AN4515

デジタル濾波でのアナログ感知器データ処理

序説

著者: Bjørnar Moe Remmen, Lloyd Clark, Phillip Olk, Microchip Technology Inc.

この応用記述はAVR® EA系MCUでの4つの異なる濾波算法を実装します。濾波は原作のｺｰﾄﾞまたはﾗｲﾌﾞﾗﾘのどちらかに基づきます。ｺｰﾄﾞ例は公に入手可能です。

ｺｰﾄﾞ例では静的な見本ﾃﾞｰﾀの組が濾波器に供給され、性能が分析されます。実際の濾波器の速度は全体の処理速度がﾃﾞｰﾀ採取の持続時間によって制限される時に2番目に重要となり得ます。一方で、効率的な濾波はCPU周期数、従って消費電力を減らします。

中央値濾波器

中央値濾波器は比較に基づき、乗算や除算のような集中的なCPU動作を必要としません。奇数採取の中央値が一連の比較によって決められます。

中央値濾波器はｶﾞｳｽ性、集中性、衝動性の雑音のような信号内の尖頭を取り去るのに秀でています。

ﾌｰﾘｴ変換

高速ﾌｰﾘｴ(Fourier)変換(FFT)は信号を各成分が周波数、振幅、位相を持つ周波数成分に分解する方法です。

FFTはかなり集中的なCPU負荷を生じますが、入力信号についての大きな一連の情報も生成します。単一FFT動作は最高点の周波数と振幅を識別することができ、従って高調波と側波帯を分離/除去するのに役立ち、ﾃﾞｼﾞﾀﾙ等価器の核心算法で有り得ます。

無限ｲﾝﾊﾟﾙｽ応答濾波器

無限ｲﾝﾊﾟﾙｽ応答(IIR:Infinite Impulse Response)濾波器は帰還に基づく、即ち、前の出力が現在の出力での役割を演じる濾波器の種類です。実装は簡単ですが、多用途で、高域と低域の通過濾波器、ﾉｯﾁと帯域通過濾波器、それと利得さえもかなり小さな係数の組で実装することができます。

帰還の原理のため、これらの濾波器は位相情報を失い、不安定になるかもしれず、にも関わらず、それらは感知器読み取り値の濾波に良好です。

ｶﾙﾏﾝ濾波器

ｶﾙﾏﾝ(Kalman)濾波器は加重平均に基づきます。重み係数は不確実性計算によって決められ、より高い確実性で重みが重くなります。加重平均から、後続する測定の予測が作成されます。その予測の”正確さ”、即ち、次の測定値が予測からどれだけ離れているかが、その測定の重みに使われます。

この濾波器は連続的で無段階の動きを持つ系で雑音の多い感知器ﾃﾞｰﾀに良く適します。

概要

ﾃﾞｼﾞﾀﾙ濾波器でのｱﾅﾛｸﾞ感知器ﾃﾞｰﾀ処理はﾏｲｸﾛ ｺﾝﾄﾛｰﾗ部(MCU)にとって資源集中型の仕事で有り得ます。この応用記述は複数のﾃﾞｼﾞﾀﾙ濾波器算法、それらの特性比較を提示し、応用特有の濾波に対する良好な開始点であるｺｰﾄﾞ例を提供します。

ﾃﾞｼﾞﾀﾙ信号処理では濾波器算法の膨大な選択がﾗｲﾌﾞﾗﾘで直ぐに利用可能か、またはCｺｰﾄﾞで容易に実装することができます。相対的に制限された8ﾋﾞｯﾄMCUの資源のため全てが適合する訳ではなく、ﾒﾓﾘ量の制限、測定、消費電力が応用設計者に妥協を強制します。この応用記述ではAVR® MCUに実装して使うのが容易ないくつかの濾波算法に対して直ぐに利用可能なﾗｲﾌﾞﾗﾘを提示します。それらの濾波器は次のとおりです。

- 中央値濾波器
- kissFFTﾗｲﾌﾞﾗﾘを使う高速ﾌｰﾘｴ変換(FFT)
- 双2次算法を使う無限ｲﾝﾊﾟﾙｽ応答(IIR)
- kalman-clibﾗｲﾌﾞﾗﾘを使うｶﾙﾏﾝ濾波器

以降の章では濾波器の短い紹介、それらの有用性と応用での注釈、それらのCPU負荷の定量化を与え、見本のｺｰﾄﾞを提供します。

目次

1. 関連ﾃﾞﾊﾞｲｽ

GitHubのｺｰﾄﾞ例はAVR EAﾃﾞﾊﾞｲｽ用に作成されています。算法を扱うｺｰﾄﾞ例の部分はこの文書で報告する一般的な原理と性能と同様に一般的に全てのAVRﾃﾞﾊﾞｲｽに対して有効です。ﾒﾓﾘ量のための制限が適用されます。ﾃﾞﾊﾞｲｽとﾃﾞｰﾀ可視器(Data Visualizer)(即ち、USART周辺機能とﾋﾟﾝ)間の通信を構成設定して実行するためのｺｰﾄﾞ部分は改造が必要とされるかもしれません。

2. 中央値濾波器

原理

中央値濾波器は尖頭(ｽﾊﾟｲｸ)を取り除くことによって信号を滑らかにします。中央値濾波器は画像と1次元信号の問題の両方に使うことができ、ここで網羅されます。中央値は数値の一覧を整列して中央値の数値を見つけることによって計算されます。

中央値濾波器は窓の大きさを持ち、どこまで遡るかを決めます。例えば、9の窓の大きさは遡る8つの要素と1つの新しい要素を持ちます。この濾波器はこれら9つの要素の中央値を返します。

例2-1. 配列の中央値の発見

配列 [31, 15, 8 ,91 ,31 ,90 ,1 ,5]を考察してください。整列後、中央の要素(ここでは5番目の位置)がこの配列の中央です。

[1 ,5, 8 ,15, 31, 31 ,90 ,91]

継続して測定を行っている時は配列内の最も古い要素が新しい値で置き換えられ、整列後、新しい中央値を決めることができます。

選んだ実装はgithub.com/accabog/MedianFilterで見つけることができ、これはAlexandru Bogdanによって書かれました。

例

本項では実装とｺｰﾄﾞ例を検討します。

main.cﾌｧｲﾙで、このｺｰﾄﾞは元の値と濾波した値の2つの整数値を持ちます。変数iはfilter/sine.hﾌｧｲﾙでの正弦波を渡って繰り返します。

```
volatile uint16_t original = 0;
volatile uint16_t filtered = 0;
uint8_t i = 0;
```

