度実効値:m/s^2" = RRV(1,$6)
$43 "座面軸合成:m/s^2" = SQR((1.4*$16)^2+(1.4*$17)^2+$18^2)
/*---------最大過渡振動値を求める-----------------*/
$44 "MTVV_X:m/S2" = MAX(300/PRD(),$16)
$45 "MTVV_Y:m/S2" = MAX(300/PRD(),$17)
$46 "MTVV_Z:m/S2" = MAX(300/PRD(),$18)
$47 "MTVV_C:m/S2" = MAX(300/PRD(),$43)
```

$ 44 には座面 X 軸 MTVV、$ 45 には座面 Y 軸 MTVV、$46 には座面 Z 軸 MTVV、$47 には座面軸合成 MTVV が格納されます。

20. 5. 四乗暴露量値(VDV:Fourth Power Vibration Dose Value)を求める

演算式を示します。

四乗暴露量値 $VDV = \left\{ \int_0^T \left[a_w(t) \right]^4 dt \right\}^{\frac{1}{4}}$

記述例:

座面 X,Y,Z 加速度から計測区間 5 分とした区間ごとの四乗暴露量値を求める場合

```
/*---------補正加速度を求める--------------------*/
$4 "座面 X 軸補正加速度:m/s^2" = WBD(#4)
$5 "座面 Y 軸補正加速度:m/s^2" = WBD(#5)
$6 "座面 Z 軸補正加速度:m/s^2" = WBK(#6)
/*---------VDV を求める----------------------*/
$48 "座面 X 軸 VDV:m/s^1.75" = SUM(300/PRD(),($4^4)*PRD())^0.25
$49 "座面 Y 軸 VDV:m/s^1.75" = SUM(300/PRD(),($5^4)*PRD())^0.25
$50 "座面 Z 軸 VDV:m/s^1.75" = SUM(300/PRD(),($6^4)*PRD())^0.25
```

$48,$49,$50 には、座面 X,Y,Z 加速度の四乗暴露量値が格納されます。

20. 6. 乗り物酔い暴露量値(Motion Sickness Dose Value)を求める

演算式を示します。

$$\mathbf{MSDV_Z} = \left(\int_0^T a_w^2(t)dt\right)^{\frac{1}{2}}$$

a_w は、瞬時加速度に補正フィルタ W_f を掛けた補正加速度、T は曝露時間

記述例:

座面 Z 軸補正加速度の計測区間 5 分ごとの乗り物酔い暴露量値を求める場合

```
$61 “座面 Z 軸乗り物酔い補正加速度:m/s^2” = WBF(#6)
$62 “MSDVz:m/s^2” = SUM(300/PRD(),($61^2)*PRD())^2
```

$61 には乗り物酔い補正加速度、$52 には MSDVz が格納されます。

※ W_f 補正フィルタは、船舶における乗り物酔い評価指数を求める為に使用されるフィルタです。
他の乗り物にそのまま適用できるか否かと対象加速度軸を合わせて検討する必要があります。

20. 7. クレストファクタ(Crest Factor)を求める クレストファクタを求

める場合 **【CFT 関数】** を使用します。**文法:**

格納先 = CFT(積分時定数,対象数列,フラグ)

引数:

【積分時定数】＜必須＞

積分時定数を記述します。

【対象数列】＜必須＞ 演算対象

数列を記述します。

【フラグ】＜省略可＞

フラグは、0 または 1 と記述します。又、記述省略した場合は 0 と見なします。フラグに:0 と記述した場合は戻り数列の長さは設定した積分時間分短くなります。

フラグに 1 と記述した場合は戻り数列の長さは対象数列と同じになりますが、先頭から積分周期を満たない部分は過渡領域となります。

記述例:

座面 X,Y,Z 補正加速度の積分区間 1sec としたクレストファクタを求める場合

```
/*----------補正加速度を求める----------*/
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
/*----------対応する補正加速度移動実効値------*/
$16 “座面 X 軸補正加速度実効値:m/s^2” = RRV(1,$4)
$17 “座面 Y 軸補正加速度実効値:m/s^2” = RRV(1,$5)
$18 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,$6)
/*----------クレストファクタを求める----------*/
$51 “座面 X 軸補正加速度クレストファクタ:” = CFT(1,$4)
$52 “座面 Y 軸補正加速度クレストファクタ:” = CFT(1,$5)
$53 “座面 Z 軸補正加速度クレストファクタ:” = CFT(1,$6)
```

$51,$52,$53 には、補正加速度移動実効値に対応したクレストファクタが格納されます。

記述例:

座面 X,Y,Z 補正加速度の計測区間 5 分とした区間ごとのクレストファクタを求める場合

```
/*----------補正加速度を求める----------*/
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
/*----------区間ごとの補正加速度実効値を求める----*/
$28 “座面 X 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),$4)
$29 “座面 Y 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),$5)
$30 “座面 Z 軸補正加速度実効値:m/s^2” = EFF(300/PRD(),$6)
/*----------クレストファクタを求める---------*/
$54 “座面 X 軸補正加速度クレストファクタ:” = MAX(300/PRD(),ABS($4))/$28
$55 “座面 Y 軸補正加速度クレストファクタ:” = MAX(300/PRD(),ABS($5))/$29
$56 “座面 Z 軸補正加速度クレストファクタ:” = MAX(300/PRD(),ABS($6))/$30
```

$54,$55,$56 には、区間ごとの加速度実効値に対応したクレストファクタが格納されます。

※ ABS( )関数は絶対値関数です。

20. 8. 振動レベルを求める

演算式を示します。

振動レベル $V_L = 20Log_{10}\left\{\frac{A_w}{10^{-5}}\right\}$

振動レベル 0dB は 10^{-5}m/s^2 となります。

記述例:

座面 X,Y,Z 補正加速度を振動レベルに変換する場合

```
/*---------補正加速度を求める----------------*/
$4 “座面 X 軸補正加速度:m/s^2” = WBD(#4)
$5 “座面 Y 軸補正加速度:m/s^2” = WBD(#5)
$6 “座面 Z 軸補正加速度:m/s^2” = WBK(#6)
/*--------移動実効値を求める----------------*/
$16 “座面 X 軸補正加速度実効値:m/s^2” = RRV(1,$4)
$17 “座面 Y 軸補正加速度実効値:m/s^2” = RRV(1,$5)
$18 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,$6)
$43 “座面軸合成:m/s^2” = SQR((1.4*$16)^2+(1.4*$17)^2+$18^2)
/*--------振動レベルを求める----------------*/
$57 “座面 X 軸振動レベル:dB” = 20*LGT($16/1e-5)
$58 “座面 Y 軸振動レベル:dB” = 20*LGT($17/1e-5)
$59 “座面 Z 軸振動レベル:dB” = 20*LGT($18/1e-5)
$60 “座面合成振動レベル:dB“ = 20*LGT($43/1e-5)
```

$57,$58,$59,$60 には、移動加速度実効値を振動レベルに変換した値が格納されます。

20. 9. 補正加速度実効値の全持続時間相当エネルギ等価振動値を求める

演算式を示します。

全持続時間相当エネルギ等価振動 $A_{w.e} = \left[\frac{\sum_i wti \cdot T}{A^2 \sum T_i}\right]^{\frac{1}{2}}$

補正加速度実効値の全時間相当エネルギ等価振動値を求める場合、補正加速度実効値の時間率頻度を求める必要があります。時間率頻度を求める場合、**【TRC 関数】**を使用します。

文法:

格納先 = TRC(セルサイズ,セルの合計個数,対象数列)

引数:

【セルサイズ】＜必須＞ 時間率頻度を求める場合のセル
サイズを記述します。

【セルの合計個数】＜必須＞
セルの合計個数を記述します。TRC 関数は正負領域を持ちます。実効値の場合、負領域は存在しませんが記述するセルの合計値は正負の領域の個数を記述します。

【対象数列】＜必須＞
演算対象数列を記述します。

時間率頻度解結果のセル中央値を求める場合、**【CNV 関数】**を使用します。

文法:

格納先 = CNV(フラグ,セルサイズ,セルの合計個数,正負フラグ)

引数:

【フラグ】＜必須＞
戻り数列の種別を意味するフラグで、セル中央値を取得する場合は 1、セル番号を取得する場合は 0 と記述します。

【セルサイズ】＜必須＞ セル
サイズを記述します。

【セルの合計個数】＜必須＞
セルの合計個数を記述します。

【正負フラグ】＜省略可＞
セルが正負領域を持つのか正領域のみかを意味するフラグで正負領域を持つ場合は 1、正領域のみの場合は 0 と記述します。時間率頻度結果のセル中央値或いはセル番号を求める場合は 1 と記述します。なお、記述省略した場合は 0 と見なします。

記述例：

座面合成補正加速度実効値をセルサイズ 0.01、最大 10m/s^2 とした時間率頻度解析を行い、全持続時間エネルギ等価振動を求める場合

```
/*---------移動実効値を求める-----------------------------------*/
$16 “座面 X 軸補正加速度実効値:m/s^2” = RRV(1,WBD(#4))
$17 “座面 Y 軸補正加速度実効値:m/s^2” = RRV(1,WBD(#5))
$18 “座面 Z 軸補正加速度実効値:m/s^2” = RRV(1,WBK(#6))
$43 “座面軸合成:m/s^2” = SQR((1.4*$16)^2+(1.4*$17)^2+$18^2)
/*--------時間率頻度を求める-----------------------------------*/
$66 “暴露時間:sec” = TRC(0.01,2000,$43)*PRD()
$67 “等価セル中央値:m/s^2” = ERC(100,199,CNV(1,0.01,200,1))
$68 “暴露量時間:m/s^2” = $66*$67
$69 “全持続時間等価振動エネルギ:” = SQR(SUM($67^2*$66)/SUM($66))
```

記述例20. 1. 解析範囲の 5 分ごとの曝露量を解析し結果シートに表示する

PcWaveForm 波形 Window で解析範囲を指定し、解析範囲の全持続時間等価振動エネルギ及び 5 分ごとの振動値、最大過渡振動値、四乗暴露量値を結果シートに表示します。

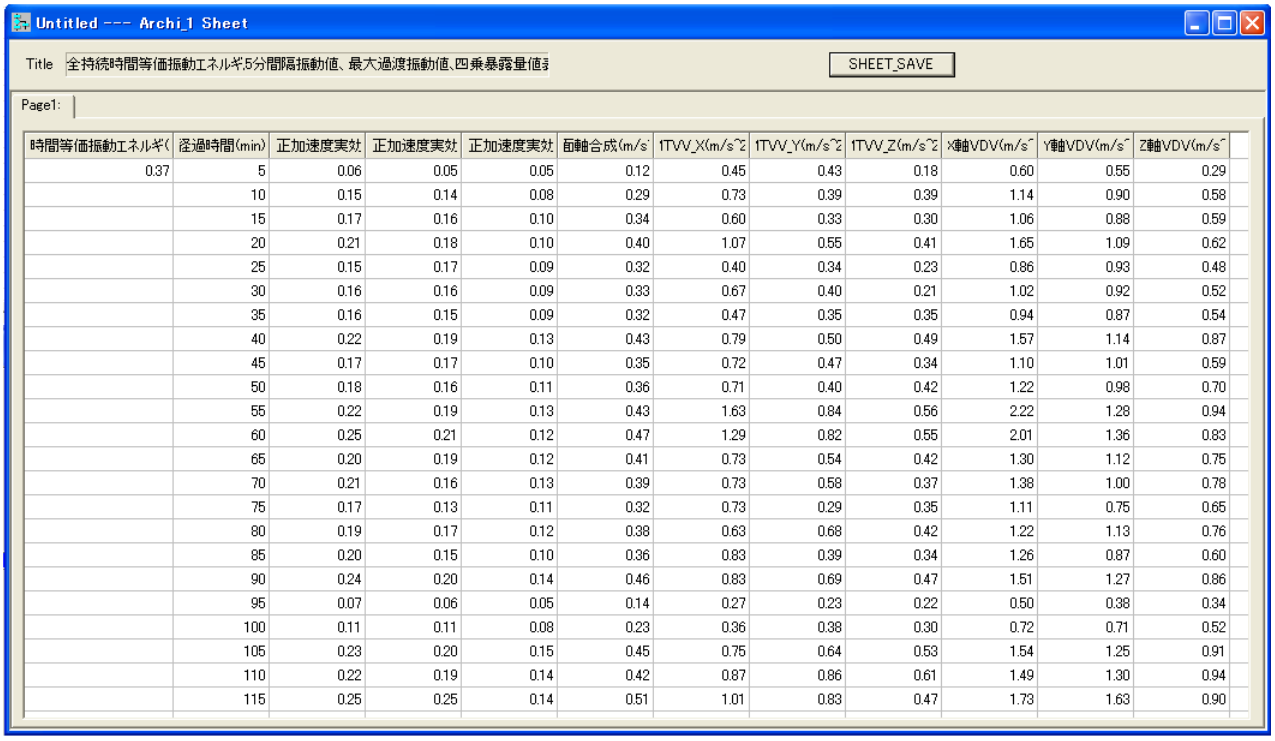
<Archi_1 Script 記述例>

```
1       /* WBV Sample 1 prc                                                    */
2       dcl menu_label "全身振動曝露解析表" 1
3       dcl sheet 1 "全持続時間等価振動エネルギ,5 分間隔振動値、最大過渡振動値、四乗暴露量値表" {
4         page 1:
5           column $69,$71,$28,$29,$30,$42,$44,$45,$46,$48,$49,$50
6           format F2,F0,F2,F2,F2,F2,F2,F2,F2,F2,F2,F2,F2,F2 2
7       }sheet
8       /*--------補正加速度を求める--------------------------------*/
9       $4 "座面 X 軸補正加速度:m/s^2" =WBD(#1)
10      $5 "座面 Y 軸補正加速度:m/s^2" =WBD(#2)
11      $6 "座面 Z 軸補正加速度:m/s^2" =WBK(#3)
12      /*--------移動実効値を求める--------------------------------*/
13      $16 "座面 X 軸補正加速度実効値:m/s^2" = RRV(1,$4)
14      $17 "座面 Y 軸補正加速度実効値:m/s^2" = RRV(1,$5)
15      $18 "座面 Z 軸補正加速度実効値:m/s^2" = RRV(1,$6)
16      /*--------計測区間ごとの実効値を求める----------------------*/
17      $28 "座面X 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBD(#1))
18      $29 "座面 Y 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBD(#2))
19      $30 "座面 Z 軸補正加速度実効値:m/s^2" = EFF(300/PRD(),WBK(#3))
20      $43 "座面軸合成:m/s^2" = SQR((1.4*$16)^2+(1.4*$17)^2+$18^2)
21      /*--------座面軸合成加速度を求める--------------------------*/
22      $42 "座面軸合成:m/s^2" = SQR((1.4*$28)^2+(1.4*$29)^2+$30^2)
23      /*--------最大過渡振動値を求める----------------------------*/
24      $44 "MTVV_X:m/s^2" = MAX(300/PRD(),$16)
25      $45 "MTVV_Y:m/s^2" = MAX(300/PRD(),$17)
26      $46 "MTVV_Z:m/s^2" = MAX(300/PRD(),$18)
27      /*---------VDV を求める-------------------------------------*/
28      $48 "座面 X 軸 VDV:m/s^1.75" = SUM(300/PRD(),($4^4)*PRD())^0.25
29      $49 "座面 Y 軸 VDV:m/s^1.75" = SUM(300/PRD(),($5^4)*PRD())^0.25
30      $50 "座面 Z 軸 VDV:m/s^1.75" = SUM(300/PRD(),($6^4)*PRD())^0.25
31      /*--------時間率頻度を求める-------------------------------*/
32      $66 "暴露時間:sec" = ERC(100,199,TRC(0.1,200,$43)*PRD())
33      $67 "等価セル中央値:m/s^2" = ERC(100,199,CNV(1,0.1,200,1))
34      $68 "暴露量:m/s^2.sec" = $67*$66
35      $69 "全持続時間等価振動エネルギ:m/s^2" = SQR(SUM($67^2*$66)/SUM($66))
36      write ch_column 1: $69
37      /*--------シート出力用 5 分間隔配列-----------------------*/
38      $71 "経過時間:min" = ACC(DAG(LEN($50),5))
39      write ch_column 1: $71,$28,$29,$30,$42,$44,$45,$46,$48,$49,$50
40      end
```

<Archi_1 Script 記述構文の説明>

3 行目～7 行目：結果シート宣言します。
9 行目～11 行目：事前処理として振動曝露フィルタ処理を行い、各軸の加速度を補正加速度に変換します。
13 行目～15 行目：事前処理として各軸の補正加速度から移動実効値演算を行い、補正加速度移動実効値を求めます。
16 行目： 事前処理として各軸の補正加速度移動実効値から軸合成移動加速度を求めます。
17 行目～19 行目：5 分間ごとに補正加速度から実効値求めます。
20 行目： 各軸の実効値から合成実効値を求めます。
24 行目～26 行目：各軸の補正加速度移動実効値から 5 分間ごとに最大過渡振動値（最大値）を求めます。
28 行目～30 行目：各軸の補正加速度から四乗暴露量値を求めます。
32 行目～35 行目：全持続時間等価振動エネルギを求めます。
32 行目： 軸合成移動実効値を時間率頻度解析を行い、0.1m/s^2 ごとの時間を求めます。
33 行目： 時間率頻度解析のセル中央値を求めます。
34 行目： 各セルごとの時間とセル中央値から曝露量を求めます。
35 行目： 全時間持続等価振動エネルギを求めます。
38 行目： 結果シート書き込み用に経過時間列を生成します。
39 行目： 演算結果を結果シートに書き込みます。

<Archi_1 Script 実行結果シート表示>


Untitled --- Archi_1 Sheet

Title 全持続時間等価振動エネルギ,5分間隔振動値、最大過渡振動値,四乗暴露量値

SHEET_SAVE

Page1:

時間等価振動エネルギ(	経過時間(min)	正加速度実効	正加速度実効	正加速度実効	面軸合成(m/s	fTVV_X(m/s^2	fTVV_Y(m/s^2	fTVV_Z(m/s^2	X軸VDV(m/s^	Y軸VDV(m/s^	Z軸VDV(m/s^
0.37	5	0.06	0.05	0.05	0.12	0.45	0.43	0.18	0.60	0.55	0.29
	10	0.15	0.14	0.08	0.29	0.73	0.39	0.39	1.14	0.90	0.58
	15	0.17	0.16	0.10	0.34	0.60	0.33	0.30	1.06	0.88	0.59
	20	0.21	0.18	0.10	0.40	1.07	0.55	0.41	1.65	1.09	0.62
	25	0.15	0.17	0.09	0.32	0.40	0.34	0.23	0.86	0.93	0.48
	30	0.16	0.16	0.09	0.33	0.67	0.40	0.21	1.02	0.92	0.52
	35	0.16	0.15	0.09	0.32	0.47	0.35	0.35	0.94	0.87	0.54
	40	0.22	0.19	0.13	0.43	0.79	0.50	0.49	1.57	1.14	0.87
	45	0.17	0.17	0.10	0.35	0.72	0.47	0.34	1.10	1.01	0.59
	50	0.18	0.16	0.11	0.36	0.71	0.40	0.42	1.22	0.98	0.70
	55	0.22	0.19	0.13	0.43	1.63	0.84	0.56	2.22	1.28	0.94
	60	0.25	0.21	0.12	0.47	1.29	0.82	0.55	2.01	1.36	0.83
	65	0.20	0.19	0.12	0.41	0.73	0.54	0.42	1.30	1.12	0.75
	70	0.21	0.16	0.13	0.39	0.73	0.58	0.37	1.38	1.00	0.78
	75	0.17	0.13	0.11	0.32	0.73	0.29	0.35	1.11	0.75	0.65
	80	0.19	0.17	0.12	0.38	0.63	0.68	0.42	1.22	1.13	0.76
	85	0.20	0.15	0.10	0.36	0.83	0.39	0.34	1.26	0.87	0.60
	90	0.24	0.20	0.14	0.46	0.83	0.69	0.47	1.51	1.27	0.86
	95	0.07	0.06	0.05	0.14	0.27	0.23	0.22	0.50	0.38	0.34
	100	0.11	0.11	0.08	0.23	0.36	0.38	0.30	0.72	0.71	0.52
	105	0.23	0.20	0.15	0.45	0.75	0.64	0.53	1.54	1.25	0.91
	110	0.22	0.19	0.14	0.42	0.87	0.86	0.61	1.49	1.30	0.94
	115	0.25	0.25	0.14	0.51	1.01	0.83	0.47	1.73	1.63	0.90

記述例20. 2. 補正加速度実効値、クレストファクタをグラフ表示する

PcWaveForm 波形 Window で解析範囲を指定し、解析範囲の補正加速度実効値及びクレストファクタをグラフ表示する。

<Archi_1 Script 記述例>

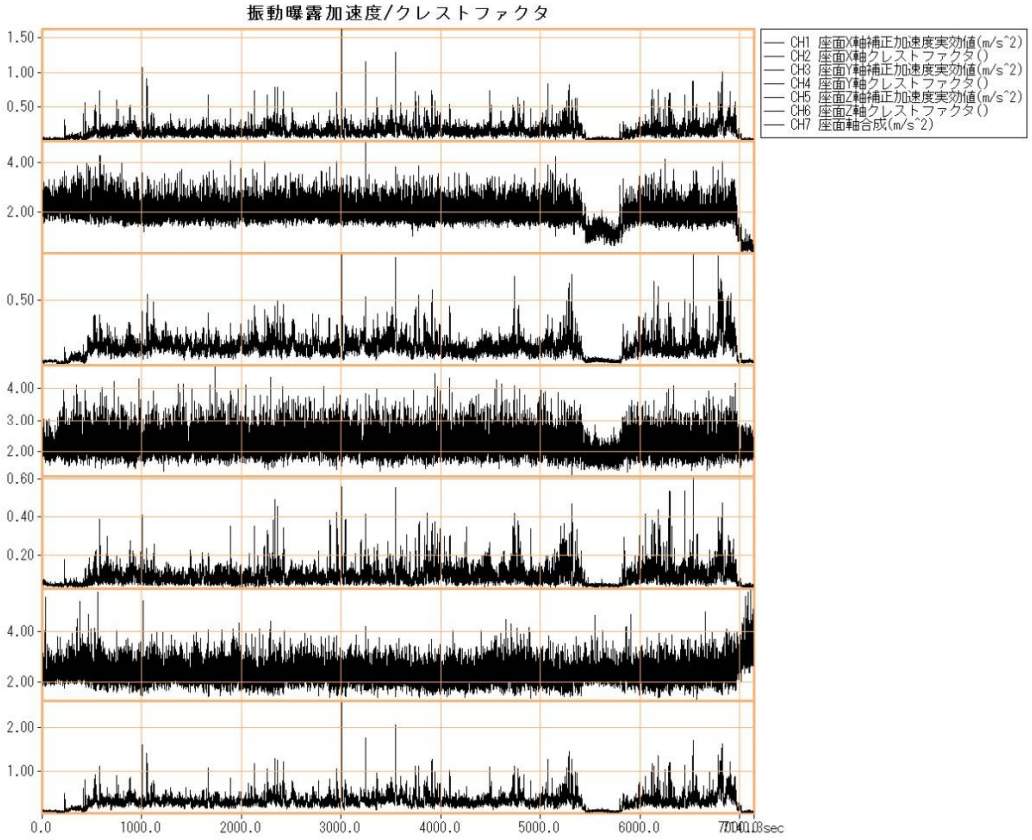
```
1       dcl menu_label "WBV_Graph1" 1
2       /*---------補正加速度を求める---------------------------------*/
3       $4 "座面 X 軸補正加速度:m/s^2" =WBD(#1)
4       $5 "座面 Y 軸補正加速度:m/s^2" =WBD(#2)
5       $6 "座面 Z 軸補正加速度:m/s^2" =WBK(#3)
6       /*---------移動実効値を求める---------------------------------*/
7       $100 "座面 X軸補正加速度実効値:m/s^2" = RRV(1,$4)
8       $102 "座面 Y軸補正加速度実効値:m/s^2" = RRV(1,$5)
9       $104 "座面 Z軸補正加速度実効値:m/s^2" = RRV(1,$6)
10      $106 "座面軸合成:m/s^2" = SQR((1.4*$100)^2+(1.4*$102)^2+$104^2)
11      /*----------クレストファクタを求める----------*/
12      $101 "座面 X 軸クレストファクタ:" =CFT(1,$4)
13      $103 "座面 Y 軸クレストファクタ:" =CFT(1,$5)
14      $105 "座面 Z 軸クレストファクタ:" =CFT(1,$6)
15      /*----------グラフ描画サブルーチン呼び出し----------*/
16      $8 "表示 ch 番号:" =ACC(DAG(7,1))
17      call proc 波形グラフ 2 7,100,$8,"振動曝露加速度/クレストファクタ"
19      end
```

※ 呼び出し先外部演算処理ブロックの記述を省略しています。実行する場合、end 行以下に外部演算処理ブロックの記述が必要です。
※ 外部演算処理ブロック「波形グラフ 2」は 11.9.3.「異なる単位の複数チャネルを、グラフ Y 軸を分割して描画する方法」に記述しています。

<Archi_1 Script 記述構文の説明>

3 行目～6 行目：各軸の加速度を補正フィルタ処理し補正加速度に変換する。
7 行目～10 行目：各軸の補正加速度から移動実効値に変換し、合成加速度実効値を求める。
12 行目～14 行目：各軸のクレストファクタを求める。
16 行目：　外部演算処理ブロックへの表示チャネル番号引数を生成する。
17 行目：　外部演算処理ブロックを呼び出しグラフ表示する

<Archi_1 Script 実行結果グラフ>



記述例20. 3. 暴露時間をグラフに表示する

PcWaveForm 波形 Window で解析範囲を指定し、解析範囲の全持続時間等価振動エネルギをもとめ、X 軸に暴露持続時間、Y 軸に暴露時間での振動の大きさをとった両対数グラフを表示します。

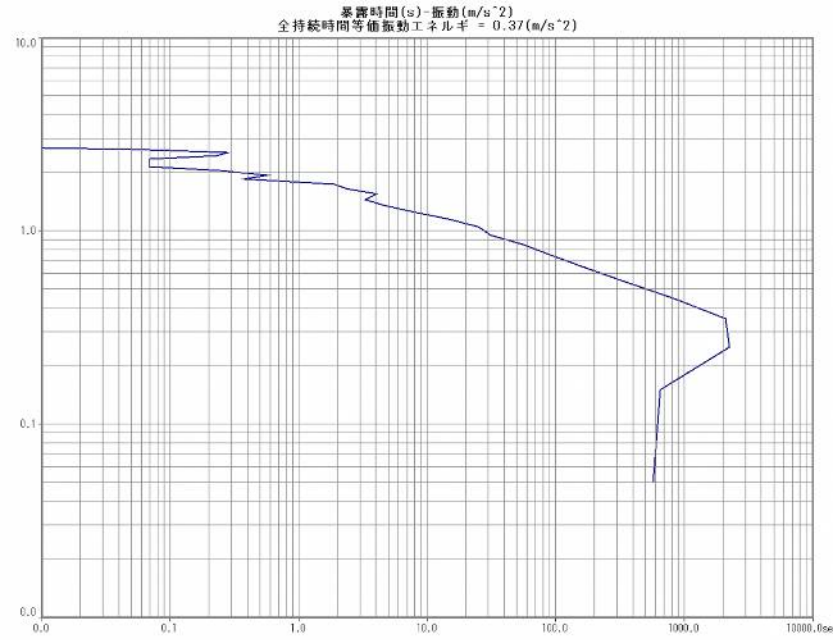
<Archi_1 Script 記述例>

```
1       /*--- WBV_test3 */
2       /*---------移動実効値を求める----------------------------------------*/
3       $16 "座面 X 軸補正加速度実効値:m/s^2" = RRV(1,WBD(#1))
4       $17 "座面 Y 軸補正加速度実効値:m/s^2" = RRV(1,WBD(#2))
5       $18 "座面 Z 軸補正加速度実効値:m/s^2" = RRV(1,WBK(#3))
6       $43 "座面軸合成:m/s^2" = SQR((1.4*$16)^2+(1.4*$17)^2+$18^2)
7       /*--------時間率頻度を求める----------------------------------------*/
8       $66 "暴露時間:sec" = ERC(100,199,TRC(0.1,200,$43)*PRD())
9       $67 "等価セル中央値:m/s^2" = ERC(100,199,CNV(1,0.1,200,1))
10      $68 "暴露量時間:m/s.sec" = $66*$67
11      $69 "全持続時間等価振動エネルギ:" = SQR(SUM($67^2*$66)/SUM($66))
12      /* ---------グラフ描画　&　グラフ格納 ---------------------------------*/
13      $72 "X 軸最小値:" =RVS(EQU(MIN($66),0),MIN($66),0.01)
14      $73 "X 軸最大値:" =MAX($66)
15      assign &1 = "全持続時間等価振動エネルギ = "|$69(F2)|"(m/s^2)"
16      def graph_id @1 "暴露時間(s)-振動(m/s^2)" &1
17      def graph_y_axis @1 0,0 F1 5 log
18      def graph_x_axis @1 $72,$73 F1 "sec" log
19      plot @1 $66 $67
20      def file_id %1 "WBV_graph2" grp
21      save plot %1 @1 $66 $67
22      end
```

<Archi_1 Script 記述構文の説明>

3 行目～5 行目：各軸の加速度に補正フィルタ処理を行い、移動実効値を求める。
6 行目：　各軸の移動加速度実効値から軸合成移動加速度実効値を求める。
8 行目：　合成加速度移動実効値をセルサイズ 0.1m/s^2として時間率頻度解析を行う。
9 行目：　セル中央値を求める
10 行目：セルごとに曝露量･時間を求める
11 行目：全持続時間等価振動エネルギを求める。
13 行目～21 行目：縦軸振動値、横軸時間とした時間率頻度解析結果を両対数グラフで表示する。
13 行目：横軸グラフ最小値を求める。
14 行目：横軸グラフ最大値を求める。
19 行目：グラフ描画する

<Archi_1 Script 実行結果グラフ>



21. 演算関数を使用して周波数解析を行う

PcWaveForm 演算関数の 1/3 オクターブ解析関連関数及び FFT 解析関連関数を使用して周波数解析を行う方法について説明します。

21. 1. 1/3 オクターブ解析

1/3 オクターブ解析とは 1kHz を中心にオクターブ(周波数で 2 倍または 1/2 倍)ごとに 3 分割したバンドパスフィルタを通過させ、バンドごとに通過した波形の実効値を表示する解析を意味します。

＜1/3 オクターブ解析の概略流れ＞

21. 1. 1. R10 系列 index を取得する

【RTI 関数】を使用します。1/3 オクターブ解析を行う為には解析周波数範囲を R10 系列 Index に変換して 1/3 オクターブ フィルタ関数に与える必要があり、解析周波数範囲の R10 系列 Index を取得する為に使用します。
※ R10 系列とは ISO 266:1997[5]に規定する 1/3 オクターブ系列の Index を意味します

文法：

格納先 = RTI(最小解析周波数,最大解析周波数,フラグ)

引数：

【最小解析周波数】＜必須＞ 解析する最小周波数を単位"Hz"で記述します。

【最大周波数】＜必須＞ 解析する最高周波数を単位"Hz"で記述します。

【フラグ】＜省略可＞ 記述された最高周波数が現在のサンプリング周波数から採り得る場合に最 高周波数を現在設定されているサンプリング周波数を参照して自動修正する か、サンプリング周波数を参照しないかのフラグを意味します。
0 または記述省略の時：現在のサンプリング周波数を参照しない。
0 以外を記述した時：現在のサンプリング周波数を参照し自動修正する。
※ 1/3 オクターブ解析する場合、上限のバンドはサンプリング周波数により規制されますので必ず 0 以外を記述し、サンプリング周波数を参照すると指定します。

記述例：

10Hz～7kHz までを解析対象とする場合
$1 = RTI(10,7000,1)

関数の戻り数列は、最小周波数から最高周波数を含むベース 2 での Index 列となり ま
R10 系列では、1000Hz を基準 index0 としてオクターブを 1/3 に分割し、それぞれのバンドの中心周波数位置の Index を意味し、1kHz より高い周波数は正、1kHz より低い周波数は負となります。

◆周波数と Index の関係を示します。

Index = 3*Ln(Frequency/1000)/Ln(2)

RTI関数の戻り数列

R10_INDEX	R10周波数(Hz)	
-20	9.8431	
-19	12.4016	Octave
-18	15.6250	
-17	19.6863	Octave
-16	24.8031	
-15	31.2500	Octave
-14	39.3725	
-13	49.6063	Octave
-12	62.5000	
-11	78.7451	Octave
-10	99.2126	
-9	125.0000	Octave
-8	157.4901	
-7	198.4251	Octave
-6	250.0000	
-5	314.9803	Octave
-4	396.8503	
-3	500.0000	Octave
-2	629.9605	
-1	793.7005	Octave
0	1000.0000	
1	1259.9210	Octave
2	1587.4011	
3	2000.0000	Octave
4	2519.8421	
5	3174.8021	Octave
6	4000.0000	
7	5039.6842	Octave
8	6349.6042	
9	8000.0000	

◆RTI 関数で生成されたバンド数の求め方

$1 = RTI(10,7000,1)
$2 = LEN($1)

$2 に生成された R10 系列 Index 数列の個数(バンド数)が格納されます。

$2 に格納された R10 系列 Index は指定された最小周波数と最大周波数を必ず含むとは限りません。最大周波数は設定されているサンプリング周期が参照され、採り得る最大周波数となります。

21. 1. 2. 1/3 オクターブ・フィルタを掛る

【OBF 関数】を使用します。OBF 関数は指定された R10 系列 Index に相当する 1/3 オクターブフィルタを掛けます。なお、実効値に変換するか否か及び実効値変換に際しての積分時定数は引数で与えます。

文法：

格納先 = OBF(R10 系列 Index 数列,実効値変換積分時定数,解析対象数列)

引数：

【R10 系列 Index 数列】＜必須＞

解析対象バンド R10 系列 Index を記述します。要素並びは昇順の必要があります。また、Index は連続している必要はなく特定のバンドのみ指定することができます。

【実効値積分時定数】＜省略可＞

0 または記述省略の場合：解析対象チャネルの全ての範囲を時定数と見なします。この場合、戻りデータ個数はバンドごとに 1 個となります。-1 と記述した場合：実効値変換を行いません。戻りデータ個数はバンドごとに解析対象数列と同じ個数となります。

0<と記述した場合：移動実効値変換時定数を記述します。戻りデータ個数はバンドごとに積分時定数分少ない数列 となります。

【解析対象数列】＜必須＞

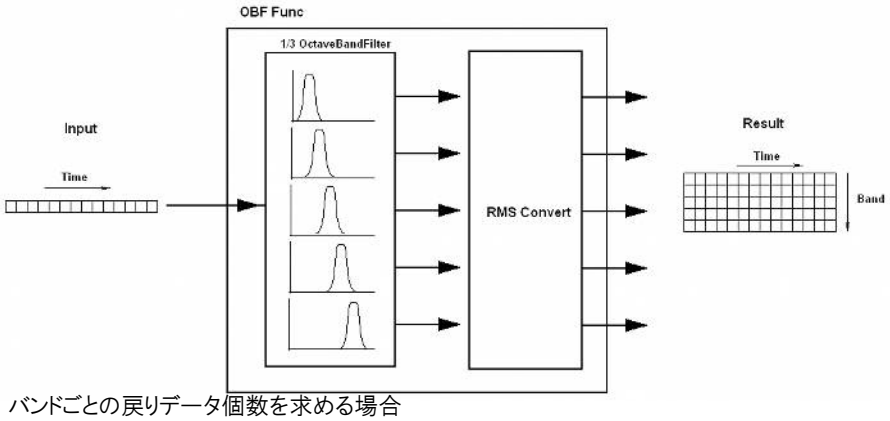
演算対象数列を記述します。

記述例：

$1 は解析対象 R10 系列 Index、$3 は解析対象チャネルとし、全範囲を実効値変換する場合

```
$4 = OBF($1,$3)
```

OBF 関数の戻り数列は、最小バンドから順に並び、仮想 2 次元数列となります。 仮想行方向はバンド順に並び、仮想列方向は時間軸となります。但し、時定数に 0 記述した場合或いは記述省略した場合
はバンドごとに実効値 1 個のデータとなりますので仮想行数は 1 列からなる一次元数列となります。
時定数に-1 を記述した場合は実効値変換せず、各バンド通過後の波形が戻ります。従って、仮想列方向は時間となり、列数は解析対象データ個数に等しくなります。なお、各仮想行方向はバンドごとの行となり、戻りデータ数は解析データ数×バンド数となり戻りデータ個数は増大します。従って、着目しているバンドのみ解析し、時間軸方向に着目したバンド範囲の波形がどの様になっているかを知りたい時などに-1 を指定します。 実効値変換の時定数を記述した場合は、記述された時定数で移動実効値演算を行います。戻りデータ数は（解析データ 数-時定数/サンプリング周波数）×バンド数となり戻りデータ数は増大します。なお、何れもメモリに余裕がる場合は全ての バンドを同時に処理しても問題ありません。つまり、時定数の 0 以外を指定した場合は、仮想 2 次元配列となりますので、受 け取った後で行ごとに分解すれば、バンドごとの時間経過となります。



◆ バンドごとの戻りデータ個数を求める場合

```
$4 = OBF($1,1.0,$3)              /* 1/3 オクターブフィルタ演算*/
$5 = LEN($4)/LEN($1)             /* バンドごとデータ数演算*/
```

$5 にバンドごとのデータ個数が格納されます。

◆ バンドごとに戻り数列を切り離す場合（仮想 2 次元行列から行の抽出）

