ＡＤＳＰシリーズ

ＡＤＳＰ６７４－３３４

ソフトウェア・ユーザーズ・マニュアル

中部電機株式会社

目　　次

１．概要

本ソフトウェアは、３２ビット浮動小数点ＤＳＰボード「ＡＤＳＰ６７４－００」を使用した多チャンネルアクティブノイズキャンセラ（ＡＮＣ）を構築するためのソフトウェアライブラリです。

適応ディジタルフィルター（ＡＤＦ）と、Ａ／Ｄ・Ｄ／Ａ変換等のマクロライブラリ／関数ライブラリを核とし、これを用いてＡＮＣを構築したアセンブラルーチンと、モニタ機能を備えたマンマシンインターフェースプログラムとで構成されています。

本製品は以下の特徴を持っています。

- アセンブラ用のマクロライブラリと、Ｃ言語用の関数ライブラリが用意されています。
- ＡＤＦ等はアセンブラ用のマクロ定義で提供されており、アセンブラソースに読み込んで使用します。
- 独立した複数のＡＤＦを容易に構築できます。
- 高速に実行できるようマクロ展開になっています。
- マクロライブラリにはＡ／Ｄ変換、Ｄ／Ａ変換とのインターフェースが含まれています。
- マクロを使用したアセンブラによるＡＮＣルーチンが含まれています。
- タイマー割り込み処理の設定による制御周期の管理等のルーチンが含まれていて、外部機器の接続によりただちに実行可能です。
- インパルス応答の画面表示等、マンマシンインターフェースが含まれていて、リアルタイムにフィルタの状態等を確認できます。
- シミュレーションモードのルーチンが含まれていますので、ＡＤＦの理論的確認が行えます。

２．インストール手順

２．１　　インストール手順

１）　ＣＤ－ＲＯＭドライブに適応デジタルフィルタのディスクを挿入します。

２）　自動でセットアッププログラムが起動されます。

３）　自動で起動されない場合は、「スタート」→「ファイル名を指定して実行」を選択し、名前の欄に“[DRIVE]:¥setup.exe”を入力し「ＯＫ」を押してください。[DRIVE]にはＣＤ－ＲＯＭドライブ名を入力してください。

セットアッププログラムが起動されたら、セットアッププログラムの指示に従って、セットアップを完了します。

注１）パスワードは、製品添付のパスワードを入力してください。

(****-****-****-****)

２．２　アンインストール手順

１）本ソフトウェアをアンインストールするには、「スタート」→「設定」→「コントロールパネル」の順でクリックし、コントロールパネルを開きます。

２）「アプリケーションの追加と削除」をダブルクリックします。

３）「セットアップと削除」タグを選択し、一覧の中から「Adaptive Digital Filter *.**.**」を選択し、「追加と削除」ボタンをクリックします。

４）　以下の確認ダイアログが表示されますので、「はい」をクリックします。

５）　適応デジタルフィルタがアンインストールされ、完了したことを伝えるダイアログが表示されますので、「ＯＫ」をクリックして終了です。

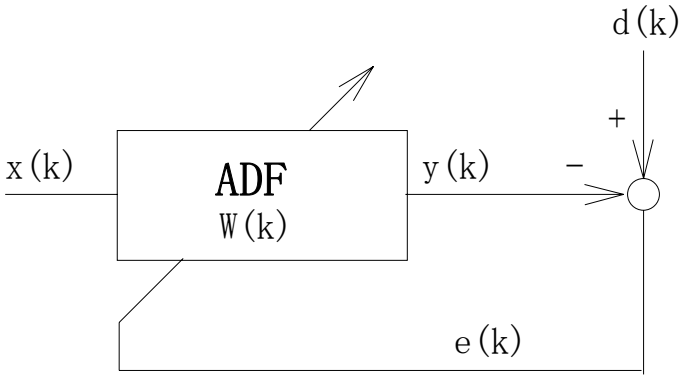
３．ＡＤＦ、ＡＮＣの計算アルゴリズム

３．１　ＡＤＦの計算アルゴリズム

本製品のＡＤＦ計算アルゴリズムは、時変な係数を有するＦＩＲフィルタで、係数更新則はＬＭＳアルゴリズムです。

適応フィルタの基本的な構成を図 1 に示します。

図 1　適応フィルタの構成図



適応フィルタの目的は、入力データx(k)を与えられた時の出力y(k)が、目標信号d(k)と等しくなるように、言い替えると、誤差信号e(k)のパワーが最小となるようにWを調整することです。

ある時刻 k におけるＡＤＦの出力y(k)は、フィルタへの入力データ列x(k)とフィルタ係数ｗi(k)を用いて次のように表せます。

$$y(k) = \sum_{i=0}^{TAP-1} wi(k)\ x(k-i) \qquad \cdots \qquad (1)$$

TAPはフィルタの係数の数で、タップ数と呼ばれます。

式を簡便にするためｗ、ｘをベクトルに置き換えると次のように表せます。

$$y(k) = W(k)^{T}\ X(k) \qquad \cdots \qquad (2)$$

この時、誤差信号e(k)のパワー（自乗平均誤差）Ｊ＝Ｅ［ｅ2(k)］を最小にする最適な係数Woは次のように表されます。(文献1,2)

$$Wo = R^{-1}P \quad \cdots \quad (3)$$

但し、

$$R = E[X(k)\ X(k)^T] \quad \cdots \quad (4)$$

$$P = E[d(k)\ X(k)^T] \quad \cdots \quad (5)$$

ここで、Ｅ［］は期待値を表します。

この最適な係数を用いた場合に得られる最小自乗平均誤差Ｊminは次式のように表されます。

$$Jmin = E[d^2(k)] - P^T Wo \quad \cdots \quad (6)$$

ところで、適応フィルタの適応プロセスは、係数をある初期値から始めて、最適な係数Woに近付くように調整してゆくことです。この調整法としてよく知られているのが最急降下法(Method of Steepest Descent)と呼ばれる方法です。この方法はＬＭＳアルゴリズムの基礎となる方法です。最急降下法では、次式によって再帰的に調整を繰り返します。

$$W(k+1) = W(k) + \mu\ (-\nabla(k)) \quad \cdots \quad (7)$$

ここで、μは毎回の繰り返しにおける補正量の大きさを制御するための正のスカラ定数で、ステップサイズ・パラメータと呼ばれます。▽(k)は時刻kにおける勾配を表します。

$$\nabla(k) = \frac{\partial J(k)}{\partial W(k)} \quad \cdots \quad (8)$$

誤差関数Ｊは２次形式であり、おわん状の曲面を（即ち単一の最小点）を持っています。従って、その時点での勾配ベクトル（導関数▽）の向きと逆向き（最急降下の向き）に係数を更新しつづければ、ついにはその最小点にたどり着くことができます。

最急降下法では勾配ベクトル▽が計算可能であることを前提としています。しかし、実際には正確な勾配ベクトルを得ることはほとんど不可能です。

ＬＭＳアルゴリズムは、この最急降下法の勾配ベクトル▽を瞬時自乗誤差e(k)を使って次式のように近似しようというものです。$e^2(k)$をＪ(k)の推定値と見なして、

$$\nabla(k) = \frac{\partial e^2(k)}{\partial W(k)} \quad \cdots \quad (9)$$

$$= 2\, e(k) \frac{\partial e(k)}{\partial W(k)} = -2\, e(k)\, X(k) \quad \cdots \quad (10)$$

