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/Kx2

サンプル・プログラム

タッチスクリーン編

対象デバイス

78K0/KB2マイクロコントローラ
78K0/KC2マイクロコントローラ
78K0/KD2マイクロコントローラ
78K0/KE2マイクロコントローラ
78K0/KF2マイクロコントローラ

資料番号　U19720JJ2V0AN00（第2版）
発行年月　January 2010 N

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

(1) 入力端子の印加波形：入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOSデバイスの入力がノイズなどに起因して，VIL（MAX.）からVIH（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，VIL（MAX.）からVIH（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

(2) 未使用入力の処理：CMOSデバイスの未使用端子の入力レベルは固定してください。未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してVDDまたはGNDに接続することが有効です。資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

(3) 静電気対策：MOSデバイス取り扱いの際は静電気防止を心がけてください。MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。また，MOSデバイスを実装したボードについても同様の扱いをしてください。

(4) 初期化以前の状態　電源投入時，MOSデバイスの初期状態は不定です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

(5) 電源投入切断順序　内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

(6) 電源OFF時における入力信号　当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

・本資料に記載されている内容は2010年1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，ソフトウエ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。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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。

注2. 本事項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，製造製品をいう。

（M8E0909J）

はじめに

対象者　このマニュアルは,78K0マイクロコントローラの応用システムを設計,開発するユーザを対象とします。

目　的　78K0マイクロコントローラでのタッチスクリーン制御方法についてユーザに理解していただくことを目的とします。

構　成　このマニュアルは，大きく分けて次の内容で構成しています。

- 概　説
- 座標検出方式
- アプリケーション実装例
- サンプル・アプリケーションのビルド方法
- サンプル・アプリケーションの実行方法

読み方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータおよびC言語に関する一般知識を必要とします。

本アプリケーションの実行評価環境を理解しようとするとき

→漢字表示デモンストレーション用ボードの**ユーザーズ・マニュアル**を参照してください。

マイクロコントローラのハードウエア機能の詳細を理解しようとするとき

→各製品の**ユーザーズ・マニュアル　ハードウエア編**を参照してください。

凡　例　データ表記の重み：左が上位桁，右が下位桁

アクティブ・ローの表記：$\overline{xxx}$（端子，信号名称に上線）

メモリ・マップのアドレス：上部－上位，下部－下位

注　：本文中に付けた注の説明

注意：気を付けて読んでいただきたい内容

備考：本文の補足説明

数の表記：2進数　… xxxxまたはxxxxB

10進数 … xxxx

16進数 … xxxxH

2のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）… $2^{10} = 1024$

M（メガ）… $2^{20} = 1024^2$

G（ギガ）… $2^{30} = 1024^3$

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

★　**ヒューマン・マシン**I/F**デモ・ボート，漢字表示デモ・ボードの関連資料**

資　料　名	資料番号	
	和　　文	英　　文
ヒューマン・マシンI/Fデモ用ベース・ボード　ユーザーズ・マニュアル	U20117J	未定
ヒューマン・マシンI/Fデモ用78K0ボード　ユーザーズ・マニュアル	U20118J	未定
ヒューマン・マシンI/Fデモ用78K0Rボード　ユーザーズ・マニュアル	U20119J	未定
ヒューマン・マシンI/Fデモ用V850ESボード　ユーザーズ・マニュアル	U20120J	未定
漢字表示デモンストレーション用ベース・ボード　ユーザーズ・マニュアル	U19207J	未定
漢字表示デモンストレーション用78K0/KF2ボード　ユーザーズ・マニュアル	U19208J	未定
漢字表示デモンストレーション用78K0R/KG3ボード　ユーザーズ・マニュアル	U19209J	未定
漢字表示デモンストレーション用V850ES/JG3ボード　ユーザーズ・マニュアル	U19210J	未定
78K0/Kx2サンプル・プログラム（簡易OS編）アプリケーション・ノート	U19214J	未定
78K0R/Kx3サンプル・プログラム（簡易OS編）アプリケーション・ノート	U19215J	未定
V850ES/Jx3サンプル・プログラム（簡易OS編）アプリケーション・ノート	U19216J	未定
フォント・ユーティリティ　ユーザーズ・マニュアル	U19527J	未定
78K0/Kx2 サンプル・プログラム（フォント選択編）アプリケーション・ノート	U19528J	未定
78K0R/Kx3 サンプル・プログラム（フォント選択編）アプリケーション・ノート	U19529J	未定
V850ES/Jx3 サンプル・プログラム（フォント選択編）アプリケーション・ノート	U19530J	未定
漢字表示デモンストレーション用拡張ボード　ユーザーズ・マニュアル	U19526J	未定
78K0/Kx2 サンプル・プログラム（ドットＬＣＤ制御編）アプリケーション・ノート	U19531J	未定
78K0R/Kx3 サンプル・プログラム（ドットＬＣＤ制御編）アプリケーション・ノート	U19532J	未定
V850ES/Jx3 サンプル・プログラム（ドットＬＣＤ制御編）アプリケーション・ノート	U19533J	未定
78K0/Kx2 サンプル・プログラム（タッチスクリーン編）アプリケーション・ノート	このノート	未定
78K0R/Kx3 サンプル・プログラム（タッチクリーン編）アプリケーション・ノート	U19721J	未定
V850ES/Jx3 サンプル・プログラム（タッチクリーン編）アプリケーション・ノート	U19722J	未定
78K0/Kx2 サンプル・プログラム（コーデック接続編）アプリケーション・ノート	U19723J[注]	未定
78K0R/Kx3 サンプル・プログラム（コーデック接続編）アプリケーション・ノート	U19724J[注]	未定
V850ES/Jx3 サンプル・プログラム（コーデック接続編）アプリケーション・ノート	U19725J[注]	未定

注.　2010**年春発行予定**

78K0**マイクロコントローラ・デバイスの関連資料**

資　料　名	資料番号	
	和　　文	英　　文
78K0/Kx2　ユーザーズ・マニュアル	U18598J	U18598E
78K0マイクロコントローラ ユーザーズ・マニュアル 命令編	U12326J	U12326E

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　　次

第1章　概　　説

この資料では，78K0用タッチスクリーン・サンプル・プログラムについて説明します。

1. 1　プログラム概要

本サンプル・プログラムは，抵抗膜式タッチスクリーンの座標検出制御を行います。実際の動作確認は，「サンプル・プログラム（ドットLCD制御編）アプリケーション・ノート(U19531)」で解説している中のタッチスクリーン付き製品２品種で行うことができます。

1. 2　開発環境

サンプル・プログラムからオブジェクト・コードを生成するソフトウエア・ツールと，生成したコードを実行する評価ボードについて説明します。

1. 2. 1　ソフトウエア・ツール

本ライブラリを使用するアプリケーション・プログラムの開発に推奨するソフトウエア・ツールは，NECエレクトロニクス製の有償ソフトウエア・パッケージ（SP78K0）です。

(1) 推奨リビジョン

ソフトウエア・ツールの推奨リビジョンです。

(a) CC78K0	：4.00以上
(b) RA78K0	：4.01以上

(2) 無償ダウンロード版の制約

無償ダウンロード版のコンパイラおよびアセンブラ（CC78K0, RA78K0）には生成できるオブジェクト・サイズに制限がありますが，本サンプル・プログラムだけであれば特に制約無く使用できます。

★ 1.2.2 評価ボード

実際にタッチスクリーン付きLCDモジュールで評価するには，下記のものが必要になります。

(1)ボードおよびタッチスクリーン

ボードとしては，内蔵A/Dコンバータ方式と専用コントローラ方式の２種類があります。

ボードの入手に関する情報については，下記のURLをご参照ください。

http://www.necel.com/micro/ja/designsupports/board/index.html

また，各種LCD評価キットの入手に関する情報については，下記のURLをご参照ください。

http://www.space-i.co.jp/sm05/

① 内蔵A/Dコンバータを使用する場合

・ヒューマン・マシンI/Fデモ用ベース・ボード（SM07A）

・ヒューマン・マシンI/Fデモ用78K0ボード（SM06E）

・タッチスクリーン付きLCDモジュール（BP240128，BG12864）

LCDモジュールの調達方法については，メーカまたは代理店にお問い合わせ下さい。

代理店はドットLCD制御編（U19531）「表2－5 連絡先・代理店一覧表」を参照ください。

② 専用コントローラを使用する場合

・漢字表示デモンストレーション用ベース・ボード（SM05A2）

・漢字表示デモンストレーション用78K0ボード（SM05F3（SM05F2は非対応です））

・漢字表示デモンストレーション用拡張ボード（SM06B2）

・各種LCD評価キット

漢字表示デモンストレーション用LCDモジュール

漢字表示デモンストレーション用LCD接続アダプタ

漢字表示デモンストレーション用LCD接続ハーネス

漢字表示デモンストレーション用タッチスクリーン接続ハーネス

(2) プログラム書き込みツール

推奨するプログラム書き込みツールは，オンチップ・デバッグ・エミュレータ（QB-MINI2）です。

PG-FP5などのフラッシュ・メモリ・プログラマ（以降，プログラマ）でも可能です。

(3) タッチペン

タッチスクリーンの精度測定に必要となります。

本書の測定例では以下のペンを使用しています。

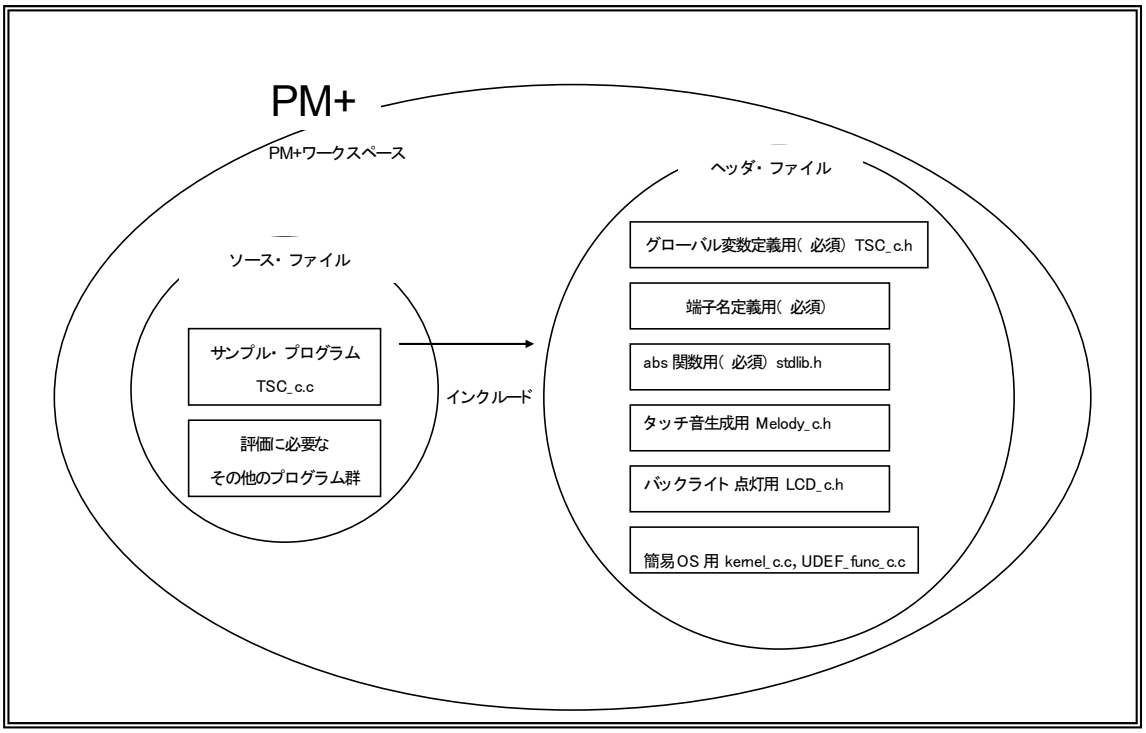
メーカ：サンワサプライ

型　番：マルチ入力ペン　PDA－PEN12

1. 3　アプリケーションの依存情報

プロジェクト・マネージャ（以降，PM+）環境下での依存情報を次の図に示します。

図1－1　PM+環境下での依存情報の関係



第2章　座標検出方式

この章では，タッチスクリーンの座標検出方式について説明します。

2. 1　タッチスクリーン構造

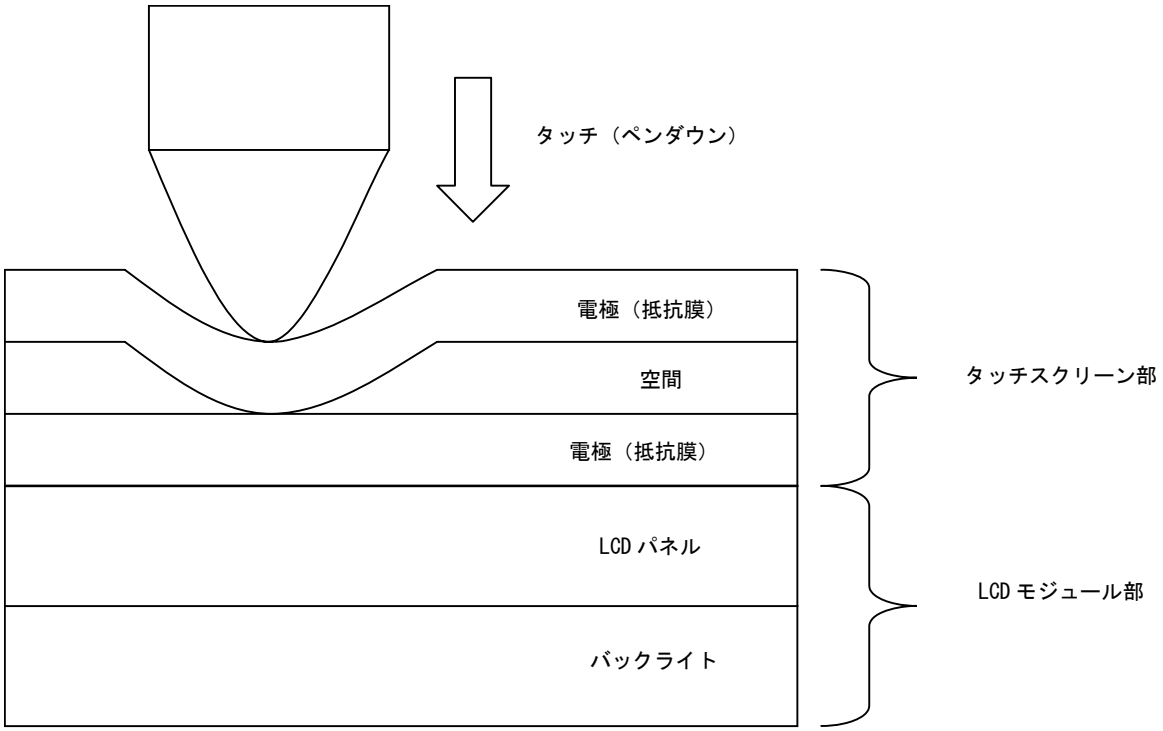
抵抗膜式タッチスクリーンは，LCDモジュール等の表示装置の前面に透明な電極を2枚設置し，指や専用ペンなどでタッチすることで電極同士が接触し，タッチした位置を得ることができます。

2. 1. 1　抵抗膜の構造

抵抗膜の構造を図2－1に示します。LCDモジュール前面の電極は抵抗膜構造になっており，タッチされていなければ抵抗膜同士には空間があり接触していません。抵抗膜に指などでタッチされると前面の抵抗膜が変形し抵抗膜同士が接触します。

抵抗膜の両端には電極が接続されており，接触した位置から両端の電極までの値を得る事で，タッチした位置を知ることができます。

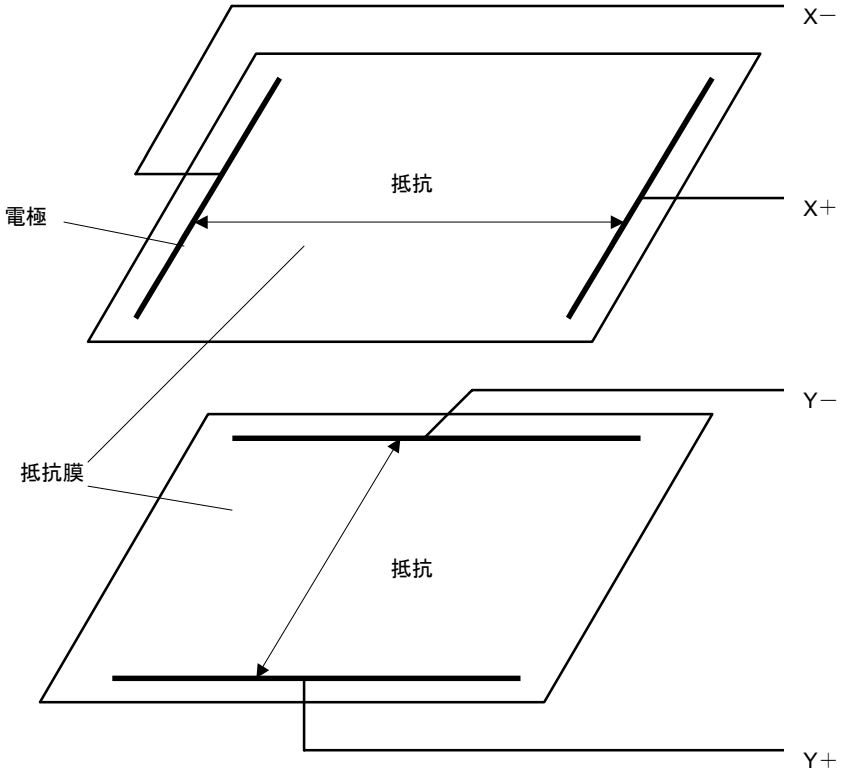
図2－1　抵抗膜構造（断面図）



2. 1. 2　4線式の電極構成

4線式の電極構成を図2－2に示します。2枚の抵抗膜の両端には棒状の電極が作られており，各抵抗膜がX方向とY方向の変化を得られるよう作られています。ここで得られる値はアナログ値となり，値取得にはX方向とY方向に分けて取得する必要があります。

図2－2　4線式の電極構成



2. 2　基本的測定方法

抵抗膜方式タッチスクリーンの測定は，構造上X値とY値に分けて測定します。

2. 2. 1　測定準備

2枚の抵抗膜のうち，測定を行いたい抵抗膜の両端に電圧を供給します。タッチスクリーンと供給電圧との関係を図2－3および図2－4に示します。供給電圧の供給方法は「2. 3. 1 内蔵A／Dコンバータを使用する例」と，「2. 3. 2 専用コントローラを使用する例」で説明します。

図2－3　検出用供給電圧　（X値の場合）

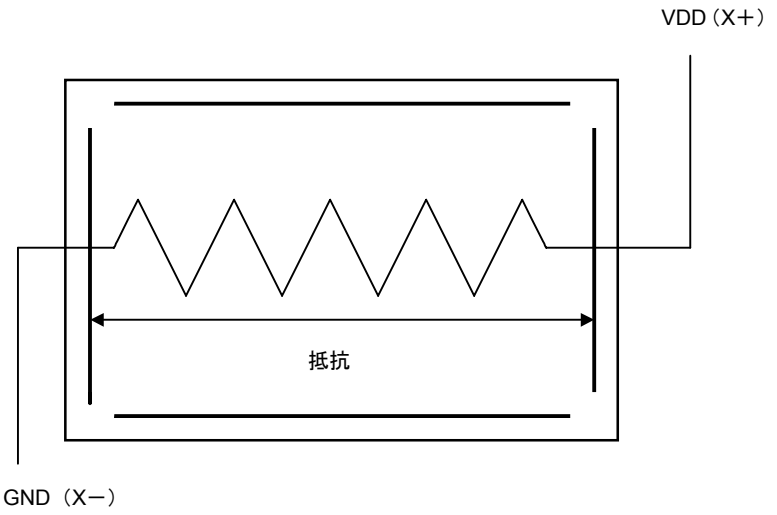
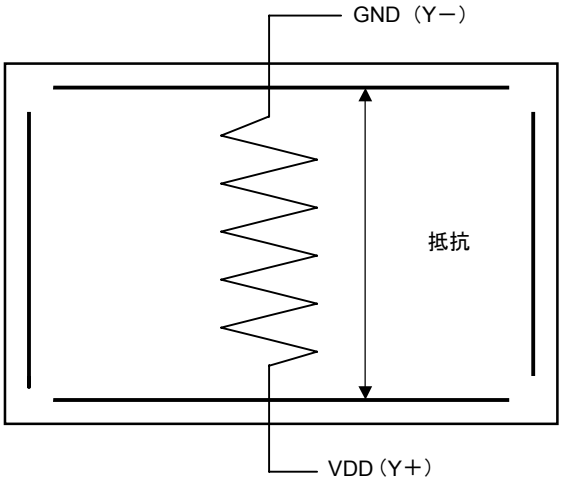


図2－4　検出用供給電圧　（Y値の場合）



2. 2. 2　測定電圧

測定を行いたい抵抗膜の両端に電圧を掛けた状態で，タッチしたX軸の電圧はY軸の＋電極（Y＋）に出力されます。ペンダウンと測定電圧との関係を図2－5および図2－6に示します。Y軸の電圧はX軸の＋電極（X＋）に出力されます。出力された値はVDDからGNDまでの範囲の分圧された電圧が出力されます。

★　なお，電圧を印可してからタッチ位置電圧が安定するまでには多少の時間がかかります。具体的には，抵抗膜間の静電容量と抵抗膜抵抗，電源インピーダンスにより決まり，後述のサンプル・アプリケーションではソース記述パラメータで安定待ち時間を変更できるようになっています（デフォルト記述は10 μ s）。

図2－5　測定電圧　（X値の場合）

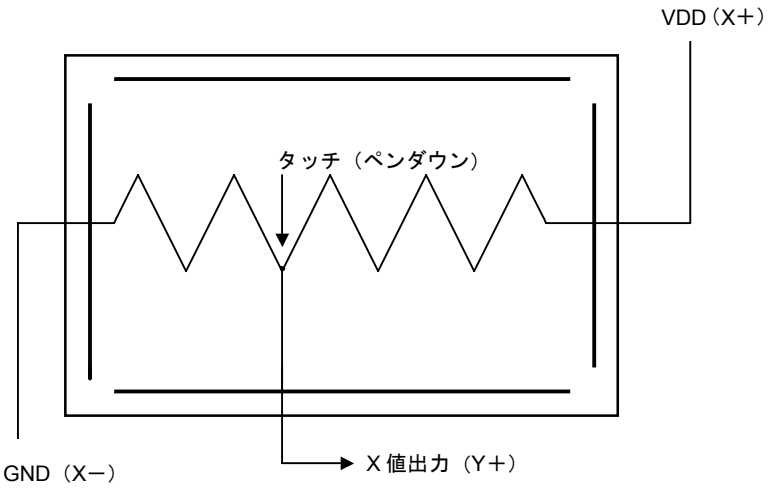
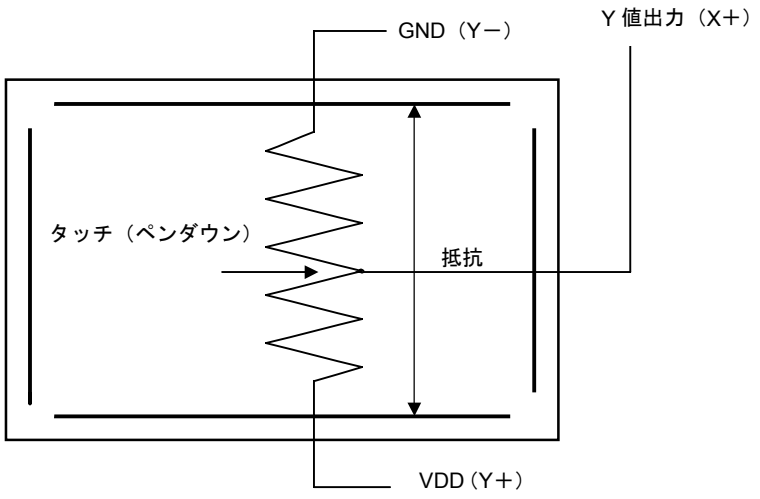


図2－6　測定電圧　（Y値の場合）



2. 3　測定回路と制御方法

2. 3. 1　内蔵A／Dコンバータを使用する例

マイクロコントローラ内蔵A/Dコンバータを使用して，タッチスクリーンのアナログ値を取得する例を説明します。

次の手順で値の取得を行います。

（1）初期化

内蔵A/Dコンバータを使用した場合の回路例を図2－7に示します。X軸，Y軸への電圧供給はポート（Port_X，Port_Y）により制御します。X軸の値を取得したい場合は，X軸に電圧供給した上で，ADC_Y+の値を読み取ります。また，供給電圧を測定するためにADC_X＋，ADC_X－の値も読み取ります。

各ポートの初期値を表2－1に示します。初期状態では，ペンダウン検出のためY－をGNDに接続し，X＋にプルアップ抵抗が接続されるようにします。

図2－7　内蔵A/Dコンバータを使用する場合の外部回路例

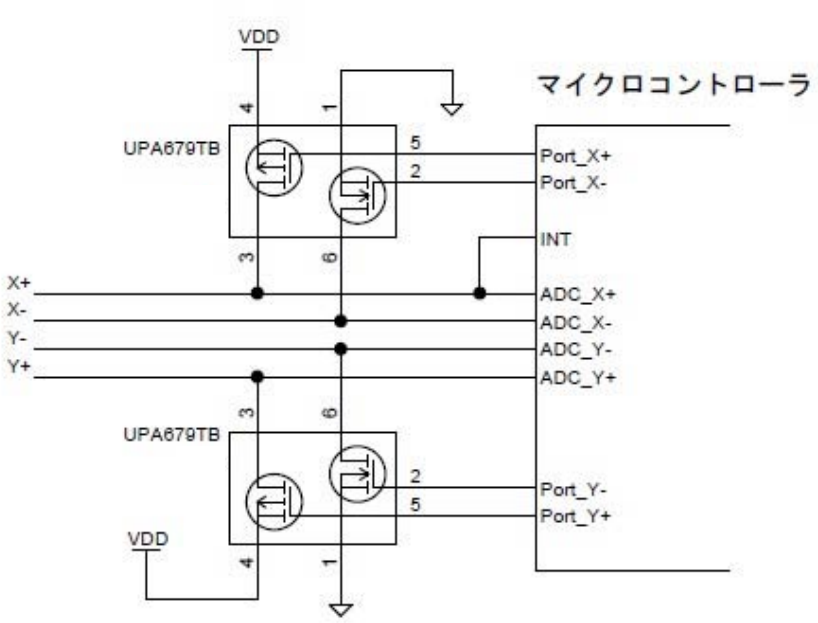


表2－1　各ポートの初期値

ポート名	値　（状態）
Port_X＋	OFF（H）
Port_X－	OFF（L）
Port_Y－	ON（H）
Port_Y＋	OFF（H）
INT（Pull_up ON/OFF）	入力有効＆プルアップON

(2) ペンダウン検出

各ポートの初期化を行うと，タッチスクリーンがタッチされたかどうかをINT端子より知ることができます。

タッチされていない状態では，INT端子の内蔵プルアップによりINT端子は，VDDに近い電圧になります。タッチされた場合，INT端子は低い電圧になります。

(3) 座標取得手順

INT端子でタッチを検出したら，内蔵のA/Dコンバータを使用しタッチした座標の取得を行います。取得は以下の手順で行います。

① X軸ポート設定（X軸検出，INT端子無効）

★ ② 安定待ち時間後，X軸値取得（ADC_X＋，ADC_X－，ADC_Y＋値取得）

③ Y軸ポート設定（Y軸検出，INT端子無効）

④ 安定待ち時間後，Y軸値取得（ADC_X＋，ADC_Y－，ADC_Y＋値取得）

⑤ 印可電圧オフ，INT端子ポート設定（INT端子有効）

X軸の値を取得するために，ポートを表2－2に示す値に設定されます。

表2－2　X軸値ポート設定

ポート名	値（状態）
Port_X＋	ON（L）
Port_X－	ON（H）
Port_Y－	OFF（L）
Port_Y＋	OFF（H）
INT（Pull_up ON/OFF）	入力無視＆プルアップOFF
ADC_X＋	入力（A/D変換）
ADC_X－	入力（A/D変換）
ADC_Y－	－
ADC_Y＋	入力（A/D変換）

