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

省電力モードからのリモコン信号受信例

要旨

赤外線リモコン受光ユニットが受信した信号を割込み入力に用いて，スタンバイモードのマイコンをアクティブモードに遷移させた後，赤外線リモコンからのデータ受信を行なう。

動作確認デバイス

H8/38024

目次

1. 仕様

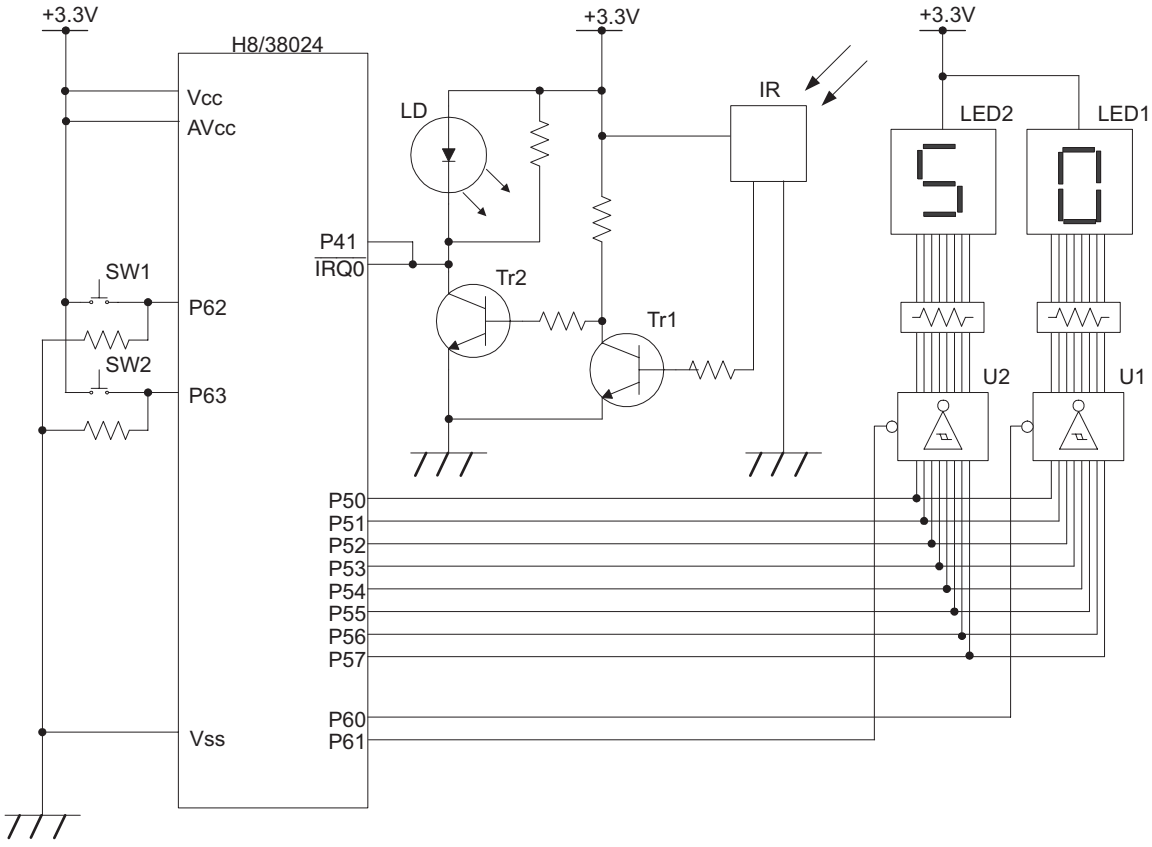
図 1 に赤外線リモコン受光ユニットを使用した，赤外線リモコンデータ受信を行なうためのハードウエア構成を示します。

本タスク例では，マイコンの省電力モードの一つであるスタンバイモードから，赤外線リモコンの信号でマイコンをアクティブモードに遷移させ，信号の受信を行ないます。

受信したデータを 7 セグメント LED に 2 桁の 16 進数 (1 バイト) で表示します。プッシュボタンスイッチ (SW1) を押すと表示が順次 1 バイトシフトします。

再度，スタンバイモードにするためには，プッシュボタンスイッチ (SW2) を押して切り換えます。

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 クロック周波数はセラミック発振子を用いて 10MHz，ウォッチクロック周波数は 32.768kHz です。



【記号説明】

IR	：赤外線リモコン受光ユニット（GP1UD281XK、シャープ）
LED1～LED2	：アノードコモン7セグメントLED
LD	：発光ダイオード
U1～U2	：3ステート出力インバータドライバ（HD74LS240）
SW1～SW2	：プッシュボタンスイッチ
Tr1～Tr2	：トランジスタ（2SC1815、東芝）

図 1　ハードウエア構成

本タスクで使用している赤外線リモコン受光ユニットは，シャープ製リモコン受光ユニット (型名：GP1UD281XK) です。以下に仕様を示します。

図 2 に赤外線リモコン受光ユニットの内部ブロックダイアグラムを示します。

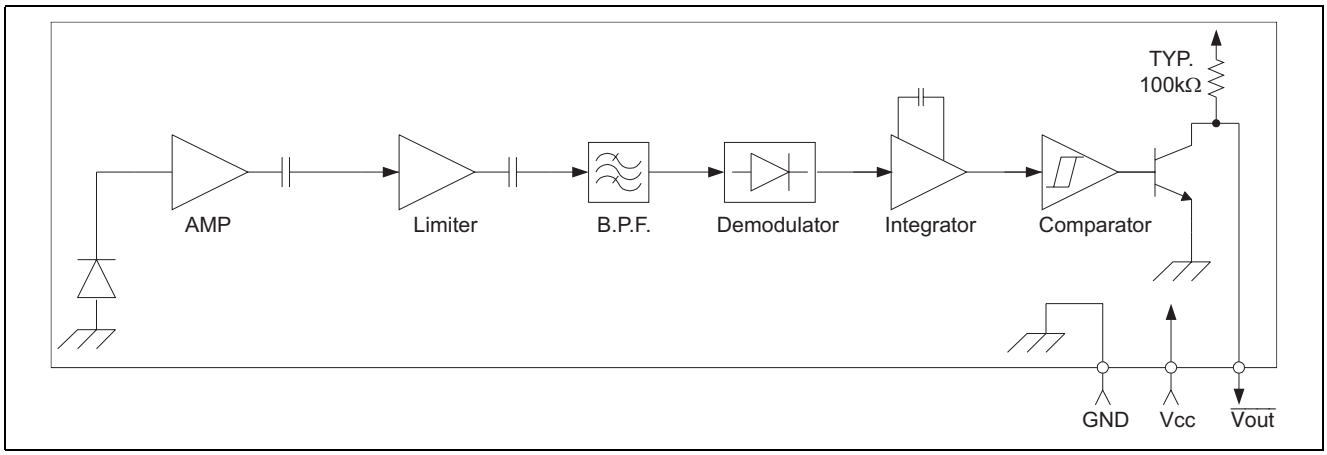


図 2　内部ブロックダイアグラム

GP1UD281XK の特長を示します。

(1) 2.7V～5.5V の電源電圧範囲
(2) キャリア周波数は 38kHz
(3) PPM (Pulse Position Modulation) 方式対応の復調回路内蔵

