お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

アプリケーション・ノート

78K0/Lx3

サンプル・プログラム（16ビットΔΣ型A/Dコンバータ）

変換結果精度補正編

本資料は，78K0/LE3，78K0/LF3マイクロコントローラに搭載する16ビットΔΣ型A/Dコンバータをより良い精度でご利用いただく方法を紹介するアプリケーション・ノートです。16ビットΔΣ型A/Dコンバータを差動入力モードで使用する場合，変換結果の誤差にある一定の特徴があります。この誤差の特徴を利用し，それをソフト的に除去することにより，より良い精度でΔΣADコンバータを使用することが可能になります。

対象デバイス

78K0/LE3マイクロコントローラ
78K0/LF3マイクロコントローラ

目　次

資料番号　U19332JJ1V0AN00（第1版）
発行年月　June 2008 NS

- 本資料に記載されている内容は2008年6月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

(注)

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

第1章　概　　要

本資料は，78K0/LE3，78K0/LF3マイクロコントローラに搭載する16ビットΔΣ型A/Dコンバータをより良い精度でご利用いただく方法を紹介するアプリケーション・ノートです。

16ビットΔΣ型A/Dコンバータを差動入力モードで使用する場合，変換結果にある一定の特徴があります。具体的には図1－1に示すように，ある特定の入力電圧（5点）で変換結果に変曲点が存在します。この特徴を利用し，それをソフト的に除去すること（5点補正）により，より良い精度でΔΣADコンバータを使用することが可能になります。

また，温度が変化した場合の変換結果においても，ある一定の特徴があります。具体的には図1－2に示しますように，温度変化により，ゲインが変化します。この特徴を利用し，それをソフト的に除去すること（温度補正）により，より良い精度でΔΣADコンバータを使用することが可能になります。

本アプリケーション・ノートでは上記の手法を具体的に説明するとともに，サンプル・プログラムを提供します。

図1－1　A/D変換結果の特徴

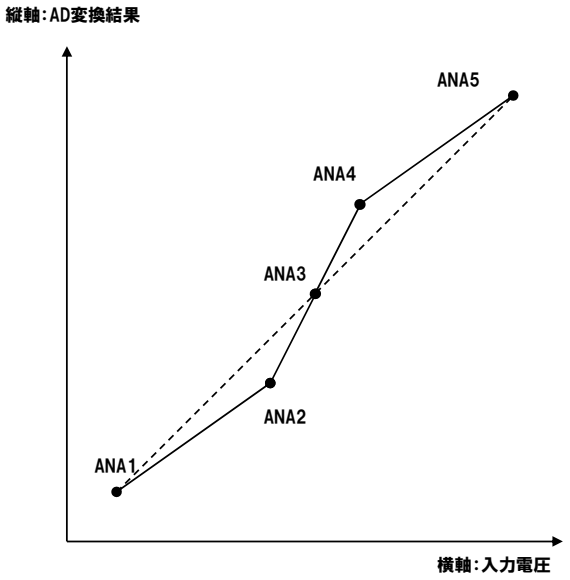
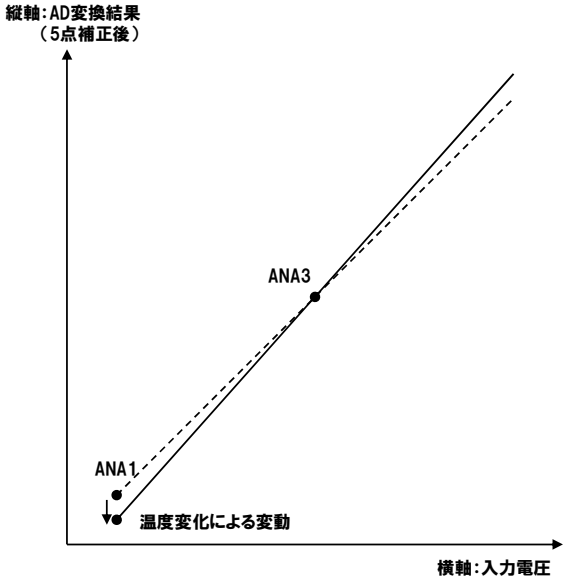
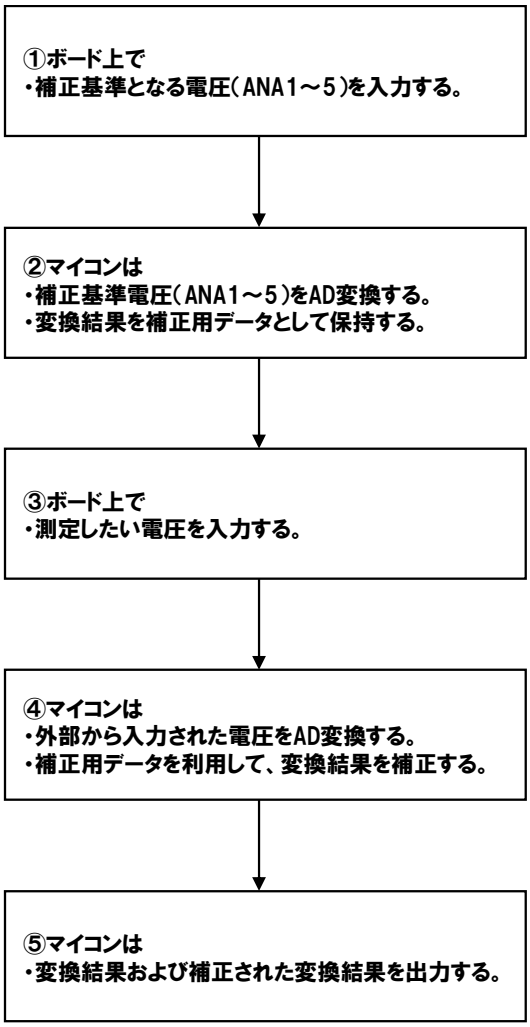


図1－2　温度変化した場合のA/D変換結果の特徴



注意　本資料に記載されている手法は，精度向上を保証するものではありません。実際のアプリケーションにおいて，十分な評価を実施してからご利用ください。

動作手順概要およびサンプル・プログラムの利用について説明します。
基本的な動作手順としては，以下の①～⑤の順序で行います。



基本的には外部基準電圧源より入力します。補正ポイントの電圧はREF+電圧に依存しますので，REF+電圧に応じた電圧を入力します。

サンプル・プログラムの“補正Aモード”を利用することにより，本動作を実行します（詳細には，①②の組み合わせで5回繰り返します）。

外部より電圧を印加します。この電圧が実際に測定したい電圧です。

サンプル・プログラムの“変換Aモード”を利用することにより，本動作を実行します。第2章で説明する補正アルゴリズムを用いて変換結果を補正算出します。

サンプル・プログラムの“変換Aモード”を利用することにより，本動作を実行します。UART経由で結果を出力します。PC（通信ソフトを利用）などで結果をモニタします。

なお，本サンプル・プログラムでは上記“補正Aモード”および“変換Aモード”に加え，“補正Bモード”および“変換Bモード”を用意しております。“補正Bモード”および“変換Bモード”は，A/D変換前に，外部DAC-ICに対し電圧設定を出力するモードです。ボード上にDAC-ICを搭載し，このモードを利用することにより，DAC-ICからの電圧を用いて評価を行うことが可能です。

以降，DAC-ICからの電圧を用いずに外部から電圧を入力する場合を“DACを使用しない場合”，DAC-ICからの電圧を用いる場合を“DACを使用する場合”と表現します。

以下にサンプル・プログラムの各処理内容について説明します。

(1) 使用する周辺の初期設定の主な内容

使用する周辺の初期設定の主な内容は次のとおりです。

●割り込みの禁止
●CPU／周辺ハードウエア・クロックを高速システム・クロック動作で，10 MHzに設定
●シリアル・インタフェースUART6をデータ送信用に設定
●8ビット・タイマH0を下記タイミング用のインターバル・タイマ（基本タイマ約100 μs）に設定
　・DAC設定後の回路の安定待ち（約500 μs）
●8ビット・タイマH1をキー・スキャン用インターバル・タイマ（約10 ms周期）に設定
●16ビットΔΣ型A/Dコンバータの設定
●ポートの設定
●割り込みの許可

(2) メイン処理の内容

キー・スキャン処理，およびイベント処理を呼び出します。また現在のモードを判断し，変換A1モード制御処理，変換A2モード制御処理，変換Bモード制御処理，補正Aモード制御処理，および補正Bモード制御処理のうち，いずれかを呼び出します。

なお，キー・スキャン処理は約10 ms周期で呼び出します。

(3) キー・スキャン処理の内容

キー入力の検出を行う処理です。

約10 msごとに7つのキー入力端子のスキャンを行います。3回の一致でチャタリング除去を行い，一致した場合のみ，現在のキー状態をコード化します。

(4) イベント処理の内容

キー・コードの変化を検出場合に，キー・コードに従ってモード（変換A1，変換A2，変換B，補正A，補正B）の切り替え，または処理実行状態（スタンバイ中，実行中，停止処理中）の切り替えを行います。処理実行状態がスタンバイ中であれば，モードの切り替えが可能です。

(5) 変換A1モード制御処理の内容

外部からの電圧入力を想定した単発変換を行うモードです。進行キーが押下されると，下記処理を1回実行します。

- アナログ入力のA/D変換
- A/D変換結果の補正[注1]
- UART送信[注2]

(6) 変換A2モード制御処理の内容

外部からの電圧入力を想定した連続変換を行うモードです。進行キーが押下されると，停止キーが押下されるまで下記処理を実行します。

- アナログ入力のA/D変換
- A/D変換結果の補正[注1]
- UART送信[注2]

(7) 変換Bモード制御処理の内容

DACからの電圧入力を想定した連続変換を行うモードです。進行キーが押下されると，設定された変換開始値および変換終了値を読み込みます。そして読み込んだ変換開始値から変換終了値までDAC設定値をインクリメントしながら下記処理を実行します。DACに変換終了値を設定して下記処理が終了した場合，または停止キーが押された場合に処理が終了します。

- DAC出力設定
- アナログ入力のA/D変換
- A/D変換結果の補正[注1]
- UART送信[注2]

注1. A/D変換結果の補正は，補正Aモードまたは補正Bモードにより，5点補正用データの測定が正常に終了していた場合のみ実行します。

2. UART送信データフォーマットについては，4. 4　UART**送信データのフォーマット**を参照してください。

(8) 補正Aモード制御処理の内容

外部からの電圧入力を想定した5点補正用データの測定を行うモードです。進行キーが押下されるごとに下記処理を行い，処理が5回実行された場合，または停止キーが押された場合に処理が終了します。

- アナログ入力のA/D変換
- UART送信[注]

5点のA/D変換結果が正しく測定できていない場合，エラー表示用LEDを点灯します。

(9) 補正Bモード制御処理の内容

DACからの電圧入力を想定した5点補正用データ測定を行うモードです。主な処理は下記のとおりです。進行キーが押下されると，下記処理を連続で行い，処理が5回実行された場合，または停止キーが押された場合に処理が終了します。DAC出力設定値はあらかじめ初期化処理にて算出します。

- DAC出力設定
- アナログ入力のA/D変換
- UART送信[注]

5点のA/D変換結果が正しく測定できていない場合，エラー表示用LEDを点灯します。

注 UART送信データフォーマットについては，4. 4　UART**送信データのフォーマット**を参照してください。

第2章　A/D変換結果補正アルゴリズム

この章では，このサンプル・プログラムで使用する5点補正および温度補正のアルゴリズムを説明します。

2. 1　5点補正について

5点補正処理の概略を以下に示します。

① あらかじめサンプルごとに使用電圧に合わせた5つの5点補正用データを測定します。

② 5点補正用データに基づき，16ビットΔΣ型A/Dコンバータの入出力特性を4つの関数に近似します。

③ 16ビットΔΣ型A/Dコンバータの入出力特性の近似関数を使用し，16ビットΔΣ型A/Dコンバータの変換結果を補正します。

(1) 5点補正用データの測定

あらかじめサンプルごとに使用電圧に合わせた5つの5点補正用データを測定します。図2－1は16ビットΔΣ型A/Dコンバータの入出力特性モデルです。図中のana1，ana2，ana3，ana4，ana5は入出力信号の値です。また，図中のcode1，code2，code3，code4，code5は，入出力信号ana1，ana2，ana3，ana4，ana5に対する出力コードです。

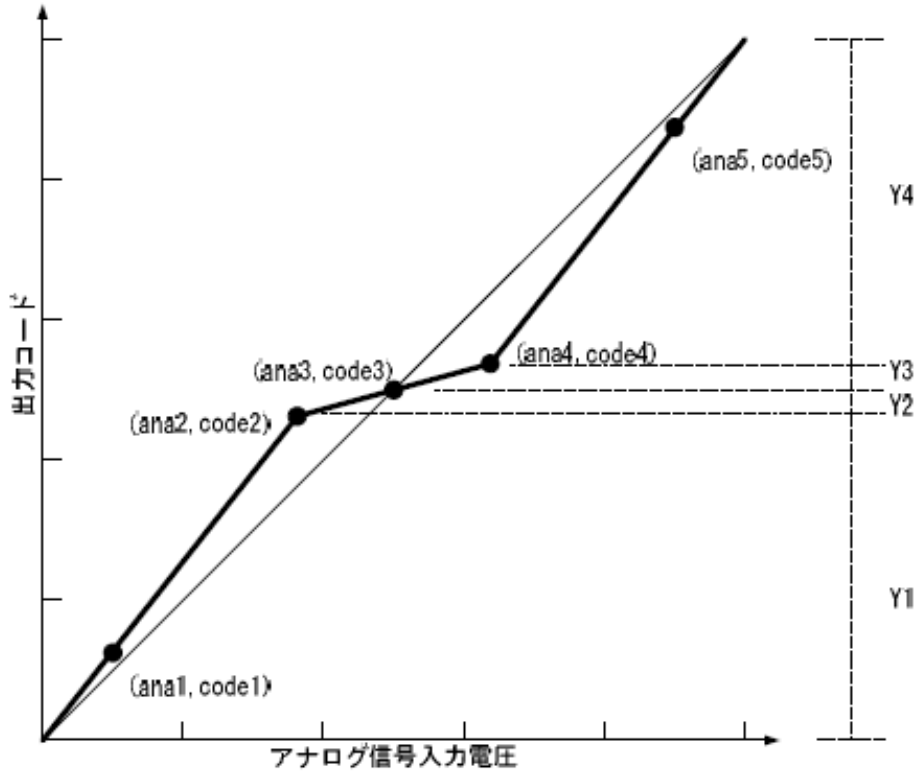
ana1～ana5におけるcode1～code5を，16ビットΔΣ型A/Dコンバータの16ビット分解能で測定します。ana1～ana5は下記の式にて算出します。code1～code5の測定は25℃で行います。

ana5：0.9 × (REF+)
ana4：0.82 × (REF+) – 0.91 注
ana3：0.5 × (REF+)
ana2：(REF+) –（0.82 × (REF+) – 0.91）注
ana1：0.1 × (REF+)

- (REF+) はリファレンス端子の正側電圧で，AV_{REF}と同電位です。
- AV_{DD}はアナログ用電源電圧です。
- 出力コードcode1～code5は，入力信号ana1～ana5をA/D変換して，求めます。

注　AV_{REF} ＝ (REF+) ＜2.84375 V の場合，ana2＞ana4となります。この場合は，ana2とana4の値を逆転させて，5点補正を行います。

図2－1　16ビット⊿Σ型A/Dコンバータの入出力特性モデル



（2）16ビット⊿Σ型A/Dコンバータ入出力特性を関数に近似

5点補正用データに基づき，16ビット⊿Σ型A/Dコンバータ入出力特性を4つの関数に近似します。表2－1に示すように，2点ずつの5点補正用データを使用し，各出力コード区間Y1，Y2，Y3，Y4（図2－1を参照）における入出力特性を関数（傾きと切片）に近似します。

表2－1　16ビット⊿Σ型A/Dコンバータ入出力特性の近似関数

区間	対象出力コード	近似関数		
			傾き	切片
Y4	code4以上	a4×A_{IN}＋b4	$a4=\frac{code5-code4}{ana5-ana4}$	$b4=\frac{ana4\times code5-ana5\times code4}{ana4-ana5}$
Y3	code3～code4	a3×A_{IN}＋b3	$a3=\frac{code4-code3}{ana4-ana3}$	$b3=\frac{ana3\times code4-ana4\times code3}{ana3-ana4}$
Y2	code2～code3	a2×A_{IN}＋b2	$a2=\frac{code3-code2}{ana3-ana2}$	$b2=\frac{ana2\times code3-ana3\times code2}{ana2-ana3}$
Y1	code2	a1×A_{IN}＋b1	$a1=\frac{code2-code1}{ana2-ana1}$	$b1=\frac{ana1\times code2-ana2\times code1}{ana1-ana2}$

備考　A_{IN}：アナログ入力電圧

(3) A/D変換結果の補正

16ビットΔΣ型A/Dコンバータ入出力特性の近似関数を使用し，A/D変換結果を補正します。表2－2に示すように，出力コード値に合わせてA/D変換結果を補正して出力します。区間の境界値（code2，code3，code4）は，原理的にどちらの補正式を使用しても同じ結果が得られます。

code1≦A/D変換結果≦code5の場合，5点補正での補正が可能です。

表2－2　補正出力の演算式

区間	5点補正対象出力コード	5点補正出力	5点補正出力（a1～a4，b1～b4を代入し展開）
Y4	code4以上	$\frac{A/D変換結果 - b4}{a4}$	$\frac{A/D変換結果(ana5 - ana4) + ana4code5 - ana5code4}{code5 - code4}$
Y3	code3～code4	$\frac{A/D変換結果 - b3}{a3}$	$\frac{A/D変換結果(ana4 - ana3) + ana3code4 - ana4code3}{code4 - code3}$
Y2	code2～code3	$\frac{A/D変換結果 - b2}{a2}$	$\frac{A/D変換結果(ana3 - ana2) + ana2code3 - ana3code2}{code3 - code2}$
Y1	code2	$\frac{A/D変換結果 - b1}{a1}$	$\frac{A/D変換結果(ana2 - ana1) + ana1code2 - ana2code1}{code2 - code1}$

2. 2　温度補正について

温度補正処理の概略を以下に示します。

① A/D変換開始前に1点の温度補正用データを測定します。

② 温度補正用データに基づき，温度による5点補正結果の傾きを算出します。

③ 算出した傾きにより，5点補正結果を温度補正します。

注意　本温度補正による補正効果は数値化されておりません。

（1）温度補正データの測定

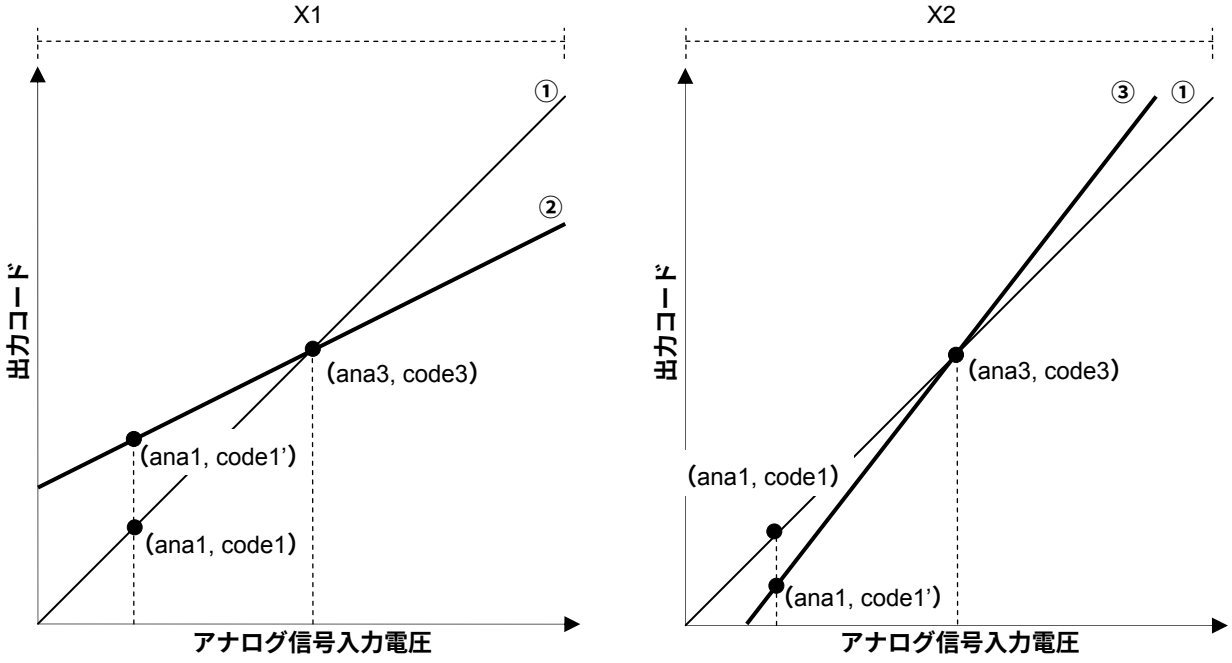
A/D変換開始前にana1における出力コードを16ビットΔΣ型A/Dコンバータの16ビット分解能で測定し，A/D値を5点補正します。この値をcode1'とします。

（2）5点補正結果の傾きを算出

ana1で温度変化により，基準温度のA/D値（code1）がcode1'まで変化します。このときA/D変換値の特性は（ana1, code1'）および（ana3, code3）を通る直線になります。

（1）で測定したcode1'と，ana1，ana3，およびあらかじめ補正しておいたcode3を基に，実際の5点補正結果の傾きを算出します。図2－2のcode1とcode3は，2.1（1）にて測定したcode1とcode3を5点補正した値です。

図2－2　5点補正結果の傾きのモデル



①：温度誤差がないデータ

②：実際の5点補正結果（温度誤差により傾きが小さくなる場合）

③：実際の5点補正結果（温度誤差により傾きが大きくなる場合）

(3) 5点補正結果を温度補正

温度誤差のある5点補正結果の傾きをcode1，code3，code1'より算出し，理想値の傾きに近づけることで温度補正を行います。

実際の温度補正演算に使用する式は表2－2に示します。途中計算が負の値にならないよう，X1，X2に場合分けします。場合分けについては図2－2を参照してください。

表2－2 温度補正の演算式

区間	温度誤差による5点補正結果の傾き	温度補正出力
X1	小さい	$\frac{5点補正結果(code3-code1)-code3(code1'-code1)}{code3-code1'}$
X2	大きい	$\frac{5点補正結果(code3-code1)+code3(code1-code1')}{code3-code1'}$

第3章　回路図

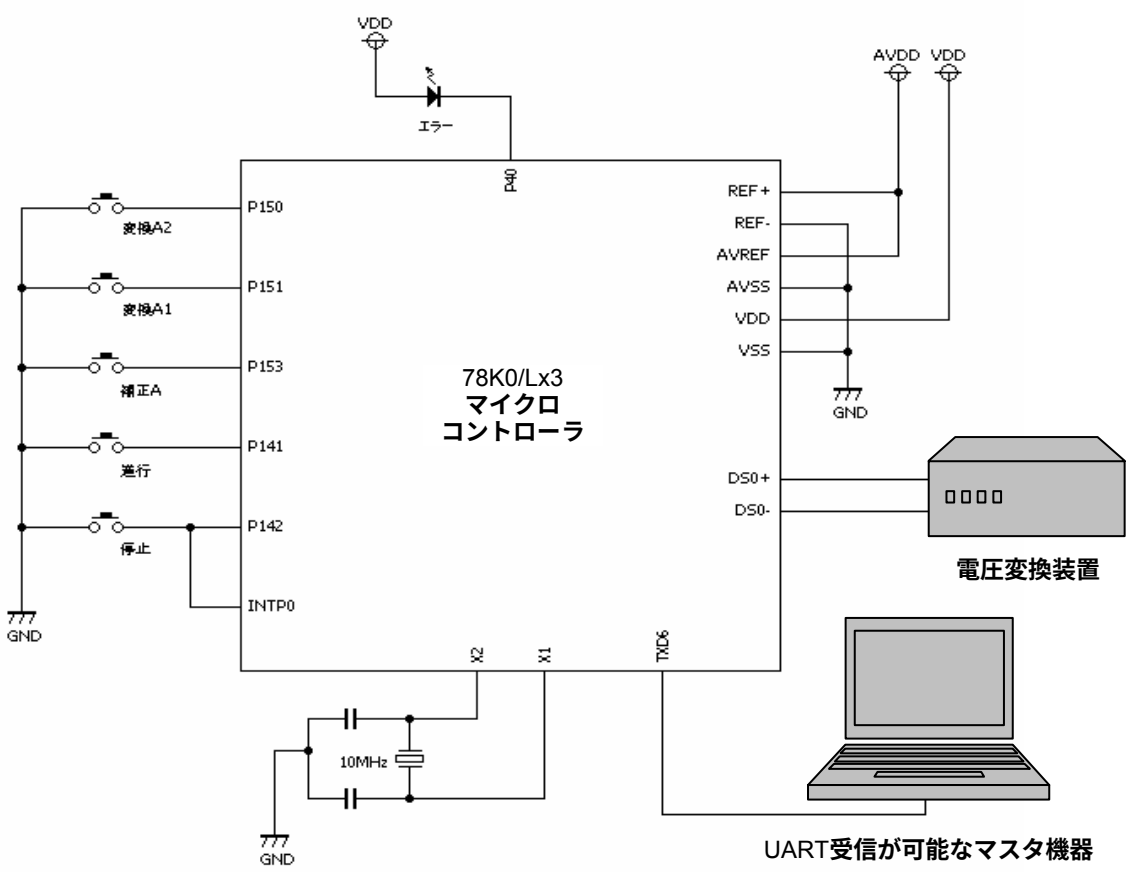
この章では，このサンプル・プログラムで使用する場合の回路図および周辺ハードウエアを説明します。

