応用での細動の理解と管理

AN5823

概要

この文書の目的はクロック細動(ジッタ)と制御繰り返しでPWM信号を利用する応用でのそれの影響の理解を提供することです。それが何処から来るかを理解する方法、それを測定する方法、細動を考慮して応用を設計する方法の情報も提供します。

目次

1. 序文

1.1. 細動て何?

細動(ｼﾞｯﾀ)は参照基準信号からのﾀｲﾐﾝｸﾞの偏差として定義されます。細動はﾀｲﾐﾝｸﾞや電気的な回路の閾値での小さな偏差によって作成されます。これらの偏差は電源の雑音と信号の品質を含む環境とｼｽﾃﾑ要素によって悪化させられ得ます。細動の種類、成分、数値化が以降の章で検討されます。ｼｽﾃﾑで雑音が増すと、細動に対応する量はいくつかの応用に対して有害になり得ます。

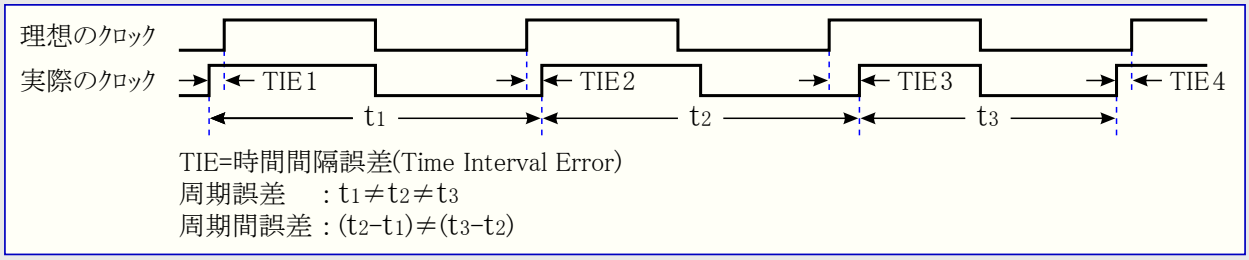
1.2. 期待値を設定

細動(ｼﾞｯﾀ)を0にすることは決してできませんが、緩和したり軽減したりできるのを理解することが重要です。いくつかの製品はｼｽﾃﾑ細動の量よりも小さいかもしれない遷移端分解能を提供します。これは応用の種類に応じて問題が存在すかもしれないし、しないかもしれません。平均または他の濾過を利用する制御繰り返しは細動が誘発した誤差に対して影響を受け難くすることができます。周期毎に制御されるｼｽﾃﾑは細動を許容することができますが、不均一な分布で脈動(ﾘｯﾌﾟﾙ)を招くかもしれません。分解能や細動が指定された試験条件のいくつかは理想環境かもしれませんが、現実の応用では電源のｽｲｯﾁﾝｸﾞから過渡応答が存在し、雑音値を上げてｼｽﾃﾑ細動になります。

2. 細動の理解

細動(ジッタ)は周期細動、周期間細動、長期細動のような様々な方法で観察して測定することができます。これらの方法は全て関連していて何の成分が存在していてそれらが応用にどう影響を及ぼし得るかの別な面の正確な理解を提供することができます。次の図はクロック元細動測定基準を示します。

図2-1. クロック元細動測定基準

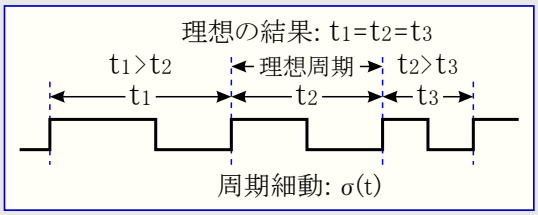


2.1. 周期細動

周期細動はどれかの単一クロック周期が期待したクロック値と一致しない時に発生します。細動がタイミング信号に影響を及ぼす時に周期がどちらかの方向に移動され得て、その周期が理想よりも大きいか小さいかのどちらになり得ることを意味します。細動を持つ周期は全ての単一周回を変えます。図2-2.でt1は期待したパルスよりも長くなっている一方で、t2は正確に望んだクロック周波数で、t3は期待したパルスよりも短くなっています。理想的な応用では3つ全てのパルスが同じ長さです。

右図はこの周期細動を示します。

図2-2. 周期細動



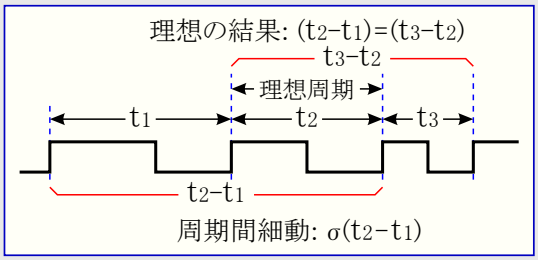
2.2. 周期間細動

周期間細動は隣接クロック周期間での周期差です。これは図2-3.で表されるように周期細動の異なる周期間の関係性で、次式によって記述されます。

式2-1.

周期間細動 = σ(t2-t1)

図2-3. 周期間細動

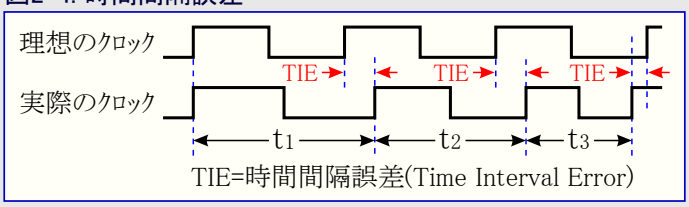


2.3. 時間間隔誤差 (TIE)

1つの時間測定から別のものまでの結果の差が時間間隔誤差(TIE:Time Interval Error)と呼ばれます。図2-4.で3つのクロック パルスのどれもが期待した時間で起こりません。これはt1の周期が期待した周波数よりも長く、波形全体を一定量の時間分移動させたからです。この時間誤差がTIEで、正または負の値を持ち得ます。例えば、t1からt2とt2からt3は両方共に正の値を持ちますが、t3は標準周期よりも短い(負の)TIEを持ち、総TIEは平均化されます。TIEはクロックに関連して違う時間尺を持つこともできます。遅いTIEは小さな誤差を持つ多くの周期として表すことができ、どちらの方向でも重要な誤差を累積するには長い時間がかかります。包絡線タイミングとTIEの大きさは独立していて様々な方法でシステム性能に影響を及ぼし得ます。無作為細動ではこれらの移動が未だ起こりますが、管理可能に留まります。

右図は時間間隔誤差を示します。

図2-4. 時間間隔誤差



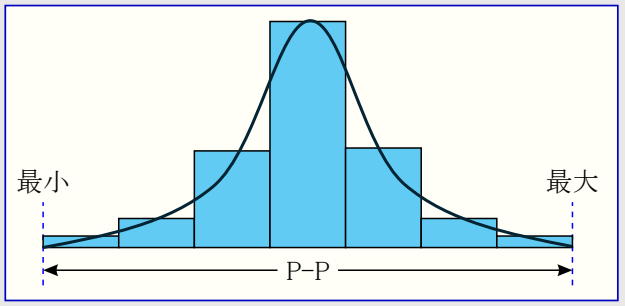
3. 細動の指標

ｸﾛｯｸ ﾀｲﾐﾝｸﾞの細動(ｼﾞｯﾀ)を測定するのにいくつかの指標があります。細動ﾃﾞｰﾀはxが理想ｸﾛｯｸ周期とのﾀｲﾐﾝｸﾞ差で、yは周期値が測定された回数の柱状図として度々表現されます。

3.1. 最小/最大

柱状図での最小と最大は最低周期測定となる最小と最高周期測定となる最大で、両側での2つの最も遠い点です。P-P(最大幅)測定は最小と最大の間の差です。これらの値は応用での全面的な分布を考慮する時に考慮されますが、P-P値は(無作為で無制約な細動のため)理論的に柱状図での計数として無限に増し、細動評価での主要な要素になるべきではありません。代わりに、柱状図の平均と標準偏差の評価が細動分析のもっと正確な形式です。右図は柱状図での最小と最大を示します。

図3-1. 最小/最大柱状図

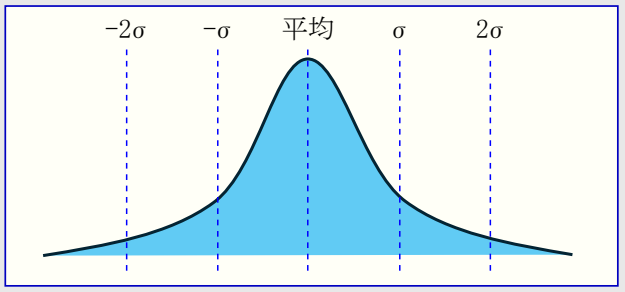


3.2. 標準偏差 (σ)

標準偏差はﾃﾞｰﾀ値の組での変動または分散の量を定量化する統計的な測定です。計算した標準偏差が低ければ、ﾃﾞｰﾀ点の大多数は平均に近いことを意味し、より高い標準偏差はより高い分散を意味します。ﾃﾞｰﾀが正規分布のように現れる標準的な分布では概ね68%のﾃﾞｰﾀが1つの標準偏差の境界内、または-1σ～1σの間になります。標準偏差はﾃﾞｰﾀの平均から導き出され、これら2つの値が細動を分析する時に最も重要な指標です。

右図は標準偏差と平均を示します。

図3-2. 標準偏差と平均



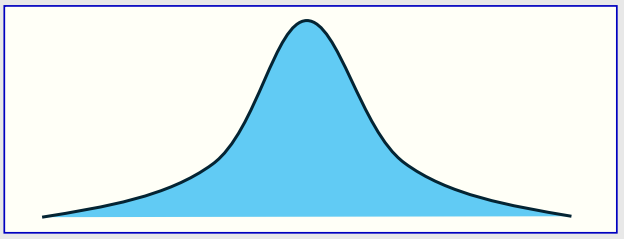
4. 細動の種類

4.1. 無作為細動

細動(ｼﾞｯﾀ)は無作為と確定的の2つの種別下になります。多くの場合、それは雑音の多い電源、貧弱なﾊｰﾄﾞｳｪｱ設計、熱雑音、漏話によって起こされます。細動は全てのｼｽﾃﾑで信号の周期での変動として現れます。無作為細動だけが存在する理想的な場合、ｸﾛｯｸ周期の柱状図は右図で示されるように、正規分布または釣鐘曲線として現れます。