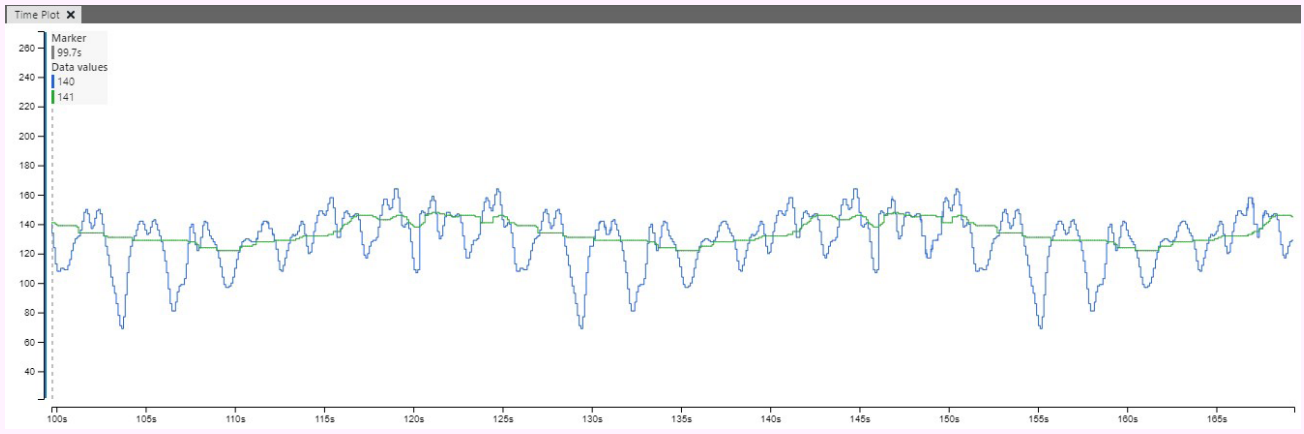
主ｺｰﾄﾞは最初にﾎﾟｰﾄDの6(PD6)ﾋﾟﾝを出力として初期化します。while繰り返しが中央値濾波器への入力として供給される新しい整数値を生成します。PD6出力が設定され、その後に中央値濾波器が呼び出されます。中央値濾波器直後にPD6出力が解除されます。従って、ｵｼﾛｽｺｰﾌﾟまたはﾛｼﾞｯｸ ｱﾅﾗｲｻﾞをPD6に接続することができます。Highﾚﾍﾞﾙ ﾊﾟﾙｽの持続時間を測定することにより、MEDIANFILTER_Insert()関数の実行時間を確定することができます。

```
original = sinewave[i++];
_delay_ms(100);
// 中央値濾波器の持続時間測定のためにﾋﾟﾝを使用
PORTD.OUTSET = PIN6_bm;                          // PD6ﾋﾟﾝを論理Lowにする。
filtered = MEDIANFILTER_Insert(&medianFilter, original);
PORTD.OUTCLR = PIN6_bm;                          // PD6ﾋﾟﾝを論理Highにする。
```

例2-2. 中央値濾波器の可視化

USART0上に元の信号と濾波した信号の両方を送るｺｰﾄﾞの塊が追加されました。この信号はﾃﾞｰﾀ可視器(Data Visualizer)を使って見ることができます。ここでの可視化は窓幅=31を使って行われます。

```
// USART上にﾃﾞｰﾀを送出
variableWrite_SendFrame(original, filtered);
```



緑は濾波した信号、青は元の信号です。2つの信号を比べるためにMPLAB®ﾃﾞｰﾀ可視器(Data Visualizer)を使って万能同期非同期送受信器(UASRT)上に送り、この2つの間には明らかな違いがあります。中央値濾波器は信号内の尖頭を取り去り、現在の信号と前の信号の中央値になります。この場合、濾波した信号はより奇麗な正弦波になります。

ﾃﾞｰﾀ可視器を使うにはLoad Workspace(作業空間読み込み)をｸﾘｯｸしてdata_visualizer.dvwsを選んでください。

性能と特性

中央値濾波器は選択可能な窓の大きさを持ちます。この窓の大きさは濾波の能力と処理にどれ位かかるかの両方に影響を及ぼします。

ﾎﾟｰﾄDの6番(PD6)ﾋﾟﾝで時間/周期数を測定するため、このﾋﾟﾝをﾛｼﾞｯｸ ｱﾅﾗｲｻﾞに接続し、ﾃﾞﾊﾞｲｽの速度を知ることによって濾波器の時間を計ることが可能です。

表2-1. 周期時間 – 中央値濾波器

濾波器の大きさ	7	15	31
周期数	215～385.8	225.8～601.8	344.8～1034

実装の形式のため、周期数は採取(試料)から採取(試料)へで変わります。この理由は試料挿入時の並び変え(整列)作業量です。それはif文が行う更新量とfor繰り返しを早めに抜け出せるかに依存します。数値が配列内の値の1つよりも小さい場合に繰り返しを抜け出ます。最悪の場合、値の一覧全体を通して繰り返すことが必要で、これが窓の大きさです。

iの値は必要とされる時に中央値を調整するために後で使われます。この算法は中央値が2つの数値の平均になる偶数での異常な状況を考慮しません。

```
for(i = 0; i < medianFilter->numNodes - 1; i++)
{
    if(sample < it->value)
    {
        if(i == 0)
        {   // 新しい節点が最小の場合に値の頭を置き換え
            medianFilter->valueHead = newNode;
        }
        break;
    }
    it = it->nextValue;
}
```

結びと使用事例

中央値濾波器の動き方と使い方を簡単に示しました。ｵｼﾛｽｺｰﾌﾟまたはﾛｼﾞｯｸ ｱﾅﾗｲｻﾞを使って応用に対する周期数を測定する方法も実演しました。

中央値濾波器は信号から雑音や尖頭を取り除くための良い方法で度々ｶﾙﾏﾝ濾波器のようなもっと高度な濾波器の前で前処理段として使うことができます。中央値濾波器ともっと高度な濾波器との違いは中央値濾波器が平均値濾波器のように信号内に極端な値を収めないことで、従って信号でのそれらの強い影響を取り去ります。

3. 無限ｲﾝﾊﾟﾙｽ応答(IIR)濾波器

原理

無限ｲﾝﾊﾟﾙｽ応答(IIR)濾波器は帰還に基づく濾波器、即ち、前の出力が現在の出力での役割を演じます。帰還の原理のため、これらの濾波器は位相情報を失い、不安定かもしれず、故にこれらは音響応用に対して常に最初の選択ではありません。しかし感知器測定に対してこれらは優れた道具の可能性があります。

この応用記述に関して、使い易さを重点に、簡単で柔軟なIIRの実装が選ばれました。

IIR濾波器は次のように微分方程式によって記述することができます。

$$y[n] = \sum_{k=0}^{M-1} b_k x[n-k] - \sum_{k=1}^{N-1} a_k y[n-k]$$

この実装では3次IIR濾波器の例を持ちます。

初期化段階ではb_{0-2}とa_{0-2}が濾波器の形式、例えば、低域通過濾波器に基づいて設定されます。濾波器走行時、x_{n-k}とy_{n-k}の値は現在の信号によって継続的に更新される一方で、a_kとb_kは一定です。