```
$4 = OBF($1,1.0,$3)      /* 1/3 オクターブフィルタ演算*/
$5 = LEN($4)/LEN($1)   /* バンドごとデータ数演算*/
$6 = 0                   /* LOOP カウンタを 0に初期化します*/
$2 = LEN($1)             /* 解析バンド数を求めます*/
repeat_case $6 < $2
 proc separate{
     $7 = $6+100       /* 格納先チャネル番号を生成します、ここでは$100 からとしています*/
    $($7) = ERC($6*$5,$6*$5+$5-1,$4)   /* バンドごとデータ切り出し(行ごとの分離)*/
    $6 = $6+1                 /* LOOP カウンタ Up
  }separate
```

仮想 2 次元行列の行ごとに指定した R10 系列 Index の先頭から対応したバンドの結果となります。

21. 1. 3. R10 系列 Index を公称バンド中心周波数に変換する

【RTF 関数】を使用します。1/3 オクターブバンドの中心周波数は、前述の RTI 関数実行例で示す様に、小数点以下を含む切りの悪い周波数となっています。この周波数を完全 BASE2 周波数と呼称しますが、グラフ等の周波数軸目盛値として不都合な為、公称周波数に変換します。この関数はグラフ表示する際に使用します。

文法：

格納先 = RTF(R10 系列 Index 数列)

引数：

【R10 系列 Index 数列】＜必須＞

公称周波数に変換する R10 系列 Index を記述します。

記述例：

$1 に格納されている R10 系列 Index を公称周波数に変換する場合

$1 = RTI(10,7000,1)

$3 = RTF($1)

$3 に$1 で与えたR10 系列 Index から公称中心周波数に変換した数列が格納されます。戻り結果を右表に示します。

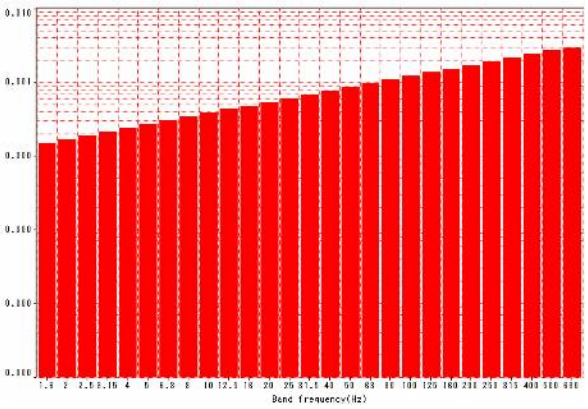
RTFの引数　　RTFの戻り数列

R10_INDEX	R10周波数(Hz)	公称周波数(Hz)
-20	9.8431	10.0
-19	12.4016	12.5
-18	15.6250	16.0
-17	19.6863	20.0
-16	24.8031	25.0
-15	31.2500	31.5
-14	39.3725	40.0
-13	49.6063	50.0
-12	62.5000	63.0
-11	78.7451	80.0
-10	99.2126	100.0
-9	125.0000	125.0
-8	157.4901	160.0
-7	198.4251	200.0
-6	250.0000	250.0
-5	314.9803	315.0
-4	396.8503	400.0
-3	500.0000	500.0
-2	629.9605	630.0
-1	793.7005	800.0
0	1000.0000	1000.0
1	1259.9210	1250.0
2	1587.4011	1600.0
3	2000.0000	2000.0
4	2519.8421	2500.0
5	3174.8021	3150.0
6	4000.0000	4000.0
7	5039.6842	5000.0
8	6349.6042	6300.0
9	8000.0000	8000.0

21. 1. 4. 1/3 オクターブ解析の留意点

フラットな周波数特性を持つ場合の 1/3 オクターブフィル タを通した結果はフラットにはなりません。

FFT 解析では、周波数ごとのスペクトラム解析が基本(実 際には Δf 間隔)の為、はグラフ上でもフラットとなりますが、1/3 オクターブ解析では、個々のバンドパスフィルタ通過 の実効値を表示する為、周波数が上昇するごとに 1/3 オ クターブフィルタの通過帯域が広がる事に寄り、周波数の 上昇に伴い表示値も大きくなりフラットにはなりません。 1/3 オクターブ解析結果は R10 系列周波数軸で等分割 される為(対数尺上同じ幅に見えますので誤解する場合 があります。



21. 2. FFT 解析

21. 2. 1. 時間軸データからフーリエ変換を行い周波数軸データに変換する

【FFT 関数】を使用します。フーリエ変換に当たって窓関数の指定、平均化処理等の指定ができます。なお、戻り値は直交座標系複素数(実数部、虚数部交互並び)で戻りますが、平均化スペクトラム、パワースペクトラム、パワースペクトラム密度解析では、虚数部は全て 0 として戻ります。

文法：

格納先 = FFT(FFT 点数,窓関数コード,解析コード,オーバラップ率,解析対象数列)

引数：

【FFT 点数】＜必須＞

FFT 点数を記述します。記述範囲は 256～32768 となります。但し、実際に使用される FFT 点数は 2^INT(log2(k))として 2 のべき乗で割り切れる値に整合化されます。

【窓関数コード】＜必須＞＜即値/数値属性参照チャネル/演算式＞

窓関数タイプをコードで記述します。

コード	窓関数形式
0	なし
1	ハミング
2	ハニング

	3	ブラックマン	
	4	カイザー・ベッセル	
	5	フラット・トップ	

【解析コード】＜必須＞ 解析タイプを
コードで記述します。

コード	解析タイプ
0	スペクトラム
1	パワースペクトラム
2	パワースペクトラム密度

【オーバラップ率】＜必須＞
オーバラップ率を 0<=%単位<=100 で記述します。小数点以下を含む場合は切り捨てられて参照されます。尚、100 以上が記述された場合は特別な意味を持ち平均化演算を行わず解析算対象数列の先頭からFFT 点数で記述したデータ個数まで 1 回のフーリエ変換となります。
【解析対象数列】＜必須＞
解析対象チャネルを記述します。解析対象数列の要素数は FFT 点数で設定した個数以上が必要です。

FFT 関数の戻り数列は index 順に実数,虚数,実数虚数と交互に並ぶ直交座標系複素数数列となり、要素数は与えた FFT 点数と同じ個数となります。なお、パワースペクトラム及びパワースペクトラム密度を指定した場合及びスペクトラムで平均化処理を行った場合は、虚数部はゼロとなります。

実数部が偶数 Index 番号に、虚数部が Index 番号に奇数に格納され、先頭の 2 個は直流分となります。
なお、直流分の虚数は必ず 0 となります。

記述例：
収録チャネル#1 を FFT 解析する場合

```
$1 = FFT(2048,5,1,80,#1)   /* FFT 点数:2048,窓関数:フラットトップ、パワースペクトラム,オーバラップ率:80%*/
$2 = SEP(0,1,$1)           /* パワースペクトラム数列の抽出*/
$3 = FFT(16384,0,0,100,#1)  /* FFT 点数:16384,窓関数:なし、スペクトラム,FFT 回数:1回*/
$4 = SEP(0,1,$3)           /* 実数部数列の抽出*/
$5 = SEP(1,1,$3)           /* 虚数部数列の抽出*/
```

$2 には、平均化パワースペクトラム解析結果が格納されます。又、$4 にはスペクトラム解析結果の実数部が、$5 には虚数部が格納されます。虚数部が意味を持つ場合はスペクトラム解析で FFT 解析 1 回と指定した場合のみとなります。他の場合は虚数部の値は全て 0と戻ります。

21. 2. 1. フーリエ変換周波数列を取得する

フーリエ変換に対応した周波数列を取得する場合【FFQ 関数】を使用します。 **文法:**

結果格納先 = FFQ(戻り属性コード,FFT 解析点数)

引数:

【戻り属性コード】＜省略可＞
FFQ 関数で求める種別をコードで記述します。

戻り属性コード	戻り内容
-1	直流分を除く周波数列
0 又は記述省略	直流分を含む周波数列
1～FFT 点数-1	スペクトラム番号と見なし対応した周波数

【FFT 解析点数】＜必須＞
フーリエ変換解析点数を記述します。

関数内で行われる演算内容は以下と等価です。
戻りコード:1～FFT 点数-1 の時:スペクトラム番号対応周波数の計算
fn = Δf×スペクトラム番号 = (1/PRD()/FFT 点数)×スペクトラム番号
※スペクトラム番号 1 の時はスペクトラム間隔周波数 Δf を意味します。
FFT 点数 2048 の場合、サンプリング周波数 2kHz の場合
Δf = 1/(PRD()*2048) = 0.9765625Hz
戻りコード:0 の時:直流分を含む周波数列
((ACC(DAG(FFT 点数/2,1))-1)×(1/PRD()/FFT 点数)

戻りコード-1 の時:直流分を除く周波数列
(ACC(DAG(FFT 点数/2-1,1))×(1/PRD()/FFT 点数)
FFT 点数 2048、直流分を除く周波数列を生成する場合
$1 "Frequency:Hz" = ACC(DAG(2048/2-1,1))*1/(PRD()*2048)

記述例

FFT 解析点数 2048 の場合

```
$1 = FFQ(1,2048)      /* スペクトラム番号 1 の周波数(Δf と等価)を取得*/
$2 = FFQ(0,2048)      /* 直流分を含む周波数列*/
$3 = FFQ(-1,2048)     /* 直流分を除く周波数列*/
```

21. 2. 3. 直交座標系複素数から極座標系複素数に変換する

【PCX 関数】を使用します。PCX 関数は直交座標系複素数数列(実数部、虚数部並び)を極座標複素数数列(絶対値、位相角並び)に変換します。FFT 解析結果のスペクトラム結果から絶対値と位相角を求める場合に使用します。なお、位相角はラジアン単位となります。この関数はスペクトラム解析で平均化処理を行わない場合に意味を持ちます。平均化処理を行った場合やパワースペクトラム、パワースペクトラム密度では虚数部は 0 となり、意味を持ちません。

文法

結果格納先 = PCX(FFT 結果数列)

引数

【FFT 結果数列】＜必須＞ 解析対象チャネルを記述します。解析対象チャネルは直交座標系複素数の必要があります。又、その要素数は必 ず偶数の必要があります。

演算内容は a:実数部、b:虚数部とすると絶対値 r は、$\sqrt{(a^2+b^2)}$ 角度 θ は、$\tan^{-1}(b/a)$となり、$\pm\pi$ 範囲となります。但し、a が 0 の場合、b>0 の時は $\pi/2$、b<0 の時は$-\pi/2$、b=0 時は 0 が戻ります。
結果格納先数列の並び順は Index 順に絶対値、位相角が交互に並ぶ極座標系複素数数列となり要素数は与えた解析チャネルの要素数と同じ個数になります。

記述例

収録チャネル#1 の FFT スペクトラム解析を行い振幅値と位相角を求める場合

```
$3 = FFT(16384,0,0,100,#1)  /* FFT 点数:16384,窓関数:なし、スペクトラム,FFT 回数:1回*/
$3 = PCX($1)                /*FFT 解析結果を極座標r,θ 数列に変換*/
$4 = SEP(0,1,$3)            /* 極座標数列から絶対値(振幅)数列の抽出*/
$5 = SEP(1,1,$3)            /* 極座標数列から位相角(rad)数列の抽出*/
$6 = SQR($4)                /* 振幅値数列を振幅実効値数列に変換*/
$7 = DEG($5)                /* 位相角数列の単位を rad から degに変換*/
```

21. 2. 4. 位相角をアンラップ処理する

【UNW 関数】を使用します。UNW 関数は位相角を$\pm\pi$ ラジアンから連続的に変化しているものと見なし、0～π 領域から 0～$-\pi$ 領域に達すると以降の位相角に 2π を加算し、逆に 0～$-\pi$ 領域から 0～π 領域に達すると以降の位相角に-2π を 加算し位相角表現をスムージングします。

文法：

結果格納先 = UNW(位相角ラジアン単位)

引数：

【位相角ラジアン単位】＜必須＞ 解析対象チャネルを記述します。解析対象チャネルはラジアン単位で$\pm\pi$ 表現の位相角の必要があります。例えば FFT 結果数列から分離した位相角数列などが相当します。

記述例

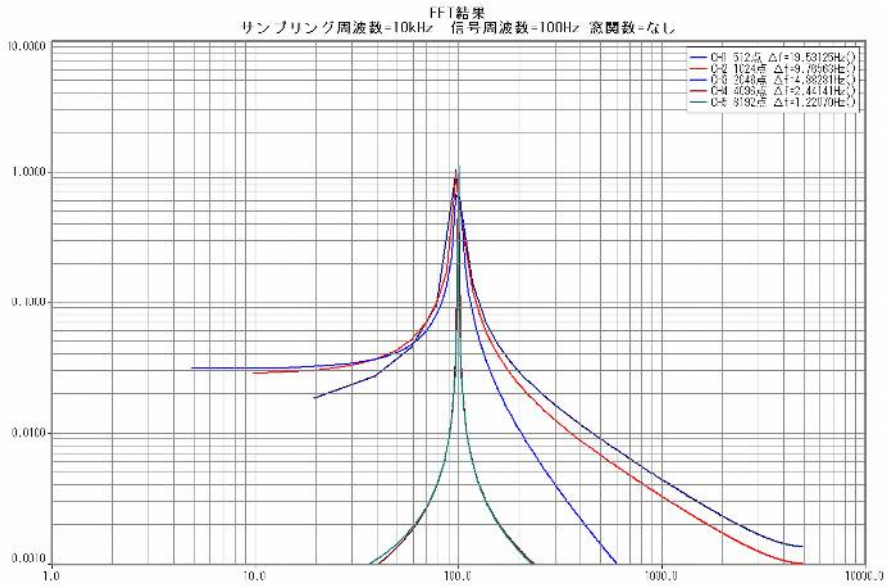
収録チャネル#1 のスペクトラム解析を行い、位相角をアンラップ処理後、単位を deg に変換する場合

```
$1 = FFT(16384,0,0,100,#1)  /* FFT 点数:16384,窓関数:なし、スペクトラム,FFT 回数:1回*/
$2 = PCX($1)                /* 極座標数列(r,θ)に変換*/
$3 "Amplitude:" = SEP(0,1,$2) /* 振幅値数列の抽出*/
$4 "Phase:deg" = DEG(UNW(SEP(1,1,$2))) /*角度(rad)数列抽出、角度アンラップ処理、角度単位変換*/
```

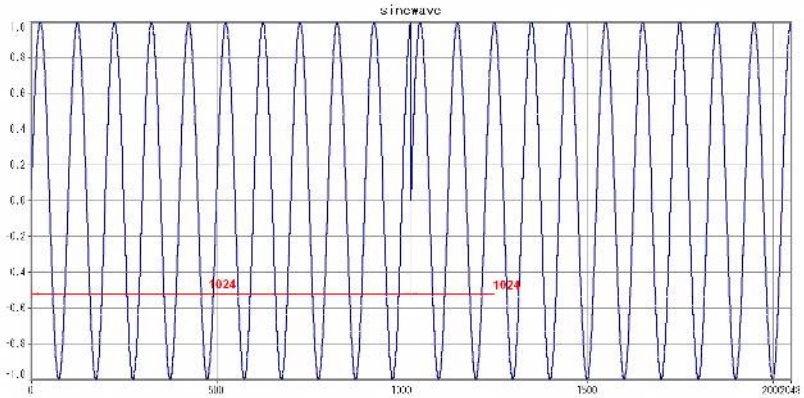
21. 2. 5. FFT 解析の留意点

FFT 解析する場合、設定する FFT 点数の違いと窓関数の違いなどにより結果が大きく異なります。

100Hz の単一正弦波を入力信号として 10kHz でサンプリングした場合の FFT 点数の違いによりどの程度異なるかをグラフで示します。

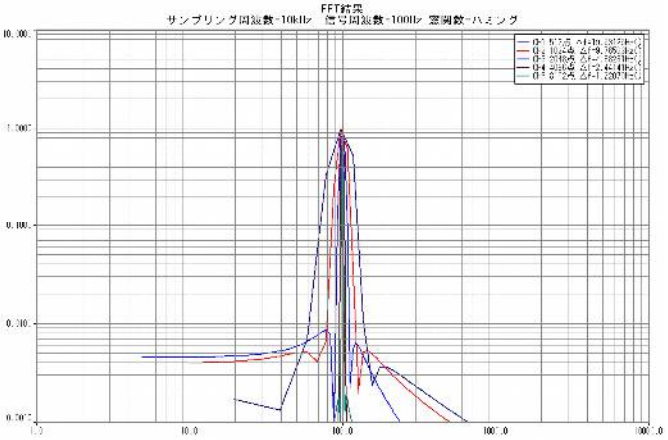


FFT 点数を増加させると、Δf が狭くなり分解能が良くなることが判ります。ピークは 100Hz を示しますが、他の周波数成分もあたかも含まれているかの様な結果となります。Δf ごとにスペクトラムが得られますが、途中の周波数スペクトラム文が漏れる事と、10kHz サンプリングで 100Hz の正弦波の連続信号を FFT した場合 FFT 点数は 2 のべき乗ですので 100Hz の周期の途中までが繰り返されているとして解析される事に起因しています。例えば 1024 点で解析した場合、次の様な継ぎ目のある波形として解析されます。

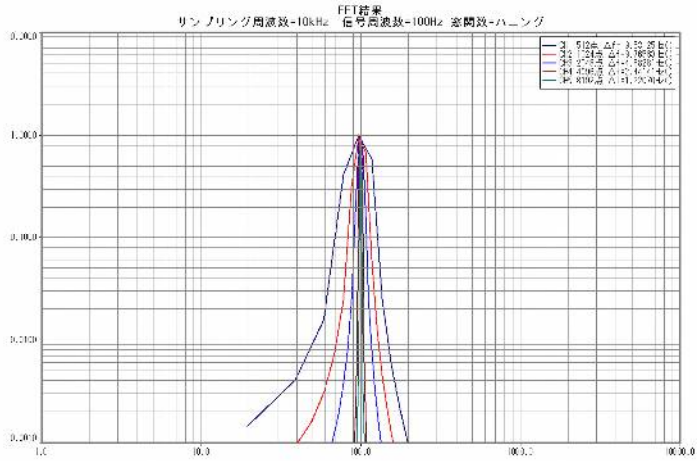


この誤差を軽減する方法として窓関数を設定して行いますが使用する窓関数により違いがあります。

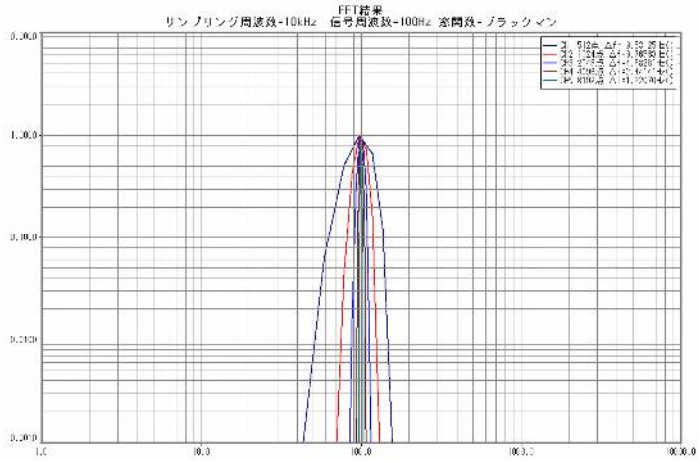
ハニング窓関数の場合



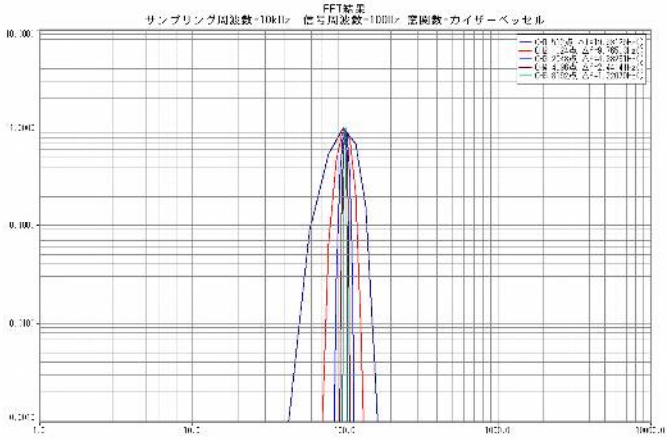
ハミング窓関数の場合



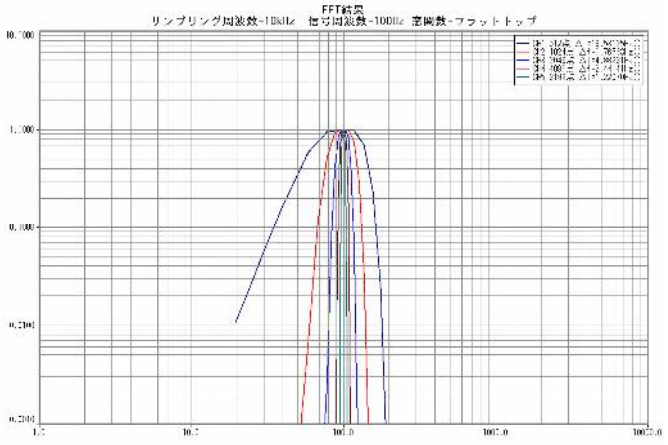
ブラックマン窓関数の場合



カイザーベッセル窓関数の場合



フラットトップ窓関数の場合



記述例21. 1. 1/3 オクターブバンドパスフィルタ OBF 関数の特性を求める

インパルス波形を生成し、生成したインパルス波形を 1/3 オクターブフィルタ関数(OBF 関数)を用いてバンドごとのインパルス応答を求めて FFT 解析を行いバンドごとの周波数特性を求めます。

<Archi_1 Script 記述例>

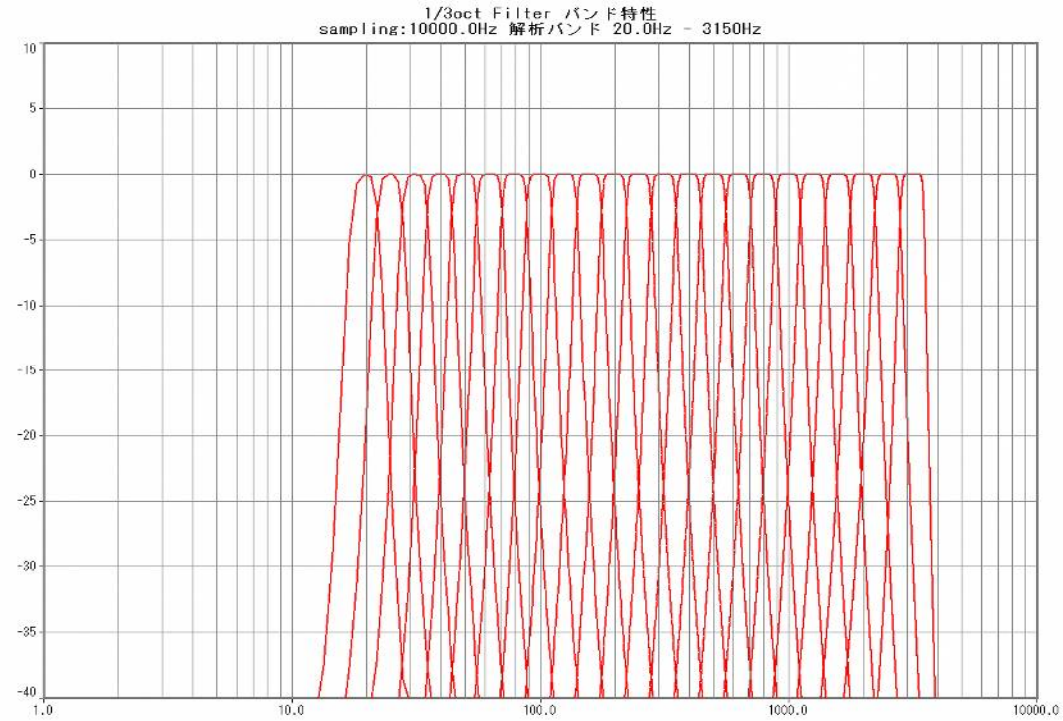
```
1       /* 1/3oct filter test----------*/
2       dcl menu_label "1/3oct フィルタ特性" 1
3       def sampl_period 100e-6 "sec"              /* サンプリング周期定義*/
4       $1 "Impulse:" = DAG(100000,0)              /* 試験波形数列準備*/
5       $1(5000) = 1                                /* impulse 波形生成*/
6       /*---------- 1/3oct Filter 処理----------------------------------------*/
7       $2 "R10_index:" = RTI(20,6000,1)          /* 解析周波数範囲 10Hz-8kHz R10Index */
8       $3 "1/3oct_filter:" = OBF($2,-1,$1)       /* 1/3oct フィルタ処理*/
9       $4 "num_band:" = LEN($2)                   /* 1/3oct バンド数取得*/
10      $6 "num_data:" = LEN($3)/$4                /* バンド当たりデータ数*/
11      $7 "BandCenterFreq:Hz" = RTF($2)           /* R10 公称周波数 取得*/
12      assign &1 "公称周波数文字列:" = $7(F1)|"Hz"
13      $5 "band_counter:" = 0                     /* カウンタ初期化*/
14      repeat_case $5 < $4
15        proc separate{
16          $8 "格納先 ch 番号:" =$5+100
17          $($8) = ERC($5*$6,$5*$6+$6-1,$3)   /* バンドごとデータ切り出し*/
18          def ch_name $($8) &1($5)             /* チャネル名定義*/
19          $5 = $5+1
20        }separate
21      /*----- フィルタ通過後波形格納-------------------------------------------*/
22      def file_id %1 "1_3oct_wave" wav
23      def wav_type %1 float
24      save wave %1 $[100,$4]
25      /*--------- FFT 処理-------------------------------------------------*/
26      $14 "sampling:Hz" = 8192                          /* FFT  点数*/
27      $5 = 0
28      repeat_case $5 < $4
29        proc fft{
30          $8 = $5+100
31          $9 "FFT 結果格納先 ch 番号:" = $5+200
32          $($9) = FFT($14,1,0,80,$($8))              /* OverLap80%,FlatTop 処理*/
33          $($9) = SEP(2,1,$($9))                      /* 直流分除く絶対値切り出し*/
34          $($9) = 20*LGT($($9)/MAX($($9)))            /* dB 変換*/
35          def ch_name $($9) &1($5)                    /* チャネル名定義*/
36          $5 = $5+1
37        }fft
38      $10 "周波数:Hz" = FFQ(-1,$14)                  /* 周波数列生成*/
39      /*---- 周波数特性グラフ-------------------------------------------------*/
40      $11 "サンプリング周波数:" = 1/PRD()
41      $12 "最小中心周波数:" = $7(0)
42      $13 "最大中心周波数:" = $7(LEN($7)-1)
43      assign &2 "sub_title:" = "sampling:"|$11(F1)|"Hz 解析バンド "|$12(F1)|"Hz - "|$13(F0)|"Hz"
44      def graph_id @1 "1/3oct Filter  バンド特性"&2
45      def graph_y_axis @1 -40,10,5 F0 5
46      def graph_x_axis @1 0,0 F1 log
47      def graph_line @1 1 "#0000FF"
48      plot @1 $10 $[200,$4]
49      def file_id %2 "One_third_filer" grp
50      save plot %2   @1 $10 $[200,$4]
51      end
```

<Archi_1 Script 記述構文の説明>

3 行目： 生成波形のサンプリング周期を定義します。
4 行目～5 行目：インパルス波形を生成します。
7 行目： 1/3 オクターブ解析周波数範囲を 20Hz～6kHz として R10 系列 Idexn を求めます。
8 行目： 1/3 オクターブフィルタ演算を実効値変換無しとして行います。
9 行目： 1/3 オクターブバンド数を求めます。
10 行目：バンドごとのデータ個数を求めます。(例ではインパルス波形数列の個数と一致します)

11 行目: バンドごとの公称中心周波数を求めます。
12 行目: バンドごと公称中心周波数をバンドごとの格納チャネルの信号名とする為、文字列に変換します。
13 行目～20 行目:1/3 オクターブフィルタ関数の戻り仮想 2 次元配列から、バンドごとのデータに分解します。
16 行目: 格納先チャネル番号を生成します。(例では$100 からバンド数分)
17 行目: 仮想 2 次元配列からバンドごとにデータを抽出します。
18 行目: 格納先チャネルに信号名を定義します。
21 行目～24 行目:インパルス応答波形を PcWaveForm で格納します。
26 行目; FFT 解析のデータ個数を定義します。
27 行目～38 行目:バンドごとインパルス応答波形を FFT 解析します。
31 行目:バンドごと FFT 解析結果格納先チャネルを定義します。(例では＄200 からバンド数分)
32 行目:オーバラップ率 80%でバンドごとにスペクトラム解析を行います。
33 行目: FFT 結果から絶対を切り出します。
34 行目; スペクトラを其々のバンド最大値を 0dB としたデシベルに変換します。
38 行目: スペクトラに対応した周波数数列を求めます。
40 行目～50 行目:周波数特性グラフの描画とグラフ格納を行います。
40 行目～43 行目:グラフの描画するサブタイトル行を生成します。
44 行目: グラフ番号を定義します。
45 行目: グラフ Y 軸を-40dB～10dB 範囲グリッド間隔 5dB で定義します。
46 行目: グラフ X 軸を Log 尺属性で定義します。
47 行目: グラフ描画線種及び線色を定義します。
48 行目: X 軸チャネル周波数数列、Y 軸チャネル周波数特性数列として全てのバンドのグラフを描画します。

<Archi_1 Script 実行結果グラフ>



※備考:FFT 点数、窓関数、オーバラップ率などは正確に周波数特性を表示する様に設定する必要があります。

記述例21. 2. 解析範囲の任意のチャネルを指定して 1/3 オクターブ解析を行う

PcWaveForm 波形 Window で解析範囲を指定し、解析範囲の任意チャネルを 0.1Hz～8kHz 範囲の 1/3 オクターブ解析を行います。実効値への変換は解析範囲を積分時定数として行い、全体の実効値は解析対象波形を実効値変換した値をグラフの再上位バンドの次に付加します。全体の実効値はバンドごとの実効値を二乗加算して√する方法も有りますが、各バンドは減衰域が隣接バンドと交差しますので解析対象波形から求めた実効値より少し大きな値となる為です。

<Archi_1 Script 記述例>

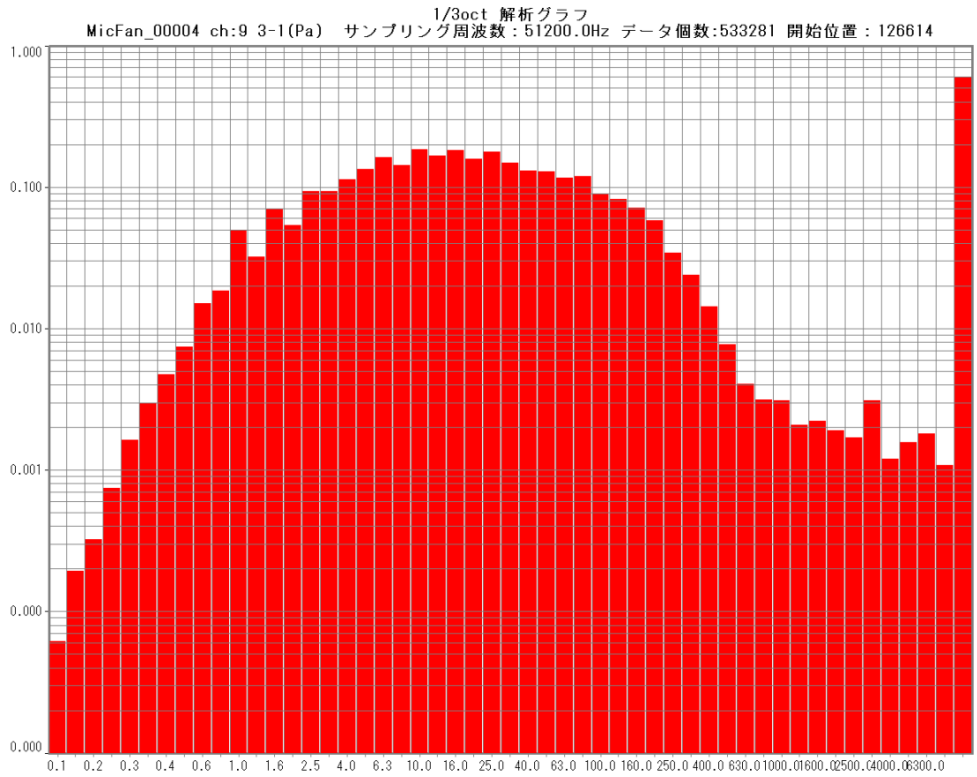
```
1       /* ---- Script メニューラベル宣言 ----------------------------------------*/
2       dcl menu_label "1/3octave Analisys" 1
3       /*------解析チャネル選択-------------------------------------------------*/
4       $8 "chNo.:" = CHS()                          /* 収録チャネル番号取得*/
5       &5 "Name:" = CHNM(0)                         /* 収録信号名取得*/
6       &6 "解析チャネル:" = CUNT(0)                  /* 収録単位名取得*/
7       $9 "num_ch:" = NCH()                         /* 収録チャネル数取得*/
8       assign &5 = $8(F0)|"ch "|&5|"("|&6|")"       /* ListBox 用配列生成*/
9       assign &6 = &5(0)                            /* ListBox 表示初期値設定*/
10      get value &5:&6 1 "解析チャネルの選択"        /* 解析対象チャネル選択*/
11      &5 = CEXT(1,CFND(1,"ch",&6)-1,&6)            /* 解析対象チャネル番号切り出し*/
12      assign $10 = &5                              /* 解析対象チャネル番号*/
13      /*----- 1/3octave 解析処理 ----------------------------------------------*/
14      $1 "R10_Index:" = RTI(0.1,8000,1)            /* R10 Index 取得*/
15      $2 "1/3oct:" = OBF($1,0,#($10))              /* 1/3Octavefilter 処理*/
16      $2 = LNK($2,EFF(#($10)))                     /* OverAll 追加収録*/
17      $4 "center_freq:Hz" = RTF($1)                /* 公称周波数変換*/
18      /*------ 1/3octave 解析結果グラフ描画処理 & 格納-------------------------*/
19      &3 "解析チャネル単位:" = CUNT($10)            /* 解析 ch 単位取得*/
20      &4 "信号名:" = CHNM($10)                     /* 信号名取得*/
21      $3 "解析対象データ個数:" = LEN(#($10))       /* 解析範囲データ個数取得*/
22      $6 "解析対象開始 Index:" = STA()             /* 解析開始 Index 取得*/
23      $7 "サンプリング周波数:" = 1/PRD()           /* サンプリング周波数取得*/
24      &2 "GraphSubTitle:" = CFNM()                 /* 解析対象ファイル名取得*/
25      assign &2 = &2|" ch:"|$10(F0)|" "|&4|"("|&3|")"  サンプリング周波数:"|$7(F1)
26      assign &2 = &2|"Hz データ個数:"|$3(F0)|" 開始位置:"|$6(F0)
27      $5 = ACC(DAG(LEN($4),2))-2                   /* 目盛位置 Index 生成*/
28      $4 = SEP(0,1,$4)                             /* 目盛 Grid1 本置き*/
29      assign &1 "X 軸目盛値:" =$4(F1)
30      def graph_id @1 "1/3oct 解析グラフ" &2
31      def graph_x_axis @1 0,0,1 &1,$5
32      def graph_y_axis @1 0,0 F3 5 log
33      def graph_line @1 40 "#0000FF"
34      plot @1 $2
35      def file_id %1 "one_third_Graph" grp
36      save plot %1 @1 $2
37      /*----- 1/3octave 解析対象波形グラフ描画処理 &格納-----------------------*/
38      $11 = SPB(0)                                 /* 時間軸生成*/
39      def graph_id @2 "1/3oct 解析対象波形"&2
40      def graph_aspect_ratio @2 2
41      def graph_x_axis @2 0,0 F2 "sec"
42      def graph_y_axis @2 0,0 F3 5
43      plot @2 $11 #($10)
44      def file_id %1 "one_third_Wave" grp
45      save plot %1 @2 $11 #($10)
46      end
```

<Archi_1 Script 記述構文の説明>

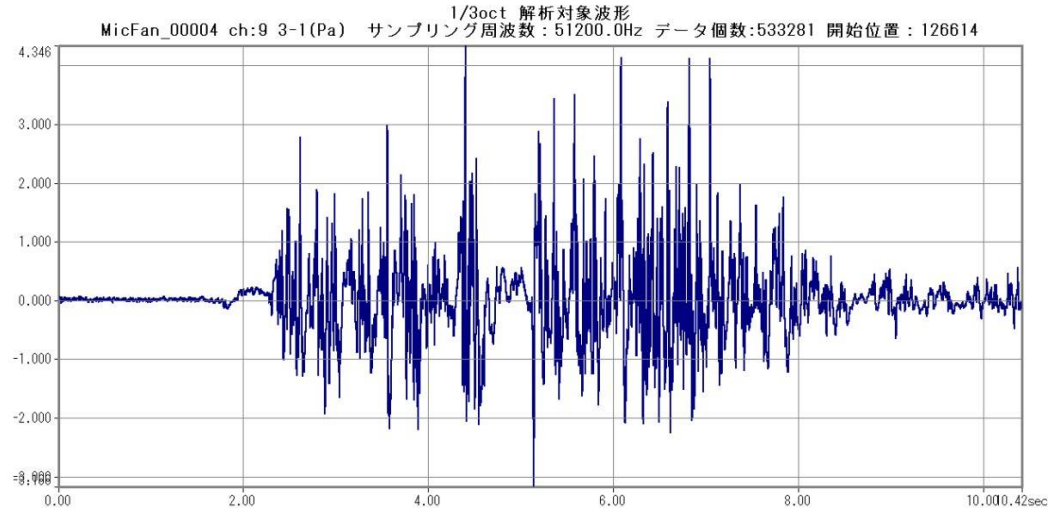
4 行目～12 行目:解析チャネルを選択させます。信号名、単位を連結させた ListBox から選択します。
14 行目: 解析周波数範囲の R10 系列 Index 数列を求めます。
15 行目: 1/3 オクターブ解析を行います。
16 行目: 1/3 オクターブ解析結果数列にオーバール実効値を追加します。
17 行目: グラフを各バンドの公称中心周波数を R10 系列Indexから求めます。
19 行目～36 行目:1/3 オクターブ解析結果グラフの描画と描画グラフの格納を行います。
19 行目～26 行目:グラフに描画するサブタイトルに必要なパラメタを取得しサブタイトルを生成します。
27 行目: グラフ X 軸目盛用に Index 数列を生成します。Index の数はバンド数+1 となります。

28 行目: X 軸目盛位置を Index ひとつおきに表示させる様に公称中心周波数を再構成します。
29 行目: X 軸目盛値は文字列指定の為、公称中心周波数を文字列に変換します。
30 行目: グラフ番号を定義します。
31 行目: X 軸自動設定、グリッド間隔 1 と表示する目盛値と表示位置を定義します。
32 行目: Y 軸自動設定、目盛表示形式小数点以下 3 桁、軸属性 log 尺として定義します。
33 行目: データ表示を棒グラフ、表示色赤として定義します。
34 行目; グラフ描画します。
35 行目～36 行目:描画したグラフを格納します。
38 行目～45 行目;解析対象波形をグラフ表示及び格納します。

<Archi_1 Script 実行結果グラフ>



<解析対象波形グラフ>



記述例21. 3. FFT 解析で 4 次バターワースフィルタ LPF 関数の周波数/位相特性を求める

演算関数-24dB/oct の遮断特性を持つローパスフィルタ関数(LPF 関数)の周波数/位相特性を求める為に、インパルス応答(インパルス波形の当該フィルタ通過後)を FFT 解析して周波数/位相特性を表示します。

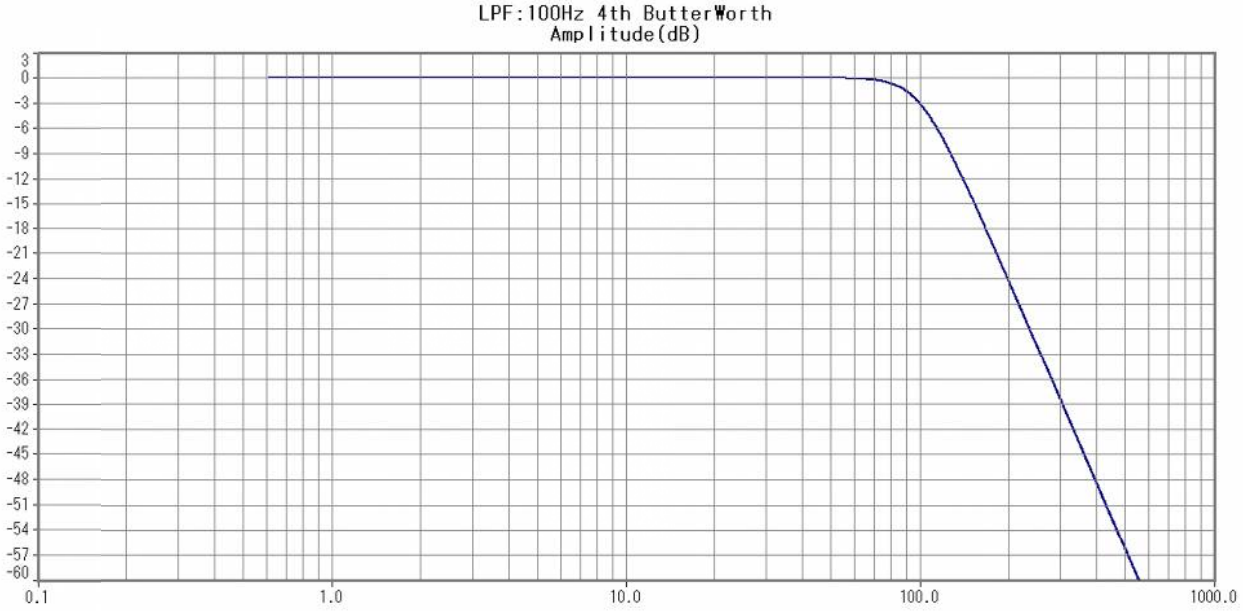
<Archi_1 Script 記述例>

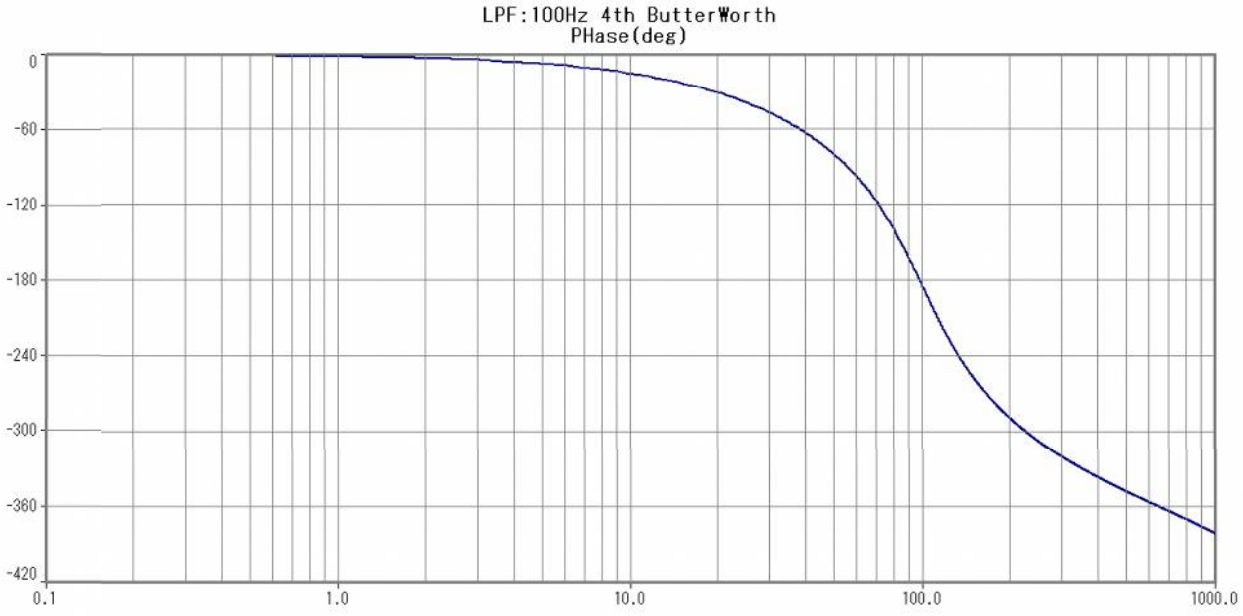
```
1       /* ---LPF 特性----------*/
2       dcl menu_label "100HzLPF 特性" 1
3       /*-----Impulse  create-------------------------------------------------*/
4       def sampl_period 100e-6 "sec"
5       $1 "ImpulseWave:" = DAG(20000,0)
6       $1(1) = 1
7       $2 "Response:" = LPF(100,$1)
8       /*------ FFT-----------------------------------------------------------*/
9       $3 "FFT_result:" = FFT(16384,0,0,100,$2) /* 点数:16384、窓関数:なし、FFT:1 回*/
10      $4 "R/θ:" = PCX($3)                      /* 極座標変換*/
11      $5 "amplitude:" = SEP(2,1,$4)            /* 振幅分離*/
12      $5 = 20*LGT($5/MAX($5))                  /* 振幅 dB 変換*/
13      $6 "phase:" = DEG(UNW(SEP(3,1,$4)))      /* 位相角分離,アンラップ処理,デグリ変換*/
14      $7 "Frequency:Hz" = FFQ(-1,16384)        /*周波数列生成*/
15      /*------Graph  Drow---------------------------------------------------*/
16      def graph_id @1 "LPF:100Hz 4th ButterWorth""Amplitude(dB)"
17      def graph_aspect_ratio @1 2
18      def graph_y_axis @1 -60,3,3 F0 5
19      def graph_x_axis @1 0,0 F1 log
20      plot @1 $7 $5
21      def graph_id @2 "LPF:100Hz 4th ButterWorth""PHase(deg)"
22      def graph_aspect_ratio @2 2
23      def graph_y_axis @2 -420,0,60 F0 5
24      def graph_x_axis @2 0.1,1e3 F1   log
25      plot @2 $7 $6
26      /*-----graph save-----------------------------------------------------*/
27      def file_id %1 "Amplitude" grp
28      save plot %1 @1 $7 $5
29      def file_id %1 "Phase" grp
30      save plot %1 @2 $7 $6
31      end
```

<Archi_1 Script 記述構文の説明>

4 行目： サンプリング周期を定義します。
5 行目～6 行目:インパルス波形を生成します。
7 行目： 4 次バターワース 100Hz ローパスフィルタのインパルス応答を求めます。
8 行目: FFT 点数 16384、窓関数なし、オーバラップ無し、スペクトラを FFT 関数で求めます。
10 行目: FFT 結果の直交座標系複素数を極座標系複素数に変換します。
11 行目: 直流を除き振幅値数列を分離抽出します。
12 行目: 振幅値を、最大値を 0dB とした dB に変換します。
13 行目: 直流を除き位相角を分離抽出し、アンラップした後、デグリ単位に変換します。
14 行目: 解析周波数列を生成します。
16 行目～20 行目:振幅特性グラフを描画します。
16 行目: グラフ番号とグラフ表題を定義します。
17 行目: グラフの縦横比を定義します。
18 行目: グラフ Y 軸を-60dB～3dB 範囲で定義します。
19 行目: グラフ X 軸を LOG 尺属性で定義します。
20 行目: グラフに X 軸周波数、Y 軸振幅として描画します。
21 行目～25 行目:位相特性グラフを描画します。
27 行目～30 行目:それぞれのグラフを格納します。

<Archi_1 Script 実行結果グラフ>


LPF:100Hz 4th ButterWorth
Amplitude(dB)
3
0
-3
-6
-9
-12
-15
-18
-21
-24
-27
-30
-33
-36
-39
-42
-45
-48
-51
-54
-57
-60
0.1
1.0
10.0
100.0
1000.0


LPF:100Hz 4th ButterWorth
PHase(deg)
0
-60
-120
-180
-240
-300
-360
-420
0.1
1.0
10.0
100.0
1000.0

記述例21. 4. 解析範囲のカレントチャネルを FFT 解析して卓越周波数と値を求める

PcWaveForm 波形表示 Window で表示しているカレントチャネルから FFT 平均化スペクトラム解析を行い、結果の振幅値数列から卓越している周波数及び振幅値を求めます。

<Archi_1 Script 記述例>