上式の▽(k)を使い、式(7)を次式のように書き直します。

$$W(k+1) = W(k) + 2\mu e(k) X(k) \quad \cdots \quad (11)$$

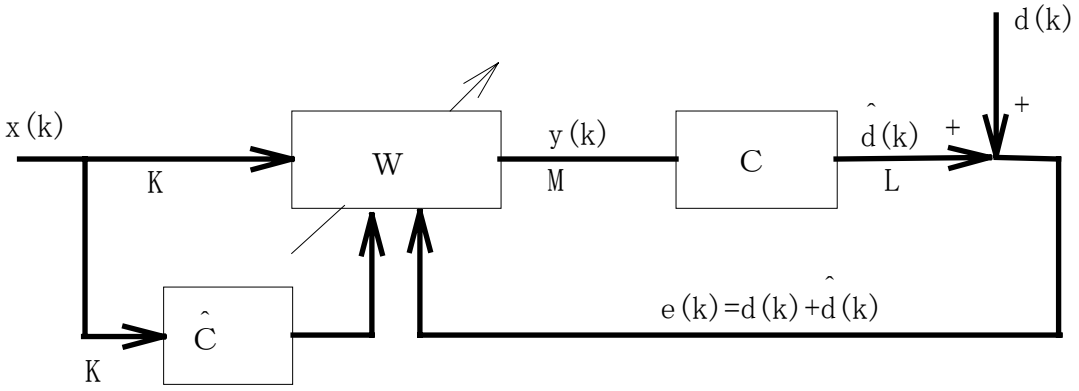
この式(11)がＬＭＳアルゴリズムによる適応プロセスの計算式です。

３．２　ＡＮＣの計算アルゴリズム

ＡＮＣ計算アルゴリズムは、Widrowにより提唱されたFilterd-Xによるものです。基本構成は、つぎに示すエラースキャニングによりＡＤＦを更新するError-Scanning Multiple-Error Filtered-X (MEFX) アルゴリズムです。(文献3)

図 2 にMEFXアルゴリズムを基本としたＡＮＣのブロック図を示します。

図 2　MEFX アルゴリズム



図のＣは制御対象の音響・電気等の特性を含んだシステムです。一次音源入力はＫ個、制御出力である二次音源はM個、エラーセンサはＬ個とします。一般にこのようなシステムを CASE(K-M-L)と表します。時刻 k に於けるｍ番目の二次音源は次の様に表されます。

$$y_m(k) = \sum_{i=0}^{I-1} w_{mi}\, x(k-i) \qquad \cdots \qquad (12)$$

制御対象システムＣは全て J次の FIRフィルタでモデル化できるものとし、ｍ番目の二次音源からｌ番目のエラーセンサに至るシステムを ｃml と表すとすると、ｌ番目のエラーセンサからの入力信号は次のように表せます。

$$e_l(k) = d_l(k) + \sum_{m=1}^{M} \sum_{j=0}^{J-1} c_{lmj}\, y_m(k-j) \qquad \cdots \qquad (13)$$

ここで、

$$r_{lm}(k) = \sum_{j=0}^{J-1} c_{lmj}\, x(k-j) \quad \cdots \quad (14)$$

とおくと、式(13)は次のように表せます。

$$e_l(k) = d_l(k) + \sum_{m=1}^{M} \sum_{i=0}^{I-1} w_{mi}\, r_{lm}(k-i) \quad \cdots \quad (15)$$

ここでマトリクス表現を用いると、エラー信号は次のように表せます。

$$e = d + R\, w \quad \cdots \quad (16)$$

ここで、

$$W_m^T = [w_{m0}, w_{m1}, \cdots, w_{mI-1}] \quad \cdots \quad (17)$$

$$w^T = [W_1^T, W_2^T, \cdots, W_M^T] \quad \cdots \quad (18)$$

$$r_{lm}^T = [r_{lm}(k), r_{lm}(k-1), \cdots, r_{lm}(k-I+1)] \quad \cdots \quad (19)$$

$$r_l^T = [r_{l1}^T, r_{l2}^T, \cdots, r_{lM}^T] \quad \cdots \quad (20)$$

$$R^T = [r_1, r_2, \cdots, r_L] \quad \cdots \quad (21)$$

$$e^T = [e_1(k), e_2(k), \cdots, e_L(k)] \quad \cdots \quad (22)$$

$$d^T = [d_1(k), d_2(k), \cdots, d_L(k)] \quad \cdots \quad (23)$$

評価関数　J　を次式に示すエラー信号の自乗平均の和とします。

$$J = E[e^T e] \quad \cdots \quad (24)$$

E［］は期待値を表します。
評価関数 J を最小とする最適なフィルタ係数 Wo は次式で表されます。

$$W_o = -[E[R^T R]]^{-1} E[R^T d] \quad \cdots \quad (25)$$

また、適当な初期値wからスタートして次式の更新を再帰的に行うことにより最適なフィルタに収束します。

$$w(k+1) = w(k) - \alpha R^T(k)\, e(k) \quad \cdots \quad (26)$$

しかしながら、K,M,L の増加に伴い膨大な計算量が必要となり、オンラインでの計算が困難となります。

そこで、エラースキャニングでは、１制御周期当りの計算量を疑似的に低減するため、フィルタ係数の更新を次の手順で行います。

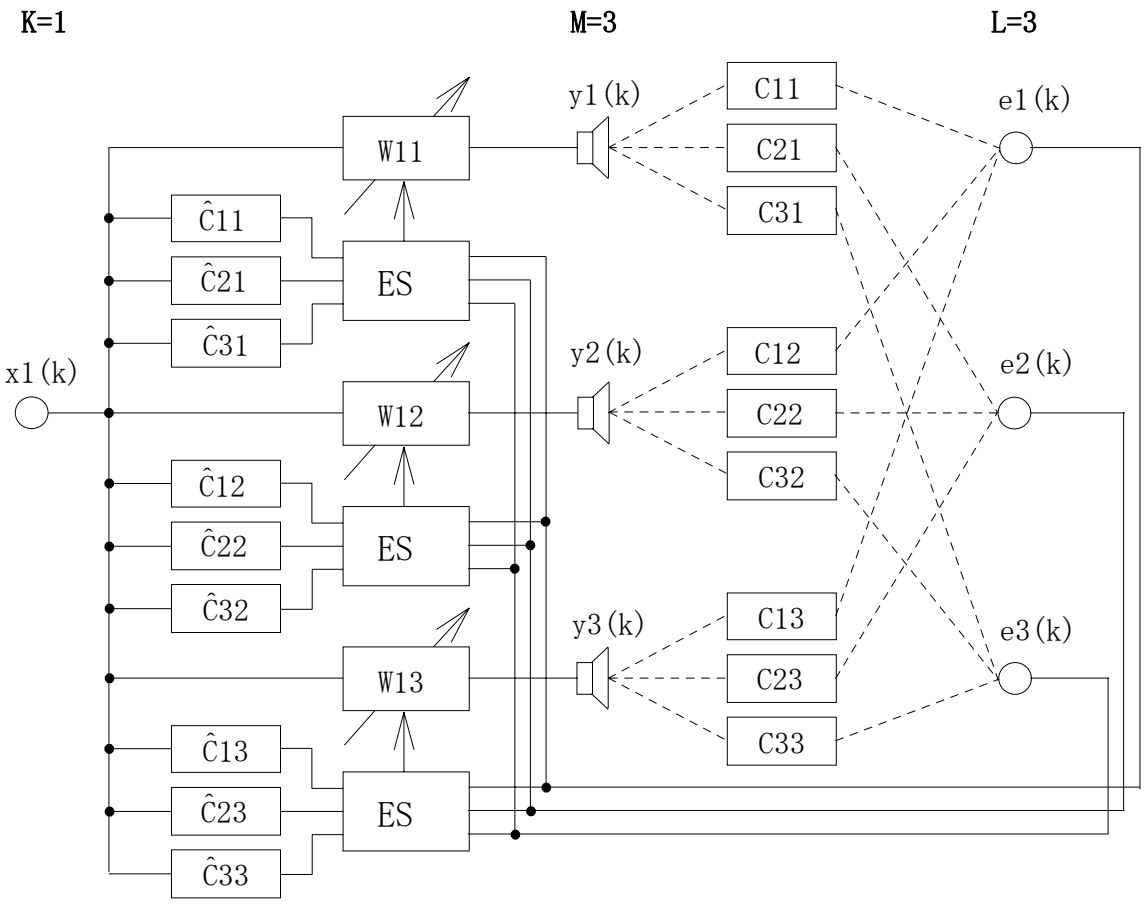
１）　毎サンプルごとのフィルタ係数更新過程で、ある１つのエラーセンサの瞬時誤差信号ｅｌ(k)にのみ着目します。

２）　誤差ｅｌ(k)に関与するシステムの伝達関数Ｃlm、を用いて、リファレンス信号ｒlm(k)を作り、全ての適応フィルタwmを Filtered-Xアリゴリズムで更新します。

