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

フォント選択編

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　U19528JJ1V0AN00（第1版）
発行年月　March 2009 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，V_{IL}（MAX.）からV_{IH}（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，V_{IL}（MAX.）からV_{IH}（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V_{DD}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。

M8E0710J

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向けフォント・アクセス・ライブラリを用いた漢字テキスト表示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
- ライブラリ仕様
- 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。

漢字表示デモンストレーション・ボードの関連資料

資　料　名	資料番号	
	和　　文	英　　文
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（漢字フォント編）アプリケーション・ノート	U19211J	未定
78K0R/Kx3サンプル・プログラム（漢字フォント編）アプリケーション・ノート	U19212J	未定
V850ES/Jx3サンプル・プログラム（漢字フォント編）アプリケーション・ノート	U19213J	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	このノート	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[注1]	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[注1]	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[注1]	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[注1]	未定

注1.　2009年夏発行予定

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向けフォント・アクセス・ライブラリFDJ78K0A.LIB/FDJ78K0B.LIBの使い方，フォント定義ファイルの生成方法，およびサンプル・プログラムにおける文字表示方法について説明します。

・FDJ78K0A.LIB：78K0非バンク用

・FDJ78K0B.LIB：78K0バンク用

1.1　ライブラリ概要

フォント・ユーティリティは，フォント・データをソース形式のフォント定義ファイルとして生成します。このフォント定義ファイルをコンパイルして得られるオブジェクト形式のフォント・データを簡単に使うための関数として，次のものが用意されています。

・DEFINE_FONT_TYPE：使用するフォント・データ名を宣言するマクロ関数

・uFont_get_next_code：　文字列から次の文字コードを得る関数

・uFont_set_type：　　以下の関数で操作するフォント・データ名を設定する関数

・uFont_get_parameter：　フォントのパラメータを得るマクロ関数

・uFont_get_font_addr：　フォントの位置を得る関数

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）でもかまいませんが，生成できるオブジェクト・サイズに制限があります

フォント・ユーティリティでは制限を超える部分は，HEXファイルとして出力されるため，プログラム実行前にHEXファイルの編集が必要になります。編集方法の手順は，「フォント・ユーティリティ　ユーザーズ・マニュアル（U19527）」を参照してください。

1.2.2 評価ボード

評価用ボードには，用途に応じて次のものがあります。

(1) 表示を伴わないライブラリ単体の評価を行う場合

QB-78K0KF2-TBまたはTK-78K0/KF2を使用できます。ただし，これらのボードでは後述のサンプル・アプリケーションを動作させることはできません。

(2) LCD表示を伴う評価を行う場合

次の3つが必要となります。

(a) 漢字表示デモンストレーション用ベース・ボード（SM05A2）

(b) 漢字表示デモンストレーション用78K0**ボード**（SM05F2）

(c) 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-FP4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。

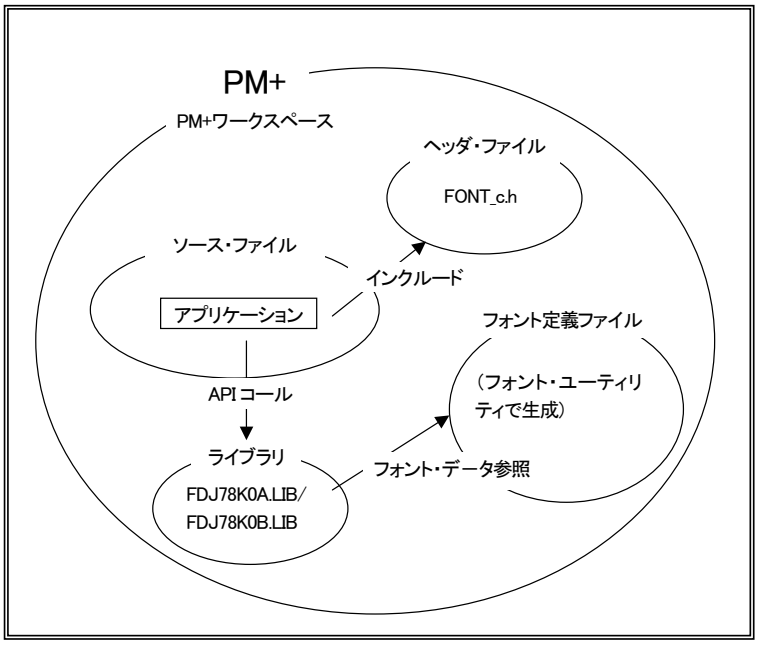
漢字表示デモンストレーション用ボードの入手に関する情報については，下記のボード設計情報をご参照ください。

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

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　ライブラリ仕様

この章では，フォント・アクセス・ライブラリの仕様について説明します。

2. 1　フォント・データ仕様

本ライブラリで使用できるフォント・データの構成は次のとおりです。

・パラメータ定義部：　32バイト固定
・テーブル定義部：　テーブル形式により，0バイト～文字コード範囲×2バイト
・フォント・データ定義部：　文字数×1文字データ・サイズ

各定義部の詳細は，「フォント・ユーティリティ　ユーザーズ・マニュアル（U19527）」を参照してください。

2. 2　関数仕様（API仕様）

本ライブラリに用意されている関数について説明します。以下，関数の詳細説明の読み方について解説します。

【機能】

各関数の機能の詳細を示します。

【ヘッダ・ファイル】

各関数を使用する場合に必要となるヘッダ・ファイル名を示します。

【関数プロトタイプ】

各関数の指定形式を示します。

【説明】

各関数の処理の詳細を示します。

2. 2. 1　関数一覧

本ライブラリで利用できる関数の一覧を表2－1に示します。

表2－1　関数一覧

名称	機能
DEFINE_FONT_TYPE	使用するフォント・データ名を宣言するマクロ関数
uFont_get_next_code	文字列から次の文字コードを得る
uFont_set_type	以下の関数で操作するフォント・データ名を設定する
uFont_get_parameter	フォントのパラメータを得るマクロ関数
uFont_get_font_addr	フォントの位置を得る

表2－2　マクロ関数を除く78K0非バンク用（FDJ78K0A.LIB）リソース一覧（数値は暫定値）

名称	コード・サイズ	RAM消費量	スタック消費量	最大実行クロック数
uFont_get_next_code	60 バイト	0	0	148 クロック
uFont_set_type	4バイト	2	0	18 クロック
uFont_get_font_addr	257 バイト	0	12	265 クロック[注1] 339 クロック[注2] 2497 クロック[注3]

注1.　総フォント・データ・サイズ32KB，テーブル無しの場合

注2.　総フォント・データ・サイズ32KB，ルックアップ・テーブルの場合

注3.　総フォント・データ・サイズ32KB，文字数1000，サーチ・テーブルの場合

表2－3　マクロ関数を除く78K0バンク用（FDJ78K0B.LIB）リソース一覧（数値は暫定値）

名称	コード・サイズ	RAM消費量	スタック消費量	最大実行クロック数
uFont_get_next_code	60 バイト	0	0	148 クロック
uFont_set_type	4バイト	2	0	18 クロック
uFont_get_font_addr	316 バイト	0	12	548 クロック[注1] 636 クロック[注2] 3018 クロック[注3]

注1.　総フォント・データ・サイズ96KB，テーブル無しの場合

注2.　総フォント・データ・サイズ96KB，ルックアップ・テーブルの場合

注3.　総フォント・データ・サイズ64KB，文字数2000，サーチ・テーブルの場合

2. 2. 2　DEFINE_FONT_TYPE（フォント・データ名を宣言する）

【機能】

使用するフォント・データ名を宣言します。

【ヘッダ・ファイル】

FONT_c.h

【関数プロトタイプ】

void DEFINE_FONT_TYPE(*gFont_data)

マクロ関数です。

関数名	引数	返り値
DEFINE_FONT_TYPE	フォント・データ名	なし

【説明】

フォント・データをリンクするために使用します。

2. 2. 3　uFont_get_next_code（文字列から文字コードを得る）

【機能】

全角／半角の混在した日本語文字列から1文字を取り出します。

【ヘッダ・ファイル】

FONT_c.h

【関数プロトタイプ】

unsigned short uFont_get_next_code (char *text) ;

関数名	引数	返り値
uFont_get_next_code	text： 文字列上の現在位置 （文字コードの取り出しアドレス）	半角の場合： 1バイト・コード（255以下） 全角の場合： JISコード（上位バイトが区に対応。ただし， 第4水準には対応していません。）

【説明】

textが指す全角／半角の混在した日本語文字列から1文字を取り出します。

全角はシフトJISコードを使用してください。

全角と判定されるのは，次の場合です。

- 第1バイト：81H～9FH, E0H～FCHの範囲[注]
- 第2バイト：40H～7EH, 80H～FCHの範囲

注　F0H～FCH（第4水準コード）は全角と判定されますが，返り値が，JISコードになりません。

2. 2. 4　uFont_set_type（対象となるフォント・データ名を設定する）

【機能】

uFont_get_parameter, uFont_get_font_addrで対象となるフォント・データ名を設定します。

【ヘッダ・ファイル】

FONT_c.h

【関数プロトタイプ】

void uFont_set_type (const struct FONT_TYPE_HEADER *gFont_data) ;

関数名	引数	返り値
uFont_set_type	gFont_data： フォント・データ名	なし

【説明】

複数のフォント・データをリンクする場合，uFont_get_parameter, uFont_get_font_addrの使用の前に，本関数で対象ファイルを設定します。

対象フォント・データが1つしかない場合は，最初に1回だけ設定すれば，繰り返し再設定する必要はありません。

2. 2. 5　uFont_get_parameter（フォントのパラメータを得る）

【機能】

フォントのボディ・サイズを得るのに使用します。

【ヘッダ・ファイル】

FONT_c.h

【関数プロトタイプ】

unsigned char uFont_get_parameter (パラメータ名) ;

マクロ関数です。

関数名	引数	返り値
uFont_get_parameter	パラメータ名： SFBH：水平ドット・サイズを得たい場合 SFBV：垂直ドット・サイズを得たい場合	ドット・サイズ

【説明】

表示位置の制御などのために，フォントのボディ・サイズを得ることができます。

2. 2. 6　uFont_get_font_addr（文字コードからフォント・データの開始位置を得る）

【機能】

文字コードから，その文字に対応したフォント・データの開始位置を取得します。

【ヘッダ・ファイル】

FONT_c.h

【関数プロトタイプ】

char* uFont_get_font_addr (unsigned short i) ;

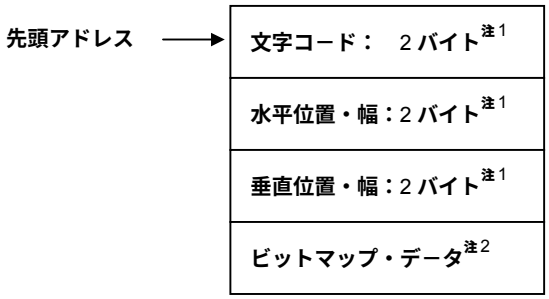
関数名	引数	返り値
uFont_get_font_addr	i： フォント・データを取り出す文字コード （uFont_get_next_codeで得た文字コード）	実装されている文字の場合： フォント・データの開始位置 未実装文字の場合： 0

注意　FDJ78K0Bの場合，本関数の実行によりメモリ・バンク選択レジスタ（BANK）を書き換えます。元の内容が必要であれば，本関数を呼び出す前に保存してください。

【説明】

iで指定される文字コードのフォント・データの先頭アドレスを返します。

フォント・データの並びについては，次のようになります。



注1.　フォント・ユーティリティの条件指定により省略可能です。

注2.　サイズとビットのスキャン順序はフォント・ユーティリティの条件指定に依存します。

第3章　アプリケーション実装例

この章では，フォント・アクセス・ライブラリを使った漢字表示方法について，サンプル・プログラムPrint_c.cを例に説明します。Print_c.cはダウンロード・ファイルのSAMPLEフォルダに格納されています。

3. 1　アプリケーションの概要

サンプル・プログラムPrint_c.cは，全角／半角混在の表示文字列をビットマップ・フォントに展開してビットマップ表示メモリに格納するプログラムです。以下，Print_c.cの中の主要な3つの関数について説明します。

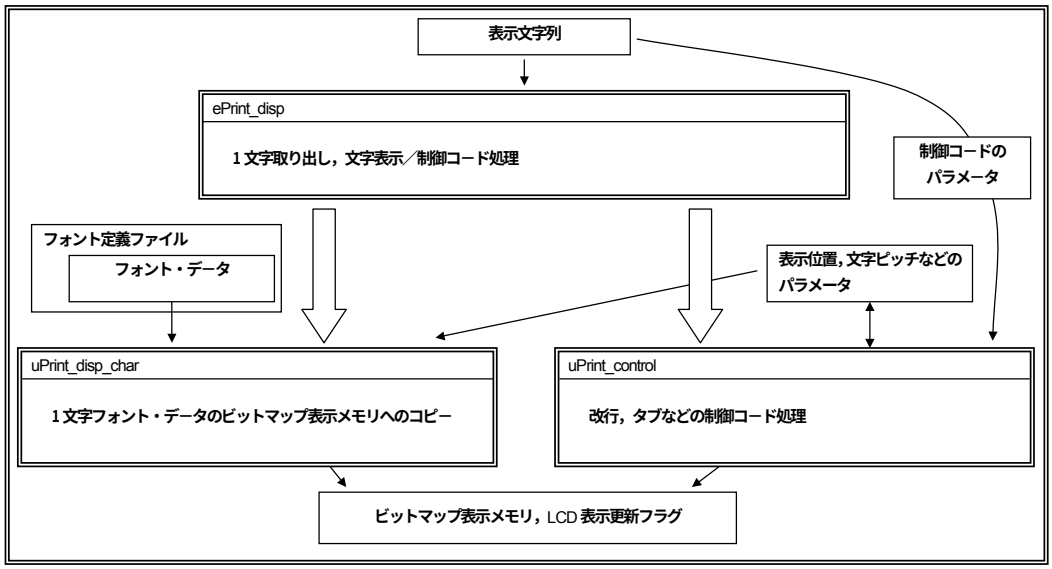
① ePrint_disp（詳細3. 2項）
文字列から1文字取り出して，文字表示または制御コード処理を行います。

② uPrint_control（詳細3. 3項）
改行，タブなどの制御コード処理を行います。

③ uPrint_disp_char（詳細3. 4項）
1文字のフォント・データをビットマップ表示メモリに転送します。

なお，Print_c.cの中の上記以外の部分は，ホスト・マシンからのコマンド処理であり，本資料では説明を省略します。
また，ビットマップ表示メモリのデータをLCDへ転送する部分は別プログラム（LCD_c.c）になっています。本資料ではLCD_c.cの説明は省略します。

図3－1　漢字表示アプリケーション処理のブロック図



3. 1. 1　フォント・データの生成条件

サンプル・プログラムPrint_c.cが取り扱うフォント・データは，次のとおりです。

(1) フォント・データの生成条件

フォント・ユーティリティを使い，次の条件で生成されたフォント・データが使用できます。

・フォント出力ビット・スキャン方向指定：　垂直（垂直ドット数は8ドットの倍数）
・フォント出力ファースト・ビット位置指定：　LSB
・フォント出力バイト順序：　垂直→水平
・補助出力：　フォント出力水平位置・幅出力指定のみをチェック