正規分布測定結果はその細動が”無作為(ﾗﾝﾀﾞﾑ)”で、原因を確定または除去できないことを示します。無作為細動は束縛されず、無限に拡張した正規分布の尾を意味しますが、大きな偏差の可能性は指数関数的に減ります。

図4-1. 無作為細動



4.2. 確定的細動

無作為と確定的の両細動を持つｼｽﾃﾑでは曲線の頂点がどちらかの側に移動されるか、または複数の頂点さえあるかもしれません。

図4-2. 確定的細動 – 移動した頂点

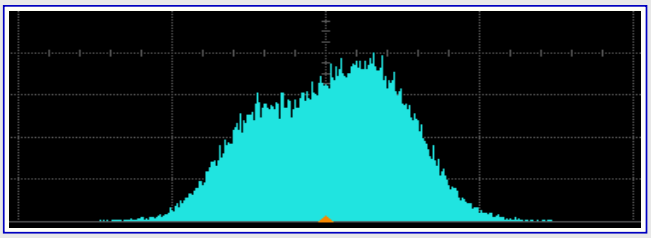
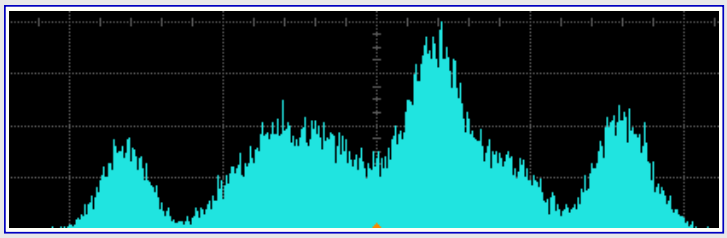
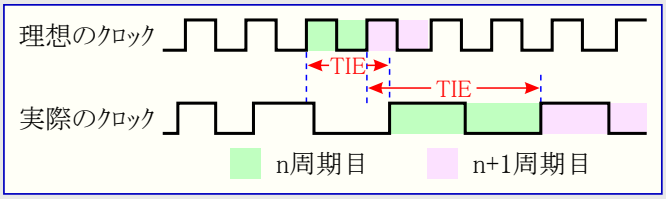


図4-3. 確定的細動 – 複数頂点



前の右図は確定的細動が応用でどう現れ得るかの代表です。単一の正規分布頂点の代わりに、いくつかの組み合わせのように見えます。これは周回時間に繰り返し影響を及ぼす応用での何かの確定的な動きを示します。図4-2.で曲線の頂点は柱状図の右側に移動され、平均ｸﾛｯｸ周期が期待したｸﾛｯｸ周期よりも長いことを意味します。これは時間と共に1方向で徐々に増加するｸﾛｯｸ周期の誤差の長期的な細動の例です。これらの場合、ｸﾛｯｸの周期が希望よりも一貫して大きいため、TIEはもっと大きくなり得ます。図4-4.は確定的細動が長期的にTIEにどう影響を及ぼすかを示します。2つの連続するｸﾛｯｸ ﾊﾟﾙｽが期待よりも大きいため、TIEが増すことに注目してください。

図4-4. 長期的な細動



5. 測定技法

測定技法と条件は結果に大きな影響を持ち得ます。使う機材は信号の速度と端速度に対して必要とする能力を持つことが必要です。不正確な振幅を最小にするため、ﾌﾟﾛｰﾌﾞの接地ﾘｰﾄﾞ長に注意すべきです。いくつかの機材は組み込みｸﾛｯｸまたは細動分析のﾂｰﾙやｿﾌﾄｳｪｱを持つかもしれません。ｼｽﾃﾑ細動を識別して定量化するためにいくつかの検査を実行することが必要かもしれません。供給源を特定するのを助けるために測定は回路の経路に沿って得ることができます。いくつかの代表的な検査点は次のとおりです。

- 外部からﾏｲｸﾛ ｺﾝﾄﾛｰﾗの場合、ｸﾛｯｸ元
- ﾋﾟﾝに持ち出すﾏｲｸﾛ ｺﾝﾄﾛｰﾗの内部ｸﾛｯｸ元
- ﾋﾟﾝに持ち出すPLL出力
- PWM出力ﾋﾟﾝ
- ｹﾞｰﾄ駆動部出力

ﾏｲｸﾛ ｺﾝﾄﾛｰﾗと他の回路がｼｽﾃﾑ雑音に追加することがあるとすれば、雑音源を分離して測定の前後で比べることが役立ち得ます。例えば、通信線やその他のPWM信号の停止は着目した信号でのそれらの影響を評価するのを助けることができます。

5.1. 測定手順

細動(ｼﾞｯﾀ)を測定には以下の手順を実行してください。

1. 電力と構成設定の要素を含む被試験体(DUT:Device Under Test)を設定してください。
2. 参照基準出力として指定されたﾋﾟﾝにｵｼﾛｽｺｰﾌﾟのﾌﾟﾛｰﾌﾞを接続してください。それが結果を隠し得るため、ｵｼﾛｽｺｰﾌﾟを平均化動作で使わないことが重要です。
3. 測定する細動の種類に対して必要とされるようにｵｼﾛｽｺｰﾌﾟを構成設定してください。隣接周期の記録は周期間細動だけでなく周期細動とTIEの計算も許します。
4. 最低10,000回手順3.を繰り返してください。大きな採取量はどの長期細動も識別されることを保証する手助けをします。
5. 検査結果から柱状図を作成して平均と標準偏差を計算してください。柱状図は各測定した周期の発生回数を示します。
6. このﾃﾞｰﾀを使い、どの確定的細動が特定され得るかと細動の水準が応用に対して受け入れ可能かを判断してください。
7. 柱状図で複数の頂点が表示されるように確定的細動が見られる場合、どの周波数が付加的な頂点を発生しているかを特定するためにｽﾍﾟｸﾄﾗﾑ分析器の使用を考慮してください。
8. 確定的細動を軽減するために以降の指示に従ってください。ｽﾍﾟｸﾄﾗﾑ分析器が使われ、望むﾀｲﾐﾝｸﾞ信号を超える付加的な周波数範囲が特定された場合、その問題を最も上手く対処する次の章の手順に注目してください。

6. 細動の軽減

様々な成分と細動(ｼﾞｯﾀ)の種類がｼｽﾃﾑで面倒な問題になり得るのが今や分かったので、ｼｽﾃﾑ内の確定的細動を減らす方法を検討する時です。無作為の成分を取り去ることはできません。以前の章で細動の原因が検討されました。細動を減らすのに使ういくつかの方法は次のとおりです。

- 供給元入力ｯｸﾛｯｸ
- PLL濾波
- 電源雑音除去
- PCBとｼｽﾃﾑの設計

6.1. ｸﾛｯｸ元の細動

外部発振器が必要でないとは言え、参照基準ｸﾛｯｸとして高速MEMS(微小電気機械ｼｽﾃﾑ)やｸﾘｽﾀﾙ発振器回路を使うことが内部供給元に比べてﾃﾞﾊﾞｲｽの細動を減らすことが分かっています。dsPIC® PLLで使う時により高い周波数の入力ｸﾛｯｸの使用がより良いｸﾛｯｸ精度を許します。これはVCO周波数補正が数ｸﾛｯｸ周期毎に起こるためで、既定の8MHz FRC速度よりも高い外部ｸﾛｯｸ周波数がVCO補正をより頻繁に起こさせることを意味します。応用に対して更なるｸﾛｯｸ精度が必要か、またはｼｽﾃﾑ内の細動の水準が受け入れ可能かを判断するためにｼｽﾃﾑを検査することが重要です。

6.2. 位相固定化閉路 (PLL)

PWM応用では必要とされる速度と分解能を達成するためにPLLが使われるのが一般的です。PLLは設計と濾波に応じて細動性能を改善または劣化のどちらかにさせ得ます。PLLの性能は帰還倍率とVCO周波数を含むそれの動作要素に依存し得ます。VCO周波数を最適化するためのPLLの事前または事後の分周器は確定的細動を減らすのに役立ちます。

6.3. 電源

複数電源供給元での細動の柱状図の獲得はｼｽﾃﾑ内の細動の供給源特定に役立ち得ます。例えば、基板上電源と実験用電源の使用時の結果間の違いを考慮してください。検査後、結果がPCBで設計した電力回路の使用時に細動がもっと大きくなったなら、いくつかの回路調整が必要です。

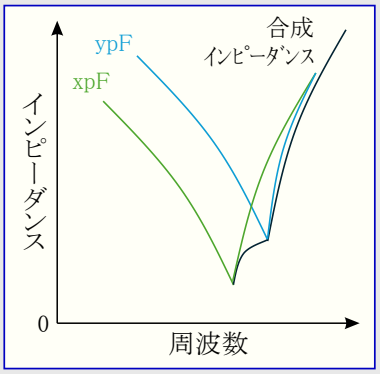
6.4. 回路調整

電力布線の設計から始め、それらの長さを考慮してください。行き先に達するのにかかるより長い布線は雑音の影響を受けやすくなります。これを解決する手っ取り早い方法はVDDとVSSの両方に独立した層を持つ最低4層のPCBを使うことです。この配置では接地経路がﾈｯﾄと接地へのﾋﾞｱを加えた長さよりももはや長くなりません。布線長を更に制限するため、部品が共に近く配置されることを確実にしてください。これはDUT(被試験体)傍の雑音分離ｺﾝﾃﾞﾝｻで特に重要です。加えて、ﾃﾞﾊﾞｲｽからﾋﾟﾝを選ぶ時に、それらが共に使われない限り、2つの高速信号は隣接ﾋﾟﾝから使われないようにすべきです。例えば、PWMのすぐ隣でのI²Cの使用は2つのﾋﾟﾝを横切る雑音をもたらし、細動を増やしてPWMの分解能を減らします。I²CとPWMが違うｸﾛｯｸ元を使っている場合、非同期信号が同期信号よりももっと誤差を発生させるため、その雑音が増します。

