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

ユーザーズ・マニュアル

SM850

システム・シミュレータ　Ver.2.00 以上

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

V850 シリーズ ™

資料番号　U14873JJ2V0UM00（第2版）
発行年月　July 2002 CP(K)

〔メ　モ〕

目次要約

V850 **シリーズ，**V852，V853，V850/SA1，V850/SB1，V850/SB2，V850/SC1，V850/SC2，V850/SC3，V850/SF1, V850/SV1, V850E/MS1, V850E/MS2, V850E/MA1, V850E/MA2, V850E/IA1, V850E/IA2 **は日本電気株式会社の商標です。**

Pentium **は** Intel Corp.**の商標です。**

Windows，WindowsNT **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，ソフトウエア，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・ソフトウエア・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，Windows®98/WindowsMe/WindowsNT™4.0/Windows2000/WindowsXP 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，Windows98/WindowsMe/WindowsNT4.0/Windows2000/WindowsXP 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 SM850 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 Windows98/WindowsMe/WindowsNT4.0/Windows2000/WindowsXP 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。

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。

○開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号	
		和文	英文
IE-703002-MC（V853™, V850/SA1™, V850/SB1™, V850/SB2™, V850/SC1™, V850/SC2™, V850/SC3™, V850/SF1™, V850/SV1™用インサーキット・エミュレータ）		U11595J	U11595E
IE-V850E-MC（V850E/IA1™, V850E/IA2™用インサーキット･エミュレータ），IE-V850E-MC-A（V850E/MA1™, V850E/MA2™用インサーキット・エミュレータ）		U14487J	U14487E
IE-703003-MC-EM1（V853用インサーキット・エミュレータ・オプション・ボード）		U11596J	U11596E
IE-703017-MC-EM1（V850/SA1用インサーキット・エミュレータ・オプション・ボード）		U12898J	U12898E
IE-703037-MC-EM1（V850/SB1, V850/SB2用インサーキット・エミュレータ・オプション・ボード）		U14151J	U14151E
IE-703040-MC-EM1（V850/SV1用インサーキット・エミュレータ・オプション・ボード）		U14337J	U14337E
IE-703079-MC-EM1（V850/SF1用インサーキット・エミュレータ・オプション・ボード）		U15447J	U15447E
IE-703102-MC（V850E/MS1™, V850E/MS2™用インサーキット・エミュレータ）		U13875J	U13875E
IE-703102-MC-EM1，IE-703102-MC-EM1-A（V850E/MS1，V850E/MS2用インサーキット・エミュレータ・オプション・ボード）		U13876J	U13876E
IE-703107-MC-EM1（V850E/MA1用インサーキット・エミュレータ・オプション・ボード）		U14481J	U14481E
IE-703116-MC-EM1（V850E/IA1用インサーキット・エミュレータ・オプション・ボード）		U14700J	U14700E
CA850 Ver.2.50 Cコンパイラ・パッケージ	操作編	U16053J	U16053E
	C言語編	U16054J	U16054E
	PM plus編	U16055J	U16055E
	アセンブリ言語編	U16042J	U16042E
ID850 Ver.2.50　統合ディバッガ	操作編	U16217J	作成予定
SM850 Ver.2.50　システム・シミュレータ	操作編	U16218J	作成予定
SM850 Ver.2.00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	このマニュアル	作成予定
RX850 Ver.3.13以上　リアルタイムOS	基礎編	U13430J	U13430E
	インストレーション編	U13410J	U13410E
	テクニカル編	U13431J	U13431E
RX850 Pro Ver.3.13　リアルタイムOS	基礎編	U13773J	U13773E
	インストレーション編	U13774J	U13774E
	テクニカル編	U13772J	U13772E
RX-NET　ネットワーク・ライブラリ（TCP/IP）		U15083J	－
RX-NET　ネットワーク・ライブラリ（PPP）		U15303J	－
RX-NET　ネットワーク・ライブラリ（DNS）		U15304J	－
RX-NET　ネットワーク・ライブラリ（DHCP）		U15382J	－
RX-NET　ネットワーク・ライブラリ（SMTP）		U15505J	－
RX-NET　ネットワーク・ライブラリ（POP）		U15539J	－
RX-NET Ver.1.00　ネットワーク・ライブラリ（telnet）		U16085J	－
RD850 Ver.3.01　タスク・ディバッガ		U13737J	U13737E
RD850 Pro Ver.3.01　タスク・ディバッガ		U13916J	U13916E
AZ850 Ver.3.10　システム・パフォーマンス・アナライザ		U14410J	U14410E
PG-FP3　フラッシュ・メモリ・プログラマ		U13502J	U13502E
PG-FP4　フラッシュ・メモリ・プログラマ		U15260J	U15260E

目次

図の目次

図番号　　タイトル，ページ

表の目次

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

SM850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。

SM850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，SM850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。

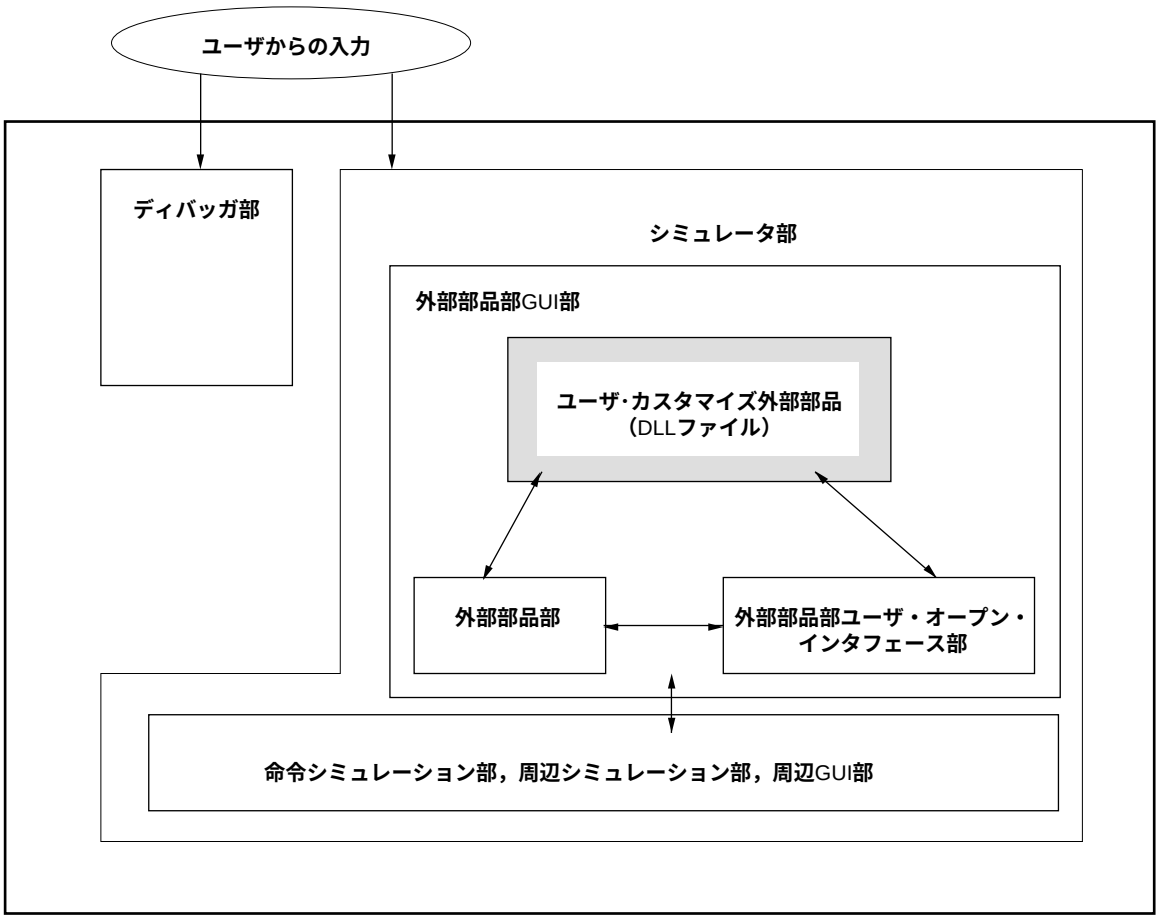
1.2.2 ユーザ作成ファイル

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 V850シミュレータの構成図



ディバッガ部	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ィ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	外部部品をウィ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。

ハードウエア環境：NEC PC-9821/PC98-NX**シリーズ**, IBM PC/AT™**互換機**
（CPU**：**Pentium™ 166 MHz**以上推奨）**
ソフトウエア環境： Windows 98 / Windows Me / WindowsNT 4.0 / Windows 2000 / Windows XP
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。

ハードウエア環境：NEC PC-9821/PC98-NX**シリーズ**, IBM PC/AT**互換機**
（CPU**：**Pentium 166 MHz**以上推奨）**
ソフトウエア環境： Windows 98 / Windows Me / WindowsNT 4.0 / Windows 2000 / Windows XP