その他の条件はメモリに実装可能な範囲で選択します。参考として同梱フォント定義ファイルの生成条件を次に示します。

表3－1　フォント定義ファイル生成条件一覧

項目	FA107LVH	FH014LVH	FK014LVH	FK024LVH
入力：フォント・ファイル	UPD7228PLUS.fon	A14.fon	K14-2004-1.fon	jiskan24-2003-1.fon
入力：バイト	1バイトx8ドット	1バイトx14ドット	2バイトx14ドット	3バイトx24ドット
テーブル	無し	無し	サーチ	サーチ
出力文字選択：範囲	半角32～255点	半角32～127点	全角デフォルト	全角デフォルト
出力文字選択：ファイル	無し	無し	SEL14_2800.csv	SEL24_5.csv
フォント出力：ドット範囲	水平5，垂直8	水平7，垂直16	水平14，垂直16	水平24，垂直24
フォント出力：オフセット	無し	垂直1	垂直1	無し
配置指定	8000H	8700H	A000H	1FE66H

(2) フォント･データの並び

フォント・データは次の順番で並んでいます。（「図3－2　フォント・データの並びの例」を参照してください。）

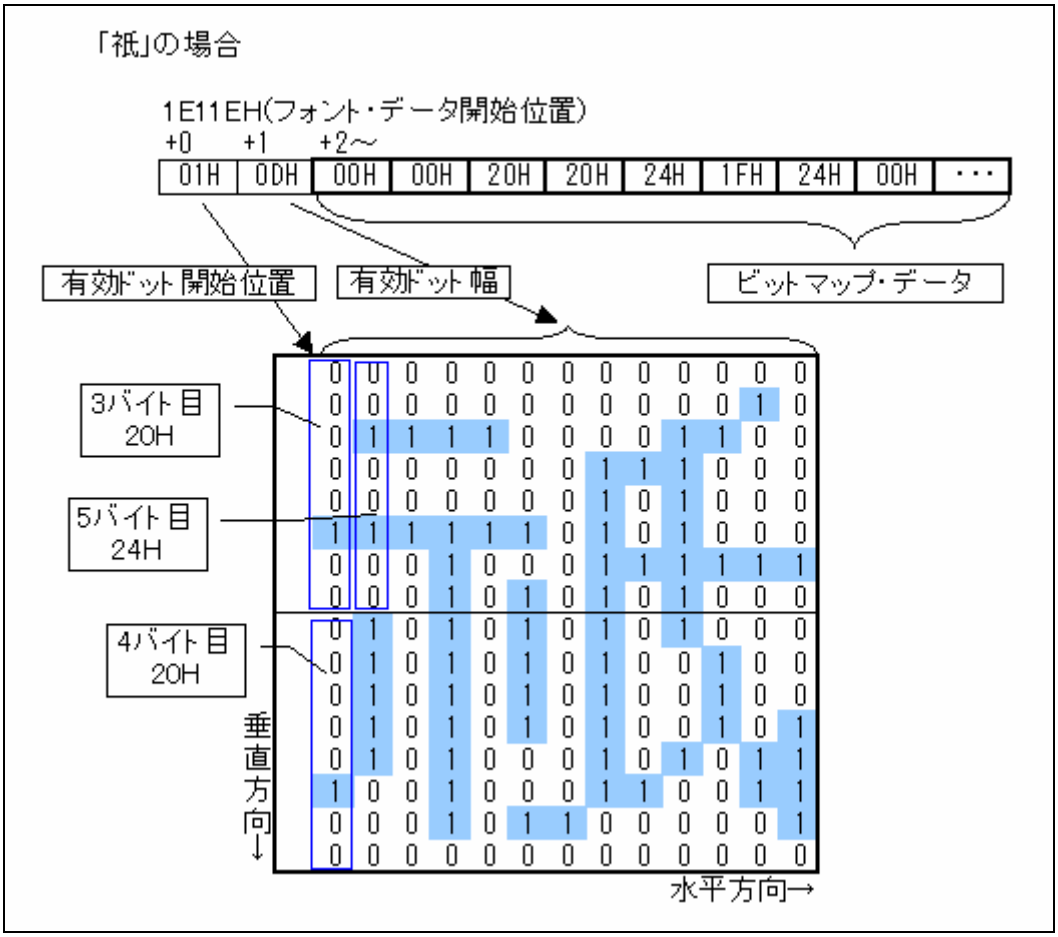
相対位置　内容

＋0 ：水平有効ドット開始位置
　ビットマップのボディに対し，実際に文字が始まる水平ドット位置を示します。
　プロポーショナル制御のために使用します。

＋1 ：水平有効ドット幅
　ビットマップのボディに対し，実際に文字が存在する水平幅を示します。
　プロポーショナル制御のために使用します。

＋2～：ビットマップ・データ
　垂直，水平のドット順に並びます。
　LSBファーストです。
　垂直ドット数は8ドット単位です。

図3－2　フォント・データの並びの例（14×16ドットの例）



ビットマップ・データは，垂直方向の1列が上～下8ドット単位で，上側バイト→下側バイトの順に設定されます。さらに，このバイトで構成される列データは，水平方向の左側の列から右側の列に向かって設定されます。

・列構成バイト数 = 垂直ドット数／8

・ビットマップ構成バイト数 = 列構成バイト数×水平ドット数

3. 1. 2　ビットマップ表示メモリの構造

ビットマップ表示メモリは，LCD表示面全体の表示ドット情報を保持します。LCD表示面ドット数はLCD_c.hの中のnLCD_xs（水平）とnLCD_ys（垂直）で定義されています。ドット数に対応して，ビットマップ表示メモリは次のようなバイト領域となっています。

ビットマップ表示メモリのバイト数 =（LCD表示面垂直ドット数／8）×LCD表示面水平ドット数

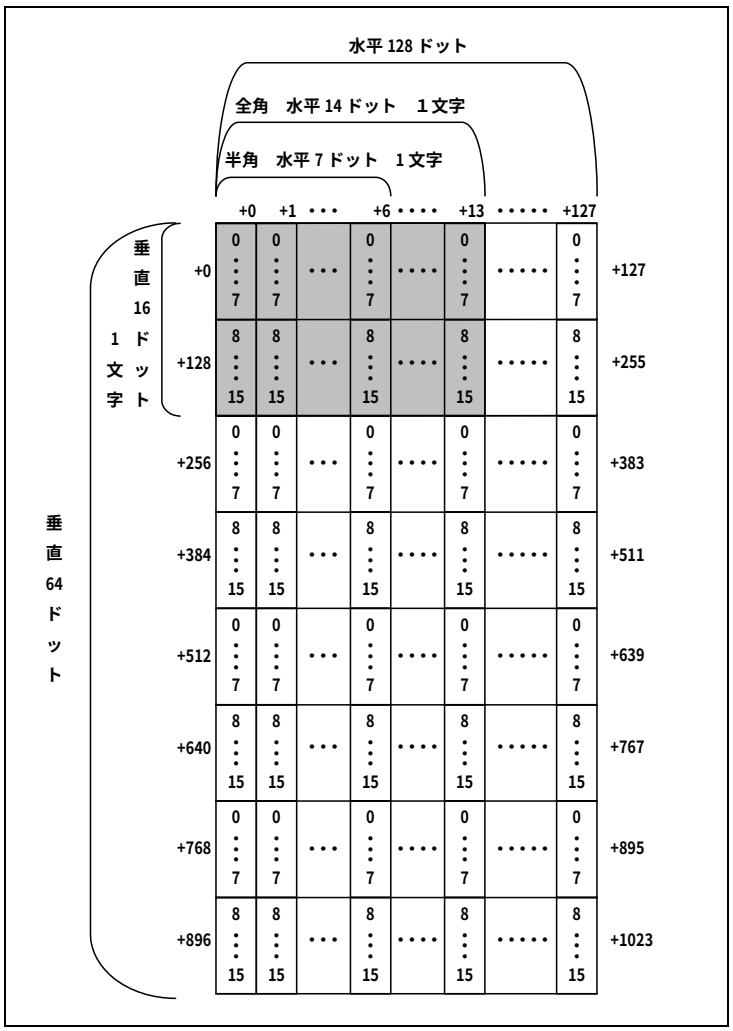
バイト数は，nLCD_textとしてLCD_c.hに定義されています。領域はバイト型配列で定義されており，水平（X）から垂直（Y）方向にデータが格納されます。

次の図は表示面が128×64ドットの例で，ビットマップ表示メモリの構造をLCD表示面に合わせ，構成バイトの並びで示しています。枠外＋0～＋1023の数字はビットマップ表示メモリでの相対バイト位置を示します。

図では全角14×14フォントでの1文字占有範囲の目安を灰色で示しています。実際には，文字種・ピッチ指定などにより占有範囲の位置は変化します。

サンプル・プログラムではビットマップ表示メモリは配列名gLCD.textとしてLCD_c.hで定義されています。

図3－3　ビットマップ表示メモリの構造



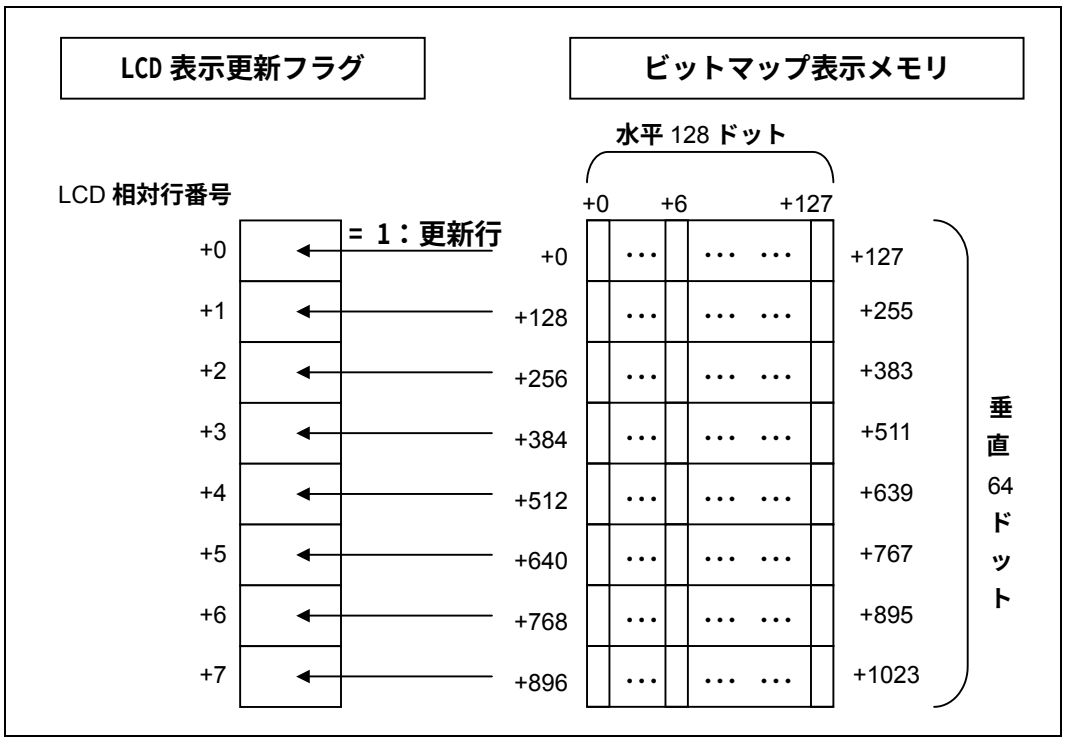
3. 1. 3　LCD表示更新フラグ

LCD表示更新フラグはフォントの展開には直接関係しませんが，変更したビットマップ表示メモリ行をLCD表示プログラム（LCD_c.c）に通知することにより，LCDへの転送負荷を低減させる役割があります。LCD表示プログラム（LCD_c.c）は，ビットマップ表示メモリの中で表示更新すべき行（8ドット・ライン単位）だけを選択してLCDへの転送を行います。

LCD相対行番号は縦8ドットを1行とした行番号です。

サンプル・プログラムではLCD表示更新フラグは配列名gLCD.dreqとしてLCD_c.hで定義されています。

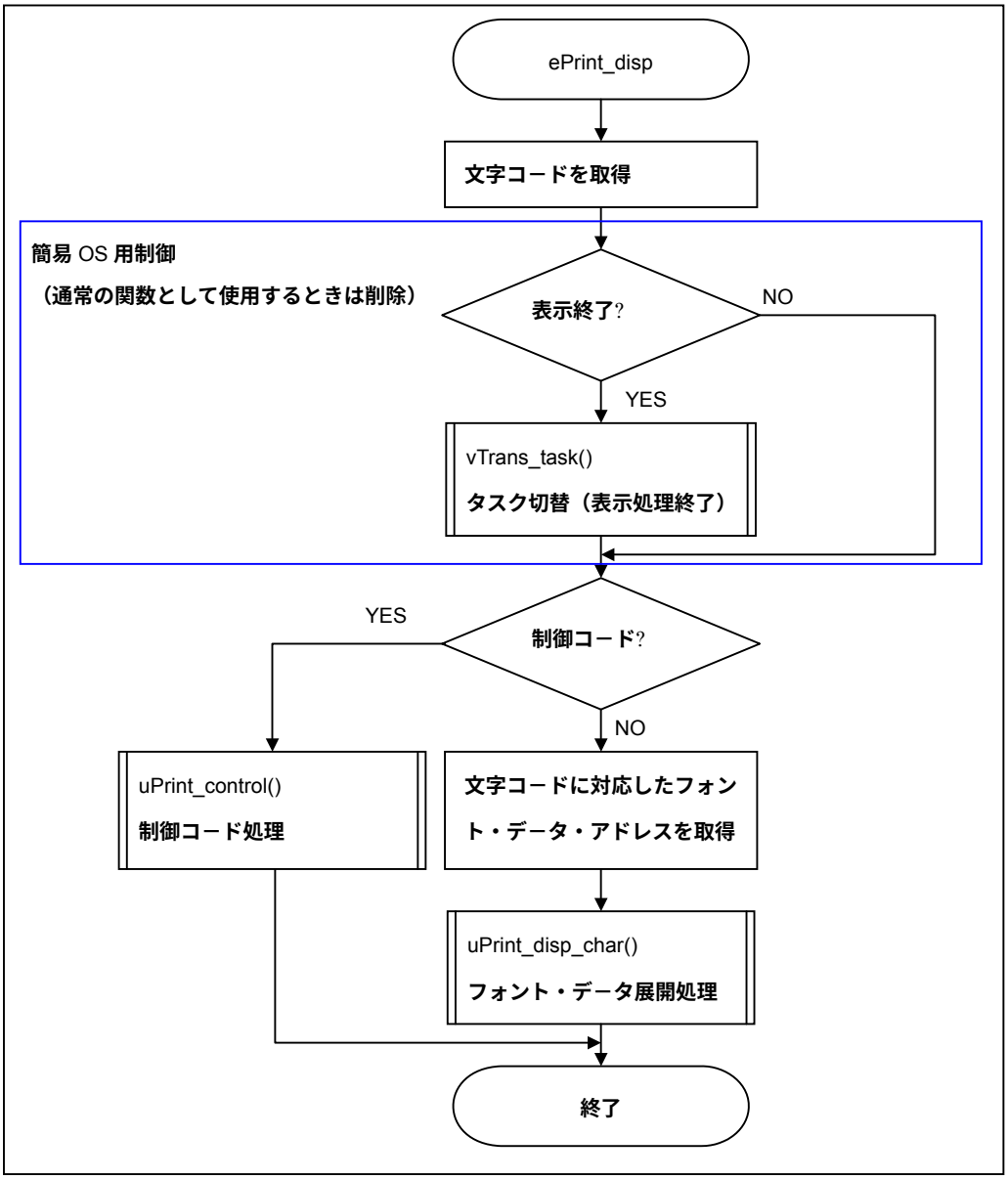
図3－4　LCD表示更新フラグ



3. 2　文字表示（ePrint_disp）

ePrint_disp関数は，1文字分のコードを取り出して全角／半角／制御コードのいずれであるかを判定し，フォント・データ展開関数（uPrint_disp_char）または制御コード処理関数（uPrint_control）により処理を行います。なお本関数には，1行だけ簡易OS用の制御が含まれていますが，その行を削除することにより通常の関数として呼び出すことができます。

