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。

H8/300L SLP シリーズ

MAXIM ΣΔAD 変換(16 ビット)接続例

要旨

16 ビット MAX1408ΣΔAD 変換結果を、クロックを与えてシリアルで H8/38024CPU に取り込んで、4 桁の 16 進数で 7 セグメント LED に表示します。

動作確認デバイス

H8/300L Super Low Power シリーズ　H8/38024CPU

目次

1. 仕様 2
2. 使用機能説明 6
3. 動作原理 9
4. ソフトウェア説明 11
5. フローチャート 15
6. プログラムリスト 25

1. 仕様

1. 図 1 にシリアル出力アナログデジタルコンバータ（以下 ADC と称します）接続例のハードウエア構成を示します。ADC のアナログ入力端子 0（IN0 端子）に入力されたアナログ電圧を 16 ビットデルタシグマ変調器で A/D 変換します。
2. 変換結果は、マイコンから外部シリアルクロックを印加して同期式シリアル通信でマイコンに取り込みます。
3. 7 セグメント LED 表示は、受信したの 16 ビットデータを 16 進数 4 桁に変換して表示します。

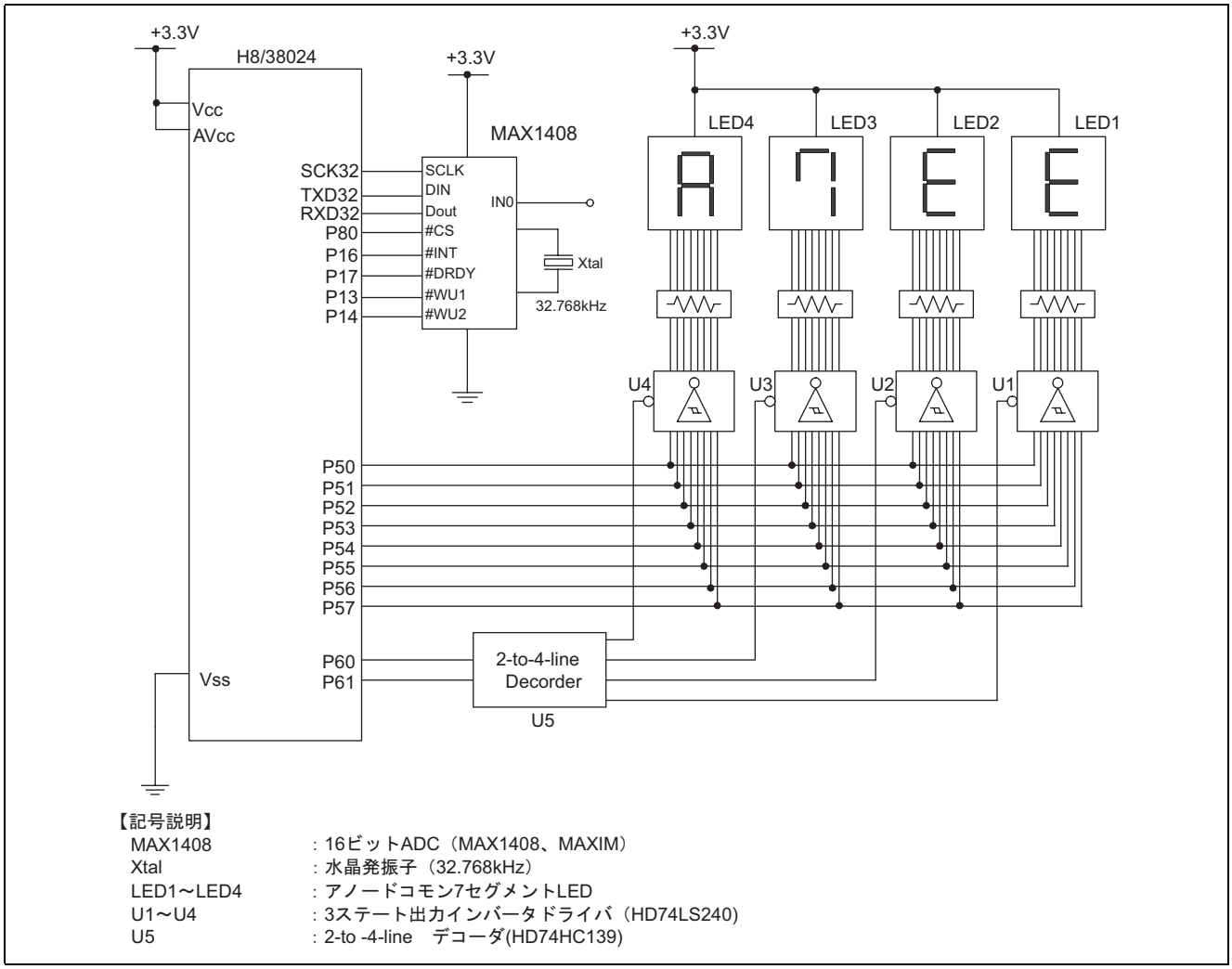


図 1 ハードウエア構成

4. 本タスク例における H8/38024 の動作電圧(Vcc)およびアナログ電源電圧(AVcc)は 3.3V、OSC クロック周波数は 10MHz、ウオッチクロック周波数は 32.768kHz です。
5. 本タスク例で使用している ADC は、MAXIM 製シリアル出力アナログデジタルコンバータ(型名：MAX1408)です。以下に仕様を示します。
   A. MAX1408 の特徴を示します。
      a. 電源電圧：+2.7V～+3.6V
      b. 消費電流：1.15mA（スリープモード時 2.5μA）
      c. パッケージ：SSOP28pin
      d. マルチチャンネル 16 ビットシグマデルタ ADC

   B. MAX1418 はデルタシグマ変調器を使用し、A/D 変換を行います。分解能が 16 ビットと比較的高いので、医療器具、工業制御機器、ポータブル機器、自動計測、ロボットなどに応用されています。

6. 本タスク例の動作は以下の通りです。
   A. MAX1418 のアナログ入力端子 0（IN0 端子）に入力された電圧を、外付けの水晶発振子のクロックを用いてデルタシグマ変調します。
   B. 16 ビット A/D に変換されたデータをシリアルクロックを与えてマイコンに取り込みます。
   C. 取り込んだ 16 ビットのデータを 16 進数 4 桁に変換して LED に表示します。
   D. 例えば、IN0 端子に電圧 0.82V が印加された場合、A/D 変換後の 16 ビットデータは「1010 0111 1110 1110」で、マイコンに取り込んで 16 進数に変換して LED に表示すると「A7EE」となります。
   E. 基準電圧の 1.25V の LED 表示は「FFFF」で、0V の LED 表示は「0000」となります。

7. 本タスク例では、７セグメント LED を表示させるためにポート出力を 3 ステート出力インバータドライバ(HD74LS240)に入力して、ドライバの出力を 7 セグメント LED のカソードに接続しています。また 4 個の 7 セグメント LED を表示させるためのポートはすべての 7 セグメント LED に接続されており、7 セグメント LED の表示切替は、3 ステートインバータドライバのイネーブル端子により制御しています。また、7 セグメント LED の表示切替を行うための信号生成は、2-to-4-line デコーダ(HD74HC139)を使用して、2 本のポート出力により制御します。図 2 に 7 セグメント LED 制御方法について示します。

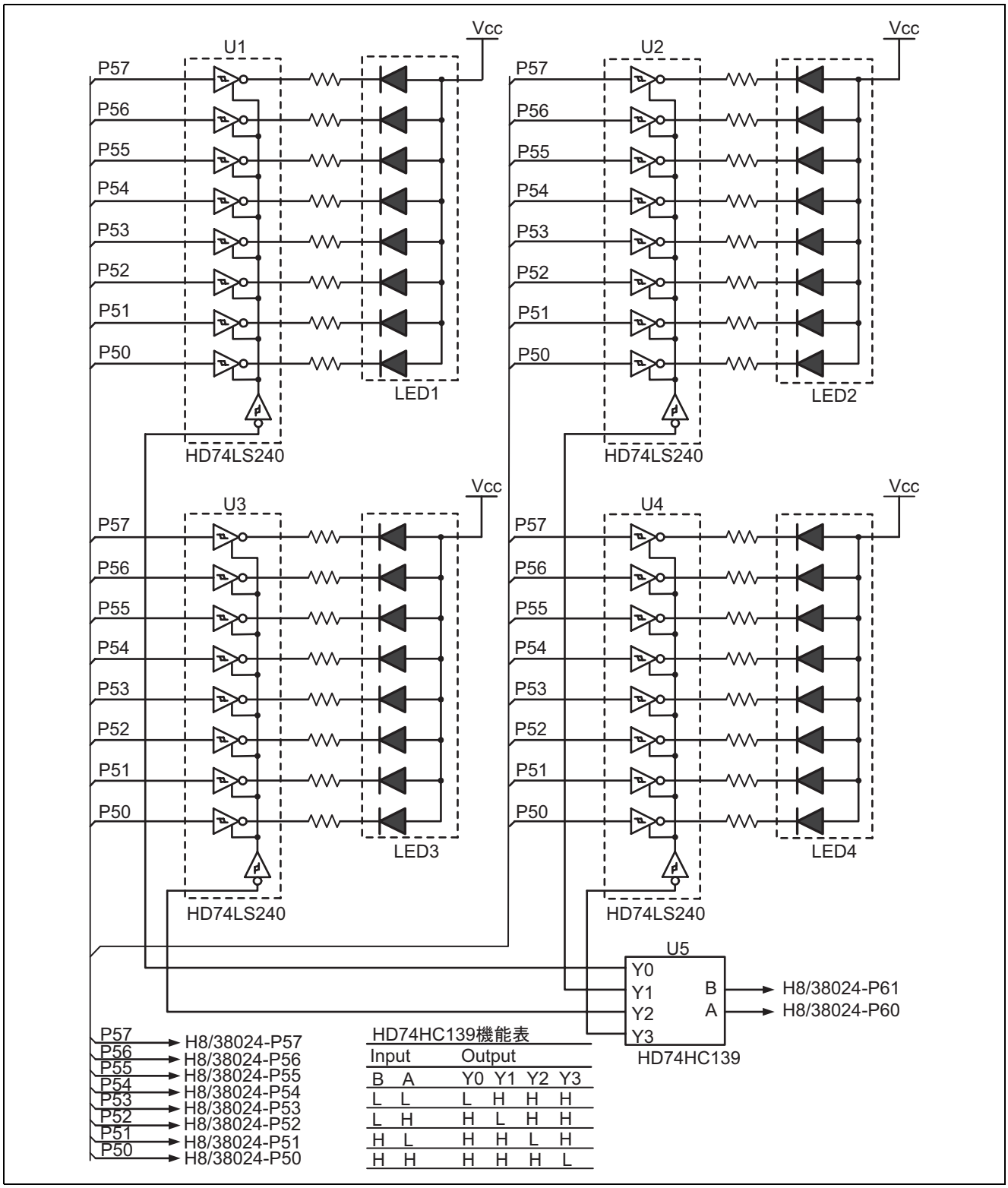


図 2　7 セグメント LED 制御方法

8. 本タスク例では、MAX1408 Σ ΔAD 変換結果である 16 ビットデータを 7 セグメント LED に 4 桁の 16 進数で表示させます。図 3 に MAX1408 Σ ΔAD 変換結果の LED 表示方法を示します。