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。

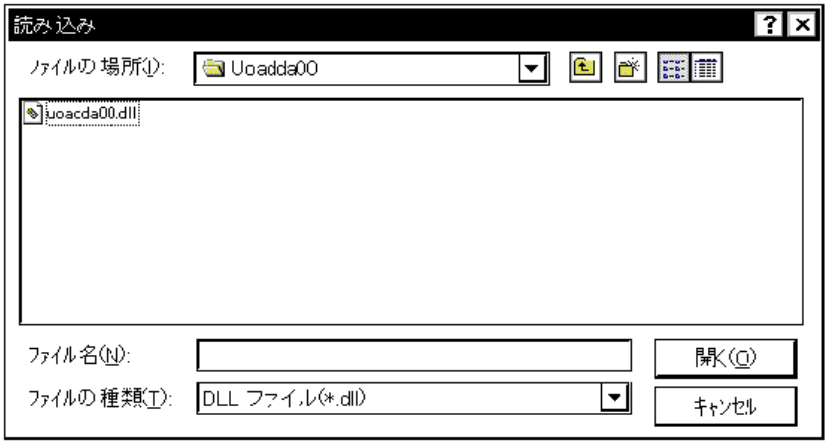
図2－1　SM850シミュレータ部＜入出力パネル＞ウィンドウ

2. 1　ダウンロード

ダウンロードの操作手順を次に示します。

(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＞ウィ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＞ウィンドウに張り付けられ，ユーザ・ウィンドウによるカスタマイズの部品であれば，ユーザ・ウィ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＞ウィンドウをクローズしても有効です。再度＜入出力パネル＞ウィ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＞ウィンドウの［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＞ウィ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＞ウィ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＞ウィンドウをクローズし，次回＜入出力パネル＞ウィンドウをオープンしたとき。

2.2　アンロード

アンロードの操作手順を次に示します。

(1) ＜入出力パネル＞ウィンドウの［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＞ウィ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＞ウィンドウから削除され，ユーザ・ウィ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・ウィ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。

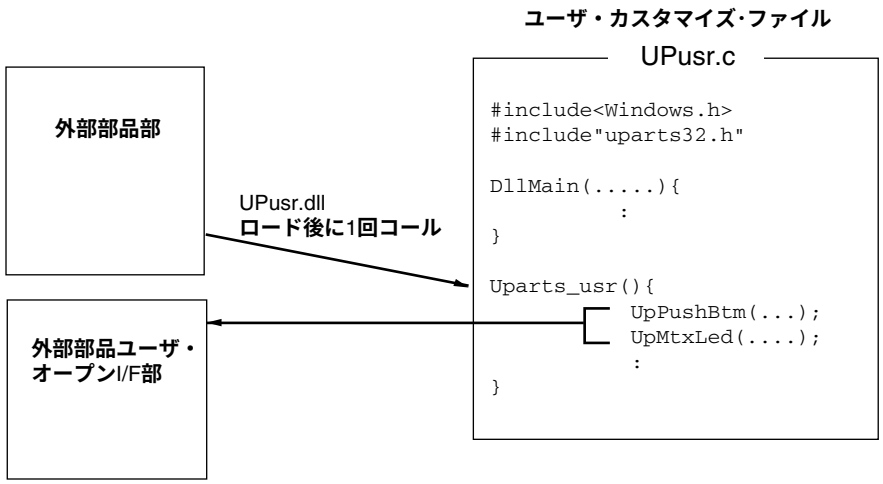
第4章で述べる関数レファレンスは，このユーザ関数内またはユーザ関数の下位の関数に記述しなければなりません。

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

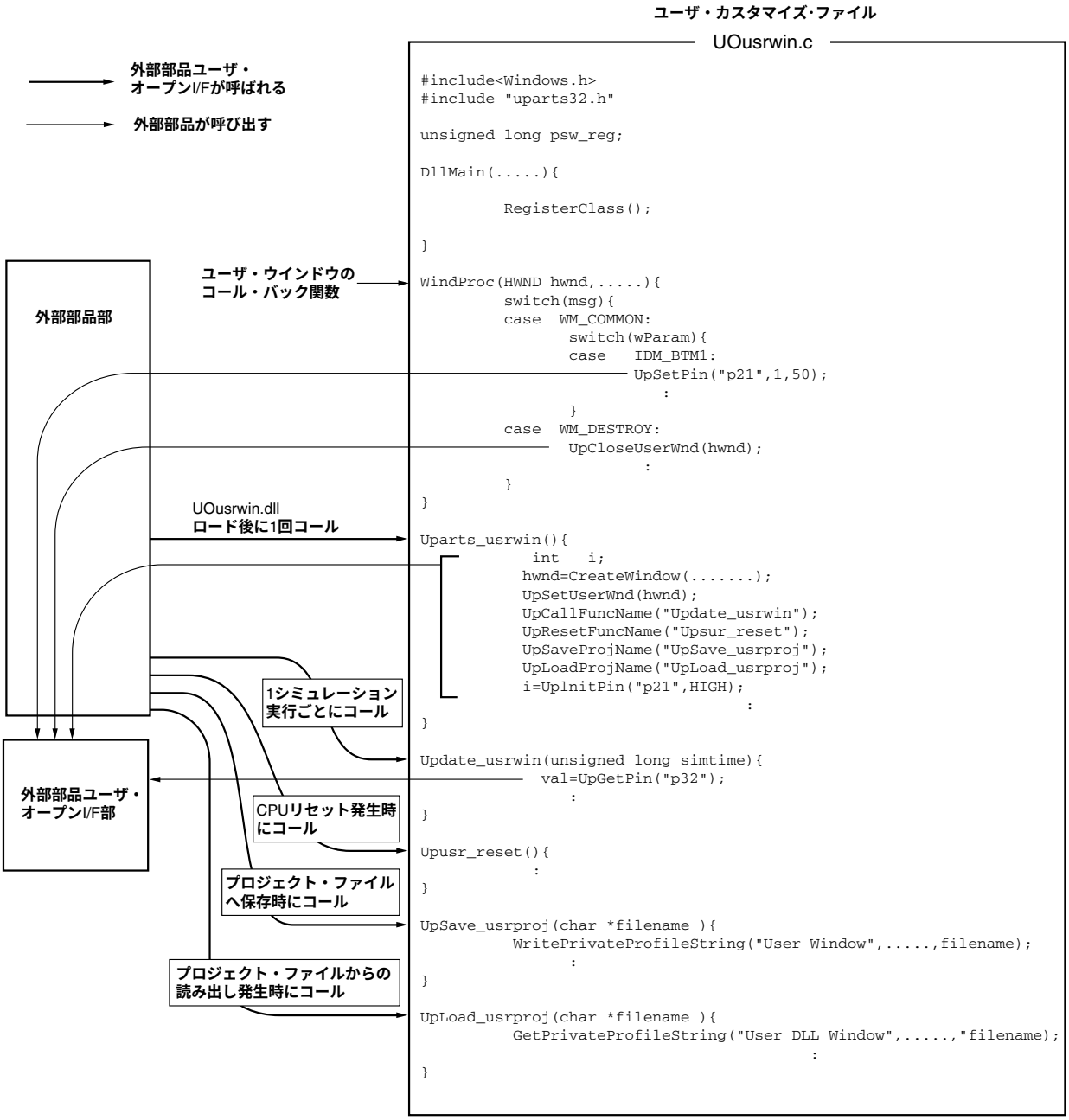
DLLファイル作成時に必要なDllMain関数と作成したウィンドウのコール・バック関数，ユーザ関数，シミュレーションの一定間隔でコールされるシミュレーション・コール関数から構成されます。
ユーザ関数またはユーザ関数の下位の関数で，シミュレーション・コール関数の通知やモータ端子名の通知などを行います。ユーザが作成したウィンドウのコール・バック関数や，シミュレーション・コール関数内で，部品の作成や入出力のアクション部分などのプログラミングを行います。