6.5. 部品

部品選びはｼｽﾃﾑ雑音での影響を持ち得ます。雑音分離(ﾊﾞｲﾊﾟｽ)ｺﾝﾃﾞﾝｻの値は望む動作周波数で最低ｲﾝﾋﾟｰﾀﾞﾝｽを持つように選ばれるべきです。これはﾋﾟﾝが交互切替する時にどのｺﾝﾃﾞﾝｻが電源で最良の濾波器として働くかを理解することを意味します。これらの付加的な雑音分離ｺﾝﾃﾞﾝｻの追加はﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄで推奨されたｺﾝﾃﾞﾝｻ配置を超えてPCB上の空間も奪うかもしれず、故にｺﾝﾃﾞﾝｻのﾃﾞｰﾀｼｰﾄを調べて応用周波数に基づく必要な値だけを選ぶことを確実にしてください。右図はこれらのﾃﾞｰﾀｼｰﾄで見つかる一般的な図を示します。

図6-1. ｲﾝﾋﾟｰﾀﾞﾝｽと周波数



6.6. その他の要素

温度のような他の変数は結果としてﾃﾞﾊﾞｲｽに細動の影響を及ぼすかもしれません。次の例は100kHzのPWM出力でCuriosity環境基板上のdsPIC33AK128MC106 GP DIMを使います。2つの図で見られるように、温度増加はPWMの出力に大きく影響を及ぼし得ます。温度が上昇すると、確定的細動も増し、これは周回時間をもっと不定期にさせます。

図6-2. PWM 100kHz – 室温

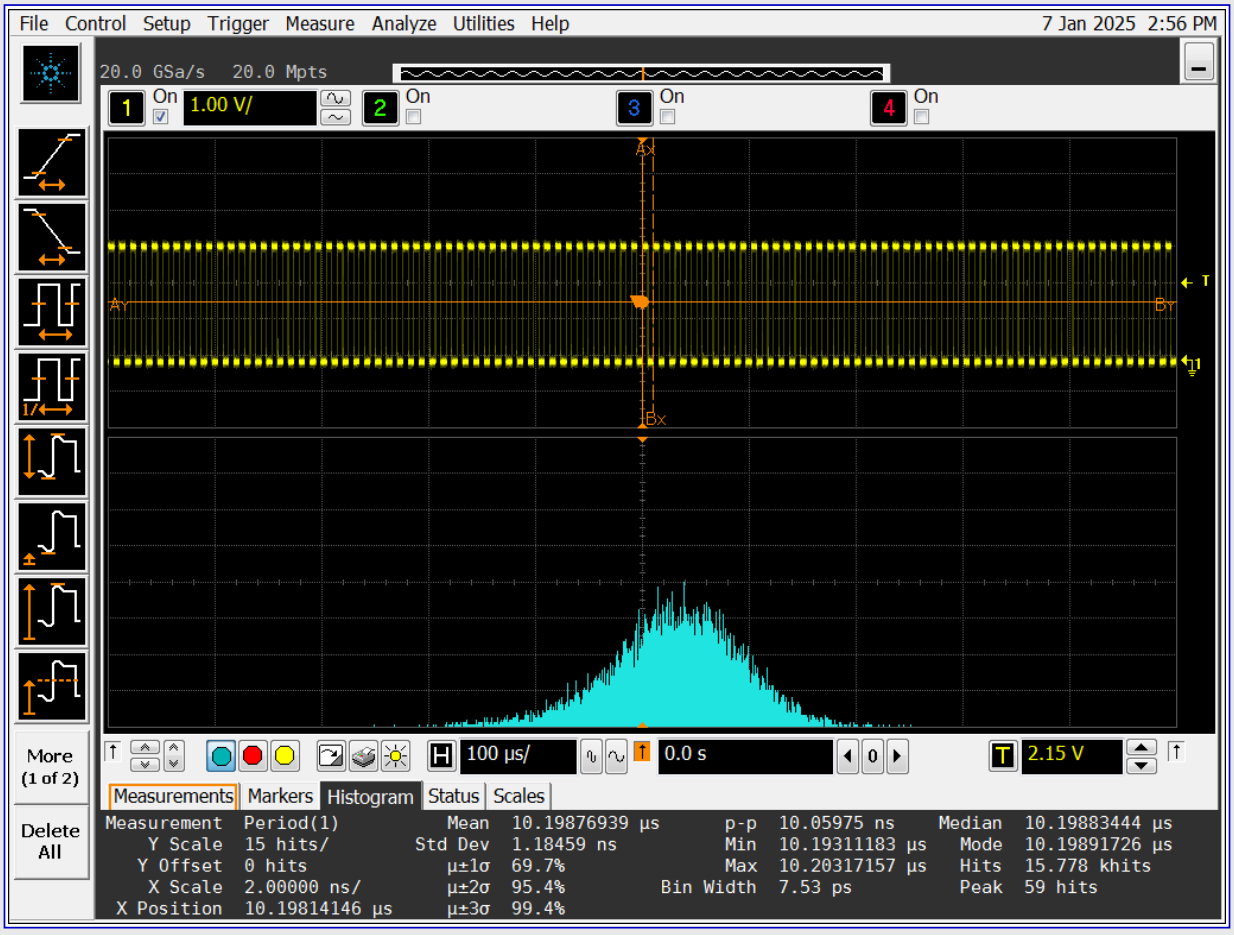
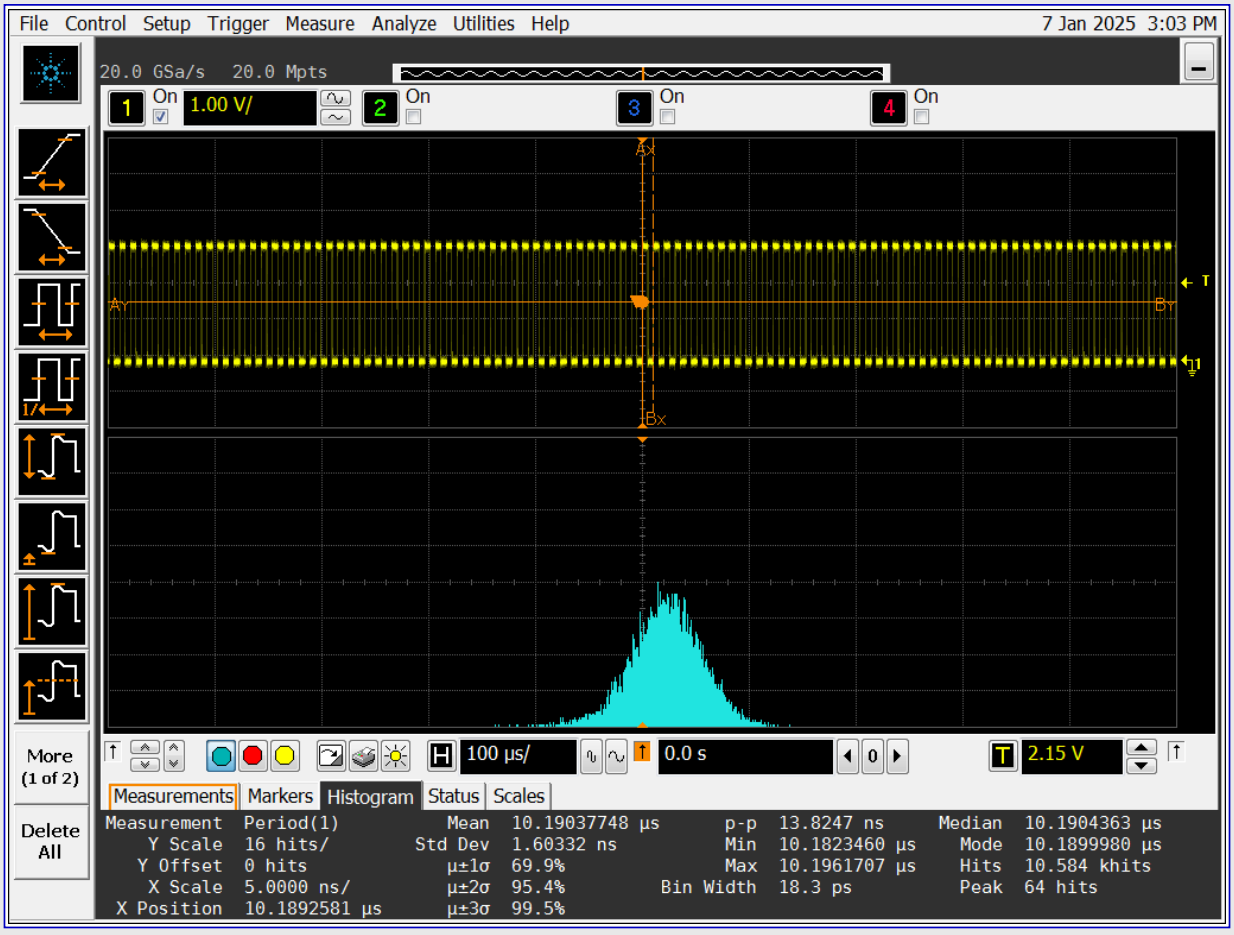


図6-3. PWM 100kHz – 高温

室温からの低下もPWM出力に影響を及ぼし得ます。

図6-4. PWM 100kHz – 低温



温度以外に細動は部品毎にも変わり得て時間と共に変化し得ます。これは製造手段、部品劣化、電源安定性のためかもしれません。装置の寿命を保証して時間と共に増える細動を防ぐ最も効果的な方法は装置動作中に制御された環境を維持することです。雑音の少ない電源は時間と共にシステムでの負荷の発生を減らし、高温から装置を守ることは経年変化を遅らせるのに役立ちます。

7. 最高性能のためのクロック構成設定例

入力クロック元とPLL構成設定はPWM出力細動(ジッタ)で重要な影響を持ち得ます。本章はdsPICのPWM応用で性能をどう最大化するかの例を提供します。この例を通して使う式は以下の式に基づいています。

式7-1.

$$F_{PLLI} \times \left(\frac{PLLBDIV}{PLLPRE \times POSTDIV1 \times POSTDIV2} \right) = F_{PLLO}$$

より小さな帰還分周値(PLLFBDIV)を使うPLL回路は細動を最小化し得ます。より小さな帰還分周値は入力信号での変動に対して修正するためのより短い濾波閉路の周期とより速い応答になります。この例では(デバイスのデータシートの電気的特性で指定されたように)最小VCO周波数は400MHzです。400MHzの出力クロック周波数を得るのにPLLを使うことは様々な方法で達成することができます。8MHzのFRCが使われた場合、PLLは次式を使って設定することができます。