```
1       /* 卓越検索*/
2       dcl menu_label "卓越周波数検索" 1
3       /*------ 結果シート使用宣言----------------------------------*/
4       dcl sheet 1 {
5           page 1:
6           column $3,$9,$14,$15
7           format F2,F2,F2,F3 2
8       }sheet
9       /* ----- 解析条件設定----------------------------------------*/
10      $1 "解析チャネル番号:" = CCH() /* 波形 Windw カレント設定 ch 番号取得*/
11      $2 "FFT 点数:" = 4096
12      $10 "窓関数:" = 5          /*フラットトップ*/
13      $24 "平均化オーバラップ率:%" = 80
14      $3 "検索窓周波数幅:"Hz" = 5
15      $8 "検索窓幅データ個数:" = INT($3*$2*PRD())+1 /* 検索周波数幅±2.5Hz*/
16      $9 "検索窓深さ:dB" = 10
17      /*-------------------------------------------------------*/
18      case $1 > 0
19        proc exec{
20          /*--------FFT 処理---------------------------------------*/
21          $4 = FFT($2,$10,0,$24,#($1))              /* FFT 平均化スペクトラム解析*/
22          $5 "振幅:" = SEP(2,1,$4)                  /* 振幅値分離*/
23          $6 "レベル:dB" = 20*LGT($5)               /* dB 変換*/
24          $7 "周波数:Hz" = FFQ(-1,$2)               /*周波数列生成*/
25          /* ---------ピーク検索-----------------------------------*/
26          $12 "peak_index:" = PDT($9,$8,$6)  /* 振幅ピーク検索*/
27          $6 = 10^($6/20)                          /*  振幅 dB 戻し*/
28          case $12 > 0
29            proc detect{
30              $11 "卓越周波数 indx:" = SEP(0,2,$12)  /* 卓越周波数 Index 分離抽出*/
31              $13 "後方周波数 index:" = SEP(2,2,$12) /* 後方周波数 Index 分離抽出*/
32              $12 "前方周波数 index:" = SEP(1,2,$12) /* 前方周波数 Index 分離抽出*/
33              $14 "卓越周波数:Hz" = PTV($11,$7)      /* index から卓越周波数変換*/
34              $15 "卓越振幅:" = PTV($11,$6)          /* index から卓越振幅値変換*/
35              $16 "前方周波数:Hz" = PTV($12,$7)      /* index から前方周波数変換*/
36              $17 "後方周波数:Hz" = PTV($13,$7)      /* index から後方周波数変換*/
37              $20 "底振幅:" = $15*10^(-$9/20)        /* 検索窓底値演算*/
38              $19 = LEN($14)                          /* 検出卓越個数演算*/
39              assign $18 "pen_ctrl:" = 1,1,1,1,1,0  /* 卓越部分囲み Pen 制御基本生成*/
40              assign $18 = $19<$18>                  /* 卓越部分囲み Pen 制御数列生成*/
41              $21 "X_addr:" = INL($14,$16,$16,$17,$17,$14) /*卓越部分囲み X 軸アドレス数列生成*/
42              $22 "Y_addr:" = INL($15,$15,$20,$20,$15,$15) /*卓越部分囲み Y 軸アドレス数列生成*/
43              write ch_column 1: $3,$9,$14,$15
44            }detect
45          /*--------周波数グラフ描画-------------------------------------*/
46          assign &2 ="なし","ハミング","ハニング","ブラックマン","カイザーベッセル","フラットトップ"
47          assign &1 = "FFT 点数="|$2(F0)|" オーバラップ="|$24(F1)|"% 適用窓関数="|&2($10)
48          def graph_id @1 "卓越周波数検索" &1    /* グラフ番号定義*/
49          def graph_y_axis @1 0,0 F4 5 log /*グラフ Y 軸Auto_Scale 定義 */
50          def graph_x_axis @1 0,0 F1 log   /* グラフ周波数軸 Auto_Scale 定義*/
51          def graph_aspect_ratio @1 2      /*  グラフ縦横比 2 定義*/
52          case $12 > 0
53            proc 卓越窓描画{
54              def graph_line_draw @1 $21,$22,$18 0 "#0000FF" /*卓越部分囲み描画定義*/
55            }卓越窓描画
56          plot @1 $7 $6                     /* グラフ描画*/
57          def file_id %1 "卓越グラフ" grp
58          save plot %1 @1 $7 $6           /* グラフ格納*/
59          /*---- 検索対象波形グラフ描画-------------------------------*/
60          $23 = SPB(0)                      /* グラフ時間軸生成*/
```

```
61              def graph_id @2 "被解析波形" /* グラフ番号定義*/
62              def graph_x_axis @2 0,0 F3     /* グラフ時間軸 Auto_Scale 定義*/
63              def graph_y_axis @2 0,0 F3 5 /* グラフ Y 軸 Auto_Scale定義*/
64              def graph_aspect_ratio @2 2 /* グラフ縦横比 2 定義*/
65              plot @2 $23 #($1)               /* グラフ描画*/
66              def file_id %1 "被試験波形" grp
67              save plot %1 @2 $23 #($1)      /* グラフ格納*/
68            }exec
69          case $1 < 0
70            proc error{
71              disp message "解析範囲が選択されていません"
72            }error
73          end
```

<Archi_1 Script 記述構文の説明>

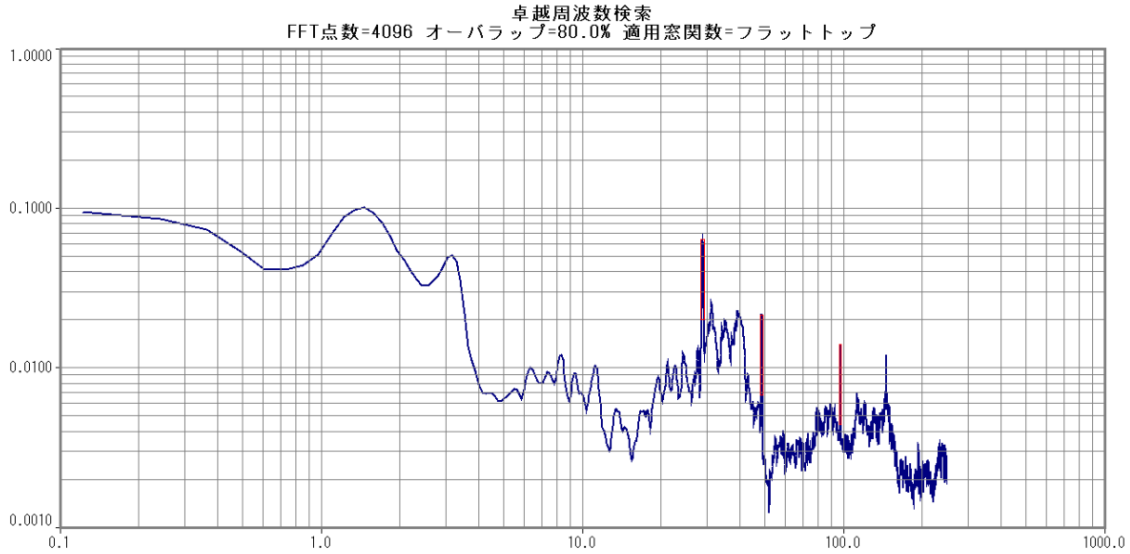
10 行目～16 行目:解析条件を設定します。
10 行目　解析チャネルを現在カレント設定されてチャネルとしてチャネル番号を取得します。
11 行目　FFT 点数を設定します。
12 行目　使用する窓関数を設定します。
13 行目　FFT 平均化オーバラップ率を設定します。
14 行目　検索窓横幅を 6Hz(±3Hz)に設定します。
15 行目　設定した検索窓横幅周波数からデータ個数を求めます。
16 行目　検索窓深さを 30dB に設定します。
21 行目～24 行目:FFT スペクトラム解析を行います。
21 行目　設定した条件で FFT 解析を行います。
22 行目　FFT 解析結果から振幅値を分離抽出します。
23 行目　卓越検索の為、一旦振幅値を dB 変換します。
24 行目　周波数列を生成します。
26 行目　卓越を設定した検索窓を使用して検索します。
27 行目　振幅値を dB から元に戻します。
30 行目～42 行目:検索した卓越が存在した場合に求めた Index から値に変換し、グラフ上での卓越部分囲み線を生成します。
30 行目: 検索結果 Index 数列から卓越周波数位置 Index を分離抽出します。
31 行目: 検索結果 Index 数列から検索窓後方 Index を分離抽出します。
32 行目: 検索結果 Index 数列から検索窓前方 Index を分離抽出します。
33 行目: 卓越周波数 Index を用いて周波数数列から卓越周波数に変換します。
34 行目: 卓越周波数 Index を用いて振幅値数列から卓越値に変換します。
35 行目: 前方 Index を用いて周波数数列から前方周波数に変換します。
36 行目: 後方Index を用いて周波数数列から後方周波数に変換します。
37 行目: 検索窓ボトム値を求めます。
38 行目: 検索結果の卓越個数を取得します。
39 行目: グラフ上卓越部分の囲み線描画の為、Pen 制御基本数列を生成します。
40 行目: ペン制御基本数列を検出した卓越個数分の数列に変換します。
41 行目: グラフ上卓越部分の囲み線描画の為、グラフ X 軸アドレス数列を生成します。
42 行目: 同じくグラフ上卓越部分を囲み線描画の為、グラフ Y 軸アドレス数列を生成します。
36 行目～67 行目:周波数グラフを描画します。

<実行結果:卓越周波数が存在した時に書き込まれる結果シート>

Page1:

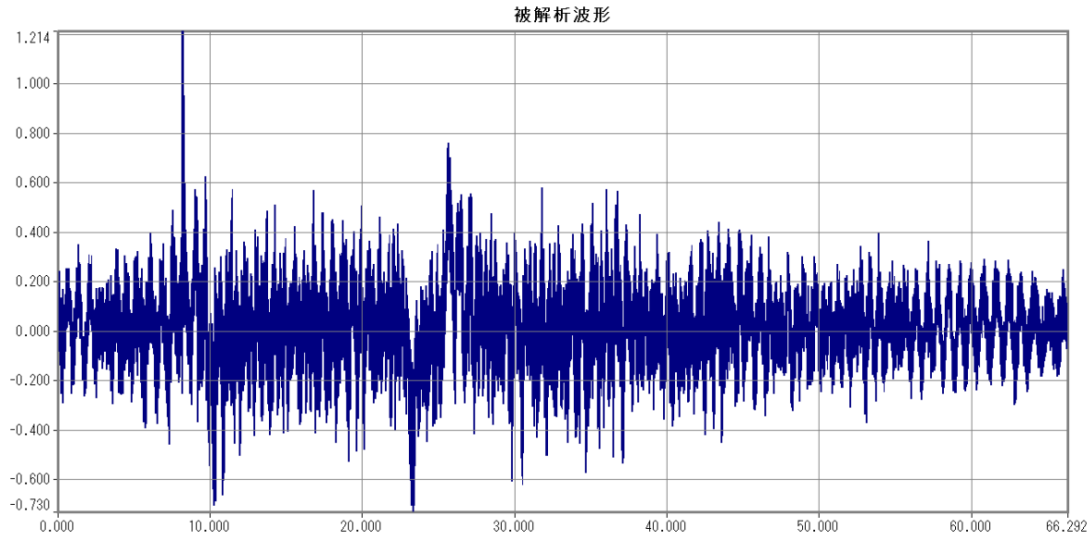
検索窓周波数幅(Hz)	検索窓深さ(dB)	卓越周波数(Hz)	卓越振幅
5.00	10.00	28.81	0.063
		48.58	0.021
		97.29	0.014

<実行結果：周波数スペクトラムグラフ>



※ グラフ内赤線枠が検出した卓越周波数検索窓を示します。

<解析対象波形グラフ>



<補足説明>

FFT スペクトラム解析結果の卓越周波数の検索はピーク検出関数（PDT 関数）を使用して、スペクトラム数列上をスライディングさせ、当該周波数±検索周波数幅範囲に於いて両側が設定した落差以上に減衰しているかで判断しまが、落差は絶対値ではなく、当該 地点の振幅値との比である必要があります。その為、振幅値は dB 値に変換して行います。例えば落差を 6dB として検索した場合、 振幅値が 0dB の場合は両側が-6dB 以下、振幅値が-20dB の場合は-26dB 以下をピークとして判定しますが落差を 0dB=1 としてリ ニア尺に置き換えた場合、振幅値が 0dB の場合の落差が 0.5、振幅値が-20dB の場合リニア尺では 0.1 ですので落差は 0.05 となり、落差を等比で設定したことと同じ意味となります。記述例では落差を 10dB、検索周波数幅を±2.5Hz として検索しています。
PDT 関数自体の記述法は 18 章の「18.3 項検索窓を設定して Index を取得する」項を参照下さい。

22. 演算関数を使用して頻度解析/疲労解析を行う

PcWaveForm 演算関数を使用して頻度解析を行う方法について説明します。

22. 1. 頻度解析手法と対応関数

22. 1. 1. 時間率頻度解析を行う

時間率頻度解析は、サンプリング周期ごとに対象波形の値がどのセル位置に該当するかを計数します。
対象数列が時系列で無い場合は度数分布を求めることと同じです。
【TRC 関数】を使用します。

文法:

格納先 = TRC(セルサイズ,セル個数,解析対象数列)

引数:

【セルサイズ】＜必須＞ セルの大きさを記述します。セル
間隔と同じ意味です。

【セル個数】＜必須＞
セルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。 なお、奇数で記述された場合は内部で＋1されて参照します。

【解析対象数列】＜必須＞ 解析
対象数列を記述します。

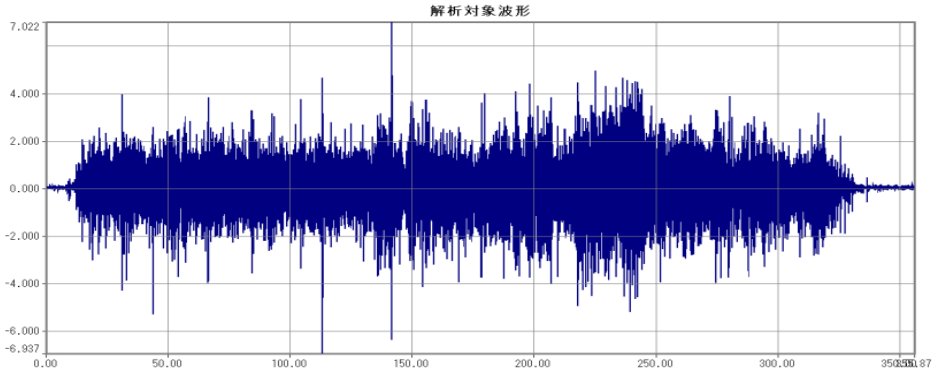
記述例:

収録カレントチャネルをセルサイズ 0.1、セル個数 128 として時間率頻度解析する場合

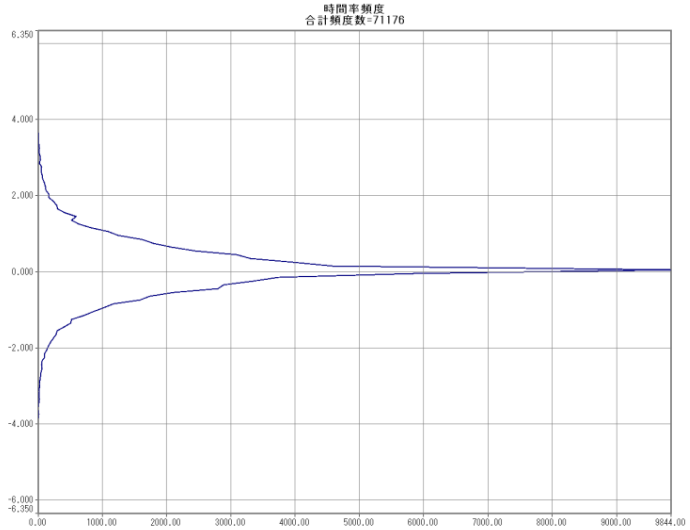
```
/*-----時間率頻度解析処理----------------------------------------------------------*/
$4 = CCH()                    /* カレントチャネル番号取得*/
$1 = TRC(0.1,128,#($4))       /* カレント ch,セルサイズ 0.1、セル個数±64、時間率頻度解析*/
$2 = ACC(DAG(64,0.1))-0.05    /* 正領域セル中央値演算*/
$2 = LNK(REV(SGN($2)),$2)     /* 正負領域セル中央値演算*/
$3 = SUM($1)                  /* 合計頻度数*/
/*-----時間率頻度解析グラフ描画処理------------------------------------------------*/
assign &1 = "合計頻度数="|$3(F0)
def graph_id @1 "時間率頻度" &1 /* 時間率頻度結果グラフ描画*/
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F3 5
plot @1 $1 $2                 /* グラフ X 軸に頻度数、Y 軸にセル中央値を記述*/
$3 = SPB(0)                   /* 解析対象波形時間軸データ生成*/
/*-------時間率解析対象波形グラフ描画処理------------------------------------------*/
def graph_id @2 "解析対象波形" /* 解析対象波形グラフ描画*/
def graph_aspect_ratio @2 2
def graph_x_axis @2 0,0 F2
def graph_y_axis @2 0,0 F3 5
plot @2 $3 #($4)
```

$1 は 128 点の要素持つ数列として戻ります。$1(0)～$1(63)がセル番号-64～-1 に相当し、$1(64)～$1(127)がセル番号 1～64 に相当した計数値が格納されます。グラフではセル中央値を表示しています。セル中央値演算は後述するセル番号/セル中央値演算関数(CNV 関数)で求めることもできます。

＜時間率頻度解析対象波形＞



＜時間率頻度解析実行結果グラフ＞



◆ 頻度数を時間へ変換する場合

時間率頻度解析で各セルに計数された頻度数の合計は、解析区間のサンプル数に等しい。従って、各セルに計数 された頻度数にサンプリング周期を掛算することで、当該セル範囲の値が存在した時間となります。

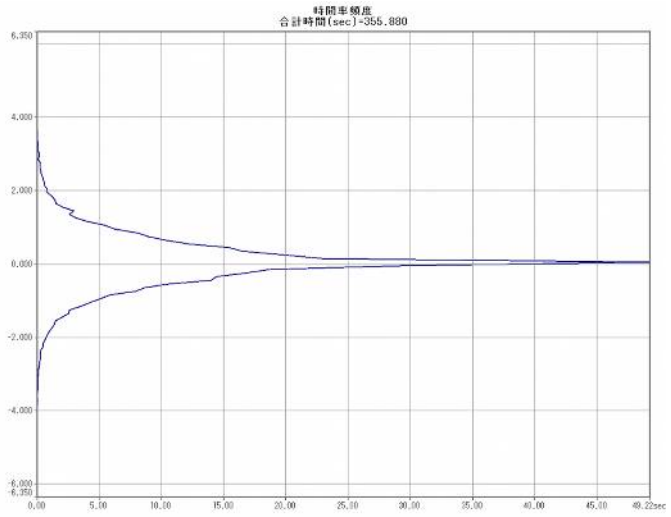
記述例：

前述記述例と同じ解析でグラフ X 軸を時間とした場合

```
$4 = CCH()                          /* カレントチャネル番号取得*/
$1 = TRC(0.1,128,#($4))*PRD()   /* カレント ch,セルサイズ 0.1、セル個数±64、時間率頻度*/
$2 = ACC(DAG(64,0.1))-0.05        /* 正領域セル中央値演算*/
$2 = LNK(REV(SGN($2)),$2)          /* 正負領域セル中央値演算*/
$3 = SUM($1)                        /* 合計時間(解析時間)*/
assign &1 = "合計時間(sec)="|$3(F3)
def graph_id @1 "時間率頻度" &1 /* 時間率頻度結果グラフ描画*/
def graph_x_axis @1 0,0 F2 "sec"
def graph_y_axis @1 0,0 F3 5
plot @1 $1 $2
$3 = SPB(0)                         /* 解析対象波形時間軸データ生成*/
def graph_id @2 "解析対象波形"   /* 解析対象波形グラフ描画*/
def graph_aspect_ratio @2 2
def graph_x_axis @2 0,0 F2
def graph_y_axis @2 0,0 F3 5
plot @2 $3 #($4)
```

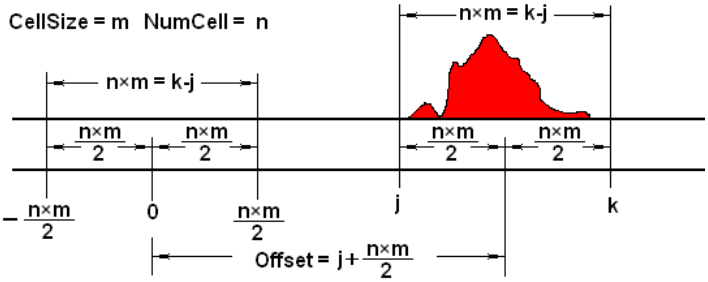
時間率頻度の頻度数にサンプリング周期を掛算することで、時間に変換します。

＜時間率頻度解析結果グラフ＞



22. 1. 2. オフセットさせた時間率頻度解析を行う

時間率頻度解析は暗黙的に 0 を中心として設定されたセルサイズで刻まれ各セルに計数します。従って解析対象チャネルが温度やエンジン回転数など、正負領域を持たない場合や正負領域何れかに偏っている場合があります。その場合、解析対象チャネルをオフセットして時間率頻度解析を行います。オフセット値を図示します。 図は解析したい範囲が上方にシフトした場合を例としています。



従って、解析対象チャネルからオフセット値=下限値 j+セルサイズ m×セル個数 n/2 を引算して時間率頻度解析します。又、解析結果のセル中央値は逆にオフセット値を加算します。

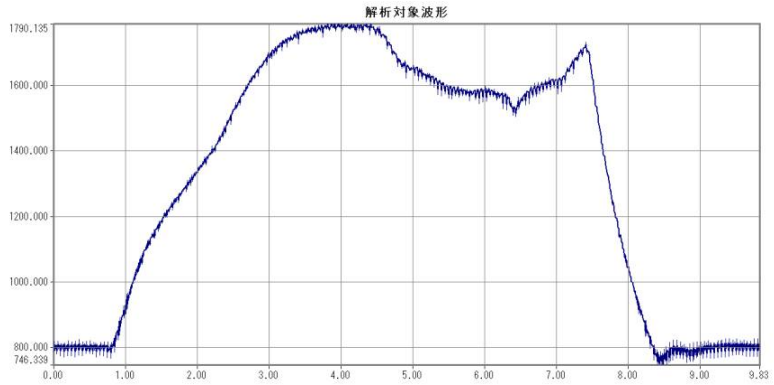
記述例:

エンジン回転数の 700～1800rpm 範囲をセルサイズ 5rpm で時間率頻度を行う場合

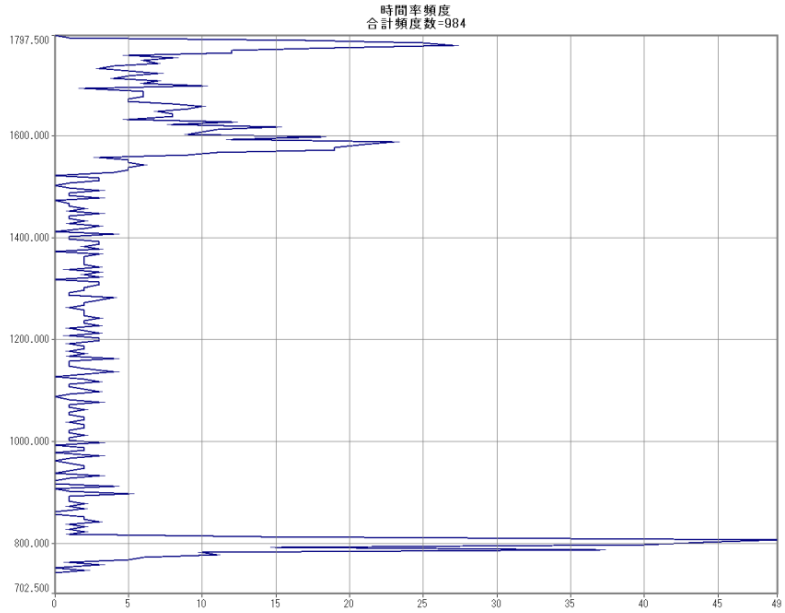
```
/*-----  オフセット時間率頻度解析処理-----------------------------------------------------------------*/
$5 "解析範囲:rpm" = 1800-700
$6 "セルサイズ:rpm" = 5
$7 "セル個数:" = INT($5/$6/2+0.5)*2 /* 必ず偶数個*/
$1 = TRC($6,$7,#2-(700+$7*$6/2)) /* カレント ch,セルサイズ 5rpm,範囲 700～1800rpm,時間率頻度*/
$2 = ACC(DAG($7,$6))+700-$6/2        /* セル中央値演算*/
$3 = SUM($1)                          /* 合計頻度数*/
/*------時間率頻度解析結果グラフ描画処理-------------------------------------------------------------*/
assign &1 = "合計頻度数="|$3(F0)
def graph_id @1 "時間率頻度" &1 /* 時間率頻度結果グラフ描画*/
def graph_x_axis @1 0,0 F0
def graph_y_axis @1 0,0 F3 5
plot @1 $1 $2
/*------オフセット時間率解析対象波形グラフ描画処理---------------------------------------------------*/
$3 = SPB(0)                           /* 解析対象波形時間軸データ生成*/
def graph_id @2 "解析対象波形"   /* 解析対象波形グラフ描画*/
def graph_aspect_ratio @2 2
def graph_x_axis @2 0,0 F2
def graph_y_axis @2 0,0 F3 5
plot @2 $3 #2
```

記述例では、セル中央値を専用関数(CNV 関数：21.2.2 参照)を使用しないで求めています。CNV 関数を使用する場合は CNV 関数の戻り中央値数列にオフセット値を加算することで良い。

＜時間率頻度解析対象波形＞



＜オフセット時間率頻度解析結果グラフ＞



22. 1. 3. 他チャネル参照時間率頻度解析を行う 通常の時間率頻度解析は、解析対象数列の要素値個々がどのセルに存在したかを計数しますが、他チャネル参照時間 率頻度解析は、セルに格納される内容が異なり、存在した位置(Index)と同じ Index 位置の参照数列の値に従属した統計 値を格納します。

他チャネル参照時間率頻度解析を行う場合【TRX 関数】を使用します。 **文法:**

格納先 = TRX(セルサイズ,セル個数,解析対象数列,解析参照数列,格納内容コード)

引数:

【セルサイズ】＜必須＞ セルの大きさを記述します。セル
間隔と同じ意味です。

【セル個数】＜必須＞
セルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。 なお、奇数で記述された場合は内部で＋1して参照します。

【解析対象数列】＜必須＞
頻度解析対象数列を記述します。

【解析参照数列】＜必須＞
頻度解析結果の Index から参照する数列を記述します。

【格納内容コード】<省略可＞ セルに格納する統計値種別を指定するコードです。省略した場合は 0 と見なします。

コード	格納内容
0	平均値
1	最大値
2	最小値
3	標準偏差
4	実効値
5	合計値

記述例:

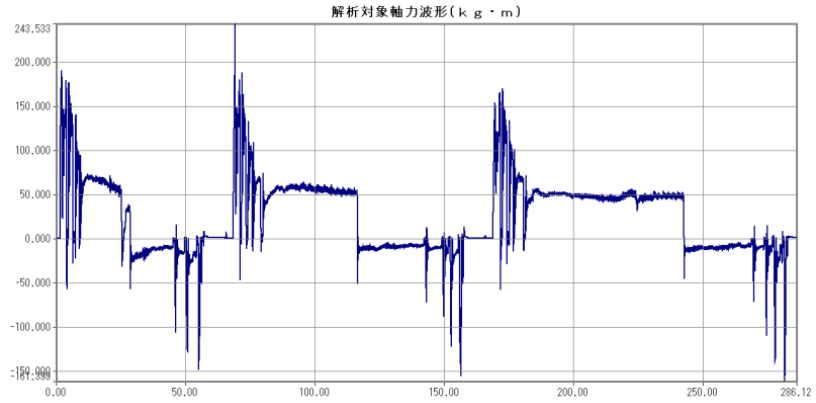
回転トルクを時間率頻度解析対象チャネルとし、各セルに当該トルクの頻度を求めるのではなく、計数したトルクの Index 位置の回転数の累積を求めるトルク累積回転数解析を行う場合