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。

ユーザ作成のウィ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方法

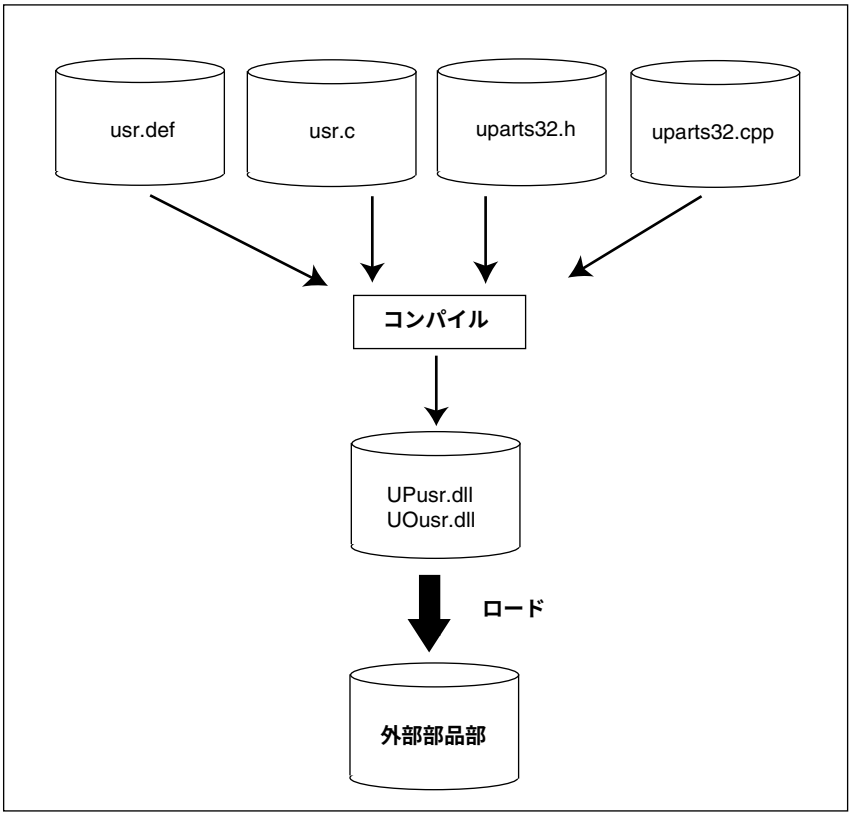
1. Windowsプログラミングに従ったDLL作成時の記述で，カスタマイズ用の外部部品をプログラミングします。その際には，“ uparts32.h ”をインクルードし，プロジェクトに“ uparts32.cpp ”を追加してください。
2. Windowsプログラミングに従ったモジュール定義（DEF）ファイル[注]，メイク・ファイル，必要であればリソース・ファイルを作成して，コンパイルしてユーザ作成DLLファイルを作成します。
   ・コンパイル時には，構造体メンバのアライメントを1バイトにするためのオプション（/Zp1）を指定してください。
   - 作成するDLLファイル名は，先頭2文字が“ 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，次回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方法

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[注]，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　カスタマイズ部品の作成フロー



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[注]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

SM850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のリンクを必ず行ってください。

- 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(*pname, actype, btmname*)	26
トグル・ボタン関数	UpTglBtm(*pname, actype, btmname*)	27
グループ式選択ボタン（排他的プッシュ・ボタン）関数	UpSelectBtm(*gname, pnames, pnum, actype, btmnames*)	28
保有時間設定関数	UpSetPBtmtime(*time*)	30
LED**関数**	UpLed(*pname, actype, ledname, pictype*)	31
ポート単位で設定できるLED**関数**	UpPortLed(*portname, actype, ledname, pictype*)	33
マトリクスLED**関数**	UpMtxLed(*pnames1, pnames2, pnum1, pnum2, actype1, actype2*)	35
DC**モータ関数**	UpDcMtr(*pname, actype, mtrname*)	37
ステッピング・モータ関数	UpStpingMtr(*pnames, num, actype, reiji, step*)	38
縦型スクロール・バー式アナログ入力関数	UpScaleInterAD(*pname, adname*)	40
基準電圧値設定関数	UpSetAVref(*avref*)	42
ボタン用ビットマップ設定関数	UpSetBtmBmp(*actbmp, nactbmp*)	43
LED**用ビットマップ設定関数**	UpSetLedBmp(*actbmp, nactbmp*)	44
DC**モータ用ビットマップ設定関数**	UpSetMtrBmp(*actbmp, nactbmp*)	45
LED**図形設定関数**	UpSetLedPic(*type, color*)	46
シリアル端子データ入力関数	UpSerial_data(*serpname, data, count, first*, *bitnum*)	47
ウィンドウのタイトル関数	UpPanelTitleName(*title*)	48
ビットマップ表示関数	UpSetUsrBmp(*bmpname*)	49
文字列表示関数	UpWriteString(*string*)	50

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，デフォールト値**0.5**ミリ秒が設定されます。**

【パラメータ】

pname	**端子名を文字列で指定します。**
actype	**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**）**を入力したい場合は**HIGH**を，**0（**ロウ**）**を入力したい場合は**LOW**を指定します。**
btmname	**プッシュ・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　　】

```
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*で指定した値です。

【パラメータ】

pname	**端子名を文字列で指定します。**
actype	**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**）**を入力したい場合は**HIGH**を，**0（**ロウ**）**を入力したい場合は**LOW**を指定します。**
btmname	**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。**

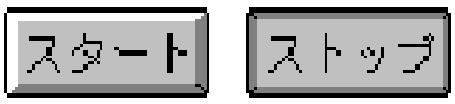
【戻　り　値】

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】

gname	グループに付ける名前を指定します。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。
pnames	端子名の文字列をボタンの数だけ指定します。
pnum	ボタンの数を指定します。
actype	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）を入力したい場合はHIGHを，0（ロウ）を入力したい場合は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。
btmnames	ボタン1つ1つに付ける名前を指定します。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。

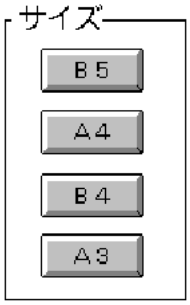
【戻　り　値】

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    UpSetPBtmtime(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	**保有時間文字列を指定します。指定の単位は，ミリ秒です。** **指定できる保有時間の範囲は，**0.001**～**999**ミリ秒となります。**

【戻　り　値】

なし

【　　例　　】

```
UpSetPBtmtime("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】

pname	端子名を文字列で指定します。
actype	LEDに表示させるときの値を指定します。アクティブ・ハイの場合は1を，アクティブ・ロウの場合は0を指定します。
ledname	LEDの機能名を指定します。この機能名はLEDの上に表示し，文字数制限はありません。
pictype	LED表示の図形を指定します。 1：　ディフォールトでは，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：　ディフォールトでは，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。

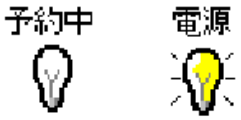
【戻　り　値】

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）。
おのおのの端子が，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指定します。

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）のときにアクティブな場合は1を, 0（ロウ）のときにアクティブな場合は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つ分指定します。

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

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

ledname　LEDに付ける名前を指定します。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。

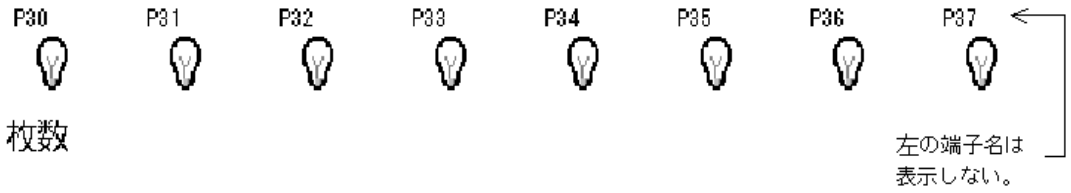
【戻　り　値】

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】

pnames1	**出力1の端子名文字列を出力1端子の数だけ指定します。**
pnames2	**出力2の端子名文字列を出力2端子の数だけ指定します。**
pnum1	**出力1の端子数を指定します。**
pnum2	**出力2の端子数を指定します。**
actype1	**出力1に表示させるときの値を指定します。アクティブ・ハイの場合は1を，アクティブ・ロウの場合は0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。**
actype2	**出力2に表示させるときの値を指定します。アクティブ・ハイの場合はHIGHを，アクティブ・ロウの場合は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。**