本タスク例の動作は以下のとおりです。

(1) (スタンバイモード) → (アクティブモード) → (スタンバイモード) の順に動作します。
(2) 赤外線リモコンから送信された赤外線の信号を，赤外線リモコン受光ユニットで受信 (受光) →復調し，信号の先頭部分によりマイコンに割込みの受付けを開始します。
(3) 割込みが受け付けられた時点でマイコンの発振波形が変化を開始し，AC 特性の「発振安定時間」で規定された時間が経過するとシステムクロックが動作します。
(4) さらに，「待機時間」 (8～16384 ステート) が経過した後，スタンバイモードが解除されて割込み例外処理を開始し，アクティブモードに遷移します。
(5) マイコンは割込みを受付け後，「発振安定待機時間」を経由し，スタンバイモードからアクティブモードに遷移します。
(6) 本タスク例では待機時間を 16 ステートに設定し，動作確認しました。待機時間の定義を図 3 に示します。
発振安定待機時間 = 発振安定時間 + 待機時間
(7) アクティブモードに遷移したマイコンは，後続信号を受信します。
(8) 受信したデータは 7 セグメント LED を 2 個使用して表示します。表示は 2 桁の 16 進数 (1 バイト) で行ない，プッシュボタンスイッチ (SW1) を 1 回押すごとに取り込んだデータを 2 桁 (1 バイト) シフトさせて次のデータを表示します。
(9) 本タスク例で用いた赤外線リモコン受信データは，4 バイト (= 32 ビット) です。7 セグメント LED を 2 個使用して，次のとおりに表示します。
「50」→「AF」→「17」→「E8」
受信データ 4 バイト表示後，7 セグメント LED を「 - -」と表示する。
(各メーカからリモコンコードが公開されていないため，一般的なリモコン信号フォーマットである LSB ファーストに準拠し，データを 1 バイト (= 8 ビット) 区切りで処理した結果)
(10) 受信データの表示が終了し，必要に応じてプッシュボタンスイッチ (SW2) を押すことでスタンバイモードになり，リモコン信号の受信待機状態になります。
(11) ご参考までに，受信データは 2 進数では次のようになり，1 バイト目および 2 バイト目，ならびに 3 バイト目および 4 バイト目はそれぞれのビット反転値の関係にあります。
「H'50」(= 0101 0000)，「H'AF」(= 1010 1111)，「H'17」(= 0001 0111)，「H'E8」(= 1110 1000)

