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

- 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、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。



78K シリーズ開発ツール

チュートリアル・ガイド

対象ツール SP78K0 Ver.2.00 SP78K0S Ver.2.00 SP78K4 Ver.2.00

資料番号 U17047JJ1V0AN00(第1版) 発行年月 January 2004 NS CP(K) [メ モ]

目次要約

- 第1章 準備編 ... 11
- 第2章 体験編 ... 14
- 第3章 入門編 ... 34
- 第4章 プログラミング編 ... 85
- 付録… 110

Microsoft, Windows, および Visual C++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。

その他,記載会社名/製品等は,各社の登録商標または商標です。

- 本資料に記載されている内容は2004年1月現在のもので、今後、予告なく変更することがあります。量 産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は,本資料の誤りに関し,一切 その責を負いません。
- 当社は、本資料に記載された当社製品の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、一切その責を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路,ソフトウエアおよびこれらに関する情報は,半導体製品の動作例,応用例を 説明するものです。お客様の機器の設計において,回路,ソフトウエアおよびこれらに関する情報を使 用する場合には,お客様の責任において行ってください。これらの使用に起因しお客様または第三者に 生じた損害に関し,当社は,一切その責を負いません。
- 当社は,当社製品の品質,信頼性の向上に努めておりますが,当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命,身体および財産に対する損害の危険を最小限度にするために,冗長設計,延焼対策設計,誤動作防止設計等安全設計を行ってください。
- 当社は,当社製品の品質水準を「標準水準」,「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また,各品質水準は,以下に示す用途に製品が使われることを意図しておりますので,当社製品の品質水準をご確認ください。
 - 標準水準:コンピュータ,OA機器,通信機器,計測機器,AV機器,家電,工作機械,パーソナル機器,産業用ロボット
 - 特別水準:輸送機器(自動車,電車,船舶等),交通用信号機器,防災・防犯装置,各種安全装置, 生命維持を目的として設計されていない医療機器
 - 特定水準:航空機器,航空宇宙機器,海底中継機器,原子力制御システム,生命維持のための医療機器,生命維持のための装置またはシステム等

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

(注)

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

M8E 02.11



NEC エレクトロニクス製開発環境を初めてご使用になる方へ NEC エレクトロニクス製開発環境の世界へようこそ。 この資料ではサンプル・プログラムを用いて 78K シリーズの開発環境の操 作をわかりやすく紹介しています。

すでに NEC エレクトロニクス製開発環境をお使いの方へ
 シミュレータを用いた疑似画面出力プログラム,スロット・プログラムなどのサンプル・プログラムを用意しました。
 開発環境の再確認のためにどうぞご使用ください。

- **対象者** この資料は,78Kシリーズ開発ツールを初めて使用されるお客様を対象としています。 なお,使用するにあたってマイクロコンピュータ,C言語,アセンブラの一般知識とWindows[®]の 操作方法に関する基礎知識を必要とします。
- **自** 的 この資料は、78Kシリーズ開発ツールの基本的な操作方法をお客様に理解していただくことを目的としています。 資料を読みながら実際にツールを使用することにより、お客様の理解をより深めていただけます。 なお、この資料では78K0の例を使って説明します。78K0Sおよび78K4で異なる部分が ある場合に限り違いについての説明をしています。
- 構 成 この資料は,次の内容で構成しています。

<u>第1章 準備編</u>

この資料で使用する 78K シリーズ開発ツールの概要とサンプル・プログラムのインストー ル方法について解説します。

<u>第2章 体験編</u>

PM plus,システム・シュミレータの基本的な操作方法を,サンプル・プログラムを使用しながら体験できます。78K0 と 78K4 が対象になります。

この章に関連するマニュアルは,関連資料のNo.,, です。

第3章 入門編

システム・シミュレータの基本的なディバグ操作を,サンプル・プログラムを使用しなが ら体験できます。78K0,78K0S,78K4 が対象になります。

この章に関連するマニュアルは,関連資料の No. , , , です。

<u> 第4章 プログラミング編</u>

78K シリーズ CPU 用の機種依存処理を C 言語で記述するための方法について,サンプ ル・プログラムを使用しながら説明します。78K0,78K0S,78K4 が対象になります。 この章に関連するマニュアルは,関連資料の No. , , です。

関連資料 この資料を使用する際,次の関連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につきましては,SP78KxxのインストールCDにPDFファイルが添付されています。

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	No.	
CC78K0 Ver.3.50 以上 C コンパイラ・パッケージ	操作編	U16613J	
	言語編	U14298J	
RA78K0 Ver.3.60 以上 アセンブラ・パッケージ	U16629J		
	言語編	U14446J	
PM plus Ver.5.10		U16569J	
ID78K0-NS Ver.2.52 統合ディバッガ	操作編	U16488J	
SM78Kシリーズ Ver.2.52 システム・シミュレータ	操作編	U16768J	
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		

資料名	資料番号	No.	
CC78K0S Ver.1.50 以上 C コンパイラ・パッケージ	操作編	U16654J	
	言語編	U14872J	
RA78K0S Ver.1.40 以上 アセンブラ・パッケージ	操作編	U16656J	
	言語編	U14877J	
PM plus Ver.5.10		U16569J	
ID78K0S-NS Ver.2.52 統合ディバッガ	操作編	U16584J	
SM78Kシリーズ Ver.2.52 システム・シミュレータ	操作編	U16768J	
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・	U15802J	
	インタフェース仕様編		

資料名	資料番号	No.	
CC78K4 Ver.2.40 以上 C コンパイラ・パッケージ	操作編	U16707J	
	言語編	U15556J	
RA78K4 Ver.1.60 以上 アセンブラ・パッケージ	U16708J		
	言語編	U15255J	
PM plus Ver.5.10	U16569J		
ID78K4-NS Ver.2.52 統合ディバッガ	操作編	U16632J	
SM78Kシリーズ Ver.2.52 システム・シミュレータ	操作編	U16768J	
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・	U15802J	
	インタフェース仕様編		

デバイスに関する資料(ユーザーズ・マニュアル)

資料名	資料番号	No.
μPD780024A, 780034A, 780024AY, 780034AY サブシリーズ	U14046J	

資料名	資料番号	No.
μPD789046 サブシリーズ	U13600J	

資料名	資料番号	No.
μPD784038, 784038Y サブシリーズ ハードウエア編	U11316J	

本サンプル・プログラムおよびプログラム動作環境は 2004 年 1 月現在の資料にもとづいて作成したもので, 今後,予告なしに変更することがあります。

製品のご採用検討に当たっては事前に販売状況等を当社販売員にご確認の上,最新ドキュメントをご参照願います。

目 次

- 第1章 準備編 … 11 使用するツール … 12
 - **サンプル環境** ... 13
- **第2章 体験編** ... 14
 - PM plusの起動 … 16 PM plusの紹介 … 17 ワークスペース・ファイルの読み込み … 19 実行形式の作成 … 21 動作の確認 … 23 システム・シミュレータ(SM78Kxx)の起動 … 24 システム・シミュレータ(SM78Kxx)の紹介 … 26 入出力パネル・ウインドウの紹介 … 27 プログラムの実行 … 28 プログラムの停止 … 31 システム・シミュレータ(SM78Kxx)の終了 … 32 PM plusの終了 … 33
- 第3章 入門編 ... 34

カウンタ・プログラムの仕様 … 35 PM plusの起動 … 37 ワークスペースの新規作成 … 38 ソースの修正と実行形式の作成(1) … 43 システム・シミュレータ(SM78Kxx)の起動 … 47 入出カパネルの設定 … 49 プログラムの実行(1) … 55 ディパグ … 58 ソースの修正と実行形式の作成(2) … 71 プログラムの実行(2) … 74 終 了 … 83

第4章 プログラミング編 ... 85

スロット・プログラムの仕様 … 86 スロット・プログラムの動作確認 … 88 ワークスペース・ファイルの読み込み … 89 実行形式の作成 … 90 システム・シミュレータ(SM78Kxx)の起動 … 91 プログラムの実行 … 92 プログラムの停止 … 94 入出力パネルの解説 … 95 終了 … 100 プログラム解説 … 102

レジスタ名を用いた特殊機能レジスタへのアクセス『#pragma sfr』 ... 103 割り込み関数の登録『#pragma interrupt』または『#pragma vect』と『__interrupt』 ... 104 部分的な割り込みの可否設定『DI();』と『EI();』 ... 106 CPU制御命令出力『HALT();』と『STOP();』と『BRK();』と『NOP();』 ... 108

付録… 110

uoVRAM.dllの作成方法 … 111 カウンタ・プログラム・ソースリスト … 116 スロット・プログラム・ソースリスト … 129



この章では,この資料で使用する開発ツールの概要とサンプル・プログラムについて説明します。

なお,この資料では SP78Kxx に含まれる開発ツールのみで,サンプル・プログラムが動作できるようになっています。

使用するツール

ここでは,この資料で使用する開発ツールの概要について説明します。 開発ツールの名称とその主な機能は,次のとおりです。

デバイス・ファイル

デバイス固有の情報は , デバイス・ファイルに入っているため , ツールを使用するには , デバイス・ ファイルが必要となります。

この資料で使用するサンプルは, 78K0 は DF780034, 78K0S は DF789046, 78K4 は DF784035 を使用してます。

CC78Kxx 78K シリーズ C コンパイラ

78K シリーズ用の組み込み制御用プログラムを C 言語で記述するために開発された,汎用性,移植性の高いCコンパイラです。

RA78Kxx 78K シリーズ アセンブラ・パッケージ

アセンブラ・ソース・プログラムから 78K シリーズで実行することができる実行コードを生成します。

PM plus

Windows 上での統合開発環境です。

エディタ,コンパイラ,ディバッガなどの開発ツールを連携して効率的な開発が行えます。

SM78Kxx 78K シリーズ システム・シミュレータ

ホスト PC 上で, 78K シリーズ用に作られた実行コードをシミュレート実行します。

この資料のサンプル・プログラムを動作させる場合は,上記開発ツールをインストールしていただく必要があ ります。

なお,開発ツールのインストール方法については,SP78Kxx に添付されている"78K シリーズ ソフトウエ ア・パッケージ SP78Kxx 使用上の留意点"をご覧ください。

この資料では,スタート・メニューへの登録は,ディフォールトの "NEC Tools32" という名称で行ったと仮 定して解説します。

サンプル環境

ここでは,この資料で使用するサンプル・プログラムの構造について説明します。

サンプル・プログラム本体のディレクトリ構造

サンプル・プログラム本体をインストールすると,お客様が指定したディレクトリに次のファイルが置かれ ます。なお,Chapter2, Chapter3, Chapter4 ディレクトリに置かれるファイルについては各章で説明します。 次に78K0の場合の例を示します。



78K0S の場合, Chapter2 はありません。



この章では,完成された 78K シリーズ用プログラムをシステム・シミュレータ(SM78Kxx^注)で操作することを 体験します。(注 以降では 78K シリーズの K0, K0S, K4 をまとめて表現するときに Kxx と表します。)

この章の例では外部 RAM を使用しますので 78KOS は対応していません。78KO または 78K4 をご利用ください。 ここでは,78K シリーズ用プログラムとして,画像メモリ(Visual RAM)にパターン画像を書き込むプログラム (以下,VRAM プログラム)を使用します。

VRAM プログラムをビルドし,SM78Kxx で操作することを通して,ツール(PM plus,SM)の基本的な操作方法と,アプリケーション・プログラムの作成時に必要なプロジェクト・ファイルの概念を理解できます。 全体の流れを,次に示します。



この章では,次の環境で VRAM プログラムを動作させます。



SM78Kxx : 78Kxx と, 78Kxx に接続される ROM, RAM および入力ボタンをシミュレートします。

仮想 VRAM 表示プログラム

: SM78Kxx 用のユーザ・カスタマイズ外部部品です。 SM78Kxx で画像 RAM の内容をディスプレイ上に疑似的に表示します。 (このユーザ・カスタマイズ外部部品は,外部部品ユーザ・オープン・インタフェース機能を使用し てこの章のために作成しています。外部部品ユーザ・オープン・インタフェース機能の詳細について は,SM78K シリーズ システム・シミュレータ Ver.2.30 以上 ユーザーズ・マニュアル 外部部品ユー ザ・オープン・インタフェース仕様編(U15802J)を参照してください。)

PM plus の起動

それでは,実際に各ツールを使用してみましょう。

まず,PM plus を起動します。



	🌯 • 🗅 📽 🖬 🎒 🗛 🕹 🖬 🖷	l 🙀 assert	🔽 + + + - 📴 💻	8
		▼ 1	主 塗 キ 本 👪	
	ProjectWindow	OutPut		
	No Workspace	LEW 3		<u>_</u>
		Ш		▼ ▶ //
٩,	ノプを表示するには [F1] を押してください			L: X: Y: //

PM plus の紹介

PM plus では,アプリケーション・プログラムや環境設定を一つの<u>プロジェクト</u>として扱い,エディタでのプロ グラム作成,ソース管理,ビルド,ディバグといった一連の作業を管理します。

メニュー・パー	ツール・バー
PM plus - No Workspace [ProjectWindow] ファイル(E) 編集(E) 検索(D) 階層(L) 表示(V) プロジェクト(P) ビルド(B) ツール(T) ウィントウ(W) ヘルフ°(H)	
_ 🗠 • 🗅 🖆 🖬 🚑 💩 🕺 🛍 🛍 🎇 assert 🔽 🔶 ♦ ♦ ♦	- 1 😫 🔜 💡
Fries Memo Files Memo	
ヘルフを表示するには [F1] を押して(次に)	

プロジェクト・ウインドウ アウトプット・ウインドウ

プロジェクト・ウインドウ : プロジェクト名やそのソース・ファイル,インクルード・ファイルがツリー構造 で表示されるウインドウです。

アウトプット・ウインドウ : ビルドの実行過程が表示されるウインドウです。

 メニュー・バー,およびツール・バーの詳細については,PM plus Ver.5.10 ユーザーズ・マニュアル (U16569J)を参照してください。

ワークスペースとは? 複数のプロジェクト・ファイルのファイル名を管理する単位です。

ワークスペース・ファイルとは? 複数のプロジェクト・ファイルのファイル名などの情報が保存されるファイルです。 ファイル名は "_____.prw"です。 プロジェクトとは?

PM plus のもとで開発されるアプリケーション・システムを指します。 PM plus は環境情報をまとめて"プロジェクト・ファイル"に保存し,参照します。

プロジェクト・ファイルとは? プロジェクトで使用するツールやソース・ファイルなど環境情報が保存されるファイルです。 ファイル名は ".prj"です。 プロジェクト・ファイルは,プロジェクトの新規作成時に設定するプロジェクト・ディレクトリに 作成されます。

ワークスペース・ファイルの読み込み

PM plus は,アプリケーション・プログラムの環境(使用するディレクトリやツール,オプション情報など) をプロジェクト・ファイルに保存します。

また、プロジェクト・ファイルの情報をワークスペース・ファイルに保存します。

この章では,あらかじめ作成されているワークスペース・ファイルとプロジェクト・ファイルを使用します。 ワークスペース・ファイルとプロジェクト・ファイルの作成方法は,「<u>第3章入門編</u>」で説明します。

この章で使用するプロジェクト・ファイルには,完成した VRAM プログラムのソース・ファイル名と SM78Kxx でシミュレートする 78Kxx, ROM, RAM および入力ボタンの設定が保存されています。

仮想 VRAM 表示プログラムは,体験編の一連の作業の中で,プロジェクト・ファイルに追加登録します。

PM plus のメニューの [ファイル(<u>F</u>)] [ワークスペースを開く(<u>W</u>)…]を選択し, "VRAM.prw"を指定してください。

□ 環境を設定していない方は「<u>サンプル環境</u>」をご覧ください。

🧱 PM plus - No Workspace [(OutPut]		
ファイル(<u>F</u>) 編集(E) 検索(D)	階層(L)	表示(V) プロジェクト(P) ビルド(B) ツール(T) ウィンドウ(W) ヘルプ(H)	
新規作成(N) 開(Q) 挿入読み込みの 閉じる(C)	Ctrl+N Ctrl+O	■ ■ ■ assert	8
		CutPut	
ワークスペースを開くいか。 ワークスペースの保存(品) ワークスペースを閉じる(M)		- [EOF]	
上書き保存(S) 名前を付けて保存(A)	Ctrl+S		
ソース・ファイル名の変更(E) すべてのソース・ファイルを保存(U)			
ED刷プレビュー(型) ED刷(Ⴒ)	Ctrl+P		
ファイルの履歴			
ワークスペースの履歴			
PM plus の終了 🖄			
既存のワークスペースを開きます			X:1 Y:1 //

		Chapter2 ディレクトリを 開いてください。
ワークスペースを開く		<u>?</u> ×
ファイルの場所型:	🔁 Chapter2	- 🗧 🔁 📸
UoVRAM_src		
ファイル名(N): ファイルの種類(T):	VRAMprw ワークスペース・ファイル (* prw)	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
	,	

" VRAM.prw "を指定して 開く(<u>O</u>) ボタンを押してください。







プロジェクトの実行形式を作成します。この作業を<u>ビルド</u>と呼びます。

PM plus のビルド・ボタン 🤽 ,またはメニューの [ビルド(<u>B</u>)] [ビルド(<u>B</u>)]を選択してください。



ビルド処理を正常に終了しました^注。

注 ここで,アウトプット・ウインドウに"Can't initialize RAM area"というメッセージが表示されますが, VRAM プログラムの動作には問題ありません。 メッセージの詳細については,RA78Kxx アセンブラ・パッケージのユーザーズ・マニュアルを参照して ください。 ビルドとは?

プロジェクトに登録されているソース・ファイルから実行形式ファイルなどを作成する機能です。 PM plus がコンパイル,アセンブル,リンクなどを自動的に実行します。 また,PM plus は,2度目以降のビルドでは,前回のビルドから更新されたファイルを自動的に 検出し,該当するファイルのみをコンパイル,アセンブルすることにより,ビルドにかかる時間 を短縮しています。

リビルドとは?