式7-2.

$$8 \times \left(\frac{100}{1 \times 2 \times 1} \right) = 400$$

8MHzのFRCでの帰還分周器が100とは100の利得倍率があることを意味します。

更なる細動削減は高精度の供給元クロックとより高い周波数で達成することができます。PLL参照基準クロック周波数が8MHzから、MEMS(微小電気機械システム)またはクリスタル発振器のどちらかで25MHz外部クロックに増された場合、式はより小さな帰還分周器値の32を与えます。

式7-3.

$$25 \times \left(\frac{32}{1 \times 2 \times 1} \right) = 400$$

参照基準クロック周波数の増加とそれによるPLL帰還分周器の調整により、PLL性能は最適化されます。これは与えられた信号での潜在的な細動を劇的に低めます。この例では細動を軽減する2つの方法が含まれますが、確定的細動を減らしてPWM用の最適な参照基準クロックを達成するために始めの方で紹介した付加的な方法も考慮されるべきです。

dsPIC33A部での細動を検査するにはdsPIC33AK128MC106 GP DIMとCuriosity基盤開発基板を使ってください。以降の例ｺｰﾄﾞはPWM信号を出力するのにmikroBUSﾍｯﾀﾞAの16番ﾋﾟﾝを使います。各例はどの細動が測定され得るかによって異なるクロック元を使います。最初の例はdsPICからの8MHz内部FRC発振器を使い、PWMは約100kHzを出力するためにそれからクロック駆動されます。2つ目の例はそれと同じFRCに加えて出力クロック信号を200MHzに設定するPLLを使います。PWM信号分周器は未だ100kHzを出力するようにそれによって調整されます。両方の場合でmikroBUSｲﾝﾀｰﾌｪｰｽAの16番ﾋﾟﾝで細動を測定するのにｵｼﾛｽｺｰﾌﾟを使うことができます。

8. ﾃﾞﾊﾞｲｽ固有の例

検査設定と結果の例が次のﾃﾞﾊﾞｲｽ系統、dsPIC33E、dsPIC33C、dsPIC33Aの各々に対して提供されます。各検査はｸﾛｯｸ元任意選択としてFRCとFRC+PLLの両方を提供し、故にFRCの結果は他のどれかの結果に対する比較点として使うことができます。細動(ｼﾞｯﾀ)はｸﾛｯｸREFO周辺機能とPWM出力の両方で検査されます。これらの設定はそれらが再現できるようにお客様の購入で入手可能な開発基板を使います。加えて、ﾃﾞｭｰﾃｨ ｻｲｸﾙとTIEのようなもっと特定の検査のためにﾃﾞｭｰﾃｨ ｻｲｸﾙと周期を容易に変更することができます。

8.1. dsPIC33E

dsPIC33E検査は対応するdsPIC33EP128GS806 DP PIMと共にﾃﾞｼﾞﾀﾙ電力開発基板の版を使います。PWM出力1はｺｰﾄﾞで構成設定され、この単位部はｺｰﾄﾞの先頭で何が定義されるかに応じてそれのｸﾛｯｸ元としてFRCまたはFRC+PLLのどちらかを使います。両方の場合でPWMは細動測定と2つの結果の比較のために100kHzを出力します。

```
// <editor-fold defaultstate="collapsed" desc="Config Bits">
// FICD
#pragma config ICS = PGD1             // ICD通信ﾁｬﾈﾙ選択ﾋﾞｯﾄ(PGECIとPGED1での通信)
//#pragma config JTAGEN = OFF         // JTAG許可ﾋﾞｯﾄ(JTAG禁止)

// FPOR
#pragma config ALTI2C1 = OFF          // 代替I2C1ﾋﾟﾝ(SDA1/SCL1ﾋﾟﾝに割り当てたI2C1)
//#pragma config ALTI2C2 = OFF        // 代替I2C2ﾋﾟﾝ(SDA2/SCL2ﾋﾟﾝに割り当てたI2C2)
#pragma config WDTWIN = WIN25         // ｳｫｯﾁﾄﾞｯｸﾞ窓選択ﾋﾞｯﾄ(WDT窓はWDT周期の25%)

// FWDT
#pragma config WDTPOST = PS32768      // ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ後置分周器ﾋﾞｯﾄ(1:32,768)
#pragma config WDTPRE = PR128         // ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ前置分周器ﾋﾞｯﾄ(1:128)
#pragma config PLLKEN = ON            // PLL許可ﾋﾞｯﾄ(PLL元へのｸﾛｯｸ切替はPLL固定化信号有効まで待機)
#pragma config WINDIS = OFF           // ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ窓許可ﾋﾞｯﾄ(非窓動作でのｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ)
#pragma config WDTEN = OFF            // ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ許可ﾋﾞｯﾄ(ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ常時許可)

// FOSC
#pragma config POSCMD = NONE          // 主発振器動作選択ﾋﾞｯﾄ(主発振器禁止)
#pragma config OSCIOFNC = ON          // OSC2ﾋﾟﾝ機能ﾋﾞｯﾄ(OSC2はｸﾛｯｸ出力)
#pragma config IOL1WAY = ON           // 周辺機能ﾋﾟﾝ選択構成設定(1つの再構成設定だけ許可)
#pragma config FCKSM = CSECMD         // ｸﾛｯｸ切替動作ﾋﾞｯﾄ(ｸﾛｯｸ切替許可、障害安全ｸﾛｯｸ監視部禁止)

// FOSCSEL
#pragma config FNOSC = FRC            // 発振器供給元選択(内部高速RC(FRC))
#pragma config PWMLOCK = OFF          // PWM固定化許可ﾋﾞｯﾄ(正しいPWMﾚｼﾞｽﾀは鍵手順後にだけ書き込み可)
#pragma config IESO = ON              // 2速発振器始動許可ﾋﾞｯﾄ(FRCでﾃﾞﾊﾞｲｽ始動後に使用者選択発振器元に切替)

// FGS
//#pragma config GWRP = OFF           // 一般区分書き込み保護ﾋﾞｯﾄ(一般区分書き込み可)
//#pragma config GCP = OFF            // 一般区分ｺｰﾄﾞ保護ﾋﾞｯﾄ(一般区分保護禁止)
// </editor-fold>

#include<xc.h>

#define use_PLL
//#define use_FRC

int main(void) {

    _TRISA4 = 0;
    _TRISC13 = 0;
#ifdef use_FRC
    PTPER = 4000000 / 6840;                    // FRCでの使用のための周期設定
#endif

#ifdef use_PLL
    //PTPER = FCY/(PWM frequency * prescale)
```

```
    PTPER = 40000000 / 68400;                   // FRC+PLLでの使用のための周期設定
    CLKDIVbits.PLLPRE = 0;
    PLLFBD = 40;
    CLKDIVbits.PLLPOST = 0b01;
    CLKDIVbits.PLLPRE = 0;
    // initiate clock switch to FRC with PLL (NOSC=1)
    __builtin_write_OSCCONH(0x01);              // NOSC=1 -> 新OSC設定
    __builtin_write_OSCCONL(OSCCONL | 0x01);    // OSWEN=1 -> クロック切替要求
    while(OSCCONbits.OSWEN != 0);               // クロック切替待ち
    while(OSCCONbits.LOCK != 1);                // PLL固定化待ち
#endif
    //REF0 setup
    _RP52R = 0b110001;                          // REF0 PPS
    _RODIV = 1;                                 // REF0分圧器
    _REFOMD = 0;                                // 参照基準単位部許可
    _ROON = 1;                                  // 参照基準出力ON
    _ROSEL = 0;                                 // 参照基準はシステム クロック

    PTCONbits.PTEN = 0;                         // PWM単位部は始動中禁止
    PWMCON1bits.ITB = 0;                        // PTPERレジスタがPWM1の時間基準を提供
    PTCONbits.PTSIDL = 0;                       // PWM時間基準は自由走行動作で作動
    PTCON2bits.PCLKDIV = 0;                     // PWM時間基準入力クロック周期はTCY(1:1前置分周)

    // Configure PWM1
    IOCON1bits.PENH = 1;                        // PWMx単位部がPWMxHピンを制御
    IOCON1bits.PENL = 0;                        // GPIO単に部がPWMxLピンを制御
    IOCON1bits.PMOD = 3;                        // 独立動作でのPWM1
    FCLCON1bits.FLTMOD = 0b11;                  // 障害動作禁止

    PDC1 = PTPER / 2;                           // PTPERの50%
    PTCONbits.PTEN = 1;                         // 他の全設定構成設定後にPWMを許可

    while(1){
        Nop();
    }
    return 0;
}
```