```
$1 = TRX(2,256,#6,#1,5)/60*PRD() /* #6:軸力,#2:回転数,トルク累積回転数解析*/
 $2 = CNV(1,2,256,1)              /* セル中央値取得 21.2.2 項参照*/
$3 = SUM($1)                      /* 合計回転数*/
assign &1 = "合計回転数="|$3(F3)
def graph_id @1 "トルク累積回転数" &1 /* 累積回転数結果グラフ描画*/
def graph_x_axis @1 0,0,50 F0 "回転"
def graph_y_axis @1 0,0,50 F3 5
plot @1 $1 $2
```

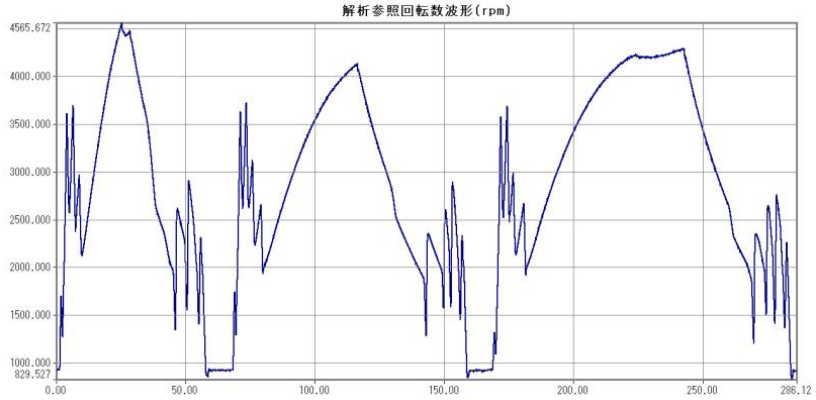
#6 は軸力チャネル、#1 は回転数チャネルを意味し、記述例では軸力をセルサイズ 2kg･m、セル個数 256(±128)とし

ています。なお、セルに計数するフラグは 5(合計値)を指定します。求めた合計回転数は rpm の為 60 で割り、サンプリング周期を掛け累積回転数に変換します。従って$1 は 256 個の要素を持ち数列として戻り、$1(0)～$1(127)がセル番号 -128～-1 に相当し、$1(128)～$1(255)がセル番号 1～128 に相当します。例えばセル番号 1 には軸力が 0<=#6(i)<2 範囲の Index 位置の回転数の合計値が格納されます。

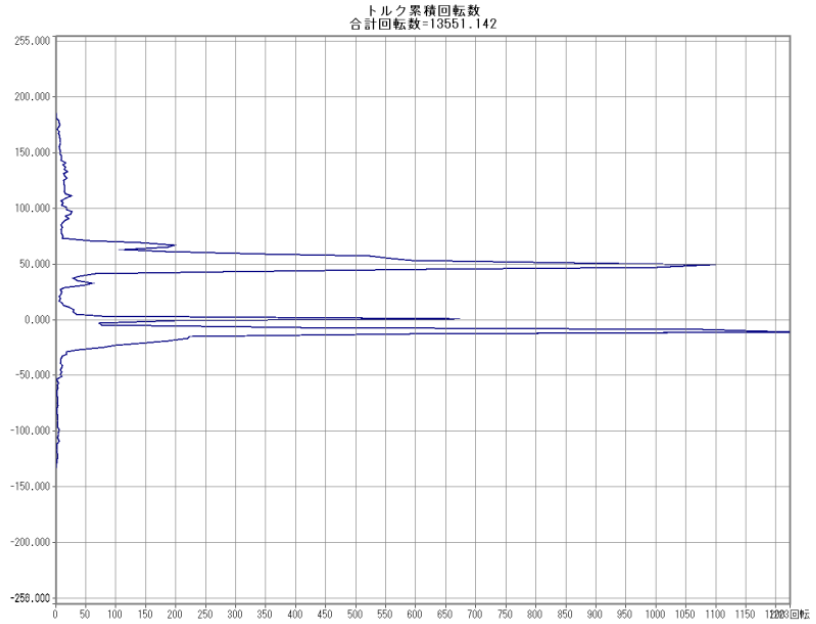
＜時間率頻度解析対象波形＞



＜時間率頻度解析参照対象波形＞



＜他チャネル参照時間率頻度解析結果グラフ:トルク累積回転数＞



22. 1. 4. 2次元時間率頻度解析を行う

2 次元時間率頻度解析は、セルが行列構造を持つ 2 次元となり、同じ index で対象数列 1 の要素値により列番号が対象数列 2 の要素値により行番号が決定されたセルに計数します。主に相互に関連した 2ch の関連性などを解析する場合に使用します。2 次元時間率頻度解析を行う場合【TR2 関数】を使用します。

文法:

格納先 = TR2(セルサイズ1,セル個数1,対象数列 1,セルサイズ 2,セル個数 2,対象数列 2)

引数:

【セルサイズ 1】＜必須＞

対象数列 1 のセルの大きさを記述します。セル間隔と同じ意味です。

【セル個数 1】＜必須＞

対象数列 1 のセルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。 なお、奇数で記述された場合は内部で＋1して参照します。

【解析対象数列 1】＜必須＞ 頻度解析

対象数列1を記述します。

【セルサイズ 2】＜必須＞

対象数列 2 のセルの大きさを記述します。セル間隔と同じ意味です。

【セル個数 2】＜必須＞

対象数列2のセルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。 なお、奇数で記述された場合は内部で＋1して参照します。

【解析対象数列 2】＜必須＞

頻度解析対象数列2を記述します。

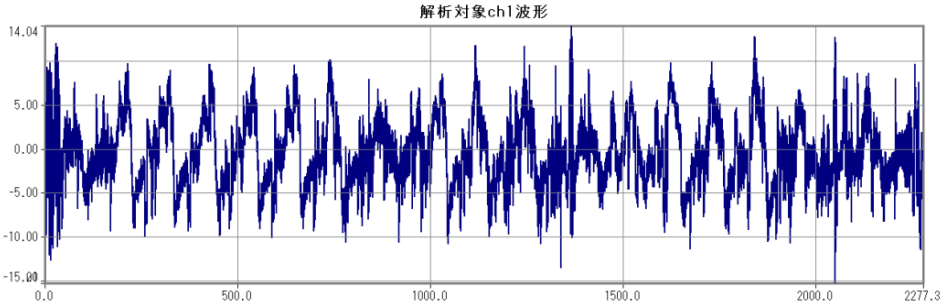
記述例:

収録チャネル#1、セルサイズ 0.5、セル個数 64 と#2、セルサイズ 2、セル個数 100 の 2 次元時間率頻度解析を行い結果を項目区切り半角カンマ”,”のテキスト形式で格納する場合

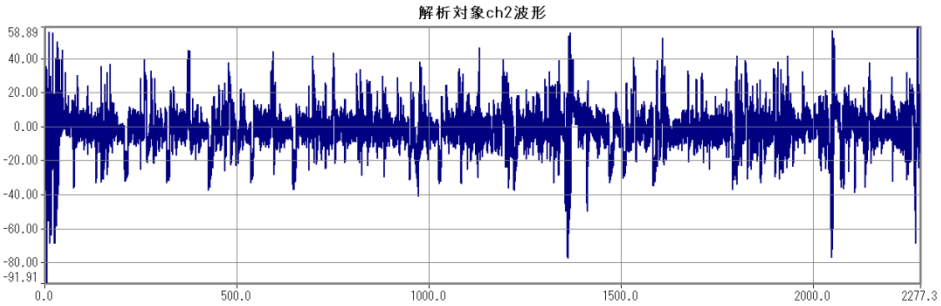
```
/* ----2 次元時間率頻度解析-------------------------------------*/
$1 = TR2(0.5,64,#1,2,100,#2)          /* ch1 と ch2 の 2 次元時間率頻度解析*/
$2 "列セル中央値:" = CNV(1,0.5,64,1)      /* #1 セル中央値取得 21.2.2 項参照*/
$7 "列セル番号:" = CNV(0,0.5,64,1)        /* #1 セル番号取得 21.2.2.参照*/
$8 "行セル中央値:" = CNV(1,2,100,1)       /* #2 セル中央値取得 21.2.2 項参照*/
$9 "行セル番号:" = CNV(0,0.2,100,1)       /* #2 セル番号取得 21.2.2.参照*/
/*---- 結果書き込み csv ファイル生成格納処理----------------------*/
def file_id %1 "2 次元時間率頻度結果" csv /*  格納先ファイル番号定義*/
def text_form %1 110,70                    /*  書き込みシートサイズ定義*/
$3 "row_counter:" = 0                      /*  行カウンタ初期化*/
$4 "格納先チャネル番号:" = 100             /*  格納先チャネル番号初期化*/
$5 "切り出し開始 index:" = 0               /* 行列切り出し開始 index 初期化*/
$6 "切り出し終了 index:" = 64-1            /*  行列切り出し終了Index 初期化*/
repeat_case $3 < 100                       /*  行列分離 行ループ*/
  proc cut_down{
    $($4) = ERC($5,$6,$1)                  /* 行列から行切り出し*/
    $4 = $4+1                              /* 格納先チャネル番号更新*/
    $5 = $5+64                             /* 切り出し開始 index 更新*/
    $6 = $6+64                             /* 切り出し終了 index 更新*/
    $3 = $3+1                              /* 行カウンタ更新*/
  }cut_down
write cell %1 1,1 0 "2 次元頻度解析結果"
write cell %1 2,1 1 "方向","チャネル","セルサイズ","セル個数"
write cell %1 3,1 1 "列","ch1","0.5","64"
write cell %1 4,1 1 "行","ch2","2.0","100"
write cell %1 5,3 0 $7(F0),$2(F3)          /* 列方向:ch1 のセル番号、セル中央値の書き込み*/
write cell %1 7,1 1 $9(F0),$8(F3)          /* 行方向:ch2 のセル番号、セル中央値の書き込み*/
write cell %1 7,3 0 $[100,100](F0)         /* 2 次元頻度解析結果の書き込み*/
save text_form %1                          /* 書き込みシートの格納*/
end
```

$1 は列数 64、行数 100 の仮想数列(64×100)となります。列方向は#1 の時間率頻度解析結果でセル番号-32～-1,1～32 に対応し、行方向は#2 の時間率頻度解析結果でセル番号-50～-1,1～50 に対応します。記述例中の演算処理ブロック”cutdown”は列方向 ch1、行方向 ch2 としたリスト形式で格納する為、行分解しています。なお、作成した結果リストファイルは、Script で標準サポートしている結果シート Window に書き込まず直接 csv ファイルの格納している為、PcWaveForm 上で表示することはできません。格納した 2 次元時間率頻度解析結果.csv をエクセル等で読み出して表示します。

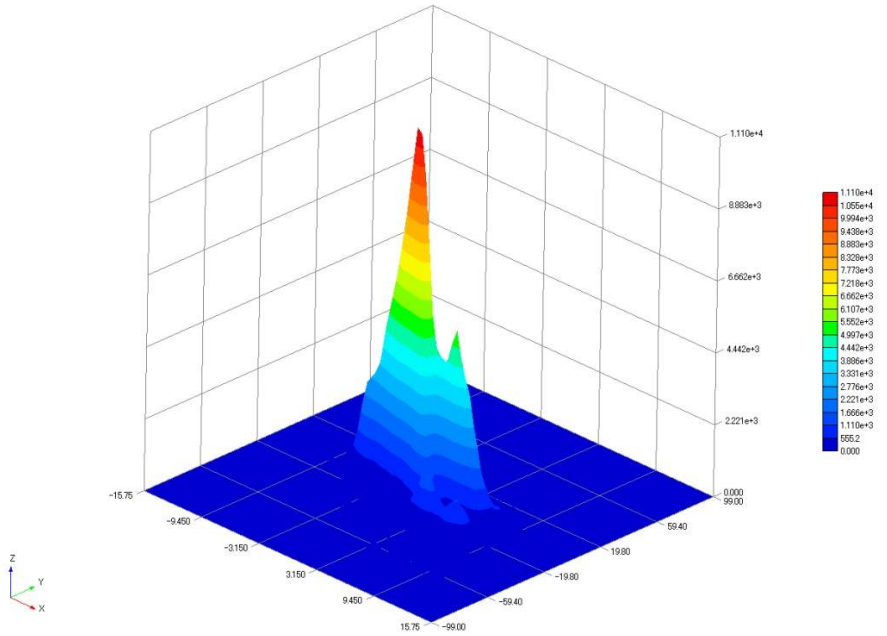
＜2 次元時間率頻度解析 X 軸対象波形＞



＜2 次元時間率頻度解析 Y 軸対象波形＞



＜2 次元時間率頻度解析結果グラフ＞



現在の Script では 3D グラフ表示機能を搭載していません。表示したグラフは csv 形式ファイルから 3D グラフ表示するフリーウェアを使用して表示しています。

#1の時間率頻度解析結果と#2 の時間率頻度解析結果を 2 次元時間率頻度解析結果から分離して求めたい場合、各列の合計の演算結果が#1 の時間率頻度結果となり、各行の合計を結果が#2 の時間率頻度解析結果となります.

記述例:

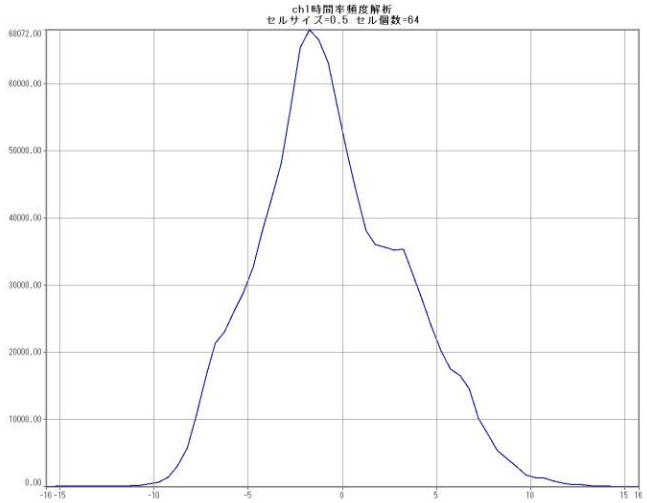
ch1 と ch2 の 2 次元時間率頻度解析結果から列合計、行合計を求め ch1 と ch2 それぞれ個別の時間率頻度解析結果を求める場合

```
/*------2 次元時間率頻度解析処理-----------------------------------------------------------*/
$1 = TR2(0.5,64,#1,2,100,#2)        /* ch1 と ch2 の 2 次元時間率頻度解析*/
$2 "列セル中央値:" = CNV(1,0.5,64,1)      /* #1 セル中央値取得 21.2.2 項参照*/
$3 "行セル中央値:" = CNV(1,2,100,1)       /* #2 セル中央値取得 21.2.2 項参照*/
/*------列(縦)合計/行(横)合計分離演算処理-----------------------------------------------------*/
$4 = MCS(64,$1)                 /*  行方向列合計 #1 の時間率頻度結果*/
```

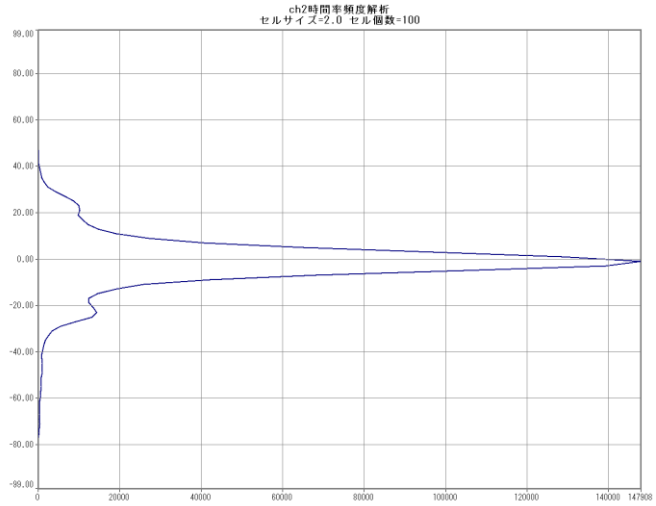
```
$5 = SUM(64,$1)              /* 列方向行合計 #2 の時間率頻度結果*/
/*-----時間率頻度解析結果グラフ列(縦)合計グラフ描画処理----------*/
def graph_id @1 "ch1 時間率頻度解析" "セルサイズ=0.5 セル個数=64"
def graph_x_axis @1 0,0 F0
def graph_y_axis @1 0,0 F2 5
plot @1 $2 $4
def file_id %1 "graph1" grp
save plot %1 @1 $2 $4
/*-----時間率頻度解析結果グラフ行(横)合計グラフ描画処理----------*/
def graph_id @1 "ch2 時間率頻度解析" "セルサイズ=2.0 セル個数=100"
plot @1 $5 $3
def file_id %1 "graph2" grp
save plot %1 @1 $5 $3
```

$4 は要素数 64、$4(0)～$4(31)がセル番号-32～-1 に、$4(32)～$4(63)がセル番号 1～64 に相当した#1 の時間率頻度解析結果の計数が格納され、$5 は要素数 100、$5(0)～$3(49)がセル番号-50～-1 に、$5(50)～$5(99)がセル番号 1～50 に相当した#2 の時間率頻度解析結果の計数が格納されます。MCS 関数(行列･列合計関数)を使用しています。

＜2 次元時間率頻度解析結果行列分離グラフ:ch1 時間率頻度解析結果＞



＜2 次元時間率頻度解析結果行列分離グラフ:ch2 時間率頻度解析結果＞



22. 1. 5. レインフロー法による頻度解析を行う レインフロー法は疲労解析に使用される頻度解析法です。疲労解析では、どの大きさを持つ応力振幅が何回掛ったかが重 要な解析因子となる為、応力波形に含まれる振幅の抽出/分解し振幅頻度を計数するレインフロー法が用いられます。レインフロー法により頻度解析を行う場合【RFM 関数】を使用します。RFM 関数は両振りで計数します、片振り変換する場合 は被害推定段階で計数を 1/2 して行います。

文法：

格納先 = RFM(セルサイズ,セル個数,無効振幅,解析対象数列)

引数：

【セルサイズ】＜必須＞ セルの大きさを記述します。セル
間隔と同じ意味です。

【セル個数】＜必須＞ セルの
個数を記述します。

【無効振幅】＜必須＞ 無効振幅値をセルの何パーセントとするかを%
単位で記述します。

【解析対象数列】＜必須＞ 解析
対象数列を記述します。

解析波形の正傾斜から負傾斜に移行した点を極大値(Peak)、負傾斜から正傾斜に移行した点を極小値(Valley)として抽出し、無効振幅除去した後、開始点から逆傾斜で開始点と等しいか過る地点間の開始点から正傾斜の場合は最大値、負傾斜の場合は最小値までを 1 解析区間として開始点から最大値又は最小値までの振幅と、間に存在する小ループを抽出し、ループの振幅×2 回を計数します。次に解析区間の終点を次の解析区間の開始点とし、同等に逆傾斜で開始点と等しいか過る地点間の最大値(最小値)までを解析区間として分解を繰り返します。セル番号を越える振幅値は最終セルに計数されます。

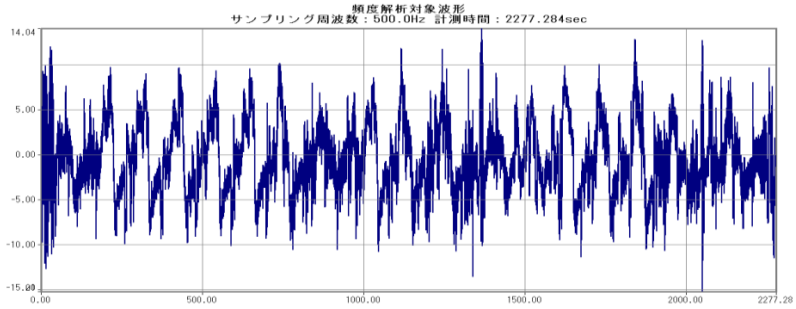
記述例：

収録チャネルch1をセルサイズ 0,5、セル個数 64、無効振幅 63%でレインフロー解析を行う場合

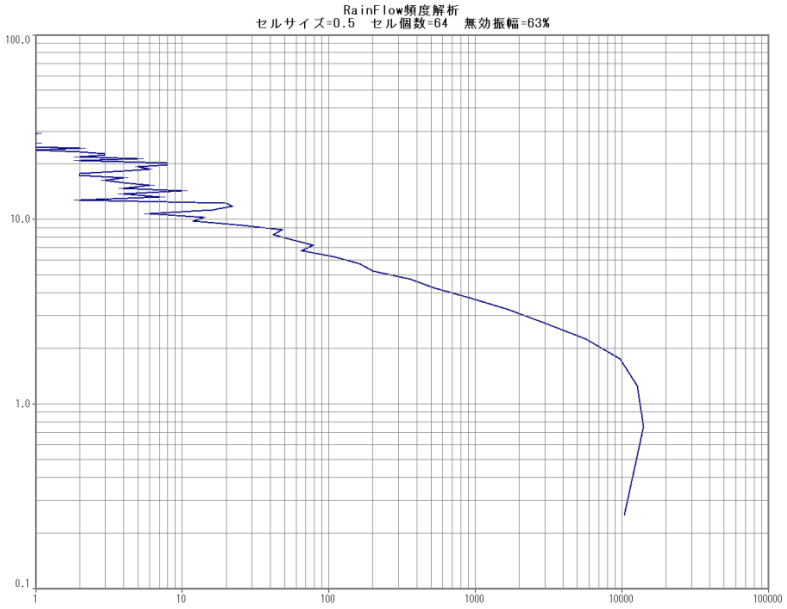
```
$/*----- ch1 レインフロー法頻度解析処理-----------------------------------------------*/
$1 = RFM(0.5,64,63,#1)          /* RainFlow 頻度解析*/
$2 = CNV(1,0.5,64)              /* セル中央値取得 21.2.2 項参照*/
/*------頻度解析結果グラフ描画処理-----------------------------------*/
$4 = RVS(3,$1,DAG(LEN($1),0.1)) /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$3 = MAX($1)                      /*  グラフ X 軸用、最大頻度数取得*/
def graph_id @1 "RainFlow 頻度解析" "セルサイズ=0.5 セル個数=64 無効振幅=63%"
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$3 F0 log
plot @1 $4 $2
def file_id %1 "freq_graph" grp
save plot %1 @1 $1 $2
/* ---頻度解析対象波形グラフ描画処理-----------------------------------*/
$4 = SPB(0)                       /*  波形グラフ X 軸用 時刻歴数列生成*/
$5 = 1/PRD()                      /* グラフ副題表示用  サンプリング周波数取得*/
$6 = LEN(#1)*PRD()                /* グラフ副題表示用  計測時間演算*/
assign &1 = "サンプリング周波数:"|$5(F1)|"Hz 計測時間:"|$6(F3)|"sec"
def graph_id @1 "頻度解析対象波形" &1
def graph_aspect_ratio @1 2
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F2 5
plot @1 $4 #1
def file_id %1 "頻度解析対象波形" grp
save plot %1 @1 $4 #1
end
```

関数の戻り数列$1 には、index 並びがセル番号並びに対応し、セル番号1から昇順に計数された数が格納されます。

＜レインフロー法頻度解析対象波形＞

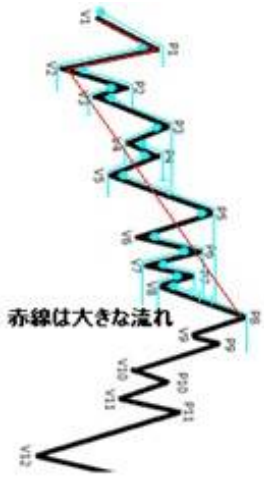


＜レインフロー法頻度解析結果グラフ＞



レインフロー処理の考え方：

左図の様に応力波形の時間経過を縦に取って説明します。



①応力は多層屋根の形をしていると仮定します。

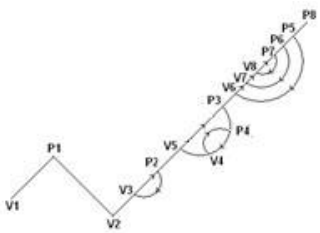
②雨滴が右に流れる場合は、屋根の左付け根に水源があるとし、左に流れる場合は屋根の右付け根にある。(V1～P8 迄に水源は 15)

③水源から発した流れは下層の屋根に落下する。

④落下した下層の屋根に既に流れている場合はそこで落下が停止する。

⑤同じ方向に流れる雨滴は、右流れの場合は、水源が、より左にあるものが優先し、合流した時点で停止する。（例えば v3 から流れた雨滴は、v2 から p2 流れ v3 から p3 に落下した時点で合流した時点で停止する。つまり、v2 の水源が V3 の水源より左に有る為、優先される）

⑥開始地点の水源より逆向きに流れる雨滴が開始地点より超えるか等しい時に大きな流れの方向が変わり、右向き流れの場合は、それまでの最大点、左向き流れの場合は最小点から流れ方向が変わり、雨滴の落下は止まる。



左図 v1 から p8 迄の流れでは、下記の振幅が抽出されます。 大きな流れ：p1-v1×1 回、p1-v2×1 回、p8-v2×1 回 小さな流れ（小ループ）：(p2-v3)×2 回、(p3-v5)×2 回、(p4-v4)×2 回、(p5-v6)×2 回、(p6-v7)×2 回、(p7-v8)×2 回

22. 1. 6. 極大/極小法による頻度解析を行う

【PVM 関数】を使用します。

文法：

格納先 = PVM(セルサイズ,セル個数,無効振幅,解析対象数列,検出フラグ)

引数：

【セルサイズ】＜必須＞ セルの大きさを記述します。セル間隔と同じ意味です。

【セル個数】＜必須＞

セルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。 なお、奇数で記述された場合は内部で＋1して参照します。

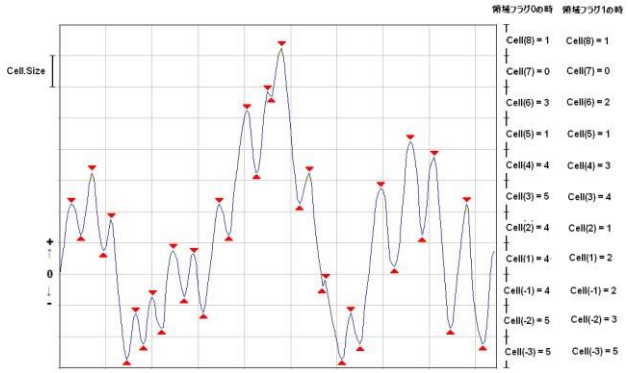
【無効振幅】＜必須＞ 無効振幅値をセルの何パーセントとするかを%単位で記述します。

【解析対象数列】＜必須＞ 解析対象数列を記述します。

【検出フラグ】＜省略可＞ 検出した極大値極小値をそのまま計数するか、或いは、正領域は極大値、負領域は極小値を計数するかのフラグを 意味します。

検出フラグ	戻り数列属性
0 又記述省略	全領域極大値、極小値
1	正領域:極大値、負領域:極小値

解析波形の正傾斜から負傾斜に移行した点を極大値(Peak)、負傾斜から正傾斜に移行した点を極小値(Valley)として抽出し、無効振幅除去した極大値極小値を該当するセルに計数します。セルを越える極大値又は極小値は最終セルに計数されます。



記述例：

収録チャネルch1をセルサイズ 0,5、セル個数 64、無効振幅 63%で極大値極小値解析を行う場合

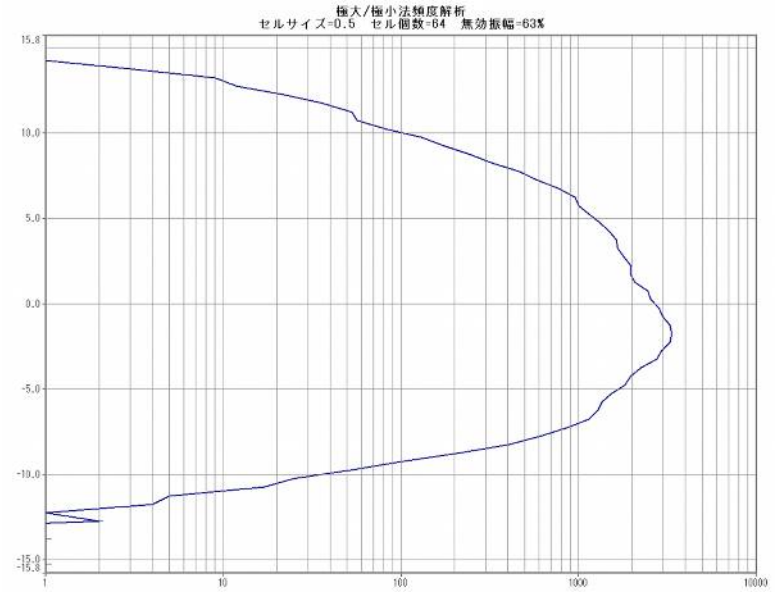
```
/*---- ch1 極大/極小法頻度解析処理----------------------------------------------*/
$1 = PVM(0.5,64,63,#1,0)        /* 正負領域の全ての極大値極小値を計数*/
$2 = CNV(1,0.5,64,1)             /* セル中央値取得 21.2.2 項参照*/
/*-----頻度解析結果グラフ描画処理-----------------------------------*/
$4 = RVS(3,$1,DAG(LEN($1),0.1)) /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$3 = MAX($1)                     /*  グラフ X 軸用、最大頻度数取得*/
def graph_id @1 "極大/極小法頻度解析" "セルサイズ=0.5 セル個数=64 無効振幅=63%"
def graph_y_axis @1 0,0 F1 5
def graph_x_axis @1 1,$3 F0 log
plot @1 $4 $2
def file_id %1 "freq_graph" grp
save plot %1 @1 $1 $2
```

関数の戻り数列$1 には index 並び順にセル番号-32 から＋32 に計数された数が格納されます。

セル番号を越える場合は何れか両端のセルに計数されます。

解析対象波形はレインフロー法記述例と同じ波形です。

＜極大/極小法頻度解析結果グラフ:全領域__極大/極小＞



極大/極小法関数は正負領域の於ける全ての極大値、極小値を計数するか、正領域は極大値を負領域は極小値を計数するかを指定できます。この機能と演算で、領域ごと、極大値/極小値ごとに分解することができます。

記述例:

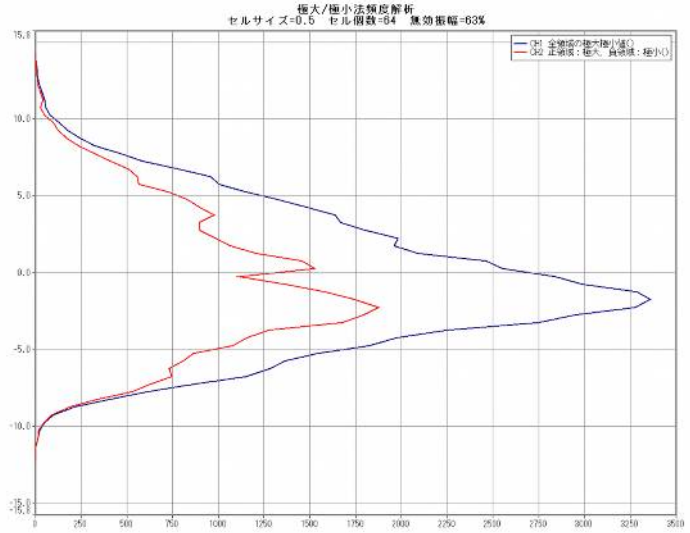
収録チャネルch1をセルサイズ 0.5、セル個数 64、無効振幅 63%で極大極小法による頻度解析処理を行い、正領域/負領域/極大値/極小値に分離する場合

```
/*------ 結果シート使用宣言-----------------------------------------*/
dcl sheet 1 {
      page 1:
      column $7,$6,$1,$2,$3,$4,$5
      format F0,F3,5(F0) 2
}sheet
/* ----- チャネル名定義処理-----------------------------------------*/
def ch_name $1 "全領域の極大極小値"
def ch_name $2 "正領域:極大、負領域:極小"
def ch_name $3 "正領域:極小、負領域:極大"
def ch_name $4 "正領域:極大、負領域:極大"
def ch_name $5 "正領域:極小、負領域:極小"
/*----- ch1 極大極小法頻度解析処理---------------------------------------------*/
$1 = PVM(0.5,64,63,#1,0)                  /* 正負領域の全ての極大値極小値を計数*/
$2 = PVM(0.5,64,63,#1,1)                  /* 正領域の極大値、負領域の極小値を計数*/
$3 = $1-$2                                /* 正領域の極小値、負領域の極大値を計数*/
$4 = LNK(ERC(0,31,$3),ERC(32,63,$2)) /* 正領域の極大値、負領域の極大値を計数*/
$5 = $1-$4                                /* 正領域の極小値、負領域の極小値を計数*/
$6 "セル中央値:" = CNV(1,0.5,64,1)      /* セル中央値取得 21.2.2 項参照*/
/* ----頻度解析結果リスト書き込み格納処理-----------------------------*/
$7 "Cell No.:" = CNV(0,0.5,64,1)        /*  セル番号取得 21.2.2 項参照*/
def sheet_title 1: "極大極小領域別"
write ch_column 1: $[1,7]
def file_id %1 "頻度処理結果シート" sht
save sheet %1
/*------頻度解析結果グラフ描画処理-----------------------------------------*/
$8 = INT((MAX($1)+499)/500)*500       /*  グラフ X 軸最大固定用演算*/
def graph_id @1 "極大/極小法頻度解析" "セルサイズ=0.5 セル個数=64 無効振幅=63%"
def graph_y_axis @1 0,0 F1 2:1
def graph_x_axis @1 0,$8,250 F0
plot @1 $1,$2 $6,$6
def file_id %1 "freq_graph1" grp
save plot %1 @1 $1,$2 $6,$6
plot @1 $3 $6
def file_id %1 "freq_graph2" grp
save plot %1 @1 $3 $6
plot @1 $4 $6
```

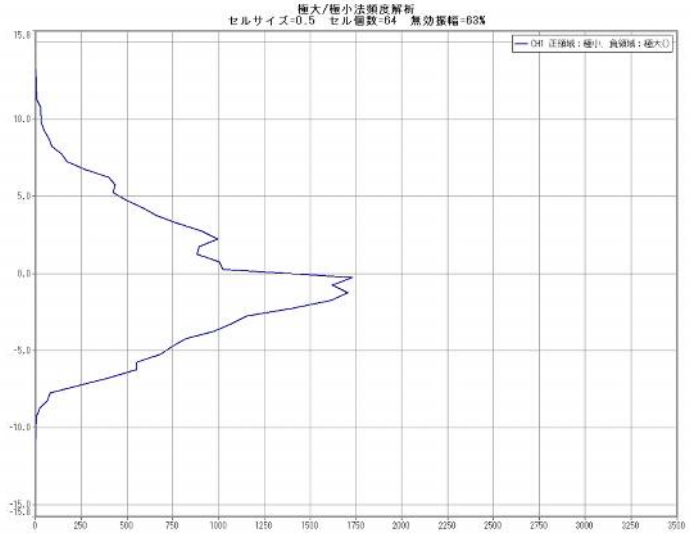
```
def file_id %1 "freq_graph3" grp
save plot %1 @1 $4 $6
plot @1 $5 $6
def file_id %1 "freq_graph4" grp
save plot %1 @1 $5 $6
end
```

解析対象波形はレインフロー法記述例と同じ波形です。

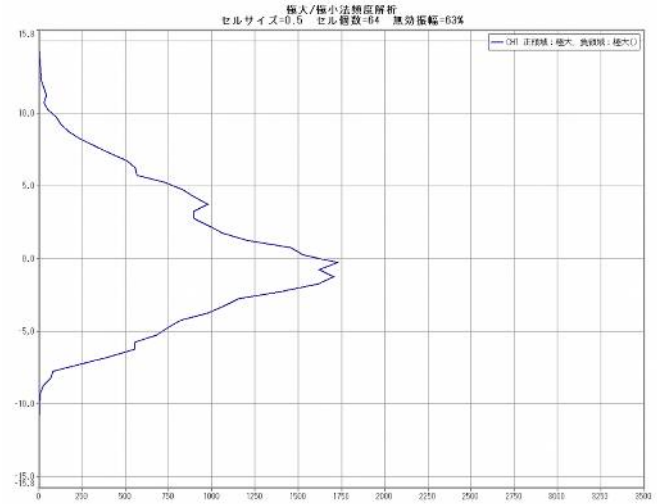
＜極大/極小法頻度解析結果グラフ：全領域-極大/極小、正領域-極大、負領域-極小＞



＜極大/極小法頻度解析結果グラフ：正領域-極小、負領域-極大＞



＜極大/極小法頻度解析結果グラフ：正領域-極大、負領域-極大＞



＜極大/極小法頻度解析結果グラフ：正領域-極小、負領域-極小＞

＜極大/極小法頻度解析結果リスト＞

Page1: 極大極小領域別

Cell No.	セル中央値	全領域の極大極小値	正領域：極大、負領域：極小	正領域：極小、負領域：極大	正領域：極大、負領域：極大	正領域：極小、負領域：極小
-32	-15.750	0	0	0	0	0
-31	-15.250	1	1	0	0	1
-30	-14.750	0	0	0	0	0
-29	-14.250	0	0	0	0	0
-28	-13.750	1	1	0	0	1
-27	-13.250	0	0	0	0	0
-26	-12.750	2	2	0	0	2
-25	-12.250	1	1	0	0	1
-24	-11.750	4	4	0	0	4
-23	-11.250	5	5	0	0	5
-22	-10.750	17	17	0	0	17
-21	-10.250	25	20	5	5	20
-20	-9.750	52	48	4	4	48
-19	-9.250	99	91	8	8	91
-18	-8.750	217	191	26	26	191
-17	-8.250	409	342	67	67	342
-16	-7.750	614	532	82	82	532
-15	-7.250	869	628	241	241	628
-14	-6.750	1150	747	403	403	747
-13	-6.250	1285	733	552	552	733
-12	-5.750	1363	808	555	555	808
-11	-5.250	1545	865	680	680	865
-10	-4.750	1821	1078	743	743	1078
-9	-4.250	1975	1156	819	819	1156
-8	-3.750	2251	1275	976	976	1275
-7	-3.250	2750	1678	1072	1072	1678
-6	-2.750	2945	1792	1153	1153	1792
-5	-2.250	3276	1876	1400	1400	1876
-4	-1.750	3359	1744	1615	1615	1744
-3	-1.250	3286	1581	1705	1705	1581
-2	-0.750	2986	1365	1621	1621	1365
-1	-0.250	2842	1115	1727	1727	1115
1	0.250	2552	1528	1024	1528	1024
2	0.750	2465	1460	1005	1460	1005

22. 1. 7. 最大/最小法による頻度解析を行う

【MMM 関数】を使用します。

文法：

格納先 = MMM(セルサイズ、セル個数,無効振幅,解析対象数列）引数：

【セルサイズ】＜必須＞

セルの大きさを記述します。セル間隔と同じ意味です。

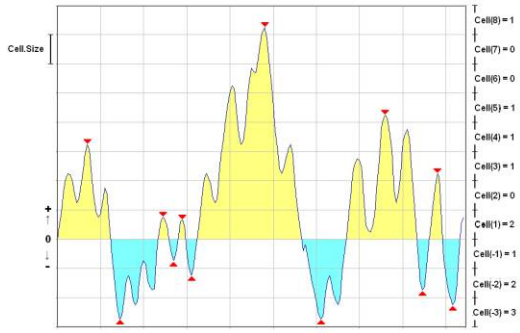
【セル個数】＜必須＞

セルの個数を記述します。正負領域を持ちますので領域ごとに、設定したセル個数/2 となります。なお、奇数で記述された場合は内部で＋1して参照します。

【無効振幅】＜必須＞ 無効振幅値をセルの何パーセントとするかを%単位で記述します。

【解析対象数列】＜必須＞ 解析対象数列を記述します。

解析波形の極大値(Peak)、極小値(Valley)として抽出し、無効振幅除去した極大値極小値からゼロを正傾斜で過ぎってから再び負傾斜でゼロを過ぎるまでの最大値(Max)、ゼロを負傾斜で過ぎってから再び正傾斜で過ぎるまでの最小値(Min)を抽出し該当セルに計数します。解析対象波形がゼロを過ぎらないと求まりません。



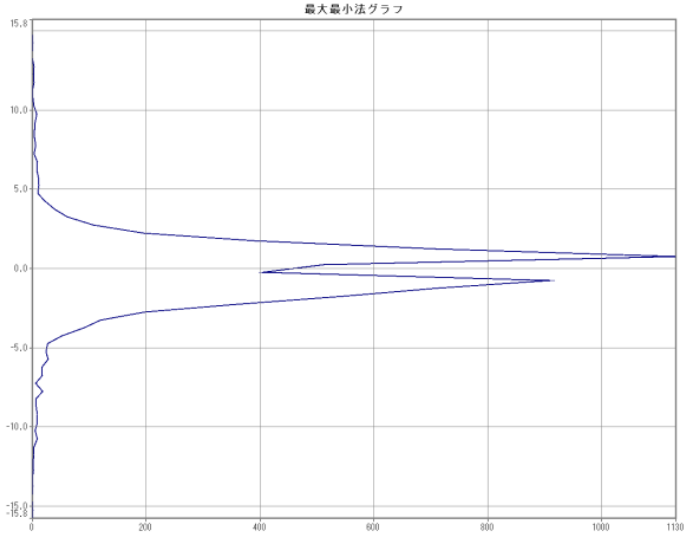
記述例：

収録チャネルch1をセルサイズ 0,5、セル個数 64、無効振幅 63%で最大値最小値解析を行う場合

```
/*---最大最小法頻度解析処理----------------------------------*/
$1 = MMM(0.5,64,63,#1)              /* 最大最小法頻度解析*/
$2 = CNV(1,0.5,64,1)              /* セル中央値取得 21.2.2 項参照*/
/*---頻度解析結果グラフ処理----------------------------------*/
def graph_id @1 "最大最小法グラフ"
def graph_y_axis @1 0,0 F1 5
def graph_x_axis @1 0,0 F0
plot @1 $1 $2
```

$1 には、index 並び順にセル番号-32 から＋32 に計数された数が格納されます。
解析対象波形はレインフロー法記述例と同じ波形です。

＜最大/最小法頻度解析結果グラフ：正領域_最大、負領域_最小＞



22. 1. 8. レベルクロス法による頻度解析を行う

【LCR 関数】を使用します。

文法：

格納先 = LCR(セルサイズ、セル個数,無効振幅,解析対象数列)

引数：

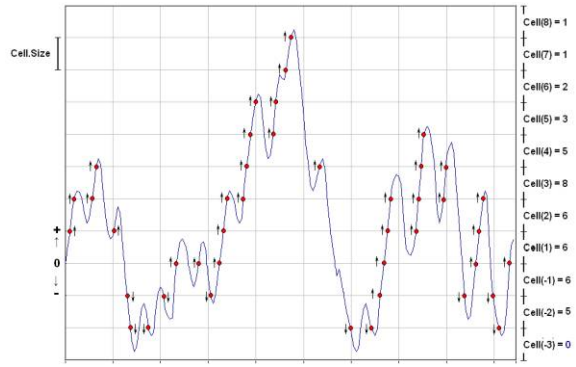
【セルサイズ】＜必須＞ セルの大きさを記述します。セル
間隔と同じ意味です。

【セル個数】＜必須＞ セルの
個数を記述します。

【無効振幅】＜必須＞
無効振幅値をセルの何パーセントとするかを%単位で記述します。

【解析対象数列】＜必須＞ 解析
対象数列を記述します。

解析波形の正傾斜から負傾斜に移行した点を極大値(Peak)、負傾斜から正傾斜に移行した点を極小値(Valley)として抽出し、無効振幅除去した極大値極小値から正領域は極小値から極大値が設定した Slice Level(セルの大きさ)を超えた時、負領域は極大値から極小値が Slice Level を下方に超えた時に計数します。又、Slice Level を飛び越しで超えた時は、飛び越された全ての Cell に計数します。

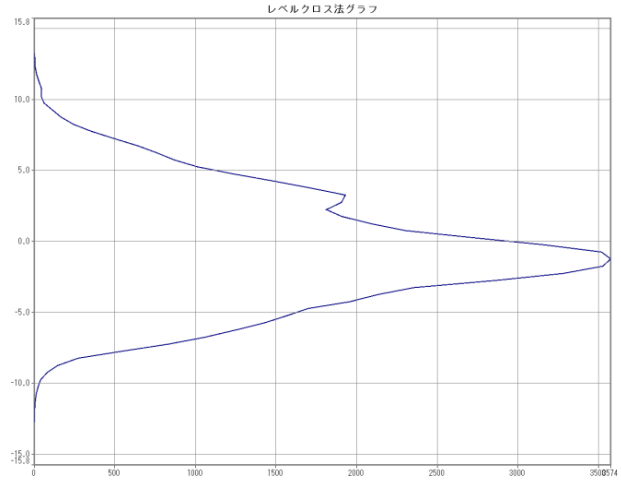


記述例：

収録チャネルch1をセルサイズ 0,5、セル個数 64(±32)でレベルクロス法頻度解析を行う場合