注: ｺｰﾄﾞの表記法は一般的な式で使われる表記法と異なります。

表3-1. ｺｰﾄﾞに対する変数説明

ｺｰﾄﾞでの変数	a_0	a_1	a_2	a_3	a_4
式での変数	$\frac{b_0}{a_0}$	$\frac{b_1}{a_0}$	$\frac{b_2}{a_0}$	$\frac{a_1}{a_0}$	$\frac{a_2}{a_0}$

```
/* 採取で双2次濾波器を計算 */
smp_type BiQuad(const smp_type sample, biquad* const b)
{
    smp_type result;

    /* 結果計算 */
    result = b->a0 * sample + b->a1 * b->x1 + b->a2 * b->x2 - b->a3 * b->y1 - b->a4 * b->y2;

    /* x1をx2へ移動、x1に採取 */
    b->x2 = b->x1;
    b->x1 = sample;

    /* y1をy2へ移動、1y1に結果 */
    b->y2 = b->y1;
    b->y1 = result;

    return result;
}                    // ｵｸﾀｰﾌﾞでの帯域幅
```

正規化係数a_0使用時、式は次のようになります。

$$y[3] = \sum_{k=0}^{3-1} b_k x[n-k] - \sum_{k=1}^{3-1} a_k y[n-k] \Rightarrow \left(\frac{b_0}{a_0} \times signal + \frac{b_1}{a_0} \times x_1 + \frac{b_2}{a_0} \times x_2\right) - \left(\frac{a_1}{a_0} \times y_1 + \frac{a_2}{a_0} \times y_2\right)$$

更なる測定についてはｶﾘﾌｫﾙﾆｱ大学ｻﾝﾃﾞｨｴｺﾞ校の音楽学部でのIIR濾波部分(musicweb.ucsd.edu/%7Etrsmyth/filters/Biquad_Section.html)を参照してください。

例え我々が使ったIIR濾波器実装が複数の濾波器に対する支援を持つとは言え、低域通過(Low-Pass)と帯域通過(Band-Pass)だけを網羅します。支援される濾波器の完全な一覧は次のとおりです。

- 低域通過濾波器
- 高域通過濾波器
- 帯域通過濾波器
- ﾉｯﾁ濾波器
- ﾋﾟｰｷﾝｸﾞ帯域EQ濾波器
- 低域ｼｪﾙﾌ濾波器
- 高域ｼｪﾙﾌ濾波器

例

本項では例ｺｰﾄﾞでの実装を調べます。

注: この実装は浮動小数点演算を使い、これは整数での実装に比べて濾波器の速度を低下させ得ます。

main.cで濾波器を説明するためにｺｰﾄﾞに直接書かれた信号を作成します。

```
const uint8_t  sinewave[] = {147, ..., 135};
```

この信号は濾波器へ渡します。濾波した結果を扱うため、ﾗｲﾌﾞﾗﾘはbq構造体を作成し、これは採取速度、周波数、帯域、利得を設定することによって作られます(利得は低域ｼｪﾙﾌと高域ｼｪﾙﾌの濾波器によってのみ使われます)。

```
biquad *bq = BiQuad_new(FILTER_TYPE, dbGain,    /* 濾波器の利得 */
                        freq,                   /* 中心周波数 */
                        srate,                  /* 採取速度 */
                        bandwidth);             /* ｵｸﾀｰﾌﾞでの帯域幅 */
```

濾波器のため、初期化したbq構造体で1つの値を双2次へ送ってください。

注: この応用記述の更に下でﾃﾞｰﾀ型切り替えが可能なことを述べます。入力値はそのﾃﾞｰﾀ型へ暗黙で収め(ｷｬｽﾄし)ます。

```
filtered = BiQuad(sinevalue, bq);
```

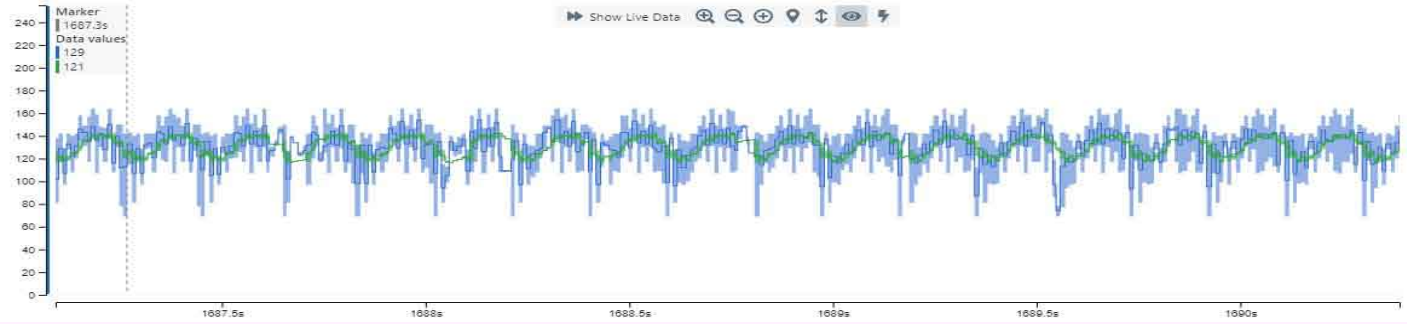
例3-1. 例: 低域通過濾波器

低域通過濾波器は与えられた閾値freqよりも低い信号を通し、その範囲の外側の信号を減衰/排除します。度々、上/下の区別を持つ濾波器について、用語の「帯域(bandwidth)」は(0～bandwidthとしての低域通過濾波器とbandwidth～無限大としての高域通過濾波器と見做す)周波数閾値を示します。ここで、bandwidth変数は上/下領域間の遷移の形を決めます。srate変数は採取(試料)間の位相Ωを決めるのに役立ちます。

```
smp_type freq = 8.0;                    // Hzでの周波数
smp_type srate = 255.0;                 // 秒毎の採取(試料)数
smp_type bandwidth = 5;                 // ｵｸﾀｰﾌﾞでの帯域
```

ﾃﾞｰﾀ可視器(Data Visualizer)を使うにはLoad Workspace(作業空間読み込み)をｸﾘｯｸしてdata-visualizer.jsonを選んでください。

図3-1. 入力信号と低域通過濾波器出力



緑は濾波した信号、青は元の信号です。2つの信号を比べるためにMPLAB®ﾃﾞｰﾀ可視器(Data Visualizer)を使って万能同期非同期送受信器(UASRT)上に送り、低域通過信号が除去された高域周波数成分を持つことが明らかです。

