

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ひ申し上げます。

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、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と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。

ユーザース・マニュアル

SM78Kシリーズ Ver.2.30以上

システム・シミュレータ

外部部品ユーザ・オープン・インタフェース仕様編

対象デバイス

78K/0シリーズ

78K/0Sシリーズ

78K/ シリーズ

〔メモ〕

目 次 要 約

第1章 概 説 ...	14
第2章 ダウンロード ...	18
第3章 プログラミング ...	20
第4章 関数レファレンス ...	27
第5章 CPUリセット時の動作 ...	83
第6章 プログラミング例 ...	84
付録A エラー・メッセージ ...	108

Pentiumは、Intel Corp.の商標です。

Windows, WindowsNT, およびMS-DOSは米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

PC/ATは、米国IBM Corp.の商標です。

- **本資料の内容は予告なく変更することがありますので、最新のものであることをご確認の上ご使用ください。**
- 文書による当社の承諾なしに本資料の転載複製を禁じます。
- 本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して、当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合、当社はその責を負うものではありませんのでご了承ください。
- 本資料に記載された回路、ソフトウェア、及びこれらに付随する情報は、半導体製品の動作例、応用例を説明するためのものです。従って、これら回路・ソフトウェア・情報をお客様の機器に使用される場合には、お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して、当社は一切その責を負いません。

M7A 98.8

巻末にアンケート・コーナーを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者 このマニュアルに記載している内容は、WindowsTM95/Windows98/Windows2000/WindowsNTTMの32ビット・アプリケーション・プログラム形式であるため、Windows95/Windows98/Windows2000/WindowsNTの32ビット・アプリケーション・プログラム作成経験者を対象とした説明となっています。

目的 このマニュアルは、システム・シミュレータSM78Kにおいて標準の外部部品で設定できない部分を、ユーザが作成できるようにインタフェース仕様を説明しています。ユーザがカスタマイズ部品のプログラムを作成するための関数とプログラミング・ルール、作成手順を説明しています。

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
- ・ダウンロード
- ・プログラミング
- ・関数レファレンス
- ・CPUリセット時の動作
- ・プログラミング例
- ・エラー・メッセージ

読み方 このマニュアルの読者には、マイクロコンピュータ、C言語に関する一般知識とWindows95/Windows98/Windows2000/WindowsNTの32ビット・アプリケーション・プログラムの作成に関する基礎知識を必要とします。

ユーザがカスタマイズ部品のプログラムを作成するための関数について知りたいとき

第4章 関数レファレンスをご覧ください。

メッセージの意味、原因などを知りたいとき

付録A エラー・メッセージをご覧ください。

対象製品 このマニュアル中の「SM78K」とは以下の製品を表します。

製品名	対応シリーズ
SM78K0	8ビット・シングルチップ・マイクロコンピュータ 78K/0シリーズ（小型汎用製品を除く）
SM78K0S	8ビット・シングルチップ・マイクロコンピュータ 78K/0Sシリーズ（小型汎用製品）
SM78K4	16ビット・シングルチップ・マイクロコンピュータ 78K/ シリーズ

また、このマニュアル中の「78KX」の表記は、ご使用になるシステム・シミュレータによって、次のように置き換えてください。

使用するシステム・シミュレータ	マニュアル中の表記	実際の名称
SM78K0	78KX	78K0
SM78K0S	78KX	78K0S
SM78K4	78KX	78K4

例 「SU78KX.DLL」は、次のように置き換えてください。

- ・ SM78K0の場合 : SU78K0.DLL
- ・ SM78K0Sの場合 : SU78K0S.DLL
- ・ SM78K4の場合 : SU78K4.DLL

- 凡 例
- データ表記の重み : 左が上位桁，右が下位桁
 - 注 : 本文中につけた注の説明
 - 注意 : 気をつ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...0XXXXX

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） $2^{20} = 1024^2$

関連資料 このマニュアルを使用する場合は、次の資料もあわせてご覧ください。

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。

78Kシリーズ開発ツールに関する資料（ユーザズ・マニュアル）

資料名		資料番号	
		和文	英文
CC78K0S Cコンパイラ Ver.2.30以上	操作編	U14871J	U14871E
	言語編	U14872J	U14872E
CC78K0 Cコンパイラ Ver.3.30以上	操作編	U14297J	U14297E
	言語編	U14298J	U14298E
CC78K4 Cコンパイラ Ver.2.20以上	操作編	U15557J	U15557E
	言語編	U15556J	U15556E
RA78K0S アセンブラ・パッケージ	操作編	U14876J	U14876E
	言語編	U14877J	U14877E
	構造化アセンブリ言語編	U11623J	U11623E
RA78K0 アセンブラ・パッケージ	操作編	U14445J	U14445E
	言語編	U14446J	U14446E
	構造化アセンブリ言語編	U11789J	U11789E
RA78K4 アセンブラ・パッケージ	操作編	U15254J	U15254E
	言語編	U15255J	U15255E
	構造化アセンブラ・プリプロセッサ	U11743J	U11743E
SM78Kシリーズ システム・シミュレータ Ver.2.30以上	操作編（Windowsベース）	U15373J	U15373E
SM78Kシリーズ システム・シミュレータ Ver.2.30以上	外部部品ユーザ・オープン・インタフェース仕様編	このマニュアル	U15006E
ID78Kシリーズ 統合ディバッガ Ver.2.30以上	操作編（Windowsベース）	U15185J	U15185E
RX78K0 リアルタイムOS	基礎編	U11537J	U11537E
	インストール編	U11536J	U11536E
RX78K4 リアルタイムOS	基礎編	U10603J	U10603E
	インストール編	U10604J	U10604E
プロジェクト・マネージャ Ver.3.12以上（Windowsベース）		U14610J	U14610E

目 次

第1章 概 説 ... 14

- 1.1 外部部品ユーザ・オープン・インタフェース仕様の概要 ... 14
- 1.2 ユーザ・カスタム部品の概要 ... 14
 - 1.2.1 カスタマイズの種類 ... 14
 - 1.2.2 ユーザ作成ファイル ... 14
 - 1.2.3 ユーザ・カスタマイズ部品の位置づけ ... 15
- 1.3 環 境 ... 16
 - 1.3.1 開発環境 ... 16
 - 1.3.2 動作環境 ... 16
- 1.4 SM78KシリーズVer.1.42以前の外部部品からVer.2.30以降に移行する場合の注意 ... 17
 - 1.4.1 ソースの変更 ... 17
 - 1.4.2 メイク環境の変更 ... 17

第2章 ダウンロード ... 18

- 2.1 ダウンロード ... 18
- 2.2 アンロード ... 19

第3章 プログラミング ... 20

- 3.1 プログラミング構成と処理フロー ... 20
 - 3.1.1 <入出力パネル> ウィンドウによるカスタマイズ ... 20
 - 3.1.2 ユーザ・ウィンドウによるカスタマイズ ... 21
- 3.2 カスタマイズ部品の作成手順 ... 23
 - 3.2.1 <入出力パネル> ウィンドウによるカスタマイズ ... 23
 - 3.2.2 ユーザ・ウィンドウによるカスタマイズ ... 23
- 3.3 基本規則 ... 25
 - 3.3.1 ユーザ関数 ... 25
 - 3.3.2 外部変数 ... 25
 - 3.3.3 機能名 ... 25
 - 3.3.4 アクティブH/L ... 26
 - 3.3.5 端子名 ... 26
 - 3.3.6 インクルード・ファイル, ソース・ファイル ... 26
- 3.4 モジュール定義(DEF)ファイル ... 26
 - 3.4.1 EXPORTS宣言 ... 26

第4章	関数レファレンス	...	27
4.1	<入出力パネル> ウィンドウによるカスタマイズ	...	27
4.2	ユーザ・ウィンドウによるカスタマイズ	...	53
第5章	CPUリセット時の動作	...	83
5.1	<入出力パネル> ウィンドウによるカスタマイズ部品	...	83
5.2	ユーザ・ウィンドウによるカスタマイズ部品	...	83
第6章	プログラミング例	...	84
6.1	<入出力パネル> ウィンドウによるカスタマイズ部品例	...	85
6.1.1	サンプル内容	...	85
6.1.2	ソース例	...	86
	ターゲット・プログラム	...	86
	カスタム品ソース・ファイル UPsw00.c	...	87
	定義ファイル UPsw00.def	...	89
	メイク・ファイル UPsw00.mak	...	90
6.2	ユーザ・ウィンドウによるカスタマイズ部品例	...	94
6.2.1	サンプル内容	...	94
6.2.2	ソース例	...	95
	ターゲット・プログラム	...	95
	カスタム品ソース・ファイル UOport.c	...	96
	定義ファイル UOport.def	...	103
	メイク・ファイル UOport.mak	...	104
付録A	エラー・メッセージ	...	108
A.1	エラー処理	...	108
A.2	エラー/ワーニング・メッセージ	...	108

図の目次

図番号	タイトル, ページ
1 - 1	78KXシリーズ シミュレータの構成図 ... 15
2 - 1	SM78KXシミュレータ部<入出力パネル>ウインドウ ... 18
2 - 2	<読み込み>ダイアログ ... 18
3 - 1	<入出力パネル>ウインドウによるカスタマイズのプログラミング構成と処理の図 ... 20
3 - 2	ユーザ・ウインドウによるカスタマイズのプログラミング構成と処理の図 ... 22
3 - 3	作成フロー ... 24
4 - 1	プッシュ・ボタン ... 28
4 - 2	トグル・ボタン ... 29
4 - 3	グループ式選択ボタン ... 31
4 - 4	ビット・マップ表示の非アクティブLED (左), アクティブLED (右) ... 34
4 - 5	図形表示の非アクティブLED (左), アクティブLED (右) ... 34
4 - 6	ポート単位で設定できるLED関数 ... 36
4 - 7	マトリクスLED関数 ... 38
4 - 8	アクティブLED (左), 非アクティブLED (右) ... 39
4 - 9	ステッピング・モータ ... 41
4 - 10	縦型スクロール・バー式アナログ入力 ... 43
6 - 1	<入出力パネル>ウインドウによるカスタマイズ部品例 ... 85
6 - 2	ユーザ・ウインドウによるカスタマイズ部品例 ... 94

表の目次

表番号	タイトル, ページ
4 - 1	<入出力パネル> ウィンドウによるカスタマイズ関数 ... 27
4 - 2	ユーザ・ウィンドウによるカスタマイズ関数 ... 53
5 - 1	CPUリセット時の<入出力パネル> ウィンドウによるカスタマイズ部品 ... 83
A - 1	エラー・メッセージ一覧 ... 109
A - 2	ワーニング・メッセージ一覧 ... 110

第1章 概 説

1.1 外部部品ユーザ・オープン・インタフェース仕様の概要

SM78KXでは、ターゲット・デバイスの動作に加え、疑似的なターゲット・システムの動作もシミュレーションできます。

SM78KXでは、疑似的なターゲット・システムを構築するために、標準的な外部部品を提供しています。標準的な外部部品は、各部品専用の設定ダイアログが用意されていますので簡単に実現できます。

そのほかに、標準の外部部品設定ダイアログで設定できない部分は、ユーザがプログラミングすることでユーザ要求の外部部品を実現できます。

外部部品ユーザ・オープン・インタフェース仕様とは、ユーザがカスタマイズ部品のプログラムを作成するのに必要な、SM78KXのインタフェース部分の関数仕様です。

1.2 ユーザ・カスタム部品の概要

1.2.1 カスタマイズの種類

ユーザのプログラミングによる部品のカスタマイズには、以下の2種類があります。

(1) <入出力パネル> ウィンドウによるカスタマイズ

端子とアクション情報をパラメータとして与えるだけで容易に部品を作成できるカスタマイズ関数を使い、部品のカスタマイズを行います。

ユーザ関数内でコールするだけで、<入出力パネル> ウィンドウに部品を張り付け、シミュレーション処理もすべて行います。

(2) ユーザ・ウィンドウによるカスタマイズ

部品とウィンドウをユーザが作成できる関数を使い、部品のカスタマイズを行います。

ユーザ・ウィンドウのハンドル通知関数によりウィンドウ処理やユーザ部品からの入力を可能にし、シミュレーション・コール関数によりユーザ部品への出力表示処理ができます。

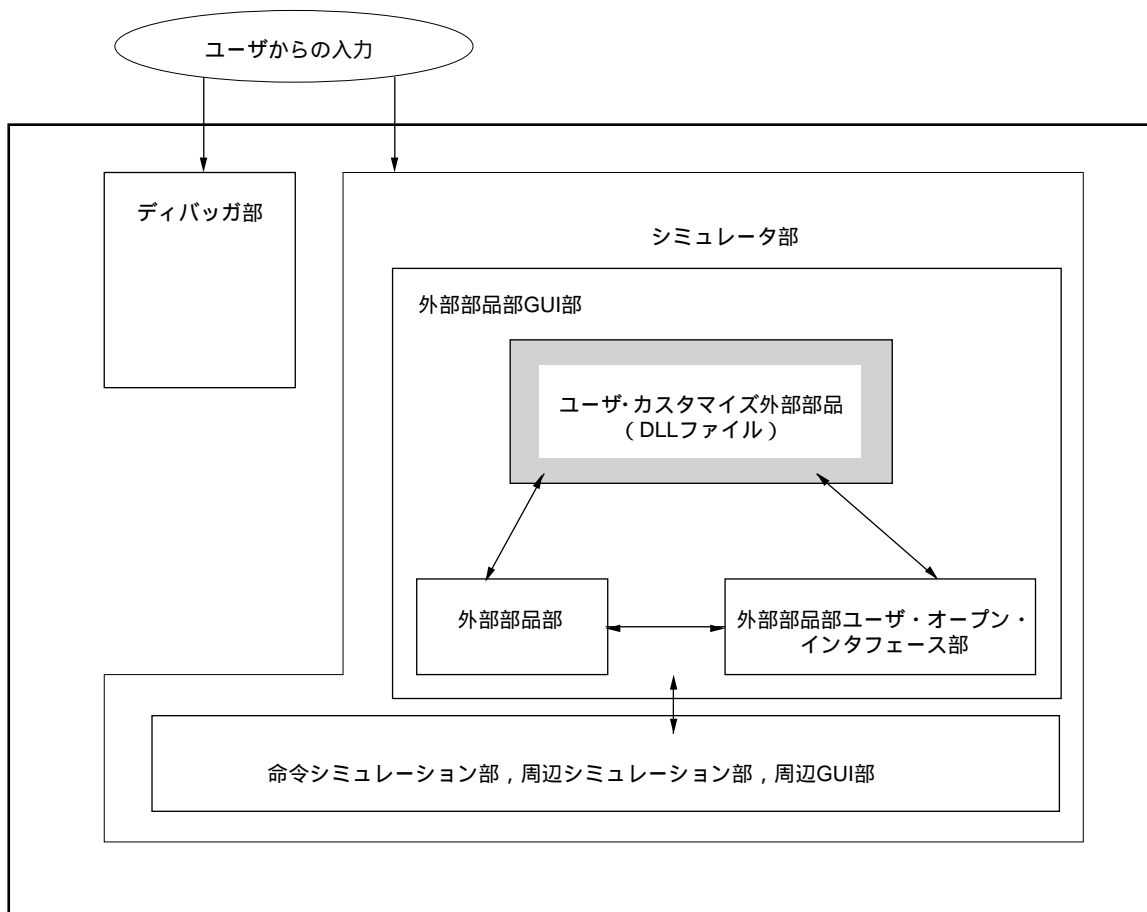
1.2.2 ユーザ作成ファイル

ユーザ・カスタマイズ部品は、このマニュアルの仕様に基づいてユーザが作成するプログラムによって実現されます。ユーザ作成のプログラムは、最終的にはDLLファイルにします。

ユーザ・カスタマイズ部品のDLLは、外部部品GUI部内にロードされシミュレーション処理されます。

1.2.3 ユーザ・カスタマイズ部品の位置づけ

図1-1 78KXシリーズのシミュレータの構成図



デバッガ部	ユーザがシミュレータに任意の機能を実行させるための指令をコマンドといい、このコマンド入力をキーボードまたはマウスによって操作する環境を提供する部分。
周辺GUI部	ユーザがポートへの自分の意図する入力情報を簡単にウインドウ上で設定できる設定環境を提供する部分。
DLL	ダイナミック・リンク・ライブラリ。Windowsアプリケーションや他のDLL内の関数から参照できる実行可能コードとデータを持つWindowsモジュール。
外部部品GUI部	外部部品をウインドウ上で動作可能としている部分。
外部部品部	外部部品GUI部の一部で、標準の外部部品を制御する部分。
ユーザ・カスタマイズ外部部品部	外部部品GUI部の一部で、ユーザが作成した外部部品の部分。
外部部品ユーザ・オープン・インタフェース部	外部部品GUI部の一部で、外部部品部とユーザ・カスタマイズ外部部品部間のインタフェースとなる部分。