```
/*---レベルクロス法頻度解析処理----------------------------------*/
$1 = LCR(0.5,64,63,#1)              /* レベルクロス法頻度解析*/
$2 = CNV(1,0.5,64,1)               /*セル中央値取得 21.2.2 項参照*/
/*---頻度解析結果グラフ処理----------------------------------*/
def graph_id @1 "レベルクロス法グラフ"
def graph_y_axis @1 0,0 F1 5
def graph_x_axis @1 0,0 F0
plot @1 $1 $2
```

関数の戻り数列$1 には、index 並び順にセル番号-32 から＋32 に計数された数が格納されます。 解析対象波形はレインフロー法記述例と同じ波形です。



正領域の下降、負領域の上昇を求めたい場合は、解析対象数列を SGN 関数を使用して符号反転した数列を LCR 関数の引数として記述します。全ての領域の上昇下降を求めたい場合は、符号反転前の結果と符号反転後の結果を加算する事で求めます。

22. 2. 頻度解析結果の整理

22. 2. 1. 正負領域を持つ計数数列から絶対値化して正領域に変換する

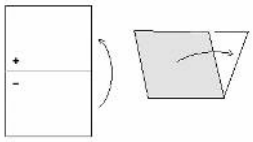
レインフロー法以外の極大極小法、最小最大法、レベルクロス法等は頻度解析結果の戻り数列は正負の領域を持ちます。
正負領域を持つ計数数列を折り畳んでセル番号を絶対値化する【FLD 関数】を使用します。

文法:

格納先 = FLD(対象数列)

引数:

【対象数列】＜必須＞ 折り畳み対象数列を記述します。対象数列の要素数は必ず偶数で
必要があり
ます。戻り数列の要素数は 1/2 となります。



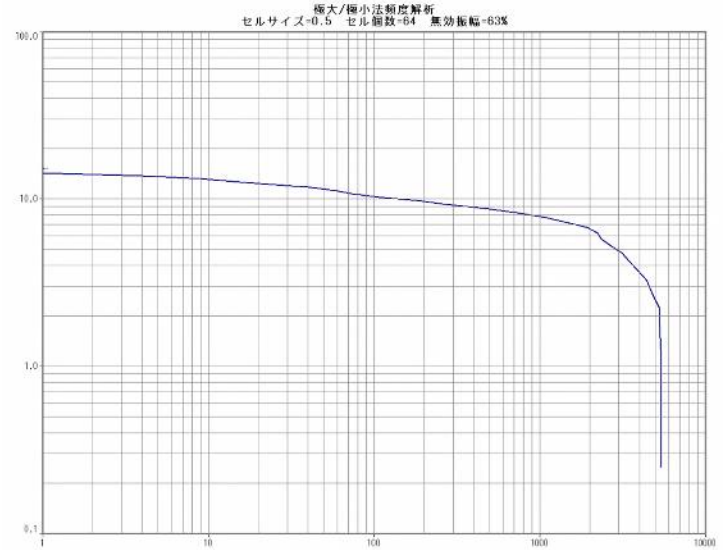
記述例:

収録チャネル ch1 をセルサイズ 0.5、セル個数 128、無効振幅セルサイズの 63%として極大極小法により求めた正負領域を折り畳みセル番号を絶対値化する場合

```
/*----- ch1 極大/極小法頻度解析処理&折り畳み----------------------------------*/
$1 = PVM(0.5,64,63,#1,0)          /* 正負領域の全ての極大値極小値を計数*/
$1 = FLD($1)                       /* 正負領域を折り畳み、セル番号を絶対値化*/
$2 = CNV(1,0.5,64)                 /* セル中央値取得 21.2.2 項参照*/
/*------頻度解析結果グラフ描画処理-------------------------------------------*/
$4 = RVS(3,$1,DAG(LEN($1),0.1))    /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$3 = MAX($1)                       /*  グラフ X 軸用、最大頻度数取得*/
def graph_id @1 "極大/極小法頻度解析" "セルサイズ=0.5 セル個数=64 無効振幅=63%"
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$3 F0 log
plot @1 $4 $2
def file_id %1 "freq_graph" grp
save plot %1 @1 $1 $2
```

解析対象波形はレインフロー法記述例と同じ波形です。

＜極大/極小法頻度解析結果折り畳み後グラフ＞



FLD 関数を使用しない場合、数列操作関数を使用して折り畳みます。演算式を示します。

負領域計数数列 = REV(ERC(0,LEN(対象数列)/2-1,対象数列)) 正領域計数数列 = ERC(LEN(対象数列)/2, LEN(対象数列)-1,対象数列) セル番号絶対値化計数数列 = 負領域計数数列+正領域計数数列

対象数列の要素数が未知を前提として数列の要素数取得関数を使用した演算式を示しましたが、対象数列の要素数は設定したセル個数を意味していますので、直接即値で記述しても良い。

記述例:

前述の記述例と同じ操作を演算式で行う場合

```
$1 = PVM(0,5,64,63,#1,0)
$2 = REV(ERC(0,32-1,$1))+ERC(32,64-1,$1)
```

$2 のセル個数は$1 の 1/2 となり、

```
$2(0)=$1(31)+$1(32)
$2(1)=$1(30)+$1(33)
```

|
$2(31)=$1(0)+$1(63)
と格納されます。

22. 2. 2. セル番号/セル中央値を求める

頻度解析結果を表形式或いはグラフ表示する場合にセル中央値数列、セル番号数列などが必要になる場合があります。セル中央値又はセル番号を求める【CNV 関数】を使用します。

文法:

格納先 = CNV(戻り値フラグ,セルサイズ,セル個数, 領域フラグ) 引数:

【戻り値フラグ】＜必須＞ セル番号数列を戻すか、セル中央値を戻すかを意味するフラグです。

戻り値フラグ	戻り数列属性
0	セル番号数列
1	セル中央値数列

【セルサイズ】＜必須＞
セルサイズを記述します。

【セル個数】＜必須＞ セル個数を記述します。

【領域フラグ】＜省略可＞ 頻度解析結果が正負領域を持つか、正領域のみかを意味するフラグです。

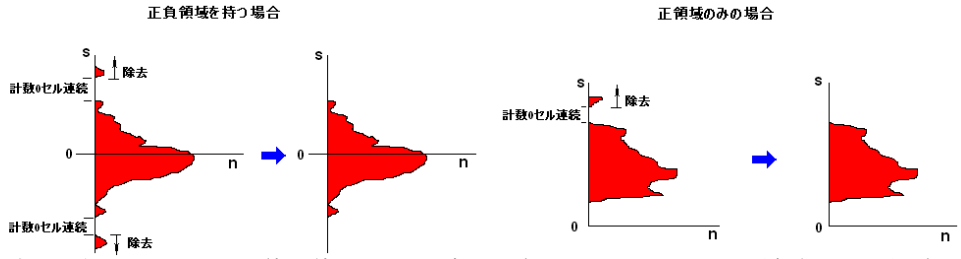
領域フラグ	戻り数列属性
0 又は記述省略	正領域数
1	負領域-正領域

記述例:

```
$1 = CNV(0,1,128,1) /*セルサイズ 1、セル個数 128、正負領域型、セル番号戻り*/
$2 = CNV(1,1,128,1) /*セルサイズ 1、セル個数 128、正負領域型、セル中央値戻り*/
$3 = CNV(0,1,64,1) /*セルサイズ 1、セル個数 64、正領域型、セル番号戻り*/
$3 = CNV(0,1,64,0) /*セルサイズ 1、セル個数 64、正領域型、セル中央値戻り*/
```

22. 2. 3. 無効計数除去処理

計数ゼロが連続した以降の計数値を無視したい場合の処理を意味します。



処理方法は何れもセル番号の絶対値最大から最初に計数ゼロ以外のセル以外が存在し、再び計数ゼロセルに出会った地点を計数 0 セル連続区間検索開始点とし、その地点から次にゼロ以外のセルに出会う迄の計数 0 のセル数が設定した連続セル数以上の場合に検索開始地点セル番号以前の計数をゼロに置き換えます。正負領域を持つ場合は正領域と負領域を分離して行い、処理後に合成します。なお、無効計数除去処理専用関数はありません。Index 検索関数及び数列操作関数を使用して処理します。

記述例:

正負領域を持つ頻度解析結果数列$1 から連続 8 セルとして無効計数除去を行う場合

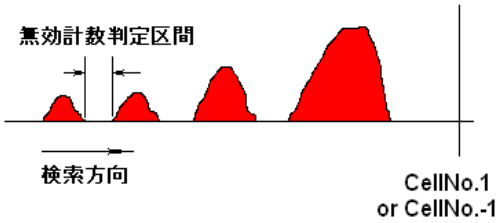
```
$2 = ERC(0,LEN($1)/2-1,$1)          /*負領域切り出し*/
call proc 無効計数セル除去 $2,8     /*無効計数除去処理 対象数列,連続数*/
$3 = REV(ERC(LEN($1)/2,LEN($1)-1,$1)) /*正領域切り出し反転*/
call proc 無効計数セル除去 $3,8     /*無効計数除去処理 対象数列,連続数*/
$1 = LNK($2,REV($3))                /*正負領域合成*/
```

正負領域を持つ頻度処理結果数列は、正領域と負領域を分離し、それぞれ無効計数セル除去処理を行った後に合成し て処理します。無効計数セル除去は外部演算処理ブロックで行います。外部演算処理ブロックの引数は、処理対象数列と 連続 0 計数セル数です。なお、正領域を処理する場合、分離した対象数列の Index 並びを反転して引数として記述しま す。

使用する外部演算処理ブロック記述例を示します。

```
proc 無効計数セル除去{
  /*-------------------------------------------------------------
```

```
呼び出し方法:
  call proc 無効計数除去 計数数列,連続セル数
引数:
  $1 <in/out>  計数数列   $2 <in> 連続セル数
  ※  計数数列はセル番号絶対値降順並びの必要があります
-------------------------------------------------------------------*/
def inherit_ch $1,$2        /* Subroutine 引数 ch*/
def local_ch $3,$4,$5       /* Subrotine ローカル ch*/
$3 = DTD(0,0.5,1,0,$1) /*判定開始 index 検索*/
case $3 > 1                 /* 開始 Index 検出*/
  proc 終端 index{
    $4 = DTD(1,0.5,1,$3,$1) /*判定終了 index 検索*/
    case $4 > 1             /*終了 index 検出*/
      proc 判定{
        $5 = $4-$3          /*連続個数演算*/
        case $5 >= $2       /*設定連続セル数以上検出*/
          proc 無効処理{
            $1 = SBV(ACC(DAG($3,1))-1,DAG($3,0),$1) /*検出区間セル計数ゼロ置換*/
          }無効処理
      }判定
  }終端 index
}無効計数セル除去
```



外部演算処理ブロック内で使用している DTD 関数は 18.1.1.項、SBV 関数は 17.2.2.項を参照下さい。
また、外部演算処理ブロックの作成に関しては、第 3 章「演算処理ブロックを構成する」を参照下さい。

記述例:

正領域のみの頻度解析結果数列$1 から連続 8 セルとして無効計数除去を行う場合

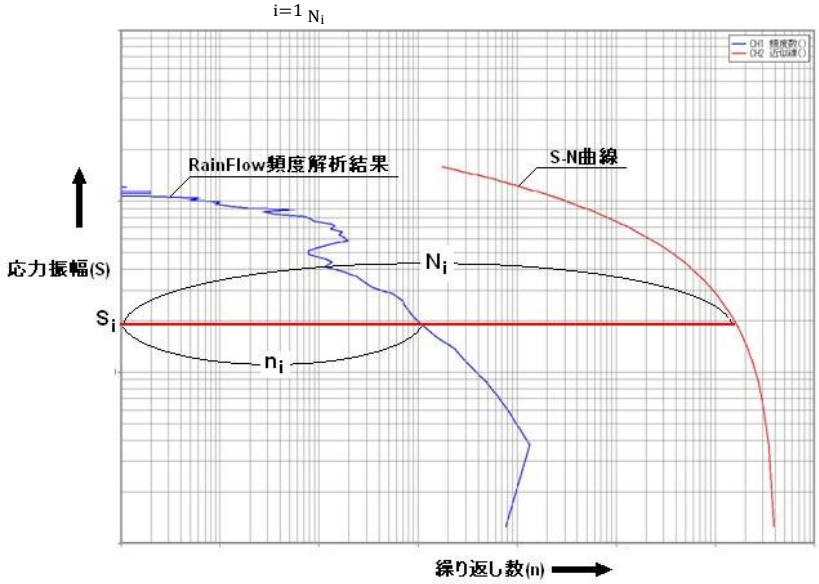
```
$1 = REV($1)                        /*反転正領域*/
call proc 無効計数セル除去 $1,8     /*対象数列,連続数*/
$1 = REV($1)                        /* 並びを元に戻す*/
```

正負領域を持つ場合同様に、無効計数セル除去は外部演算処理ブロックで行います。頻度解析結果数列を特に分離する必要はありませんが、Index 並びを反転する必要があります。

22. 3. 被害推定を行う

レインフロー法により頻度解析した結果の振幅頻度 n と与える S-N 曲線(ストレス S(多く場合応力振幅)と当該ストレス下での繰り返し回数 n)から求めた繰り返し数 N との比の合計を被害度と言います。
n を当該セルに於ける頻度数、N を S-N 曲線から求めた当該セル中央値に於ける繰り返し数とし、k を最大セル番号すると、合計被害度(ダメージ)D は次式により演算します。従って D は 1 以下でないと壊れている事を意味します。

$$D = \sum_{i=1}^{k} \frac{n_i}{N_i}$$



推定寿命は解析対象データの計測時間或いは走行時間を求めた合計被害度で割り算した値となります。

推定走行寿命 = 解析対象データの走行距離／合計被害度
推定寿命 = 解析データの計測時間/合計被害度

但し、何れも定義する S-N 曲線の定義根拠に依存します。また、定義した S-N 曲線が規定した時間或いは走行距離での許容疲労を意味する場合、レインフロー法により求めた各セルの頻度数(計数)を規定した時間或いは走行距離への換算が

必要になります。換算は同じ応力振幅が続くことを前提としています。 **校正した頻度数=(S-N 曲線基準走行距離/解析対象データの走行距離)×頻度数 校正した頻度数=(S-N 曲線基準時間/解析対象データの計測時間)×頻度数**

22. 3. 1. S-N 曲線をテーブルで定義して被害推定する

S-N 曲線をテーブルで定義する場合代入文などを使用して S 値と対応する N 値を定義します。テーブルは LOG-LOG 尺上で定義されたと見なして、LOG-LOG 尺上で補間/補外して頻度解析結果のセル中央値に対応した N 値数列を生成して被害度演算を行います。S 値、N 値自体は頻度解析した材料や部位に依存します。

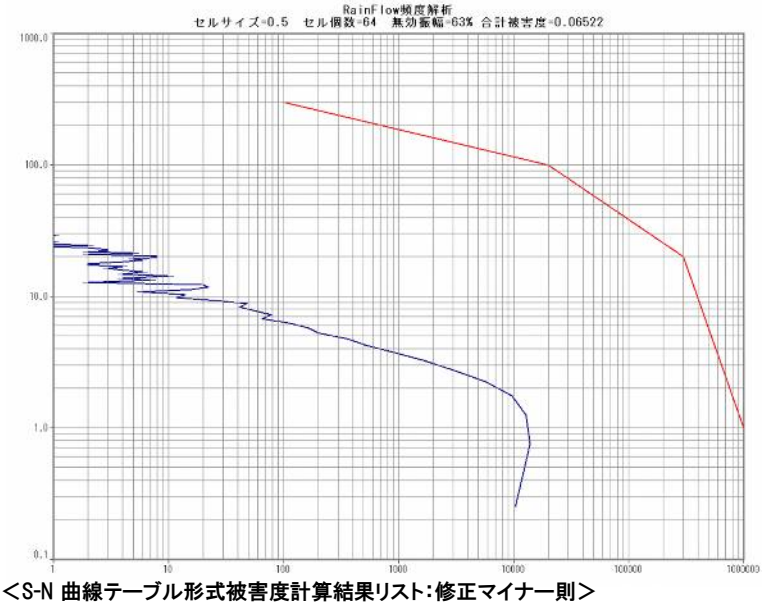
記述例:

セルサイズ 2.5、セル個数 64 でレインフロー法頻度解析結果$5 から 4 点テーブルで定義した S-N 曲線から修正マイナー則で被害度を求める場合

```
/*--------結果シート宣言処理-----------------------------------------------------------*/
dcl sheet 1 {
     page 1: "頻度解析結果"
     column &2,&3,&4,$12,$6,$5,$13,$8
     format 3(A),F0,F2,F2,F0,F3,F6 2
}sheet
/*--------S-N 曲線テーブル定義---------------------------------------------------------*/
assign $1 = 300,100,20,1                    /*S 値テーブル*/
assign $2 = 1e2,2e4,3e5,1e6                 /*N 値テーブル*/
$3 = SRT(2,$1)                              /*S 値の昇順並び Index 取得*/
$1 = QUE($3,$1)                             /*取得した Index 順に S 値テーブル並び替え*/
$2 = QUE($3,$2)                             /*取得した Index 順に N 値テーブル並び替え*/
/*-------レインフロー法頻度解析処理---------------------------------------------------*/
$5 "頻度数:" = RFM(0.5,64,63,#1)            /*ch1 のレインフロー法頻度解析処理*/
$6 "セル中央値:" = CNV(1,0.5,64)            /*セルサイズ 0.5,セル個数 64,正領域セル中央値取得*/
$12 "セル番号:" = CNV(0,0.5,64)             /*セルサイズ 0.5,セル個数 64,正領域セル番号取得*/
/*---被害度演算処理 修正マイナー則適用 両振り-----------------------------------------*/
$7 = 10^ITP(LGT($1),LGT($2),LGT($6))        /*セル中央値に対応した N 値数列の演算*/
$8 "被害度:" = $5/$7                        /*セルごと修正マイナ則被害度演算 S-N 線両振り対応*/
$9 "合計被害度:" = SUM($8)                  /*合計被害度演算*/
/*----結果シート書き込み処理-----------------------------------------------------------*/
$14 = SUM($5)
$13 "頻度数割合:%" = $5/$14*100             /* 頻度数比率演算*/
assign &2 "解析条件:" = "解析手法","Cell サイズ","Cell 個数","無効振幅%","S-N 曲線定義"
assign &3 = "RainFlow","0.5","64","63","4 点テーブル方式"
assign &4 "解析結果:" = "総頻度数="|$14(F0),"合計被害度="|$9(F6)
write ch_column 1: &2,&3,&4,$12,$6,$5,$13,$8
/*---S-N 線図グラフ処理----------------------------------------------------------------*/
$10 = RVS(3,$5,DAG(LEN($5),0.1))            /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$11 = MAX($2)                               /*  グラフ X 軸用、最大頻度数取得*/
assign &1 = "セルサイズ=0.5 セル個数=64 無効振幅=63% 合計被害度="|$9(F5)
def graph_id @1 "RainFlow 頻度解析" &1
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$11 F0 log
plot @1 $10,$2 $6,$1
def file_id %1 "freq_graph" grp
save plot %1 @1 $10,$2 $6,$1
end
```

S 値テーブルと N 値テーブルは Index ごとに対応している事が前提となります。また S 値テーブルが昇順で記述されている場合は並び替え処理の必要ありません。なお、S-N 線の N 値テーブルは両振り対応を前提としています。である必要があります。与えた S-N 曲線が片振り対応の場合は、セルごと被害度演算結果を 1/2 します。(`$8 "被害度:" = $5/$7/2`) 解析対象波形はレインフロー法記述例と同じ波形です。

＜S-N 曲線テーブル形式被害度計算結果グラフ＞



＜S-N 曲線テーブル形式被害度計算結果リスト:修正マイナー則＞

Page1: 頻度解析結果

解析条件		解析結果	セル番号	セル中央値	頻度数	頻度数割合(%)	被害度
解析手法	RainFlow	総頻度数=60222	1	0.25	10472	17.389	0.005999
Cellサイズ	0.5	合計被害度=0.065220	2	0.75	14057	23.342	0.012522
Cell個数	64		3	1.25	12828	21.301	0.014032
無効振幅%	63		4	1.75	9710	16.124	0.012159
S-N曲線定義	4点テーブル方式		5	2.25	5726	9.508	0.007932
			6	2.75	2979	4.947	0.004473
			7	3.25	1670	2.773	0.002682
			8	3.75	910	1.511	0.001548
			9	4.25	523	0.868	0.000936
			10	4.75	362	0.601	0.000677
			11	5.25	202	0.335	0.000393
			12	5.75	163	0.271	0.000329
			13	6.25	110	0.183	0.000230
			14	6.75	66	0.110	0.000142
			15	7.25	78	0.130	0.000173
			16	7.75	56	0.093	0.000128

対象とする部材或いは部位が疲労限界(これ以下のストレスであれば疲労しない)を持つ場合、被害度演算をその疲労限界値以下は演算しないマイナー則を使用します。マイナー則で被害度演算を行う場合、被害度演算時にセル中央値が設定した疲労限界以上の範囲で行います。

記述例:

前述の記述例をマイナー則で疲労限界 2 として被害度を求める場合

```
/*--------結果シート宣言処理-----------------------------------------------------------*/
dcl sheet 1 {
     page 1: "頻度解析結果"
     column &2,&3,&4,$12,$6,$5,$13,$8
     format 3(A),F0,F2,F0,F3,F6 2
}sheet
/*--------S-N 曲線テーブル定義---------------------------------------------------------*/
assign $1 = 300,100,20,1                  /*S 値テーブル*/
assign $2 = 1e2,2e4,3e5,1e6               /*N 値テーブル*/
$3 = SRT(2,$1)                            /*S 値の昇順並び Index 取得*/
$1 = QUE($3,$1)                           /*取得した Index 順に S 値テーブル並び替え*/
$2 = QUE($3,$2)                           /*取得した Index 順に N 値テーブル並び替え*/
/*-------レインフロー法頻度解析処理---------------------------------------------------*/
$5 "頻度数:" = RFM(0.5,64,63,#1)          /*ch1 のレインフロー法頻度解析処理*/
$6 "セル中央値:" = CNV(1,0.5,64)          /*セルサイズ 0.5,セル個数 64,正領域セル中央値取得*/
$12 "セル番号:" = CNV(0,0.5,64)           /*セルサイズ 0.5,セル個数 64,正領域セル番号取得*/
/*---被害度演算処理 修正マイナー則適用 両振り-----------------------------------------*/
$7 = 10^ITP(LGT($1),LGT($2),LGT($6))      /*セル中央値に対応した N 値数列の演算*/
$8 "被害度:" = GT_($6,2)*$5/$7            /*セルごとマイナ則被害度演算 S-N線両振り対応*/
$9 "合計被害度:" = SUM($8)                /*合計被害度演算*/
/*----結果シート書き込み処理-----------------------------------------------------------*/
```

```
$14 = SUM($5)
$13 "頻度数割合:%" = $5/$14*100          /* 頻度数比率演算*/
assign &2 "解析条件:" = "解析手法","Cell サイズ","Cell 個数","無効振幅%","S-N 曲線定義"
assign &3 = "RainFlow","0.5","64","63","4 点テーブル方式"
assign &4 "解析結果:" = "総頻度数="|$14(F0),"合計被害度="|$9(F6)
write ch_column 1: &2,&3,&4,$12,$6,$5,$13,$8
/*---S-N 線図結果グラフ処理-----------------------------------------------------------------*/
$10 = RVS(3,$5,DAG(LEN($5),0.1))          /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$11 = MAX($2)                             /*  グラフ X 軸用、最大頻度数取得*/
assign &1 = "セルサイズ=0.5 セル個数=64 無効振幅=63% 合計被害度="|$9(F5)
def graph_id @1 "RainFlow 頻度解析" &1
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$11 F0 log
plot @1 $10,$2 $6,$1
def file_id %1 "freq_graph" grp
save plot %1 @1 $10,$2 $6,$1
end
```

記述例赤文字で示した行がマイナー則での被害度演算行となります。比較演算関数(GT_関数)を使用してセル中央値が 2 以下の被害度を 0 とします。

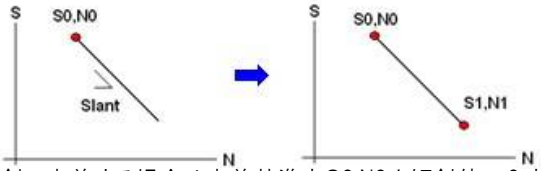
＜S-N 曲線テーブル形式被害度計算結果リスト:修正マイナー則＞

Page1: 頻度解析結果

解析条件		解析結果	セル番号	セル中央値	頻度数	頻度数割合(%)	被害度
解析手法	RainFlow	頻度数=60222	1	0.25	10472	17.389	0.000000
Cellサイズ	0.5	害度=0.020509	2	0.75	14057	23.342	0.000000
Cell個数	64		3	1.25	12828	21.301	0.000000
無効振幅%	63		4	1.75	9710	16.124	0.000000
S-N曲線定義	点テーブル方式		5	2.25	5726	9.508	0.007932
	マイナー則		6	2.75	2979	4.947	0.004473
			7	3.25	1670	2.773	0.002682
			8	3.75	910	1.511	0.001548
			9	4.25	523	0.868	0.000936
			10	4.75	362	0.601	0.000677
			11	5.25	202	0.335	0.000393
			12	5.75	163	0.271	0.000329

22. 3. 2. S-N 曲線を基準点と傾斜で定義する

S-N 曲線が LOG-LOG 尺上で S と N の関係が直線で負傾斜している場合に使用します。 言い換えると S-N 曲線定義を 2 点のテーブル方式で定義した事と同じ考え方です。



傾斜で定義する場合は定義基準点S0,N0と傾斜値の 3 点が必要となります。傾斜値(Slant)は S 値が1ディケード小さくなる時の N 値の倍率を意味します。作成する 2 点テーブルは、S1 と N1 を生成定義し、

S 値テーブル = LNK(基準 S 値/10,基準 S 値) 昇順並び
N 値テーブル = LNK(基準 N 値×倍率,基準 N 値) S 値テーブル並び対応順

定義したテーブルを用いて、ITP 関数でセル中央値に対応した S-N 曲線の N 値を生成します。

S-N 曲線 N 値数列 = ITP(LGT(S 値テーブル),LGT(N 値テーブル),LGT(セル中央値数列))

なお、N0 値及び倍率を指数部のみで扱う場合もあります。その時、N0 値は 10^{N0} 倍率は $10^{倍率}$ として、テーブル作成します。

記述例:

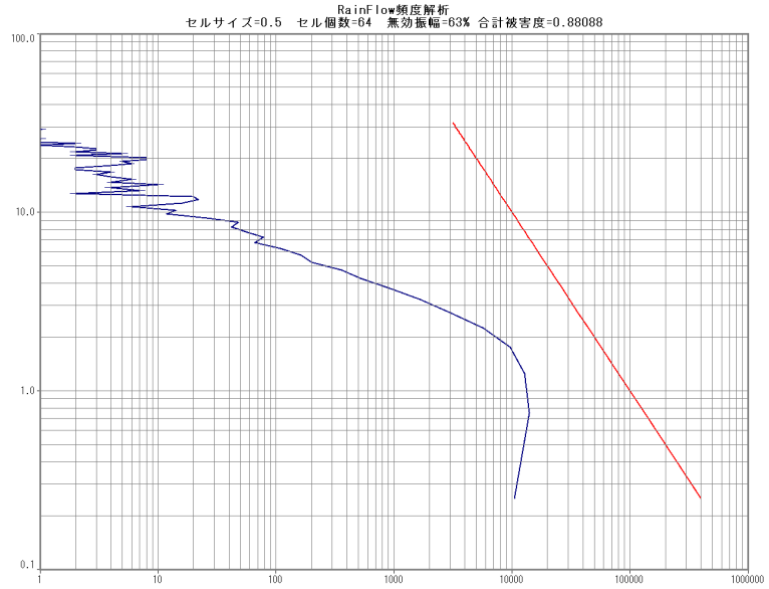
S-N 曲線を S が 100 の時、N は 1000、倍率 10 として定義し修正マイナー則で被害度演算を行う場合。

```
/*-------結果シート宣言処理-----------------------------------------------------------*/
dcl sheet 1 {
    page 1: "頻度解析結果"
    column &2,&3,&4,$12,$6,$5,$13,$8
    format 3(A),F0,F2,F0,F3,F6 2
}sheet
/*-------S-N 曲線定義:原点倍率-------------------------------------------------------*/
$1 = 100                                  /*S 値基準点*/
$2 = 1000                                 /*N 値基準点*/
$3 = 10                                   /*倍率(傾斜)*/
```

```
$1 = LNK($1/10,$1)                              /*S 値テーブル*/
$2 = LNK($2*$3,$2)                              /*N 値テーブル*/
/*------レインフロー法頻度解析処理-----------------------------------------------------*/
$5 "頻度数:" = RFM(0.5,64,63,#1)              /*ch1 のレインフロー法頻度解析処理*/
$6 "セル中央値:" = CNV(1,0.5,64)              /*セルサイズ 0.5,セル個数 64,正領域セル中央値取得*/
$12 "セル番号:" = CNV(0,0.5,64)               /*セルサイズ 0.5,セル個数 64,正領域セル番号取得*/
/*---被害度演算処理 修正マイナー則適用 両振り-----------------------------------*/
$7 = 10^ITP(LGT($1),LGT($2),LGT($6))          /*セル中央値に対応した N 値数列の演算*/
$8 "被害度:" = $5/$7                           /*セルごと被害度演算修正マイナ則 S-N 線両振り対応*/
$9 "合計被害度:" = SUM($8)                     /*合計被害度演算*/
/*----結果シート書き込み処理-----------------------------------------------------------*/
$14 = SUM($5)
$13 "頻度数割合:%" = $5/$14*100          /* 頻度数比率演算*/
assign &2 "解析条件:" = "解析手法","Cell サイズ","Cell 個数","無効振幅%","S-N 曲線定義"
assign &3 = "RainFlow","0.5","64","63","基準点傾斜方式","修正マイナー則"
assign &4 "解析結果:" = "総頻度数="|$14(F0),"合計被害度="|$9(F6)
write ch_column 1: &2,&3,&4,$12,$6,$5,$13,$8
/*---S-N 線図結果グラフ処理-------------------------------------------------------------*/
$10 = RVS(3,$5,DAG(LEN($5),0.1))              /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$11 = MAX($7)                                   /*  グラフ X 軸用、最大頻度数取得*/
assign &1 = "セルサイズ=0.5  セル個数=64  無効振幅=63% 合計被害度="|$9(F5)
def graph_id @1 "RainFlow 頻度解析" &1
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$11 F0 log
plot @1 $10,$7 $6,$6
def file_id %1 "freq_graph" grp
save plot %1 @1 $10,$7 $6,$6
end
```

解析対象波形はレインフロー法記述例と同じ波形です。

＜S-N 曲線基準点傾斜定義形式被害度計算結果グラフ＞



＜S-N 曲線基準点傾斜定義形式被害度計算結果リスト＞

Page1: 頻度解析結果

解析条件		解析結果	セル番号	セル中央値	頻度数	頻度数割合(%)	被害度
解析手法	RainFlow	総頻度数=60222	1	0.25	10472	17.389	0.026180
Cellサイズ	0.5	合計被害度=0.880885	2	0.75	14057	23.342	0.105428
Cell個数	64		3	1.25	12828	21.301	0.160350
無効振幅%	63		4	1.75	9710	16.124	0.169925
S-N曲線定義	基準､傾斜方式		5	2.25	5726	9.508	0.128835
	修正マイナー則		6	2.75	2979	4.947	0.081923
			7	3.25	1670	2.773	0.054275
			8	3.75	910	1.511	0.034125
			9	4.25	523	0.868	0.022228
			10	4.75	362	0.601	0.017195
			11	5.25	202	0.335	0.010605
			12	5.75	163	0.271	0.009373

22. 3. 3. S-N 曲線を演算式で定義する

S-N 曲線を演算式で定義する場合は、セル中央値に対応した N 値の式として記述します。例えば、セル中央値 = -2.0412×LOG(N)+78.1 とした場合は、N の式に変換して記述します。
N= EXP((セル中央値-78.1)/(-2.0412))として記述します。

記述例:

```
/*------レインフロー法頻度解析処理-----------------------------------------------------*/
$5 "頻度数:" = RFM(0.5,64,63,#1)          /*ch1 のレインフロー法頻度解析処理*/
$6 "セル中央値:" = CNV(1,0.5,64)          /*セルサイズ 0.5,セル個数 64,正領域セル中央値取得*/
$12 "セル番号:" = CNV(0,0.5,64)           /*セルサイズ 0.5,セル個数 64,正領域セル番号取得*/
/*---被害度演算処理 修正マイナー則適用 両振り-------------------------------------*/
$7 = EXP($6-78.1)/(-2.0412))              /*セル中央値に対応した N 値数列の演算*/
$8 "被害度:" = $5/$7                      /*セルごと被害度演算修正マイナ則 S-N 線両振り対応*/
$9 "合計被害度:" = SUM($8)                /*合計被害度演算*/
/*---S-N 線図結果グラフ処理-------------------------------------------------------*/
$10 = RVS(3,$5,DAG(LEN($5),0.1))          /* グラフ x 軸 Log 尺用、頻度数 0 セル修飾*/
$11 = MAX($7)                             /*  グラフ X 軸用、最大頻度数取得*/
assign &1 = "セルサイズ=0.5  セル個数=64  無効振幅=63%  合計被害度="|$9(F5)
def graph_id @1 "RainFlow 頻度解析" &1
def graph_y_axis @1 0,0 F1 5 log
def graph_x_axis @1 1,$11 F0 log
plot @1 $10,$7 $6,$6
def file_id %1 "freq_graph" grp
save plot %1 @1 $10,$7 $6,$6
end
```

22. 4. 強度解析を行う

22. 4. 1. Smith 線図を使用して強度判定する

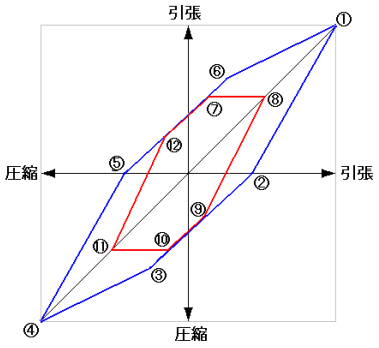
Smith 線図の各座標を計算し、応力計測データ(完全両振り)の最大振幅値(最大値-最小値)が有効枠内に入っているか否かを確認します。

求める座標

座標	X 軸値	Y 軸値
①	σ_u	σ_u
②	$\sigma_{mt}(0)$	0
③	$-\sigma_{mt}(0)$	$-2\times S_{max}(-1)$
④	$-\sigma_u$	$-\sigma_u$
⑤	$-\sigma_{mc}(0)$	0
⑥	$2/3\times\sigma_{mc}(0)$	$5/3\times S_{max}(-1)$
⑦	$\sigma_y\times\sigma_{mc}(0)/S_{max}(-1)-\sigma_{mc}(0)$	σ_y
⑧	σ_y	σ_y
⑨	$\sigma_y\times\sigma_{mc}(0)/S_{max}(-1)-\sigma_{mc}(0)$	$-(S_{max}(-1)\times(1-(\sigma_y\times\sigma_{mc}(0)/S_{max}(-1)-\sigma_{mc}(0))/\sigma_{mt}(0)))$
⑩	$-(\sigma_y\times\sigma_{mt}(0)/S_{max}(-1)-\sigma_{mt}(0))$	$-\sigma_y$
⑪	$-\sigma_y$	$-\sigma_y$
⑫	$-(\sigma_y\times\sigma_{mt}(0)/S_{max}(-1)-\sigma_{mt}(0))$	$S_{max}(-1)\times(1-(\sigma_y\times\sigma_{mt}(0)/S_{max}(-1)-\sigma_{mt}(0)))/\sigma_{mc}(0))$

入力されるパラメタ

σ_u:引張強さ
σ_y:引張降伏点
$S_{max}(-1)$:両振り引張圧縮疲労限度



入力されたパラメタから求めるパラメタ

完全片振り応力限度
引張応力限度:5/3・$S_{max}(-1)$とする
圧縮応力限度:2・$S_{max}(-1)$とする
平均応力
$\sigma_{mt}(0)$:引張平均応力は引張応力限度/2 とする $\sigma_{mc}(0)$:圧縮平均応力は圧縮応力限度/2 とする

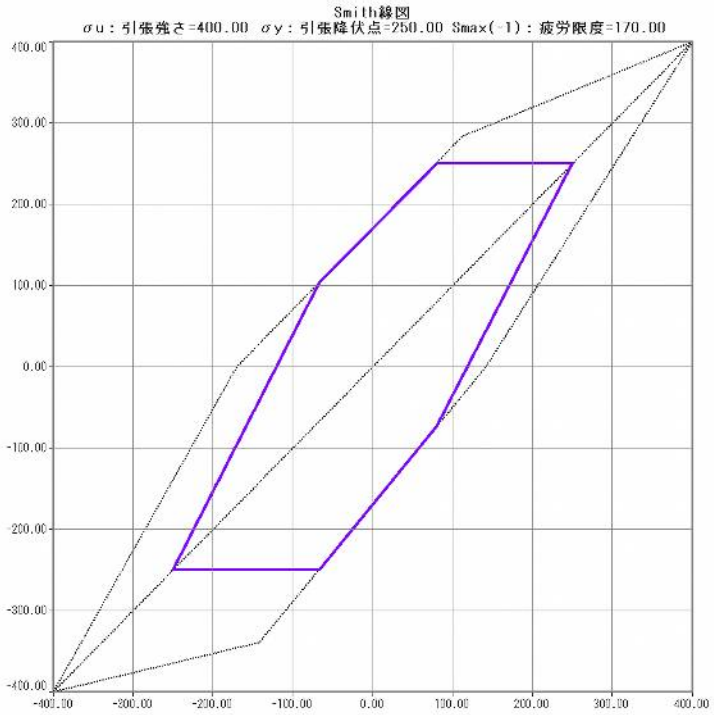
記述例:

σ_u:引張強さ 400、σ_y:引張降伏点 250、$S_{max}(-1)$:両振り引張圧縮疲労限度 170 として Smith 線図を描画する場合

```
/*----- Smith 線図 パラメタ設定-------------------------------------*/
$1 "σu:" = 400                  /* σu:引張強さ*/
$2 "σy:" = 250                  /* σy:引張降伏点*/
```

