EPSON
EXCEED YOUR VISION

S1C17W18
温度補償
アプリケーションノート

セイコーエプソン株式会社

Rev.1.0

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

目　次

1. 周波数誤差の概要

最近のほとんどのMCUは、それぞれ異なる周波数と周波数誤差/偏差範囲を持つ複数のクロックソースに対応しています。具体的には、S1C17W18は以下のクロックソースを持っています。

- OSC3: 250kHz、384kHz、500kHz、1MHz、2MHz、4MHz内蔵発振、25°CにおいてMax. ±5%(50,000ppm)の周波数偏差
- IOSC: 700kHz内蔵発振、-40～85°CにおいてMax. ±7%(70,000ppm)の周波数偏差
- OSC1: 32.768kHz水晶発振(要外部振動子)、25°CにおいてTyp. ±20ppmの周波数偏差

周波数誤差/偏差の主な要因には次の2つがあります。

- 製造ばらつき
- 温度ドリフト

水晶振動子や発振器の仕様には、通常、製造ばらつきを示す25°Cにおける周波数偏差が記載されています。また、MCUの動作温度により、周波数にはずれが生じます。

本書は、製造ばらつきと温度ドリフトを補償する方法を説明します。S1C17W18は温度センサを内蔵しているため、外部に温度測定回路を設けることなく、その温度データを周波数誤差補償アルゴリズムで使用することができます。内蔵温度センサの温度補正方法も記載しています。

このアプリケーションノートには、本書に記載されているアルゴリズムのいくつかを実現し、補正のための周波数計測を容易にするための手引きとなるサンプルソフトウェアが用意されています。サンプルソフトウェアの動作については、4章で説明します。

1.1 OSC3周波数誤差

S1C17W18 OSC3内蔵発振回路の公称周波数は250kHz、384kHz、500kHz、1MHz、2MHz、4MHzの6種類で、使用する周波数をソフトウェアにより選択することができます。

S1C17W18テクニカルマニュアルの“電気的特性”の章内、“クロックジェネレータ(CLG)特性”の節にOSC3発振回路特性表が記載されており、周波数特性を見ることができます。環境温度が25°C、電源電圧が1.6～3.6Vの周波数誤差は、最悪の場合、±5%(50,000ppm)です。
また、OSC3内蔵発振周波数-温度特性グラフも掲載されており、下降方向の傾斜を持つ直線的なグラフから、温度が上がるにつれ周波数が下がることが分かります。

OSC3クロックの周波数はCLGTRIM.OSC3AJ[4:0]ビットを使用してトリミング(調整)することができます。これらのビットを最大値および最小値に設定した場合の4MHz OSC3クロック周波数の対温度特性を下図に示します。

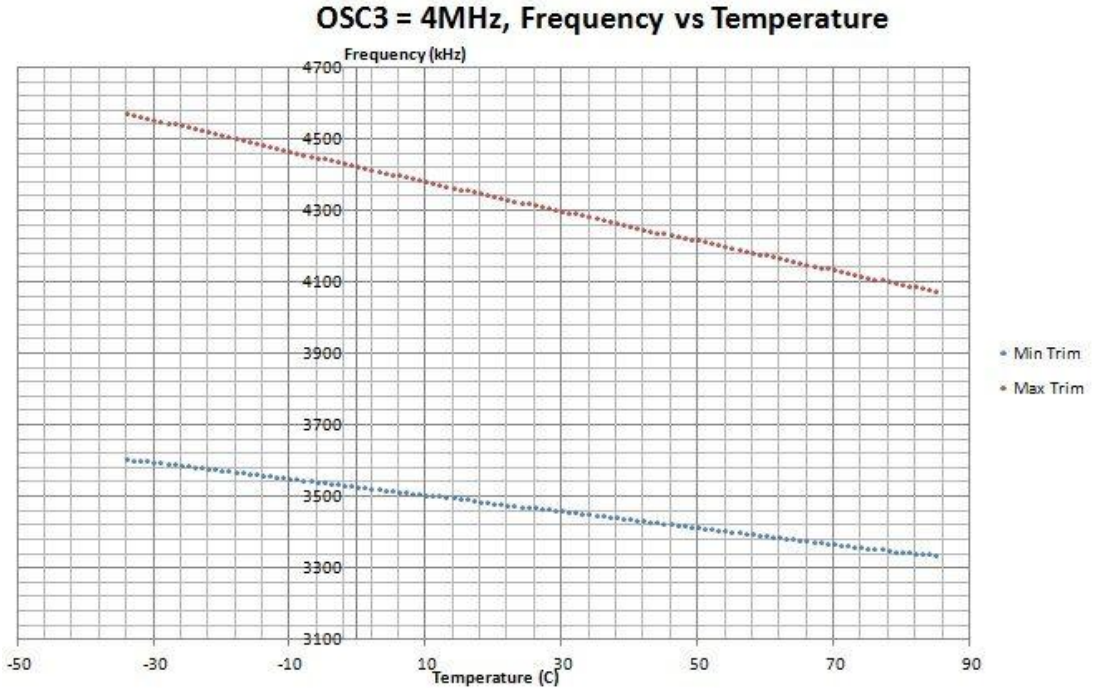

図1-1　最大/最小トリミング設定時の4MHz OSC3周波数-温度特性

1.2 IOSC周波数誤差

S1C17W18 IOSC内蔵発振回路の公称周波数は700kHzです。

S1C17W18テクニカルマニュアルの“電気的特性”の章内、“クロックジェネレータ(CLG)特性”の節にIOSC発振回路特性表が記載されており、周波数特性を見ることができます。環境温度が-40～85°C、電源電圧V_{DD}が1.6～3.6Vの周波数誤差は、最悪の場合、±7% (70,000ppm)です。
また、IOSC内蔵発振周波数-温度特性グラフも掲載されており、上昇方向の傾斜を持つ直線的なグラフから、温度が上がるにつれ周波数も上がることが分かります。

IOSCクロックの周波数はCLGTRIM.IOSCAJ[5:0]ビットを使用してトリミング(調整)が可能です。また、CLGIOSC.IOSCSTMビットに1を書き込むことにより、周波数を自動的に調整するオートトリミング機能を実行することもできます。

1.3 OSC1周波数誤差

OSC1は外付けの音叉型水晶振動子を使用する発振回路で、公称周波数は32.768kHz、25°Cの環境温度における標準偏差は±20ppmです。音叉型水晶振動子の周波数-温度特性は、下図のように25°Cをピークとする(25°Cを中心として周波数が反転する)逆放物線になります(25°Cで偏差が0ppmとなる標準的な水晶振動子の場合)。

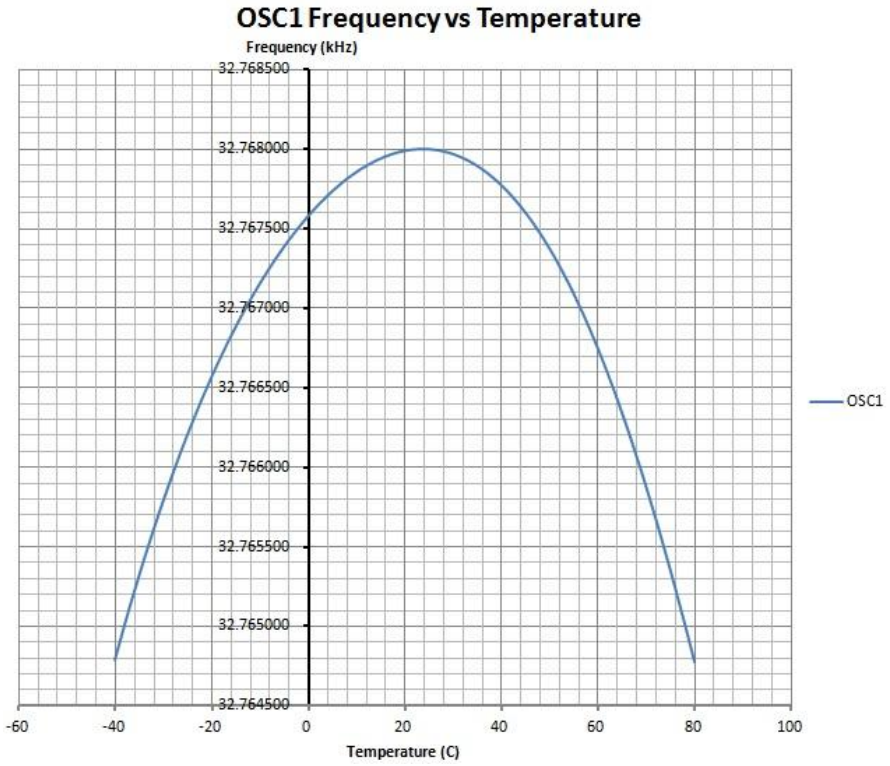

図1-2　OSC1周波数-温度特性

2. 温度センサの校正

2.1 温度センサ特性

S1C17W18は温度センサ/基準電圧生成回路(TSRVR)を内蔵しています。TSRVRは、温度に従って直線的に変化するセンサ出力電圧を生成します。この電圧は、内蔵の12ビットA/D変換器に入力され、A/D変換器に供給される基準電圧に応じたデジタル値に変換されます。TSRVRの構成図はS1C17W18テクニカルマニュアルの“温度センサ/基準電圧生成回路(TSRVR)”の章を参照してください。

S1C17W18テクニカルマニュアルの“電気的特性”の章内、“温度センサ/基準電圧生成回路(TSRVR)特性”の節に温度センサ出力電圧-温度特性グラフが掲載されています。このグラフから、センサ出力電圧と温度には直線的な関係があることが分かります。グラフの傾斜を表す値が電圧温度係数で、ΔV_{TEMP}で示されます。電圧温度係数のTyp.値は3.6mV/°Cです。次の式により、基準温度T_{REF}における既知の電圧V_{TREF}を基に、温度センサ出力電圧から温度を算出できます。

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} \qquad (式2-1)$$

ここで、T_{SEN}は算出される環境温度(°C)、V_{TSEN}は温度がT_{SEN}の場合に温度センサが出力する電圧(V)です。

温度センサが出力する電圧V_Tは、次の式から算出できます。

$$V_T = \frac{ADC}{4096} * V_{REFA} \qquad (式2-2)$$

ここで、V_Tは温度Tにおける温度センサ出力電圧、ADCは温度がTの場合にA/D変換器が変換した値、V_{REFA}はA/D変換器に供給されている基準電圧です。25°Cの室温において温度センサが出力する電圧値は1.04V～1.11Vの範囲で、Typ.値は1.07Vです。
TSRVRは、下表に示すTSRVR*n*VCTL.VREFAMD[1:0]ビットの設定により、基準電圧を内蔵のA/D変換器に供給することができます。