３）次のサンプル時には別のエラー信号について上記の計算を行います。こうして次々エラーセンサーをスキャンしてゆきます。

注）本ソフトウェアに添付されているＡＮＣプログラムは上記アルゴリズムによるCASE(1-1-1)であるシングル版です。

図 3　CASE(1-3-3)の ANC ブロック図例



４．ソフトウェア説明

４．１　ファイル構成

本製品は、ライブラリファイルセットと、その使用例として、シミュレーション用ソフトウェアセット、実制御用ソフトウェアセット等が含まれています。各セットはＣＤ－ＲＯＭに以下に示すディレクトリに分けて収録されています。

¥LIB	ライブラリーファイルセット
¥LIB¥OCDMEM	ライブラリーファイル（内部ＲＡＭ使用）
¥LIB¥SAMPLE	Ｃ言語用ライブラリ関数のサンプルプログラム
¥ANC¥DSP	ＡＮＣ制御プログラムセット（ＤＳＰ）
¥ANC¥HOST	ＡＮＣ制御プログラムセット（ＨＯＳＴ）

各ディレクトリには、一部、同一名称のファイルが収録されている場合があります。同一名称のファイルどうしは、大体においては同等の処理内容ですが、ファイルによっては内容が一部異なる場合がありますから、混同しないよう注意してください。各ファイルの概要は次の通りです。

４．２　ファイルの概要

¥ｾｯﾄｱｯﾌﾟﾃﾞｨﾚｸﾄﾘ (ﾃﾞﾌｫﾙﾄ C:¥A674_334)

- **LIB［関数ライブラリ］**
  - FILTER67.SRC　アセンブラマクロソース
  - LMS67.H　ヘッダ File(共通)
  - *.ASM　ソース File
  - *.OBJ　オブジェクト File
  - LMS67.LIB　関数ライブラリ。
  - A674.CMD　リンカーコマンド File
  - Add.CMD　関数ライブラリ作成用コマンド File (ar6x にて使用)
  - **OCDMEM［内部 RAM 使用］**
    - *.ASM　ソース File
    - *.OBJ　オブジェクト File
    - LMS67on.LIB　関数ライブラリ
    - A674.CMD　リンカーコマンド File
    - Add_ON.CMD　関数ライブラリ作成用コマンド File （ar6x にて使用）
  - **SAMPLE［関数仕様例］**
    - T_ADPT.C　lmsadpt67()のサンプル
    - T_CLR.C　lmsclr67()のサンプル
    - T_FIR.C　lmsfir67()のサンプル
    - T_FIR0.C　lmscir67()と lmsfir670()のサンプル
    - T_FIR2.C　lmscir672()と lmsfir670()のサンプル
    - T_LMS.C　lms67()のサンプル

- ANC［ｱｸﾃｨﾌﾞﾉｲｽﾞｷｬﾝｾﾗｰﾗｲﾌﾞﾗﾘ］
  - DSP［ＤＳＰ側プログラム］
    - ANCS.C　メインプログラム
    - TIMSUB6x.C　割り込みタイマルーチン
    - FIRCOEF.C　ＦＩＲフィルタ係数
    - ANCS.H　メイン用File
    - TIMSUB6x.H　割り込みタイマルーチン用ヘッダFile
    - INTFUNC6.ASM　割り込み処理ソースFile
    - ANCS.CMD　リンカーコマンドFile
    - ＊.OBJ　オブジェクトFile
    - ANCS.OUT　実行File
  - HOST［ＨＯＳＴ側プログラム］
    - ANC.VBP　ＶＢプロジェクト
    - ANC.EXE　実行プログラム
    - A67XDLL.BAS　ＤＳＰ（６７）ボード用モジュールFile
    - Module1.BAS　ＡＮＣ用モジュールFile
    - Form1.frm　ＡＮＣ用フォームFile
    - Dialog.frm　ＡＮＣ用ダイログFile
    - *.COF　フィルタ係数File(ＤＳＰﾀﾞｳﾝﾛｰﾄﾞ用)

５．マクロライブラリ

５．１　ファイル名

FILTER67.SRC

５．２　形式

TMS320C6000浮動小数点ＤＳＰ用アセンブラソース

５．３　マクロエントリー一覧

CIRCUL67フィルター入力データー遅延バッファ構築（１）

CIRCUL672　　フィルター入力データー遅延バッファ構築（２）

FIRFILT67　　ＦＩＲフィルター出力の計算

LMSADPT67　　ＬＭＳアルゴリズムによる適応フィルターの係数更新

RANDGEN67　　M系列白色乱数の生成

AD_STRT　　Ａ／Ｄ変換の開始

AD_EOC　　Ａ／Ｄ変換の完了確認

READ_AD　　Ａ／Ｄ変換データの入力

WRIT_DA　　Ｄ／Ａ変換データの出力

５．４　外部参照される大域変数

Ａ／ＤおよびＤ／Ａ変換に関するマクロは、ハードウェアをアクセスする際に、以下に示す大域変数を参照しています。

_ad_start　　Ａ／Ｄ変換開始ポートを指標します。

_ad_busyＡ／Ｄ変換ビジーフラグポートを指標します。

_ad_baseＡ／Ｄ変換データポートを指標します。

_da_baseＤ／Ａ変換データポートを指標します。

各変数はユーザーがメインルーチンで定義し、かつ、その内容を各ポートを参照するように初期化してください。例として、Ｃとアセンブラの各言語による定義と初期化を示します。

【Ｃ言語】

```
int       *ad_start = (int*)0x033F0000;
int       *ad_busy =  (int*)0x033F0018;
int       *ad_base =  (int*)0x033F0000;
int       *da_base =  (int*)0x033F0008;
```

【アセンブラ言語】

```
                    .global _ad_start
                    .global _ad_busy
                    .global _ad_base
                    .global _da_base
_ad_start:          .word   033F0000h
_ad_busy:           .word   033F0018h
_ad_base:           .word   033F0000h
_da_base:           .word   033F0008h
```

参考：

1）Ｃ言語で定義した変数名は、Ｃコンパイラによりアセンブラに展開される際に、変数名の前に、’_’が、自動的に付加されます。

2）上記の例の定義をした場合、各変数は、Ｃ言語では .bss セクションに、アセンブラ言語では .text セクションに配置されます。コマンドファイルを操作してメモリー配置を変更する際には、注意が必要です。

使用上の注意:

すべてのマクロは、スモールモデルを想定して作られています。したがって、メインルーチンを記述する場合次の様な注意が必要です。

【Ｃ言語】

コンパイルはスモールモデルで行ない、ポインタで間接参照する領域以外はすべて、６４Ｋワード境界をまたがない連続する６４Ｋワード内に配置してください。

【アセンブラ言語】

レジスタで間接参照する領域以外はすべて、６４Ｋワード境界をまたがない連続する６４Ｋワード内に配置してください。

５．５　マクロの詳細

【凡例】

マクロ定義　MACRO_NAME　　Param1 , Param2 [,Param3]
MACRO_NAMEはマクロの名称を、Param1,Param2等は仮引数を表わしています。
[,Param3]は、引数Param3が省略可能である事を表わしています。

機能　マクロの機能の詳細を述べています。

引数詳細　マクロの各引数の意味、利用可能なアドレッシング等詳細な説明が述べられています。

出力　主にレジスタにより出力されるデータについて述べています。

参照変数　マクロの正常な実行に必要なメモリー/レジスタ変数について述べています。ここで説明している変数は、明示的な引数として表われておらず、暗黙に参照されているものです。

作業領域　マクロの実行に必要、または、実行の対象となるメモリ上の作業領域のうち、特に重要なものついて説明しています。

使用レジスタ　マクロ内で作業用に使用されるレジスタのうち、マクロ実行前の値が必ずしも保存されないレジスタを示しています。

[前提条件]

- **フィルタ入力（circul67,circul672），ＦＩＲフィルタ出力計算（FIRFILT67），ＬＭＳ適応フィルタ係数更新（LMSADPT67）はＢ６レジスタをサーキュラアドレッシングモード設定済として動作します。よって、各マクロ使用前にＡＭＲレジスタへ必ずサーキュラサイズ、レジスタの設定を行ってください。設定方法は（５．６データ遅延）を参照してください。**

ＣＩＲＣＵＬ６７

マクロ定義　CIRCUL67NewData , BufPtr

機能　ＦＩＲフィルター（カスケード接続不可）への入力データ列を構築します。データの遅延はＤＳＰのサーキュラアドレッシングを使用して実現しています。与えられた最新データは、マクロ呼び出し前すなわち１時刻前の時点に於ける最古のデータ上に上書きされ、同時に最古のデータを指標するポインターが更新されます。