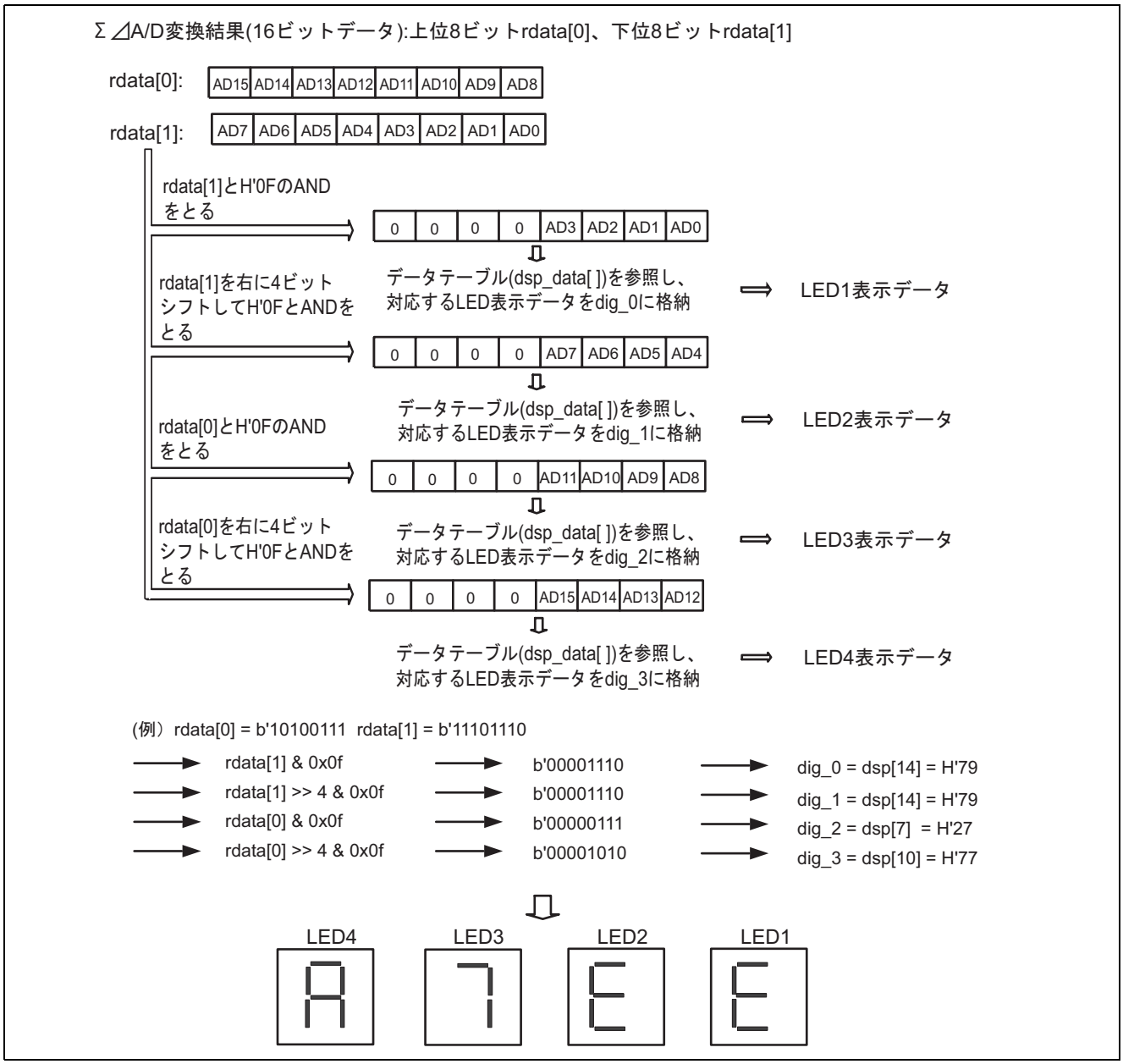


図 3　MAX1408 Σ ΔAD 変換結果の LED 表示方法

2. 使用機能説明

1. 図 4 に本タスク例における H8/38024 の使用機能のブロック図を、表 1 に機能割付けを示します。

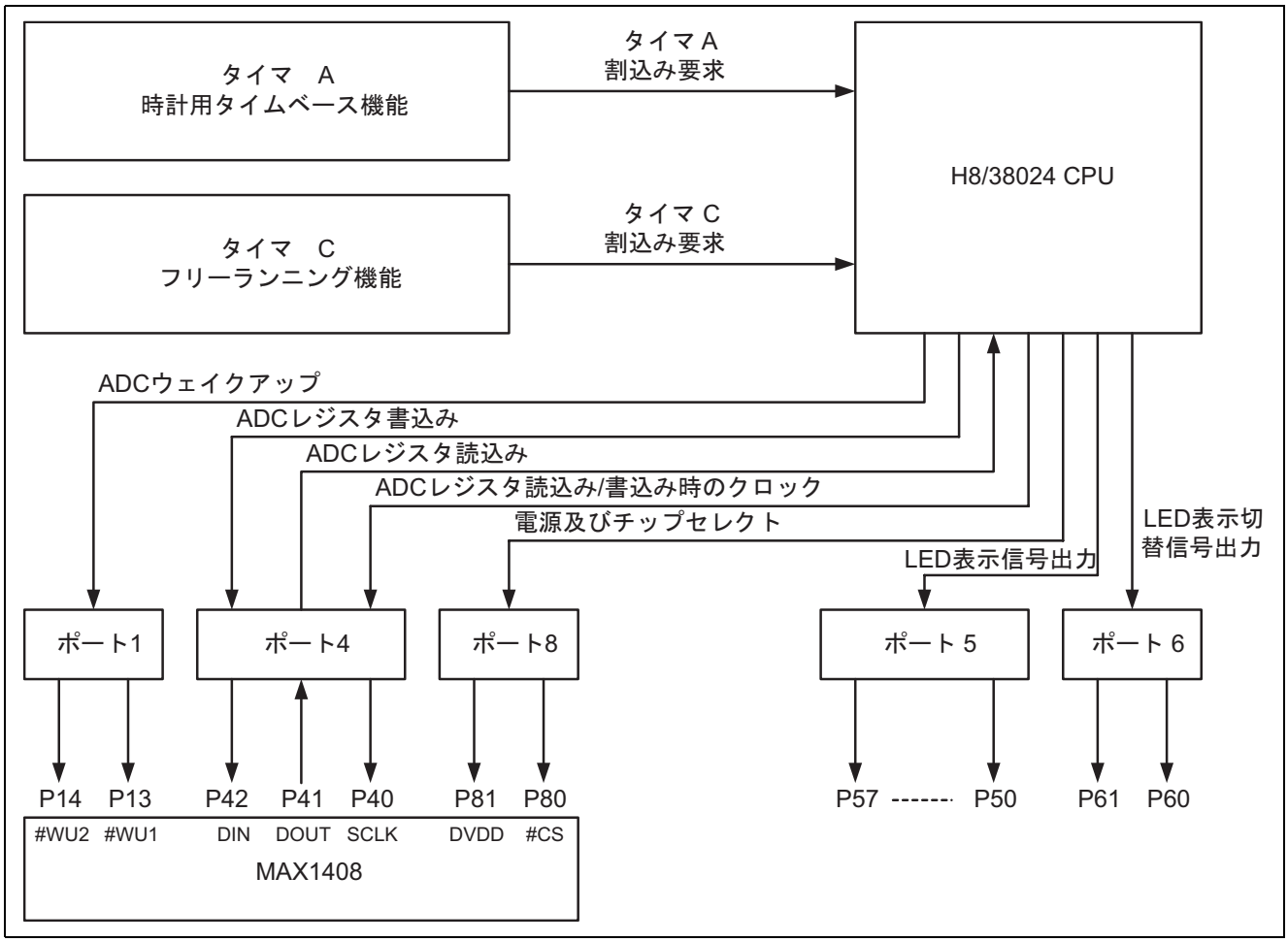


図 4　使用機能ブロック図

表 1　機能割付け

使用機能	機能割付け
タイマ A	タイマ A 時計用タイムベース機能使用して、MAX1408 ΣΔAD 変換結果である DATA レジスタの値を LED 表示データに変換し RAM に格納します。
タイマ C	タイマ C フリーランニング機能を使用して 7 セグメント LED の表示切替制御を行います。タイマ C オーバフロー周期 3.2768ms 毎に 4 個の 7 セグメント LED を順番に点灯させることによるダイナミック点灯を行います。
ポート 1	ポート 1 の P13、P14 出力端子により、MAX1408 ΣΔADC ウェイクアップを行います。
ポート 4	ポート 4 の P42 出力端子により、MAX1408 ΣΔADC レジスタへの書込みを行います。 ポート 4 の P40 出力端子により、MAX1408 ΣΔADC レジスタ読込み/書込み時のクロックにセットします。 ポート 4 の P41 入力端子により、MAX1408 ΣΔADC レジスタからの読込みを行います。
ポート 5	ポート 5 の P50～P57 出力端子により、7 セグメント LED の表示を行います。MAX1408 ΣΔAD 変換結果である 16 ビットデータを 4 桁の 16 進数表示データに変換して LED に出力します。
ポート 6	ポート 6 の P60、P61 出力端子により、4 個の 7 セグメント LED の表示切替を行います。P60、P61 出力端子は 2−to−4−line デコーダの入出端子に接続されています。
ポート 8	ポート 8 の P80、P81 出力端子により、MAX1408 ΣΔADC の電源及びチップセレクトにセットします。

2. 使用する 7 セグメント LED の接続図を図 5 に示します。図 5 に示すようにポート 5 から”High”を出力することにより対応する LED のセグメントが点灯します。また、ポート 5 出力と LED 表示データの関係を表 2 に示します。

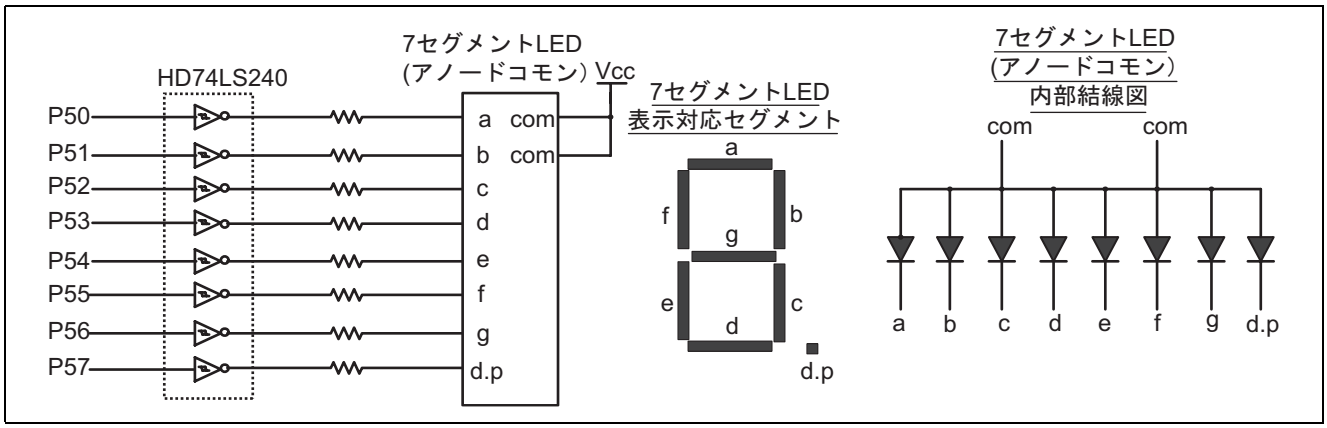
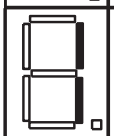
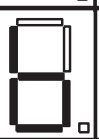
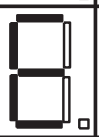
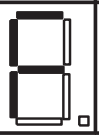
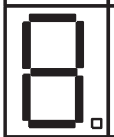


図 5　7 セグメント LED 接続図および内部結線図

表 2　ポート 5 出力と 7 セグメント LED 表示データの関係

LED 表示	ポート5出力データ								LED 表示	ポート5出力データ							
	P57	P56	P55	P54	P53	P52	P51	P50		P57	P56	P55	P54	P53	P52	P51	P50
	0	0	1	1	1	1	1	1		0	1	1	1	0	1	1	1
	0	0	0	0	0	1	1	0		0	1	1	1	1	1	0	0
	0	1	0	1	1	0	1	1		0	0	1	1	1	0	0	1
	0	1	0	0	1	1	1	1		0	1	0	1	1	1	1	0
	0	1	1	0	0	1	1	0		0	1	1	1	1	0	0	1
	0	1	1	0	1	1	0	1		0	1	1	1	0	0	0	1
	0	1	1	1	1	1	0	1									
	0	0	1	0	0	1	1	1									
	0	1	1	1	1	1	1	1									
	0	1	1	0	1	1	1	1									

3. 動作原理

1. 図 6 にタイマ A を使用した、MAX1408ΣΔAD 変換結果である 16 ビット DATA レジスタの値を、LED 表示データに変換し RAM に格納を示します。図 6 に示すように、本タスク例では、tmra ルーチンの中で、タイマ A 割込み周期より、16 ビット MAX1408ΣΔAD 変換結果を LED 表示データに変換し RAM に格納します。