ビルドは,前回から更新されたソース・ファイルのみをコンパイル,アセンブルしますが,リビルド ではすべてのソース・ファイルをコンパイル,アセンブルします。

コンパイラ・オプション等,各種設定を変更したときは,ビルドではなくリビルドを選択する必要が あります。

また,ビルドで更新されたファイルの検出に失敗する可能性がある次の場合にも,リビルドを選択 する必要があります。

・修正したソースを修正前の古いソース・ファイルにファイルコピーなどで置き換えたとき

・前回のビルドの後,ホスト・マシンの時刻を訂正したとき

・時刻がずれているホスト・マシン間でプロジェクト環境を移動したとき

動作の確認

NEC エレクトロニクスではユーザ・アプリケーションの実行環境として,<u>統合ディバッガ</u>,および<u>システム・</u> <u>シミュレータ</u>を提供しています。

この資料では、システム・シミュレータ(SM78Kxx)を起動し、動作を確認します。

 統合ディバッガ(ID78Kxx)とは?
 統合ディバッガは,インサーキット・エミュレータとターゲット・システムを接続した開発環境で ディバグを行う Windows ベースのソフトウエア・ツールです。
 C ソース・レベル,またはアセンブラ・レベルでのディバグが可能です。
 インサーキット・エミュレータの持つイベント設定機能を利用して,リアルタイムに実行,検証 できます。

システム・シミュレータ (SM78Kxx)とは? システム・シミュレータは,ホスト・マシン上でターゲット・システムの動作をシミュレーション しながら,ディバグを行う Windows ベースのソフトウエア・ツールです。 C ソース・レベル,またはアセンブラ・レベルでのディバグが可能です。 アプリケーションの論理検証をハードウエア開発から独立して行えます。

システム・シミュレータ (SM78Kxx)の起動

SM78Kxx を起動します。

PM plus のディバグ・ボタン 闘

,またはメニューの [ビルド(<u>B</u>)] [ディバグ(<u>D</u>)] を選択してくださ

L١。

ここで,ディバグ・ボタンが表示されていない場合は,メニューの[ツール(<u>T</u>)] [ディバッガの選択 (<u>D</u>)…]で"SM78Kxx システム・シミュレータ"を選択してください。 詳細については,「<u>第3章 入門編</u>」をご覧ください。





SM78Kxx が起動します。



次に,入出力パネル・ウインドウのメニューの [カスタム(<u>C</u>)] [ロード(<u>L</u>)]を選択し, "uoVRAM.dll" を指定してください。

🔜 入出力パネル 📃 🗖 💌	
ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B)	
カスタム(②) 図形(①) オブション(②) ヘルブ(凹)	
<u> </u>	
1 2 3	
PAUSE	

読み込み			? ×
ファイルの場所型:	🔄 Chapter2	•	🗈 📸 🔳
uoVRAM_src			
」 ファイル名(N):	uoVRAM.dll		- IIK(Q)
ファイルの種類(エ):	DLL ファイル (*.dll)		▼ ↑ ♥𝒴 €₩
	読み込み ファイルの場所型: □uoVRAM_src ■uoVRAMdIII 100VRAMdIII ファイル名(型): ファイルの種類(T):	読み込み ファイルの場所型: Ghapter2 ↓uoVRAM_src ↓uoVRAMdII ファイル名(型): uoVRAMdII ファイルの種類①: DLL ファイル(*.dll)	読み込み ファイルの場所型: Chapter2 UoVRAM_src UoVRAM(dll ファイル名(<u>N</u>): UoVRAM(dll ファイルの種類①: DLL ファイル(*,dll)

ここで, "uoVRAM.dll"ファイルが表示されない場合は, "ファイル名(<u>N</u>):"の欄に "uoVRAM.dll"と直接入力するか,または,エクスプローラでDLLファイルを表示できるように設定 してください。



ここで扱う" uoVRAM.dll "は, VRAM プログラム用に作成されたものですので,使用するにあたって, 設定変更などを行う必要はありません。 " uoVRAM.dll "のソースからの作成方法については,「<u>付録 uoVRAM.dll の作成方法</u>」をご覧ください。 また,詳細については, SM78K シリーズ システム・シミュレータ Ver.2.30 以上 ユーザーズ・マニュア ル 外部部品ユーザ・オープン・インタフェース仕様編(U15802J)を参照してください。

システム・シミュレータ(SM78Kxx)の紹介

SM78Kxx は, メイン・ウインドウとシミュレータ GUI ウインドウから構成されています。

メイン・ウインドウ : CPU コア内部のステータスの表示,およびシミュレータ実行の制御を行い ます。

シミュレータ GUI ウインドウ :外部部品の制御を行います。

SM78Kxxの初期画面は次のとおりです。



ウインドウ表示エリア

ステータス表示エリア

<シミュレータ GUI ウインドウ>



シミュレーション対象ファイル名表示エリア

→ 各エリア,メニュー・バー,およびツール・バーの詳細については,SM78K シリーズ システム・シミュ レータ Ver.2.52 ユーザーズ・マニュアル 操作編(U16768J)を参照してください。

入出力パネル・ウインドウの紹介

SM78Kxx は,疑似的なターゲット・システムを構築するために,ボタンや LED などの標準的な部品を提供しています。

入出力パネル・ウインドウは, それらの部品を設定, および使用するためのウインドウです。

78Kxx シミュレータ GUI ウインドウのメニューの [外部部品(<u>E</u>)] [入出力パネル(<u>P</u>)...]を選択し,開く ことができます。

→ 入出力パネルの設定方法は,「<u>第3章入門編</u>」で説明します。

→ メニュー・バー,およびツール・バーの詳細については,SM78K シリーズ システム・シミュレータ Ver.2.52 ユーザーズ・マニュアル 操作編(U16768J)を参照してください。

	メニュー・バー	ツール・バー	
	/	/	
🧱 入出力パネル			
ファイル(E) モード(<u>M</u>)	編集(E) ▶ 接続(P) ビッ	トマップ(B) カスタ <mark>ム(C) 図形(D)</mark>	オブション(0) ヘルブ(円)
回翻浴圈>	8. 🔐 🔊 i das Reset 👙	®IJ≩∭A/∎C	

ここで用意されていない部品は, "uoVRAM.dll"のように「Microsoft™ Visual C++™」を使用して,自由に 作成することができます。

→ ユーザ作成の外部部品の詳細については,SM78K シリーズ システム・シミュレータ Ver.2.30 以上 ユー ザーズ・マニュアル 外部部品ユーザ・オープン・インタフェース仕様編(U15802J)を参照してください。

プログラムの実行

次に、プログラムを実行します。

SM78Kxx のリスタート・ボタン **・** , またはメニューの[実行(<u>R</u>)] [リスタート(<u>R</u>)]を選択してくださ い。VRAM プログラムを実行します。

SMK032 : VRAM.prj	. <u> </u>
	1 sea 1
Search << >> Watch Quick Refresh Close	
88 /************************************	*/
┃ * 103 statePaint = nowTile; /* 現在の状態をLタイル模種抽画中Jに設 Ready main.c#37 processOpenning 082E	定*/
プログラムを実行し プログラムを実行し	ます。
- menuloce (************************************	
Source (main.c)	_ 0 ×
158 void main()	
167 /####################################	
* 175 End_int(); /* CPUIこ割り込み状態終了を通知 */ * 176 Cng bank0(); /* BANK0に切り替え */	

プログラムの実行中はステータス表示エリアが赤く変化します。

次に,実行中の VRAM プログラムを操作します。

入出力パネルの各ボタンを操作して,画面の変化を確認してください。



< 画面 > <画面 > < 画面 > ■仮想 VRAM - 🗆 × 🚮 仮想 VRAM - U × - 🗆 ×







これで VRAM プログラムが正常に動作していることが確認できました。

ターゲット CPU のリセットとは? ここでいうターゲット CPU とは, SM78K0 がシミュレートしている仮想のµPD780034 です。 78K4の時は µ PD784035 になります。 ターゲット CPU のリセットを行うと, 仮想 µ PD780034 の RESET 端子に low 信号が入力された 状態を SM78K0 がシミュレートします。 これにより,仮想 µ PD780034 上で動作している VRAM プログラムは初期状態に戻ります。 したがって,この操作は,SM78Kxx が動作しているパソコンをリセットすることではありません。



プログラムを停止します。

SM78Kxxの停止ボタン II , またはメニューの [実行(<u>R</u>)] [ストップ(<u>S</u>)]を選択してください。

iii S	MK03:	2 : VRA	Mprj	- D ×				
771	ファイル(E) 編集(E) 表示(V) オブション(Q) 実行(R) イベント(N) ブラウズ(B) ジャンプ(U) ウィンドウ(M) ヘルプ(H)							
(п								
	Sourc	e (mair	no)	_ 🗆 🗵				
	Searc	h	Watch Quick Refresh Close					
ШС	$\left \Box \right $	158	void main()					
*	IP II	159	1 Γ ΓΓΓ΄)• /ω ΦΡΙΙΙΙΑΛΙΖΙΛΦΈΛΙΕ ω/					
*		161	Init(): /* 周辺機能の初期化 */					
*		162	statePause = FALSE; /* 一時停止状態を[非停止(処理中)]に設定 */					
*		163	[] EI(); /* 割り込み許可 */					
*		164	processUpenning(); /* 油画処理 [私期画面]を起動 */					
аран (т. 1997). 1917 — Прила Саран (т. 1917). 1917 — Прила (т. 1917). 1917 —		166						
ш.	167//***********************************							
ш.		168	ボタン ″1″が押された(割り込み)					
ш.		169	Z1%/r• 4am-					
ш.		171	21以 - 流					
ш.		172	********					
		173	yoid pushButton1(void)					
*		1/4	{ 「 「 」 「 」 」 「 」 」 「 」 」 「 」 」 「 」 」 」 「 」 」 」 「 」					
*		176	Chg hank(): /* GFOIC割り込み(A感報) を通知 */	-				
<u> </u>								



<mark>講</mark> SMK032:\ ファイル(E) 編	RAM.prj 集促) 表示(V) オフジョン(Q) 実行(R) イベント(N) ブラウズ(B) ジャンプ(J) ウィントウ(W) ヘルレプ(H)	
∬ II ► ►	<u>- ► ► ► ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ </u>	
Search	<u>≪ >> Watch Quick… Refresh Close</u> 86 HALT(); /* Halt状態に移行 */ 97 }	<u> </u>
	88 89 91 1年画処理 <i>* タイル</i> 寝様* 91 92 31鼓: 無 93 返値: 無 41 ************************************	
* * * * 1	30 [1] FI(); /* 割り込み許可 */ 98 if(statePaint == nowTile){ /* 現在の状態が[タイル模様描画中]なら */ 99 statePaint = nowClear; /* 現在の状態を[画面消去中]に設定 */ 00 paintClear(); /* 画面消去処理 */ 01 }	
*	02 for(;;){ 03 statePaint = nowTile; /* 現在の状態を[タイル模様描画中]に設定 */ 04 paintTile(); /* タイル模様描画処理 */ 1	× 4 ×
Ready	main.c#87 processOpenning 082E B	REAK

プログラムを停止すると,ステータス表示エリアの色が元に戻ります。

システム・シミュレータ(SM78Kxx)の終了

SM78Kxx メニューの [ファイル(<u>F</u>)] [終了(<u>X</u>)]を選択してください。

🚆 SM78K0 : VRAM.prj	
ファイル(E) 編集(E) 表示(V) オプション(Q) 実行(R) イベント(N) フ	ブラウズ(B) ジャンプ(J) ウィンドウ(W) ヘルプ(H)
開(@) Ctrl+O 名前を付けて保存(<u>A</u>) 閉じる(<u>C</u>)	
ダウンロード(<u>(D</u>) アップロード(<u>U</u>)	
プロジェクト(① 環境(E) デバッザ(Hzwh(P))	▶ /* 割り込み禁止 */
於了⊗	/* 播画処理 [初期画面]を起動 */
1_C#NECTools32¥78K0_sample#Chapter2¥VRAMprj 2_C#NECTools32¥78K0_sample#Chapter2#character.Imf	
170 引数:無 171 返値:無 172 ************************************	·/
* 175 End_int(); * 176 Cng_bank0(); * 176 *	/* CPUに割り込み状態終了を通知 */ /* BANKUに切り替え */ /* BANKUに切り替え */
main.c#159 main 0628	



| はい(<u>Y)</u> | を押してください。SM78Kxx が終了します。

PM plus の終了

PM plus のメニューの [ファイル(<u>F</u>)] [PM plus の終了(<u>X</u>)]を選択してください。



PM plus が終了します。



この章では,システム・シミュレータ(SM78Kxx)を用いた基本的なディバグ操作を,サンプル・プログラム を使用しながら解説します。

ここでは, サンプル・プログラムとして<u>カウンタ・プログラム</u>を使用します。

カウンタ・プログラムには,あらかじめいくつかのバグが埋め込まれていますので,実際にこのバグを修正し ながら操作を進めていきましょう。

全体の流れを,次に示します。


カウンタ・プログラムの仕様

この章のディバグ操作を行う前に,カウンタ・プログラムの概要を理解していただく必要があります。 基本的な外部仕様は次のとおりです。

外部仕様

ある装置に,ボタンと2桁の7セグメント LED があり,ボタンを押すとカウンタがカウント・アップします。 78K0Sの場合は,INTWTを使用します。78K4の場合は,INTC00を使用します。



この章では"カウント・アップ機能"と"LED 表示機能"を実装します。ボタン入力と初期化などの処理は, SM78Kxxのディバグ機能を利用した,ディバグ用メイン・ルーチンを使用します。

基本仕様("カウント・アップ機能", "LED 表示機能")

- カウント・アップ機能
 - ➢ INTTM00 割り込みが発生したら、1 つずつカウント・アップする 2 桁の 10 進カウンタです。
 - 99までカウントしたら次は0に戻します(ループ・カウンタにします)。
- LED 表示機能
 - ▶ 10 進カウンタの値を7 セグメント LED に出力します。
 - ▶ 7 セグメント LED の制御には 78Kxx の入出力ポートを使用します。

ディバグ用メイン・ルーチン基本仕様

- カウンタの値を0で初期化します。
- <u>7 セグメント LED</u>制御用の入出力ポートを初期化します。
- INTTM00 割り込みの発生は, SM78K0 で疑似的に内部割り込みを発生させる"<u>内部割り込みボタン</u>"を 利用し,疑似割り込みを受け取るのに必要な部分のみを実装します。
- カウント処理時以外は, CPU を HALT モードにします。

入出力ポートとは? 78K シリーズのほとんどのデバイスには,CPU 外部にある部品などの制御や CPU 外部から信号を 取得する手段の 1 つとして,入出力ポートが装備されています。 入出力ポートの詳細については,各デバイスのユーザーズ・マニュアルを参照してください。

内部仕様は次のとおりです。

内部仕様

● カウンタの値はグローバル変数 count1, count10 に保持します。

	変数	内容				
i	int count1	1桁目(1の位)の値を保持				
i	int count10	2 桁目(10 の位)の値を保持				

● プログラムは, LED の表示, カウント・アップ, ディバグ用メイン・ルーチンの3つの関数で構成します。

処理関数	処理内容
LED の表示	・カウンタの値を LED に表示
void putLED()	
カウント・アップ	・INTTM00 割り込みで起動
void interrupt1()	・カウンタのカウント・アップと桁溢れを処理
	・カウント・アップ後,putLED()関数を起動
ディバグ用メイン・ルーチン	・LED を制御する入出力ポートを初期化
void main()	・INTTM00 割り込みを受け取り可能状態に設定
	・カウンタを 0 で初期化し,putLED()関数を起動し,
	初期値0を表示
	・CPU を HALT モードに移行

● 7 セグメント LED の制御には 78K0, 78K4 は入出力ポート P4, P5 を, 78K0S は P0,P1 を使用します。 P4 は表示内容の設定, P5 は表示する桁を指定します。

ポート	ビット位置	略号	ビットの役割
P4	0	P40 又は P00	0 1の時, P1に出力している内容を1桁目に反映
又は P0	1	P41 又は P01	0 1 の時, P1 に出力している内容を2桁目に反映
P5	0	P50 又は P10	下の横棒の点灯状態を設定 (1…点灯,0…無点灯)
又は P1	1	P51 又は P11	左下の縦棒の点灯状態を設定(1…点灯,0…無点灯)
	2	P52 又は P12	右下の縦棒の点灯状態を設定(1…点灯,0…無点灯)
	3	P53 又は P13	中の横棒の点灯状態を設定 (1…点灯,0…無点灯)
	4	P54 又は P14	左上の縦棒の点灯状態を設定(1…点灯,0…無点灯)
	5	P55 又は P15	右上の縦棒の点灯状態を設定(1…点灯,0…無点灯)
	6	P56 又は P16	上の横棒の点灯状態を設定 (1…点灯,0…無点灯)
	7	P57 又は P17	右下ドットの点灯状態を設定(1…点灯,0…無点灯)



PM plus の起動

まず, PM plus を起動します。

Windows スタート・メニューの [プログラム(<u>P</u>)] [NEC Tools32] [PM plus] を選択してください。



📅 PM plus - No Workspace [ProjectWindow	1				
7ァイル(E) 編集(E) 検索(D) 階層(L) 表	示☑ フロシェクト᠐	ビルド(B) ツール(T)	<u>ሳሪ⊦ን</u> ₩ ∿	ルフ°(<u>H</u>)	
💁 • 🗅 🚅 🖬 🚑 💁 X 🖻	🔁 🛛 🎇 assert		• + +	+ - 12	2 8
		▼ ¥	l 🕸 🖗 🗡		
Files Memo	CutPut - [EOF]				
	I				
^					L: X: Y: //

ワークスペースの新規作成

ワークスペースを新規作成します。

PM plus のメニューの [ファイル(F)] [ワークスペースの新規作成(E)…]を選択してください。



ワークスペース情報を,次のように設定して	\
ください。	
ワークスペース・ファイル名(<u>W</u>)	
counter	
フォルダ位置(<u>F</u>)	
78K0 の場合,	•
¥78K0_sample¥Chapter3	
(参照(<u>R</u>) ボタンで <u>サンプルを展開</u>	
<u>したディレクトリ</u> を選択してください)	ワークスペースの新規(作成) - ステップ*1/8 [ワークスペース情報の回致定] ×1
78K0S の場合 ,	* ケースペース* プァロルーA1020 · counter 「 空のワークスペースを作成する(C) 2. リアルクイペンス情報の設定 2. リアルクイムのSの現在(#)
¥78K0S_sample¥Chapter3	* 7# 人学位置(¹):
78K4 の場合,	71ジェナト・ウルーフを2(2): 5. リンウ・ディレクテロ・ファイルの指定(4) counter program 6. ソース・ファイルの設定
¥78K4_sample¥Chapter3	ソーズ名(S): 78K/0 Series アBK/0 Series PBK/0 SF
ノロジェクト・クルーノ名(<u>G</u>)	デバイス登録(D) (第) 2~5 は Vobul Series (U.M)
counter program	ワーウスペースとフロジェントの基本情報を設定します。
ンリース石(<u>3</u>) 78K/0 Series (78K0 の埋合)	→ ¥は必須入力項目です。
78K/09 Series (78K/09 の場合)	/=====
78K/4 Series (78K4の場合)	
デバイス名(D)	
μ PD780034 (78K0 の場合)	
μ PD789046 (78K0S の場合)	
μ PD784035 (78K4 の場合)	/
	·