例3-2. 例: 帯域通過濾波器

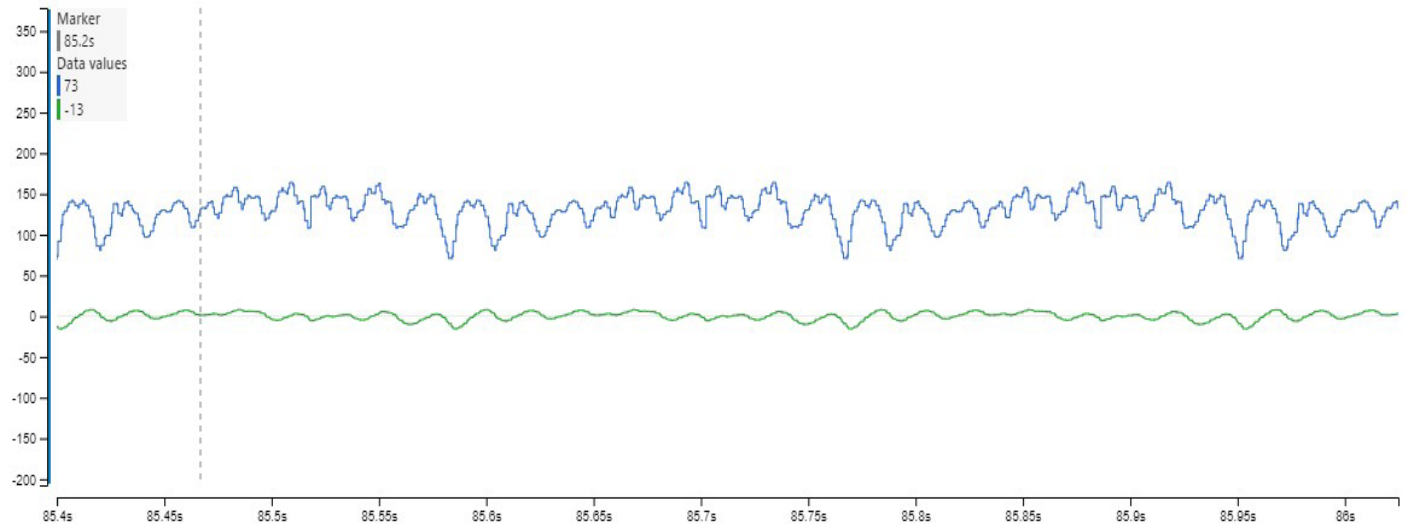
帯域通過濾波器は与えられた周波数範囲の外側の信号を排除または減衰し、それらの内側を容認します。

理想的には、帯域通過濾波器は範囲の内側の信号を減数または増幅せずに、周波数範囲の外側の信号を完全に取り去ります。帯域通過濾波器の中心周波数はbandwidthによって決められます。

```
smp_type freq = 3.0;                    // Hzでの周波数
smp_type srate = 255.0;                 // 秒毎の採取(試料)数
smp_type bandwidth = 1;                 // ｵｸﾀｰﾌﾞでの帯域
```

注: ｵｸﾀｰﾌﾞでの帯域の計算はwww.sengpielaudio.com/calculator-bandwidth.htmで見つけることができます。

図3-2. 帯域通過濾波器前後の低域通過濾波器出力



2つの信号を比べるためにMPLAB®ﾃﾞｰﾀ可視器(Data Visualizer)を使って万能同期非同期送受信器(UASRT)上に送り、濾波器の効果を見ることができます。帯域通過濾波器はIIR濾波器に対するﾊﾟﾗﾒｰﾀを使って設定した周波数の外側の雑音を取り去り、その結果が緑の濾波した信号です。ﾃﾞｰﾀ可視器(Data Visualizer)を使うにはLoad Workspace(作業空間読み込み)をｸﾘｯｸしてdata-visualizer.jsonを選んでください。

入力信号(青)のDC変位(ｵﾌｾｯﾄ)が0Hzの周波数と等しいため、信号は0を中心にされます。これは濾波器の通過帯域の下で、従って取り除かれ、帯域通過濾波器出力(緑)はDC変位を持たず、0を中心にされることを意味します。

性能と特性

IIR濾波器はこの濾波器に対してdoubleまたはfloatのどちらも使うことができますが、これは濾波器の速さに少々影響します。

時間/周期数を測定するため、main.cでONLY_SEND_USARTを0に設定してください。これはﾎﾟｰﾄBの2番(PB2)ﾋﾟﾝで刻時間の時間測定を許します。このﾋﾟﾝをﾛｼﾞｯｸ ｱﾅﾗｲｻﾞに接続し、ﾃﾞﾊﾞｲｽの速度を知ることによって濾波器の時間を計ることが可能です。

コンパイラとデバイス基本構造のいくつかの組み合わせはdoubleの大きさの変更、例えば、32ビットから64ビットへを支援します。増加は精度を改善しますが、より遅い走行時間の犠牲でです。整数はこの実装で動きません。

濾波器実装でデータ型を変更するにはbiquad.hでtypedef float smp_type;型定義を使ってください。コンパイラが-fno-short-doubleのような追加の引数を必要とするかもしれません。詳細についてはコンパイラの資料をご覧ください。

表3-2. 周期時間 – IIR濾波器

データ型	float
周期数	1280

結びと使用事例

AVRデバイスで双2次IIR濾波器を使う方法を実演しました。加えて、コードで濾波器の時間を計る方法も紹介しました。

ここで適用したIIRはいくつかの濾波任意選択を提供します。CPU負荷との組み合わせで、IIRはデジタル信号処理、特に8ビット マイクロコントローラでの有益な道具です。応用に応じて、IIRは雑音や変位変動の除去に使うことができ、これは望む信号成分の分離に役立ちます。CPU負荷がかなり低いおかげで、IIRは負荷やタイミングに敏感な応用に容易に適合することができます。

4. 高速フーリエ変換(FFT)

原理

高速フーリエ変換(FFT)は信号分析のための多用途の道具です。電子信号に関する一般的な概念は与えられた(時間領域での)信号を(周波数領域での)正弦形状成分に分解することです。各成分は周波数、位相、振幅を持ちます。逆演算は全ての成分を加算し、理論限界内で元の入力信号と同一の信号を返します。成分のパラメータを提供することにより、FFTは濾波の道具として機能することができます。雑音の多い環境で周期的な信号を分離し、振幅と周波数を定量化し、信号の周波数移動を検出し、ディスコの照明を制御するために高周波数帯域と低周波数帯域の振幅を区別したりすることができます。

FFTを実装してAVRデバイスで走行することができるいくつかのライブラリが利用可能です。ここでは、簡素さと簡潔さに注目するため、kissFFTが選ばれました。これはAVRプロジェクトへの統合が容易で、Cで実装され、便利な(3条項BSD)許認可様式を持ちます。一方で、(流れの中からデータの塊を選ぶ時の暗黙の長方形窓を別にして)窓関数と尖頭位置特定のような分析機能はなく、これは応用次第です。ライブラリに別の基準を選んだり、独自のFFT機能を実装することさえもできますが、この文書の一般的な結論はそこでも有効です。

例

本項では実装と作成したコード例を調べます。

このコードはFFTを走らせて実行速度を測定する方法を示します。簡単さのため、FFT算法に対して次のように直接書き込んだ入力信号を使います。

```
const int16_t sinewave[1024] = { ... };
```

信号は「無限インパルス応答(IIR)」章で使ったのと同じ周期信号です。以下のようなこれらの特性を持ちます。

- 基本周波数(f_0) = 440Hz
- 採取速度(f_S) = 44.1kHz
- DC変位(オフセット)
- 強い倍音 (故意に加えられた"ファズ(Fuzz)"効果)

nfftパラメータはkissFFTで採取長、即ち、FFT動作に対してどれ位のデータ点数が使われるかを選びます。動作の出力での周波数ビン(bin:入れ物)数(nfft/2)、従って、各ビンの幅(f_S/nfft)も決めます。その結果として、nfftは濾波器の速度に影響を及ぼします。