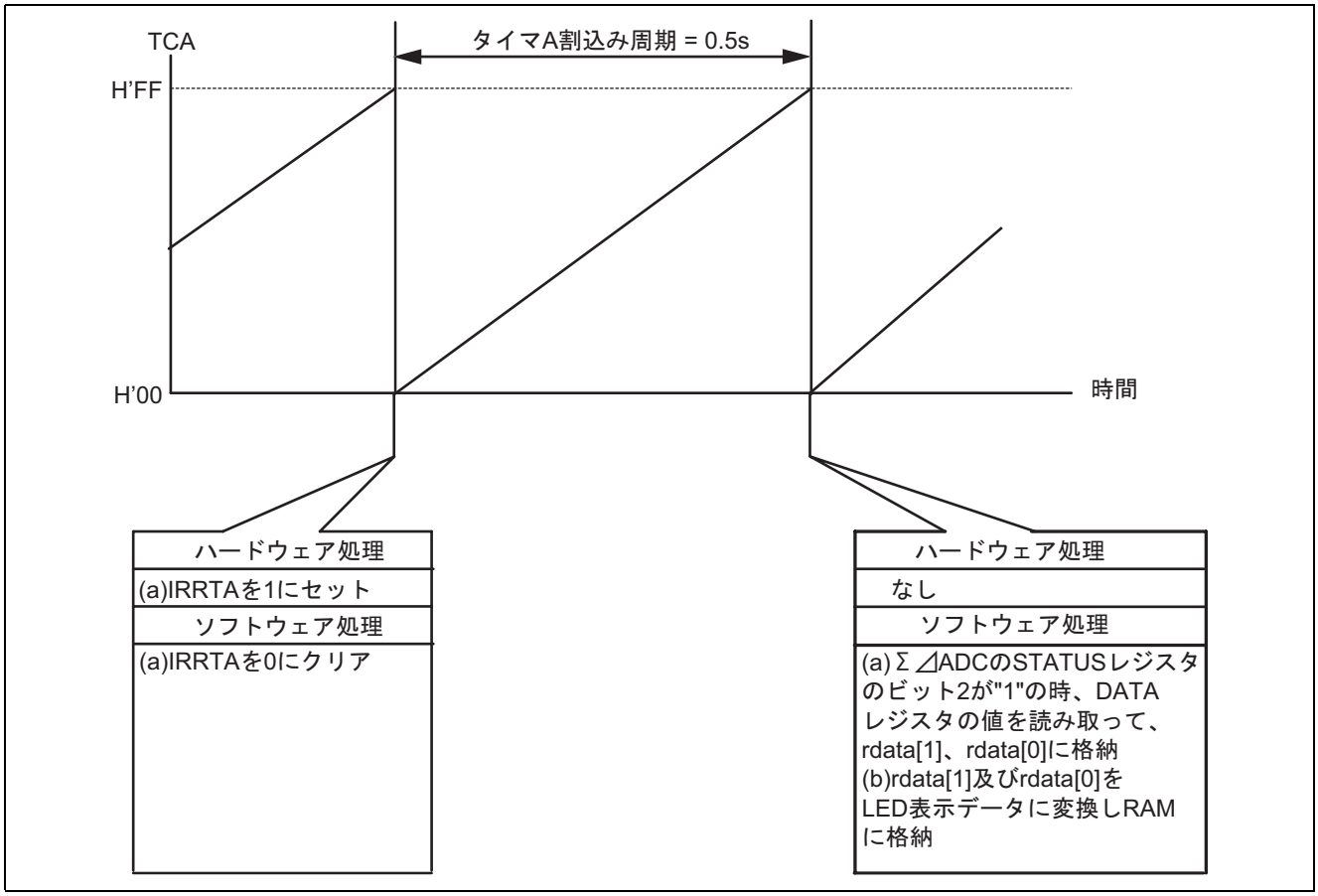


図 6　タイマ A を使用しΣΔA/D 変換結果を RAM に格納する動作原理

2. 7 セグメント LED の表示制御の動作原理について説明します。図 7 は LED4～LED1 に”A7EE”を表示する場合の動作原理について説明しています。図 7 に示すようにタイマ C オーバフロー周期ごとに LED1～LED4 を順番に表示させることにより 7 セグメント LED のダイナミック表示を行っています。

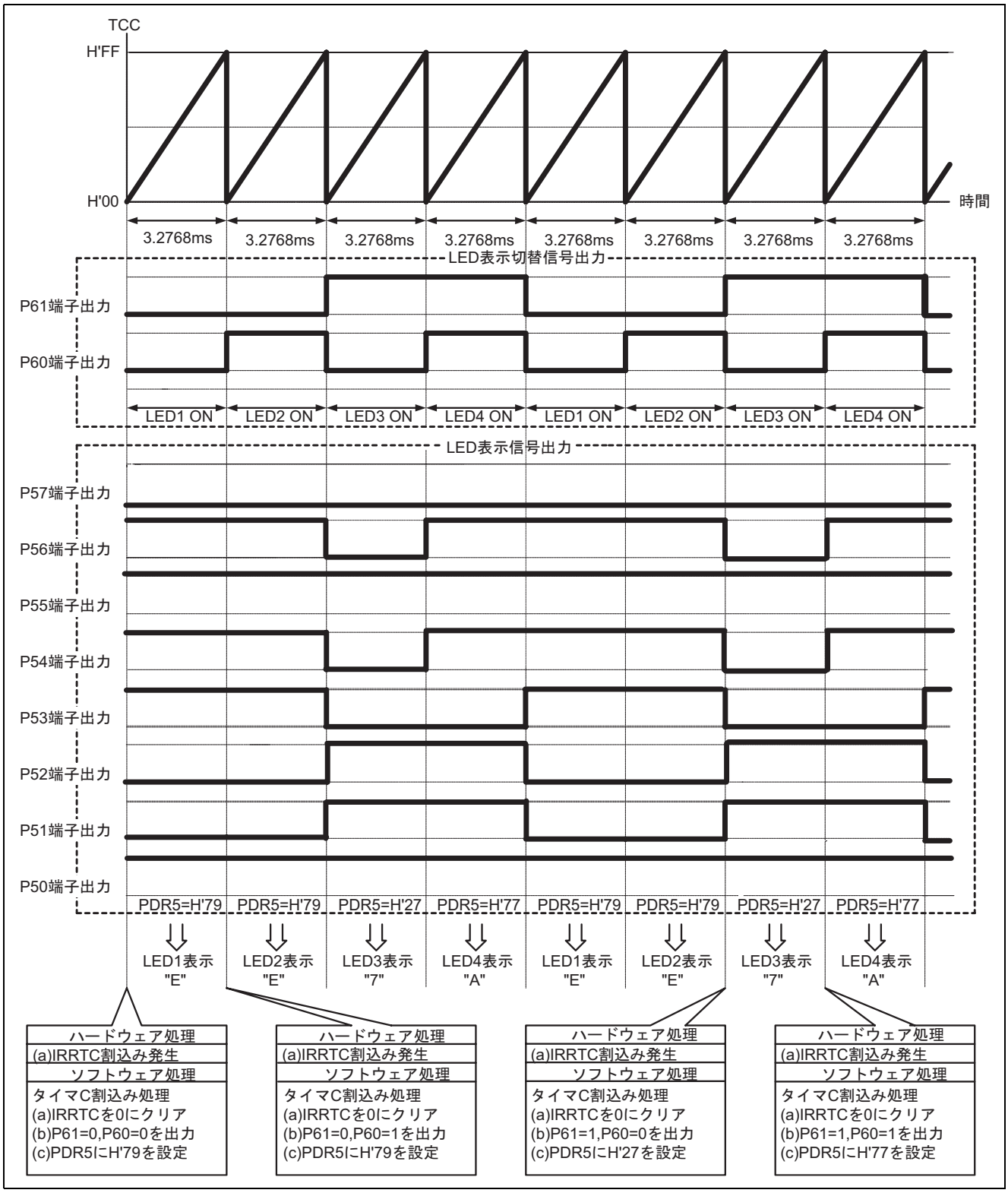


図 7　7 セグメント LED 表示制御の動作原理

4. ソフトウェア説明

1. モジュール説明

表 3 に本タスク例におけるモジュール説明を示します。

表 3　モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	初期設定、MAX1408 Σ Δ AD 変換器の初期設定関数を呼び出し、割込み許可
タイマ A 割込み処理ルーチン	tmra	割込みフラグのクリア、STUTAS レジスタのビット 2 は"1"になってから、MAX1408 Σ Δ AD 変換結果である DATA レジスタの値を LED 表示データに変換し RAM に格納
タイマ C 割込み処理ルーチン	tmrc	割込みフラグのクリア、LED 表示データの出力と LED 表示切替の制御
ADC_Init( )処理ルーチン	ADC_Init	MAX1408 Σ Δ AD 変換器の初期設定
DataIn( )処理ルーチン	DataIn	MAX1408 Σ Δ AD 変換器のレジスタへの書込み及び読込み
DataOut( )処理ルーチン	DataOut	MAX1408 Σ Δ AD 変換器のレジスタへの書込み

2. 引数説明

本タスク例では、引数を使用しておりません。

3. 使用内部レジスタ説明

本タスク例の使用内部レジスタを表 4 に示します。

表 4　使用内部レジスタ説明

レジスタ名		機能説明	アドレス	設定値
TMA		タイマモードレジスタ A ：プリスケーラ、入力クロックの選択	H'FFB0	H'0C （初期設定時）
	TMA3	インターナルクロックセレクト 3 ：タイマ A の動作モードの選択 TMA3 = 1 設定時、 ：プリスケーラ W の出力をカウントする時計用タイムベースとして動作	ビット 3	1
	TMA2	インターナルクロックセレクト 2～0 ：TMA3 = 1 のとき、時計用タイムベース(32.768kHz)を選択 TMA2 = 1、TMA1 = 0、TMA0 = 0 設定時、 ：TCA リセット TMA2 = 0、TMA1 = 0、TMA0 = 1 設定時、 ：TCA オーバフロー周期は 0.5s	ビット 2	0/1
	TMA1		ビット 1	0
	TMA0		ビット 0	0/1
TMC		タイマモードレジスタ C ：オートリロードの選択、カウンタのアップ／ダウン制御、入力クロックの制御	H'FFB4	H'1B
	TMC7	オートリロード機能の選択 ：TMC7 = 0 のとき、インターバル機能を選択	ビット 7	0
	TMC6	カウンタアップ／ダウン制御 ：TMC6 = 0、TMC5 = 0 のとき、TCC はアップカウンタ	ビット 6	0
	TMC5		ビット 5	0
	TMC2	クロックセレクト ：TMC2 = 0、TMC1 = 1、TMC0 = 1 のとき、内部クロック φ/64 でカウント	ビット 2	0
	TMC1		ビット 1	1
	TMC0		ビット 0	1
TLC		タイマロードレジスタ C ：TCC のリロード値を設定	H'FFB5	H'00

レジスタ名		機能説明	アドレス	設定値
CKSTRP1		クロック停止レジスタ 1 ：ビット 5：SCI3 をモジュールスタンバイモード制御(S32CKSTP)、1 のとき SCI3 のモジュールスタンバイモードは解除される	H'FFFA	H'FF
PDR1		ポートデータレジスタ 1	H'FFD4	H'00
PCR1		ポートコントロールレジスタ 1 ：ポート 1 の各端子 P17、P16、P14、P13 の入出力をビットごとに制御 PCR1 = H'3F のとき、 ：P14、P13 端子は出力端子として機能	H'FFE4	H'3F
PUCR1		ポートプルアップコントロールレジスタ 1 ：ポートの各端子 P17、P16、P14、P13 のプルアップ MOS をビットごとに制御 PUCR1 = H'00 のとき、 ：P17、P16、P14、P13 端子のプルアップ MOS はオフ	H'FFE0	H'00
PMR1		ポートモードレジスタ 1	H'FFC8	H'42
	IRQ3	P17/IRQ3/TMIF 端子機能切り替え ：0 のとき P17 汎用入出力ポート機能	ビット 7	0
	IRQ4	P14/IRQ4/ADTRG 端子機能切り替え ：0 のとき P14 汎用入出力ポート機能	ビット 4	0
	TMIG	P13/TMIG 端子機能切り替え ：0 のとき P13 汎用入出力ポート機能	ビット 3	0
PMR2		ポートモードレジスタ 2 ：P35 端子の PMOS の ON/OFF、ウォッチドッグタイマクロック選択、TMIG ノイズキャンセルセレクト、P43/IRQ0 端子機能の切り替えを制御	H'FFC9	H'D8
PDR4		ポートデータレジスタ 4	H'FFD7	H'F8
PCR4		ポートコントロールレジスタ 4 ：ポート 4 の各端子 P42～P40 の入出力をビットごとに制御 PCR4 = H'FD のとき、 ：P42、P40 端子は出力端子として機能、P41 端子は入力端子として機能	H'FFE7	H'FD
PUCR6		ポートプルアップコントロールレジスタ 6 ：入力ポートに設定されたポート 6 の各端子のプルアップ MOS をビットごとに制御 PUCR6 = H'00 のとき、 ：P67～P60 端子のプルアップ MOS はオフ	H'FFE3	H'00
PDR6		ポートデータレジスタ 6 ：ポート 6 の汎用入出力ポートデータレジスタ	H'FFD9	H'00
PCR6		ポートコントロールレジスタ 6 ：ポート 6 の汎用入出力ポートとして使用する端子の入出力をビットごとに選択 PCR6 = H'FF のとき、 ：P67～P60 端子は汎用出力端子として機能	H'FFE9	H'FF
PMR5		ポートモードレジスタ 5 ：ポート 5 の端子機能を設定	H'FFCC	H'00
	WKP7	P57/_WKP7/SEG7 端子機能切り替え ：0 のとき P57 汎用入出力ポート機能	ビット 7	0
	WKP6	P56/_WKP6/SEG6 端子機能切り替え ：0 のとき P56 汎用入出力ポート機能	ビット 6	0