設定が完了したら, 次へ(N)> ボタンを押してください。





使用するディバッガを指定します。

ワークスペースの新規作成 - ステップ7/8 [ディバッガの選択]	×
使用するティハッカを選択してください。 * 選択ティハッカ(D): 78K/0 システム・シミュレータ マ 	1. ワークスペース情報の設定 2. リアルタイムOSの選択(#)
771ル名(E): C¥NECTools32¥BIN¥SMK032.EXE	 スタート・アップ・ファイルの指定(#) レジスタ・モードの選択(#) リンウ・ディレクティブ・ファイルの指定(#) ソース・ファイルの設定 ア、ディハッカの選択 設定の確認 (#) 2~5 (1)(850 Series (0.3)
ここで選択したディバッかは後から[ツール]メニュー →[ディバッかの設定]で変更すること **は必須入力項目です。	が可能です。
」 〈戻る(B) (次へ(b)	> キャンセル ヘルプ

次へ(<u>N</u>)> ボタンを押してください。



設定した内容を確認します。

アーウスペース・ファイル名: 	 ► ► 1. ワークスペース情報の設定 2. リアルタイムOSの選択(#) 3. スタート・アップ・ファイルの指定(#) 4. レジスタ・モードの選択(#) 5. リンク・ディレクティブ・ファイルの指定(#) 6. ソース・ファイルの設定 7. ディハッカの選択 >> 8. 設定の確認 (#) 2~5 (はV850 Series のみ
✓	王 キャンセル ヘルプ

設定した内容が間違っていなければ, 完了 ボタンを押してください。



これで,ワークスペースの作成は完了です。

ソース・ファイルは,後から随時追加登録することができます。

ソースの修正と実行形式の作成(1)

プロジェクトをビルドします。

PM plus のビルド・ボタン 🗽 ,またはメニューの [ビルド(<u>B</u>)] [ビルド(<u>B</u>)]を選択してください。



カウンタ・プログラムのソース中にエラーを検出したので,エラー・メッセージが表示されました。 OK ボタンを押してください。

では,エラーの修正を始めましょう。

ソースを修正します。

アウトプット・ウインドウに詳細なエラー・メッセージが表示されています。

エラーの表示行 " counter.c(52):F756 Too many initializers'box " をダブルクリックしてください。





52 行目では,初期値として記述された { }内のデータの個数(10 個)に対して,box 配列の領域が不足しています。

"box[9]"を"box[10]" に修正してください。

	。 C¥NECTools32¥78K0_sample¥Chapter3¥counter.c(変更)	<u>- 🗆 ×</u>
F	**	-
F.	**************	
E.	***************************************	
E.	yoid putLED() *	
E.		
E.	static unsigned char /ギ バビクメント LEU じ - 9 UJバダンギ/*	14. 51
E.	box[10] {Ux//,Ux24,Ux0b,Ux0d,Ux3C,Ux3d,Ux3f,Ux/4,Ux/f,Ux/d};	/*←EI
E.	* /* 7セガリント LED にパターンを転送 (1の位) */*	
L.	$P_4 = 0.1$	
Ŀ.	P5 = hox[count1]:k	
ŀ.	P4 = 1: *	
F		_

これで, ソースの修正が終了しました。

エラーの修正が終了したので,再びビルドします。 PM plus のビルド・ボタン **…** ,またはメニューの [ビルド(<u>B</u>)] [ビルド(<u>B</u>)] を選択してください。 PM plus のエディタ機能の場合,ソースの修正内容はビルド時に自動的にセーブされます。



ビルド処理が正常に終了し,実行形式が作成されました。

実行形式のディフォールトのファイル名は,先頭に登録されたソース名.lmfです。

システム・シミュレータ (SM78Kxx)の起動

SM78K0 を起動します。

PM plus のディバグ・ボタン 🞇 ,またはメニューの [ビルド(<u>B</u>)] [ディバグ(<u>D</u>)]を選択してください。



このダイアログでは,メモリ・マッピング,クロックなどの設定を行います。

ただし,この章で扱うカウンタ・プログラムは µPD78xxxx に内蔵されている ROM と RAM で動作するため, ここでは設定を変更する必要はありません。

OK ボタンを押してください。



SM78Kxx のメイン・ウイン ドウが開きます。

	SMKO	32 : cou	nter.prj			6.51.60					JN
77	чルŒ)漏果	£) 表示(⊻)	オフッション(<u>O</u>)	美行吧	1^\2N <u>₪</u>	- ブラワズ(<u>B</u>) 97697°(<u>J</u>	$(-97)^{-1}$	• ^//ブ(<u>H</u>	,
Ι		►) FI FI 4	- IT 🗾	s 🗈 [D 🖷 💐	0 🔏		1 📢 💽	📰 📍 🛛	7
	Sour	ce (cour	iter.c)							?	< I
		ch 75 76 77 78 78 80 80 81 82 84 85 86 85 88 84 85 86 85 88 89 90 91 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 92 93 94 95 93 95 95 95 95 95 95 95 95 95 95 95 95 95	★ ・LED * ・力ウ * ・ INT * ・ INT * ・ INT * * ・ INT * ま次 * 関引り の * 関引り の の * で の り の り の の り の の の の の の の の の の の の	Watch 表示の為にP ワンタ値(cou TM00割込み; iNTTM00割込の))INTTM00割う)INTTM00割う : なし ローバル変数 int count1 int count1 int count1 ************* **********************	Quic 5, P4の: nt1, cot を初期すたし、 を発生みが約 込みを待: 0 **********************************	 k Ref モードをPMI John 10)を初し、 (PUをHAL) CPUをHAL (PUでのののののののののののののののののののののののののののののののののののの	resh 5, PM41こよ 期化する。 を許可する だ許可する ではいたHALT ************************************	Close り設定する る。 *********************************	c ************************************		
Rea	ady				COL	unter.c # 96		main	008A		

入出カパネルの設定

プログラムを実行する前に,カウンタ・プログラムで使用するボタンと 7 セグメント LED の設定を行います。 まず,入出力パネルでボタンの設定を行います。

入出力パネル・ウインドウは, 78Kxx シミュレータ GUI ウインドウから開くことができます。

では,タスク・バーの"78Kxx シミュレータ GUI"を選択して,78Kxx シミュレータ GUI ウインドウを開い てください。

🏦 ८५ 📙 🍘 🗊 🧭 🗍 🧱 Project Manager	- co 🛛 🚔 idea-L(counter.prj) - c 🕅	\$MK032:counter .
		78Kxx シミュレータ GUI ウインドウが開きます。
〒78K/0 シミュレータ GUI ファイル(E) 外部部品(E) P25/ASCK0 に 0 を入	外部回路(©) ヘルプ(<u>H</u>) 力しました	
Down Load File:	Output Data	a File:

78Kxx シミュレータ GUI ウインドウのメニューの [外部部品(E)] [入出力パネル(P)...]を選択してください。





🔜 入出力パネル	<u> </u>
ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B) カスタム(C) 図形(D) オブション(D)	ヘルプ(円)
□ ▦ 懣 斷 ※ 8. Ձ Ø 神 fff ● ず 彡 M A/□○○△	

<u>内部割り込みボタン</u>の設定をします。

まず,入出力パネル・ウインドウの内部割り込みボタン 📂 ,またはメニューの[接続(P)] [内部割り込みボタン [い...]を選択してください。

内部割り込みボタン設定ダイアログが開きますので,割り込み名の"#1 "を 78K0 は" INTTM00 ", 78K0S は"INTWT", 78K4 は"INTC00"に変更してください。



OK ボタンを押してください。





	🧱 入出力パ	パネル							_ 🗆 🗵
	ファイル(E)	モード(<u>M</u>)	編集(E)	接続(<u>P</u>)	ビットマップ(<u>B</u>)	カスタム(©)	図形(D)	オプション(0)	ヘルプ(円)
			8. 🔍	s inder	RESET 🛞 🔛	≩∭ 4	A / C		2
C	INTTMOO								
1									

入出力パネル上に内部割り込みボタンが表示されます。

内部割り込みボタンとは?

SM78Kxxのディバグ機能の1つです。タイマなどのCPU周辺機能が発生させる内部割り込みを, ユーザが任意にボタンを押すことにより,疑似的に発生させることができます。 内部割り込みが発生する条件を整えるのが難しい場合でも,この機能を使用すると,ディバグ時に 容易に割り込みを発生させて割り込み処理の動作確認を行うことができます。 <u>7 セグメント LED</u>端子の設定を行います。 7 セグメント LED 端子に入出力ポート P1, P3 の各ビットを接続します。 まず,入出力パネル・ウインドウの 7 セグメント LED 端子設定ボタン **●**,またはメニューの [接続(<u>P</u>)] [7 セグメント LED(<u>S</u>)…]を選択してください。

7 セグメント LED 端子設定ダイアログが開きますので,7 セグメント LED を下図のように設定してください。

■入出力パネル	<u>■■×</u>
ファイル(E) モード(M) (手)(E) 目	鉄値 ビットマップ® カスタム© 図形① オプション© ヘルプ(H)
(回) (国) (図) (図) (図) (図) (図) (図) (図) (図) (図) (図	ジ 瞬 1995 100 100 100 100 100 100 100 100 100 10
7セグメント LED 端子を	アセグメント LED 端子設定
次のように設定します。	ダイアログが開きます。 アセグメント LED端子設定 アビグメント LED端子設定 原宿告1 OK 附宿号2 読み込み 附宿号3 (保存) 附宿号5 キャンセル 附宿号6 ヘルブ 竹宿号7 〇日 〇日 所宿号8 前頁
P56 P54 P53 P51 P50 P50 P50 P57 セグメントH/L でH でH ・日	新信号1 P40 OK 桁信号2 P41 読み込み 新信号3 (保存 桁信号4 クリア 桁信号5 キャンセル 桁信号6 ヘルブ 桁信号7 前頁 次頁

設定が終了したら, OK ボタンを押してください。



入出力パネル上に仮想 LED が表示されます。

🧱 入出力パネル						
ファイル(<u>F</u>) モード(M)編集(E) 接続(E) ビットマップ(<u>B</u>)	カスタム(©)	図形(D)	オプション(Q)	ヘルプ(円)
□퐯蕊龞	× 8. Q Ø:	æ RESET 🛞 🛒	事版 4	$ / \Box$		2
INTTMOO						

これで,ボタンと7セグメント LED の設定が終了しました。

7 セグメント LED とは? 疑似的なターゲット・システムを構築するために SM78Kxx が用意している外部部品の1つです。 主に数値を表示するために、入出力ポートと接続して使用します。操作に使用する信号には、 各桁共通の"セグメント信号"と各桁独立の"桁信号"があります。"セグメント信号"端子に 表示する値, "桁信号"端子に表示する桁位置の順に入出力ポートから出力します。 詳細については, SM78K シリーズ システム・シミュレータ Ver.2.52 ユーザーズ・マニュアル 操作 編(U16768J)を参照してください。

プログラムの実行(1)

<u>カウンタ・プログラム</u>を実行します。

SM78Kxx のリスタート・ボタン **I** , またはメニューの [実行(<u>R</u>)] [リスタート(<u>R</u>)]を選択してください。

この操作により,エミュレーション CPU をリセットしてから,プログラム実行を行います。



プログラムの実行中は,ステータス表示エリアが赤く変化します。入出力パネルには"00"が表示され, INTTMO0 ボタンによる入力待ち状態になっています。78K0Sは INTWT ボタン,78K4は INTC00 ボタンとなります。

ここで,上記の動作が起きない場合は,次の操作を行ってください。 LED が点灯しない場合 再度, リスタートしてください。 リスタートしても動作が改善されない場合は,7セグメントLED端子の設定をやり直してください。 LED が "00"以外の表示を行った場合 7 セグメント LED 端子の設定をやり直してください。

INTTM00 ボタンを数回押してください。

INTTM00 ボタンを押すたびに,カウンタが1つずつカウント・アップする仕様です。

SMK032 : counter.prj - [Source (counter.c)]
□ ファイル(E) 編集(E) 表示(V) オブション(Q) 実行(R) イヘント(N) ブラウズ(E) ジャンプ(J) ウィンドウ(W) ヘルプ(H) <u>- #</u>
Search K >> Watch Quick Refresh Close
31 Yoid main() Yo
0.0 0000000000000000000000000000000000
* > <u>96</u> PM5 = 0x00; /*P5 (P50-P57)は出力 */ * 97 PM4 = 0x00; /*P4 (P40-P47)は出力 */
38 93 * INTTM00 割込みレベルの設定と割込みの有効化 */ 100 TMPR00 = 0:/* INTTM00割込みレベルを寄傷先順位レベルにする */
* 101 TMMKOO = 0;/* INTTMOO割込みの受付を許可する */
103 /* カウンタの初期化 */ * 104 count1 = 0:/* LED に表示する数値の初期化(1の位) */ count1 = 0:/* LED に表示する数値の初期化(100位) */
108 108 107 107 /* 初期値の表示 */
•
この 「「「「「「「」」」 「「」 「「」 「」 「」 「」 「 「 」 「 」
Search K >> Watch Quick Refresh Close
91 void main(){ /************************************
93) * 2D期代L処理 94 ************************************
95 /* 7セグメント LED に出力するポートのモードを設定 */ * > <mark>96 PM5 = 0×00; /*P5 (P50-P57)は出力 */</mark>
* 97 PM4 = 0×00; /*P4 (P40-P47)は出力 */ 98
93 /* INITMOO 割込みしベルの設定と割込みの有効化 */ * 100 TMPROO = 0:/* INITMOO割込みレベルを高優先順位レベルにする */
* 101 MMKUU = 0;/* INTIMUU割込みの)文目を計可する */ 102 101 100 /# 10:00000000000000000000000000000000000
* 103 7* カウンタのがあれて */ * 104 count1 = 0; /* LED に表示する数値の初期化(1の位) */
Ready ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B) カスタム(D) 図形(D) オブション(D) ヘルプ(H)

INTTM00 ボタンを押すと, LED の表示は 10 の位のみカウント・アップし, 1 の位は変化しません。 したがって, 仕様に合った動作をしていないことがわかります。

ここで,上記の動作が起きない場合は,次の操作を行ってください。
 INTTMOO ボタンを押しても,何もアクションが起きない場合
 内部割り込みボタンの設定をやり直してください。
 LED が上記の動作(10の位のみカウント・アップ)でない場合
 7セグメント LED 端子の設定をやり直してください。

カウント・アップが正常に処理されていないので , ディバグを行いましょう。

プログラムを停止します。

SM78Kxxの停止ボタン II , またはメニューの [実行(R)] [ストップ(S)]を選択してください。



プログラムを停止すると,ステータス表示エリアの色は元に戻ります。

ディバグ

count1 には LED の 1 の位の値が, count10 には LED の 10 の位の値が入っています。

(二) 詳細については , 「<u>カウンタ・プログラムの仕様</u>」をご覧ください。

まず, LED 表示ルーチン (putLED()関数)を実行したときに count1, count10 にどのような値が設定されてい るのか調べてみましょう。

78K0 と 78K4 の場合は 68 行目に, 78K0S の場合は 69 行目にブレークポイントを設定します。

ブレークポイントの設定できる行には"*"が表示されています。

78K0 と 78K4 の場合は 68 行目の"*"を, 78K0S の場合は 69 行目をクリックしてください。



68 行目 (78K0 と 78K4) または 69 行目 (78K0S) にブレークポイントが設定されました。

それでは , プログラムを実行します。

SM78Kxx のリスタート・ボタン **・** , またはメニューの [実行(<u>R</u>)] [リスタート(<u>R</u>)] を選択してください。

Security Security Watch Guick Refresh Close 1 57 24 = 1; ** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 50 /** 1** 1** 51 /** 1** 1** 52 /** 1** 1** 53 /** 1** 1** 54 24 = 0: 1** 1** 55 1** 1** 1** 56 2** 1** 1** 57 ** 1** 1** 58 2** 1** 1** 57 ** *** 1** 58 2** *** 1** 57 *** **** 1*** 58 *** ***** 1************************************
0 000 700 70
プログラムを実行します。
<mark>■SMK032 : counter.prj - [Source (counter.c)] </mark>
II ■ ▶ ■ ▶ ■ ▶ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
** 100 TMPR00 = 0; /* INTIMO動込みレベルを高優先順位レベルにする */ 101 ** 101 TMMK00 = 0; /* INTIMO動込みの受付を許可する */ 102 102 /* カウンタの初期化 */ ** ** 104 ** 105 ** 105 ** 105 ** 105 ** 105 ** 105 ** 106 107 /* 初期値の表示 */
100 100 100 100 110 /************************************

プログラムを実行するとすぐに,ブレークポイントでプログラムが停止します。 プログラムが停止した行には ">"が表示され,黄色に反転します。 変数の値を見るには,ウォッチ・ウインドウを使用します。

ウォッチ・ウインドウを開き,このときの count1, count10 それぞれの値を確認しましょう。 56 行目の " count1 " をダブルクリックし反転表示させてから, Watch ボタンを押してください。

##SMK032 : counter.prj - [Source (counter.c)]	
Search.	
52	
% 55 P4 = 0; % 56 P5 = bolicount1] % 57 P4 = 1;	
◎ 1 NOP(); ◎ 61 NOP(); 62 (# 745 / 2016 LED = 1の方言であた後(10の(点) # /	
 84 P4 = 0; 85 P5 = box[count10]; 8 e5 P5 = box[count10]; 	
m 00 r4-2; B 68 ;	
Ready counter c#96 main 008A HALT	
□ ■ 選 ■ ※ 8. Q Q >> == == == == == == == == == == == ==	
	•
In SMK032 : counter.prj	
ファイル(2) 編集(2) 表示(2) オブヘン(2) 実行(3) イヘント(2) フラウズ(3) シャンブ(2) ウィンドウ(4) ヘルブ(4) Ⅲ▶▶▼▼灯(7) ▲(*) 図(20) (11) (11) (11) (11) (11) (11) (11) (1	
Add Delete Up Down Refresh Close Pulck Pulcks Close count1 0000H 0x77,0x24,0x8b,0x86,0x8c,0x8d,0x1 0x8b,0x8d,0x8c,0x8d,0x1	
D にバターンを転送(1の位)*/	
□ [1/3 - 2/2載2 (100)世) */];	