1.3 環 境

1.3.1 開発環境

ユーザがこのマニュアルの仕様に基づいてプログラミングし、DLLファイルまで作成する開発環境を次に示します。

ハードウェア環境 : NEC PC-9821/PC98-NXシリーズ, IBM PC/ATTM互換機
(CPU : PentiumTM 120 MHz以上推奨)

ソフトウェア環境 : Windows95/Windows98/Windows2000/WindowsNT4.0
Microsoft Visual C++ V5.00以上

1.3.2 動作環境

ユーザが作成したファイルをロードして動作するシミュレータの動作環境を次に示します。

ハードウェア環境 : NEC PC-9821/PC98-NXシリーズ, IBM PC/AT互換機
(CPU : Pentium 120 MHz以上推奨)

ソフトウェア環境 : Windows95/Windows98/Windows2000/WindowsNT4.0

1. 4 SM78Kシリーズ Ver.1.42以前の外部部品からVer.2.30以降に移行する場合の注意

SM78Kシリーズ Ver.1.42以前の版で作成した外部部品をVer.2.30以降で使用するためには、外部部品のソースを一部修正し、外部部品の作り直しをする必要があります。

ここでは、その変更点について説明します。

1. 4. 1 ソースの変更

Cソースの変更点は次の2点です。

- ・インクルードするファイルをuparts.hからuparts32.hに変更してください。
uparts32.hは、SM78Kのインストール・ディレクトリ下（例：c:\nec\tools32）の.\smp78kx\sm78kxにあります。
- ・LibMain()、WEP() 関数の代わりにDllMain() を使用してください。

1. 4. 2 メイク環境の変更

新たにWin32のメイク環境を作成する必要があります。VC++上でメイク環境を作成する際は、次の2点に注意してください。

- (1) プロジェクトにuparts32.cppファイルを追加してください。
uparts32.cppは、SM78Kのインストール・ディレクトリ下（例：c:\nec\tools32）の.\smp78kx\sm78kxにあります。
- (2) 構造体メンバのアラインメントは1バイトにしてください。

第2章 ダウンロード

この章では、第3章、第4章に従って作成されたユーザ・カスタマイズ部品を、シミュレータへダウンロードする手順について説明します。

作成したカスタマイズ外部部品（DLLファイル）を実際に使用するためには、シミュレータにロードする必要があります。

ロードしたカスタマイズ外部部品（DLLファイル）を取り消すには、シミュレータからアンロードします。

ユーザが作成したカスタマイズ外部部品DLLのロードおよびアンロードは、<入出力パネル> ウィンドウで行います。

図2 - 1 SM78KXシミュレータ部<入出力パネル> ウィンドウ

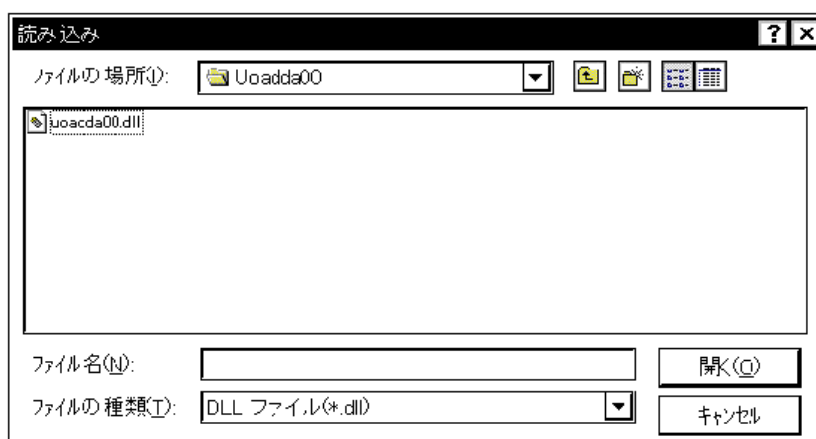


2.1 ダウンロード

操作手順

- (1) <入出力パネル> ウィンドウの [カスタム] メニュー [ロード] を選択すると、次の<読み込み>ダイアログが表示されます。

図2 - 2 <読み込み>ダイアログ



(2) <読み込み>ダイアログで、カスタマイズ外部部品DLLファイル名を指定して<開く>ボタンをクリックします。指定したDLLファイルは、シミュレータ内にロードされます。すると、<入出力パネル>ウインドウによるカスタマイズ関数で作った部品は、<入出力パネル>ウインドウに張り付けられ、ユーザ・ウインドウによるカスタマイズの部品であれば、ユーザ・ウインドウを表示します。

(a) シミュレータにロードできるユーザ作成のカスタマイズ外部部品DLLは、**最大6つ**となります。

(b) シミュレータにダウンロードしたカスタマイズ外部部品DLLファイルは、<入出力パネル>ウインドウをクローズしても有効です。再度<入出力パネル>ウインドウをオープンしたときに、このDLLファイルを自動的にダウンロードします。

(c) ロードしたユーザ作成のカスタマイズ外部部品DLLのファイル名は、<入出力パネル>ウインドウの[カスタム]メニューのプルダウン・メニューに追加されます。

(d) <入出力パネル>ウインドウ内に表示されたユーザ作成のカスタマイズ外部部品は、配置変更ができます。しかし、配置変更の情報は保存できません。配置変更後に次の操作を行った場合は、各部品の位置が保存または終了したときの状態ではありません。このため、再度部品の配置をしてください。

- ・プロジェクト・ファイル (xxxx.prj) や<入出力パネル>ウインドウの表示情報を保存するファイル (xxxx.pnl) に状態を保存したあと、これらのファイルを読み込んだとき。

- ・カスタマイズ外部部品DLLの情報をロードしたまま<入出力パネル>ウインドウをクローズし、次回<入出力パネル>ウインドウをオープンしたとき。

2.2 アンロード

操作手順

(1) <入出力パネル>ウインドウの[カスタム]メニュー [アンロード]を選択します。

(2) 現在シミュレータにロードされているカスタマイズ外部部品DLLがすべて取り消されます。<入出力パネル>ウインドウによるカスタマイズ関数で作った部品は、<入出力パネル>ウインドウから削除され、ユーザ・ウインドウによるカスタマイズのプログラムであれば、ユーザ・ウインドウがクローズします。

第3章 プログラミング

3.1 プログラミング構成と処理フロー

<入出力パネル> ウィンドウによるカスタマイズとユーザ・ウィンドウによるカスタマイズのそれぞれについて、基本的なプログラミングの構成を説明します。

3.1.1 <入出力パネル> ウィンドウによるカスタマイズ

構成

DLLファイル作成時に必要なDllMain関数，DLLファイルをロードした後に1回だけコールされるユーザ関数から構成されます。

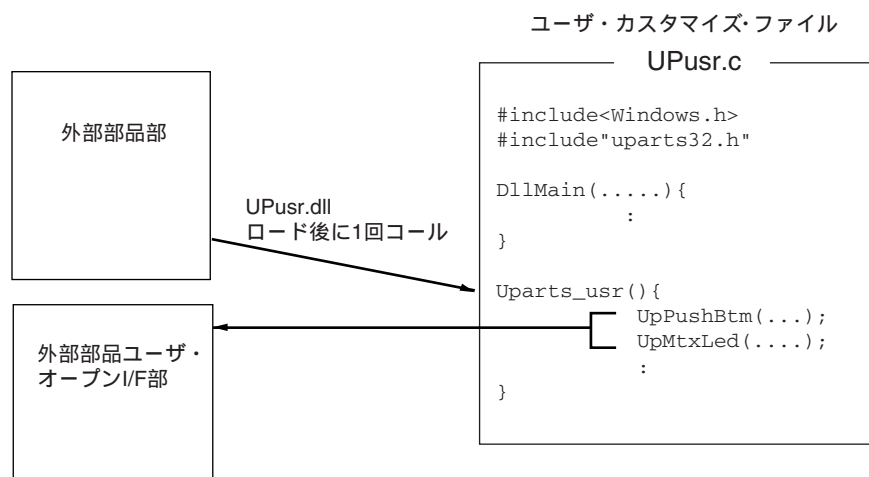
第4章で述べる関数レファレンスは，このユーザ関数内またはユーザ関数の下位の関数に記述しなければなりません。

処理フロー

シミュレータの外部部品部は，記述された関数の部品情報により部品を作成し，シミュレータの外部部品部の中で部品に関するすべてのシミュレーションを行います。

ユーザ作成のDLLファイルとシミュレータ内の外部部品との関係，関数の構成を図3 - 1に示します。

図3 - 1 <入出力パネル> ウィンドウによるカスタマイズのプログラミング構成と処理の図



3.1.2 ユーザ・ウインドウによるカスタマイズ

構成

DLLファイル作成時に必要なDllMain関数と作成したウインドウのコール・バック関数、ユーザ関数、シミュレーションの一定間隔でコールされるシミュレーション・コール関数から構成されます。

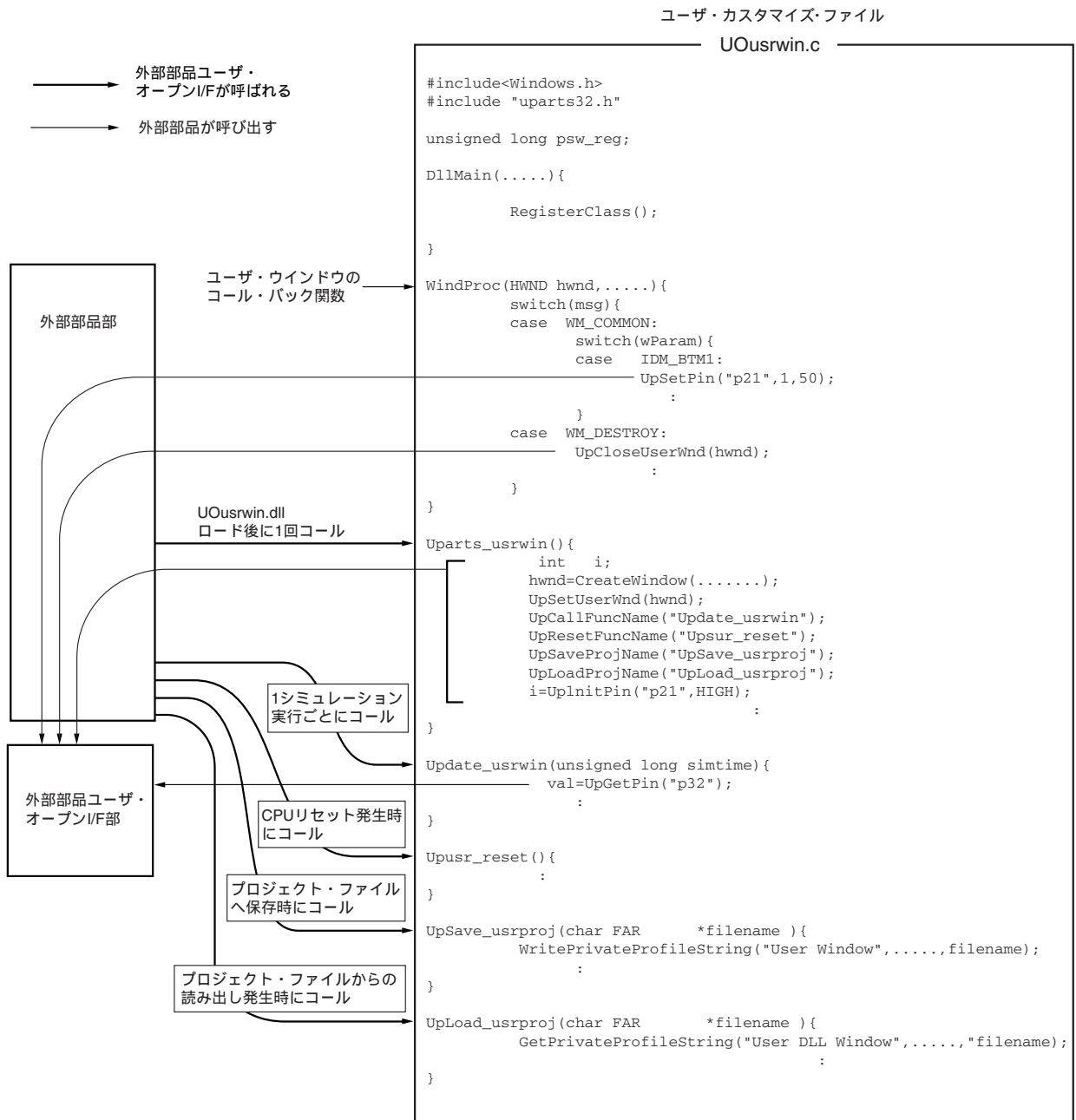
ユーザ関数またはユーザ関数の下位の関数で、シミュレーション・コール関数の通知やモータ端子名の通知などを行います。ユーザが作成したウインドウのコール・バック関数や、シミュレーション・コール関数内で、部品の作成や入出力のアクション部分などのプログラミングを行います。

処理フロー

端子、ポートの入出力情報の取得関数およびセット関数を使用してシミュレータの外部部品部とやり取りを行い、カスタマイズ部品のシミュレーションを行います。また、シミュレーション・コール関数を外部部品部からコールしてもらうことにより、端子の出力情報の再絵画などの処理を行います。

ユーザ作成のウインドウによるカスタマイズのDLLファイルとシミュレータ内の外部部品との関係、関数の構成を図3 - 2に示します。

図3-2 ユーザ・ウィンドウによるカスタマイズのプログラミング構成と処理の図



3.2 カスタマイズ部品の作成手順

3.2.1 <入出力パネル> ウィンドウによるカスタマイズ

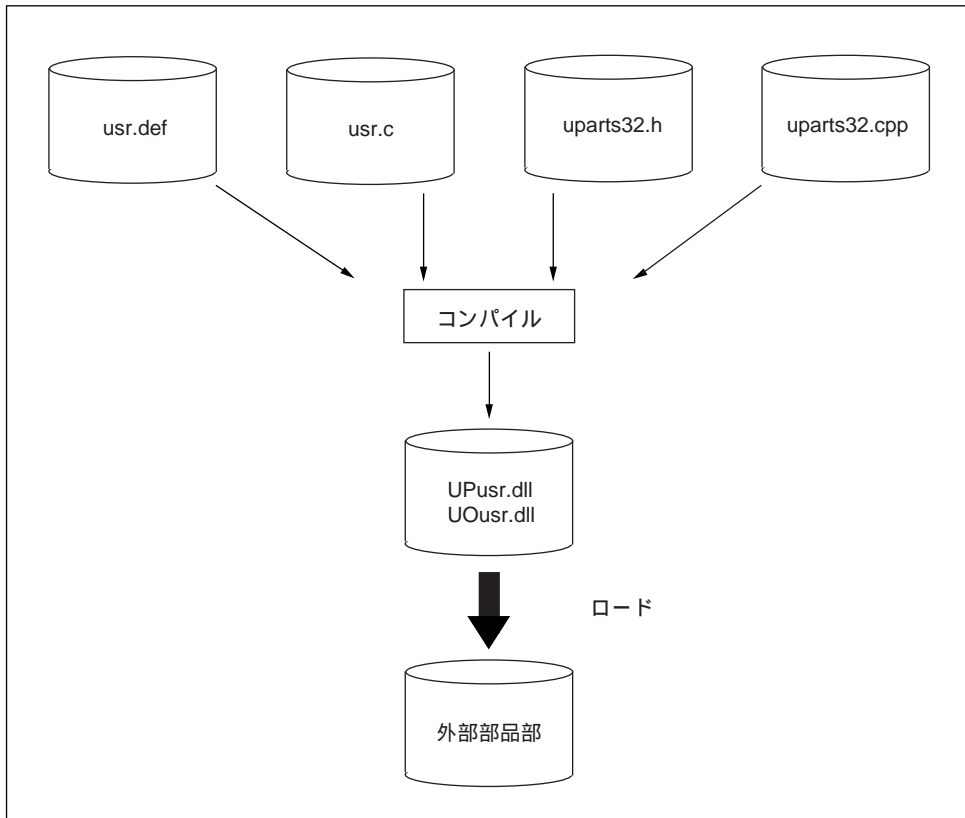
1. Windowsプログラミングに従ったDLL作成時の記述で、カスタマイズ用の外部部品をプログラミングします。その際には、“uparts32.h”をインクルードし、プロジェクトに“uparts32.cpp”を追加してください。
2. Windowsプログラミングに従ったモジュール定義(DEF)ファイル[※]、メイク・ファイル、必要であればリソース・ファイルを作成して、コンパイルしてユーザ作成DLLファイルを作成します。
 - ・コンパイル時には、構造体メンバのアライメントを1バイトにするためのオプション(/Zp1)を指定してください。
 - ・作成するDLLファイル名は、先頭2文字が“UP”で始まるようにしてください。
 - ・Microsoft Visual C++がインストールされていない環境でDLLファイルを動作させるには、DLLファイルをリリース版で作成してください。
3. シミュレータの外部部品のカスタマイズ・ファイル指定箇所[※]にユーザ作成DLLファイル名を設定します。(参照: 2.1 ダウンロード)
4. すでに、<入出力パネル>ウィンドウに表示している標準部品に加えて、ユーザ作成のカスタマイズ部品を表示します。
5. <入出力パネル>ウィンドウを配置モードにして、部品の配置を行います。
6. <入出力パネル>ウィンドウの[ファイル]メニュー [名前を付けて保存]を選択して状態を保存することにより、次のシミュレーションからはユーザ作成DLLをロードする必要はありません。