X軸の値を知るためには次の計算をします。

X軸値 = （ADC_Y+値 － ADC_X-値）÷（ADC_X+値 － ADC_X-値）　[%]

Y軸の値を取得するために，ポートを表2－3に示す値に設定されます。

表2－3　Y軸ポート設定

ポート名	値（状態）
Port_X＋	OFF（H）
Port_X－	OFF（L）
Port_Y－	ON（H）
Port_Y＋	ON（L）
INT（Pull_up ON/OFF）	入力無視＆プルアップOFF
ADC_X＋	入力（A/D変換）
ADC_X－	－
ADC_Y－	入力（A/D変換）
ADC_Y＋	入力（A/D変換）

Y軸の値を知るためには次の計算をします。

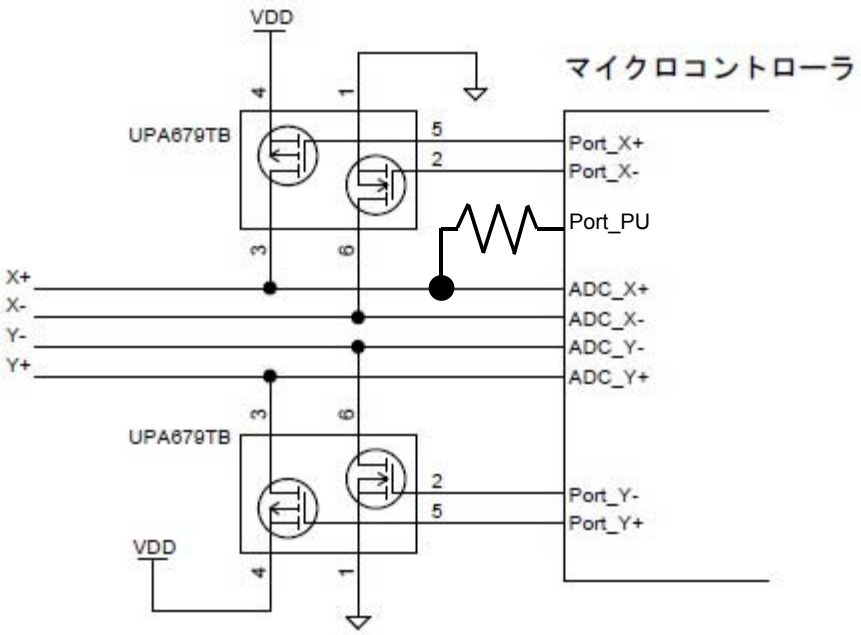
Y軸値 ＝ （ADC_X+値 － ADC_Y-値）÷（ADC_Y+値 － ADC_Y-値）　[%]

★　（4）SM07Aベース・ボード使用時の制御方法

SM07Aベース・ボードを使用した場合の回路を図2－8に示します。プルアップする場合はPort_PUを ”H ” にし，プルアップをOFFする場合はPort_PUを ” Hi-z ” にします。

ペンダウン検出はADC_X+端子で行います。

図2－8　SM07Aベース・ボードの外部回路



2. 3. 2　専用コントローラを使用する例

デモンストレーション用拡張ボード（SM06B2）に搭載されているタッチスクリーン専用コントローラTSC2046を使用する例を説明します。

次の手順で値の取得を行います。

(1) 初期化

タッチスクリーン専用コントローラTSC2046はA/Dコンバータを内蔵しており，タッチスクリーンからのアナログ電圧をディジタル値に変換します。X軸，Y軸への電圧供給の切り替え等はTSC2046にコマンドを送ることにより行います。TSC2046への接続はCSI（3線式シリアル・インタフェース）にて接続します。

TSC2046への接続を表2－4に示します。TSC2046の制御タイミングを図2－9に示します。

TSC2046初期化のために次のコマンドを送ります。

90H　（ADC停止[注]，ペンダウン検出有効）

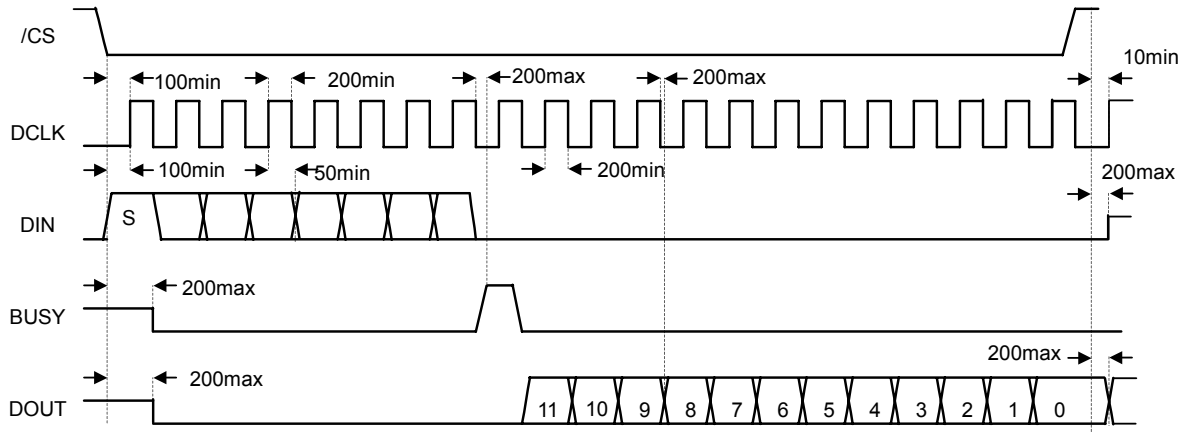
注　通常TSC2046のA/Dコンバータは停止させ，測定を行う時のみ動作させます。

表2－4　TSC2046への接続

TSC2046端子名	コネクタ信号名（サンプル・プログラムでの定義名）
/CS（チップセレクト）	PIO0（zCSI_CS2）
DCLK（クロック）	PIO4（zCSI_SPC）
DIN（コマンド入力）	CTS1（zCSI_SPO）
BUSY（ビジー・チェック）	無接続
DOUT（データ出力）	PIO6（zCSI_SPI）
PENIRQ（ペンダウン検出）	PIO8（zPEN_INQ）

図2－9　TSC2046 信号タイミング

単位：ns



(2) ペンダウン検出

ペンダウン検出を有効にした場合，TSC2046の内部では図2－9のような接続に切り替わります。
タッチがされていない状態では，PENIRQ端子は内部抵抗によりVDDに近い電圧になります。
タッチがされた場合， PENIRQ端子は低い電圧になります。
ペンダウン検出時の関係を図2－10および図2－11に示します

図2－10　ペンダウン検出（タッチしない場合）

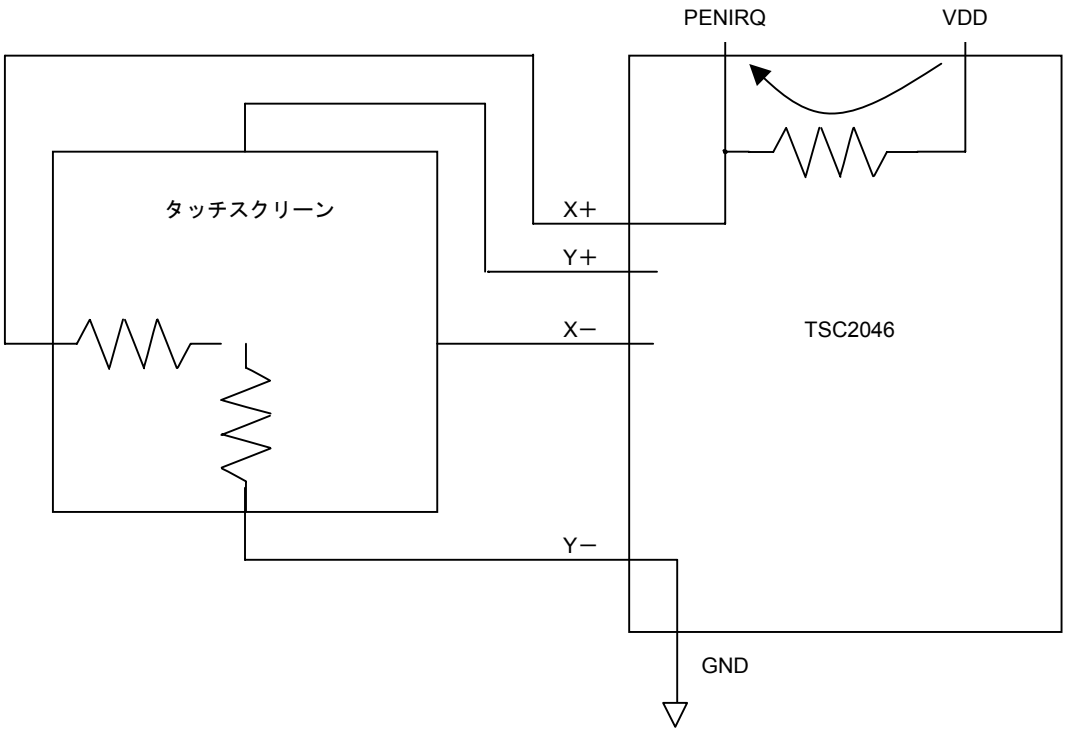
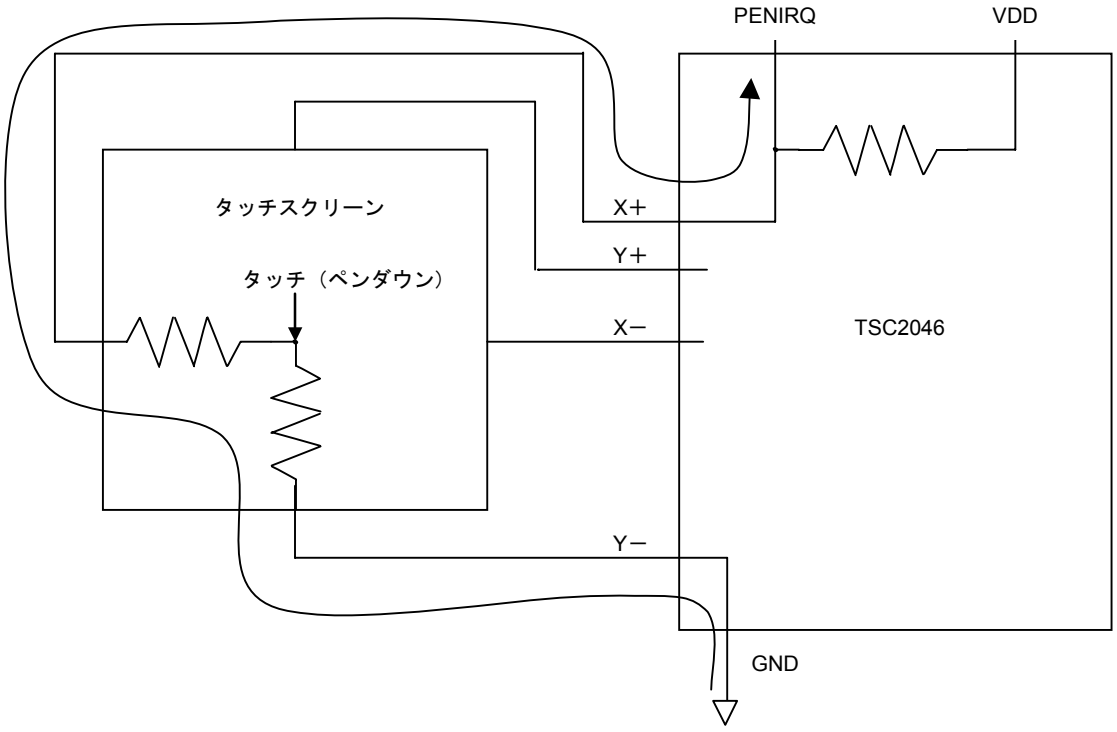


図2－11　ペンダウン検出（タッチした場合）



(3) 座標取得手順

TSC2046のPENINQ信号でペンダウンを検出したら，コマンドを送ってタッチした座標の取得を行います。取得は以下の手順で行います。

① D1H コマンド送信（X軸有効X軸電圧供給，ペン割り込み無効）

★ ② 安定待ち時間後，D1H コマンド再送し，X軸のデータを受信

③ 91H コマンド送信（Y軸有効Y軸電圧供給，ペン割り込み無効）

④ 安定待ち時間後，91H コマンド再送し，Y軸のデータを受信

⑤ 90H コマンド送信（ADC停止，ペン割り込み有効）

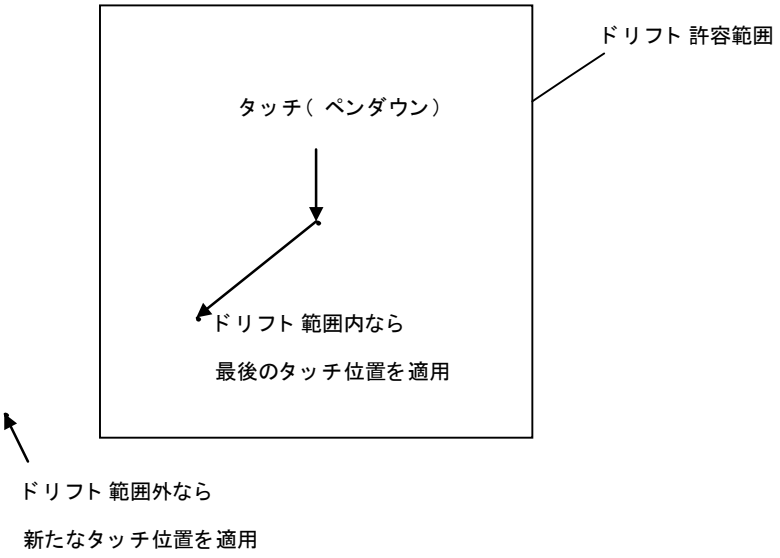
2. 4　読み取り安定化方法

タッチスクリーンから得られる値は，周辺ノイズやユーザの押し方の影響を受けます。そのため読み取った値の安定化を図る必要があります。

2. 4. 1　ドリフト許容範囲

ペンダウン後，ユーザの押し位置が微妙に揺れたり，周辺ノイズ等の影響により取得した値は常に変動する可能性があります。そのため取得した値にある程度の許容範囲を設ける必要があります。後述のサンプル・プログラムではドリフト許容範囲の設定をコマンドにより設定可能としています。ペンダウンとドリフト許容範囲の関係を図2－12に示します。

★　図2－12　ドリフト許容範囲

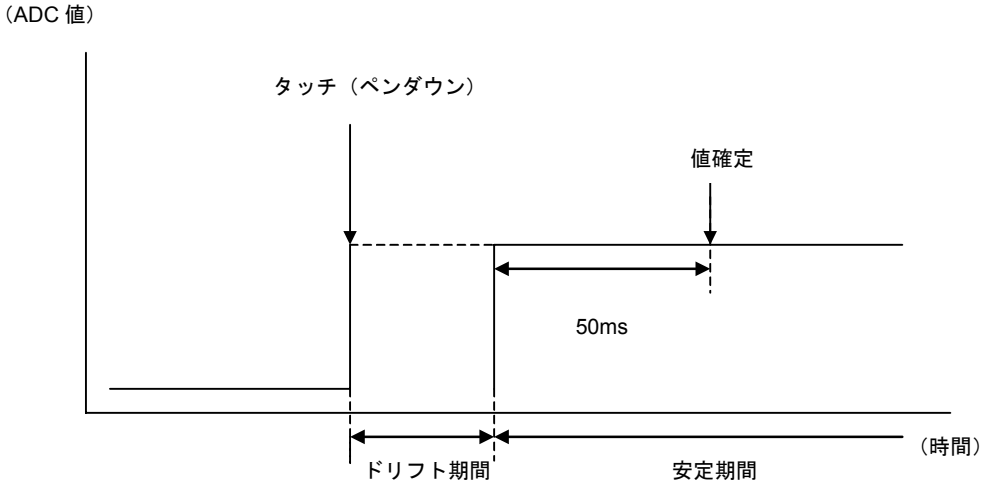


2. 4. 2　読み取り時間

ペンダウン直後は，ペンが揺れたりして微妙なドリフトが出る可能性があるため，取得値が安定するまでの時間を取る必要があります。

後述のサンプル・プログラムのデフォルト設定では，50ms（10ms間隔5回測定）の間，値が安定していれば値を読み取るように設定されています。ペンダウンと読み取り時間との関係を図2－13に示します。ペンの移動をリアルタイムで検出したい場合はこの機能の削除する必要があります。

図2－13　読み取り時間



2.5　座標補正

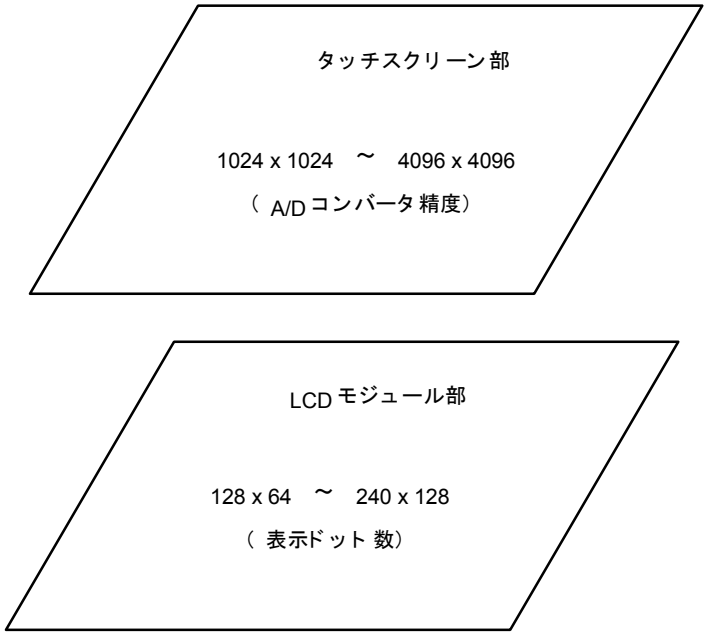
2.5.1　概　　要

タッチスクリーン部とLCDモジュール部の関係を図2－14に示します。タッチスクリーンから得られる値はA/Dコンバータの値であり，LCDモジュールの座標とは異なります。そのため，LCDのドット座標に合わせるためには得られた座標値を補正する必要があります。

座標補正は以下の手順で行います。

- タッチスクリーン・グリッド取得　：LCD上の表示位置に対する補正前のタッチ位置を取得します
- 回帰直線算出　：表示位置と補正前の取得位置から補正パラメータを作成します
- 補正パラメータ検証　：補正パラメータを設定し，正しく補正されたか検証します

図2－14　タッチスクリーン関係図



★ 2. 5. 2 **準　　備**

座標補正のための測定を行うには，LCD**モジュールに基準となる十字マークや格子を表示する必要があります。本例では，より正確な補正の例として多数の格子点を測定する方法を用いています。**

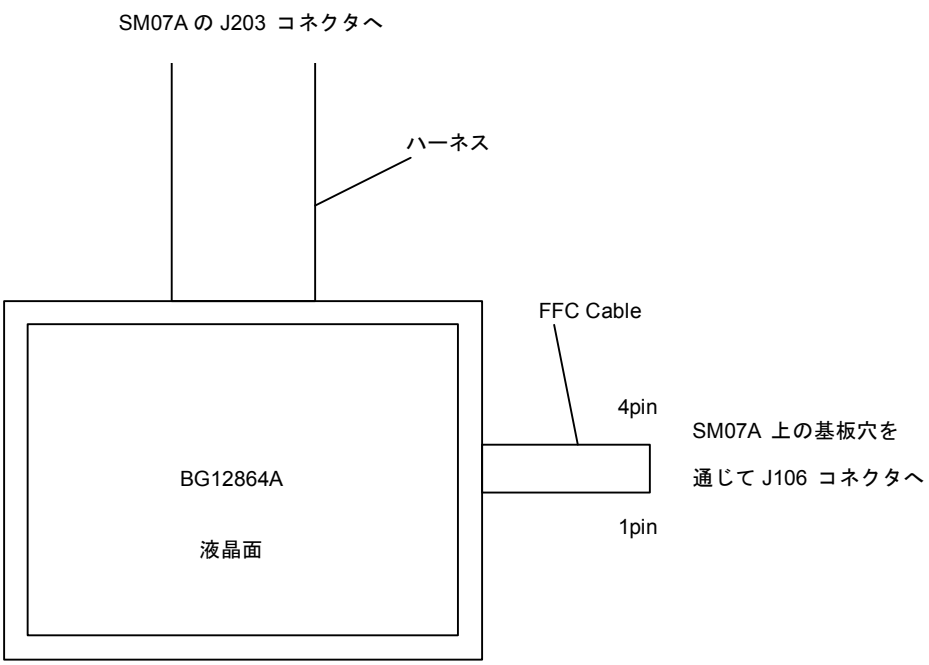
（1**）**BG12864A**（**128x64**ドット）**LCD**モジュール**

① SM07A**デモ・ボード（内蔵**A/D**コンバータ方式）へ取り付ける場合**

BG12864A**タッチスクリーン付き**128 x 64**ドット**LCD**モジュールを，ヒューマン・マシン**I/F**デモ用ベース・ボード（**SM07A**）へ接続する場合について説明します。**BG12864A**の詳細は「**78K0/Kx2**サンプル・プログラム ドット**LCD**制御編（**U19531J**）」を参照してください。**

BG12864A**を**SM07A**へ接続する方法は，「ヒューマン・マシン**I/F**デモ用ベース・ボード　ユーザーズ・マニュアル（**U20117**）　2. 2. 2 （2）BG12864AGPHHhpn207d$」を参照してください。**

図2**ー**15　BG12864A**を**SM07A**へ取り付け**



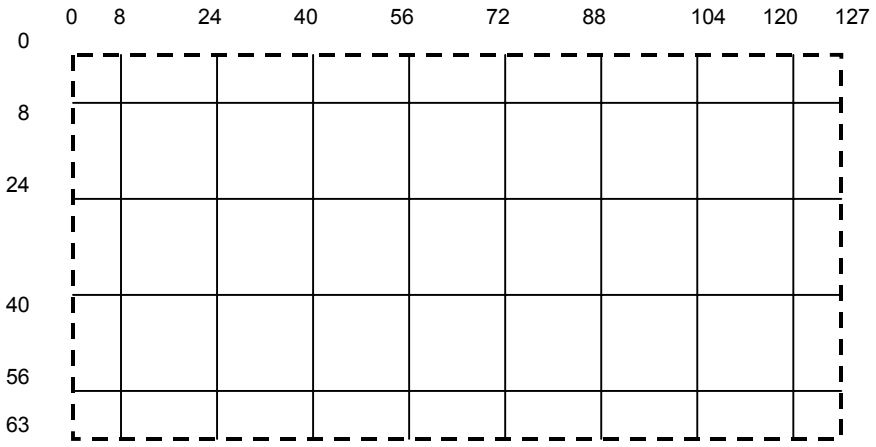
BG12864A をSM07Aへ接続する場合，後述のサンプル・アプリケーション実装例では，以下のようにパラメータ・アドレス1の設定と格子表示コマンドを行い，LCDモジュールに格子の表示と設定をします。

```
$gW'1 FA
$JF'00 00 42 A A A A
$NP'00 08 7F 08 00 01
$NP'00 18 7F 18 00 01
$NP'00 28 7F 28 00 01
$NP'00 38 7F 38 00 01
$NP'08 00 08 3F 20 FF
$NP'18 00 18 3F 20 FF
$NP'28 00 28 3F 20 FF
$NP'38 00 38 3F 20 FF
$NP'48 00 48 3F 20 FF
$NP'58 00 58 3F 20 FF
$NP'68 00 68 3F 20 FF
$NP'78 00 78 3F 20 FF
```

格子表示コマンドにて図2－16のような格子がLCDに表示されます。

図2－16　BG12864A格子表示

単位：ドット



（実線が表示される格子）

② SM05A2+SM06B2デモ・ボード（専用コントローラ方式）へ取り付ける場合

BG12864Aタッチスクリーン付き128 x 64ドットLCDモジュールを，拡張ボード（SM06B2）へ接続する場合について説明します。BG12864Aの詳細は「78K0/Kx2 サンプル・プログラム ドットLCD制御編（U19531J）」を参照してください。

BG12864Aのタッチスクリーン部接続を図2－17および図2－18に示します。

図2－17　BG12864Aタッチスクリーン図

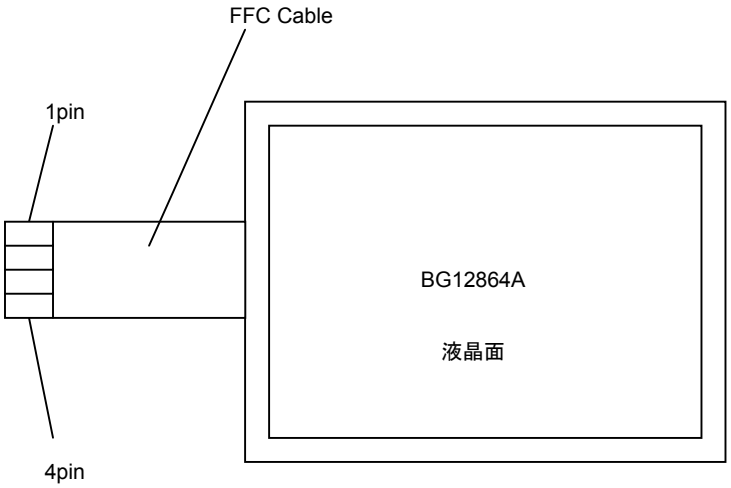
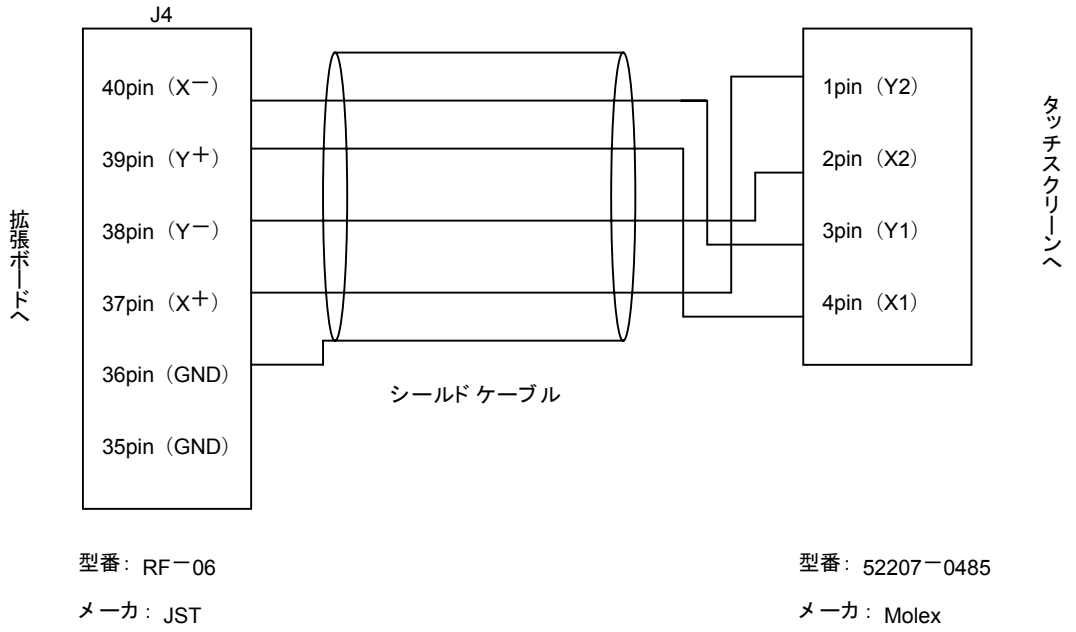