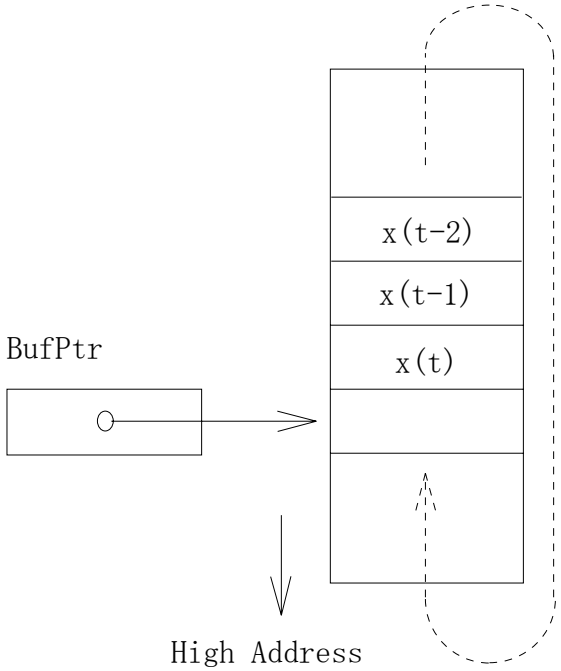
引数詳細　NewData　最新データx(t)を格納したレジスタを与えます。

BufPtr　遅延バッファの最古のデータを指標するポインタを与えます。（**５．６　データ遅延**を参照）

出力　特にありません。

参照変数　BKレジスタ（**５．６　データ遅延**を参照）

作業領域　CIRCUL67実行直後



使用レジスタ　B6(サーキュラアドレス),B7

ＣＩＲＣＵＬ６７２

マクロ定義　CIRCUL672　　NewData , OldData , BufPtr , Taps

機能　ＦＩＲフィルター（カスケード接続可）への入力データ列を構築します。データの遅延はＤＳＰのサーキュラアドレッシングを使用して実現しています。与えられた最新データは、マクロ呼び出し前すなわち１時刻前の時点に於ける最古のデータ上に上書きされ、同時に最古のデータを指標するポインターが更新されます。更に、引数で指定されたタップ数よりあふれ出たデータがOldData として得られます。これを後続段のＦＩＲフィルタへの最新データとする事により、ＦＩＲフィルタのカスケード接続が可能です。

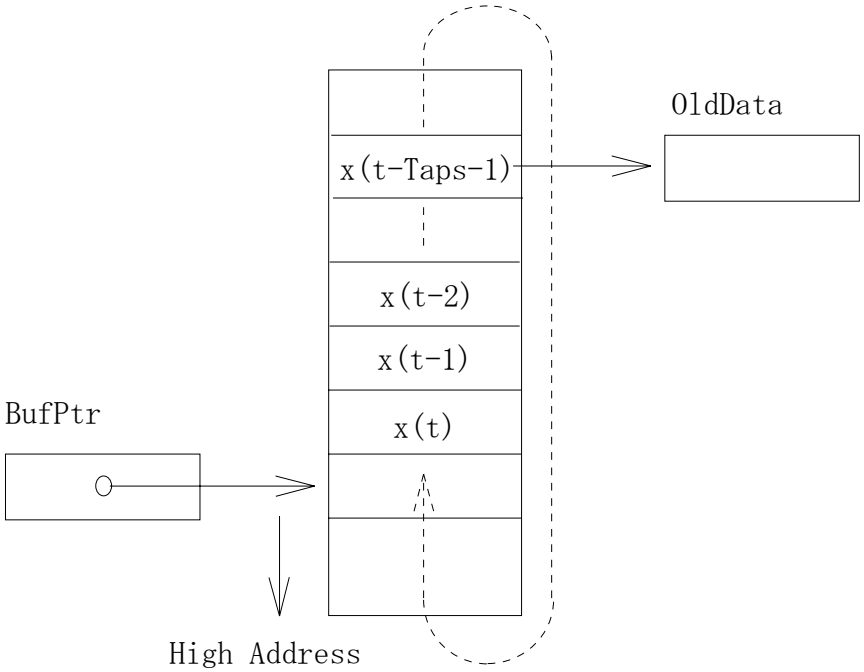
引数詳細

NewData	最新データx(t)を格納したレジスタを与えます。
OldData	遅延バッファからあふれ出たデータx(t-Taps)を格納する実数レジスタを与えます。NewData と同じレジスタを指定できます。
BufPtr	遅延バッファの最古のデータを指標するポインタを与　え　ます。（**５．６　データ遅延**を参照）
Taps	該当するＦＩＲフィルタのタップ数を与えます。

出力　OldData　遅延によりバッファからあふれ出たデータが得られます。

参照変数

作業領域　　CIRCUL672実行直後



バッファ長とタップ数が同じ場合（即ち、最新データ x(t) を格納する領域とあふれ出るデータ x(t-Taps-1)が同じ領域の場合）、x(t-Taps-1)はx(t)が書き込まれる前に読み出されます。

使用レジスタ　　ありません

ＦＩＲＦＩＬＴ６７

マクロ定義　FIRFILT67　　Input , Coef , Tap [, Gain]

機能　　ＦＩＲフィルターの出力を求めます。オプションで出力ゲイン変数を使ってゲインを可変にできます。入力データ列、インパルス応答列、タップ長等を引数で与えているので、複数の個別なＦＩＲフィルターを構築できます。

引数詳細

Input　　ＦＩＲフィルターへの入力データ列を指標するポインターを与えます。通常は、マクロ CIRCUL67 で最新データを遅延バッファに格納した後、同マクロの第二引数と同じものを与えます。（**５．６　データ遅延**を参照）

Coef　　インパルス応答列の先頭番地を指標するポインターを与えます。インパルス応答列は先頭番地が h(0) です。

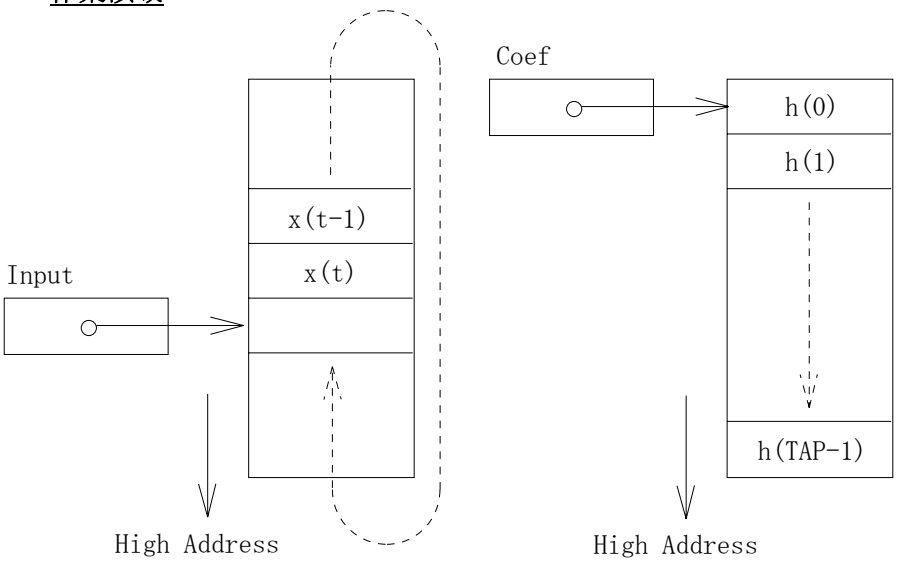
Tap　　ＦＩＲフィルターのタップ長を指示します。

Gain　　ＦＩＲフィルターの出力ゲインを決める実数型の定数または変数を与えます。Gainは省略可能で、省略時のゲインは１です。

出力　　B0　　ＦＩＲフィルターの出力です。

$$B0 = \sum_{k=0}^{TAP-1} x(t-k)\cdot h(k) \cdot Gain$$

作業領域



使用レジスタ　　A1,A2,A6～A13,B0,B1,B2,B6～B11

ＬＭＳＡＤＰＴ６７

マクロ定義　LMSADPT67　　Input , Coef , Tap , Error , Mu2

機能　ＦＩＲフィルターの係数（インパルス応答列）を、ＬＭＳアルゴリズムにて更新することにより、適応フィルターを構築します。ＦＩＲフィルターの出力y(t)は、FIRFILT67で求めます。この出力と目標信号d(t)との誤差と、ステップサイズパラメーター μ をもとに適応の計算をします。計算式は、更新後のフィルター係数をｈ’とすると次の式で計算されます。

$$e(t) = d(t) - y(t) \quad \cdots\cdots \quad (1)$$

$$h'(k) = h(k) + 2 \cdot \mu \cdot e(t) \cdot x(t-k) \quad \cdots\cdots \quad (2)$$