Ready counter c#96 main NGA HALT RUN/	
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	

ウォッチ・ウインドウはシンボル名表示エリアとデータ値表示 / 設定エリアで構成されています。

 詳細については, SM78K シリーズ システム・シミュレータ Ver.2.52 ユーザーズ・マニュアル 操作編 (U16768J)を参照してください。 同様に, ソース・テキスト・ウインドウ (Source ウインドウ)の 65 行目の "count10"をダブルクリックし, 反転表示させてから, Watch ボタンを押してください。

ウォッチ・ウインドウに count10 が追加されます。

	↓ ブラウズ(B) ジャンブ(J) ウィンドウ(W) ヘルブ(H)
	3 Q⊿∰& ≥ 3 5 1 1 2 7 7 8 8
Source (counter.c)	
Search	lefresh Close
NOP();	
● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	ンを転送(10の位)*/
<pre>* 65 P5 = box[count10]; * 86 P4 = 2;</pre>	
67] return;	
P Watch	
Add Delete Up Down Refre	Si Close
count10 0000H count1 0000H	
<u>.</u>	
Ready counter.c#96	main 008A HALT RUN/
	└└────── └ └ count1,count10 ともに 0 を表示します。

ウォッチ・ウインドウを見ると, count1, count10 ともに初期値が0 であることが確認できます。 同様に,入出力パネルの LED も初期値00 を表示していることが確認できます。 ウォッチ・ウインドウのデータ値表示 / 設定エリアは 16 進表示ですが, SM78Kxx の[表示(<u>V</u>)]メニューに より変更することができます。

では, count1の値を10進表示に変更してみましょう。

ウォッチ・ウインドウの count1 をクリックして反転させたあとに, SM78Kxx のメニューの[表示(<u>V</u>)] [10 進(<u>D</u>)]を選択してください。

SMK032 : count	ter.prj				
ファイル(E) 編集(E	② 表示(⊻) わりょン(2)	実行(E) イヘシト(N)	ブラウズ(<u>B</u>) ジャンプ(<u>J</u>)	ウィンドウ(₩) ∿ルプ(⊞)	
	▶ 検索(S) CH 移動(V) CH	1+G 1+J	<u> 4 # B</u> <u>-</u>	式 🖾 🛅 📍 🖤	🕫 🔞
🔟 Source (counti	クイックウォッチ(Q) CI	(1+\))/			- 🗆 🛛
Search	<< ワォッチ登録(W)… 一 ウォッチ追加の	k Re	fresh Close		
* 61 62	ウォッチ削除(L) De シンボル変換(Y)	1			
63 * 64	2進(B)	ニバターン	を転送(10の位)*/		
* 65 * 66	Bi進(①) 105準(D)	-			
	10/2(1)				Ţ.
	1 文字列① ●自動(P)				
Watch	De /Yr/KE)	1 Refrec			
count10		DODOH			
	通合(<u>A</u>)	DOOOH			
	アップ(<u>P</u>)				
	3.72/10/				
		•	•		
		counter.c#96	main	008A HAL	.T RUN/ //



🚟 SMK032 : counter.prj	······································			_ 🗆 🗙
ファイル(E) 編集(E) 表示(V) オプション(0)	実行(R) イベント(N) フ	[、] ラウス [、] (<u>B</u>) ジャンフ [。] (<u>J</u>)	ሳብ≻ን™ ∿ዞን°	Y(Η)
			🐔 🛅 📍	V 🕫 🔞
🔟 Source (counteric)				
Search 🤇 🚿 🛛 Watch	Quick Refres	h Close		
80 NOP(); 81 NOP();				
62 63 /* 7セグメン	トIFD にパターンをi	転送(10の位)*/		
* 64 P4 = 0; 85 P5 = box[con	nt10]•			
* 66 P4 = 2;	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,			
				F
🔟 Watch				
Add Delete Up	Down Refresh	Close		
count10	H0000			
	\rightarrow			
Ready	counter.c#96	main	008A	HALT RUN/

同様に count10 の値も 10 進表示に変更してください。

次に, INTTM00 ボタンを押したときに, LED に情報が転送され, 正しく表示されているかどうかを確認します。

SM78Kxx のスタート・ボタン
, またはメニューの [実行(<u>R</u>)] [継続して実行(<u>G</u>)]を選択してく
ださい。

プログラム実行を停止行から継続して行うため, "リスタート"ではなく"スタート"を使用します。

ファイル(E) 編集(E)	表示(V) オフション(Q) 実行(B) イベント(N) フラウス*(B) ジャンフ*(J) ウイントウ(M) ヘルフ*(H)
п •(>) > >	INAN 280 DRA QARB BKS BI !V?
B Source (counter.	
Search <<	>> Watch Quick Refresh Close
59	
* 61	NOP();
62	/* 7セグメント LED にパターンを転送(10の位)*/
* 64	P4 = 0; P5 = box[count10];
* 66	P4 = 2;
اهالمالحقفا كم	
D Watch	
Add De	lete In Down Refrech Glose
count10	
count1	ОТ
•	
Ready	counter.c#68 putLED 013D E
20102	



〒SMK002:counterprj □ ファイル-印 編集印 表示(い オフウャン(い 実行(P) イヘット(い フラウァズ(D) ジャンブ(い ウィントウ(い ヘルブ(t))) □ ● ▶ エ 戸 河 ▲ ゴ 『 ◎ ■ 圓 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	
Cource (counterc) 」ロメ Secrit: >>>>>>>>>>>>>>>>>>>>>>>>>>>>	
Bill Wetch Image: Count 1 Add. Delete Count 1 0T Count 1 0T	
赤く変化します。	

INTTM00 ボタンを押したときの,LED 表示ルーチン(putLED()関数)の動作を見てみましょう。INTTM00 ボタンを押して,ウォッチ・ウインドウの表示を確認してください。



INTTM00 ボタンを押すと,ブレークポイントでプログラムが停止します。

ここで,カウント・アップが正常に動作していれば, count10に0, count1に1の値が設定されます。

しかし,ウォッチ・ウインドウを見ると, count10 が1, count1 が0となっており,変数に正しくデータが設 定されていないことがわかります。

一方,入出力パネルの LED は 10 となっており,ウォッチ・ウインドウの変数の値が LED に反映されています。

これにより, count10, count1の値を LED に表示する処理は,正しく行われていることが確認できます。

表示処理は正しいことが確認できましたので,次はカウント・アップ処理が正しく行われているかどうかを確認しましょう。

これから,LED 表示ルーチン(putLED()関数)とは別の箇所にあると考えられるエラーを探しますので, putLED()でプログラムが停止しないようにブレークを解除します。

■ SMK032:counter.prj ファイル(E) 編集(E) 表示(V) オフジョン(Q) 浮	実行(B) イベント(N) フラウス	((<u>B</u>) ジャンフ°(<u>。</u>	ມ ¢ብ/ኑኻ₩) ∿#	<mark>ー</mark> レフ*(H)		
	🔟 🖳 📲 🛄 .	4	2 3 8 2	¶ ▼ ₹	<u>ð</u>	
Source (counter.c)						
Search <u><>></u> Watch 62	Quick Refresh	Close	24		-	
83 /# 7セグメント 84 P4 = 0; 65 P5 = box[coun 66 P4 = 2; 87 return; 88	LED にパターンを転送 t10];	(10の位) *	1			
	*****	******	*****		*	
Watch Add Delete IIn	Down Befresh	Close		17		
count1						
		1	10100			- 🗆 🗵
Ready		putLED	013D		E //	
	回翻遊醫浴	8. 👷 🔗	a RESET 部以	き Mi A		
						
						Ŧ

" B "をクリックすると" * "に戻り,ブレークポイントが解除されます。

次に, INTTM00 ボタンを押したときのカウント・アップ処理の動作を確認します。

内部割り込みが発生したときに実行される関数(interrupt1())の中で, count1, count10の値がどのように変化するかを調べてみましょう。

144 行目にブレークポイントを設定します。

144 行目の" * "をクリックしてください。

SMK032 : counter.prj					_ 🗆 X	
ファイル(E) 編集(E) 表示(V) オブショ	ン(Q) 実行(R) イベント(N) フミ	iウス(<u>B</u>) ジャンプ(<u>J</u>)	ሳ⁄ントウ₩ ^	ルフ°(<u>H</u>)		
			- 🔏 🛅 🛅	፻ 🐨 🐨 🐼	J	
🔟 Source (counter.c)						
Search < >> W	itch Quick Refresh	Close				
141 ***********************************	**************************************	*****	********			
* 144 { 145 /******* 146 * カウご 147 ******* * 148 count1+・	************************************	********** ************/ ップ */			• •	
Add Delete Up	Down Refresh	Close				
count10	1T					
count1	OT					
•	► I	Þ				
Ready	counter.c#96	main	008A	HALT	RUN/ //	100
		蕭登圖☆	8. 🔍 🖉 i	a RESET 😤 🗾	事版	A / I
	II	ТТМОО				



SMK032 : counter.prj		
ファイル(E) 編集(E) 表示(W) オブション(Q) 実行(R) イベント(N) ブラウズ(B) ジャンプ(J) ウィンドウ(W) ヘルプ(H)		
Source (counter.c)		
Search. Watch Ouick. Perfresh Close 140 ************************************		
144 /************************************	× A ×	
Watch		
Add. Delete Up Down Refresh Close 1T count10 1T Count1 0T		
x		
Ready counter.c#68 putLED 013D	E	
		-
INTIMO 3		

144 行目にブレークポイントが設定されました。

リスタートします。

SM78Kxx のリスタート・ボタン ・ , またはメニューの [実行(<u>R</u>)] [リスタート(<u>R</u>)]を選択してくだ さい。







INTTM00 ボタンを押してください。



INTTM00 ボタンを押すと,144 行目のブレークポイントでプログラムが停止します。

ここからはソース・プログラムを1行ずつ実行し, count1, count10の値を確認してみましょう。

ステップ実行を行います。

SM78Kxx のステップイン・ボタン ▶ 「 , またはメニューの [実行(<u>R</u>)] [ステップイン(<u>T</u>)] を選択して3回選択してください。



148 行目をステップ実行後, ウォッチ・ウインドウの count1 の値を確認してください。 count1 が1になっているので,正しく動作していることが確認できます。 続けて,153 行目をステップ実行します。

SM78Kxx のステップイン・ボタン ▶ , またはメニューの [実行(<u>R</u>)] [ステップイン(<u>T</u>)]を選択して ください。

🚟 SMK032 : counter.prj	- 🗆 🗡	
ファイル(E) 編集(E) 表示(W) わかょン(Q) 実行(R) イベント(W) フラウズ(B) ジャンフ(U) ウィントや(W) ヘルフ(H)		
▥▶▶ਙ▶≫▲ਖ਼ ◙◙◙ ◙ॠॡ Qame book !!♥	🕫 😚	
👿 Source (counter.c)		
Search << >> Watch Quick Refresh Close		
B 144 /####################################	<u> </u>	
151 /* 10 位の桁上がりが必要かどうか? */ 152 /* 10 位の桁上がりが必要かどうか? */ 153 if(count!=10) /*+→ERROR*/ 154 (/* 10 位が10 なので、桁上がり処理を行う */	× (4) *	
Watch X Add Outs Op Count Otsee OT count1 10T 10T OT OT OT		
x		<u>- 0 ×</u>
Ready counter.c#155 interrupt1 017D	B //.	
□ 囲 ジ S 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	Å A / [
		<u></u>
		•

153 行目をステップ実行すると, count1 の値が1から10に変化しました。

153 行目は,桁上がり処理が必要かどうかを判断する条件式であり, count1の値の変化はないはずです。 これで,153 行目に問題があることがわかりました。

153 行目を確認すると, if 文の条件式で「count1 と 10 とを比較する式(count1==10)」を記述すべきところが,「count1 に 10 を代入する式(count1=10)」になっています。

エラー箇所が確認できましたので,ブレークを解除します。

" B "をクリックすると" * "に戻り, ブレークポイントが解除されます。

sMK032 : counter.prj				
ファイル(E) 編集(E) 表示(V) オブション(Q) 実	『行(E) イベント(N) ブラウズ(E) シ	/`ャンプ(』) ウィンドウ() ヘルプ(Ŀ	()	
	🗈 🕒 🖷 🤁 🞑 🛋 💼	N 🗄 🕷 🛅 🖠 🕅	V 🕫 🔞	
Source (counter.c)				
Search << >> Watch	Quick Refresh Clos	ie -		
141 ***********************************	************************************	*****************		
● 14/1 ************************************	****の********************************	***/	× 4 ×	
Wetch Add. Delete Up D count10 count1	lown Refresh Close 0T 10T			
	<u> </u>	<u>)</u>		×
Ready	counter.c#155 inter	rrupt1 017D	B //	
	□ ▦ 淡 醫 ※ 8. 9	2 2 10 10 10 10 10 10 10 10 10 10 10 10 10		1
				•
ソースの修正と実行形式の作成(2)

プログラムのエラー箇所を修正します。

SM78Kxxのメニューの[編集(E)] [ソースの修正(E)]を選択してください。





エディタが開きます。





153 行目の if 文の条件式 " count1=10 " を " count1==10 " に修正してください。

修正が終了したら, PM plus のビルド->ディバグ・ボタン 、 またはメニューの [ビルド(<u>B</u>)] [ビルド->ディバグ(<u>A</u>)] を選択してください。

PM plus のエディタ機能の場合 , ソースの修正内容はビルド時に自動的にセーブされます。





ビルドが終了すると, SM78Kxx は自動的に実行形式ファイルのダウンロードを行います。

プログラムの実行(2)

リスタートします。

SM78Kxx のリスタート・ボタン **・** , またはメニューの [実行(<u>R</u>)] [リスタート(<u>R</u>)]を選択してください。



カウント・アップ動作が正常に行われるかを確認します。

まず,1の位が正しくカウント・アップしているかを確認するためにボタンを数回押してください。

SMK032 : counter.prj ファイル(F) 編集(F) 表示(V) わやみ)(O) 見	第元(R) イベット(N) フラウス?(B) ッションフッ(L) ウィットウ (W) ヘルフッ(H)	
	D	** 🔯
🔲 Source (counter.c)		
Search 🤇 🏷 Watch	Quick Refresh Close	
95 /*7セグメント * > 96 PM5 = 0×00; /	LED に出力するボートのモードを設定 */ *P5 (P50-P57)は出力 */	
97 PM4 = U×UU; , 98	*P4 (P4U-P4/)は出力 */	
99 /* INTIMUU 割) * 100 TMPR00 = 0;/* 101 TMPR00 = 0;/*	ムみの割込みレベルの設定と割込みの有効化 */ INTTMOO割込みレベルを高優先順位レベルにする */	
* 101 「MMK00 = 0;/* 102 109 (# 古中にものが	INTIMUU割込みの文字で計画する */	
* 104 count1 = 0;//	#110 */ LED に表示する数値の初期化(1の位) */	
		*
🖲 Watch		
Add Delete Up [iown Refresh Close	
count1	3T	
	ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B) カスタム オプション(O) ヘルプ(H)	(C) (2)H3(D)
	INTTMOO	-

1の位は,正常にカウント・アップ処理しています。

次に,桁上がり処理が正しく行われているかを確認するために, INTTM00 ボタンを 10 回以上押してください。



桁上がり処理は正常に行われています。

したがって,カウント・アップ動作は正常に行われていることが確認できました。

最後に,桁溢れ処理が正しく行われているかを確認します。

INTTM00 ボタンを 100 回以上押して確認することもできますが,ここでは,より簡単に確認する方法を紹介します。

まず,SM78Kxxの停止ボタン III ,またはメニューの[実行(<u>R</u>)] [ストップ(<u>S</u>)]を選択し,プログラム を停止してください。

■SMK082 : counterprj フェ(Iv(E) 編集(E) 表示(M) わりねン(Q) 実行(B) イベント(M) フラウス(B) ジャンフ(Q) ウイントウ(M) へいつ(H)	
III FIMAL BER DER QAAD BNE BNE I VVVO	
B Source (counter.c)	
Search < >> Watch Quick Refresh Close	
* 37 PM4 = 0×00; /*F4 (P40-P47)は出力 */	
98 99 /* INTTMOO 割込みの割込みレベルの設定と割込みの有効化 */ 100 TMPROO = 0;/* INTTMOO割込みレベルを高優先順位レベルにする */ 101 TMMKOO = 0;/* INTTMOO割込みの受付を許可する */	
103 /* カウンタの初期化 */ ▼ 104 count1 = 0; /* LED に表示する数値の初期化(100位) */ ▲ 105 count10 = 0; /* LED に表示する数値の初期化(100位) */ ▼	
Watch	
Add Delete Up Down Refresh Close	
count1 IT ST	
	×
ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B) カスタム(Q) 図形(D)	
	-
INTIMOO 3 C S	•







次に,ウォッチ・ウインドウのデータ値表示/設定エリアの値を変更します。

ウォッチ・ウインドウのデータ値表示 / 設定エリアにカーソルを合わせ, count10 の値を"9"に, count1 の値を"7"に変更します。







その状態で,リターン・キーを押してください。

count10, count1の値が,赤字から黒字に変わります。

SMK032 : counter.prj	- 🗆 🗙	
ファイル(E) 編集(E) 表示(W) オブション(Q) 実行(R) イベント(N) ブラウズ(B) ジャンプ(J) ウィントウ(W) ヘルプ(H)		
III>>III>>III=>	1 100	
🔟 Source (counteric)	-OX	
Search << >> Watch Quick Refresh Close		
neci();		
124// ###################################		
126 * カウントアップ処理 127 * (INTIMODで呼び出される割込み開数)	*	
	\$	
	Þ	
🗉 Watch		
Add Delete Up Down Refresh Close		
count10 9T		
		- 🗆 🗵
Ready counter.c#119 main 015A	B //	
<u>□ </u>	A / \Box	\circ
		-