3.2.2 ユーザ・ウィンドウによるカスタマイズ

1. Windowsプログラミングに従ったDLL作成時の記述で、カスタマイズ用の外部部品をプログラミングします。その際には、“uparts32.h”をインクルードし、プロジェクトに“uparts32.cpp”を追加してください。
2. Windowsプログラミングに従ったモジュール定義(DEF)ファイル[※]、メイク・ファイル、必要であればリソース・ファイルを作成して、コンパイルしてユーザ作成DLLファイルを作成します。
 - ・コンパイル時には、構造体メンバのアライメントを1バイトにするためのオプション(/Zp1)を指定してください。
 - ・作成するDLLファイル名は、先頭2文字が“UO”で始まるようにしてください。
 - ・Microsoft Visual C++がインストールされていない環境でDLLファイルを動作させるには、DLLファイルをリリース版で作成してください。
3. シミュレータの外部部品のカスタマイズ・ファイル指定箇所[※]にユーザ作成DLLファイル名を設定します。(参照: 2.1 ダウンロード)
4. ユーザが作成したウィンドウとのカスタマイズ部品を表示します。

注 3.4 モジュール定義(DEF)ファイルを参照してください。

図3 - 3 作成フロー



3.3 基本規則

ユーザがカスタマイズ部品をプログラミングする際の基本規則を説明します。

3.3.1 ユーザ関数

ユーザ関数とは、ユーザが記述するメイン関数です。

- (1) シミュレータへユーザ作成のDLLファイルをロードした際に、シミュレータがコールする関数となります。
- (2) 第4章で説明する関数レファレンスは、ユーザ関数内またはユーザ関数の下位の関数に記述しなければなりません。
- (3) ユーザ関数は、“UParts_”にユーザ作成DLLファイル名の先頭2文字を除いた名前を付加した関数名とします。
- (4) ユーザ作成DLLファイル名の先頭2文字は固定です。
 - (a) <入出力パネル> ウィンドウによるカスタマイズ
ユーザ作成DLLファイル名の先頭2文字は“UP”にしてください。
例 UPusr.dll UParts_usr()
 - (b) ユーザ・ウィンドウによるカスタマイズ
ユーザ作成DLLファイル名の先頭2文字は“UO”にしてください。
例 UOusr.dll UParts_usr()
- (5) ユーザ関数は、パラメータなしのvoid型にしてください。
- (6) ユーザ関数は、モジュール定義ファイルでEXPORTS宣言^註してください。

注 3.4.1 EXPORTS宣言を参照してください。

3.3.2 外部変数

外部変数を使用する場合は、“UP”を先頭に付加してください。

例 int UPglobal

3.3.3 機能名

機能名は、ユーザ作成の外部部品に与える名前のことです。

機能名を部品名として表示したくない場合には、部品を作成する関数のパラメータに、NULL文字列を記述してください。

3.3.4 アクティブH/L

「アクティブH/L」とは、端子の値とアクティブ状態（端子に接続した部品が動作する状態）との関連付けを指定するものです。部品を作成する関数に、「アクティブH/L」を指定するパラメータがある場合には、次のどちらかのマクロを指定してください（マクロ“HIGH”，“LOW”は、uparts32.hで定義されています）。

1（ハイ）で動作：HIGH

0（ロウ）で動作：LOW

3.3.5 端子名

部品を作成する関数のパラメータに、端子名やポート名を指定するものがあります。このとき、端子名やポート名は文字列で指定しますが、その名前は対象になるデバイスのユーザーズ・マニュアルに明記されている名前にしてください。ただし、大文字、小文字の制限はありません。

3.3.6 インクルード・ファイル、ソース・ファイル

SM78Kの製品パッケージに、ユーザ・カスタマイズ用のインクルード・ファイルuparts32.hとユーザ・カスタマイズ用のソース・ファイル“uparts32.cpp”が含まれています。

uparts32.hのインクルードと“uparts32.cpp”のリンクを行ってください。

“uparts32.h”および“uparts32.cpp”は、SM78Kのインストール・ディレクトリ下（例：c:\necools32）の.\\$smp78kx\\$sm78kxにあります。

- ・uparts32.hには、「アクティブH/L」情報のマクロ定義や、**第4章 関数レファレンス**で説明する関数のIMPORTS宣言が記述されています。
- ・コンパイルの際には、uparts32.hの存在するディレクトリにインクルード・パスを設定してください。

3.4 モジュール定義（DEF）ファイル

Windowsプログラミングのモジュール定義ファイルの記述に従って、次のようなEXPORTS宣言を記述したモジュール定義（DEF）ファイルを作成する必要があります。

IMPORTS宣言については、Uparts32.hで記述済みですので考慮する必要はありません。

3.4.1 EXPORTS宣言

ユーザ関数、シミュレーション・コール関数は、必ずEXPORTS宣言してください。

また、プロジェクト・ファイルの読み込み関数、プロジェクト・ファイルの保存関数、リセット関数などを使用する場合には、それらもEXPORTS宣言する必要があります。

```
例      EXPORTS      UParts_usrwin
                          UPdata_usrwin
```

第4章 関数レファレンス

4.1 <入出力パネル> ウィンドウによるカスタマイズ

ユーザ関数内でコールするだけで、<入出力パネル> ウィンドウに部品を張り付け、シミュレーション処理もすべて行う関数を次に示します。

関数は端子とアクション情報をパラメータとして与えるだけで容易に部品を作成できます。

ただし、ユーザがウィンドウを作成した場合でも、こちらで作成した部品はすべて<入出力パネル> ウィンドウに張り付けられます。

表4 - 1 <入出力パネル> ウィンドウによるカスタマイズ関数

関数名	プロト・タイプ	ページ
プッシュ・ボタン関数	UpPushBtm(<i>pname, actype, btmname</i>)	28
トグル・ボタン関数	UpTglBtm(<i>pname, actype, btmname</i>)	29
グループ式選択ボタン (排他的プッシュ・ボタン) 関数	UpSelectBtm(<i>gname, pnames, pnum, actype, btmnames</i>)	30
保有時間設定関数	UpSetPBtmtime(<i>time</i>)	32
LED関数	UpLed(<i>pname, actype, ledname, pictype</i>)	33
ポート単位で設定できるLED関数	UpPortLed(<i>portname, actype, ledname, pictype</i>)	35
マトリクスLED関数	UpMtxLed(<i>pnames1, pnames2, pnum1, pnum2, actype1, actype2</i>)	37
DCモータ関数	UpDcMtr(<i>pname, actype, mtrname</i>)	39
ステッピング・モータ関数	UpStpingMtr(<i>pnames, num, actype, reiji, step</i>)	40
縦型スクロール・バー式アナログ入力関数	UpScaleInterAD(<i>pname, adname</i>)	42
基準電圧値設定関数	UpSetAVref(<i>avref</i>)	44
ボタン用ビット・マップ設定関数	UpSetBtmBmp(<i>actbmp, nactbmp</i>)	45
LED用ビット・マップ設定関数	UpSetLedBmp(<i>actbmp, nactbmp</i>)	46
DCモータ用ビット・マップ設定関数	UpSetMtrBmp(<i>actbmp, nactbmp</i>)	47
LED図形設定関数	UpSetLedPic(<i>type, color</i>)	48
シリアル端子データ入力関数	UpSerial_data(<i>serpname, data, count, first, bitnum</i>)	49
ウィンドウのタイトル関数	UpPanelTitleName(<i>title</i>)	50
ビット・マップ表示関数	UpSetUsrBmp(<i>bmppname</i>)	51
文字列表示関数	UpWriteString(<i>string</i>)	52

プッシュ・ボタン関数

```
void    UpPushBtm(pname, actype, btmname)
char    *pname;          /* 端子名      */
int     actype;         /* アクティブH/L */
char    *btmname;      /* 機能名      */
```

【機能】

1つのプッシュ式ボタンを作成します。プッシュ・ボタンとは、ボタンをクリックした直後から保有時間だけ入力状態になるボタンです。保有時間は保有時間設定関数UpSetPBtmtime()で設定します。

本関数の直前に記述されているUpSetPBtmtimeで設定した時間を保有時間とします。保有時間の設定がない場合は、デフォルト値0.5msecが設定されます。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。
<i>actype</i>	プッシュ・ボタンにより入力したときの値を指定します。1(ハイ)を入力したいときはHIGH, 0(ロウ)を入力した時はLOW指定します。
<i>btmname</i>	プッシュ・ボタンの機能名を記述します。この機能名はボタン上に表示されるため、文字数は最大半角で16文字分の文字数制限があります。

【戻り値】

なし

【例】

```
UpSetPBtmtime(50);
UpPushBtm("p20",HIGH,"スタート");
UpPushBtm("p20",LOW,"ストップ");
```

図4 - 1 プッシュ・ボタン



トグル・ボタン関数

```
void    UpTglBtm(pname, actype, btmname)
char    *pname;          /* 端子名      */
int     actype;          /* アクティブH/L */
char    *btmname;       /* 機能名      */
```

【機能】

1つのトグル式ボタンを作成します。トグル式ボタンとは、ボタンをクリックして、次に同じボタンをクリックするまで入力状態を保持するボタンです。

初期状態は非アクティブであり、最初にボタンをクリックして入力する値がパラメータ*actype*で設定した値です。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。
<i>actype</i>	トグル・ボタンにより入力した時の値を指定します。1（ハイ）を入力したいときはHIGH，0（ロウ）を入力したときはLOWを指定します。
<i>btmname</i>	トグル・ボタンの機能名を指定します。この機能名はボタン上に表示されるため、文字数は最大半角で16文字分の文字数制限があります。

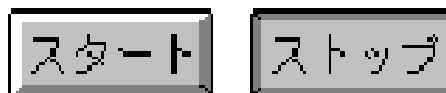
【戻り値】

なし

【例】

```
UpTglBtm("p22",HIGH,"スタート");
UpTglBtm("p23",LOW,"ストップ");
```

図4-2 トグル・ボタン



グループ式選択ボタン（排他的プッシュ・ボタン）関数

```
void    UpSelectBtm(gname, pnames, pnum, actype, btmnames)
char    *gname;          /* グループ名 */
char    **pnames;        /* 端子名 */
int     pnum;            /* ボタンの数 */
int     actype;          /* アクティブH/L */
char    **btmnames;     /* 機能名 */
```

【機能】

複数のボタンをグループとする排他的ボタンを作成します。グループ・ボタンは枠で囲まれグループ・ボタンの中でクリックしたボタンのみアクティブな値が入力されます。

また、その入力状態は次に他のボタンをクリックするまで保持されます。つまり、グループ・ボタンの中でアクティブなボタンは1つだけとなります。

【パラメータ】

<i>gname</i>	グループに付ける名前を記述します。このグループ名はグループ式選択ボタンの先頭に表示されます。
<i>pnames</i>	端子名の文字列をボタンの数だけ記述します。
<i>pnum</i>	ボタンの数を記述します。
<i>actype</i>	グループ式選択ボタンにより入力したときの値を指定します。1（ハイ）を入力したいときはHIGH，0（ロウ）を入力したときはLOWを指定します。グループ・ボタンのすべてのアクティブ状態は同じです。
<i>btmnames</i>	ボタン1つ1つに付ける名前を記述します。この機能名はボタン上に表示されるため、文字数は最大半角で10文字分の文字数制限があります。

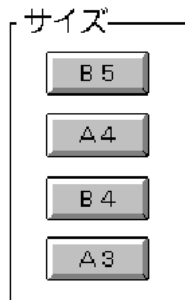
【戻り値】

なし

【 例 】

```
static char *sizePin[4] = {"p30","p31","p32","p33"};  
static char *sizeName[4] = {"B5","A4","B4","A3"};  
UpSelectBtm("サイズ", sizePin, 4, HIGH, sizeName);
```

図4 - 3 グループ式選択ボタン



保有時間設定関数

```
void    UpSetPBmtime(time)
char    *time;          /* 保有時間 */
```

【機能】

プッシュ・ボタンの保有時間を設定します。

【パラメータ】

time 保有時間文字列を設定します。設定の単位はミリ秒です。
設定できる保有時間の範囲は、0.001 ~ 999 msecとなります。

【戻り値】

なし

【例】

```
UpSetPBmtime("0.2");
```

LED関数

```
void    UpLed(pname, actype, ledname, pictype)
char    *pname;          /* 端子名      */
int     actype;         /* アクティブH/L */
char    *ledname;       /* 機能名      */
char    pictype;        /* 図の形式    */
```

【機能】

1つのLEDを作成します。

指定した端子がアクティブな値になったときにアクティブ・ビット・マップ（あるいは色つき図形）を表示し、非アクティブな値になったときに非アクティブ・ビット・マップ（あるいは色なし図形）を表示します。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。				
<i>actype</i>	LEDに表示させるときの値を指定します。アクティブ・ハイのときはHIGH、アクティブ・ロウのときはLOWを指定します。				
<i>ledname</i>	LEDの機能名を記述します。この機能名はLEDの上に表示し、文字数制限はありません。				
<i>pictype</i>	LED表示の図形を指定します。 <table> <tbody> <tr> <td>1の場合</td> <td>デフォルトでは、電球型のビット・マップを表示します。 UpSetLedBmp()でビット・マップを指定した場合には、そのビット・マップを表示します。</td> </tr> <tr> <td>0の場合</td> <td>デフォルトでは、長方形を表示します。 UpSetLedPic()で図形を指定した場合には、その図形を表示します。</td> </tr> </tbody> </table>	1の場合	デフォルトでは、電球型のビット・マップを表示します。 UpSetLedBmp()でビット・マップを指定した場合には、そのビット・マップを表示します。	0の場合	デフォルトでは、長方形を表示します。 UpSetLedPic()で図形を指定した場合には、その図形を表示します。
1の場合	デフォルトでは、電球型のビット・マップを表示します。 UpSetLedBmp()でビット・マップを指定した場合には、そのビット・マップを表示します。				
0の場合	デフォルトでは、長方形を表示します。 UpSetLedPic()で図形を指定した場合には、その図形を表示します。				

【戻り値】

なし

【 例 】

```
UpLed("p40",LOW,"予約中",1);
```

```
UpLed("p21",HIGH,"電源",1);
```

図4 - 4 ビット・マップ表示の非アクティブLED (左) , アクティブLED (右)



```
UpLed("p41",LOW," L ",0);
```

```
UpLed("p22",HIGH," H ",0);
```

図4 - 5 図形表示の非アクティブLED (左) , アクティブLED (右)



ポート単位で設定できるLED関数

```
void      UpPortLed(portname, actype, ledname, pictype)
char      *portname;          /* ポート名      */
unsigned  char  actype;       /* アクティブH/L */
char      *ledname;          /* 機能名        */
char      pictype;           /* 図の形式      */
```

【機能】

1ポート内の各端子に対応したLEDを作成します（つまり8個のLEDで1セットとなります）。
 おおのこの端子が、アクティブな状態のときはアクティブ・ビット・マップ（または色つき図形）を表示し、
 非アクティブな状態のときは非アクティブ・ビット・マップ（または色なし図形）を表示します。

【パラメータ】

pnames ポート名を文字列で記述します。
actype アクティブ・ビット・マップを表示させるときの値を指定します。値が1（ハイ）のときアクティブならば1, 0（ロウ）のときはアクティブならば0を指定します。
 8つのLEDの状態を8ビット・データで1ビットずつ指定します。ポートの最下位端子を最下位ビットとして1ビットずつ順に8つ分設定します。

例

ポート3のLEDでp30, p31がアクティブ・ロウで、その他がアクティブ・ハイの場合（*actype*の下位2ビットを0にする）

```
UpPortLed("p3", 0xfc, "枚数", 1);
```

ledname LEDに付ける名前を記述します。この機能名は表示したビット・マップの下に表示し、文字数制限はありません。
pictype LED表示の図形を指定します。1の場合はビット・マップ, 0の場合は長方形の図形となります。