```
$3 "Smax(-1):" = 170          /* Smax(-1):両振り引張圧縮疲労限度*/
$4 "Stmax(0):" = $3*5/3
$5 "Scmax(0):" = $3*2
$6 "σmt(0):" = $4/2          /* 引張平均応力*/
$7 "σmc(0):" = $5/2          /* 圧縮平均応力*/
/*----- Smith 線図 描画処理-----------------------------------------------*/
$8 "外枠 X 座標:" = LNK($1,$6,-$6,-$1,-$7,2/3*$7,$1)
$9 "外枠 Y 座標:" = LNK($1,0,-$5,-$1,0,$4,$1)
$10 "対角線 X:" = LNK(-400,400)
$11 "対角線 Y:" = LNK(-400,400)
$12 "内枠 X 座標:" = LNK($2*$7/$3-$7,$2,$2*$7/$3-$7,-($2*$6/$3-$6),-$2,-($2*$6/$3-$6),$2*$7/$3-$7)
$13 "内枠 Y 座標:" = LNK($2,$2,-($3*(1-$12(0)/$6)),-$2,-$2,$3*(1+$12(3)/$7),$2)
$14 "GraphScaleMln:" = -$1
assign &1 "line_color:" = "#000000","#000000","#FF0080"
$15 "line_type:" = LNK(11,11,3)
assign &2 = "σu:引張強さ="|$1(F2)|" σy:引張降伏点="|$2(F2)|" Smax(-1):疲労限度="|$3(F2)
def graph_id @1 "Smith 線図" &2
def graph_aspect_ratio @1 1
def graph_x_axis @1 $14,$1 F2
def graph_y_axis @1 $14,$1 F2 5
def graph_line @1 $15 &1
plot @1 $8,$10,$12 $9,$11,$13
end
```

＜実行結果の Smith 線図＞



紫色で表示された範囲が合格範囲で、計測結果の最大振幅がグラフ縦方向に枠内入っている時、合格となります。

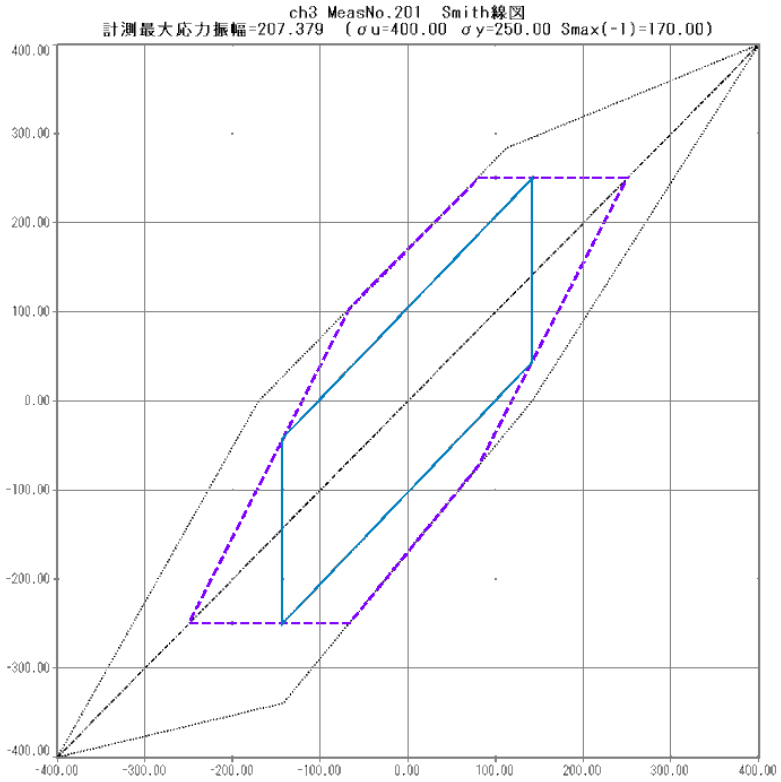
記述例:

応力計測結果データの波形表示しているカレントチャネルを最大応力振幅が Smith 線図の有効範囲にあるかを Smith 線図で確認する場合

```
/*----- Smith 線図 パラメタ設定-----------------------------------------*/
$1 "σu 引張強さ:" = 400          /* σu:引張強さ*/
$2 "σy 引張降伏点:" = 250          /* σy:引張降伏点*/
$3 "Smax(-1)疲労限度:" = 170       /* Smax(-1):両振り引張圧縮疲労限度*/
get value $1(F3),$2(F3),$3(F3)    /* パラメタ入力*/
$4 "Stmax(0):" = $3*5/3
$5 "Scmax(0):" = $3*2
$6 "σmt(0):" = $4/2               /* 引張平均応力*/
$7 "σmc(0):" = $5/2               /* 圧縮平均応力*/
/*----- Smith 線図 座標演算処理-----------------------------------------*/
$8 "外枠 X 座標:" = LNK($1,$6,-$6,-$1,-$7,2/3*$7,$1)
```

```
$9 "外枠 Y 座標:" = LNK($1,0,-$5,-$1,0,$4,$1)
$10 "対角線 X:" = LNK(-400,400)
$11 "対角線 Y:" = LNK(-400,400)
$12 "内枠 X 座標:" = LNK($2*$7/$3-$7,$2,$2*$7/$3-$7,-($2*$6/$3-$6),-$2,-($2*$6/$3-$6),$2*$7/$3-$7)
$13 "内枠 Y 座標:" = LNK($2,$2,-($3*(1-$12(0)/$6)),-$2,-$2,$3*(1+$12(3)/$7),$2)
/* -----有効判定 非線形テーブル定義-----------------------------------------*/
$16 "TBL1_X:" = LNK($12(4),$12(3),$12(2),$12(1))          /*座標番号⑪,⑩,⑨,⑧*/
$17 "TBL1_Y:" = LNK($13(4),$13(3),$13(2),$13(1))
$18 "TBL2_X:" = LNK($12(4),$12(5),$12(0),$12(1))          /*座標番号⑪,⑫,⑦,⑧*/
$19 "TBL2_Y:" = LNK($13(4),$13(5),$13(0),$13(1))
$22 = ABS($12(4)-$12(1))                                    /*Smith 演算応力幅*/
$22 "応力範囲:" = (ACC(DAG(1000,1))-1)*$22/1000+$12(4)   /* 演算応力範囲数列生成*/
$30 "上側:" = ITP($18,$19,$22)                             /* Smith 判定上限数列生成*/
$31 "下側:" = ITP($16,$17,$22)                             /* Smith 判定下限数列生成*/
$23 "Smith 振幅:" = $30-$31                                 /* Smith 判定振幅数列生成*/
/*-----カレントチャネル最大応力振幅演算＆合否判定-----------------------------*/
$20 "カレントチャネル番号:" = CCH()                /* カレントチャネル番号取得*/
$21 "最大応力振幅:" = MAX(#($20))-MIN(#($20)) /* 計測 ch 最大応力振幅*/
$24 "判定フラグ:" = SUM(GT_($23,$21))     /* 合否判定:Smith 判定振幅>計測最大応力振幅*/
$33 "合格フラグ:" = 0                       /* 合格フラグ初期化*/
case $24 >= 1                               /* 一箇所でもあれば合格*/
  proc 合格{
    /*------ 合格範囲描画アドレス生成-----------------------------------------*/
    $25 = DTD(0,$21,0,MXP($23),$23) /*下降通過,閾値:最大振幅,降順,開始:smith 最大*/
    $26 = DTD(0,$21,1,MXP($23),$23) /*下降通過,閾値:最大振幅,降順,開始:smith 最大*/
    $27 "範囲 X:" = LNK(PTV($25,$22),PTV($26,$22),PTV($26,$22),PTV($25,$22),PTV($25,$22))
    $28 "範囲 Y:" = LNK(PTV($25,$30),PTV($26,$30),PTV($26,$31),PTV($25,$31),PTV($25,$30))
    $29 "pen_ctrl:" = DAG(5,1)
    $33 = 1          /*  合格フラグ set*/
  }合格
/*-----smith 線図 描画処理-----------------------------------------------------*/
assign &1 "line_color:" = "#000000","#000000","#FF0080" /*描画線色設定*/
$15 "line_type:" = LNK(11,31,23)                          /*描画線種設定*/
assign &2 = "計測最大応力振幅="|$21(F3)|" (σu="|$1(F2)|" σy="|$2(F2)|" Smax(-1)="|$3(F2)|")"
assign &4 = CHNM($20)                    /* チャネル名取得*/
assign &3 = "ch"|$20(F0)|" "|&4|" Smith 線図"
def graph_id @1 &3 &2
def graph_aspect_ratio @1 1               /*グラフ番号定義*/
$14 "GraphScaleMin:" = -$1                /*グラフ左端/下端値*/
def graph_x_axis @1 $14,$1 F2             /*グラフ X 軸定義*/
def graph_y_axis @1 $14,$1 F2 5           /*グラフ Y 軸定義*/
def graph_line @1 $15 &1                  /* グラフ線種/線色定義*/
case $33 = 1
  proc 合格線{
    def graph_line_draw @1 $27,$28,$29 2 "#C08000" /*合格範囲描画*/
  }合格線
plot @1 $8,$10,$12 $9,$11,$13             /*Smith 線図描画*/
def file_id %1 "graph" grp               /*グラフ格納先ファイル番号定義*/
save plot %1 @1 $8,$10,$12 $9,$11,$13 /*Smith 線図格納*/
end
```

＜実行結果の Smith 線図＞



合格した場合、青色範囲を描画し、計測データの応力最大振幅合格範囲を示します。不合格の場合は青色枠は描画しません。

記述例22. 1. フォルダ内に格納されているファイルを一括頻度処理する 指定したフォルダ内に格納されている計測データファイルを選択し、指定した単位と同じ収録チャネルを一括頻度解析し、解析対 象ファイルごとに頻度解析結果を csv ファイルに格納する。

<Archi_1 Script 記述例>

```
/* ---------記述例21.1-------------------------*/
dcl menu_label "一括頻度解析"
def ch_name &1 "file_name"
def ch_name &2 "解析対象チャネル単位"
def ch_name &3 "methode"
def ch_name &4 "頻度解析手法"
def ch_name &5 "file_date_stump"
def ch_name &6 "file_time_stump"
def ch_name &7 "ch_unit"
def ch_name &8 "格納先ファイル名"
def ch_name $1 "num_file"
def ch_name $2 "file_counter"
def ch_name $3 "セルサイズ"
def ch_name $4 "num_cell"
def ch_name $5 "オフセット値(時間率頻度解析時有効)"
def ch_name $6 "file_select_flag"
def ch_name $7 "ch_counter"
def ch_name $8 "num_ch"
def ch_name $9 "ch_series"
def ch_name $10 "頻度解析実行フラグ"
def ch_name $11 "頻度"
def ch_name $12 "セル番号"
def ch_name $13 "セル中央値"
def ch_name $14 "頻度%"
def ch_name $15 "セル個数"
def ch_name $16 "無効振幅%"
def ch_name $17 "書き込み列番号"
def ch_name $18 "fileSaveFlag"
def ch_name $19 "total_count"
$18 = 0
/*---解析条件設定-----------------------------------------------*/
assign &2 = "Pa"
assign &3 = "RainFlow","Peak/Valley","Min/Max","LevelCross","TimeRate"
assign &4 = &3(0)          /*  解析手法初期値ReinFlow*/
$3 = 0
repeat_case $3 <= 0
  proc condition_set{
    $3 = 0.5                /*  セルサイズ初期値0.5*/
    $5 = 0                  /*  オフセット初期値0*/
    $16 = 0                 /*  無効振幅初期値0%*/
    get value &3:&4,&2,$3(F2),$16(F1),$5(F2)
  }condition_set
/*---- 解析対象ファイル格納先選択-----------------------------*/
get folder_select
read file_info &1 &5 &6 $1 "hdr"   /* $1は存在ファイル数*/
case $1 > 0
  proc file_select{
    $6 = DAG(LEN($1),1)
    get check_box_status &1 $6 "解析するファイルを選択して下さい"
    $1 = SUM($6)
    case $1 > 0
      proc exec{
        &1 = CREC($6,&1)              /* 選択されたファイルだけ再構成*/
        /* ---- ファイル  ループ-----------------------------------*/
        $2 = 0
        repeat_case $2 < $1
          proc file_loop{
            assign &9 "ProgressMessage:" = &1($2)|"  読み込み "|$2(F0)|"/"|$1(F0)
            write progress_status &9
            def file_id %1 &1($2) wav       /* 解析対象ファイル番号定義*/
            read wave %1                    /* 解析対象ファイル読み出し*/
```

```
assign &8 = &1($2)|"_"|&4|"頻度解析結果"
def file_id %2 &8 csv          /* 頻度解析結果ファイル番号定義*/
$10 = 0                        /* 解析実行結果存在フラグ初期化*/
$8 = NCH()                     /* 収録チャネル数取得*/
$9 = CHS()                     /* 収録チャネル番号取得*/
$17 = $8*4
def text_form %2 1010,$17      /* 頻度解析結果書き込み仮想シート定義*/
write cell %2 1,1 1 "解析手法","セルサイズ","無効振幅%"
write cell %2 2,1 1 &4,$3(F2),$16(F2)
/* ----- ファイル内チャネル  ループ ------------------------*/
$7 = 0                         /* ch counter init*/
repeat_case $7 < $8
  proc ch_loop{
    &7 = CUNT($9($7))          /* 収録チャネル単位取得*/
    assign &9 = &1($2)|"ch"|$9($7)(F0)|" "|$2(F0)|"/"|$1(F0)
    write progress_status &9
    case &7 = &2               /* 収録チャネル単位一致*/
      proc freq_anal{
        /* ----- 頻度解析実行----------------------------------*/
        assign &9 = &1($2)|"ch"|$9($7)(F0)|" "|&4|" "|$2(F0)|"/"|$1(F0)
        write progress_status &9
        case &4 = "RainFlow"
          proc rainflow{
            $15 = INT((MAX(#($9($7)))-MIN(#($9($7))))/$3+1) /* セル個数取得*/
            $11 = RFM($3,$15,$16,#($9($7)))        /* RainFlow 解析*/
            $12 = CNV(0,$3,$15)                    /* セル番号取得*/
            $13 = CNV(1,$3,$15)                    /* セル中央値取得*/
            $14 = $11/SUM($11)*100                 /* 頻度%演算*/
          }rainflow
        case &4 = "Peak/Valley"
          proc peakvalley{
            $15 = INT(MAX(ABS(#($9($7))))/$3)*2    /* セル個数取得*/
            $11 = PVM($3,$15,$16,#($9($7)),1)      /* 極大/極小解析*/
            $12 = CNV(0,$3,$15,1)                  /* セル番号取得*/
            $13 = CNV(1,$3,$15,1)                  /* セル中央値取得*/
            $14 = $11/SUM($11)*100                 /* 頻度%演算*/
          }peakvalley
        case &4 = "Min/Max"
          proc minmax{
            $15 = INT(MAX(ABS(#($9($7))))/$3)*2    /* セル個数取得*/
            $11 = MMM($3,$15,$16,#($9($7)))        /* 最大/最小解析*/
            $12 = CNV(0,$3,$15,1)                  /* セル番号取得*/
            $13 = CNV(1,$3,$15,1)                  /* セル中央値取得*/
            $14 = $11/SUM($11)*100                 /* 頻度%演算*/
          }minmax
        case &4 = "LevelCross"
          proc levelcross{
            $15 = INT(MAX(ABS(#($9($7))))/$3)*2    /* セル個数取得*/
            $11 = LCR($3,$15,$16,#($9($7)))        /* レベルクロス解析*/
            $12 = CNV(0,$3,$15,1)                  /* セル番号取得*/
            $13 = CNV(1,$3,$15,1)                  /* セル中央値取得*/
            $14 = $11/SUM($11)*100                 /* 頻度%演算*/
          }levelcross
        case &4 = "TimeRate"
          proc timerate{
            $15 = INT(MAX(ABS(#($9($7)))-$5)/$3)*2 /* セル個数取得*/
            $11 = TRC($3,$15,#($9($7)))            /* 時間率解析*/
            $12 = CNV(0,$3,$15,1)                  /* セル番号取得*/
            $13 = CNV(1,$3,$15,1)+$5               /* セル中央値取得*/
            $14 = $11/SUM($11)*100                 /* 頻度%演算*/
          }timerate
        /*---- 結果書き込み----------------------------------*/
        $17 = $10*4+1                              /* 書き込み列カウンタ更新*/
        assign &9 = &1($2)|"結果書き込み "|$2(F0)|"/"|$1(F0)
        write progress_status &9
        &9 = CHNM($9($7))
        $19 = SUM($11)
```

```
129                         assign &9 = "ch"|$9($7)(F0),&9," ",$19(F0)
130                         assign &10 = "Cell_No.","Cell 値:"|&2,"頻度数","%"
131                         write cell %2 3,$17 0 &9,&10
132                         write cell %2 5,$17 1 $12(F0),$13(F2),$11(F0),$14(F3)
133                         $10 = $10+1
134                       }freq_anal
135                     $7 = $7+1              /* ch counter up*/
136                   }ch_loop
137                 /* ----- 結果格納----------------------------------------------*/
138                 case $10 > 0
139                   proc result_save{
140                     assign &9 = &1($2)|" 結果格納  "|$2(F0)|"/"|$1(F0)
141                     write progress_status &9
142                     save text_form %2
143                     $18 = $18+1
144                   }result_save
145                 $2 = $2+1                  /* file counter up*/
146               }file_loop
147           }exec
148       }file_select
149     case $18 = 0
150       proc file_not_save{
151         disp message "頻度解析は一致した単位が有りませんでした"
152       }file_not_save
153     end
```

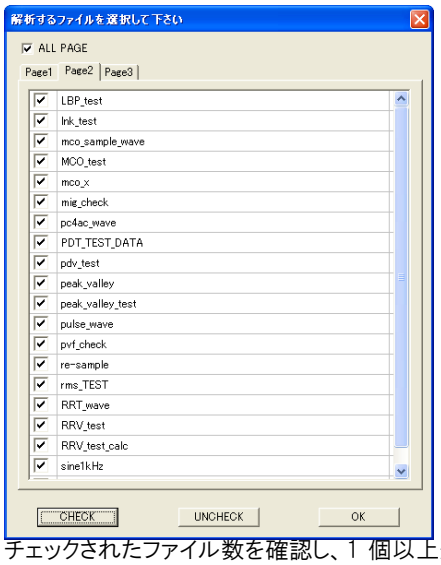
<Archi_1 Script 記述構文の説明>

31 行目～42 行目：頻度解析条件を設定します。頻度解析条件は頻度解析手法、解析対象チャネルの単位、セルサイズ、無効振幅及び時間率頻度解析設定時のオフセット値を設定します。設定はセルサイズがゼロより大きい場合に LOOP を抜ける様にします。

44 行目：解析対象ファイル格納先フォルダを選択します。

45 行目：フォルダ内に格納されている拡張子".hdr"ファイルのファイル名を取得します。

49 行目：フォルダ内に格納されているファイル名を表示し解析対象ファイルを選択します。



50 行目：チェックされたファイル数を確認し、1 個以上選択されている場合に次に進みます。

53 行目：チェックされたファイル名だけにファイル名配列を再構成します。

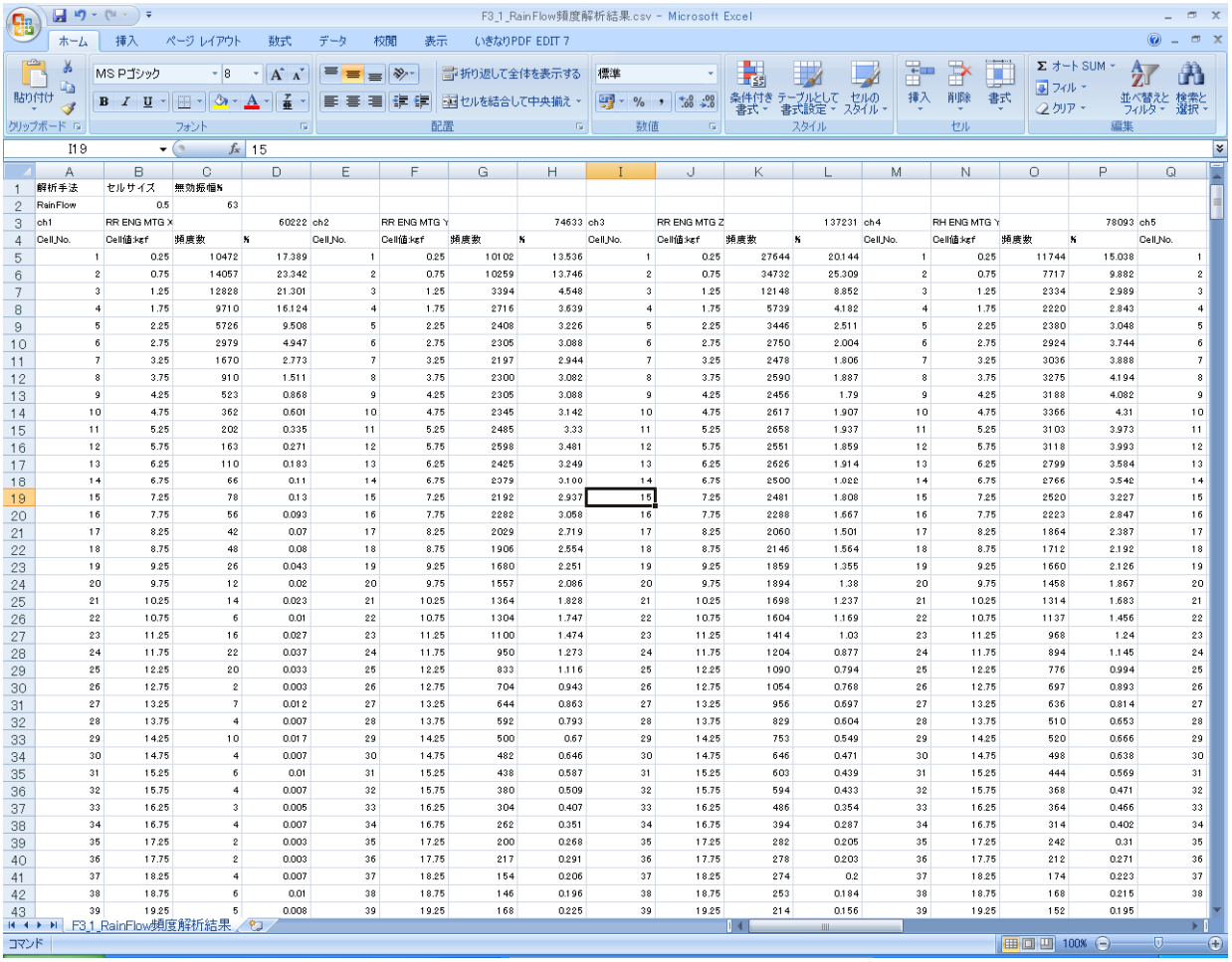
56 行目：ファイル数分の繰り返し LOOP です。

60 行目: 解析対象ファイル番号を定義します。
61 行目: 解析対象ファイルを読み出します。
63 行目: 頻度解析結果格納先ファイル番号を定義します。ファイル名は解析対象ファイル名に頻度解析手法と"頻度解析結果"を付加したファイル名で定義します。_
65 行目: 解析対象ファイルの収録チャネル数を取得します。
66 行目: 解析対象ファイルの収録チャネル番号を取得します。
67 行目: 解析結果書き込み用仮想シートの列数を計算します。列数はチャネルごとに 4 列占有します。
68 行目: 解析結果書き込み用仮想シートを定義します。行数は頻度解析結果行が 1004 行までと少し大きめに定義しています。
69 行目～70 行目:仮想シートの 1 行目と 2 行目に解析条件を書き込みます。
73 行目: 解析対象ファイル内のチャネルループです。解析は 1ch ごとに単位が一致しているか確認し一致しているチャネルのみ頻度解析します。
75 行目: 解析対象チャネルの単位を取得します。
78 行目: 解析対象チャネルの単位と解析条件で設定した単位の一致を検査します。
79 行目～134 行目:頻度解析実行演算処理ブロックです。
84 行目～90 行目:レインフロー法頻度解析演算処理ブロックです。
85 行目: レインフロー法頻度解析を行う場合の合計セル個数を演算します。
86 行目: レインフロー法頻度解析を行います。
87 行目: セル番号数列を取得します。
88 行目:セル中央値(代表値)数列を取得します。
89 行目: 当該セルが合計頻度数の何%で在るかを演算します。
92 行目～98 行目:極大/極小法頻度解析演算処理ブロックです。
93 行目: 極大/極小法頻度解析を行う場合の合計セル個数を演算します。
94 行目: 極大/極小法頻度解析を行います。正領域は極大値を計数、負領域は極小値を計数します。
95 行目: セル番号数列を取得します。
96 行目: セル中央値(代表値)数列を取得します。
97 行目: 当該セルが合計頻度数の何%で在るかを演算します。
100 行目～106 行目:最大/最小法頻度解析演算処理ブロックです。
101 行目:最大/最小法頻度解析を行う場合の合計セル個数を演算します。
102 行目:最大/最小法頻度解析を行います。
103 行目:セル番号数列を取得します。
104 行目:セル中央値(代表値)数列を取得します。
105 行目:当該セルが合計頻度数の何%で在るかを演算します。
108 行目～114 行目:レベルクロス法頻度解析演算処理ブロックです。
109 行目:レベルクロス法頻度解析を行う場合の合計セル個数を演算します。
110 行目:レベルクロス法頻度解析を行います。
111 行目:セル番号数列を取得します。
112 行目:セル中央値(代表値)数列を取得します。
113 行目:当該セルが合計頻度数の何%で在るかを演算します。
116 行目～122 行目:時間率頻度解析演算処理ブロックです。
117 行目:時間率法頻度解析を行う場合の合計セル個数を演算します。
118 行目:時間率頻度解析を行います。
119 行目:セル番号数列を取得します。
120 行目:セル中央値(代表値)数列を取得します。
121 行目:当該セルが合計頻度数の何%で在るかを演算します。
124 行目～132 行目:仮想シートに頻度解析結果を書き込みます。
124 行目:仮想シート書き込み列番号を演算します。
127 行目:解析対象チャネル名を取得します。
128 行目:合計頻度数を演算します。
129 行目～131 行目:解析チャネルごとヘッダー行を仮想シートに書き込みます。
132 行目:セル番号、セル中央値、頻度数、割合を仮想シートに書き込みます。
135 行目:チャネルループカウンタをインクリメントします。
139 行目～144 行目:仮想シートを格納する演算処理ブロックです。仮想シートへの書き込みが行われない場合は格納しません。
145 行目:ファイルループカウンタをインクリメントします。

※ 記述した Script では、仮想シートに書き込んだ頻度解析結果は表示せず、解析対象ファイルごとに、csv ファイルを生成します。csv ファイルを読み出す場合、エクセル等を使用して読み出します。

※ 頻度解析に当たって、セルの合計個数は設定したセルサイズと解析対象チャネルの最大/最小値から全ての範囲を含む様に 自動設定しています。但し、セルの合計個数が 1004 を越える場合は、超えたセル番号のデータは格納されません。

<Archi_1 Script 実行結果リスト:エクセルで表示>



F3_1_RainFlow頻度解析結果.csv - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	解析手法	セルサイズ	無効振幅%														
2	RainFlow	0.5	63														
3	ch1	RR ENG MTG X		60222	ch2	RR ENG MTG Y		74633	ch3	RR ENG MTG Z		137231	ch4	RH ENG MTG Y		78093	ch5
4	Cell_No.	Cell値kgf	頻度数	%	Cell_No.	Cell値kgf	頻度数	%	Cell_No.	Cell値kgf	頻度数	%	Cell_No.	Cell値kgf	頻度数	%	Cell_No.
5	1	0.25	10472	17.389	1	0.25	10102	13.536	1	0.25	27644	20.144	1	0.25	11744	15.038	1
6	2	0.75	14057	23.342	2	0.75	10259	13.746	2	0.75	34732	25.309	2	0.75	7717	9.882	2
7	3	1.25	12828	21.301	3	1.25	3394	4.548	3	1.25	12148	8.852	3	1.25	2334	2.989	3
8	4	1.75	9710	16.124	4	1.75	2716	3.639	4	1.75	5739	4.182	4	1.75	2220	2.843	4
9	5	2.25	5726	9.508	5	2.25	2408	3.226	5	2.25	3446	2.511	5	2.25	2380	3.048	5
10	6	2.75	2979	4.947	6	2.75	2305	3.088	6	2.75	2750	2.004	6	2.75	2924	3.744	6
11	7	3.25	1670	2.773	7	3.25	2197	2.944	7	3.25	2478	1.806	7	3.25	3036	3.888	7
12	8	3.75	910	1.511	8	3.75	2300	3.082	8	3.75	2590	1.887	8	3.75	3275	4.194	8
13	9	4.25	523	0.868	9	4.25	2305	3.088	9	4.25	2456	1.79	9	4.25	3188	4.082	9
14	10	4.75	362	0.601	10	4.75	2345	3.142	10	4.75	2617	1.907	10	4.75	3366	4.31	10
15	11	5.25	202	0.335	11	5.25	2485	3.33	11	5.25	2658	1.937	11	5.25	3103	3.973	11
16	12	5.75	163	0.271	12	5.75	2598	3.481	12	5.75	2551	1.859	12	5.75	3118	3.993	12
17	13	6.25	110	0.183	13	6.25	2425	3.249	13	6.25	2626	1.914	13	6.25	2799	3.584	13
18	14	6.75	66	0.11	14	6.75	2379	3.188	14	6.75	2500	1.822	14	6.75	2766	3.542	14
19	15	7.25	78	0.13	15	7.25	2192	2.937	15	7.25	2481	1.808	15	7.25	2520	3.227	15
20	16	7.75	56	0.093	16	7.75	2282	3.058	16	7.75	2288	1.667	16	7.75	2223	2.847	16
21	17	8.25	42	0.07	17	8.25	2029	2.719	17	8.25	2060	1.501	17	8.25	1864	2.387	17
22	18	8.75	48	0.08	18	8.75	1906	2.554	18	8.75	2146	1.564	18	8.75	1712	2.192	18
23	19	9.25	26	0.043	19	9.25	1680	2.251	19	9.25	1859	1.355	19	9.25	1660	2.126	19
24	20	9.75	12	0.02	20	9.75	1557	2.086	20	9.75	1894	1.38	20	9.75	1458	1.867	20
25	21	10.25	14	0.023	21	10.25	1364	1.828	21	10.25	1698	1.237	21	10.25	1314	1.683	21
26	22	10.75	6	0.01	22	10.75	1304	1.747	22	10.75	1604	1.169	22	10.75	1137	1.456	22
27	23	11.25	16	0.027	23	11.25	1100	1.474	23	11.25	1414	1.03	23	11.25	968	1.24	23
28	24	11.75	22	0.037	24	11.75	950	1.273	24	11.75	1204	0.877	24	11.75	894	1.145	24
29	25	12.25	20	0.033	25	12.25	833	1.116	25	12.25	1090	0.794	25	12.25	776	0.994	25
30	26	12.75	2	0.003	26	12.75	704	0.943	26	12.75	1054	0.768	26	12.75	697	0.893	26
31	27	13.25	7	0.012	27	13.25	644	0.863	27	13.25	956	0.697	27	13.25	636	0.814	27
32	28	13.75	4	0.007	28	13.75	592	0.793	28	13.75	829	0.604	28	13.75	510	0.653	28
33	29	14.25	10	0.017	29	14.25	500	0.67	29	14.25	753	0.549	29	14.25	520	0.666	29
34	30	14.75	4	0.007	30	14.75	482	0.646	30	14.75	646	0.471	30	14.75	498	0.638	30
35	31	15.25	6	0.01	31	15.25	438	0.587	31	15.25	603	0.439	31	15.25	444	0.569	31
36	32	15.75	4	0.007	32	15.75	380	0.509	32	15.75	594	0.433	32	15.75	368	0.471	32
37	33	16.25	3	0.005	33	16.25	304	0.407	33	16.25	486	0.354	33	16.25	364	0.466	33
38	34	16.75	4	0.007	34	16.75	262	0.351	34	16.75	394	0.287	34	16.75	314	0.402	34
39	35	17.25	2	0.003	35	17.25	200	0.268	35	17.25	282	0.205	35	17.25	242	0.31	35
40	36	17.75	2	0.003	36	17.75	217	0.291	36	17.75	278	0.203	36	17.75	212	0.271	36
41	37	18.25	4	0.007	37	18.25	154	0.206	37	18.25	274	0.2	37	18.25	174	0.223	37
42	38	18.75	6	0.01	38	18.75	146	0.196	38	18.75	253	0.184	38	18.75	168	0.215	38
43	39	19.25	5	0.008	39	19.25	168	0.225	39	19.25	214	0.156	39	19.25	152	0.195	

23. 演算関数を使用して計測データの統計解析を行う

23. 1. チャネルごとの分布/基本統計量を求める

23. 1. 1. 収録チャネルのヒストグラムを描画する 演算対象チャネルの時間率頻度解析を行い、結果をヒストグラム表現します。
時間率頻度解析を行う場合【TRC 関数】を 使用します。

TRC 関数の文法：

結果格納先 = TRC(セルサイズ,セル合計個数,解析対象数列)

※ TRC 関数の詳細は 22 章「22.1.1.時間率頻度解析を行う」項を参照下さい。

セルサイズとセル合計個数を求める演算手順：

時間率頻度解析関数(TRC 関数)の引数として必要なセルサイズ、合計セル個数が既知の場合は、関数の引数にそのまま記述しますが、セルサイズ、合計セル数が不明な場合に解析対象データから求める方法を記述します。
演算は、片領域の概略セル個数と正負何れかの領域の絶対値の大きい方を指定し、最適なセルサイズとセル個数を求める演算を行います。なお、セルサイズはヒストグラム描画を意識して 1,2,5 ステップの倍数となる様に求めます。

① 絶対値最大と概略片領域セル個数の割り算を行い、結果の $\log_{10}$ をとり、指数部を求める
指数部 = INT(LOG(絶対値最大/概略片領域セル個数)

② ①で求めた指数部が負数の場合、指数部を-1 する
指数部 = 指数部-LT_(指数部,0)

③ セルサイズ数列を生成する(1,2,5 ステップの初期値を用意し、求めた指数部を使用して 10 のべき乗する)
セルサイズ数列 = LNK(0.1,0.2,0.5,1,2,5,10,20,50)*10^指数部
※ セルサイズ数列は 1,2,5 ステップで求めたセルサイズ採用候補数列を意味します。

④ セル個数数列を生成する(絶対値最大をセルサイズで割り、小数点以下が存在した場合+1 する)
小数点以下が存在するか否かの演算
小数点以下存在フラグ = GT_(絶対値最大/セルサイズ-INT(絶対値最大/セルサイズ),0) 存在した場合：(小数点以下存在フラグ = 1 の時)
セル個数数列 = INT(絶対値最大/セルサイズ)+1) 存在しない場合：(小数点以下存在フラグ = 0 の時
セル個数数列 = 絶対値最大/セルサイズ
※ セル個数数列は、片領域のセル個数採用候補数列を意味します。

⑤ 設定された概略片領域セル個数に最も近いセル個数数列の index を取得する
最適 index = MNP(ABS(概略片領域セル個数-セル個数数列))

⑥ セルサイズ数列の最適 Index 値を抽出してセルサイズとする
セルサイズ = PTV(最適 index,セルサイズ数列)

⑦ セル個数数列の最適 index 値を抽出して 2 倍しセル合計個数とする
セル合計個数 = PTV(最適 index,セル個数数列)*2

記述例：

セルサイズとセル合計個数を求める外部演算処理ブロックを記述する

```
/* ---セルサイズ、セル個数を求める外部演算処理ブロック----------*/
proc CellSize{
 /*
  呼び出し方向：
    call proc CellSize 絶対値最大,概略片領域セル数,セルサイズ,合計セル数
  引数：
    $1：<入力> 絶対値最大    $2：<入力> 概略片領域セル数
    $3：<出力> セルサイズ    $4 <出力> 合計セル個数
-------------------------------------------------------------------------*/
  def inherit_ch $1,$2,$3,$4
  def local_ch $5,$6
  $5 = INT(LGT($1/$2))     /* 指数部演算*/
  $5 = $5-LT_($5,0)        /* 指数部負数時-1処理*/
  $3 "セルサイズ:" = LNK(0.1,0.2,0.5,1,2,5,10,20,50)*10^$5
  $4 "セル個数:" = RVS(GT_($1/$3-INT($1/$3),0),$1/$3,INT($1/$3)+1)
  $5 "最適 index:" = MNP(ABS($2-$4)) /*最も差異の少ない index 取得*/
  $3 = PTV($5,$3)                    /*最適セルサイズ*/
  $4 = PTV($5,$4)*2                  /*合計セル個数*/
}CellSize
```

絶対値最大が 1.214、概略片領域セル個数を 100 とした場合、実行結果のセルサイズは 0.01、合計セル個数は 244 と戻ります。 又、概略片領域セル個数を 64 とした場合、実行結果のセルサイズは 0.02、合計セル個数は 122 と戻ります。

時間率頻度解析対象数列の記述方法:

TRC 関数では、与えられたセル個数の 1/2 を片領域として正負両領域の時間率頻度解析を行います。例えば、合計セル個数が 64 とするとセル番号は-32～-1,1～32 となります。その場合、セル番号-1 は-セルサイズ<=データ<0 が計数され、セル番号 1 は 0<=データ<セルサイズが計数される為、ゼロを跨ぐセルは存在しません。その為、TRC 関数に与える解析対象数列からセルサイズ/2 を減算して引数とします。

解析対象数列が全体にセルサイズ/2 だけ負方向に移動し、セル番号 1 は-セルサイズ/2<=データ<セルサイズ/2 を計数することができます。

時間率頻度解析結果格納先 = TRC(セルサイズ,合計セル個数,解析対象数列-セルサイズ/2)

ヒストグラムの描画方法:

ヒストグラムは一般的には棒グラフで表現しますが、グラフ描画線種に Bar (棒)を指定した場合、グラフの X 軸は描画するデータの Index 固定となり、任意の X 軸を採る事ができません。従って、ヒストグラム(棒グラフ)の場合は、目盛位置と表示する目盛値を明示的に指定する必要があります。

表示目盛値とその目盛位置を求める演算手順:

① 目盛数は 7 本以上を目安としてセルサイズの 5 倍間隔で目盛線 Index 間隔を求める。
目盛線 Index 間隔 = INT(INT(合計セル数/7)/5)*5

② 正領域仮目盛値表示 Index 数列を生成します。
仮想 index 数列 = ACC(DAG(INT(合計セル数/2/目盛 index 間隔),目盛 index 間隔))

③ 仮目盛値表示 Index を正負領域に拡張する
仮想 index 数列 = LNK(REV(SGN(仮想 index 数列)),0,仮想 index 数列)

④ 表示目盛 index 数列を生成する
目盛表示 Index 数列 = 仮想 Index 数列+合計セル数/2

⑤ 表示目盛値数列を生成する
表示目盛数列 = 仮想 Index 数列*セルサイズ

⑥ 表示目盛値数列を文字列に変換する
assign 表示目盛文字列 = 表示目盛数列(表示フォーマット指定)

この結果をグラフ X 軸定義に記述します。
def graph_x_axis @グラフ番号 0,0 表示目盛文字列,目盛値表示 Index 数列

記述例:

表示目盛値と目盛表示位置を求める外部演算処理ブロックを記述する。