これで, count10 に "9", count1 に "7"の値が設定されました。

それでは,この状態から継続して実行してみましょう。

SM78Kxx のスタート・ボタン
, またはメニューの[実行(<u>R</u>)] [継続して実行(<u>G</u>)]を選択してください。

The own counter.prj		
ファイル(E) 編集(E) 表示(V) オブション(Q)	実行(B) イベント(N) フラウス(B) ジャンフペリ ウィントウ(W) ヘルフペH)	
	I III I I I I I I I I I I I I I I I I	e ^y (0)
🔟 Source (counter.c)		
Search << >> Watch	Quick Refresh Close	
HALI();		
* 121 }		
123		
124 /************************************	***************************************	
126 * カウントアップ処理		.
127 * (INTIMUU C呼び出る 128 *	される割込み関数)	÷
	******	Ē
NI Watak		10 m
P1 Watch		
Add Delete Up	Down Refresh Close	
Add Delete Up	Down Refresh Close 9T	
Add_ Delete Up count10 count1	Down Refresh Olose 9T 7T	
Add. Delete Up count1	Down Refresh Close	
Add. Delete Up	Down Refresh Close 9T 7T	
Add. Delete Up	Down Refresh Close	_10×
AddDeleteUp count10 count1	Down Refresh Close	- I ×
Ad. Delete Up count10 count1	Down Refresh Close	
Add. Delete Up count10 count1 Ready	Down Refresh Close 9T 7T 7T counter.c#119 main 015A ■■◎■◎ ② ③ ☞ #### ⑧ ◎ ◎	B / 0 / 1
AddDeleteUp count10 count1	Down Refresh Close	B /
Add. Delete Up count10 count1	Down Refresh Olose 9T 7T 7T pr pr pr pr pr pr pr pr pr pr pr pr pr	
Add. Delete Up count10 count1 	Down Refresh Close 9T 7T 7T counter.c#119 main 015A imain 015A imain 015A	



これで, INTTMOO ボタンを 98 回押したときと同じ状態になりました。 それでは, さらに INTTMOO ボタンを押してください。



桁溢れ処理は,正常に行われています。 これで,プログラムのすべての確認が終了しました。 プログラムを停止します。 SM78Kxxの停止ボタン <u>III</u>,またはメニューの [実行(<u>R</u>)] [ストップ(<u>S</u>)]を選択してください。





SM78Kxx を終了します。

SM78Kxx メニューの [ファイル(<u>F</u>)] [終了(<u>X</u>)]を選択してください。

SMK032 : co	unter.prj								_ 🗆 ×
ァイル(E) 編集	€(E) 表示(⊻)	わ ⁹ ション(<u>0</u>)	実行(R)	イベント(N)	ブラウス(日)	ジャンプ(J)	ታለን⊦ን∰	^/レフ°(<u>H</u>)	
開く(Q) 名前を付けて(閉じる(<u>C</u>)	(保存(<u>A</u>)					Ctrl+O		I ! V	
ダウンロード(<u>D</u> アップロード(<u>U</u>))								
プロジェクト(<u>J</u>)						•			
デバッガリセット	·(<u>R</u>)								
終了🛛							******		
<u>1</u> c:¥78K_sam <u>2</u> D:¥Mydocu <u>3</u> D:¥Mydocu	nple¥Chapter3¥c ment¥ファーストス ment¥ファーストス	counter.lmf 、テップガイド¥7 、テップガイド¥7	8K_sample¥ 8K_sample¥	∉Chapter2¥\ ∉Chapter2¥c	/RAM.prj :haracter.lmf		*****		* *
	•								•
Watch								- 🗆 🗙	
Add	Delete	Up	Down	Refresh	i Close				
count10					OT OT				
					01				





この章で設定した入出力パネルの状態などをプロジェクト・ファイルに保存する場合は はい(<u>Y</u>) を,保存 しない場合は いいえ(<u>N</u>) を押してください。

「環境」とは,外部部品,ウインドウの状態などのことです。

→ 詳細については, SM78K シリーズ システム・シミュレータ Ver.2.52 ユーザーズ・マニュアル 操作編 (U16768J)を参照してください。 PM plus を終了します。

PM plus のメニューの [ファイル(<u>F</u>)] [PM plus の終了(<u>X</u>)]を選択してください。

🚟 PM plus - counter.prw [OutPut]	
ファイル(E) 編集(E) 検索(D) 階層	夏(L) 表示(V) プロジェウト(P) ビルド(B) ツール(T) ウィンドウ(W) ヘルプ(H)
新規作成№ C 開く@ C 挿入読み込み@ 閉じる@	ttrl+N ttrl+O ug Build
ワーカスヘヘースの新規作成(E) ワーカスペースを聞く(W) ワーカスペースを閉く(M) ワーカスペースを閉じる(<u>K</u>)	<pre>PoutPutX - * - 78K/0 Series Linker E3.60a [3 Apr 2003]* - Copyright(C) NEC Electronics Corporation 1990,2003* +</pre>
上書き保存(<u>S</u>) C 名前を付けて保存(<u>A</u>)	tri+S - Target chip : uPD780034+ - Device file : V1.01+
ソース・ファイル・名の変更(E) すべてのソース・ファイルを保存(LL)	 Link complete, 0 error(s) and 0 warning(s) found.* C:¥NECTools32¥BIN¥oc78K0.exe -fcounter.poc*
印刷プレビュー(\/) 印刷(P) C	trl+P - 78K/O Series Object Converter E3.60a [3 Apr 2003]↓ trl+P - Copyright(C) NEC Electronics Corporation 1990,2003↓
<u>1</u> counter.c	- Target chip : uPD780034+
C:¥NECTools32¥¥counter.prw C:¥NECTools32¥¥VRAM.prw	- Object Conversion Complete, D error(s) and D warning(s)
PM plus の終了 🕗	Build Total error(s) : 0 Total warning(s) : 0 🕴
	・ C:¥NECTools32¥78K0_sample¥Chapter3¥counter.Imf をディバゥガに友*ウンロマ ・
PM plusを終了し、変更したファイルの保存	字を確認します L:1 X:1 Y:31 //

PM plus のプロジェクト情報は逐次保存されているため,保存の確認はされません。

第4章 プログラミング編

この章では,78K シリーズ CPU 用の機種依存処理を C 言語で記述するための方法について,サンプル・プロ グラムを使用しながら解説します。

ここでは,サンプル・プログラムとして,スロット・プログラムを使用します。

スロット・プログラムでは,拡張C言語仕様の「レジスタ名を用いた特殊機能レジスタへのアクセス」,

「割り込み / 例外関数の記述」,および「部分的な割り込み禁止の設定」の機能を使用しています。 この章の全体の流れを,次に示します。



スロット・プログラムの仕様

この章の操作を行う前に、スロット・プログラムの概要を理解していただく必要があります。

外部仕様は次のとおりです。

外部仕様
● ある装置に,5桁の7セグメント LED,3つの四角いボタン,リセット・ボタンがあります。
8. 1. 8. 1. 8. Out₂y►
STOP(L) STOP(C) STOP(R)
 ● LED は 1 つおきに使用し,0から9までの数字を連続で表示し,9を表示した次は0に戻ります。 ● 四角いボタンの表面には,それぞれ "STOP(L)", "STOP(C)", "STOP(R)"と書かれており,このボタン
を押すと連続して数字を表示している LED の動きが止まります(ある数字を表示したままになります)。
➤ STOP(L) ボタンを押すと,LED の左端の桁の数字が停止します。
➢ STOP(C) ボタンを押すと,LED の中央の桁の数字が停止します。
➢ STOP(R) ボタンを押すと,LED の右端の桁の数字が停止します。
● リセット・ボタンを押すと初期状態に戻り,止まっていた LED が動き出します。

基本仕様は次のとおりです。

基本仕様 ● スロット表示 ▶ 0~9までの数値をカウント・アップ表示します。 ▶ 9を表示したら,0に戻ります。 ● ボタンの動作 ➢ STOP(L) ボタンを押すと, INTPO 割り込みが発生します。 ▶ STOP(C) ボタンを押すと, INTP1 割り込みが発生します。 ▶ STOP(R) ボタンを押すと, INTP2 割り込みが発生します。 割り込み関数の設定 ▶ INTPO 割り込みが発生したら, 関数 stop btn Left を実行します。 ▶ INTP1 割り込みが発生したら,関数 stop_btn_Center を実行します。 INTP2 割り込みが発生したら,関数 stop_btn_Right を実行します。 割り込み関数の処理 関数 stop_btn_Left を実行すると,左端の桁の表示を固定します。 関数 stop_btn_Center を実行すると,中央の桁の表示を固定します。 関数 stop btn Right を実行すると,右端の桁の表示を固定します。 ● ターゲット CPU 環境の初期化 ▶ 使用するポートの初期化を行います。 > 割り込みの使用を可能にし,割り込みの優先順位の設定を行います。

内部仕様は次のとおりです。

内部仕様

● LED を点灯する位置と表示する数字は,次の変数で保持します。

変数	内容
unsigned char place;	表示(点灯)位置の指定
unsigned char num_data[10];	表示用数字データ

● プログラムは,メイン関数,ターゲット CPU 環境の初期化部,スロットの表示部,および割り込み関数 で構成します。

ファイル名	処理関数	処理内容
slot.c	メイン関数	・ターゲット CPU 環境の初期化関数
	void main();	(init_target())の呼び出し
		・スロット表示関数(slot())の呼び出し
	ターゲット CPU 環境の初期化	・ポート,割り込みレベルなど,ターゲット
	void init_target(void);	CPU 環境の初期化
		・割り込みを受け取り可能状態に設定
	スロット表示	・表示用数字(0~9)をループさせ LED に表示
	void slot(void);	・ループ中での割り込みを許可
interrupt_func.h	STOP(L)ボタン処理	・INTP0 割り込みで起動
(関数宣言)	interrupt void stp_btn_Left(void);	・左端の桁の表示を現在の数値のまま固定
	STOP(C)ボタン処理	・INTP1 割り込みで起動
interrupt_func.c	interrupt void stp_btn_Center(void);	・中央の桁の表示を現在の数値のまま固定
(関数定義)	STOP(R)ボタン処理	・INTP2 割り込みで起動
	interrupt void stp_btn_Right(void);	・右端の桁の表示を現在の数値のまま固定

● 使用する割り込みは次の3つです。

- > INTP0
- > INTP1
- ➢ INTP2
- LED の制御と割り込みの入力には,入出力ポートとして 78K0 は P0, P5, P4, 78K0S は P2, P1, P4, 78K4 は P2, P5, P4 を使用します。

78K0 78K0S 78K4

\succ	P0	P2	P2	(割り込みの入力用)
---------	----	----	----	------------

- ▶ P5 P1 P5 (7セグメント LED 点灯用)
- P4 P4 P4 (7セグメント LED 桁指定用)

スロット・プログラムの動作確認

スロット・プログラムの動作を確認するために,まず,PM plus を起動します。 Windows スタート・メニューの[プログラム(P)] [NEC Tools32] [PM plus]を選択してください。





📆 PM plus - No Workspace [ProjectWindow]		- D ×
_ ファイル(E) 編集(E) 検索(D) 階層(L) 表示(V) プロジェクト(P) ビルト	(18) ツール(17) ウィントウ(12) ヘルフ*(11)	
] 💁 🔹 🗋 🚅 見 🎒 🗟, 🕺 🖿 💼 🎇 assert	▼ + + + - 12	?
	■ ▲ 柴 キ 本 器	
ProjectWindow		
Files Memo		<u> </u>
		_
ヘルフ [®] を表示するには [F1] を押してください		X: Y: //

ワークスペース・ファイルの読み込み

この章では,あらかじめ作成されているワークスペース・ファイルを使用します。

PM plus のメニューの [ファイル(<u>F</u>)] [ワークスペースを開く(<u>W)</u>…]を選択し, "slot.prw"を指定して ください。

□ 環境を設定していない方は「<u>サンプル環境</u>」をご覧ください。







プロジェクトの実行形式を作成します。

PM plus のビルド・ボタン , またはメニューの [ビルド(<u>B</u>)] [ビルド(<u>B</u>)] を選択してください。



ビルド処理を正常に終了し,実行形式が作成されました。

システム・シミュレータ (SM78Kxx)の起動

SM78Kxx を起動します。

PM plus のディバグ・ボタン 🧱 ,またはメニューの [ビルド(<u>B</u>)] [ディバグ(<u>D</u>)]を選択してください。





SMK032 : slot.prj								
ファイル(E) 編集(E) 計	表示──	オフション(型)	実行(<u>R</u>)	イヘジト(N) こ	ドラウズ(目) ジャンフ。	ሠ ሳ心Ւኻ₩	^/レフ°(<u>H</u>)	
 ▶ ► ▼ ►) ▲	•			2 🖪 💼 🕓	2 🕺 🐻 🛅	1 🕈 🔍 🕫 🚳	
🔟 Source (slot.c)								_ 🗆 ×
Search <<	>>>	Watch	Quic	k Refre	sh Close			10000
* > 57 /* 58 ir	: ターケ <mark>iit_tars</mark>	ットCPU塚 <mark>et();</mark>	境の利期	1E */				<u> </u>
59 60 /*	表示位	置指定変換	数の初期化	*/				
* 61 pl 62	ace = 0	×15; /*	10101 :	LEDを一つお	きに3つ点灯する	. */		
* 63 /* 64 P4	:7セグ> + = 0×00	シトLED ;	消灯 */					-
* 65 P5	: = 0×00 4 = 0×FF	;						
00 13								
67 68 /*	、スロッ	ト(数字点	(灯)の処理	2 */				
67 68 /* * 69 sl	・スロッ ot();	ト(数字点	(灯)の処理	£ */				
* 67 68 /+ 8 69 sl 70 * 71 } /+ 72	* スロッ ot(); * main *	ト(数字点 /	(灯)の処理	£ */				
* 67 * 69 * 70 * 71 72 73 74 /****	8 スロッ lot(); 3 main *	ト(数字点 / ********	(灯)の処理	*******	*****	*****	****	
* 67 68 /# * 68 sl 70 * 71 } /# 72 } 73 74 /*** 75 * 7 76 *	* スロッ lot(); * main * ******** ポート, 1	ト(数字点 / ******** 割り込みし	灯)の処理 ********** レベル等の	【 */ ☆*********** ゆターゲットC	************************************	**************************************	****	× 4
* 67 * 68 * 70 * 71 * 71 * 71 * 71 * 71 * 70 * 71 * 70 * 71 * 70 * 71 * 70 * 71 * 70 * 70	* スロッ lot(); * main * ******** ポート,]	ト(数字点 / ********* 割り込みし 	灯)の処野 *********** レベル等の 1	』 */ ☞********** ッターゲットC	*************** PU環境の补刀期化ド	************* 周鼓	****	× 4×
* 67 / / * 68 / / * 70 * 71 * 72 74 / ##* 75 * 7 76 * * 76 * * 76 * *	* スロッ lot(); * main * ********* ポート, 1	ト(数字点 / ********** 割り込みし :-:+ +	(灯)の処野 ********* レベル等の * slo	!*/ ************** ターゲットC tc#58	************************************	*************************************	*****	× 4: *
eedy ■ 2420 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	* スロッ lot(); * main * ******** ポート, 1	ト(数字点 / ***********************************	(灯)の処当 ********** レベル等の slo >	【 */ ***********************************	************************************	*************************************	***** /////	× * * *
eady	* スロッ lot(); * main * ******** ボート, 1 増**タ・ (の) 図用	ト(数字点 / ********** 割り込み! ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	(灯)の処理 *********** レベル等の slo ション(0)	! */ ***********************************	************************************	************************************	***** ////////////////////////////////	×
	* スロッ oot(); * main * ********* ポート, 1 間**タ・ 」 (の) 図讯	ト (数字点 / ********** 割り込み ('- '- ' ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	(灯)の処理 (*********** レベル等の ・ Slo > ション(2)	! */ :***********************************	************************************	************************************	******** /////////////////////////////	×
	* スロッ ot(); * main * ******** ポート, 1 **** (の) 図用 を .	ト (数字点 / *********** 割り込み ((灯)の処理 (*********** レベル等の ション(2) (RESET 震	! */ ***********************************	************************************	************************************	******* 7(H)	* * *
	* スロッ lot(); * main * ******** ポート, 1 時**ク・ (の) 図用 を を、 の	ト (数字点 / *********** 割り込み 目 接続(P) (Q) オプ	(灯)の処理 (######### レベル等の ション(①) (#ESET (変)	! */ **************** な考ターゲットC たま58 11 〒77(ル(E) 〒03(こ〇)	**************** PU環境のや対明化 main たまりたからり 外部部品(0) タ を入力しまし	************************************	****** ///////////////////////////////	* * ×
	* スロッ lot(); * main * ******** ポート, 1 項**ク・ (の) 図研 (の) 図研	ト(数字点 / ********** りり込み(- 14 - 1- 接続(P) (Q) オブ (Q) オブ (Q) オブ	(灯)の処理 (************************************	と */ ***********************************	**************** PU環域の?)加病化 main たこりテン(の) 外部部品(2) タ を入力しまし	****************************** 現設 」008A 部回路② ヘル た	***** ////////////////////////////////	× 44
	* Z□ッ lot(); * main * * * ******** * * , 1 * * , 1 * * * , 1 * * * * * * * * * * * * * * * * * * *	ト(数字点 / ************ 割り込み () () () () () () () () () ((灯)の処明 *********** レベル等の ・・・ 家回 ションの 第1日 ※ リセット	te#58	**************** PU環境の約30時(上) 「main 外部部品(2) か を入力しまし	###################################	****** ?(д) ut Data File•	× *

プログラムの実行

<u>スロット・プログラム</u>を実行します。

SM78Kxx のリスタート・ボタン **I**▶ ,またはメニューの [実行(<u>R</u>)] [リスタート(<u>R</u>)]を選択してくだ

さい。

SMK032 : slot.prj					- 🗆 ×
ファイル(E) 編集(E) 表示(V) わ	ション(0) 実行(12) イベント(12) ブ	ラウス(国) ジャンフ(山)	• ሳለን⊦ነት 🕑 🗠 ∿ዞ	7°(<u>H</u>)	
			× 🔊 💼 🧐	! ▼ ₹ 8	
🔟 Source (slot.c)					
Search << >>>	Watch Quick Refres	h Close			
<u>57</u> /* ターゲッ	・CPU環境の初期化 */				
* > <mark> 58</mark> init_target 59	.);				
60 /* 表示位置打	8定変数の初期化 */		872		
61 place = 0x1	;; /* 10101 : LEDを一つおる	きに8つ点灯する.	*/		
63 /* 7セグメン	トLED 消灯 */				
64 P4 = 0×00;	1 11.0.1				
65 P5 = 0×00;					
66 P4 = 0×FF;					
	我实上你) 小加速 。/				
60 /* 人口ツF\ * 69 slot()・	数子点灯100処理 #/				
70					
* 71 } /* main */					
72					
73					
		****************	************	****	•
/5 * 不一下,割5	ルンタレベル寺のターケットい	'0球現の利期1日関野	<i>i</i> X		\$
	1 11				¥
Ready	alot c#58	main	0084		-