図2－18　BG12864Aタッチスクリーン・ハーネス図



型番：RF－06
メーカ：JST

型番：52207－0485
メーカ：Molex

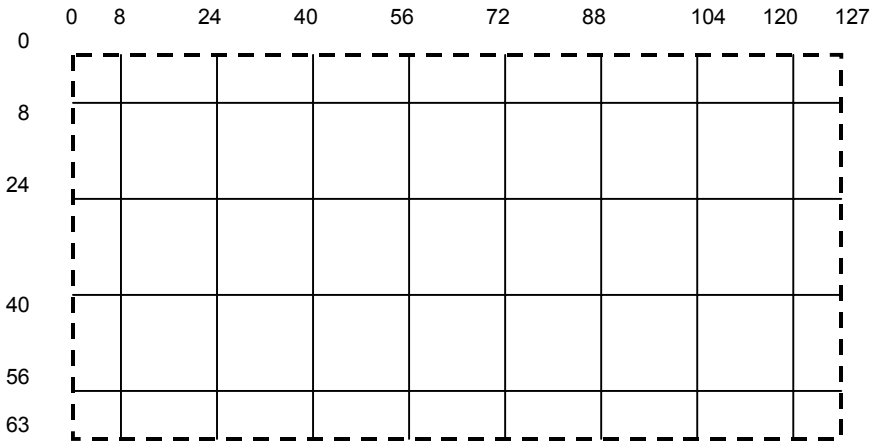
BG12864Aを拡張ボード（SM06B2）へ接続する場合，後述のサンプル・アプリケーション実装例では，以下のようにパラメータ・アドレス1の設定と格子表示コマンドを行い，LCDモジュールに格子の表示と設定をします。

```
$gW'1 F2
$JF'00 00 42 A A A A
$NP'00 08 7F 08 00 01
$NP'00 18 7F 18 00 01
$NP'00 28 7F 28 00 01
$NP'00 38 7F 38 00 01
$NP'08 00 08 3F 20 FF
$NP'18 00 18 3F 20 FF
$NP'28 00 28 3F 20 FF
$NP'38 00 38 3F 20 FF
$NP'48 00 48 3F 20 FF
$NP'58 00 58 3F 20 FF
$NP'68 00 68 3F 20 FF
$NP'78 00 78 3F 20 FF
```

格子表示コマンドにて図2－19のような格子がLCDに表示されます。

図2－19　BG12864A格子表示

単位：ドット



（実線が表示される格子）

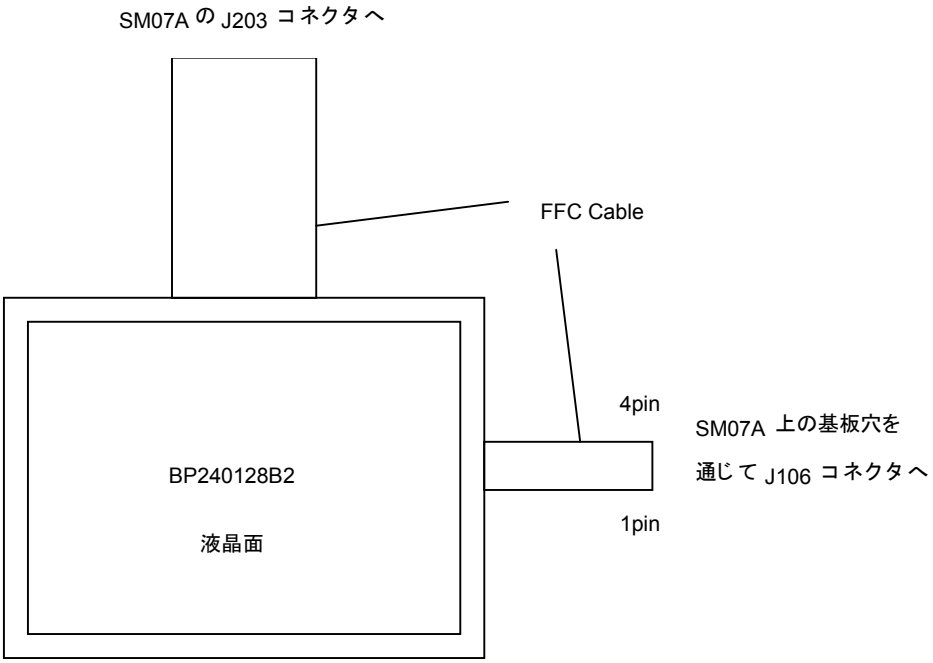
(2) BP240128（240x128ドット）LCDモジュール

①　SM07Aデモ・ボード（内蔵A/Dコンバータ方式）へ取り付ける場合

BP240128B2タッチスクリーン付き240 x 128ドットLCDモジュールを，ヒューマン・マシンI/Fデモ用ベース・ボード（SM07A）へ接続する場合について説明します。BP240128B2の詳細は「78K0/Kx2 サンプル・プログラム ドットLCD制御編（U19531J）」を参照してください。

BP240128B2をSM07Aへ接続する方法は，「ヒューマン・マシンI/Fデモ用ベース・ボード　ユーザーズ・マニュアル（U20117）　2. 2. 2 （1）BP240128B2FPHHhp$」を参照してください。

図2－20　BG12864AをSM07Aへ取り付け



BP240128B2を SM07Aへ接続する場合，後述のサンプル・アプリケーション実装例では，以下のようにパラメータ・アドレス1の設定と格子表示コマンドを行い，LCDモジュールに格子の表示と設定をします。

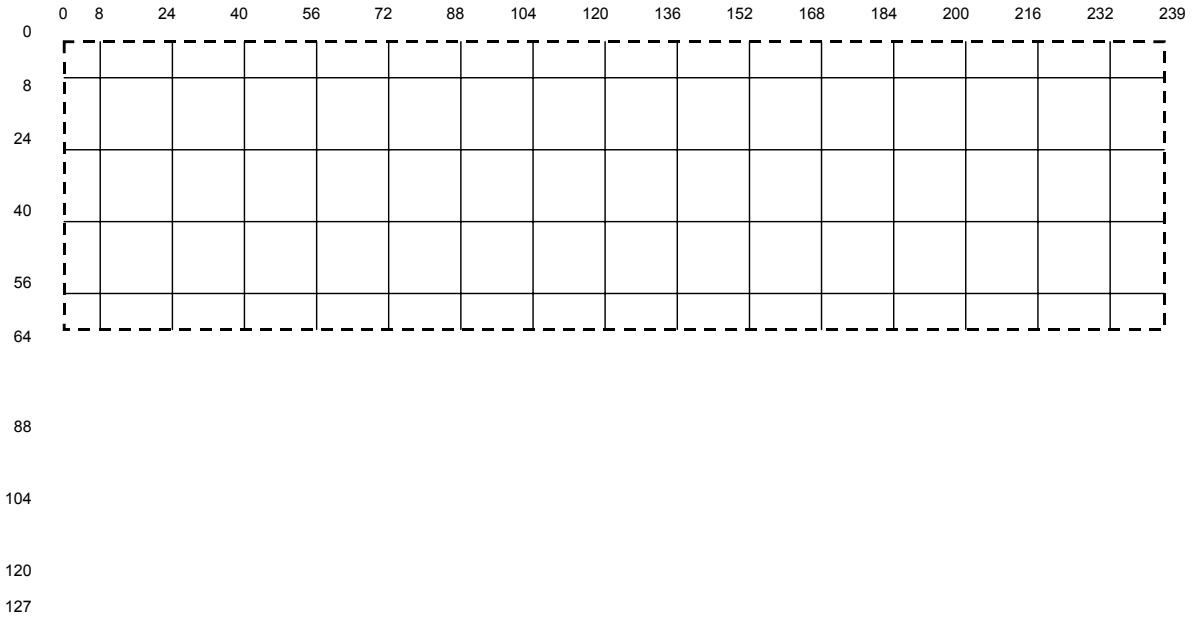
```
$gW'1 FF
$JF'00 00 42 A A A A A A A A
$NP'00 08 EF 08 00 01
$NP'00 18 EF 18 00 01
$NP'00 28 EF 28 00 01
$NP'00 38 EF 38 00 01
$NP'08 00 08 3F 20 FF
$NP'18 00 18 3F 20 FF
$NP'28 00 28 3F 20 FF
$NP'38 00 38 3F 20 FF
$NP'48 00 48 3F 20 FF
$NP'58 00 58 3F 20 FF
$NP'68 00 68 3F 20 FF
$NP'78 00 78 3F 20 FF
$NP'88 00 88 3F 20 FF
$NP'98 00 98 3F 20 FF
$NP'A8 00 A8 3F 20 FF
$NP'B8 00 B8 3F 20 FF
$NP'C8 00 C8 3F 20 FF
$NP'D8 00 D8 3F 20 FF
$NP'E8 00 E8 3F 20 FF
```

格子表示コマンドにて図2－21のような格子がLCDに表示されます。

BP240128B2を78K0で使用する場合，画面上半分のみ格子を表示します。

図2－21　BP240128B2格子表示

単位：ドット



（ 実線が表示される格子）

② SM05A2+SM06B2デモ・ボード（専用コントローラ方式）へ取り付ける場合

BP240128B2タッチスクリーン付き240 x 128ドットLCDモジュールを，拡張ボード（SM06B2）へ接続する場合について説明します。BP240128B2の詳細は「78K0/Kx2 サンプル・プログラム ドットLCD制御編（U19531J）」を参照してください。

BP240128B2のタッチスクリーン部接続を図2－22および図2－23に示します。

図2－22　BP240128B2タッチスクリーン図

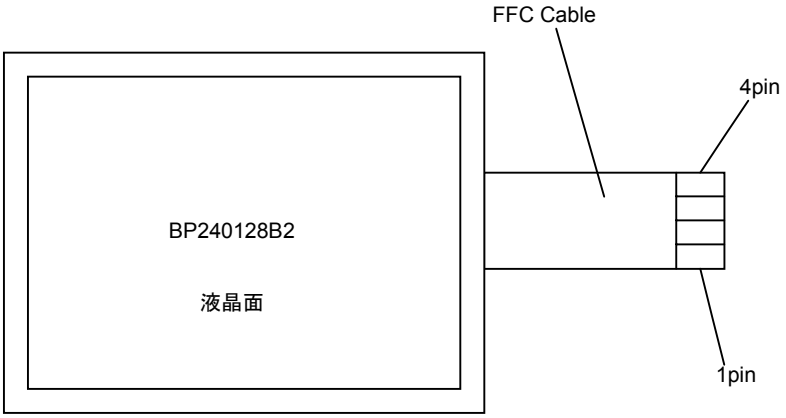
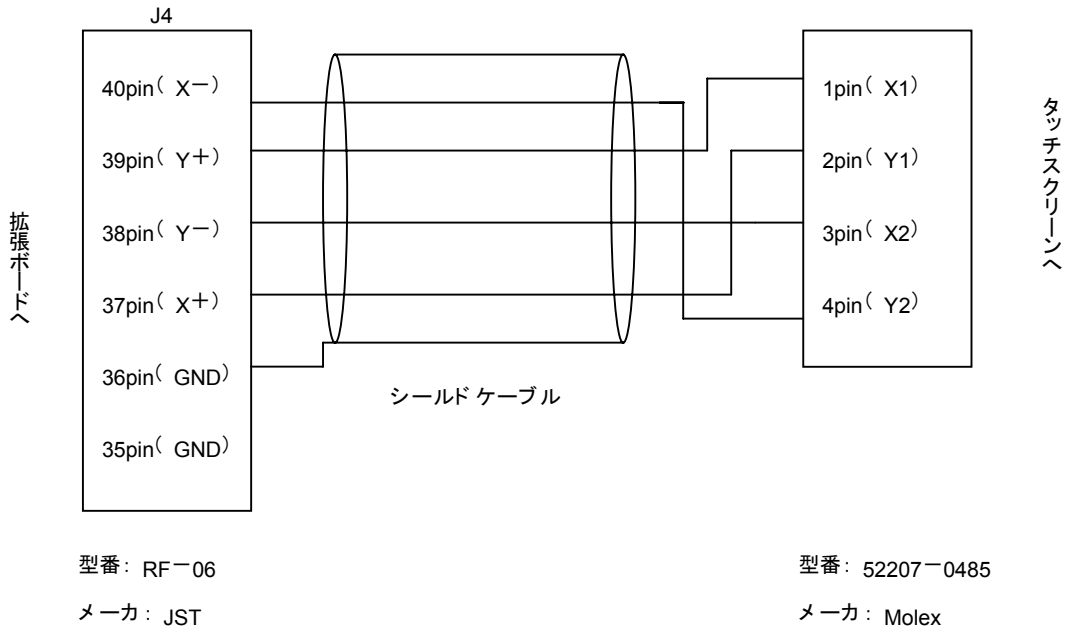


図2－23　BP240128B2タッチスクリーン・ハーネス図



型番：RF－06
メーカ：JST

型番：52207－0485
メーカ：Molex

BP240128B2を拡張ボード（SM06B2）へ接続する場合，後述のサンプル・アプリケーション実装例では，以下のようにパラメータ・アドレス1の設定と格子表示コマンドを行い，LCDモジュールに格子の表示と設定をします。

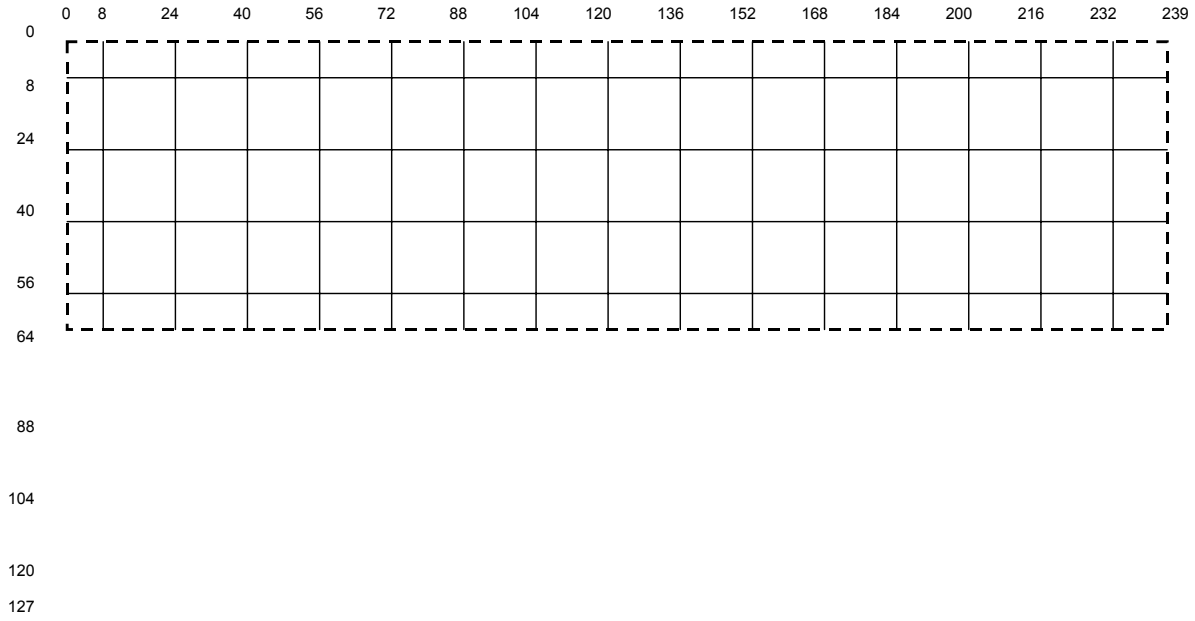
```
$gW'1 FF
$JF'00 00 42 A A A A A A A
$NP'00 08 EF 08 00 01
$NP'00 18 EF 18 00 01
$NP'00 28 EF 28 00 01
$NP'00 38 EF 38 00 01
$NP'08 00 08 3F 20 FF
$NP'18 00 18 3F 20 FF
$NP'28 00 28 3F 20 FF
$NP'38 00 38 3F 20 FF
$NP'48 00 48 3F 20 FF
$NP'58 00 58 3F 20 FF
$NP'68 00 68 3F 20 FF
$NP'78 00 78 3F 20 FF
$NP'88 00 88 3F 20 FF
$NP'98 00 98 3F 20 FF
$NP'A8 00 A8 3F 20 FF
$NP'B8 00 B8 3F 20 FF
$NP'C8 00 C8 3F 20 FF
$NP'D8 00 D8 3F 20 FF
$NP'E8 00 E8 3F 20 FF
```

格子表示コマンドにて図2－24のような格子がLCDに表示されます。

BP240128B2を78K0で使用する場合，画面上半分のみ格子を表示します。

図2－24　BP240128B2格子表示

単位：ドット



（ 実線が表示される格子）

2. 5. 3　タッチスクリーン・グリッド取得

タッチスクリーンの補正を行うためには，補正される前の値を取得する必要があります。

ここでは後述のサンプル・プログラムを使う場合について説明します。

まず，次のコマンドで生データを出力するように設定します。

$gW'3｛ドリフト許容範囲｝ ｛X軸オフセット｝ ｛Y軸オフセット｝ ｛X軸スケール｝ ｛Y軸スケール｝

ここで補正前は，以下の値にします。

★ ・ドリフト許容範囲：0.25ドット単位数。反応可能な最小値を設定します。標準では
8（専用コントローラ方式），14（16進数。内蔵10bitA/Dコンバータ方式）

・X軸，Y軸オフセット：オフセット量 × 128を設定します。ただし補正前は0を設定。

・X軸スケール：1.0（設定値は32768倍した8000Hをリトル・エンディアンで0080と入力）。
オフセット0の場合は，この設定により座標出力範囲が0～FFFHになります。

・Y軸スケール：0.5（おおむね画面の縦横比に合わせるため。設定値は32768倍した4000Hを
リトル・エンディアンで0040と入力）。オフセット0の場合は，この設定により
座標出力範囲が0～7FFHになります。

次に，LCDモジュールに表示されている格子の交点をタッチすると，その座標値が以下の形式で通知されます。

#gD!｛X軸値｝ ｛Y軸値｝

ここで，各軸値は，2バイトの16進数でビッグ・エンディアン表記となります。

通知例

#gD!'01EB01AD

X軸値：01EB（16進）→ 491（10進）

Y軸値：01AD（16進）→ 429（10進）

記録はX軸値とY軸値別々に行い，格子の交点を全て記録します。

本書ではエクセルの回帰分析機能を使うため，タッチした座標と通知された座標値をエクセルに記入していきます。

座標記入例を表2－5に示します。この例では全交点を複数回測定していますが，4隅の交点を1回づつ測定するだけでも補正計算は可能です。

表2－5　補正前座標記入例（BG12864A）

	A	B	C	D
1	X	x	Y	y
2	8	491	8	429
3	8	493	8	430
4	8	492	8	436
5	8	481	8	440
6	24	951	8	437
7	24	939	8	431
8	24	946	8	443
9	24	945	8	439
10	40	1392	24	820
11	40	1394	24	824
12	40	1393	24	820
13	40	1394	24	836
14	56	1826	24	830
15	56	1840	24	831
16	56	1849	24	845
17	56	1837	24	825
18	72	2292	40	1229
19	72	2286	40	1228
20	72	2289	40	1225
21	72	2282	40	1235
22	88	2759	40	1230
23	88	2758	40	1230
24	88	2751	40	1232
25	88	2749	40	1224
26	104	3214	56	1636
27	104	3209	56	1630
28	104	3204	56	1625
29	104	3209	56	1628
30	120	3671	56	1623
31	120	3666	56	1620
32	120	3663	56	1618
33	120	3656	56	1631

X：タッチしたX軸の座標（表示されている格子のドット位置）

x：タッチにより得られたX軸値

Y：タッチしたY軸の座標（表示されている格子のドット位置）

y：タッチにより得られたY軸値

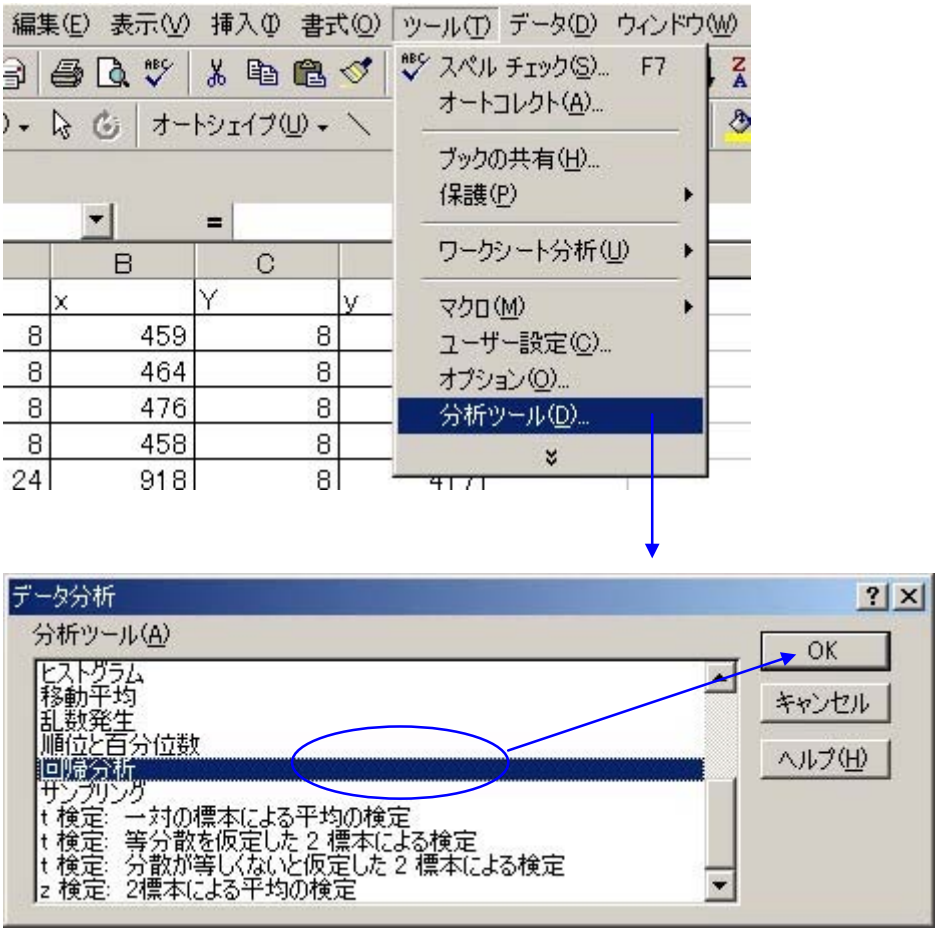
2. 5. 4　回帰直線算出

2. 5. 3で得られた座標値から回帰直線を用いて補正パラメータを算出します。

回帰直線はX軸Y軸別々に計算し，計算結果を補正パラメータに加工します。

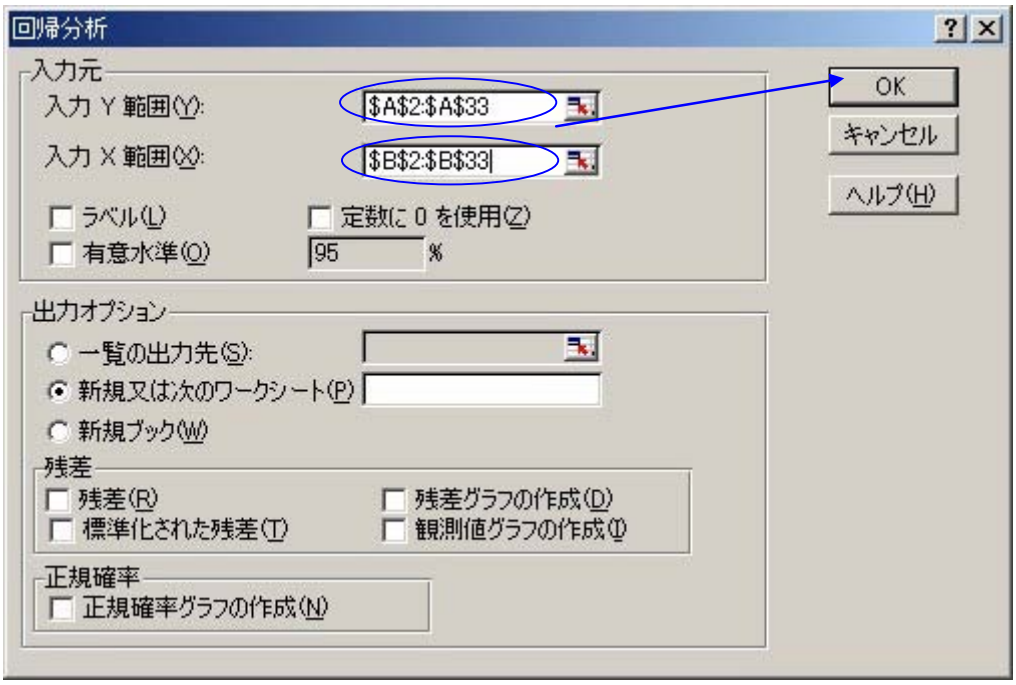
エクセルを用いた回帰直線算出方法を説明します。

①　「ツール」メニューの分析ツール[注]を実行し，回帰分析を選択します



注　分析ツールをアドオンから組み込む必要があります。詳しくはエクセルのヘルプを参照してください。

② 回帰ウインドウが表示されるので，「入力Y範囲」にはX（タッチしたX軸の座標）を指定し，「入力X範囲」にはx（タッチにより得られたX軸値）を指定し，回帰分析を実行します。



③ 回帰分析を実行すると，別シートに分析結果が表示されます。回帰分析はX軸とY軸別々に行います。

	A	B	C	
1	概要			
2				
3	回帰統計			
4	重相関 R	0.999973		
5	重決定 R2	0.999945		
6	補正 R2	0.999943		
7	標準誤差	0.280261		
8	観測数	32		
9				
10	分散分析表			
11		自由度	変動	
12	回帰	1	43005.64	43
13	残差	30	2.35638	0.0
14	合計	31	43008	
15				
16		係数	標準誤差	
17	切片	-9.1552	0.110585	-8
18	X 値 1	0.035298	4.77E-05	73

X 軸結果

	A	B	C	
1	概要			
2				
3	回帰統計			
4	重相関 R	0.999912		
5	重決定 R2	0.999824		
6	補正 R2	0.999818		
7	標準誤差	0.245139		
8	観測数	32		
9				
10	分散分析表			
11		自由度	変動	
12	回帰	1	10238.2	1
13	残差	30	1.80279	0.0
14	合計	31	10240	
15				
16		係数	標準誤差	
17	切片	-9.47791	0.109434	-8
18	X 値 1	0.04027	9.76E-05	41

Y 軸結果

★　④　解析結果の「係数」を基に，後述のサンプル・プログラムで使用する補正パラメータを作成します。補正パラメータはプログラム内でシフト処理されるため，シフト量に応じた値をあらかじめ掛けておきます。

X軸オフセット設定値＝－切片×128　　（小数点以下四捨五入）

Y軸オフセット設定値＝－切片×128　　（小数点以下四捨五入）

X軸スケール = X値1 × 32768 × 補正前スケール　　（小数点以下四捨五入）

Y軸スケール = X値1 × 32768 × 補正前スケール　　（小数点以下四捨五入）

各補正パラメータの計算結果を表2－6に示します。

表2－6　補正パラメータ計算結果

名称	計算結果	パラメータ（16進）
X軸オフセット	1171.8656（＝－（－9.1552）×128）	494H
Y軸オフセット	1213.17248（＝－（－9.47791）×128）	4BDH
X軸スケール	1156.644864（＝0.035298 × 32768 × 1）	485H
Y軸スケール	659.78368（＝0.04027 × 32768 × 0.5）	294H

補正パラメータを設定するコマンドは以下になります。パラメータ値はリトル・エンディアンになっています。

```
$gW'3 00 9404 BD04 8504 9402
```