表2-1　TSRVRが出力する基準電圧

TSRVR*n*VCTL.VREFAMD[1:0]ビット	Vref電圧
0x3	2.5V
0x2	2.0V
0x1	V_{DD}
0x0	Hi-Z (外部印加可能)

“算出例”の節に示すとおり、電圧温度係数をTyp.値の3.6mV/°Cとした場合、-40°C～85°Cの動作温度範囲の温度センサ出力は0.830V～1.290Vになります。また、Vrefを2.0Vに設定した場合、温度センサが温度変化を検出する分解能(最小温度差)は、およそ0.3°Cです。

S1C17W18は電圧の代わりにA/D変換値を読み出すため式2-1は、現在のA/D変換値(ADC_{TSEN})、基準温度のA/D変換値(ADC_{TREF})、基準温度(T_{REF})から温度(T_{SEN})を計算する次の式に書き換えることができます。

$$T_{SEN} = \frac{(ADC_{TSEN} - ADC_{TREF})}{\Delta ADC_{TEMP}} + T_{REF} \qquad (式2-3)$$

ここで、ΔADC_{TEMP}はA/D変換器の温度係数(A/D変換値-温度特性の傾き)です。ΔADC_{TEMP}の値は次の式から求められます。

$$\Delta ADC_{TEMP} = \frac{(\frac{\Delta V_{TEMP}}{1000})}{Vref} * 4096 \qquad (式2-4)$$

電圧温度係数をTyp.値の3.6mV/°C、25°Cにおける温度センサ出力電圧を1.07Vとした場合の、2種類のVref電圧に対応したΔADC_{TEMP}の期待値を表2-2に示します。

表2-2　ΔADC_{TEMP}公称値

Vref	ΔADC_{TEMP}
2.0V	7.3728
2.5V	5.89824

温度計算に整数乗除算を使用する場合、T_{SEN}の式は次のように記述できます。

$$\boldsymbol{T_{SEN}} = \frac{(\boldsymbol{ADC_{TSEN}} - \boldsymbol{ADC_{TREF}})}{\boldsymbol{M}} * \boldsymbol{65536} + \boldsymbol{T_{REF}} \qquad (式2-5)$$

ここで、M = ΔADC_{TEMP} * 65536です。

2種類のVref電圧に対応したMの期待値を表2-3に示します。

表2-3　M公称値(ΔADC_{TEMP} * 65536)

Vref	M
2.0V	483184
2.5V	386547

本アプリケーションノートのサンプルソフトウェアは､式2-5を使用してA/D変換値から温度を算出します。

2.2　算出例

以下の例では、電圧温度係数を動作温度範囲全体にわたり3.6mV/°C、25°Cの室温における基準電圧を1.07Vと想定しています。温度センサが出力可能な電圧の範囲は以下のように計算できます。

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF}$$

$$V_{TSEN} = \frac{T_{SEN} - T_{REF}}{1000} * \Delta V_{TEMP} + V_{TREF}$$

最小出力電圧と最大出力電圧は、T_{SEN}をそれぞれ最低温度の-40°C、最高温度の85°Cに置き換えることにより求められます。

$$V_{TSEN,MIN} = \frac{(-40\,°C) - 25\,°C}{1000} * 3.6\frac{mV}{°C} + 1.07\,V = 0.836\,V$$

$$V_{TSEN,MAX} = \frac{(85\,°C) - 25\,°C}{1000} * 3.6\frac{mV}{°C} + 1.07\,V = 1.286\,V$$

したがって、温度センサ出力電圧の範囲は、おおよそ0.83V～1.29Vと見込まれます。ただし、デバイスによる電圧温度係数や25°Cにおける出力電圧の違いにより、これらの値はわずかに変化します。

ここでは再度、温度範囲全体にわたり電圧温度係数をTyp.値の3.6mV/°C、25°Cの室温における校正電圧を1.07V、A/D変換器に供給されている基準電圧を2.0Vとして、温度センサの分解能(検出可能な最小の温度差)を算出します。この室温におけるA/D変換値は次のように求められます。

$$V_T = \frac{ADC}{4096} * V_{REFA}$$

$$ADC = \frac{4096}{V_{REFA}} * V_T = \frac{4096}{2.0\,V} * 1.07\,V = 2191.36 \searrow 2191$$

温度センサが検出可能な温度の最小変化は、A/D変換値が+1または-1となって現れます。この新たなA/D変換値を使用することによって、対応する温度が次のように計算できます。

$$V_T = \frac{ADC}{4096} * V_{REFA} = \frac{2190}{4096} * 2.0\,V = 1.0693\,V$$

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} = \frac{(1.0693\,V - 1.07\,V) * 1000}{3.6\ {}^{mV}/_{°C}} + 25\,°C = 24.8055\,°C$$

元のA/D変換値は2191.36から2191に小数点以下が切り捨てられているため、これによってわずかな誤差が生じ、2191のA/D変換値から得られる温度は十分な精度とは言えません。この誤差を最小限に抑えるため、2192のA/D変換値に相当する温度も計算し、2つの値を使用して2191からの温度差の平均を取ります。

$$V_T = \frac{ADC}{4096} * V_{REFA} = \frac{2192}{4096} * 2.0\,V = 1.0703\,V$$

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} = \frac{(1.0703\,V - 1.07\,V) * 1000}{3.6\ {}^{mV}/_{°C}} + 25\,°C = 25.0833\,°C$$

$$\Delta T_1 = 25\,°C - 24.8055\,°C = 0.1945\,°C$$

$$\Delta T_2 = 25.0833\,°C - 25\,°C = 0.0833\,°C$$

$$\Delta T_{AVG} = \frac{0.1945\,°C + 0.0833\,°C}{2} = 0.1389\,°C$$

したがって、温度センサは0.2°C以内の温度差を検出可能と言うことができます。
異なる基準電圧では異なるA/D変換値が得られますが、温度変化にそれほどの違いはありません。

2.3 校正方法

製造公差/バラツキなどにより、$\mathbf{T_{REF}}$における**M**と$\mathbf{ADC_{REF}}$はデバイスにより変わります。したがって、高い温度測定精度を得るには、温度センサの校正が必要です。測定のキーパラメータは**M**(A/D変換値-温度特性の傾き)です。$\mathbf{T_{REF}}$における$\mathbf{ADC_{REF}}$は通常、室温(25°C)で記録されます。

Mを決定するには、いろいろな方法があります。最も分かりやすい方法は、製品の動作温度範囲全体のA/D変換値を小さな単位(たとえば1°Cステップ)で記録し、高度なデータ解析(最小二乗法など)を行ってA/D変換値-温度特性の傾向線を得ることです。もう少し実用的で簡単な方法は、動作温度範囲両端(最低および最高温度)のA/D変換値のみを記録し、この2点のデータを基に傾き(**M**)を計算することです。使用する方法は、必要な精度と実用性/校正コストの兼ね合いを検討して決定します。

以下に、量産時の温度センサ校正方法をいくつか紹介します。

方法1: 製造ラインにおける全温度範囲の特性評価

製造ラインの製品個々に、その動作温度範囲の全体にわたりA/D変換値を記録します。特性の傾斜(**M**)は、そのデータから最小二乗法などにより計算します。$\mathbf{T_{REF}}$における$\mathbf{ADC_{REF}}$を選択し、**M**、$\mathbf{ADC_{REF}}$、および$\mathbf{T_{REF}}$値をEEPROMに書き込みます。出荷後の通常動作時は、式2-5により温度を計算します。この方法は最も正確です。ただし、製造ラインの資源(検査時間や機器)に関しては最も費用がかかり、多くの製品に対して現実的とは言えません。

方法2: 同一のM、ADC_{REF}、T_{REF}値を全製品に適用

この方法では、製造ラインの製品をすべて同一の**M**、$\mathbf{ADC_{REF}}$、$\mathbf{T_{REF}}$の値でプログラムします。出荷後の通常動作時は、式2-5により温度を計算します。製品開発時や動作試験時に、サンプルの製品群を使用して動作温度範囲全体のA/D変換値-温度特性を計測し、全製品に適用する平均的な**M**、$\mathbf{ADC_{REF}}$、$\mathbf{T_{REF}}$の値を決定します。

方法3: 室温での測定結果 + M平均値

この方法では、方法2によって製品すべてに共通の**M**(傾斜)の平均値を算出します。一方、室温($\mathbf{T_{REF}}$)における$\mathbf{ADC_{REF}}$の値は製品個々に記録し、固定の**M**とともにEEPROMに書き込みます。