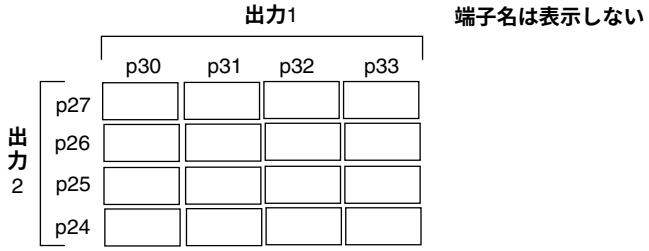
【戻　り　値】

なし

【　例　】

```
static   char   *out1[4] = {"p30","p31","p32","p33"};
static   char   *out2[4] = {"p24","p25","p26","p27"};
UpMtxLed((char *)out1, (char *)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，シミュレーションが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】

pname	端子名を文字列で指定します。
actype	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を，アクティブ・ロウの場合はLOWを指定します。
mtrname	DCモータの機能名を指定します。この機能名はモータの下に表示します。

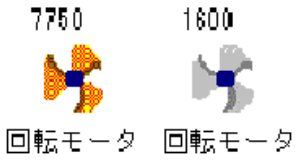
【戻　り　値】

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】

pname	端子名文字列を端子の数だけ指定します。
num	1チャネルの端子の数（4または8）を指定します。
actype	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を，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。
reiji	励磁方式を指定します。1相の場合は0を，1-2相の場合は1を指定します。
step	最小ステップ角は360を割り切れる整数を指定します。

【戻　り　値】

なし

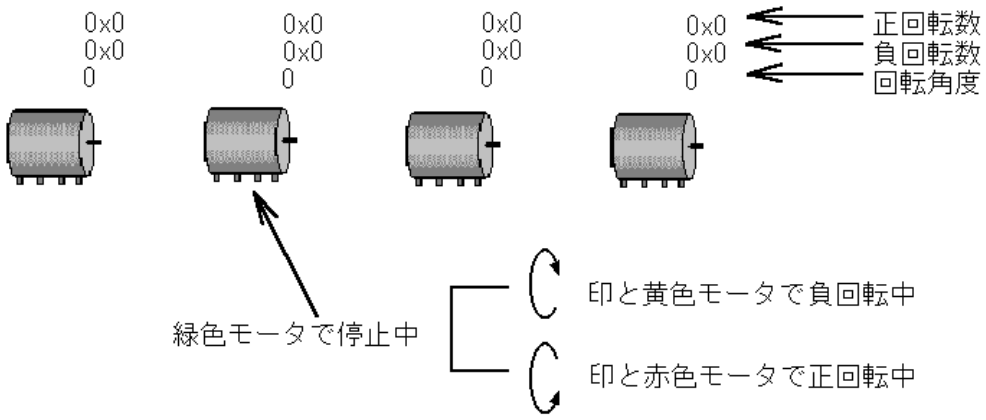
【備　　考】

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((char *)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】

pname	アナログ入力端子名を文字列で指定します。
adname	スクロール・バー式入力部品の機能名を指定します。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[注]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，ディフォールト値5.0 Vとして動作します。

注　SM850 **システム・シミュレータ** Ver.2.50　**操作編の第7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指定します。

【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，ディフォールト値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指定します。
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指定します。

【戻　り　値】

なし

【備　　考】

この関数を事前に指定せずにボタン関数を指定すると，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】

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指定します。
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指定します。

【戻　り　値】

なし

【備　　考】

この関数を事前に指定せずにLED関数を指定すると，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】

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指定します。
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指定します。

【戻　り　値】

なし

【備　　考】

この関数を事前に指定せずにDCモータ関数を指定すると，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】

type　　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）。

PIC_ELL　　：長方形

PIC_RECT　　：円

となります。

color　　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）。

PIC_RED　　：赤

PIC_YELLOW　　：黄

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(MSG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】

serpname	**シリアル・データ入力端子名文字列を指定します。**
data	**シリアル・データ入力端子に指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指定します。**
count	**シリアル・データ入力端子に指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指定します。**
first	**転送データのビット数分のデータをMSBから順に値に指定するかLSBから順に指定するかを指定します。MSBからの場合は1を，LSBからの場合は0を指定します。**
bitnum	**転送データのビット数を指定します。** **UART（Universal Asynchronous Reciever/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から順に指定する場合

```
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

ウィ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＞ウィンドウのタイトル・バーに名前を表示します。

【パラメータ】

title	**＜入出力パネル＞ウィンドウのタイトル・バーに表示したい名前を文字列で指定します。**

【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指定します。ビットマップ・ファイルの存在するビットマップ・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。

【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指定します。**

【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(*hUwnd*)	53
ウィンドウ・クローズ関数	UpCloseUserWnd(*hwnd*)	53
シミュレーション・コール関数	UpCallFuncName(*fname*)	54
モータ端子通知関数	UpInitMtrPin(*pname,actype*)	55
ステッピング・モータ通知関数	UpInitStpingMtr(*pname, num, actype, reiji, step*)	56
端子のアクティブ値通知関数	UpInitPin(*pname, actype*)	57
ポートのアクティブ値通知関数	UpInitPort(*portname, actype*)	58
AD入力端子の通知関数	UpInitAD(*pname*)	59
プロジェクト・ファイル読み込み関数名の通知関数	UpLoadProjName(*funcname*)	60
プロジェクト・ファイル保存関数名の通知関数	UpSaveProjName(*funcname*)	61
リセット関数名の通知関数	UpResetFuncName(*funcname*)	62
端子の値取得関数	UpGetPin(*pname, val*)	63
ポートのデータ取得関数	UpGetPort(*portname, data*)	64
DA出力端子の値取得関数	UpGetDA(*pname, val*)	65
メモリ領域のデータ取得関数	UpGetMem(*addr, data*)	66
DCモータ用アクティブ時間クリア関数	UpClrMtrAcClk(*pname*)	67
ステッピング・モータ情報取得関数	UpGetStpingMtr(*pnames, num, posrev, negrev, angle*)	68
端子に値をセットする関数	UpSetPin(*pname, val, time*)	69
ポートにデータをセットする関数	UpSetPort(*portname, data, time*)	70
AD入力端子に値をセットする関数	UpSetAD(*pname, val*)	71
メモリ領域にデータをセットする関数	UpSetMem(*addr, data*)	72
モータ用アクティブ時間通知関数	UpGetMtrAcClk(*pname, val, actime*)	73
1メイン・システム・クロック秒換算通知関数	UpSimtimeSec(*void*)	74
USB機能でHOSTからパケット送信を行う関数	UpSetUSBPack(*total, total_bit, data*)	75
USB機能でFunctionからパケットを受け取る関数	UpGetUSBPack(*total, data*)	76
USB機能でHOSTからの信号送信関数	UpSetUSBSig(*sig*)	77
USB機能でFunctionからの信号を受け取る関数	UpGetUSBSig(*sig*)	78

ウィンドウのハンドルの通知関数

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

【機　　能】

ユーザが作成したウィ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ィンドウを作成した直後に指定します。

【パラメータ】

hUwnd	**ユーザが作成したウィンドウのハンドルです。**

【戻　り　値】

なし

【　　例　　】

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

ウィンドウ・クローズ関数

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

【機　　能】

ユーザが作成したウィ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ィンドウ・コール・バック関数のメッセージWM_DESTROY**でこの関数を指定します。**

【パラメータ】

hwnd	**ユーザが作成した，クローズするウィ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( )**の中に指定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指定します。**

【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    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( )の中に指定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，この関数を指定する必要はありません。

【パラメータ】

pname	モータに接続した端子名を文字列で指定します。
actype	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( )**の中に指定してください。ただし，ステッピング・モータを使用しない場合には，この関数を指定する必要はありません。**

【パラメータ】

pnames	**端子名文字列を端子の数だけ指定します**
num	**1チャネルの端子の数（**4**または**8**）を指定します。**
actype	**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**を，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。**
reiji	**励磁方式を指定します。**1**相の場合は**0**を，**1-2**相の場合は**1**を指定します。**
step	**最小ステップ角は**360**を割り切れる整数を指定します。**

【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((char *)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( )の中に指定してください。ただし，端子に値を入力しない場合には，この関数を指定する必要はありません。

【パラメータ】

pname	端子名を文字列で指定します。
actype	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を，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( )の中に指定してください。ただし，ポートに値を入力しない場合には，この関数を指定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指定します。
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を，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つ分指定します。

【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( )に指定してください。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，この関数を指定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指定します。

【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

```
int     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・ウィ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( )の中に指定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 *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・ウィ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 UpLoad_usrproj(char *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

```
int     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・ウィ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( )の中に指定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指定します。

【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 *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・ウィ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 UpSave_usrproj(char *filename)
```

【　例　】

```
UpSaveProjName(“UpSave_usrproj”);
```

リセット関数名の通知関数

```
int     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・ウィ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( )**の中に指定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リセット関数名を指定します。**

【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 Upreset_usrwin(void)
```

【　例　】

```
UpResetFunName("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】

pname	端子名を文字列で指定します。
val	端子の値を格納する領域へのポインタを指定します。

【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指定します。
data	ポートのデータを格納する領域へのポインタを指定します。

【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[注]　0

注　ポートの値に不定値が含まれていた場合も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指定します。
val	DA出力端子の値を格納する領域へのポインタを指定します。

【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	データ格納領域を指定します。

【戻　り　値】

正しくデータが取得できた場合　　1

正しく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指定します。**

【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】

pnames	端子名文字列を端子の数だけ指定します。
num	1チャネルの端子の数（4または8）を指定します。
posrev	正方向回転数を格納する領域を指定します。
negrev	負方向回転数を格納する領域を指定します。
angle	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((char *)mtrpin, 4,HIGH,1,10);
    :
UpGetStpingMtr((char *)mtrpin, 4,&posrev,&negrev,&angle);
```

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	端子名を文字列で指定します。
val	端子のアクティブ時の値を指定します。
time	データを保持する時間を指定します。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す。

【　例　】

UpInitPin(“p31”,HIGH)と指定し，端子P31をアクティブ・ハイと通知しておいた場合，次の指定で，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指定します。
data	ポートに設定する値を指定します。
time	データを保持する時間を指定します。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)と指定し，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，次の指定で，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指定します。
val	AD入力端子に入力する値を指定します。

【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	**データを指定します。**

【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】

pname	**モータに接続した端子名を文字列で指定します。**
val	**端子の値を設定します。**
actime	**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。** actime[1]×0x100000000＋actime[0] **例**　actime[1] = 0x390;　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));
```

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】

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

【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指定します。
data	送信したいパケット・データ配列へのポインタを指定します。