ＬＭＳアルゴリズムおよび計算式の詳細については２章を参照してください。

引数詳細

Input　ＦＩＲフィルターへの入力データ列を指標するポインターを与えます。基本的な適応フィルターを構築する場合は、マクロFIRFILT67 で出力を計算後、同マクロの第一引数と同じものを与えます。

（５．６ データ遅延を参照）

Coef　インパルス応答列の先頭番地を指標するポインターを与えます。インパルス応答列は先頭番地が h(0) です。

Tap　ＦＩＲフィルターのタップ長を指示します。

Error　目標信号とフィルター出力との誤差、式(1)を与えます。

Mu2　ステップサイズパラメータ、すなわち式(2)の $2 \cdot \mu$ を正規化したうえ与えます。ステップサイズパラメータは、適応の速さや精度を左右する重要なパラメータで､小さすぎると適応速度が遅く、大きすぎると精度が低下したり、場合によっては発散してしまいます｡通常0.001～0.01程度とします｡ステップサイズパラメータを０とすると適応を停止します｡ステップサイズパラメータは、フィルターに入力する信号の振幅により正規が必要で、信号の振幅をＷとすると正規化されたMu2は、

$Mu2 = 2 \cdot \mu \ / \ W^2$

とします。

出力　特にありません

作業領域　FIRFILT67を参照してください。

使用レジスタ　A0,A1,A2,A6～A10,B0～B12

ＲＡＮＤＧＥＮ６７

マクロ定義　RANDGEN67　　［ Gain ］

機能　Ｍ系列の乱数による白色ノイズを生成します。マクロ内では１５ビットのＭ系列整数乱数を生成し、これを実数変換し、±Gain または ±１の振幅の乱数としています。

引数詳細　Gain　　生成するノイズのピーク値を指示します。省略可能です。例えば、０．５を指示すると、ピーク値±０．５のノイズが、また、省略すると、ピーク値±１のノイズが生成されます。

出力　B0　　ノイズデータ

参照変数　ありません

作業領域　以下の局所変数（定数）を使用しています。

mpycnst,addcnst,divcnst,rndseed

これらの変数（定数）は、マクロファイル内に定義されており、ユーザが用意する必要はありません。

使用レジスタ　A1,A2,B1,B2

備考　同一ファイル内で、複数のルーチンからこのマクロを参照した場合、これらはいずれもひとつの系列の乱数となります。個別にソースファイルを用意すれば、ソースファイルごとに独立した系列の乱数列を生成することができます。

ＡＤ＿＿ＳＴＲＴ

マクロ定義　AD_STRT

機能　Ａ／Ｄ変換を開始します。変換を開始するのみで、変換完了は待機しませんから、その間他の処理を行なうことができます。Ａ／Ｄ変換データの読み込みに先立ち、AD_EOC で変換完了を待機してください。

引数詳細　ありません。

出力　ありません。

参照変数　_ad_start　Ａ／Ｄ変換開始ポートを指標するポインターです。予めＡ／Ｄ変換開始ポートアドレスを格納しておきます。

作業領域　ありません。

使用レジスタ　B0,B1

備考　このマクロは単一のＡ／Ｄボードにのみ対応しています。

ＡＤ＿＿ＥＯＣ

機能　Ａ／Ｄ変換完了を待機します。Ａ／Ｄ変換データの読み込みに先立ち、AD_EOC で変換完了を待機してください。Ａ／Ｄ変換が完了するまでマクロ内でループしています。

引数詳細　ありません。

出力　ありません。

参照変数　_ad_busy　Ａ／Ｄ変換ビジーポートを指標するポインターです。予めＡ／Ｄ変換ビジーポートアドレスを格納しておきます。

作業領域　ありません。

使用レジスタ　B1,B2

備考　このマクロは単一のＡ／Ｄボードにのみ対応しています。

ＲＥＡＤ＿ＡＤ

マクロ定義　READ_AD　　Port , Reg

機能　Ａ／Ｄ変換されたデータをレジスタに読み込みます。Ａ／Ｄ変換ボードからの入力データはオフセットバイナリですが、マクロ内で実数に変換し指定されたレジスタに格納します。実数への変換は必要最小限のステップに押さえるため、スケーリングは行なっていません。従って１２ビットＡ／Ｄでは出力データは－２０４８～２０４７の範囲の値となります。

引数詳細

Port	入力するポート番号、０～１を指示します。
Reg	入力データを格納するレジスタを指定します。

出力

Reg	入力および変換された実数データ。

参照変数

_ad_base	Ａ／Ｄ変換データポートを指標するポインターです。Ａ／Ｄポートのうち先頭のポートアドレスを予め格納しておきます。

作業領域　ありません。

使用レジスタ　A0,A1,B0,B1,Reg

備考　このマクロは単一のＡ／Ｄボードにのみ対応しています。

ＷＲＩＴ＿ＤＡ

マクロ定義　WRIT_DA　　Reg , Port [, Work]

機能　レジスタ Reg の実数データ（－２０４８～２０４７）をＤ／Ａへ出力します。実数データはマクロ内でオフセットバイナリに変換したうえＤ／Ａ変換ボードへ出力されます。オフセットバイナリへの変換は必要最小限のステップに押さえるため、スケーリングは行なっていません。また、ＭＡＸチェックを行なっていませんから、計算結果がＤ／Ａで取り扱える値を越える可能性がある場合はマクロ実行前にクリッピングしてください。

引数詳細

Reg	出力データが格納されているレジスタを指定します。
Port	出力するポート番号、０～１を指示します。
Work	バイナリ変換作業に使用するレジスターを指示します。省略可能で、省略した場合は、B2が使用されます。

出力　ありません。

参照変数

_da_base	Ｄ／Ａ変換データポートを指標するポインターです。Ｄ／Ａポートのうち先頭のポートアドレスを予め格納しておきます。

作業領域　ありません。

使用レジスタ　B1,A0,B2又はWork

備考　このマクロは単一のＡ／Ｄボードにのみ対応しています。

５．６　データ遅延

ＦＩＲフィルタはその出力の計算に際し、フィルタへの過去の入力が必要となります。一定期間の過去の入力を保存するため、一般には、１時刻経過する毎に１つずつデータがシフトする遅延バッファを用います。ＤＳＰは、こうしたデータの遅延バッファを容易に実現できるよう、サーキュラ・アドレッシング機構を備えています。これはサーキュラバッファと呼ばれ、バッファ内をリング状に移動するポインタを用意し、これを用いて、時刻の更新の際に、保存されているデータ自体は移動させず、不要となった最古データ上に最新データを上書きをする機構です。

ＡＭＲレジスタはサーキュラバッファのサイズとレジスタを決定する重要なレジスタです。ＡＭＲレジスタの値と、バッファの配置は次の条件を満たさなくてはいけません。（バッファサイズは2^n 指定です。ｎ－１をサイズ指定します。）

＜サーキュラアドレッシングモード設定例＞

Ｂ６レジスタへ　サイズ１０２４バイト（タップ数　２５６）
のサーキュラーアドレッシングモードを設定

１）バッファ先頭アドレスはバッファサイズ１０２４の整数倍に設定

２）ＡＭＲレジスタへレジスタ、バッファサイズを指定

```
mvkl    0x1000,A2          Ｂ６レジスタ指定
mvkh    0x00091000,A2      サイズ指定
mvc     A2,AMR
```

本ソフトウェアでは、フィルタタップ数の最大を、256 とし、従ってAMRレジスタはフィルタ関連マクロで 256 に、各遅延バッファの先頭アドレスはメインルーチンで 256*4 の整数倍に、それぞれ設定されています。