3. 1　回路図

3. 1. 1　DACを使用しない場合の回路図

⊿Σ型A/Dコンバータのアナログ入力にDACを使用しない場合の回路図を次に示します。

図3－1　DACを使用しない場合の接続



注意1.　2.7 V≦AV_{REF}＝REF+≦V_{DD}≦5.5 Vの電圧範囲で使用してください。

2.　AV_{SS}端子はV_{SS}と同電位にし，GNDに直接接続してください。

3.　回路図中の端子以外の未使用のポート機能端子はすべて出力ポートのため，オープン（未接続）にしてください。

4.　DS0-/DS0+は，REF-からREF+の範囲でアナログ入力可能な電圧変換装置に接続します。

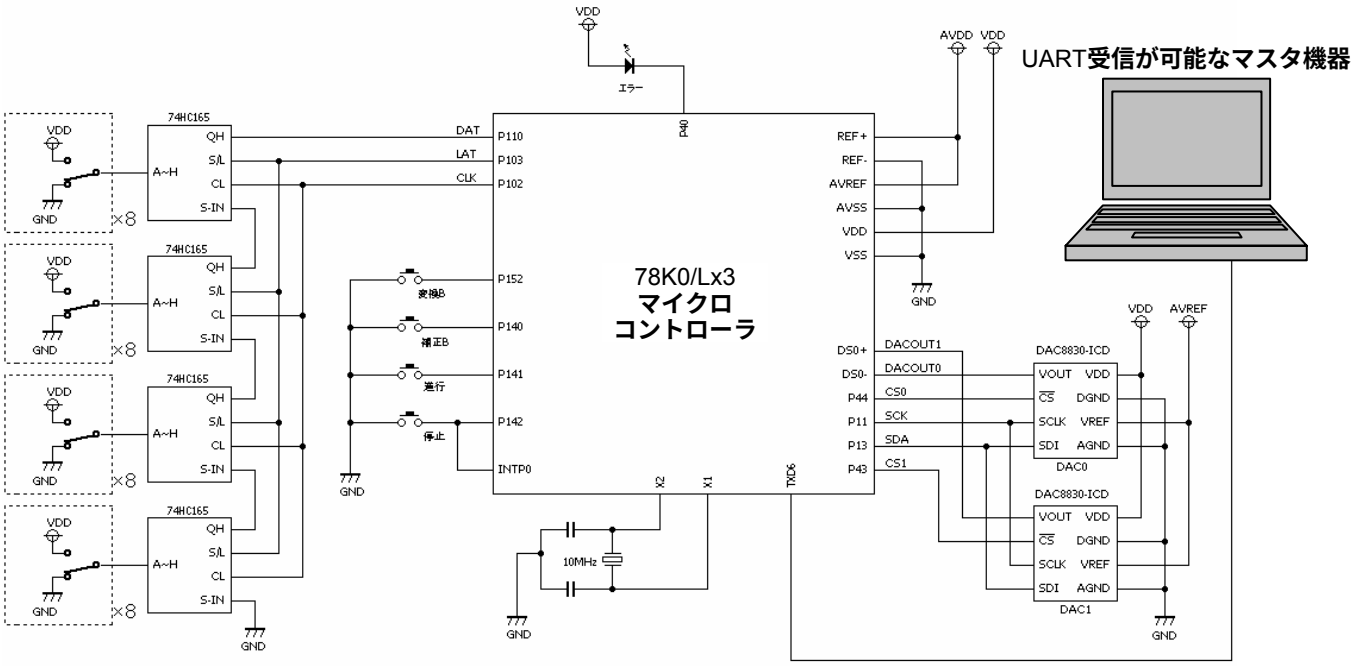
5.　TxD6端子は，UART受信が可能な機器に接続します。

3. 1. 2　DACを使用する場合の回路図

⊿Σ型A/Dコンバータのアナログ入力にDACを使用する場合の回路図を次に示します。

なお，本サンプル・プログラムではDACにTI社製DAC8830ICDを使用しています。

図3－2　DACを使用する場合の接続



注意1.　2.7 V≦AV$_{REF}$＝REF+≦V$_{DD}$≦5.5 Vの電圧範囲で使用してください。

2.　AV$_{SS}$端子はV$_{SS}$と同電位にし，GNDに直接接続してください。

3.　回路図中の端子以外の未使用のポート機能端子はすべて出力ポートのため，オープン（未接続）にしてください。

4.　DS0-/DS0+は，REF-からREF+の範囲でアナログ入力可能な電圧変換装置に接続します。

5.　TxD6端子は，UART受信が可能な機器に接続します。

3. 2　周辺ハードウエア

使用する周辺ハードウエアを次に示します。

（1）水晶振動子

X1，X2に10 MHzの水晶振動子を接続します。

（2）UART**通信用機器（**TxD6**）**

TxD6端子にUART受信用の機器を接続します。

（3）キー・スイッチ（P140-P142，P150-P153**）**

DACを使用しない場合，キー入力ポート（P150，P151，P153，P141，P142）に計5個のキー・スイッチを接続します。

DACを使用する場合，キー入力ポート（P152，P140-P142）に4個のキー・スイッチを接続します。

（4）16**ビット⊿Σ型**A/D**コンバータのアナログ入力端子（**DS0-/DS0+**）**

アナログ入力にDACを使用しない場合，DS0-/DS0+端子は，AV_{SS}からAV_{REF}の範囲でアナログ入力可能な電圧変換装置に接続します。

アナログ入力にDACを使用する場合，DS0-/DS0+端子は，それぞれDACの出力端子に接続します。本サンプル・プログラムは，DAC8830ICDを計2個使用しています。

（5）8**ビット・シリアル・シフト・レジスタ（**P110，P102，P103**）**

DACを使用する場合，変換開始値（16ビット）および変換終了値（16ビット）を設定するため，8 ビット・シリアル・シフト・レジスタ　74HC165を4個接続します。P110をデータラインに，P103をラッチに，P102をクロック・ラインに使用します。

（6）エラー出力用LED**（**P40**）**

5点補正演算係数のエラー表示用に，P40にエラー出力用LEDを接続します。

第4章　ソフトウエアについて

この章では，ダウンロードする圧縮ファイルのファイル構成，使用するマイコンの内蔵周辺機能，サンプル・プログラムの初期設定と動作概要，UART送信データのフォーマット，およびフロー・チャートを説明します。

4.1　ファイル構成

ダウンロードする圧縮ファイルのファイル構成は，次のようになっています。

ファイル名	説 明	同封圧縮（*.zip）ファイル	
			
main.c	マイコンのハードウエア初期化処理，メイン処理，キー・スキャン処理，イベント処理，変換A1モード制御処理，変換A2モード制御処理，変換Bモード制御処理，補正Aモード制御処理，補正Bモード制御処理のソース・ファイル[注]	●	●
K0Lx3_DSAD.prw	統合開発環境 PM+用ワーク・スペース・ファイル		●
K0Lx3_DSAD.prj	統合開発環境 PM+用プロジェクト・ファイル		●

注　ダウンロードするソース・ファイルには，A/D変換のアナログ入力にDACを使用しない場合と，DACを使用する場合の両方の処理が含まれています。動作環境に合わせて，不要な処理を削除してください。

備考　  　：ソース・ファイルのみ同封

 　：統合開発環境 PM+で使用するファイルを同封

4. 2　使用する内蔵周辺機能

このサンプル・プログラムでは，マイコンに内蔵する次の周辺機能を使用します。

・キー・スキャン周期のタイミングの計測用

　　：8ビット・タイマH1を約10 ms（$f_{PRS}/2^{12}$×23）周期のインターバル・タイマとして使用

・DAC設定後の周辺回路安定待ち時間計測用

　　：8ビット・タイマH0を約100 μs（$f_{PRS}/2^{10}$）周期を基準時間とするインターバル・タイマとして使用

・A/D変換用

　　：16ビットΔΣ型A/Dコンバータの DS0-/DS0+を使用

・A/D変換結果出力用

　　：シリアル・インタフェース UART6を使用

4. 3　初期設定と動作概要

このサンプル・プログラムでは，初期設定にて，クロック周波数の選択， 8ビット･タイマH0の設定，8ビット･タイマH1の設定，シリアル・インタフェースUART6の設定，16ビットΔΣ型A/Dコンバータの設定，および入出力ポートの設定を行います。

動作概要は，4.3.1　DACを使用しない場合の動作と4.3.2　DACを使用する場合の動作に示します。

本サンプル・プログラムはA/D変換のアナログ入力にDACを使用しない場合と，DACを使用する場合の両方の処理が含まれます。動作環境に合わせて，不要な処理を削除し，4.3.1　DACを使用しない場合の動作，または4.3.2　DACを使用する場合の動作を参照してください。

4. 3. 1　DACを使用しない場合の動作

使用する周辺の初期設定完了後，変数を初期化し，キー・スキャン処理，イベント処理，変換A1モード制御処理，変換A2モード制御処理，補正Aモード制御処理を呼び出します。なお，キー・スキャン処理は約10 ms周期で呼び出します。また変換A1モード制御処理，変換A2モード制御処理，および補正Aモード制御処理は，現在のモードとシステムの状態に合わせて呼び出します。

(1) キー・スキャン処理

7つのキー入力ポートからキー状態をスキャンします。チャタリング除去回数は3回で，キーの状態をコード化します。

(2) イベント処理

キー・コードに変化があった場合，キー・コードにしたがって，モード（変換A1，変換A2，補正A）の切り替え，または変換実行状態（変換実行中，スタンバイ中，停止中）の切り替えを行います。変換状態がスタンバイ中の場合のみ，モードの切り替えが可能です。

(3) 変換A1モード制御処理

進行キーが押されるたびに，以下の処理を1回行います。停止キーが押されると，変換動作を終了します。

- 16ビットΔΣ型A/Dコンバータにて A/D変換
- 5点補正（5点補正用データ測定済みの場合のみ実行）
- 温度補正（5点補正完了かつ温度誤差があった場合のみ実行）
- UARTでA/D変換結果，5点補正結果，および温度補正結果を出力[注]

温度補正は，変換A1モードに入った直後に測定した5点補正用データ（code1）に，温度誤差があった場合に行います。そのため，変換A1モードに入る場合，アナログ入力にana1の値を入力しておく必要があります。

A/D変換結果，5点補正結果，温度補正結果はUARTで出力しますが，5点補正，温度補正がそれぞれ行われなかった場合は，送信データの該当部分が「＊＊＊＊」となります[注]。

注　UARTの送信データの詳細については，4.4　UART送信データのフォーマットを参照してください。

(4) 変換A2モード制御処理

進行キーが押されると，以下の処理を連続で行います。停止キーが押されると，変換動作を終了します。

- 16ビットΔΣ型A/Dコンバータにて A/D変換
- 5点補正（5点補正用データ測定済みの場合のみ実行）
- 温度補正（5点補正完了，かつ温度誤差があった場合のみ実行）
- UARTでA/D変換結果，5点補正結果，および温度補正結果を出力[注]

温度補正は，変換A2モードに入った直後に測定した5点補正用データ（code1）に，温度誤差があった場合に行います。そのため変換A2モードに入る場合，アナログ入力にana1の値を入力しておく必要があります。

A/D変換結果，5点補正結果，温度補正結果はUARTで出力しますが，5点補正，温度補正がそれぞれ行われなかった場合は，送信データの該当部分が「＊＊＊＊」となります[注]。

(5) 補正Aモード制御処理

補正Aモードは，5点補正用データを測定するモードです。進行キーが押されると，以下の処理を1回行います。停止キーが押されると，変換動作を終了します。また以下の処理が5回行われると，変換動作を終了します。

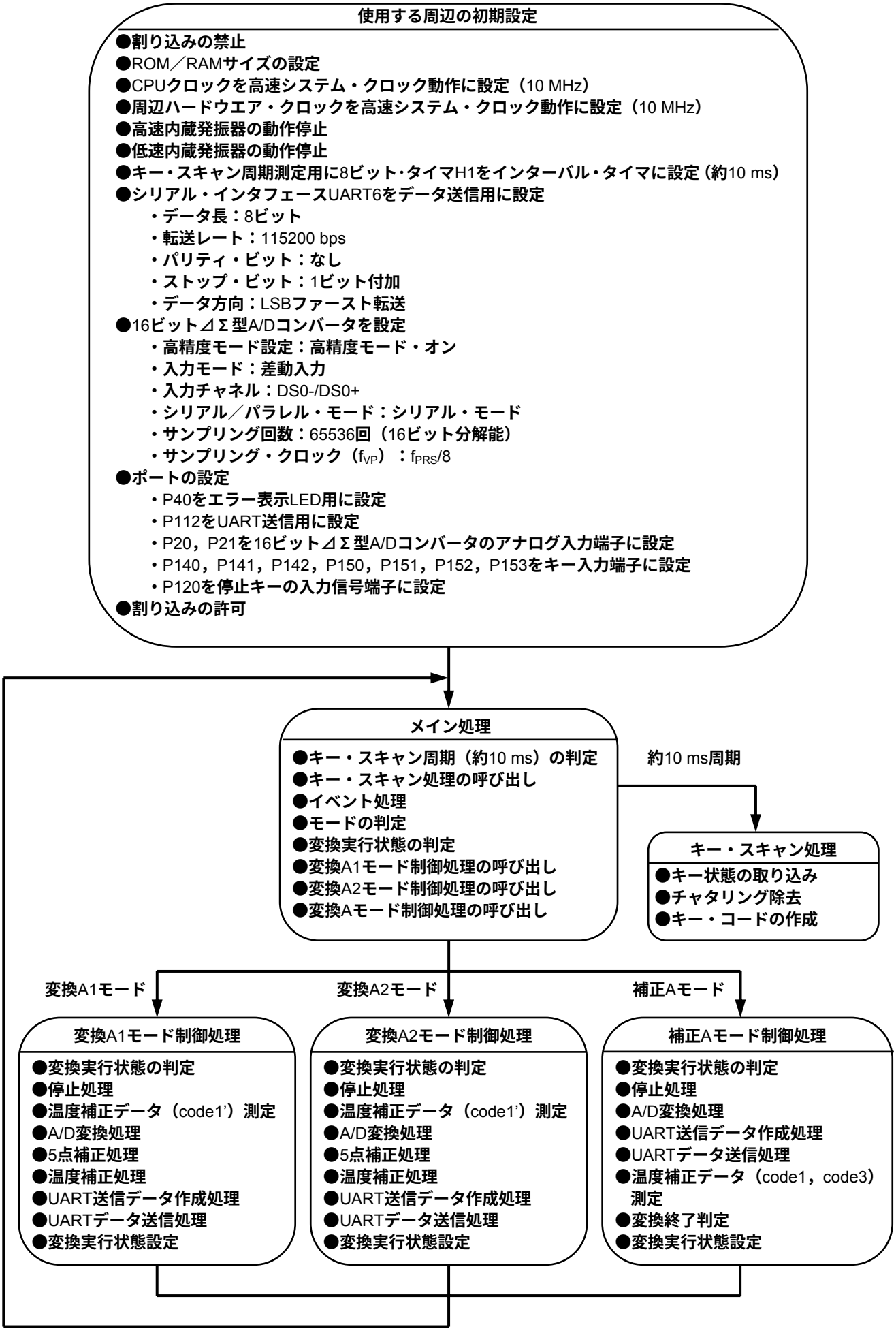
- 16ビットΔΣ型A/Dコンバータにて A/D変換
- UARTでA/D変換結果を出力[注]

進行キーが押されるごとにcode1からcode5を順に測定します。そのため，アナログ入力にはana1からana5を順に入力する必要があります。

5点補正用データcode1～code5がcode1＜code3＜code5となっていなければ，エラー表示用LEDを点灯します。5点補正用データが正常に測定できた場合，code1およびcode3を5点補正にて補正し，温度補正用データとして保存します。

注　UARTの送信データの詳細については，4.4　UART**送信データのフォーマット**を参照してください。

詳細については，次の状態遷移図（ステート・チャート）に示します。


使用する周辺の初期設定
●割り込みの禁止
●ROM／RAMサイズの設定
●CPUクロックを高速システム・クロック動作に設定（10 MHz）
●周辺ハードウエア・クロックを高速システム・クロック動作に設定（10 MHz）
●高速内蔵発振器の動作停止
●低速内蔵発振器の動作停止
●キー・スキャン周期測定用に8ビット・タイマH1をインターバル・タイマに設定（約10 ms）
●シリアル・インタフェースUART6をデータ送信用に設定
・データ長：8ビット
・転送レート：115200 bps
・パリティ・ビット：なし
・ストップ・ビット：1ビット付加
・データ方向：LSBファースト転送
●16ビット⊿Σ型A/Dコンバータを設定
・高精度モード設定：高精度モード・オン
・入力モード：差動入力
・入力チャネル：DS0-/DS0+
・シリアル／パラレル・モード：シリアル・モード
・サンプリング回数：65536回（16ビット分解能）
・サンプリング・クロック（fVP）：fPRS/8
●ポートの設定
・P40をエラー表示LED用に設定
・P112をUART送信用に設定
・P20，P21を16ビット⊿Σ型A/Dコンバータのアナログ入力端子に設定
・P140，P141，P142，P150，P151，P152，P153をキー入力端子に設定
・P120を停止キーの入力信号端子に設定
●割り込みの許可
メイン処理
●キー・スキャン周期（約10 ms）の判定
●キー・スキャン処理の呼び出し
●イベント処理
●モードの判定
●変換実行状態の判定
●変換A1モード制御処理の呼び出し
●変換A2モード制御処理の呼び出し
●変換Aモード制御処理の呼び出し
約10 ms周期
キー・スキャン処理
●キー状態の取り込み
●チャタリング除去
●キー・コードの作成
変換A1モード
変換A2モード
補正Aモード
変換A1モード制御処理
●変換実行状態の判定
●停止処理
●温度補正データ（code1'）測定
●A/D変換処理
●5点補正処理
●温度補正処理
●UART送信データ作成処理
●UARTデータ送信処理
●変換実行状態設定
変換A2モード制御処理
●変換実行状態の判定
●停止処理
●温度補正データ（code1'）測定
●A/D変換処理
●5点補正処理
●温度補正処理
●UART送信データ作成処理
●UARTデータ送信処理
●変換実行状態設定
補正Aモード制御処理
●変換実行状態の判定
●停止処理
●A/D変換処理
●UART送信データ作成処理
●UARTデータ送信処理
●温度補正データ（code1，code3）測定
●変換終了判定
●変換実行状態設定

4. 3. 2　DACを使用する場合の動作

使用する周辺の初期設定完了後，変数を初期化し，キー・スキャン処理，イベント処理，変換Bモード制御処理，および補正Bモード制御処理を呼び出します。なお，キー・スキャン処理は約10 ms周期で呼び出します。また変換Bモード制御処理，および補正Bモード制御処理は，現在のモードとシステムの状態に合わせて呼び出します。

(1) キー・スキャン処理

7つのキー入力ポートからキー状態をスキャンします。チャタリング除去回数は3回で，キーの状態をコード化します。

(2) イベント処理

イベント処理では，キー・コードに変化があった場合，キー・コードにしたがって，モード（変換B，補正B）の切り替え，または変換実行状態（変換実行中，スタンバイ中，停止中）の切り替えを行います。変換状態がスタンバイ中の場合のみモードの切り替えが可能です。

(3) 変換Bモード制御処理

変換Bモードに入ると，まず74HC165から変換開始値と変換終了値を読み込み，進行キーが押されると以下の処理を連続で行います。停止キーが押されるか，またはDAC出力に設定した値が読み込んだ変換終了値と等しくなると変換動作を終了します。

- DAC出力の設定
- 16ビットΔΣ型A/Dコンバータにて A/D変換
- 5点補正（5点補正用データ測定済みの場合のみ実行）
- 温度補正（5点補正完了かつ温度誤差があった場合のみ実行）
- UARTでA/D変換結果，5点補正結果，および温度補正結果を出力[注]

DACへ設定する値は，A/D変換をするたびにインクリメントされます。読み込んだ変換開始値から変換終了値までを順にDAC出力に設定し，A/D変換します。変換終了値＜変換開始値だった場合，DAC設定値をラウンドさせて変換開始値から変換終了値までをDAC出力に設定します。

温度補正は，変換Bモードに入った直後に測定した5点補正用データ（code1）に，温度誤差があった場合に行います。code1の測定はDACにana1を設定して行います。

A/D変換結果，5点補正結果，温度補正結果はUARTで出力しますが，5点補正，温度補正がそれぞれ行われなかった場合は，送信データの該当部分が「＊＊＊＊」となります[注]。

注　UARTの送信データの詳細については，4.4　UART**送信データのフォーマット**を参照してください。

(4) 補正Bモード制御処理

補正Bモードは，5点補正用データを測定するモードです。進行キーが押されると以下の処理を5回連続で行います。停止キーが押されるか，または以下の処理が5回終わると，変換動作を終了します。

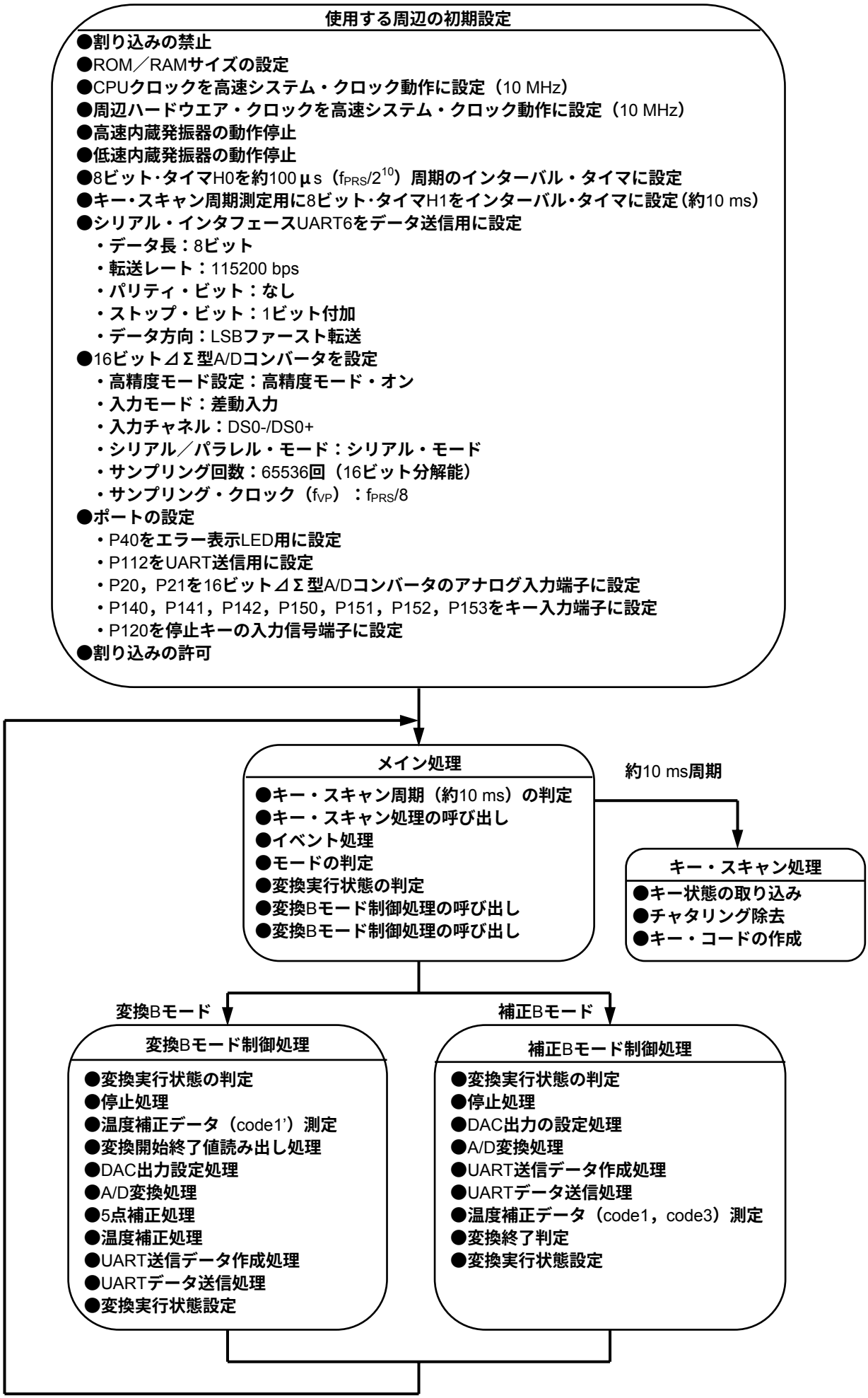
- DAC出力の設定
- 16ビットΔΣ型A/Dコンバータにて A/D変換
- UARTでA/D変換結果を出力[注]

ana1～ana5をDAC出力に順に設定し，code1～code5を測定します。code1～code5がcode1＜code3＜code5となっていなければ，エラー表示用LEDを点灯します。5点補正用データが正常に測定できた場合，code1およびcode3を5点補正にて補正し，温度補正用データとして保存します。