【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

シミュレータのディバッガ部から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

シミュレータのディバッガ部から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。

6.1.2，6.2.2で挙げるソース・プログラムのうち，

①は，ターゲットのプログラムです。
CA850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.OUT）を作成します。

②以降は，カスタマイズ部品の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オプションを指定してください。
（ /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。）

備考　カスタマイズ部品のソース・プログラム例は，16ビットのダイナミック・リンク・ライブラリから32ビットのダイナミック・リンク・ライブラリ作成への変更をわかりやすくするためifdef文などを使用して書かれています。
SM850 V2.00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。

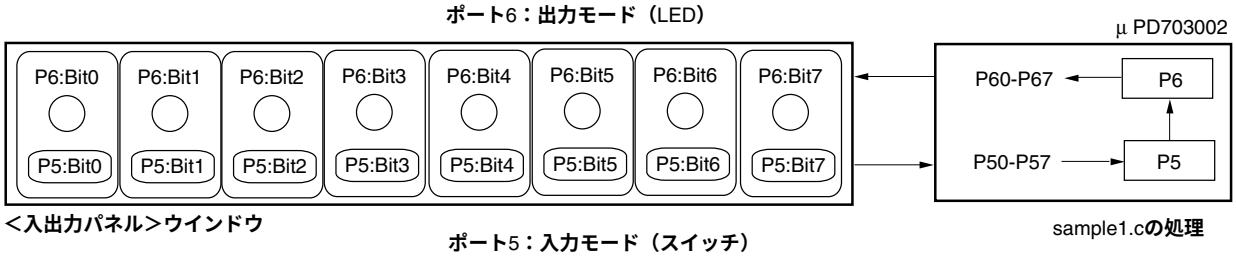
6. 1　＜入出力パネル＞ウインドウによるカスタマイズ部品例

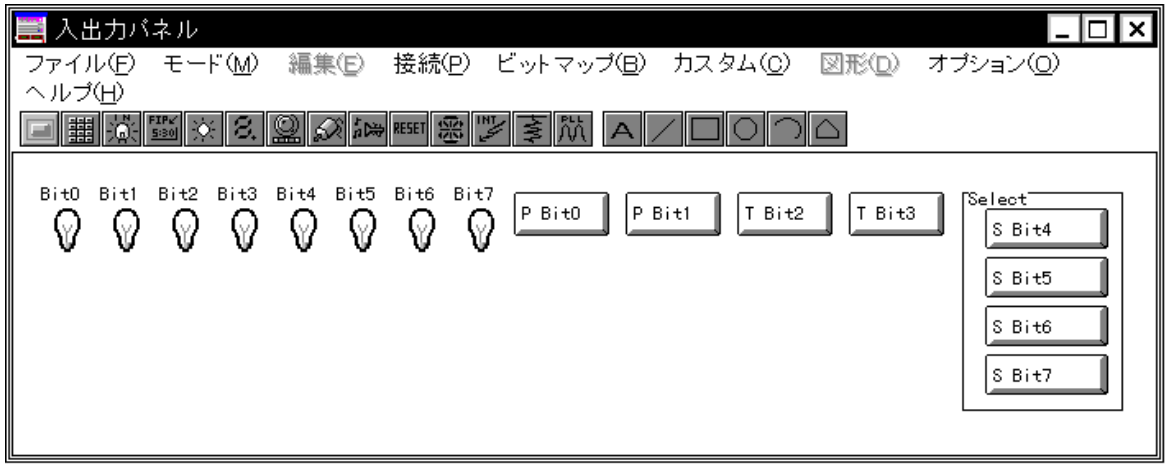
6. 1. 1　サンプル内容

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＞ウインドウに表示し，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＞ウインドウによるカスタマイズ部品例





6. 1. 2　ソース例

①ターゲット・プログラム（V852用のプログラム）

（1/1）SAMPLE1.C

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

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

②カスタム品ソース・ファイル　UPsw00.c

（1/3）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;


#if 1         /*16ビット版のときは0にしてください。*/
  #include    "uparts32.h"

#else
  #include    "uparts.h"

#endif



#ifdef WIN32
  BOOL APIENTRY DllMain(HANDLE, DWORD, LPVOID);

#else
  BOOL LibMain(HANDLE, WORD, WORD, LPSTR);
  int  WEP(int);

#endif

void UParts_sw00(void);
```

（2/3）UPsw00.c

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

#ifdef WIN32

  BOOL APIENTRY DllMain(HANDLE hModele, DWORD ul_reason_for_call, LPVOID lpReserved)
  {
     return(TRUE);
  }

#else

  BOOL LibMain(HANDLE hInstance, WORD wDataSeg, WORD cbHeapSize, LPSTR lpszCmdLine)
  {
     if(cbHeapSize > 0) {
            UnlockData(0);
     }
    return(TRUE);
  }


#endif

#ifndef  WIN32
/************************************************************************/
/*     WEP                                                              */
/************************************************************************/

  int WEP(int nParameter)
  {
     switch(nParameter) {
              case WEP_SYSTEM_EXIT:
                                 break;
              case WEP_FREE_DLL:
                                 break;
     }

     return(1);
  }

#endif
```

（3/3）UPsw00.c

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

③定義ファイル　UPsw00.def

（1/1）UPsw00.def

```
LIBRARY          UPSW00

;EXETYPE         WINDOWS 3.1

DESCRIPTION      'User Open I/F Panel sw00'

;STUB            'WINSTUB.EXE'

CODE             PRELOAD MOVEABLE DISCARDABLE
DATA             PRELOAD SINGLE

HEAPSIZE  3072

EXPORTS
;  WEP                          @1
   UParts_sw00                  @2

;IMPORTS
;  SU850.UpSetPBtmtime
;  SU850.UpLed
;  SU850.UpPushBtm
;  SU850.UpTglBtm
;  SU850.UpSelectBtm
```

④メイク・ファイル　UPsw00.mak

（1/6）UPsw00.mak

```
# Microsoft Developer Studio Generated NMAKE File, Based on upsw00.dsp
!IF "$(CFG)" == ""
CFG=upsw00 - Win32 Debug
!MESSAGE 構成が指定されていません。ﾃﾞﾌｫﾙﾄの 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 ﾋﾞﾙﾄﾞ ﾓｰﾄﾞ "$(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 ｺﾏﾝﾄﾞ 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 選択可能なﾋﾞﾙﾄﾞ ﾓｰﾄﾞ:
!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

!IF  "$(CFG)" == "upsw00 - Win32 Release"

OUTDIR=.¥Release
INTDIR=.¥Release
# Begin Custom Macros
OutDir=.¥Release
# End Custom Macros

ALL : "$(OUTDIR)¥upsw00.dll"
```

（2/6）UPsw00.mak

```
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=cl.exe
CPP_PROJ=/nologo /Zp1 /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_MBCS"
/D "_USRDLL" /D "UPSW00_EXPORTS" /Fp"$(INTDIR)¥upsw00.pch" /YX /Fo"$(INTDIR)¥¥" /Fd"
$(INTDIR)¥¥" /FD /c

.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) $<
<<
```

（3/6）UPsw00.mak

```
.cxx{$(INTDIR)}.sbr::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

MTL=midl.exe
MTL_PROJ=/nologo /D "NDEBUG" /mktyplib203 /win32
RSC=rc.exe
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 /in
cremental:no /pdb:"$(OUTDIR)¥upsw00.pdb" /machine:I386 /def:".¥Upsw00.def" /out:"$(OU
TDIR)¥upsw00.dll" /implib:"$(OUTDIR)¥upsw00.lib"
DEF_FILE= ¥
       ".¥Upsw00.def"
LINK32_OBJS= ¥
       "$(INTDIR)¥Upsw00.obj" ¥
       ".¥si85032.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"
```

（4/6）UPsw00.mak

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

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

CPP=cl.exe
CPP_PROJ=/nologo /Gd /Zp1 /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS
" /D "_MBCS" /D "_USRDLL" /D "UPSW00_EXPORTS" /Fp"$(INTDIR)¥upsw00.pch" /YX /Fo"$(INT
DIR)¥¥" /Fd"$(INTDIR)¥¥" /FD /c

.c{$(INTDIR)}.obj::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

.cpp{$(INTDIR)}.obj::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

.cxx{$(INTDIR)}.obj::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

.c{$(INTDIR)}.sbr::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<
```

（5/6）UPsw00.mak

```
.cpp{$(INTDIR)}.sbr::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

.cxx{$(INTDIR)}.sbr::
   $(CPP) @<<
   $(CPP_PROJ) $<
<<

MTL=midl.exe
MTL_PROJ=/nologo /D "_DEBUG" /mktyplib203 /win32
RSC=rc.exe
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 /in
cremental:yes /pdb:"$(OUTDIR)¥upsw00.pdb" /debug /machine:I386 /def:".¥Upsw00.def" /o
ut:"$(OUTDIR)¥upsw00.dll" /implib:"$(OUTDIR)¥upsw00.lib" /pdbtype:sept
DEF_FILE= ¥
       ".¥Upsw00.def"
LINK32_OBJS= ¥
       "$(INTDIR)¥Upsw00.obj" ¥
       ".¥si85032.lib"