2.4 個体の校正例

次のグラフは、SVTmini17W18ボードで校正したA/D変換値-温度特性の例です。

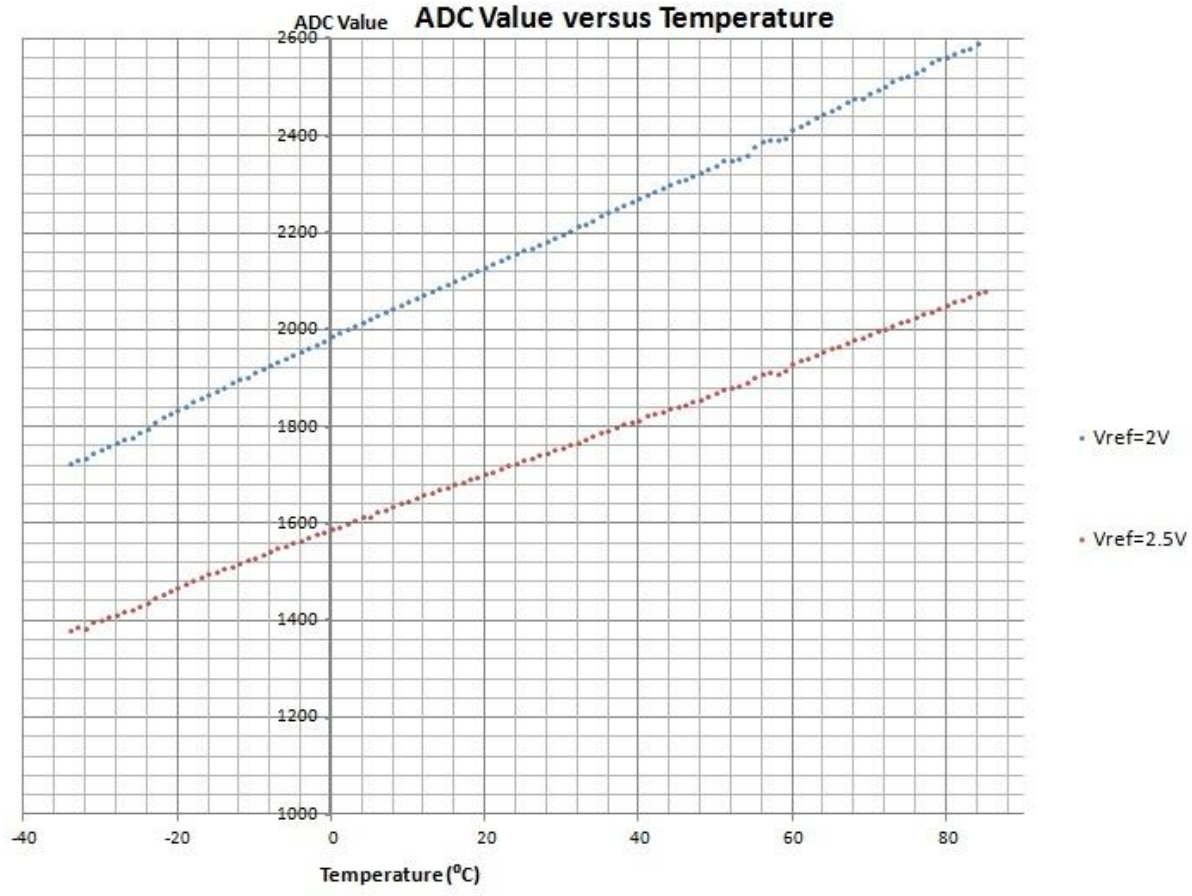

図2-1　A/D変換値-温度特性グラフの例

このグラフはVref = 2VおよびVref = 2.5VのA/D変換値-温度特性を示しています。下記の傾斜値は最低温度と最高温度の差を測定して計算したものです。$\mathbf{ADC_{REF}}$は$\mathbf{T_{REF}}$ = 30°Cにおける値です。

表2-4　ΔADC_{TEMP}とMの測定値

Vref	ΔADC_{TEMP}	M	ADC_{REF} @T_{REF} = 30°C
2.0V	7.361344	482433	2200
2.5V	5.882353	385506	1759

この**M**、$\mathbf{ADC_{REF}}$、$\mathbf{T_{REF}}$の値は、サンプルソフトウェアでの温度計算に使用しています。また、使用したボードは温度槽で再試験を行い、ソフトウェアの温度計算結果と実際の温度を比較しています。この結果を、Vref = 2VとVref = 2.5Vそれぞれについて表2-5に示します。

表2-5 計算値と実際の温度との比較結果

	Vref = 2.0V			Vref = 2.5V		
温度(°C)	A/D変換値	算出温度(°C)	温度誤差(°C)	A/D変換値	算出温度(°C)	温度誤差(°C)
-35	1734	-35.0	0.0	1385	-35.3	-0.3
-30	1772	-29.9	0.1	1415	-29.8	0.2
-25	1808	-24.9	0.1	1444	-25.0	0.0
-20	1837	-20.2	-0.2	1472	-19.9	0.1
-15	1881	-14.8	0.2	1505	-15.0	0.0
-10	1913	-10.3	-0.3	1529	-10.3	-0.3
-5	1952	-5.2	-0.2	1560	-5.0	0.0
0	1990	0.2	0.2	1590	0.0	0.0
5	2025	5.1	0.1	1619	5.0	0.0
10	2066	10.2	0.2	1647	9.8	-0.2
15	2098	14.6	-0.4	1677	15.0	0.0
20	2135	19.9	-0.1	1705	19.8	-0.2
25	2164	24.8	-0.2	1727	24.8	-0.2
30	2200	30.0	0.0	1759	29.8	-0.2
35	2239	35.2	0.2	1789	35.0	0.0
40	2274	40.2	0.2	1817	39.8	-0.2
45	2312	45.2	0.2	1843	45.1	0.1
50	2343	49.9	-0.1	1873	49.9	-0.1
55	2383	55.1	0.1	1903	55.2	0.2
60	2416	60.0	0.0	1931	60.0	0.0
65	2448	65.1	0.1	1958	64.9	-0.1
70	2489	70.0	0.0	1991	70.1	0.1
75	2526	74.9	-0.1	2019	75.1	0.1
80	2564	80.0	0.0	2051	80.0	0.0
85	2601	85.0	0.0	2079	85.0	0.0

個体の校正例のまとめ

動作温度範囲の両端のデータのみを基に決定した**M**の値を使用しても、算出された温度はVref = 2Vおよび Vref = 2.5Vのどちらの場合も±0.5°Cの精度に収まることを、上記校正例は示しています。したがって、かなり正確な温度測定が必要な場合でも、恐らく校正方法2と3で十分と言えます。また、室温近辺の3箇所(たとえば、20°C、24°C、28°C)のA/D変換値のみから**M**、$\mathbf{ADC_{REF}}$、$\mathbf{T_{REF}}$の値を得て温度計算に使用しても十分かもしれません。

3. 周波数誤差補償

各クロックソースは、周波数誤差を補償する機能やアルゴリズムを持っています。本章では、それらについて、個別に説明します。

3.1 OSC3周波数誤差補償

3.1.1 OSC3周波数特性と調整

OSC3内蔵発振回路の周波数はCLGTRIM.OSC3AJ[4:0]ビットを使用してトリミング(調整)することができます。これらのビットを最大値および最小値に設定した場合の各OSC3クロック周波数(250kHz、384kHz、500kHz、1MHz、2MHz、4MHz)の対温度特性を以下に示します。