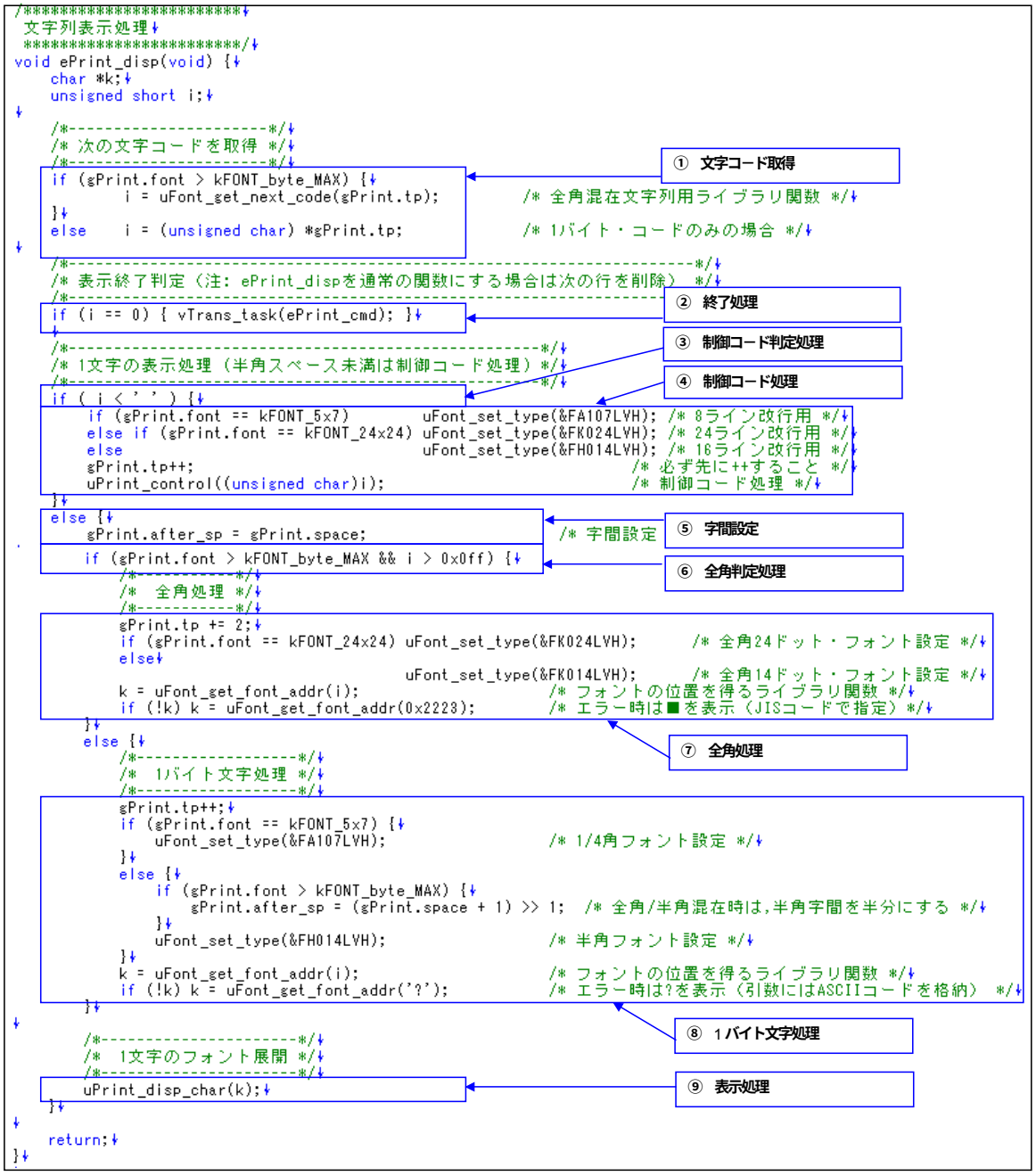
（1）概略フロー



(2) 使用領域

定義場所	定義方法	型	変数名	意味
Print_c.h	構造体	char*	gPrint.tp	処理すべき文字へのポインタ
		char	gPrint.font	文字をどのフォントで表示するかの指定値 ホスト・マシンからのコマンドで格納されます。 ・KFONT_5x7　（ = 0）：1/4角 ・KFONT_5x14　（ = 1）：半角 ・KFONT_14x14（ = 2）：14×14ドット・フォント ・KFONT_24x24（ = 4）：24×24ドット・フォント
		char	gPrint.space	ホスト・マシンから設定された字間ドット数
		unsigned char	gPrint.after_sp	補正後の字間ドット数を格納するワーク変数
	#define		kFONT_byte_MAX	= 1：1/4角，半角などの1バイト・コードの最大値

(3) プログラム・リスト




```
/************************
 文字列表示処理
 ************************/
void ePrint_disp(void) {
    char *k;
    unsigned short i;

    /*----------------------*/
    /* 次の文字コードを取得 */
    /*----------------------*/
    if (gPrint.font > kFONT_byte_MAX) {
          i = uFont_get_next_code(gPrint.tp);       /* 全角混在文字列用ライブラリ関数 */
    }
    else    i = (unsigned char) *gPrint.tp;         /* 1バイト・コードのみの場合 */

    /*-------------------------------------------------------------*/
    /* 表示終了判定（注: ePrint_dispを通常の関数にする場合は次の行を削除） */
    /*-------------------------------------------------------------*/
    if (i == 0) { vTrans_task(ePrint_cmd); }

    /*-----------------------------------------------*/
    /* 1文字の表示処理（半角スペース未満は制御コード処理） */
    /*-----------------------------------------------*/
    if ( i < ' ' ) {
        if (gPrint.font == kFONT_5x7)        uFont_set_type(&FA107LVH); /* 8ライン改行用 */
        else if (gPrint.font == kFONT_24x24) uFont_set_type(&FK024LVH); /* 24ライン改行用 */
        else                                 uFont_set_type(&FH014LVH); /* 16ライン改行用 */
        gPrint.tp++;                                                    /* 必ず先に++すること */
        uPrint_control((unsigned char)i);                               /* 制御コード処理 */
    }
    else {
        gPrint.after_sp = gPrint.space;                                 /* 字間設定 */
        if (gPrint.font > kFONT_byte_MAX && i > 0x0ff) {
            /*----------*/
            /*  全角処理 */
            /*----------*/
            gPrint.tp += 2;
            if (gPrint.font == kFONT_24x24) uFont_set_type(&FK024LVH);      /* 全角24ドット・フォント設定 */
            else
                                            uFont_set_type(&FK014LVH);      /* 全角14ドット・フォント設定 */
            k = uFont_get_font_addr(i);                                     /* フォントの位置を得るライブラリ関数 */
            if (!k) k = uFont_get_font_addr(0x2223);                        /* エラー時は■を表示（JISコードで指定） */
        }
        else {
            /*------------------*/
            /*  1バイト文字処理 */
            /*------------------*/
            gPrint.tp++;
            if (gPrint.font == kFONT_5x7) {
                uFont_set_type(&FA107LVH);                                  /* 1/4角フォント設定 */
            }
            else {
                if (gPrint.font > kFONT_byte_MAX) {
                    gPrint.after_sp = (gPrint.space + 1) >> 1;  /* 全角/半角混在時は,半角字間を半分にする */
                }
                uFont_set_type(&FH014LVH);                                  /* 半角フォント設定 */
            }
            k = uFont_get_font_addr(i);                                     /* フォントの位置を得るライブラリ関数 */
            if (!k) k = uFont_get_font_addr('?');                           /* エラー時は?を表示（引数にはASCIIコードを格納） */
        }

        /*----------------------*/
        /*  1文字のフォント展開 */
        /*----------------------*/
        uPrint_disp_char(k);
    }

    return;
}
```

(4) 詳細説明

①　文字コード取得

表示する文字コードを取得します。

・全角／半角混在時は，uFont_get_next_code関数を呼び出して，文字コードを取得します。

・1バイト・コードだけの場合は，文字ポインタを使い文字コードを取り出します。

②　終了処理

取り出した文字コードが0の場合は，文字列終了となります。

簡易OSを使用する場合に，コマンド処理タスクへ制御を戻します。

簡易OSを使用しない場合には，当該行を削除するか，コメントにします。この場合，本関数を次のように呼び出します。

```
while (*gPrint.tp) ePrint_disp();
```

③　制御コード判定処理

取り出した文字コードが，制御コードか判定します。

・半角スペース（20H）未満の場合は，制御コード処理を行います。

・半角スペース（20H）以上の場合は，表示文字処理を行います。

④　制御コード処理

制御コード処理を行います。

・フォント種別に応じた表示フォントをuFont_set_type関数を呼び出して選択します。

・文字ポインタを更新します。

　制御コードによっては，引き続くパラメータを，文字列バッファから取り出す場合がありますので，制御コード処理関数を呼び出す前に，ポインタを更新します。

・uPrint_control関数を呼び出して制御コードに応じた処理を行います。

・本関数を終了します。

⑤　字間設定

表示文字コードの場合に，字間スペース・パラメータを設定します。

⑥　全角判定処理

漢字フォントが指定されていて，文字コードが全角の場合は，全角フォント処理を行います。そうでない場合には，1バイト文字処理を行います。

⑦　全角フォント処理

全角フォント・データ・アドレスを取得します。

・文字ポインタの更新（2バイト分）

・表示フォント・データ選択

　フォント種別に応じて，uFont_set_type関数を呼び出し，表示フォント・データを選択します。

・フォント・データのアドレス取得

　uFont_get_font_addr関数を呼び出して，フォント・データのアドレスを取得します。フォント・データのない文字コードの場合は，フォント・データ・アドレスとして，0が戻されま

す。この場合は，表示用フォント・データとして，“■”を使用します。

⑧ 1バイト文字処理

1バイト文字のフォント・データ・アドレスを取得します。

・文字ポインタの更新（1バイト分）

・フォント種別に応じた表示フォント・データの選択

フォント種別に応じて，uFont_set_type関数を呼び出し，表示フォント・データを選択します。

・文字間スペースの調整

全角／半角混在文字列における半角フォントの場合は，文字間スペースを設定値の半分（小数部切上げ）にします。

・フォント・データのアドレス取得

uFont_get_font_addr関数を呼び出して，フォント・データのアドレスを取得します。フォント・データのない文字コードの場合は，フォント・データ・アドレスとして，0が戻されます。この場合は，表示用フォント・データとして，“？”を使用します。

⑨ 表示処理

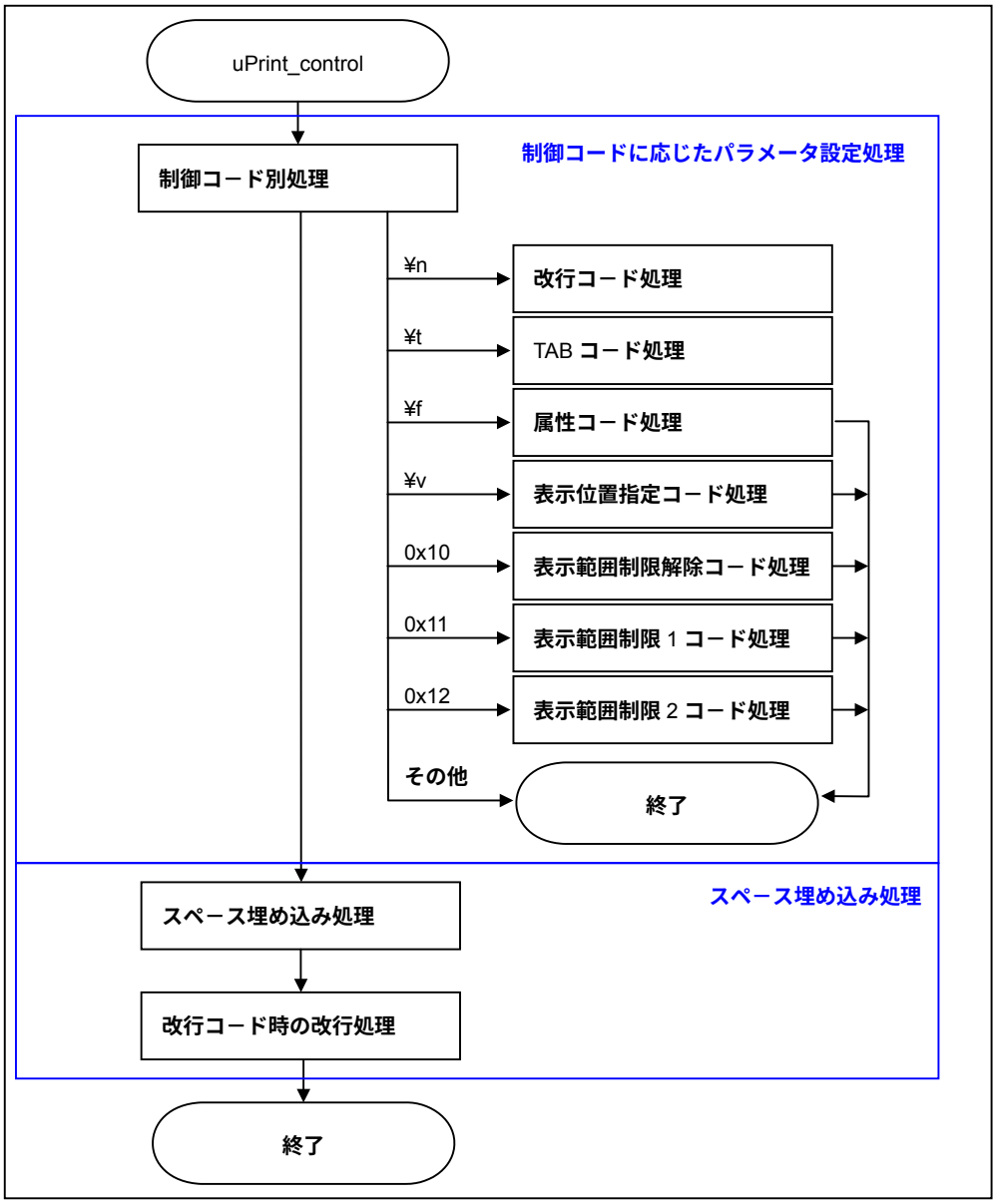
uPrint_disp_char関数を呼び出して，フォント・データを表示位置のビットマップ表示メモリに展開して，LCDに表示させます。

注意 FDJ78K0Bの場合，uFont_get_font_addr関数の実行によりメモリ・バンク選択レジスタ（BANK）を書き換えます。元の内容が必要であれば，uFont_get_font_addr関数を呼び出す前に保存してください。また，uFont_get_font_addr関数の実行後は，対象フォント・データが格納されたバンクを指し示すため，フォント・データをアクセスする前にバンク操作を行う場合は，いったんBANKの内容を保存して下さい。

3. 3　制御コード処理（uPrint_control）

制御コードの処理を行います。本関数は，制御コードに応じたパラメータ設定処理部分とスペース埋め込処理部分からなります。（制御コードの詳細は「5. 4　コマンドの書式」を参照してください。）