この例では次のように選びます。

```
int nfft = 800;
```

従って、次のとおりです。

- ビン数 = nfft/2 = 400
- ビン幅 = f_S/nfft ≒ 110Hz

結果のビン幅は既知の基本周波数、例の440Hzで上手く動きますが、速度最適化のためには、nfftに対して2のべき乗(例えば、64や256)の値を使います。

(一連のADC変換が提供するのと同様に)時間領域の実際の信号に対して、kissFFTライブラリは次のようなこの関数を推奨します。

```
kiss_fftr(cfg, cpx_in , cpx_out);                    // 実際のFFT操作
```

ここで、cfg = kiss_fftr_alloc(nfft, 0, 0, 0)は構成パラメータの組を必要とし、cpx_inは(実数)入力データでcpx_outは複素数出力データです。

注: kissFFTライブラリは複素数入力を持つかもしれない他の関数を提供します。この理由のため、入力変数名は接頭辞cpxを持ちます。それにも関わらず、入力信号は一連のADC採取値で、従って、実数だけを含みます。

今や応用は進行し、その目的のためにcpx_out[n]内の特定情報を評価することができます。我々の場合、パワー スペクトルpwrを計算し、USART周辺機能を使ってPCに送り、データ可視器(Data Visualizer)を使ってPC上にそれを作図します。

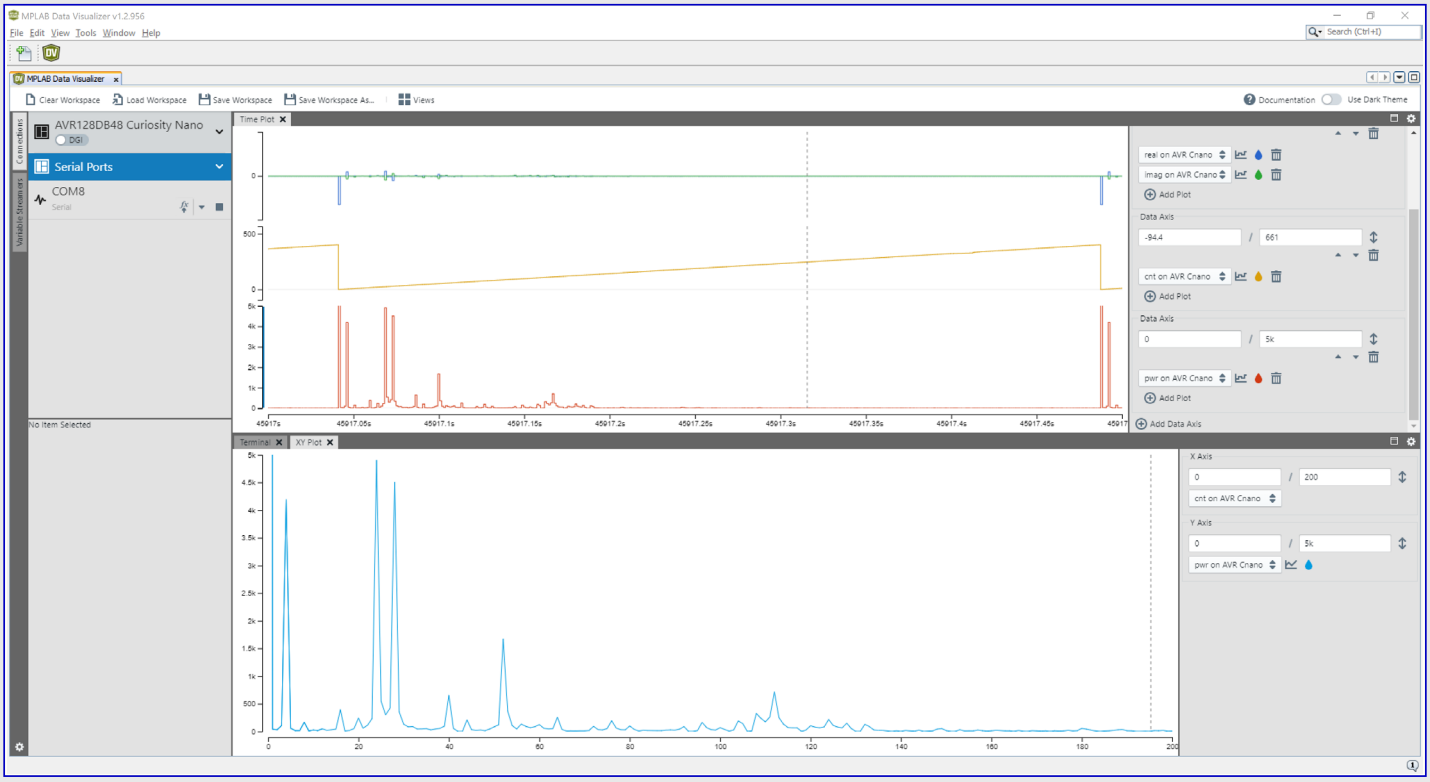
最初に、次のようにpwrを計算してください。

```
// パワー スペクトルを計算
pwr = sqrt(cpx_out[n].r * cpx_out[n].r + cpx_out[n].i * cpx_out[n].i);
```

各周波数ビン*n*に対して、複素結果ベクトルcpx_out[n]の絶対値は各々、それの実数と虚数の成分、cpx_out[n].rとcpx_out[n].iから計算されます。

デバイスがPCに接続されると、信号分析にMPLAB®データ可視器(Data Visualizer)を使うことができます。以下の図は前述の入力信号のFFTを示します。上側の図はUASRTによって送られた変数値を表示します。cpx_out[n]の実数と虚数の成分は1つの目盛りで共用します。次は黄色の反復計数器、次に赤色のパワー スペクトルです。

図4-1. 低振幅を持つ正弦波のFFT



“USART送信時間”(赤色)上のパワー スペクトルの表示は正確な尖頭位置を正確に示すことが難しいため、殆ど役に立ちません。MPLABデータ可視器のXY作図機能は、反復計数器をX値として使って周波数ビン番号に渡るパワー スペクトルを作図することができ、上図の下側の図(青)をご覧ください。1つのビンの幅が概ね110Hzなことを念頭に置くと、尖頭の位置と振幅を以下のように識別して定量化することができます。

- ビン0はkissFFTライブラリによって信号のDC変位(オフセット)を含むように定義されます(可読性のために切り取り)。
- ビン4は基本波、約440Hzを含みます。
- 自然高調波は次のように識別されます。
  - ビン8での第2高調波
  - (ビン12での第3高調波成分はありません。)
  - ビン16での第4高調波
  - ビン20での第5高調波
- 元の信号の”ぼやけた”見た目はビン24とビン28の支配的な成分によって引き起こされ、各々、約2.65kHzと3.09kHzの第6と第7の高調波を表します。
- 更に高い高調波や高周波雑音成分も識別でき、例えばビン40での第10高調波などです。