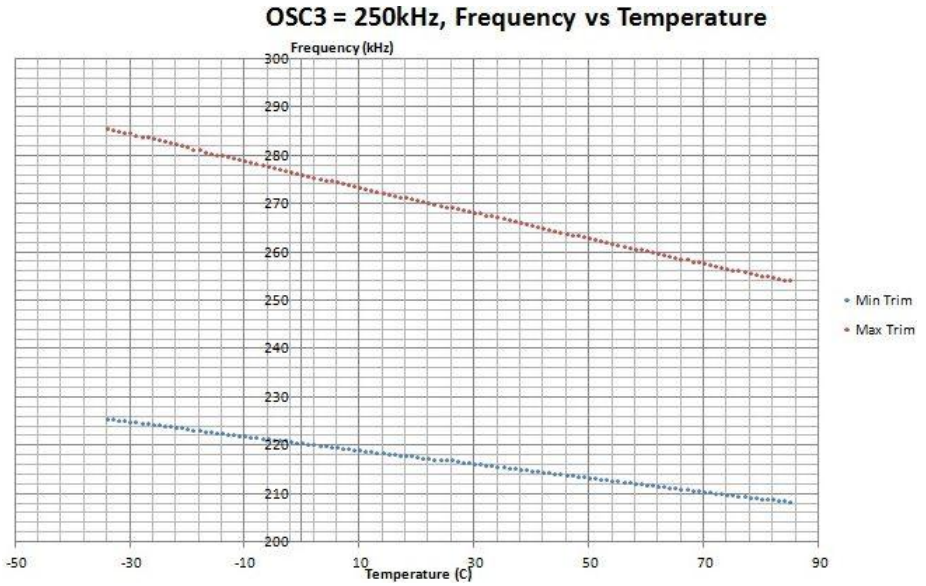

図3-1　最大/最小トリミング設定時の250kHz OSC3周波数-温度特性

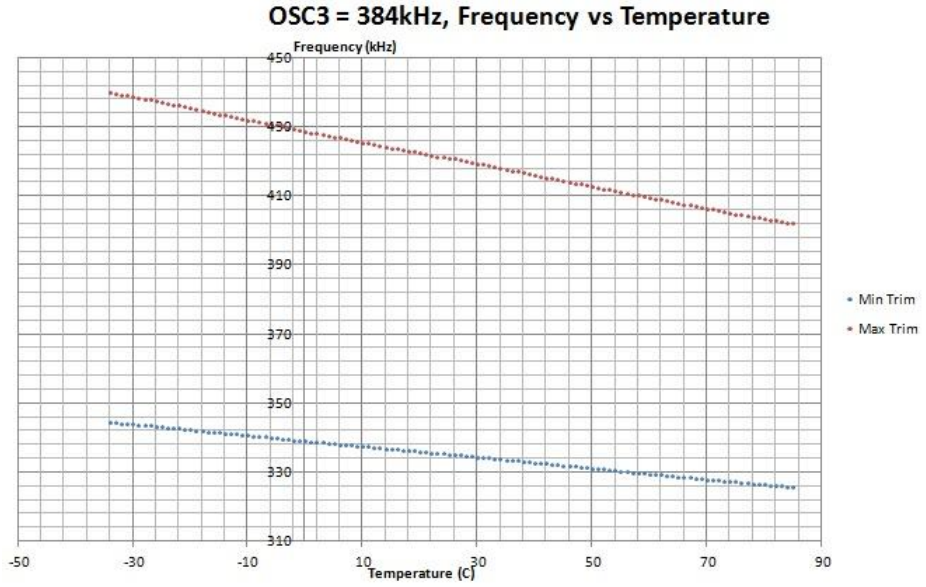

図3-2　最大/最小トリミング設定時の384kHz OSC3周波数-温度特性

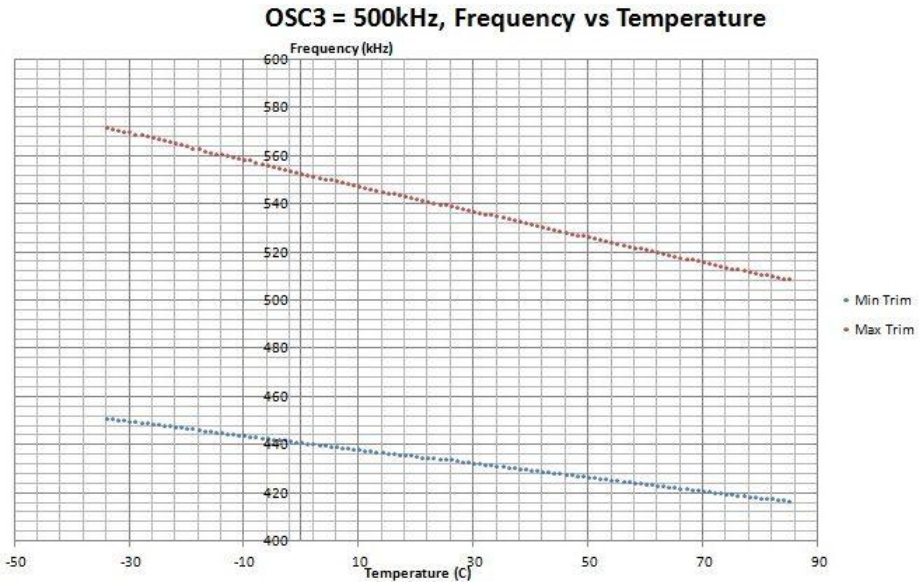

図3-3　最大/最小トリミング設定時の500kHz OSC3周波数-温度特性

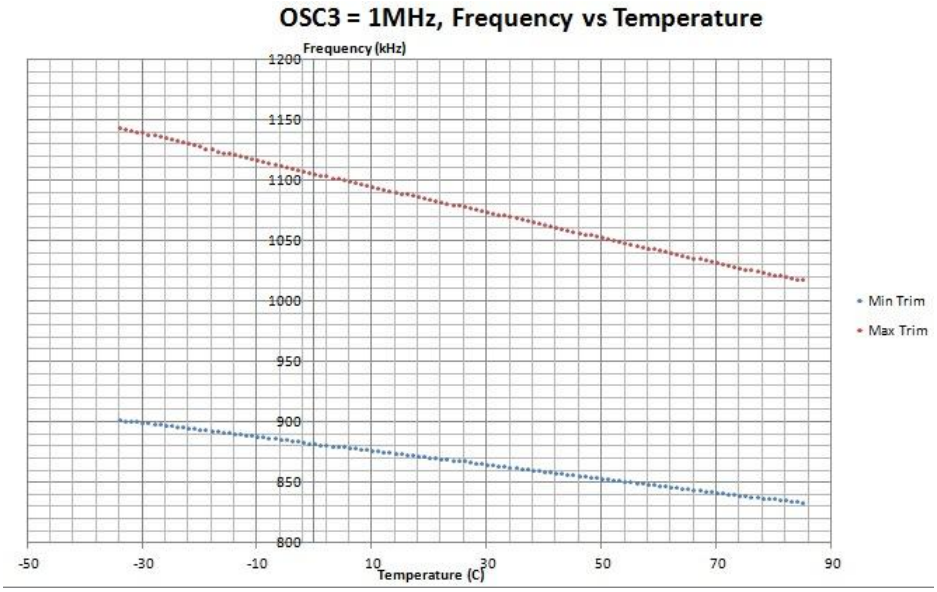

図3-4　最大/最小トリミング設定時の1MHz OSC3周波数-温度特性

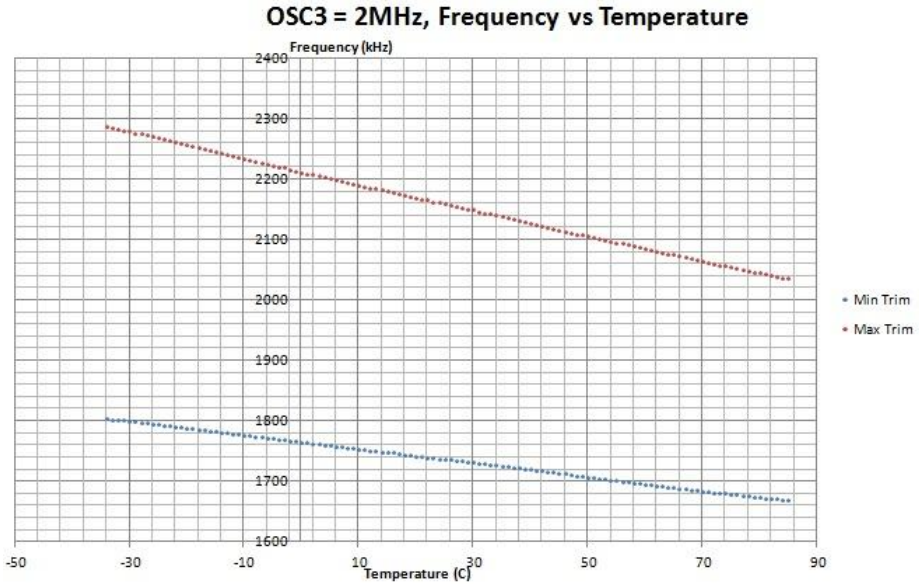

図3-5　最大/最小トリミング設定時の2MHz OSC3周波数-温度特性

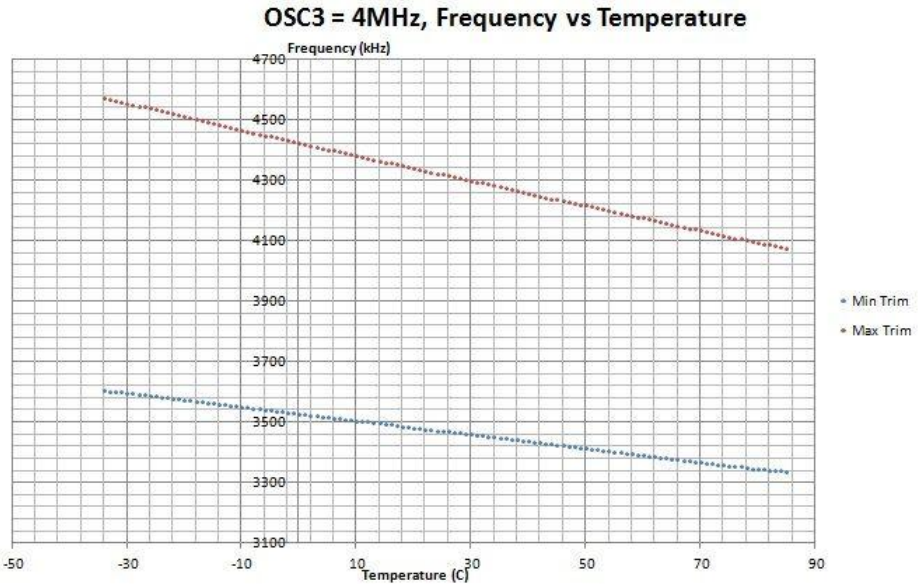

図3-6　最大/最小トリミング設定時の4MHz OSC3周波数-温度特性

OSC3周波数は、FOUT出力をポートに割り当て、FOUTのクロックソースにOSC3を選択することにより、S1C17W18のP16ポート上で測定できます。

これらのグラフから分かるとおり、CLGTRIM.OSC3AJ[4:0]ビットで設定するOSC3トリミング回路は、OSC3周波数を動作温度範囲全体にわたり、Typ.値に調整するのに十分な能力を持っています。

3.1.2 OSC1を使用するOSC3のソフトウェアトリミング

このソフトウェア補償方法は、OSC1をクロックソースとして使用し、OSC3周波数を測定します。その結果を基にして、公称周波数からの誤差が最小となるように、反復ループでCLGTRIM.OSC3AJ[4:0]ビットのトリミング値を調整します。高精度のOSC1クロック(300ppmまたは0.03%未満の周波数誤差)を使用して、OSC3クロックの精度(最大10%の周波数誤差)を改善(1%以下の周波数誤差に)します。

この方法では、16ビットタイマ(T16)を2チャネル使用します。一方のタイマをOSC1クロックで、もう一方をOSC3クロックで動作させます(以降、OSC1タイマとOSC3タイマ)。OSC1タイマは*m*クロック周期で割り込みを発生するようにプログラムします。OSC3タイマはカウンタを0にクリアし、両方のカウンタをスタートさせます。OSC1タイマの割り込み(*m*クロック経過時)が発生したところでOSC3タイマを停止させ、そのカウンタ値*n′*を読み出して*n*(OSC3タイマのスタート後のカウント数0x10000 - *n′*)を取得します。*m*と*n*、およびOSC1周波数(typ. 32.768kHz)から、OSC3周波数は次の式で計算できます。

$$Fosc3 = \frac{(\boldsymbol{n} * Fosc1)}{\boldsymbol{m}} \qquad (式3-1)$$

測定したFosc3と公称周波数をソフトウェアで比較します。この結果を基にトリミング値(CLGTRIM.OSC3AJ[4:0]ビット)を調整し、上記の周波数測定を繰り返します。この反復ループを、周波数誤差が最小になるまで実行します。

ソフトウェアの待ち時間およびカウンタ値の取り込み誤差による周波数測定誤差も、次の式のように*nerr*として見込んでおく必要があります。

$$Fosc3 = \frac{\big((\boldsymbol{n} + nerr) * Fosc1\big)}{\boldsymbol{m}} \qquad (式3-2)$$

*nerr*の影響を最小に抑えるには、*n*(および*m*)を所望の測定誤差(*nerr*/*n*)仕様を満たす十分に大きな値にする必要があります。

*m*と*n*値の計算例

Fosc3周波数が250kHz、*nerr*を10(Fosc3クロックサイクル数)とした場合に、*m*と*n*値を変えて計算した*nerr*/*n*比の例を表3-1に示します。

表3-1 OSC1を使用してOSC3周波数を測定するための*m*と*n*値の計算例

Fosc1公称周波数(Hz)		32768
nerr		10
Fosc3		250000
Fosc3/Fosc1		7.63
m	***n***	***nerr/n***
10	76.29	13.107%
100	762.94	1.311%
1000	7629.39	0.131%
8000	61035.16	0.016%

*m*を100(最大3msの測定時間)～1000(最大30msの測定時間)の間に設定すると、*nerr*/*n*比はおおむね1%程度になります。

3.1.3 個体のOSC3ソフトウェアトリミング例

3.1.2節で説明した方法を採用したサンプルソフトウェアを使用して、-35°C～85°Cの温度範囲内の何点かでSVTmini17W18ボード上のOSC3周波数の調整と測定を行いました。各OSC3周波数をOSC1を使用して測定した際に使用した*m*と*n*の値を表3-2に示します。

表3-2　各公称周波数でOSC3の調整に使用した*m*と*n*値の例

Fosc3	m	n
250kHz	8000	61035
384kHz	5000	58593
500kHz	4000	61035
1MHz	2000	61035
2MHz	1000	61035
4MHz	500	61035