６．関数ライブラリ

６．１　ファイル名

LMS67.LIB　　スモールモデル用ライブラリ

LMS67_ON.lib　スモールモデル用ライブラリ（内部ＲＡＭ使用）

LMS67.H　　ヘッダファイル

６．２　形式

TMS320C6000 浮動小数点ＤＳＰ用Ｃコンパイラーライブラリ

６．３　関数一覧

lms67()	フィルタへのデーター入力、出力計算と係数更新
lmsadpt67()	フィルタ係数更新
lmsclr67()	フィルター初期化
lmscir67()	フィルター入力データー遅延バッファ構築（１）
lmscir672()	フィルター入力データー遅延バッファ構築（２）
lmsfir67()	フィルターへのデータ入力と出力計算
lmsfir670()	フィルタ出力計算

６．４　管理構造体

関数ライブラリは、個々のフィルターを構造体で管理します。構造体は、ヘッダファイルに定義されていて、以下に示すメンバーにより構成されています。

```
struct lmsparameter {
        float   *h;
        float   *x;
        int     tap;
        int     bk;
        float   beta;
 };
```

float　*h;　フィルタの係数（インパルス応答列）の先頭を指標します。先頭がh(0)で、h(1)、h(2)の順に上位アドレスへ延びて行きます。

float　*x;　フィルタへの入力データ列を保持する遅延バッファ内の最古のデータを指標しています。このバッファの配置には制約があります。詳しくは、**５．６　データ遅延**を参照してください。

int　tap;　フィルタの長さを指定します。指定する値は、タップ数としてください。

int　bk;　入力データ遅延バッファの長さを決定する定数です。この値はバッファの配置と深い関連があります。詳しくは、**５．６　データ遅延**を参照してください。

float　beta;　係数更新のステップサイズパラメータを与えます。詳しくは、マクロ LMSADPT67 の引数 Mu2 を参照してください。

６．５ 関数の詳細

lms67()

関数定義　float lms67(float _xn,float _err,struct lmsparameter *_lmsprm);

機能　データの入力、フィルタの出力、係数更新の全ての処理を１回の呼出しで実行します。出力計算と係数更新の処理順序は、まず係数の更新を行い、続いて出力の計算順で行われます。係数の更新は１時刻前の誤差を元にして行われるため、動作的には、「１時刻前の誤差による係数の更新を今回まで保留していて、それと、今回の出力の計算とを同時に行う」事になります。lmsadpt67(),lmsfir67()等の個別の関数を用いるより最も処理効率の高い関数です。

引数　float　_xn;　今回のフィルタへの入力データです。
float　_err;　１時刻前の誤差を与えます。
struct lmsparameter *_lmsprm;
フィルタ管理構造体へのポインタを与えます。

戻り値　今回のフイルタの出力を返します。

使用例

```
#include<lms67.h>
#define TAP     10
#define BK      5
#define BETA    0.25
#define REPEAT  600

#pragma DATA_ALIGN(x1,64);
float   x1[16];

float   coef[TAP] = {0,0,0,0,0,0,0,0,0,0};
lmsprm  prms = {&coef[0],&x1[0],TAP,BK,BETA};
```

```
void main(int argc,char **argv)
{
        int     t;
        float   xn,dn,yn,err;
        lmsclr67(&prms);
        for( err=0,t=0; t<REPEAT; t++ ){
                xn = NewInput();        /* フィルタへの入力取得 */
                dn = DesiredOutput();           /* 目標信号取得 */
                yn = lms67(xn,err,&prms);       /* 出力と適応計算 */
                err = dn - yn;                  /* 今回の誤差 */
        }
}
```

lmsclr67()

関数定義　　void　lmsclr67(struct lmsparameter *_lmsprm);

機能　　データ遅延バッファを０でクリアすることによりフィルタを初期化します。

引数　　struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　　ありません。

使用例

```
#include<lms67.h>
#define TAP     10
#define BK      5
#define BETA    0.001

#pragma DATA_ALIGN(x1,64);
float   x1[16];

float   coef[TAP] = {0,0,0,0,0,0,0,0,0,0};
lmsprm  prms = {&coef[0],&x1[0],TAP,BK,BETA};

void main(int argc,char **argv)
{
        lmsclr67(&prms);
}
```

lmsadpt67()

関数定義　　void　lmsadpt67(float _err,struct lmsparameter *_lmsprm);

機能　　係数の更新のみを行います。これに先立ち、フィルタへのデータの入力と出力の計算及び誤差の計算を行っておく必要があります。この関数を用いる事で、Filtered-X等のアルゴリズムを構築できます。

引数　　float　_err;　誤差を与えます。

struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　　ありません。

使用例

```
#include<lms67.h>
#define TAP     10
#define BK      5
#define BETA    0.003
#define REPEAT  5000

#pragma DATA_ALIGN(x1,64);
float   x1[16];
float   coef[TAP] = {0,0,0,0,0,0,0,0,0,0};
lmsprm  prms = {&coef[0],&x1[0],TAP,BK,BETA};

void main(int argc,char **argv)
{
        int     t;
float   xn,dn,yn,err;
lmsclr67(&prms);
for( err=0,t=0; t<REPEAT; t++ ){
                xn = NewInput();        /* フィルタへの入力取得 */
                dn = DesiredOutput();           /* 目標信号取得 */
                yn = lmsfir67(xn,&prms);        /* 出力計算 */
                err = dn - yn;                  /* 今回の誤差 */
                lmsadpt67(err,&prms);           /* 適応計算 */
        }
}
```

lmsfir67()

関数定義　float　lmsfir67(float _xn,struct lmsparameter *_lmsprm);

機能　フィルタへのデータの入力と出力の計算を行います。一般のＦＩＲフィルタと同じ動作をします。データ入力により不要となった過去のデータは消失します。従ってカスケード接続する場合は、lmscir672(),lmsfir670()等を使ってください。

引数　float　_xn;　今回のフィルタへの入力データです。

float　_err;　１時刻前の誤差を与えます。

struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　フィルタからの出力です。

使用例　（関数 lmsadpt67()の使用例を参照してください。）

lmsfir670()

関数定義　float　lmsfir670(struct lmsparameter *_lmsprm);

機能　フィルタ出力の計算を行います。lmsfir()との相違点は、lmsfir670()はデータの入力を伴わず、畳み込み演算のみを行っている点です。Lmsfir670()ではデータの入力は、lmscir672()等を用いて予め行います。フィルタをカスケード接続する場合に使用します。

引数　struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　フィルタからの出力です。

使用例　（関数 lmscir67(),lmscir672()の使用例を参照してください。）

lmscir67()

関数定義　void lmscir67(float xn,struct lmsparameter *_lmsprm);

機能　フィルタへの入力データを遅延データバッファへ挿入します。フィルタの出力計算は、lmsfir670()を用いて行い、また、係数の更新は、lmsadpt67()を用いて行います。この関数は、lmscir672()のカスケード接続の為の機能を省いたものです。この関数は、出力計算と、係数の更新に別の入力データ列を使用するFiltered-X等のアルゴリズムの実現に有用です。

引数　float　_xn;　今回のフィルタへの入力データです。

struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　ありません。

使用例

```
#include<lms67.h>
#define TAP     12
#define BK      7
#define BETA    0.003
#define REPEAT  5000
#pragma DATA_ALIGN(x1,64);
float   x1[16];
float   coef[TAP] = {0,0,0,0,0,0,0,0,0,0,0,0};
lmsprm  prms = {&coef[0],&x1[0],TAP,BK,BETA};
void main(int argc,char **argv)
{
        int     t;
float   xn,dn,yn,err;
lmsclr67(&prms);
for( err=0,t=0; t<REPEAT; t++ ){
                xn = NewInput();        /* フィルタへの入力取得 */
                dn = DesiredOutput();   /* 目標信号取得 */
                lmscir67(xn,&prms);     /* 入力データ挿入 */
                yn = lmsfir670(&prms);  /* 出力計算 */
                err = dn - yn;          /* 今回の誤差 */
                lmsadpt67(err,&prms);   /* 適応計算 */
        }
}
```

lmscir672()

関数定義　float lmscir672(float xn,struct lmsparameter *_lmsprm);

機能　フィルタへの入力データを遅延データバッファへ挿入します。フィルタの出力計算は、lmsfir670()を用いて行い、また、係数の更新は、lmsadpt67()を用いて行います。この関数は、データ挿入により押し出された過去のデータが出力として得られます。これを用いカスケード接続が可能です。

引数　float　_xn;　今回のフィルタへの入力データです。

struct lmsparameter *_lmsprm;