レジスタ名		機能説明	アドレス	設定値
	WKP5	P55/_WKP5/SEG5 端子機能切り替え ：0 のとき P55 汎用入出力ポート機能	ビット 5	0
	WKP4	P54/_WKP4/SEG4 端子機能切り替え ：0 のとき P54 汎用入出力ポート機能	ビット 4	0
	WKP3	P53/_WKP3/SEG3 端子機能切り替え ：0 のとき P53 汎用入出力ポート機能	ビット 3	0
	WKP2	P52/_WKP2/SEG2 端子機能切り替え ：0 のとき P52 汎用入出力ポート機能	ビット 2	0
	WKP1	P51/_WKP1/SEG1 端子機能切り替え ：0 のとき P51 汎用入出力ポート機能	ビット 1	0
	WKP0	P50/_WKP0/SEG0 端子機能切り替え ：0 のとき P50 汎用入出力ポート機能	ビット 0	0
PUCR5		ポートプルアップコントロールレジスタ 5 ：入力ポートに設定されたポート 5 の各端子のプルアップ MOS をビットごとに制御 PUCR5 = H'00 のとき、 ：P57～P50 端子のプルアップ MOS はオフ	H'FFE2	H'00
PDR5		ポートデータレジスタ 5 ：ポート 5 の汎用入出力ポートデータレジスタ	H'FFD8	H'00
PCR5		ポートコントロールレジスタ 5 ：ポート 5 の汎用入出力ポートとして使用する端子の入出力をビットごとに選択 PCR5 = H'FF のとき、 ：P57～P50 端子は汎用出力端子として機能	H'FFE8	H'FF
PDR8		ポートデータレジスタ 8 ：ポート 8 の汎用入出力ポートデータレジスタ	H'FFDB	H'00
PCR8		ポートコントロールレジスタ 8 ：ポート 8 の汎用入出力ポートとして使用する端子の入出力をビットごとに選択 PCR8 = H'FF のとき、 ：P87～P80 端子は汎用出力端子として機能	H'FFEB	H'FF
IENR1		割込み許可レジスタ 1 ：割り込み要求の許可／禁止を制御	H'FFF3	-
	IENTA	タイマ A 割り込み要求イネーブル ：1 のとき、タイマ A のオーバフロー割り込み要求を許可	ビット 5	1
IRR1		割り込み要求レジスタ 1 ：タイマ A、IRQ4、IRQ3、IRQAEC、IRQ1、IRQ0 割り込み要求が発生すると対応するフラグが 1 にセットされる	H'FFF6	-
	IRRTA	タイマ A 割り込み要求フラグ ：タイマ A のカウンタ値がオーバフロー(H'FF→H'00)したときに 1 にセット ：IRRTA に 0 をライトしたときに 0 クリア	ビット 7	0/1
IENR2		割込み許可レジスタ 2 ：割り込み要求の許可／禁止を制御	H'FFF4	-
	IENTC	タイマ C 割り込み要求イネーブル ：1 のとき、タイマ C のオーバフローまたはアンダーフロー割り込み要求を許可	ビット 1	1

レジスタ名		機能説明	アドレス	設定値
IRR2		割り込み要求レジスタ 2 ：直接遷移、A/D 変換器、タイマ G、タイマ FH、タイマ FL、タイマ C、非同期イベントカウンタ割り込み要求が発生すると対応するフラグが 1 にセットされる	H'FFF7	-
	IRRTC	タイマ C 割り込み要求フラグ ：タイマ C のカウンタ値がオーバフロー(H'FF→H'00)、またはアンダーフロー(H'00→H'FF)したときに 1 にセット ：IRRTC に 0 をライトしたときに 0 クリア	ビット 7	0/1

4. 使用 RAM 説明

表 5 に本タスク例における使用 RAM 説明を示します。

表 5　使用 RAM 説明

ラベル名	機能	アドレス	使用モジュールラベル名
ptr	Dig_0 のアドレスを格納するポイント	H'FB82	tmrc,
dig_0	LED1 の表示データを格納(1byte)	H'FB84	main, tmra, tmrc
dig_1	LED2 の表示データを格納(1byte)	H'FB85	main, tmra, tmrc
dig_2	LED3 の表示データを格納(1byte)	H'FB86	main, tmra, tmrc
dig_3	LED4 の表示データを格納(1byte)	H'FB87	main, tmra, tmrc
cnt	LED1～LED4 の表示切替のための 8 ビットカウンタ (1byte)	H'FB88	main, ttmrc
sdata	MAX1408 ΣΔAD の制御させるレジスタ値を格納	H'FB89	tmra,ADC_Init,DataIn,DataOut
rdata	MAX1408 ΣΔAD 変換結果である DATA レジスタ値を格納	H'FB8E	tmra, DataIn,

5. データテーブル説明

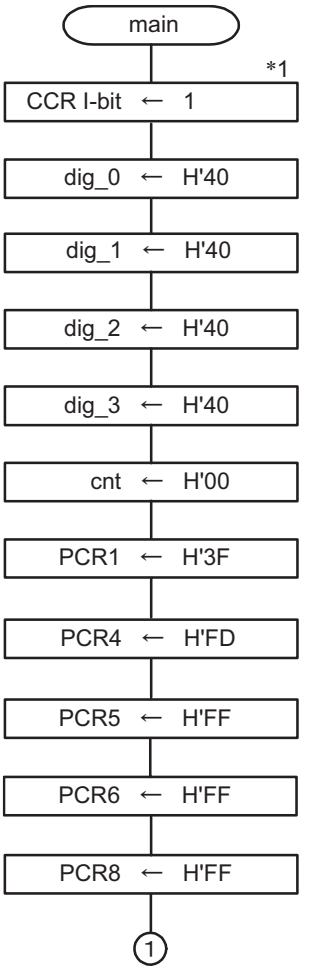
本タスク例では 7 セグメント LED の表示データを 1 次元配列のデータテーブルとして ROM に格納しています。表 6 に 7 セグメント LED 表示データテーブル(dsp_data [ ])説明を示します。

表 6　7 セグメント LED 表示データテーブル(dsp_data[])説明

配列名	データ	データ説明	データサイズ	アドレス
dsp_data[0]	H'3F	LED に"0"を表示させるためのポート 5 出力データ	1 byte	H'7B4
dsp_data[1]	H'06	LED に"1"を表示させるためのポート 5 出力データ	1 byte	H'7B5
dsp_data[2]	H'5B	LED に"2"を表示させるためのポート 5 出力データ	1 byte	H'7B6
dsp_data[3]	H'4F	LED に"3"を表示させるためのポート 5 出力データ	1 byte	H'7B7
dsp_data[4]	H'66	LED に"4"を表示させるためのポート 5 出力データ	1 byte	H'7B8
dsp_data[5]	H'6D	LED に"5"を表示させるためのポート 5 出力データ	1 byte	H'7B9
dsp_data[6]	H'7D	LED に"6"を表示させるためのポート 5 出力データ	1 byte	H'7BA
dsp_data[7]	H'27	LED に"7"を表示させるためのポート 5 出力データ	1 byte	H'7BB
dsp_data[8]	H'7F	LED に"8"を表示させるためのポート 5 出力データ	1 byte	H'7BC
dsp_data[9]	H'6F	LED に"9"を表示させるためのポート 5 出力データ	1 byte	H'7BD
dsp_data[10]	H'77	LED に"A"を表示させるためのポート 5 出力データ	1 byte	H'7BE
dsp_data[11]	H'7C	LED に"b"を表示させるためのポート 5 出力データ	1 byte	H'7BF
dsp_data[12]	H'39	LED に"C"を表示させるためのポート 5 出力データ	1 byte	H'7C0
dsp_data[13]	H'5E	LED に"d"を表示させるためのポート 5 出力データ	1 byte	H'7C1
dsp_data[14]	H'79	LED に"E"を表示させるためのポート 5 出力データ	1 byte	H'7C2
dsp_data[15]	H'71	LED に"F"を表示させるためのポート 5 出力データ	1 byte	H'7C3

5. フローチャート

1. メインルーチン(main)



CCRの割込みマスクビット(I)を1セットし、割込み要求をマスク

使用RAM領域初期設定
dig_0 = H'40 :LED1初期表示データ "–"
dig_1 = H'40 :LED2初期表示データ "–"
dig_2 = H'40 :LED3初期表示データ "–"
dig_3 = H'40 :LED4初期表示データ "–"
cnt = H'00 :LEDイネーブルカウンタ初期値 = 0

ポート1初期設定
P13/TMIG :P13出力端子(Pull-up MOS = OFF)
P14/_IRQ4/_ADTRG :P14出力端子(Pull-up MOS = OFF)
P16 :P16入力端子(Pull-up MOS = OFF)
P17/_IRQ3/TMIF :P17入力端子(Pull-up MOS = OFF)

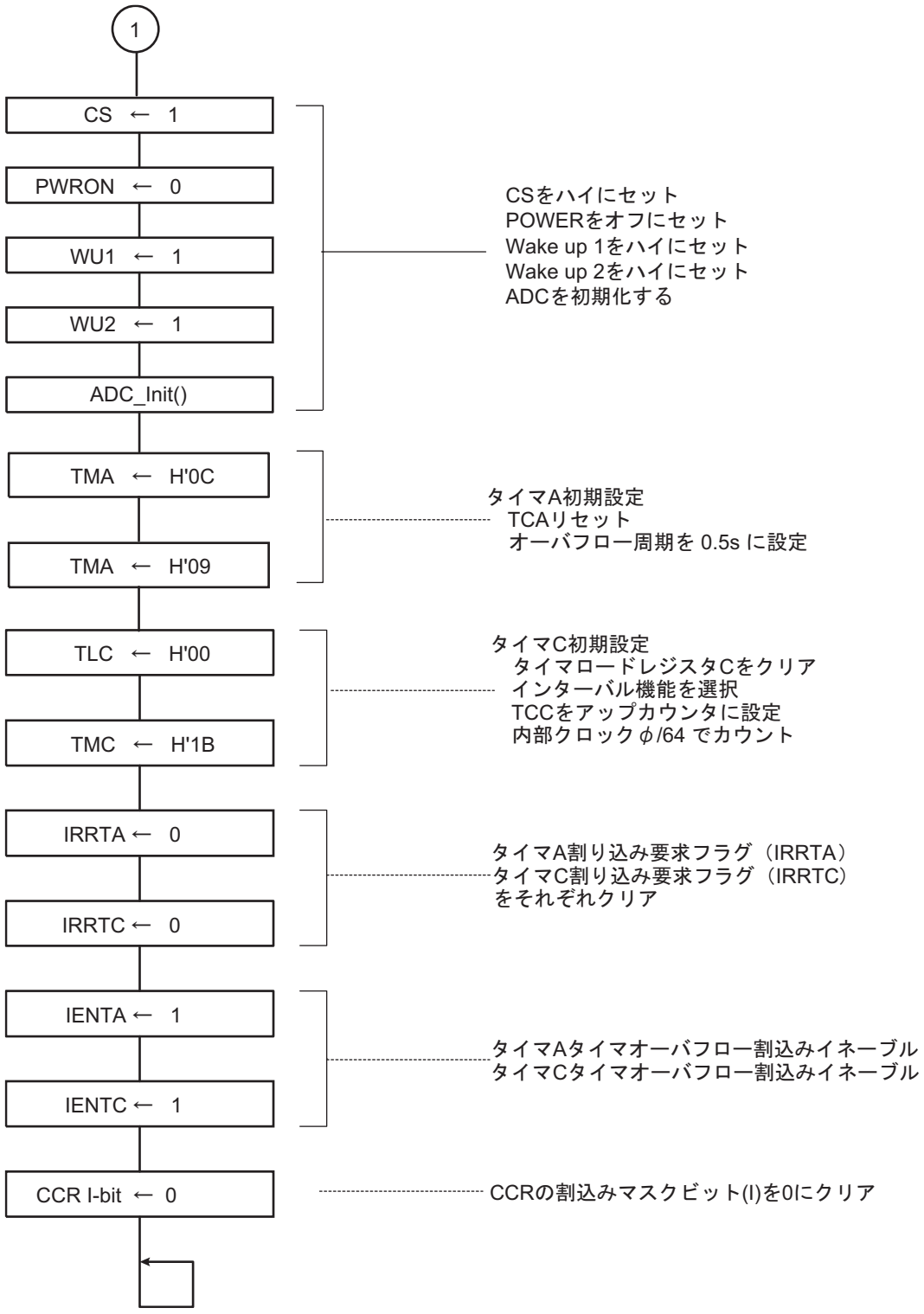
ポート4初期設定
P40/SCK32 :P40出力端子
P41/RXD32 :P41入力端子
P42/TXD32 :P42出力端子