表3-3に測定結果を示します。

表3-3　OSC3調整結果

	Fosc3 (kHz)						誤差					
温度(°C)	250kHz	384kHz	500kHz	1MHz	2MHz	4MHz	250kHz	384kHz	500kHz	1MHz	2MHz	4MHz
-35	248.2091	381.2728	496.2003	0.993968	1.990044	3.978941	-0.72%	-0.71%	-0.76%	-0.60%	-0.50%	-0.53%
-30	248.3115	381.1763	497.29	0.997008	1.981189	3.96682	-0.68%	-0.74%	-0.54%	-0.30%	-0.94%	-0.83%
-25	247.4197	382.3204	496.4046	0.994296	1.990685	3.953357	-1.03%	-0.44%	-0.72%	-0.57%	-0.47%	-1.17%
-20	247.6933	381.46	497.8195	0.997921	1.984484	3.969056	-0.92%	-0.66%	-0.44%	-0.21%	-0.78%	-0.77%
-15	248.4112	381.0009	496.3384	0.994126	1.990439	3.981598	-0.64%	-0.78%	-0.73%	-0.59%	-0.48%	-0.46%
-10	247.4713	382.0398	497.3162	0.996988	1.996818	3.96672	-1.01%	-0.51%	-0.54%	-0.30%	-0.16%	-0.83%
-5	248.1674	381.6902	498.7376	0.993678	1.989273	3.980129	-0.73%	-0.60%	-0.25%	-0.63%	-0.54%	-0.50%
0	247.8297	382.4103	497.2776	0.996172	1.993677	3.964099	-0.87%	-0.41%	-0.54%	-0.38%	-0.32%	-0.90%
5	247.495	381.9617	497.6373	0.997659	1.985905	3.974854	-1.00%	-0.53%	-0.47%	-0.23%	-0.70%	-0.63%
10	247.3466	380.7218	496.3263	0.994347	1.990254	3.959508	-1.06%	-0.85%	-0.73%	-0.57%	-0.49%	-1.01%
15	248.1812	381.3468	497.056	0.996759	1.989227	3.968671	-0.73%	-0.69%	-0.59%	-0.32%	-0.54%	-0.78%
20	248.0652	381.0207	498.7286	0.993477	1.987868	3.97754	-0.77%	-0.78%	-0.25%	-0.65%	-0.61%	-0.56%
25	248.2492	382.249	498.1023	0.996824	1.996932	3.969753	-0.70%	-0.46%	-0.38%	-0.32%	-0.15%	-0.76%
30	247.7843	381.8519	497.9084	0.998062	1.988705	3.952351	-0.89%	-0.56%	-0.42%	-0.19%	-0.56%	-1.19%
35	247.6091	382.7473	496.6247	0.99475	1.991045	3.965723	-0.96%	-0.33%	-0.68%	-0.53%	-0.45%	-0.86%
40	248.1637	381.3272	497.8699	0.996457	1.994292	3.966153	-0.73%	-0.70%	-0.43%	-0.35%	-0.29%	-0.85%
45	248.0597	381.0818	498.3574	0.993356	1.988863	3.977342	-0.78%	-0.76%	-0.33%	-0.66%	-0.56%	-0.57%
50	248.296	382.1071	497.3051	0.99756	1.986639	3.977203	-0.68%	-0.49%	-0.54%	-0.24%	-0.67%	-0.57%
55	248.1846	380.6994	497.8899	0.996613	1.99646	3.97005	-0.73%	-0.86%	-0.42%	-0.34%	-0.18%	-0.75%
60	247.7203	382.4157	496.9384	0.995465	1.992601	3.96304	-0.91%	-0.41%	-0.61%	-0.45%	-0.37%	-0.92%
65	248.0809	381.835	497.5432	0.995668	1.994443	3.96975	-0.77%	-0.56%	-0.49%	-0.43%	-0.28%	-0.76%
70	248.1419	381.0061	498.6667	0.993548	1.989444	3.959367	-0.74%	-0.78%	-0.27%	-0.65%	-0.53%	-1.02%
75	247.7209	381.3081	496.8563	0.99528	1.990856	3.96371	-0.91%	-0.70%	-0.63%	-0.47%	-0.46%	-0.91%
80	248.227	380.7919	497.1257	0.996705	1.990346	3.972165	-0.71%	-0.84%	-0.57%	-0.33%	-0.48%	-0.70%
85	248.2741	381.8569	496.295	0.994263	1.990977	3.962823	-0.69%	-0.56%	-0.74%	-0.57%	-0.45%	-0.93%

表3-3の結果は、3.1.2節で説明したアルゴリズムを使用して適切なトリミング値を設定することにより、OSC3周波数を公称値に近い状態に調整可能であることを示しています。公称周波数に最も近い2つのトリミング設定の周波数の精度は、公称周波数が2つのトリミング設定の間のどの当たりに位置するかによって変わります。表3-3のデータは、最も精度が上がるトリミング設定によって、OSC3クロック周波数がおおむね公称値の-1%以内に収まることを示しています。

3.1.4 補償方法

3.1.4.1 OSC1を使用するOSC3のトリミング

この方法では、3.1.2節で説明したソフトウェアによるトリミングアルゴリズムを使用して、OSC3の周波数誤差を補正します。その後、温度センサを使用して温度の変化をチェックし、周期的に3.1.2節のトリミング動作を行うことで、温度変化による周波数誤差/ドリフトを補償します。

3.1.4.2 校正データによるOSC3のトリミング

この方法では、製品開発時に、OSC3トリミングデータ(公称周波数に最も近づく値)と温度を対応させた6種類のOSC3周波数別校正データを集約して、ファームウェアにルックアップテーブルを組み込んでおきます。

3.2 IOSC周波数誤差補償(オートトリミング)

3.2.1 周波数誤差補正

IOSC内蔵発振回路の周波数は、CLGTRIM.IOSCAJ[5:0]ビットの設定により調整することができます。S1C17W18は、高精度なOSC1クロックを校正用クロックソースとして使用し(3.1.2節のOSC3トリミング方法と同様)、ハードウェアによってIOSC周波数を自動的に調整するIOSCオートトリミング機能も持っています。オートトリミングはCLGIOSC.IOSCSTMビットへの1書き込みにより開始します(このビットはオートトリミングの終了時まで1を保持します)。オートトリミングが終了すると、CLGINTF.IOSCSTAIF割り込みフラグが1にセットされます。

3.2.2 補償方法

電源投入直後に、ハードウェアによるオートトリミングまたは3.1.2節のソフトウェアトリミングアルゴリズムを使用してIOSC周波数誤差を補正します。その後、温度センサを使用して温度の変化をチェックし、周期的にオートトリミングまたは3.1.2節のソフトウェアトリミングを行うことで、温度変化による周波数誤差/ドリフトを補償します。

3.3 OSC1周波数誤差補償

3.3.1 RTC計時誤差の補正

S1C17W18が内蔵するRTC(リアルタイムクロック)は、計時にOSC1クロックを使用し、経過日時を保持します。そのため、OSC1周波数誤差はできるだけ小さくしておく必要があります。たとえば、20ppmの周波数誤差があると、1日に1.7秒、1月に52秒、1年で10.5分の時間のずれが生じます。S1C17W18には、OSC1水晶発振周波数をソフトウェアで調整する機能はありません。ただし、S1C17W00シリーズの全MCUが搭載するRTCには、効果的にOSC1の周波数誤差を補正する、論理緩急と言う周波数誤差補償回路が内蔵されています。

S1C17W00シリーズのRTCは256Hzの内部クロックで動作します。この256Hzクロックは、OSC1クロック(Typ. 32.768kHz)を128分周(32768/128 = 256)して生成されます。また、この256Hzクロックによって0～255のカウントを行う8ビットカウンタが動作します。このカウンタは最大値の255になると0に戻り、RTC1Sパルスを生成します。もしOSC1に周波数誤差がなければ、RTC1Sパルスは正確に1秒周期で生成されます。ただし、発振回路部品の製造バラツキや温度ドリフトにより、OSC1発振周波数は公称周波数の32.768kHzからずれ、RTC1Sパルスは正確に1秒間隔とはなりません。

周波数誤差を補償するため、論理緩急ハードウェアがトリガによって起動すると、RTC1Sの1Hz信号の精度を保持するため8ビットカウンタの値を修正します。たとえば、OSC1発振回路が公称周波数の32.768kHzより低い周波数で発振していた場合、8ビットカウンタは1Hzより長い周期で255になります。これを補正するため、論理緩急ハードウェアがトリガによって起動すると、ソフトウェア(RTCCTL.RTCTRM[6:0]ビット)で設定した値だけ8ビットカウンタの値を増加させ、カウンタが255に達する時間を早めます。なお、論理緩急のトリガは、RTCCTL.RTCTRM[6:0]ビットへの補正値の書き込みによって発行されます。論理緩急動作中かどうかについてはRTCCTL.RTCTRMBSYビットで確認でき、論理緩急が終了するとRTCINTF.RTCTRMIF割り込みフラグが1にセットされます。

RTCCTL.RTCTRM[6:0]ビットには、論理緩急補正値を2の補数表現にして設定します(***cv***)。論理緩急回路は蓄積された計時誤差を(***cv***/256)秒補正します。したがって、論理緩急回路は計時誤差補正回路と言えます。

周波数誤差の結果として累積された計時誤差を計算する式に必要な項を、ここで説明しておきます。

- T_s = 周波数誤差のチェックと温度の検出を行う理論上のサンプリング間隔(秒単位の整数)
- $f_a(i)$ = サンプル'*i*'における実際の周波数
- f_0 = OSC1公称周波数 = 32768kHz
- $f_{err}(i)$ = サンプル'*i*'における周波数誤差、$f_{err}(i) = f_a(i) - f_0$。通例では、$f_a(i) > f_0$であれば$f_{err}(i)$を正数とみなす
- $T_{err}(i)$ = サンプル'*i*'における、1つ前のサンプリングからの計時誤差
- $T_a(i)$ = サンプル'*i*'における、1つ前のサンプリングからの実際の経過時間、$T_a(i) = T_s + T_{err}(i)$
- $E_a(n)$ = '*n*'回のサンプリングが終了するまでにかかった実際の時間
- $E_0(n)$ = '*n*'回のサンプリングが終了するまでにかかる理論上の時間、$E_0(n) = n*T_s$
- $E_{err}(n)$ = '*n*'回のサンプリングが終了した時点の累積計時誤差