性能と特性

他の濾波技法と比べて、FFTはnfft(と応用の要件)に応じてCPU負荷にとって楽ではなく、装置のCPUは相当な時間の間多忙になり得ます。

実際の装置がアクセス可能の時の1つのFFTの持続時間はオシロスコープやロジック アナライザでデジタル出力ピンの出力水準を観測することによって測定することができます。例えば、Curiosity Nano基板のLEDを交互切り替えすることができます。

```
PORTD.OUTSET = PIN6_bm;                    // PD6出力を論理Highにします。
kiss_fftr(cfg, cpx_in , cpx_out);          // 実際のFFT動作
PORTD.OUTCLR = PIN6_bm;                    // PD6出力を論理Lowにします。
```

測定した持続時間は性能比較やｸﾛｯｸ周波数での尺度調整のために”MHz当たりのFFT数”に変換することができます。この方法はﾀｲﾐﾝｸﾞとCPU負荷が重要な応用に対して推奨されます。

装置や必要とする道具がｱｸｾｽ不可の時に、模倣(ｼﾐｭﾚｰﾄ)したAVRﾃﾞﾊﾞｲｽで算法の速度を測定するのに次のようにMPLAB Xを使うことができます。

1. ﾌﾟﾛｼﾞｪｸﾄ特性で接続した道具として”Simulator(ｼﾐｭﾚｰﾀ)”を選んでください。
2. kiss_fftr(～);関数呼び出しの直前と直後に適切な中断点(ﾌﾞﾚｰｸ ﾎﾟｲﾝﾄ)を追加してください。
3. ｽﾄｯﾌﾟｳｫｯﾁ ｳｨﾝﾄﾞｳを見つけてください(Window(ｳｨﾝﾄﾞｳ)⇒Debugging(ﾃﾞﾊﾞｯｸﾞ)⇒Stopwatch(ｽﾄｯﾌﾟｳｫｯﾁ))。
4. ﾃﾞﾊﾞｯｶﾞを動かしてください。
5. ﾃﾞﾊﾞｯｶﾞが中断点で停止すると、開始または直前の中断点から現在の中断点間に使った仮想ｸﾛｯｸ周期数を表示します。

この手法は原理的な性能上限の素早く容易な推定を提供します。

次表はCuriosity Nanoで行い測定した性能を一覧にします。

表4-1. 周期時間 – FFT濾波器

nfft	4	8	16	32	64	128	256	512
FFT当たりの周期数	5,300	12,000	30,100	67,000	161,000	348,000	810,000	1,750,000

例のようにnfft=64でのFFTはFFT当たり約161k周期を使い、MHz当たり6.2回FFTに相当します。その結果、20MHzのｺｱ周波数で走行するﾃﾞﾊﾞｲｽは20倍の回数、即ち、秒当たり124回FFTを実行することができます。ﾃﾞﾊﾞｲｽが32.768kHzの外部ｸﾘｽﾀﾙだけを使う場合、同じFFT動作は概ね5秒かかります。

注: 最大nfft値はﾃﾞﾊﾞｲｽの実際のﾒﾓﾘ構成と応用の残りのﾒﾓﾘ要件に依存します。

速度最適化のため、nfftに2のべき乗(例えば、64や256)の値を使ってください。

結びと使用事例

第三者の(kissFFT)ﾗｲﾌﾞﾗﾘを使ってAVRﾃﾞﾊﾞｲｽが高速ﾌｰﾘｴ変換を実行できることを示しました。FFTは他のﾃﾞｼﾞﾀﾙ濾波技法ができない価値ある情報を提供することができますが、CPU周期数の犠牲でです。これは応用の要件とFFTが継続的に走行するかどうかの詳細に依存します。代わりに、FFTは他のより速い濾波器用にﾊﾟﾗﾒｰﾀを最適化するために時々使われることがあります。

留意すべき別の点はﾊﾟﾜｰ ｽﾍﾟｸﾄﾙによって示唆される仮想精度と測定の実際の精度です。

1つの面は入力採取(試料)の品質と使った時間基準です。時間基準を提供するｵｼﾛｽｺｰﾌﾟは数%の温度変動を受け得ます。これは信号分析用自己完結応用、特に採取とFFT使用間が短時間の時にそれ程本題ではありません。けれども、それら自身の時間基準を持つ外部装置と相互作用する時に関係があります。

他の面は信号の採取速度とnfftによって決められる周波数ﾋﾞﾝの幅です。上の例では、ﾊﾟﾜｰ ｽﾍﾟｸﾄﾙが高調波を識別するのに上手く適合されていました。一方で第10高調波を利用する時さえ、基本成分が実際に440Hzに調整されたのか、音楽的に言えば離調されたのかのどちらかを判定することができません。

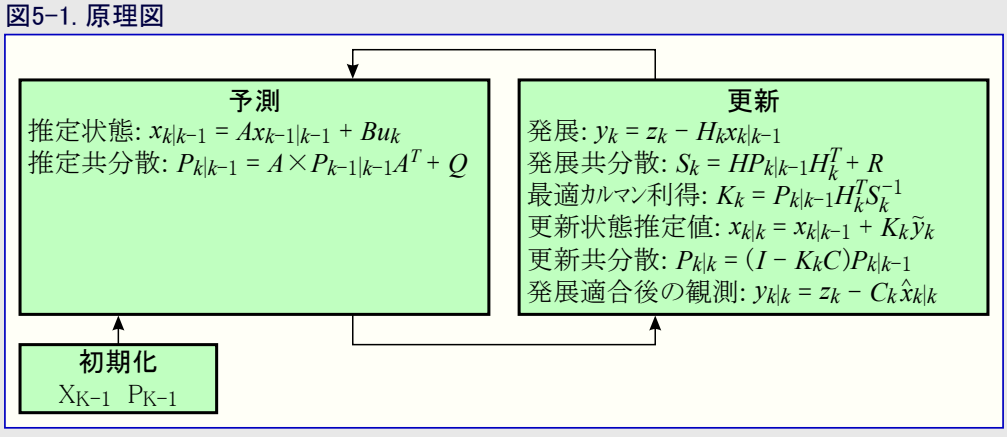
5. カルマン濾波器

原理

線形2次推定(LQE:linear quadratic estimation)とも呼ばれるｶﾙﾏﾝ濾波器は将来の未知の変数を推定するのに一連の測定を使う算法です。状態推定は、例えば、壁にぶつかるのを避けるようにﾛﾎﾞｯﾄ掃除機の配置を予測したり、平衡を保つﾛﾎﾞｯﾄを作成したりするのに使うことができます。

下図はｶﾙﾏﾝ濾波器の一般的な原理を示します。初期化後、算法走行時に予測と更新の段階が起きます。更新段階は予測での誤差に基づいて変数を更新します。

図5-1. 原理図



例

本項はGitHubでの例ｺｰﾄﾞを検討します。使った例は3×3の配列ですが、必要ならば他の例に変更することができます。

```
int main(void)
{
    SYSTEM_Initialize();

    PORTD.DIRSET = PIN6_bm;

    matrix_unittests();
    while(1)
    {

        if(SEND_OVER_USART)
        {
            // USART上にﾃﾞｰﾀを送るｺｰﾄﾞ
            kalman_gravity_demo_usart();
        }else{
            // ﾋﾟﾝでのより速い測定周回のためのUSARTなしでの通常のｺｰﾄﾞ
            kalman_gravity_demo();
        }

        kalman_gravity_demo_lambda();
    }
}
```