```
/*--------棒グラフ X 軸目盛生成外部演算処理ブロック----------------------------*/
proc X_axis_scale[
  /*
   呼び出し方法:
      call proc X_axis_scale セルサイズ,セル合計個数,位置 index,目盛文字列 引
   数:
      $1 <入力> セルサイズ   $2 <入力> セル合計個数
      $3 <出力> 位置 index    &1 <出力> 目盛文字列
-------------------------------------------------------------------------------*/
  def inherit_ch $1,$2,$3,&1
  def local_ch $5,$6
  $5 "目盛 index 間隔:" = INT(INT($2/7)/5)*5  /*目盛表示 7 箇所以上セルサイズの 5 倍として*/
  $6 = ACC(DAG(INT($2/2/$5),$5))              /*仮正領域目盛位置数列生成*/
  $6 = LNK(REV(SGN($6)),0,$6)                 /*正負領域目盛位置数列生成*/
  $5 "目盛値数列:" = $6*$1                    /*目盛値数列演算*/
  $3 "目盛位置 index:" = $6+$2/2              /*目盛位置 index 数列演算*/
  assign &1 = $5(F2)                          /*目盛文字列変換*/
]X_axis_scale
```

セルサイズ 0.02、セル合計個数 66 とした場合、

目盛文字列&1 は-0.6,-0.5,-0.4,-0.3,-0.2,-0.1,0,0.1,0.2,0.3,0.4,0.5,0.6 が戻り、目盛 index$3 は 3,8,13,18,23,28,33,38,43,48,53,58,63 が戻ります。

記述例:

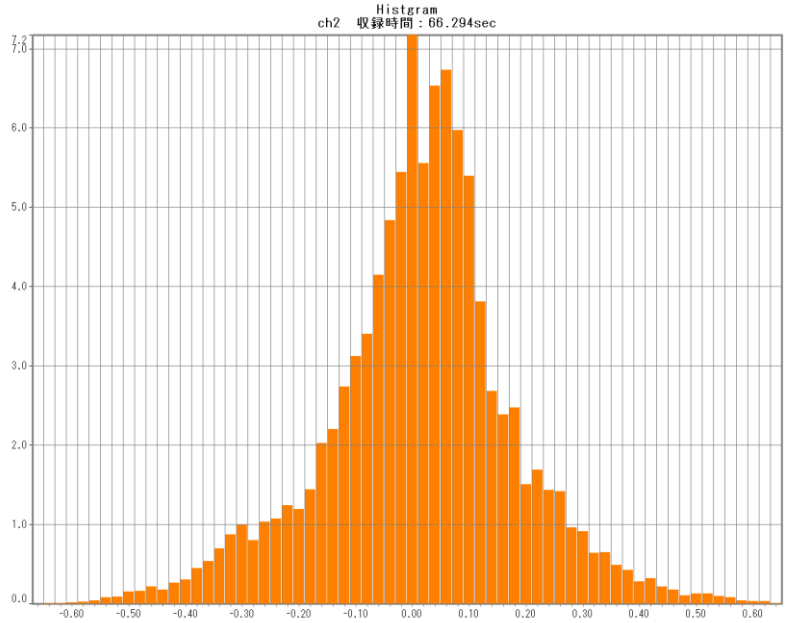
収録データ･カレントチャネルを概略片領域セル個数 32 としてヒストグラムを描画する場合

```
/*---カレントチャネルのヒストグラム----------------------*/
$1 "カレントチャネル番号:" = CCH()
$2 "絶対値最大:" = RVS(3,ABS(MAX(#($1))),ABS(MIN(#($1))))
$3 "概略片領域セル個数:" = 32
def ch_name $4 "セルサイズ"
def ch_name $5 "セル個数"
call proc CellSize $2,$3,$4,$5      /*外部演算処理ブロック呼び出し*/
$6 = TRC($4,$5,#($1)-$4/2)          /*時間率頻度解析*/
$7 = $6/SUM($6)*100                 /*Y 軸を%に変換*/
```

```
/*-----ヒストグラム描画処理----------------------------*/
$8 "収録時間:"sec" = LEN(#($1))*PRD()
assign &1 = "ch"|$1(F0)|"  収録時間:"|$8(F3)|"sec"
call proc X_axis_scale $4,$5,$11,&2 /*外部演算処理ブロック呼び出し*/
def graph_id @1 "Histgram" &1
def graph_x_axis @1 0,0 &2,$11
def graph_y_axis @1 0,0 F1 5
def graph_line @1 40 "#0080FF"    /*線種コードに Bar を設定*/
plot @1 $7
def file_id %1 "graph" grp
save plot %1 @1 $7
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:ヒストグラム>



※ ヒストグラムの X 軸は値、Y 軸は、パーセンタイル(%)として表示しています。

23. 1. 2. 収録チャネルの振幅ヒストグラム(棒グラフ)を描画する

ゼロを上昇で過ってから再びゼロを上昇で過るまでの最大値-最小値を振幅と見なし、ヒストグラムを描画する場合、DTM 関数を使用してゼロを上昇で過る地点 Index 数列を求め、最大値関数及び最小値関数の引数として与え、求めた最大値数列―最小値数列から振幅数列を求め TRC 関数を使用して時間率頻度解析を行います。最大値関数(MAX 関数)、最小値関数(MIN 関数)の引数詳細については後述する 23.1.3.項を参照下さい。なお、時間率頻度解析結果が正負両領域を持ちますが、負領域は計数が存在しない為、ヒストグラム描画時に正領域のみに縮退します。

記述例:

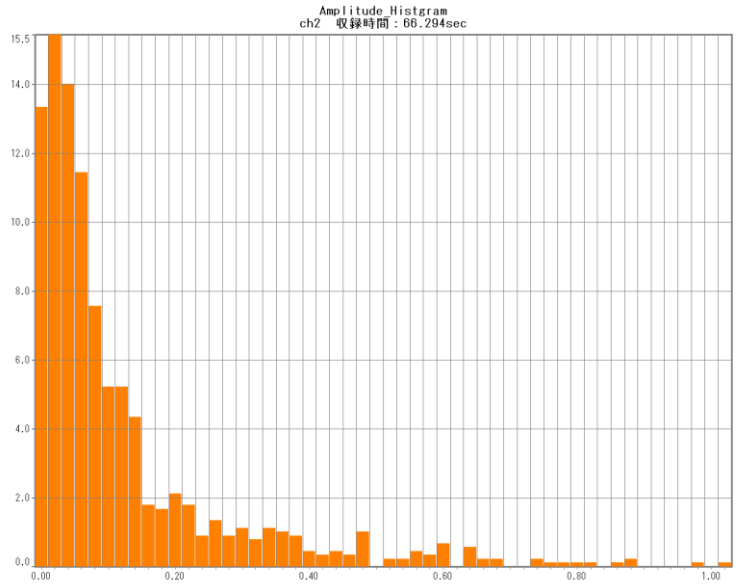
収録データ･カレントチャネルを概略片領域セル個数 64 として振幅ヒストグラムを描画する場合

```
/*---カレントチャネルの振幅ヒストグラム--------------------*/
$1 "カレントチャネル番号:" = CCH()
$2 "ゼロ通過 Index 数列:" = DTM(1,0,1,0,#($1))
$3 "振幅値数列:" = MAX($2,#($1))-MIN($2,#($1))
$4 "振幅最大:" = MAX($3)
def ch_name $5 "セルサイズ"
def ch_name $6 "セル個数"
call proc CellSize $4,64,$5,$6    /*外部演算処理ブロック呼び出し*/
$7 = TRC($5,$6,$3-$5/2)          /*時間率頻度解析*/
$8 = $7/SUM($7)*100               /*Y 軸を%に変換*/
/*-----振幅ヒストグラム描画処理----------------------------*/
$9 "収録時間:"sec" = LEN(#($1))*PRD()
assign &1 = "ch"|$1(F0)|"  収録時間:"|$9(F3)|"sec"
call proc X_axis_scale $5,$6,$11,&2 /*外部演算処理ブロック呼び出し*/
$8 = ERC($6/2,$6-1,$8)                  /*頻度解析結果正領域のみ抽出*/
&2 = CREC(GTE($11,$6/2),&2)             /*目盛表示値正領域のみ抽出*/
$11 = ZSP(GTE($11,$6/2),$11)-$6/2    /*目盛表示 Index 正領域のみ抽出*/
def graph_id @1 "Amplitude_Histgram" &1
```

```
def graph_x_axis @1 0,0 &2,$11
def graph_y_axis @1 0,0 F1 5
def graph_line @1 40 "#0080FF"        /*線種コードに Bar を設定*/
plot @1 $8
def file_id %1 "graph" grp
save plot %1 @1 $8
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:ヒストグラム>



※ ヒストグラムの X 軸は振幅値、Y 軸は、パーセンタイル(%)として表示しています。

23. 1. 3. 平均値/合計値/標準偏差値/最大値/最小値/実効値を求める

平均値を求める場合【MEA 関数】、合計値を求める場合【SUM 関数】、標準偏差値を求める場合【STD 関数】、最大値を求める場合【MAX 関数】、最小値を求める場合【MIN 関数】、実効値を求める場合【EFF 関数】を使用します

文法:

平均値結果格納先 = MEA(演算個数,解析対象数列) 合計値結果格納先 = SUM(演算個数,解析対象数列) 標準偏差値結果格納先 = STD(演算個数,解析対象数列) 最大値結果格納先 = MAX(演算個数,解析対象数列) 最小値結果格納先 = MIN(演算個数,解析対象数列) 実効値結果格納先 = EFF(演算個数,解析対象数列)

引数:

【演算個数】＜省略可＞ 演算範囲のデータ個数を記述します。記述の仕方により演算範囲の設定が異なります。 記述省略した場合:

解析対象数列の全ての範囲が演算対象となります。 即

値或いは要素数1個の参照チャネル番号で記述した場合:

演算範囲個数と見なし、先頭から記述した個数分に区切って演算範囲とします。例えば 100 と記述した場合は、index0～99,100～199,200～299,,,,として演算を行い、解析対象数列の終端部が、記述した個数に満たない余り部分は演算範囲から除外されます。

複数要素を持つ参照チャネルで記述し、その要素数が解析対象数列より少ない場合:

解析象数列の Index と見なし、隣接した index 値間を演算範囲と見なします。なお、複数要素の index 数列で記述した場合、同値を含まず昇順並びの必要があります。例えば、100,250,300 と 3 つの要素を持つ参照チャネルで記述した場合は index100～249,250～299 の 2 箇所の演算を行います。

複数要素を持つ参照チャネルで記述し、要素数が解析対象数列と同じの場合: 論理数列と見なし、論理"1"の範囲を演算範囲とします。論理数列の演算したい部分を論理"1"とすることで、演 算したい範囲を不連続で指定出来ます。

【解析対象数列】＜必須＞

解析対象数列を記述します。

※ 関数内部演算式を示します。

MEA 関数内部演算式 $Ans = \frac{1}{n}\sum_{i=1}^{n} X_i$

SUM 関数内部演算式 $Ans = \sum_{i=1}^{n} X_i$

STD 関数内部演算式 $Ans = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

EFF 関数演算式 $Ans = \sqrt{\frac{1}{n}\int_{i=1}^{n} x_i^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$

※ 分散を求める場合は、STD 関数の戻り値を二乗するか、又は、演算式で次の様に記述します。分散値 = SUM(対象数列-MEA(対象数列))/LEN(対象数列)

※ 自由度 n-1 の不偏分散及び実験標準偏差値を求める場合は演算式で次の様に記述します。
不偏分散値 = SUM(対象数列-MEA(対象数列))/(LEN(対象数列)-1) 実験標準偏差値 = SQR(SUM(対象数列-MEA(対象数列))/(LEN(対象数列)-1)) 実験標準偏差値 = SQR(不偏分散値)

記述例:演算範囲を全範囲とする場合

収録データのチャネルごとの基本統計値を求め結果シートに書き込む場合

```
/*---- 結果シート使用宣言-----------------------------------------*/
dcl sheet 1
  { page 1:
     column $2,&1,&2,$[3,6]
     format F0,2(A),6(F4) 2
}sheet
/*-------- チャネルごと基本統計値演算---------------------------------*/
$1 = NCH()                    /*収録データチャネル数取得*/
$2 "Ch_No.:" = CHS()          /*収録チャネル番号取得*/
&1 "信号名:" = CHNM(0)        /*信号名取得*/
&2 "単位:" = CUNT(0)          /*単位取得*/
/* 演算は収録チャネルごとに行い、index を指定して格納しますので格納先を記述します*/
$3 "平均値:" = DAG($1,0)      /*チャネル数分の 0 数列準備*/
$4 "合計値:" = $3             /*チャネル数分の 0 数列準備*/
$5 "標準偏差値:" = $3         /*チャネル数分の 0 数列準備*/
$6 "最大値:" = $3             /*チャネル数分の 0 数列準備*/
$7 "最小値:" = $3             /*チャネル数分の 0 数列準備*/
$8 "実効値:" = $3             /*チャネル数分の 0 数列準備*/
$9 = 0                        /*チャネルカウンタ初期化*/
repeat_case $9 < $1           /*チャネル LOOP*/
  proc calc{      /*チャネルごと基本統計値演算処理ブロック*/
    $3($9) = MEA(#($2($9)))   /*平均値演算*/
    $4($9) = SUM(#($2($9)))   /*合計値演算*/
    $5($9) = STD(#($2($9)))   /*標準偏差値演算*/
    $6($9) = MAX(#($2($9)))   /*最大値演算*/
    $7($9) = MIN(#($2($9)))   /*最小値演算*/
    $8($9) = EFF(#($2($9)))   /*実効値演算*/
    $9 = $9+1                 /*チャネルカウンタ UP*/
  }calc
/*------ 結果シート書き込み処理-------------------------------------*/
write ch_column 1: &1,&2,$[2,7]
end
```

<実行結果:結果シート書き込み結果>

Page1:

Ch_No.	信号名	単位	平均値	合計値	標準偏差値	最大値	最小値	実効値
1	信号1	deg/s	-0.5191	-928.1999	11.3506	21.7773	-19.8989	11.3624
2	信号2	deg/s	-0.9850	-1761.2420	20.5919	37.6586	-36.3280	20.6155
3	信号3	deg	0.0980	175.2153	2.1205	4.0858	-3.4286	2.1227
4	信号4	N・m	0.4212	753.0339	2.9111	6.3958	-5.7020	2.9414
5	信号5	N・m	-0.9847	-1760.6871	3.0188	5.2217	-6.5884	3.1754

記述例:演算範囲を設定した閾値以上として演算する場合

収録データ・カレントチャネルの設定した閾値 5 以上の基本統計量を求め結果シートに書き込む場合

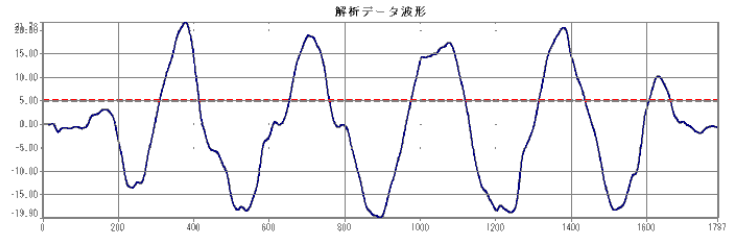
① 演算対象チャネルを GT_関数で閾値と比較し論理数列に変換します。この結果、論理数列の論理"1"はデータが閾値以上のデータである事を意味します。

② 生成した論理数列を ZSP 関数の引数に記述しカレントチャネルデータから論理"0"の index データを削除した演算対象数列を生成します。

※ ZSP 関数は、論理値チャネル$3 の論理値"1"の要素のみに数列を再構成する関数です。

```
/*---- 結果シート使用宣言-----------------------------------------------*/
dcl sheet 1
  { page 1:
     column $1,$2,$11,$[5,6]
     format F0,F3,F0,6(F4) 2
}sheet
/*-------- 演算範囲データの生成-----------------------------------------
$1 "Ch_No.:" = CCH()                          /*カレントチャネル番号取得*/
$2 "閾値:" = 5                                /*設定閾値*/
$3 "閾値以上論理数列:" = GT_(#($1),$2)        /*閾値で論理化*/
$4 "閾値以上対象数列再構成:" = ZSP($3,#($1)) /*論理"1"のみに縮退*/
/*------ 演算範囲生成済み基本統計量演算処理-----------------------------*/
$5 "平均値:" = MEA($4)                        /*平均値演算*/
$6 "合計値:" = SUM($4)                        /*合計値演算*/
$7 "標準偏差:" = STD($4)                      /*標準偏差値演算*/
$8 "最大値:" = MAX($4)                        /*最大値演算*/
$9 "最小値:" = MIN($4)                        /*最小値演算*/
$10 "実効値:" = EFF($4)                       /*実効値*/
$11 "データ個数:" = SUM($3)                   /*閾値以上データ個数演算*/
write ch_column 1: $1,$2,$11,$[5,6]           /*結果シート書き込み*/
/*----- 解析データグラフ描画処理-----------------------------------------*/
$12 = LNK($2,$2)
$13 = LNK(0,LEN(#($1))-1)
$14 = ACC(DAG(LEN(#($1))-1,1))-1
$15 = LNK(2,23)
def graph_id @1 "解析データ波形"
def graph_x_axis @1 0,0 F0
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
def graph_line @1 $15
plot @1 $14,$13 #($1),$12
def file_id %1 "wave" grp
save plot %1 @1 $14,$13 #($1),$12
end
```

<解析対象波形>



※ グラフ赤点線表示は設定した閾値を示します。

<実行結果:結果シート書き込み結果>

Page1:

Ch_No.	閾値	データ個数	平均値	合計値	標準偏差	最大値	最小値	実効値
1	5.000	534	13.4542	7184.5236	4.5540	21.7773	5.0094	14.2040

記述例:演算区間を固定のデータ数で区切って演算する場合 収録データ・カレントチャネルの先頭から 100 点ごとに区間を区切って基本統計量を求める場合 区間ごとに求める場合、設定した区間値(演算対象データ数)をそのまま演算関数の引数に記述します。

```
/*---- 結果シート使用宣言-----------------------------------------------*/
dcl sheet 1
  { page 1:
     column $[1,11]
     format 5(F0),6(F4) 2
}sheet
/*-------- 区間ごとの基本統計演算準備-----------------------------------*/
$1 "Ch_No.:" = CCH()                          /*カレントチャネル番号取得*/
$2 "区間データ数:" = 100                      /*設定区間データ数*/
$3 "区間番号:" = ACC(DAG(INT(LEN(#($1))/$2),1)) /*区間番号生成*/
$4 "区間開始 Index:" = ($3-1)*$2              /*区間開始 index*/
$5 "区間終了 Index:" = $4+$2-1                /*区間終了 Index*/
/*-------- 演算対象区間-------------------------------------------------*/
$6 "平均値:" = MEA($2,#($1))                  /*平均値演算*/
$7 "合計値:" = SUM($2,#($1))                  /*合計値演算*/
```

```
$8 "標準偏差:" = STD($2,#($1))                /*標準偏差値演算*/
$9 "最大値:" = MAX($2,#($1))                  /*最大値演算*/
$10 "最小値:" = MIN($2,#($1))                 /*最小値演算*/
$11 "実効値:" = EFF($2,#($1))                 /*実効値*/
write ch_column 1: $[1,11]                    /*結果シート書き込み*/
end
```

<実行結果:結果シート書き込み結果>

Page1:

Ch_No.	区間データ数	区間番号	区間開始Index	区間終了Index	平均値	合計値	標準偏差	最大値	最小値	実効値
1	100	1	0	99	-0.5968	-59.6791	0.4875	0.1646	-1.6174	0.7706
		2	100	199	1.2920	129.2026	1.6239	3.1121	-3.1224	2.0752
		3	200	299	-9.1145	-911.4534	4.2717	1.1436	-13.4950	10.0659
		4	300	399	14.5631	1456.3081	6.0011	21.7773	1.5573	15.7511
		5	400	499	-3.7548	-375.4772	6.8431	13.9198	-15.5619	7.8055
		6	500	599	-13.5552	-1355.5165	5.6142	-2.7214	-18.3950	14.6718
		7	600	699	6.4755	647.5459	6.8978	18.6109	-2.5506	9.4610
		8	700	799	9.1499	914.9924	7.8085	18.9047	-0.5546	12.0289
		9	800	899	-12.9045	-1290.4531	6.7541	-0.3639	-19.8989	14.5652
		10	900	999	-3.8898	-388.9772	9.8246	13.5349	-19.4295	10.5666
		11	1000	1099	15.3516	1535.1645	1.2018	17.3861	12.1925	15.3986
		12	1100	1199	-6.3635	-636.3516	9.5330	11.8173	-18.3200	11.4618
		13	1200	1299	-12.8731	-1287.3123	6.5419	-0.6474	-18.7847	14.4400
		14	1300	1399	13.9489	1394.8890	6.2545	20.5007	-0.3703	15.2869
		15	1400	1499	0.4543	45.4316	8.7749	14.2521	-15.8197	8.7867
		16	1500	1599	-12.1736	-1217.3574	6.3569	3.3406	-18.2966	13.7334
		17	1600	1699	5.4747	547.4737	3.5262	10.1642	0.3952	6.5120

※ 区間終了 Index が 1699 と言う事は、カレントチャネルのデータ数が 1800 個に満たないことを意味しています。

記述例:演算区間が連続で存在する場合

収録データ・カレントチャネルの値がゼロを過るごとの区間基本統計値を求める場合

```
/*---- 結果シート使用宣言-------------------------------------------*/
dcl sheet 1
  { page 1:
      column $1,$[6,8]
      format 3(F0),6(F4) 2
}sheet
/*-------- ゼロクロスごと区間 index の生成処理-------------------------------------*/
$1 "Ch_No.:" = CCH()                          /*カレントチャネル番号取得*/
$2 "ゼロ上昇通過 index:" = DTM(1,0,1,0,#($1))    /*ゼロ上昇通過 index*/
$3 "ゼロ下降通過 index:" = DTM(0,0,1,0,#($1))     /* ゼロ下降通過 index*/
$4 "ゼロ通過 index:" = SRT(1,LNK($2,$3))          /*ゼロ通過 index 昇順並び替え*/
$6 "区間開始 Index:" = ERC(0,LEN($4)-2,$4)        /*区間開始 index*/
$7 "区間終了 Index:" = ERC(1,LEN($4)-1,$4)-1      /*区間終了 Index*/
/*-------- ゼロクロス区間ごとの基本統計値演算-----------------------------*/
$8 "平均値:" = MEA($4,#($1))                  /*平均値演算*/
$9 "合計値:" = SUM($4,#($1))                  /*合計値演算*/
$10 "標準偏差:" = STD($4,#($1))               /*標準偏差値演算*/
$11 "最大値:" = MAX($4,#($1))                 /*最大値演算*/
$12 "最小値:" = MIN($4,#($1))                 /*最小値演算*/
$13 "実効値:" = EFF($4,#($1))                 /*実効値*/
write ch_column 1: $1,$[6,8]                  /*結果シート書き込み*/
end
```

※ 記述例では、特にエラートラップ処理を記述していない為、対象チャネルが必ずゼロを過ることを前提としています。

<実行結果:結果シート書き込み結果>

Page1:

Ch_No.	区間開始Index	区間終了Index	平均値	合計値	標準偏差	最大値	最小値	実効値
1	1	14	0.0373	0.5216	0.0207	0.0669	0.0056	0.0426
	15	20	-0.0311	-0.1863	0.0125	-0.0140	-0.0463	0.0335
	21	29	0.0958	0.8618	0.0572	0.1646	0.0017	0.1116
	30	119	-0.8259	-74.3337	0.2980	-0.0561	-1.6174	0.8780
	120	191	2.1763	156.6935	0.8174	3.1121	0.0535	2.3247
	192	296	-8.8350	-927.6717	4.3268	-0.1002	-13.4950	9.8376
	297	422	12.7983	1612.5855	6.6913	21.7773	0.0168	14.4419
	423	609	-10.1550	-1898.9910	6.0709	-0.1359	-18.3950	11.8313
	610	622	0.3035	3.9449	0.1607	0.5049	0.0362	0.3434
	623	630	-0.0468	-0.3740	0.0253	-0.0074	-0.0745	0.0532
	631	773	11.0551	1580.8829	6.4305	18.9047	0.0391	12.7893
	774	961	-10.3796	-1951.3654	7.1859	-0.0030	-19.8989	12.6243
	962	1129	11.8565	1991.8941	4.9422	17.3861	0.2507	12.8453
	1130	1301	-12.3076	-2116.9139	6.0745	-0.0779	-18.7847	13.7251
	1302	1454	11.7261	1794.0890	6.3520	20.5007	0.0425	13.3360
	1455	1590	-11.6716	-1587.3383	5.5381	-0.2028	-18.2966	12.9189
	1591	1703	4.9988	564.8644	3.5925	10.1642	0.0164	6.1558

記述例:演算区間が連続で存在する場合 収録データ･カレントチャネルがゼロを上昇で過り再び上昇で過るまでの区間で絶対値最大が 2 以上の区間の基本統 計値を求める場合。区間の絶対最大値閾値を越えない場合は、演算区間と見なさず、次の演算区間に含み処理し ています。

区間を求める演算手順:

① 演算対象チャネルデータがゼロを上昇で過る index を DTM 関数で取得します。
② 取得した Index 列を MAX 関数、MIN 関数の引数に記述し絶対値最大数列を求めます。
③ 求めた区間ごとの絶対値最大を設定した閾値を越えているか GTE 関数で判定します。
④ 判定結果の論理数列の論理"0"位置 Index のゼロ上昇通過 Index 数列から削除します。 但し、最後にゼロ上昇した Index は除外します。
⑤ 削除結果のゼロ上昇通過 Index 数列の最後にゼロ上昇通過した Index を連結します。
⑥ ⑤で生成したゼロ通過 Index 数列を其々の演算関数の引数に記述します。

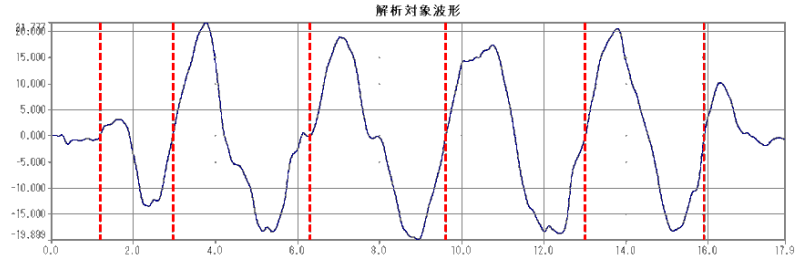
```
/*---- 結果シート使用宣言-----------------------------------------*/
dcl sheet 1
  { page 1:
     column $1,$15,$16,$[8,7]
     format 3(F0),7(F4) 2
}sheet
/*----- 演算ゼロクロス有効区間抽出-----------------------------------------
$1 "Ch_No.:" = CCH()                              /*カレントチャネル番号取得*/
$2 "ゼロ上昇通過 index:" = DTM(1,0,1,0,#($1))     /*ゼロ上昇通過 index*/
$3 "閾値:" = 2
$4 = RVS(3,MAX($2,#($1)),SGN(MIN($2,#($1))))     /*区間内絶対値最大演算*/
$5 = GTE($4,$3)                                    /*区間有効無効判定*/
$6 = LNK(ZSP($5,ERC(0,LEN($2)-2,$2)),$2(LEN($2)-1)) /*有効区間再構成*/
/*-------- ゼロクロス有効区間ごとの基本統計値演算-----------------------------*/
$8 "平均値:" = MEA($6,#($1))                      /*平均値演算*/
$9 "合計値:" = SUM($6,#($1))                      /*合計値演算*/
$10 "標準偏差:" = STD($6,#($1))                   /*標準偏差値演算*/
$11 "最大値:" = MAX($6,#($1))                     /*最大値演算*/
$12 "最小値:" = MIN($6,#($1))                     /*最小値演算*/
$13 "実効値:" = EFF($6,#($1))                     /*実効値演算*/
$14 "振幅値:" = $11-$12                           /*振幅値演算*/
$15 "開始 index:" = ERC(0,LEN($6)-2,$6)           /*有効区間開始 index 演算*/
$16 "データ個数:" = ERC(1,LEN($6)-1,$6)-$15       /*区間内データ個数演算*/
write ch_column 1: $1,$[8,9]                      /*結果シート書き込み*/
/*---- 解析対象波形グラフ描画格納処理-----------------------------------*/
$16 = SPB(0)                                      /*グラフ X 軸生成*/
$17 = PTV(MGR($6,$6),$16)                         /*追加線アドレス生成*/
$18 = MGR(DAG(LEN($6),MAX(#($1))),DAG(LEN($6),MIN(#($1)))) /*追加線 Y 軸アドレス生成*/
$19 = MGR(DAG(LEN($6),1),DAG(LEN($6),0))          /*PEN 制御情報生成*/
def graph_id @1 "解析対象波形"
def graph_x_axis @1 0,0 F1
def graph_y_axis @1 0,0 F3 5
def graph_aspect_ratio @1 3
def graph_line_draw @1 $17,$18,$19 23 "#0000FF"
plot @1 $16 #($1)
def file_id %1 "wave" grp
save plot %1 @1 $16 #($1)
end
```

<実行結果:結果シート書き込み結果>

Page1:

Ch_No.	有効区間開始 index	データ個数	平均値	合計値	標準偏差	最大値	最小値	実効値	振幅値
1	120	177	-4.3558	-770.9782	6.3746	3.1121	-13.4950	7.7207	16.6071
	297	334	-0.8468	-282.8346	12.5037	21.7773	-18.3950	12.5323	40.1723
	631	331	-1.1193	-370.4825	12.6464	18.9047	-19.8989	12.6959	38.8035
	962	340	-0.3677	-125.0198	13.2926	17.3861	-18.7847	13.2976	36.1708
	1302	289	0.7154	206.7506	13.1218	20.5007	-18.2966	13.1413	38.7973

<解析対象波形>



※ 赤点線は有効区間示します。有効区間は赤点線と赤点線の間で 5 箇所存在します。

記述例:演算区間が不連続で存在する場合 収録データ・カレントチャネルがゼロを上昇で過ってから再び上昇で過るまでの区間で振幅最大値が 5 以上の区間の 基本統計値を求める場合

※ 演算区間を"1"とし無効区間を"0"とした論理数列を生成して、演算関数の引数に与え論理"1"部分の統計量を求めます。

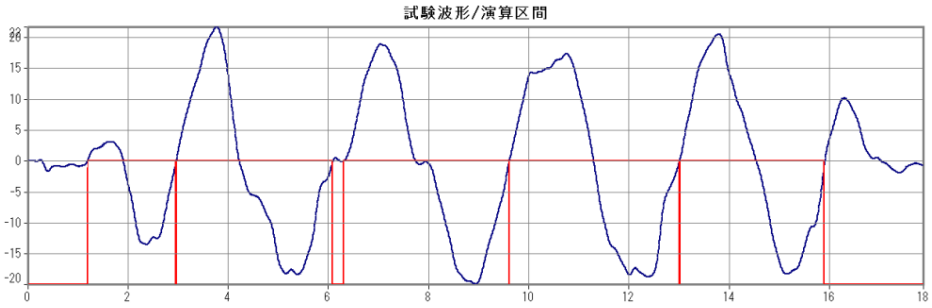
```
/*-----結果シート使用宣言----------------------------*/
dcl sheet 1 {
      page 1:
      column &3,&1,&2,$[8,9]
      format 3(A),2(F0),7(F4) 2
      }sheet
$1 "カレント ch 番号:" = CCH()      /* カレントch 番号取得*/
&1 "信号名:" = CHNM($1)             /* 信号名取得*/
&2 "単位:" = CUNT($1)                /* 単位取得*/
assign &3 "ch_No.:" = $1(F0)|"ch"
/*------ 演算区間検索---------------------------------*/
$2 "ゼロ上昇通過 index:" = DTM(1,0,1,0,#($1))
$5 = GTE(MAX($2,#($1))-MIN($2,#($1)),5)
$3 "開始 index:" = ZSP($5,ERC(0,LEN($2)-2,$2))
$4 "終了 index:" = ZSP($5,ERC(1,LEN($2)-1,$2)-1)
$6 "ソートキー:" = SRT(2,LNK($3,$4))
$7 "演算区間:" = ACC(VRP(0,LEN(#($1)),QUE($6,LNK($3,$4)),QUE($6,LNK(DAG(LEN($3),1),DAG(LEN($4),-1)))))
/*--------統計量演算-----------------------------------*/
$8 "開始 index:" = $3
$9 "終了 index:" = $4
$10 "時間:sec" = ($4-$3)*PRD()
$11 "平均値:" = MEA($7,#($1))
$12 "最大値:" = MAX($7,#($1))
$13 "最小値:" = MIN($7,#($1))
$14 "振幅値:" = $12-$13
$15 "実効値:" = EFF($7,#($1))
$16 "標準偏差値:" = STD($7,#($1))
write ch_column 1: &[1,3],$[8,9]   /* 結果シート書き込み*/
/*---- グラフ描画処理-------------------------------*/
$18 "時刻歴:" = SPB(0)
$7 = ($7-0.5)*10
def graph_id @1 "試験波形/演算区間"
def graph_x_axis @1 0,0 F0
def graph_y_axis @1 0,0 F0 5
def graph_aspect_ratio @1 3
plot @1 $18 #($1),$7
end
```

<実行結果:結果シート書き込み結果>

Page1:

ch_No.	信号名	単位	開始index	終了index	時間(sec)	平均値	最大値	最小値	振幅値	実効値	標準偏差値
3ch	信号1	deg/s	120	296	1.7600	-4.3800	3.1121	-13.4950	16.6071	7.7426	6.3846
			297	609	3.1200	-0.9175	21.7773	-18.3950	40.1723	12.9664	12.9339
			631	961	3.3000	-1.1224	18.9047	-19.8989	38.8035	12.7151	12.6654
			962	1301	3.3900	-0.3686	17.3861	-18.7847	36.1708	13.3172	13.3121
			1302	1590	2.8800	0.7199	20.5007	-18.2966	38.7973	13.1641	13.1444

<実行結果:解析対象波形と演算区間>



23. 1. 4. 中央値/最頻値/尖度値を求める

中央値(median)とは:

解析対象数列の値を昇順並びとした時の Index 中心の値を意味します。解析対象数列が偶数の場合は真中の値 2 個の平均値を取ります。中央値を求める場合【MED 関数】を使用します。

文法:

中央値格納先 = MED(演算個数,解析対象数列) 引数:

平均値関数(MEA 関数)と同じ指定です。

記述例:MED 関数を使用して中央値を求める場合

解析対象数列$1 の中央値を$2 に求める場合

```
$2 = MED($1)          /* 中央値演算*/
```

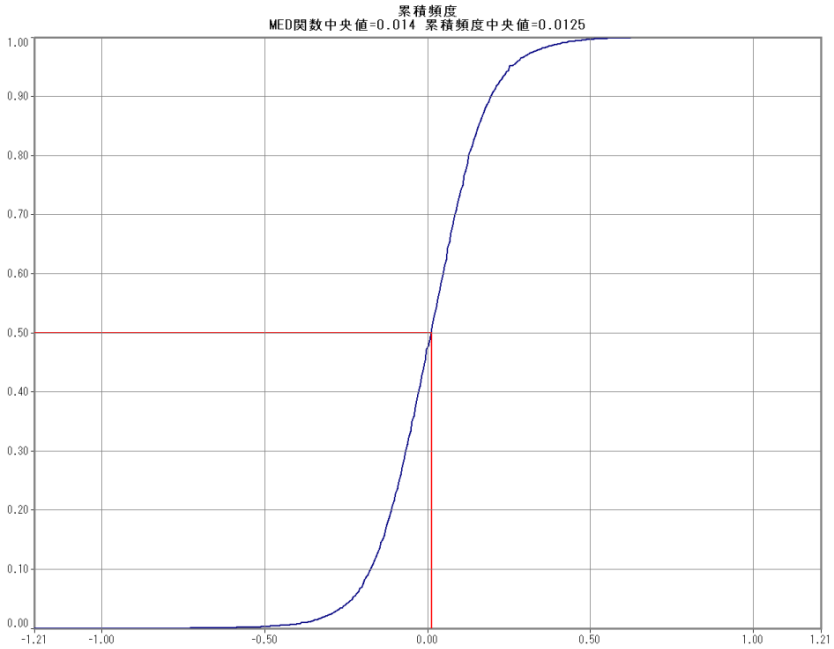
なお、MED 関数は、関数内部で並び替え操作が行われます。その為、データ数の大きな数列の場合は演算時間が掛ります。大きな数列を解析対象とする場合は、時間率頻度解析を行いその結果から中央値を求める場合があります。但し、精度は設定されるセルサイズ以下にはなりません。

記述例:時間率頻度解析の結果から概略中央値を求める場合

収録データ・カレントチャネルをセルサイズ 0.005 とした時間率頻度解析結果から概略中央値を求める場合

```
/*-------------------------------------------------------------------*/
$1 "カレント ch 番号:" = CCH()
$2 "median:" = MED(#($1))
/*------ 頻度解析から中央値演算処理-----------------------------------------*/
$3 "絶対値最大:" = RVS(3,ABS(MIN(#($1))),ABS(MAX(#($1))))
$5 "セルサイズ:" = 0.005
$6 "合計セル数:" = INT($3/$5)*2
$13 "セル中央値:" = CNV(1,$5,$6,1)
$6 "時間率頻度解析:" = TRC($5,$6,#($1))
$7 "累積頻度:" = ACC($6)/SUM($6)
$8 "50%通過 index:" =  DTD(1,0.5,1,0,$7)     /*累積頻度数 50%通過index 検索*/
$14 "中央値:" = PTV($8,$13)                   /*累積頻度 50%通過セル代表値*/
/*----- グラフ描画処理-----------------------------------------------*/
$9 = LNK(MIN($13),$14,$14)
$10 = LNK(0.5,0.5,0)
assign &1 = "MED 関数中央値="|$2(F3)|" 累積頻度中央値="|$14(F4)
def graph_id @1 "累積頻度" &1
def graph_x_axis @1 0,0 F2
def graph_y_axis @1 0,0 F2 5
plot @1 $13,$9 $7,$10
def file_id %1 "graph" grp
save plot %1 @1 $13,$9 $7,$10
end
```

<実行結果：累積頻度グラフ>



最頻値(mode)とは：
時間率頻度解析を行った結果、セル（区画)| への計数値が最も大きいセル(区画)| の中央値(代表値)意味します。

尖度値(kuritosis)とは：
分布の尖り具合を表します。尖り具合とは平均値の周辺にデータが集まっているのか(分布の裾野が狭い)などを意味しま す。
尖度を求める場合、専用の関数は用意されていませんので、演算式で求めます。
演算式を示します。

$$Ans = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

記述例：

解析対象数列$1 の尖度を$3 に求める場合

```
$2 = MEA($1)
$3 = SUM(($1-$2)^4)/SUM(($1-$2)^2)
```

記述例：

収録データのカレントチャネルの平均値、最大値､最小値、中央値、最頻値、標準偏差値、尖度値を求める場合

```
/*---- 結果シート使用宣言-------------------------------------*/
dcl sheet 1
  { page 1:
     column $1,$5,$2,$3,$4,$6,$7,$9,$15
     format 2(F0),8(F4),F0,F4 2
}sheet
/*---- 統計量演算処理-----------------------------------------*/
$1 "カレント ch 番号:" = CCH()
$2 "平均値:" = MEA(#($1))
$3 "最大値:" = MAX(#($1))
$4 "最小値:" = MIN(#($1))
$5 "データ個数:" = LEN(#($1))
$6 "中央値:" = MED(#($1))
$7 "標準偏差値:" = STD(#($1))
$9 "尖度値:" = SUM((#($1)-$2)^4)/SUM((#($1)-$2)^2)
/*----- 最頻値演算処理-----------------------------------------*/
$8 "絶対値最大:" = RVS(3,ABS(MIN(#($1))),ABS(MAX(#($1))))
$16 "概略片領域セル数:" = 64
call proc CellSize $8,$16,$11,$10    /*外部演算処理ブロック呼び出し*/
$12 "時間率頻度解析:" = TRC($11,$10,#($1)-$11/2)
$13 "セル中央値:" = CNV(1,$11,$10,1)
$14 "最頻セル index:" = MXP($12)
$15 "最頻値:" = PTV($14,$13)-$11/2
/*---- 結果シート書き込み&グラフ描画処理---------------------*/
```