8.1.1 dsPIC33Eの結果

下はdsPIC33EP128GS806 DP PIMでの細動検査からの結果です。

図8-1. FRCでのdsPIC33EP128GS806 PWM細動結果

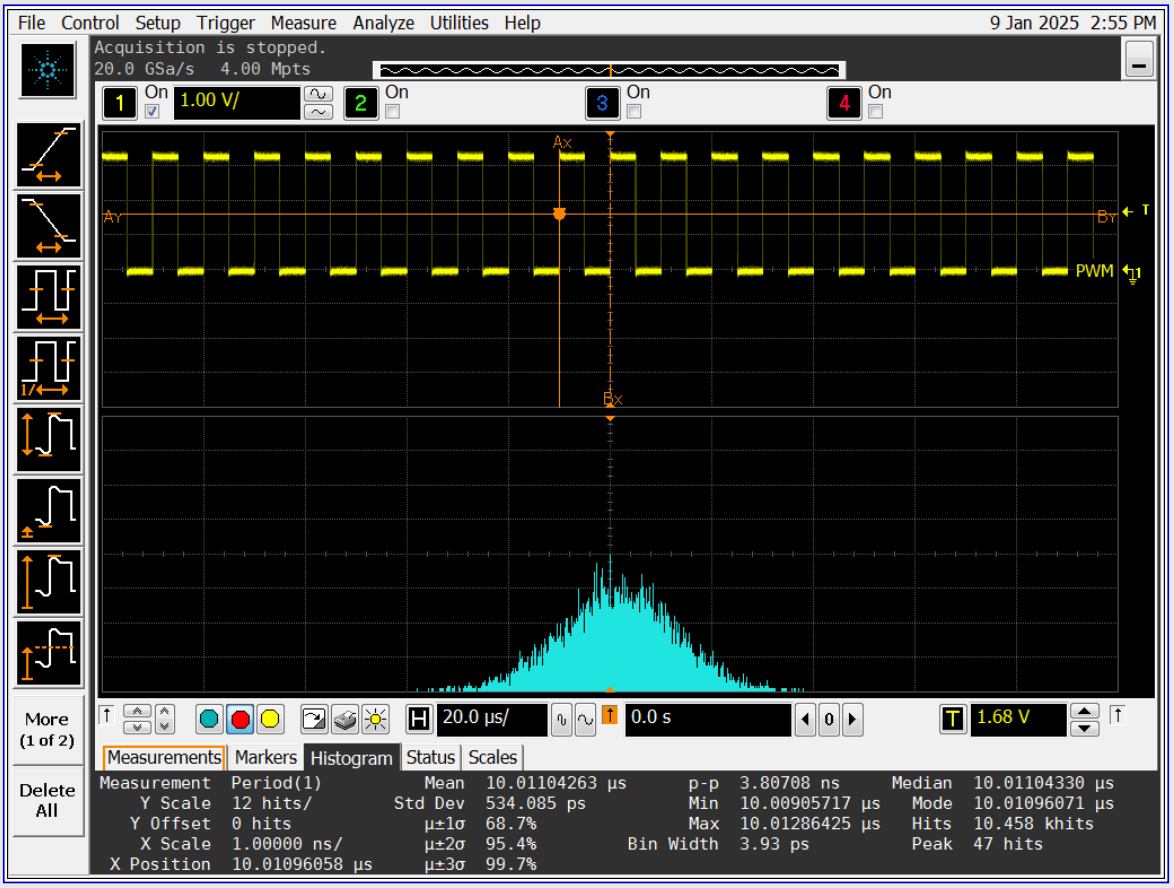
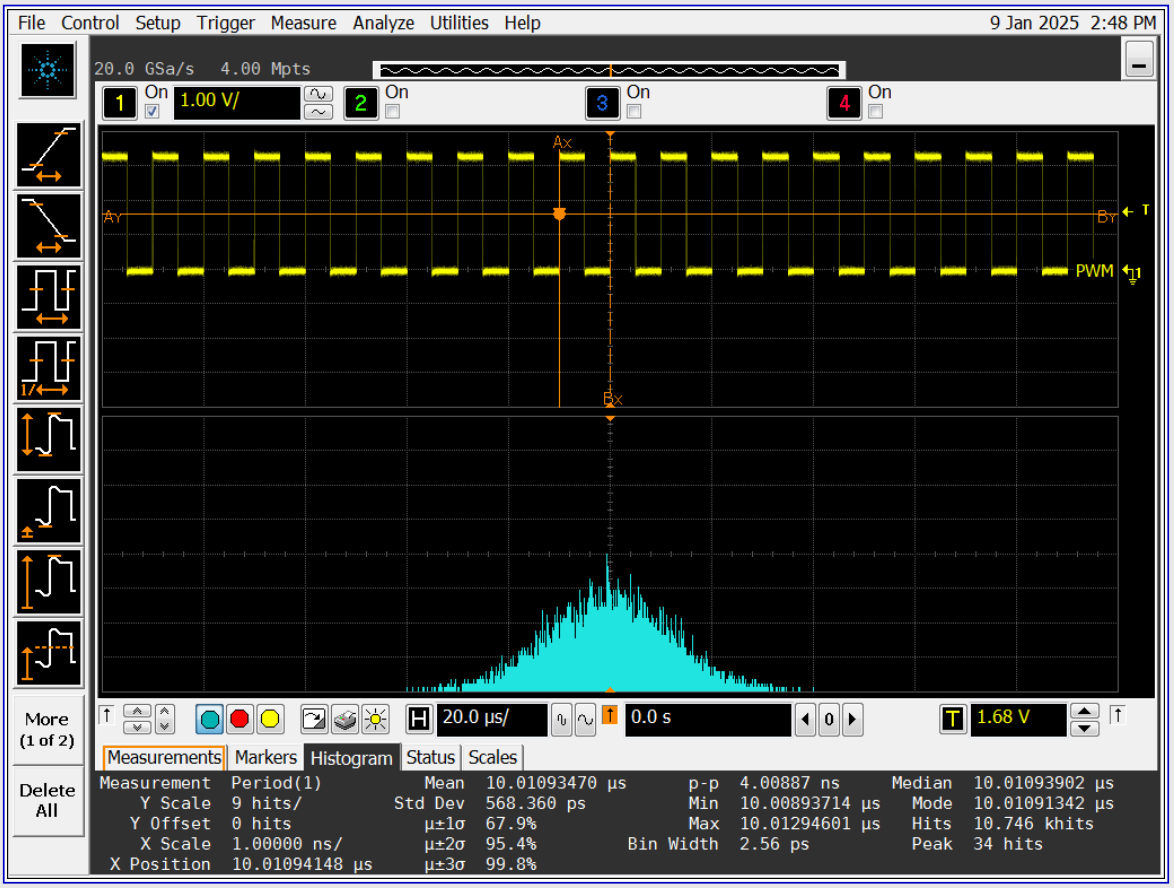


図8-2. FRC+PLLでのdsPIC33EP128GS806 PWM細動結果



8.2. dsPIC33C

dsPIC33C検査は基板上のdsPIC33CK1024MP710と共にdsPIC33C Touch CAN LIN Curiosity開発基板を使います。提供した例コードはPEMとクロック参照基準出力(REFO)周辺機能の両方の検査を許します。両出力はコードの先頭で何が定義されるかに応じてFRCまたはFRC+PLLのどちらかを使って検査をすることができます。

```
// <editor-fold defaultstate="collapsed" desc="Config Bits">
// FOSCSE
#pragma config FNOSC = FRCDIV        // 発振器供給元選択(前置分周器付き内部高速RC(FRC)発振器)
#pragma config IESO = ON             // 2速発振器始動許可ビット(FRCでデバイス始動後に使用者選択発振器元に切替)

// FOSC
#pragma config POSCMD = NONE         // 主発振器動作選択ビット(主発振器禁止)
#pragma config OSCIOFNC = OFF        // OSC2ピン機能ビット(OSC2はクロック出力)
#pragma config FCKSM = CSECMD        // クロック切替動作ビット(クロック切替許可、障害安全クロック監視部禁止)
#pragma config PLLKEN = ON           // PLL固定化状態制御(PLL固定化信号は固定化消失時にPLLクロック出力禁止に使用)
#pragma config XTCFG = G3            // XT構成設定(24～32MHzクリスタル)
#pragma config XTBST = ENABLE        // XT促進(始動促進)

// FICD
#pragma config ICS = PGD1            // ICD通信チャネル選択ビット(PGC1とPGD1で通信)
#pragma config JTAGEN = OFF          // JTAG許可ビット(JTAG禁止)
#pragma config NOBTSWP = DISABLED    // BOOTSWP命令禁止ビット(BOOTSWP命令禁止)
// </editor-fold>

#include "xc.h"
#include <libpic30.h>
//#define use_PL
L#define use_FRC
int main(void) {
    _RP52R = 14;                                // デバイスの80番ピン、RC4

#ifdef use_PLL
    CLKDIVbits.PLLPRE   = 1;                    // N1=1
    PLLFBDbits.PLLFBDIV = 125;                  // M=125 PLLFBD
    PLLDIVbits.POST1DIV = 5;                    // N2=5 PLLDIV
    PLLDIVbits.POST2DIV = 2;                    // N3=1

    __builtin_write_OSCCONH(0x01);              // NOSC=1 -> 新OSC設定
    __builtin_write_OSCCONL(OSCCONL | 0x01);    // OSWEN=1 -> クロック切替要求
    while(OSCCONbits.OSWEN != 0);               // クロック切替待ち
    while(OSCCONbits.LOCK != 1);                // PLL固定化待ち
    PG7PER = 0x200;                             // FRC+PLLで周期設定
    PG7DC = 0x100;                              // FRC+PLLでデューティサイクル設定
#endif

#ifdef use_FRC
    PG7PER = 0x50;                              // FRCクロック元で周期設定
    PG7DC = 0x25;                               // FRCクロック元でデューティサイクル設定
#endif

    _ANSELE0 = 0;                               // E0でのデジタル機能
    _ROOUT = 1;                                 // 参照基準クロック出力許可
    _ROSEL = 0;                                 // 参照基準元はシステム クロック
    _RODIV = 0;                                 // 基準クロック値の分周なし
    _ROEN = 1;                                  // 参照基準クロック許可
    _TRISD1 = 0;                                // D1はデジタル出力設定

    PCLKCONbits.MCLKSEL = 0b00;                 // 0はFOSC、2はPLL分周器出力
    PG7CONHbits.TRGMOD = 1;                     // 再起動可能動作
    PG7CONLbits.CLKSEL = 0b01;                  // MCLKSELによって設定したクロックを使用
```

```
    PG7IOCONHbits.PMOD = 1;                 // PWMは独立動作で作動
    PG7IOCONHbits.PENH = 1;                 // PWMがピンの出力を制御
    PG7CONLbits.ON = 1;                     // PWM 7許可

    while(1){
        Nop();
    }
    return 0;
}
```

8.2.1 dsPIC33Cの結果

下は提供した検査ｺｰﾄﾞを使ってdsPIC33C Touch CAN LIN Curiosity基板で8MHz FRC対FRC+PLLによってｸﾛｯｸ駆動されているPWM出力に対する柱状図の比較です。