補正パラメータは測定回数や測定対象を増やすことで精度を上げることができます。

2. 5. 5　補正パラメータ検証

2. 5. 4 で得られた補正パラメータを用いて補正パラメータの検証を行います。

次のコマンドにて補正パラメータの設定します。

$gW'3 00 9404 BD04 8504 9402

LCDモジュールに表示されている格子の交点をタッチすると，格子点の表示ドット位置とほぼ同一の座標値が通知されます。測定例を表2－7に示します。

表2－7　補正パラメータを設定して取得した座標値

	A	B	C	D
1	X	x	Y	y
2	8	8	8	7
3	8	8	8	7
4	8	8	8	7
5	8	8	8	7
6	24	23	8	7
7	24	23	8	7
8	24	24	8	7
9	24	24	8	7
10	40	39	24	23
11	40	39	24	23
12	40	40	24	24
13	40	39	24	24
14	56	55	24	24
15	56	55	24	24
16	56	56	24	24
17	56	56	24	23
18	72	71	40	39
19	72	71	40	39
20	72	71	40	39
21	72	71	40	39
22	88	87	40	40
23	88	87	40	39
24	88	87	40	40
25	88	87	40	40
26	104	103	56	55
27	104	103	56	55
28	104	103	56	56
29	104	103	56	55
30	120	118	56	56
31	120	119	56	56
32	120	119	56	55
33	120	118	56	56

X：タッチしたX軸の座標（表示されている格子のドット位置）

x：タッチにより得られたX軸値

Y：タッチしたY軸の座標（表示されている格子のドット位置）

y：タッチにより得られたY軸値

第3章　アプリケーション実装例

この章では，タッチスクリーン・サンプル・プログラムTSC_c.c**について説明します。**
TSC_c.c**はサンプル・プログラムのダウンロード・ファイルに含まれています。**

3. 1　アプリケーションの概要

3. 1. 1　機能概要

サンプル・プログラムTSC_c.c**は，タッチスクリーン上の押された位置を取得するプログラムです。**

機能概要を図3―1**および図**3―2**に示します。なお，アプリケーション全体としては簡易**OS**や格子を**LCD**に表示するプログラムが含まれていますが，**TSC_c.c**以外の説明は省略します。液晶表示制御部分（**LCD_c.c**）については 「**78K0サンプル・プログラム　ドット**LCD**制御編（**U19531**）」を参照してください。**

図3―1　**機能概要図（内蔵**A/D**コンバータを使用する場合）**

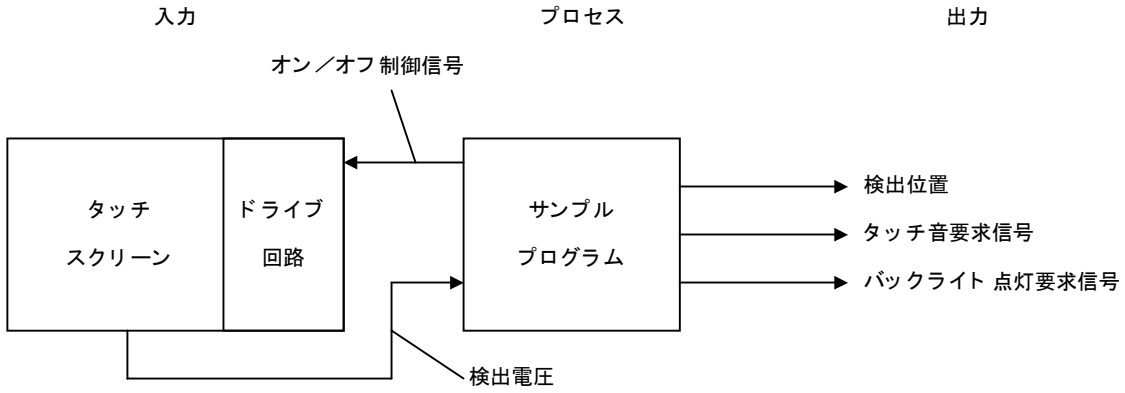
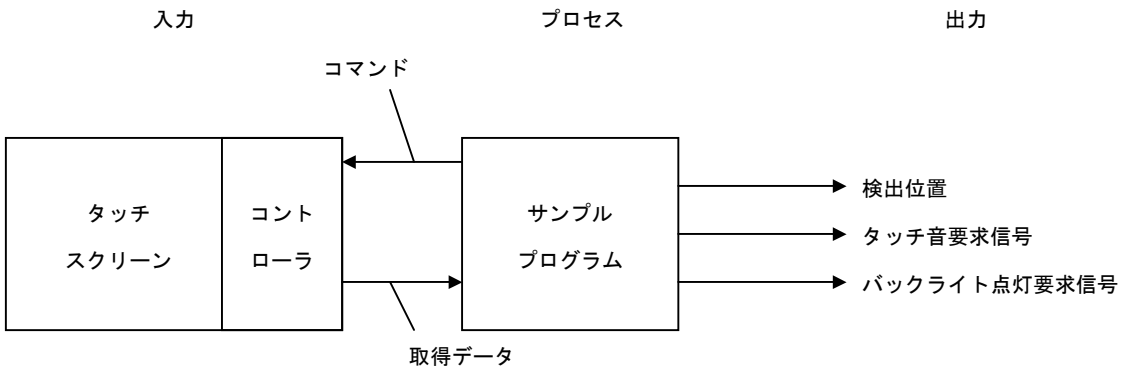


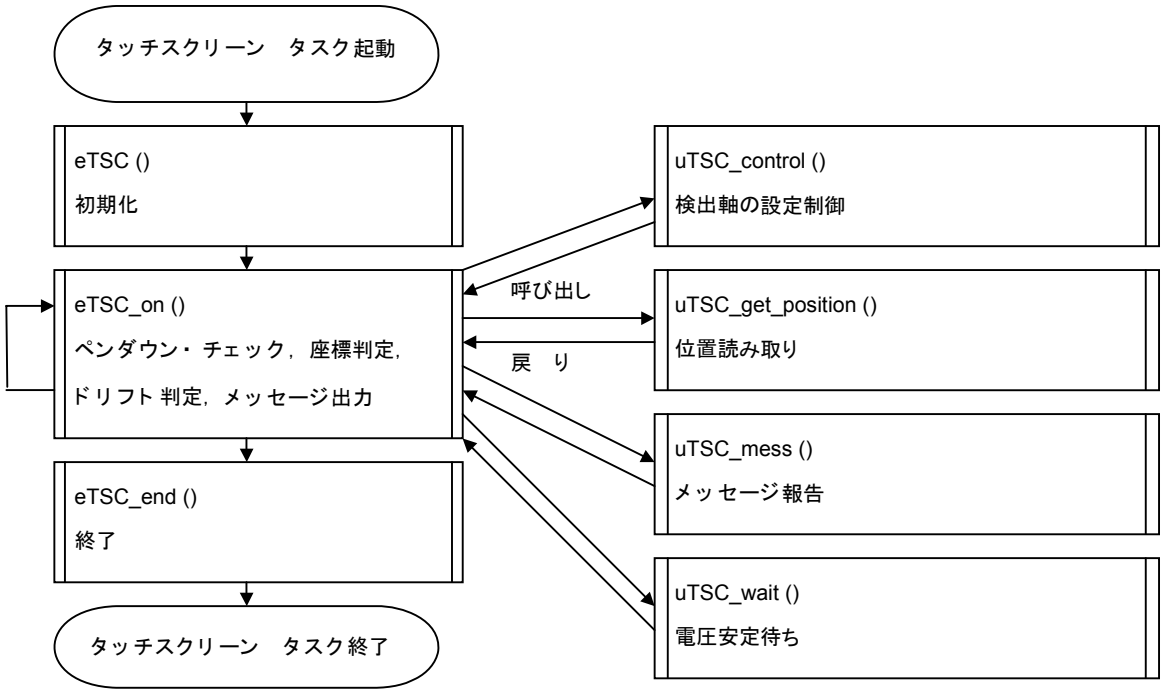
図3―2　**機能概要図（専用コントローラを使用する場合）**



3. 1. 2　制御手順の概要

TSC_c.c中の主要な関数の流れを図3－3に示します。本ソース・プログラムには，簡易OS依存部が多少含まれますが，簡易OSを使わない場合の修正方法がコメントに記載されています。簡易OSを使わない場合，eTSC_on は10 ms周期で呼び出すことにより動作します。

図3－3　全体フローチャート



3. 1. 3　変数一覧

タッチスクリーン・サンプル・プログラムで使用する変数・定数・端子定義を表3－1，表3－2，表3－3に示します。

表3－1　タッチスクリーン　パラメーター一覧　（1/2）

定義場所	定義方法	型	変数名	意味	
TSC_c.h	構造体	unsigned char	gTSC.req	ホスト・コマンド用	ペンリクエスト有無
		unsigned char	gTSC.cmd		コマンド文字
		unsigned char	gTSC.num		引数の長さ
		unsigned char	gTSC.emuxH		ペンエミュレーションX軸値
		unsigned char	gTSC.emuxL		ペンエミュレーションX軸値
		unsigned char	gTSC.emuyH		ペンエミュレーションY軸値
		unsigned char	gTSC.emuyL		ペンエミュレーションY軸値
		unsigned char	gTSC.cmp	パラメータ設定用	一致時間（10ms単位）
		unsigned char	gTSC.xyc		X軸Y軸の交換指定
		unsigned char	gTSC.mess1		メッセージ出力有無
		unsigned char	gTSC.xp		X軸極性
		unsigned char	gTSC.yp		Y軸極性
		unsigned char	gTSC.local		ローカル・コード生成指定
		unsigned char	gTSC.BLon		バックライト点灯指定
		unsigned char	gTSC.click		タッチ音生成指定
		unsigned char	gTSC.off		ペンアップ・コード生成指定
		unsigned char	gTSC.mess_uid		通知先設定
		unsigned char	gTSC.drift		ドリフト許容範囲
		unsigned short	gTSC.basex		X軸オフセット
		unsigned short	gTSC.basey		Y軸オフセット
		unsigned short	gTSC.pitchx		X軸スケール
		unsigned short	gTSC.pitchy		Y軸スケール
		short	gTSC.codex	検出座標出力	ローカル通知(X)
		short	gTSC.codey		ローカル通知(Y)

表3－1　タッチスクリーン　パラメーター覧　(2/2)

TSC_c.h		short	gTSC.lastx	内部制御用	前回確定値（X）
		short	gTSC.lasty		前回確定値（Y）
	#define		kTSC_off	ペンアップ・コード	
	#define		kTSC_no	ペン情報無し	
UDEF_func_c.h	#define		Task_Melody	メロディ出力タスク有無	
	#define		Task_LCD	表示データをLCDへ転送するタスク有無	
TSC_c.c	#define		uTSC_wait_clock	タッチスクリーン・コントローラへのクロック速度調整	
	#define		uTSC_wait_data	タッチスクリーン・コントローラへのデータ速度調整	
	#define		kADM	内蔵A/D変換速度指定	
	#define		kTSC_WAIT	安定待ち時間（μs単位）	
	#define		kTSC_WAITC	安定待ち時間（カウント単位）	
		unsigned char	tcTSC_cnt	一致回数カウンタ	
		short	tjTSC_prevx	直前検出値（X）	
		short	tjTSC_prevy	直前検出値（Y）	
	#define		TSC_ON	タッチスクリーン・コントローラへのコマンド（動作ON）	
	#define		X_POSITION	タッチスクリーン・コントローラへのコマンド（X軸設定）	
	#define		Y_POSITION	タッチスクリーン・コントローラへのコマンド（Y軸設定）	
	#define		TSC_OFF	タッチスクリーン・コントローラへのコマンド（動作OFF）	
	#define		ALREADY_DETECTED	検出済み	
		short	tjTSC_hold_onx	メッセージ報告用テンポラリ変数(X)	
		short	tjTSC_hold_ony	メッセージ報告用テンポラリ変数(Y)	
		boolean	tbTSC_hold_on1	メッセージ報告有無	
	構造体	const unsigned short	tTSC	タッチスクリーン・プログラム初期設定値	

★ **表3－2　端子および特殊機能レジスタ（SFR）にかかわる定義（内蔵A/Dコンバータ方式）**

定義場所	定義方法	型	変数名	意味
UDEF_port_c.h	#define		zSWXP	X＋端子制御
	#define		zSWXM	X－端子制御
	#define		zSWYP	Y＋端子制御
	#define		zSWYM	Y－端子制御
	#define		zPEN_INQ	ペンダウン制御用プルアップ
	#define		ANXP_ch	X＋値読み取り，ペンダウン検出
	#define		ANXM_ch	X－値読み取り
	#define		ANYP_ch	Y＋値読み取り
	#define		ANYM_ch	Y－値読み取り
UDEF_port_c.h	#define		yBTIMER_TM	基本周期タイマ・カウント値
UDEF_BOARD_NUMBER_c.h	#define		kBTIMER	基本周期タイマの周期

★ **表3－3　端子および特殊機能レジスタ（SFR）にかかわる定義（専用コントローラ方式）**

定義場所	定義方法	型	変数名	意味
UDEF_port_c.h	#define		zCSI_SPC	クロック
	#define		zCSI_SPO	コマンド入力
	#define		zCSI_SPI	データ出力
	#define		zCSI_CS2	チップセレクト
	#define		zPEN_INQ	ペンダウン検出
UDEF_port_c.h	#define		yBTIMER_TM	基本周期タイマ・カウント値
UDEF_BOARD_NUMBER_c.h	#define		kBTIMER	基本周期タイマの周期

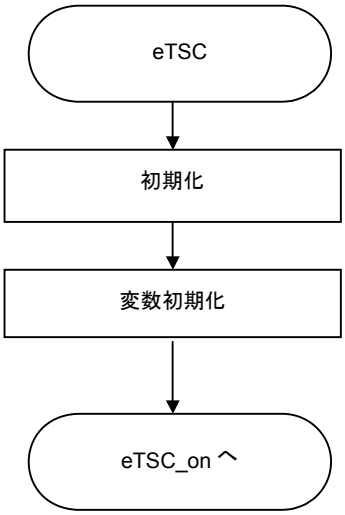
3. 2　タッチスクリーン初期化（eTSC）

eTSC関数は，タッチスクリーンで使用するポートの初期化，A/Dコンバータまたはタッチスクリーン・コントローラの初期化，変数の初期化を行います。

（1）概略フロー

eTSC関数の流れを図3－4に示します。

図3－4　eTSC関数フローチャート



★　(2) eTSC関数プログラム・リスト

① 内蔵A/Dコンバータ方式のリスト

内蔵A/Dコンバータを使用する場合のeTSC関数のプログラム・リストを図3－5に示します。

図3－5　eTSC関数プログラム・リスト（内蔵A/Dコンバータ方式）

```
/************************************************************************
 ************************************************************************
 初期設定
 ************************************************************************
 ************************************************************************/
void eTSC(void) {

    /* コントローラ初期設定 */
    uTSC_control(TSC_OFF);

    /* 変数初期化 */
    gTSC.codex = kTSC_no;
    gTSC.codey = kTSC_no;
    gTSC.lastx = kTSC_no;
    gTSC.lasty = kTSC_no;

    tcTSC_cnt = 0;
    tjTSC_prevx = kTSC_no;
    tjTSC_prevy = kTSC_no;
    tbTSC_hold_on1 = 0;

    vActive_task(hTSC);                 /* 簡易OSを使用しない場合は削除 */
    vTrans_task(eTSC_on);               /* 簡易OSを使用しない場合は，return文に変更 */
}
```

① 初期化

② 変数初期化

③ eTSC_onへ

② 専用コントローラ方式のリスト

専用コントローラを使用する場合のeTSC関数のプログラム・リストを図3－6に示します。

図3－6　eTSC関数プログラム・リスト（専用コントローラ方式）

```
/************************************************************************
 ************************************************************************
 初期設定
 ************************************************************************
 ************************************************************************/
void eTSC(void) {
    /* チップセレクト初期設定 */
    zCSI_CS2 = 1;
    zmCSI_CS2 = 0;

    /* コントローラ初期設定 */
    if (!vPolling_sema(hsSPI)) return;      /* シリアル・ポートの共有制御が不要なら削除 */

    /* シリアル・ポート端子設定 */
    zCSI_SPC = 0;
    zmCSI_SPC = 0;
    zCSI_SPO = 0;
    zmCSI_SPO = 0;

    /* コントローラ初期化（ペンダウン検出待ちに設定） */
    zCSI_CS2 = 0;
    uTSC_control(TSC_OFF);
    zCSI_CS2 = 1;
    vSignal_sema(hsSPI);                    /* シリアル・ポートの共有制御が不要なら削除 */

    /* 変数初期化 */
    gTSC.codex = kTSC_no;
    gTSC.codey = kTSC_no;
    gTSC.lastx = kTSC_no;
    gTSC.lasty = kTSC_no;

    tcTSC_cnt = 0;
    tjTSC_prevx = kTSC_no;
    tjTSC_prevy = kTSC_no;
    tbTSC_hold_on1 = 0;

    vActive_task(hTSC);                 /* 簡易OSを使用しない場合は削除 */
    vTrans_task(eTSC_on);               /* 簡易OSを使用しない場合は，return文に変更 */
}
```

① 初期化

② 変数初期化

③ eTSC_onへ

(3) 詳細説明

eTSC関数の詳細は以下になります。

①　初期化

後述の制御関数uTSC_controlを使用して初期化（ペンダウン検出待ちに設定）を行います。専用コントローラ方式ではシリアル・ポート用の端子設定も行います。

②　変数初期化

タッチスクリーンで使用する変数の初期化を行います。

③　eTSC_onへ

eTSC_onへ移行します。簡易OSを使用しない場合はreturn文に変更し，最初に1回だけ本関数を呼び出します。

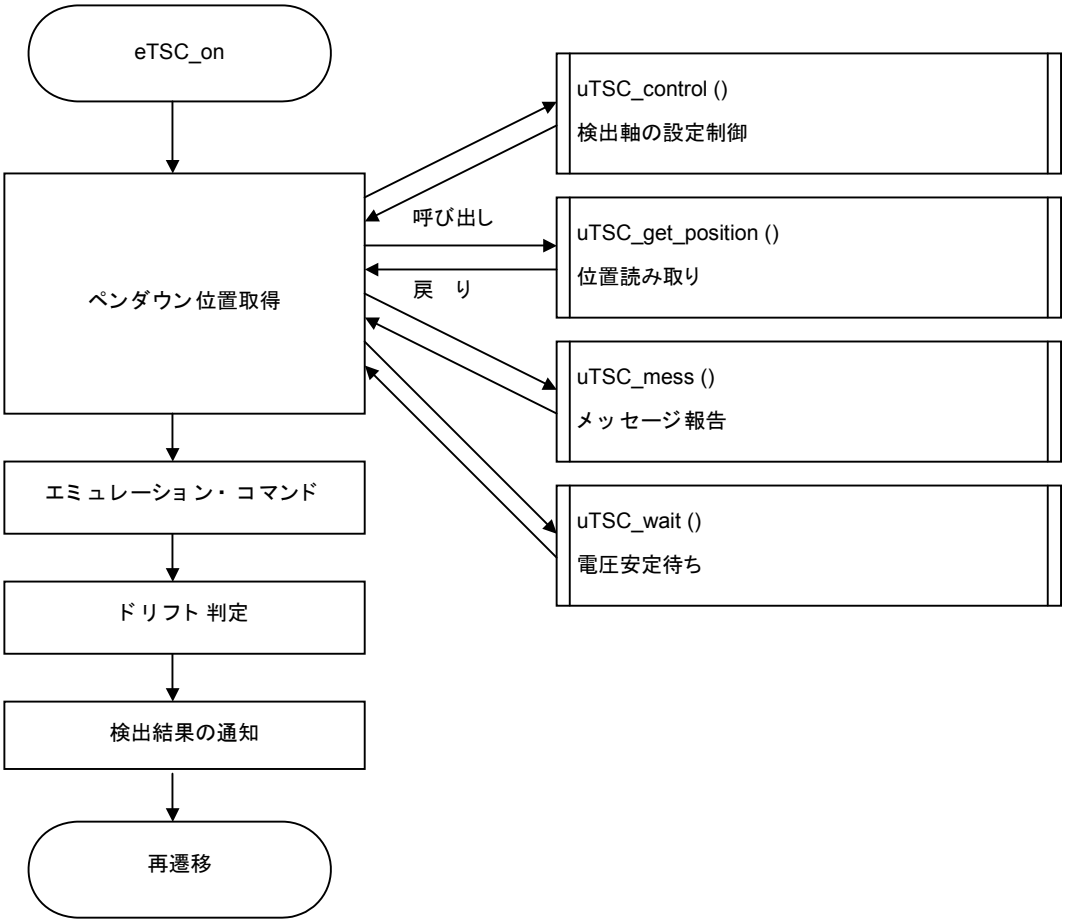
3. 3　ペン位置検出（eTSC_on）

eTSC_on関数は，ペンダウン・チェック，座標判定，ドリフト判定，メッセージ出力を行います。

(1) 概略フロー

eTSC_on関数の流れを図3－7に示します。

図3－7　eTSC_on関数フローチャート

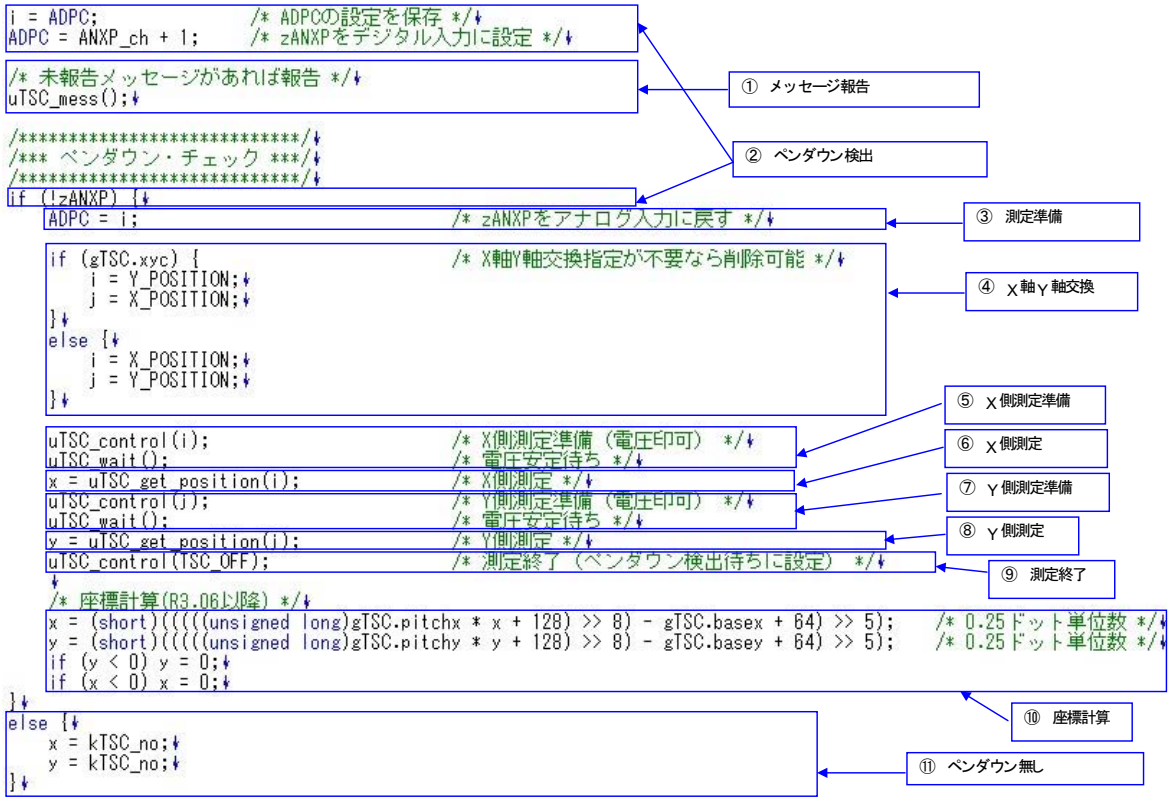


（2）eTSC_on関数プログラム・リスト（ペンダウン位置取得）

★　**①　内蔵A/Dコンバータ方式のリスト（ペンダウン位置取得）**

内蔵A/ Dコンバータを使用する場合のeTSC_on関数プログラム・リストを図3－8に示します。

図3－8　eTSC_on関数プログラム・リスト（内蔵A/ Dコンバータ方式）

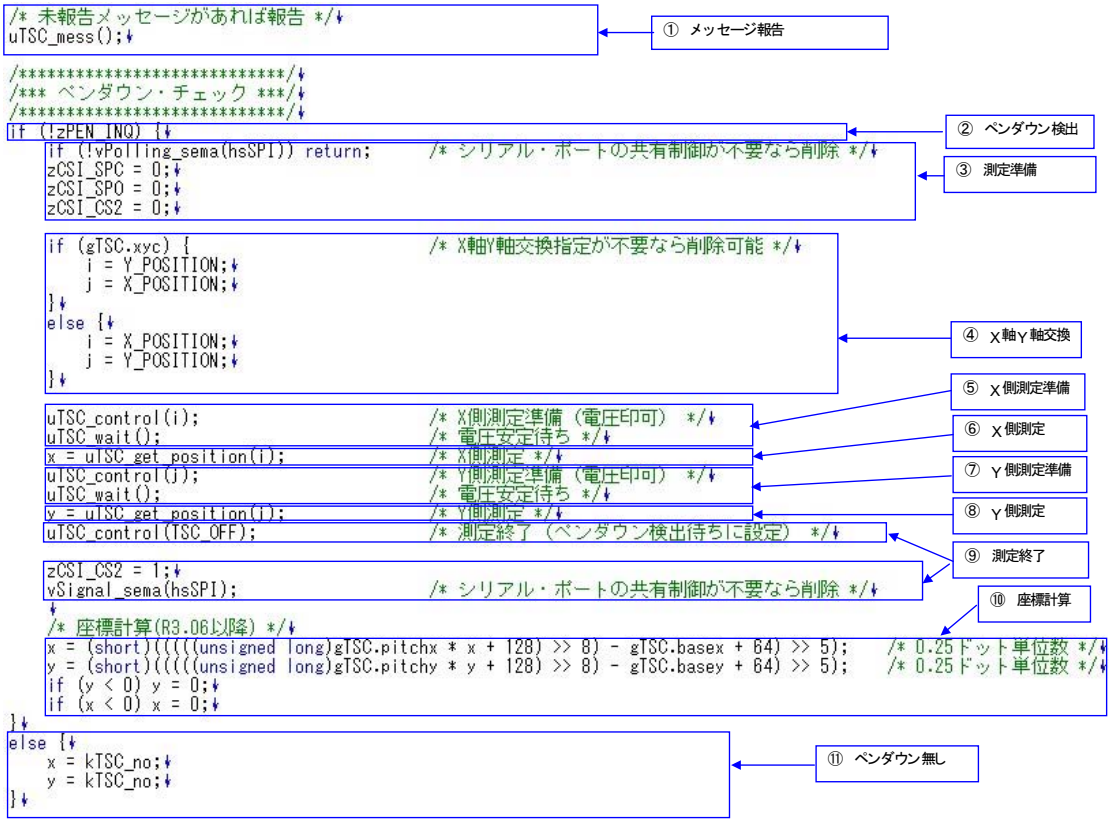



```
i = ADPC;                   /* ADPCの設定を保存 */
ADPC = ANXP_ch + 1;         /* zANXPをデジタル入力に設定 */

/* 未報告メッセージがあれば報告 */
uTSC_mess();

/*****************************/
/*** ペンダウン・チェック ***/
/*****************************/
if (!zANXP) {
    ADPC = i;                           /* zANXPをアナログ入力に戻す */

    if (gTSC.xyc) {                     /* X軸Y軸交換指定が不要なら削除可能 */
        i = Y_POSITION;
        j = X_POSITION;
    }
    else {
        i = X_POSITION;
        j = Y_POSITION;
    }

    uTSC_control(i);                    /* X側測定準備（電圧印可） */
    uTSC_wait();                        /* 電圧安定待ち */
    x = uTSC_get_position(i);           /* X側測定 */
    uTSC_control(j);                    /* Y側測定準備（電圧印可） */
    uTSC_wait();                        /* 電圧安定待ち */
    y = uTSC_get_position(j);           /* Y側測定 */
    uTSC_control(TSC_OFF);              /* 測定終了（ペンダウン検出待ちに設定） */

    /* 座標計算(R3.06以降) */
    x = (short)(((((unsigned long)gTSC.pitchx * x + 128) >> 8) - gTSC.basex + 64) >> 5);    /* 0.25ドット単位数 */
    y = (short)(((((unsigned long)gTSC.pitchy * y + 128) >> 8) - gTSC.basey + 64) >> 5);    /* 0.25ドット単位数 */
    if (y < 0) y = 0;
    if (x < 0) x = 0;
}
else {
    x = kTSC_no;
    y = kTSC_no;
}
```