フィルタ管理構造体へのポインタを与えます。

戻り値　不要となった過去のデータです。入力をx(n)とすると、x(n-TAP)が得られます。

使用例

```
#include<lms67.h>
#define TAP     8
#define BK      5
#define BETA    0.001
#define REPEAT  30
#pragma DATA_ALIGN(x1,32);
float   x1[8];
#pragma DATA_ALIGN(x2,32);
float   x2[8];

float   coef1[TAP] = {0,0,0,0,0,0,0,0};
lmsprm  prms1 = {&coef1[0],&x1[0],TAP,BK,BETA};
float   coef2[TAP] = {0,0,0,0,0,0,0,0};
lmsprm  prms2 = {&coef2[0],&x2[0],TAP,BK,BETA};
```

```
void main(int argc,char **argv)
{
      int   t;
      float xn,xnn,dn,yn,err;
      lmsclr67(&prms);
      for( err=0,t=0; t<REPEAT; t++ ){
            xn = NewInput();                /* フィルタへの入力取得 */
            dn = DesiredOutput();           /* 目標信号取得 */
            xnn = lmscir672(xn,&prms1);     /* ＃１入力データ挿入 */
            lmscir67(xnn,&prms2);           /* ＃２入力データ挿入 */
            yn = lmsfir670(&prms1) + lmsfir670(&prms2);
                                            /* 出力計算 */
      }
}
```

７．プログラム例

７．１　割り込み処理ルーチン

適応フィルタの計算や消音制御等は、全て割り込み処理ルーチンの中で行われています。割り込み処理は速度が重要視されますので、アセンブラで記述されています。ファイル名は INTFUNC6.ASM です。割り込みはタイマーにより定期的に発生するようメインルーチン内で初期化され、その処理アドレスとしてこの処理ルーチンをメインルーチンがセットしています。

また、フィルタ係数やデータ遅延バッファ等はメインルーチン側で定義しています。

本ソフトでは、割り込み処理ルーチンとして次の６つを用意しています。

_c_int10　無実行ルーチンです。

_c_int11　Ｃ１検出系同定ルーチンです。

_c_int12　Ｃ２消音系同定ルーチンです。

_c_int2*　消音制御ルーチンです。_c_int1*で同定したＣ１、Ｃ２を使って消音処理を行います。

同定処理は１つのスピーカーから乱数ノイズを出力し、それをプライマリセンサ又はエラーセンサから取り込み、ＡＤＦにて同定しています。

７．２　ＤＳＰ側メインルーチン

ＤＳＰ側のメインルーチンは作業領域の初期化、タイマーの初期化と割り込み処理の登録等、初期化処理を終えた後、ホスト側プログラムとハンドリングにより通信を行います。通信の主導権はホスト側プログラムが持ち、これより送られて来たコマンドに応答する形式で処理を進めます。

ホストからのコマンドには次のものが用意されていて、ヘッダファイル ANCS.H に定義されています。

```
#define     C_NOP           0       無実行処理起動
#define     C_ID1_GO1       検出系C1の同定開始
#define     C_ID1_READ      2       検出系C1の係数送信
#define     C_ID2_GO4       消音系C2の同定開始
#define     C_ID2_READ      5       消音系C2の係数送信
#define     C_ID_END7       同定終了
#define     C_ANC_GO8       消音処理開始
#define     C_ANC_END       9       消音処理終了
#define     C_ANC_READ      10      消音用フィルタW1を送信
#define     C_PRM_SET       12      パラメータ変更
#define     C_ID_C1_CL      13      検出系C1係数初期化
#define     C_ID_C2_CL      14      消音系C2係数初期化
#define     C_CTRL_COEF_CL  15      フィルタ係数クリア
#define     C_FIR_COEF      16      フィルタ係数交換
#define     C_GAIN_SET      17      ゲイン設定
#define     C_LOG_START     33      データロギング開始
#define     C_LOG_STATUS    34      データロギング状況取得
#define     C_PRIM_READ     35      プライマリ波形読出し
#define     C_ERROR_READ    36      エラー波形読出し
#define     C_CTRL_READ     37      制御波形読出し
```

これらのコマンドは、ホスト側プログラムの関数 CommandTfr() が送信し、これをＤＳＰ側の関数 CommandRead() で受信しています。コマンドによっては付随するパラメータを構造体 struct handling （ANCS.H内に定義）のメンバー iprm1 等を使用して送信するもの、また、データとして80003000H番地 を使用して送信するもの、等があります。

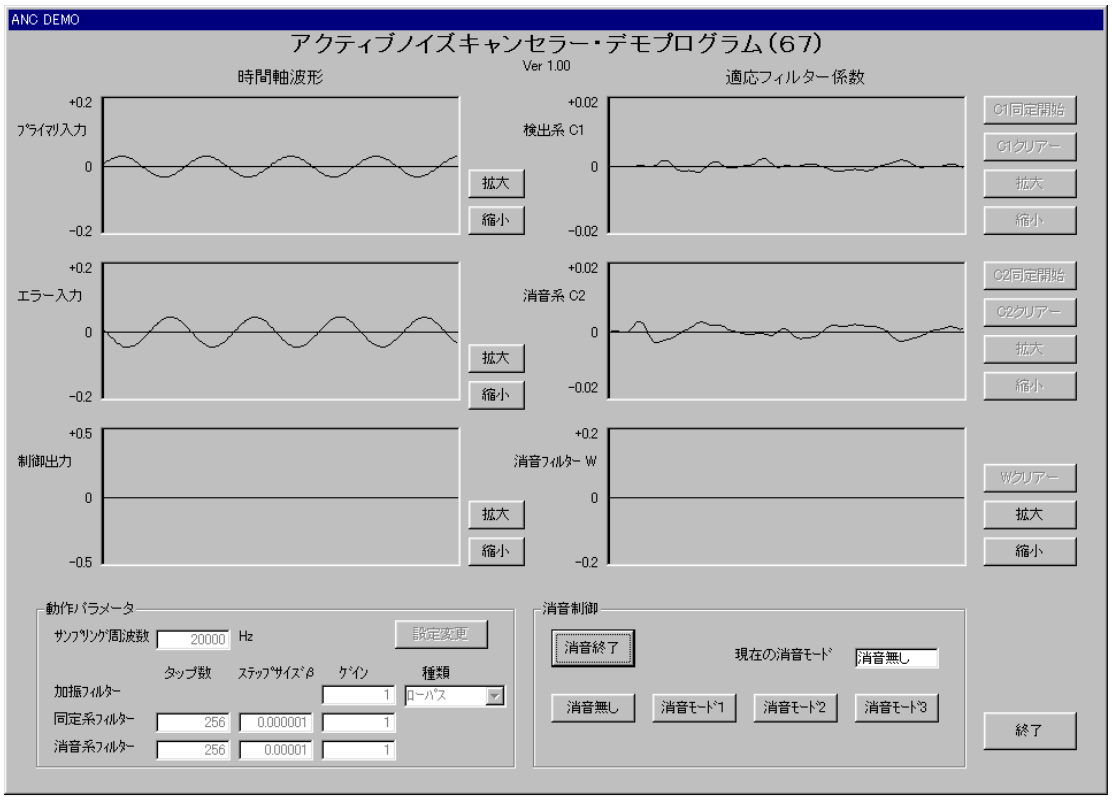
これら対ホストのコマンド処理は、割り込み処理による同定や制御処理等と並行して行われます。割り込み処理で参照している変数へ上書きするようなコマンドの場合には注意が必要です。こういった処理をする場合は、ホスト側で一旦割り込み処理を停止したうえで行う事をおすすめします。