プログラムを実行します。



プログラムの実行中は,ステータス表示エリアが赤く変化します。

プログラムの実行中,入出力パネルの LED の各桁は,数字(0~9)をカウント・アップ表示する動作を繰り 返しています。



では,この実行中のスロット・プログラムを操作してみましょう。

入出力パネルの各ボタンを操作して,LED表示の変化を確認してください。

🔜 入出力パネル	
ファイル(E) モード(M) 編集(E) 接続(E))
ビットマップ(B) カスタム(C) 図形(D) オコ	1ション(<u>O</u>)
🔲 퐯 澈 🎬 🖄 🕄 🔛 🔊 🏛	₩ RESET 🛞 🕻
김홍금홍물 🤇	リセット
	I
STOP(L) STOP(C) STOP(R)	I

stop(L) ボタンを押すと,LEDの左端の桁の数字が停止します。

STOP(C) ボタンを押すと,LEDの中央の桁の数字が停止します。

STOP(R) ボタンを押すと,LEDの右端の桁の数字が停止します。

リセット ボタンを押すと,初期状態に戻り,LEDの各桁はカウントアップを開始します。

これで,スロット・プログラムの動作確認を終了します。

プログラムの停止

プログラムを停止します。

SM78Kxxの停止ボタン II , またはメニューの [実行(<u>R</u>)] [ストップ(<u>S</u>)]を選択してください。

🚟 SMK032 : slotprj	_ 🗆 🗙
フェイル(E) 編集(E) 表示(V) オフション(Q) 実行(B) イヘント(N) ブラウズ(B) ジャンプ(J) ウィンドウ(W) ヘルプ(H)	
II) NINAK BER DRAQARD BNE II I VIII	
B Source (slot.c)	
Search K >> Watch Quick Refresh Close	
57 /* ターゲットCPU環境の初期化 */	<u> </u>
<pre></pre>	
* 「1000000000000000000000000000000000000	×



ファイル(E) 編集(E) 表示(V) オフション(()	2) 実行(R) イヘット(N)	ブラウズ® ジャンプ(』 ◯ ⊿ 🛋 🖪) ウィンドウ\@ ヘルフ°(H) ฿ ๙ Ლ ഈ 	•* ©
🔟 Source (slot.c)				
Search < >> Watc 173 174 175 176 176 177 177 ** while (1) { /* 第 ** 177 177 178 178 178 178 178 178 178 178	h Quick Refre 無限ループ */ , [num_idx]; 要素は10個なので,添う への値は0に戻す必要があ	sh <u>Close</u> こを表すnum_idx が ある。 */	10になったら,	
<pre># 185 if(num_idx >= num_idx = 187 } 187 } 187 } 188 189 } /* While */ 190 191 } /* slot */ 192 </pre>	= 10) { 0;			

プログラムを停止すると,ステータス表示エリアの色が元に戻ります。

入出カパネルの解説

スロット・プログラムで使用する入出力パネルには,「7 セグメント LED」,「ボタン(3つ)」, 「リセット・ボタン」が設定されています。

それぞれの設定について解説します。



まず,7セグメント LED 端子の設定について解説します。

入出力パネル・ウインドウの7セグメント LED 端子設定ボタン 🛛 , またはメニューの[接続(P)] [7セグメントLED(S)...]を選択してください。

7 セグメント LED 端子設定ダイアログが開きます。





7 セグメント LED 端子設定 ダイアログが 開きます。



78K0 と 78K4 は入出力ポート P5, P4, 78K0S は P1, P4 の各ビットを, 7 セグメント LED 端子に接続していま す。

スロット・プログラムでは,LEDは5桁使用するため,桁信号1から桁信号5まで設定しています。 ↓ 7 セグメント LED 端子設定の詳細については、「<u>第3章 入門編</u>」をご覧ください。

次に,ホタンの設定について解説しま

入出力パネル・ウインドウのボタン (), またはメニューの[接続(P)] [ボタン(B)...]を選択してください。

ボタン端子設定ダイアログが開きます。



 #2
 3
 STOP(C)
 INTP1
 C H
 (保存

 #3
 3
 STOP(R)
 INTP2
 C H
 クリア

 #4
 3
 C H
 キャンセル

 (保有時間)
 0.05
 msec
 前頁
 次頁

ボタン端子設定ダイアログでは,入出力パネル・ウインドウに表示するボタンの接続情報を設定します。

ボタンの接続は,任意の端子に対して可能で,表示されたボタンを押すことで入力値を与えることができます。 この章では,ボタンを外部割り込み端子(INTP0 ~ 2)に接続しています。

内部割り込みボタンは,ボタンを押すことで内部割り込みを発生させますが,ここではボタンを端子に接続し, 端子の Low-High 状態を認識して,割り込みのトリガにしています。

→ 詳細については,「<u>スロット・プログラムの仕様</u>」をご覧ください。

ボタンを設定する場合,あわせて<u>プルアップ/プルダウンの設定</u>が必要です。

 入出力パネル・ウインドウのボタン
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

 ・
 ・

プルアップ / プルダウン設定ダイアログが開きます。



この章では,ボタンを接続した外部割り込み端子(INTP0 ~ 2)をプルダウンに設定しています。

プルアップ / プルダウンの設定とは?
SM78Kxxの外部部品の一部には,非動作時の出力が未定義なものがあります。
ボタンもその一つで,このような外部部品では,プルアップ / プルダウン設定により,非動作時の
端子状態を設定する必要があります。
なお,プルアップ / プルダウン設定は,ボタンなどの外部部品の設定の前に行う必要があります。
詳細については,ユーザーズ・マニュアル「SM78K シリーズ システム・シミュレータ 操作編」
を参照してください。

次に,リセット・ボタンの設定について解説します。

入出力パネル・ウインドウのボタン **ESSET** , またはメニューの [接続(<u>P</u>)] [リセットボタン(<u>R</u>)]を選択 すると,リセット・ボタンは初期状態の位置に移動します。





リセット・ボタンが,初期状態 の位置(左上隅)に移動します。

🗮 入出力パネル 📃 🗖 🗵
ファイル(E) モード(M) 編集(E) 接続(P) ビットマップ(B)
カスタム(©) 図形(0) オプション(0) ヘルプ(出)
<u>- #炎醫※8.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2.2</u>
Ovz> 2. 8. 8. 8. 8.
STOP(L) STOP(C) STOP(R)

リセット・ボタンの位置は, [モード(<u>M</u>)] [配置(<u>E</u>)]を選択後, リセット・ボタンをドラッグ&ドロップ することにより, 自由に変更できます。

位置の変更後は, [モード(<u>M</u>)] [実行(<u>S</u>)]を選択し, 実行モードに戻してください。 シミュレーション中にリセット・ボタンを押すと, リセット信号がシミュレータに入力されます。

これで,入出力パネルの各設定の解説を終わります。



SM78Kxx を終了します。

SM78Kxx メニューの [ファイル(<u>F</u>)] [終了(<u>X</u>)]を選択してください。

■ SM78K0:slot ファイル(E) 編集	.prj (F) 表示(V)	オフ ^ペ /a ¹ /(O) 5	≢行(R) イベ	νk(N) ブラウフ	*(B) ジャンフ*(.l)	ታሪታኑነት የለለን	ላሆኋ ₍ H)		<u>_ ×</u>
// n/cD *mm本 開く(Q) 名前を付けて() 問じる(C)	e 10/110 森(<u>A</u>)			Ctrl+O				¥7 🕜	
ダウンロード(<u>D</u>) アップロード(<u>U</u>)					Close				
プロジェクト(<u>J</u>) 環境(<u>E</u>))	- ふよいする	a./			
デバッガリセット 終了⊗	(<u>R</u>)				<u>- 2 жлу</u> а.	Φ/			
1 C/¥NECTool 2 C/¥NECTool 3 C/¥NECTool 4 C/¥NECTool	s32¥78KO_samp s32¥78KO_samp s32¥78KO_samp s32¥78KO samp	le¥Chapter4¥s le¥Chapter4¥ii le¥Chapter3¥c le¥Chapter2¥\	:lot.prj nterrupt_func.l :ounter.lmf /RAM.prj	Imf	-				
* 71 72 73 74 75 76 76 77 78 80 81	} /* main * /********** * ポート,: * 周数名: * 見り値: * 使用グロ ***********	*/ ************** ** init_target なし なし ーバル変数 **********	*********** い等のター : : : なし ************	:*********** ゲットCPU環 :*********	************************************	**************************************	*****		- 411-
slot.c#58	main	1	008A						



この章で設定した入出力パネルの状態などをプロジェクト・ファイルに保存する場合は はい(Y) を,保存 しない場合は いいえ(N) を押してください。 PM plus を終了します。

PM plus のメニューの [ファイル(<u>F</u>)] [PM plus の終了(<u>X</u>)]を選択してください。

🧱 PM plus - slot.prw [OutPut]		
ファイル(<u>F)</u> 編集(E) 検索(D) β	皆層(L) 表	ミ示(V) プロジュウト(P) ビルド(B) ツール(T) ウィントウ(W) ヘルフペ田)
新規作成(N) 開く(Q) 挿入読み込み(D)	Ctrl+N Ctrl+O	Image: Constraint of the second se
ାଲାରେଲି 		
ワークスペースの新規作成(E) ワークスペースを開く(\\) ワークスペースの保存(R) ワークスペースを閉じる(k)		Compilation complete, 0 error(s) and 0 warning(s) found. C:#NECTools32#BIN¥Ik78K0.exe -finterrupt_func.plk* 78K/0 Series Linker E3 60a [3 Apr 2003]*
上書き保存(<u>S)</u> 名前を付けて保存(<u>A</u>)	Ctrl+S	- Copyright (C) NEC Electronics Corporation 1990,2003+
ソース・ファイル・名の変更(E) すべてのソース・ファイルを保存(U)		- Device file : V1.01+ - + - +
印刷プレビュー(\/) 印刷(<u>P</u>)	Ctrl+P	- C:¥NECTools32¥BIN¥oc78K0.exe -yC:¥NECTools32¥DEV¥ interrupt_fun
1 C:¥NECTools32¥¥counter.c		Copyright(C) NEC Electronics Corporation 1990,2003+
C:¥NECTools32¥¥slot.prw C:¥NECTools32¥¥counter.prw C:¥NECTools32¥¥VRAM.prw		- Target chip : uPD780034+ - Device file : V1.01+ - +
		- Object Conversion Complete, O error(s) and O warning(s)
		Build Total error(s) : 0 Total warning(s) : 0 [EOF]
- PM plusを終了し、変更したファイルの	保存を確認	Lat JL: 1 X: 1 Y: 38

PM plus のプロジェクト情報は逐次保存されているため,保存の確認はされません。

プログラム解説

ここでは,スロット・プログラムで使用している次の機能をC言語で記述する方法について解説します。

- ・デバイス内蔵の特殊機能レジスタへのアクセス
- ・割り込みや例外発生時の関数の定義
- ・割り込みを制御する関数
- CPU を制御する命令

なお,スロット・プログラムのソース・リストは「<u>付録 スロット・プログラム・ソースリスト</u>」に添付して ありますので,あわせてご覧ください。

CC78Kxx C コンパイラは, ANSI 規格で規定された言語仕様をサポートしています。 また,機種依存処理をできるだけ C 言語で記述可能にするために, ANSI 規格の C 言語仕様を 拡張しています。 この拡張言語仕様により,割り込み時処理や SFR の参照など,機種に依存した処理を,オブジェ

クト効率を保ちながら C 言語で記述することが可能となり,プログラムの継承性や開発効率を 向上させることができます。

拡張仕様には,次のものがあります。

- ・外部変数の saddr 領域への割り付け指定 (sreg 変数)
- ・関数引数や自動変数の saddr 領域やレジスタへの割り付け指定 (nerec, noauto 関数)
- ・短い命令での関数呼び出し指定(callt 関数, callf 関数^{注1})
- ・SFR へのアクセス
- ・割り込み処理のC言語記述(レジスタ・バンク^{注2}切り替え可能)
- ・割り込み禁止 / 許可命令の出力
- ・C 言語ソース・プログラムへのアセンブラ記述挿入
- ・CPU 制御命令の出力
- ・2 進数定数の記述

拡張言語仕様部分についての詳細は,ユーザーズ・マニュアル 「CC78Kxx C コンパイラ・ パッケージ 言語編」を参照してください。

注1. callf 命令は, 78KOS ではサポートされていません。

2. 78K0S ではサポートされていません。

レジスタ名を用いた特殊機能レジスタへのアクセス 『#pragma sfr』

各デバイスで内蔵している周辺機能のためのレジスタを,特殊機能レジスタと呼びます。 特殊機能レジスタを使うには,ソースの先頭部分にプラグマ指令『#pragma sfr』を記述します。

#pragma sfr

スロット・プログラムのソースでは, slot.c の始めに, #pragma sfr を記述しています。

[slot.c]

```
/* 特殊機能レジスタ名 (SFR 名) の有効化 */
```

#pragma sfr

プラグマ指令『#pragma sfr』を記述すると、特殊機能レジスタ名を、通常の符号なし(unsigned)外部変数 のように使用することができますが、『#pragma sfr』の記述なしに、特殊機能レジスタ名を使用すると「変数 が未定義である(error: E2210: 特殊機能レジスタ名: not defined)」というエラーが出ます。

例

```
/* #pragma sfr の記述がない場合 */
main() {
P1 = 0; /* P1 の未定義エラー */
}
```

→ 特殊機能レジスタの詳細については,各デバイスのユーザーズ・マニュアル を参照してください。

割り込み関数の登録

『#pragma interrupt』または『#pragma vect』と『__interrupt』

割り込みとは,現在実行中の処理を強制的に中断させて,別の処理を行うことです。割り込みで中断された 処理は,割り込んだ処理の終了後に再開します。

割り込みを発生させることを,割り込み要求と呼びます。

割り込み要求時に行われる処理は,関数として記述でき,割り込みの要因によって,実行する関数を指定する ことができます。このような関数を"割り込み関数"と呼びます。

ある関数を,割り込み関数とするには,次の作業が必要です。

- ・関数名と割り込みの要因(割り込み要求名)とを対応づける
- ・関数を割り込み関数に指定する

関数名と割り込み要求名とを対応づけるには、プラグマ指令『#pragma interrupt』または『#pragma vect』を 使用します。

#pragma interrupt 割り込み要求名 関数名

#pragma vect 割り込み要求名 関数名

このプラグマ指令に記述された割り込み要求名に従い,関数を割り込み関数として登録します。 計定可能な割り込み要求名については,各デバイスのユーザーズ・マニュアルを参照してください。

関数を割り込み関数に指定するには,関数定義(もしくは関数宣言)に,修飾子 __interrupt を付けます。

_interrupt 関数定義 or 関数宣言

割り込み関数に指定された関数は,通常のレジスタの退避/復帰に加え,割り込みに対応したレジスタの 退避/復帰も行います。また,reti命令で復帰します。

割り込み関数に指定できる関数は,通常,戻り値も引き数もない関数("void Func(void)"型)です。

スロット・プログラムのソースでは,関数名と割り込み要求名との対応づけを slot.c で,関数の割り込み 関数の指定を interrupt_func.h で行っています。

[slot.c]

#pragma interrupt INTP0 stp_btn_Left
#pragma interrupt INTP1 stp_btn_Center
#pragma interrupt INTP2 stp_btn_Right

[interrupt_func.h]

__interrupt void stp_btn_Left(void);

__interrupt void stp_btn_Center(void);

__interrupt void stp_btn_Right(void);

部分的な割り込みの可否設定 『DI();』と『EI();』

ある処理を行っている間はマスカブル割り込みを禁止にし,処理が終わったらマスカブル割り込みを受け付け るようにするということもC言語で記述できます。

割り込みの受け付けを部分的に禁止もしくは許可するには,割り込み制御関数(DI/EI)を利用します。 まず『#pragma DI』, 『#pragma EI』を指定します。

#pragma DI #pragma EI

DI();

DI 関数は,割り込みを禁止(di 命令を生成)します。

EI();

EI 関数は,割り込みを許可(ei 命令を生成)します。
指定例:

```
#pragma DI
#pragma EI
void func() {
    int i,j,k;
    .....
DI(); /* 割り込み禁止 */
    /*
    * 割り込み禁止状態で行う必要のある処理
    */
    EI(); /* 割り込み許可 */
    .....
return;
}
```

スロット・プログラムのソースでは,ボタンが押されたことによって割り込みを発生させるため,数字をカウ ント・アップするループの前に,割り込みを有効にしています。

[slot.c]

```
/* 割り込み許可 */
El();
```

CPU 制御命令出力 『HALT();』と『STOP();』と『BRK();』と『NOP();』

CPU 制御を行う命令を,関数形式でC言語で記述できます。 BRK 命令は 78K0, 78K4 のみサポートされています。

機能を使用する場合には, #pragma 指令で使用することを宣言します。

#pragma HALT

#pragma STOP

#pragma BRK

#pragma NOP

HALT();

78K0, 78K0S の場合, HALT 関数は, halt 命令を生成します。 78K4 の場合, STBC を操作するコードを生成します。

STOP();

78K0,78K0Sの場合,STOP 関数は,stop 命令を生成します。78K4の場合,STBC を操作するコードを生成します。

BRK();

78K0, 78K4 の場合, BRK 関数は, brk 命令を生成します。

NOP();

NOP 関数は , nop 命令を生成します。

指定例:

```
#pragma HALT
#pragma STOP
#pragma BRK
#pragma NOP
void func() {
    .....
    HALT(); /* halt 命令出力 */
    STOP(); /* stop 命令出力 */
    BRK(); /* brk 命令出力 */
    NOP(); /* nop 命令出力 */
    .....
    return;
}
```