②　専用コントローラ方式のリスト（ペンダウン位置取得）

専用コントローラを使用する場合のeTSC_on関数プログラム・リストを図3－9　に示します。

図3－9　eTSC_on関数プログラム・リスト（専用コントローラ方式）




```
/* 未報告メッセージがあれば報告 */
uTSC_mess();

/*****************************/
/*** ペンダウン・チェック ***/
/*****************************/
if (!zPEN_INQ) {
    if (!vPolling_sema(hsSPI)) return;      /* シリアル・ポートの共有制御が不要なら削除 */
    zCSI_SPC = 0;
    zCSI_SPO = 0;
    zCSI_CS2 = 0;

    if (gTSC.xyc) {                         /* X軸Y軸交換指定が不要なら削除可能 */
        i = Y_POSITION;
        j = X_POSITION;
    }
    else {
        i = X_POSITION;
        j = Y_POSITION;
    }

    uTSC_control(i);                        /* X側測定準備（電圧印可） */
    uTSC_wait();                            /* 電圧安定待ち */
    x = uTSC_get_position(i);               /* X側測定 */
    uTSC_control(j);                        /* Y側測定準備（電圧印可） */
    uTSC_wait();                            /* 電圧安定待ち */
    y = uTSC_get_position(j);               /* Y側測定 */
    uTSC_control(TSC_OFF);                  /* 測定終了（ペンダウン検出待ちに設定） */

    zCSI_CS2 = 1;
    vSignal_sema(hsSPI);                    /* シリアル・ポートの共有制御が不要なら削除 */

    /* 座標計算(R3.06以降) */
    x = (short)(((((unsigned long)gTSC.pitchx * x + 128) >> 8) - gTSC.basex + 64) >> 5);    /* 0.25ドット単位数 */
    y = (short)(((((unsigned long)gTSC.pitchy * y + 128) >> 8) - gTSC.basey + 64) >> 5);    /* 0.25ドット単位数 */
    if (y < 0) y = 0;
    if (x < 0) x = 0;
}
else {
    x = kTSC_no;
    y = kTSC_no;
}
```

(3) 詳細説明（ペンダウン位置取得）

eTSC_on関数，ペンダウン位置取得部プログラムの詳細は以下になります。

①　メッセージ報告

検出済みの座標値をホスト・マシンに報告する場合はuTSC_mess関数の呼び出しを行います。この機能が不要であれば削除可能です。

②　ペンダウン検出

内蔵A/Dコンバータ方式では，zANXP端子をデジタル・ポートに設定した上で，zANXP端子のレベルを判定します。専用コントローラ方式では，zPEN_INQ端子のレベルを判定します。

③　測定準備

内蔵A/Dコンバータ方式では，zANXP端子をアナログ・ポートに戻します。

専用コントローラ方式では，シリアル・ポートの設定をします。

④　X軸Y軸交換

X軸Y軸の軸設定を交換します。

⑤　X側測定準備

X側測定準備のために制御端子に電圧印加を行い，端子電圧が安定するまで待ちます。

⑥　X側測定

X側測定を行います。

⑦　Y側測定準備

Y側測定準備のために制御端子に電圧印加を行い，端子電圧が安定するまで待ちます。

⑧　Y側測定

Y側測定を行います。

⑨　測定終了

ペンダウン検出待ちに戻します。専用コントローラ方式では，シリアル・ポートの開放を行います。

⑩　座標計算

補正パラメータを使い，取得した値の座標補正を行います。

⑪　ペンダウン無し

ペン・オフ・コードを設定します。

(4) eTSC_on関数（エミュレーション・コマンド）

eTSC_on関数内，エミュレーション・コマンド部プログラム・リストを図3－10に示します。

図3－10　eTSC_on関数プログラム・リスト（エミュレーション・コマンド）

```
/****************/
/*** 座標判定 ***/
/****************/
if ( x == kTSC_no && gTSC.lastx == kTSC_no) {
    /*** 直前値クリア ***/
    tjTSC_prevx = kTSC_no;
    tjTSC_prevy = kTSC_no;
    tcTSC_cnt = 0;

    /*** ペン・エミュレーション（ホスト・マシンからのコマンドが不要ならば，このifブロックを削除） ***/
    if (gTSC.cmd) {
        if (p=uUDEF_get_REG(gTSC.req)) {
            /* コマンド解釈実行 */
            if (gTSC.cmd != 'E')    i = 'c';
            else if (gTSC.num < 4) i = 'p';
            else {
                if (gTSC.local) {
                    gTSC.codex = ((short)gTSC.emuxH << 8) + gTSC.emuxL;
                    gTSC.codey = ((short)gTSC.emuyH << 8) + gTSC.emuyL;
                }
                i = 'o';
            }

            /* 応答 */
            if (p != UDEF_REQ_ERROR) {
                *p++ = (unsigned char) hTSC;
                *p++ = gTSC.cmd;
                *p++ = 1;
                *p++ = i;
            }

            gTSC.cmd = 0;
        }
    }
}
```

① ペン無しかつ前回情報無し
② ペン直線値クリア
③ コマンド有り
④ ペン・リクエスト有り
⑤ コマンド文正常
⑥ 値取り込み
⑦ リクエストエラー無し
⑧ 応答出力
⑨ コマンドモード

eTSC_on関数内，エミュレーション・コマンド部の詳細は以下になります。

① ペン無しかつ前回情報無し

ペン・オフ・コード情報と前回情報が無いか，判定を行います。

② ペン直前値クリア

直前のペン値をクリアします。

③ コマンド有り

ホスト･マシンからのペン・エミュレーション･コマンドが要求されたか判定を行います。ホスト・マシンからのコマンド機能が不要であれば，このifブロック部分は削除可能です。

④ ペン・リクエスト有り

ペンがリクエストされたか判定を行います。

⑤ コマンド文正常

コマンド内容が正しいか判定を行い，エラーの場合，報告を行います。

⑥ 値取り込み

コマンドの内容の取り込みを行います。

⑦ リクエストエラー無し

リクエストにエラーが無いか判定を行います。

⑧ 応答出力

応答メッセージの出力を行います。

⑨ コマンドモード

ペン・エミュレーション・コマンドをクリアし，再遷移します。

(5) eTSC_on関数（ドリフト判定）

eTSC_on関数内，ドリフト判定部プログラム・リストを図3－11に示します。

図3－11　eTSC_on関数プログラム・リスト（ドリフト判定）

```
else {
    i = 0;
    if (abs(tjTSC_prevx - x) < (short)gTSC.drift) i++;
    if (abs(tjTSC_prevy - y) < (short)gTSC.drift) i++;
    if (i != 2) {
        if (abs(gTSC.lastx - x) < (short)gTSC.drift &&
            abs(gTSC.lasty - y) < (short)gTSC.drift    ) {
            /*** 検出済継続 ***/
            tjTSC_prevx = gTSC.lastx;
            tjTSC_prevy = gTSC.lasty;
            tcTSC_cnt = ALREADY_DETECTED;
        }
        else {
            /*** 異なる値検出 ***/
            tjTSC_prevx = x;
            tjTSC_prevy = y;
            tcTSC_cnt = 0;
        }
    }
    else {
        if (tcTSC_cnt <= gTSC.cmp)  ++tcTSC_cnt;
        else if (tcTSC_cnt != ALREADY_DETECTED) {
            /*-------------------------*/
            /* 検出結果出力            */
            /*-------------------------*/
            tcTSC_cnt = ALREADY_DETECTED;

            /*** 位置出力 ***/
            gTSC.lastx = x;
            gTSC.lasty = y;
```

① ドリフト 範囲内
② 前回値と異なる値
③ ドリフト 範囲内
④ 前回の確定値コピー
⑤ 今回の値使用
⑥ 一致時間以下
⑦ 時間経過
⑧ 検出値確定

eTSC_on関数内，ドリフト判定部の詳細は以下になります。

① ドリフト範囲内

取得した値がドリフト範囲内にあるか判定を行い，範囲内の場合は判定用カウンタをインクリメントします。

② 前回値と異なる値

取得した値と前回値が異なるか，判定を行います。

③ ドリフト範囲内

取得した値がドリフト範囲内にあるか判定を行います。

④ 前回の確定値コピー

前回の確定値に設定します。

⑤ 今回の値使用

今回の取得した値を設定します。

⑥ 一致時間以下

パラメータで設定した一致時間以下であれば，一致回数カウンタをインクリメントします。

⑦ 時間経過

確定した値がすでに出力済かどうか判定します。

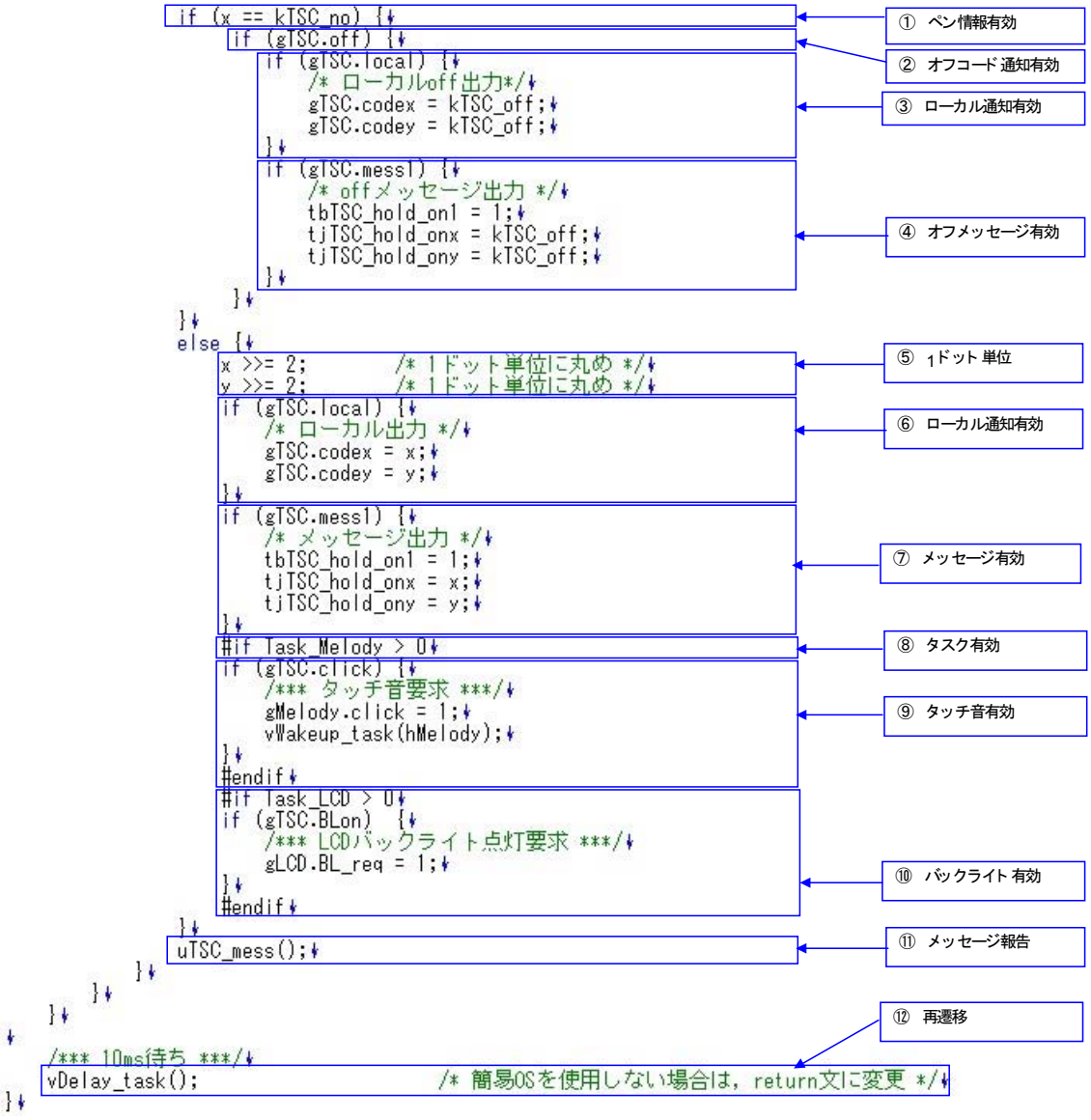
⑧ 検出値確定

今回検出した値を確定します。

(6) eTSC_on関数プログラム・リスト（検出結果の通知）

eTSC_on関数内，検出結果の通知部プログラム・リストを図3－12に示します。

図3－12　eTSC_on関数プログラム・リスト（検出結果の通知）




```
if (x == kTSC_no) {
    if (gTSC.off) {
        if (gTSC.local) {
            /* ローカルoff出力*/
            gTSC.codex = kTSC_off;
            gTSC.codey = kTSC_off;
        }
        if (gTSC.mess1) {
            /* offメッセージ出力 */
            tbTSC_hold_on1 = 1;
            tjTSC_hold_onx = kTSC_off;
            tjTSC_hold_ony = kTSC_off;
        }
    }
}
else {
    x >>= 2;        /* 1ドット単位に丸め */
    y >>= 2;        /* 1ドット単位に丸め */
    if (gTSC.local) {
        /* ローカル出力 */
        gTSC.codex = x;
        gTSC.codey = y;
    }
    if (gTSC.mess1) {
        /* メッセージ出力 */
        tbTSC_hold_on1 = 1;
        tjTSC_hold_onx = x;
        tjTSC_hold_ony = y;
    }
    #if Task_Melody > 0
    if (gTSC.click) {
        /*** タッチ音要求 ***/
        gMelody.click = 1;
        vWakeup_task(hMelody);
    }
    #endif
    #if Task_LCD > 0
    if (gTSC.BLon)  {
        /*** LCDバックライト点灯要求 ***/
        gLCD.BL_req = 1;
    }
    #endif
    }
    uTSC_mess();
    }
    }
}

/*** 10ms待ち ***/
vDelay_task();                  /* 簡易OSを使用しない場合は，return文に変更 */
}
```

eTSC_on関数内，検出結果の通知部の詳細は以下になります。

① ペン情報有効

ペンダウン座標確定またはペンアップ確定の情報があるか判定を行います。

② オフコード通知有効

パラメータ設定でオフコード通知が指定されている場合オフコード通知を行います。

③ ローカル通知有効

パラメータ設定で他のタスクに対するローカル通知が指定されている場合にオフコード通知を行います。

④ オフメッセージ有効

パラメータ設定でホスト・マシンへのメッセージ通知が指定されている場合にオフメッセージ通知の設定を行います。

⑤ 1ドット単位

取得した値を1ドット単位に丸めます。

⑥ ローカル通知有効

パラメータ設定で他のタスクに対するローカル通知が指定されている場合に取得した座標を通知します。

⑦ メッセージ有効

パラメータ設定でホスト・マシンへのメッセージ通知が指定されている場合にメッセージ通知の設定を行います。

⑧ タスク有効

Melodyユニットが実装されているか判定を行います。

⑨ タッチ音有効

Melodyユニットが実装されている場合，タッチ音要求を行います。

⑩ バックライト有効

LCDユニットが実装されている場合，LCDバックライト要求を行います。

⑪ メッセージ報告

uTSC_mess関数の呼び出しを行い，メッセージ報告を行います。

⑫ 再遷移

10 ms後にeTSC_onタスクに再遷移します。簡易OSを使用しない場合はreturn文に変更し，10 ms周期でeTSC_on関数を呼び出してください。

★ 3. 4　検出軸の設定制御（uTSC_control）

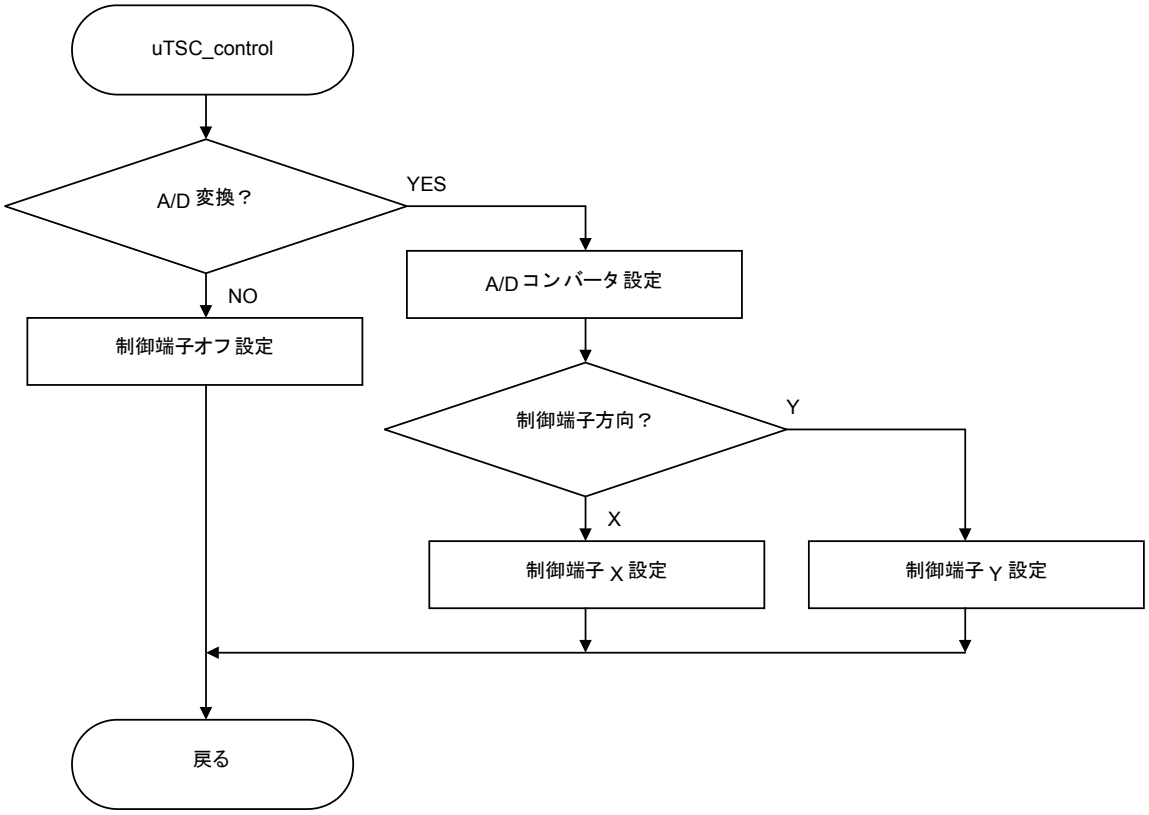
uTSC_control関数は，位置読み取りに必要な制御を行います。制御方法は内蔵A/Dコントローラを使用する場合と専用コントローラを使用する場合で異なります。

3. 4. 1　内蔵A/Dコンバータ方式の軸設定

（1）概略フロー

内蔵A/ Dコンバータを使用する場合のuTSC_control関数の流れを図3－13に示します。

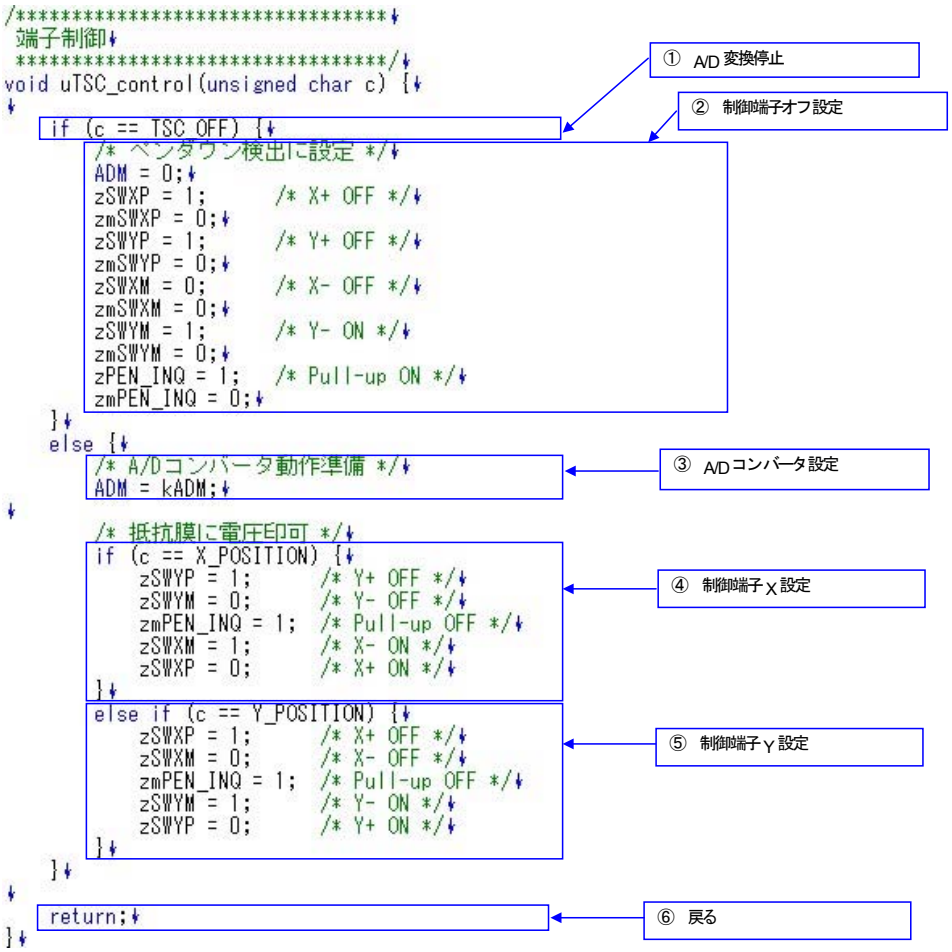
図3－13　uTSC_control関数フローチャート（内蔵A/ Dコンバータ方式）



(2) プログラム・リスト

内蔵A/Dコンバータを使用する場合のuTSC_control関数のプログラム・リストを図3－14に示します。

図3－14　uTSC_control関数プログラム・リスト（内蔵A/Dコンバータ方式）




```
/*********************************
 端子制御
 *********************************/
void uTSC_control(unsigned char c) {

    if (c == TSC_OFF) {
        /* ペンダウン検出に設定 */
        ADM = 0;
        zSWXP = 1;      /* X+ OFF */
        zmSWXP = 0;
        zSWYP = 1;      /* Y+ OFF */
        zmSWYP = 0;
        zSWXM = 0;      /* X- OFF */
        zmSWXM = 0;
        zSWYM = 1;      /* Y- ON */
        zmSWYM = 0;
        zPEN_INQ = 1;   /* Pull-up ON */
        zmPEN_INQ = 0;
    }
    else {
        /* A/Dコンバータ動作準備 */
        ADM = kADM;

        /* 抵抗膜に電圧印可 */
        if (c == X_POSITION) {
            zSWYP = 1;      /* Y+ OFF */
            zSWYM = 0;      /* Y- OFF */
            zmPEN_INQ = 1;  /* Pull-up OFF */
            zSWXM = 1;      /* X- ON */
            zSWXP = 0;      /* X+ ON */
        }
        else if (c == Y_POSITION) {
            zSWXP = 1;      /* X+ OFF */
            zSWXM = 0;      /* X- OFF */
            zmPEN_INQ = 1;  /* Pull-up OFF */
            zSWYM = 1;      /* Y- ON */
            zSWYP = 0;      /* Y+ ON */
        }
    }

    return;
}
```

(3) 詳細説明

内蔵A/Dコンバータを使用する場合のuTSC_control関数の詳細は以下になります。

① A/D変換停止

A/D変換を停止するか開始するか判定します。

② 制御端子オフ設定

A/D変換を停止する場合の，制御端子を設定します。

③ A/Dコンバータ設定

A/D変換速度などの設定を行います。

④ 制御端子X設定

X値を読み取るために制御端子を設定します。

⑤ 制御端子Y設定

Y値を読み取るために制御端子を設定します。

⑥ 戻る

内蔵A/Dに必要な制御を終了します。

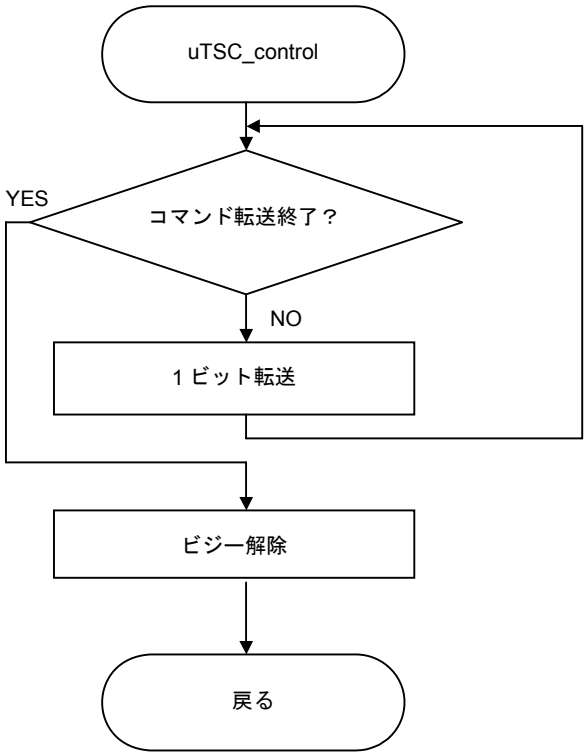
3. 4. 2　専用コントローラ方式の軸設定

専用コントローラを使用する場合，ポート制御によりシリアル・データ転送を行っています。

(1) 概略フロー

専用コントローラを使用する場合のuTSC_control関数の流れを図3－15に示します。

図3－15　uTSC_control関数フローチャート（専用コントローラ方式）



(2) プログラム・リスト

専用コントローラを使用する場合のuTSC_control関数のプログラム・リストを図3－16に示します。

図3－16　uTSC_control関数プログラム・リスト（専用コントローラ方式）

```
/*********************************
 コントローラ制御
 *********************************/
void uTSC_control(unsigned char c) {
    char i;

    /* コマンド転送 */
    for (i=0; i<8; i++) {
        if (c & 0x80)   zCSI_SPO = 1;
        else            zCSI_SPO = 0;
        c <<= 1;
        uTSC_wait_data;
        zCSI_SPC = 1;
        uTSC_wait_clock;
        zCSI_SPC = 0;
    }

    /* ビジー解除 */
    zCSI_SPO = 0;
    uTSC_wait_clock;
    zCSI_SPC = 1;
    uTSC_wait_clock;
    zCSI_SPC = 0;

    return;
}
```