A/D変換，5点補正，温度補正の結果はUARTで出力します。ただし5点補正，温度補正がそれぞれ行われなかった場合，送信データの該当部分は「＊＊＊＊」となります[注]。停止キーが押されると変換動作を終了します。

注　UARTの送信データの詳細については，4.4　UART**送信データのフォーマット**を参照してください。

詳細については，次の状態遷移図（ステート・チャート）に示します。


使用する周辺の初期設定
●割り込みの禁止
●ROM／RAMサイズの設定
●CPUクロックを高速システム・クロック動作に設定（10 MHz）
●周辺ハードウエア・クロックを高速システム・クロック動作に設定（10 MHz）
●高速内蔵発振器の動作停止
●低速内蔵発振器の動作停止
●8ビット・タイマH0を約100μs（fPRS/2^10）周期のインターバル・タイマに設定
●キー・スキャン周期測定用に8ビット・タイマH1をインターバル・タイマに設定（約10 ms）
●シリアル・インタフェースUART6をデータ送信用に設定
・データ長：8ビット
・転送レート：115200 bps
・パリティ・ビット：なし
・ストップ・ビット：1ビット付加
・データ方向：LSBファースト転送
●16ビットΔΣ型A/Dコンバータを設定
・高精度モード設定：高精度モード・オン
・入力モード：差動入力
・入力チャネル：DS0-/DS0+
・シリアル／パラレル・モード：シリアル・モード
・サンプリング回数：65536回（16ビット分解能）
・サンプリング・クロック（fVP）：fPRS/8
●ポートの設定
・P40をエラー表示LED用に設定
・P112をUART送信用に設定
・P20，P21を16ビットΔΣ型A/Dコンバータのアナログ入力端子に設定
・P140，P141，P142，P150，P151，P152，P153をキー入力端子に設定
・P120を停止キーの入力信号端子に設定
●割り込みの許可
メイン処理
●キー・スキャン周期（約10 ms）の判定
●キー・スキャン処理の呼び出し
●イベント処理
●モードの判定
●変換実行状態の判定
●変換Bモード制御処理の呼び出し
●変換Bモード制御処理の呼び出し
約10 ms周期
キー・スキャン処理
●キー状態の取り込み
●チャタリング除去
●キー・コードの作成
変換Bモード
補正Bモード
変換Bモード制御処理
●変換実行状態の判定
●停止処理
●温度補正データ（code1'）測定
●変換開始終了値読み出し処理
●DAC出力設定処理
●A/D変換処理
●5点補正処理
●温度補正処理
●UART送信データ作成処理
●UARTデータ送信処理
●変換実行状態設定
補正Bモード制御処理
●変換実行状態の判定
●停止処理
●DAC出力の設定処理
●A/D変換処理
●UART送信データ作成処理
●UARTデータ送信処理
●温度補正データ（code1，code3）測定
●変換終了判定
●変換実行状態設定

4. 4　UART送信のデータ・フォーマット

UART6により送信するデータの内容を説明します。

UARTの設定は下記のとおりです。

・ボー・レート：115200 bps

・データのキャラクタ長：8ビット

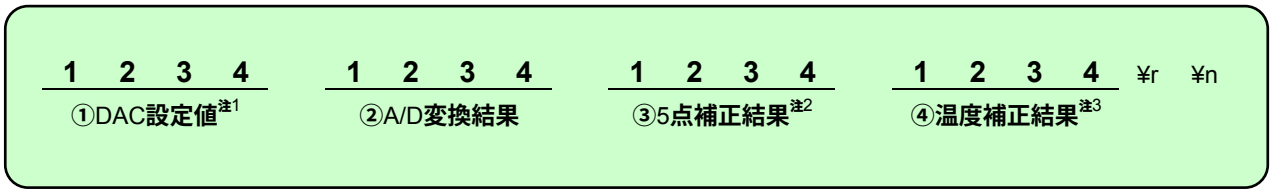
・パリティ・ビット：出力しない

・ストップ・ビット数：1

データの送信は，A/D変換1回につき1回行います。1回の送信データの長さは21バイトです。DAC設定値，A/D変換結果，5点補正結果，温度補正結果をそれぞれ16進数でASCIIコードに変換して送信します。

データの内容は図4－1に示します。

図4－1　UART送信のデータ・フォーマット



注1．DACを使用しない場合，データは「＊＊＊＊」となります。

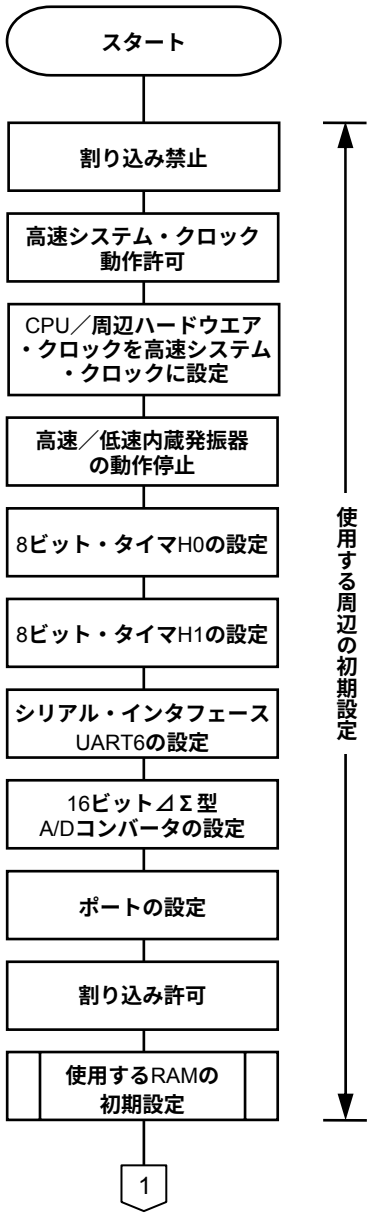
2．5点補正を行わない場合，データは「＊＊＊＊」となります。

3．温度補正を行わない場合，データは「＊＊＊＊」となります。

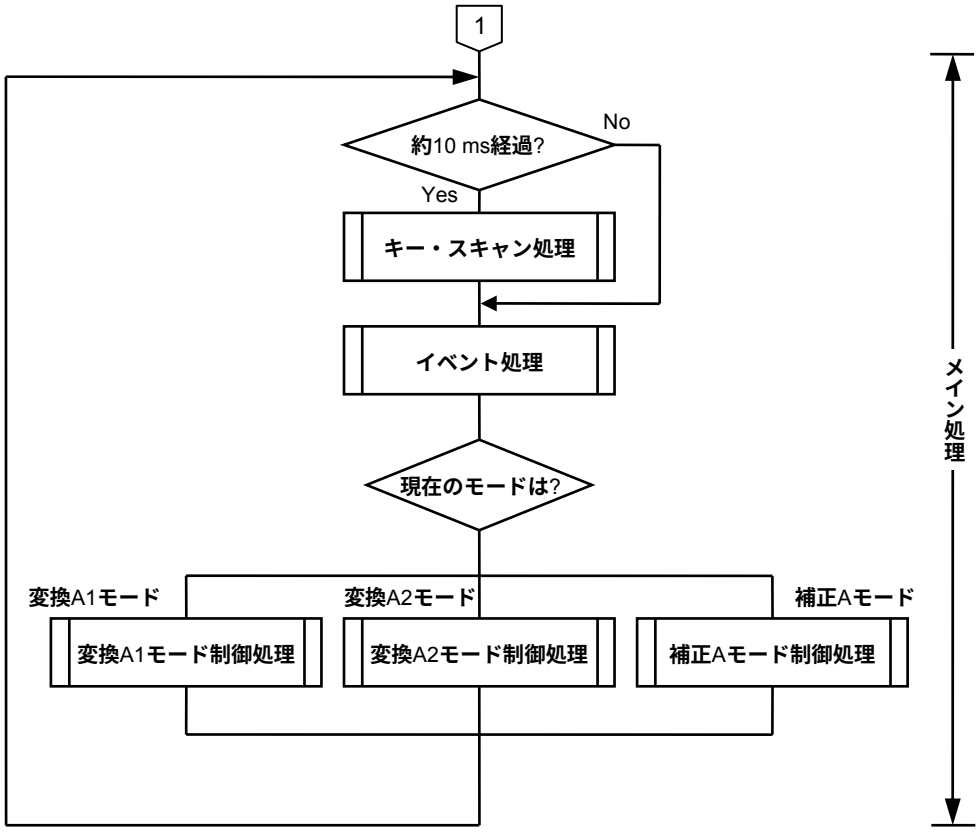
4.5　フロー・チャート

このサンプル・プログラムのフロー・チャートを次に示します。

＜リセット解除後の初期化処理＞



●DACを使用しない場合のメイン処理



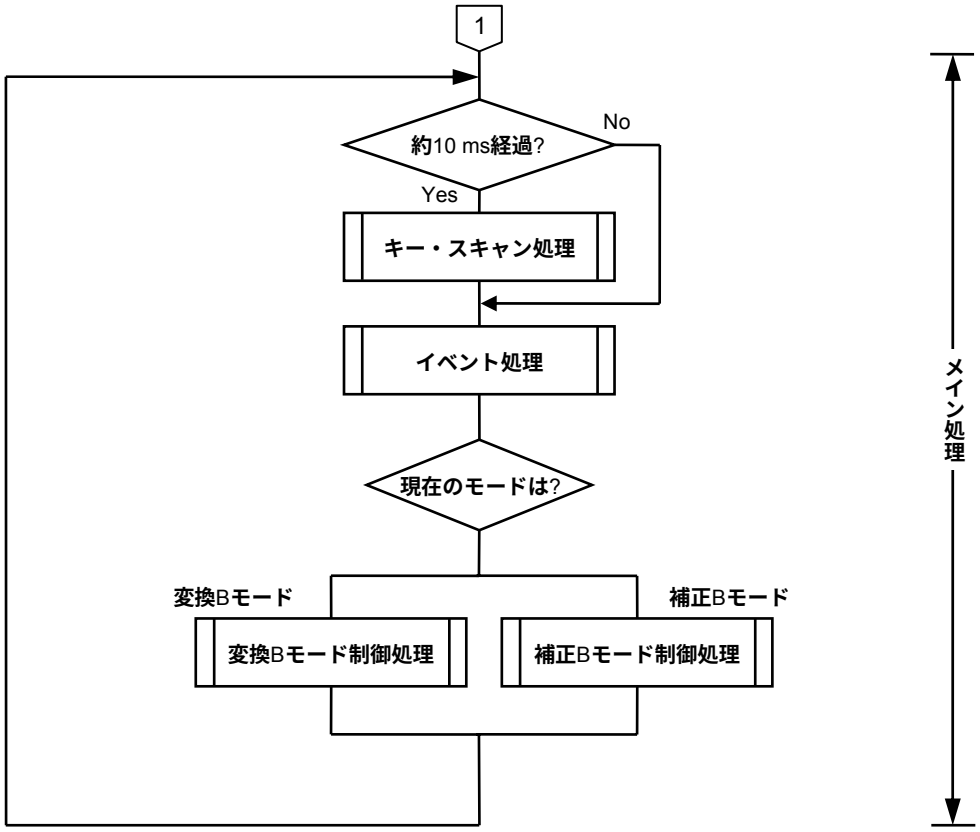
注意1. **オプション・バイトは，**RA78K0**のリンカ・オプションにて設定してください。設定の仕方については，**RA78K0　**アセンブラ・パッケージ　ユーザーズ・マニュアルを参照してください。**

オプション・バイトでは，次の内容を設定します。

- **ウォッチドッグ・タイマの動作**
- **リセット解除時（電源立ち上げ時）の**LVI**の設定**
- **オンチップ・デバッグの動作制御**

2. **本サンプル・プログラムには，アナログ入力に**DAC**を使用する場合と使用しない場合の両方の処理が含まれています。動作環境に合わせて，不要な処理は削除してください。**

●DACを使用する場合のメイン処理

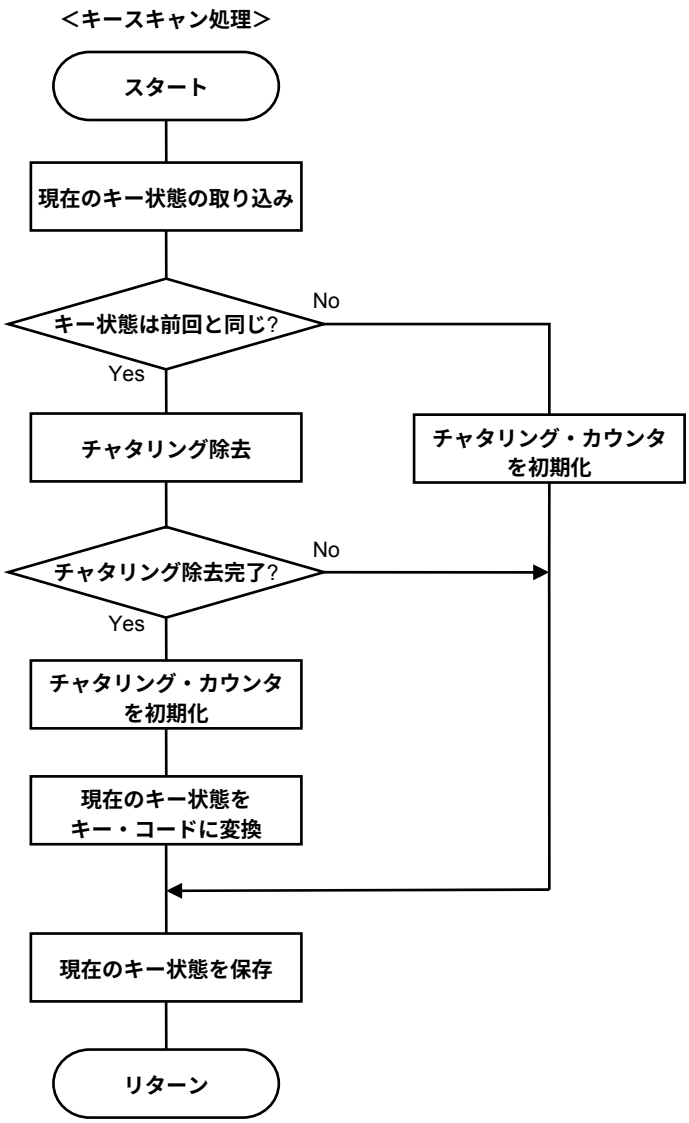


注意1.　**オプション・バイトは，**RA78K0**のリンカ・オプションにて設定してください。設定の仕方については，**RA78K0　**アセンブラ・パッケージ　ユーザーズ・マニュアルを参照してください。**

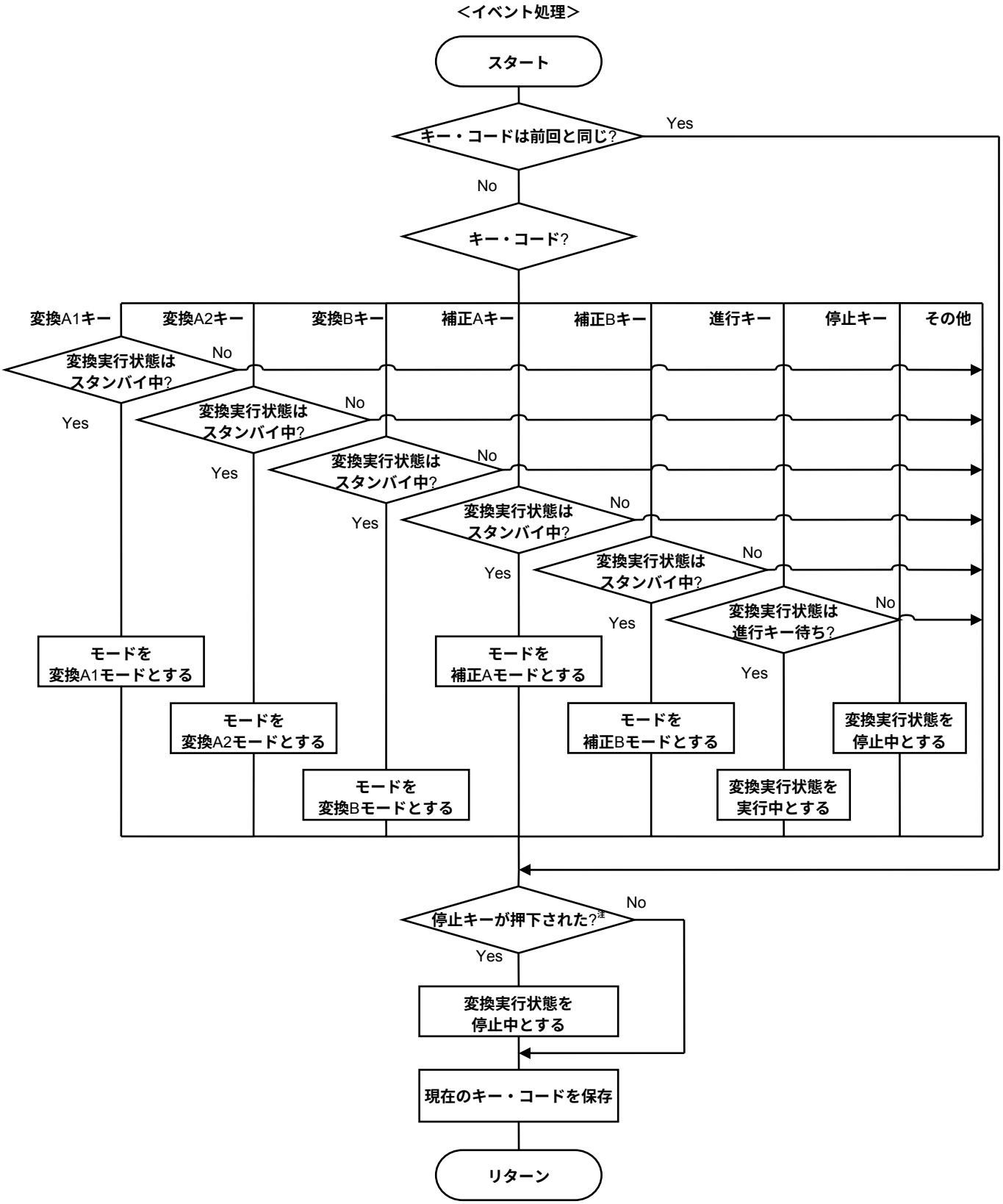
オプション・バイトでは，次の内容を設定します。

- **ウォッチドッグ・タイマの動作**
- **リセット解除時（電源立ち上げ時）の**LVI**の設定**
- **オンチップ・デバッグの動作制御**

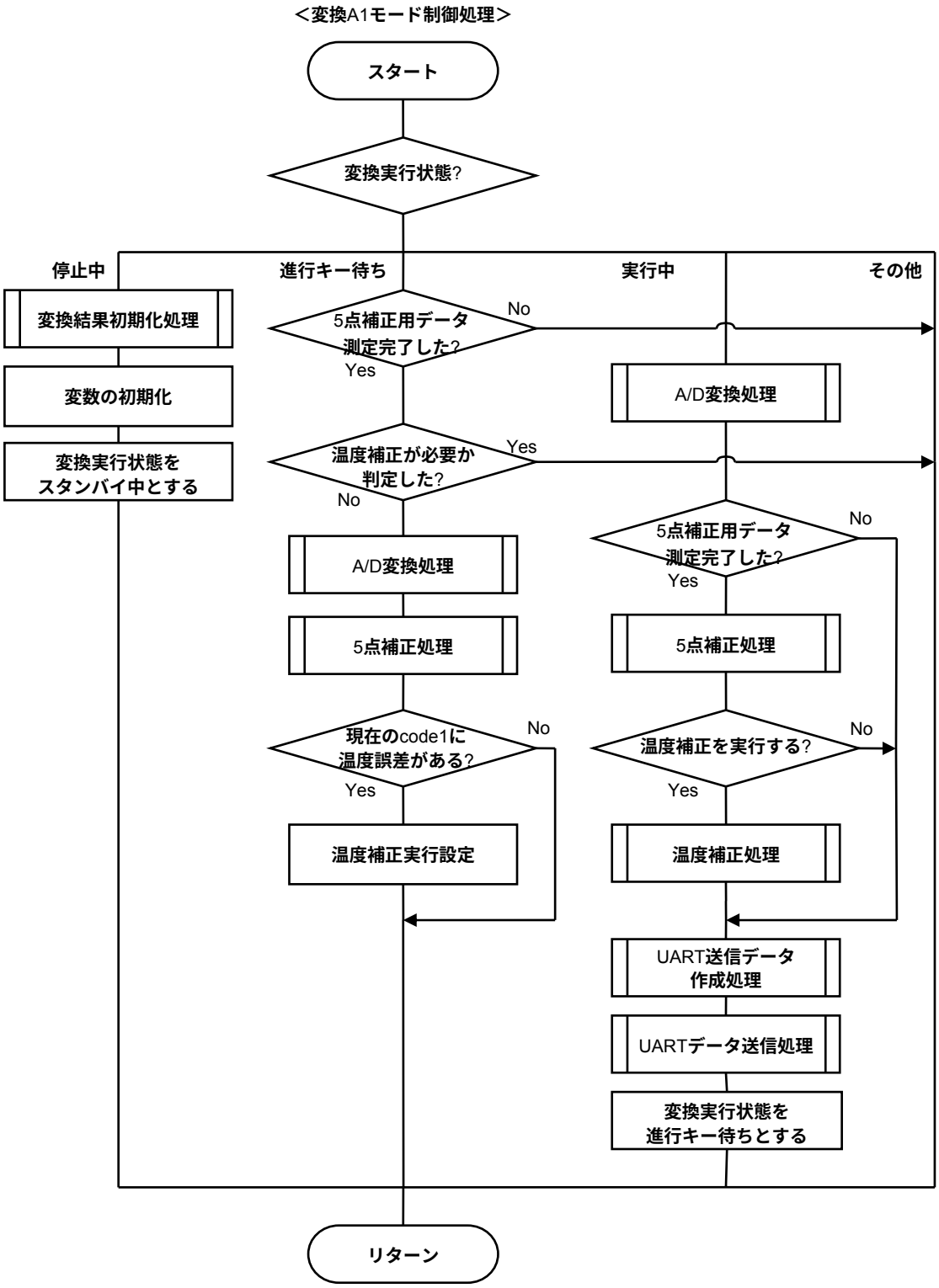
2.　**本サンプル・プログラムには，アナログ入力に**DAC**を使用する場合と使用しない場合の両方の処理が含まれています。動作環境に合わせて，不要な処理は削除してください。**