【注意】

CC78K4 で, HALT(), STOP()は, STBC のCK1/CK0の値を調べて, それに応じた HALT 用または STOP 用の値を STP/HLT に設定する命令を出力します。

(STBC に対しては,「MOV STBC,#値」でしか設定できません。)

そのため, STBC の 2, 3, 6, 7 ビット目には, 0 とした命令が出力されます。

2, 3, 6, 7 ビット目が, 0 固定ではないデバイスの場合には, HALT(), STOP()は, 使用できませんので, 注意してください。

7	6	5	4	3	2	1	0
0	0	CK1	CK0	0	0	STP	HLT



ここでは付録として,各章で扱った次の項目の詳細について紹介します。

- <u>uoVRAM.dllの作成方法</u>
- <u>カウンタ・プログラム・ソースリスト</u>
- counter.c
- <u>スロット・プログラム・ソースリスト</u>
- <u>slot.c</u>
- interrupt func.h
- interrupt_func.c

uoVRAM.dll の作成方法

付 録

ここでは,第2章で使用する仮想 VRAM の外部部品(uoVRAM.dll)を Microsoft Visual C++(以後 VC++)で作 成する方法を紹介します。

なお,第2章のサンプル環境には,完成された uoVRAM.dll が添付されていますので,ここでの操作は第2章を 進めるにあたって必要な操作ではありません。本節はお客様ご自身で外部部品ユーザ・オープン・インタフェース を使用した外部部品を作成する際の参考資料です。

 外部部品ユーザ・オープン・インタフェースの詳細については、SM78K シリーズ システム・シミュレータ
 Ver.2.30 以上 ユーザーズ・マニュアル 外部部品ユーザ・オープン・インタフェース仕様編(U15802J)を 参照してください。

uoVRAM.dllのソース・ファイルは,他のサンプル・プログラムと一緒に入っています。作成にはVC++ Ver.5 以上が必要です。ここでは,VC++ Ver.6 での作成方法を紹介します。

< 使用するファイル >

uoVRAM.dll の作成には,自己展開型圧縮ファイルに入っているソース・ファイル(vram.c,vram.def)と, SM78K0 と同時にインストールされる外部部品ユーザ・オープン・インタフェース用ファイル(upart32.cpp, uparts32.h)を使用します。



<uoVRAM.dllの作成手順>

VC++ Ver.6 でビルドする手順は次のとおりです。

VC++を起動し,新規に"Win32 Dynamic-Link Library"プロジェクトを作成します。
 まず,メニューの[ファイル(<u>F</u>)] [新規作成(<u>N</u>)...]を選択してください。



設定が完了したら, OK ボタンを押してください。

注 プロジェクト名は "uoVRAM" としてください。これ以外のプロジェクト名を使用する場合は, VC++が出力する DLL のファイル名が "uoVRAM.dll"になるようにオプションを変更してください。SM78Kxx の外部部品ユー ザ・オープン・インタフェースでは, DLL のファイル名と, DLL から EXPORT する関数名を対応づける必要があ ります。本ソースをそのまま利用する場合, DLL のファイル名に uoVRAM.dll 以外を使用すると, SM78Kxx は DLL を正常に読むことができません。詳細については, SM78K シリーズ システム・シミュレータ Ver.2.30 以上 ユーザーズ・マニュアル 外部部品ユーザ・オープン・インタフェース仕様編 (U15802J)を参照してください。



DLL の種類を選択します。

" 空の DLL プロジェクト(<u>E</u>) " を選択してください。

Win32 Dynamic-Link Library - ステップ゜1 / 1		? ×
	作成する DLL の種類を選択してください。 ② 空の DLL 7百ジェクト(E) ③ 単純な DLL 7百ジェクト(S) ③ ジンボルをエクスポートする DLL(D)	
< 戻る(<u>B</u>)	次へ(11) (終7(E) +v	ンセル

選択後, 終了(<u>F</u>) ボタンを押すとプロジェクトが生成されます。

- 1.でプロジェクトを作成したディレクトリに,仮想 VRAM のソース・ファイル (vram.c, vram.def)と外部部品 ユーザ・オープン・インタフェース用ファイル (uparts32.cpp, uparts32.h)をコピーします (ファイルのある位 置については<u><使用するファイル></u>をご覧ください)。
- 2.でコピーしたファイル (vram.c, vram.def, uparts32.cpp, uparts32.h)を 1.で作成したプロジェクトに登録します。VC++のプロジェクト・ワークスペース・ウインドウの "File View " タブの中で設定します。

まず, "File View"タブを選択してください。

次に, "uoVRAM ファイル"を右クリックしてプルダウン・メニューを表示し, [ファイルをプロジェクトへ追加(<u>F</u>)…]を選択してください。



"ファイルの種類(<u>T</u>)"を"すべてのファイル (*.*)"にすると,4つのファイルをまとめて選択できます。 選択後,OK」ボタンを押すと,プロジェクトにファイルが登録されます。 ビルドを実行します。

メニューの [ビルド(<u>B</u>)] [ビルド(<u>B</u>)] を選択してください。

∬ファイル(E) 編集(E) 表示(V) 挿入Φ プロシェクト(P)	ビルド(13) ツール(17) ウィンドウ(14) ヘルフ(14)
" uoVRAM.dll " と表示されて いることを確認してください。	 ジハギ(ル(2) Otrl+F7 ビルト*(B. uoVRAMdI) F7 ドリーン(E) デ*ハ*ッケ*の開始(D) デ*ハ*ッケ*の開始(D)
	! 実行 ⊗ Ctrl+F5
	アクティアな構成の設定(O) 構成(E) ア ¹ ロファイル(<u>P</u>)

以上の作業により, " uoVRAM.dll " が作成されます。

カウンタ・プログラム・ソースリスト

(1/5)

```
[counter.c]
  78K0 のソースリスト
/*
* Copyright (C) NEC Electronics Corporation 2000,2004
* All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely
* for the purpose for which it was furnished by NEC Electronics Corporation. No part
* of thisProgram may be reproduced or disclosed to others, in any form,
* without the prior written permission of NEC Electronics Corporation.
*/
          *****
  *****
* カウンタプログラム(µPD780034用)
      ****
/* コンパイラ定義 */
#pragma NOP
#pragma HALT
#pragma El
/* 特殊機能レジスタ名(SFR 名)の有効化 */
#pragma sfr
/* 関数 interrupt1()を, INTTMOOの割込み関数に設定 */
#pragma vect INTTM00 interrupt1
/* カウンタ値保持変数 */
volatile int count1; /* LED に表示する数値(1の位) */
volatile int count10; /* LED に表示する数値(10の位) */
  ******
      *******
* 7 セグメント LED に数値を表示
* count1に保持している値を LED の1桁目(1の位), count10 に
* 保持している値を LED の 2 桁目(10 の位)に表示する。
```

```
* 関数名:putLED
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
     int count1
     int count10
*****
  ******
void putLED()
{
     static unsigned char /* 7 セグメント LED '0' - '9'のパターン */
          box[10]={0x77,0x24,0x6b,0x6d,0x3c,0x5d,0x5f,0x74,0x7f,0x7d};/* ERROR*/
     /* 7 セグメント LED にパターンを転送 (1 の位) */
     P4 = 0;
    P5 = box[count1];
     P4 = 1;
     /* 時間調整 */
    NOP();
    NOP();
     /* 7 セグメント LED にパターンを転送 (10 の位) */
     P4 = 0;
    P5 = box[count10];
     P4 = 2;
     return;
}
      *****
       ******
* デバッグ用メイン関数
* 以下の処理を行う。
*
 ・LED 表示の為に P5, P4 のモードを PM5, PM4 により設定する。
* ・カウンタ値(count1, count10)を初期化する。
* ・INTTMOO 割込みを初期化し,割込みを許可する。
  ・INTTMOO 割込み発生まで, CPU を HALT 状態にする。
   また, INTTMOO 割込みが終了したら CPU を HALT 状態にし,
*
   次の INTTMOO 割込みを待つ。
* 関数名:main
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
     int count1
     int count10
     ******
```

```
(3/5)
```

```
*****
void main(){
    /*******
     * 初期化処理
     /* 7 セグメント LED に出力するポートのモードを設定 */
    PM5 = 0x00; /*P5 (P50-P57)は出力 */
    PM4 = 0x00; /*P4 (P40-P47)は出力 */
    /* INTTMOO 割込みの割込みレベルの設定と割込みの有効化 */
    TMPROO = 0:/* INTTMOO 割込みレベルを高優先順位レベルにする */
    TMMK00 = 0;/* INTTM00 割込みの受付を許可する */
    /* カウンタの初期化 */
    count1 = 0; /* LED に表示する数値の初期化(1の位) */
    count10 = 0;/* LED に表示する数値の初期化(10の位) */
    /* 初期値の表示 */
    putLED(); /* LED に数値を表示 */
    /******
     * メインループ
     /* この後の処理は割込み処理のみなので、
      割り込み処理の無い時は、HALT 状態にする。*/
    EI(); /* 割込みの許可 */
    while(1)
    {
      HALT();
    }
}
```

(4/5)

```
******
* カウントアップ処理
* (INTTMOO で呼び出される割込み関数)
* count1, count10 で構成されるカウンタを1カウントアップする。
* 99までカウントしたら次は0にする。
* また, putLED 関数で, LED にカウンタの値を表示する。
*
* 関数名:interrupt1
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
*
   int count1
*
   int count10
*******
```

(5/5)

```
*****
___interrupt
void interrupt1()
{
    /******
     * カウンタのカウントアップ
     count1++; /* 1 の位をカウントアップ */
    /* 桁上がり処理 */
    /* 1の位の桁上がりが必要かどうか? */
    if(count1=10)
                            /* ERROR*/
    { /* 1 の位が 10 なので,桁上がり処理を行う */
         count1 = 0; /* 1の位を0に設定 */
         count10++; /* 10 の位をカウントアップ */
         /* 10の位の桁上がりが必要かどうか? */
         if(count10==10)
         {
         /* 10 の位が 10 なので,桁上がり処理を行う */
           count10 =0; /* 10の位を0に設定 */
                  /* LED が2桁なので99の次は0に戻す*/
         }
    }
    /********
     * LED に数値を表示
     putLED(); /* LED に数値を表示 */
    return;
}
```

付 録

78K0S のソースリスト

/* * Copyright (C) NEC Electronics Corporation 2000,2004 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely * for the purpose for which it was furnished by NEC Electronics Corporation. No part * of thisProgram may be reproduced or disclosed to others, in any form, * without the prior written permission of NEC Electronics Corporation. */ ****** * カウンタプログラム (µPD789046用) /* コンパイラ定義 */ #pragma NOP #pragma HALT #pragma El /* 特殊機能レジスタ名(SFR 名)の有効化 */ #pragma sfr /* 関数 interrupt1()を, INTWT の割込み関数に設定 */ #pragma vect INTWT interrupt1 /* カウンタ値保持変数 */ volatile int count1; /* LED に表示する数値(1の位) */ volatile int count10; /* LED に表示する数値(10の位) */ /**** ***** * 7 セグメント LED に数値を表示 * count1 に保持している値を LED の 1 桁目(1 の位), count10 に * 保持している値を LED の 2 桁目(10 の位)に表示する。 * 関数名: putLED * 引数 :なし * 戻り値:なし * 使用グローバル変数 : int count1 int count10

(1/4)

```
void putLED()
{
     static unsigned char /* 7 セグメント LED '0' - '9'のパターン */
          box[9]={0x77,0x24,0x6b,0x6d,0x3c,0x5d,0x5f,0x74,0x7f,0x7d};/* ERROR*/
     /* 7 セグメント LED にパターンを転送 (1 の位) */
     P0 = 0;
     P1 = box[count1];
     P0 = 1;
     /* 時間調整 */
     NOP();
     NOP();
     /* 7 セグメント LED にパターンを転送 (10 の位) */
     P0 = 0;
     P1 = box[count10];
     P0 = 2;
     return;
}
      *******
       *****
* デバッグ用メイン関数
* 以下の処理を行う。
*
  ・LED 表示の為に P1, P0 のモードを PM1, PM0 により設定する。
* ・カウンタ値(count1, count10)を初期化する。
*
  ・INTWT 割込みを初期化し,割込みを許可する。
  ・INTWT 割込み発生まで, CPU を HALT 状態にする。
   また, INTWT 割込みが終了したら CPU を HALT 状態にし,
   次の INTWT 割込みを待つ。
* 関数名:main
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
     int count1
     int count10
      ****
******
void main(){
     /******
     * 初期化処理
     /* 7 セグメント LED に出力するポートのモードを設定 */
     PM1 = 0x00; /*P1 (P10-P17)は出力 */
     PMO = 0x00; /*PO (POO-PO7)は出力 */
```

```
(3/4)
```

```
/* INTWT 割込みの割込みレベルの設定と割込みの有効化 */
    WTMK = 0;/* INTWT 割込みの受付を許可する */
    /* カウンタの初期化 */
    count1 = 0; /* LED に表示する数値の初期化(1の位) */
    count10 = 0; /* LED に表示する数値の初期化(10の位) */
    /* 初期値の表示 */
    putLED(); /* LED に数値を表示 */
    /**********
     * メインループ
    ******
    /* この後の処理は割込み処理のみなので,
     割り込み処理の無い時は, HALT 状態にする。 */
    EI(); /* 割込みの許可 */
    while(1)
    {
      HALT(); /* */
    }
}
        *******
* カウントアップ処理
* (INTWT で呼び出される割込み関数)
* count1, count10 で構成されるカウンタを1カウントアップする。
* 99までカウントしたら次は0にする。
* また, putLED 関数で, LED にカウンタの値を表示する。
* 関数名: interrupt1
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
    int count1
*
    int count10
     ******
*****
_interrupt
void interrupt1()
{
    /***********
    * カウンタのカウントアップ
    *****
    count1++; /* 1 の位をカウントアップ */
```

```
/* 桁上がり処理 */
/* 1の位の桁上がりが必要かどうか? */
if(count1=10)
                           /* ERROR*/
{ /* 1の位が10なので,桁上がり処理を行う*/
     count1 = 0; /* 1の位を0に設定 */
     count10++; /* 10 の位をカウントアップ */
     /* 10 の位の桁上がりが必要かどうか? */
     if(count10==10)
     {
     /* 10 の位が 10 なので,桁上がり処理を行う*/
       count10 =0; /* 10の位を0に設定 */
               /* LED が2桁なので99の次は0に戻す*/
     }
}
/********
* LED に数値を表示
******
putLED();
        /* LEDに数値を表示 */
return;
```

78K4 のソースリスト

Copyright (C) NEC Electronics Corporation 2000,2004 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely * for the purpose for which it was furnished by NEC Electronics Corporation. No part * of thisProgram may be reproduced or disclosed to others, in any form, * without the prior written permission of NEC Electronics Corporation. */ ****** ***** * カウンタプログラム (µPD784035用) /* コンパイラ定義 */ #pragma NOP #pragma HALT #pragma El /* 特殊機能レジスタ名(SFR 名)の有効化 */ #pragma sfr /* 関数 interrupt1()を, INTCOOの割込み関数に設定 */ #pragma vect INTCO0 interrupt1 /* カウンタ値保持変数 */ volatile int count1; /* LED に表示する数値(1の位) */ volatile int count10; /* LED に表示する数値(10の位) */ /***** ***** * 7 セグメント LED に数値を表示 * count1 に保持している値を LED の 1 桁目(1 の位), count10 に 保持している値を LED の 2 桁目(10 の位)に表示する。 * 関数名:putLED * 引数 :なし * 戻り値:なし * 使用グローバル変数 : int count1 int count10 ******

アプリケーション・ノート U17047JJ1V0AN

(1/4)

```
void putLED()
{
     static unsigned char /* 7 セグメント LED '0' - '9'のパターン */
          box[9]={0x77,0x24,0x6b,0x6d,0x3c,0x5d,0x5f,0x74,0x7f,0x7d};/* ERROR*/
     /* 7 セグメント LED にパターンを転送 (1 の位) */
     P4 = 0;
     P5 = box[count1];
     P4 = 1;
     /* 時間調整 */
     NOP();
     NOP();
     /* 7 セグメント LED にパターンを転送 (10 の位) */
     P4 = 0;
     P5 = box[count10];
     P4 = 2;
     return;
}
         *******
      *****
* デバッグ用メイン関数
*
  以下の処理を行う。
* ・LED 表示の為に P5, P4 のモードを PM5, PM4 により設定する。
* ・カウンタ値(count1, count10)を初期化する。
  ・INTCOO 割込みを初期化し,割込みを許可する。
* ・ INTCOO 割込み発生まで, CPU を HALT 状態にする。
    また, INTCOO 割込みが終了したら CPU を HALT 状態にし,
   次の INTCOO 割込みを待つ。
* 関数名:main
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
      int count1
      int count10
  *****
void main(){
     /*******
      * 初期化処理
             ***********************************
     /* 7 セグメント LED に出力するポートのモードを設定 */
     PM5 = 0x00; /*P5 (P50-P57)は出力 */
     PM4 = 0x00; /*P4 (P40-P47)は出力 */
```

```
/* INTCOO 割込みの割込みレベルの設定と割込みの有効化 */
    CIC00 = CIC00 & 0xF0;/* INTC00 割込みレベルを高優先順位レベルにする(CPR001=0,CPR000=0) */
    CMKOO = 0;
               /* INTCOO 割込みの受付を許可する */
    /* カウンタの初期化 */
    count1 = 0;/* LED に表示する数値の初期化(1の位) */
    count10 = 0; /* LED に表示する数値の初期化(10 の位) */
    /* 初期値の表示 */
    putLED(); /* LED に数値を表示 */
    /*****
     * メインループ
     *****
    /* この後の処理は割込み処理のみなので,
      割り込み処理の無い時は, HALT 状態にする。 */
    EI(); /* 割込みの許可 */
    while(1)
    {
      HALT(); /* */
    }
      * カウントアップ処理
* (INTCOO で呼び出される割込み関数)
* count1, count10 で構成されるカウンタを1カウントアップする。
* 99までカウントしたら次は0にする。
* また, putLED 関数で, LED にカウンタの値を表示する。
* 関数名: interrupt1
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数:
    int count1
*
    int count10
*******
___interrupt
void interrupt1()
```