① コマンド転送終了

② 1ビット転送

③ ビジー解除

④ 戻る

(3) 詳細説明

専用コントローラを使用する場合のuTSC_control関数の詳細は以下になります。

① コマンド転送終了

タッチスクリーン・コントローラへのコマンド転送数をカウントします。

② 1ビット転送

タッチスクリーン・コントローラへ1ビット転送します。

③ ビジー解除

タッチスクリーン・コントローラのビジー状態を解除します。

④ 戻る

タッチスクリーン・コントローラへのコマンド転送を終了します。

★ 3. 5　位置読み取り（uTSC_get_position）

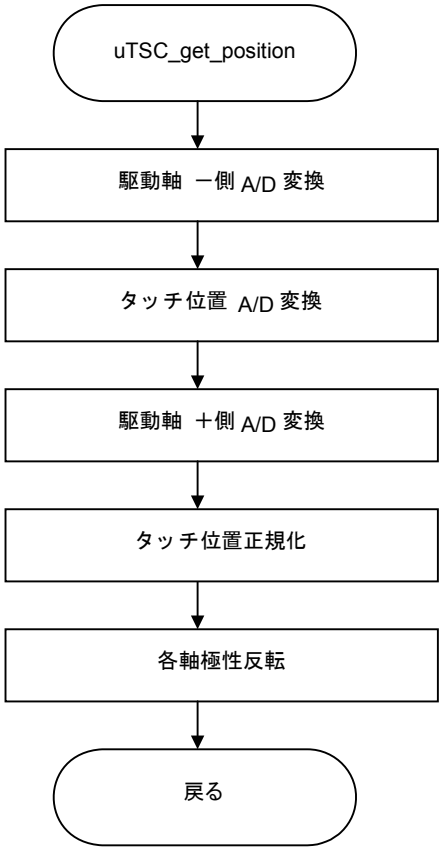
uTSC_get_position関数は，位置読み取りを行います。読み取り方法は内蔵A/Dコンバータを使用する場合と専用コントローラを使用する場合で異なります。

3. 5. 1　内蔵A/Dコンバータ方式の位置読み取り

（1）概略フロー

内蔵A/Dコンバータを使用する場合のuTSC_get_position関数の流れを図3－17に示します。

図3－17　uTSC_get_position関数フローチャート（内蔵A/Dコンバータ方式）



(2) プログラム・リスト

内蔵A/Dコンバータを使用する場合のuTSC_get_position関数のプログラム・リストを図3－18に示します。

図3－18　uTSC_get_position関数プログラム・リスト（内蔵A/Dコンバータ方式）

```
/*********************************
 位置読み取り
 *********************************/
unsigned short uTSC_get_position(unsigned char c) {
    unsigned short x, y, z;
    char  i;

    /*** 駆動軸 -側A/D ***/
    if (c == X_POSITION) ADS = ANXM_ch;
    else                 ADS = ANYM_ch;
    ADIF=0;
    ADCS=1;
    while (!ADIF);
    x = ADCR;

    /*** タッチ位置A/D ***/
    if (c == X_POSITION) ADS = ANYP_ch;
    else                 ADS = ANXP_ch;
    ADIF=0;
    while (!ADIF);
    y = ADCR;
    if (y < x)  y = 0;
    else        y -= x;

    /*** 駆動軸 +側A/D ***/
    if (c == X_POSITION) ADS = ANXP_ch;
    else                 ADS = ANYP_ch;
    ADIF=0;
    while (!ADIF);
    z = ADCR - x;

    /*** タッチ位置を0～FFFH範囲に正規化 ***/
    x = (unsigned short)( (((unsigned long)y << 16) / z + 8) >> 4 );
    if (x > 0x0fff) x = 0x0fff;

    /*** 各軸の極性反転機能 ***/
    if (c == X_POSITION && gTSC.xp || c == Y_POSITION && gTSC.yp) x ^= 0x0fff;

    return(x);
}
```

① 駆動軸－側A/D変換

② タッチ位置A/D変換

③ 駆動軸＋側A/D変換

④ タッチ位置正規化

⑤ 各軸極性反転

⑥ 戻る

(3) 詳細説明

内蔵A/Dコンバータを使用する場合のuTSC_get_position関数の詳細は以下になります。

① **駆動軸－側A/D変換**

A/Dチャンネルの指定と駆動軸－側のA/D変換を行います。

② **タッチ位置A/D変換**

A/Dチャンネルの指定とタッチ位置のA/D変換を行います。

③ **駆動軸＋側A/D変換**

A/Dチャンネルの指定と駆動軸＋側のA/D変換を行います。

④ **タッチ位置正規化**

取得した位置を0～100%（FFFH）に換算します。

⑤ **各軸極性反転**

各軸の極性反転が設定されている場合，値を反転します。

⑥ **戻る**

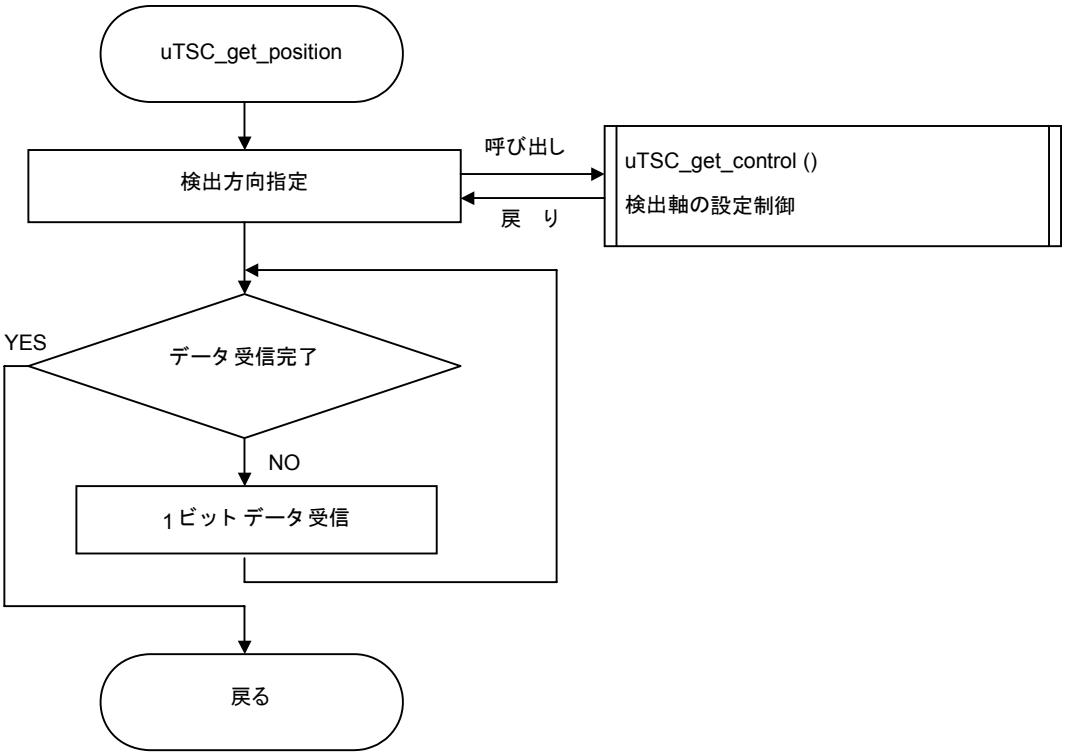
位置読み取りを終了します。

3. 5. 2　専用コントローラ方式の位置読み取り

(1) 概略フロー

専用コントローラを使用する場合のuTSC_get_position関数の流れを図3－19に示します。

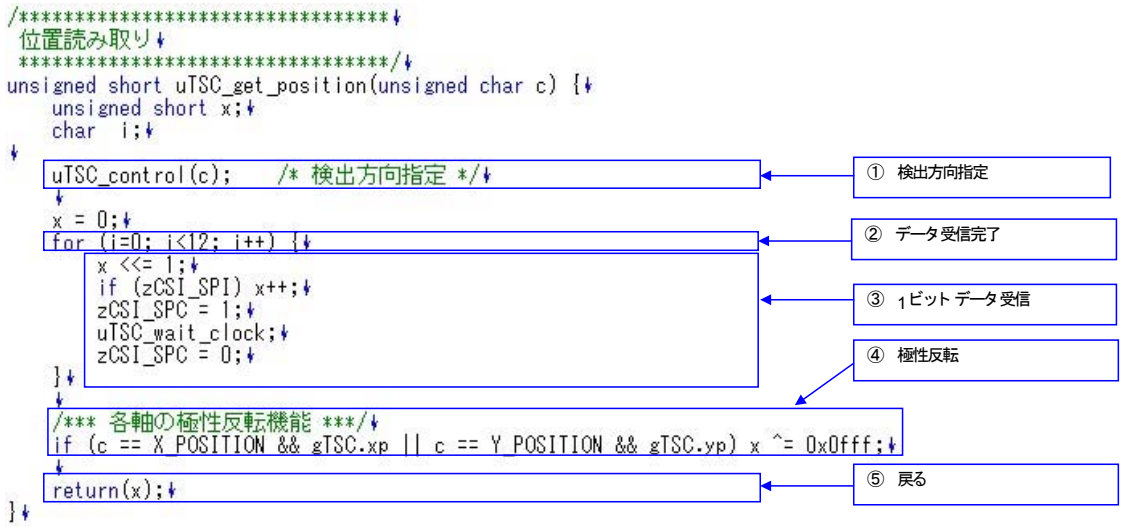
図3－19　uTSC_get_position関数フローチャート（専用コントローラ方式）



(2) プログラム・リスト

専用コントローラを使用する場合のuTSC_get_position関数のプログラム・リストを図3－20に示します。

図3－20　uTSC_get_position関数プログラム・リスト（専用コントローラ方式）



(3) 詳細説明

専用コントローラを使用する場合のuTSC_get_position関数の詳細は以下になります。

① 検出方向指定

uTSC_get_control関数の呼び出しを行い，検出方向の設定を行います。

② データ受信完了

タッチスクリーン・コントローラからの受信データ数をカウントします。

③ 1ビットデータ受信

タッチスクリーン・コントローラから1ビットデータを取得します。

④ 極性反転

各軸の極性反転が設定されている場合，値を反転します。

⑤ 戻る

タッチスクリーン・コントローラからのデータ受信を終了します。

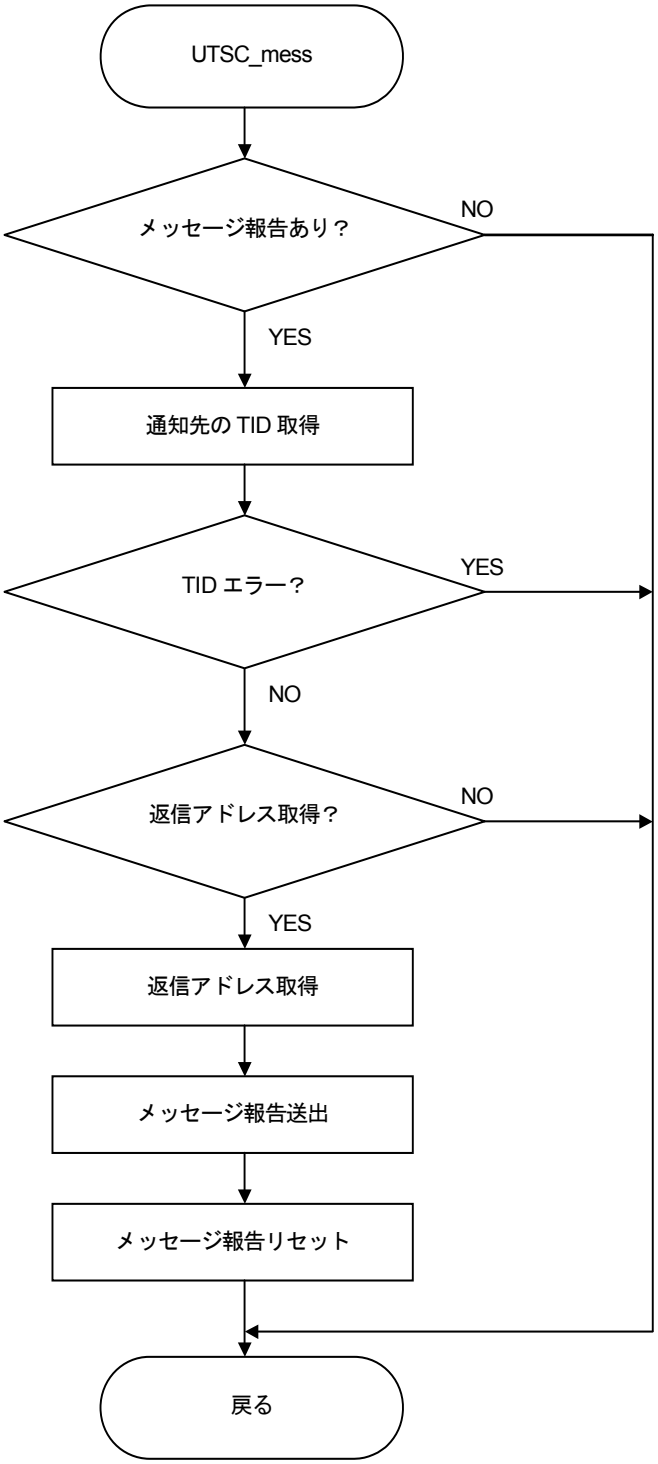
3. 6　メッセージ報告（uTSC_mess）

uTSC_mess関数は，ホスト・マシンへのメッセージ報告（検出座標通知）を行います。この機能が不要であれば本関数は削除可能です。

（1）概略フロー

uTSC_mess関数の流れを図3－21に示します。

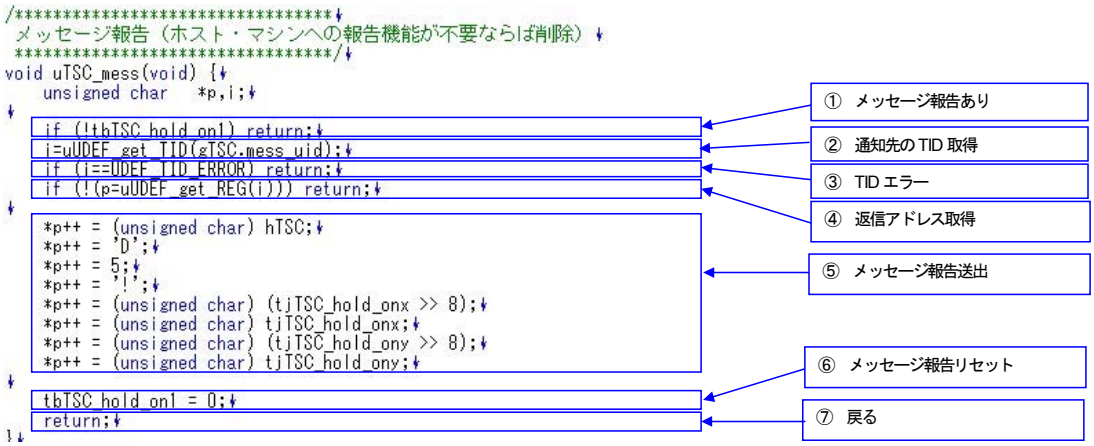
図3－21　uTSC_mess関数フローチャート



(2) uTSC_mess関数プログラム・リスト

uTSC_mess関数のプログラム・リストを図3－22に示します。

図3－22　uTSC_mess関数プログラム・リスト



(3) uTSC_mess関数詳細

uTSC_mess関数の詳細は以下になります。

① メッセージ報告あり

メッセージ報告があるか判定し，報告がなければ終了します。

② 通知先のTID取得

通知先のTIDを取得します。TIDについては「78K0/Kx2サンプル・プログラム・簡易OS編（U19214J）」を参照してください。

③ TIDエラー

TIDにエラーがあるか判定し，エラーの場合終了します。

④ 返信アドレス取得

返信アドレスを取得し，取得できなかった場合終了します。

⑤ メッセージ報告送出

メッセージ報告を送出します。

⑥ メッセージ報告リセット

メッセージ報告の状態をリセットします。

⑦ 戻る

メッセージ報告を完了し，終了します。

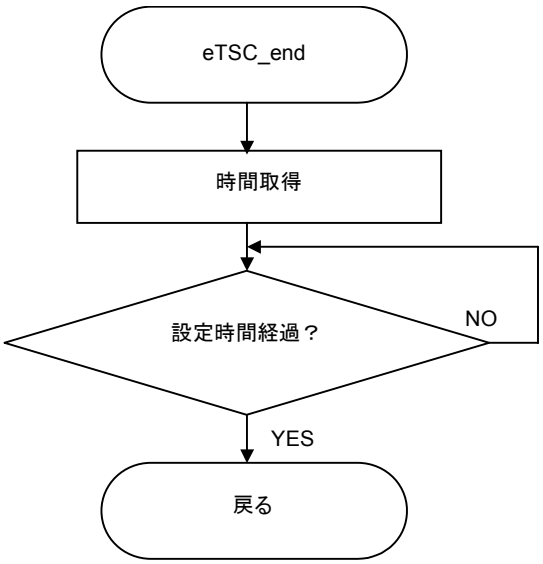
★ 3. 7　電圧安定待ち（eTSC_wait）

eTSC_wait関数は，端子電圧が安定するまで待機を行います。

(1) 概略フロー

eTSC_wait関数の流れを図3－23に示します。

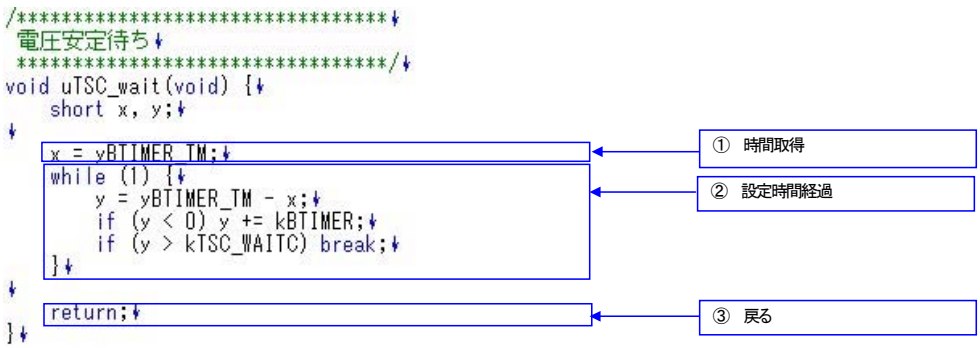
図3－23　eTSC_end関数フローチャート



(2) eTSC_wait関数プログラム・リスト

eTSC_wait関数のプログラム・リストを図3－24に示します。

図3－24　eTSC_wait関数プログラム・リスト



(3) eTSC_wait関数プログラム・リスト

eTSC_wait関数の詳細は以下になります。

① 時間取得

現在の時間を取得します。

② 設定時間経過

設定された時間まで待機します。

③ 戻る

待機を終了します。

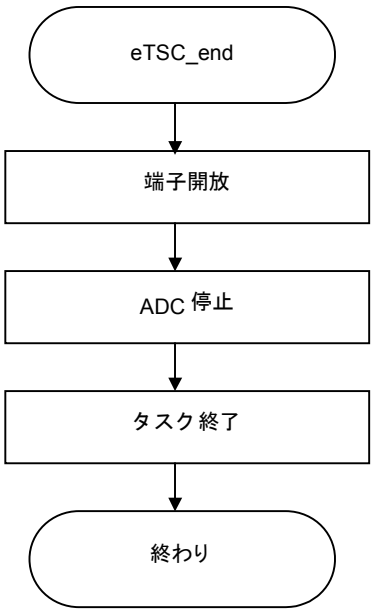
3. 8　終　　了（eTSC_end）

eTSC_end関数は，TSCタスクの終了を行います。簡易OSを使用しない場合で，端子開放が不要ならば省略できます。

（1）概略フロー

eTSC_end関数の流れを図3－25に示します。

図3－25　eTSC_end関数フローチャート



(2) プログラム・リスト

★ **(a) 内蔵A/Dコンバータ方式のリスト**

内蔵A/Dコンバータを使用する場合のeTSC_end関数のプログラム・リストを図3－26に示します。

図3－26　eTSC_end関数プログラム・リスト（内蔵A/Dコンバータ方式）

```
/************************************************************************
 ************************************************************************
 終了
 ************************************************************************
 ************************************************************************/
void eTSC_end(void) {

    /*** 端子解放（スタンバイ状態に設定）  ***/
    zmSWXP = 1;
    zmSWYP = 1;
    zmSWXM = 1;
    zmSWYM = 1;
    zmPEN_INQ = 1;

    /* ADC停止（念のため）*/
    ADM = 0;

    vExit_task();                   /* 簡易OSを使用しない場合は，return文に変更 */
}
```

① 端子開放

② ADC停止

③ タスク終了

(b) 専用コントローラ方式のリスト

専用コントローラを使用する場合のeTSC_end関数のプログラム・リストを図3－27に示します。

図3－27　eTSC_end関数プログラム・リスト（専用コントローラ方式）

```
/************************************************************************
 ************************************************************************
 終了
 ************************************************************************
 ************************************************************************/
void eTSC_end(void) {

    /*** 端子解放（スタンバイ状態に設定）  ***/
    zmCSI_SPC = 1;
    zmCSI_SPO = 1;
    zmCSI_CS2 = 1;

    vExit_task();                   /* 簡易OSを使用しない場合は，return文に変更 */
}
```

① 端子開放

③ タスク終了

(3) 詳細説明

eTSC_end関数の詳細は以下になります。

① 端子開放

内蔵A/D用端子を解放します。

② ADC停止

ADCを停止します。

③ タスク終了

TSCタスクを終了します。

★ 第4章　サンプル・アプリケーションのビルド方法

この章では，本サンプル・アプリケーションのビルド方法の詳細について説明します。

4. 1　フォルダ構成

サンプル・アプリケーションは下記からダウンロードできます。

http://www.necel.com/micro/ja/designsupports/sampleprogram/78k0/kx2/index.html

ダウンロードしたファイルの名称は，次のようになっています。

SM06EC_Rxxx（xxx**はリビジョン）・・・内蔵**A/D**コンバータ方式**

SM06FC3_Rxxx（xxx**はリビジョン）・・・専用コントローラ方式**

フォルダの構成は次のようになっています。

サンプル・アプリケーションのフォルダ構成

SM06EC_Rxxx／SM06FC3_Rxxx　（xxx はリビジョンです。）

- SM06ECj.TXT／SM06FC3j.TXT　・リリース・ノート（変更情報などが記載されています。）
- SM06EC. prw／SM06FC3.prw　・ワークスペース・ファイル（通常使用する場合はこれを開いてください）
- SM06EC. prj／SM06FC3.prj　・プロジェクト・ファイル（自動参照）
- ¥FONT_LIB　・フォント・データおよびライブラリ
- ¥KERNEL　・簡易 OS 関連ファイル
- ¥TASK_A　・サンプル・アプリケーション共通ファイル
- ¥TASK_E／TASK_F
  - TSC_c.c　・本書で解説するプログラム
  - TSC_c.h　・本書で解説するプログラムのヘッダ･ファイル
  - LCD_lib_c.h　・使用する LCD モジュールを選択します。
  - その他のファイル　・各種サンプル・プログラムの共通ファイル
- ¥TASK_EC／TASK_FC　・本サンプル・アプリケーション専用ファイル

4.2　実行モジュールの作成

本サンプル・プログラムの実行モジュール作成について説明します。

必要とするソフトウエア・ツールのバージョンが異なる場合，ソフトウエア・ツールの入手を行ってください。

なお，本サンプル・プログラムは，使用するLCDモジュールに応じてソースを１箇所書き換える必要があります。

書換えは，後述のプロジェクト・ファイルの設定を行った後に，ダウンロード・ファイル内のLCD_lib_c.hの下記場所に対して行います。

（1）BG12864Aモジュール＋SM06B2ボードを選択する場合

LCD_lib_c.hファイル内の「LCD_CODE_TYPE」を「302」に変更します。

```
/*****************************************************************************/
/**    1. ＬＣＤの種別（Code）を表から選んで，LCD_CODE_TYPEに設定してください。  **/
        #define       LCD_CODE_TYPE   302                                    /**/
/**                                                                          **/
/**    2. ビットマップ領域数（1～4）を，nLCD_plane に設定してください。        **/
        #define       nLCD_plane      1                                      /**/
/**                                                                          **/
/**    3. CPU速度（fCPU周波数）を，BOARD_FCPU に設定してください。             **/
/**       （デモ・プログラムでは，UDEF_BOARD_NUMBER_c.hの中で設定）           **/
/*****************************************************************************/
```

（2）BG12864Aモジュール＋SM07Aボードを選択する場合

LCD_lib_c.hファイル内の「LCD_CODE_TYPE」を「309」に変更します。

```
/*****************************************************************************/
/**    1. ＬＣＤの種別（Code）を表から選んで，LCD_CODE_TYPEに設定してください。  **/
        #define       LCD_CODE_TYPE   309                                    /**/
/**                                                                          **/
/**    2. ビットマップ領域数（1～4）を，nLCD_plane に設定してください。        **/
        #define       nLCD_plane      1                                      /**/
/**                                                                          **/
/**    3. CPU速度（fCPU周波数）を，BOARD_FCPU に設定してください。             **/
/**       （デモ・プログラムでは，UDEF_BOARD_NUMBER_c.hの中で設定）           **/
/*****************************************************************************/
```

（3）BP240128B2モジュールを選択する場合

LCD_lib_c.hファイル内の「LCD_CODE_TYPE」を「401」に変更します。

```
/*****************************************************************************/
/**    1. ＬＣＤの種別（Code）を表から選んで，LCD_CODE_TYPEに設定してください。  **/
        #define       LCD_CODE_TYPE   401                                    /**/
/**                                                                          **/
/**    2. ビットマップ領域数（1～4）を，nLCD_plane に設定してください。        **/
        #define       nLCD_plane      1                                      /**/
/**                                                                          **/
/**    3. CPU速度（fCPU周波数）を，BOARD_FCPU に設定してください。             **/
/**       （デモ・プログラムでは，UDEF_BOARD_NUMBER_c.hの中で設定）           **/
/*****************************************************************************/
```