図8-3. FRCでのdsPIC33CK1024MP710 PWM細動結果

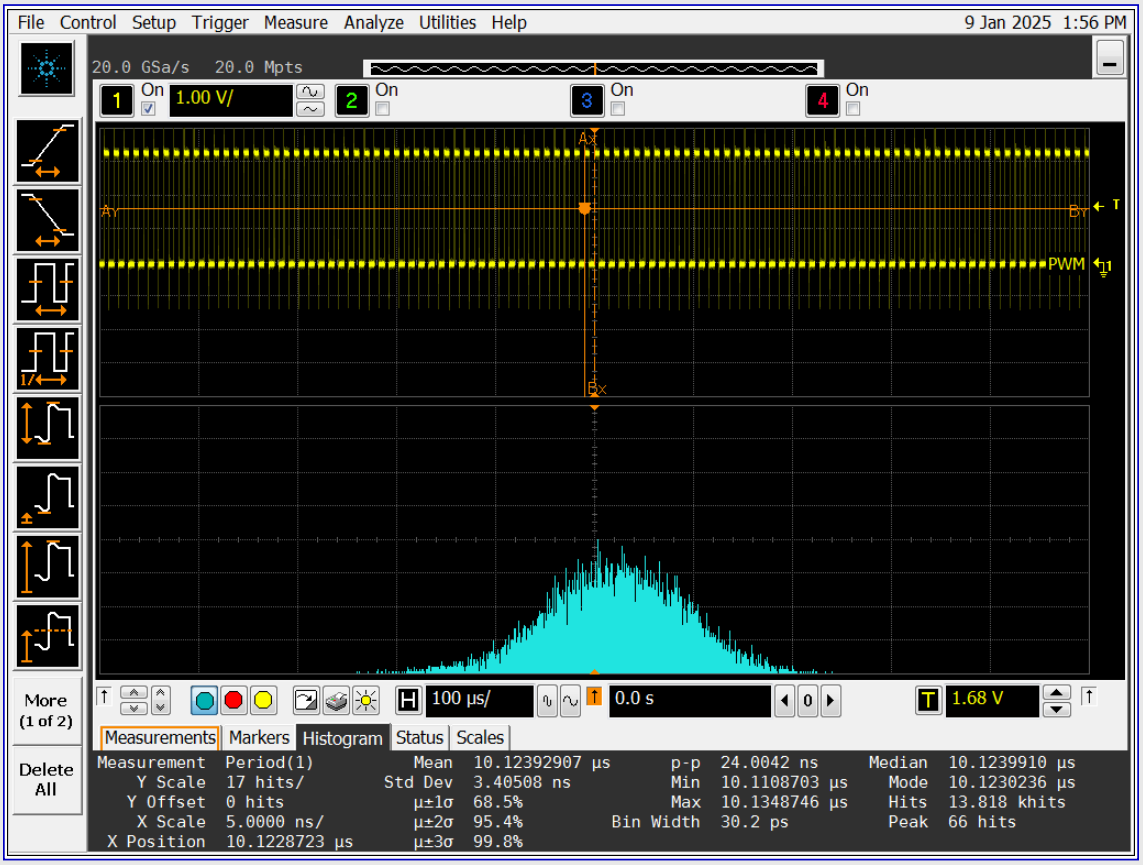
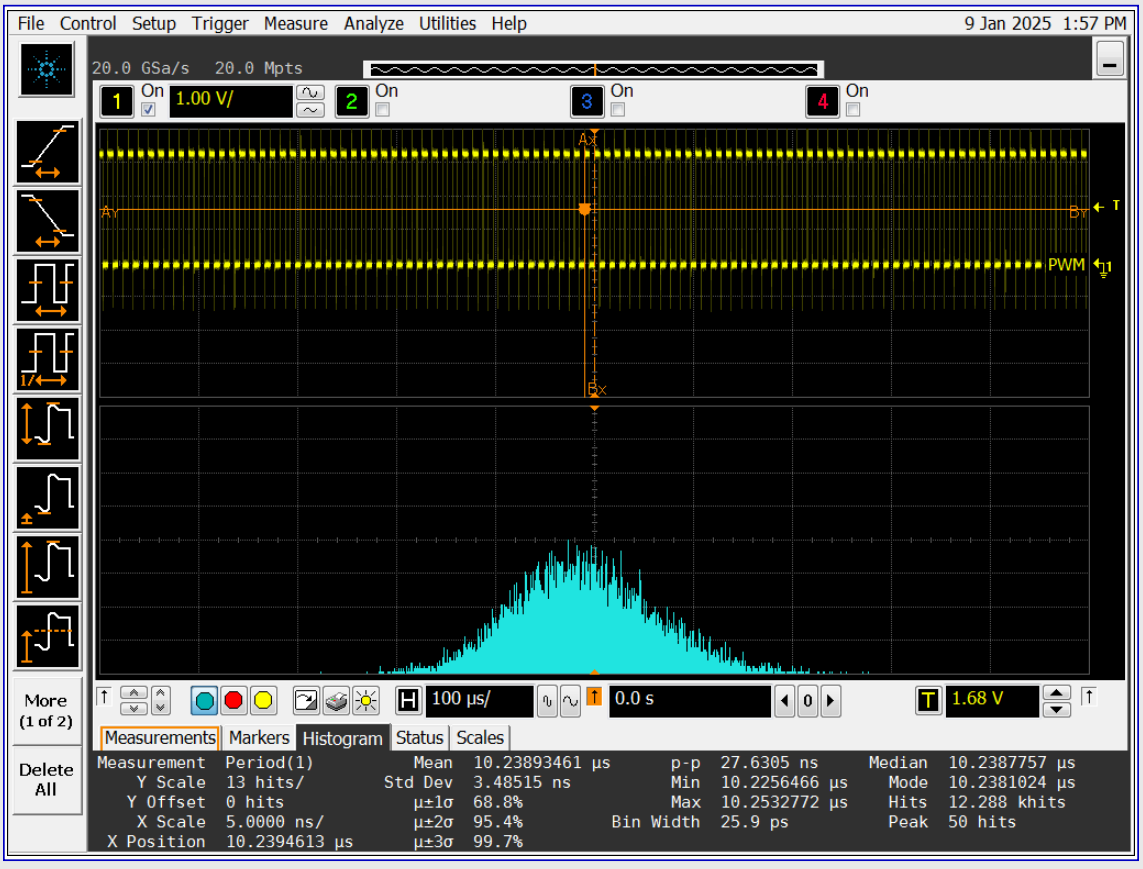