"$(OUTDIR)¥upsw00.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF



!IF "$(NO_EXTERNAL_DEPS)" != "1"
!IF EXISTS("upsw00.dep")
!INCLUDE "upsw00.dep"
```

（6/6）UPsw00.mak

```
!ELSE
!MESSAGE Warning: cannot find "upsw00.dep"
!ENDIF
!ENDIF


!IF "$(CFG)" == "upsw00 - Win32 Release" || "$(CFG)" == "upsw00 - Win32 Debug"
SOURCE=.¥Upsw00.c

"$(INTDIR)¥Upsw00.obj" : $(SOURCE) "$(INTDIR)"



!ENDIF
```

6. 2　ユーザ・ウインドウによるカスタマイズ部品例

6. 2. 1　サンプル内容

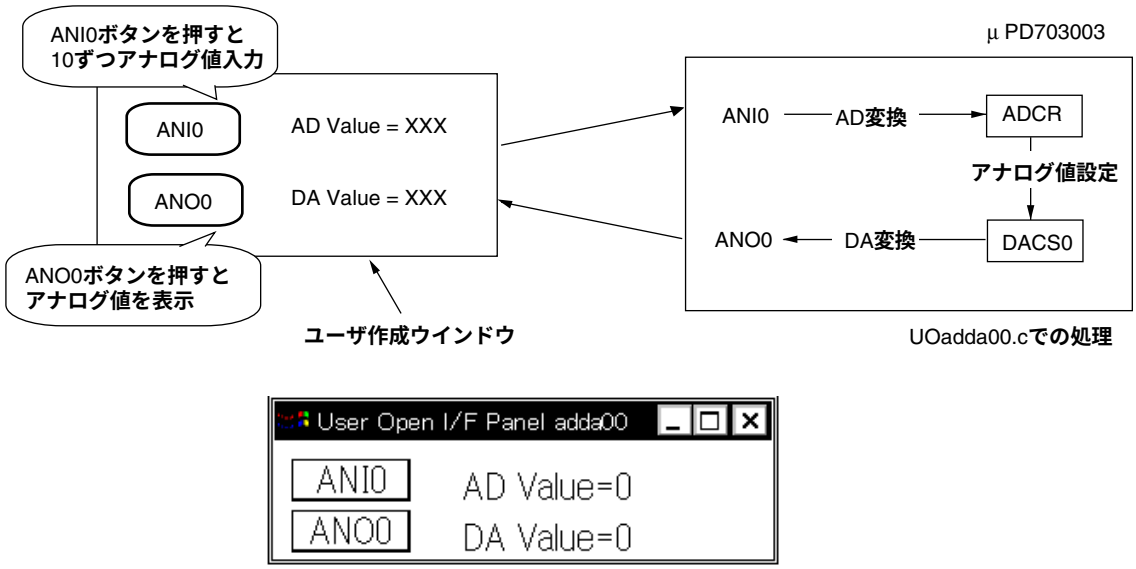
ユーザ・カスタム・ウインドウに，AD入力端子に値を設定する部品とDA出力端子からの値を取得する部品を表示します。

ユーザ・カスタム・ウインドウのANI0ボタンを押すと，アナログ入力端子ANI0のアナログ値入力が10ずつ加算されていきます。「AD Value = 250」となると，0に戻ります。

このANI0からのアナログ値をAD変換し，結果のデジタル値をさらにDA変換しANO0へアナログ値を出力させます。ユーザ・カスタム・ウインドウのANO0ボタンを押したタイミングで，「DA 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・ウインドウによるカスタマイズ部品例



6. 2. 2 ソース例

①ターゲット・プログラム（V853用のプログラム）

（1/1）SAMPLE2.C

```
#pragma ioreg
#pragma          interrupt INTAD  func1
__interrupt void func1(void);

__interrupt void func1(void)
{
__DI();
       DACS0 = (char)ADCR0;
       ADM0 = 0x90;
__EI();
return;
}

main()
{
       ADM1 = 0x07;
       ADM0 = 0x90;
       DAM = 0x01;
       ADIC = 0x02;
       __EI();
       while(1)
       {
       }
}
```

②カスタム品ソース・ファイル　UOadda00.c

（1/8）UOadda00.c

```
/*
 *     User Open I/F  Sample Program     (UOadda00.c)
 */

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

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

#ifdef WIN32
  #include    <Windows.h>

#else
  #include    <Windowsx.h>

#endif

#if 1         /*16ビット版の時は，0にしてください。*/
  #include    "uparts32.h"

#else
  #include    "uparts.h"

#endif

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


#ifdef WIN32
  BOOL APIENTRY DllMain(HANDLE, DWORD, LPVOID);

#else
  BOOL LibMain(HANDLE, WORD, WORD, LPSTR);
  int  WEP(int);

#endif
```

（2/8）UOadda00.c

```
void UParts_adda00(void);
LONG UParts_adda00WndProc(HWND, unsigned, WORD, LONG);
void UParts_adda00Call(ULONG);
void UParts_adda00Reset(void);
void UParts_adda00LoadProj(char *);
void UParts_adda00SaveProj(char *);

/* Window point */
#define      UParts_adda00WIDTH  300
#define      UParts_adda00HEIGHT 100

/* Title Strings */
#define      STR_UP_TITLE  "User Open I/F Panel adda00"

/* Window Class Name */
const BYTE     cnUParts_adda00[] = "UParts_adda00Win";

HANDLE hInst;
HWND   hUParts_adda00Wnd;
HWND   btm_hwnd[2];
char   *strbuf[2] = {"ANI0","ANO0"};
USHORT adval = 0;
USHORT daval = 0;
char   UParts_Veiw_str[7];
char   UParts_Rect_str[23];

#ifdef WIN32
/************************************************************************/
/*     DLL Main                                                         */
/************************************************************************/
  BOOL APIENTRY DllMain(HANDLE hModele, DWORD ul_reason_for_call, LPVOID lpReserved)
  {
    WNDCLASS  wndclass;

    switch(ul_reason_for_call){
       case DLL_PROCESS_ATTACH:
           hInst = hModele;
           wndclass.lpszClassName = (LPSTR)cnUParts_adda00;
           wndclass.hInstance     = hInst;
           wndclass.lpfnWndProc    = (WNDPROC)UParts_adda00WndProc;
           wndclass.hCursor        = NULL;
           wndclass.hIcon          = NULL;
           wndclass.lpszMenuName  = NULL;
           wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
```

（3/8）UOadda00.c

```
          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);
}

#else

  BOOL LibMain(HANDLE hInstance, WORD wDataSeg, WORD cbHeapSize, LPSTR lpszCmdLine)
  {
    WNDCLASS  wndclass;

    if(cbHeapSize > 0) {
        UnlockData(0);
        UnlockSegment(wDataSeg);
    }

    wndclass.lpszClassName  = (LPSTR)cnUParts_adda00;
    wndclass.hInstance      = hInstance;
    wndclass.lpfnWndProc    = (WNDPROC)UParts_adda00WndProc;
    wndclass.hCursor        = NULL;
    wndclass.hIcon          = NULL;
    wndclass.lpszMenuName   = NULL;
    wndclass.hbrBackground  = COLOR_WINDOW + 1;
    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;

    RegisterClass(&wndclass);

     hInst = hInstance;
```

（4/8）UOadda00.c

```
    return(TRUE);
}

#endif

#ifndef WIN32
/*************************************************************************/
/*     WEP                                                               */
/*************************************************************************/
  int WEP(int nParameter)
  {
    switch(nParameter) {
          case WEP_SYSTEM_EXIT:
                       break;
          case WEP_FREE_DLL:
                       break;
  }
    return(1);
}

#endif

/*************************************************************************/
/*     UParts_adda00(void)                                               */
/*************************************************************************/
void UParts_adda00(void)
{
    if(!hUParts_adda00Wnd){
        hUParts_adda00Wnd = CreateWindow((LPSTR)cnUParts_adda00, /* Class name */
        STR_UP_TITLE,                                            /* Title.  */
        WS_OVERLAPPEDWINDOW | WS_BORDER   | WS_VISIBLE,          /* Style bits. */
        CW_USEDEFAULT,                                           /* x - default.*/
        CW_USEDEFAULT,                                           /* y - default. */
        UParts_adda00WIDTH,                                      /* cx - default.*/
        UParts_adda00HEIGHT,                                    /* cy - default.*/
        NULL,                                                   /* No parent.*/
        NULL,                                                   /* Class memu.*/
        hInst,                                                  /* Creator */
        NULL);                                                  /* Params. */
    }
    if(hUParts_adda00Wnd){
        UpSetUserWnd(hUParts_adda00Wnd);
        ShowWindow(hUParts_adda00Wnd, SW_SHOW);
        UpCallFuncName((char *)"UParts_adda00Call");
```

（5/8）UOadda00.c