７．３　ホスト側ルーチン

主に、マンマシンインターフェースを司るプログラムです。

このプログラムは、Ｖｉｓｕａｌ　Ｂａｓｉｃ６．０にて作成しています。

以下に操作方法について説明します。



１）ＨＯＳＴプログラム実行時に、ＤＳＰボードのリセット、ＤＳＰプログラムのロード等を行います。

２）動作パラメータを確認します。
（変更時は設定変更ダイアログにて変更してください）

３）検出系Ｃ１の同定処理を行います。

４）消音系Ｃ２の同定処理を行います。

５）擬似騒音を発生させ、消音を開始します。

７．４　ＤＳＰ～ホスト通信異常

もし、ＤＳＰ側の割り込み処理が過負荷で制御周期内に完了しない場合、メイン処理、即ちホストからのコマンド処理が全く行われなくなります。ホスト側は通信関数の中で通信の所要時間を監視しており、一定時間（５～６秒）経過しても通信が完了しない場合はこの状態に陥っていると判断します。復旧するにはＤＳＰをリセットする方法しかありません。ＤＳＰのリセットとプログラムの再ロードを行い復旧します。本プログラムでは通信ルーチンの中で自動的に行っています。ＤＳＰはプログラムの再ロードにより、同プログラム中に定義されているデフォルト値が有効となり、プログラム起動直後の状態となります。しかしホスト側で保持しているパラメータはそのままになっています。したがって、このままの状態では、ＤＳＰ内のパラメータとホスト内のパラメータの各値に相違がある事になります。不都合が生じるといけませんから、その場合は必ず、パラメータを再設定してください。ＤＳＰ側の割り込み処理の負荷を低減するには、①フィルタのタップ数を減らす、②制御周期を長くする、のいずれかです。

７．５　シミュレーションルーチン

シミュレーション用のプログラムでは、ＡＮＣの処理以外に、次に示すシミュレーションのための処理を行なっています。

●ホスト側ルーチン

シミュレーション用フィルタ係数ファイルをディスクから読み込み、ＤＳＰボードへセットします。

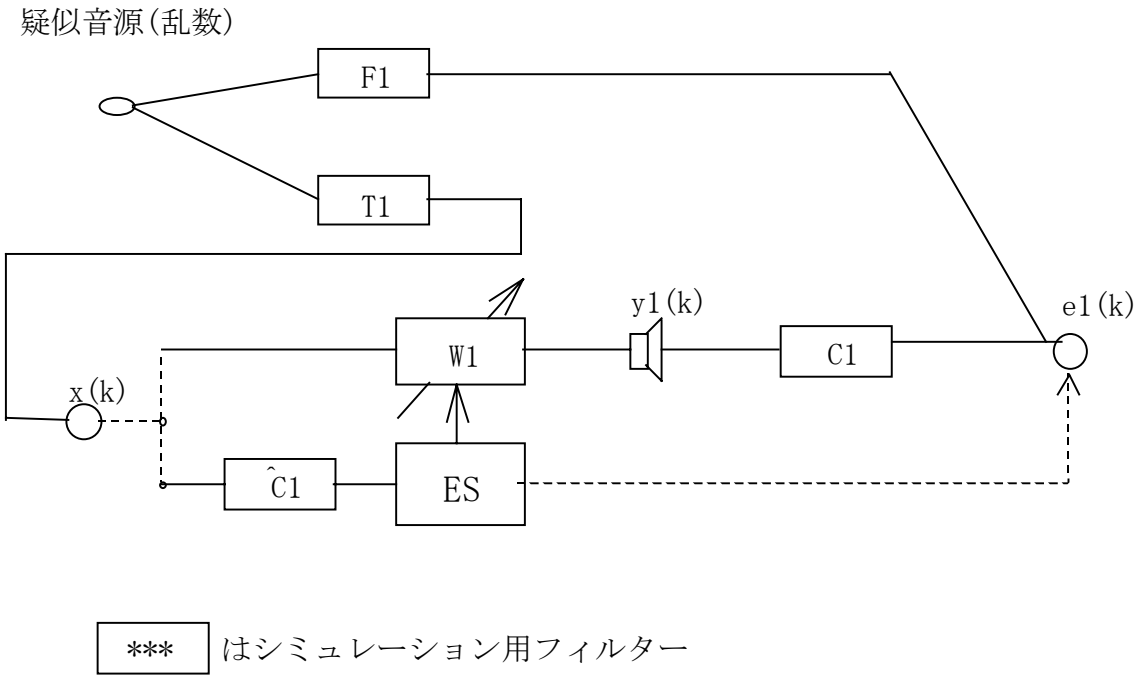
●ＤＳＰ側メインルーチン

Ａ／Ｄ、Ｄ／Ａ変換ボードのポートを指標するポインターを、内部メモリーを指標するように初期化します。これにより、入出力はメモリーに対して行なわれます。

- 割り込み処理側ルーチン

メモリに出力された二次音源をシミュレーション用フィルタに入力し、制御対象システムをシミュレーションします。図 4 は、シミュレーション用フィルタのブロック図です。

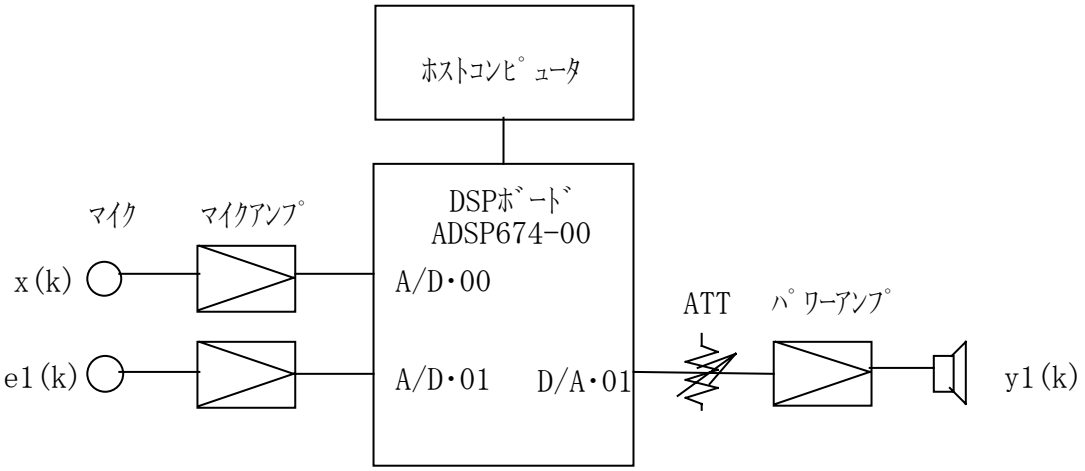
図 4　シミュレーション用フィルタブロック図



８．機器構成例

この章では、先のプログラム例の章で示した例に適応するハードウェア例を示します。

図 5　機器ブロック図



Ａ／Ｄ変換のレンジは±２．５Ｖですから、マイクに市販のマイク等を使用する場合、その出力レベルに応じてマイクアンプを用意します。できれば交流結合のアンプが良いでしょう。ゲインの定量的な精度はさほど重要ではありません。重要なことは、最大レベルの信号が入力されたときにＡ／Ｄ変換への入力がオーバーしないこと、歪みが少ないこと、ゲインが安定であること、等です。

Ｄ／Ａ変換のレンジは±１Ｖですから、パワーアンプの入力レベルに適合するよう、必要に応じてアッテネータ（ＡＴＴ）を通してパワーアンプに接続します。Ａ／Ｄ変換同様、ゲイン精度は重要でなく、歪み、ゲインの安定性などに留意します。

９．コンパイラのバージョン

使用したコンパイラのバージョンは以下の通りです。

ＤＳＰ	ＣＰＬ３２０６Ｘ－ＭＳ	Ver1.0
ホスト	ＶｉｓｕａｌＢａｓｉｃ	Ver6.0

１０．参考文献

（1）S.Haykin , "Introduction to Adaptive Filters"
Macmillan , New York , (1984)
武部　幹　訳 , 「適応フィルター入門」
現代工学社 (1987)

（2）浜田　春夫 , "アダプティブフィルタの基礎"
日本音響学会誌 , Vol.48 , No.7 (1992)

（3）浜田　春夫 , "適応ディジタルフィルタのアクティブコントロールへの応用"
日本機械学会　講習会教材 , No.920-106 (1992)

・本マニュアルの内容は製品の改良のため予告無しに
変更される事がありますので、ご了承下さい。

中部電機株式会社

〒440-0004　愛知県豊橋市忠興 3 丁目 2-8

TEL <0532>61-9566

FAX <0532>63-1081

URL ： http://www.chubu-el.co.jp

E-mail ：csg@chubu-el.co.jp

ADSP324-334

適応ﾃﾞｼﾞﾀﾙﾌｨﾙﾀﾗｲﾌﾞﾗﾘ(67)

ｿﾌﾄｳｪｱ・ﾕｰｻﾞｰｽﾞ・ﾏﾆｭｱﾙ

2004.07　　第 2 版発行