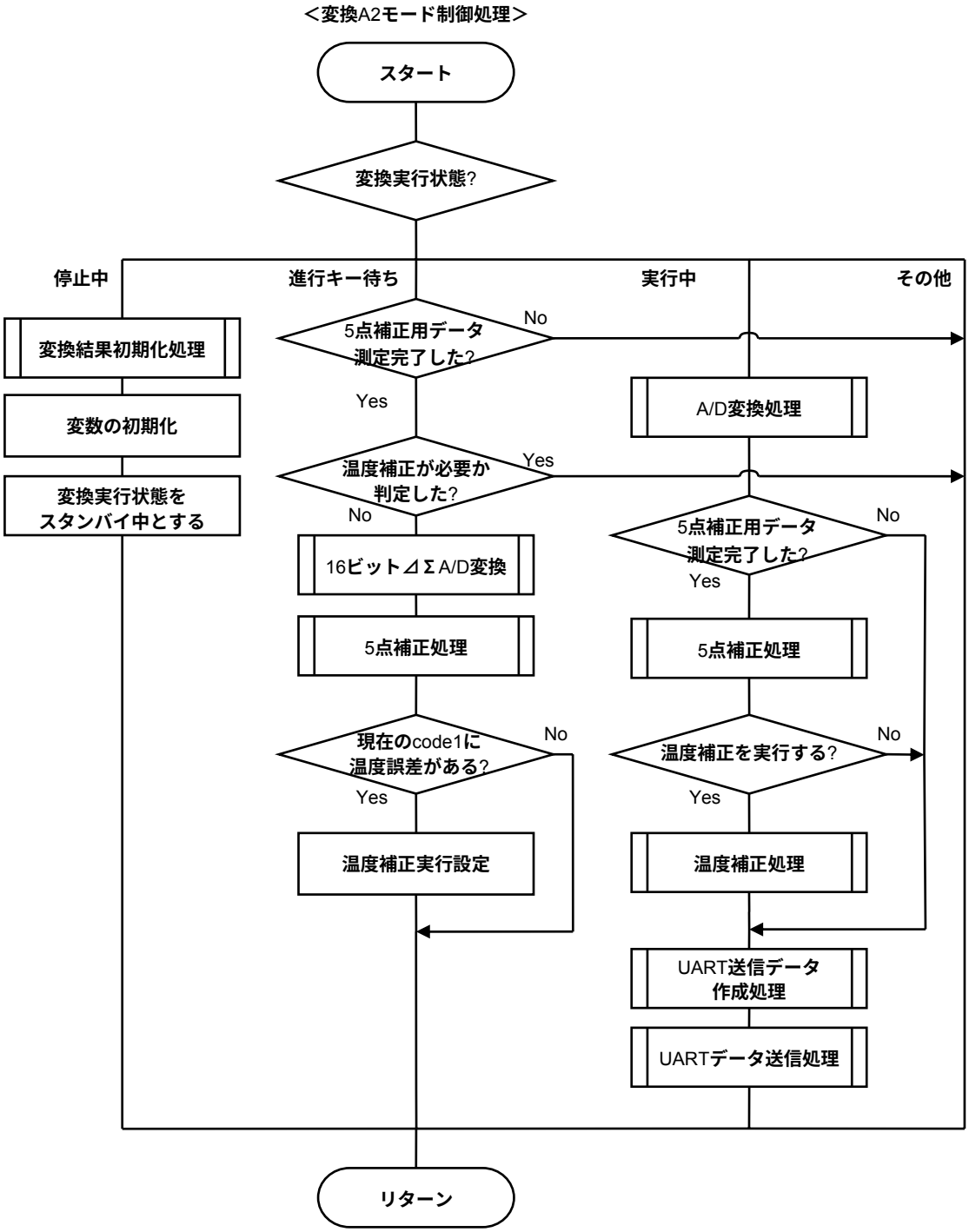


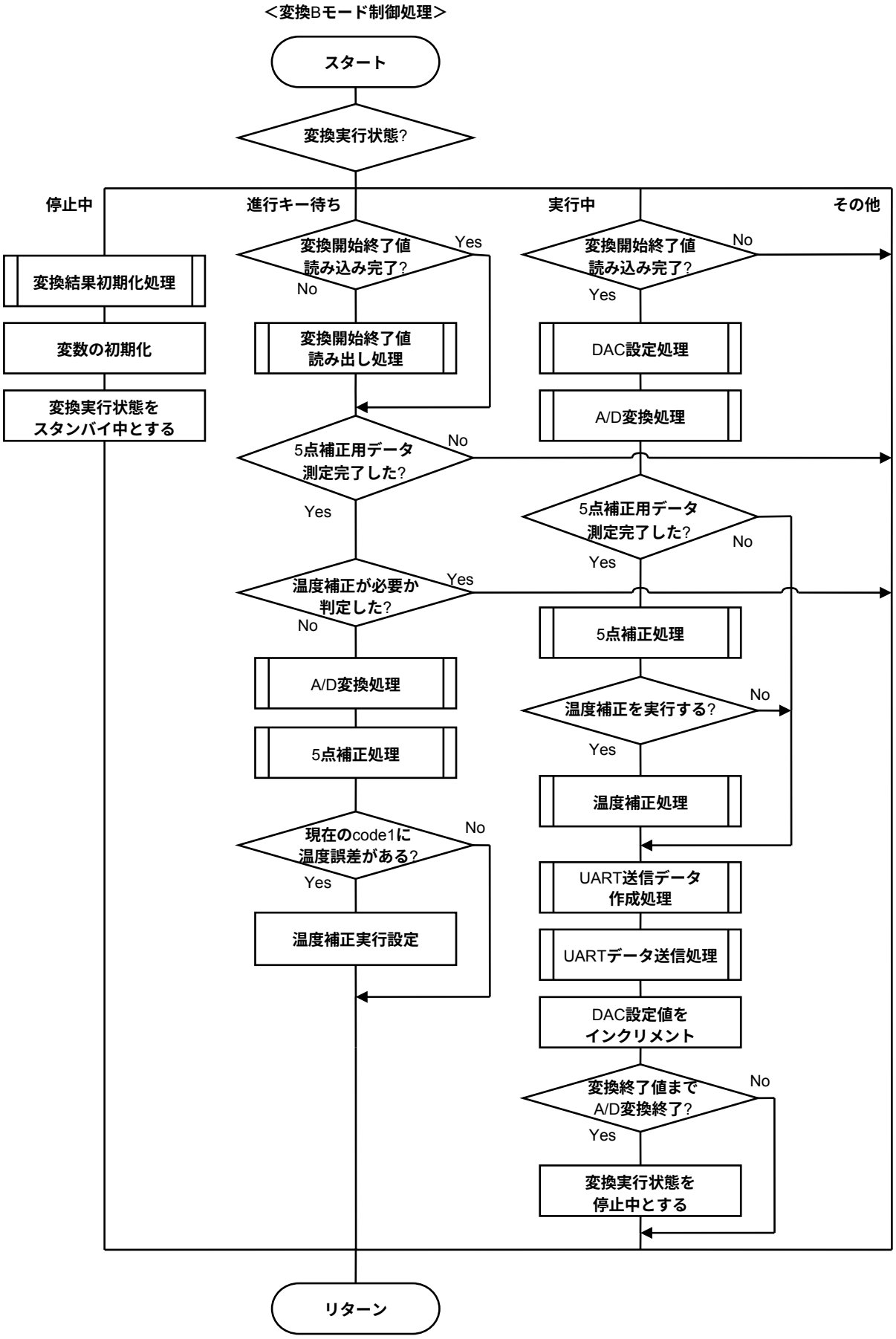
注意　キーの多重押しは無効です。

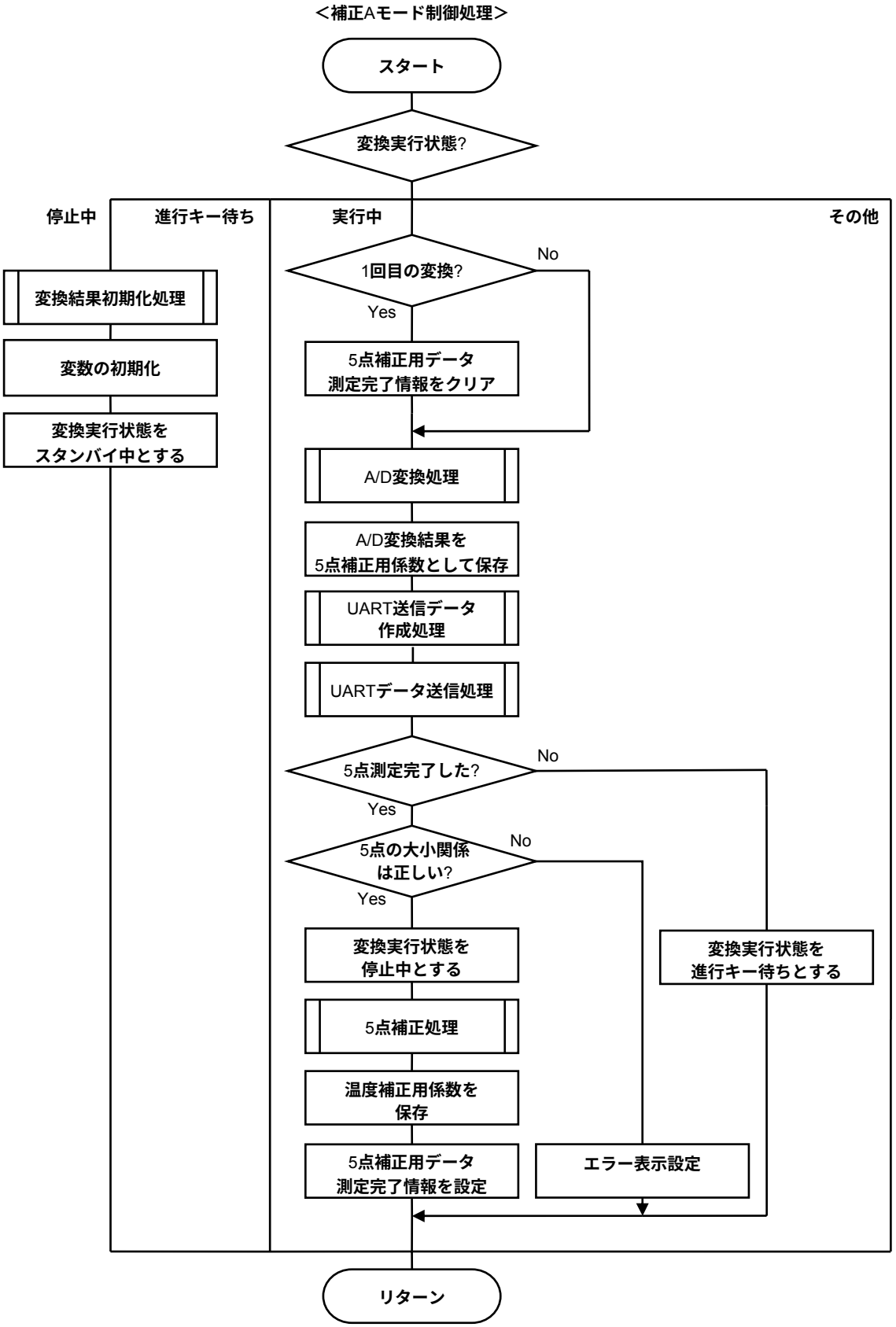


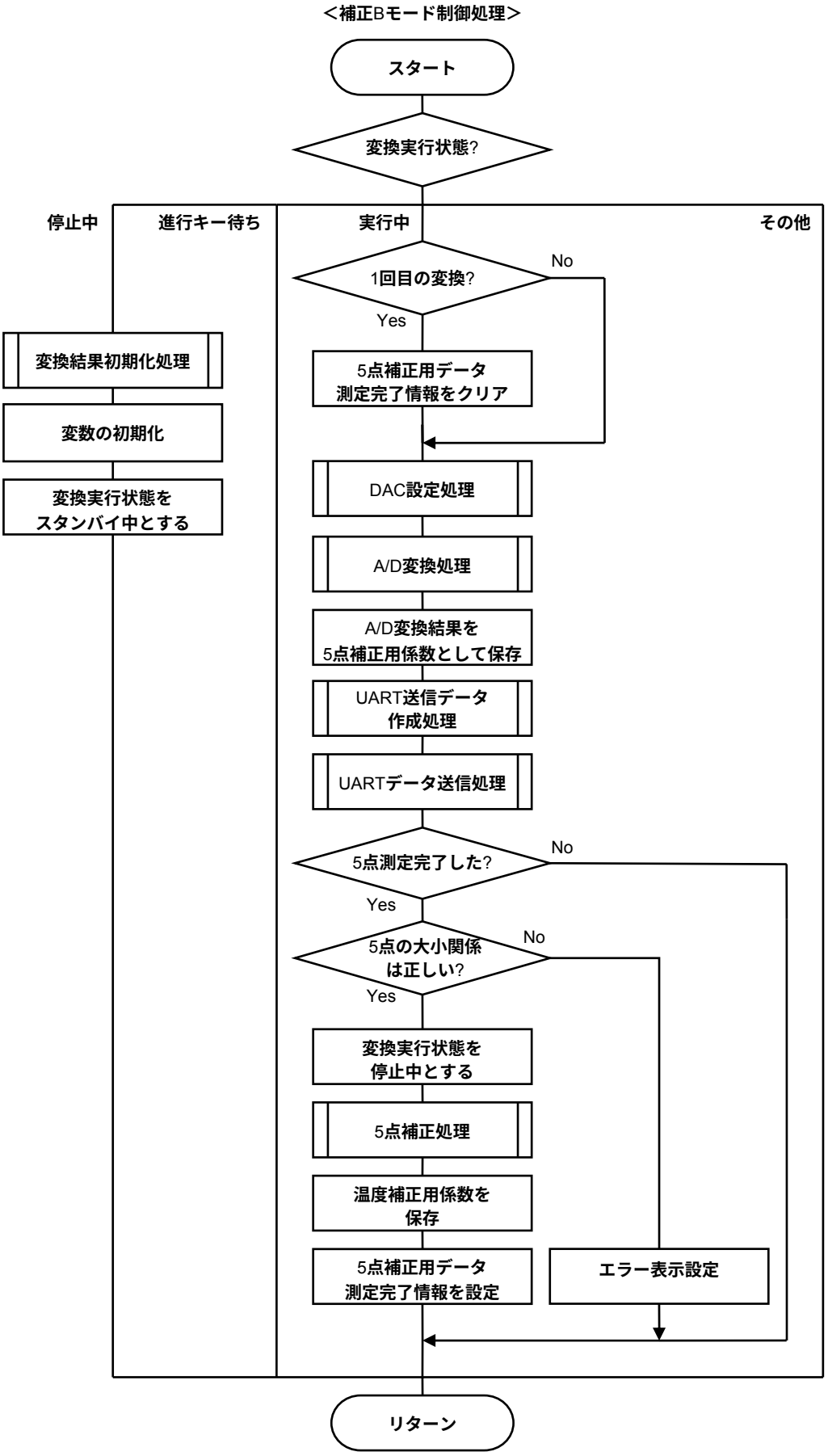
注　停止キーが押下されたかを，割り込み要求フラグ（PIF0）により判定します。キー・スキャン処理は約10 ms周期で実行されますが，A/D変換処理などにより，遅れることがあります。


＜変換A1モード制御処理＞
スタート
変換実行状態?
停止中
進行キー待ち
実行中
その他
変換結果初期化処理
5点補正用データ
測定完了した?
No
Yes
変数の初期化
A/D変換処理
変換実行状態を
スタンバイ中とする
温度補正が必要か
判定した?
Yes
No
5点補正用データ
測定完了した?
No
Yes
A/D変換処理
5点補正処理
5点補正処理
現在のcode1に
温度誤差がある?
No
Yes
温度補正を実行する?
No
Yes
温度補正実行設定
温度補正処理
UART送信データ
作成処理
UARTデータ送信処理
変換実行状態を
進行キー待ちとする
リターン

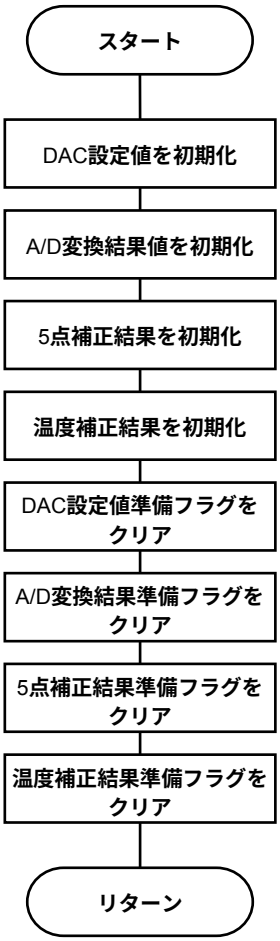

＜変換A2モード制御処理＞
スタート
変換実行状態?
停止中
進行キー待ち
実行中
その他
変換結果初期化処理
変数の初期化
変換実行状態を
スタンバイ中とする
5点補正用データ
測定完了した?
No
Yes
温度補正が必要か
判定した?
Yes
No
16ビットΔΣA/D変換
5点補正処理
現在のcode1に
温度誤差がある?
No
Yes
温度補正実行設定
A/D変換処理
5点補正用データ
測定完了した?
No
Yes
5点補正処理
温度補正を実行する?
No
Yes
温度補正処理
UART送信データ
作成処理
UARTデータ送信処理
リターン


＜変換Bモード制御処理＞
スタート
変換実行状態?
停止中
進行キー待ち
実行中
その他
変換開始終了値
読み込み完了?
Yes
No
変換結果初期化処理
変数の初期化
変換実行状態を
スタンバイ中とする
変換開始終了値
読み出し処理
DAC設定処理
A/D変換処理
5点補正用データ
測定完了した?
温度補正が必要か
判定した?
5点補正処理
温度補正を実行する?
温度補正処理
現在のcode1に
温度誤差がある?
温度補正実行設定
UART送信データ
作成処理
UARTデータ送信処理
DAC設定値を
インクリメント
変換終了値まで
A/D変換終了?
変換実行状態を
停止中とする
リターン


＜補正Aモード制御処理＞
スタート
変換実行状態?
停止中
進行キー待ち
実行中
その他
1回目の変換?
No
Yes
変換結果初期化処理
変数の初期化
変換実行状態を
スタンバイ中とする
5点補正用データ
測定完了情報をクリア
A/D変換処理
A/D変換結果を
5点補正用係数として保存
UART送信データ
作成処理
UARTデータ送信処理
5点測定完了した?
No
Yes
5点の大小関係
は正しい?
No
Yes
変換実行状態を
停止中とする
変換実行状態を
進行キー待ちとする
5点補正処理
温度補正用係数を
保存
5点補正用データ
測定完了情報を設定
エラー表示設定
リターン


＜補正Bモード制御処理＞
スタート
変換実行状態?
停止中
進行キー待ち
実行中
その他
1回目の変換?
No
Yes
変換結果初期化処理
変数の初期化
変換実行状態を
スタンバイ中とする
5点補正用データ
測定完了情報をクリア
DAC設定処理
A/D変換処理
A/D変換結果を
5点補正用係数として保存
UART送信データ
作成処理
UARTデータ送信処理
5点測定完了した?
No
Yes
5点の大小関係
は正しい?
No
Yes
変換実行状態を
停止中とする
5点補正処理
温度補正用係数を
保存
5点補正用データ
測定完了情報を設定
エラー表示設定
リターン

＜変換結果初期化処理＞

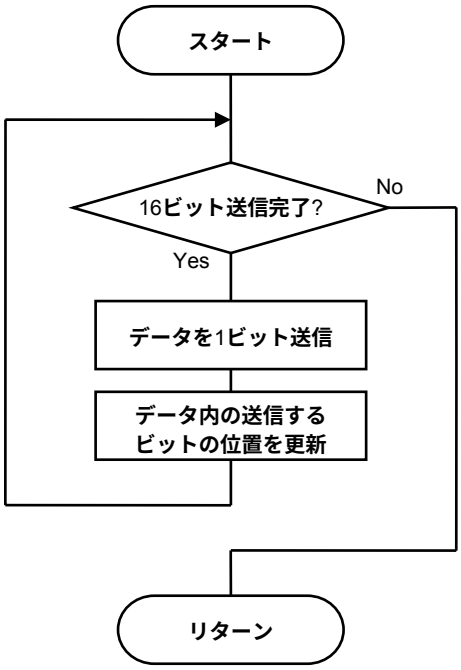

スタート
DAC設定値を初期化
A/D変換結果値を初期化
5点補正結果を初期化
温度補正結果を初期化
DAC設定値準備フラグをクリア
A/D変換結果準備フラグをクリア
5点補正結果準備フラグをクリア
温度補正結果準備フラグをクリア
リターン

＜変換開始終了値読み出し処理＞

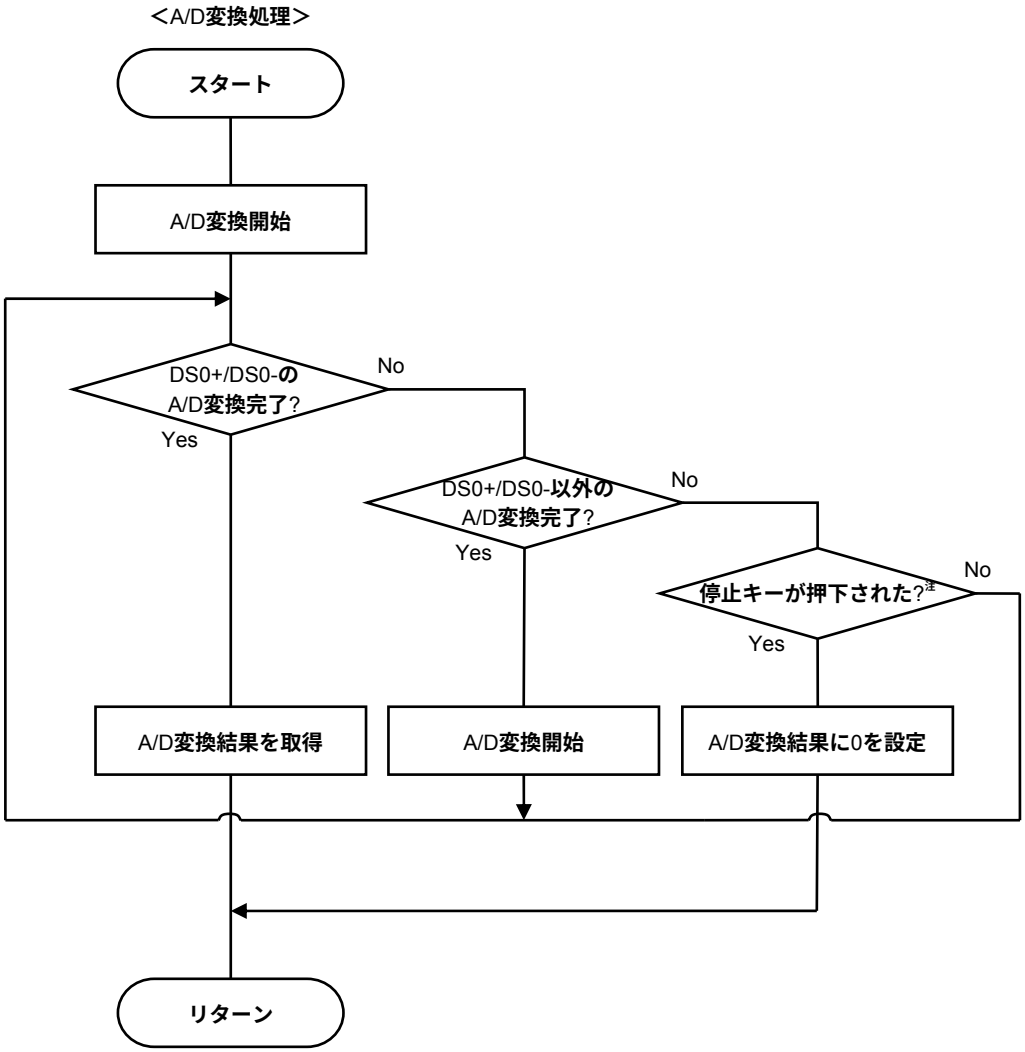
＜DAC設定処理＞


スタート
DAC1の設定値より
DAC0の設定値を算出
DAC1送信開始設定
ソフトウエア・
シリアル送信処理
DAC1送信終了設定
DAC0送信開始設定
ソフトウエア・
シリアル送信処理
DAC0送信終了設定
回路の安定待ち（500μs）
リターン