例5-1. カルマン重力実演

この実演は自由落下試験体の位置Sに基づいて重力gを計算し、Markus Mayerによって書かれています。ｿｰｽはhttps://github.com/sunsided/kalman-clibで見つけることができます。

このｺｰﾄﾞは3つの状態(位置、速度、重力)を使い、ここで重力は走行距離、速度、直前の推定加速度に基づく推定加速度です。ｺｰﾄﾞで言及されるように、この試みに対する式は次のとおりです。

$S = S + v \times T + g \times 0.5 \times T^2$

$v = v + g \times T$

$g = g$

で、

- Sはmでの位置
- vはm/sでの速度
- Tは時間、1秒単位で進行
- gは6 $\frac{m}{s^2}$の初期値での重力。これは"悪い推定"で、ｶﾙﾏﾝ濾波器が改善するでしょう。

Sの値は"雑音の多い"位置で、それらの間を1秒で記録されます。

ｺｰﾄﾞは初期値を準備し、試験体の相対加速度を予測するために距離を反復します。この手順は次のとおりです。

1. 車間距離を予測します。
2. 実際の車間距離を測定します。
3. 誤差を計算します。
4. Z配列を更新します。

Z配列は更にkalman_correct()ﾙｰﾁﾝでの使われ、これは変数を更新して修正し、故にこの手順を繰り返すことができます。

この手順を2、3回経験すると、計算したgの値は直ぐに最終値に近づきます。

ｶﾙﾏﾝ濾波器の速度を測定するため、ﾎﾟｰﾄDの6(PD6)ﾋﾟﾝにｵｼﾛｽｺｰﾌﾟかﾛｼﾞｯｸ ｱﾅﾗｲｻﾞを接続してこのﾋﾟﾝがHighの時の時間を測定してください。ｶﾙﾏﾝ濾波器の使用法のもっと綿密な説明はｺｰﾄﾞのREADME.mdです。

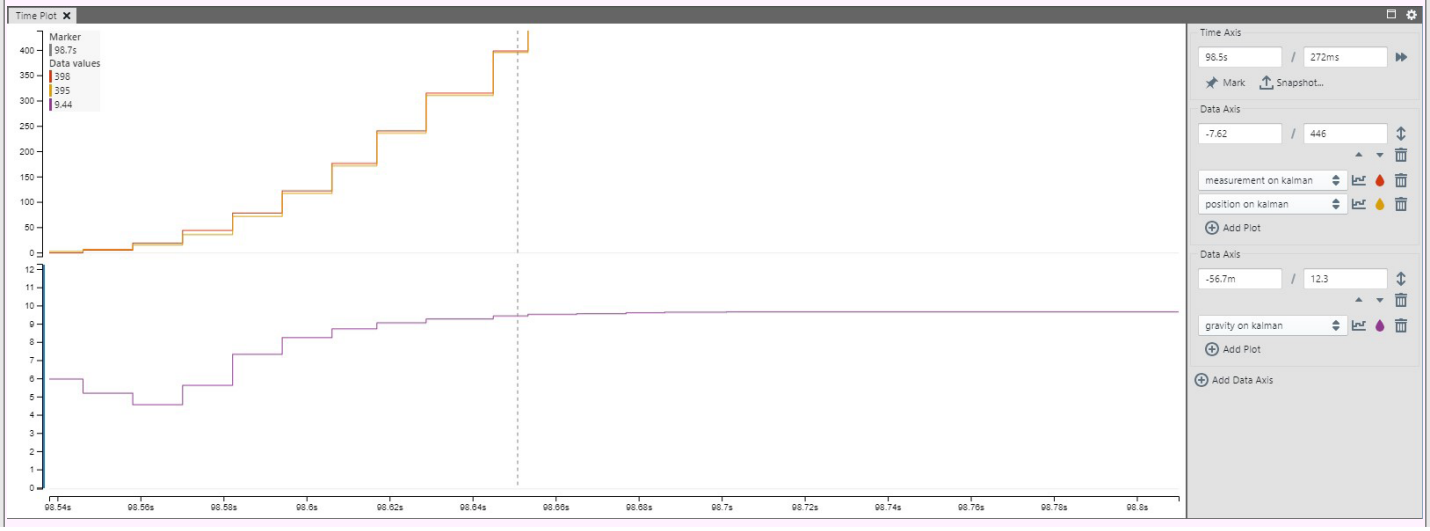
```
// USART上にﾃﾞｰﾀを送出
variableWrite_SendFrame(original, filtered);

void kalman_gravity_demo()
{
    PORTD.DIRSET = PIN6_bm;
```

```
    // 濾波器初期化
    PORTD.OUTSET = PIN6_bm;
    kalman_gravity_init();
    PORTD.OUTCLR = PIN6_bm;

    // 構造体取得
    kalman_t *kf = &kalman_filter_gravity;
    kalman_measurement_t *kfm = &kalman_filter_gravity_measurement_position;

    matrix_t *x = kalman_get_state_vector(kf);
    matrix_t *z = kalman_get_measurement_vector(kfm);

    // 濾波!
    for (int i = 0; i < MEAS_COUNT; ++i)
    {
        PORTD.OUTSET = PIN6_bm;
        // 予測
        kalman_predict(kf);
        // x[0] -> s_i -> 次の位置を推定
        // x[1] -> v_i -> 次の速度を推定
        // g[2] -> g_i -> 次の重力を推定
        // 測定 ...
        matrix_data_t measurement = real_distance[i] + measurement_error[i];
        // floatをuint8配列に変換するため位置を読み込み
        volatile uint8_t* measurement_b = (uint8_t*) &measurement;
        matrix_set(z, 0, 0, measurement);

        // 更新
        kalman_correct(kf, kfm);
        PORTD.OUTCLR = PIN6_bm;
        PORTD.DIRCLR = PIN6_bm;
    }

    // 推定したgを取得
    matrix_data_t g_estimated = x->data[2];
    assert(g_estimated > 9 && g_estimated < 10);
}
```

推定の継続的な改善を図解するためにperipherals/usart.hでSEND_OVER_USARTをTRUEに設定してください。これはUSART経由でデータを送るkalman_gravity_demo_usart()の使用を保証します。

注: 算法のもっと正確な周期測定を得るには周期測定を行っている時にSEND_OVER_USARTをFALSEに設定してください。

Load Workspace(作業空間読み込み)を選ぶことによってデータ可視器(Data Visualizer)を構成設定してください。構成設定ファイルのdata_visualizer.dvwsを選んでください。