ポート5初期設定
P50/_WKP0/SEG1 :P50出力端子（出力データ = 0)
P51/_WKP1/SEG2 :P51出力端子（出力データ = 0)
P52/_WKP2/SEG3 :P52出力端子（出力データ = 0)
P53/_WKP3/SEG4 :P53出力端子（出力データ = 0)
P54/_WKP4/SEG5 :P54出力端子（出力データ = 0)
P55/_WKP5/SEG6 :P55出力端子（出力データ = 0)
P56/_WKP6/SEG7 :P56出力端子（出力データ = 0)
P57/_WKP7/SEG8 :P57出力端子（出力データ = 0)

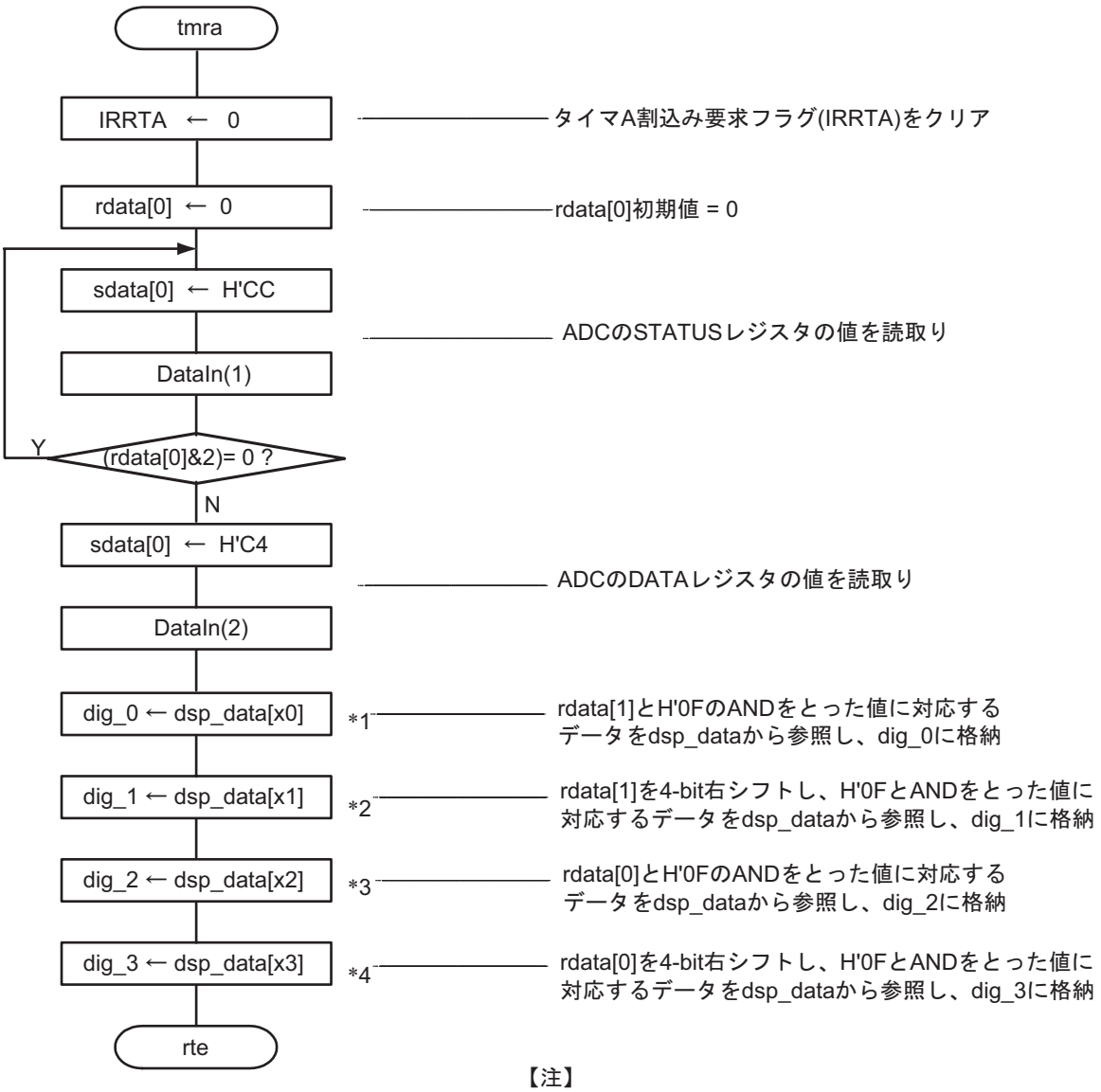
ポート6初期設定
P60/SEG9 :P60出力端子
P61/SEG10 :P61出力端子
P62/SEG11 :P62出力端子
P63/SEG12 :P63出力端子
P64/SEG13 :P64出力端子
P65/SEG14 :P65出力端子
P66/SEG15 :P66出力端子
P67/SEG16 :P67出力端子

ポート8初期設定
P80/SEG25 :P80出力端子
P81/SEG26 :P81出力端子
P82/SEG27 :P82出力端子
P83/SEG28 :P83出力端子
P84/SEG29 :P84出力端子
P85/SEG30 :P85出力端子
P86/SEG31 :P86出力端子
P87/SEG32 :P87出力端子

【注】
*1：本タスク例ではスタックポインタの設定はINIT.SRC（アセンブリ言語）で行っています。

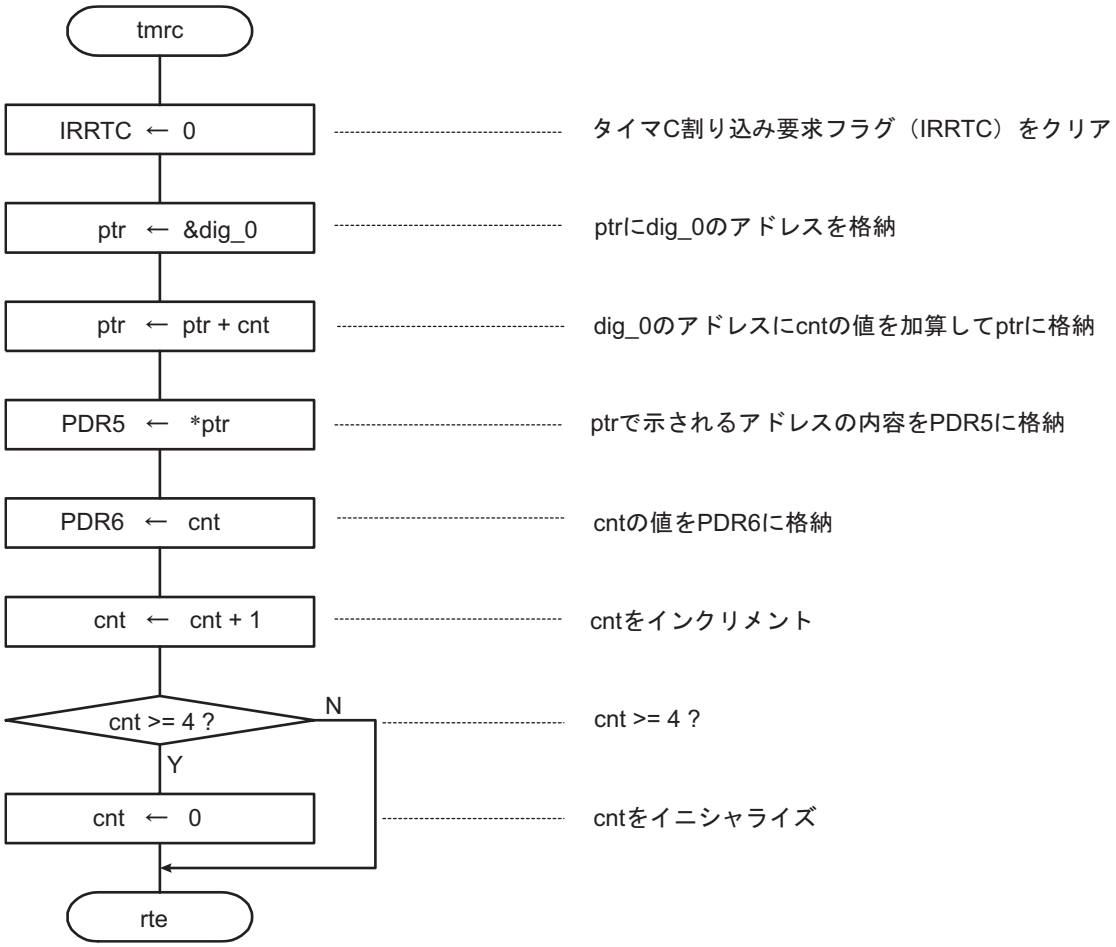

1
CS ← 1
PWRON ← 0
WU1 ← 1
WU2 ← 1
ADC_Init()
CSをハイにセット
POWERをオフにセット
Wake up 1をハイにセット
Wake up 2をハイにセット
ADCを初期化する
TMA ← H'0C
TMA ← H'09
タイマA初期設定
TCAリセット
オーバフロー周期を 0.5s に設定
TLC ← H'00
TMC ← H'1B
タイマC初期設定
タイマロードレジスタCをクリア
インターバル機能を選択
TCCをアップカウンタに設定
内部クロック φ/64 でカウント
IRRTA ← 0
IRRTC ← 0
タイマA割り込み要求フラグ（IRRTA）
タイマC割り込み要求フラグ（IRRTC）
をそれぞれクリア
IENTA ← 1
IENTC ← 1
タイマAタイマオーバフロー割込みイネーブル
タイマCタイマオーバフロー割込みイネーブル
CCR I-bit ← 0
CCRの割込みマスクビット(I)を0にクリア

2. タイマ A 割込み処理ルーチン(tmra)

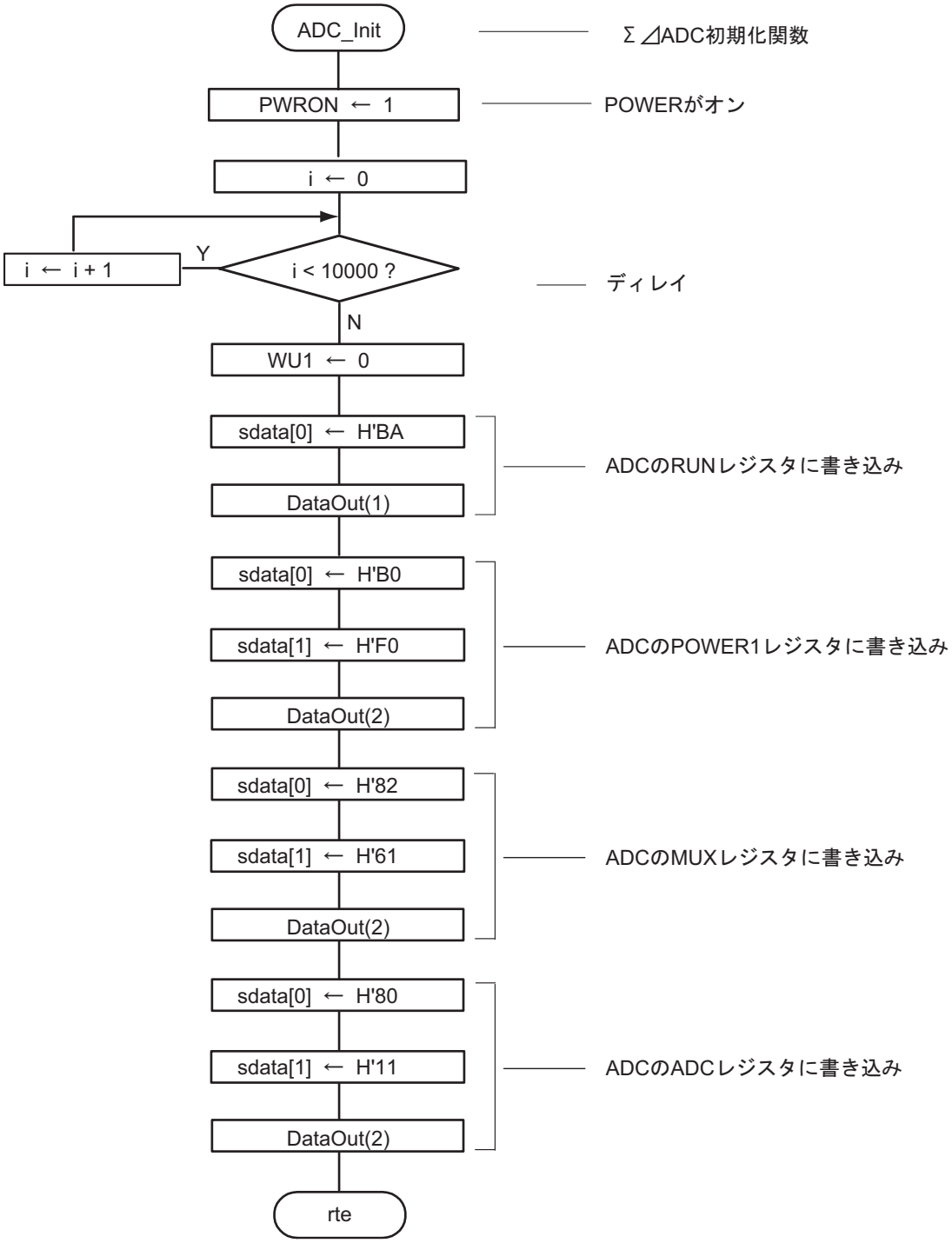


【注】
*1 : x0 = rdata[1] & H'0F
*2 : x1 = rdata[1] >> 4 & H'0F
*3 : x2 = rdata[0] & H'0F
*4 : x3 = rdata[0] >> 4 & H'0F