'*n*'回のサンプリングが終了するまでにかかった実際の時間$E_a(n)$を求める式から、'*n*'回のサンプリングが終了した時点の累積計時誤差$E_{err}(n)$を導き出せます。

$$E_a(n) = \sum_1^n Ta(i) = \sum_1^n (Ts + Terr(i)) = (n * Ts) + \sum_1^n (Terr(i))$$
$$= E_0(n) + E_{err}(n)$$

$$E_0(n) = n * T_s$$

$$\boldsymbol{E_{err}(n) = \sum_1^n (Terr(i))} \quad (式\mathbf{3-3})$$

サンプル'*i*'における実際のサンプリング間の経過時間を求める式$T_a(i)$から、サンプリング間の計時誤差$T_{err}(i)$を導き出せます。

$$Ta(i) = Ts + Terr(i) = Ts * \left(\frac{fo}{fa(i)}\right)$$

$$Terr(i) = Ts * \left(\left(\frac{fo}{fa(i)}\right) - 1\right)$$

$$\boldsymbol{Terr(i) = Ts * \left(\frac{fo - fa(i)}{fa(i)}\right) = -Ts * \frac{fa(i) - fo}{fa(i)} = -Ts * \frac{ferr(i)}{fo + ferr(i)}} \quad (式\mathbf{3-4})$$

式3-4を式3-3に代入することにより、'*n*'回のサンプリングが終了した時点の累積計時誤差を算出する式は、次のようになります。

$$\boldsymbol{E_{err}(n)} = \sum_1^n \left(\boldsymbol{Terr(i)}\right) = -\boldsymbol{Ts} * \sum_1^n \left(\frac{ferr(i)}{fo+ferr(i)}\right) \quad (式\mathbf{3-5})$$

補正値 ***cv*** の符号は、累積計時誤差の符号と同じです。クロック周波数が公称値より低い場合、ferr(i)の符号は負になります。したがって、計時誤差が正となり、補正値 cv も 256 サイクルの 1 秒カウンタを速めるため正となります。クロック周波数が公称値より高い場合、$\boldsymbol{f_{err}(i)}$の符号は正になります。したがって、計時誤差が負となり、補正値 ***cv*** も 256 サイクルの 1 秒カウンタを遅らせるため負となります。

サンプル***n***における補正値***cv(n)***は、累積計時誤差から次の式により求められます。

$$\left(\frac{cv(n)}{256}\right) = Eerr(t) = -Ts * \sum_1^n \left(\frac{ferr(i)}{fo + ferr(i)}\right)$$

$$cv(n) = -256 * Ts * \sum_1^n \left(\frac{ferr(i)}{fo + ferr(i)}\right)$$

総和の項は小さな分数で、追跡は容易ではありませんので以下のように計算を簡略化します。
$\boldsymbol{f_0} >> \boldsymbol{f_{err}(i)}$ ($\boldsymbol{f_0}$は$\boldsymbol{f_{err}(i)}$より十分に大きい)のため、

$$(fo + ferr(i)) \cong fo$$

とすると、***cv***は次の式で近似できます。

$$\boldsymbol{cv(n)} \cong -\frac{\boldsymbol{256 * Ts}}{\boldsymbol{fo}} * \sum_{\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{ferr(i)} = -\frac{\boldsymbol{256 * Ts}}{\boldsymbol{32768}} * \sum_{\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{ferr(i)} = -\frac{\boldsymbol{Ts}}{\boldsymbol{128}} * \sum_{\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{ferr(i)} \quad (式\mathbf{3-6})$$

論理緩急による補正を実行後は、この補正分を加味し、累積周波数誤差の総和$\sum_1^n ferr(i)$を、次のように修正してください。

$$\sum_1^n ferr(i) \ <== \sum_1^n ferr(i) \ -(-\frac{cv * 32768}{256 * Ts})$$

$$\sum_1^n ferr(i) \ <== \sum_1^n ferr(i) \ + \frac{\boldsymbol{cv * 128}}{\boldsymbol{Ts}} \quad (式\mathbf{3-7})$$

論理緩急による補正を使用して温度補償付き計時誤差補償を実行するには、製品開発時に校正を行い、計時誤差の計算に使用する温度別の周波数誤差を記録したルックアップテーブルを作成しておく必要があります。

3.3.2 RTCの温度補償付き補正方法

論理緩急を使用して温度補償付き計時誤差補正を行うための推奨アルゴリズムを、以下に説明します。

1) 使用するサンプリング間隔/周期T_sを決定します。サンプリング周期を長く取ると、消費電力を抑えられます。ただし、短いサンプリング周期の場合に比べ、周波数誤差の変動をソフトウェアによって正確に検出できないことが考えられます。したがって、消費電力と温度変化による周波数変動の検出精度の兼ね合いを検討する必要があります。

2) 現在の周波数誤差、前回の周波数誤差、累積周波数誤差を扱うための3つの変数、*freqerr*、*prev_freqerr*、*freqerrsum*を定義します

3) 以下の初期化を行います。
   a. *freqerrsum*を0に設定します。
   b. 温度を読み出します。
   c. 得られた温度に対応する周波数誤差を、ルックアップテーブルから読み出します。
   d. 読み出した値を*prev_freqerr*に代入します。

4) T_s秒ごとに割り込みが発生するようにRTCをプログラムします。これがサンプリング間隔/周期になります。

5) RTC割り込みが発生したところで、以下の処理を行います。
   a. 温度を読み出します。
   b. 得られた温度に対応する周波数誤差を、ルックアップテーブルから読み出します。
   c. 読み出した値を*freqerr*に代入します。
      これは現在の周波数誤差です。
   d. *freqerr* = (*freqerr* + *prev_freqerr*) / 2
      これは前回と現在の周波数誤差の中間値を求める式で、2点のサンプリング間における周波数誤差のおおよその平均値が得られます。
   e. *freqerrsum* = *freqerrsum* + *freqerr*
      累積周波数誤差を更新します。
   f. 式3-6から補正値*cv*を計算します。
   g. 以下を実行します。
      - 論理緩急補正を開始させます。
      - 補正終了後、式3-7を使用して*freqerrsum*を更新します。
      - ステップ4に移行します。

3.3.3 室温における個体の計時誤差補正の例

サンプルソフトウェアは、3.3.2節で説明した温度補償付き計時誤差補正アルゴリズムを実装しています。テストは、室温において約+45ppmのOSC1周波数誤差を持つSVTmini17W18ボードで行いました。サンプリング周期T_sは15分(900秒)に設定しています。OSC1周波数-温度特性は図3-6のように測定されました。これを基に周波数誤差-温度ルックアップテーブルを作成し、温度補償付き計時誤差補正アルゴリズムのために使用しています。

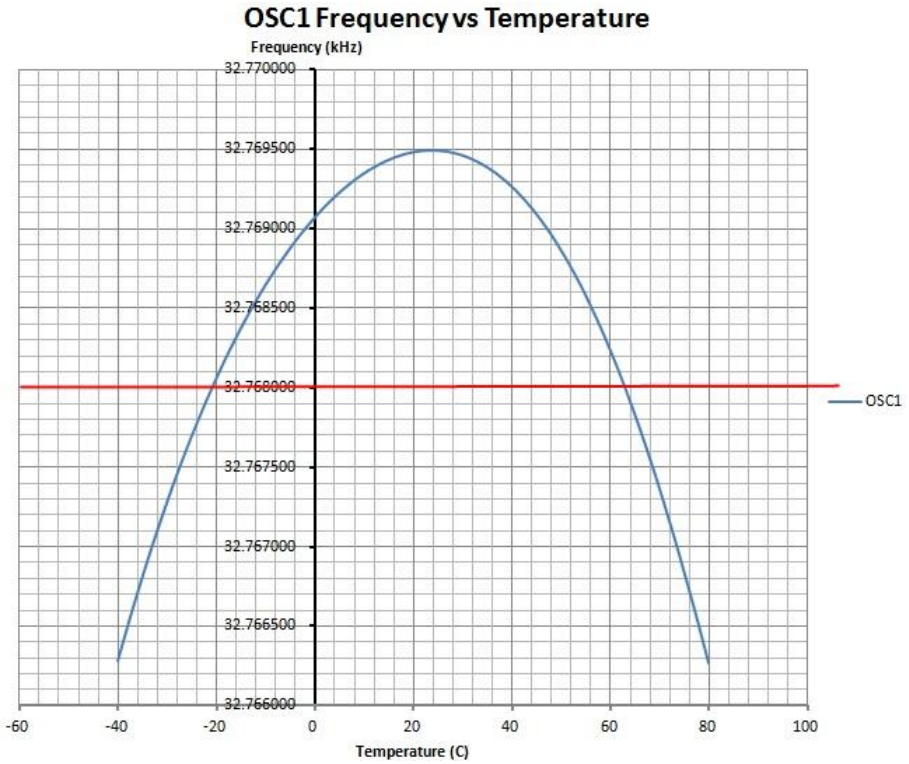

図3-6 OSC1周波数-温度特性グラフ

テストは、S1C17W18サンプルソフトウェアによって表示された日時とWEBサイト“time.gov”で表示された時刻を視覚的に(目視で)捉え、3日間記録しました。このため、約±0.5秒の測定誤差があります。表3-4にテスト結果を示します。

表3-4 計時誤差補正結果の例

“time.gov” (原子時計)		S1C17W18の時間		“time.gov”	S1C17W18	計時誤差	補正なしの	補正時の推定計時
日	時刻	日	時刻	経過秒数	経過秒数	(秒)	推定計時誤差(秒)	誤差(ppm)
12/18/2015	15:00:29.2	0	00:01:00.0	0.0	0.0	0.0	0.0	***0.00***
12/19/2015	15:33:00.1	1	00:33:31.0	88350.9	88351.0	0.1	4.0	***1.13***
12/20/2015	23:07:29.0	2	08:08:00.0	202019.8	202020.0	0.2	9.1	***0.99***
12/21/2015	15:00:29.0	3	00:01:00.0	259199.8	259200.1	0.3	11.7	***1.16***

+45ppmの周波数誤差のまま、補正なしで3日間計時を行った場合の推定計時誤差は約11.7秒です。補正を行うと、計時誤差は1秒未満に減少し、ppm誤差はおよそ+1ppmになります。

計時誤差補正アルゴリズムをさらに厳密にテストするには、被試験器を恒温槽に置き、温度の上昇/下降サイクルも通した方が確かです。また、長いサンプリング周期は短いサンプリング周期に比べ、補正後に高いppm誤差になるということを確認するため、データはサンプリング設定を変えて集めてください。

3.3.4 OSC1周波数誤差校正方法

3.3.2節で説明した推奨アルゴリズムを使用して温度補償付き計時誤差補正を行うには、温度別の周波数誤差を記録したルックアップテーブルが必要です。このテーブルを作成するには、校正が必要となります。