図8-4. FRC+PLLでのdsPIC33CK1024MP710 PWM細動結果



8.3. dsPIC33A

dsPIC33A部での細動検査はdsPIC33AK128MC106 GP DIMとCuriosity基盤開発基板を使います。

例ｺｰﾄﾞは細動を検査するためのいくつかのｸﾛｯｸ設定を提供します。この例はFRC、FRC+PLL、8MHz MEMS、MEMS+PLLによってｸﾛｯｸ駆動されているPWMを含みます。PWM信号分周器は全ての場合で100kHzを出力するためにそれによって調整され、出力を測定するのにｵｼﾛｽｺｰﾌﾟを使うことができます。

```
#include "xc.h"
#include <stdio.h>
#define use_FRC
//#define use_PLL
//#define use_MEMS
//#define use_MEMS_PLL

void pwm_Init();
void clock_PWM_at_FRC();
void clock_PWM_at_MEMS();
void clock_PWM_at_400MHz_from_PLL2_Fout();
void clock_PWM_at_400MHz_EC_PLL2_Fout();

void clock_PWM_at_400MHz_from_PLL2_Fout() {
    PLL2CONbits.ON = 1;             // 既に許可されていなければPLL生成器2を許可
    PLL2CONbits.NOSC = 1;           // PLL2のｸﾛｯｸ元としてFRCを選択
    PLL2CONbits.OSWEN = 1;          // PLL2ｸﾛｯｸ切替要求
    while(PLL2CONbits.OSWEN);       // PLL2ｸﾛｯｸ切替完了待ち

    // PLL2分周器を200MHz出力に設定
    PLL2DIVbits.PLLPRE = 1;         // 参照基準入力は8MHz、分周なし
    PLL2DIVbits.PLLFBDIV = 200;     // Fvco=8MHz×200=1600MHz
    PLL2DIVbits.POSTDIV1 = 4;       // Fvcoを4分周
    PLL2DIVbits.POSTDIV2 = 1;       // Fplllo=Fvco/4/1=400MHz

    // PLLSWENﾋﾞｯﾄはPLL帰還分周器への変更を制御
    PLL2CONbits.PLLSWEN = 1;        // PLL2帰還分周器切替要求
    while(PLL2CONbits.PLLSWEN);     // PLL2帰還分周器切替完了待ち

    // FOUTSWENﾋﾞｯﾄはPLL出力分周器への変更を制御
    PLL2CONbits.FOUTSWEN = 1;       // PLL2出力分周器切替要求
    while(PLL2CONbits.FOUTSWEN);    // PLL2出力分周器切替完了待ち

    CLK5CONbits.ON = 1;             // CLKGEN5許可
    CLK5DIVbits.INTDIV = 0;         // CLKGEN5分数分周器を比率1:1にﾘｾｯﾄ
    CLK5DIVbits.FRACDIV = 0;
    CLK5CONbits.DIVSWEN = 1;        // CLKGEN5分数分周器切替要求
    while(CLK5CONbits.DIVSWEN);     // CLKGEN5分数分周器切替完了待ち

    CLK5CONbits.NOSC = 6;           // PLL2のFoutをCLKGEN5ｸﾛｯｸ元として設定
    CLK5CONbits.OSWEN = 1;          // CLKGEN5ｸﾛｯｸ切替要求
    while (CLK5CONbits.OSWEN);      // CLKGEN5ｸﾛｯｸ切替完了待ち

    PCLKCONbits.MCLKSEL = 1;        // CLKGEN5をPWM主ｸﾛｯｸ元として選択
}

void clock_PWM_at_400MHz_EC_PLL2_Fout() {
    _POSCMD = 0b00;
    PLL2CONbits.ON = 1;             // 既に許可されていなければPLL生成器2を許可
    PLL2CONbits.NOSC = 3;           // PLL2のｸﾛｯｸ元としてFRCを選択
    PLL2CONbits.OSWEN = 1;          // PLL2ｸﾛｯｸ切替要求
    while(PLL2CONbits.OSWEN);       // PLL2ｸﾛｯｸ切替完了待ち

    // PLL2分周器を200MHz出力に設定
```

```
    PLL2DIVbits.PLLPRE = 1;          // 参照基準入力は8MHz、分周なし
    PLL2DIVbits.PLLFBDIV = 200;      // Fvco=8MHz×200=1600MHz
    PLL2DIVbits.POSTDIV1 = 4;        // Fvcoを4分周
    PLL2DIVbits.POSTDIV2 = 1;        // Fplllo=Fvco/4/1=400MHz

    // PLLSWENﾋﾞｯﾄはPLL帰還分周器への変更を制御
    PLL2CONbits.PLLSWEN = 1;         // PLL2帰還分周器切替要求
    while(PLL2CONbits.PLLSWEN);      // PLL2帰還分周器切替完了待ち

    // FOUTSWENﾋﾞｯﾄはPLL出力分周器への変更を制御
    PLL2CONbits.FOUTSWEN = 1;        // PLL2出力分周器切替要求
    while(PLL2CONbits.FOUTSWEN);     // PLL2出力分周器切替完了待ち

    CLK5CONbits.ON = 1;              // CLKGEN5許可
    CLK5DIVbits.INTDIV = 0;          // CLKGEN5分数分周器を比率1:1にﾘｾｯﾄ
    CLK5DIVbits.FRACDIV = 0;
    CLK5CONbits.DIVSWEN = 1;         // CLKGEN5分数分周器切替要求
    while(CLK5CONbits.DIVSWEN);      // CLKGEN5分数分周器切替完了待ち

    CLK5CONbits.NOSC = 6;            // PLL2のFoutをCLKGEN5ｸﾛｯｸ元として設定
    CLK5CONbits.OSWEN = 1;           // CLKGEN5ｸﾛｯｸ切替要求
    while (CLK5CONbits.OSWEN);       // CLKGEN5ｸﾛｯｸ切替完了待ち

    PCLKCONbits.MCLKSEL = 1;         // CLKGEN5をPWM主ｸﾛｯｸ元として選択
}

void clock_PWM_at_FRC(){
    CLK5CONbits.ON = 1;              // CLKGEN5許可
    CLK5DIVbits.INTDIV = 0;          // CLKGEN5分数分周器を比率1:1にﾘｾｯﾄ
    CLK5DIVbits.FRACDIV = 0;
    CLK5CONbits.DIVSWEN = 1;         // CLKGEN5分数分周器切替要求
    while(CLK5CONbits.DIVSWEN);      // CLKGEN5分数分周器切替完了待ち
    CLK5CONbits.NOSC = 1;            // FRCをCLKGEN5ｸﾛｯｸ元として設定
    CLK5CONbits.OSWEN = 1;           // CLKGEN5ｸﾛｯｸ切替要求
    while (CLK5CONbits.OSWEN);       // CLKGEN5ｸﾛｯｸ切替完了待ち
    PCLKCONbits.MCLKSEL = 1;         // CLKGEN5をPWM主ｸﾛｯｸ元として選択
}

void clock_PWM_at_MEMS(){
    _POSCMD = 0b00;
    CLK5CONbits.ON = 1;              // CLKGEN5許可
    CLK5DIVbits.INTDIV = 0;          // CLKGEN5分数分周器を比率1:1にﾘｾｯﾄ
    CLK5DIVbits.FRACDIV = 0;
    CLK5CONbits.DIVSWEN = 1;         // CLKGEN5分数分周器切替要求
    while(CLK5CONbits.DIVSWEN);      // CLKGEN5分数分周器切替完了待ち
    CLK5CONbits.NOSC = 3;            // MEMSをCLKGEN5ｸﾛｯｸ元として設定
    CLK5CONbits.OSWEN = 1;           // CLKGEN5ｸﾛｯｸ切替要求
    while (CLK5CONbits.OSWEN);       // CLKGEN5ｸﾛｯｸ切替完了待ち
    PCLKCONbits.MCLKSEL = 1;         // CLKGEN5をPWM主ｸﾛｯｸ元として選択
}

void pwm_Init(){
    //PG1PER = 0x10000;              // PLLとで100kHz
    //PG1DC = 0x8000;                // PLLとで100kHz

    PG1CONbits.UPDMOD = 0b000;       // PWM緩衝更新動作はUPDREQ=1の場合に次のPWM周回で開始
    PG1CONbits.TRGMOD = 0b01;        // PWM生成器1は単一起動動作で作動
    PG1CONbits.SOCS = 0b0000;        // 周回の開始は局所EOC

    PG1CONbits.ON = 0;               // PWM生成器1禁止(まだ開始しません。)
    PG1CONbits.TRGCNT = 1;           // PWM生成器1は起動時に1つのPWM周期を生成
```

```
    PG1CONbits.CLKSEL = 0b01;       // PWM生成器1は非分周、非尺度調整のPWM主ｸﾛｯｸを使用
    PG1CONbits.MODSEL = 0b000;      // PWM生成器1は独立端PWM動作で作動

    PG1IOCONbits.PMOD = 0b01;       // PWM生成器1出力動作は独立動作
    PG1IOCONbits.PENH = 1;          // PWM生成器1はPWM1H出力ﾋﾟﾝを制御
    PG1IOCONbits.PENL = 1;          // PWM生成器1はPWM1L出力ﾋﾟﾝを制御
    PG1CONbits.ON = 1;
}

int main(void) {

#ifdef use_FRC
    clock_PWM_at_FRC();
    PG1PER = 0x500;                 // FRCでの100kHz
    PG1DC = 0x250;                  // FRCでの100kHz
#endif

#ifdef use_PLL
    clock_PWM_at_400MHz_from_PLL2_Fout();
    PG1PER = 0xFB11;                // PLLでの100kHz
    PG1DC = 0x7D81;                 // PLLでの100kHz
#endif

#ifdef use_MEMS
    clock_PWM_at_MEMS();
    PG1PER = 0x500;                 // MEMSでの100kHz
    PG1DC = 0x250;                  // MEMSでの100kHz
#endif

#ifdef use_MEMS_PLL
    clock_PWM_at_400MHz_EC_PLL2_Fout();
    PG1PER = 0xFB11;                // PLLでの100kHz
    PG1DC = 0x7D81;                 // PLLでの100kHz
#endif

    pwm_Init();

    while(1){
        Nop();
    }
}
```

8.3.1 dsPIC33Aの結果

下は8MHz FRCやFRC+PLLによって400MHzでクロック駆動されている時のPWM出力での細動に対する検査結果です。FRC+PLL検査からの結果は(FRC検査の1,7nsと比べて1.2nsの)より低い標準偏差だけでなくより定義した中央点であることを示します。