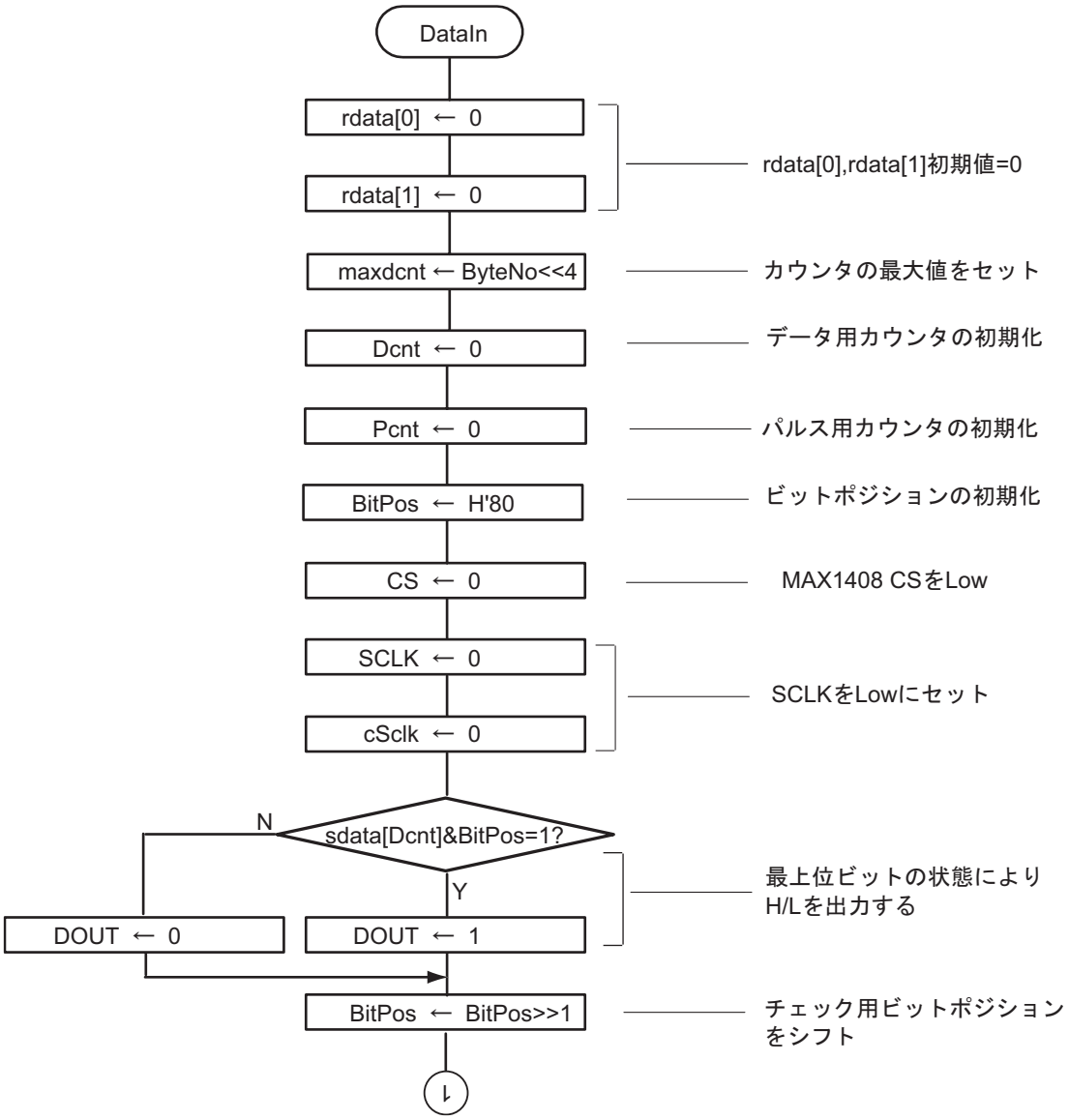
3. タイマ C 割込み処理ルーチン(tmrc)