方法1: 製造ラインにおける全温度範囲の特性評価

製造ラインの製品個々に、その動作温度範囲全体をとおして周波数誤差を記録します。そのデータから周波数誤差-温度ルックアップテーブルを作成してEEPROMに書き込みます。
これは、水晶発振回路部品の製造公差と温度ドリフトも考慮に入れた最も正確な方法です。ただし、製造ラインの資源(検査時間や機器)に関しては最も費用がかかり、多くの製品に対して現実的とは言えません。

方法2: 同一の周波数誤差データを全製品に適用

この方法では、製造ラインの製品をすべて同一の周波数誤差-温度データルックアップテーブルを使用してプログラムします。製品開発時や動作試験時に、サンプルの製品群を使用して動作温度範囲全体の周波数誤差を測定し、その平均値を基に全製品に適用する周波数誤差-温度ルックアップデータを作成します。

方法3: 方法2 + 室温オフセット

この方法では、室温における周波数誤差を、生産ラインの製品個々に測定します。ここで測定した誤差と方法2で作成したルックアップテーブルの室温に対応した周波数誤差との差を取り、これをオフセット値としてルックアップテーブルのすべての誤差値に加えて使用します。周波数誤差-温度特性曲線の形は全製品に共通で、デバイスごとの違いにより曲線が上下します。

4. サンプルソフトウェア

4.1 動作

本アプリケーションノートには、温度校正と周波数誤差補償アルゴリズムの実証と、アプリケーションの容易な作成を支援するためのサンプルソフトウェアが用意されています。評価ボードは、UM232H (USB-シリアル変換モジュール)などの外部シリアルインタフェースボードを使用して、UART経由でPCに接続できます。UARTインタフェースは、転送速度 = 38400bps、パリティ = なし、スタートビット = 1ビット, ストップビット = 1ビットに設定されます。S1C17W18のP32ポートとP33ポートを、それぞれUARTのUSIN入力(PC→S1C17W18)、USOUT出力(S1C17W18→PC)として使用します。PC上のTeraTermなどのターミナルエミュレータを使用して、サンプルプログラムとの情報交換を行います。

サンプルソフトウェアは、月/日/時刻、RTC補償情報、A/D変換値、毎秒のA/D変換結果から算出した温度の更新と表示を行います。次の画面は、サンプルソフトウェアが出力するTeraTerm上の表示例です。

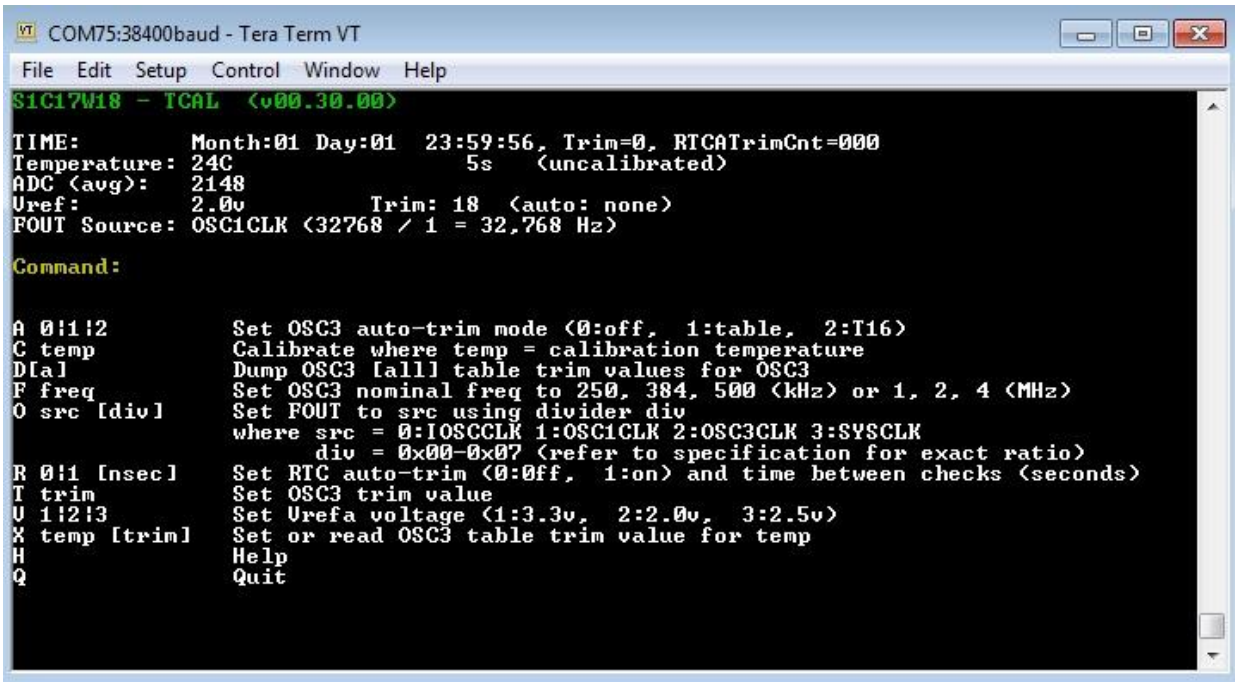

図4-1 サンプルソフトウェアの表示出力

電源が投入されると、RTCの月と日はどちらもデフォルトの01になります。時刻は、サンプルソフトウェアが23:59:51に初期化します。

ステータス“Trim=x”はRTC計時誤差補正アルゴリズムがイネーブル/アクティブかどうかを示します。‘1’はアクティブ状態、‘0’はディスエーブル状態を意味します。“RTCATrimCnt=xxx”は、RTC計時誤差補正がアクティブの場合に論理緩急補正を何回実行したかを示します。

“ADC (avg): xxxx”は、最新の温度センサ出力のA/D変換結果を表示します。ノイズによるA/D変換結果への影響を減らすため、サンプルソフトウェアは16回分のA/D変換値を読み出して平均を取った上で、ここに表示します。

温度は上記のA/D変換平均値から計算され、整数で表示されます

UARTインタフェースのクロックソースはIOSC発振回路です。デバイスを恒温槽に置き、その温度を変化させた場合でもボーレートの精度を維持できるように、ハードウェアによるIOSCのオートトリミングは5分ごとに実行されます。

ユーザは単一キーコマンドの入力により、各種の機能を実行することができます。使用可能なコマンドの一覧を表示させるには、'?'または'h'を入力して<ENTER>を押してください。コマンドは、"src/main.c"内の"process()"関数によって処理されます。

使用可能なコマンドの一覧を表4-1に示します。[ ]で囲まれたパラメータはオプションです。

表4-1　ソフトウェアコマンド一覧

コマンド	パラメータ	説明
AまたはA	0 | 1 | 2	OSC3トリミングモードを設定します。 0: マニュアル: OSC3トリミングなし、または't'コマンドでトリミング値を設定 1: テーブル: トリミング値-温度ルックアップテーブルを使用してOSC3をトリミング 2: オートトリミング: OSC1を基準にOSC3をトリミング(3.1.2節参照)
CまたはC	*temp*	現在の温度を*temp*の値で指定し、基準温度として設定します。 この値は式2-5の**Tref**に相当します。現在のA/D変換結果が**ADCref**として設定されます。
DまたはD	–	OSC3用トリミング値-温度ルックアップテーブル内のトリミング値を、すべてダンプ表示します。
FまたはF	*freq*	OSC3周波数を選択します。*freq*で指定可能な値は以下のとおりです。 250 (kHz) 384 (kHz) 500 (kHz) 1 (MHz) 2 (MHz) 4 (MHz)
OまたはO	*src* [*div*]	FOUT端子から出力する周波数のクロックソースを設定します。オプションの*div*により、選択したクロックソースの周波数を分周して出力することもできます。 srcで指定可能な値は以下のとおりです。 0: IOSCCLK 1: OSC1CLK 2: OSC3CLK 3: SYSCLK *div*の指定可能範囲は0x00～0x07です。
RまたはR	0 | 1 [*nsec*]	RTCの計時誤差補正アルゴリズム(3.3.2節参照)をイネーブルにします。 0: 計時誤差補正ディスエーブル 1: 計時誤差補正イネーブル サンプリング周期Δ*n*のデフォルトは900秒です。これ以外の値に変更するには、オプションの*nsec*でその秒数を指定します。
TまたはT	*trim*	OSC3のトリミング値をマニュアル設定します。0～31の範囲で指定可能です。
VまたはV	1 | 2 | 3	A/D変換器の基準電圧を設定します。指定可能な電圧は以下のとおりです。 1: V_{DD} 2: 2.0V 3: 2.5V
XまたはX	*temp* [*trim*]	OSC3ルックアップテーブル内の任意の温度に対応するトリミング値を読み出します。*temp*で温度を指定します。オプションの*trim*を指定すると、指定した値が温度*temp*に対応するトリミング値として設定されます。
H、hまたは?	–	ヘルプメニューを表示します。
QまたはQ	–	Quit (サンプルソフトウェアを終了します。)

4.1.1 温度計算

サンプルソフトウェアは、式2-5と表2-4に基づき温度を計算して表示します。“src/temptsrvr.c”内の“readTempTsrvr()”関数がA/D変換値を読み出してこの計算を行います。ユーザは‘c’コマンドを使用して、A/D変換基準値と基準温度を変更することができます。サンプルソフトウェアは、‘c’コマンドで指定された温度を基準温度として、現在のA/D変換結果をA/D変換基準値として使用し、以降の温度計算を行います。

温度センサ値の変換でA/D変換器が使用する基準電圧は、‘v’コマンドにより内部電圧の2Vまたは2.5V、もしくは外部電圧V_{DD}に設定可能です。

温度センサの校正のためには、デバイスを恒温槽に置き、温度範囲全体に渡ってA/D変換値を記録します。この結果から、A/D変換値-温度特性の傾きが得られます。

4.1.2 FOUT端子の選択

ユーザが選択したクロック信号をFOUT端子(P16ポート)から出力することができます。FOUT端子から出力するクロック(クロックソースと分周比)は、‘o’コマンドで指定します。また、OSC3内蔵発振回路の周波数は、‘f’コマンドで指定します。FOUT端子にオシロスコープ等を接続して、温度によるクロック周波数の変化を観測できます。

4.1.3 OSC3トリミング

ユーザは‘a’コマンドを使用してOSC3のトリミングモードを選択することができます。

マニュアルトリミング(“a 0”コマンド)を選択した場合、‘t’コマンドでトリミング値を指定できます。

テーブルトリミング(“a 1”コマンド)を選択した場合、“src/main.c”内の“autoOSC3trimTemp()”関数がトリミング値-温度ルックアップテーブルからトリミング値を読み出してトリミングレジスタに設定します。