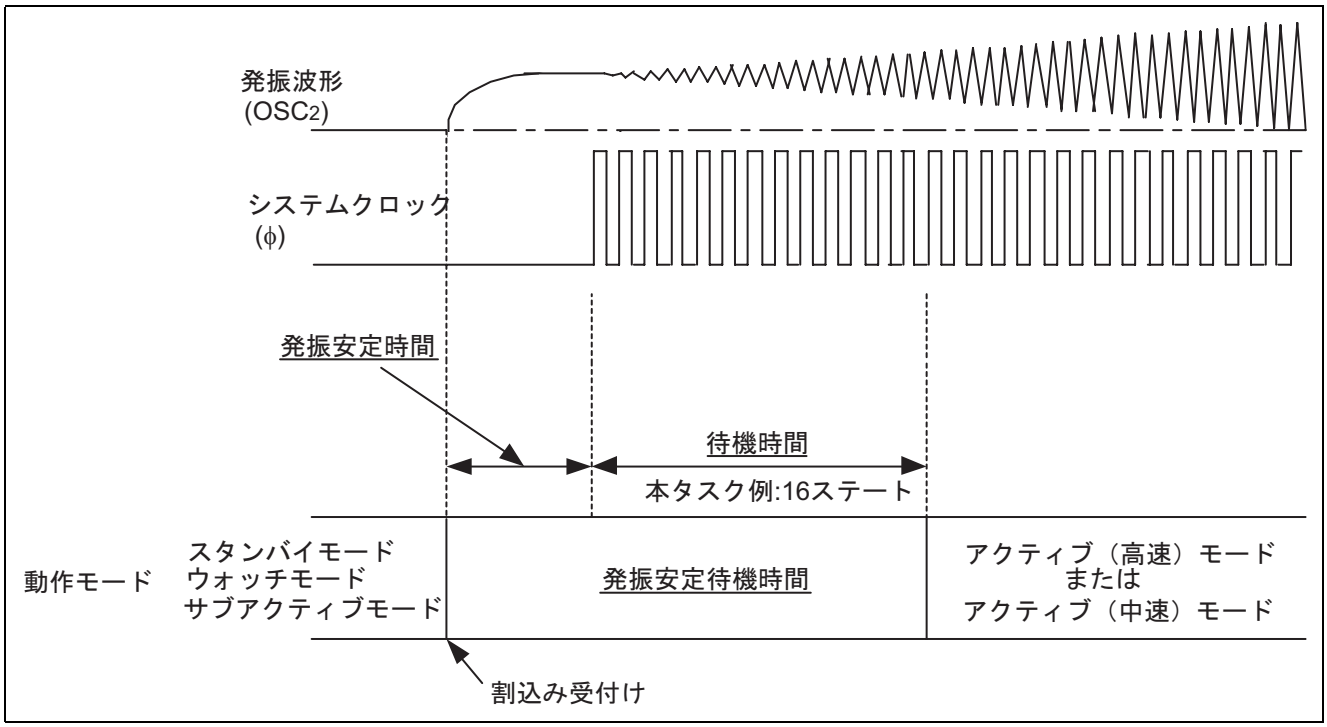


図 3　発振安定待機時間

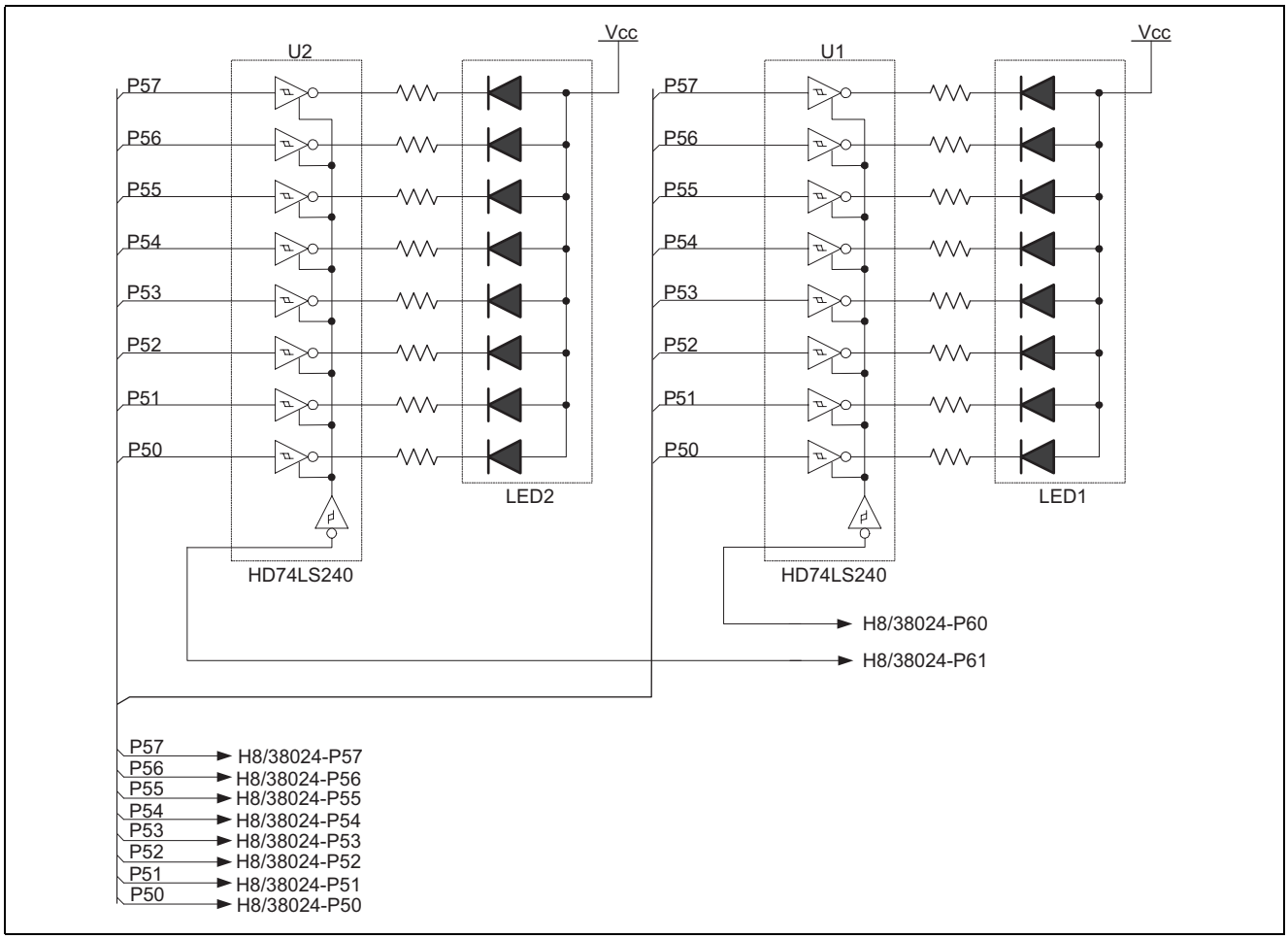


図 4　7 セグメント LED 制御方法

本タスク例では，リモコン入力結果を 7 セグメント LED に 16 進数で表示 (H'FF～H'00) させます。図 5 にリモコン入力結果の LED 表示方法を示します。