```
write ch_column 1: $[1,7],$9,$15
assign &1 = "平均="|$2(F3)|" 最小="|$4(F3)|" 最大="|$3(F3)|" 中央="|$6(F3)
assign &1 = &1|" 標準偏差="|$7(F3)|" 尖度="|$9(F3)|"  最頻値="|$15(F3)
call proc X_axis_scale $11,$10,$16,&2 /*外部演算処理ブロック呼び出し*/
def graph_id @1 "histogram" &1
def graph_x_axis @1 0,0 &2,$16
def graph_y_axis @1 0,0 F2 5
def graph_line @1 40 "#0080FF"          /*線種に Bar 設定*/
plot @1 $13 $12
def file_id %1 "graph" grp
save plot %1 @1 $12
end
```

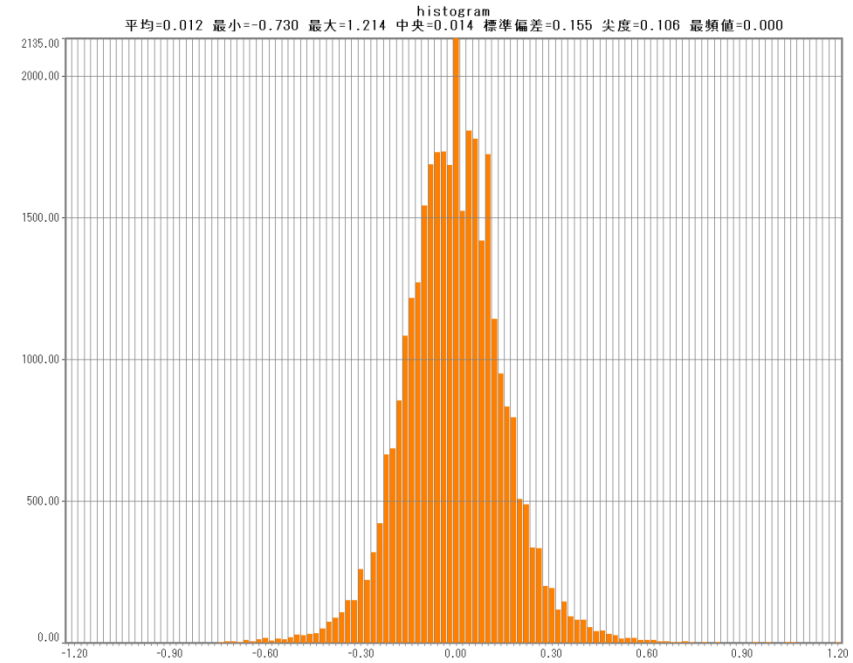
※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:結果シート書き込み内容>

Page1:

カレントch番号	データ個数	平均値	最大値	最小値	中央値	標準偏差値	尖度値	最頻値
1	33147	0.0124	1.2140	-0.7300	0.0140	0.1545	0.1056	0.0000

<実行結果:分布グラフ>



※ X 軸は解析チャネルの値、Y 軸は計数値を示す。

23. 2. 収録チャネル間の相関を求める

23. 2. 1. 信号間の散布図を描画し相関係数/最小二乗法近似直線を求める

相関係数を求める場合【CRR 関数】を使用します。

文法:

相関係数格納先 = CRR(対象数列 X,対象数列 Y)

引数:

【対象数列 X】＜必須＞ 対象
数列を記述します。

【対象数列 Y】＜必須＞
対象数列を記述します。

関数内部演算式を示します。

$$r = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}}$$

記述例:

収録データ･チャネル 1 と 2 の相関係数を求める場合

$1 "相関係数:" = CRR(#1,#2)

$1 に#1 と#2 の相関係数が格納されます。

最小二乗法による直線方程式の傾きとオフセット値を求める場合【LSM 関数】を使用します。 **文法:**

傾きオフセット格納先 = LSM(対象数列 X,対象数列 Y)

引数:

【対象数列 X】＜必須＞ 対象数

列 X を記述します。

【対象数列 Y】＜必須＞ 対象数

列 Y を記述します。

関数内部演算式を示します。

$$a = \frac{n\sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2}$$

$$b = \frac{n\sum_{i=1}^{n} X_i^2 \sum_{i=1}^{n} Y_i^2 - \sum_{i=1}^{n} X_i Y_i \sum_{i=1}^{n} X_i}{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2}$$

記述例:

収録データ･チャネル 1 と 2 の最小二乗法直線方程式を求める場合

$1 “傾きオフセット:” = LSM(#1,#2)

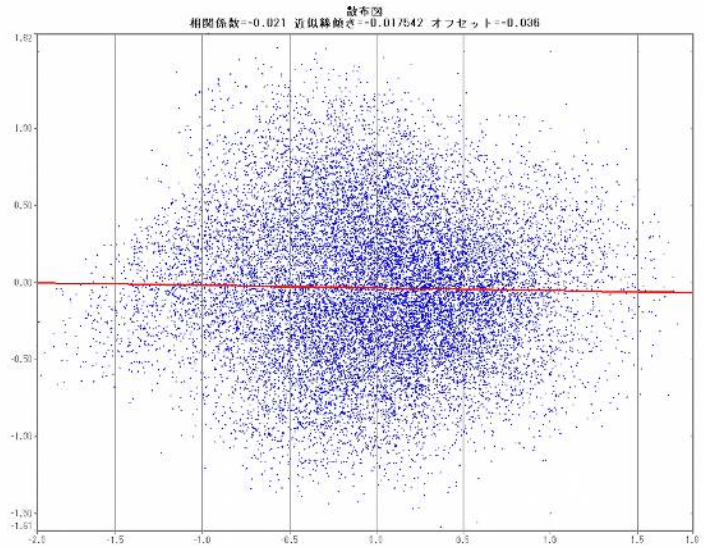
$1(0)に傾きa、$1(1)にオフセット b が格納されます。(y=aX+b)

記述例:

X 軸加速度波形と Y 加速度波形の散布図を描画し、相関係数と近似線を描画する場合

```
/*-----相関係数と最小二乗法近似線を求める----------*/
$5 "相関係数:" = CRR(#1,#2)        /*ch1 と ch2 の相関係数を求める*/
$6 "傾きオフセット:" = LSM(#1,#2) /*ch1 と ch2 の直線近似線を求める*/
/*----グラフ描画処理----------------------------------*/
$7 "近似線 X 軸:" = LNK(MIN(#1),MAX(#1))
$8 "近似線 Y 軸:" = LNK($6(0)*$7(0)+$6(1),$6(0)*$7(1)+$6(1))
$9 "linetype:" = LNK(50,2)
assign &1 = "相関係数="|$5(F3)|" 近似線傾き="|$6(0)(F6)|" オフセット="|$6(1)(F3)
assign &2 = "#FF0000","#0000FF"
def graph_id @1 "散布図" &1
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
def graph_line @1 $9 &2          /*線種にデータを Dot、近似線に太線を設定*/
plot @1 #1,$7 #2,$8
def file_id %1 "graph" grp
save plot %1 @1 #1,$7 #2,$8
end
```

<実行結果:散布図>



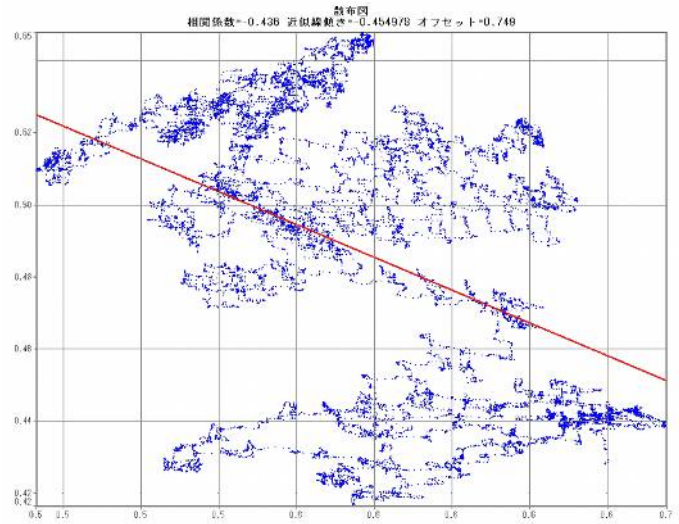
※ X 軸 ch1 加速度、Y 軸 ch2 加速度を示す。相関係数=-0.021 と殆ど無相関である事が判る。

記述例:

X 軸加速度実効値と Y 加速度実効値の散布図を描画し、相関係数と近似線を描画する場合
振動加速度実効値は積分時定数 0.25sec として移動実効値に変換しています。

```
/*-- 加速度実効値を求める----------*/
$3 "X 加速度実効値:m/s^2" = RRV(0.25,#1) /*振動加速度から実効値変換*/
$4 "Y 加速度実効値:m/s^2" = RRV(0.25,#2) /*振動加速度から実効値変換*/
/*-----相関係数と最小二乗法近似線を求める----------*/
$5 "相関係数:" = CRR($3,$4)        /*ch1 と ch2 の相関係数を求める*/
$6 "傾きオフセット:" = LSM($3,$4) /*ch1 と ch2 の直線近似線を求める*/
/*----グラフ描画処理----------*/
$7 "近似線 X 軸:" = LNK(MIN($3),MAX($3))
$8 "近似線 Y 軸:" = LNK($6(0)*$7(0)+$6(1),$6(0)*$7(1)+$6(1))
$9 "linetype:" = LNK(50,2)
assign &1 = "相関係数="|$5(F3)|" 近似線傾き="|$6(0)(F6)|" オフセット="|$6(1)(F3)
assign &2 = "#FF0000","#0000FF"
def graph_id @1 "散布図" &1
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
def graph_line @1 $9 &2          /*線種にデータを Dot、近似線に太線を設定*/
plot @1 $3,$7 $4,$8
def file_id %1 "graph" grp
save plot %1 @1 $3,$7 $4,$8
end
```

<実行結果:散布図>



※ X 軸 ch1 加速度実効値、Y 軸 ch2 加速度実効値を示す。相関係数=-0.438 と逆相関値を示すが散布図から必ず相関しているとは言えない。

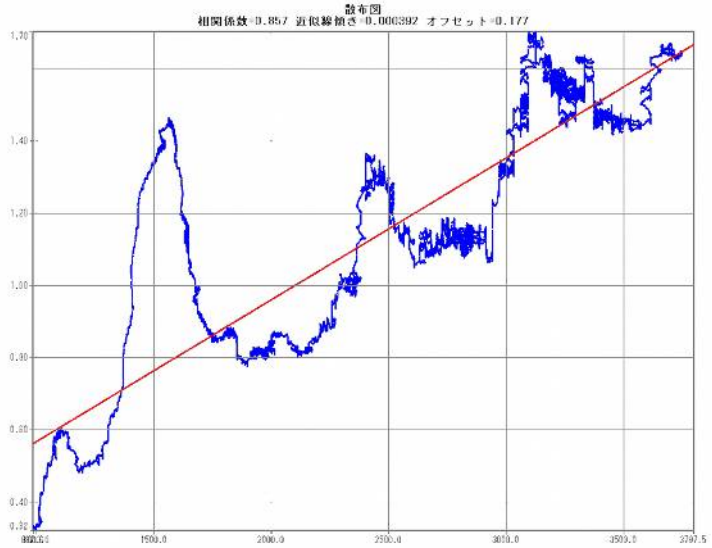
記述例:

エンジン回転数と Z 軸振動加速度実効値の散布図を描画し、相関係数と近似線を描画する場合 回転数は収録データ 9ch の回転パルスから求め、振動加速度実効値は収録データ 3ch の加速度から積分時定数 0.25sec で移 動実効値に変換しています。

```
/*-- 回転数と加速度実効値を求める----------*/
$1 "回転パルス:" = TPC(1,#9)    /*演算に不要なパルスを除去*/
$3 "回転数:rpm" = FVC(1,$1,1) /*タコパルスから回転数に変換*/
$4 "加速度実効値:m/s^2" = RRV(0.25,#3) /*振動加速度から実効値変換*/
/*-----相関係数と最小二乗法近似線を求める----------*/
$5 "相関係数:" = CRR($3,$4)        /*相関係数を求める*/
$6 "傾きオフセット:" = LSM($3,$4) /*直線近似線を求める*/
/*----グラフ描画処理----------*/
$7 "近似線 X 軸:" = LNK(MIN($3),MAX($3))
$8 "近似線 Y 軸:" = LNK($6(0)*$7(0)+$6(1),$6(0)*$7(1)+$6(1))
$9 "linetype:" = LNK(50,2)
assign &1 = "相関係数="|$5(F3)|" 近似線傾き="|$6(0)(F6)|" オフセット="|$6(1)(F3)
assign &2 = "#FF0000","#0000FF"
def graph_id @1 "散布図" &1
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
```

```
def graph_line @1 $9 &2          /*線種にデータを Dot、近似線に太線を設定*/
plot @1 $3,$7 $4,$8
def file_id %1 "graph" grp
save plot %1 @1 $3,$7 $4,$8
end
```

<実行結果:散布図>



※ X 軸回転数、Z 軸振動加速度実効値を示す。相関係数=0.857 と正相関している事が判る。

23. 2. 2. 信号間の相関関数を求める 相関関数を求める場

合【COR 関数】を使用します。・ **文法:**

相関関数格納先 = COR(対象数列 X,対象数列 Y)

引数:

【対象数列 X】＜必須＞ 対象

数列を記述します。

【対象数列 Y】＜必須＞

対象数列を記述します。

対象数列 X と対象数列 Y に同じ数列を記述した場合は自己相関関数を意味し、異なる数列を記述した場合は相互相関関数を意味します。相関関数は波形相互の似具合を確認できます。

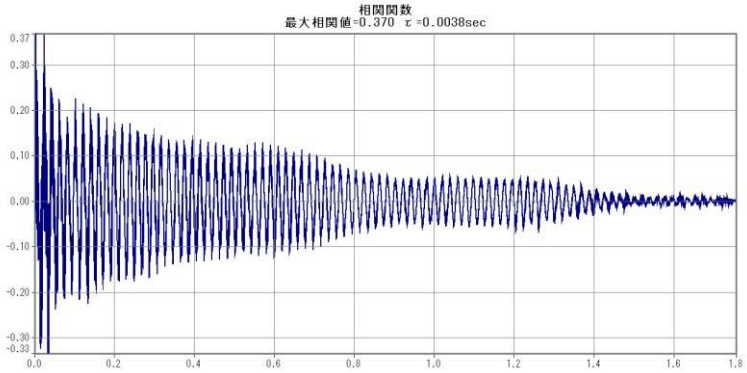
※ 相関関数は大きな数列を処理する場合、演算時間が掛ります。

記述例:

X 軸振動加速度 ch1 と Y 軸振動加速度 ch2 の相互相関関数を求める場合

```
/*-- 相互相関関数を求める----------------------------*/
$1 "相互相関関数:" = COR(#1,#2) /*相互相関関数を求める*/
$2 "τ:sec" = MXP($1)*PRD()      /*相関関数最大値位置時間 τ を求める*/
$3 "相関最大値:" = MAX($1)       /*相互相関関数最大値を求める*/
/*----グラフ描画処理----------------------------------*/
$4 "グラフ X 軸:" = SPB(0)          /*グラフ X 軸数列生成*/
assign &1 = "最大相関値="|$3(F3)|" τ ="|$2(F4)|"sec"
def graph_id @1 "相関関数" &1
def graph_y_axis @1 0,0 F2 5
def graph_x_axis @1 0,0 F1
def graph_aspect_ratio @1 2
plot @1 $4 $1
def file_id %1 "graph" grp
save plot %1 @1 $4 $1
end
```

<実行結果：相互相関関数グラフ>



※ 相互相関関数が規則的に正相関/逆相関になっている事から位相差(τから約 3.8ms)でを持つ似た波形で或る事が判る。

23. 2. 2. 信号間の移動相関係数を求める

移動相関係数を求める場合【MCO 関数】を使用します。移動相関係数とは、記述する同じ長さの参照数列と比較数列の数列の同じ時間軸上を指定したデータ個数分をスライディングしながら相関係数を求める設定(同じ時間軸上で似た波形箇所を検索する場合)と、記述する短い数列で記述する参照波形を長い数列で比較数列上をスライディングしながら相関係数を求める設定(参照波形と似た波形箇所を検索する場合)の何れかを設定出来ます。

文法：

相関係数格納先 = MCO(データ個数,参照数列,比較数列) 引数：

【データ個数】＜必須＞

スライディング幅をデータ個数で記述します。データ個数を 0 とした場合は参照数列は比較数列より短い必要があり、参照数列自体を比較数列上スライディングして相関係数を求めます。データ個数を正数で記述した場合は、記述したデータ個数分を参照数列と比較数列のスライディングしながら相関係数を求めます。 なお、データ個数を正数で記述した場合は参照数列と比較数列は同じ長さでなければなりません。

【参照数列】＜必須＞ 参照数列を記述します。

【比較数列】＜必須＞

比較数列を記述します。

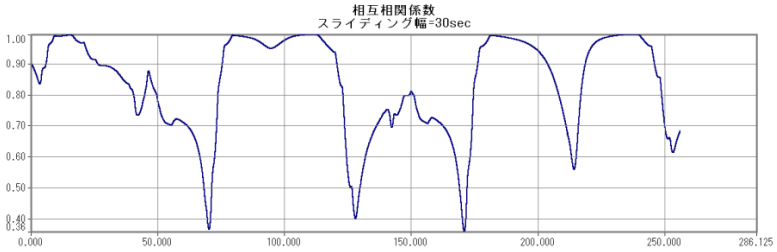
記述例：

収録データ･ch1 回転数波形と ch4 速度波形の似ている箇所を探す場合

スライディング幅 30sec として相互相関係数を求める。

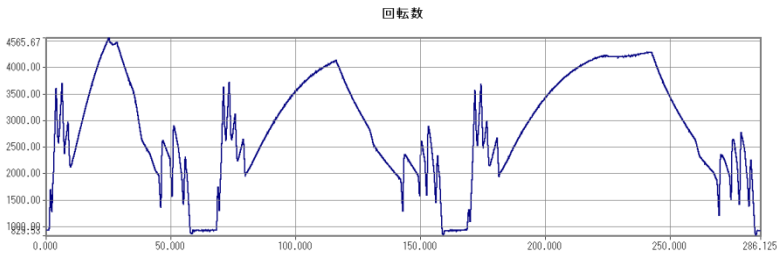
```
/*-------チャネル間スライディング相互相関係数演算--------------------*/
$1 "スライディングデータ個数:" = INT(30/PRD()) /*30 秒間のデータ個数*/
$2 "相互相関係数:" = MCO($1,#1,#4)          /*相互相関係数演算*/
/*-----グラフ描画処理-------------------------------*/
$3 = SPB(0)                                 /*グラフ X 軸数列生成*/
def graph_id @1 "相互相関係数" "スライディング幅=30sec"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $3 $2
def file_id %1 "wave" grp
save plot %1 @1 $3 $2
def graph_id @1 "回転数" " "
plot @1 $3 #1
def file_id %1 "回転数波形" grp
save plot %1 @1 $3 #1
def graph_id @1 "速度" " "
plot @1 $3 #4
def file_id %1 "速度波形" grp
save plot %1 @1 $3 #4
end
```

<実行結果:相互相関係数グラフ>

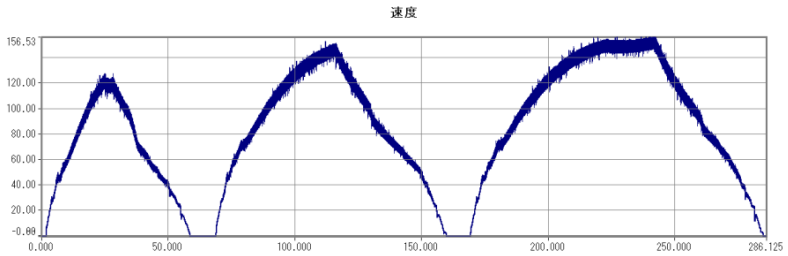


※ 相互相関係数結果数列は設定したスライディング幅分短くなり、相互相関係数のグラフ開始点が前方にスライディング幅分偏 っている様に表示されます。

<解析に使用した回転数波形>



<解析に使用した速度波形>

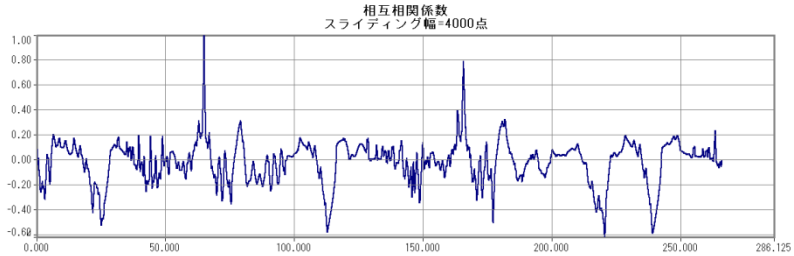


記述例:

収録波形の一部分を切り取り、同じチャネルで切り取った波形と似た箇所が他に出現するかを探す場合

```
/*-------チャネル内スライディング相互相関係数演算----------------*/
$1 "検索波形:" = ERC(13000,16999,#5)     /*検索波形切り取り*/
$2 "相互相関係数:" = MCO(0,$1,#5)         /*相互相関係数演算*/
/*-----------グラフ描画処理-----------------------------------*/
$3 = SPB(0)                                /*グラフ x 軸生成*/
def graph_id @1 "相互相関係数" "スライディング幅=4000 点"
def graph_x_axis @1 0,0 F3
def graph_y_axis @1 0,0 F2 5
def graph_aspect_ratio @1 3
plot @1 $3 $2
end
```

<実行結果:相互相関係数グラフ>



※ 切り取った検索波形と一致した点と、同じ様な波形が相関係数 0.8 で後方に 1 箇所存在する事が判る。

23. 3. 確率紙を作成する

23. 3. 1. ワイブル確率紙を作成する ワイブル(weibull)確率紙は寿命推定や実験データの整理など良く使用されます。ワイブル分布の確率密度関数

$$f(x) = \frac{m}{\eta} \cdot \left(\frac{x}{\eta}\right)^{m-1} \cdot \exp\left\{-\left(\frac{x}{\eta}\right)^{m}\right\}$$

累積密度関数

$$F(x) = 1 - \exp\left\{-\left(\frac{x}{\eta}\right)^{m}\right\}$$

※ m:=形状母数、η :=尺度母数

ワイブル確率紙の書き方:

ワイブル確率紙の X 軸はデータ値 x の $\log_{10}$ 尺とし、Y 軸は故障率 Ft の Ln(Ln(1/(1-Ft))尺となります。
X 軸の左端値はデータの(最小値を含むディケードの開始値または更に 1 ディケードの手前の開始値、右端値はデータの最大値を含むディケードの終端値あるいはその次ディケードの終端値とします。
Y 軸の開始値はデータ数に依存しますが通常 0.1%(0.001):=-6.907255 で問題ありません。
Y 軸の最大値は 100%としますが、尺として演算できないため 2 に固定します。
データ値のプロットは
X 軸値は、データ値を x とすると

$$X_i = \ln(x_i)$$

演算時は ln(x)とし、描画する場合、グラフ属性に log と定義しますので X 値はそのままデータ値 x 記述します。データ値には 0 を含まず、昇順並びである必要があります。
Y 軸値は累積故障率を F とすると

$$Y_i = \ln\left(\ln\left(\frac{1}{1 - F_i}\right)\right)$$

累積故障率 F は、昇順に並び替えたデータ番号(1～n)をからランク法により求めます。
i をデータ番号、***n*** をサンプル数とすると メジアンランク法:
(但し近似式)

$$F_i = \frac{i - 0.3}{n + 0.4}$$

ミーンランク法:

$$F_i = \frac{1}{n + 1}$$

※ ランク法は標本数が 20 個以下の場合メジアンランク法、以上の場合ミーンランク法を使用します。

形状母数 m、尺度母数 η の求め方

形状母数 m は最小二乗近似直線の傾きを意味します。

$$m = \frac{S_{xy} \cdot S_x - S_{xx} \cdot S}{S_x \cdot S_x - S_{xx} \cdot S}$$

※ 形状母数は分布の形に関わりを持ち、m が大きくなると傾斜が大きくなり、言い換えれば分布の幅が狭くなり、m が小さくなると傾斜が緩やかになり、言い換えれば分布の幅が広くなり、解析の信頼性も低くなります。
尺度母数 η は累積故障率 Ft の 63.21%地点の x に相当します。

$$\eta = \exp\left(-\left(\frac{S_x \cdot S_{xy} - S_y \cdot S_{xx}}{S_x \cdot S_x - S_{xx} \cdot S}\right)/m\right)$$

Sx、Sxx、Sy、Sxy は次の通り

$$S_x = \sum_{i=1} X_i \quad S_{xx} = \sum_{i=1} X_i^2 \quad S_y = \sum_{i=1} Y_i \quad S_{xy} = \sum_{i=1} (X_i \cdot Y_i)$$

※ 上記式中の Xi は *Ln(xi)*を意味し、Yi は *Ln(Ln(1/(1-Fi))*を意味します。

平均寿命 MTTF の求め方

ガンマ関数「 を使用して

$$MTTF = \eta \cdot \Gamma\left(1 + \frac{1}{m}\right)$$

最小二乗法近似直線の書き方

X と Y を最小二乗法により直線方程式 y=ax+b の a と b を求めます。
傾き aは形状母数演算と同じです。
オフセット b は

$$\hat{\beta} = \frac{n\sum x_i^2 \ln x_i - \sum x_i^2 \sum \ln x_i}{n\sum x_i^2 \ln x_i - \sum x_i \sum x_i \sum \ln x_i}$$

記述例:

データ値から故障率 Ft を求めるランク付け処理を行う外部演算処理ブロックを作成する データ数列の同値除去処理、昇順並び替え処理を行った後、データ数によりメジアンランク又はミーンランクにより Ft を求めます。 proc lank_calc{

```
  /*-------------------------------------------------------
  呼び出し方法:
    call proc lank_calc データ数列,Ft,同値除去フラグ 引
  数:
    $100 : <in> データ数列   $101 :<out> Ft
    $104 <in> 同値除去フラグ  0:なし,1:除去
  -------------------------------------------------------*/
  def inherit_ch $100,$101,$104
  def local_ch $102,$103
  case_false $104       /*同値除去なしの場合*/
    proc 同値除去無し{
      $100 = SRT(1,PTV(NEQ($100,0,1),$100))
    }同値除去無し
  case_true $104       /*同値除去ありの場合*/
    proc 同値除去する{
      $100 = UNQ(SRT(1,PTV(NEQ($100,0,1),$100)))
    }同値除去する
  $102 = LEN($100)                          /* データ数取得*/
  $103 = ACC(DAG($102,1))                   /* データ番号生成*/
  $101 = RVS(GTE($102,20),($103-0.3)/($102+0.4),$103/($102+1)) /*N<20:Median N>20:Mean*/
  $101 = LOG(LOG(1/(1-$9)))                 /* y⇒Ft 尺変換*/
}lank_calc
```

記述例:

データ値と故障率 Ft から m 値、η 値、MTTF 値及び近似線 X 軸値と Y 軸値を求める外部演算処理ブロックを作成 する

```
proc param_calc{
  /*---------m 値,η 値,MTTF 近似線演算-----------------------------------------------
  呼び出し方法:
    call proc param_calc データ数列,Ft 数列,m 値,η 値,MTTF 値,近似線 X 軸数列,Y 軸数列 引
  数:
    $100 <in> data   $101 <in> Ft
    $102 <out> 形状母数:m 値  $103 <out> 尺度母数:η 値  $104 <out> MTTF
    $105 <out> 近似線 X 軸  $106 <out> 近似線 Y 軸
  ------------------------------------------------------------------------------------*/
  def inherit_ch $100,$101,$102,$103,$104,$105,$106
  def local_ch $107,$108,$109,$110,$111
  $107 = SUM(LOG($100))                     /* Sx 演算*/
  $108 = SUM($101)                          /* Sy 演算*/
  $109 = MAC(LOG($100),LOG($100))           /* Sxx 演算*/
  $110 = MAC(LOG($100),$101)                /* Sxy 演算*/
  $102 = LSM(LOG($100),$101)                /* m 演算*/
  $103 = EXP(0-(($107*$110-$108*$109)/($107*$107-$109*LEN($100)))/$102(0)) /* η 値演算*/
  $104 = $17*GMM(1+1/$102(0))                 /* MTTF:平均寿命*/
  assign $105 = 5,10,100,1000,2000            /* 近似線 X 軸値生成*/
  $106 = $102(0)*LOG($105)+$102(1)            /* 近似線 Y 軸値演算*/
}param_calc
```

記述例:

ワイブル確率紙の Y 軸値(故障率)から相当値を求める外部演算処理ブロックを作成する

```
proc weibull_value{
  /*-------------------------------------------------------------------
  呼び出し方法:
     call weibull_value 故障率数列,対応する相当値数列,形状母値 m 引
  数:
    $100 <in> Ft%   $101 <out> Weibull Value   $104 <in> m 値
  -------------------------------------------------------------------*/
  def inherit_ch $100,$101,$104
  def local_ch $102,$103
  $102 = LEN($100)          /*演算個数取得*/
```

```
  $103 = 0                    /*loop カウンタ初期化*/
  repeat_case $103 < $102     /*演算個数 loop*/
    proc calc{
      $101 = EXP((LOG((LOG(1/(1-$100/100))))-$104(1))/$104(0))
      $103 = $103+1           /*loop カウンタ up*/
    }calc
}weibull_value
```

記述例:

半角カンマ区切りの拡張子"csv"形式ファイルの列ごとにデータ値が記述されていると見なし、指定されたファイルを呼び出しワイブル確率紙にプロットする列を選択して描画する場合

csv ファイルの各列の 1 行目は列名が記述されている必要があります。結果はワイブル確率紙グラフと求めた各パラメタの csv 形式ファイルを生成します。

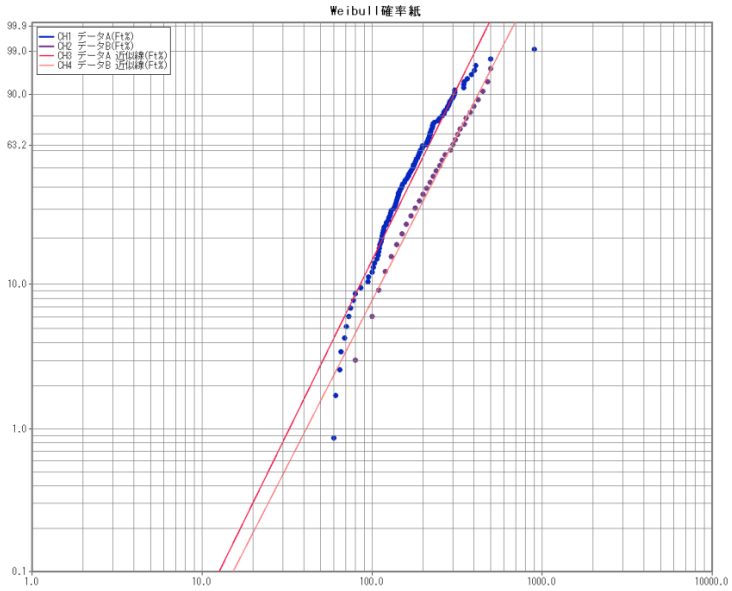
```
/*---------ワイブル確率紙----------*/
def ch_name $1 "Y 軸(Ft%)グリッド線位置生成"
def ch_name $2 "Y 軸(Ft%)目盛表示グリッド線番号"
def ch_name $3 "グリッド線位置尺変換"
def ch_name $5 "x:データ"
def ch_name $6 "Y 軸(F1%)表示目盛値"
def ch_name $9 "Ft%"
def ch_name $10 "データ格納先チャネル番号"
def ch_name $11 "Ft 格納先チャネル番号"
def ch_name $12 "近似線 X 軸格納先チャネル番号"
def ch_name $13 "近似線 Y 軸格納先チャネル番号"
def ch_name $14 "同値除去フラグ"
def ch_name $17 "η 値:尺度母数
" def ch_name $18 "MTTF"
def ch_name $19 "近似線傾き:形状母数"
def ch_name $21 "近似線 X"
def ch_name $22 "近似線 Y"
def ch_name $23 "63.2%η"
def ch_name $24 "63.2%x"
def ch_name $25 "線種"
def ch_name $30 "データ列数"
def ch_name $31 "データ列選択"
def ch_name $32 "temp"
def ch_name $33 "格納 Ft%"
def ch_name $34 "text_form_counter"
def ch_name $35 "weibull_value"
def ch_name &1 "Y 軸目盛"
def ch_name &2 "線色"
def ch_name &3 "結果表示"
def ch_name &4 "データファイル名"
def ch_name &5 "データ名"
def ch_name &6 "temp"
def ch_name &7 "テキスト格納先ファイル名"
def ch_name &8 "描画線色"
def ch_name &9 "近似線色"
assign &8 = "#C02605","#893D74","#5A3D7E","#F79362","#B5B524","#346F9E","#3DAD25"
assign &9 = "#4B1AF7","#8080FF","#4080FF","#C080FF","#FF80FF","#FF00FF","#A589ED"
/* ----------csv file_read ----------*/
get file_name &4 $30 "csv" /*データファイル名取得*/
def file_id %10 &4(0) csv  /*データファイル名定義*/
read cell %10 1,1 0 &5     /*データ列名読み出し*/
$30 = ELM(&5)              /*データ列数取得*/
$31 = DAG($30,1)           /*データ列選択フラグ初期化*/
get check_box_status &5 $31 "データ列を選択して下さい"
$32 = ACC(DAG($30,1))  /* 選択列番号初期化*/
$30 = SUM($31)         /* 選択列数取得*/
case $30 > 0           /* 解析対象列存在*/
  proc exec{
    &5 = CREC(0,$31,&5)        /* 選択列名再構成*/
    $31 = ZSP($31,$32)         /* 選択列番号再構成*/
    &8 = CREC(0,$31,&8)        /* 描画線色再構成*/
    &9 = CREC(0,$31,&9)        /* 近似線色再構成*/
    /*-----同値除去する/しない選択----------*/
    $14 = 1
    get reply "同値除去しますか" "はい" "いいえ" $14
```

```
/*----- 故障率指定から相当値計算する ft%の入力----------*/
assign $33 = 99.9,90,80,70   /* 格納 Ft%初期化*/
get value $33(0)(F1),$33(1)(F1),$33(2)(F1),$33(3)(F1) "計算する Ft%値を入力して下さい"
$33 = ZSP(NEQ($33,0),$33)  /* Ft 入力 0 除外*/
/* ----ワイブル確率演算結果格納ファイル準備----------*/
assign &7 = &4(0)|"_ワイブル確率演算結果"
def file_id %11 &7 csv
def text_form %11 20,10      /* 20 行,10 列テキストフォーム定義*/
&3 = CNDT()                  /* 現在年月日取得*/
&6 = CNTM()                  /* 現在時刻取得*/
write cell %11 1,1 1 &3,&6 /* 現在年月日,時刻書き込み*/
case_true $14
  proc 同値除去コメント{
    write cell %11 1,4 1 "同値除去あり"
  }同値除去コメント
case_false $14
  proc 同値除去無しコメント{
    write cell %11 1,4 1 "同値除去なし"
  }同値除去無しコメント
assign &3 = &3|" "|&6
assign &6 = "データ名","形状母数(m)","尺度母数(η)","MTTF",$33(F1)|"%" /* ヘッダー行生成*/
write cell %11 2,1 0 &6       /*結果テキストヘッダー書き込み*/
/*--------- ワイブル確率紙演算処理----------*/
$32 = 0                       /* データ列カウンタ初期化*/
repeat_case $32 < $30         /*  データ列loop*/
  proc column_read{
    $10 = 50+$32              /* データ格納先チャネル番号*/
    $11 = 60+$32              /* Ft 格納先チャネル番号*/
    def ch_name $($11) &5($32)
    def ch_unit $($11) "Ft%"
    $12 = 70+$32              /* 近似線 X 軸格納先チャネル番号*/
    $13 = 80+$32              /* 近似線 Y 軸格納先チャネル番号*/
    assign &6 = &5($32)|" 近似線:Ft%"
    def ch_name   $($13) &6
    $34 = 3+$32               /* テキストフォーム書き込み行*/
    read cell %10 2,$31($32) 1 $5   /* データ列読み出し*/
    call proc lank_calc $5,$9,$14     /*外部演算処理ブロック呼び出し*/
    $($10) = $5
    $($11) = $9
    call proc param_calc $5,$9,$19,$17,$18,$21,$22 /* 外部演算処理ブロック呼び出し*/
    $($12) = $21
    $($13) = $22
    write cell %11 $34,1 1 &5($32),$19(0)(F4),$17(F4),$18(F4) /*名称,m 値,η 値,MTTF 書き込み
    */ call proc weibull_value $33,$35,$19          /* 相当値演算*/
    write cell %11 $34,5 0 $35(F4)
    $32 = $32+1
  }column_read
save text_form %11
/*---- weibull グラフ描画処理----------*/
assign &3 = &4(0)|" Weibull 解析 "|&3
/*  グラフ Y 軸目盛、グリッド線生成処理----------*/
$1 = SRT(1,LNK(ACC(DAG(9,0.1)),ACC(DAG(9,1)),ACC(DAG(9,10)),63.21,99,99.9)) /*グリッド線位置*/
assign $6 = 0.1,1,10,63.21,90,99,99.9        /* 表示目盛値生成*/
assign &1 = $6(F1)                           /* 目盛値文字列変換*/
$2 = EQP(2,$6,$1)                            /* 表示グリッド線番号抽出*/
$3 = LOG(LOG(1/(1-$1/100)))                  /*  グリッド線位置Ft 尺変換*/
assign $25 = $30<52>,$30<1>                  /*  描画線種*/
assign &2 = &8,&9                           /*線色*/
def graph_id @1 &3                           /*グラフ番号定義*/
def graph_line @1 $25 &2                      /* 線種 Dot 設定*/
def graph_x_axis @1 1,1e4 F1 log              /* グラフ X 軸定義*/
def graph_y_axis @1 $3(0),2,$3 &1,$2 1 linear /* グラフ Y 軸定義*/
/*--------- data plot ----------*/
/*assign $24 = 1,$17,$17      η 値描画する場合の座標計算------
- assign $23 = 63.21,63.21,0.1
$23 = LOG(LOG(1/(1-$23/100)))----------*/
plot @1 $[50,$30],$[70,$30] $[60,$30],$[80,$30]
def file_id %1 &4(0) grp
```

```
    save plot %1 @1 $[50,$30],$[70,$30] $[60,$30],$[80,$30]
  }exec
end
```

※ 外部演算処理ブロックの記述は省略していますが、実行に際しては end 行以降に使用する外部演算処理ブロックの記述が必要です。

<実行結果:ワイブル確率紙グラフ>



株式会社 デイシー
〒198-0024 東京都青梅市新町 9-2190
電話: 0428-34-9860
メール: info@deicy.co.jp

File: PL-U4101C1_PcWaveForm_取扱説明書_Script 記述方法編_Rev104_20120907.docx