（1）概略フロー



(2) 使用領域

定義場所	定義方法	型	変数名	意味
Print_c.h	構造体	char*	gPrint.tp	表示文字ポインタ
		char	gPrint.space	字間ドット数の指定値 属性バイトのbit3-0で指定されます。
		char	gPrint.pitch	ピッチ（固定ピッチ／プロポーショナル）の指定値 属性バイトのbit4で指定されます。
		char	gPrint.font	文字をどのフォントで表示するかの指定値 属性バイトのbit7-5で指定されます。 ・KFONT_5x7　（ = 0）：1/4角 ・KFONT_5x14　（ = 1）：半角 ・KFONT_14x14（ = 2）：14×14ドット・フォント ・KFONT_24x24（ = 4）：24×24ドット・フォント
		unsigned char	gPrint.xs1 gPrint.ys1 gPrint.xe1 gPrint.ye1	表示範囲制限領域1 ・gPrint.xs1：水平開始位置XS1 ・gPrint.ys1：垂直開始位置YS1 ・gPrint.xe1：水平終了位置XE1 ・gPrint.ye1：垂直終了位置YE1 ホスト・マシンからのコマンドで設定されます。
		unsigned char	gPrint.xs2 gPrint.ys2 gPrint.xe2 gPrint.ye2	表示範囲制限領域2 ・gPrint.xs2：水平開始位置XS2 ・gPrint.ys2：垂直開始位置YS2 ・gPrint.xe2：水平終了位置XE2 ・gPrint.ye2：垂直終了位置YE2 ホスト・マシンからのコマンドで設定されます。
		unsigned char	gPrint.xs gPrint.ys gPrint.xe gPrint.ye	現在選択されている表示範囲制限領域 ・gPrint.xs：水平開始位置XS ・gPrint.ys：垂直開始位置YS ・gPrint.xe：水平終了位置XE ・gPrint.ye：垂直終了位置YE 表示範囲指定制御コードで，対応した表示範囲制限領域値が設定されます。 初期状態および，表示範囲制限解除制御コードで，LCD全体が表示領域値として設定されます。
		unsigned char	gPrint.x	水平表示位置X
		unsigned char	gPrint.y	垂直表示位置Y
LCD_c.h	#define		nLCD_xm	= 128：ビットマップ表示メモリの横ドット数
			nLCD_ym	= 64：ビットマップ表示メモリの縦ドット数
			KDISP_REQUEST	= 1：表示更新要求あり
	構造体	unsigned char	gLCD.dreq	LCD表示更新フラグ配列（1：更新あり）
		unsigned char	gLCD.text	ビットマップ表示メモリ

FONT_c.h	構造体	unsigned char	SFBV	フォントの垂直ボディ・サイズ領域 uFont_get_parameter関数を使用してフォントの垂直ボディ・サイズを取り出すのに使用します。

(3) プログラム・リスト1

```
/*************************************
 制御コード処理
    引数：制御コード（0～31）
    戻値：無し
 *************************************/
void uPrint_control(char c) {
    unsigned char  i, j, ie, ys, ye;
    unsigned short y, position;

    /*-----------------*/
    /* 制御の種類別処理 */
    /*-----------------*/
    switch (c) {
        case '¥n': /* 行末までスペースで埋めて改行 */
                   ie = gPrint.xe;
                   break;
        case '¥t': /* 指定カラムまでスペースで埋める */
                   ie = (unsigned char) *gPrint.tp++;
                   if (ie > gPrint.xe) ie = gPrint.xe;
                   break;
        case '¥f': /* 属性を変更する */
                   i = (unsigned char) *gPrint.tp++;
                   gPrint.space = i & 0x0f;
                   gPrint.pitch = (i >> 4) & 1;
                   gPrint.font = i >> 5;
                   return;
        case '¥v': /* 描画位置を変更する */
                   gPrint.x = (unsigned char) *gPrint.tp++;
                   gPrint.y = (unsigned char) *gPrint.tp++;
                   return;
        case 0x10: /* 描画許可範囲を画面全体にする */
                   gPrint.xs = 0;
                   gPrint.ys = 0;
                   gPrint.xe = nLCD_xm -1;
                   gPrint.ye = nLCD_ym -1;
                   return;
        case 0x11: /* 描画許可範囲をパラメータ1の範囲にする */
                   gPrint.xs = gPrint.xs1;
                   gPrint.ys = gPrint.ys1;
                   gPrint.xe = gPrint.xe1;
                   gPrint.ye = gPrint.ye1;
                   return;
        case 0x12: /* 描画許可範囲をパラメータ2の範囲にする */
                   gPrint.xs = gPrint.xs2;
                   gPrint.ys = gPrint.ys2;
                   gPrint.xe = gPrint.xe2;
                   gPrint.ye = gPrint.ye2;
                   return;
        default:   return;
    }
```

① ¥n: 改行コード処理

② ¥t: タブ・コード処理

③ ¥f: 属性コード処理

④ ¥v: 表示位置指定コード処理

⑤ 10H: 表示範囲制限解除コード処理

⑥ 11H: 表示範囲制限１指定コード処理

⑦ 12H: 表示範囲制限２指定コード処理

⑧ 未定義制御コード処理

（つづく）

(4) 詳細説明1

制御コードに応じて，各種パラメータ値を設定します。

①　‘¥n’：改行コード処理

行末までスペースを埋め込むため，スペース埋め込み終了位置パラメータを行末に設定します。（スペース埋め込み処理の前準備です。）

②　‘¥t’：タブ・コード処理

指定カラムまでスペースを埋め込むため，スペース埋め込み終了位置パラメータに，制御コードに続けて入力されている指定位置を設定します。（スペース埋め込み処理の前準備です。）

③　‘¥f’：属性コード処理

制御コードに続けて入力されている属性値を，パラメータに設定して，本関数を終了します。

・字間ドット数：属性バイトのbit3－0

・ピッチ指定：　属性バイトのbit4

・フォント指定：属性バイトのbit7－5

④　‘¥v’：表示位置指定コード処理

制御コードに続けて入力されている表示位置（X, Y**）データを，表示位置パラメータに設定して，本関数を終了します**

⑤　10H**：表示範囲制限解除コード処理**

表示範囲が制限されている状態を解除するために，表示範囲パラメータをLCD**全面に設定して，本関数を終了します。**

⑥　11H**：表示範囲制限**1**指定コード処理**

表示範囲パラメータに，表示範囲1**を設定して，本関数を終了します。**

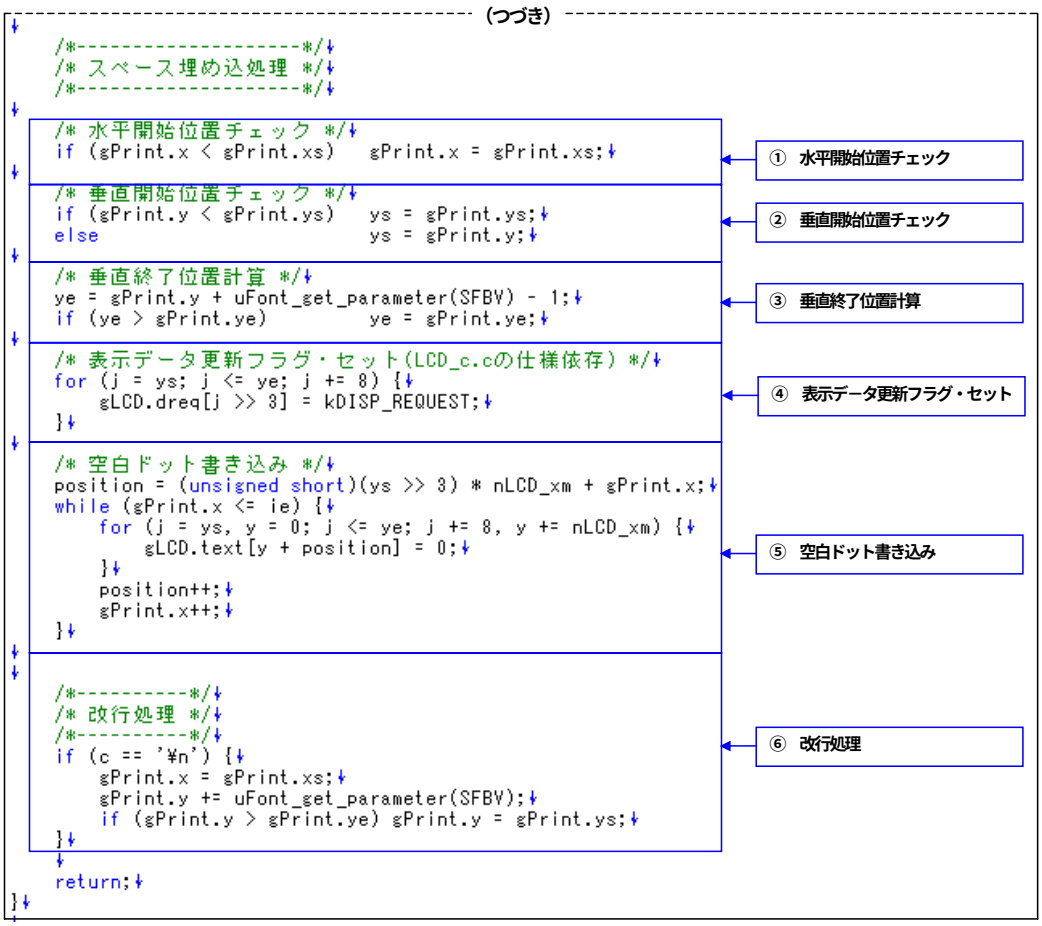
⑦　12H**：表示範囲制限**2**指定コード処理**

表示範囲パラメータに，表示範囲2**を設定して，本関数を終了します。**

⑧　未定義制御コード処理

何もせずに，本関数を終了します。

(5) プログラム・リスト2



(6) 詳細処理2

スペースを埋め込む位置を計算して，必要な分のスペースとして，空白ドットをビットマップ表示メモリに書き込みます。スペースを埋め込んだ行の表示更新フラグをセットします。さらに，改行コードの場合は，表示位置を次行の先頭へ移動します。

①　水平開始位置チェック

水平表示位置が，表示範囲左端より左側にある場合は，水平表示位置を，表示範囲左端に補正します。この水平表示位置がスペース埋め込みの水平開始位置となります。

②　垂直開始位置チェック

垂直表示位置が，表示範囲上端より上側にある場合は，スペース埋め込み垂直開始位置を，表示範囲上端に補正します。そうでない場合は，垂直表示位置をスペース埋め込みの垂直開始位置とします。

③　垂直終了位置計算

スペース埋め込み垂直終了位置 ＝ 垂直表示位置＋フォント垂直ドット数－1

スペース埋め込み垂直終了位置が表示範囲下端より下側にある場合は，垂直終了位置を表示範囲下端に補正します。

④　表示データ更新フラグ・セット

表示を更新するLCD相対行全部のLCD表示更新フラグを設定します。

・LCD相対行番号 = 垂直表示位置／8

⑤　空白ドット書き込み

空白ドットをビットマップ表示メモリにセットします。

・ビットマップ表示メモリの格納先を計算

格納位置 ＝ （LCD相対行番号×LCD表示水平ドット数）＋水平表示位置

・空白ドットの書き込み

ビットマップ表示メモリに，空白ドットとして，0を書き込みます。書き込みは，垂直8ドット単位での行数分埋め込みを，水平開始位置から水平終了位置まで，列単位で繰り返します。

1列分書き込みごとに，水平表示位置を更新します。

⑥　改行処理

制御コードが改行コードの場合は，次行の先頭に表示位置を移動します。

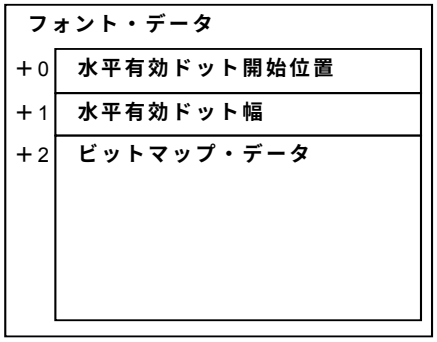
・水平表示位置：水平表示範囲左端

・垂直表示位置：1行分（フォント垂直ボディ・サイズ分）進めます。表示範囲下端を超えた場合は，表示範囲上端に戻します。

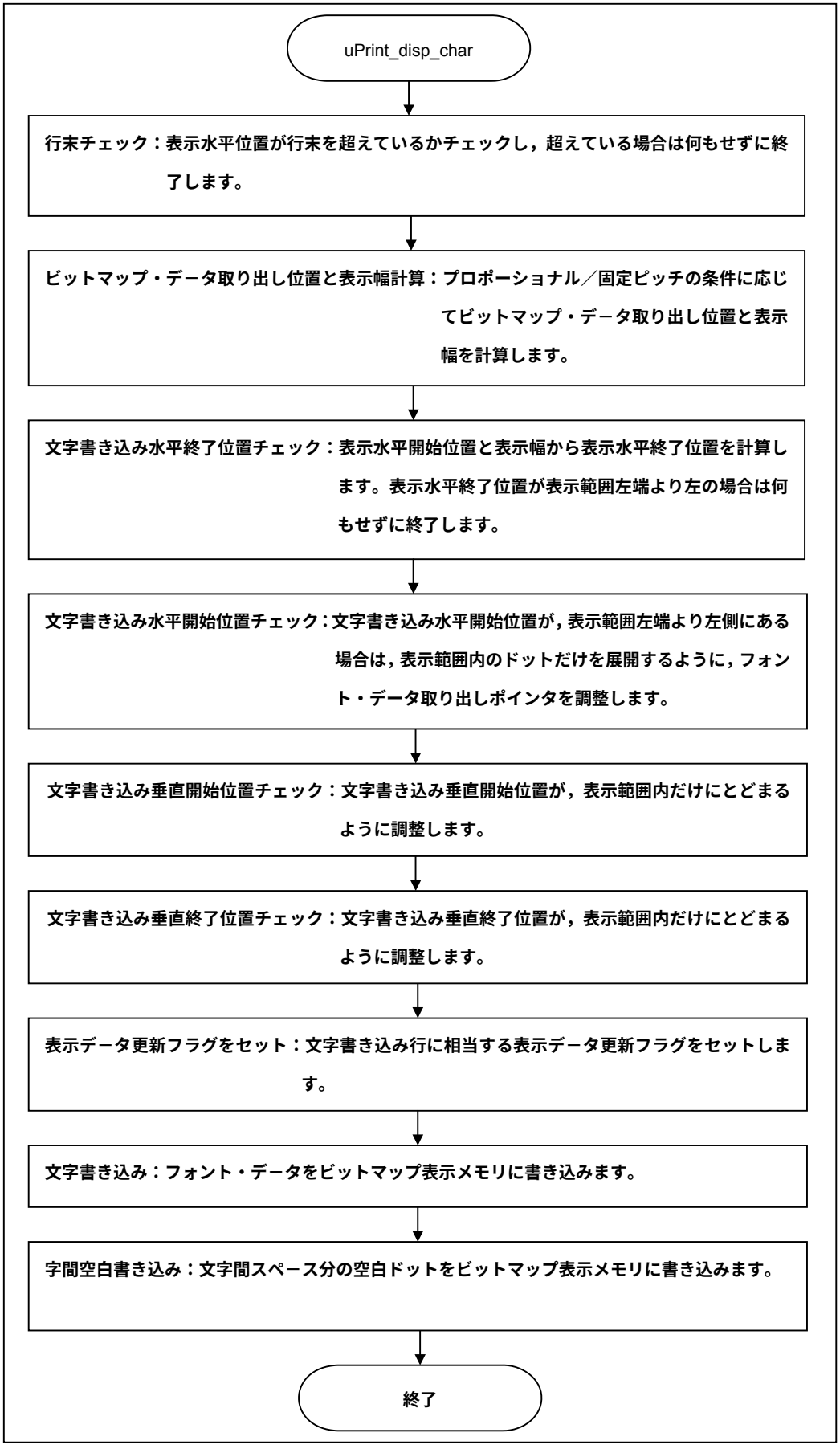
3. 4　フォント・データ展開処理（uPrint_disp_char）

フォント・データをビットマップ表示メモリに展開します。引数は次の図のフォント・データの先頭位置となります。（フォント・データの詳細は，「3. 1. 1　フォント・データの生成条件」を参照してください。）