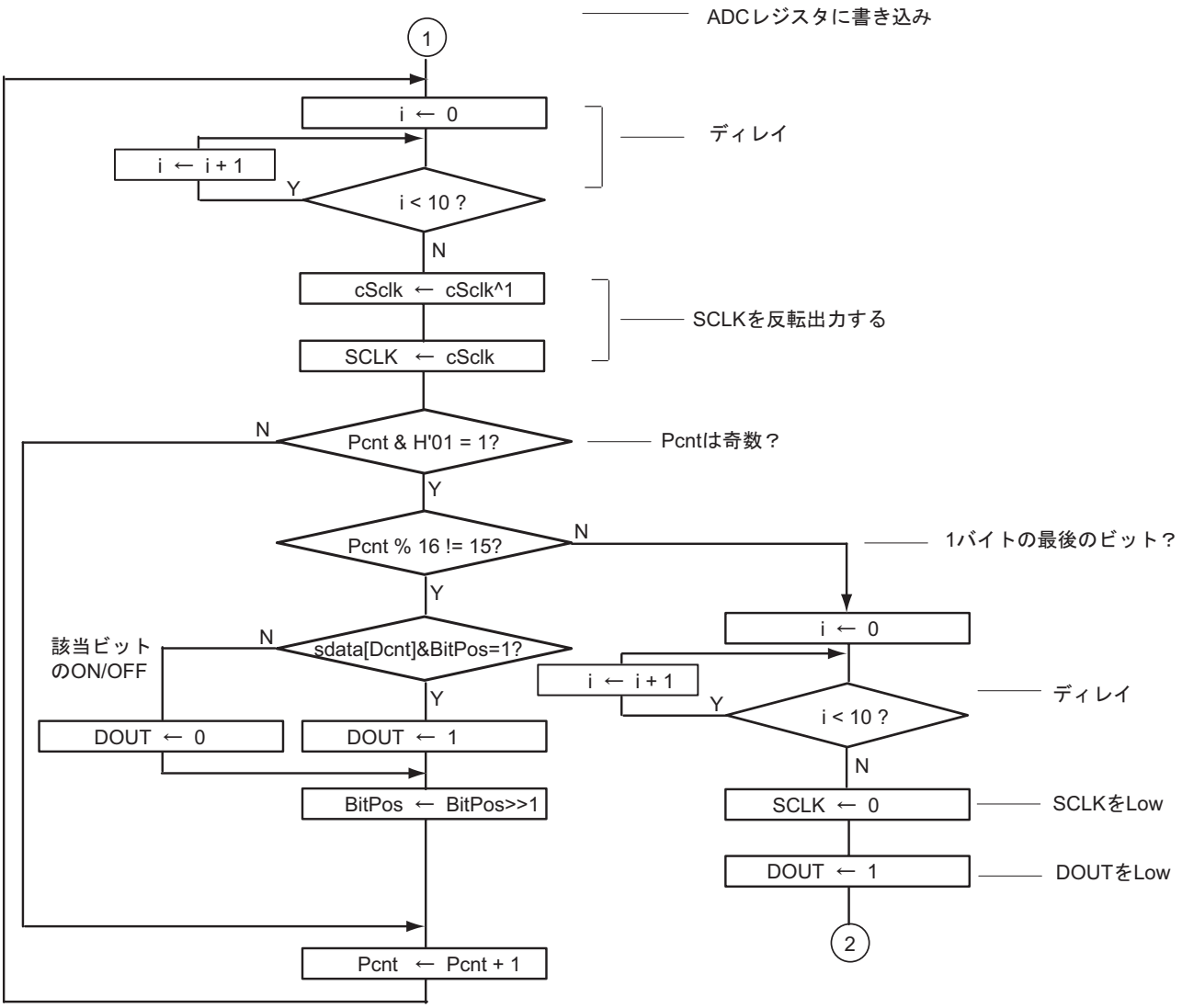


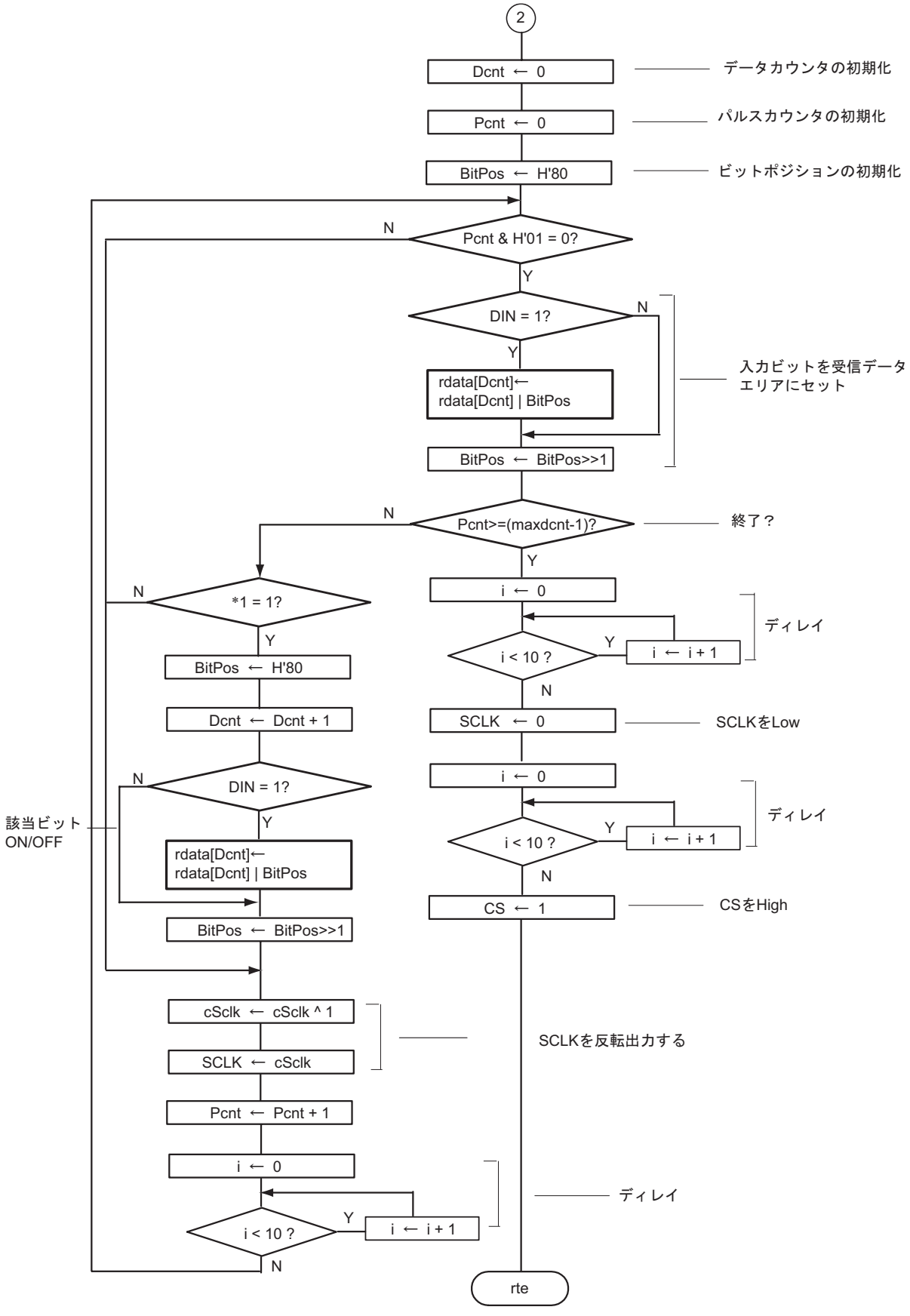
4. ADC_Init()処理ルーチン



5. DataIn()処理ルーチン

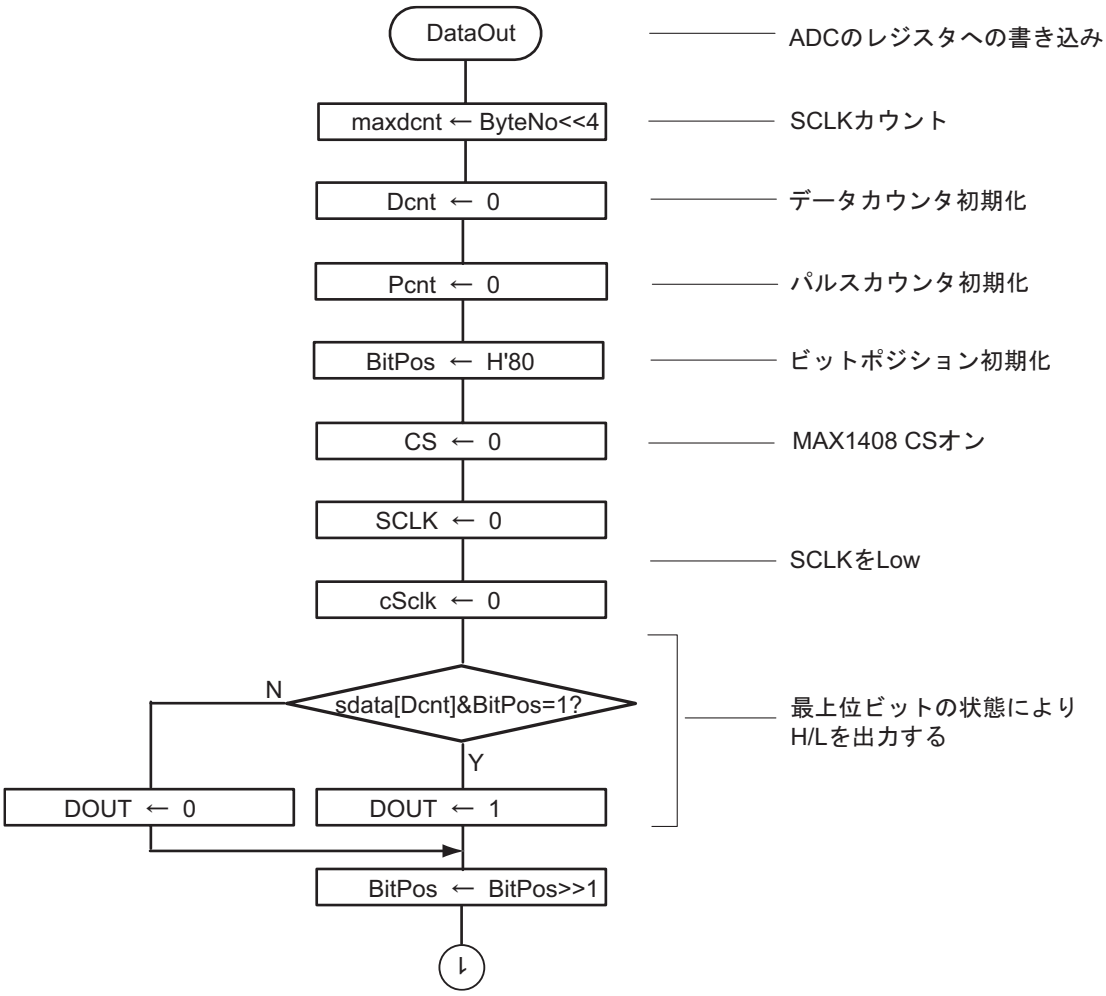


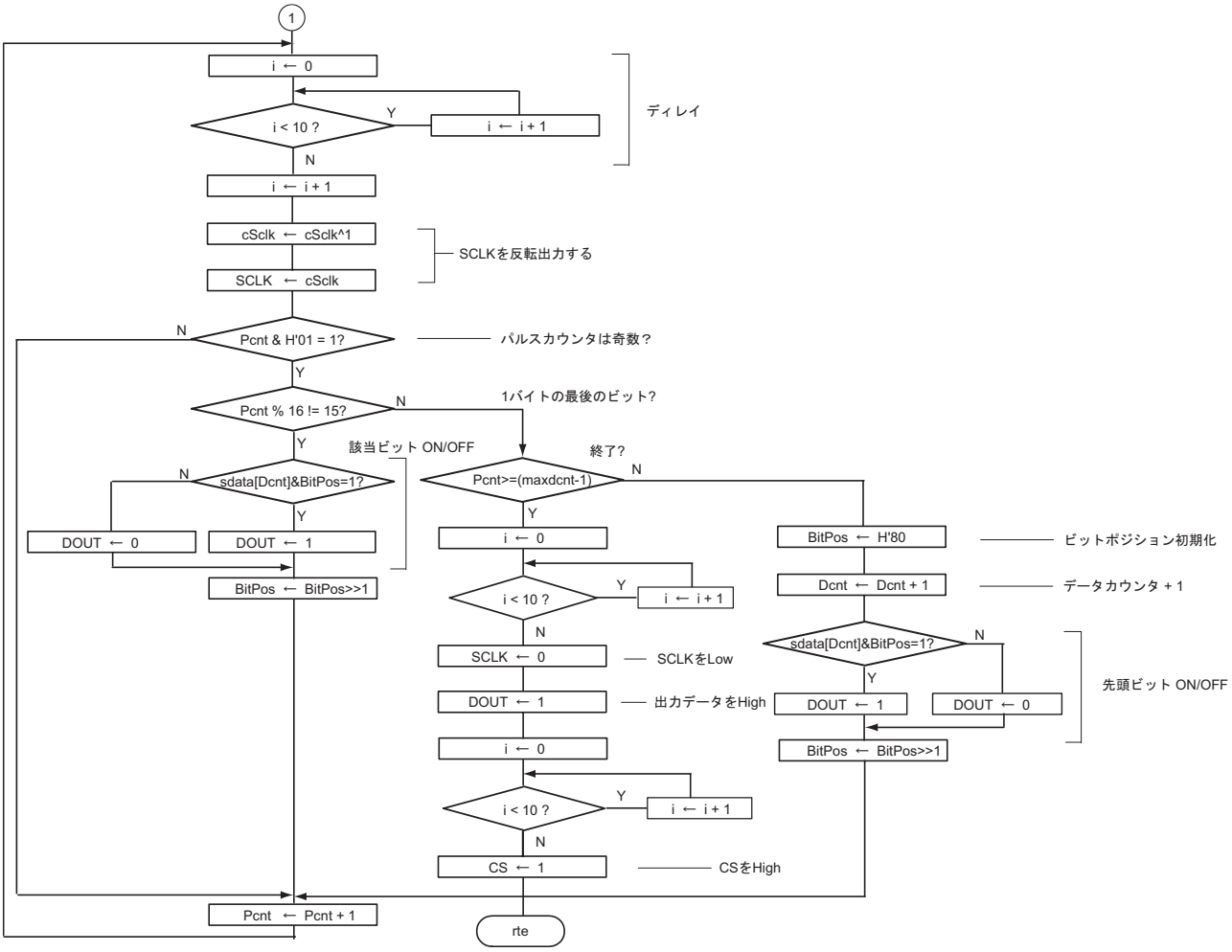

ADCレジスタに書き込み
1
i ← 0
ディレイ
i ← i + 1
Y
i < 10 ?
N
cSclk ← cSclk^1
SCLKを反転出力する
SCLK ← cSclk
N
Pcnt & H'01 = 1?
Pcntは奇数？
Y
Pcnt % 16 != 15?
N
1バイトの最後のビット？
Y
該当ビット
のON/OFF
N
sdata[Dcnt]&BitPos=1?
Y
DOUT ← 0
DOUT ← 1
BitPos ← BitPos>>1
Pcnt ← Pcnt + 1
i ← 0
i ← i + 1
Y
i < 10 ?
ディレイ
N
SCLK ← 0
SCLKをLow
DOUT ← 1
DOUTをLow
2



*1 = ((Pcnt % 16) == 0) && (Pcnt != 0)

6. DataOut()処理ルーチン




1
i ← 0
i < 10 ?
Y
i ← i + 1
N
ディレイ
i ← i + 1
cSclk ← cSclk^1
SCLK ← cSclk
SCLKを反転出力する
N
Pcnt & H'01 = 1?
パルスカウンタは奇数？
Y
Pcnt % 16 != 15?
N
1バイトの最後のビット?
Y
該当ビット ON/OFF
終了?
N
sdata[Dcnt]&BitPos=1?
Pcnt>=(maxdcnt-1)
N
Y
DOUT ← 0
DOUT ← 1
BitPos ← BitPos>>1
i ← 0
BitPos ← H'80
ビットポジション初期化
i < 10 ?
Y
i ← i + 1
Dcnt ← Dcnt + 1
データカウンタ + 1
N
SCLK ← 0
SCLKをLow
sdata[Dcnt]&BitPos=1?
N
DOUT ← 1
出力データをHigh
Y
DOUT ← 1
DOUT ← 0
先頭ビット ON/OFF
i ← 0
BitPos ← BitPos>>1
i < 10 ?
Y
i ← i + 1
N
CS ← 1
CSをHigh
Pcnt ← Pcnt + 1
rte