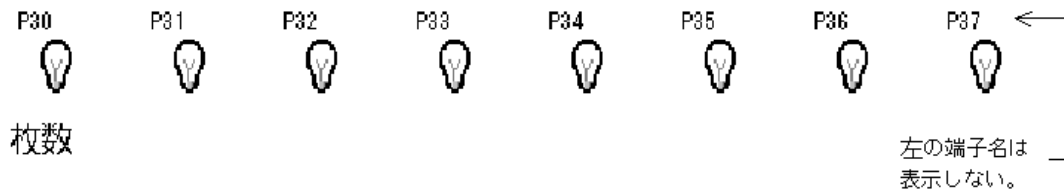
【戻り値】

なし

【 例 】

```
UpPortLed("p3",0xfc,"枚数",1);
```

図4 - 6 ポート単位で設定できるLED関数



マトリクスLED関数

```
void    UpMtxLed(pnames1, pnames2, pnum1, pnum2, actype1, actype2)
char    **pnames1;          /* 出力1端子名列 */
char    **pnames2;          /* 出力2端子名列 */
int     pnum1;              /* 出力1端子数   */
int     pnum2;              /* 出力2端子数   */
int     actype1;            /* 出力1のアクティブH/L */
int     actype2;            /* 出力2のアクティブH/L */
```

【機能】

マトリクス上のLEDを作成します。出力1端子と出力2端子のマトリクス上の交点でどちらもアクティブになったときにアクティブなビット・マップ（ビット・マップは指定できません，固定です）を表示するマトリクスLEDを作成します。

【パラメータ】

<i>pnames1</i>	出力1の端子名文字列を出力1端子の数だけ記述します。
<i>pnames2</i>	出力2の端子名文字列を出力2端子の数だけ記述します。
<i>pnum1</i>	出力1の端子数を記述します。
<i>pnum2</i>	出力2の端子数を記述します。
<i>actype1</i>	出力1に表示させるときの値を指定します。アクティブ・ハイのときはHIGH，アクティブ・ロウのときはLOWを指定します。出力1のアクティブな状態は，すべて同じとします。
<i>actype2</i>	出力2に表示させるときの値を指定します。アクティブ・ハイのときはHIGH，アクティブ・ロウのときはLOWを指定します。出力2のアクティブな状態は，すべて同じとします。

【戻り値】

なし

【 例 】

```
static char *out1[4] = {"p30","p31","p32","p33"};
static char *out2[4] = {"p24","p25","p26","p27"};
UpMtxLed(out1, out2, 4, 4, HIGH, HIGH);
```

図4 - 7 マトリクスLED関数



DCモータ関数

```
void UpDcMtr(pname, actype, mtrname)
char *pname;          /* 端子名 */
int actype;          /* アクティブH/L */
char *mtrname;      /* 機能名 */
```

【機能】

DCモータを作成します。指定した端子がアクティブになったときにアクティブ・ビット・マップを表示し、非アクティブになったときに非アクティブ・ビット・マップを表示します。

また、シミュレーションを開始してからのトータルなアクティブ時間を表示します。表示単位はシステム・メイン・クロックで、リセットがかかるか、10進数で10桁の値を越えた場合に0クリアします。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。
<i>actype</i>	モータがアクティブな表示をするときの状態を指定します。アクティブ・ハイのときはHIGH、アクティブ・ロウのときはLOWを指定します。
<i>mtrname</i>	DCモータの機能名を記述します。この機能名はモータの下に表示します。文字数の制限はありません。

【戻り値】

なし

【例】

```
UpDcMtr("p41",HIGH,"回転モータ");
```

図4 - 8 アクティブLED (左) , 非アクティブLED (右)



ステッピング・モータ関数

```
void    UpStpingMtr(pnames,num,actype,,reiji,step)
char    **pnames;          /* 端子名 */
int     num;              /* 1チャンネルの端子数 */
int     actype;          /* アクティブH/L */
char    reiji;           /* 励磁方式 */
short   step;            /* 最小ステップ角 */
```

【機能】

複数の端子により回転するステッピング・モータを作成します。

モータの回転方向に従ってモータを表示するとともに、回転数、角度を表示します。

【パラメータ】

<i>pname</i>	端子名文字列を端子の数だけ記述します。
<i>num</i>	1チャンネルの端子の数（4または8）を指定します。
<i>actype</i>	モータがアクティブな表示をするときの状態を指定します。アクティブ・ハイのときはHIGH、アクティブ・ロウのときはLOWを指定します。すべての端子のアクティブ状態は同じです。
<i>reiji</i>	励磁方式を指定します。1相の場合0、1-2相の場合1を設定します。
<i>step</i>	最小ステップ角は360を割り切れる整数を設定します。

【戻り値】

なし

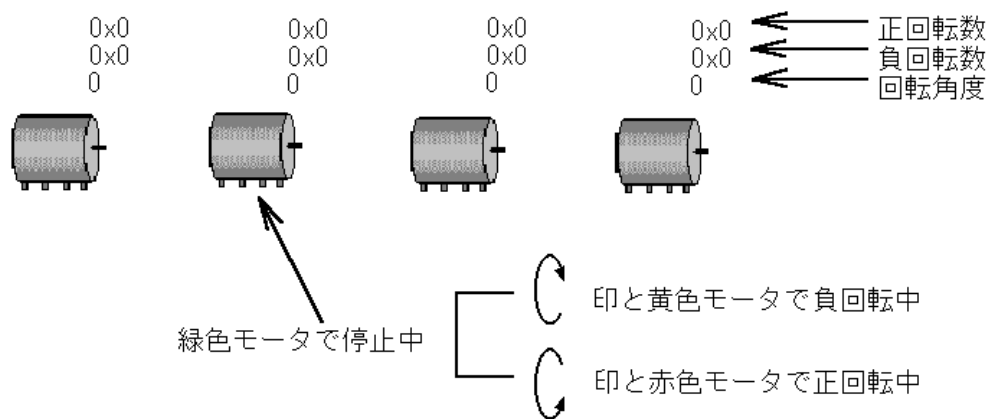
【備考】

この関数の動作開始は、接続された端子に最初に0以外の値を出力したときで、その値を初期値とします。このとき、ステッピング・モータは停止状態で表示され、回転は行いません。

【 例 】

```
char *mtrpin[4] = {"p00","p01","p02","p03"};
UpStpingMtr(mtrpin, 4,HIGH,1,10);
```

図4 - 9 ステッピング・モータ



縦型スクロール・バー式アナログ入力関数

```
void    UpScaleInterAD(pname, adname)
char    *pname;          /* 端子名 */
char    *adname;        /* 機能名 */
```

【機能】

縦型のスクロール・バー式のアナログ入力部品を作成します。

スクロール・サムを移動し、スクロール・バー上でマウスの右ボタンをクリックすることによりアナログ・データを入力し、入力した値を赤色に表示する部品を作成します。

【パラメータ】

<i>pname</i>	アナログ入力端子名を文字列で記述します。
<i>adname</i>	スクロール・バー式入力部品の機能名を記述します。この機能名はスクロール・バー式入力部品の上に表示し、文字数制限はありません。

【戻り値】

なし

【備考】

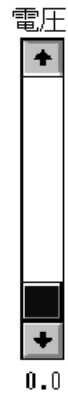
スクロール・バーの動作範囲は、基準電圧値設定関数UpSetAVref()で設定するか、標準のレベル・ゲージ端子設定ダイアログ[※]の基準電圧値設定により決定されます。どちらの設定もなかった場合は、デフォルト値5.0Vとして動作します。

注 SM78Kシリーズ システム・シミュレータ Ver.2.30以上 操作編 (Windowsベース) (U15373J) の第6章 ウィンドウ・レファレンスを参照してください。

【 例 】

```
UpScaleInterAD("ani1","電圧");
```

図4 - 10 縦型スクロール・バー式アナログ入力



基準電圧値設定関数

```
void    UpSetAVref(avref)
char    *avref;          /* 基準電圧値 */
```

【機能】

A/Dコンバータ用の基準電圧値を設定する関数です。

この基準電圧値により、アナログ入力部品の動作範囲が決定します。

動作電源電圧の範囲内（各デバイスのユーザーズ・マニュアル参照）の設定ができます。

ただし、小数点第1位まで設定できますが、それ以降は切り捨てられます。

【パラメータ】

avref 基準電圧値を文字列で記述します。

【戻り値】

なし

【備考】

標準の設定とこの関数で設定しなかった場合は、デフォルト値5.0 Vとしてアナログ入力部品は動作します。

【例】

```
UpSetAVref("3.5");
```

ボタン用ビット・マップ設定関数

```
void    UpSetBtmBmp(actbmp,nactbmp)
char    *actbmp;          /* アクティブ・ビット・マップ名文字列 */
char    *nactbmp;        /* 非アクティブ・ビット・マップ名文字列 */
```

【機能】

ボタンのビット・マップを設定する関数です。設定したいボタン関数の直前に設定することによりボタン表示を変更できます。次にこの関数で設定するまで、ビット・マップ表示は同じです。ビット・マップ・ファイルはシミュレータ本体が存在するディレクトリか、フル・パスでファイル名を指定してください。

【パラメータ】

actbmp アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。
nactbmp 非アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。

【戻り値】

なし

【備考】

この関数を事前に記述せずにボタン関数を記述すると、標準のボタンのビット・マップ（図4-1のイメージ）を表示します。この関数設定時には、ボタン名は表示されません。

【例】

```
UpSetBtmBmp("on.bmp","off.bmp");
UpPushBtm("p21",LOW,"スタート");
```

LED用ビット・マップ設定関数

```
void    UpSetLedBmp(actbmp,nactbmp)
char    *actbmp;          /* アクティブ・ビット・マップ名文字列 */
char    *nactbmp;        /* 非アクティブ・ビット・マップ名文字列 */
```

【機能】

LEDのビットマップを設定する関数です。設定したいLED関数の直前に設定することによりLED表示を変更できます。ただし、この関数はLED関数でビット・マップを指定した場合に有効となります。次にこの関数で設定するまで、ビット・マップ表示は同じです。ビット・マップ・ファイルはシミュレータ本体が存在するディレクトリか、フル・パスでファイル名を指定してください。

【パラメータ】

<i>actbmp</i>	アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。
<i>nactbmp</i>	非アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。

【戻り値】

なし

【備考】

この関数を事前に記述せずにLED関数を記述すると、標準のLEDのビット・マップ（図4 - 4のイメージ）を表示します。

【例】

```
UpSetLedBmp("lighton.bmp","lightoff.bmp");
UpLed("p31",HIGH,"電源",1);
```


DCモータ用ビット・マップ設定関数

```
void    UpSetMtrBmp(actbmp,nactbmp)
char    *actbmp;          /* アクティブ・ビット・マップ名文字列 */
char    *nactbmp;        /* 非アクティブ・ビット・マップ名文字列 */
```

【機能】

DCモータのビット・マップを設定する関数です。設定したいDCモータ関数の直前に設定することによりDCモータ表示を変更できます。次にこの関数で設定するまで、ビット・マップ表示は同じです。ビット・マップ・ファイルはシミュレータ本体が存在するディレクトリか、フル・パスでファイル名を指定してください。

【パラメータ】

<i>actbmp</i>	アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。
<i>nactbmp</i>	非アクティブなときに表示するビット・マップ・ファイル名文字列を記述します。

【戻り値】

なし

【備考】

この関数を事前に記述せずにDCモータ関数を記述すると、標準のDCモータのビット・マップ（図4 - 8のイメージ）を表示します。

【例】

```
UpSetMtrBmp("trun.bmp","stop.bmp");
UpDcMtr("p32",HIGH,"モータ");
```

LED図形設定関数

```
void    UpSetLedPic(type,color)
char    type;          /* 図形のタイプ */
char    color;        /* アクティブ時に図形を塗りつぶす色 */
```

【機能】

LED表示で使用する図形とアクティブ時の色を設定する関数です。設定したいLED関数の直前に設定することによりLED表示を変更できます。この関数はLED関数で図形を指定した場合のみ有効です。次にこの関数で設定するまで、図形表示は同じです。

【パラメータ】

<i>type</i>	図形のタイプを指定します（マクロは、uparts32.hで定義しています）。 マクロ PIC_RECT：長方形 マクロ PIC_ELL：円 となります。
<i>color</i>	アクティブ時に塗りつぶす色を指定します（マクロは、uparts32.hで定義しています）。 マクロ PIC_RED：赤 マクロ PIC_YELLOW：黄 マクロ PIC_GREEN：緑 となります。

【戻り値】

なし

【例】

```
UpSetLedPic(PIC_RECT,PIC_GREEN);
UpLed("p32",HIGH,"テスト",0);
```

シリアル端子データ入力関数

```

void    UpSerial_data(serpname,data,count,first,bitnum)
char    *serpname;          /* シリアル端子名文字列 */
unsigned short *data;       /* データの配列へのポインタ */
unsigned short count;       /* データの配列の個数 */
char    first;              /* 先頭ビット(MSBかLSB) */
char    bitnum;             /* 転送データのビット数 */

```

【機能】

シリアル端子に転送データのビット数を1単位とし、データを指定した先頭ビットから順番に値を設定する関数です。

【パラメータ】

<i>serpname</i>	シリアル・データ入力端子名文字列を記述します。
<i>data</i>	シリアル・データ入力端子に設定する値を転送データ・ビット数単位で格納した配列へのポインタを記述します。
<i>count</i>	シリアル・データ入力端子に設定する値を転送データ・ビット数単位で格納した配列の個数を記述します。
<i>first</i>	転送データのビット数分のデータをMSBから順に値に設定するかLSBから順に設定するかを指定します。MSBからの場合1, LSBからの場合0を設定します。
<i>bitnum</i>	転送データのビット数を指定します。 UART (Universal Asynchronous Receiver/Transmitter) の場合は、スタート・ビット、パリティ・ビット、ストップ・ビットもデータおよびデータ・ビット数に含まれます。

【戻り値】

なし

【例】

シリアル端子SER1に8ビット・データをLSBから順に設定する場合

```
unsigned short data[3] = {0xff, 0xa0, 0x3b};
```

```
UpSerial_data("SER1",data,3,0,8);
```

SER1にデータは、次のように入っていきます。

```
111111110000010111011100
```

ウインドウのタイトル関数

```
void    UpPanelTitleName(title)
char    *title;          /* タイトル名 */
```

【機能】

<入出力パネル> ウインドウのタイトル・バーに名前を表示します。

【パラメータ】

title <入出力パネル> ウインドウのタイトル・バーに表示したい名前を文字列で記述します。

【戻り値】

なし

【例】

```
UpPanelTitleName("プリンタ用システム");
```

ビット・マップ表示関数

```
void    UpSetUsrBmp(bmpname)  
char    *bmpname;        /* ビットマップ・ファイル名 */
```

【機能】

シミュレーションに関係なく常に指定したビット・マップを表示します。

ビット・マップは現在表示している部品のうち、最も右下にあるものの右となりに表示します。右に表示するとウィンドウ内に収まらなくなる場合には下に表示します。

【パラメータ】

bmpname ビット・マップ・ファイル名を文字列で記述します。ビット・マップ・ファイルは、シミュレータ本体が存在するディレクトリ上のファイル名かフル・パスでファイル名を指定してください。

【戻り値】

なし

【例】

```
UpSetUsrBmp("printer.bmp");
```

文字列表示関数

```
void UpWriteString(string)
char *string;          /* 表示文字列 */
```

【機能】

文字列を表示します。

文字列は現在表示している部品のうち、最も右下にあるものの右となりに表示します。右に表示するとウィンドウ内に収まらなくなる場合には下に表示します。

【パラメータ】

string 表示したい文字列を記述します。文字数の制限はありません。

【戻り値】

なし

【例】

```
UpWriteString("電源");
```

4.2 ユーザ・ウィンドウによるカスタマイズ

ユーザが作成したウィンドウと部品で自由にカスタマイズができるように次の関数が用意されています。ユーザ・ウィンドウのハンドラ通知関数により、ウィンドウ処理やユーザ部品からの入力ができるようになり、シミュレーション・コール関数によりユーザ部品への出力表示処理ができます。