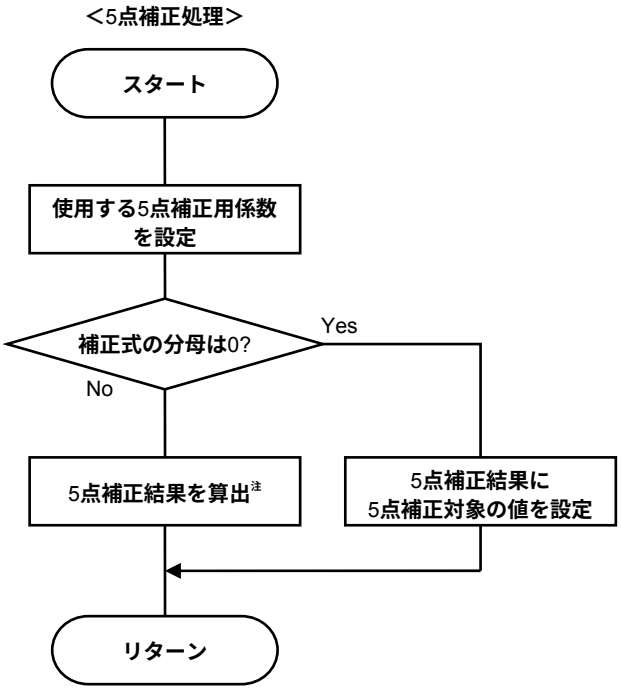
＜ソフトウエア・シリアル送信処理＞



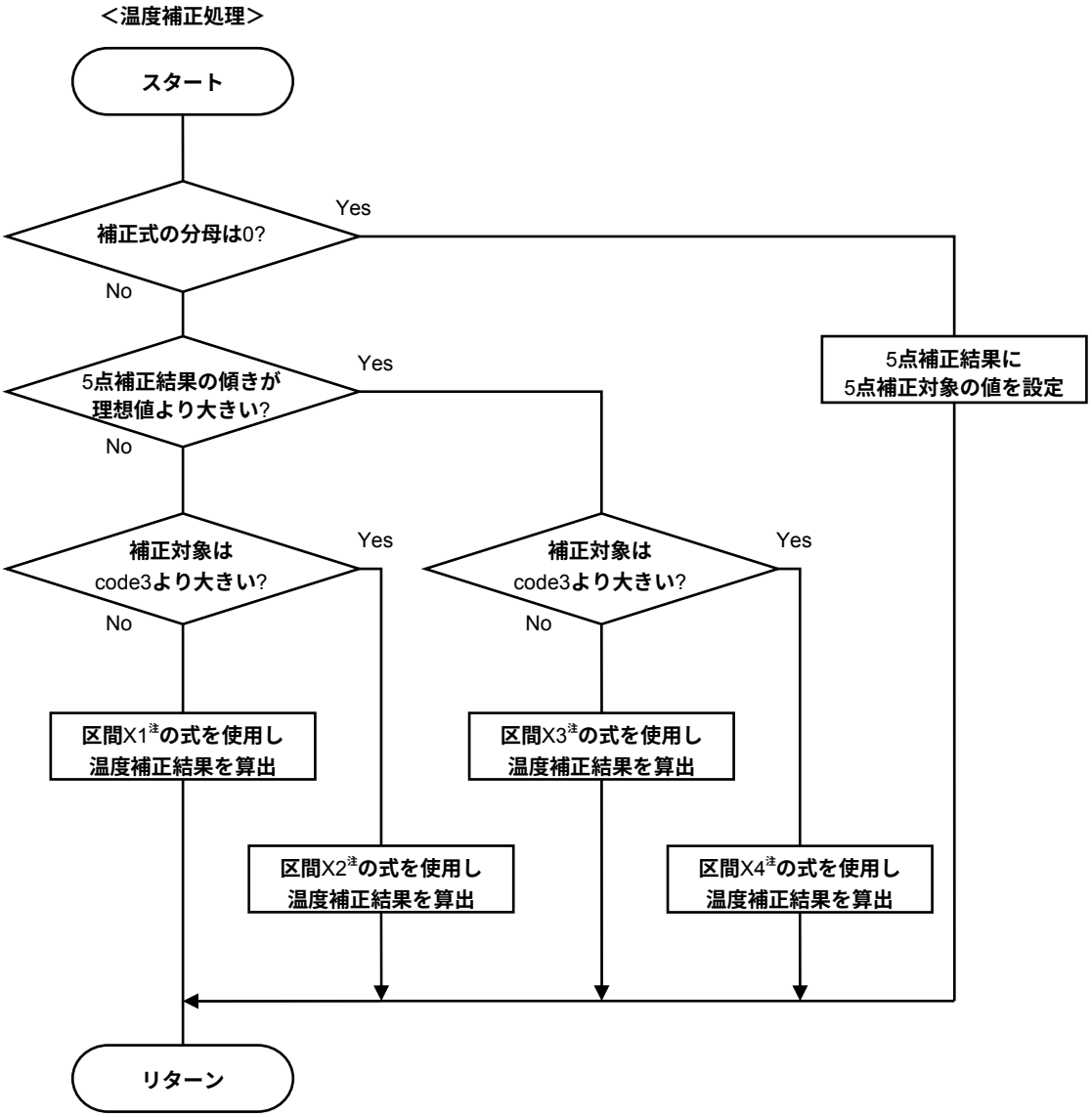
注意　データはMSB**から順に**1**ビットずつ送信します。**



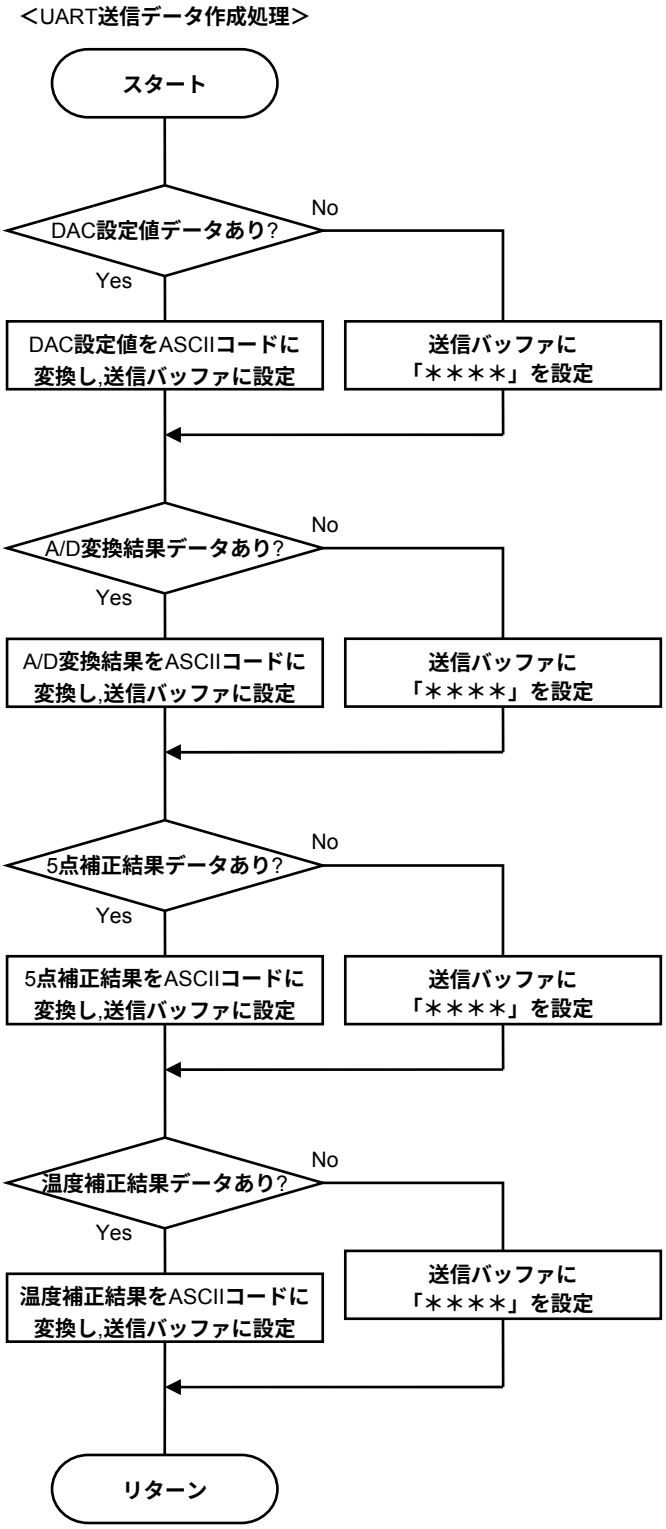
注　停止キーが押下されたかを割り込み要求フラグ（PIF0）により判定します。キー・スキャン処理は約10 ms周期で実行されますが，A/D変換処理などにより遅れることがあります。



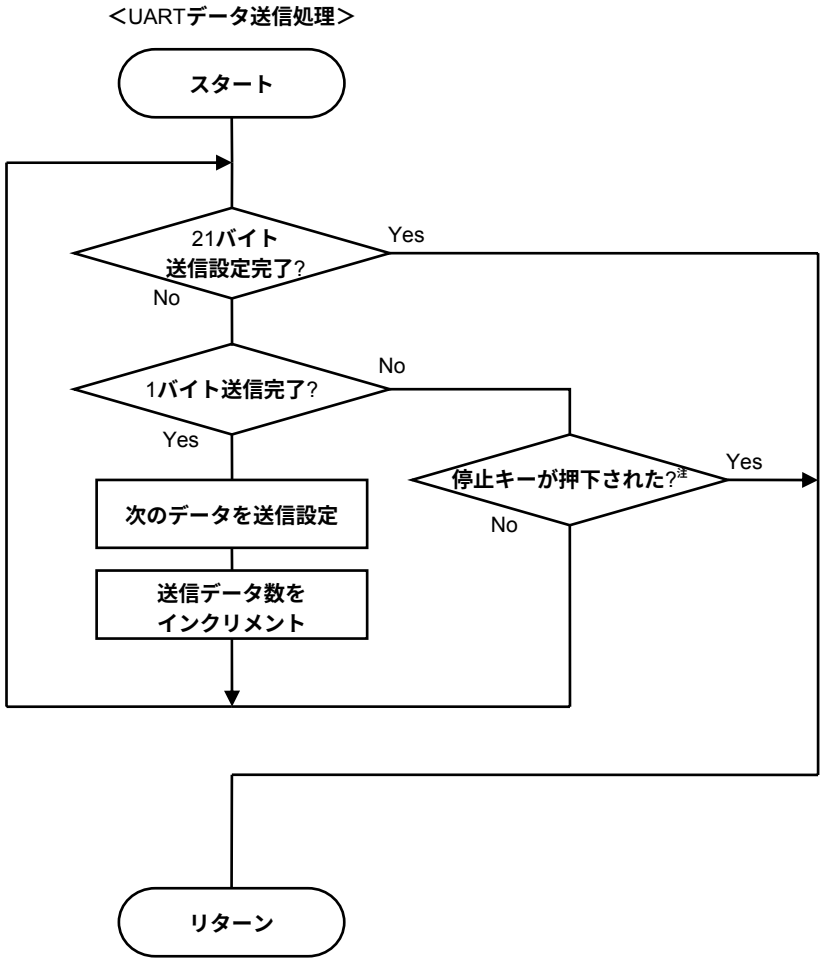
注　5点補正演算式の詳細は，2.1　5**点補正について**を参照してください。



注　温度補正演算式の詳細は，2.2　**温度補正について**を参照してください。



注意　UART**の送信データの詳細については，**4.4　UART**送信データのフォーマットを参照してください。**



注　停止キーが押下されたかを割り込み要求フラグ（PIF0）により判定します。キー・スキャン処理は約10 ms周期で実行されますが，A/D変換処理などにより遅れることがあります。

注意　データはMSBから順に1バイトずつ送信します。

第5章　設定方法について

この章では，ΔΣA/D処理について説明します。

レジスタ設定方法などの詳細については，各製品のユーザーズ・マニュアル（78K0/LE3，78K0/LF3）を参照してください。

5. 1　使用する周辺の初期設定

リセット解除後の初期化処理では，次のSFRの初期設定を行います。

① A/Dポート・コンフィギュレーション・レジスタ0（ADPC0）
アナログ入力として使用する端子を指定します。

② ポート・モード・レジスタ2（PM2）
アナログ入力ポートを入力モードに設定します。

③ A/Dコンバータ電源（ADDPON）
A/Dコンバータ電源をONします。

④ 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ1（ADDCTL1）
サンプリング回数，サンプリング・クロック，動作モードを設定します。

⑤ 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ0（ADDCTL0）
A/D動作をΔΣに設定します。

⑥ A/Dコンバータ電源ON→動作許可まで5 μs以上空ける。
A/D電源の安定を待ちます。

⑦ 変換動作許可ビット（ADDCE）
A/D変換動作を許可します。

⑧ A/D変換完了割り込み要求フラグ（DSADIF）
A/D変換完了に使用するため，割り込み要求フラグをクリアしておきます。

```
/*-------------------------------------------------------------------------
        16bit⊿Σ型A/Dコンバータ
-------------------------------------------------------------------------*/
①       ADPC0 = 0b00000000;             /* A/Dポート・コンフィギュレーション・レジスタ0 */
                /*||||++++--------------- ADPC03/ADPC02/ADPC01/ADPC00:P20-P27 アナログ入力(ΔΣ型) 設定 */
                /*++++------------------- <0000固定> */

②       PM2 =   0b11111111;                     /* PM2入出力設定 */
                /*|||||||+--------------- PM20:入力(1) DS0-として使用 */
                /*||||||+---------------- PM21:入力(1) DS0+として使用 */
                /*|||||+----------------- PM22:入力(1) DS1-として使用 */
                /*||||+------------------ PM23:入力(1) DS1+として使用 */
                /*|||+------------------- PM24:入力(1) DS2-として使用 */
                /*||+-------------------- PM25:入力(1) DS2+として使用 */
                /*|+--------------------- PM26:入力(1) REF-として使用 */
                /*+---------------------- PM27:入力(1) REF+として使用 */
        P2 =    0b00000000;                     /* P2初期値設定 */
                /*++++++++--------------- P27/P26/P25/P24/P23/P22/P21/P20:Lo(0) */

③       ADDPON = 1;                             /* A/Dコンバータ電源ON */
④       ADDCTL1=0b01100111;             /* 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ1 */
                /*|||||+++--------------- ADDN2/ADDN1/ADDN0:サンプリング回数(分解能) 65536回(16bit) 指定 */
                /*|||++----------------- <00固定> */
                /*||+------------------- ADDTS:シリアルモード設定 */
                /*++-------------------- ADDFS1/ADDFS0: サンプリングクロック fPRS/8 選択*/

⑤       ADDCTL0=0b10110000;             /* 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ0 */
                /*||||||++--------------- ADDS1/ADDS0:アナログ入力 DS0+/DS0- 指定 */
                /*||||++----------------- <00固定> */
                /*|||+------------------- AINMOD:入力モード 差動入力 指定 */
                /*||+-------------------- HAC:高精度モードON */
                /*|+--------------------- ADDCE:変換停止 */
                /*+---------------------- ADDPON:A/Dコンバータ電源ON */

⑥       for( i = 20; i < 0; i--);       /* A/Dコンバータ電源ON→動作許可まで5μsec以上空ける */
⑦       ADDCE = 1;                      /* 変換動作許可 */
⑧       DSADIF = 0;                             /* A/D変換完了割り込み要求フラグクリア */
```

5. 2　A/D変換処理

ΔΣA/D変換処理では，次の動作を行います。

アナログ入力チャネル0（DS0+/DS0-）に印加されている電圧のA/D変換動作を行います。A/D変換の詳細は，各製品のユーザーズ・マニュアル（78K0/LE3，78K0/LF3）を参照してください。

① A/D変換動作を許可します。
② A/D変換終了タイミングをA/D変換完了割り込み要求で見ているので，割り込み要求をクリアしておきます。
③ 本アプリケーションではA/D変換は常時動作させているので，A/D変換終了を待って，変換結果を読み込みます。
本アプリケーションで使用しているのは，チャネル0のみなので，変換終了チャネル（ADDSTR）をチェックする必要はありませんが，念のため，ADDSTRが目的のチャネル0であるかを確認しています。
もし，他のチャネルであった場合は再変換を行います。
④ 本アプリケーションではA/D変換終了までループしますので，途中終了の方法として，外部割り込みをチェックします。
⑤ A/D変換結果を戻り値として返します。

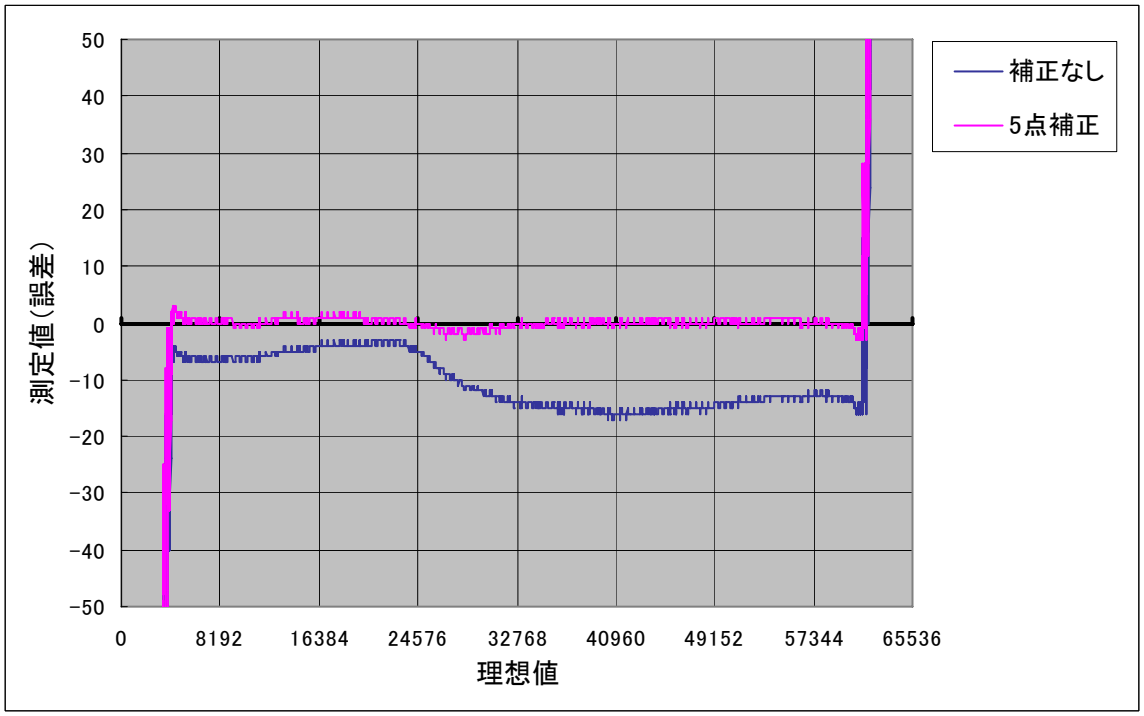
```
/************************************************************************

        16bitΔΣ型A/D変換

------------------------------------------------------------------------
        ΔΣ型A/D変換の開始し、A/D変換の結果を返します。
************************************************************************/
static unsigned short fn_GetAD(void)
{
        unsigned short ret = 0;             /* A/D変換結果用戻り値 */

        /* A/D変換開始 */
①       ADDCE = 1;                                          /* 変換動作許可 */
②       DSADIF = 0;                                         /* A/D変換完了割り込み要求クリア */

        /* A/D変換結果取得 */
        while(1){   /* A/D変換が完了するor停止キーが押されるまで待つ */
                if(( DSADIF )&&( ADDSTR == 0 )){  /* DS0+のA/D変換が完了した場合 */
                        ret = ADDCR;                        /* A/D変換結果読み出し */
③                       break;
                }else if(( DSADIF )&&( ADDSTR != 0 )){      /* 目的のチャンネルでない値を変換した場合 */
                        ADDCE = 1;                          /*   A/D変換やり直し   */
                        DSADIF = 0;
                }else if(PIF0){                             /* 停止キーが押された場合 */
④                       ADDCE = 0;                          /* A/D変換動作停止 */
                        break;                              /* A/D変換中止 */
                }else
                        ;
        }

⑤       return      ( ret );                                /* A/D変換結果を返す */
}}
```

第6章　デバイスでの動作確認例

この章では，A/D変換値の補正例を示します。

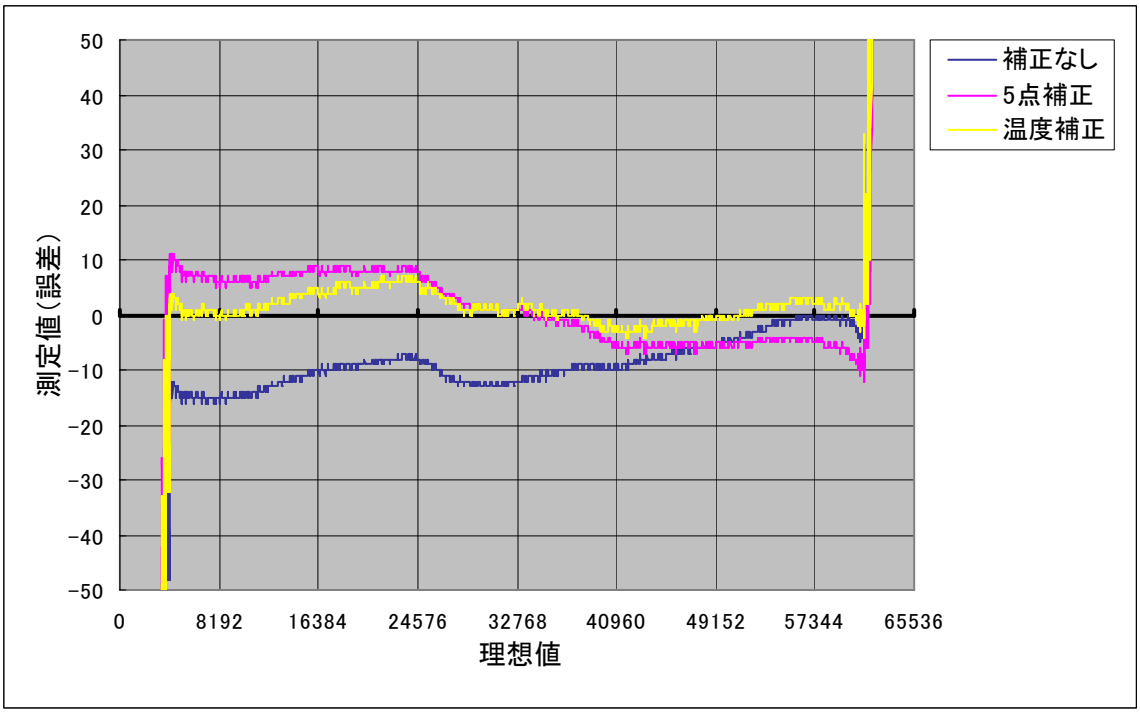
6. 1　5点補正

5点補正の補正例を示します。



6. 2　温度補正

25℃時の測定値を基準にして，0℃時に補正を行った温度補正例を示します。



第7章　関連資料

資　料　名		和文／英文
78K0/LE3　ユーザーズ・マニュアル		PDF
78K0/LF3　ユーザーズ・マニュアル		PDF
78K/0シリーズ　命令編　ユーザーズ・マニュアル		PDF
RA78K0　アセンブラ・パッケージ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
CC78K0　Cコンパイラ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
PM+　プロジェクト・マネージャ　ユーザーズ・マニュアル		PDF

付録A　プログラム･リスト

プログラム･リスト例として，78K0/LF3マイクロコントローラのソース･プログラムを次に示します。

● main.c （C言語版）

```
/************************************************************************************

        NEC Electronics      78K0/Lx3シリーズ

************************************************************************************
        78K0/LF3シリーズ          サンプル･プログラム
************************************************************************************
        16bit⊿Σ型ADC精度補正AN開発
************************************************************************************
【履歴】
        2008.1.--        新規作成
************************************************************************************

【概要】
  このサンプル･プログラムは，アナログ入力を16ビット⊿Σ型A/Dコンバータによ りA/D
 変換し， A/D値を補正するプログラムです。測定したA/D値に対し，5点補正と，温度補
 正を行います。
  5点補正は，使用電圧に合わせて測定した5つの5点補正用データからA/D変換の入出力
 特性の近似関数を算出して行います。
  温度補正では，取得した温度補正用データと5点補正用データを用いて，5点補正結果
 の温度変化による誤差を補正します。


  本サンプル・プログラムは，変換A1，変換A2，変換B，補正A，補正Bの5つのモードから
 構成されています。

 ≪変換A1モード≫
   進行キーが押下されるごとにDAC以外から入力される電圧を1回A/D変換し，A/D値を補正
  します。

 ≪変換A2モード≫
   進行キーが押下されると停止キーが押下されるまで，DAC以外から入力される電圧をA/D
  変換し，A/D値を補正します。

 ≪変換Bモード≫
```

変換開始値と変換終了値を読み込み， DACへ出力電圧の設定し，DACからのアナログ入力をA/D変換し，A/D値の補正を行います。DACへ設定する出力電圧を開始値からインクリメントしながら，設定値が終了値となるまで連続で処理を行います。