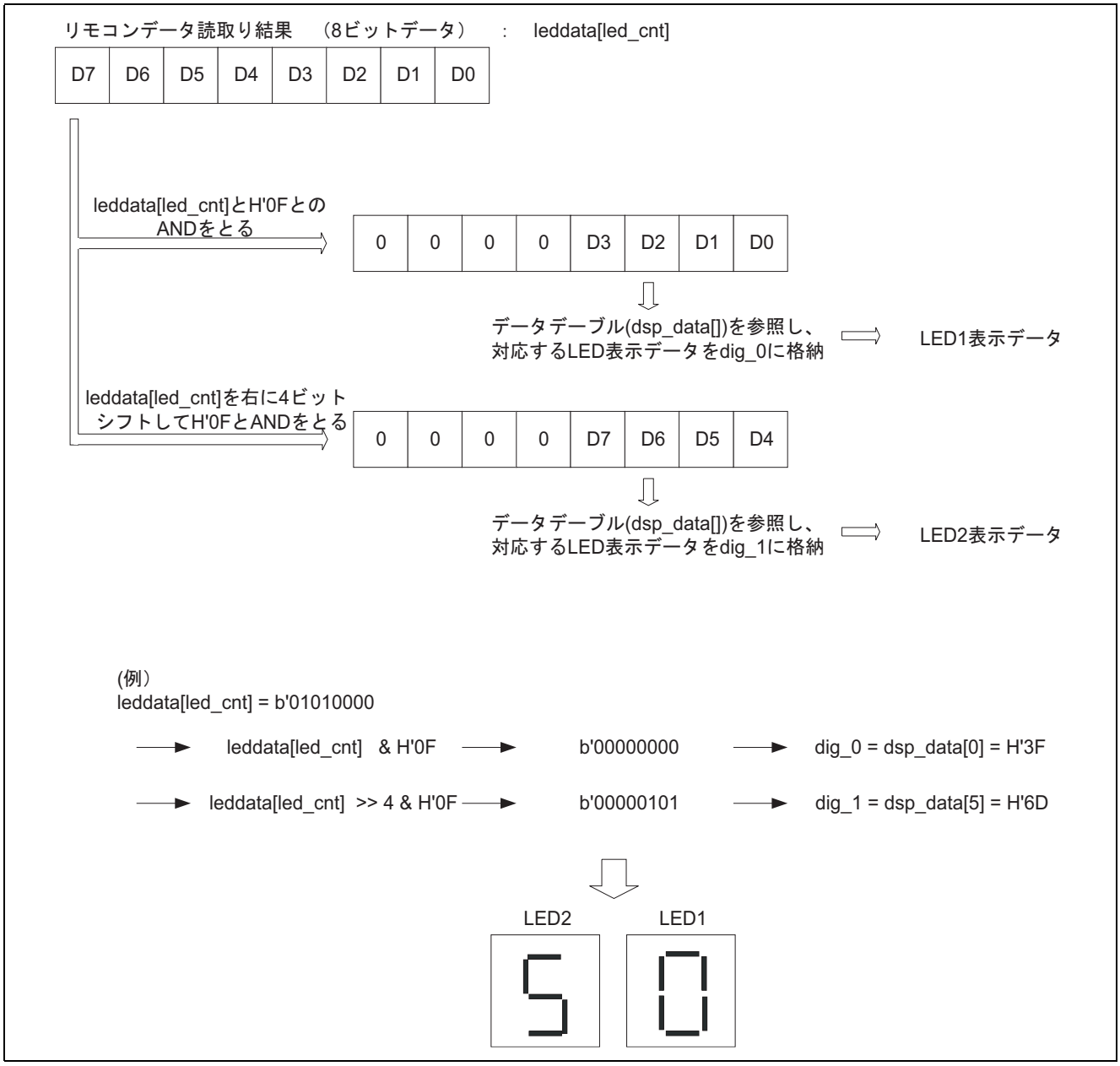


図 5　リモコン入力結果の LED 表示方法

2. 使用機能説明

図 6 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。

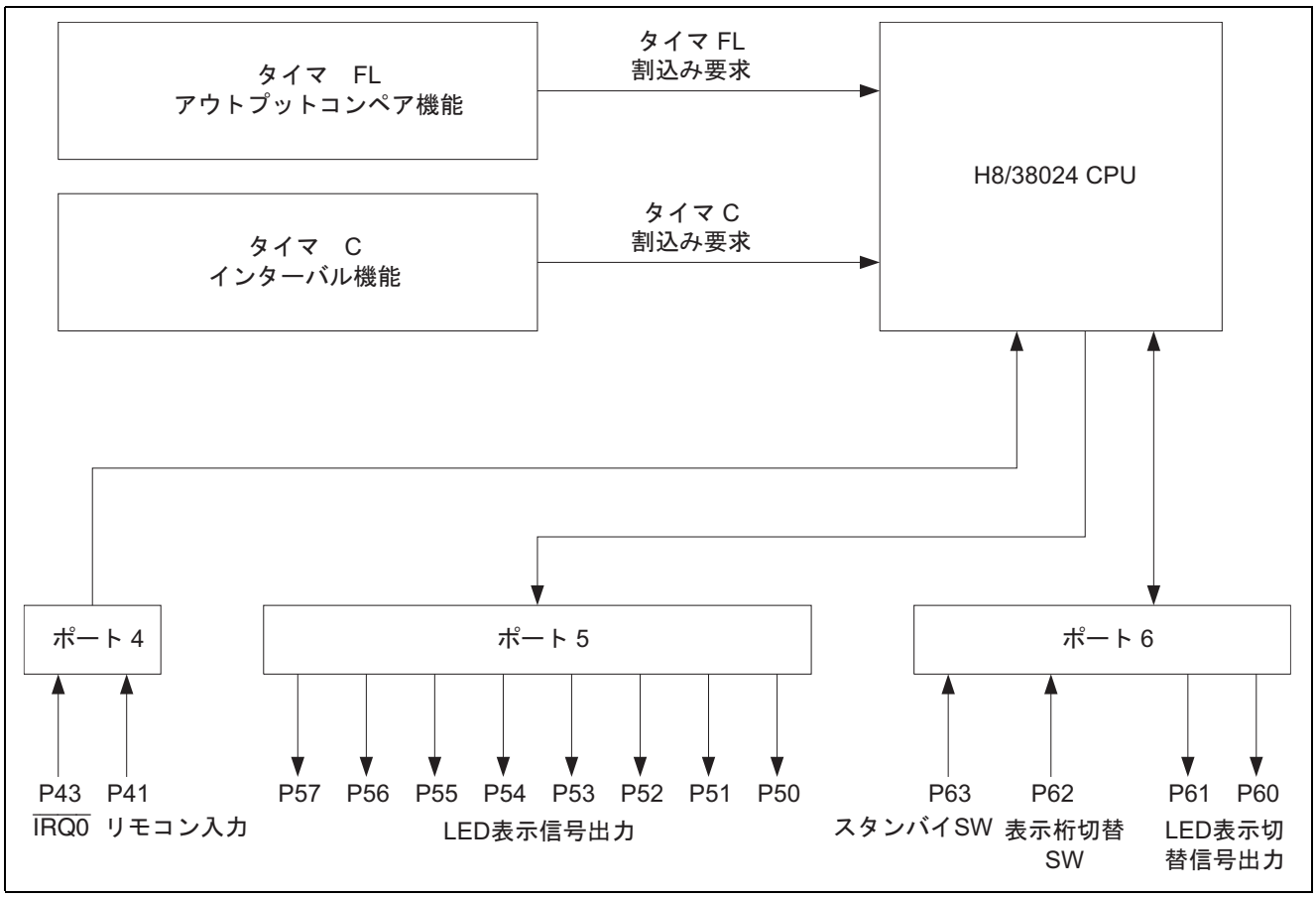


図 6　使用機能ブロック図

表 1　機能割付け

使用機能	機能割付け
タイマ FL	アウトプットコンペア機能を使用して，0.1ms ごとに P41 入力端子からリモコンの赤外線データ入力を行ないます。
タイマ C	インターバル機能を使用して，7 セグメント LED の表示切換え制御を行ないます。タイマ C オーバフロー周期 3.2768ms ごとに 2 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ないます。
ポート 4	P41 入力端子からリモコンの赤外線データを受信し，P43 の $\overline{\text{IRQ0}}$ 割込みによってスタンバイモードからアクティブ（高速）モードに遷移します。
ポート 5	P50～P57 出力端子により，7 セグメント LED の表示を行ないます。P41 端子からのリモコンコードデータを 2 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	P61, P60 を交互に ON/OFF することによって 2 個の 7 セグメント LED を交互に点灯させます。 P62 の表示切換え SW を押下することにより，複数バイト入力されたリモコンコードを順番に表示します。P63 のスタンバイ SW を押下することにより，アクティブ（高速）モードからスタンバイモードへ遷移します。