図3－5　フォント・データ構造



(1) 概略フロー


uPrint_disp_char
行末チェック：表示水平位置が行末を超えているかチェックし，超えている場合は何もせずに終了します。
ビットマップ・データ取り出し位置と表示幅計算：プロポーショナル／固定ピッチの条件に応じてビットマップ・データ取り出し位置と表示幅を計算します。
文字書き込み水平終了位置チェック：表示水平開始位置と表示幅から表示水平終了位置を計算します。表示水平終了位置が表示範囲左端より左の場合は何もせずに終了します。
文字書き込み水平開始位置チェック：文字書き込み水平開始位置が，表示範囲左端より左側にある場合は，表示範囲内のドットだけを展開するように，フォント・データ取り出しポインタを調整します。
文字書き込み垂直開始位置チェック：文字書き込み垂直開始位置が，表示範囲内だけにとどまるように調整します。
文字書き込み垂直終了位置チェック：文字書き込み垂直終了位置が，表示範囲内だけにとどまるように調整します。
表示データ更新フラグをセット：文字書き込み行に相当する表示データ更新フラグをセットします。
文字書き込み：フォント・データをビットマップ表示メモリに書き込みます。
字間空白書き込み：文字間スペース分の空白ドットをビットマップ表示メモリに書き込みます。
終了

(2) 使用領域

定義場所	定義方法	型	変数名	意味
Print_c.h	構造体	char	gPrint.pitch	ピッチ（固定ピッチ／プロポーショナル）の指定値 属性バイトのbit4で指定されます。
		unsigned char	gPrint.xs gPrint.ys gPrint.xe gPrint.ye	現在選択されている表示範囲制限領域 ・gPrint.xs = 水平開始位置XS ・gPrint.ys = 垂直開始位置YS ・gPrint.xe = 水平終了位置XE ・gPrint.ye = 垂直終了位置YE 表示範囲指定制御コードで，対応した表示範囲制限領域値が設定されます。 初期状態および，表示範囲制限解除制御コードで，LCD全体が表示領域値として設定されます。
		unsigned char	gPrint.x	水平表示位置X
		unsigned char	gPrint.y	垂直表示位置Y
		unsigned char	gPrint.after_sp	字間ドット数が格納されているワーク変数
LCD_c.h	#define		nLCD_xm	= 128：ビットマップ表示メモリの横ドット数
			KDISP_REQUEST	= 1：表示更新要求あり
	構造体	unsigned char	gLCD.dreq	LCD表示更新フラグ配列（1：更新あり）
		unsigned char	gLCD.text	ビットマップ表示メモリ
FONT_c.h	構造体	unsigned char	SFBV	フォントの垂直ボディ・サイズ領域 uFont_get_parameter関数を使用してフォントの垂直ボディ・サイズを取り出すのに使用します。
		unsigned char	SFBH	フォントの水平ボディ・サイズ領域 uFont_get_parameter関数を使用してフォントの水平ボディ・サイズを取り出すのに使用します。

(3) プログラム・リスト

```
/**********************************
 1文字のフォントを展開
    引数：フォントの先頭アドレス
    戻値：無し
 **********************************/
void uPrint_disp_char(char *k) {
    unsigned char i, j, ys, ye, ie;
    unsigned short y, position;

    /* 行末超えチェック */
    if (gPrint.x > gPrint.xe) goto uPrint_disp_char_end;

    /* 表示幅計算 */
    if (gPrint.pitch) {
        /* プロポーショナル */
        i = *k++;
        ie = *k++;
        k += i * (uFont_get_parameter(SFBV) >> 3);
    }
    else {
        /* 固定ピッチ */
        k += 2;
        ie = uFont_get_parameter(SFBH);
    }

    /* 文字書き込み水平終了位置チェック */
    ie = ie + gPrint.x - 1;
    if (ie < gPrint.xs) {
            gPrint.x = ie;
            goto uPrint_disp_char_end;
    }
    if (ie > gPrint.xe) ie = gPrint.xe;

    /* 文字書き込み水平開始位置チェック */
    if (gPrint.x < gPrint.xs) {
        i = gPrint.xs - gPrint.x;
        k += i * (uFont_get_parameter(SFBV) >> 3);
        gPrint.x = gPrint.xs;
    }

    /* 垂直開始位置チェック */
    if (gPrint.y < gPrint.ys)    ys = gPrint.ys;
    else                         ys = gPrint.y;

    /* 垂直終了位置チェック */
    ye = gPrint.y + uFont_get_parameter(SFBV) - 1;
    if (ye > gPrint.ye)          ye = gPrint.ye;

    /* 表示データ更新フラグ・セット(LCD_c.cの仕様依存) */
    for (j = ys; j <= ye; j += 8) {
        gLCD.dreq[j >> 3] = kDISP_REQUEST;
    }


    /* 文字書き込み */
    position = (unsigned short)(gPrint.y >> 3) * nLCD_xm + gPrint.x;
    while (gPrint.x <= ie) {
        for (j = gPrint.y, y = 0; j < gPrint.y + uFont_get_parameter(SFBV); j += 8, y += nLCD_xm) {
            if (j >= ys && j <= ye) {
                gLCD.text[y + position] = *k;
            }
            k++;
        }
        position++;
        gPrint.x++;
    }

    /* 字間加算 */
    ie = gPrint.x + gPrint.after_sp;
    if (ie > gPrint.xe) ie = gPrint.xe;

    /* 空白ドット書き込み */
    while (gPrint.x < ie) {
        for (j = ys, y = 0; j <= ye; j += 8, y += nLCD_xm) {
            gLCD.text[y + position] = 0;
        }
        position++;
        gPrint.x++;
    }

uPrint_disp_char_end:
    /* ■■■■■■■■■■■■■■■■■■■■■■■■■■■ */
    /* ■■■ 注意:本版ではバンクを0にして戻ります。 ■■■ */
    /* ■■■■■■■■■■■■■■■■■■■■■■■■■■■ */
    BANK=0;
    return;
}
```

① 行末超えチェック処理

② ビットマップ・データ取り出し位置と表示幅計算処理

③ 文字書き込み水平終了位置チェック処理

④ 文字書き込み水平開始位置チェック処理

⑤ 垂直開始位置チェック処理

⑥ 垂直終了位置チェック処理

⑦ 表示データ更新フラグ・セット処理

⑧ 文字書き込み処理

⑨ 字間空白ドット書き込み処理

⑩ BANK を 0 にします

(4) 詳細説明

1文字をLCDに表示させるために，フォント・データのビットマップ・データをビットマップ表示メモリに格納します。

まず，引き渡されたフォント・データのアドレスから，ビットマップ・データ取り出し位置を計算します。また，書き込み行に対応したLCD表示更新フラグをセットします。さらに，表示を行うフォント・データをビットマップ表示メモリに格納し，最後に，文字間空白ドットを格納します。表示位置は，次の表示位置に更新されます。

関数から戻るときに，BANKを0にして戻ります。

① 行末超えチェック処理

水平表示位置が，表示範囲外（表示右端より右側）の場合は，何もせずに，BANKを0にして本関数を終了します。

② ビットマップ・データ取り出し位置と表示幅計算処理

ビットマップ・データの取り出し位置と表示幅を計算します。

・プロポーショナル指定の場合

フォント・データから水平有効ドット開始位置と水平有効ドット幅を取り出します。水平有効ドット開始位置からビットマップ・データ取り出し位置を計算します。

取り出しバイト位置 = ビットマップ・データ先頭
＋（水平有効ドット開始位置×垂直バイト数）

（垂直バイト数は，フォント垂直ボディ・サイズを8で割った値です。）

表示幅はフォント水平有効ドット幅を使います。

・固定ピッチ指定の場合

ビットマップ・データ取り出し位置をフォント・データ中のビットマップ・データ先頭とします。

取り出しバイト位置 = ビットマップ・データ先頭
= フォント・データ先頭位置＋2

表示幅は，フォント水平ボディ・サイズを使います。

③ 文字書き込み水平終了位置チェック処理

・書き込み水平終了位置が，表示範囲左端より左側であれば，文字全体が表示範囲から外れているので，表示現在位置を更新し，BANKを0にして本関数を終了します。

・書き込み水平終了位置が，表示範囲右端より右側であれば，書き込み水平終了位置を表示範囲右端までとします。

④ 文字書き込み水平開始位置チェック処理

書き込み水平開始位置が，表示範囲より左側にある場合は，次の手順で，表示範囲右端位置に相当するビットマップ・データ取り出し位置に補正して，表示範囲内のビットマップ・データだけを取り出すようにします。

・取り出しバイト位置 ＋= （表示範囲右端までのドット数×垂直バイト数）

（垂直バイト数は，フォント垂直ボディ・サイズを8で割った値です。）

・水平表示位置を表示範囲右端に補正します。

⑤　垂直開始位置チェック処理

垂直表示位置が表示範囲上端より上側であれば，垂直表示開始位置を，表示範囲上端に補正します。そうでない場合は，垂直表示開始位置は垂直表示位置とします。

⑥　垂直終了位置チェック処理

垂直表示終了位置を計算し，表示範囲下端を超える場合は，表示範囲下端に補正します。

・垂直表示終了位置 = 垂直表示位置＋フォント垂直ボディ・サイズ－1

⑦　表示データ更新フラグ・セット処理

表示領域を更新するLCD相対行番号に対応したLCD表示更新フラグを設定します。

・LCD相対行番号 = 垂直表示位置／8

⑧　文字書き込み処理

ビットマップ・データをビットマップ表示メモリに転送します。

・ビットマップ表示メモリの格納先を計算

格納位置 ＝ （LCD相対行番号×LCD表示水平ドット数）＋水平表示位置

・ビットマップの書き込み

ビットマップ表示メモリに，フォント垂直ボディ・サイズ分のビットマップを，垂直8ドット行単位で，水平開始位置から水平終了位置まで，コピーします。ただし，垂直表示位置が表示範囲にある場合のみ，データを書き込みます。1列分（フォント垂直ボディ・サイズ分）転送ごとに，水平表示位置を更新します。

⑨　字間空白ドット書き込み処理

文字後の字間スペース用の空白ドットをビットマップ表示メモリに格納します。

・字間加算

文字間空白を書き込む水平終了位置を計算します。水平終了位置が，表示範囲右端を超える場合は，水平終了位置は表示範囲右端までとします。

空白ドット終了位置 = 水平表示位置＋空白ドット数

・空白ドット書き込み

ビットマップ表示メモリに，空白ドットとして，0を書き込みます。書き込みは，垂直8ドット単位での行数分の埋め込みを，水平開始位置から水平終了位置まで，繰り返します。

1列分書き込みごとに，水平表示位置を更新します。

⑩　BANKを0にします

本関数から戻るときに，BANKを0にして戻ります。もし，元のBANKの値に戻す必要があれば，事前に保存したBANK値をここで戻します。FDJ78K0Aを使用する場合は，この行を削除するかコメントにします。

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　有償版／無償版ツールでのビルド方法の違い

開発ツールが有償版と無償版とでは，フォント・データのサイズや配置によっては，使用するフォント定義ファイルが異なります。本サンプル・アプリケーションでは，14×14および24×24ドット・フォントについてはそれぞれのツールに対応した異なるファイルが添付されています。

有償版の場合：フォント・ユーティリティで，有償版を指定して作成したフォント定義ファイルが，使用できます。生成されたフォント定義ファイルを組み込み，ビルドを行いますと，そのままデバッガを立ち上げるか，生成されたHEXファイルをプログラマでデバイスに書き込めば実行できます。

無償版の場合：フォント・ユーティリティで，フリー版を指定して作成したフォント定義ファイルが，使用できます。生成されたフォント定義ファイル（アセンブラ・ソース版）を組み込み，ビルドを行います。フォント・ユーティリティで，同名のHEX版フォント定義ファイルが生成された場合は，HEXファイルの編集が必要です。
（HEXファイルの編集につきましては，「フォント・ユーティリティのユーザーズ・マニュアル（U19527）」を参照してください。）

表4－1　有償版／無償版開発ツール使用上の違い

	有償版	無償版
フォント・定義ファイル（アセンブラ・ソース版）	プロジェクトに組み込んでビルド	プロジェクトに組み込んでビルド
フォント・定義ファイル（HEX版）	不要です。（フォント・ユーティリティでは生成されません。）	ビルドで生成されたHEXファイルに組み込むための編集操作が必要です。（フォント・データが，無償版のオブジェクト・サイズ制限内に収まっている場合は，HEX版は生成されませんので，編集操作は不要となります。）

それぞれのサンプル・ファイルにつきましては，「4.2　フォルダ構成」を参照してください。

4.2　フォルダ構成

サンプル・プログラムは下記からダウンロードできます。

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

ダウンロードしたファイルのフォルダの構成は次のようになっています。

図4－1　サンプル・アプリケーションのフォルダ構成

SM05FA2_R3xx	（3xx はリビジョンです。）
SM05FA2j.TXT	・リリース・ノート（変更情報などが記載されています。）
SM05FA2.prw	・ワークスペース・ファイル （有償版開発環境を使用する場合はこれを開いてください）
SM05FA2.prj	・プロジェクト・ファイル（自動参照）（有償用）
SM05FA2_F.prw	・ワークスペース・ファイル （無償版開発環境を使用する場合はこれを開いてください）
SM05FA2_F.prj	・プロジェクト・ファイル（自動参照）（無償用）
¥FONT_LIB	
FONT_c.h	・ヘッダ・ファイル
FDJ78K0A.LIB	・フォント・アクセス・ライブラリ（非 BANK 用）
FDJ78K0B.LIB	・フォント・アクセス・ライブラリ（BANK 用）
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.asm	・14 ドット　漢字フォント定義ファイル（有償用）
FK014LVH_F.asm	・14 ドット　漢字フォント定義ファイル（無償用）
FK014LVH_F.HEX	・14 ドット　漢字フォント定義ファイル HEX 版（無償用）
FK024LVH.asm	・24 ドット　漢字フォント定義ファイル（有償用）
FK024LVH_F.asm	・24 ドット　漢字フォント定義ファイル（無償用）
FK024LVH_F.HEX	・24 ドット　漢字フォント定義ファイル HEX 版（無償用）
SEL14_2800.csv	・14 ドット　漢字フォント用文字登録ファイル（参考用）
SEL14_2800.txt	・14 ドット　漢字フォント用表示テキスト・ファイル（参考用）
SEL24_5.csv	・24 ドット　漢字フォント用文字登録ファイル（参考用）
SEL24_5.txt	・24 ドット　漢字フォント用表示テキスト・ファイル（参考用）
¥SAMPLE	
Printc.c	・本書で解説するプログラム
Print_c.h	・本書で解説するプログラムのヘッダ･ファイル
SM05FA2.hex	・ビルド済実行モジュール （ビルドを行わずに動作確認を行う場合に使用します）
その他のファイル	・デモ・ボードの実行モジュールをビルドするため に必要なファイル群