}

{

```
/********
* カウンタのカウントアップ
count1++; /* 1 の位をカウントアップ */
/* 桁上がり処理 */
/* 1の位の桁上がりが必要かどうか? */
                        /* ERROR*/
if(count1=10)
{ /* 1 の位が 10 なので,桁上がり処理を行う */
    count1 = 0; /* 1の位を0に設定 */
    count10++; /* 10 の位をカウントアップ */
    /* 10 の位の桁上がりが必要かどうか? */
    if(count10==10)
    {
    /* 10 の位が 10 なので,桁上がり処理を行う */
      count10 =0; /* 10の位を0に設定 */
             /* LED が2桁なので99の次は0に戻す*/
    }
}
/*****
* LED に数値を表示
putLED();
       /* LED に数値を表示 */
return;
```

}

スロット・プログラム・ソースリスト

```
[Slot.c]
78K0 のソースリスト
```

(1/4)

/* * Copyright (C) NEC Electronics Corporation 2000,2004 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely * for the purpose for which it was furnished by NEC Electronics Corporation. No part * of thisProgram may be reproduced or disclosed to others, in any form, * without the prior written permission of NEC Electronics Corporation. */ * スロット・プログラム (uPD780034 用) ****** /* コンパイラ定義 */ #pragma El /* 特殊機能レジスタ名(SFR 名)の有効化 */ #pragma sfr /* INTPO, INTP1, INTP2 それぞれの割り込み関数として, * 関数 stp_btn_Left(), stp_btn_Center(), stp_btn_Right()を設定 */ #pragma interrupt INTPO stp_btn_Left #pragma interrupt INTP1 stp_btn_Center #pragma interrupt INTP2 stp_btn_Right #include "interrupt_func.h" /*割り込み関数の宣言 */ /* 表示(LED 点灯)位置の指定 */ unsigned char place; /* 表示(LED 点灯)用数字データ */ unsigned char num_data[10] = { 0x77, 0x24, 0x6b, 0x6d, 0x3c, 0x5d, 0x5f, 0x74, 0x7f, 0x7d, }; /* '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' */ /* ポート,割り込みレベル等のターゲット CPU 環境の初期化関数 */ void init_target(void); /* スロット表示用関数 */ void slot(void);

(2/4)

```
/***********
* スロット用メイン関数
   表示用数字(0~9)をループさせて, LED に表示する.
   ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ,
   表示している数値を固定する.
* 関数名:main
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 :
    unsigned char place;
          *******
void main(void)
{
 /* ターゲット CPU 環境の初期化 */
 init_target();
 /* 表示位置指定変数の初期化 */
 place = 0x15; /* 10101 : LED を一つおきに3つ点灯する. */
 /* 7 セグメント LED 消灯 */
 P4 = 0x00;
 P5 = 0x00;
 P4 = 0xFF;
 /* スロット(数字点灯)の処理 */
 slot();
} /* main */
* ポート,割り込みレベル等のターゲット CPU 環境の初期化関数
* 関数名: init target
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数 : なし
void init_target(void){
 /*
 * Port0 を 割り込みの入力に使用
 * Port5 を 7 セグメント LED 点灯用に使用
 * Port4 を 桁指定用に使用
 */
 /* Port0 の全bit を入力モードにする. */
 PMO = 0xFF; /* モード・レジスタ(PMO)の全 bit を入力(1)に設定 */
```

(3/4)

```
/* Port5 の全 bit を出力モードにする. */
 PM5 = 0x00; /* モード・レジスタ(PM5)の全bitを出力(0)に設定 */
 /* Port4 の全bit を出力モードにする. */
 PM4 = 0x00; /* モード・レジスタ(PM4)の全bitを出力(0)に設定 */
 /* Port5,Port4 を入出力ポートモードで使用するために
 * メモリ拡張モードレジスタをポートモードに設定 */
 MEM = 0x00;
 /*
 * 外部端子による外部割り込み要求の有効エッジは,
 * 外部割り込み立ち上がりエッジ許可レジスタ(EGP)
 * 外部割り込み立ち下がりエッジ許可レジスタ(EGN)で指定する.
 * 本プログラムでは,外部割り込み要求に INTPO,INTP1,INTP2 を使用するため
 * 外部割り込み立ち上がりエッジ許可レジスタ(EGP)を,それぞれ立ち上がりエッジ有効に設定し,
 * 外部割り込み立ち下がりエッジ許可レジスタ(EGN)を無効に設定する.
 */
 EGP = 0x07; /* 0x07 = XX XX 0 1 1 1
                      | | | INTPO
                      | | INTP1
                      | INTP2
                      *
                      INTP3
            */
 EGN = 0x00; /* 0x00 = XX XX 0 0 0 0
            *
                      | | | INTPO
                      | | INTP1
                      | INTP2
                      INTP3
            * EGP | EGN |
            *_____
            * 0 | 0 | 割込み禁止
            * 0 | 1 | 立ち下がりエッジ
            * 1 | 0 | 立ち上がりエッジ
            * 1 | 1 | 立ち上がり, 立ち下がり両エッジ
            */
 PMK0 = 0; /* INTPO の割り込みの受け付け許可 */
 PMK1 = 0; /* INTP1 の割り込みの受け付け許可 */
PMK2 = 0; /* INTP2 の割り込みの受け付け許可 */
} /* init_target */
```

(4/4)

```
******
/**********
* スロット表示用関数
*
    表示用数字(0~9)をループさせて, LED に表示する.
    ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ,
    表示している数値を固定する.
* 関数名:slot
* 引数 :なし
* 戻り値:なし
* 使用グローバル変数:
     unsigned char place;
     unsigned int num_data[];
                    *******
void slot(void) {
 /*
  * 表示用数字を 0~9 までループさせる.
  * 数字の点灯位置は, place 変数に従う.
 */
 /* 表示数値(num_data)用インデックス */
 int num_idx = 0;
 /* 割り込み許可 */
 EI();
 while (1) { /* 無限ループ */
  /* 数字表示 */
  P5 = num_data[num_idx];
  P4 = place;
  num_idx++ ;
  /* num_dataの要素は10個なので,添字を表すnum_idxが10になったら,
   * インデックスの値は0に戻す必要がある. */
  if (num_i dx >= 10) {
     num_idx = 0;
  }
 } /* While */
} /* slot */
```

付 録

78K0S のソースリスト

/* * Copyright (C) NEC Electronics Corporation 2000,2004 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely * for the purpose for which it was furnished by NEC Electronics Corporation. No part * of thisProgram may be reproduced or disclosed to others, in any form, * without the prior written permission of NEC Electronics Corporation. */ * スロット・プログラム (uPD789046 用) /* コンパイラ定義 */ #pragma El /* 特殊機能レジスタ名(SFR 名)の有効化 */ #pragma sfr /* INTP0, INTP1, INTP2 それぞれの割り込み関数として, * 関数 stp_btn_Left(), stp_btn_Center(), stp_btn_Right()を設定 */ #pragma interrupt INTPO stp_btn_Left #pragma interrupt INTP1 stp_btn_Center #pragma interrupt INTP2 stp_btn_Right #include "interrupt_func.h" /*割り込み関数の宣言 */ /* 表示(LED 点灯)位置の指定 */ unsigned char place; /* 表示(LED 点灯)用数字データ */ unsigned char num_data[10] = { 0x77, 0x24, 0x6b, 0x6d, 0x3c, 0x5d, 0x5f, 0x74, 0x7f, 0x7d, }; /* '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' */ /* ポート,割り込みレベル等のターゲット CPU 環境の初期化関数 */ void init_target(void); /* スロット表示用関数 */ void slot(void); * スロット用メイン関数 表示用数字(0~9)をループさせて, LED に表示する. * ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ, * 表示している数値を固定する.

(1/4)

```
関数名:main
•
   引数 :なし
•
   戻り値:なし
•
  使用グローバル変数:
•
  unsigned char place;
      *****
void main(void)
{
 /* ターゲット CPU 環境の初期化 */
 init_target();
 /* 表示位置指定変数の初期化 */
 place = 0x15; /* 10101 : LED を一つおきに3つ点灯する. */
 /* 7 セグメント LED 消灯 */
 P0 = 0x00;
 P1 = 0x00:
 P0 = 0xFF;
 /* スロット(数字点灯)の処理 */
 slot();
} /* main */
ポート,割り込みレベル等のターゲット CPU 環境の初期化関数
•
 関数名:init_target
•
  引数 :なし
•
•
   戻り値:なし
  使用グローバル変数 : なし
•
              *****
* * * * * * * * * * * * * * * * * * *
void init_target(void){
 /*

    Port2 を 割り込みの入力に使用

    Port1 を 7 セグメント LED 点灯用に使用

    Port0 を 桁指定用に使用

 */
 /* Port2 の全bit を入力モードにする. */
 PM2 = 0xFF; /* モード・レジスタ(PM2)の全bitを入力(1)に設定 */
 /* Port1 の全 bit を出力モードにする. */
 PM1 = 0x00; /* モード・レジスタ(PM1)の全 bit を出力(0)に設定 */
```

```
/* Port0 の全 bit を出力モードにする. */
 PMO = 0x00; /* モード・レジスタ(PMO)の全bitを出力(0)に設定 */
 /*
  外部端子による外部割り込み要求の有効エッジは、
  外部割り込みモード・レジスタ INTMO で指定する.
.
  本プログラムでは,外部割り込み要求に INTP0,INTP1,INTP2 を使用するため
  外部割り込みモード・レジスタ0(INTMO)を,それぞれ立ち上がりエッジ有効に設定する.
 */
 INTMO = 0x54; /* 0x54 = 0 1 0 1 0 1 XX
  •
•
   •
  | | INTPO
•
  | INTP1
•
•
  •
  INTP2
 00: 立ち下がりエッジ
•
•
  01: 立ち上がりエッジ
  10: RFU(予約)
•
 11:立ち上がり,立ち下がり両エッジ
•
          */
PMK0 = 0; /* INTPO の割り込みの受け付け許可 */
PMK1 = 0; /* INTP1 の割り込みの受け付け許可 */
PMK2 = 0; /* INTP2 の割り込みの受け付け許可 */
} /* init_target */
•
  スロット表示用関数
  表示用数字(0~9)をループさせて, LED に表示する.
•
•
  ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ,
  表示している数値を固定する.
•
•
  関数名:slot
•
  引数 :なし
  戻り値:なし
•
  使用グローバル変数:
•
  unsigned char place;
 unsigned int num_data[];
```

```
void slot(void) {
 /*
  * 表示用数字を 0~9 までループさせる.
  * 数字の点灯位置は , place 変数に従う .
  */
 /* 表示数値(num_data)用インデックス */
 int num_idx = 0;
 /* 割り込み許可 */
 EI();
 while (1) { /* 無限ループ */
   /* 数字表示 */
   P0 = 0x00;
   P1 = num_data[num_idx];
   P0 = place;
   num_idx++ ;
   /* num_dataの要素は10個なので,添字を表すnum_idxが10になったら,
   * インデックスの値は0に戻す必要がある. */
   if( num_idx >= 10 ) {
      num_idx = 0;
   }
 } /* While */
} /* slot */
```

付 録

78K4 のソースリスト

/* * Copyright (C) NEC Electronics Corporation 2000,2004 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely * for the purpose for which it was furnished by NEC Electronics Corporation. No part * of thisProgram may be reproduced or disclosed to others, in any form, * without the prior written permission of NEC Electronics Corporation. */ * スロット・プログラム (uPD784035 用) /* コンパイラ定義 */ #pragma El /* 特殊機能レジスタ名(SFR 名)の有効化 */ #pragma sfr /* INTP0, INTP1, INTP2 それぞれの割り込み関数として, * 関数 stp_btn_Left(), stp_btn_Center(), stp_btn_Right()を設定 */ #pragma interrupt INTP0 stp_btn_Left #pragma interrupt INTP1 stp_btn_Center #pragma interrupt INTP2 stp_btn_Right #include "interrupt_func.h" /*割り込み関数の宣言 */ /* 表示(LED 点灯)位置の指定 */ unsigned char place; /* 表示(LED 点灯)用数字データ */ unsigned char num_data[10] = { 0x77, 0x24, 0x6b, 0x6d, 0x3c, 0x5d, 0x5f, 0x74, 0x7f, 0x7d, }; /* '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' */ /* ポート,割り込みレベル等のターゲット CPU 環境の初期化関数 */ void init_target(void); /* スロット表示用関数 */ void slot(void); /***** * スロット用メイン関数 表示用数字(0~9)をループさせて, LED に表示する. ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ, * 表示している数値を固定する.

(1/4)

* 関数名:main * 引数 :なし * 戻り値:なし * 使用グローバル変数: unsigned char place; ********** ****** void main(void) { /* ターゲット CPU 環境の初期化 */ init_target(); /* 表示位置指定変数の初期化 */ place = 0x15; /* 10101 : LED を一つおきに3つ点灯する. */ /* 7 セグメント LED 消灯 */ P4 = 0x00;P5 = 0x00: P4 = 0xFF;/* スロット(数字点灯)の処理 */ slot(); } /* main */ * ポート,割り込みレベル等のターゲット CPU 環境の初期化関数 * 関数名: init_target * 引数 :なし * 戻り値:なし * 使用グローバル変数 : なし ***** * * * * * * * * * * * * * * * * * void init_target(void){ /* * Port2 を 割り込みの入力に使用 * Port5 を 7 セグメント LED 点灯用に使用 * Port4 を 桁指定用に使用 */ /* Port2 は入力専用ポートの為,入力モードの設定はいらない. */ /* Port5 の全 bit を出力モードにする. */ PM5 = 0x00; /* モード・レジスタ(PM5)の全bitを出力(0)に設定 */ /* Port4 の全 bit を出力モードにする. */ PM4 = 0x00; /* モード・レジスタ(PM4)の全 bit を出力(0)に設定 */

```
(3/4)
```

```
/* Port5,Port4 を入出力ポートモードで使用するために
•
  メモリ拡張モードレジスタをポートモードに設定 */
 MM = 0 \times 00;
/*
•
  外部端子による外部割り込み要求の有効エッジは,
  外部割り込みモード・レジスタ INTMn(n=1-0)で指定する.
 本プログラムでは、外部割り込み要求に INTPO、INTP1、INTP2 を使用するため
•
  外部割り込みモード・レジスタ0(INTMO)を,それぞれ立ち上がりエッジ有効に設定する.
•
 */
 INTMO = 0x54; /* 0x54 = 0 1 0 1 0 1 XX
•
   •
.
   | | INTPO
•
•
   | INTP1
•
   •
  INTP2
•

    00: 立ち下がりエッジ

  01: 立ち上がりエッジ
•
  10: RFU(予約)
•
 11: 立ち上がり , 立ち下がり両エッジ
•
         */
 PMK0 = 0; /* INTPO の割り込みの受け付け許可 */
 PMK1 = 0; /* INTP1 の割り込みの受け付け許可 */
 PMK2 = 0; /* INTP2 の割り込みの受け付け許可 */
} /* init_target */
/*****
            *******
•
  スロット表示用関数
•
  表示用数字(0~9)をループさせて,LEDに表示する.
  ループ中に割り込みが発生すると,対応する割り込み関数が呼ばれ,
  表示している数値を固定する.
•
  関数名 : slot
  引数 :なし
•
  戻り値:なし
  使用グローバル変数:
.
•
  unsigned char place;
  unsigned int num_data[];
```

```
void slot(void) {
 /*
  * 表示用数字を 0~9 までループさせる.
  * 数字の点灯位置は , place 変数に従う .
  */
 /* 表示数値(num_data)用インデックス */
 int num_idx = 0;
 /* 割り込み許可 */
 EI();
 while (1) { /* 無限ループ */
   /* 数字表示 */
   P4 = 0x00;
   P5 = num_data[num_idx];
   P4 = place;
   num_idx++ ;
   /* num_dataの要素は10個なので,添字を表すnum_idxが10になったら,
   * インデックスの値は0に戻す必要がある. */
   if( num_idx >= 10 ) {
      num_idx = 0;
   }
 } /* While */
} /* slot */
```

[interrupt_func.h]

/*
* Copyright (C) NEC Electronics Corporation 2000,2004
* All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely
* for the purpose for which it was furnished by NEC Electronics Corporation. No part
* of thisProgram may be reproduced or disclosed to others, in any form,
* without the prior written permission of NEC Electronics Corporation.
*/
/* スロットの各桁の表示を止める関数 */
/* __interrupt 関数修飾子によって,
* stp_btn_Left(), stp_btn_Center(), stp_btn_Right()が,
* 割り込み関数であることを宣言する.
*/
__interrupt void stp_btn_Left(void);
__interrupt void stp_btn_Right(void);
__interrupt void stp_btn_Right(void);
__interrupt void stp_btn_Right(void);
__interrupt void stp_btn_Right(void);

付録

[interrupt_func.c]

```
/*
 * Copyright (C) NEC Electronics Corporation 2000,2004
 * All rights reserved by NEC Electronics Corporation. ThisProgram must be used solely
 * for the purpose for which it was furnished by NEC Electronics Corporation. No part
 * of thisProgram may be reproduced or disclosed to others, in any form,
  without the prior written permission of NEC Electronics Corporation.
*/
#include "interrupt_func.h"
extern unsigned char place; /* 点灯位置の指定 */
/* 7 セグメント LED は,
 * 桁信号に割り当てた端子(本ソースでは port5)の出力がアクティブ時に,
 * 対応する LED を点灯/消灯させることができる.
 * 接続端子の出力情報を受け取り,その値に従って表示し,値が変化するまで
 * 点灯し続ける.
 */
void stp_btn_Left(void) {
 /*
  * 左端の桁の表示を,現在の数値のまま固定する.
      (入出力パネル上では,数値の動きが止まったように見える.)
  */
 /* 表示位置指定変数(place)の,
  * 表示を止めたい(固定したい)桁のビットを0にする.*/
  place = place & 0xEF; /* 0xEF = 1110 1111 */
}
void stp_btn_Center(void) {
 /* 中央の桁の表示を,現在の数値のまま固定する.*/
 place = place & 0xFB; /* 0xFB = 1111 1011 */
}
void stp_btn_Right(void) {
 /* 右端の桁の表示を,現在の数値のまま固定する.*/
 place = place & 0xFE; /* 0xFE = 1111 1110 */
}
```
[メ モ]

【発 行】

NECエレクトロニクス株式会社

〒211-8668 神奈川県川崎市中原区下沼部1753 電話(代表):**044(435)5111**

----- お問い合わせ先--

【ホームページ】

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

【営業関係,技術関係お問い合わせ先】

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

【資料請求先】

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