使用する 7 セグメント LED の接続図を図 6 に示します。図 7 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。

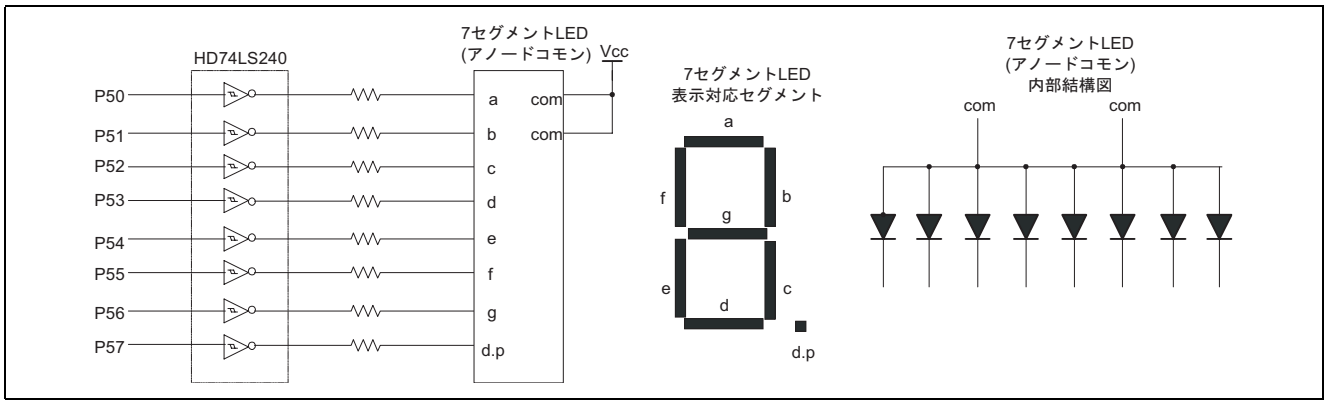


図 7　7 セグメント LED 接続図および内部結線図

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							
	P57	P56	P55	P54	P53	P52	P51	P50
	0	0	1	1	1	1	1	1
	0	0	0	0	0	1	1	0
	0	1	0	1	1	0	1	1
	0	1	0	0	1	1	1	1
	0	1	1	0	0	1	1	0
	0	1	1	0	1	1	0	1
	0	1	1	1	1	1	0	1
	0	0	1	0	0	1	1	1

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

3. 動作説明

図 8 にタイマ FL を使用した，リモコン受信を行なう際の動作原理を示します。動作モードをスタンバイモードに設定し，リモコンの赤外線信号が P41 より入力され，それによる IRQ0 割込みでマイコンはアクティブ (高速) モードへ遷移し，タイマ FL の 0.1ms ごとの割込みによって赤外線信号の状態を入力します。

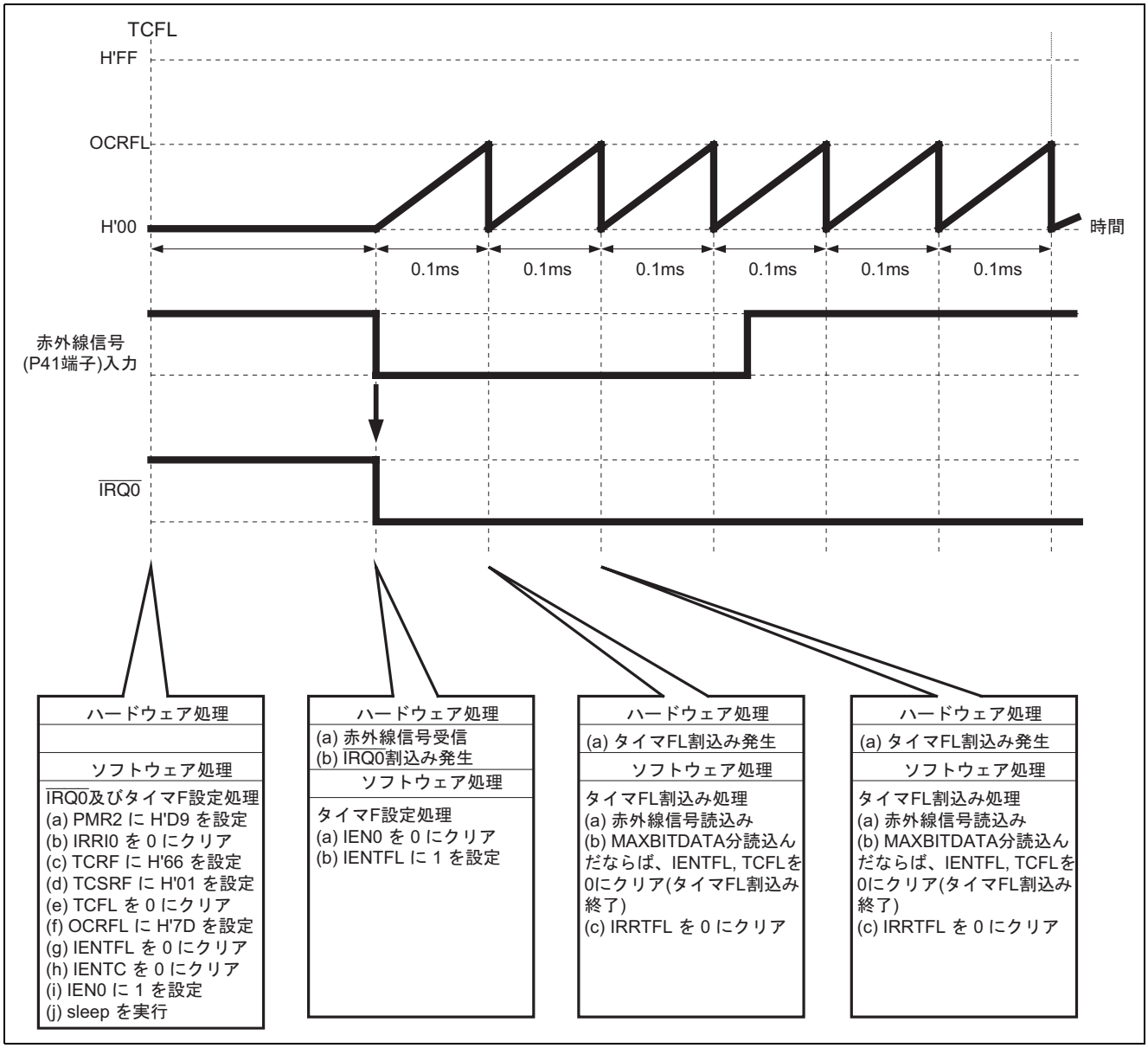


図 8　タイマ FL を使用したリモコン受信の動作原理

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 9 は LED2～LED1 に”50”を表示する場合の動作原理について説明しています。図 9 に示すようにタイマ C オーバフロー周期ごとに LED1～LED2 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なっています。