表4－2　主要リリース・モジュール差分情報一覧

モジュール名	SM05FA2_R211 →SM05FA2_R301	SM05FV2_R301 →SM05FA2_R301
Print_c FDJ78K0A.LIB FDJ78K0B.LIB	新規追加	同一
FA107LVH FH014LVH FK014LVH（旧FK114LVH） FK024LVH Font_c.h	・データ部はフォント・ユーティリティで生成されたものです。 ・API部は，FDJ78K0A.LIB/FDJ78K0B.LIBに分離しました。	同一
User_c	廃止	－
UDEF_a.h UDEF_func_c.h	ビルド対象をUser_cからPrint_cへ変更しました。各種改良を反映しています。	ビルド対象を限定しています。
LCD_c Kernel_a Kernel_OPT_a Data_a Main_c COM1_rx_c COM1_tx_c BTimer_c BTimer_INT_a Kernel_data_a Kernel_init_c UDEF_func_c.c UDEF_data_a	各種改良を反映しています。 （一部のファイルは同一です。）	同一

4. 3　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。インストールされているツールのバージョンが異なる場合は，ツール選択だけは必要になります。

4. 3. 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。

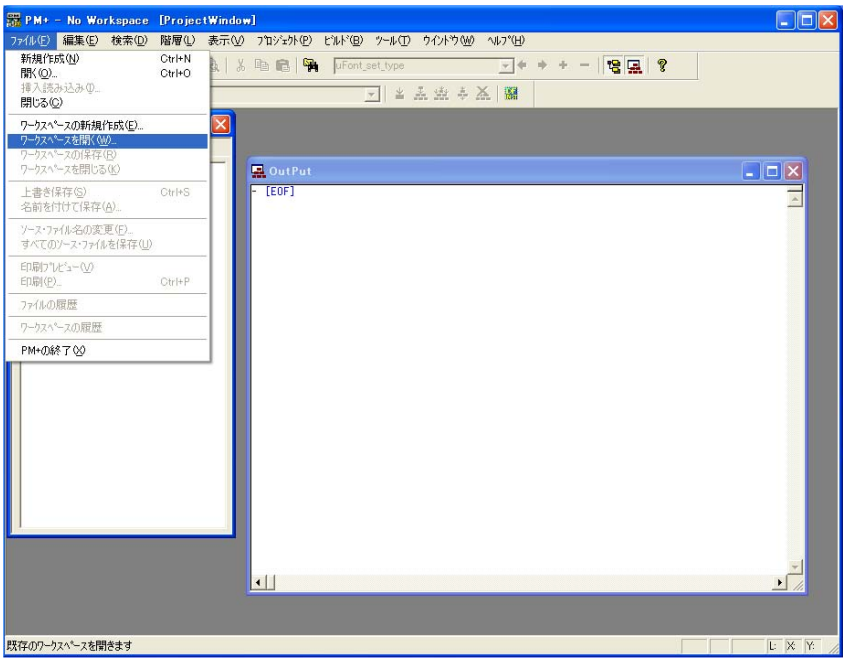
・有償版開発環境時：SM05FA2.prw

・無償版開発環境時：SM05FA2_F.prw

以下では，有償版開発環境での操作を説明します。

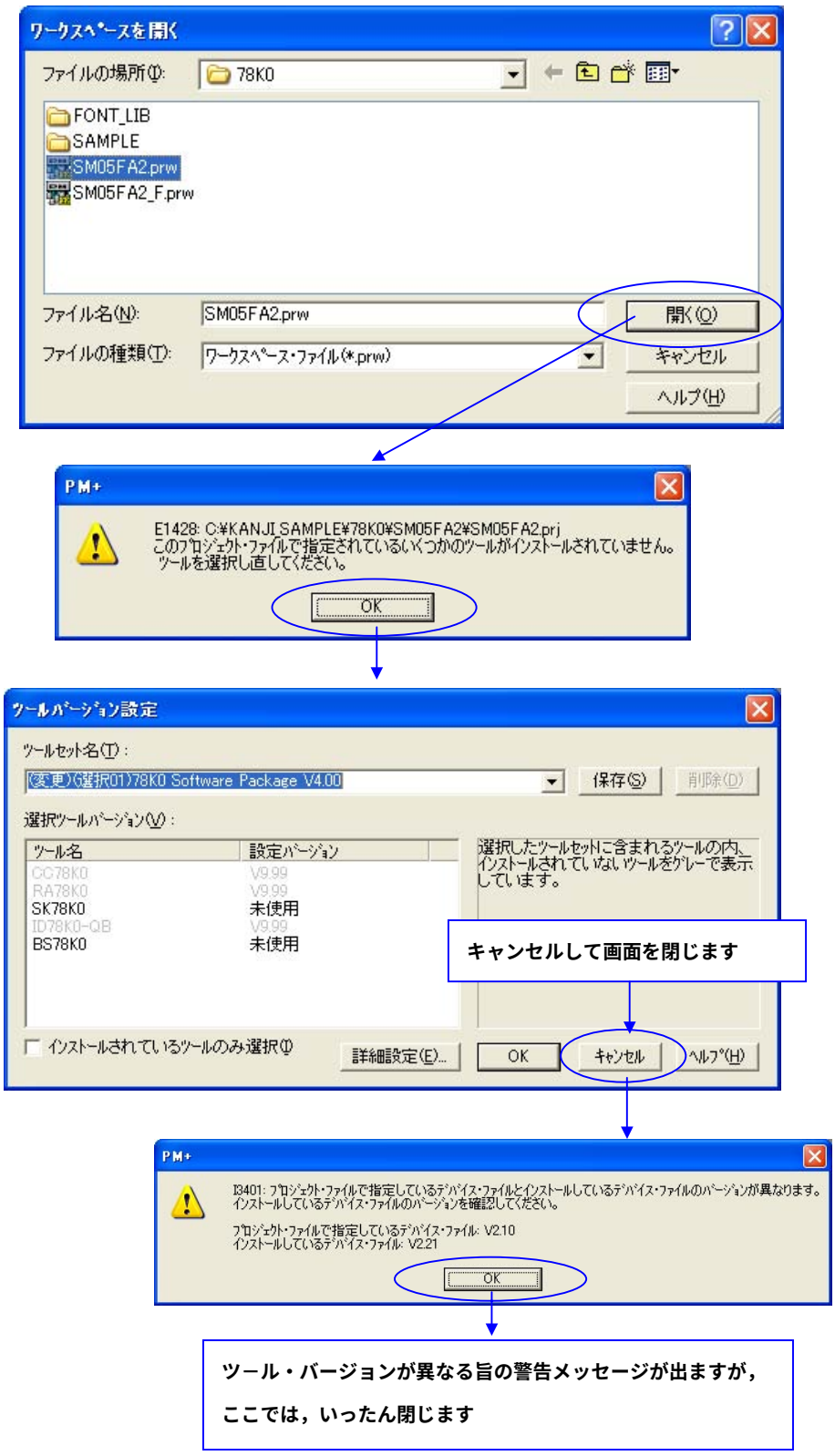
(1) PM+を起動します。

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。



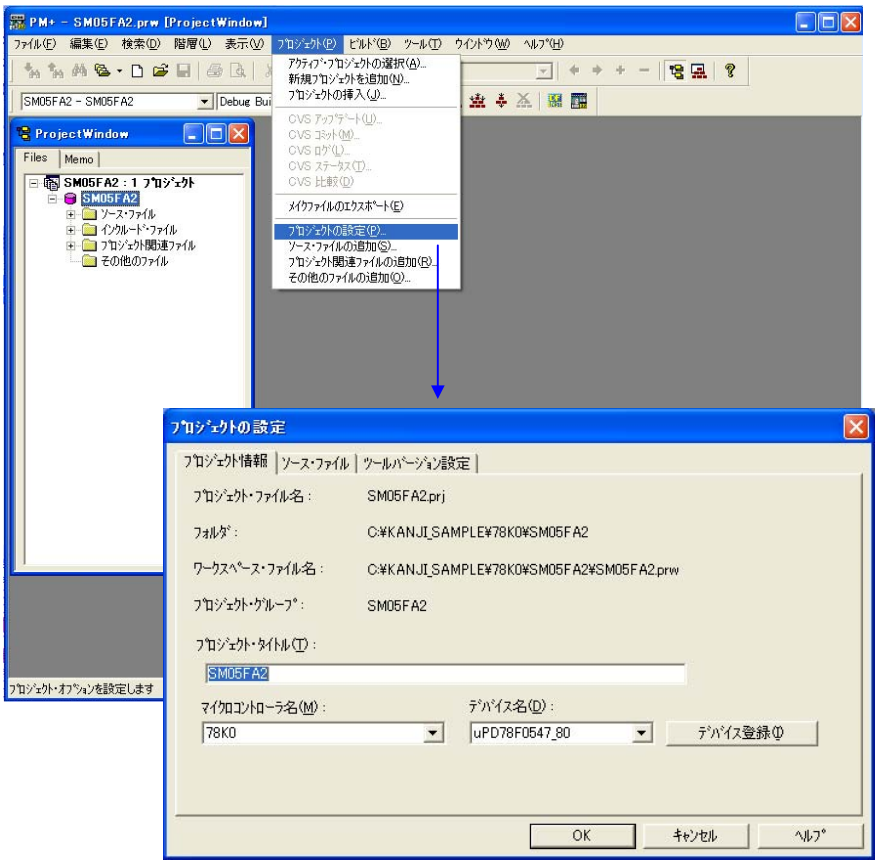
(2) ワークスペースを選択します

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。



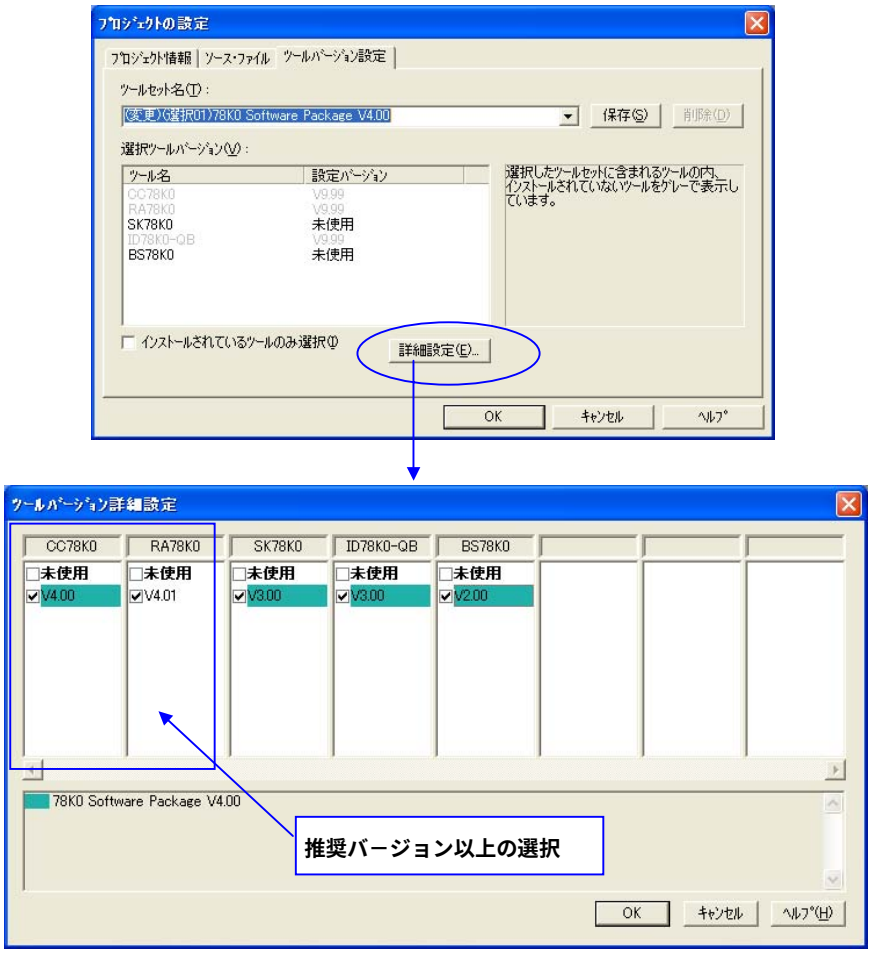
(3) プロジェクト設定画面

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。

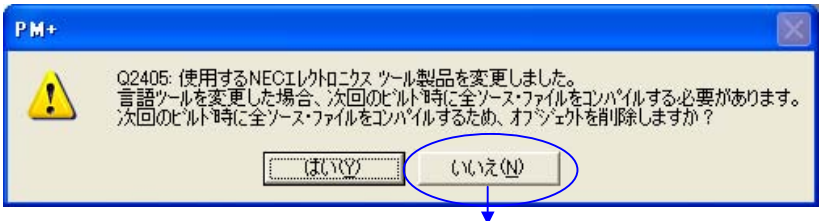


ツールは，推奨バージョン以上を選択してください。

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。

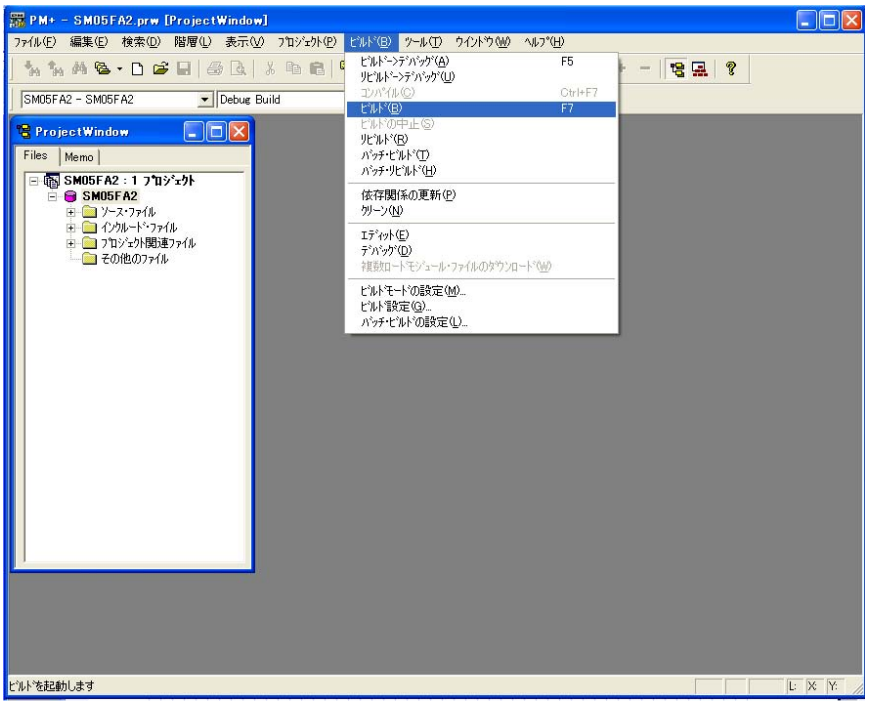
ツール選択後，ツール変更に伴うオブジェクト削除の問い合わせメッセージが表示された場合は，「いいえ」を選択してください。ツールを変更した後では，オブジェクト更新のために，「リビルド」を行ってください。



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。

(5) 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. 3. 2　ファイルの選択」～「4. 3. 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. 3. 2　ファイルの選択