```
        UpResetFuncName((char *)"UParts_adda00Reset");
        UpLoadProjName((char *)"UParts_adda00LoadProj");
        UpSaveProjName((char *)"UParts_adda00SaveProj");
        UpInitAD("ANI0");
        UpInitAD("ANI1");
    }
    return;
}

/*****************************************************************************/
/*     UParts_adda00WndProc(HWND, unsigned WORD, LONG)                       */
/*****************************************************************************/
LONG UParts_adda00WndProc(HWND hWnd, unsigned iMessage, WORD wParam, LONG lParam)
{
    HDC             hDC;
    PAINTSTRUCT     ps;
    RECT      wRect;
    int       i;
    char      strval[20];

 #ifdef WIN32
    long      wx,wy;

 #else
    WORD      wx, wy;

 #endif

    switch(iMessage){
    case WM_CREATE:
        for(i = 0; i < 2; i++){

          #ifdef  WIN32

            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);

          #else
                    btm_hwnd[i] = CreateWindow((LPSTR)"button", strbuf[i],
```

（6/8）UOadda00.c

```
                    WS_CHILD | BS_PUSHBUTTON | WS_VISIBLE | WS_TABSTOP,
                   10, 10 + 30 * i,
                    BTN_WIDTH, BTN_HIGHT,
                    hWnd, IDD_PIN_BUTTON + i, hInst, NULL);
        #endif
    }
    return(FALSE);

case WM_COMMAND:
    switch(wParam){
      case IDD_PIN_BUTTON:
           if(adval <= 245)
               adval += 10;
           else
               adval = 0;
           UpSetAD("ANI0", adval);
           InvalidateRect(hWnd, NULL, TRUE);
           UpdateWindow(hWnd);
           break;
      case IDD_PIN_BUTTON + 1:
           UpGetDA("ANO0", &daval);
           InvalidateRect(hWnd, NULL, TRUE);
           UpdateWindow(hWnd);
           break;
    }
    return(FALSE);

case WM_PAINT:
    hDC = BeginPaint(hWnd, &ps);
    wsprintf(strval, "AD Value=%u¥0", adval);
    TextOut(hDC, BTN_WIDTH + 40, 15, strval, lstrlen(strval));
    wsprintf(strval, "DA Value=%u¥0", daval);
    TextOut(hDC, BTN_WIDTH + 40, 45, strval, lstrlen(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;
```

（7/8）UOadda00.c

```
        if((wx != 36) && (wy != 36)) {
                wsprintf(UParts_Rect_str, "%d, %d, %d, %d", wRect.left, wRect.top, wx, w
y);
        }
        InvalidateRect(hWnd, NULL, TRUE);
        break;

    case WM_SIZE:
        if(wParam == SIZEICONIC) {
                lstrcpy(UParts_Veiw_str, "Icon");
        } else {
                GetWindowRect(hWnd, &wRect);
                lstrcpy(UParts_Veiw_str, "Normal");
                wsprintf(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_adda00Call(ULONG)                                         */
/************************************************************************/
void UParts_adda00Call(ULONG time)
{
        return;
}
/************************************************************************/
/*     UParts_adda00Reset(void)                                         */
/************************************************************************/
void UParts_adda00Reset(void)
{
    adval = 0;
    daval = 0;
    InvalidateRect(hUParts_adda00Wnd, NULL, TRUE);
}
```

（8/8）UOadda00.c

```
/************************************************************************/
/*     UParts_adda00LoadProj(char *)                                    */
/************************************************************************/
void UParts_adda00LoadProj(char *fname)
{
   char      *next;
   WORD      x, y, wx, wy;

   GetPrivateProfileString("UOadda00", "Window", "Hide", UParts_Veiw_str, 7, fname);
   if(!lstrcmp(UParts_Veiw_str, "Icon")){      /* "Icon" mode */
      ShowWindow(hUParts_adda00Wnd, SW_SHOWMINNOACTIVE);
   }
   else {     /* "Normal" mode */
      GetPrivateProfileString("UOadda00", "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_adda00Wnd, x, y, wx, wy, TRUE);
      }
      ShowWindow(hUParts_adda00Wnd, SW_SHOWNORMAL);
   }
}

/************************************************************************/
/*     UParts_adda00SaveProj(char *)                                    */
/************************************************************************/
void UParts_adda00SaveProj(char *fname)
{
   WritePrivateProfileString("UOadda00", "Window",   UParts_Veiw_str, fname);
   WritePrivateProfileString("UOadda00", "Geometry", UParts_Rect_str, fname);
}

/* UOadda00.c */
```

③定義ファイル　UOadda00.def

（1/1）　UOadda00.def

```
LIBRARY             UOADDA00

;EXETYPE            WINDOWS 3.1

DESCRIPTION   'User Open I/F Panel adda00'

;STUB         'WINSTUB.EXE'

CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE SINGLE

HEAPSIZE      4096

EXPORTS
;             WEP                         @1
              UParts_adda00               @2
              UParts_adda00WndProc        @3
              UParts_adda00Call           @4
              UParts_adda00Reset          @5
              UParts_adda00LoadProj       @6
              UParts_adda00SaveProj       @7

;IMPORTS      SU78K0.UpSetUserWnd
;             SU850.UpCloseUserWnd
;             SU850.UpCallFuncName
;             SU850.UpResetFuncName
;             SU850.UpLoadProjName
;             SU850.UpSaveProjName
;             SU850.UpInitAD
;             SU850.UpSetAD
;             SU850.UpGetDA
```

④メイク・ファイル　UOadda00.mak

（1/5）UOadda00.mak

```
# Microsoft Developer Studio Generated NMAKE File, Based on uoadda00.dsp
!IF "$(CFG)" == ""
CFG=uoadda00 - Win32 Debug
!MESSAGE 構成が指定されていません。ﾃﾞﾌｫﾙﾄの uoadda00 - Win32 Debug を設定します。
!ENDIF

!IF "$(CFG)" != "uoadda00 - Win32 Release" && "$(CFG)" != "uoadda00 - Win32 Debug"
!MESSAGE 指定された ﾋﾞﾙﾄﾞ ﾓｰﾄﾞ "$(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 ｺﾏﾝﾄﾞ ﾗｲﾝ上でﾏｸﾛの設定を定義します。例:
!MESSAGE
!MESSAGE NMAKE /f "uoadda00.mak" CFG="uoadda00 - Win32 Debug"
!MESSAGE
!MESSAGE 選択可能なﾋﾞﾙﾄﾞ ﾓｰﾄﾞ:
!MESSAGE
!MESSAGE "uoadda00 - Win32 Release" ("Win32 (x86) Dynamic-Link Library" 用)
!MESSAGE "uoadda00 - 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)" == "uoadda00 - Win32 Release"

OUTDIR=.¥Release
INTDIR=.¥Release
# Begin Custom Macros
```

（2/5）UOadda00.mak

```
OutDir=.¥Release
# End Custom Macros

ALL : "$(OUTDIR)¥uoadda00.dll"



CLEAN :
       -@erase "$(INTDIR)¥Uoadda00.obj"
       -@erase "$(INTDIR)¥vc60.idb"
       -@erase "$(OUTDIR)¥uoadda00.dll"
       -@erase "$(OUTDIR)¥uoadda00.exp"
       -@erase "$(OUTDIR)¥uoadda00.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 "_USRDLL" /D "UOADDA00_EXPORTS" /Fp"$(INTDIR)¥uoadda00.pch" /YX /Fo"$(INTDIR)¥¥" /
Fd"$(INTDIR)¥¥" /FD /c
MTL_PROJ=/nologo /D "NDEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)¥uoadda00.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 /in
cremental:no /pdb:"$(OUTDIR)¥uoadda00.pdb" /machine:I386 /def:".¥Uoadda00.def" /out:"
$(OUTDIR)¥uoadda00.dll" /implib:"$(OUTDIR)¥uoadda00.lib"
DEF_FILE= ¥
       ".¥Uoadda00.def"
LINK32_OBJS= ¥
       "$(INTDIR)¥Uoadda00.obj" ¥
       ".¥si85032.lib"

"$(OUTDIR)¥uoadda00.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<
```

（3/5）UOadda00.mak

```
!ELSEIF  "$(CFG)" == "uoadda00 - Win32 Debug"

OUTDIR=.¥Debug
INTDIR=.¥Debug
# Begin Custom Macros
OutDir=.¥Debug
# End Custom Macros

ALL : "$(OUTDIR)¥uoadda00.dll"