表4-2 ユーザ・ウィンドウによるカスタマイズ関数

関数名	プロト・タイプ	ページ
ウィンドウのハンドル通知関数	UpSetUserWnd(<i>hUwnd</i>)	54
ウィンドウ・クローズ関数	UpCloseUserWnd(<i>hwnd</i>)	55
シミュレーション・コール関数	UpCallFuncName(<i>fname</i>)	56
モータ端子通知関数	UpInitMtrPin(<i>pname, actype</i>)	57
ステッピング・モータ通知関数	UpInitStpingMtr(<i>pname, num, actype, reiji, step</i>)	58
端子のアクティブ値通知関数	UpInitPin(<i>pname, actype</i>)	59
ポートのアクティブ値通知関数	UpInitPort(<i>portname, actype</i>)	60
AD入力端子の通知関数	UpInitAD(<i>pname</i>)	61
プロジェクト・ファイル読み込み関数名の通知関数	UpLoadProjName(<i>funcname</i>)	62
プロジェクト・ファイル保存関数名の通知関数	UpSaveProjName(<i>funcname</i>)	63
リセット関数名の通知関数	UpResetFuncName(<i>funcname</i>)	64
端子の値取得関数	UpGetPin(<i>pname, val</i>)	65
ポートのデータ取得関数	UpGetPort(<i>portname, data</i>)	66
DA出力端子の値取得関数	UpGetDA(<i>pname, val</i>)	67
メモリ領域のデータ取得関数	UpGetMem(<i>addr, data</i>)	68
DCモータ用アクティブ時間クリア関数	UpClrMtrAcClk(<i>pname</i>)	69
ステッピング・モータ情報取得関数	UpGetStpingMtr(<i>pnames, num, posrev, negrev, angle</i>)	70
制御レジスタのデータ取得関数	UpGetReg(<i>type</i>)	71
端子に値をセットする関数	UpSetPin(<i>pname, val, time</i>)	72
ポートにデータをセットする関数	UpSetPort(<i>portname, data, time</i>)	73
AD入力端子に値をセットする関数	UpSetAD(<i>pname, val</i>)	74
メモリ領域にデータをセットする関数	UpSetMem(<i>addr, data</i>)	75
モータ用アクティブ時間通知関数	UpGetMtrAcClk(<i>pname, val, actime</i>)	76
制御レジスタにデータをセットする関数	UpSetReg(<i>type, data</i>)	77
1メイン・システム・クロック秒換算通知関数	UpSimtimeSec(<i>void</i>)	78
USB機能でHOSTからパケット送信を行う関数	UpSetUSBPack(<i>total, total_bit, data</i>)	79
USB機能でFunctionからパケットを受け取る関数	UpGetUSBPack(<i>total, data</i>)	80
USB機能でHOSTからの信号送信関数	UpSetUSBSig(<i>sig</i>)	81
USB機能でFunctionからの信号を受け取る関数	UpGetUSBSig(<i>sig</i>)	82

ウインドウのハンドルの通知関数

```
void      UpSetUserWnd(hUwnd)
HANDLE    hUwnd;      /* ユーザ・ウインドウのハンドル */
```

【機能】

ユーザが作成したウインドウのハンドルをシミュレータに通知します。
この関数はユーザがウインドウを作成した直後に記述します。

【パラメータ】

hUwnd ユーザが作成したウインドウのハンドルです。

【戻り値】

なし

【例】

```
HWND      hwnd;
hwnd = CreateWindow(.....);
UpSetUserWnd(hwnd);
```


ウインドウ・クローズ関数

```
void      UpCloseUserWnd(hwnd)
HWND      hwnd;      /*クローズするウインドウ・ハンドル*/
```

【機能】

ユーザが作成したウインドウをクローズすることをシミュレータに通知します。

ユーザが作成したウインドウ・コール・バック関数のメッセージWM_DESTROYでこの関数を記述します。

【パラメータ】

hwnd ユーザが作成した、クローズするウインドウのハンドルです。

【戻り値】

なし

【例】

```
WM_DESTROY:
    :
    :
    UpCloseUserWnd(hwnd);
```

シミュレーション・コール関数

```
void    UpCallFuncName(fname)
char    *fname;          /* シミュレーション・コール関数名 */
```

【機能】

シミュレーションの一定間隔^注でシミュレータからコールしてもらう関数名を通知します。
この関数はユーザ関数UParts_xxx()の中に記述しておかなければなりません。

注 1命令実行中に1度コールされます。

【パラメータ】

fname シミュレータからコールしてもらう関数名を記述します。

【戻り値】

なし

【備考】

シミュレーション・コール関数は次のような関数仕様にしてください。

unsigned long型のパラメータで、シミュレーションの実行時間を受け取ります。

シミュレーションの実行時間とは、前回コールされてからの経過時間であり、メイン・システム・クロック単位となります。

また、シミュレーション・コール関数はモジュール定義ファイルでEXPORTS宣言を（3. 4. 1 EXPORTS宣言を参照）してください。

```
void FAR PASCAL Update_usrwin(unsigned long simtime)
```

【例】

```
UpCallFuncName("Update_usrwin");
```

モータ端子通知関数

```
void    UpInitMtrPin(pname,actype)
char    *pname;          /* 端子名 */
int     actype;          /* アクティブH/L */
```

【機能】

モータ用アクティブ時間通知関数でモータの値とアクティブ時間を取得するために、モータ用に指定する端子名を通知する関数です。

モータ用端子を使用する場合は、この関数をユーザ関数UParts_xxx()の中に記述しておく必要があります。

モータ用端子を使用しない場合には、この関数を記述する必要はありません。

【パラメータ】

<i>pname</i>	モータに接続した端子名を文字列で記述します。
<i>actype</i>	モータがアクティブな状態を指定します。アクティブ・ハイの場合はHIGH、アクティブ・ロウの場合はLOWを指定します。

【戻り値】

なし

【備考】

ユーザ関数内であらかじめ通知しなければ、シミュレーション中にモーター用アクティブ時間通知関数UpGetMtrAcClk()で情報を取り込んでも値は保証されません。

【例】

```
UpInitMtrPin("p41",HIGH)
```

ステッピング・モータ通知関数

```
int    UpInitStpingMtr(pnames,num,actype,reiji,step)
char  **pnames;          /* 端子名 */
int    num;              /* 1チャンネルの端子数 */
int    actype;          /* アクティブH/L */
char   reiji;           /* 励磁方式 */
short  step;            /* 最小ステップ角 */
```

【機能】

複数の端子により回転するステッピング・モータを指定した端子に接続します。

ステッピング・モータを使用する場合は、この関数をユーザ関数UParts_xxx()の中に記述してください。ただし、ステッピング・モータを使用しない場合には、この関数を記述する必要はありません。

【パラメータ】

<i>pnames</i>	端子名文字列を端子の数だけ記述します
<i>num</i>	1チャンネルの端子の数（4または8）を指定します。
<i>actype</i>	モータがアクティブな表示をするときの状態を指定します。アクティブ・ハイの場合はHIGH、アクティブ・ロウの場合はLOWを指定します。すべての端子のアクティブ状態は同じです。
<i>reiji</i>	励磁方式を指定します。1相の場合0、1-2相の場合1を設定します。
<i>step</i>	最小ステップ角は360を割り切れる整数を設定します。

【戻り値】

正しく設定できた場合 1

正しく設定できなかった場合 0

【例】

```
char *mtrpin[4] = {"p00","p01","p02","p03"};
UpInitStpingMtr(mtrpin, 4,HIGH,1,10);
```

端子のアクティブ値通知関数

```
int    UpInitPin(pname,actype)
char  *pname;      /* 端子名 */
int    actype;     /* 端子のアクティブ値 */
```

【機能】

1つの端子のアクティブ状態の値を設定します。

端子に値を入力することがある場合は、この関数をユーザ関数UParts_xxx()の中に記述してください。ただし、端子に値を入力しない場合には、この関数を記述する必要はありません。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。
<i>actype</i>	端子のアクティブ値を指定します。アクティブ・ハイの場合はHIGH, アクティブ・ロウの場合はLOWを指定します。

【戻り値】

正しく端子のアクティブ値を設定できた場合 1

正しく端子のアクティブ値を設定できなかった場合 0

【例】

端子P46をアクティブ・ハイ（入力値が1のとき）で動作するように設定する場合

```
int ret;
ret=UpInitPin("P46",HIGH);
```

ポートのアクティブ値通知関数

```
int          UpInitPort(portname,actype)
char        *portname;          /* ポート名 */
unsigned    char    actype;      /* ポートのアクティブ値 */
```

【機能】

1つのポート内の各端子のアクティブ状態の値を設定します。

ポートに値を入力することがある場合は、この関数をユーザ関数UParts_xxx()の中に記述してください。ただし、ポートに値を入力しない場合には、この関数を記述する必要はありません。

【パラメータ】

portname ポート名を文字列で記述します。

actype ポートの各端子のアクティブ値を指定します。
 ポートの各端子に対し、アクティブ・ハイの場合は1、アクティブ・ロウの場合は0を指定します。
 ポートの最下位端子を最下位ビットとして1ビットずつ順に8つ分設定します。

【戻り値】

正しくポートのアクティブ値を設定できた場合 1

正しくポートのアクティブ値を設定できなかった場合 0

【例】

ポート4の端子のうち、P40, P41がアクティブ・ハイ、P42からP47がアクティブ・ロウの設定の場合

```
int ret;
ret=UpInitPort("P4",0x03);
```

ポート2の端子のうち、P27のみがアクティブ・ハイ、P20からP26がアクティブ・ロウの設定の場合

```
int ret;
ret=UpInitPort("P2",0x80);
```

AD入力端子の通知関数

```
int    UpInitAD(pname)
char   *pname;          /* AD入力端子名*/
```

【機能】

ユーザ・オープン・インタフェース関数から値を入力するAD入力端子をシミュレータに通知する関数です。
UpSetAD()でAD入力端子に値を入力することがある場合は、この関数をUParts_XXXX()に記述してください。
ただし、ユーザ・オープン・インタフェース関数で、そのAD入力端子に値を入力することがなければ、この関数を記述する必要はありません。

【パラメータ】

pname AD入力端子名を文字列で記述します。

【戻り値】

正常終了時 1
異常終了時 0 (そのAD入力端子が現在シミュレーション中のデバイスに存在しない場合)

【例】

```
UpInitAD("ANI0");
```

プロジェクト・ファイル読み込み関数名の通知関数

```
void UploadProjName(funcname)
char *funcname; /* プロジェクト・ファイル読み込み関数名 */
```

【機能】

シミュレータのプロジェクト・ファイル読み込み時に、ユーザ・ウインドウの情報もこのプロジェクト・ファイルから同時に読み込む関数名を通知します。

この関数は、ユーザ関数UParts_xxx()の中に記述しておく必要があります。

【パラメータ】

funcname シミュレータからコールしてもらうプロジェクト・ファイル読み込み関数名を指定します。

【戻り値】

なし

【備考】

プロジェクト・ファイル読み込み関数の仕様は、次のようにしてください。

- ・ char FAR *型のパラメータで、プロジェクト・ファイル名文字列を受け取ります。
- ・ パラメータで受け取ったプロジェクト・ファイル名のファイルから、ユーザ・ウインドウの情報を読み取ります。このとき、読み取り時に使用するライブラリは関数GetPrivateProfileStringまたはGetPrivateProfileIntを使用してください。
- ・ ユーザの使用するセクション名は、“ User DLL Window ” とします。
- ・ プロジェクト・ファイル読み込み関数は、モジュール定義ファイルでEXPORTS宣言する必要があります。

```
void FAR PASCAL Upload_usrproj(char FAR *filename)
```

【例】

```
UploadProjName("Upload_usrproj");
```


プロジェクト・ファイル保存関数名の通知関数

```
void    UpSaveProjName(funcname)
char    *funcname;      /* プロジェクト・ファイル保存関数名 */
```

【機能】

シミュレータのプロジェクト・ファイル保存時に、ユーザ・ウインドウの情報もこのプロジェクト・ファイルへ同時に保存する関数名を通知します。

この関数をユーザ関数UParts_xxx()の中に記述しておく必要があります。

【パラメータ】

funcname シミュレータからコールしてもらうプロジェクト・ファイル保存関数名を記述します。

【戻り値】

なし

【備考】

プロジェクト・ファイル保存関数の仕様は、次のようにしてください。

- ・ char FAR *型のパラメータで、プロジェクト・ファイル名文字列を受け取ります。
- ・ パラメータで受け取ったプロジェクト・ファイル名のファイルへ、ユーザ・ウインドウの情報を書き込みます。このとき、書き込み時に使用するライブラリは関数WritePrivateProfileStringを使用してください。
- ・ ユーザの使用するセクション名は、“ User DLL Window ” とします。
- ・ プロジェクト・ファイル保存関数は、モジュール定義ファイルでEXPORTS宣言する必要があります。

```
void FAR PASCAL UpSave_usrproj(char FAR *filename)
```

【例】

```
UpSaveProjName("UpSave_usrproj");
```

リセット関数名の通知関数

```
void    UpResetFuncName(funcname)
char    *funcname;      /* リセット関数名 */
```

【機能】

シミュレータからのCPUリセット発生時に、ユーザ・ウインドウのリセット処理を行う関数名を通知します。
この関数は、ユーザ関数UParts_xxx()の中に記述しておく必要があります。

【パラメータ】

funcname シミュレータからコールしてもらいリセット関数名を記述します。

【戻り値】

なし

【備考】

リセット関数の仕様は、次のようにしてください。

- ・パラメータはないので、VOID型です。
- ・リセット関数は、モジュール定義ファイルでEXPORTS宣言する必要があります。

```
Void FAR PASCAL Upreset_usrwin(VOID)
```

【例】

```
UpResetFuncName("Upreset_usrwin");
```

端子の値取得関数

```
int    UpGetPin(pname,val)
char  *pname;          /* 端子名 */
char  *val;           /* 端子の値を格納する領域へのポインタ */
```

【機能】

1つの端子の値を取得します。

【パラメータ】

<i>pname</i>	端子名を文字列で記述します。
<i>val</i>	端子の値を格納する領域へのポインタを記述します。

【戻り値】

正しく端子の値を取得できた場合 1
正しく端子の値を取得できなかった場合^注 0

注 端子の値が不定値であった場合も0を返します。

【例】

```
char  val;
int    ret;
ret = UpGetPin("p46",&val);
```

ポートのデータ取得関数

```
int          UpGetPort(portname , data)
char         *portname;          /*ポート名 */
unsigned    char   *data;        /*ポートのデータを格納する領域へのポインタ*/
```

【機能】

ポートのデータを取得します。

【パラメータ】

portname ポート名を文字列で記述します。
data ポートのデータを格納する領域へのポインタを記述します。

【戻り値】

正しくポートのデータを取得できた場合 1
正しくポートのデータを取得できなかった場合[※] 0

注 ポートの値に不定値が含まれていた場合にも0を返します。

【例】

```
unsigned char data;
int ret;
ret = UpGetPort("p4",&data);
```

DA出力端子の値取得関数

```
int          UpGetDA(pname,val)
char         *pname;          /*DA出力端子名*/
unsigned     short  *val;     /*DA出力値*/
```

【機能】

DA出力端子の値を設定します。

【パラメータ】

pname DA出力端子名を文字列で記述します。
val DA出力端子の値を格納する領域へのポインタを記述します。

【戻り値】

正常終了時 1
異常終了時^注 0

注 DA出力端子の値が不定値の場合にも0を返します。

【例】

```
unsigned short daval;
UpGetDA("ANO0",&daval);
```

メモリ領域のデータ取得関数

```
int          UpGetMem(addr,data)
unsigned long  addr;          /* アドレス */
unsigned char *data;         /* データ格納領域 */
```

【機能】

メモリ領域内のデータを取得します。

【パラメータ】

addr 取得したいメモリ領域内のアドレスを指定します。
data データ格納領域を記述します。

【戻り値】

正しくデータが取得できた場合 1
データが取得できなかった場合 0

【例】

```
unsigned char  data;
int  ret;
ret = UpGetMem(0xffe000,&data);
```

DCモータ用アクティブ時間クリア関数

```
VOID UpClrMtrAcClk(pname)  
char *pname;      /* 端子名 */
```

【機能】

モータ用に指定した端子のアクティブな時間を0クリアする関数です。

【パラメータ】

pname モータに接続した端子名を文字列で記述します。

【戻り値】

なし

【備考】

この関数を使用する場合は、モータ端子通知関数UpInitMtrPin()をユーザ関数内でコールし、端子名をあらかじめ通知してください。

【例】

```
UpClrMtrAcClk("p41");
```

ステッピング・モータ情報取得関数

```
int      UpGetStpingMtr(pnames,num,posrev,negrev,angle)
char    **pnames;          /* 端子名 */
int      num;              /* 1チャンネルの端子の数(4または8を指定する)*/
unsigned long  *posrev;    /* 正方向回転数を格納する領域 */
unsigned long  *negrev;    /* 負方向回転数を格納する領域 */
unsigned long  *angle;     /* 角度を格納する領域 */
```

【機能】

ステッピング・モータ通知関数UpInitStpingMtrで通知済みの端子名に接続したステッピング・モータの正/負の回転数と現在の角度を取得します。

【パラメータ】

<i>pnames</i>	端子名文字列を端子の数だけ記述します。
<i>num</i>	1チャンネルの端子の数(4または8)を指定します。
<i>posrev</i>	正方向回転数を格納する領域を指定します。
<i>negrev</i>	負方向回転数を格納する領域を指定します。
<i>angle</i>	角度を格納する領域を指定します。

【戻り値】

正しく取得できた場合 1
正しく取得できなかった場合 0

【例】

```
char  *mtrpin[4] = {"p00","p01","p02","p03"};
unsigned long  posrev;
unsigned long  negrev;
unsigned long  angle;
UpInitStpingMtr(mtrpin, 4,HIGH,1,10);
:
UpGetStpingMtr(mtrpin,4,&posrev,&negrev,&angle);
```