ワークスペースの作成とファイルの選択について説明します。

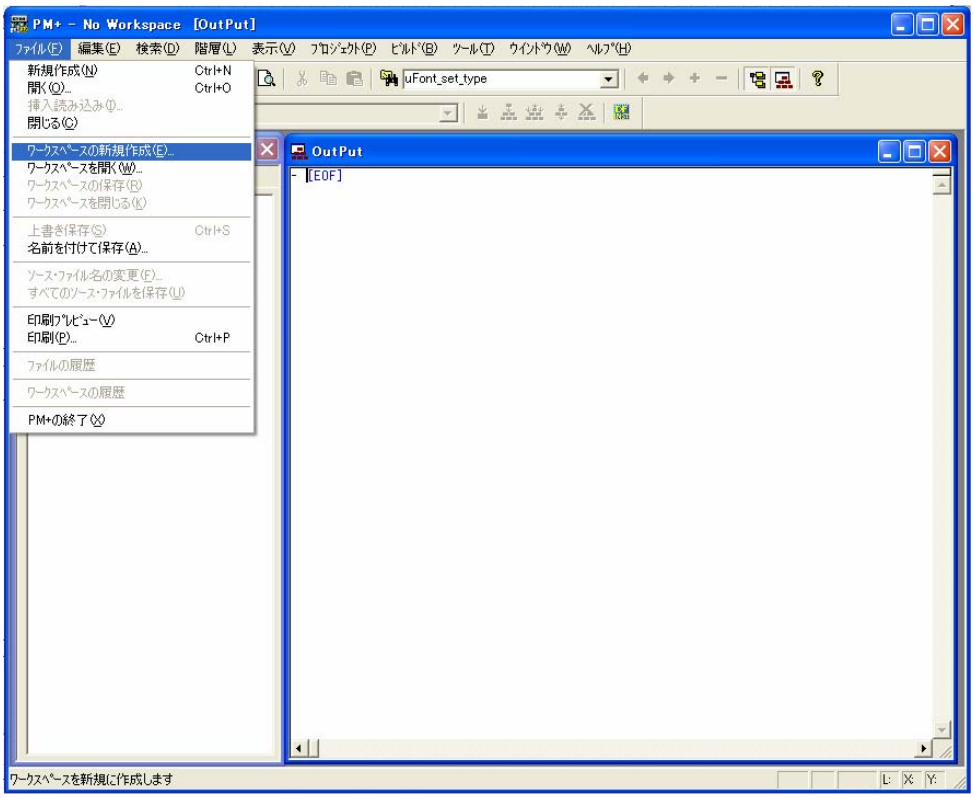
(1) ワークスペースの生成を行います。

①　ワークスペース作成の準備を行います。

サンプル・プログラム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SM05FA2として以下のフォルダにあるとします。
C:¥KANJI_SAMPLE¥78K0¥SM05FA2

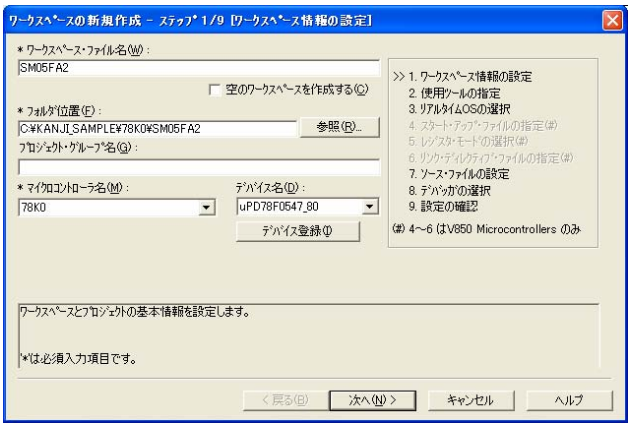
②　PM+を起動します。

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。



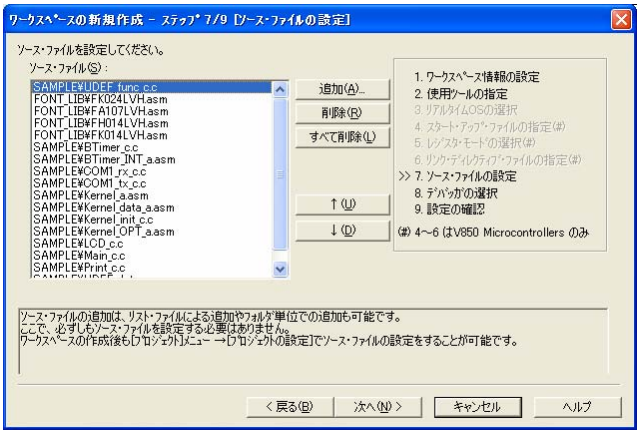
③　ワークスペースを新規作成します。

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。



④　以下のファイルを選択します。

Print_c.c	：漢字表示サンプル・プログラム（サンプル・タスク）。
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.asm	：14ドット　漢字フォント定義ファイル（有償版開発環境用）
	：無償版開発環境の場合は，FK014LVH_F.asmを選択します。
FK024LVH.asm	：24ドット　漢字フォント定義ファイル（有償版開発環境用）
	：無償版開発環境の場合は，FK024LVH_F.asmを選択します。
LCD_c.c	：表示データをLCDへ転送するタスク・プログラム。
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（割り込みハンドラ）。
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。



4. 3. 3　オプションの設定

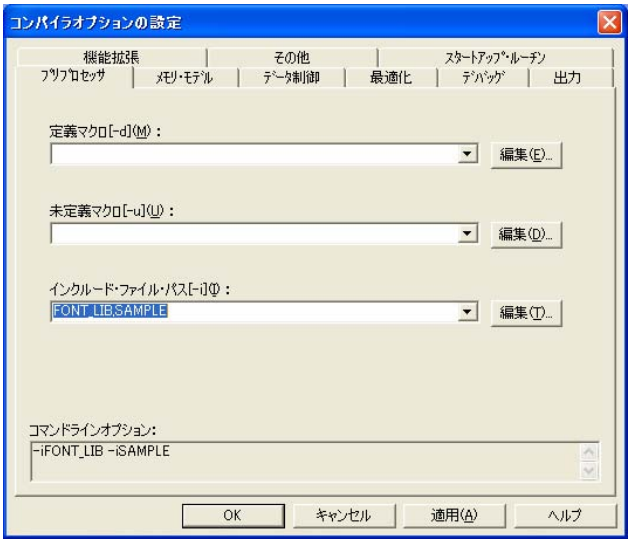
本サンプル・プログラムの実行モジュールを作成するために必要なコンパイラとリンカのオプション設定を説明します。

(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。

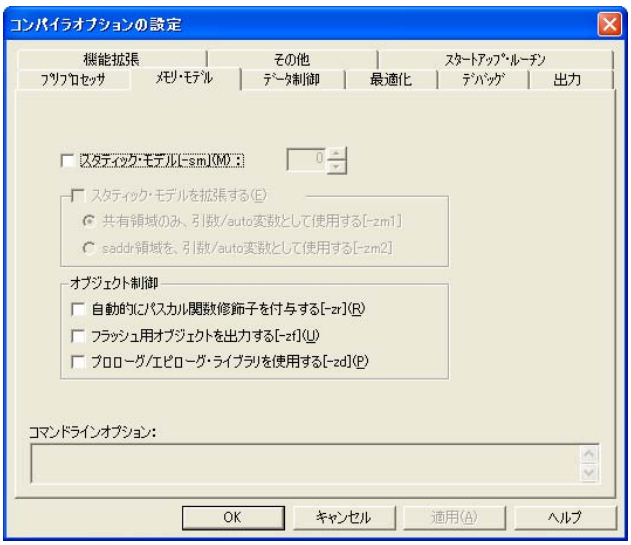
①　インクルード・パスを追加します

「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「SAMPLE」フォルダと「FONT_LIB」フォルダを追加します。



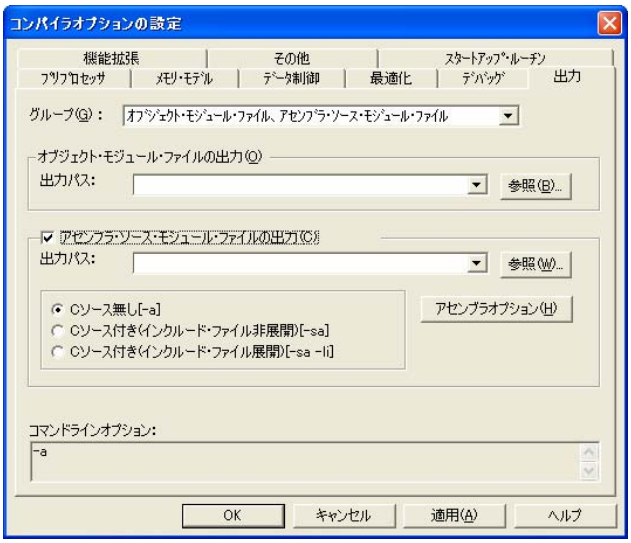
②　メモリ・モデルを選択します。

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。

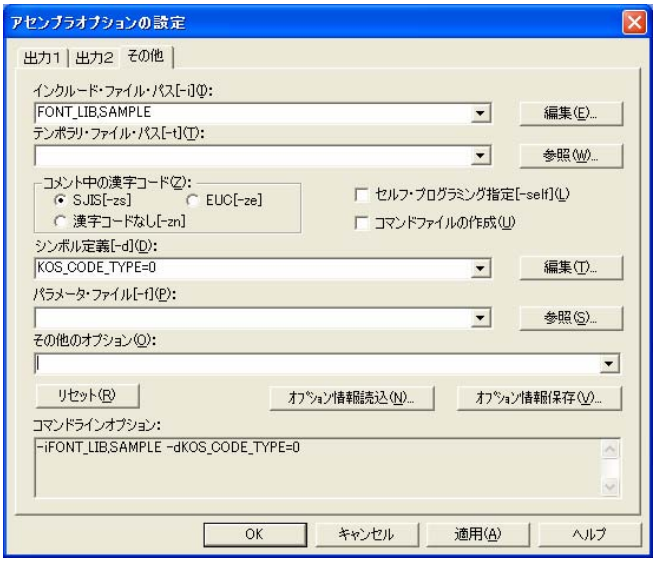


(2) アセンブラのオプションを設定します。

「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「SAMPLE」フォルダと「FONT_LIB」フォルダを追加します。
- 「シンボル定義」に「KOS_CODE_TYPE=0」を追加します。



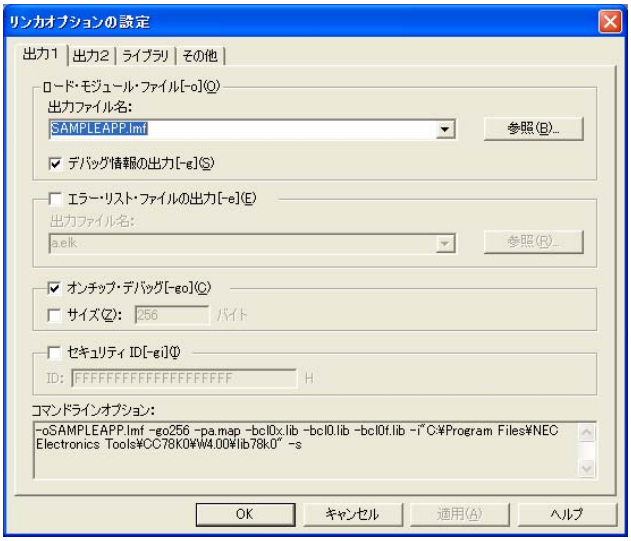
(3) リンカのオプションを設定します。

「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」を設定します。

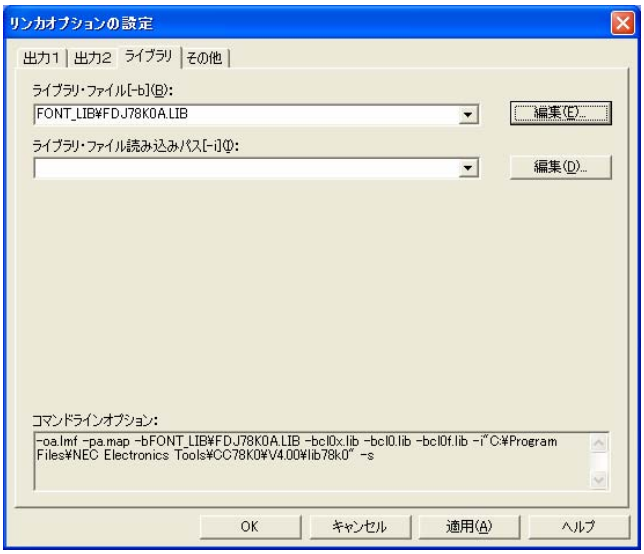


②　ライブラリを指定します。

「ライブラリ」タブでFDJ78K0A.LIBまたはFDJ78K0B.LIBを指定します。

FDJ78K0A.LIB：非BANK用

FDJ78K0B.LIB：BANK用



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。

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（Print_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，「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）」および「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）」を参照してください。

5. 1　プログラムの書き込みと起動方法

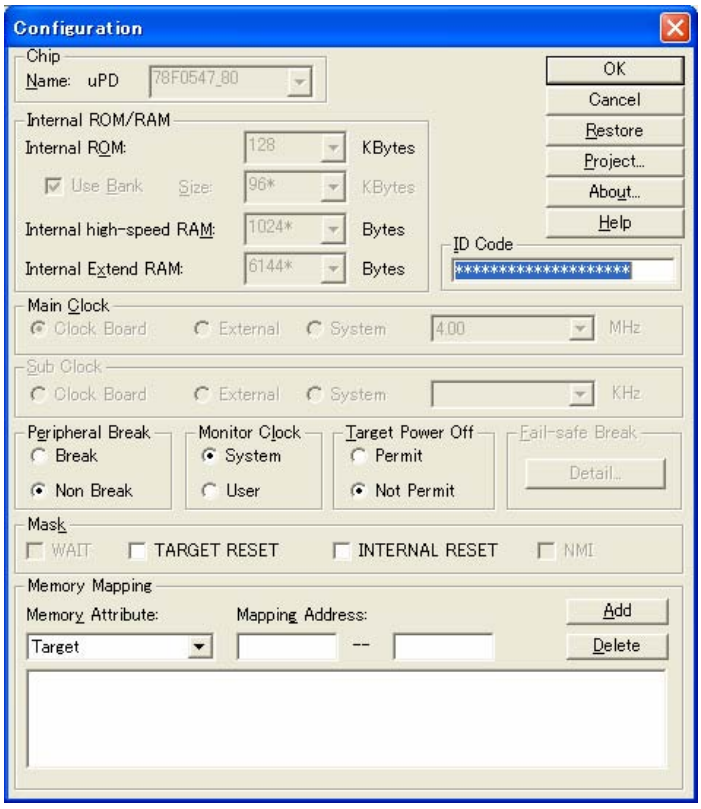
デバッガによる起動とデバッガを使わずにボード単体で起動する方法について説明します。

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) 設定画面が表示されます。設定画面でOKボタン押下後、デバッガ画面が表示されます。