4. 2. 1　プロジェクト・ファイルによるビルド

添付のワークスペース・ファイルを使用してビルドを行う操作について説明します。

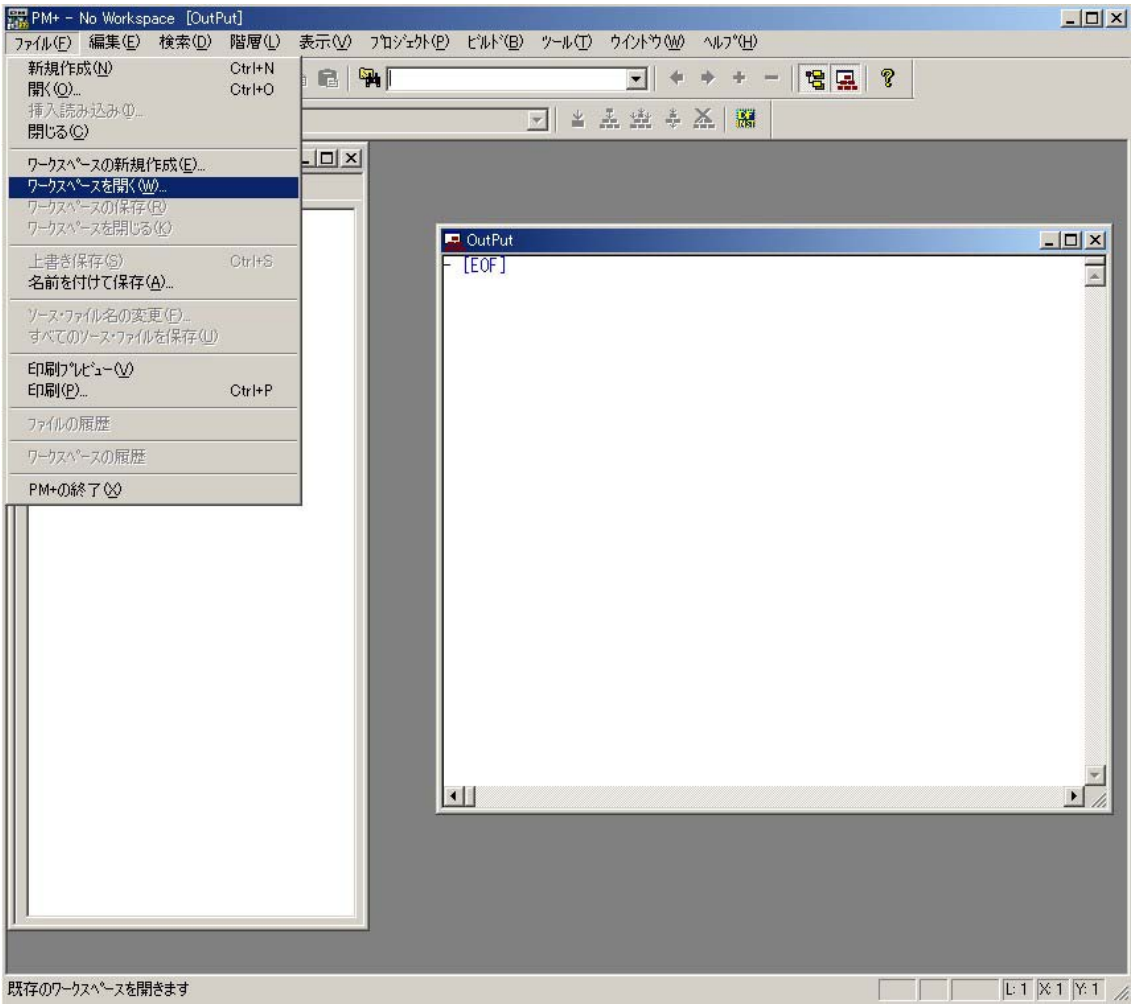
使用するワークスペース・ファイルは次のとおりです。

・SM06EC.prw／SM06FC3.prw

以下の画面はSM06FC3**の例です。**

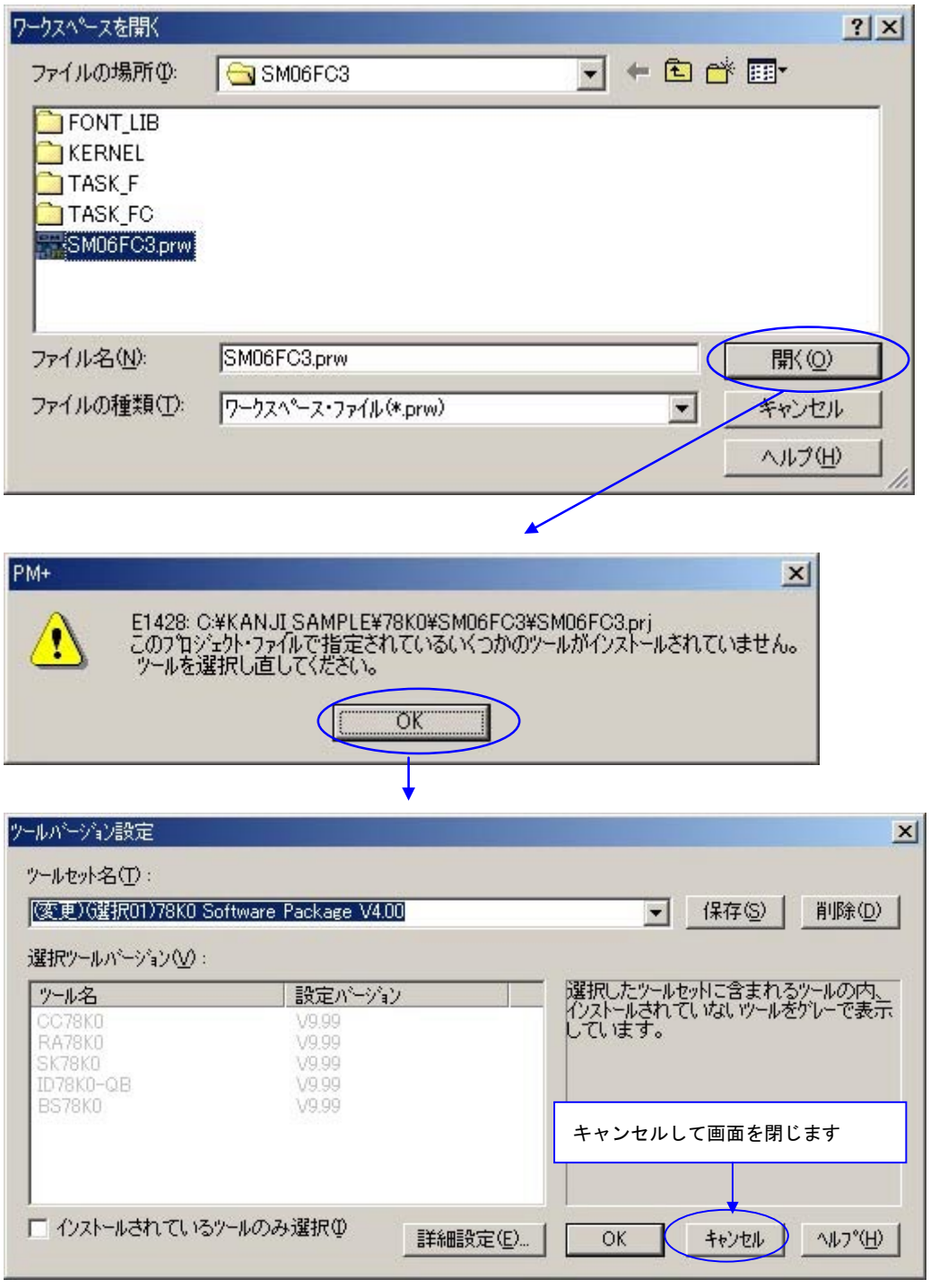
（1）PM＋**を起動します。**

PM＋**を起動後，「ファイル」メニューの「ワークスペースを開く」を実行します。**



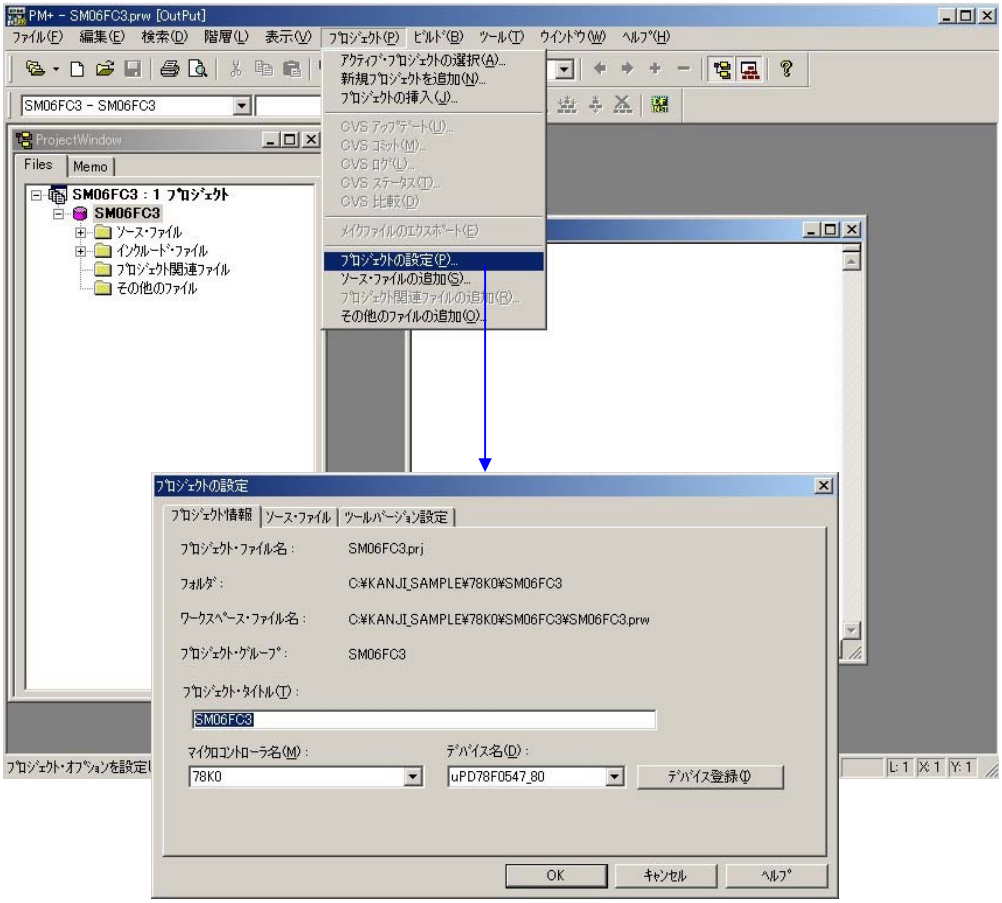
(2) ワークスペースを選択します

ワークスペースを開く画面で，ワークスペース・ファイルを選択します。ワークスペース・ファイルを選択しますと，ツール選択の警告メッセージが表示されます。OKを押すと「ツールバージョン設定」画面が表示されますが，ここではキャンセルを選択して，いったん「ツールバージョン設定」画面を閉じます。



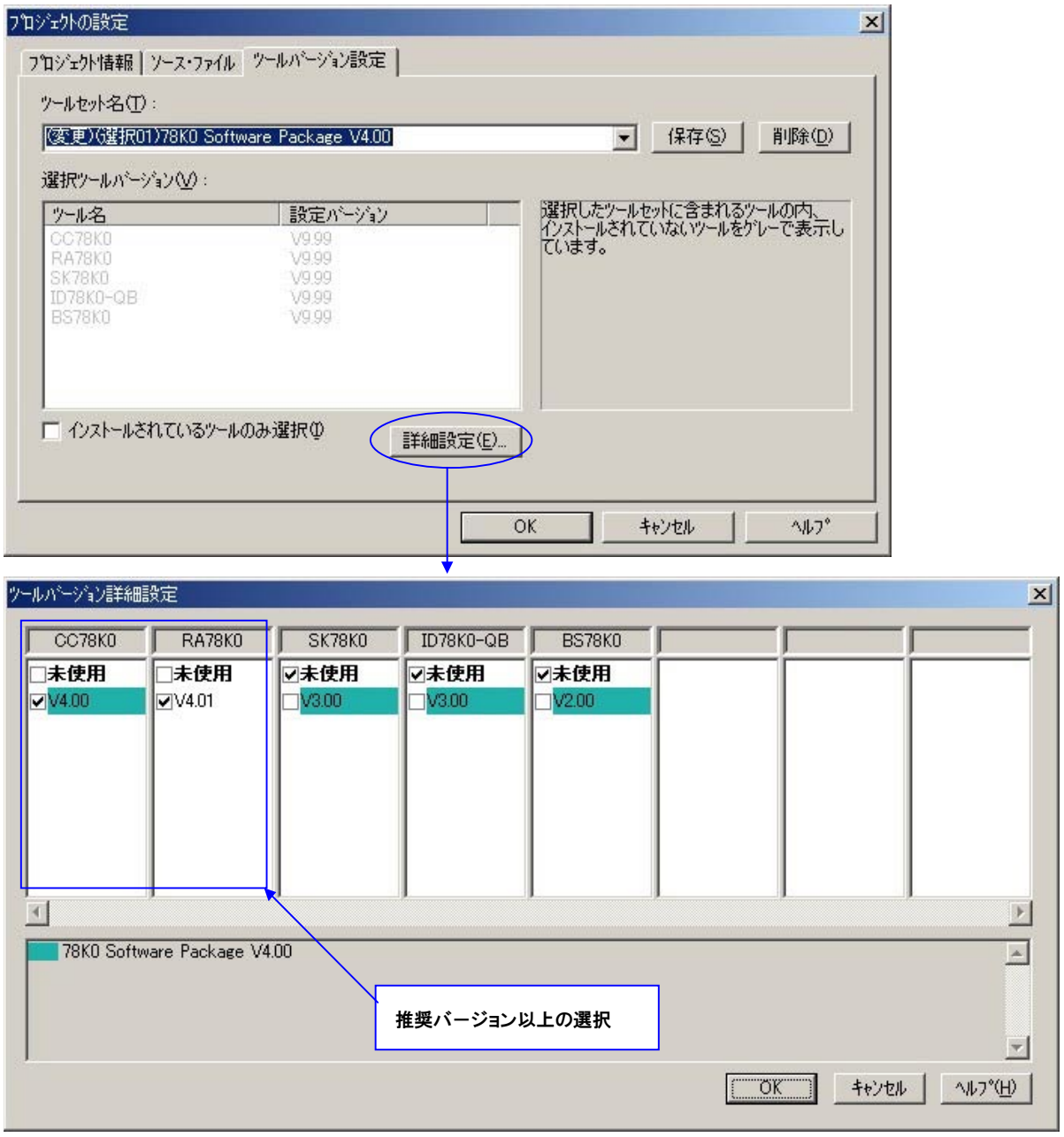
(3) プロジェクト設定画面

ツールを選択するために，「プロジェクト」メニューの「プロジェクトの設定」を実行します。



(4) ツールの選択

「プロジェクトの設定」画面の「ツールバージョン設定」タブの「詳細設定」ボタンを押して，ツールの選択を行います。

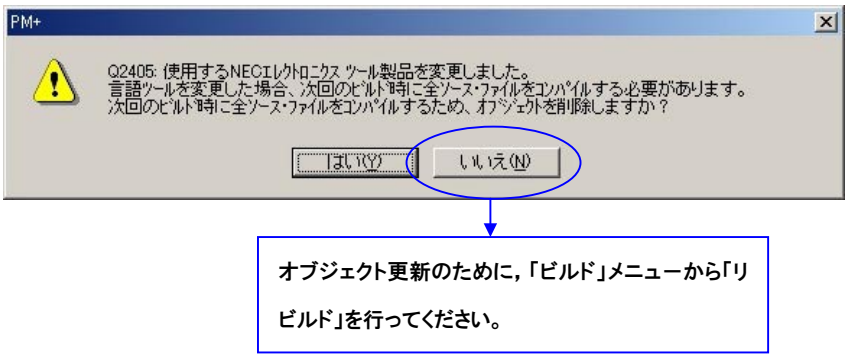


ツールは，推奨バージョン以上を選択してください。

推奨バージョン以上が表示されない場合は，ツールを新しいバージョンにアップデートしてください。

推奨バージョンにつきましては，「1.2　開発環境」の「1. 2. 1　ソフトウエア・ツール」を参照してください。

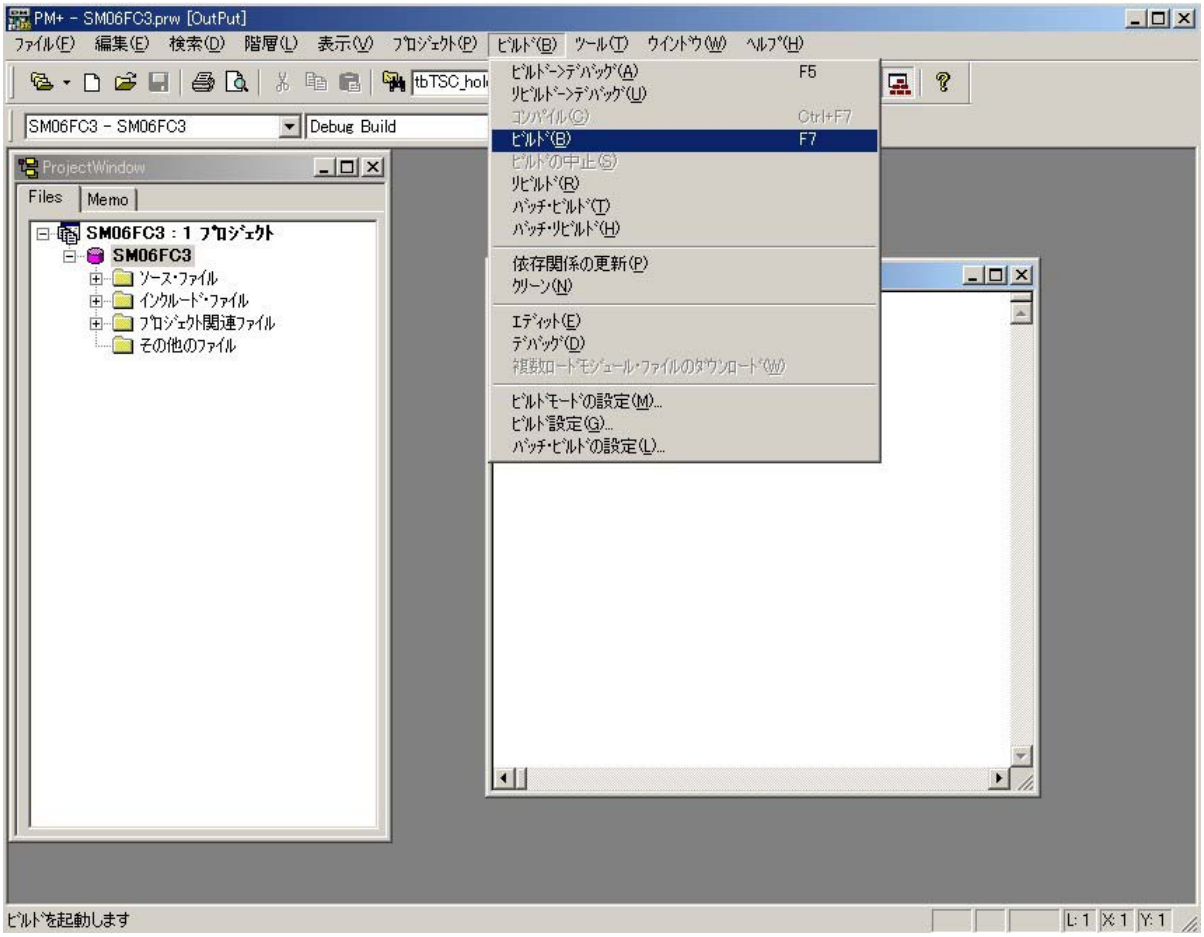
ツール選択後，ツール変更に伴うオブジェクト削除の問い合わせメッセージが表示された場合は，「いいえ」を選択してください。ツールを変更した場合は，オブジェクト更新のために，「リビルド」を行ってください。



オブジェクト更新のために，「ビルド」メニューから「リビルド」を行ってください。

(6) ビルドの実行

「ビルド」メニューの「ビルド」を実行します。ツール・バージョン変更後の最初のビルドは「リビルド」を実行してください。



ビルドできない場合は，「4. 2. 2　プロジェクト・ファイルの新規作成」～「4. 2. 3　オプションの設定」の手順で，ワークスペースを新規作成してビルドを行ってください。

4.2.2　プロジェクト・ファイルの新規作成

ダウンロード・ファイルに同梱されているSM06EC.prw／SM06FC3.prwを使わずに，新規にプロジェクト・ファイルを作成してビルドする方法について説明します。

以下の画面はSM06FC3の例です。

(1) ワークスペースの生成を行います。

① ワークスペース作成の準備を行います。

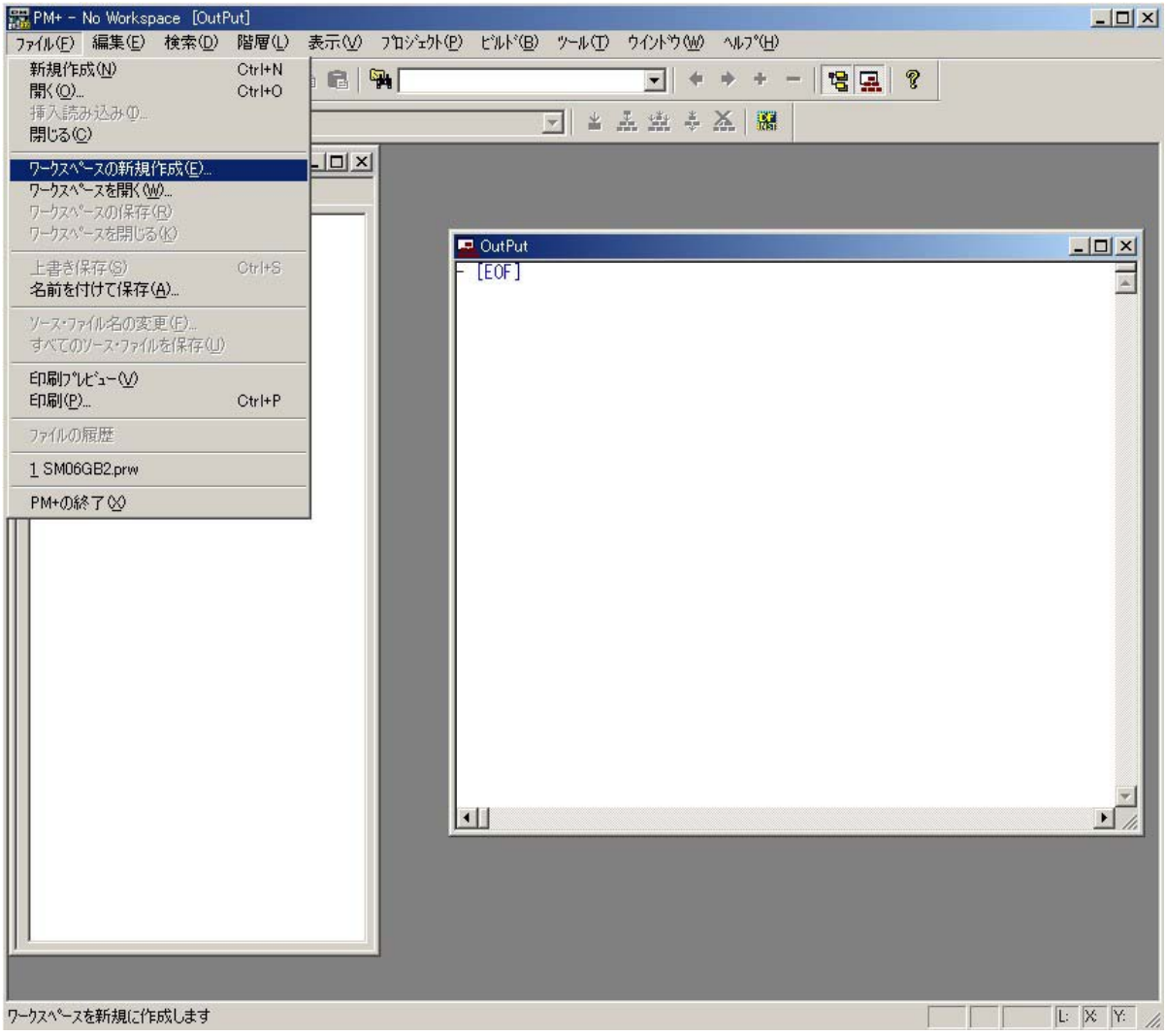
サンプル・アプリケーションをダウンロードしたフォルダ構成のまま使用します。

ここではワークスペースの名称をSM06FC3として以下のフォルダにあるとします。

C:¥KANJI_SAMPLE¥78K0¥SM06FC3

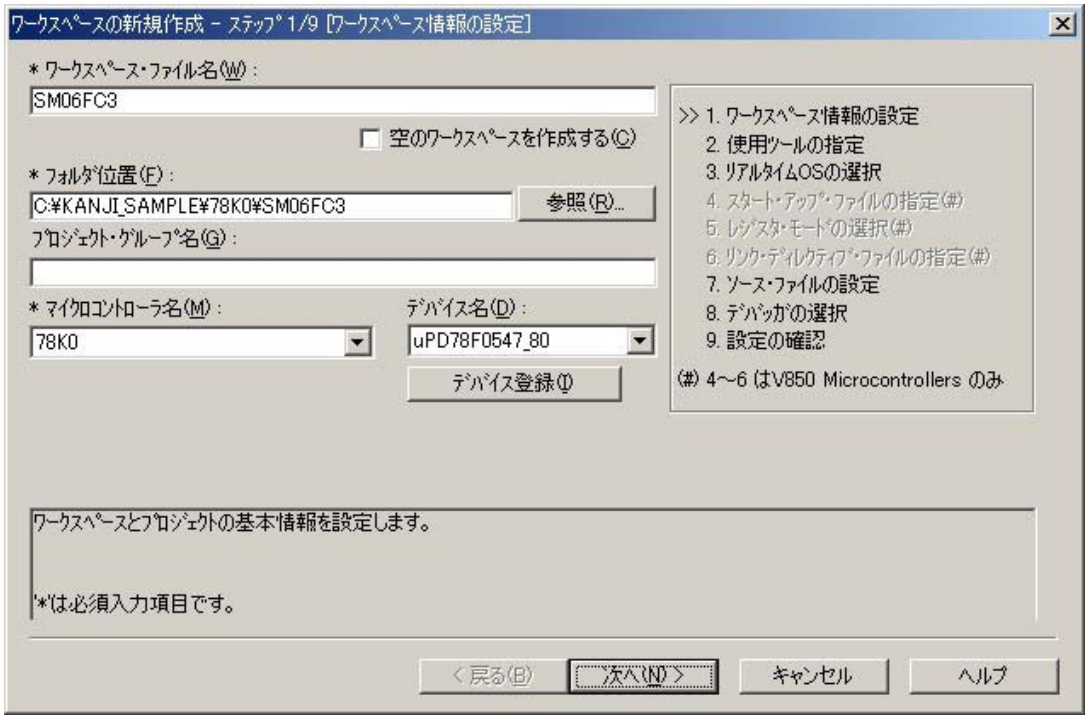
② PM+を起動します。

PM+を起動後，「ファイル」メニューの「ワークスペースの新規作成」を実行します。



③　ワークスペースを新規作成します。

ワークスペース・ファイル名とフォルダ位置を指定し，マイクロコントローラ名とデバイス名を選択します。SM05F3ボードの場合はuPD78F0547_80，SM06Eボードの場合はuPD78F0537_64を選びます。



④　使用ツールの設定。

ツールの選択を行います。詳しくは「4. 2. 1　(4) ツールの選択」を参照してください。

⑤ ソースファイルの選択

使用するソースファイルを選択します。

TSC_c.c	：タッチスクリーン　サンプル・プログラム
Melody_c.c	：メロディ出力タスク・プログラム ペンダウン時のタッチ音生成のために使います。
ADPCM_c.c	：コーデック制御タスク・プログラム タッチ音をヘッドフォンまたはスピーカから出力するために使います。
Graph_c.c	：グラフィック描画タスク・プログラム 格子を表示するために使います。
Print_c.c	：漢字表示タスク・プログラム
LCD_c.c	：表示データをLCDへ転送するタスク・プログラム
LCD_lib.c	：表示データをLCDへ転送するタスク・プログラム（ライブラリ）
COM1_rx_c.c	：ホスト・マシンからデータを受信するタスク・プログラム
COM1_tx_c.c	：ホスト・マシンへ応答データを送信するタスク・プログラム
BTimer_c.c	：OS予約のタイマ処理等
BTimer_INT_a.asm	：OS予約のタイマ処理等（SM06FC3のみ）
Kernel_a.asm	：OSコード部
Kernel_data_a.asm	：OSデータ部
Kernel_OPT_a.asm	：オプションバイト（OS依存性はありません）
Kernel_init_c.c	：初期化プログラム（OS依存性はありません）
Main_c.c	：メイン（システム制御）タスク・プログラム
UDEF_func_c.c	：システム共通処理プログラム
UDEF_data_a.asm	：システム共通処理用データ
FA107LVH.asm	：1/4角フォント定義ファイル
FH014LVH.asm	：半角フォント定義ファイル
FK014LVH_108.asm	：14ドット　漢字フォント定義ファイル
FK024LVH_28.asm	：24ドット　漢字フォント定義ファイル
Data_a.asm	：バンク0配置データ

4.2.3　オプションの設定

本サンプル・プログラムの実行モジュールを作成するために必要なオプション設定について説明します。なお，ダウンロード・ファイルに同梱のSM06EC.prw／SM06FC3.prwを使用してビルドする場合は，これらの値は自動的に設定されます。