≪補正Aモード≫

進行キーが押下されるごとにDAC以外から入力される電圧を1回A/D変換します。処理を5回行い，5点補正の演算に使用する5つの補正データを取得します。

≪補正Bモード≫

AVDDの電圧から算出した値をDACに設定し，DACからのアナログ入力をA/D変換します。処理を5回行い，5点補正の演算に使用する5つの補正データを取得します。

DAC以外からのアナログ入力をA/D変換する場合は変換A1，変換A2，および補正Aモードを使用します。DACからのアナログ入力をA/D変換する場合は変換B，および補正Bモードを使用します。不要な処理は動作環境に合わせて削除してください。

```
*************************************************************************/
#pragma SFR                             /* 特殊機能レジスタ(SFR)名を記述可能にする */
#pragma DI                              /* DI命令を記述可能にする */
#pragma EI                              /* EI命令を記述可能にする */

/*=========================================================================

        ポート定義

=========================================================================*/
/* キー入力ポート */
#define P_KEY0  P15     /* キー(変換A1、変換A2、変換B、補正A)の入力ポート */
#define P_KEY1  P14     /* キー(補正B、進行、停止)の入力ポート */

/* ソフトウェアシリアルインタフェース */
#define P_CS0   P4.4    /* チップセレクト0：DAC0用 */
#define P_CS1   P4.3    /* チップセレクト1：DAC1用 */
#define P_SCK   P1.1    /* クロック：DAC1,DAC0共通 */
#define P_SDA   P1.3    /* データ：DAC1,DAC0共通 */

/* HD74HC165通信用パラレル→シリアル入力 */
#define P_CLK   P11.0   /* クロック */
#define P_DAT   P10.3   /* データ */
#define P_LAT   P10.2   /* ラッチ */

/* エラー表示 */
#define P_NOERR P4.0
```

```
/*=========================================================================

        構造体定義

=========================================================================*/
struct st_Result {      /* 処理結果を格納する構造体 */
        unsigned short ushDAC;                  /* DAC設定値 */
        unsigned short ushAD;                   /* A/D変換結果 */
        unsigned short ushCorrect;              /* 5点補正結果 */
        unsigned short ushHeatCorr;             /* 温度補正結果 */
        unsigned char bDAC_ready        :1;     /* DAC設定値準備フラグ */
        unsigned char bAD_ready         :1;     /* A/D変換結果準備フラグ */
        unsigned char bCorrect_ready    :1;     /* 5点補正結果準備フラグ */
        unsigned char bHeatCorr_ready   :1;     /* 温度補正結果準備フラグ */
};

/*=========================================================================

        関数プロトタイプ宣言

=========================================================================*/
static void fn_Init(void);
static void fn_Init_st_Result(struct st_Result *result);
static void fn_KeyScan(void);
static void fn_KeyEvent(void);
static void fn_Control_ConvertA1(void);
static void fn_Control_ConvertA2(void);
static void fn_Control_ConvertB(void);
static void fn_Control_CorrectA(void);
static void fn_Control_CorrectB(void);
static void fn_ImportStartEnd(void);
static void fn_SetDAC(unsigned short DAC1_value);
static void fn_SSIO_16bit(unsigned short value);
static unsigned short fn_GetAD(void);
static unsigned short fn_CorrectADresult( unsigned short ADresult );
static unsigned short fn_HeatCorrect(unsigned short ushNowCode1, unsigned short ushCode);
static void fn_SetSendData(struct st_Result s_Result, unsigned char *p_ucTxBuffer);
static void fn_ADResultOut(unsigned char ucTxBuffer[16]);


/*=========================================================================
```

ＲＯＭの定義

```
================================================================================*/
/* ⊿Σ型A/Dコンバータの初期化処理用 */
#define ADD0TCR0                (*(volatile unsigned char*) 0x0FA26)
#define ADD0TCR1                (*(volatile unsigned char*) 0x0FA27)
#define ADD0TCR2                (*(volatile unsigned char*) 0x0FA28)
#define ADD0TCR3                (*(volatile unsigned char*) 0x0FA29)

/*================================================================================
```

ＲＡＭの定義

```
================================================================================*/
#define ANA_AVDD                5               /* AVDD = 5V */
#define ANA_REFP                ANA_AVDD        /* REFP = AVDD */
#define ANA_MIN                 0x0000          /* デジタル電圧最小値 */
#define ANA_MAX                 0xFFFF          /* デジタル電圧最大値 */
#define ANA_THRESHOLD           2.84375         /* ana2 > ana4 となってしまう場合の閾値 */

#define CONVERT_B_STEP                  1       /* 開始値から終了値まで変換する場合、1H刻みで連続変換
を行う */

#define TXDATA_SIZE                     21      /* 表示用データサイズ */
/* 表示内容：DAC設定値(4桁) SP A/D変換結果(4桁) SP 5点補正結果(4桁) SP 温度補正結果(4桁) \r\n */

#define READY_NO                        0       /* データ無し */
#define READY_OK                        1       /* データ有り */

#define CKEYCHAT                        3       /* キースキャンチャタリング除去回数 */
static unsigned char ucKeyCode;                 /* キーコード */
#define KYCODE_OFF                      0       /* 押下キー無し */
#define KYCODE_CONVERT_A1               1       /* 変換A1キー押下 */
#define KYCODE_CONVERT_A2               2       /* 変換A2キー押下 */
#define KYCODE_CONVERT_B                3       /* 変換Bキー押下 */
#define KYCODE_CORRECT_A                4       /* 補正Aキー押下 */
#define KYCODE_CORRECT_B                5       /* 補正Bキー押下 */
#define KYCODE_GO                       6       /* 進行キー押下 */
#define KYCODE_STOP                     7       /* 停止キー押下 */
#define KYCODE_INVALID                  9       /* キー多重押し状態(無効) */

static unsigned char ucSystemMode;              /* システムの状態 */
#define MODE_STANDBY                    0       /* スタンバイ状態 / キー入力待ち */
```

```
#define MODE_CONVERT_A1                 1       /* 単発変換モード(DAC以外からの電圧入力) */
#define MODE_CONVERT_A2                 2       /* 連続変換モード(DAC以外からの電圧入力) */
#define MODE_CONVERT_B                  3       /* 連続変換モード(DACからの電圧入力) */
#define MODE_CORRECT_A                  4       /* 5点補正モード(DAC以外からの電圧入力) */
#define MODE_CORRECT_B                  5       /* 5点補正モード(DACからの電圧入力) */

static unsigned char ucState;                   /* A/D変換/補正の実行状態 */
#define STATE_STANDBY                   0       /* スタンバイ中 */
#define STATE_WAIT_GO                   1       /* 進行キー待ち */
#define STATE_EXEC                      2       /* 実行中 */
#define STATE_STOP                      3       /* 停止 / 処理終了 */

#define CMP00_AFTER_DAC         (500/100)       /* 8bitTMH1のカウント：DAC設定後回路安定待 */

static unsigned char uc5CodeReady;              /* 5点補正用データ測定完了状態 */
#define CODE5_NOT_COMP                  0       /* 5点補正用データ測定未完了 */
#define CODE5_COMPLETE                  1       /* 5点補正用データ測定完了 */

static float fAna[5];                           /* 5点補正出力演算用係数 */
static unsigned short ushCode[5];               /* 5点補正出力演算用係数 */
static unsigned short ushCorrCode1;             /* 温度補正出力演算用係数 code1補正結果 */
static unsigned short ushCorrCode3;             /* 温度補正出力演算用係数 code3補正結果 */

#define JDG_HEATCORR_NO                 0       /* 温度補正不要 */
#define JDG_HEATCORR_EX                 1       /* 温度補正必要 */
#define JDG_NOT_YET                     2       /* 温度補正用/不要 未判定 */

static unsigned short ushStart;                 /* A/D変換開始値 */
static unsigned short ushEnd;                   /* A/D変換終了値 */
static unsigned char ucImportState;             /* A/D変換開始/終了値取り込み状態 */
#define IMPORT_NO                       0       /* A/D変換開始/終了値取り込み未完了 */
#define IMPORT_OK                       1       /* A/D変換開始/終了値取り込み完了 */

/************************************************************************************

        リセット解除後の初期化処理

************************************************************************************/
void hdwinit(void)
{

        unsigned short i;       /* ワーク領域 */
```

```
        DI();                           /* 割り込み禁止 */
/*--------------------------------------------------------------------------
        ROM/RAMサイズの設定
--------------------------------------------------------------------------/
        モデルにより設定値が異なるので注意してください。
        使用モデルの設定を有効にしてください。（デフォルトではuPD78F0495）
--------------------------------------------------------------------------*/
        /* uPD78F0461,uPD78F0491使用時の設定 */
        /*IMS = 0x04;                   /* ROMサイズの設定 */
        /*IXS = 0x0C;                   /* 内部拡張RAMサイズの設定 */

        /* uPD78F0462,uPD78F0492使用時の設定 */
        /*IMS = 0xC6;                   /* ROMサイズの設定 */
        /*IXS = 0x0C;                   /* 内部拡張RAMサイズの設定 */

        /* uPD78F0463,uPD78F0493使用時の設定 */
        /*IMS = 0xC8;                   /* ROMサイズの設定 */
        /*IXS = 0x0C;                   /* 内部拡張RAMサイズの設定 */

        /* uPD78F0464,uPD78F0494使用時の設定 */
        /*IMS = 0xCC;                   /* ROMサイズの設定 */
        /*IXS = 0x0A;                   /* 内部拡張RAMサイズの設定 */

        /* uPD78F0465,uPD78F0495使用時の設定 */
        IMS =   0xCF;                   /* ROMサイズの設定 */
        IXS =   0x0A;                   /* 内部拡張RAMサイズの設定 */

/*--------------------------------------------------------------------------
        クロック周波数の設定
--------------------------------------------------------------------------
        高速システム・クロックで動作が行えるように設定します。
--------------------------------------------------------------------------*/
        OSCCTL =0b01000000;             /* クロック動作モード */
                /*||||++++--------------- <0固定> */
                /*|||+------------------- OSCSELS: 入力ポート */
                /*||+-------------------- <0固定> */
                /*++--------------------- EXCLK/OSCSEL: X1発振モード */

        MOC = 0;                        /* X1発振回路動作 */

        while (OSTC.0 == 0)             /* X1発振回路 発振安定時間待ち */

        MCM =   0b00000101;             /* 供給クロック選択 */
```

```
            /*|||||+|+--------------- XSEL/MCM0: */
            /*|||||  |                 メイン・システム・クロック(fXP)=fXH */
            /*|||||  |                 周辺ハードウエア・クロック(fPRS)=fXH */
            /*|||||  +---------------- MCS: Read Only */
            /*+++++------------------- <0固定> */

        PCC =   0b0000000;              /* CPUクロック(fCPU)の選択 */
            /*|||+|+++--------------- CSS/PCC2/PCC1/PCC0: */
            /*|||  |                          CPUクロック(fCPU)=fXP */
            /*|||  +------------------ <0固定> */
            /*||+-------------------- CLS: メイン・システム・クロック */
            /*++--------------------- <0固定> */

        RCM =   0b0000011;              /* CPUクロック(fCPU)の選択 */
            /*|||||+----------------- LSRSTOP:低速内蔵発振器の停止 */
            /*|||||+----------------- RSTOP:高速内蔵発振器の停止 */
            /*|++++------------------ <0固定> */
            /*+---------------------- RSTS: Read Only */

/*-------------------------------------------------------------------------------
        UART6の設定
-------------------------------------------------------------------------------*/

        CKSR6 = 0b00000000;     /* UART6基本クロック選択 */
            /*||||++++-------- TPS63-60: 基本クロック(fXCLK6) = fPRS */
            /*++++------------ <0固定> */

        /* ボー･レート用クロックの分周値設定 */
        BRGC6 = 43;     /* ボーレート：115200bps ← 116279bps(ERR:0.94%) */

        ASIM6 = 0b01000101;             /* UART6動作モード選択 */
            /*|||||||+---------------- ISRM6: 受信エラー発生時にINTSR6を割り込み */
            /*||||||+----------------- SL6: ストップ･ビット数=1 */
            /*|||||+------------------ CL6: データ長=8 */
            /*|||++------------------- PS61-60: パリティなし */
            /*||+--------------------- RXE6: 受信動作禁止 */
            /*|+---------------------- TXE6: 送信動作許可 */
            /*+----------------------- POWER6: 内部動作クロックの動作禁止 */

        ASICL6 =0b00010110;             /* 先頭ビット,TxD6出力反転選択 */
            /*|||||||+---------------- TXDLV6: TxD6通常出力 */
            /*||||||+----------------- DIR6: 先頭ビットLSB */
            /*|||+++------------------ SBL62-60: 未使用 */
```

```
        /*||+------------------- SBTT6: 未使用 */
        /*|+-------------------- SBRT6: Read Only */
        /*+--------------------- SBRF6: 未使用 */

    PF1 =   0b00000000;             /* P16使用設定 */
        /*+|++|+++-------------- <000000固定> */
        /* |  +----------------- PF13: 未使用 */
        /* +-------------------- PF16: 未使用 (KO/LF3時) */
        /* +-------------------- <0固定> (KO/LE3時)*/

    ISC =   0b00001000;             /* 入力切り替え制御(LF3使用時) */
        /*|||||||+-------------- ISC0: 未使用 */
        /*||||||+--------------- ISC1: TI000入力をそのまま使用する(通常動作) */
        /*|||||+---------------- ISC2: 未使用 */
        /*||||+----------------- ISC3: RxD6/P113入力許可 */
        /*||++------------------ ISC5-4: TxD6=P112,RxD6=P113 */
        /*++-------------------- <0固定> */

    POWER6 =        1;              /* 内部動作クロックの動作許可 */

    /*-------------------------------------
      送信設定
    -------------------------------------*/
    PM11.2 = 0;                     /* P112 = TxD6 */
    P11.2 = 1;                      /* P112 = Hi */

    /*-------------------------------------
      受信設定
    -------------------------------------*/
    PM11.3 = 1;                     /* P113 = RxD6 */
    PU11.3 = 1;                     /* 受信動作はしないため、内蔵プルアップを接続 */

/*---------------------------------------------------------------------------
    100usインターバル･タイマの設定
---------------------------------------------------------------------------
    ウェイト時間計測用タイマの設定(100us単位でインターバル時間の設定が可能)
        ・DAC設定後の回路安定待ち時間(500us)
---------------------------------------------------------------------------*/
    TMHMD0 = 0b01000000;            /* タイマ･クロック選択レジスタ */
        /* |||||||+-------------- TOEN1: タイマ出力禁止 */
        /* ||||||+--------------- TOLEV0: タイマ出力レベル = 未使用 */
        /* ||||++---------------- TMMD01/00: タイマ動作 = インターバル */
        /* |+++----------------- CKS02/01/00: カウント･クロック fPRS/2^10 (fPRS = 10MHz より
```

```
9.77kHz) */
                /* +--------------------- TMHE0: タイマ動作禁止(タイマ使用時にCMP00を設定し許可) */

        TMIFH0 = 0;                     /* 割り込み要求クリア */
        TMMKH0 = 1;                     /* 割り込み禁止 */

/*-------------------------------------------------------------------------------
        10msインターバル設定
-------------------------------------------------------------------------------
        TMH1を使用し、10msのインターバルを作成
-------------------------------------------------------------------------------*/
        TMHMD1 = 0b01000000;            /* タイマ･クロック選択レジスタ */
                /* |||||||+-------------- TOEN1: タイマ出力禁止 */
                /* ||||||+--------------- TOLEV1: タイマ出力レベル=未使用 */
                /* ||||++---------------- TMMD11/10: タイマ動作=インターバル */
                /* |+++------------------ CKS12/11/10: カウント･クロック fPRS/2^12 (fPRS = 10MHz より
2.44kHz) */
                /* +--------------------- TMHE1: タイマ動作禁止(タイマ設定完了後許可) */
        CMP01 = 24-1;                   /* 10msインターバル：(fPRS/2^12)*0.01[sec]=24.4 */

        TMHE1  = 1;                     /* タイマ動作開始 */
        TMIFH1 = 0;                     /* 割り込み要求クリア */
        TMMKH1 = 1;                     /* 割り込み禁止 */

/*-------------------------------------------------------------------------------
        DAC設定用ソフトウェアシリアルインタフェース設定
-------------------------------------------------------------------------------*/
        PM1 =   0b00000000;             /* P1入出力設定 *//* K0/LF3で動作させる場合のみコメントを外し
てください */
                /*|||||||+-------------- PM10:未使用(0) */
                /*||||||+--------------- PM11:出力(0) SCKとして使用 */
                /*|||||+---------------- PM12:未使用(0) */
                /*||||+----------------- PM13:出力(0) SDAとして使用 */
                /*++++------------------ PM17/PM16/PM15/PM14:未使用(0) */
        P1 =    0b00000010;             /* P1初期値 *//* K0/LF3で動作させる場合のみコメントを外してく
ださい */
                /*|||||||+-------------- P10:未使用(0) */
                /*||||||+--------------- P11:Hi(1) */
                /*|||||+---------------- P12:未使用(0) */
                /*||||+----------------- P13:Lo(0) */
                /*++++------------------ P17/P16/P15/P14:未使用(0) */
/*      PM1 =   0b11100000;     */      /* P1入出力設定 *//* K0/LE3で動作させる場合のみコメントを外し
てください */
```

```
                /*|||||||+--------------- PM10:未使用(0) */
                /*||||||+---------------- PM11:出力(0) SCKとして使用 */
                /*|||||+----------------- PM12:未使用(0) */
                /*||||+------------------ PM13:出力(0) SDAとして使用 */
                /*|||+------------------- PM14:未使用(0) */
                /*+++-------------------- <111固定> */
/*      P1 =    0b00000010;     */      /* P1初期値 *//* K0/LE3で動作させる場合のみコメントを外してください */
                /*|||||||+--------------- P10:未使用(0) */
                /*||||||+---------------- P11:Hi(1) */
                /*|||||+----------------- P12:未使用(0) */
                /*||||+------------------ P13:Lo(0) */
                /*|||+------------------- P14:未使用(0) */
                /*+++-------------------- <000固定> */
        PM4 =   0b00000000;             /* P4入出力設定 *//* K0/LF3で動作させる場合のみコメントを外してください */
                /*|||||+++---------------- PM42/PM41/PM40:未使用(0) (P40は後にエラー出力用に設定) */
                /*||||+------------------ PM43:出力(0) CS1として使用 */
                /*|||+------------------- PM44:出力(0) CS0として使用 */
                /*+++-------------------- PM47/PM46/PM45:未使用(0) */
        P4 =    0b00011000;             /* P4初期値 *//* K0/LF3で動作させる場合のみコメントを外してください */
                /*|||||+++---------------- PM42/PM41/PM40:未使用(0) (P40は後にエラー出力用に設定) */
                /*||||+------------------ PM43:Hi(1) */
                /*|||+------------------- PM44:Hi(1) */
                /*+++-------------------- PM47/PM46/PM45:未使用(0) */
/*      PM4 =   0b11100000;     */      /* P4入出力設定 *//* K0/LE3で動作させる場合のみコメントを外してください */
                /*|||||+++---------------- PM42/PM41/PM40:未使用(0) (P40は後にエラー出力用に設定) */
                /*||||+------------------ PM43:出力(0) CS1として使用 */
                /*|||+------------------- PM44:出力(0) CS0として使用 */
                /*+++-------------------- <111固定> */
/*      P4 =    0b00011000;     */      /* P4初期値 *//* K0/LE3で動作させる場合のみコメントを外してください */
                /*|||||+++---------------- PM42/PM41/PM40:未使用(0) (P40は後にエラー出力用に設定) */
                /*||||+------------------ PM43:Hi(1) */
                /*|||+------------------- PM44:Hi(1) */
                /*+++-------------------- <000固定> */

/*--------------------------------------------------------------------------------
        16bitΔΣ型A/Dコンバータ
--------------------------------------------------------------------------------*/
        ADPC0 = 0b00000000;             /* A/Dポート・コンフィギュレーション・レジスタ0 */
```

```
            /*||||++++---------------- ADPC03/ADPC02/ADPC01/ADPC00:P20-P27 アナログ入力(ΔΣ型) 設
定 */
            /*++++------------------- <0000固定> */

        PM2 =   0b11111111;             /* PM2入出力設定 */
            /*|||||||+--------------- PM20:入力(1) DS0-として使用 */
            /*||||||+---------------- PM21:入力(1) DS0+として使用 */
            /*|||||+----------------- PM22:入力(1) DS1-として使用 */
            /*||||+------------------ PM23:入力(1) DS1+として使用 */
            /*|||+------------------- PM24:入力(1) DS2-として使用 */
            /*||+-------------------- PM25:入力(1) DS2+として使用 */
            /*|+--------------------- PM26:入力(1) REF-として使用 */
            /*+---------------------- PM27:入力(1) REF+として使用 */
        P2 =    0b00000000;             /* P2初期値設定 */
            /*++++++++--------------- P27/P26/P25/P24/P23/P22/P21/P20:Lo(0) */

        ADDPON = 1;                     /* A/Dコンバータ電源ON */

        ADDCTL1=0b01100111;             /* 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ1 */
            /*|||||+++--------------- ADDN2/ADDN1/ADDN0:サンプリング回数(分解能) 65536回(16bit) 指
定 */
            /*|||++----------------- <00固定> */
            /*||+------------------ ADDTS:シリアルモード設定 */
            /*++------------------- ADDFS1/ADDFS0: サンプリングクロック fPRS/8 選択*/

        ADDCTL0=0b10110000;             /* 16ビットΔΣ型A/Dコンバータ・コントロール・レジスタ0 */
            /*||||||++--------------- ADDS1/ADDS0:アナログ入力 DS0+/DS0- 指定 */
            /*||||++----------------- <00固定> */
            /*|||+------------------- AINMOD:入力モード 差動入力 指定 */
            /*||+-------------------- HAC:高精度モードON */
            /*|+--------------------- ADDCE:変換停止 */
            /*+---------------------- ADDPON:A/Dコンバータ電源ON */

        /* SFRの初期化 */
        ADD0TCR0 = 0b10000000;
        ADD0TCR0 = 0b11000000;
        ADD0TCR1 = 0b00100000;
        ADD0TCR2 = 0b01100000;
        ADD0TCR3 = 0b00010011;

        for( i = 20; i < 0; i--);       /* A/Dコンバータ電源ON→動作許可まで5μsec以上空ける */
        ADDCE = 1;                      /* 変換動作許可 */
        DSADIF = 0;                     /* A/D変換完了割り込み要求フラグクリア */
```

```
/*------------------------------------------------------------------------------
        キースキャン設定
------------------------------------------------------------------------------
        キースキャン用のポート設定を行う。
                ポート割り付け
                 P150 : 変換A1   P140 : 補正B
                 P151 : 変換A2   P141 : 進行キー
                 P152 : 変換B    P142 : 停止
                 P153 : 補正A
------------------------------------------------------------------------------*/
        PM15 =  0b11111111;             /* P15入出力設定 */
                /*||||++++--------------- PM153/PM152/PM151/PM150:入力(1) */
                /*++++------------------- <0000固定> */
        P15 =   0b00000000;             /* P15初期値 */
                /*||||++++--------------- P153/P152/P151/P150:Lo(0) */
                /*++++------------------- <0000固定> */
        PU15 =  0b00001111;             /* P15内蔵プルアップ接続設定 */
                /*||||++++--------------- PU153/PU152/PU151/PU150:内蔵プルアップ抵抗を接続(1) */
                /*++++------------------- <0000固定> */

        PM14 =  0b11111111;             /* P14入出力設定 */
                /*||||++++--------------- PM143/PM142/PM141/PM140:入力(1) */
                /*++++------------------- <1111固定> */
        P14 =   0b00001111;             /* P14初期値設定 */
                /*||||++++--------------- P143/P142/P141/P140:Hi(1) */
                /*++++------------------- <0000固定> */
        PU14 =  0b00001111;             /* P14内蔵プルアップ接続設定 */
                /*||||++++--------------- PU143/PU142/PU141/PU140:内蔵プルアップ抵抗を接続(1) */
                /*++++------------------- <0000固定> */

        /* STOPキーによる割り込み設定 */
        PM12 =  0b11111111;             /* P12入出力設定 */
                /*|||||||+--------------- P120:入力(1) INTP0 */
                /*+++++++---------------- <0000000固定> */
        P12 =   0b00000001;             /* PM12初期値設定 */
                /*|||||||+--------------- PM120:Lo(0) */
                /*+++++++---------------- <0000000固定> (K0/LF3時) */
                /*+++++++---------------- 設定不可(0) (K0/LE3時) */
        PU12 =  0b00000001;             /* P12内蔵プルアップ接続設定 */
                /*|||||||+--------------- PU120:内蔵プルアップ抵抗を接続(1) */
                /*+++++++---------------- <0000000固定> */
```

```
        EGP = 0b00000000;               /* 外部割り込み立ち上がりエッジ許可レジスタ */
                /*|||||||+--------------- EGP0: EGN0と合わせて立下りエッヂ検出有効 */
                /*||+++++---------------- EGP5/EGP4/EGP3/EGP2/EGP1:未使用 */
                /*++---------------------- <00固定> */
        EGN = 0b00000001;               /* 外部割り込み立ち下がりエッジ許可レジスタ */
                /*|||||||+--------------- EGN0: EGP0と合わせて立下りエッヂ検出有効*/
                /*||+++++---------------- EGN5/EGN4/EGN3/EGN2/EGN1:未使用 */
                /*++---------------------- <00固定> */

        PMK0 = 1;                       /* 割り込み処理禁止 */
        PIF0 = 0;                       /* 割り込み要求クリア */

/*--------------------------------------------------------------------------
        HD74HC165通信用パラレル→シリアル入力設定
--------------------------------------------------------------------------*/
        P10 =   0b00000000;             /* P10初期値設定 */
                /*||||||++--------------- P101/P100:Lo(0) */
                /*||||++----------------- P103/P102:Lo(0) */
                /*++++------------------- <0000固定> */
        PM10 =  0b11111000;             /* P10入出力設定 */
                /*||||||++--------------- PM101/PM100:未使用(0) */
                /*|||||+----------------- PM102:出力(0) = LAT */
                /*||||+------------------ PM103:入力(1) = DAT */
                /*++++------------------- <1111固定> */
        PU10 =  0b00001000;             /* P10内蔵プルアップ接続設定 */
                /*||||||++--------------- PU101/PU100:未使用(0) */
                /*|||||+----------------- PU102:未使用(0) */
                /*||||+------------------ PU103:内蔵プルアップ抵抗を接続(1) */
                /*++++------------------- <0000固定> */

        P11.0 = 0;                      /* P110:Lo(0) */
        PM11.0 = 0;                     /* PM110:出力(0) = CLK */


/*--------------------------------------------------------------------------
        エラー出力ポート設定
--------------------------------------------------------------------------*/
        PM4.0 = 0;                      /* PM40:出力(0) エラー出力に使用 */
        P4.0 = 1;                       /* P40:Hi(1) */

/*--------------------------------------------------------------------------
        未使用ポートの初期化
--------------------------------------------------------------------------*/
```

```
        /* P3 */
        PM3 =   0b11100000;             /* P3入出力設定 *//* KO/LF3で動作させる場合のみコメントを外し
てください */
                /*|||+++++--------------- PM34/PM33/PM32/PM31/PM30:未使用(0) */
                /*+++-------------------- <111固定> */
        P3 =    0b00000000;             /* P3初期値設定 *//* KO/LF3で動作させる場合のみコメントを外し
てください */
                /*|||+++++--------------- P34/P33/P32/P31/P30:Lo(0) */
                /*+++-------------------- <000固定> */
/*      PM3 =   0b11100001;     */      /* P3入出力設定 *//* KO/LE3で動作させる場合のみコメントを外し
てください */
                /*|||||||+--------------- <1固定> */
                /*|||++++---------------- PM34/PM33/PM32/PM31:未使用(0) */
                /*+++-------------------- <111固定> */
/*      P3 =    0b00000000;     */      /* P3初期値設定 *//* KO/LE3で動作させる場合のみコメントを外し
てください */
                /*|||||||+--------------- <0固定> */
                /*|||++++---------------- P34/P33/P32/P31:Lo(0) */
                /*+++-------------------- <000固定> */

        /* P8 */
        PM8 =   0b11110000;             /* P8入出力設定 */
                /*||||++++--------------- PM83/PM82/PM81/PM80:未使用(0) */
                /*++++-------------------- <1111固定> */
        P8 =    0b00000000;             /* P8初期値設定 */
                /*||||++++--------------- P83/PM2/P81/P80:Lo(0) */
                /*++++-------------------- <0000固定> */

        /* P9 */
        PM9 =   0b11110000;             /* P9入出力設定 *//* KO/LF3で動作させる場合のみコメントを外し
てください */
                /*||||++++--------------- PM93/PM92/PM91/PM90:未使用(0) */
                /*++++-------------------- <1111固定> */
        P9 =    0b00000000;             /* P9初期値設定 *//* KO/LF3で動作させる場合のみコメントを外し
てください */
                /*||||++++--------------- P93/P92/P91/P90:Lo(0) */
                /*++++-------------------- <0000固定> */

        /* P11 */
        PM11.1 = 0;                     /* PM111:未使用(0) */
        P11.1 = 0;                      /* P111:Lo(0) */

        /* P13 */
```

```
        PM13 =  0b11110000;             /* P13入出力設定 *//* K0/LF3で動作させる場合のみコメントを外
してください */
                /*||||++++--------------- PM130/PM133/PM132/PM131:未使用(0) */
                /*++++------------------- <1111固定> */
        P13 =   0b00000001;             /* P13初期値設定 *//* K0/LF3で動作させる場合のみコメントを外
してください */
                /*||||++++--------------- P133/P132/P131/P130:Lo(0) */
                /*++++------------------- <0000固定> */

        PFALL = 0b00000000;             /* セグメントと兼用のポートの出力設定 */
                /*|++|++|+--------------- PF15ALL/PF14ALL/PF11ALL/PF10ALL/PF08ALL:セグメント出力以外
として使用 */
                /*|  +--+---------------- PF13ALL/PF09ALL:セグメント出力以外として使用（K0/LF3時）*/
                /*|  +--+---------------- <00固定> (K0/LE3時) */
                /*+---------------------- <0固定> */

        EI();                           /* 割り込み許可 */
}

/************************************************************************************

        変数の初期化処理

------------------------------------------------------------------------------

************************************************************************************/
static void fn_Init(void)
{
        unsigned char i;                /* 初期化用ワーク変数 */
        float temp;

        /* キーコード */
        ucKeyCode = KYCODE_OFF;                 /* キーオフ */

        /* システムの状態 */
        ucSystemMode = MODE_STANDBY;            /* スタンバイ */

        /* A/D変換/補正の実行状態 */
        ucState = STATE_STANDBY;                /* スタンバイ中 */

        /* 5点補正用データ測定完了状態 */
        uc5CodeReady = CODE5_NOT_COMP;          /* 5点補正用データ測定未完了 */
```

```
        /* 補正出力演算用係数 ana1～ana5の電圧値をデジタル値で設定 */
        fAna[0] = ((0.1) * ANA_REFP)*(ANA_MAX/ANA_REFP);
        fAna[1] = (ANA_AVDD - (0.82*ANA_AVDD) + 0.91)*(ANA_MAX/ANA_REFP);
        fAna[2] = (0.5 * ANA_REFP)*(ANA_MAX/ANA_REFP);
        fAna[3] = (0.82 * ANA_AVDD - 0.91)*(ANA_MAX/ANA_REFP);
        fAna[4] = (0.9 * ANA_REFP)*(ANA_MAX/ANA_REFP);

        if( ANA_AVDD < ANA_THRESHOLD ){         /* AVDD = REFP < 2.84375V の場合 */
                temp = fAna[1];                 /* ana2 > ana4となるため、値を入れ替える */
                fAna[1] = fAna[3];
                fAna[3] = temp;
        }

        /* 補正出力演算用係数 code1～code5 */
        for( i = 0; i < 5; i++ ){
                ushCode[i] = ANA_MIN;
        }

        /* A/D変換開始/終了値取り込み */
        ushStart = ANA_MIN;                     /* A/D変換開始値 最小値で初期化 */
        ushEnd = ANA_MAX;                       /* A/D変換終了値クリア */
        ucImportState = IMPORT_NO;              /* A/D変換開始/終了値取り込み未完了 */
}

/*****************************************************************************

        st_Result型変数の初期化処理

*****************************************************************************/
static void fn_Init_st_Result(struct st_Result *result)
{
                (*result).ushDAC = ANA_MIN;             /* DAC設定値の初期化 */
                (*result).ushAD = ANA_MIN;              /* A/D変換結果の初期化 */
                (*result).ushCorrect = ANA_MIN;         /* 5点補正結果の初期化 */
                (*result).ushHeatCorr = ANA_MIN;        /* 温度補正結果の初期化 */
                (*result).bDAC_ready = READY_NO;        /* DAC設定値データ無し */
                (*result).bAD_ready = READY_NO;         /* A/D変換結果データ無し */
                (*result).bCorrect_ready = READY_NO;    /* 5点補正結果データ無し */
                (*result).bHeatCorr_ready = READY_NO;   /* 温度補正結果データ無し */
}

/*****************************************************************************
```