制御レジスタのデータ取得関数

```
unsigned long UpGetReg(type)
unsigned char type;      /* 制御レジスタを識別する値 */
```

【機能】

制御レジスタ（PSW, PC, SPのいずれか）のデータを取得します。

【パラメータ】

type データを設定したい制御レジスタの識別値を記述します。
マクロで指定します。

PSW マクロREG_PSW
PC マクロREG_PC
SP マクロREG_SP

【戻り値】

指定した制御レジスタの値。

PCを指定した場合は次に実行する命令のアドレスです。

PSWまたはSPを指定した場合は、現在の命令実行後の結果となります。

【例】

```
unsigned long psw_val;
psw_val = UpGetReg(REG_PSW);
```

端子に値をセットする関数

```
void      UpSetPin(pname,val,time)
char      *pname;          /* 端子名 */
char      val;            /* アクティブ値 */
unsigned  long   time;     /* 保有時間 */
```

【機能】

端子の値を設定します。

【パラメータ】

pname 端子名を文字列で記述します。
val 端子のアクティブ時の値を設定します。
time データを保持する時間を設定します。単位はメイン・システム・クロックです。

【戻り値】

なし

【備考】

この関数を使用する場合は、端子のアクティブ値通知関数UpInitPin()をユーザ関数内でコールし、端子名をあらかじめ通知してください。この関数UpInitPinで通知した端子のアクティブ値をマクロHIGHと設定した場合、この関数UpSetPinのアクティブ値として1を設定すると、端子がアクティブ状態になります。同様にUpInitPinで通知した端子のアクティブ値をマクロLOWとした場合、この関数UpSetPinのアクティブ値として0を設定すると、端子がアクティブ状態になります。また、保有時間に0を設定した場合には、アクティブ値が保持されません。

【例】

UpInitPin("p31",HIGH)と記述し、端子P31をアクティブ・ハイと通知しておいた場合、次の記述で、端子P31にアクティブ・ハイ入力を50メイン・システム・クロック保持することになります。

```
char val;   val = 1;
UpSetPin("p31",val,50L);
```

ポートにデータをセットする関数

```
void      UpSetPort(portname,data,time)
char      *portname;          /* ポート名 */
unsigned  char  data;         /* データ */
unsigned  long  time;         /* 保有時間 */
```

【機能】

ポート単位でデータを設定します。

【パラメータ】

portname ポート名を文字列で記述します。
data ポートに設定する値を記述します。
time データを保持する時間を設定します。単位はメイン・システム・クロックです。

【戻り値】

なし

【備考】

この関数を使用する場合は、ポートのアクティブ値通知関数UpInitPort()をユーザ関数内でコールし、端子名をあらかじめ通知してください。この関数UpInitPortでポート内の各端子のアクティブ値を、アクティブ・ハイならば1、アクティブ・ロウならば0をビット単位で設定しておきます。この関数UpSetPortでポート内のある端子をアクティブ状態にする場合は、ポートに設定するデータのその端子に対応したビット値を、UpSetPortで設定したアクティブ値のその端子に対応したビット値と同じにしてください。また、保有時間に0を設定した場合には、アクティブ値が保持されます。

【例】

UpInitPort("p4",0x03)と記述し、ポートP4の端子P40, P41をアクティブ・ハイ, P42からP47をアクティブ・ロウと通知しておいた場合、次の記述で、ポートP40の端子P40, P42, P43をアクティブ状態で50メイン・システム・クロックの間、保持します。

```
unsigned char data;
data = 0xf1;
UpSetPort("p4",data,50L);
```

AD入力端子に値をセットする関数

```
int          UpSetAD(pname,val)
char         *pname;          /*AD入力端子名*/
unsigned     short   val;      /*AD入力値*/
```

【機能】

AD入力端子の値を設定します。

【パラメータ】

pname AD入力端子名を文字列で記述します。
val AD入力端子に入力する値を設定します。

【戻り値】

正常終了時 1
異常終了時^注 0

注 そのAD入力端子が現在シミュレーション中のデバイスに存在しない場合

【備考】

この関数を使用する場合は、AD入力端子の接続通知関数UpInitAD()をユーザ関数内でコールし、AD入力端子名をあらかじめ通知する必要があります。

【例】

```
unsigned short adval;

adval = 10;
UpSetAD("ANI0",adval);
```

メモリ領域にデータをセットする関数

```
int          UpSetMem(addr,data)
unsigned long  addr;          /* アドレス */
unsigned char  data;         /* データ  */
```

【機能】

メモリ領域内にデータを設定します。

【パラメータ】

addr 設定したいメモリ領域内のアドレスを指定します。
data データを記述します。

【戻り値】

正しく値が設定できた場合 1
正しく値が設定できなかった場合 0

【例】

```
int ret;
ret = UpSetMem(0xffe300, 0x72);
```

モータ用アクティブ時間通知関数

```
int          UpGetMtrAcClk(pname, val, actime)
char        *pname;          /* 端子名 */
char        *val;           /* 値 */
unsigned    long    *actime; /* アクティブな時間 */
```

【機能】

モータ用に指定した端子のアクティブな時間を取り込む関数です。

モータ端子通知関数UpInitMtrPin()であらかじめ作成したモータ部品の端子のみ有効となります。

アクティブ時間とは、シミュレーションが開始してからのトータル時間であり、リセットがかかるか、10進数で10桁を越えた場合に0クリアします。

アクティブ時間はメイン・システム・クロックで表現します。

【パラメータ】

<i>pname</i>	モータに接続した端子名を文字列で記述します。
<i>val</i>	端子の値を設定します。
<i>actime</i>	アクティブ時間を、シミュレーション開始からのトータル時間を二次元配列で表現します。 $\text{actime}[1] \times 0x100000000 + \text{actime}[0]$

例 $\text{actime}[1] = 0x390; \text{actime}[0] = 0x10052688;$
トータル時間 = $0x39010052688$ メイン・システム・クロック

【戻り値】

DCモータ関数で設定した端子 0

DCモータ関数で設定した端子以外の端子を設定した場合 - 1

【備考】

この関数を使用する場合は、モーター端子通知関数UpInitMtrPin()をユーザ関数内でコールし、端子名をあらかじめ通知してください。

【例】

```
char    val;
unsigned long    actime[2];
UpGetMtrAcClk("p41",&val,actime);
wsprintf(timebuf,"回転時間 =  %#lx%08lx \n",actime[1],actime[0]);
TextOut(hdc,240,320,timebuf,sizeof(timebuf));
```

制御レジスタにデータをセットする関数

```
void      UpSetReg(type, data)
unsigned  char   type;          /* 制御レジスタを識別する値 */
unsigned  long   data;         /* データ */
```

【機能】

制御レジスタ (PSW, PC, SPのいずれか) にデータを設定します。

【パラメータ】

type データを設定したい制御レジスタの識別値を記述します。
 マクロで指定します。
 PSW マクロREG_PSW
 PC マクロREG_PC
 SP マクロREG_SP

data 指定した制御レジスタに設定するデータを記述します。
 ただし、レジスタのサイズを越えた部分は無視されます。

【戻り値】

なし

【備考】

この関数で設定した制御レジスタのデータは、1シミュレーションの終わりに設定されます。このため、1命令のシミュレーション途中 (命令により2-3回のシミュレーションを行う場合) に、この関数を使って制御レジスタの値を書き換えても、残りのシミュレーションの結果により制御レジスタの値が変わることがあります。1命令のシミュレーションごとにデータを設定したい場合は、ユーザがUpGetReg関数で得たPCの値が変化したとき、制御レジスタのデータを書き換える方法をお勧めします。

【例】

```
UpSetReg(REG_PSW, 0x2);
```

1 メイン・システム・クロックの秒換算通知関数

```
unsigned long UpSimtimeSec(void)
```

【機能】

1メイン・システム・クロックをナノ秒に換算します。

【パラメータ】

なし

【戻り値】

1メイン・システム・クロックをナノ秒に換算した値を返します。

【例】

```
unsigned long simtime;  
simtime = UpSimtimeSec( );
```


USB機能でHOSTからパケット送信を行う関数

```
BOOL      UpSetUSBPack(total,total_bit,data)
unsigned  char      total;          /* data配列の個数 */
unsigned  char      total_bit;    /* 送信するデータのビット数 */
unsigned  char      *data;        /* パケット・データ配列へのポインタ */
```

【機能】

USB機能を使用してHOST側からのパケット送信情報をセットする関数です。

【パラメータ】

<i>total</i>	パケット・データ配列の個数を記述します。
<i>total_bit</i>	送信するデータをビット数に換算した値を記述します。
<i>data</i>	送信したいパケット・データ配列へのポインタを記述します。

【戻り値】

正常終了時	1
異常終了時	0

【備考】

この関数は、USB機能をもつデバイスでのみサポートしています。

USB機能でFunctionからパケットを受け取る関数

```
void          UpGetUSBPack(total,data)
unsigned char total;      /* data配列の個数 */
unsigned char **data;    /* パケット・データ配列へのポインタ */
```

【機能】

USB機能を使用してFunction側からのパケット情報を受信する関数です。

【パラメータ】

total パケット・データ配列の個数を記述します。
data 送信したいパケット・データ配列へのポインタを記述します。

【戻り値】

なし

【備考】

この関数は、USB機能をもつデバイスでのみサポートしています。

USB機能でHOSTからの信号送信関数

```
void      UpSetUSBSig(sig)
unsigned  char    sig;          /* 送信信号ID */
```

【機能】

USB機能を使用してHOST側からの信号を送信する関数です。

【パラメータ】

sig 送信信号IDを指定します。
0 : USBresetの場合
1 : Resumeの場合

【戻り値】

なし

【備考】

この関数は、USB機能をもつデバイスでのみサポートしています。

USB機能でFunctionからの信号を受け取る関数

```
void      UpGetUSBSig(sig)
unsigned  char    *sig;          /* 受信信号ID */
```

【機能】

USB機能を使用してFunction側からの信号を受信する関数です。

【パラメータ】

sig 受信信号IDを指定します。
 0 : USBresetの場合
 1 : Resumeの場合

【戻り値】

なし

【備考】

この関数は、USB機能をもつデバイスでのみサポートしています。

第5章 CPUリセット時の動作

シミュレータのデバッグ部からCPUリセットが発生したときの、カスタム部品の動作を示します。

5.1 <入出力パネル> ウィンドウによるカスタマイズ部品

<入出力パネル> ウィンドウによるカスタマイズで記述した各関数の部品は次のようになります。

表5 - 1 CPUリセット時の<入出力パネル> ウィンドウによるカスタマイズ部品

部品名	状 態
プッシュ・ボタン	すべて非アクティブ状態になります。
トグル・ボタン	すべて非アクティブ状態になります。
グループ式選択ボタン	すべて押されていない状態になります。
LED	すべて非アクティブ状態になります。
ポート単位設定LED	すべて非アクティブ状態になります。
マトリクスLED	すべて点灯しない状態になります。
DCモータ	すべて非アクティブ状態になり、トータル・アクティブ時間は0になります。
ステッピング・モータ	すべて非アクティブ状態になり、正回転数、負回転数、回転角度は0になります。
縦型スクロール・バー式アナログ入力	入力値0、スクロール・バーのスクロール・サムは下端になります。
シリアル端子データ入力	データの先頭に戻ります。

5.2 ユーザ・ウィンドウによるカスタマイズ部品

シミュレータのデバッグ部からCPUリセットが発生したとき、リセット関数名通知関数 `UpResetFuncName()` で事前に通知されている場合は、ユーザ・ウィンドウのリセット処理関数の処理を行います。

第6章 プログラミング例

この章では、カスタマイズ部品の例を記載します。

以降のソースのうち、

は、ターゲットのプログラムです。

CC78K シリーズでコンパイル、リンクを行い、ロード・モジュール・ファイル (xxxx.lmf) を作成します。

以降は、カスタマイズ部品作成に必要なファイルです。

このマニュアルに従い Visual C++ で、ダイナミック・リンク・ライブラリ (xxxx.DLL) を作成します。

コンパイル時には、/Zp1 オプションを付けてください。

(/Zp1 オプション：構造体メンバのアライメントを 1 バイトにする。)

[プロジェクト] メニューの [設定] で [C/C++] を選択し、“カテゴリ” “コード生成” において“構造体のアライメント”を“1 バイト”に設定してください。

備考 SM78K シリーズ V2.30 以上を使用する場合には、32 ビットのダイナミック・リンク・ライブラリを作成してください。

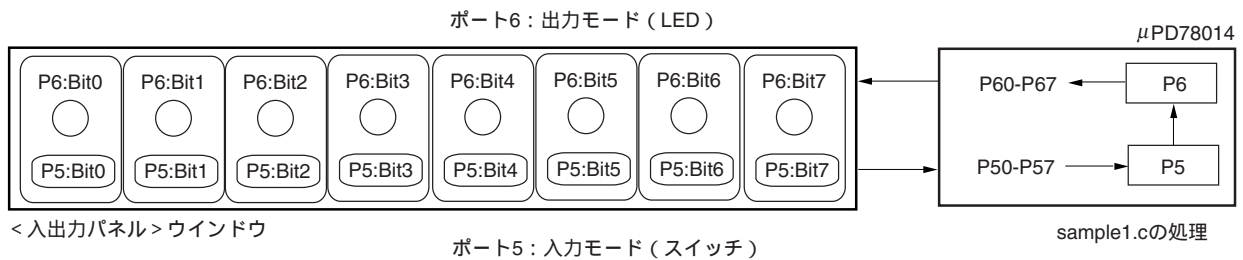
6.1 <入出力パネル> ウィンドウによるカスタマイズ部品例

6.1.1 サンプル内容

8つのLEDと、8つのスイッチ(2つのプッシュ(P50, P51), 2つのトグル(P52, P53), 4つのセレクト(P54, P55, P56, P57))を<入出力パネル> ウィンドウに表示し、スイッチのON/OFFで、各スイッチに対応するLEDがON/OFFするものです。

この例では、次のようになっています。

図 6 - 1 <入出力パネル> ウィンドウによるカスタマイズ部品例



6.1.2 ソース例

ターゲット・プログラム

(1/1) SAMPLE1.C

```
#pragma sfr
void main()
{
    MM = 0xB0;
    PM5 = 0xFF;
    PM6 = 0;
    P6 = 0;

    while(1)
    {
        P6 = P5;
    }
}
```


カスタム品ソース・ファイル UPsw00.c

(1/2) UPsw00.c

```
/*
 *      User Open I/F Sample Program      (UPsw00.c)
 *
 *      P50 (I) : Switch 0          P60 (O) : LED 0
 *      P51 (I) : Switch 1          P61 (O) : LED 1
 *      P52 (I) : Switch 2          P62 (O) : LED 2
 *      P53 (I) : Switch 3          P63 (O) : LED 3
 *      P54 (I) : Switch 4          P64 (O) : LED 4
 *      P55 (I) : Switch 5          P65 (O) : LED 5
 *      P56 (I) : Switch 6          P66 (O) : LED 6
 *      P57 (I) : Switch 7          P67 (O) : LED 7
 */

#include      <Windows.h>
#include      <string.h>

typedef unsigned char    UCHAR;
typedef unsigned short   USHORT;
typedef unsigned long    ULONG;

#include      "uparts32.h"

BOOL WINAPI DllMain(HANDLE, DWORD, LPVOID);

void FAR PASCAL UParts_sw00(void);

/*****
/*      DLL Main
*****/
BOOL WINAPI DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    return(TRUE);
}
```

```

/*****
/*      UParts_sw00(void)                                */
/*****

void FAR PASCAL
UParts_sw00(void)
{
    static char *pin[4] = { "P54", "P55", "P56", "P57" };
    static char *name[4] = { "S Bit4", "S Bit5", "S Bit6", "S Bit7" };

    UpSetPBtmtime("3.0");
    UpLed("P60", HIGH, "Bit0", 1);
    UpLed("P61", HIGH, "Bit1", 1);
    UpLed("P62", HIGH, "Bit2", 1);
    UpLed("P63", HIGH, "Bit3", 1);
    UpLed("P64", HIGH, "Bit4", 1);
    UpLed("P65", HIGH, "Bit5", 1);
    UpLed("P66", HIGH, "Bit6", 1);
    UpLed("P67", HIGH, "Bit7", 1);
    UpPushBtm("P50", HIGH, "P Bit0");
    UpPushBtm("P51", HIGH, "P Bit1");
    UpTglBtm("P52", HIGH, "T Bit2");
    UpTglBtm("P53", HIGH, "T Bit3");
    UpSelectBtm("Select", pin, 4, HIGH, name);
}

/* UPsw00.c */

```

定義ファイル UPsw00.def

(1/1) UPsw00.def

```
LIBRARY UPSW00

DESCRIPTION      'User Open I/F Panel sw00'

CODE             PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD SINGLE

HEAPSIZE        3072

EXPORTS
    UParts_sw00          @2
```