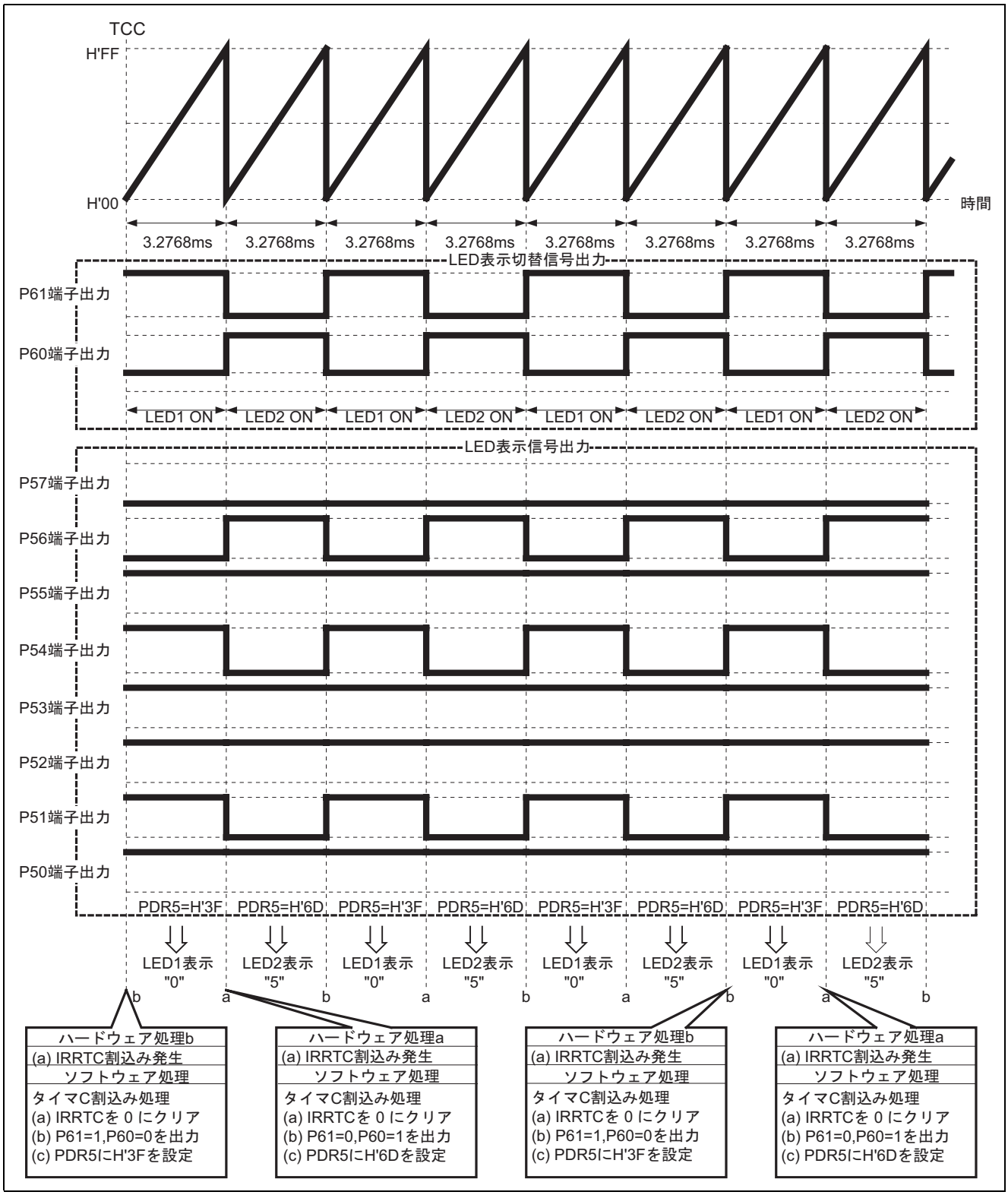


図 9　7 セグメント LED 表示制御の動作原理

4. ソフトウェア説明

4.1 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	初期設定後， スタンバイモードへ遷移，データ取込み終了待ち，コード決定処理，LED 表示処理を繰り返す。
コード決定処理	code_decision	リモコンより入力されたデータからコード部分を抽出する。
LED 表示処理	led_disp	表示切換え SW が ON で,複数バイト入力されたリモコンコードを順番に表示します。スタンバイ SW が ON で，アクティブ (高速) モードからスタンバイモードへ遷移します。
ソフトウェアディレイ処理	delay	約 300ms のソフトウェアタイマとして使用
$\overline{\text{IRQ0}}$ 割込み処理	irq0	$\overline{\text{IRQ0}}$ 割込み禁止
タイマ C 割込み処理	tmrc	割込みフラグのクリア，LED 表示データの出力と LED 表示切換えの制御
タイマ F 割込み処理	tmrf	割込みフラグのクリア，赤外線信号の状態を入力する。

4.2 引数説明

本タスク例では，引数を使用しません。

4.3 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
TCRF		タイマコントロールレジスタ F ：16 ビットモード，8 ビットモードの切換え，4 種類の内部クロックおよび外部イベントの選択，TMOFH，TMOFL 端子の出力レベルの設定	H'FFB6	H'66
	TOLH	トグルアウトプットレベル H ：TMOFH 端子の出力レベルを設定 TOLH = 0 設定時， ：Low レベル	ビット 7	0
	CKSH2	クロックセレクト H ：CKSH2 = 1, CKSH1 = 1, CKSH0 = 0 のとき，内部クロックφ/4 でカウント	ビット 6	1
	CKSH1		ビット 5	1
	CKSH0		ビット 4	0
	TOLL	トグルアウトプットレベル L ：TMOFL 端子の出力レベルを設定 TOLL = 0 設定時， ：Low レベル	ビット 3	0
	CKSL2	クロックセレクト L ：CKSL2 = 1, CKSL1 = 1, CKSL0 = 0 のとき，内部クロックφ/4 でカウント	ビット 2	1
	CKSL1		ビット 1	1
	CKSL0		ビット 0	0
TMC		タイマモードレジスタ C ：オートリロード機能の選択，カウンタのアップ／ダウン制御，および入力クロックの選択	H'FFB4	H'1B
	TMC7	オートリロード機能選択 ：タイマ C のオートリロード機能を選択 ：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 ：アップカウンタとするかダウンカウンタとするかを選択 ：TMC6 = 0, TMC5 = 1 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 ：TCC に入力するクロックを選択する ：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