6. プログラムリスト

```
/* Super Low Power Series -H8/38024- Application note */
/* 応用編 */
/* ΣΔADC接続例 */

#include <machine.h>

/* Symbol defnition */
struct BIT {
   unsigned char b7:1;   /* bit 7 */
   unsigned char b6:1;   /* bit 6 */
   unsigned char b5:1;   /* bit 5 */
   unsigned char b4:1;   /* bit 4 */
   unsigned char b3:1;   /* bit 3 */
   unsigned char b2:1;   /* bit 2 */
   unsigned char b1:1;   /* bit 1 */
   unsigned char b0:1;   /* bit 0 */
};

#define PDR1  *(volatile unsigned char *)0xFFD4     /* Port data register 1 */
#define PCR1  *(volatile unsigned char *)0xFFE4     /* Port control register 1 */
#define PUCR1 *(volatile unsigned char *)0xFFE0     /* Port pull-up control register 1 */
#define PMR1  *(volatile unsigned char *)0xFFC8     /* Port mode register 1 */
#define PMR2  *(volatile unsigned char *)0xFFC9     /* Port mode register 2 */
#define PDR1_BIT (*(struct BIT *)0xFFD4)
#define WU1      PDR1_BIT.b3                        /* MAX1408 WU1 */
#define WU2      PDR1_BIT.b4                        /* MAX1408 WU2 */

#define PDR4  *(volatile unsigned char *)0xFFD7     /* Port data register 4 */
#define PCR4  *(volatile unsigned char *)0xFFE7     /* Port control register 4 */
#define PDR4_BIT (*(struct BIT *)0xFFD7)
#define SCLK  PDR4_BIT.b0                           /* MAX1408 SCLK */
#define DOUT  PDR4_BIT.b2                           /* MAX1408 DOUT */
#define DIN      PDR4_BIT.b1                        /* MAX1408 DIN */

#define PMR5  *(volatile unsigned char *)0xFFCC     /* Port mode register 5 */
#define PUCR5 *(volatile unsigned char *)0xFFE2     /* Port pull-up control register 5 */
#define PDR5  *(volatile unsigned char *)0xFFD8     /* Port data register 5 */
#define PCR5  *(volatile unsigned char *)0xFFE8     /* Port control register 5 */

#define PUCR6 *(volatile unsigned char *)0xFFE3     /* Port pull-up control register 6 */
#define PDR6  *(volatile unsigned char *)0xFFD9     /* Port data register 6 */
#define PCR6  *(volatile unsigned char *)0xFFE9     /* Port control register 6 */

#define PDR8  *(volatile unsigned char *)0xFFDB     /* Port data register 8 */
#define PCR8  *(volatile unsigned char *)0xFFEB     /* Port control register 8 */
#define PDR8_BIT (*(struct BIT *)0xFFDB)
#define CS       PDR8_BIT.b0                        /* MAX1408 #CS */
#define PWRON PDR8_BIT.b1                           /* MAX1408 POWER */

#define TMA      *(volatile unsigned char *)0xFFB0  /* Timer mode register A */
#define CKSTPR1  *(volatile unsigned char *)0xFFFA  /* Clock stop register 1 */

#define TMC      *(volatile unsigned char *)0xFFB4  /* Timer mode register C */
#define TLC      *(volatile unsigned char *)0xFFB5  /* Timer Load register C */

#define IRR1  *(volatile unsigned char *)0xFFF6     /* Interrupt request register 1 */
```

```
#define IRR1_BIT (*(struct BIT *)0xFFF6)
#define IRRTA IRR1_BIT.b7                          /* Timer A interrupt request flag */
#define IENR1 *(volatile unsigned char *)0xFFF3    /* Interrupt enable register 1 */
#define IENR1_BIT (*(struct BIT *)0xFFF3)
#define IENTA IENR1_BIT.b7                         /* Timer A interrupt enable */

#define IRR2  *(volatile unsigned char *)0xFFF7    /* Interrupt request register 2 */
#define IRR2_BIT (*(struct BIT *)0xFFF7)
#define IRRTC IRR2_BIT.b1                          /* Timer C interrupt request flag */
#define IENR2 *(volatile unsigned char *)0xFFF4    /* Interrupt enable register 2 */
#define IENR2_BIT (*(struct BIT *)0xFFF4)
#define IENTC IENR2_BIT.b1                         /* Timer C interrupt enable */

#pragma interrupt (tmra)
#pragma interrupt (tmrc)

/* 関数定義 */
extern void INIT(void);                            /* Stack pointer set */
void main(void);                                   /* main routine */
void tmra(void);                                   /* Timer A interrupt routine */
void tmrc(void);                                   /* Timer C interrupt routine */
void ADC_Init(void);                               /* ADC initialize   */
void DataIn(int ByteNo);                           /* ADC read */
void DataOut(int ByteNo);                          /* ADC write */

/* Data table */
const unsigned char dsp_data[16] =
{
   0x3f,                                           /* LED display data = "0" */
   0x06,                                           /* LED display data = "1" */
   0x5b,                                           /* LED display data = "2" */
   0x4f,                                           /* LED display data = "3" */
   0x66,                                           /* LED display data = "4" */
   0x6d,                                           /* LED display data = "5" */
   0x7d,                                           /* LED display data = "6" */
   0x27,                                           /* LED display data = "7" */
   0x7f,                                           /* LED display data = "8" */
   0x6f,                                           /* LED display data = "9" */
   0x77,                                           /* LED display data = "A" */
   0x7c,                                           /* LED display data = "b" */
   0x39,                                           /* LED display data = "C" */
   0x5e,                                           /* LED display data = "d" */
   0x79,                                           /* LED display data = "E" */
   0x71                                            /* LED display data = "F" */
};

/* RAM define */
unsigned char dig_0;                               /* Dig-0 LED display data store */
unsigned char dig_1;                               /* Dig-1 LED display data store */
unsigned char dig_2;                               /* Dig-2 LED display data store */
unsigned char dig_3;                               /* Dig-3 LED display data store */
unsigned char cnt;                                 /* LED enable counter */
unsigned int temp;
unsigned char *ptr;                                /* Pointer set */
volatile unsigned char   sdata[5];                 /* send data area         */
volatile unsigned char   rdata[5];                 /* receive data area*/
```

```
/* Vector address */
#pragma section V1                                  /* Vector section set */
void (*const VEC_TBL1[])(void) = {
   INIT                                             /* H'0000 Reset vector */
};
#pragma section V2                                  /* Vector section set */
void (*const VEC_TBL2[])(void) = {
   tmra                                             /* H'0016 Timer A interrupt vector */
};
#pragma section V3                                  /* Vector section set */
void (*const VEC_TBL3[])(void) = {
   tmrc                                             /* H'001a Timer C interrupt vector */
};
#pragma section                                     /* P */

/************************************************************/
/* Main program                                             */
/************************************************************/
void main(void)
{
   set_imask_ccr(1);                                /* CCR I-bit = 1 */

   dig_0 = 0x40;                                    /* Used RAM area initialize */
   dig_1 = 0x40;                                    /* Used RAM area initialize */
   dig_2 = 0x40;                                    /* Used RAM area initialize */
   dig_3 = 0x40;                                    /* Used RAM area initialize */
   cnt   = 0x00;                                    /* Used RAM area initialize */

   PCR1  = 0x3f;                                    /* Port 1 initialize */
   PCR4  = 0xfd;                                    /* Port 4 initialize */
   PCR5  = 0xff;                                    /* Port 5 initialize */
   PCR6  = 0xff;                                    /* Port 6 initialize */
   PCR8  = 0xff;                                    /* Port 8 initialize */

   CS = 1;                                          /* CS High    */
   PWRON = 0;                                       /* POWER OFF  */
   WU2 = 1;                                         /* Wake up OFF      */
   WU2 = 1;                                         /* Wake up OFF      */

   ADC_Init();                                      /* ADC initialize   */

   TMA   = 0x0c;                                    /* Clear Timer Counter A to 0 */
   TMA   = 0x09;                                    /* Timer A initialize */
   TLC   = 0x00;                                    /* Clear Timer Load register C to 0 */
   TMC   = 0x1b;                                    /* Timer C initialize */

   IRRTA = 0;                                       /* Clear IRRTA to 0 */
   IRRTC = 0;                                       /* Clear IRRTC to 0 */
   IENTA = 1;                                       /* Timer A interrupt enable */
   IENTC = 1;                                       /* Timer C interrupt enable */

   set_imask_ccr(0);                                /* CCR I-bit = 0 */

   while(1);
}
```

```
/*********************************************************/
/* Timer A Interrupt                                     */
/*********************************************************/
void tmra(void)
{
    IRRTA = 0;                                  /* Clear IRRTA to 0 */

    rdata[0] = 0x00;
    do {
        sdata[0] = 0xcc;                        /* Read to Status Register */
        DataIn(1);
    } while((rdata[0] & 0x02) == 0);
    sdata[0] = 0xc4;                            /* Read DATA Register value */
    DataIn(2);

    dig_0 = dsp_data[rdata[1] & 0x0f];          /* Dig-0 LED display data set */
    dig_1 = dsp_data[rdata[1] >> 4 & 0x0f];     /* Dig-1 LED display data set */
    dig_2 = dsp_data[rdata[0] & 0x0f];          /* Dig-2 LED display data set */
    dig_3 = dsp_data[rdata[0] >> 4 & 0x0f];     /* Dig-3 LED display data set */
}

/*********************************************************/
/* Timer C Interrupt                                     */
/*********************************************************/
void tmrc(void)
{
    IRRTC = 0;                                  /* Clear IRRTC to 0 */

    ptr = &dig_0;                               /* LED display data store address set*/
    ptr += cnt;                                 /* LED display data read */
    PDR5 = *ptr;                                /* LED display data output */
    PDR6 = cnt;                                 /* LED enable data output */

    cnt++;                                      /* "cnt" increment */
    if (cnt >= 4){                              /* 4 times end ? */
        cnt = 0;                                /* "cnt" initialize */
   }
}

/*********************************************************/
/* ADC initialize function                               */
/*********************************************************/
void ADC_Init(void)
{
    long i;

    PWRON = 1;                                  /* POWER ON   */

    for(i=0;i<10000;i++);                       /* delay      */
    WU1 = 0;

    sdata[0] = 0xba;                            /* Write RUN Register */
    DataOut(1);

    sdata[0] = 0xb0;                            /* Write POWER1 Register */
    sdata[1] = 0xf0;
    DataOut(2);
```

```
    sdata[0] = 0x82;                                /* Write MUX Register */
    sdata[1] = 0x61;
    DataOut(2);

    sdata[0] = 0x80;                                /* Write ADC Register */
    sdata[1] = 0x11;
    DataOut(2);
}

/*************************************************************/
/* ADC register write and read function                      */
/*************************************************************/
void DataIn(int ByteNo)
{
    int cSclk, Dcnt, maxdcnt, i, Pcnt;
    unsigned  char BitPos;

    rdata[0] = rdata[1] = 0x00;                     /* data area initialize    */

    maxdcnt = (ByteNo << 4);
    Dcnt = Pcnt = 0;                                /* counter initialize      */
    BitPos = 0x80;                                  /* bit position initialize */
    CS = 0;                                         /* CS ON      */
    SCLK = cSclk = 0;                               /* SCLK Low   */
    if(sdata[Dcnt]  & BitPos)                       /* output data      */
       DOUT = 1;
    else
       DOUT = 0;
    BitPos >>= 1;                                   /* next bit position*/

    /* write register   */
    while(1){
       for(i=0;i<10;i++);                           /* delay      */
       cSclk ^= 1;
       SCLK = cSclk;                                /* output reverse SCLK     */
       if(Pcnt & 0x01) {
          if((Pcnt % 16) != 15){                    /* last bit ? */
             if(sdata[Dcnt] & BitPos)               /* output data      */
                DOUT = 1;
             else
                DOUT = 0;
             BitPos >>= 1;                          /* next bit position*/
          } else {
             for(i=0;i<10;i++);                     /* delay      */
             SCLK = 0;                              /* SCLK Low   */
             DOUT = 1;                              /* DOUT Low   */
             break;
          }
       }
       Pcnt++;                                      /* pulse count increment   */
    }

    /* read register */
    Dcnt = Pcnt = 0;                                /* SCLK count */
    BitPos = 0x80;                                  /* bitb position initialize      */
    while(1){
```

```
        if((Pcnt & 0x01) == 0x00) {
              if(DIN)
                 rdata[Dcnt] |= BitPos;             /* input data */
              BitPos >>= 1;                         /* next bit position*/
              if(Pcnt >= (maxdcnt - 1)) {           /* end ?      */
                 for(i=0;i<10;i++);                 /* delay      */
                 SCLK = 0;                          /* SCLK Low   */
                 for(i=0;i<10;i++);                 /* delay      */
                 CS = 1;                            /* CS High    */
                 return;
              } else if(((Pcnt % 16) == 0) && (Pcnt != 0)){/* next data */
                 BitPos = 0x80;                     /* bitb position initialize      */
                 Dcnt++;                            /* data count increment    */
                 if(DIN)                            /* input data */
                    rdata[Dcnt] |= BitPos;
                 BitPos >>= 1;                      /* next bit position*/
              }
        }
        cSclk ^= 1;
        SCLK = cSclk;                               /* reverse SCLK     */
        Pcnt++;                                     /* pulse count increment   */
        for(i=0;i<10;i++);                          /* delay      */
    }
}

/*************************************************************/
/* ADC register write function                               */
/*************************************************************/
void DataOut(int ByteNo)
{
    int cSclk, Dcnt, maxdcnt, i, Pcnt;
    unsigned  char BitPos;

    maxdcnt = (ByteNo << 4);
    Dcnt = Pcnt = 0;                                /* counter initialize      */
    BitPos = 0x80;                                  /* bit position initialize */
    CS = 0;                                         /* CS ON      */
    SCLK = cSclk = 0;                               /* SCLK Low   */
    if(sdata[Dcnt]  & BitPos)                       /* output data      */

        DOUT = 1;
    else
        DOUT = 0;
    BitPos >>= 1;                                   /* next bit position*/

    while(1){
        for(i=0;i<10;i++);                          /* delay      */
        cSclk ^= 1;
        SCLK = cSclk;                               /* reverse SCLK     */
        if(Pcnt & 0x01) {
           if((Pcnt % 16) != 15){                   /* not last bit ?   */
              if(sdata[Dcnt] & BitPos)              /* output data      */
                 DOUT = 1;
              else
                 DOUT = 0;
              BitPos >>= 1;                         /* next bit position*/
           } else {
```

```
            if(Pcnt >= (maxdcnt - 1)) {          /* end ?     */
               for(i=0;i<10;i++);                /* delay     */
               SCLK = 0;                         /* SCLK Low  */
               DOUT = 1;                         /* output High      */
               for(i=0;i<10;i++);                /* delay     */
               CS = 1;                           /* CS OFF    */
               return;
            } else {
               BitPos = 0x80;                    /* bit position initialize */
               Dcnt++;                           /* data count increment    */
               if(sdata[Dcnt] & BitPos)          /* output data      */
                  DOUT = 1;
               else
                  DOUT = 0;
               BitPos >>= 1;                     /* next bit position*/
            }
         }
      }
      Pcnt++;                                  /* pulse count increment   */
   }
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.18	—	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。