メイク・ファイル UPsw00.mak

(1/4) UPsw00.mak

```
# Microsoft Developer Studio Generated NMAKE File, Based on upsw00.dsp
!IF "$(CFG)" == ""
CFG=upsw00 - Win32 Release
!MESSAGE 構成が指定されていません。デフォルトの upsw00 - Win32 Debug を設定します。
!ENDIF
!IF "$(CFG)" != "upsw00 - Win32 Release" && "$(CFG)" != "upsw00 - Win32 Debug"
!MESSAGE 指定された ビルド・モード "$(CFG)" は正しくありません。
!MESSAGE NMAKE の実行時に構成を指定できます
!MESSAGE コマンド・ライン上でマクロの設定を定義します。例:
!MESSAGE
!MESSAGE NMAKE /f "upsw00.mak" CFG="upsw00 - Win32 Debug"
!MESSAGE
!MESSAGE 選択可能なビルド・モード:
!MESSAGE
!MESSAGE "upsw00 - Win32 Release" ("Win32 (x86) Dynamic-Link Library" 用)
!MESSAGE "upsw00 - Win32 Debug" ("Win32 (x86) Dynamic-Link Library" 用)
!MESSAGE
!ERROR 無効な構成が指定されています。
!ENDIF
!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe
!IF "$(CFG)" == "upsw00 - Win32 Release"
OUTDIR=.\Release
INTDIR=.\Release
# Begin Custom Macros
OutDir=.\Release
# End Custom Macros
ALL : "$(OUTDIR)¥upsw00.dll"
```

```

CLEAN :

    -@erase "$(INTDIR)¥Upsw00.obj"
    -@erase "$(INTDIR)¥vc60.idb"
    -@erase "$(OUTDIR)¥upsw00.dll"
    -@erase "$(OUTDIR)¥upsw00.exp"
    -@erase "$(OUTDIR)¥upsw00.lib"

"$ (OUTDIR) " :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
CPP_PROJ=/nologo /Zp1 /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_MBCS" /D
"_USRDL" /D "UPSW00_EXPORTS" /Fp"$(INTDIR)¥upsw00.pch" /YX /Fo"$(INTDIR)¥¥"
/Fd"$(INTDIR)¥¥" /FD /c
MTL_PROJ=/nologo /D "NDEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)¥upsw00.bsc"
BSC32_SBRS= ¥

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll
/incremental:no /pdb:"$(OUTDIR)¥upsw00.pdb" /machine:I386 /def:".¥Upsw00.def"
/out:"$(OUTDIR)¥upsw00.dll" /implib:"$(OUTDIR)¥upsw00.lib"
DEF_FILE= ¥
    ".¥Upsw00.def"
LINK32_OBJS= ¥
    "$(INTDIR)¥Upsw00.obj" ¥
    ".¥sik032.lib"
"$ (OUTDIR)¥upsw00.dll" : "$ (OUTDIR) " $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
    $(LINK32_FLAGS) $(LINK32_OBJS)
<<
!ELSEIF "$ (CFG) " == "upsw00 - Win32 Debug"
OUTDIR=¥Debug
INTDIR=¥Debug
# Begin Custom Macros
OutDir=¥Debug
# End Custom Macros
ALL : "$(OUTDIR)¥upsw00.dll"

```

```

CLEAN :

    -@erase "$(INTDIR)¥Upsw00.obj"
    -@erase "$(INTDIR)¥vc60.idb"
    -@erase "$(INTDIR)¥vc60.pdb"
    -@erase "$(OUTDIR)¥upsw00.dll"
    -@erase "$(OUTDIR)¥upsw00.exp"
    -@erase "$(OUTDIR)¥upsw00.ilc"
    -@erase "$(OUTDIR)¥upsw00.lib"
    -@erase "$(OUTDIR)¥upsw00.pdb"

"$ (OUTDIR) " :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
CPP_PROJ=/nologo /Gz /Zp1 /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS"
/D "_MBCS" /D "_USRDLL" /D "UPSW00_EXPORTS" /Fp "$(INTDIR)¥upsw00.pch" /YX /Fo "$(INTDIR)¥¥"
/Fd "$(INTDIR)¥¥" /FD /c
MTL_PROJ=/nologo /D "_DEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o "$(OUTDIR)¥upsw00.bsc"
BSC32_SBRS= ¥

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll
/incremental:yes /pdb:"$(OUTDIR)¥upsw00.pdb" /debug /machine:I386 /def:".¥Upsw00.def"
/out:"$(OUTDIR)¥upsw00.dll" /implib:"$(OUTDIR)¥upsw00.lib" /pdbtype:sept
DEF_FILE= ¥
    ".¥Upsw00.def"
LINK32_OBJS= ¥
    "$(INTDIR)¥Upsw00.obj" ¥
    ".¥sik032.lib"
"$ (OUTDIR)¥upsw00.dll" : "$(OUTDIR) " $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
    $(LINK32_FLAGS) $(LINK32_OBJS)
<<
!ENDIF
.c{$(INTDIR)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<

```

```
.cpp{$(INTDIR)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cxx{$(INTDIR)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.c{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cpp{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cxx{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
!IF "$(NO_EXTERNAL_DEPS)" != "1"
!IF EXISTS("upsw00.dep")
!INCLUDE "upsw00.dep"
!ELSE
!MESSAGE Warning: cannot find "upsw00.dep"
!ENDIF
!ENDIF
!IF "$(CFG)" == "upsw00 - Win32 Release" || "$(CFG)" == "upsw00 - Win32 Debug"
SOURCE=.\Upw00.c
"$(INTDIR)\Upw00.obj" : $(SOURCE) "$(INTDIR)"
!ENDIF
```

6.2 ユーザ・ウィンドウによるカスタマイズ部品例

6.2.1 サンプル内容

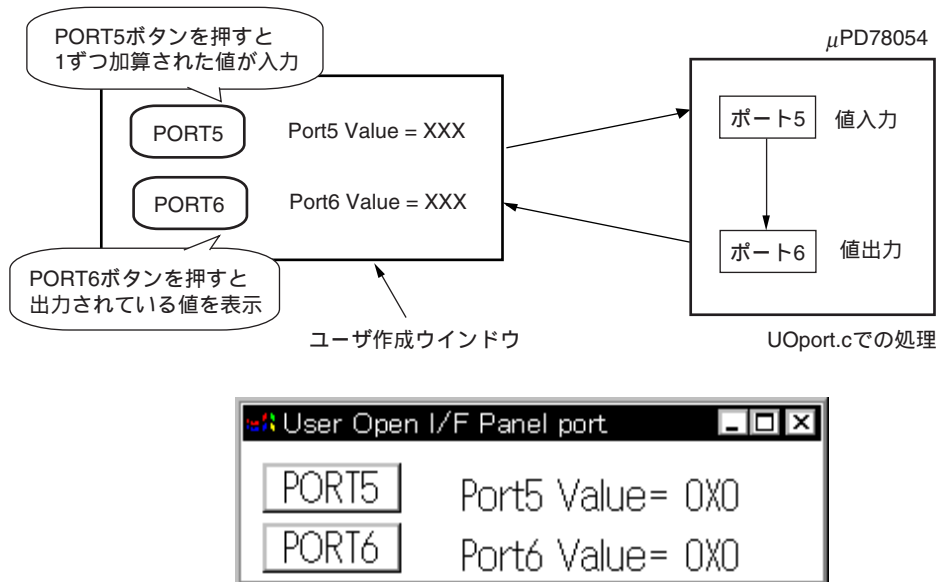
ユーザ・カスタム・ウィンドウに、ポート5に値を設定する部品とポート6からの値を取得する部品を表示します。

ユーザ・カスタム・ウィンドウのPORT5ボタンを押すと、ポート5への入力値が1ずつ加算されていきます。「Port5 Value = 0x7f」となると、0に戻ります。

このポート5からの入力値をポート6へ出力させます。ユーザ・カスタム・ウィンドウのPORT6ボタンを押したタイミングで、「Port6 Value = XXXX」の表示が現在の値に更新されます。

この例では、次のようになっています。

図6 - 2 ユーザ・ウィンドウによるカスタマイズ部品例



6.2.2 ソース例

ターゲット・プログラム

(1/1) Sample2.c

```
#pragma sfr

void main()
{
    MM = 0xB0;
    PM5 = 0xFF;
    PM6 = 0;
    P6 = 0;

    while(1)
    {
        P6 = P5;
    }
}
```

カスタム品ソース・ファイル UOport.c

(1/7) UOport.c

```
/*
 *      User Open I/F Sample Program    (UOport.c)
 */

#include      <stdio.h>
#include      <stdlib.h>
#include      <string.h>

typedef unsigned char   UCHAR;
typedef unsigned short  USHORT;
typedef unsigned long   ULONG;

#include      <Windows.h>
#include      "uparts32.h"

#define IDM_PAST          0x1111
#define IDM_NEWWIN       0x1112
#define BTN_WIDTH        70
#define BTN_HIGHT       25
#define IDD_PIN_BUTTON   0x10

void FAR PASCAL UParts_port(void);
LONG FAR PASCAL UParts_portWndProc(HWND, unsigned, WPARAM, LPARAM);
void FAR PASCAL UParts_portCall(ULONG);
void FAR PASCAL UParts_portReset(void);
void FAR PASCAL UParts_portLoadProj(char FAR *);
void FAR PASCAL UParts_portSaveProj(char FAR *);

/* Window point */
#define UParts_portWIDTH      300
#define UParts_portHEIGHT    100

/* Title Strings */
#define STR_UP_TITLE          "User Open I/F Panel port"

/* Window Class Name */
const BYTE   cnUParts_port[] = "UParts_portWin";
```

```
HANDLE hInst;
HWND hUParts_portWnd;
HWND btm_hwnd[2];
char *strbuf[2] = {"PORT5","PORT6"};
char port5val = 0;
char port6val = 0;
char UParts_Veiw_str[7];
char UParts_Rect_str[23];

/*****
/*      DLL Main
*****/

BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    WNDCLASS wndclass;

    switch(ul_reason_for_call){
        case DLL_PROCESS_ATTACH:
            hInst = hModule;
            wndclass.lpszClassName = (LPSTR)cnUParts_port;
            wndclass.hInstance = hInst;
            wndclass.lpfnWndProc = (WNDPROC)UParts_portWndProc;
            wndclass.hCursor = NULL;
            wndclass.hIcon = NULL;
            wndclass.lpszMenuName = NULL;
            wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
            wndclass.style = CS_HREDRAW | CS_VREDRAW;
            wndclass.cbClsExtra = 0;
            wndclass.cbWndExtra = DLGWINDOWEXTRA;

            RegisterClass(&wndclass);

            break;
        case DLL_THREAD_ATTACH:
            break;
        case DLL_THREAD_DETACH:
            break;
        case DLL_PROCESS_DETACH:
            break;
    }
}
```

```
    }

    return(TRUE);
}

/*****
/*      UParts_port(void)
/*****

void FAR PASCAL UParts_port(void)
{
    if(!hUParts_portWnd) {
        hUParts_portWnd = CreateWindow((LPSTR)cnUParts_port, /* Class name */
            STR_UP_TITLE, /* Title. */
            WS_OVERLAPPEDWINDOW | WS_BORDER | WS_VISIBLE, /* Style bits. */
            CW_USEDEFAULT, /* x - default. */
            CW_USEDEFAULT, /* y - default. */
            UParts_portWIDTH, /* cx - default.*/
            UParts_portHEIGHT, /* cy - default.*/
            NULL, /* No parent. */
            NULL, /* Class memu. */
            hInst, /* Creator */
            NULL); /* Params. */
    }

    if(hUParts_portWnd) {
        UpSetUserWnd(hUParts_portWnd);
        ShowWindow(hUParts_portWnd, SW_SHOW);
        UpCallFuncName("UParts_portCall");
        UpResetFuncName("UParts_portReset");
        UpLoadProjName("UParts_portLoadProj");
        UpSaveProjName("UParts_portSaveProj");
        UpInitPort("P5", 0xFF);
    }

    return;
}
```

```
/*
 * UParts_portWndProc(HWND, unsigned WPARAM, LPARAM)
 */
LONG PASCAL FAR UParts_portWndProc(HWND hWnd, unsigned iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC          hDC;
    PAINTSTRUCT  ps;
    RECT wRect;
    int          i;
    char strval[20];
    long wx,wy;
    switch(iMessage){
    case WM_CREATE:
        for(i = 0; i < 2; i++){
            btm_hwnd[i] = CreateWindow((LPSTR)"button",strbuf[i],
                WS_CHILD|BS_PUSHBUTTON|WS_VISIBLE|WS_TABSTOP,
                10,10+30*i,
                BTN_WIDTH,BTN_HIGHT,
                hWnd, (HMENU) (IDD_PIN_BUTTON+i),hInst,NULL);
        }
        return(FALSE);
    case WM_COMMAND:
        switch(wParam){
        case IDD_PIN_BUTTON:
            if(port5val < 0x7f)
                port5val++;
            else
                port5val = 0;
            UpSetPort("P5", port5val, 0);
            InvalidateRect(hWnd, NULL, TRUE);
            UpdateWindow(hWnd);
            break;
        case IDD_PIN_BUTTON + 1:
            UpGetPort("P6", &port6val);
            InvalidateRect(hWnd, NULL, TRUE);
            UpdateWindow(hWnd);
            break;
        }
    }
}
```

```
return(FALSE);

case WM_PAINT:
    hDC = BeginPaint(hWnd, &ps);
    sprintf(strval, "Port5 Value=%#2X¥0", port5val);
    TextOut(hDC, BTN_WIDTH + 40, 15, strval, strlen(strval));
    sprintf(strval, "Port6 Value=%#2X¥0", port6val);
    TextOut(hDC, BTN_WIDTH + 40, 45, strval, strlen(strval));
    EndPaint(hWnd, &ps);
    return(FALSE);

case WM_SYSCOLORCHANGE:
    InvalidateRect(hWnd, NULL, TRUE);
    break;

case WM_MOVE:
    GetWindowRect(hWnd, &wRect);
    wx = wRect.right - wRect.left;
    wy = wRect.bottom - wRect.top;
    if((wx != 36) && (wy != 36)) {
        sprintf(UParts_Rect_str, "%d, %d, %d, %d",
                wRect.left, wRect.top, wx, wy);
    }
    InvalidateRect(hWnd, NULL, TRUE);
    break;

case WM_SIZE:
    if(wParam == SIZE_ICONIC) {
        strcpy(UParts_Veiw_str, "Icon");
    } else {
        GetWindowRect(hWnd, &wRect);
        strcpy(UParts_Veiw_str, "Normal");
        sprintf(UParts_Rect_str, "%d, %d, %d, %d",
                wRect.left,
                wRect.top,
                wRect.right - wRect.left,
                wRect.bottom - wRect.top);
    }
    break;

case WM_DESTROY:
    UpCloseUserWnd(hWnd);

default:
    return DefWindowProc(hWnd, iMessage, wParam, lParam);
}
```

```
    return 0L;
}
/*****
/*      UParts_portCall (ULONG)                                */
*****/
void FAR PASCAL
UParts_portCall (ULONG time)
{
    return;
}
/*****
/*      UParts_portReset (void)                                */
*****/
void FAR PASCAL
UParts_portReset (void)
{
    port5val = 0;
    port6val = 0;
    InvalidateRect (hUParts_portWnd, NULL, TRUE);
}
/*****
/*      UParts_portLoadProj (char FAR *)                       */
*****/
void FAR PASCAL
UParts_portLoadProj (char FAR *fname)
{
    char *next;
    WORD x, y, wx, wy;
    GetPrivateProfileString ("UOport", "Window",
                             "Hide", UParts_Veiw_str, 7, fname);
    if (!lstrcmp (UParts_Veiw_str, "Icon")) { /* "Icon" mode */
        ShowWindow (hUParts_portWnd, SW_SHOWMINNOACTIVE);
    } else { /* "Normal" mode */
        GetPrivateProfileString ("UOport", "Geometry",
                                 "0, 0, 0, 0", UParts_Rect_str, 23, fname);
        if (lstrcmp (UParts_Rect_str, "0, 0, 0, 0")) {
            next = strtok (UParts_Rect_str, ",");
            x = (WORD) strtoul (next, NULL, 10);
            next = strtok (NULL, ",");
            y = (WORD) strtoul (next, NULL, 10);
            next = strtok (NULL, ",");

```

```
        wx = (WORD) strtoul(next, NULL, 10);
        next = strtok(NULL, "");
        wy = (WORD) strtoul(next, NULL, 10);
        MoveWindow(hUParts_portWnd, x, y, wx, wy, TRUE);
    }
    ShowWindow(hUParts_portWnd, SW_SHOWNORMAL);
}
}
/*****
/*      UParts_portSaveProj(char FAR *)          */
/*****
void FAR PASCAL
UParts_portSaveProj(char FAR *fname)
{
    WritePrivateProfileString("UOport", "Window",  UParts_Veiw_str, fname);
    WritePrivateProfileString("UOport", "Geometry", UParts_Rect_str, fname);
}
/* UOport.c */
```