メイン・ループ

```
************************************************************************/
void main(void)
{
        hdwinit();                      /* ハードウェアの初期化 */
        fn_Init();                      /* 変数等の初期化 */

        while(1)
        {
                if( TMIFH1 )                    /* 10msec毎の処理 */
                {
                        TMIFH1 = 0;
                        fn_KeyScan();           /* Keyの取り込み */
                }
                fn_KeyEvent();                  /* キーコードによるイベント処理 */
                switch( ucSystemMode ){         /* 現在のモードによる分岐 */
                case MODE_CONVERT_A1:           /* 変換A1モード制御処理 */
                        fn_Control_ConvertA1();
                        break;
                case MODE_CONVERT_A2:           /* 変換A2モード制御処理 */
                        fn_Control_ConvertA2();
                        break;
                case MODE_CONVERT_B:            /* 変換Bモード制御処理 */
                        fn_Control_ConvertB();
                        break;
                case MODE_CORRECT_A:            /* 補正Aモード制御処理 */
                        fn_Control_CorrectA();
                        break;
                case MODE_CORRECT_B:            /* 補正Bモード制御処理 */
                        fn_Control_CorrectB();
                        break;
                default:
                        break;
                }
        }

}

/************************************************************************
```

キースキャン処理

```
--------------------------------------------------------------------------------
        キー状態を取り込み、チャタリング除去を行います。
        キースキャン周期：10msec
********************************************************************************/
static void fn_KeyScan(void)
{
        static unsigned char LastKeyState = 0;
        static unsigned char ChatCounter = 0;
        unsigned char NewKeyState;

        NewKeyState = ( P_KEY1 & 0b00001111 ) << 4 ;            /* 現在のキー状態取り込み */
        NewKeyState = NewKeyState | ( P_KEY0 & 0b00001111 );

        if( LastKeyState == NewKeyState ){                      /* キー状態の状態に変化がない場合 */
                ChatCounter--;
                if( ChatCounter == 0 ){
                        ChatCounter = CKEYCHAT;                 /* チャタリングカウンタ初期化 */
                        /* キー状態によるキーーコードの設定 */
                        if( NewKeyState == 0b11111111 ){        /* キーが全てOFF */
                                ucKeyCode = KYCODE_OFF;         /* キーコード：キーオフ */
                        } else if( NewKeyState == 0b11111110 ){ /* 変換A1キーのみON */
                                ucKeyCode = KYCODE_CONVERT_A1;  /* キーコード：変換A1 */
                        } else if( NewKeyState == 0b11111101 ){ /* 変換A2キーのみON */
                                ucKeyCode = KYCODE_CONVERT_A2;  /* キーコード：変換A2 */
                        } else if( NewKeyState == 0b11111011 ){ /* 変換Bキーのみ0N */
                                ucKeyCode = KYCODE_CONVERT_B;   /* キーコード：変換B */
                        } else if( NewKeyState == 0b11110111 ){ /* 補正Aキーのみ0N */
                                ucKeyCode = KYCODE_CORRECT_A;   /* キーコード：補正A */
                        } else if( NewKeyState == 0b11101111 ){ /* 補正Bキーのみ0N */
                                ucKeyCode = KYCODE_CORRECT_B;   /* キーコード：補正B */
                        } else if( NewKeyState == 0b11011111 ){ /* 進行キーのみON */
                                ucKeyCode = KYCODE_GO;          /* キーコード：進行 */
                        } else if( NewKeyState == 0b10111111 ){ /* 停止キーのみON */
                                ucKeyCode = KYCODE_STOP;        /* キーコード：停止 */
                        } else {                                /* 複数のキーがON */
                                ucKeyCode = KYCODE_INVALID;     /* キーコード：キー無効 */
                        }
                }
        }
        else{                                   /* キー状態に変化があった場合 */
                ChatCounter = CKEYCHAT;         /* チャタリングカウンタ初期化 */
        }
        LastKeyState = NewKeyState;             /* 現在のキー状態を保存 */
```

```
}

/*****************************************************************************

        キーイベント処理

------------------------------------------------------------------------------
        キーにより、モードまたは変換実行状態を遷移させます。
*****************************************************************************/
static void fn_KeyEvent(void)
{
        static unsigned char ucLastKeyCode;

        if( ucKeyCode != ucLastKeyCode ){
                switch ( ucKeyCode ){
                case KYCODE_CONVERT_A1:         /* 変換A1キーが押された場合 */
                        if( ucState == STATE_STANDBY ){         /* 変換/補正等が行われていなければ */
                                ucSystemMode = MODE_CONVERT_A1; /* 単発変換モード(DAC以外からの電圧入
力)へ */
                                ucState = STATE_WAIT_GO;        /* 進行キー待ち状態とする */
                        }
                        break;

                case KYCODE_CONVERT_A2:         /* 変換A2キーが押された場合 */
                        if( ucState == STATE_STANDBY ){         /* 変換/補正等が行われていなければ */
                                ucSystemMode = MODE_CONVERT_A2; /* 連続変換モード(DAC以外からの電圧入
力)へ */
                                ucState = STATE_WAIT_GO;        /* 進行キー待ち状態とする */
                        }
                        break;

                case KYCODE_CONVERT_B:          /* 変換Bキーが押された場合 */
                        if( ucState == STATE_STANDBY ){         /* 変換/補正等が行われていなければ */
                                ucSystemMode = MODE_CONVERT_B;  /* 連続変換モード(DACからの電圧入力)
へ */
                                ucState = STATE_WAIT_GO;        /* 進行キー待ち状態とする */
                        }
                        break;

                case KYCODE_CORRECT_A:          /* 補正Aキーが押された場合 */
                        if( ucState == STATE_STANDBY ){         /* 変換/補正等が行われていなければ */
                                ucSystemMode = MODE_CORRECT_A;  /* 5点補正モード(DAC以外からの電圧入
```

```
力)へ */
                        ucState = STATE_WAIT_GO;        /* 進行キー待ち状態とする */
                }
                break;

        case KYCODE_CORRECT_B:          /* 補正Bキーが押された場合 */
                if( ucState == STATE_STANDBY ){         /* 変換/補正等が行われていなければ */
                        ucSystemMode = MODE_CORRECT_B;  /* 5点補正モード(DACからの電圧入力)へ
*/
                        ucState = STATE_WAIT_GO;        /* 進行キー待ち状態とする */
                }
                break;

        case KYCODE_GO:                 /* 進行キーが押された場合 */
                if( ucState == STATE_WAIT_GO ){         /* 進行キー待ち状態であれば */
                        ucState = STATE_EXEC;           /* 変換実行中待ち状態とする */
                }
                break;

        case KYCODE_STOP:               /* 停止キーが押された場合 */
                ucState = STATE_STOP;                   /* 停止処理へ */
                break;

/*      case KYCODE_INVALID: */         /* キーが無効の場合 */
        default:
                ;                                       /* 何もしない */
        }
}
if(PIF0){                                               /* 停止キーが押された場合 */
        ucState = STATE_STOP;                           /* 停止処理へ */
        PIF0 = 0;                                       /* 割り込み要求クリア */
}
ucLastKeyCode = ucKeyCode;                              /* 前回のキーコード更新 */
}

/************************************************************************

        A/D変換：変換A1モード制御処理

------------------------------------------------------------------------
        単発変換モード(DAC以外からの電圧入力)の処理を制御します。
        ※A/D変換のアナログ入力にDACを使用しない場合のみ使用。
************************************************************************/
```

```
static void fn_Control_ConvertA1(void)
{
        struct st_Result sResult;                       /* 変換結果格納 */
        static unsigned char ucTxBuffer[TXDATA_SIZE];   /* 表示用データバッファ */
        static unsigned short ushNowCode1;              /* 現在の温度で再取得したcode1 */
        static unsigned char HeatCorrJdg;               /* 温度補正が必要かどうかの情報 */

        switch( ucState ){      /* 実行状態による分岐 */
        case STATE_STOP:        /* 停止処理 */
                fn_Init_st_Result( &sResult );                  /* 変換結果の初期化 */
                ucState = STATE_STANDBY;                        /* 動作状態をスタンバイに */
                ushNowCode1 = ANA_MIN;                          /* 現在温度のcode1を初期化 */
                HeatCorrJdg = JDG_NOT_YET;                      /* 温度補正要/不要未判定 */
                break;

        case STATE_WAIT_GO:     /* 進行キー待ち */
                if(( uc5CodeReady == CODE5_COMPLETE )&&( HeatCorrJdg == JDG_NOT_YET ))
                {       /* 5点補正用データ測定が完了していて、温度補正の要/不要が未判定の場合、code1
を取り直して判定 */
                        ushNowCode1 = fn_GetAD();                       /* ana1 A/D変換結果再取得 */
                        ushNowCode1 = fn_CorrectADresult( ushNowCode1 );/* 現在のcode1を補正 */
                        if( ushNowCode1 != ushCorrCode1 )               /* 温度変化有りなら */
                                HeatCorrJdg = JDG_HEATCORR_EX;          /* 温度補正を実行するよう設定
*/
                        else
                                HeatCorrJdg = JDG_HEATCORR_NO;          /* 温度補正をしないよう設定 */
                }
                break;

        case STATE_EXEC:        /* 変換処理実行 */
                sResult.bDAC_ready = READY_NO;                  /* DAC設定値データ無し */
                sResult.ushAD = fn_GetAD();                     /* A/D変換結果取得 */
                sResult.bAD_ready = READY_OK;                   /* A/D変換結果データ準備完了 */

                if( uc5CodeReady == CODE5_COMPLETE )            /* 5点補正用データ測定が完了していた
場合 */
                {
                        sResult.ushCorrect = fn_CorrectADresult( sResult.ushAD );       /* A/D変換結
果5点補正 */
                        sResult.bCorrect_ready = READY_OK;      /* 5点補正結果データ準備完了 */
                        if( HeatCorrJdg == JDG_HEATCORR_EX )    /* 温度補正が必要な場合 */
                        {
                                sResult.ushHeatCorr = fn_HeatCorrect( ushNowCode1,
```

```
sResult.ushCorrect);/* 温度補正 */
                                sResult.bHeatCorr_ready = READY_OK;/* 温度補正結果データ準備完了 */
                        }
                        else
                                sResult.bHeatCorr_ready = READY_NO;/* 5点補正結果データ無し */
                }
                else                                            /* 5点補正用データ測定が未完了の場合
*/
                {
                        sResult.bCorrect_ready = READY_NO;      /* 5点補正結果データ無し */
                        sResult.bHeatCorr_ready = READY_NO;     /* 5点補正結果データ無し */
                }

                fn_SetSendData( sResult, ucTxBuffer );          /* 変換結果出力設定 */
                fn_ADResultOut( ucTxBuffer );                   /* データ出力 */
                ucState = STATE_WAIT_GO;                        /* 進行キーを待って次の変換を行う */
                break;

        default:
                break;
        }
}

/*****************************************************************************

        A/D変換：変換A2モード制御処理

-----------------------------------------------------------------------------
        連続変換モード(DAC以外からの電圧入力)の処理を制御します。
        ※A/D変換のアナログ入力にDACを使用しない場合のみ使用。
*****************************************************************************/
static void fn_Control_ConvertA2(void)
{
        struct st_Result sResult;                       /* 変換結果格納 */
        static unsigned char ucTxBuffer[TXDATA_SIZE];   /* 表示用データバッファ */
        static unsigned short ushNowCode1;              /* 現在の温度で再取得したcode1 */
        static unsigned char HeatCorrJdg;               /* 温度補正が必要かどうかの情報 */

        switch( ucState ){      /* 実行状態による分岐 */
        case STATE_STOP:        /* 停止処理 */
                fn_Init_st_Result( &sResult );                  /* 変換結果の初期化 */
                ucState = STATE_STANDBY;                        /* 動作状態をスタンバイに */
                ushNowCode1 = ANA_MIN;                          /* 現在温度のcode1を初期化 */
```

```
                HeatCorrJdg = JDG_NOT_YET;                      /* 温度補正要/不要未判定 */
                break;

        case STATE_WAIT_GO:     /* 進行キー待ち */
                if(( uc5CodeReady == CODE5_COMPLETE )&&( HeatCorrJdg == JDG_NOT_YET ))
                {       /* 5点補正用データ測定が完了していて、温度補正の要/不要が未判定の場合、code1
を取り直して判定 */
                        ushNowCode1 = fn_GetAD();                       /* ana1 A/D変換結果再取得 */
                        ushNowCode1 = fn_CorrectADresult( ushNowCode1 );/* 現在のcode1を補正 */
                        if( ushNowCode1 != ushCorrCode1 )               /* 温度変化有りなら */
                                HeatCorrJdg = JDG_HEATCORR_EX;          /* 温度補正を実行するよう設定
*/
                        else
                                HeatCorrJdg = JDG_HEATCORR_NO;          /* 温度補正をしないよう設定 */
                }
                break;

        case STATE_EXEC:        /* 変換処理実行 */
                sResult.bDAC_ready = READY_NO;                  /* DAC設定値データ無し */
                sResult.ushAD = fn_GetAD();                     /* A/D変換結果取得 */
                sResult.bAD_ready = READY_OK;                   /* A/D変換結果データ準備完了 */

                if( uc5CodeReady == CODE5_COMPLETE )            /* 5点補正用データ測定が完了していた
場合 */
                {
                        sResult.ushCorrect = fn_CorrectADresult( sResult.ushAD );       /* A/D変換結
果5点補正 */
                        sResult.bCorrect_ready = READY_OK;      /* 5点補正結果データ準備完了 */
                        if( HeatCorrJdg == JDG_HEATCORR_EX )    /* 温度補正が必要な場合 */
                        {
                                sResult.ushHeatCorr = fn_HeatCorrect( ushNowCode1,
sResult.ushCorrect);/* 温度補正 */
                                sResult.bHeatCorr_ready = READY_OK;/* 温度補正結果データ準備完了 */
                        }
                        else
                                sResult.bHeatCorr_ready = READY_NO;/* 5点補正結果データ無し */
                }
                else                                            /* 5点補正用データ測定が未完了の場合
*/
                {
                        sResult.bCorrect_ready = READY_NO;      /* 5点補正結果データなし */
                        sResult.bHeatCorr_ready = READY_NO;     /* 5点補正結果データ無し */
                }
```

```
                fn_SetSendData( sResult, ucTxBuffer );          /* 変換結果出力設定 */
                fn_ADResultOut( ucTxBuffer );                   /* データ出力 */
                /* 次回も変換を行うため、A/D実行状態は変えない */
                break;

        default:
                break;
        }
}

/*****************************************************************************

        A/D変換：変換Bモード制御処理

-----------------------------------------------------------------------------
        連続変換モード(DACからの電圧入力)の処理を制御します。
        ※A/D変換のアナログ入力にDACを使用する場合のみ使用。
*****************************************************************************/
static void fn_Control_ConvertB(void)
{
        static struct st_Result sResult;                /* 変換結果格納 */
        static unsigned char ucTxBuffer[TXDATA_SIZE];   /* 表示用データバッファ */
        static unsigned short ushNowCode1;              /* 現在の温度で再取得したcode1 */
        static unsigned char HeatCorrJdg;               /* 温度補正が必要かどうかの情報 */
        unsigned char temp;

        switch( ucState ){      /* 実行状態による分岐 */
        case STATE_STOP:        /* 停止処理 */
                fn_Init_st_Result( &sResult );                  /* 変換結果の初期化 */
                ushStart = ANA_MIN;                             /* A/D変換開始値クリア */
                ushEnd = ANA_MAX;                               /* A/D変換終了値クリア */
                ucImportState = IMPORT_NO;                      /* A/D変換開始/終了値取り込み状態クリ
ア */
                ucState = STATE_STANDBY;                        /* 動作状態をスタンバイに */
                ushNowCode1 = ANA_MIN;                          /* 現在温度のcode1を初期化 */
                HeatCorrJdg = JDG_NOT_YET;                      /* 温度補正要/不要未判定 */
                break;

        case STATE_WAIT_GO:     /* 進行キー待ち */
                if( ucImportState == IMPORT_NO )                /* A/D変換開始/終了値取り込んでいない
場合 */
                {
                        fn_ImportStartEnd();                    /* 変換開始/終了値を取り込み */
```

```
                        sResult.ushDAC = ushStart;              /* DAC設定値初期化 */
                }
                if(( uc5CodeReady == CODE5_COMPLETE )&&( HeatCorrJdg == JDG_NOT_YET ))
                {       /* 5点補正用データ測定が完了していて、温度補正の要/不要が未判定の場合、code1
を取り直して判定 */
                        fn_SetDAC((unsigned short)fAna[1-1] );          /* DAC出力にana1を設定 */
                        ushNowCode1 = fn_GetAD();                       /* ana1 A/D変換結果再取得 */
                        ushNowCode1 = fn_CorrectADresult( ushNowCode1 );/* 現在のcode1を補正 */
                        if( ushNowCode1 != ushCorrCode1 )               /* 温度変化有りなら */
                                HeatCorrJdg = JDG_HEATCORR_EX;          /* 温度補正を実行するよう設定
*/
                        else
                                HeatCorrJdg = JDG_HEATCORR_NO;          /* 温度補正をしないよう設定 */
                }
                break;

        case STATE_EXEC:        /* 変換処理実行 */
                if( ucImportState == IMPORT_NO )                /* 開始/終了値の取り込みがまだの場合 */
                {
                        ucState = STATE_WAIT_GO;                /* 再度開始/終了値の読み込みを行う */
                        return;
                }

                fn_SetDAC( sResult.ushDAC );                    /* DAC出力を設定 */
                sResult.bDAC_ready = READY_OK;                  /* DAC設定値データ準備完了 */
                sResult.ushAD = fn_GetAD();                     /* A/D変換結果取得 */
                sResult.bAD_ready = READY_OK;                   /* A/D変換結果データ準備完了 */

                if( uc5CodeReady == CODE5_COMPLETE )            /* 5点補正用データ測定が完了していた
場合 */
                {
                        sResult.ushCorrect = fn_CorrectADresult( sResult.ushAD );       /* A/D変換結
果補正 */
                        sResult.bCorrect_ready = READY_OK;      /* 5点補正結果データ準備完了 */
                        if( HeatCorrJdg == JDG_HEATCORR_EX )    /* 温度補正が必要な場合 */
                        {
                                sResult.ushHeatCorr = fn_HeatCorrect( ushNowCode1,
sResult.ushCorrect);/* 温度補正 */
                                sResult.bHeatCorr_ready = READY_OK;/* 温度補正結果データ準備完了 */
                        }
                        else
                                sResult.bHeatCorr_ready = READY_NO;/* 5点補正用データ測定結果データ無
し */
```

```
                }
                else                                            /* 5点補正が未完了の場合 */
                {
                        sResult.bCorrect_ready = READY_NO;      /* 5点補正結果データ無し */
                        sResult.bHeatCorr_ready = READY_NO;/* 5点補正結果データ無し */
                }
                fn_SetSendData( sResult, ucTxBuffer );          /* 変換結果出力設定 */
                fn_ADResultOut( ucTxBuffer );                   /* データ出力 */

                /* DAC設定値を更新 */
                /*if( sResult.ushDAC == ANA_MAX )       */      /* 最大値まで変換終了した場合は */
                /*      sResult.ushDAC = 0;     */              /* 最小値から変換を続ける */
                /*else*/
                /*      sResult.ushDAC++;       */
                /*if(( ushEnd == ANA_MAX )&&( sResult.ushDAC == 0 ))*//* 変換終了値まで変換が終わって
いた場合 */
                /*      ucState = STATE_STOP;   */      /* 変換を終了 */
                /*else if( sResult.ushDAC == ( ushEnd + 1 ))*/
                /*      ucState = STATE_STOP;           *//* 連続変換を終了 */

                /* DAC設定値を更新 */
                for( temp = 0; temp < CONVERT_B_STEP; temp++ )
                {
                        sResult.ushDAC ++;
                        if( sResult.ushDAC == ( ushEnd + 1 ))
                        {
                                ucState = STATE_STOP;           /* 連続変換を終了 */
                                break;
                        }
                }
                break;

        default:
                break;
        }
}

/*****************************************************************************

        A/D変換：補正Aモード制御処理

-----------------------------------------------------------------------------
        5点補正モード(DAC以外からの電圧入力)の処理を制御します。
```

```
        ※A/D変換のアナログ入力にDACを使用しない場合のみ使用。