黄色の線は位置の測定(即ち、感知器測定)で、赤色の線は次の測定に対する予測位置、紫色の線は計算した重力です。
1秒毎に1つの新しい測定が取られ、次の位置と重力gの推定のための帰還を与えます。重力gの作図は初期値6 $\frac{m}{s^2}$から始まり、9.81 $\frac{m}{s^2}$に向かう推定を改善します。

性能と特性

カルマン濾波器は高度な算法で、従って、この算法の長い走行時間は予想外ではありません。より大きな配列の大きさに対して時間消費が更に増します。より小さな配列の大きさはより速く走行します。

表5-1. 周期時間 – カルマン濾波器

配列の大きさ	初期化	1推定
A=3×3、B=null	1700	440

結びと使用事例

この応用記述はこの濾波器の使い方の概要を与え、濾波器の代表的な周期時間を提供し、そして位置測定からの加速を計算する例を一通り説明します。

カルマン濾波器は前の入力に基づいて次の状態を知ることが役立つ多くの状況で使うことができます。カルマン濾波器の使用事例は平衡を取るロボット、掃除機、さらに自動運転車のような移動性応用でも度々見られます。

6. 結び

この文書は紹介した4つのデジタル信号濾波法と実演したAVRデバイスでのそれらの使用を持ちます。各濾波法は利点と欠点の範囲を持ちます。

中央値濾波器 度々見落とされる中央値濾波器はかなり簡単ですが、効果的な”信号平滑化”技法です。尖頭や瞬間異常を上手く取り去り、少しのCPU周期数しか必要としません。欠点は結果が滑らかすぎて”丸く”なり得ることです。

IIR濾波器 無限インパルス応答濾波器は”古典的な”信号濾波からいくつかの機能、即ち、低域通過、高域通過、帯域通過の変種を提供します。多用途性と手頃なCPU周期消費はこれを魅力的な道具にします。

FFT 高速フーリエ変換それ自体は古典的な濾波器ではありませんが、入力信号についての多くの情報が明らかにします。従って、FFTはデジタル信号処理の主流です。8ビットMCUの視点からはFFTがメモリとCPU周期数に関して多くの資源を必要とし得ます。注意深く適用すると、入力信号の解釈での相当な助けを提供し得ます。

カルマン濾波器 どの古典的な濾波器でもないカルマン濾波器は制御器との類似点を持ちます。(出力)パラメータを決めるのに雑音の多い入力信号を使うことができます。CPU消費は中程度です。カルマン濾波器は実際の値が継続的に伝播することが予想される設定で優れています。

7. GitHubからのコード例取得

コード例は画像的使用者インターフェース(GUI:Graphical User Interface)を通して応用コードを提供するウェブに基づくサーバーであるGitHubを通して入手可能です。

GitHubウェブ頁: GitHub

コード例

GitHubでコード例を見てください。
貯蔵庫を閲覧するにはクリックしてください。

Clone(複製)またはDownload(ダウンロード)の釦をクリックすることによってGitHubの例頁から.zipファイルとしてコードをダウンロードしてください。

8. 改訂履歴

文書改訂	日付	注釈
A	2022年4月	初版文書公開

Microchip情報

Microchipウェブ サイト

Microchipはwww.microchip.com/で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にするのに使われます。利用可能な情報のいくつかは以下を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、最新ソフトウェア配布と保管されたソフトウェア
- **全般的な技術支援** – 良くある質問(FAQ)、技術支援要求、オンライン検討グループ、Microchip設計協力課程会員一覧
- **Microchipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、Microchip営業所の一覧、代理店と代表する工場

製品変更通知サービス

Microchipの製品変更通知サービスはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ツールに関連する変更、更新、改訂、障害情報がある場合に必ず電子メール通知を受け取ります。

登録するにはwww.microchip.com/pcnへ行って登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのチャネルを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 組み込み解決技術者(ESE:Embedded Solutions Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、またはESEに連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はwww.microchip.com/supportでのウェブ サイトを通して利用できます。

Microchipデバイス コード保護機能

Microchip製品での以下のコード保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- Microchipは動作仕様内で意図した方法と通常条件下で使われる時に、その製品系統が安全であると考えます。
- Microchipはその知的所有権を尊重し、積極的に保護します。Microchip製品のコード保護機能を侵害する試みは固く禁じられ、デジタル ミレニアム著作権法に違反するかもしれません。
- Microchipや他のどの半導体製造業者もそれのコードの安全を保証することはできません。コード保護は製品が”破ることができない”ことを当社が保証すると言うことを意味しません。コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。

法的通知

この刊行物と契約での情報は設計、試験、応用とのMicrochip製品の統合を含め、Microchip製品でだけ使えます。他の何れの方法でのこの情報の使用はこれらの条件に違反します。デバイス応用などに関する情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。追加支援については最寄りのMicrochip営業所にお問い合わせ頂くか、www.microchip.com/en-us/support/design-help/client-support-servicesで追加支援を得てください。

この情報はMicrochipによって「現状そのまま」で提供されます。Microchipは非侵害、商品性、特定目的に対する適合性の何れの黙示的保証やその条件、品質、性能に関する保証を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。

如何なる場合においても、Microchipは情報またはその使用に関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、情報またはその使用に関連する全ての請求に対するMicrochipの全責任は、もしあれば、情報のためにMicrochipへ直接支払った料金を超えないものとします。生命維持や安全応用でのMicrochipデバイスの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

品質管理ｼｽﾃﾑ

Microchipの品質管理ｼｽﾃﾑに関する情報についてはwww.microchip.com/qualityを訪ねてください。

日本語© *HERO* 2024.

本応用記述はMicrochipのAN4515応用記述(DS00004515A-2022年4月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **ｱﾄﾗﾝﾀ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **ｵｰｽﾁﾝ** TX Tel: 512-257-3370 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ﾛｰﾘｰ** NC Tel: 919-844-7510 **ﾆｭｰﾖｰｸ** NY Tel: 631-435-6000 **ｻﾝﾎｾ** CA Tel: 408-735-9110 Tel: 408-436-4270 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港特別行政区** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 廈門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ソウル** Tel: 82-2-554-7200 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ヴェルス** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4485-5910 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - エスポー** Tel: 358-9-4520-820 **フランス - パリ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ハーン** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ハイルブロン** Tel: 49-7131-72400 **ﾄﾞｲﾂ - カールスルーエ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ミュンヘン** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ローゼンハイム** Tel: 49-8031-354-560 **イスラエル - ラーナナ** Tel: 972-9-744-7705 **イタリア - ミラノ** Tel: 39-0331-742611 Fax: 39-0331-466781 **イタリア - パドバ** Tel: 39-049-7625286 **オランダ - デルーネン** Tel: 31-416-690399 Fax: 31-416-690340 **ノルウェー - トロンハイム** Tel: 47-72884388 **ポーランド - ワルシャワ** Tel: 48-22-3325737 **ルーマニア - ブカレスト** Tel: 40-21-407-87-50 **スペイン - マドリード** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **スウェーデン - イェーテボリ** Tel: 46-31-704-60-40 **スウェーデン - ストックホルム** Tel: 46-8-5090-4654 **イギリス - ウォーキンガム** Tel: 44-118-921-5800 Fax: 44-118-921-5820