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
TCSRF		タイマコントロールステータスレジスタ F ：カウンタクリアの選択，オーバフローフラグのセット，コンペアマッチフラグのセット，オーバフローによる割込み要求の許可の制御	H'FFB7	H'01
	OVFH	タイマオーバフローフラグ H ：TCFH がオーバフロー (H'FF→H'00) したことを示すステータスフラグ	ビット 7	
	CMFH	コンペアマッチフラグ H ：TCFH と OCRFH がコンペアマッチしたことを示すステータスフラグ	ビット 6	
	OVIEH	タイマオーバフローインタラプトイネーブル H ：OVIEH = 0 のとき，TCFH のオーバフローによる割込み要求を禁止	ビット 5	
	CCLRH	カウンタクリア H ：CCLRH = 1 のとき，コンペアマッチによる TCF のクリアを許可	ビット 4	
	OVFL	タイマオーバフローフラグ L ：TCFL がオーバフロー (H'FF→H'00) したことを示すステータスフラグ	ビット 3	
	CMFL	コンペアマッチフラグ L :TCFL と OCRFL がコンペアマッチしたことを示すステータスフラグ	ビット 2	
	OVIEL	タイマオーバフローインタラプトイネーブル L ：OVIEL = 0 のとき，TCFL のオーバフローによる割込み要求を禁止	ビット 1	
	CCLRL	カウンタクリア L ：CCLRL = 1 のとき，コンペアマッチによる TCFL のクリアを許可	ビット 0	
TCFL		8 ビットタイマカウンタ ：TCFL は 8 ビットのリード／ライト可能なアップカウンタ	H'FFB9	H'00
OCRFL		8 ビットアウトプットコンペアレジスタ ：TCFL とのコンペアマッチによる割込みを発生させる	H'FFBB	H'7D
IENR1		割込み許可レジスタ 1	H'FFF3	H'01 (初期設定時)
	IEN0	：IEN0＝1 のとき，$\overline{\text{IRQ0}}$ 端子の割込み要求を許可	ビット 0	1/0
IENR2		割込み許可レジスタ 2	H'FFF4	H'00 (初期設定時)
	IENTFL	：IENTFL = 1 のとき，タイマ FL の割込み要求を許可	ビット 2	1/0
	IENTC	：IENTC = 1 のとき，タイマ C の割込み要求を許可	ビット 1	1/0
IRR1		割込み要求レジスタ 1	H'FFF6	H'00 (初期設定時)
	IRRI0	：IRRI0 = 1 の状態で IRRI0 に 0 をライトしたときにクリアできる ：$\overline{\text{IRQ0}}$ 端子が割込み入力に設定されており，かつ当該端子に指定されたエッジが入力されたときに 1 がセットされる	ビット 0	1/0

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
IRR2		割込み要求レジスタ 2	H'FFF7	H'00 (初期設定時)
	IRRTFL	：IRRTFL = 1 の状態で IRRTFL に 0 をライトしたときにクリアできる ：8 ビットタイマモードで TCFL と OCRFL が一致したときに 1 がセットされる	ビット 2	1/0
	IRRTC	：IRRTC = 1 の状態で IRRTC に 0 をライトしたときにクリアできる ：タイマ C のカウンタ値がオーバフロー (H'FF→H'00)，またはアンダフロー (H'00→H'FF) したときに 1 がセットされる	ビット 1	1/0
SYSCR1		システムコントロールレジスタ 1 ：低消費電力モードの制御を行なう	H'FFF0	H'F7
	SSBY	ソフトウェアスタンバイ ：1 アクティブモードで SLEEP 命令実行後，スタンバイモードあるいはウォッチモードに遷移	ビット 7	1
	STS2	スタンバイタイマセレクト 2～0 ：7 待機時間を 16 ステートとする	ビット 6	1
	STS1		ビット 5	1
	STS0		ビット 4	1
	LSON	ロースピードオンフラグ ：0 CPU の動作クロックをシステムクロック (φ) とする	ビット 3	0
	MA1	アクティブ (中速) モードクロックセレクト ：3 φosc/128	ビット 1	1
	MA0		ビット 0	1
PDR4		ポートデータレジスタ 4 ：ポート 4 の汎用入出力ポートデータレジスタ	H'FFD7	H'00
PCR4		ポートコントロールレジスタ 4 ：ポート 4 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 PCR4 = H'00 のとき， ：P47～P40 端子は汎用入力端子として機能	H'FFE7	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
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'03
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'03 のとき， ：P61～P60 端子は汎用出力端子として機能 ：P67～P62 端子は汎用入力端子として機能	H'FFE9	H'03