*************************************************************************/
static void fn_Control_CorrectA(void)
{
        static struct st_Result sResult;                /* 変換結果格納 */
        static unsigned char ucTxBuffer[TXDATA_SIZE];   /* 表示用データバッファ */
        static unsigned char ADCount = 0;               /* 変換回数 */

        switch( ucState ){      /* 実行状態による分岐 */
        case STATE_STOP:        /* 停止処理 */
                fn_Init_st_Result( &sResult );          /* 変換結果の初期化 */
                ADCount = 0;                            /* 変換回数クリア */
                ucState = STATE_STANDBY;                /* 動作状態をスタンバイに */
                break;

        case STATE_WAIT_GO:     /* 進行キー待ち */
                break;

        case STATE_EXEC:        /* 変換処理実行 */
                if( ADCount == 0 )
                        uc5CodeReady = CODE5_NOT_COMP;  /* 5点補正用データ測定未完了 */

                sResult.bDAC_ready = READY_NO;          /* DAC設定値データ無し */

                sResult.ushAD = fn_GetAD();             /* A/D変換結果取得 */
                ushCode[ADCount] = sResult.ushAD;       /* A/D変換結果保存 */
                sResult.bAD_ready = READY_OK;           /* A/D変換結果データ準備完了 */

                sResult.bCorrect_ready = READY_NO;      /* 5点補正結果データ無し */
                sResult.bHeatCorr_ready = READY_NO;     /* 5点補正結果データ無し */

                fn_SetSendData( sResult, ucTxBuffer );  /* 変換結果出力設定 */
                fn_ADResultOut( ucTxBuffer );           /* データ出力 */

                ADCount++;                              /* DAC設定値を更新 */

                if( ADCount >= 5 ){                     /* 5点全ての変換が終わっていた場合 */
                        ucState = STATE_STOP;           /* 変換を終了 */
                        if(( ushCode[0] < ushCode[2] )  /* code1,code3,code5 の大小関係が正しい場合 */
                        && ( ushCode[2] < ushCode[4] )){
                                uc5CodeReady = CODE5_COMPLETE;  /* 5点補正用データ測定完了を通知 */
                                P_NOERR = 1;            /* エラー表示無し */
                                ushCorrCode1 = fn_CorrectADresult( ushCode[1-1] );      /* 温度補正用
```

```
にcode1を補正して保存 */
                        ushCorrCode3 = fn_CorrectADresult( ushCode[3-1] );      /* 温度補正用
にcode3を補正して保存 */
                }
                else{                           /* code1 ～ code5 の大小関係が正しくない場合
*/
                        uc5CodeReady = CODE5_NOT_COMP;  /* 5点補正用データ測定未完了 */
                        P_NOERR = 0;            /* エラー表示設定 */
                }
        }
        else{                                   /* 5点の補正が未完了の場合 */
                ucState = STATE_WAIT_GO;        /* 進行キーを待って次の変換を行う */
        }
        break;

    default:
        break;
    }
}

/*****************************************************************************

    A/D変換：補正Bモード制御処理

-----------------------------------------------------------------------------
    5点補正モード(DACからの電圧入力)の処理を制御します。
    ※A/D変換のアナログ入力にDACを使用する場合のみ使用。
*****************************************************************************/
static void fn_Control_CorrectB(void)
{
    static struct st_Result sResult;            /* 変換結果格納 */
    static unsigned char ucTxBuffer[TXDATA_SIZE];   /* 表示用データバッファ */
    static unsigned char ADCount = 0;           /* 変換回数 */


    switch( ucState ){      /* 実行状態による分岐 */
    case STATE_STOP:        /* 停止処理 */
        fn_Init_st_Result( &sResult );          /* 変換結果の初期化 */
        ADCount = 0;                            /* 変換回数クリア */
        ucState = STATE_STANDBY;                /* 動作状態をスタンバイに */
        break;

    case STATE_WAIT_GO:     /* 進行キー待ち */
```

```
            break;

        case STATE_EXEC:        /* 変換処理実行 */
            if( ADCount == 0 )
                uc5CodeReady = CODE5_NOT_COMP;  /* 5点補正用データ測定未完了 */

            sResult.ushDAC = (unsigned short)fAna[ADCount];
            fn_SetDAC( sResult.ushDAC );            /* DAC出力を設定 */
            sResult.bDAC_ready = READY_OK;          /* DAC設定値データ準備完了 */

            sResult.ushAD = fn_GetAD();             /* A/D変換結果取得 */
            ushCode[ADCount] = sResult.ushAD;       /* A/D変換結果保存 */
            sResult.bAD_ready = READY_OK;           /* A/D変換結果データ準備完了 */

            sResult.bCorrect_ready = READY_NO;      /* 5点補正結果データ準備完了 */
            sResult.bHeatCorr_ready = READY_NO;     /* 5点補正結果データ無し */

            fn_SetSendData( sResult, ucTxBuffer );  /* 変換結果出力設定 */
            fn_ADResultOut( ucTxBuffer );           /* データ出力 */

            ADCount++;                              /* DAC設定値を更新 */
            if( ADCount >= 5 ){                     /* 5点全ての変換が終わっていた場合 */
                ucState = STATE_STOP;               /* 変換を終了 */
                if(( ushCode[0] < ushCode[2] )  /* code1,code3,code5 の大小関係が正しい場合 */
                && ( ushCode[2] < ushCode[4] )){
                    uc5CodeReady = CODE5_COMPLETE;  /* 5点補正用データ測定完了を通知 */
                    P_NOERR = 1;                /* エラー表示無し */
                    ushCorrCode1 = fn_CorrectADresult( ushCode[1-1] );      /* 温度補正用
にcode1を補正して保存 */
                    ushCorrCode3 = fn_CorrectADresult( ushCode[3-1] );      /* 温度補正用
にcode3を補正して保存 */
                }
                else{                               /* code1 ～ code5 の大小関係が正しくない場合
*/
                    uc5CodeReady = CODE5_NOT_COMP;  /* 5点補正用データ測定未完了 */
                    P_NOERR = 0;                /* エラー表示設定 */
                }
            }
            break;

        default:
            break;
        }
```

```
}

/*****************************************************************************

        HD74HC165からA/D変換開始値＆終了値の取り込み

-----------------------------------------------------------------------------
        スイッチで設定されたA/D変換開始値(16bit)と終了値(16bit)を読み込みます。
        ※A/D変換のアナログ入力にDACを使用する場合のみ使用。
*****************************************************************************/
static void fn_ImportStartEnd(void)
{
        unsigned short temp;            /* データ取り込み用一時変数 */
        unsigned char count;            /* データ取り込み回数カウンタ */

        /*--------------------------------------
                A/D変換開始値取り込み
        --------------------------------------*/
        temp = 0;
        P_LAT = 1;                              /* データ取り込み開始 */
        for( count = 16; count != 0; count-- ){ /* A/D変換開始値16bitをMSBから読み込み */
                temp = ( temp << 1 );
                if( P_DAT )                     /* 取り込んだデータが'1'であれば、該当bitに1を設定 */
                        temp++;
                P_CLK = 0;
                P_CLK = 1;                      /* 次のデータのシフトを要求 */
        }
        ushStart = temp;                        /* A/D変換開始値設定 */

        /*--------------------------------------
                A/D変換終了値取り込み
        --------------------------------------*/
        temp = 0;
        for( count = 16; count != 0; count-- ){ /* A/D変換開始値16bitをMSBから読み込み */
                temp = ( temp << 1 );
                if( P_DAT )                     /* 取り込んだデータが'1'であれば、該当bitに1を設定 */
                        temp++;
                P_CLK = 0;
                P_CLK = 1;                      /* 次のデータのシフトを要求 */
        }
        ushEnd = temp;                          /* A/D変換終了値設定 */

        /* 全てのデータ取り込み終了 */
```

```
        P_LAT = 0;
        P_CLK = 0;

        ucImportState = IMPORT_OK;              /* 取り込み完了 */
}

/*****************************************************************************

        DAC設定

-----------------------------------------------------------------------------
        DAC1,DAC2へ出力電圧の設定を行います。
        ※A/D変換のアナログ入力にDACを使用する場合のみ使用。
*****************************************************************************/
static void fn_SetDAC(unsigned short DAC1_value)
{
        unsigned short DAC0_value;              /* DAC0用設定データ */

        /* DAC0へ設定する値を算出 */
        DAC0_value = ~DAC1_value;               /* DAC1の設定値を反転 */
        if(( DAC0_value == ANA_MAX )||( DAC0_value == ANA_MIN ))
                ;                               /* 最小/最大設定の場合はDAC1を反転した値を設定 */
        else
                DAC0_value += 1;                /* 最大/最小以外の設定はDAC1の反転 +1を設定 */

        /*--------------------------------------
                        DAC1への設定
        --------------------------------------*/
        /* DAC1へ設定送信 */
        P_CS1 = 0;                              /* 送信開始 */
        fn_SSIO_16bit( DAC1_value );            /* データ出力 */
        P_CS1 = 1;                              /* 送信終了 */

        /*--------------------------------------
                        DAC0への設定
        --------------------------------------*/
        /* DAC0へ設定送信 */
        P_CS0 = 0;                              /* 送信開始 */
        fn_SSIO_16bit( DAC0_value );            /* データ出力 */
        P_CS0 = 1;                              /* 送信終了 */

        /* シリアル送信後DACが安定するまでウェイト */
        CMP00 = CMP00_AFTER_DAC;                /* 500us設定 */
```

```
        TMIFH0 = 0;                             /* 割り込み要求クリア */
        TMHE0  = 1;                             /* タイマ動作開始 */
        while( !TMIFH0 )        ;               /* 500us経過待ち */
        TMHE0  = 0;                             /* タイマ動作停止 */
}

/*****************************************************************************

        ソフトウェアシリアル送信

-----------------------------------------------------------------------------
        3線式シリアル通信で、DACへデータ(16bit)を送信する。
        ※A/D変換のアナログ入力にDACを使用する場合のみ使用。
*****************************************************************************/
static void fn_SSIO_16bit(unsigned short value)
{
        unsigned short mask = 0b1000000000000000;       /* マスク用データ */

        for( ; mask != 0; mask >>= 1 )          /* MSBから1bitずつ送信 */
        {
                P_SCK = 0;                      /* SCK : LOW */
                if(( value & mask ) == 0 )      /* 送信データはLow？ */
                        P_SDA = 0;              /* Yes:送信データ(Low)出力 */
                else
                        P_SDA = 1;              /* No:送信データ(High)出力 */
                P_SCK = 1;                      /* SCK : HIGH (DACが送信データを読む) */
        }
}

/*****************************************************************************

        16bitΔΣ型A/D変換

-----------------------------------------------------------------------------
        ΔΣ型A/D変換の開始し、A/D変換の結果を返します。
*****************************************************************************/
static unsigned short fn_GetAD(void)
{
        unsigned short ret = 0;         /* A/D変換結果用戻り値 */

        /* A/D変換開始 */
        ADDCE = 1;                                      /* 変換動作許可 */
        DSADIF = 0;                                     /* A/D変換完了割り込み要求クリア */
```

```
        /* A/D変換結果取得 */
        while(1){       /* A/D変換が完了するor停止キーが押されるまで待つ */
                if(( DSADIF )&&( ADDSTR == 0 )){        /* DS0+のA/D変換が完了した場合 */
                        ret = ADDCR;                    /* A/D変換結果読み出し */
                        break;
                }else if(( DSADIF )&&( ADDSTR != 0 )){  /* 目的のチャンネルでない値を変換した場合 */
                        ADDCE = 1;                      /*  A/D変換やり直し  */
                        DSADIF = 0;
                }else if(PIF0){                         /* 停止キーが押された場合 */
                        ADDCE = 0;                      /* A/D変換動作停止 */
                        break;                          /* A/D変換中止 */
                }else
                        ;
        }

        return  ( ret );                                /* A/D変換結果を返す */
}

/*****************************************************************************

        5点補正

-----------------------------------------------------------------------------
        A/D変換結果を、補正出力用の演算式(下記)により補正します。
        演算の分母が0になる場合は、5点補正演算をせずにA/D結果をそのまま返します。

                 AD変換結果 - bn
                ―――――――――――
                       an

            ※ 補正係数説明
                      codeA - codeB
                an = ―――――――――――
                       anaA - anaB

                      anaB × codeA - anaA × codeB
                bn = ―――――――――――――――――――――
                                anaB - anaA
                n:1～4
                n = 1 : anaA = ana2, anaB = ana1, codeA = code2, codeB = code1
                n = 2 : anaA = ana3, anaB = ana2, codeA = code3, codeB = code2
                n = 3 : anaA = ana4, anaB = ana3, codeA = code4, codeB = code3
```

```
                n = 4 : anaA = ana5, anaB = ana4, codeA = code5, codeB = code4
*************************************************************************/
static unsigned short fn_CorrectADresult( unsigned short ADresult )
{
        float   ret = ADresult;         /* 演算結果 初期値は補正前の値 */
        unsigned short  temp;
        unsigned char   n;

        /* 使用する係数を判定 */
        if( ADresult >= ushCode[4-1] )          /* A/D変換結果が code4 以上の場合 */
                n = 4;
        else if(( ADresult >= ushCode[3-1] ))   /* A/D変換結果が code3～code4(code3を含む)の場合 */
                n = 3;
        else if(( ADresult > ushCode[2-1] ))    /* A/D変換結果が code2～code3の場合 */
                n = 2;
        else                                    /* A/D変換結果が code2 以下の場合 */
                n = 1;

        /* A/D変換結果を補正 */
        temp = ushCode[n] - ushCode[n-1];
        if(temp != 0){          /* 演算の分母が0になってしまう場合は、補正演算をせずA/D値をそのまま返
す */
                ret = (float)ADresult*(fAna[n] - fAna[n-1]) + fAna[n-1]*(float)ushCode[n] -
fAna[n]*(float)ushCode[n-1];
                ret = ret / temp;
        }

        return  ( unsigned short )( ret );      /* 補正結果の整数部分(16bit)を返す */
}

/*************************************************************************

        温度補正

-------------------------------------------------------------------------
        温度補正を行う関数です。
        温度誤差のある5点補正の傾きを、理想値の傾きに変換することにより，
        5点補正結果を補正します。
        下記の2つの場合に場合分けし、演算を行います。
                ①5点補正結果の傾きが理想値より小さい場合。(高温時)
                ②5点補正結果の傾きが理想値より大きい場合。(低温時)
        演算の分母が0になる場合は、温度補正演算をせずにA/D結果をそのまま返します。
```

```
*****************************************************************************/
static unsigned short fn_HeatCorrect(unsigned short ushNowCode1, unsigned short ushCode){

        unsigned long ret = ushCode;            /* 温度補正結果 */
        unsigned short temp;                    /* 演算用ワーク領域 */

        temp = ushCorrCode3 - ushNowCode1;
        if( temp != 0 ){                /* 分母が0であれば、補正演算をせず補正対象をそのまま変えす */
                if( ushNowCode1 > ushCorrCode1 )        /* ① */
                {
                        ret = (unsigned long)ushCode*(unsigned long)(ushCorrCode3 - ushCorrCode1);
                        ret = ret - (unsigned long)ushCorrCode3*(unsigned long)(ushNowCode1 -
ushCorrCode1);
                }
                else                                    /* ② */
                {
                        ret = (unsigned long)ushCode*(unsigned long)(ushCorrCode3 - ushCorrCode1);
                        ret = ret + (unsigned long)ushCorrCode3*(unsigned long)(ushCorrCode1 -
ushNowCode1);
                }
                ret = ret / temp;
        }

        return  ( unsigned short )( ret );      /* 補正結果を返す */

}

/*****************************************************************************

        UARTで出力するデータを作成

------------------------------------------------------------------------------
        引数の数値をUARTで出力するデータに変換し、送信バッファに設定します。
*****************************************************************************/
static void fn_SetSendData(struct st_Result s_Result, unsigned char *p_ucTxBuffer)
{
        unsigned char temp;     /* データ作成用ワーク領域 */
        unsigned char n;

        /* DAC設定値 送信データ設定 */
        if( s_Result.bDAC_ready )
        {       /* DAC設定値の送信データがある場合 */
                for( n = 0; n < 4; n++ ){
```

```
                /* 送信データに変換する桁を取り出し */
                temp = (unsigned char)((s_Result.ushDAC & ( 0xf000 >> 4*n )) >> (4*(3 - n)));
                if( temp >= 0xa )       /* 送信データが0xa～0xfの場合 */
                        p_ucTxBuffer[n] = temp + ( 'A' - 0xa );         /* 'A'～'F'に変換 */
                else                    /* 送信データが0x0～0x9の場合 */
                        p_ucTxBuffer[n] = temp + '0';                   /* '0'～'9'に変換 */
        }
}else{  /* DAC設定値の送信データがない場合 */
        for( n = 0; n < 4; n++ )
                p_ucTxBuffer[n] = '*';          /* DAC設定値の変わりに'*'を出力 */
}

p_ucTxBuffer[4] = ' ';          /* データ間SP */

/* A/D変換値 送信データ設定 */
if( s_Result.bAD_ready )
{       /* AD設定値の送信データがある場合 */
        for( n = 0; n < 4; n++ ){
                /* 送信データに変換する桁を取り出し */
                temp = (unsigned char)((s_Result.ushAD & ( 0xf000 >> 4*n )) >> (4*(3 - n)));
                if( temp >= 0xa )       /* 送信データが0xa～0xfの場合 */
                        p_ucTxBuffer[n + 5] = temp + ( 'A' - 0xa );     /* 'A'～'F'に変換 */
                else                    /* 送信データが0x0～0x9の場合 */
                        p_ucTxBuffer[n + 5] = temp + '0';               /* '0'～'9'に変換 */
        }
}else{  /* AD設定値の送信データがない場合 */
        for( n = 0; n < 4; n++ )
                p_ucTxBuffer[n + 5] = '*';              /* DAC設定値の変わりに'*'を出力 */
}

p_ucTxBuffer[9] = ' ';          /* データ間SP */

/* 5点補正 送信データ設定 */
if( s_Result.bCorrect_ready )
{       /* 5点補正の送信データがある場合 */
        for( n = 0; n < 4; n++ ){
                /* 送信データに変換する桁を取り出し */
                temp = (unsigned char)((s_Result.ushCorrect & ( 0xf000 >> 4*n )) >> (4*(3 - n)));
                if( temp >= 0xa )       /* 送信データが0xa～0xfの場合 */
                        p_ucTxBuffer[n + 10] = temp + ( 'A' - 0xa );    /* 'A'～'F'に変換 */
                else                    /* 送信データが0x0～0x9の場合 */
                        p_ucTxBuffer[n + 10] = temp + '0';              /* '0'～'9'に変換 */
        }
```

```
        }else{  /* 5点補正の送信データがない場合 */
                for( n = 0; n < 4; n++ )
                        p_ucTxBuffer[n + 10] = '*';             /* DAC設定値の変わりに'*'を出力 */
        }

        p_ucTxBuffer[14] = ' ';         /* データ間SP */

        /* 温度補正結果 送信データ設定 */
        if( s_Result.bHeatCorr_ready )
        {       /* 温度補正結果の送信データがある場合 */
                for( n = 0; n < 4; n++ ){
                        /* 送信データに変換する桁を取り出し */
                        temp = (unsigned char)((s_Result.ushHeatCorr & ( 0xf000 >> 4*n )) >> (4*(3 - n)));
                        if( temp >= 0xa )       /* 送信データが0xa～0xfの場合 */
                                p_ucTxBuffer[n + 15] = temp + ( 'A' - 0xa );    /* 'A'～'F'に変換 */
                        else                    /* 送信データが0x0～0x9の場合 */
                                p_ucTxBuffer[n + 15] = temp + '0';              /* '0'～'9'に変換 */
                }
        }else{  /* A/D変換結果補正値の送信データがない場合 */
                for( n = 0; n < 4; n++ )
                        p_ucTxBuffer[n + 15] = '*';             /* DAC設定値の変わりに'*'を出力 */
        }

        p_ucTxBuffer[19] = '\r';        /* キャリッジリターン */
        p_ucTxBuffer[20] = '\n';        /* 改行 */
}

/*****************************************************************************

        UART6で変換結果出力

-----------------------------------------------------------------------------
        送信バッファの内容をUARTで送信します。
*****************************************************************************/
static void fn_ADResultOut(unsigned char *p_TxBuffer)
{
        unsigned char ucTxBufferCounter = 0;            /* 送信カウンタ */

        /* 送信動作開始 */
        STIF6 = 1;                                      /* 割り込み要求セット */

        while( ucTxBufferCounter < TXDATA_SIZE ){       /* UART6送信完了待ち */
                if( STIF6 ){
```

```
                STIF6 = 0;                      /* 割り込み要求クリア */

                TXB6 = p_TxBuffer[ucTxBufferCounter];   /* TxD6: データ送信 */
                ucTxBufferCounter++;            /* 送信カウンタ更新 */
            }

            if(PIF0){                           /* 停止キーが押された場合 */
                STIF6 = 1;                      /* 割り込み要求セット */
                break;
            }
        }
}
```

付録B　改版履歴

版　数	発行年月	改版箇所	改版内容
第1版	June 2008	－	－

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。