Main Clockは，8MHzを指定します。

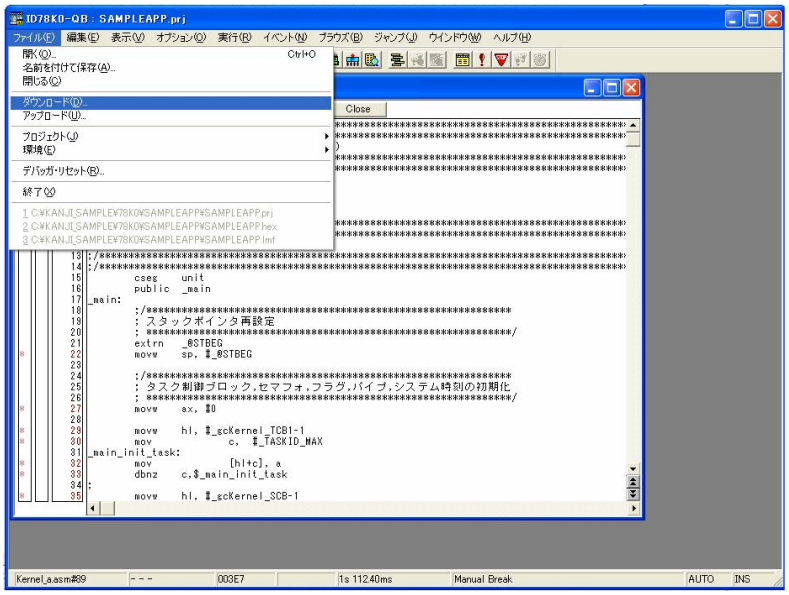


(2) 無償版開発ツールを使用した場合

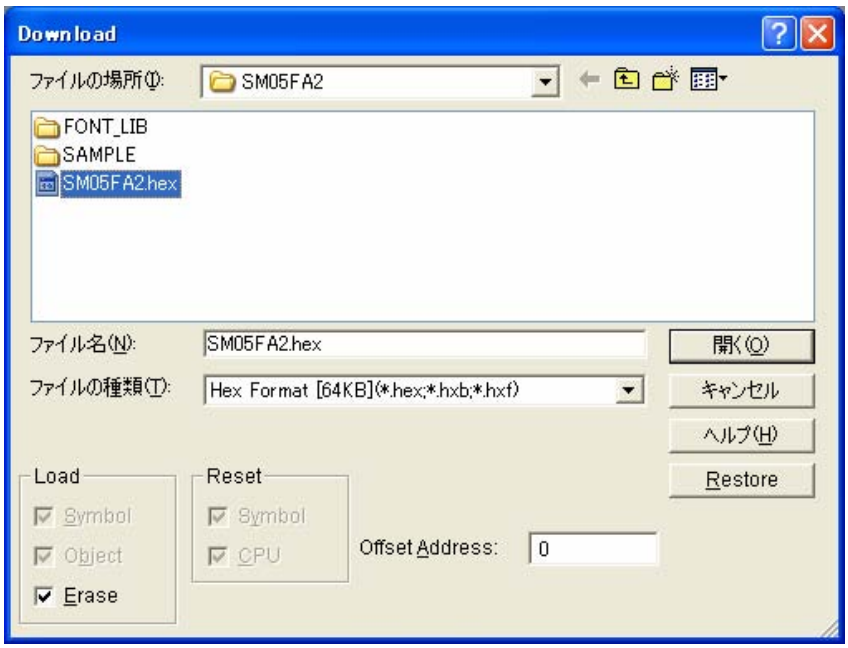
ビルド後，デバッガを起動した場合，正しく表示されない場合があります。フォント・データを実装するための編集を行ったHEXファイルをダウンロードして実行することにより，正しく表示することができます。

（HEXファイルの編集につきましては，「フォント・ユーティリティ　ユーザーズ・マニュアル（U19527）」を参照してください。）

(a) 「ファイル」メニューの「ダウンロード」を実行します。



(b) 編集済みHEXファイルを選択して，「開く」を実行するとダウンロードが開始されます。



(c) ダウンロード完了後に，プログラムを起動すると漢字が正しく表示されます。

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ファイルまたはSAMPLEフォルダダにあるSM05FA2.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。

無償版開発ツールを使用した場合には，ビルド後にHEXファイルの編集が必要です。編集を行わずに書き込み後実行しますと，正しく表示されないことがあります。HEXファイルの編集につきましては，「フォント・ユーティリティ　ユーザーズ・マニュアル（U19527）」を参照してください。

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　文字の表示方法

文字を表示するには，次の順でコマンドを入力します。

(1) パラメータ・ライト・コマンドで表示制限領域を指定します。

パラメータ・ライト・コマンドを使用して表示制限領域を指定します。表示制限領域が不要の場合は省略することができます。

(2) 文字表示コマンドで表示する日本語テキストや制御コードを指定します。

文字表示コマンドで表示位置，表示属性や表示する日本語テキストを指定します。制御コードを入力することにより，表示条件を変えて表示させることができます。

5. 4　コマンドの書式

(1) パラメータ・ライト・コマンド

書式：$JW' {パラメータ・アドレス}　{書き込みデータの並び}

指定したパラメータを設定します。

アドレスと設定データの内容は「表5－1　パラメーター一覧表」を参照してください。

16進数表記（A～Fは大文字のみ）で指定します。

表5－1　パラメーター一覧表

アドレス	内容
0	表示制限領域1の水平開始ドット位置を設定します。 有効範囲は0H～7FHです。 設定されたパラメータは，制御コードの表示制限領域1指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます
1	表示制限領域1の垂直開始ドット位置を設定します。 8ドット単位数での指定が可能です。 設定されたパラメータは，制御コードの表示制限領域1指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます。
2	表示制限領域1の水平終了ドット位置を設定します。 有効範囲は0H～7FHです。 設定されたパラメータは，制御コードの表示制限領域1指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます
3	表示制限領域1の垂直終了ドット位置を設定します。 8ドット単位数－1の指定が可能です。 設定されたパラメータは，制御コードの表示制限領域1指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます。
4	表示制限領域2の水平開始ドット位置を設定します。 有効範囲は0H～7FHです。 設定されたパラメータは，制御コードの表示制限領域2指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます
5	表示制限領域2の垂直開始ドット位置を設定します。 8ドット単位数での指定が可能です。 設定されたパラメータは，制御コードの表示制限領域2指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます。
6	表示制限領域2の水平終了ドット位置を設定します。 有効範囲は0H～7FHです。 設定されたパラメータは，制御コードの表示制限領域2指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます
7	表示制限領域2の垂直終了ドット位置を設定します。 8ドット単位数－1の指定が可能です。 設定されたパラメータは，制御コードの表示制限領域2指定で有効となり，$JFコマンドによる文字表示時に，表示制限領域として使用されます。

(2) パラメータ・リード・コマンド

書式：$JR'｛パラメータ・アドレス｝　｛リード・バイト数｝

指定したパラメータ・アドレスから指定したリード・バイト数分の設定内容を読み出せます。

［パラメータ・アドレス＋リード・バイト数］は8以下（パラメータ領域サイズ以下）で指定してください。

(3) 文字表示コマンド

書式：$JF'｛水平表示位置 垂直表示位置 表示属性｝　｛表示文字列／制御コード｝

指定した文字列を表示します。また，制御コードを指定して，表示制御を行うことも可能です。

(a) 水平表示位置

表示を開始する水平ドット位置を16進数表記で指定します。

(b) 垂直表示位置

表示を開始する垂直ドット位置を16進数表記で指定します。

(c) 表示属性バイト

表示属性バイトを16進数表記で指定します。属性値は次のとおりです。

表5－2　表示属性バイト

位置	属性	説明
bt3－0	空白ドット数	ピッチの指定に応じて次のようになります。 ただし，全角／半角混在時は，指定値の半分（小数切上げ）を，空白ドット数とします。 ・固定ピッチの場合 フォントの水平ボディ・サイズ＋空白ドット数が文字ピッチになります ・プロポーショナル 空白ドット数が字間ドット数になります。
bt4	ピッチ指定	ピッチを指定します。 ＝0：固定ピッチ ＝1：プロポーショナル
bit7―5	フォント指定	フォントの種別を指定します。 ＝0：1/4角（5×7ドット・フォント） ＝1：半角（7×14ドット・フォント） ＝2：全角（14×14ドット・フォント），半角（7×14ドット・フォント）混在 ＝4：全角（24×24ドット・フォント），半角（7×14ドット・フォント）混在

(d) 表示文字列

表示文字列は，'（アポストロフィ）で囲んで書きます。'（アポストロフィ）で囲まれていない部分は16進数表記による文字コードとみなされます。20H未満の文字コードは，制御コードとして処理されます。

全角はシフトJISコードを使います。

表示後は，表示位置が更新されます。

(e) 制御コード

フォント表示の制御を行うための制御コードを,$JFコマンド中に16進数表記で指定することができます。制御コードによっては，コードに続けて，パラメータの指定が必要なものがあります。

書式：制御コード　{パラメータ1～3}

表5－3　制御コード一覧表

制御コード	16進値	説明
¥n	0AH	行末まで，空白で埋めて，改行します。
¥t	09H	続く1バイトで指定される水平ドット位置まで，空白を埋めます。 ① パラメータ1：水平ドット位置
¥f	0CH	属性（フォント指定，ピッチ指定，空白ドット数）を，続く1バイトの値に変更します。 （属性の詳細は，「（c）表示属性」を参照してください。） ① パラメータ1のbit3-0：空白ドット数 ② パラメータ1のbit4：ピッチ指定 ③ パラメータ1のbit7-5：フォント指定
¥v	0BH	表示位置（x, y）を続く2バイトで指定される値に変更します。 ① パラメータ1：表示位置X ② パラメータ2：表示位置Y
10H	10H	表示範囲制限を解除します。 次の11Hまたは12Hを指定した後で有効になります。$JFコマンド先頭で自動的に解除されます。
11H	11H	表示範囲制限1による表示範囲制限指定 パラメータ・アドレス0～3で指定される範囲だけに表示を行うように制限します。パラメータは$JWコマンドであらかじめ書き込んでおく必要があります。
12H	12H	表示範囲制限2による表示範囲制限指定 パラメータ・アドレス4～7で指定される範囲だけに表示を行うように制限します。パラメータは$JWコマンドであらかじめ書き込んでおく必要があります。

5.5　コマンド例

コマンドの使用例について説明します。表示例はそれぞれLCDに表示されるイメージを表現したものですので，個々の文字フォントや文字間スペースなどは実際の表示と異なるものがあります。

5.5.1　全角／半角文字表示コマンド例

(1) 全角／半角混在，プロポーショナル

$JF'0 0 51'Printユニット（'27'J'27'）は，'A'日本語テキストを表示'A'するdemonstration'A'プログラムです。'A

```
Printユニット（'J'）は，
日本語テキストを表示
するdemonstration
プログラムです。
```

(2) 全角，16ドット固定ピッチ

以降の表示例では，平家物語の冒頭部分を引用しています。固定ピッチ表示の場合はプロポーショナル表示との差異を示すために，同じ行数で一部文字を省いて表示しています。

$JF'0 0 42'祇園精舎の鐘の声'A'諸行無常の響有り'A'沙羅双樹の花の色'A'盛者必衰の理表す'A

```
祇園精舎の鐘の声
諸行無常の響有り
沙羅双樹の花の色
盛者必衰の理表す
```

$JF'0 0 42'驕者も久しからず'A'唯春の夜の夢の如'A'猛者も遂に亡びぬ'A'偏に風前の塵に同'A

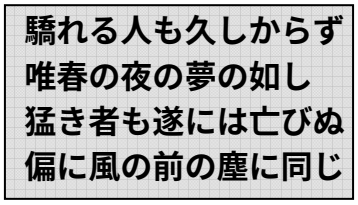
```
驕者も久しからず
唯春の夜の夢の如
猛者も遂に亡びぬ
偏に風前の塵に同
```

(3) 全角，プロポーショナル

$JF'0 0 53'祇園精舎の鐘の声'A 0C52'諸行無常の響き有り'A 0C53'沙羅双樹の花の色'A 0C52'盛者必衰の理を表す'A

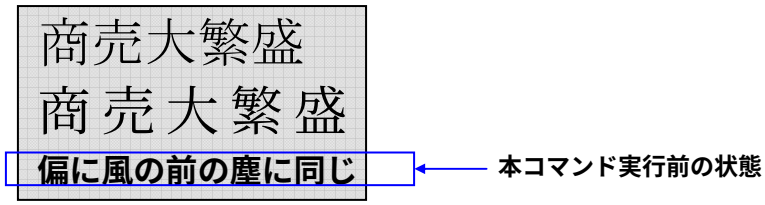
```
祇園精舎の鐘の声
諸行無常の響き有り
沙羅双樹の花の色
盛者必衰の理を表す
```

$JF’0 0 51'驕れる人も久しからず'A 0C52'唯春の夜の夢の如し'A 0C51'猛き者も遂には亡びぬ'A'偏に風の前の塵に同じ'A



（4）全角，24ドット

$JF'0 0 81'商売大繁盛'A 0C92'商売大繁盛'A



5. 5. 2　1/4角文字表示コマンド例

（1）1/4角，6ドット固定ピッチとプロポーショナル字間1ドットの交互表示

$JF'0 00 01'ABCDEFGHIJKLMNOPQRSTU'A

$JF'0 08 11'ABCDEFGHIJKLMNOPQRSTU'A

$JF'0 10 01'VWXYZabcdefghijklmnop'A

$JF'0 18 11'VWXYZabcdefghijklmnop'A

$JF'0 20 01'qrstuvwxyz0123456789.'A

$JF'0 28 11'qrstuvwxyz0123456789.'A

$JF'0 30 01'!"#$%&'27'()=~¥-^|@`[{<>'A

$JF'0 38 11'!"#$%&'27'()=~¥-^|@`[{<>ｱｶｻﾀ'A

5. 5. 3　半角限定文字表示コマンド例

（1）半角限定，8ドット固定ピッチ

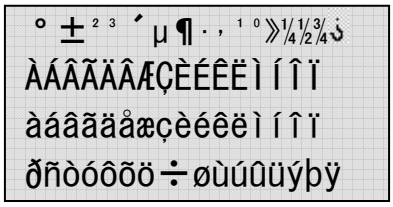
$JF'0 00 21'ｰｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿ'A

$JF'0 10 21'ﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏ'A

$JF'0 20 21E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF0A

$JF'0 30 21F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF0A

（半角フォントは英語用を組み込んでいるため半角カタカナの代わりにラテン文字が表示されます。）



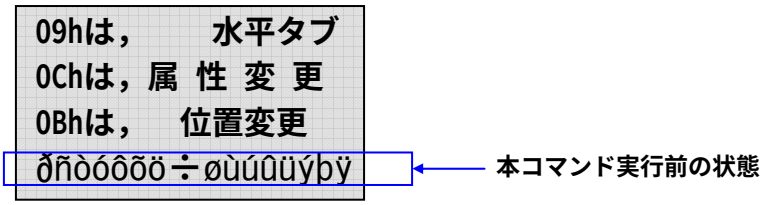
5. 5. 4　制御コードを使用したコマンド例

（1）水平タブ，属性変更，位置変更

$JF'00 00 42'09hは,'9 40'水平タブ'A

$JF'00 10 51'0Chは,'C 44'属性変更'A

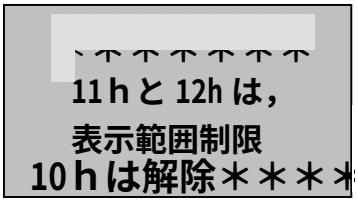
$JF'30 20 42'位置変更'A B 00 20'0Bhは,'



（2）表示範囲制限

$JW'0 18 08 77 37

$JF'00 00 42 A A A A 11'＊＊＊＊＊＊＊＊＊'A'11hと12hは，＊'A'表示範囲制限です'10 A'10hは解除＊＊＊＊'A



〔メ　モ〕

【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

【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/

【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。

お問い合わせ先

【営業関係，デバイスの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

【マイコン開発ツールの技術関係お問い合わせ先】

開発ツールサポートセンター

E-mail　：toolsupport-micom@ml.necel.com

【漢字表示プログラム／ボードの技術関係お問い合わせ先】　E-mail　：kanji-demo@ml.necel.com