定義ファイル UOport.def

(1/1) UOport.def

```
LIBRARY      UOPORT

DESCRIPTION  'User Open I/F Panel port'

HEAPSIZE    4096

EXPORTS

    UParts_port
    UParts_portWndProc
    UParts_portCall
    UParts_portReset
    UParts_portLoadProj
    UParts_portSaveProj
```

メイク・ファイル UOport.mak

(1/4) UOport.mak

```
# Microsoft Developer Studio Generated NMAKE File, Based on UOport.dsp
!IF "$(CFG)" == ""
CFG=UOport - Win32 Release
!MESSAGE 構成が指定されていません。デフォルトの UOport - Win32 Debug を設定します。
!ENDIF
!IF "$(CFG)" != "UOport - Win32 Release" && "$(CFG)" != "UOport - Win32 Debug"
!MESSAGE 指定された ビルド・モード "$(CFG)" は正しくありません。
!MESSAGE NMAKE の実行時に構成を指定できます
!MESSAGE コマンド・ライン上でマクロの設定を定義します。例：
!MESSAGE
!MESSAGE NMAKE /f "UOport.mak" CFG="UOport - Win32 Debug"
!MESSAGE
!MESSAGE 選択可能なビルド・モード：
!MESSAGE
!MESSAGE "UOport - Win32 Release" ("Win32 (x86) Dynamic-Link Library" 用)
!MESSAGE "UOport - Win32 Debug" ("Win32 (x86) Dynamic-Link Library" 用)
!MESSAGE
!ERROR 無効な構成が指定されています。
!ENDIF
!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe
!IF "$(CFG)" == "UOport - Win32 Release"
OUTDIR=.\Release
INTDIR=.\Release
# Begin Custom Macros
OutDir=.\Release
# End Custom Macros
ALL : "$(OUTDIR)UOport.dll"
```

```

CLEAN :

    -@erase "$(INTDIR)¥Uoport.obj"
    -@erase "$(INTDIR)¥vc60.idb"
    -@erase "$(OUTDIR)¥Uoport.dll"
    -@erase "$(OUTDIR)¥Uoport.exp"
    -@erase "$(OUTDIR)¥Uoport.lib"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
CPP_PROJ=/nologo /Zp1 /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_MBCS" /D
"_USRDL" /D "UOPORT_EXPORTS" /Fp"$(INTDIR)¥Uoport.pch" /YX /Fo"$(INTDIR)¥¥"
/Fd"$(INTDIR)¥¥" /FD /c
MTL_PROJ=/nologo /D "NDEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)¥Uoport.bsc"
BSC32_SBRS= ¥

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll
/incremental:no /pdb:"$(OUTDIR)¥Uoport.pdb" /machine:I386 /def:".¥Uoport.def"
/out:"$(OUTDIR)¥Uoport.dll" /implib:"$(OUTDIR)¥Uoport.lib"
DEF_FILE= ¥
    ".¥Uoport.def"
LINK32_OBJS= ¥
    "$(INTDIR)¥Uoport.obj" ¥
    ".¥sik032.lib"
"$(OUTDIR)¥Uoport.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
    $(LINK32_FLAGS) $(LINK32_OBJS)
<<
!ELSEIF "$(CFG)" == "UOport - Win32 Debug"
OUTDIR=¥Debug
INTDIR=¥Debug
# Begin Custom Macros
OutDir=¥Debug
# End Custom Macros
ALL : "$(OUTDIR)¥Uoport.dll"

```

```

CLEAN :

    -@erase "$(INTDIR)¥Uoport.obj"
    -@erase "$(INTDIR)¥vc60.idb"
    -@erase "$(INTDIR)¥vc60.pdb"
    -@erase "$(OUTDIR)¥Uoport.dll"
    -@erase "$(OUTDIR)¥Uoport.exp"
    -@erase "$(OUTDIR)¥Uoport.ilc"
    -@erase "$(OUTDIR)¥Uoport.lib"
    -@erase "$(OUTDIR)¥Uoport.pdb"

"$ (OUTDIR) " :
    if not exist "$ (OUTDIR) / $ (NULL) " mkdir "$ (OUTDIR) "
CPP_PROJ=/nologo /Zp1 /MTd /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /D
"_MBCS" /D "_USRDLL" /D "UOPORT_EXPORTS" /Fp"$ (INTDIR)¥Uoport.pch" /YX /Fo"$ (INTDIR)¥¥"
/Fd"$ (INTDIR)¥¥" /FD /GZ /c
MTL_PROJ=/nologo /D "_DEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$ (OUTDIR)¥Uoport.bsc"
BSC32_SBR5= ¥

LINK32=link.exe
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /dll
/incremental:yes /pdb:"$ (OUTDIR)¥Uoport.pdb" /debug /machine:I386 /def:".¥Uoport.def"
/out:"$ (OUTDIR)¥Uoport.dll" /implib:"$ (OUTDIR)¥Uoport.lib" /pdbtype:sept
DEF_FILE= ¥
    ".¥Uoport.def"
LINK32_OBJS= ¥
    "$ (INTDIR)¥Uoport.obj" ¥
    ".¥sik032.lib"
"$ (OUTDIR)¥Uoport.dll" : "$ (OUTDIR) " $ (DEF_FILE) $ (LINK32_OBJS)
    $ (LINK32) @<<
    $ (LINK32_FLAGS) $ (LINK32_OBJS)
<<
!ENDIF
.c{$ (INTDIR)}.obj::
    $ (CPP) @<<
    $ (CPP_PROJ) $<
<<

```

```
.cpp{$(INTDIR)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cxx{$(INTDIR)}.obj::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.c{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cpp{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
.cxx{$(INTDIR)}.sbr::
    $(CPP) @<<
    $(CPP_PROJ) $<
<<
!IF "$(NO_EXTERNAL_DEPS)" != "1"
!IF EXISTS("UOport.dep")
!INCLUDE "UOport.dep"
!ELSE
!MESSAGE Warning: cannot find "UOport.dep"
!ENDIF
!ENDIF
!IF "$(CFG)" == "UOport - Win32 Release" || "$(CFG)" == "UOport - Win32 Debug"
SOURCE=.\UOport.c
"$(INTDIR)\UOport.obj" : $(SOURCE) "$(INTDIR)"
!ENDIF
```

付録A エラー・メッセージ

A.1 エラー処理

- (a) シミュレーションする品種に存在しない端子名を指定した場合は、エラー・メッセージ・ダイアログにより、エラーを通知します。
- (b) ユーザ・パネル・カスタム関数と<入出力パネル>カスタム関数を混ぜた1つのDLLファイルを読み込んだ場合、DLLファイル名に従って、そのDLLファイルに記述すべきでない関数を最初に取り込んだときに、ワーニング・メッセージをダイアログにより通知します。
- (c) ユーザ作成カスタムDLLを読み込んだときにエラーが発生した場合、エラー発生部品は作成しません。
- (d) ユーザ・パネル・カスタム関数のうち、UpGetPin(), UpGetPort(), UpGetMem(), UpClrMtrAcClk(), UpGetStpingMtr()は、エラーまたはワーニングが一度発生すると、エラーが発生した以降にこれらの関数が現れても、エラー値を返したり関数の機能を果たさない状態になります。このため、エラーやワーニングが発生した場合、ソースを修正して再度DLLファイルを作成し、再ロードし直してください。

A.2 エラー/ワーニング・メッセージ

関数の実行時に発生する可能性のあるエラー・メッセージおよびワーニング・メッセージを次に示します。関数名の項には、エラーの発生する関数の名前を下記の短縮した名前で記述してあります。

ステッピング・モータ関数	UpStpingMtr(), UpSetStpingMtr(), UpGetStpingMtr()
LED図形設定関数	UpSetLedPic()
LED関数	UpLed(), UpPortLed()
マトリックスLED関数	UpMtxLed()
シリアル端子データ入力関数	UpSerial_data()
ポート値設定/取得関数	UpPortLed(), UpGetPort(), UpSetPort()
保有時間設定関数	UpSetPBtmtime()
縦型スクロール・バー式アナログ入力関数	UpScaleInterAD()
基準電圧値設定関数	UpSetAVref()
関数名通知関数	UpCallFuncName(), UploadProjName(), UpSaveProjName(), UpResetFuncName()
ビット・マップ設定関数	UpSetBtmBmp(), UpSetLedBmp(), UpSetMtrBmp()
ボタン関数	UpPushBtm(), UpTglBmp(), UpSelectBtm()
レジスタ関係関数	UpGetReg(), UpSetReg()

表A-1 エラー・メッセージ一覧(1/2)

番号	ウインドウ名	メッセージ	原因	ユーザ処置
E1	パラメータに端子名の記述があるすべての関数	指定した端子は存在しません。	記述した端子が存在しません。	対象デバイスに存在する端子名を記述してください。
E2	パラメータに端子名の記述があるすべての関数	端子名が全角で記述されています。	記述した端子名が全角で記述されています。	半角文字で端子名を記述してください。
E3	パラメータにアクティブH/Lの記述があるすべての関数	アクティブH/LがHIGHまたはLOWではありません。	記述したアクティブH/LがHIGHおよびLOWではない値です。	アクティブH/LにHIGHまたはLOWを記述してください。
E4	パラメータにチャンネル数の記述のあるステップング・モータ関数	チャンネル数が4または8ではありません。	記述したチャンネル数が4および8でない値です。	チャンネル数に4または8(端子数の数に合わせて)を記述してください。
E5	パラメータに励磁の記述のあるステップング・モータ関数	励磁が0または1ではありません。	記述した励磁の値が0および1でない値です。	励磁に0または1(励磁方式に従った)を記述してください。
E6	パラメータに最小ステップ角の記述があるステップング・モータ関数	最小ステップ角が360を割り切れません。	記述した最小ステップ角が360を割り切れない値です。	最小ステップ角に360を割り切れる整数を記述してください。
E7	LED図形設定関数	図形のタイプがPIC_RECTまたはPIC_ELLではありません。	パラメータの図形タイプに記述した値がPIC_RECTおよびPIC_ELLでない値です。	図形のタイプにPIC_RECTまたはPIC_ELLを記述してください。
E8	LED図形設定関数	色のタイプがPIC_REDまたはPIC_YELLOWまたはPIC_GREENではありません。	パラメータの色のタイプに記述した値がPIC_REDおよびPIC_YELLOWおよびPIC_GREENでない値です。	色のタイプにPIC_REDまたはPIC_YELLOWまたはPIC_GREENを記述してください。
E9	パラメータに図の形式の記述のあるLED関数	表示の図形指定が0または1ではありません。	記述した図の形式の値が0および1でない値です。	パラメータの図の形式に値0または1を記述してください。
E10	シリアル端子データ入力関数	シリアル入力の先頭ビット指定がMSB(1)またはLSB(0)ではありません。	パラメータの先頭ビットに記述した値が0および1でない値です。	パラメータの先頭ビットに値1または0を記述してください。
E11	マトリクスLED関数	出力1のアクティブH/LがHIGHまたはLOWではありません。	パラメータの出力1のアクティブH/Lに記述した値がHIGHおよびLOWでない値です。	パラメータの出力1のアクティブH/LにHIGHまたはLOWを記述してください。
E12	マトリクスLED関数	出力2のアクティブH/LがHIGHまたはLOWではありません。	パラメータの出力2のアクティブH/Lに記述した値がHIGHおよびLOWでない値です。	パラメータの出力2のアクティブH/LにHIGHまたはLOWを記述してください。
E13	ポート値設定/取得関数	ポート名が全角で記述されています。	パラメータのポート名が全角で記述されています。	パラメータのポート名を半角で記述してください。
E14	ポート値設定/取得関数	指定したポートは存在しません。	パラメータのポート名に存在しないポート名が記述されています。	対象デバイスに存在するポート名を記述してください。

表A-1 エラー・メッセージ一覧(2/2)

番号	ウインドウ名	メッセージ	原因	ユーザ処置
E15	保有時間設定関数	保有時間が不正です。	保有時間が範囲内に設定されていない、または数ではありません。	保有時間を0.001 msecから999 msecの範囲で設定してください。
E16	縦型スクロール・バー式アナログ入力関数	設定した端子がアナログ入力端子ありません。	アナログ用の端子でない端子名が記述されています。	アナログ用の端子を記述してください。
E17	基準電圧値設定関数	AV _{REF} が動作電源電圧の範囲内にありません。	AV _{REF} が動作電源電圧の範囲内にありません。	動作電源電圧内の範囲で設定してください。
E18	ポインタ型のパラメータを持つすべての関数	パラメータがNULLポインタです。	関数に不正なパラメータを記述しています。	正しいポインタを記述してください。
E19	4. 1に記述したすべての関数	関数XXXXは、UOxxx.dllに記述できません。	UOxxx.dllにこの関数を記述しています。	UPxxx.dllで使用してください。
E20	4. 2に記述したすべての関数	関数XXXXは、UPxxx.dllに記述できません。	UPxxx.dllにこの関数を記述しています。	UOxxx.dllで使用してください。
E21	事前に通知関数が必要な関数	関数XXXXを事前に通知していません。	必要な通知関数を事前に通知していません。	事前に必要な通知関数をコールしてください。
E22	レジスタ関連関数	制御レジスタのタイプがREG_PC、REG_PSWまたはREG_SPではありません。	制御レジスタ以外のレジスタを指定しています。	制御レジスタ以外は指定しないでください。
E23	ビット・マップ設定関数	指定したビット・マップ・ファイルが不正です。	指定したファイルへのアクセス、または領域確保に失敗しています。	正しいファイルを指定してください。または他のアプリケーションを削除してメモリを確保してください。
E24	ボタン関数	指定した端子は既に指定されています。	すでに指定済みの端子を再度設定しようとしています。	すでに設定済みの端子を再度設定しないでください。
E25	全関数	領域が確保できません。	メモリの確保ができませんでした。	他のアプリケーションを削除してメモリを確保してください。

表A-2 ワーニング・メッセージ一覧

番号	ウインドウ名	メッセージ	原因	ユーザ処置
W1	基準電圧値設定関数	AV _{REF} の設定がありません。5.0 Vになりますがよろしいですか？	基準電圧値文字列に” ”を記述しています。	基準電圧値文字列に設定したい値の文字列を記述してください。
W2	保有時間設定関数	保有時間の設定がありません。0.5 msecになりますがよろしいですか？	保有時間文字列に” ”を記述しています。	保有時間文字列に設定したい値の文字列を記述してください。

〔メモ〕

— お問い合わせ先 —

【技術的なお問い合わせ先】

NEC半導体テクニカルホットライン
(電話：午前 9:00～12:00，午後 1:00～5:00)

電話 : 044-435-9494
FAX : 044-435-9608
E-mail : info@lsi.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部

東京 (03)3798-6106, 6107,
6108

大阪 (06)6945-3178, 3200,
3208, 3212

広島 (082)242-5504

仙台 (022)267-8740

郡山 (024)923-5591

千葉 (043)238-8116

第二販売事業部

東京 (03)3798-6110, 6111,
6112

立川 (042)526-5981, 6167

松本 (0263)35-1662

静岡 (054)254-4794

金沢 (076)232-7303

松山 (089)945-4149

第三販売事業部

東京 (03)3798-6151, 6155, 6586,
1622, 1623, 6156

水戸 (029)226-1702

前橋 (027)243-6060

鳥取 (0857)27-5313

太田 (0276)46-4014

名古屋 (052)222-2170, 2190

福岡 (092)261-2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロニクス デバイス ホームページ】

NECエレクトロニクスデバイ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.ic.nec.co.jp/>

アンケート記入のお願い

お手数ですが、このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

SM78K シリーズ Ver.2.30 以上 ユーザーズ・マニュアル

[ドキュメント名] 外部部品ユーザ・オープン・インタフェース仕様編

(U15802JJ1V0UM00 (第1版))

[お名前など] (さしつかえのない範囲で)

御社名(学校名, その他) ()
ご住所 ()
お電話番号 ()
お仕事の内容 ()
お名前 ()

1. ご評価 (各欄に をご記入ください)

項 目	大変良い	良 い	普 通	悪 い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン, 字の大きさなど					
その他 ()					
()					

2. わかりやすい所 (第 章, 第 章, 第 章, 第 章, その他)
理由 []

3. わかりにくい所 (第 章, 第 章, 第 章, 第 章, その他)
理由 []

4. ご意見, ご要望

5. このドキュメントをお届けしたのは
NEC 販売員, 特約店販売員, その他 ()

ご協力ありがとうございました。

下記あてに FAX で送信いただくか, 最寄りの販売員にコピーをお渡しください。

日本電気(株) NEC エレクトロニクス
半導体テクニカルホットライン

FAX : (044) 435-9608

2000.6