オートトリミング(“a 2”コマンド)を選択した場合、“src/tcomp.c”内の“SWTrimOSC3()”関数がOSC1を基準周波数として使用し、3.1.2節で説明したとおりOSC3のトリミングを行います。

4.1.4 RTC計時誤差補正

RTCの論理緩急補正(3.3.2節参照)は、‘r’コマンドにより実行するかしないかを選択できます。同時に、オプションのパラメータ*nsec*を指定し、サンプリング周期T_s(秒数)を設定することもできます。*nsec*の指定を省略すると、サンプリング周期は900秒となります。

“src/tcomp.c”内の“tcompSetRtcAlarm()”および“RTCATrim()”関数が、3.3.2節で説明した論理緩急補正を実行します。

4.2 ソフトウェアモジュール

本アプリケーションノートに付随するサンプルソフトウェアは、以下のモジュールを含んでいます。

adc12a.c: A/D変換を行うADC12Aルーチン
boot.c: ブートルーチン
clg.c: クロックジェネレータルーチン
init.c: システム初期化ルーチン
rtca.c: リアルタイムクロックルーチン
tsrvr.c: 温度センサ関連ルーチン
uart.c: UARTルーチン

temptsrvr.c: TSRVR温度センサからの温度データ取得関連ルーチン
tcomp.c: 温度補償関連ルーチン
main.c: メイン/ユーザインタフェース/校正/測定ルーチン

4.2.1 “temptsrvr.c”

このモジュールは、S1C17W18の内蔵TSRVR温度センサを使用した温度の取得に関連する下記の関数を含みます。また、ADC12Aの割り込み処理関数(*intAd12()*)も含め、“adc12a.c”と“tsrvr.c”内の関数も使用します。

readADC(): ADC12Aによる温度センサ電圧の変換結果を読み出す
initTempTsrvr(): 温度読み出しモジュールを初期化
readTempTsrvr(): TSRVR温度センサを使用して温度を読み出す

“*initTempTsrvr()*”関数は、温度読み出しモジュールを初期化するために“main()”から1度だけコールします。この関数には、A/D変換器が使用する基準電圧(VREF)の選択値が引数として渡されます。この関数は、ADC12Aのクロックソース(T16 Ch.3)、TSRVR基準電圧生成回路用出力を割り当てるGPIOポート、A/D変換器用基準電圧を設定します。

初期化関数をコールした後、温度を読み出すために“*readTempTsrvr()*”関数をコールします。この関数は、摂氏温度を返します。この関数は“*readADC()*”関数をコールして温度センサの検出値を読み出し、式2-5を使用して温度を計算します。“*readADC()*”関数は16サンプルのA/D変換結果を取得してその平均値を算出し、グローバル変数“adcAvg”にセットします。

温度計算用(式2-5と表2-4参照)の*Tref[]*、*ADCref[]*、*M[]*の3つの配列は、単一試験器の測定データを基にしたデフォルト値に初期化されます。このデフォルト値は、より正確な校正データを基にした別の値に変更することができます。

4.2.2 “tcomp.c”

このモジュールは、OSC1、OSC3、IOSCの温度補償を実現するいくつかの関数を含んでいます。これらの中には、OSC1を基準とするOSC3ソフトウェアトリミングに使用するT16 Ch.0の割り込み処理関数(*intT16Ch0()*)や、以下のパブリック関数も含まれます。

HWTrimIOSC(): ハードウェアによるIOSCオートトリミングを実行
SWTrimOSC3(): OSC1を基準クロックとしてOSC3ソフトウェアトリミングを実行
tcompSetRtcAlarm(): *alarmsec*秒後にアラーム割り込みを発生するようにRTCをプログラム
initRTCATrim(): RTC論理緩急トリミング用の変数を初期化
RTCATrim(): RTC論理緩急トリミングを実行

“*HWTrimIOSC()*”と“*SWTrimOSC3()*”関数は、それぞれIOSCオートトリミングとOSC3ソフトウェアトリミングの必要に応じてコールします。一般的なアプリケーションでは、大きな温度変化を検出したときにコールすればいいでしょう。

*SWTrimOSC3()*関数

OSC1を基準クロックとして使用するOSC3ソフトウェアトリミングでは“*targetOSC1OSC3[6][2]*”配列が使用され、その初期値には表3-2の値を使用しています。

“*SWTrimOSC3()*”関数はOSC3の周波数誤差を最小にするため、OSC1を基準クロックとしてOSC3トリミング値を調整する以下の処理を実行します。

1) 現在のトリミング値を読み出し、“*trim_ref*”に格納します。
2) “*countT16()*”関数をコールし、OSC3周波数を測定します。“*countT16()*”関数(および“*intT16Ch0()*”割り込みハンドラ)は、OSC3クロックをカウントしたタイマの値をグローバル変数“*measuredOSC3*”に格納します。
3) “*measuredOSC3*”の値と“*targetOSC1OSC3[][1]*”配列から読み出した期待値との差を算出し、“*delta_ref*”に格納します。
4) “*delta_ref*”が正の値の場合はOSC3トリミング値を1だけ減らし、それ以外の場合は1だけ増やします。新しいトリミング値を“*trim_new*”に格納します。
5) “*countT16()*”関数をコールしてトリミング実施後のOSC3周波数を測定し、“*measuredOSC3*”に新たな測定値を取得します。
6) “*measuredOSC3*”の値と“*targetOSC1OSC3[][1]*”配列から読み出した期待値との差を算出し、“*delta_new*”に格納します。
7) “*delta_new*”の絶対値が“*delta_ref*”の絶対値より大きかった場合、OSC3トリミングレジスタを“*trim_ref*”の値に設定して終了します。それ以外の場合、“*trim_ref*”を“*trim_new*”の値に、“*delta_ref*”を“*delta_new*”の値に設定します。
8) ステップ4に戻ります。

RTC補償関連の関数

“*tcompSetRtcAlarm()*”関数は、割り込みを発生させるRTCアラームの設定を行います。この関数には、現時点からアラーム割り込みを発生させるまでの秒数を引数として渡します。通常この関数は、RTCの論理緩急補正の実行間隔を設定するために使用します。

“*RTCATrim()*”関数に必要な変数を初期化するために、“*initRTCATrim()*”関数をコールします。この関数には、現在の周波数誤差“*curr_freqerr*”(ミリヘルツ)を引数として渡します。“*initRTCATrim()*”関数は、グローバル変数“*freqerrsum_mHz*”を0に、“*prev_osc1FreqErr_mHz*”を“*curr_freqerr*”の値に初期化します。

“*RTCATrim()*”関数は、RTCの論理緩急補正を実行するためにコールされます。通常は、RTCアラーム割り込み発生時にコールします。この関数には、サンプリング周期“*tsamp*”と現在の周波数誤差“*curr_freqerr_mHz*”(ミリヘルツ)を引数として渡します。“*tsamp*”の値は、式3-6の$\boldsymbol{T_s}$の値に相当します。グローバル変数“*freqerrsum_mHz*”は、周波数誤差の累積和である、式3-6に示した***ferr(n)***の総和の値に相当します。

“*RTCATrim()*”関数は以下の処理を実行します。

1) 現在の周波数誤差“*curr_freqerr_mHz*”と前回の周波数誤差“*prev_osc1FreqErr_mHz*”を加算後、2で割って平均周波数誤差を取得します。
2) 平均周波数誤差を累積誤差“*freqerrsum_mHz*”に加算します。
3) 算出された最新の累積誤差“*freqerrsum_mHz*”の補正に必要なトリミング値(“*rtcTrim*”変数)を、式3-6を使用して計算します。
4) 論理緩急補正を実行します。さらに、“*freqerrsum_mHz*”の値をトリミング値の分だけ減少させます(式3-7）。
5) “*tsamp*”秒後に割り込みが発生するように、RTCアラームを設定(“*tcompSetRtcAlarm()*”関数をコール)します。

4.2.3 “main.c”

このモジュールは、“main()”およびUARTインタフェースを介してユーザとのやり取りを行うための以下の関数を含んでいます。

initClocks():	アプリケーションで使用する時計を初期化します。
setTref():	‘c’または‘C’コマンドによりコールされ、現在の温度および現在選択されている基準電圧(VREF)によるA/D変換値を、それぞれ基準温度および基準A/D変換値として設定します。
setOSC3trim():	OSC3トリミング値を設定します。OSC3は一旦停止し、トリミング値を設定後に再起動します。
getOSC3freq():	現在設定されているOSC3の公称周波数(Hz)を返します。
getClgSystemClkFreq():	現在設定されているシステムクロックの公称周波数(Hz)を返します。
autoOSC3trimTemp():	温度が変化するたびに、トリミング値-温度ルックアップテーブルを使用してOSC3トリミングレジスタを設定します。この関数は、OSC3トリミングモードが1(“a 1”コマンドで設定)の場合に、温度が変化するとコールされます。
sprintfcomma():	10進数を3桁ごとにカンマで区切った文字列に変換します。
clrMsgArea():	メッセージ領域をクリアします。
updateTemp():	温度ステータス行の表示内容を更新します。
updateOSC3():	OSC3ステータス行の表示内容を更新します。
updateVref():	VREFステータス行の表示内容を更新します。
updateFOUT():	FOUTステータス行の表示内容を更新します。
updateCmdPrompt():	コマンドプロンプトを表示します。
displayHeader():	ヘッダを表示します。
updateDisplay():	表示をすべて更新します。
displayHelp():	ヘルプの内容を表示します。
dumpOSC3TrimTable():	OSC3トリミング値のテーブルを表示します。
nextarg():	文字列中の次のトークン/引数を探します。
process():	ユーザのキー操作を処理します。

RTCトリミングには、“*osc1FreqErr_mHz[]*”配列をOSC1周波数誤差(ミリヘルツ)-温度(-40～85°C、1°C単位)ルックアップテーブルとして使用します。このテーブルは、単一のSVTmini17W18ボードによる校正結果を基に初期化されます。初期値のデータは、OSC1水晶発振回路の設計に合わせ、別の校正値のセットに変更することも可能です。

改訂履歴表

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.0	2016/10/25	全ページ	新規	新規制定

セイコーエプソン株式会社

マイクロデバイス事業部　デバイス営業部

東京　〒191-8501　東京都日野市日野421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

大阪　〒541-0059　大阪市中央区博労町3-5-1　御堂筋グランタワー15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：413349000
2016年 11月 作成 Ⓛ