(1) コンパイラのオプションを設定します。

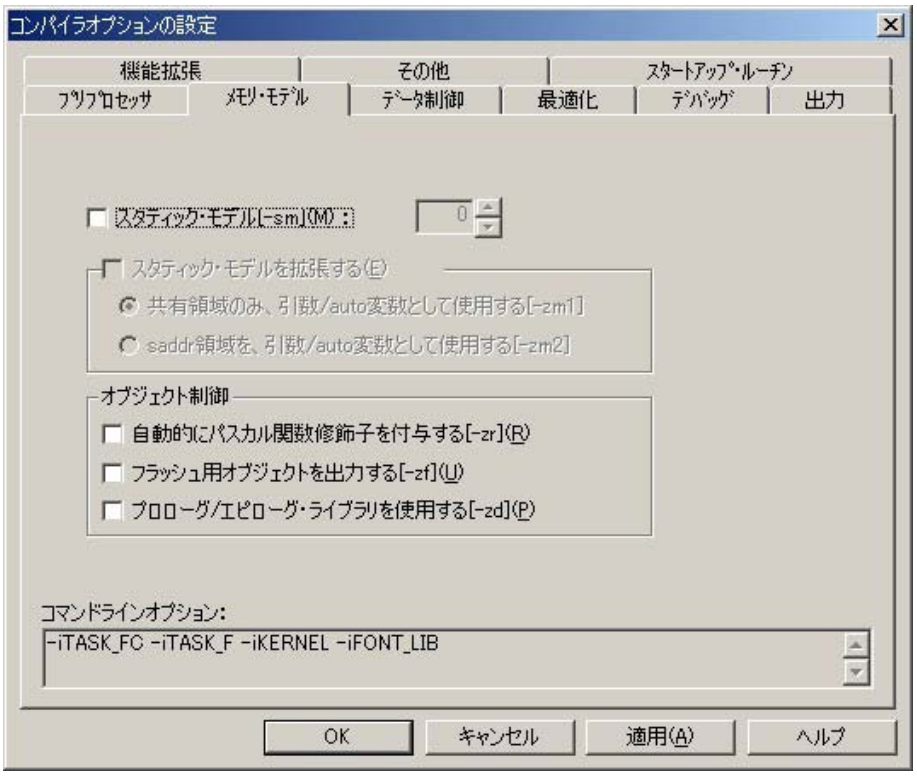
「ツール」メニューの「コンパイラオプションの設定」を選択し，設定画面を表示します。

① インクルード・パスを追加します

「プリプロセッサ」タブを選択して，「インクルード・ファイル・パス」に「FONT_LIB」フォルダと「KERNEL」フォルダと「TASK_A」フォルダと「TASK_E/TASK_F」フォルダおよび「TASK_EC/TASK_FC」フォルダを追加します。また「定義マクロ」に「KOS_CODE_TYPE=0」を追加します。

② メモリ・モデルを選択します。

メモリ・モデルは「ノーマル」にのみ対応します。「メモリ・モデル」タブを選択し，「スタティック・モデル」オプションのチェックが外れていることを確認してください。



③　アセンブラ・ソース・モジュール・ファイルの出力をチェックします。

「出力」タブを選択し，「アセンブラ・モジュール・ファイルの出力」チェックをオンにします。

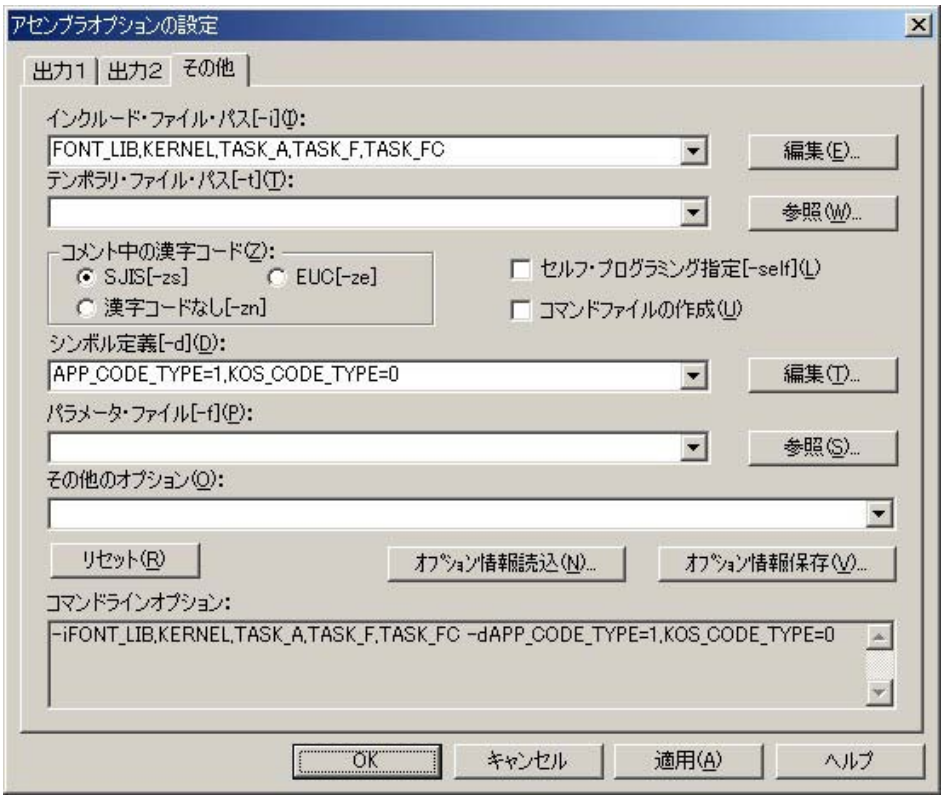
設定が完了したら，ウインドOKボタンを押します

(2) アセンブラのオプションを設定します。

「ツール」メニューの「アセンブラオプションの設定」を選択し，設定画面を表示します。

① 「その他」タブのインクルード・ファイル・パスとシンボル定義を追加します。

- **「インクルード・ファイル・パス」に「**KERNEL**」フォルダと「**FONT_LIB**」フォルダと「**TASK_A**」フォルダと「**TASK_E/TASK_F**」フォルダおよび「**TASK_EC/TASK_FC**」フォルダを追加します。**
- **「シンボル定義」に「**APP_CODE_TYPE=1**」を追加します。（**R3.10**以降は設定不要）**
- **「シンボル定義」に「**KOS_CODE_TYPE=0**」を追加します。**



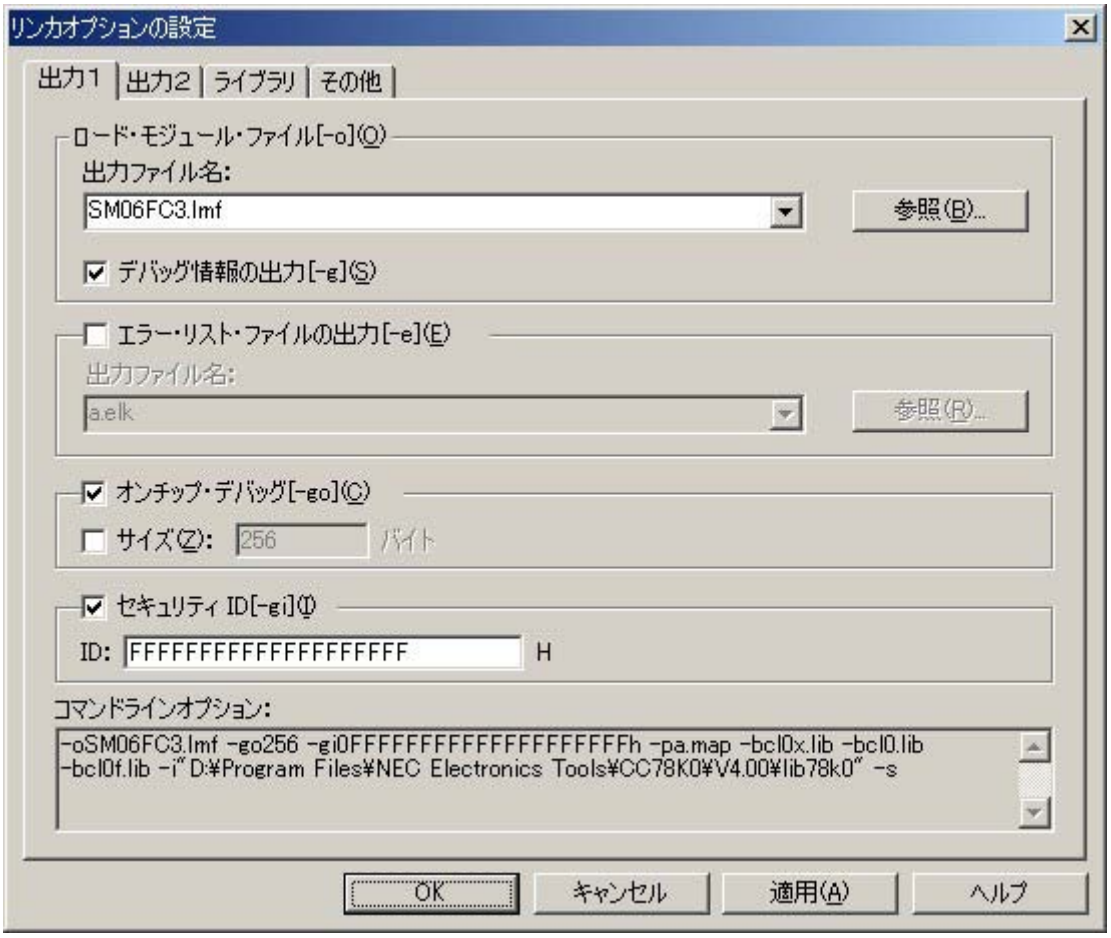
(3) リンカのオプションを設定します。

「ツール」メニューの「リンカオプションの設定」を選択し，設定画面を表示します。

① 「出力1」タブのオプションを設定します。

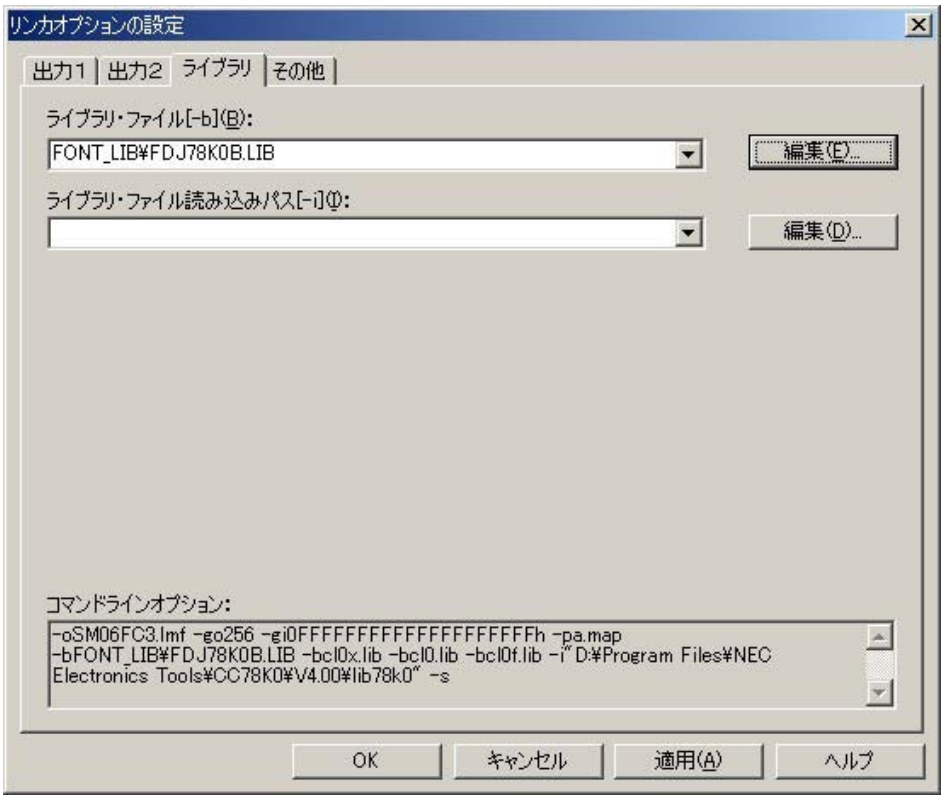
「ロード・モジュール・ファイル名」，「セキュリティID」を必要に応じて設定します。

デバッグ時は，「オンチップ・デバッグ・オプション・バイト」を設定します。



② **ライブラリを指定します。**

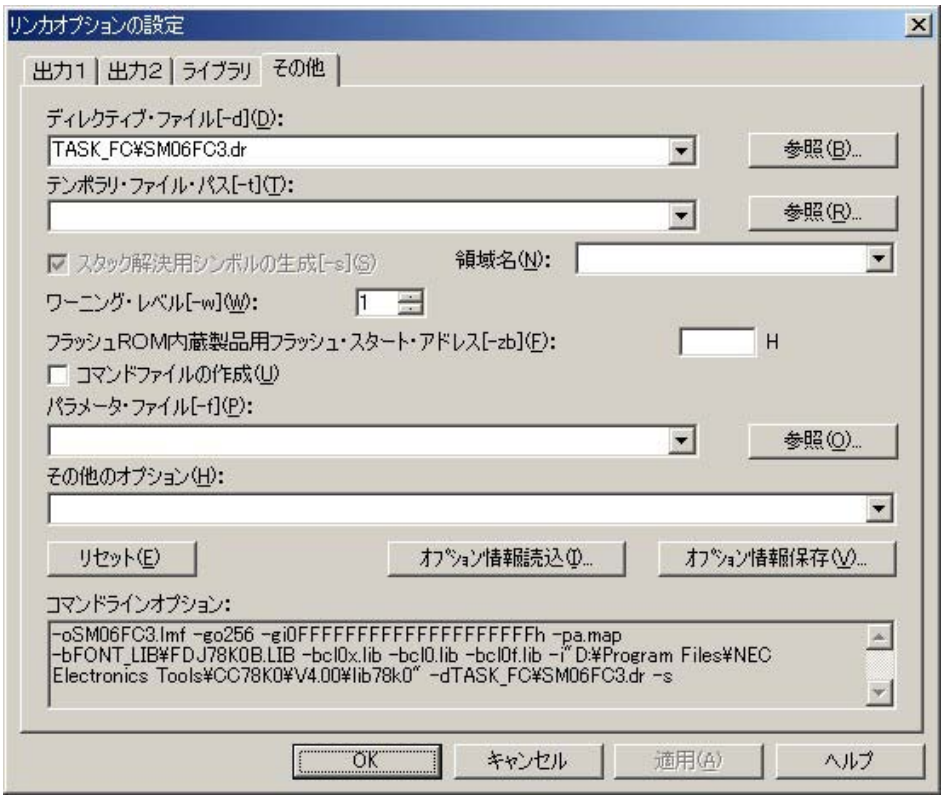
「ライブラリ」タブの「ライブラリ・ファイル」に「FDJ78K0B.LIB**」を指定します。**



ライブラリの指定は, ProjectWindow**のプロジェクト関連ファイルで追加を行うこともできます。**

③ **リンクディレクティブ・ファイルを指定します。**

「その他」タブでSM06FC3.dr**を指定します。**



第5章　サンプル・アプリケーションの実行方法

この章では，プログラムの実行方法とホスト・マシンからサンプル・プログラム（TSC_c.c）を動作させるためのコマンド例について説明します。

デモ用ボードの設定とホスト・マシンのターミナル・ソフトウエアの設定方法については，使用するボードによって異なります。以下のマニュアルを参照してください。

★ （1）内蔵A/Dコンバータ方式（SM07Aベース・ボード，SM06Eデバイス・ボード）

- 「ヒューマン・マシンI/Fデモ用ベース・ボード　ユーザーズ・マニュアル（U20117J）」
- 「ヒューマン・マシンI/Fデモ用78K0ボード　ユーザーズ・マニュアル（U20118J）」
- 「78K0サンプル・プログラム（ドットLCD制御編）アプリケーション・ノート（U19531）」

（2）専用コントローラを使用する場合（SM05A2，SM05F3，SM06B2）

- 「漢字表示デモンストレーション用ベース・ボード　ユーザーズ・マニュアル（U19207J）」
- 「漢字表示デモンストレーション用78K0R/KF2ボード　ユーザーズ・マニュアル（U19208J）」
- 「漢字表示デモンストレーション用拡張ボード ユーザーズ・マニュアル（U19526）」
- 「78K0サンプル・プログラム（ドットLCD制御編）アプリケーション・ノート（U19531）」

5. 1　プログラムの書き込みと起動方法

デバッガによる起動とデバッガを使わずにボード単体で起動する方法について説明します。

5. 1. 1　デバッガで起動する場合

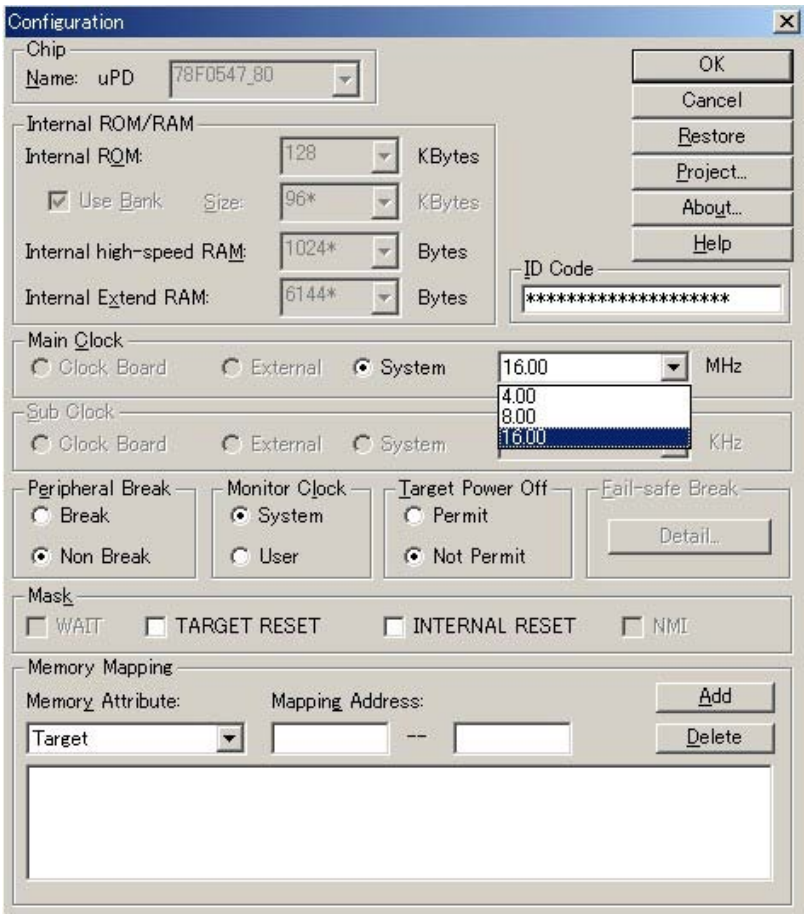
ビルド後，デバッガ（ここではID78K0-QBを使用します）を起動します。

（1）初回起動時にエラーが出た場合は消去（イレース）を行います。

初回起動時に「このデバイスではオンチップデバッグができません」とメッセージが表示されることがあります。この場合はQB-Programmerなどで消去を行ってからデバッガを起動してください。

★　**(2) メイン・クロックの周波数設定**

OCDボード上に発振器をつけていない場合は，システムクロックの選択が出来ますので，16MHzを選択してください。ここで選択した周波数は動作周波数とは異なります。動作周波数はデバイス・ボードに実装されている発振子周波数となり，SM05F3は16MHz，SM06Eは20MHzになります。



5. 1. 2　プログラマで書き込んで起動する場合

ビルドで作成されたHEXファイルをプログラマで書き込み後，ベース・ボードからプログラマを外すとデバイス・ボードが起動します。

5. 2　ターミナル・ソフトウエアの操作について

コマンド入力時の操作について説明します。

(1) キーボードからのコマンド入力

コマンド入力時に注意が必要な点を説明します。

①　コマンドの編集

コマンドはリターン・キーを押すと確定します。リターン・キーを押すまでは，バックスペースによる修正はできますが，そのほかの編集キーによる操作は無効です。

②　デモボードからの表示

入力中にデモボードからのメッセージが表示されても，表示がなかったものとして，続けて入力してください。

(2) コピー＆ペーストによるコマンド入力

コマンドは，後述のコマンド例をコピー＆ペーストしても動作します。コメントも含めて複数行を一括で貼り付けしてもかまいません。

PDFファイルからのテキスト・コピーでは動作しない場合は，ダウンロード・ファイルのリリース・ノートの後ろにあるコマンド・サンプル集からコピーしてください。

5. 3　座標の取得方法

本サンプル・プログラムは起動されるとすぐにデフォルト・パラメータに基づいて座標取得動作を行います。補正パラメータの変更やタッチ音の変更などは，パラメータ・ライト・コマンドで行います。

5. 4　コマンドの書式

(1) パラメータ・ライト・コマンド

書式：$gW' {パラメータ・アドレス}　{書き込みデータの並び}

★

アドレス	内容
0	検出座標の一致期間を10 ms単位で設定します。デフォルト値は5です。
1	動作モードを設定します。デフォルト値はF2Hです。 bit7：1ならペン・アップ時にオフ・コード（-1，-1）を生成します。 bit6：1ならペン・ダウン時にクリック音を出すようにMelodyユニットへ要求します。 bit5：1ならペン・ダウン時にバックライトを点灯するようにLCDユニットへ要求します。 bit4：1なら他のユニットに対して検出座標やオフ・コードを公開します。 bit3：1ならY軸の極性を反転します。 bit2：1ならX軸の極性を反転します。 bit1：1なら指定ユニットに対して検出座標やオフ・コードを自動的に報告します。 bit0：1ならX軸とY軸を入れ替えます。
2	自動報告を行う場合に，報告先ユニット記号を指定します。デフォルト値は”u”です。
3	ドリフト許容範囲（絶対値）を0.25ドット単位で設定します。 デフォルト値は8H（SM06B2ボード用），14H（SM07Aボード用）です。
4～5[注]	X軸オフセット。設定値 = オフセット量 × 128。デフォルト値は0です。
6～7[注]	Y軸オフセット。設定値 = オフセット量 × 128。デフォルト値は0です。
8～9[注]	X軸スケール。設定値 = 縮小比 × 32768。デフォルト値は8000Hです。
AH～BH[注]	Y軸スケール。設定値 = 縮小比 × 32768。デフォルト値は4000Hです。

注. リトル・エンディアンで設定します。

(2) ペンダウン・エミュレーション

書式：$gE {X座標}　{Y座標}

・座標

0～7FFFHの範囲の座標を2バイトのビッグ・エンディアンで指定します。

オフ・コードは，FFFFHを指定します。

このコマンドの実行により指定した値が，タッチスクリーン制御プログラムのグローバル変数gTSC.coedex，gTSC.codeyに設定されます。

拡張ボードのデモプログラム（SM06EY/SM06FW3 R3.10以上）では，メニューで「タッチスクリーン」を選択するとgTSC.coedex，gTSC.codeyに設定された値がLCDに表示されます。

5. 5　コマンド例

(1) クリック音，軸極性などの設定例

オフ・コード生成あり，タッチ音あり，検出座標の自動報告ありのコマンド例を次に示します。

SM06B2**ボード使用時**　　$gW'1 F2

SM07A**ボード**＋BG12864A　　$gW'1 FA

SM07A**ボード**＋BP240128　　$gW'1 FF

なお軸極性はボードによって異なります。

(2) 補正パラメータの設定例

タッチスクリーンから座標を得るためには，補正パラメータをコマンドから設定します。

①　240x128**ドット**LCD**パネル設定例（**BP24128B2**）**

```
$JF'00 00 42 A A A A A A A
$NP'00 08 EF 08 00 01
$NP'00 18 EF 18 00 01
$NP'00 28 EF 28 00 01
$NP'00 38 EF 38 00 01
$NP'08 00 08 3F 20 FF
$NP'18 00 18 3F 20 FF
$NP'28 00 28 3F 20 FF
$NP'38 00 38 3F 20 FF
$NP'48 00 48 3F 20 FF
$NP'58 00 58 3F 20 FF
$NP'68 00 68 3F 20 FF
$NP'78 00 78 3F 20 FF
$NP'88 00 88 3F 20 FF
$NP'98 00 98 3F 20 FF
$NP'A8 00 A8 3F 20 FF
$NP'B8 00 B8 3F 20 FF
$NP'C8 00 C8 3F 20 FF
$NP'D8 00 D8 3F 20 FF
$NP'E8 00 E8 3F 20 FF
$gW'3 00 C404 D504 2008 8E04
```

上記コマンドを設定すると格子が表示され，タッチスクリーンをタッチ・ペンなどでタッチすると，タッチした座標がメッセージ通知されます。

左上格子交点をタッチした時の通知例を次に示します。

通知例

#gD!'0008000B

X**軸値：**8（10**進**）

Y**軸値：**11（10**進**）

② 128x64ドットLCDパネル設定例（BG12864A）

$JF'00 00 42 A A A A

$NP'00 08 7F 08 00 01

$NP'00 18 7F 18 00 01

$NP'00 28 7F 28 00 01

$NP'00 38 7F 38 00 01

$NP'08 00 08 3F 20 FF

$NP'18 00 18 3F 20 FF

$NP'28 00 28 3F 20 FF

$NP'38 00 38 3F 20 FF

$NP'48 00 48 3F 20 FF

$NP'58 00 58 3F 20 FF

$NP'68 00 68 3F 20 FF

$NP'78 00 78 3F 20 FF

$gW'3 00 0005 0004 9004 8802

上記コマンドを設定すると格子が表示され，タッチスクリーンをタッチ・ペンなどでタッチすると，タッチした座標がメッセージ通知されます。

左上格子交点をタッチした時の通知例を次に示します。

通知例

#gD!'0008000C

X軸値：8（10進）

Y軸値：12（10進）

（3）ペンダウン・エミュレーション例

X座標＝128（0080H），Y座標＝64（0040H）の例を次に示します。

$gE'0080 0040

★

付録A　改版履歴

A. 1　本版で改訂された主な箇所

箇　所	内　容
はじめに	
p.7	**関連資料**を変更
第1章　概　説	
p.11	1. 1. 2　**評価ボード**に内蔵A/Dコンバータを使用する場合を追加
第2章　座標検出方式	
p.16	2. 2. 2　**測定電圧**に電圧安定待ち時間の記述を追加
p.18	2. 3. 1（3）　**座標取得手順**に安定待ち時間を追加
p.19	2. 3. 1（4）　SM07A**ベース・ボード使用時の制御方法**を追加
p.22	2. 3. 2（3）　**座標取得手順**に安定待ち時間を追加
p.23	**図**2－12　**ドリフト許容範囲**の範囲内適用位置を“最後”に変更
p.26～35	2. 5. 2　**準備**の内容を変更
p.36	2. 5. 3　**タッチスクリーン・グリッド取得**のコマンド・パラメータを変更
p.40	2. 5. 4④　**設定値算出のための係数**を変更
第3章　アプリケーション実装例	
p.42	**図**3－1　**機能概要図（内蔵**A/D**コンバータを使用する場合）**を追加
p.46	**表**3－2，**表**3－3　**端子および特殊機能レジスタ（ＳＦＲ）にかかわる定義**を追加
p.48	3. 2（2）　eTSC**関数プログラム・リスト**に内蔵A/Dコンバータ方式のリストを追加
p.51	3. 3（2）　eTSC_on**関数プログラム・リスト（ペンダウン位置取得）**に内蔵A/Dコンバータ方式を追加
p.60	3. 4　**検出軸の設定制御**に変更し，方式ごとに項番分離
p.65	3. 5　**位置読み取り**の内容を方式ごとに項番分離
p.72	3. 7　**電圧安定待ち**を追加
p.75	3. 8（2）　**プログラム・リスト**に内蔵A/Dコンバータ方式のリストを追加
第4章　サンプル・アプリケーションのビルド方法	
p.76～87	**サンプル・プログラム**SM06ECを追加
第5章　サンプル・アプリケーションの実行方法	
p.93	（1）**内蔵**A/D**コンバータ方式**の参照資料を追加
p.94	5. 1. 1（2）　**メイン・クロックの周波数設定**に解説を追加
p.97	5. 4（1）　**パラメータ・ライト・コマンド**のパラメータ内容を変更
付録A　**改版履歴**	
p.100	**付録**A　**改版履歴**を追加

〔メ　モ〕

【発行】NEC**エレクトロニクス株式会社（**http://www.necel.co.jp/**）**

【問い合わせ先】http://www.necel.com/contact/ja/