CLEAN :
       -@erase "$(INTDIR)¥Uoadda00.obj"
       -@erase "$(INTDIR)¥vc60.idb"
       -@erase "$(INTDIR)¥vc60.pdb"
       -@erase "$(OUTDIR)¥uoadda00.dll"
       -@erase "$(OUTDIR)¥uoadda00.exp"
       -@erase "$(OUTDIR)¥uoadda00.ilk"
       -@erase "$(OUTDIR)¥uoadda00.lib"
       -@erase "$(OUTDIR)¥uoadda00.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 "UOADDA00_EXPORTS" /Fp"$(INTDIR)¥uoadda00.pch" /YX /Fo"$(INTD
IR)¥¥" /Fd"$(INTDIR)¥¥" /FD /GZ /c
MTL_PROJ=/nologo /D "_DEBUG" /mktyplib203 /win32
BSC32=bscmake.exe
BSC32_FLAGS=/nologo /o"$(OUTDIR)¥uoadda00.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 /in
cremental:yes /pdb:"$(OUTDIR)¥uoadda00.pdb" /debug /machine:I386 /def:".¥Uoadda00.def
" /out:"$(OUTDIR)¥uoadda00.dll" /implib:"$(OUTDIR)¥uoadda00.lib" /pdbtype:sept
DEF_FILE= ¥
       ".¥Uoadda00.def"
```

（4/5）UOadda00.mak

```
LINK32_OBJS= ¥
       "$(INTDIR)¥Uoadda00.obj" ¥
       ".¥si85032.lib"

"$(OUTDIR)¥uoadda00.dll" : "$(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) @<<
```

（5/5）UOadda00.mak

```
   $(CPP_PROJ) $<
<<


!IF "$(NO_EXTERNAL_DEPS)" != "1"
!IF EXISTS("uoadda00.dep")
!INCLUDE "uoadda00.dep"
!ELSE
!MESSAGE Warning: cannot find "uoadda00.dep"
!ENDIF
!ENDIF


!IF "$(CFG)" == "uoadda00 - Win32 Release" || "$(CFG)" == "uoadda00 - Win32 Debug"
SOURCE=.¥Uoadda00.c

"$(INTDIR)¥Uoadda00.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。
関数の項目には，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。

表A－1　エラーの発生する関数の名前

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( )

A. 2. 1　エラー・メッセージ

エラー・メッセージはエラー・ダイアログと共に表示します。

エラー・ダイアログの＜OK＞ボタンをクリックすることにより引き続き処理を行います。

E10100:	指定した端子は存在しません。
原因	設定した端子が存在しません。
ユーザー処置	対象デバイスに存在する端子名を設定してください。
関数	パラメータに端子名の記述があるすべての関数

E10101:	端子名が全角で記述されています。
原因	設定した端子名が全角で記述されています。
ユーザー処置	半角文字で端子名を記述してください。
関数	パラメータに端子名の記述があるすべての関数

E10103:	保有時間が不正です。
原因	保有時間が範囲内に設定されていないか，または数値ではありません。
ユーザー処置	保有時間を999ミリ秒から0.001ミリ秒の範囲で設定してください。
関数	保有時間設定関数

E10107:	AVrefが動作電源電圧の範囲内にありません。
原因	AVrefが動作電源電圧の範囲にありません。
ユーザー処置	動作電源電圧内の範囲で設定してください。
関数	基準電圧値設定関数

E1010c:	指定したビットマップファイルが不正です。
原因	ビットマップの登録ダイアログで指定したビットマップ・ファイル名がビットマップ・ファイルではありません。
ユーザー処置	ビットマップ形式のファイルを設定してください。
関数	ビット・マップ設定関数

E10124:	指定した端子はすでに設定されています。
原因	ボタン端子設定ダイアログですでに設定済みの端子を再度設定しようとしました。
ユーザー処置	すでに設定済みの端子は再度設定しないようにしてください。
関数	ボタン関数

E10139:	領域が確保できません。
原因	メモリの確保ができませんでした。
ユーザー処置	他のアプリケーションを終了して，メモリを確保してください。
関数	全関数

E10200:　アクティブH/LがHIGHまたはLOWではありません。
原因　記述したアクティブH/LがHIGHおよびLOWでない値です。
ユーザー処置　アクティブH/LにHIGHまたはLOWを記述してください。
関数　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

E10201:　チャネル数が4または8ではありません。
原因　記述したチャネル数が4および8でない値です。
ユーザー処置　チャネル数に4または8（端子名の数にあわせて）を記述してください。
関数　パラメータにチャネル数の記述があるステッピング・モータ関数

E10202:　励磁が0または1ではありません。
原因　記述した励磁の値が0および1でない値です。
ユーザー処置　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。
関数　パラメータにチャネル数の記述があるステッピング・モータ関数

E10203:　最小ステップ角が360を割り切れません。
原因　記述した最小ステップ角が360を割り切れない値です。
ユーザー処置　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。
関数　パラメータにチャネル数の記述があるステッピング・モータ関数

E10204:　図形のタイプがPIC_RECTまたはPIC_ELLではありません。
原因　パラメータの図形のタイプに記述した値がマクロPIC_RECTおよびPIC_ELLでない値です。
ユーザー処置　図形のタイプに　PIC_RECTまたはPIC_ELLを記述してください。
関数　LED図形設定関数

E10205:　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。
原因　パラメータの色のタイプに記述した値がマクロPIC_REDおよびPIC_YELLOWおよびPIC_GREENでない値です。
ユーザー処置　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。
関数　LED図形設定関数

E10206:　表示の図形指定が0または1ではありません。
原因　記述した図の形式の値が0および1でない値です。
ユーザー処置　パラメータ図の形式に値0または1を記述してください。
関数　パラメータに図の形式があるLED関数

E10207:　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。
原因　パラメータの先頭ビットに記述した値が1および0でない値です。
ユーザー処置　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。
関数　シリアル端子データ入力関数

E10208:　出力1のアクティブH/LがHIGHまたはLOWではありません。
原因　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。
ユーザー処置　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。
関数　マトリックスLED関数

E10209:　出力2のアクティブH/LがHIGHまたはLOWではありません。
原因　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。
ユーザー処置　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。
関数　マトリックスLED関数

E1020a:　ポート名が全角で記述されています。
原因　パラメータのポート名が全角で記述されています。
ユーザー処置　パラメータのポート名を半角で記述してください。
関数　ポート値設定／取得関数

E1020b:　指定したポートは存在しません。
原因　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。
ユーザー処置　対象デバイスに存在するポート名を記述してください。
関数　ポート値設定／取得関数

E1020c:　パラメータがNULLポインタです。
原因　関数に不正なパラメータを記述しました。
ユーザー処置　正しいパラメータを記述してください。
関数　ポインタ型のパラメータを持つすべての関数

E1020d:　関数xxxxはUOxxx.dllに記述できません。
原因　UOxxx.dllにこの関数を記述しました。
ユーザー処置　UPxxx.dllで使用してください。
関数　4.1に記述したすべての関数

E1020e:　関数xxxxはUPxxx.dllに記述できません。
原因　UPxxx.dllにこの関数を記述しました。
ユーザー処置　UOxxx.dllで使用してください。
関数　4.2に記述したすべての関数

E1020f:　関数xxxxを事前に通知していません。
原因　必要な通知関数を事前に通知していません。
ユーザー処置　事前に必要な通知関数をコールしてください。
関数　事前に通知関数が必要な関数

E10210:　制御レジスタのタイプがREG_PCまたはREG_PSWまたはREG_SPではありません。
原因　制御レジスタ以外のレジスタを指定しました。
ユーザー処置　制御レジスタ以外は指定しないでください。
関数　レジスタ関係関数

A. 2. 2　ワーニング・メッセージ

ワーニング・メッセージはワーニング・ダイアログと共に表示します。

ワーニング・ダイアログの＜いいえ＞ボタンをクリックすることにより，省略時の値の設定を行わずに処理を続けます。

＜はい＞ボタンをクリックすると，メッセージどおりの処理を行ない処理を続行します。

W10180:　保有時間の設定がありません。0.5msecになりますがよろしいですか？
原因　保有時間の設定がありません。
ユーザー処置　保有時間を設定してください。
関数　保有時間設定関数

W10181:　AVrefの設定がありません。5.0Vになりますがよろしいですか？
原因　AVrefの設定がありません。
ユーザー処置　AVrefの設定を行ってください。
関数　基準電圧値設定関数

W10280:　バックトレース実行中に外部部品からのデータ入力はできません。
原因　バックトレース中のデータ入力は禁止です。
ユーザー処置　バック実行中はデータ入力しないでください。
関数　縦型スクロール・バー式アナログ入力関数

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】

システムLSI第一営業事業部

東　京　(03)3798−6106, 6107, 6108, 6155
大　阪　(06)6945−3178, 3200, 3208
名古屋　(052)222−2375
仙　台　(022)267−8740
水　戸　(029)226−1702
広　島　(082)242−5504
鳥　取　(0857)27−5313
松　山　(089)945−4149

システムLSI第二営業事業部

東　京　(03)3798−6110, 6111, 6112, 6151, 6156
名古屋　(052)222−2170, 2190
松　本　(0263)35−1662
前　橋　(027)243−6060
立　川　(042)526−5981
静　岡　(054)254−4794
金　沢　(076)232−7303
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エレクトロンデバイス　ホームページ】

NECエレクトロン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。

SM850　システム・シミュレータ Ver.2.00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

（U14873JJ2V0UM00（第2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 エレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