図8-5. FRCでのdsPIC33AK128MC106 PWM細動結果

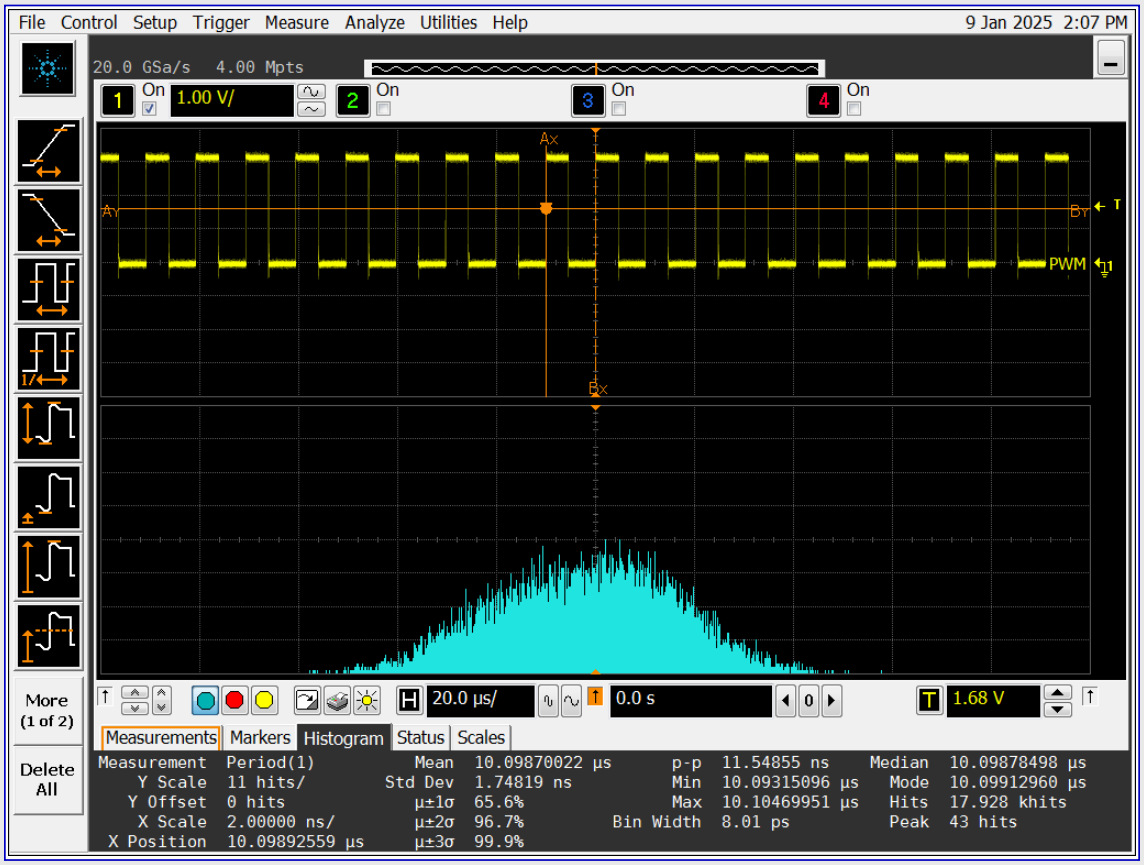


図8-6. FRC+PLLでのdsPIC33AK128MC106 PWM細動結果

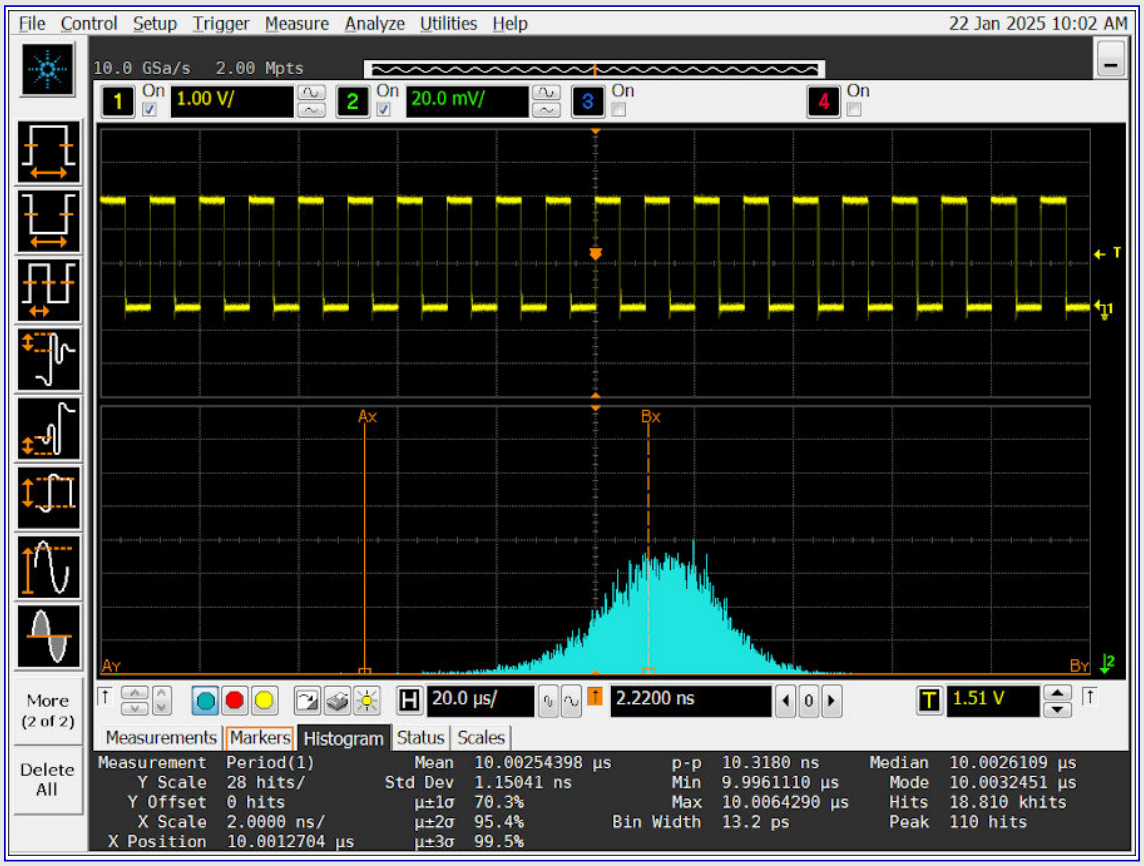


図8-7. 8MHz MEMSでのdsPIC33AK128MC106 PWM細動結果

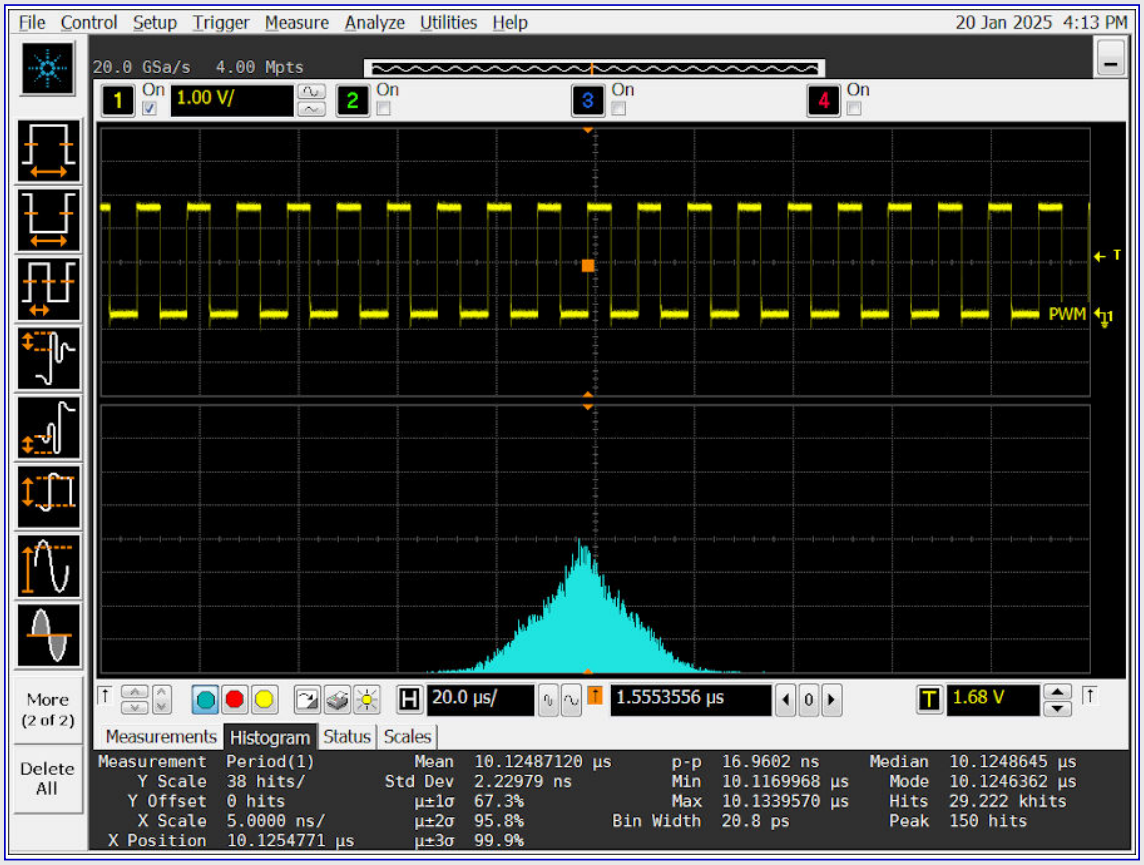
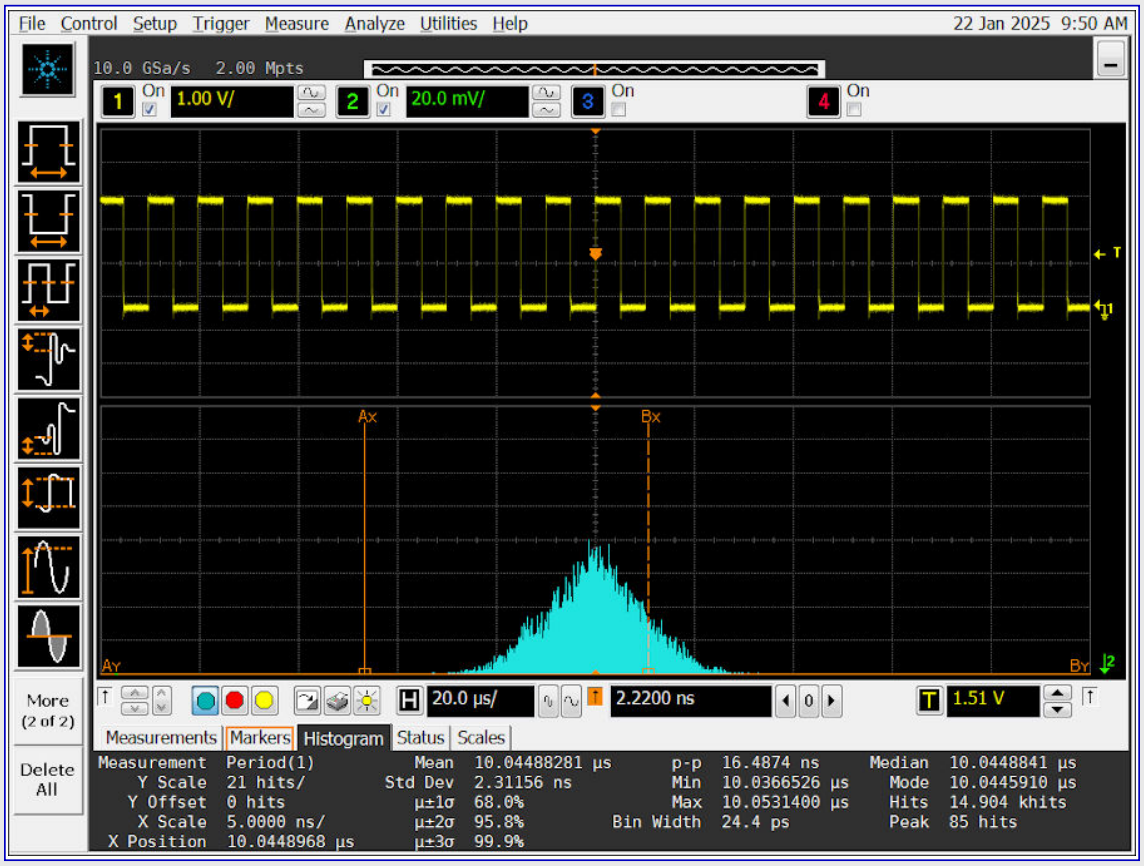


図8-8. 8MHz MEMS+PLLでのdsPIC33AK128MC106 PWM細動結果



9. 結び

この文書は応用で細動(ｼﾞｯﾀ)に対する期待値を設定し、細動を理解して測定し、可能な時に減らす方法を網羅します。どのｼｽﾃﾑにも無作為細動が存在するとは言え、この文書で提供された道具を使い、確定的細動を理解して管理することができます。これらの道具の使用は応用を改善し、その結果は装置と周辺ｼｽﾃﾑからのより良い結果をもたらします。

10. 改訂履歴

この改訂履歴はこの文書で実施された変更を記述します。変更は最も過去の公開で始まる改訂によって一覧にされます。

文書改訂	日付	注釈
A	2025年3月	初版

Microchip情報

商標

“Microchip”の名称とﾛｺﾞ、“M”のﾛｺﾞ、それと他の名称、ﾛｺﾞ、商標は米国や他の国に於けるMicrochip Technology Incorporatedまたはその系列会社や子会社の登録または未登録の商標です(“Microchip商標”)。Microchip商標に関する情報はhttps://www.microchip.com/en-us/about/legalinformation/microchip-trademarksで見つけることができます。

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。ﾃﾞﾊﾞ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ﾃﾞﾊﾞ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。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

Microchip製品での以下のｺｰﾄﾞ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ﾃﾞ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製品のｺｰﾄﾞ保護機能を侵害する試みは固く禁じられ、ﾃﾞｼﾞﾀﾙ 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ﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は製品が”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。ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。

日本語© *HERO* 2025.

本応用記述はMicrochipのAN5823応用記述(DS00005823A-2025年3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。