4.4 使用 RAM 説明

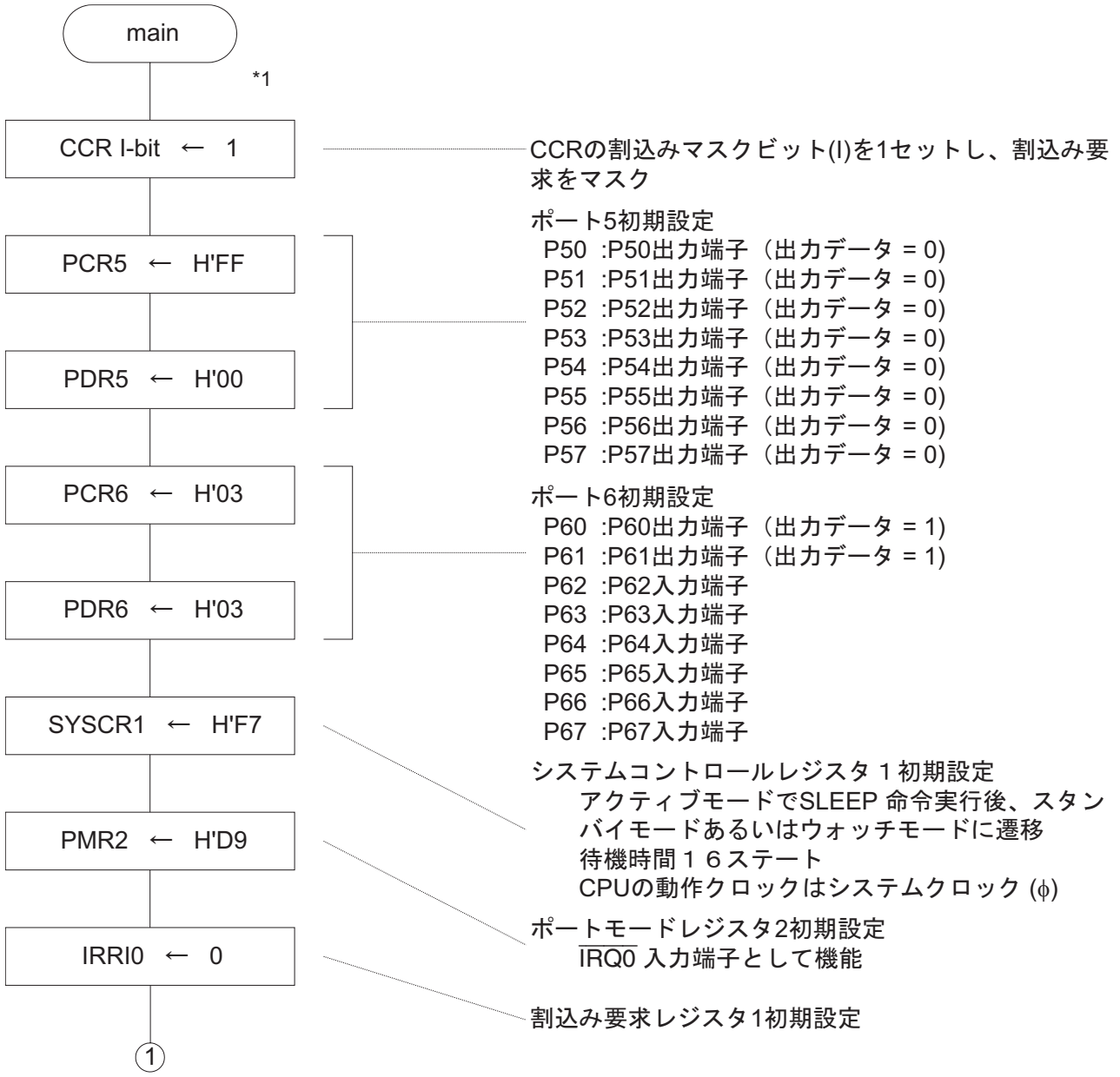
表 5 に本タスク例における使用 RAM 説明を示します。

表 5　使用 RAM 説明

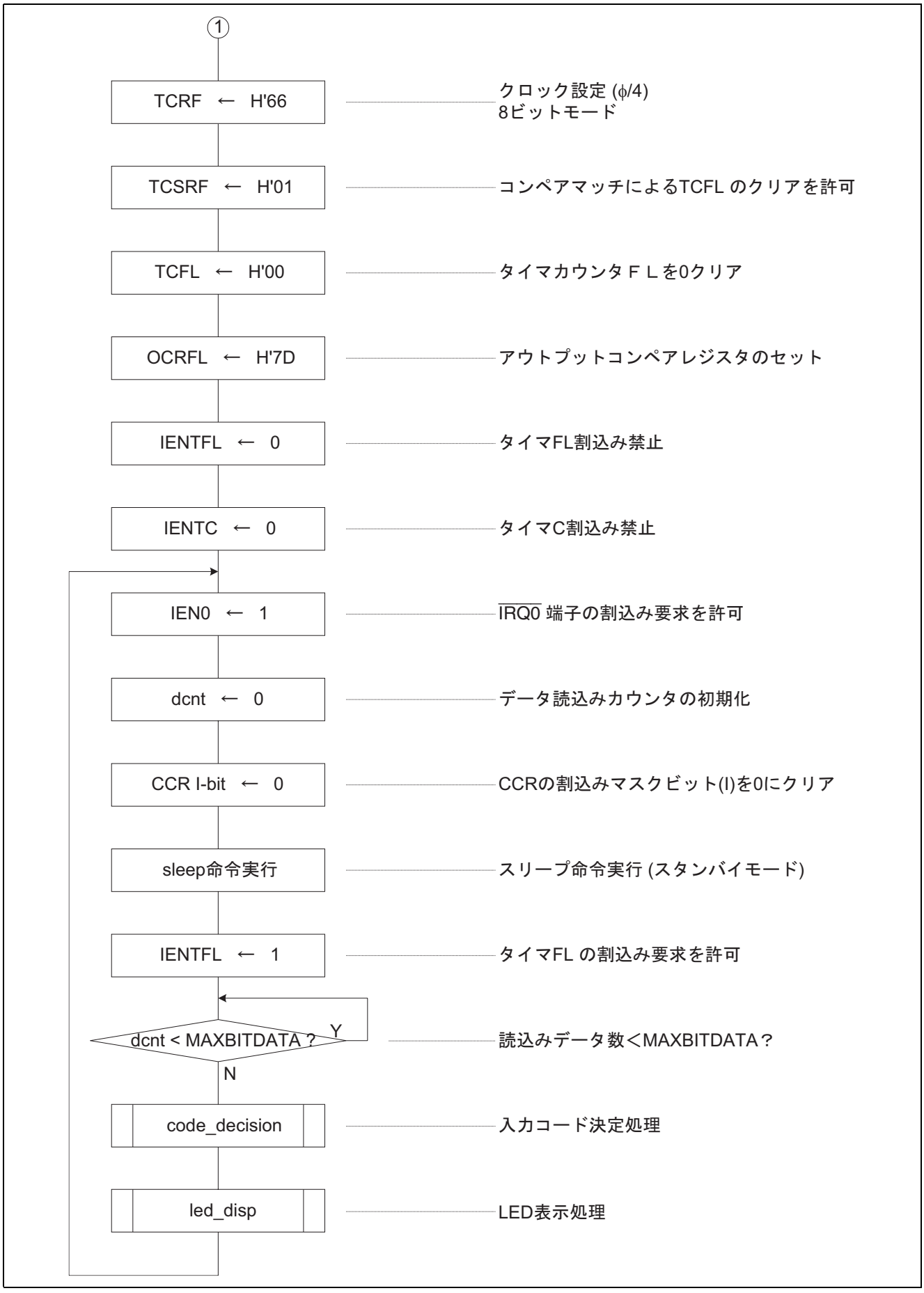
ラベル名	機能	アドレス	使用モジュールラベル名
dig_0	LED1 の表示データを格納 (1 byte)	H'FB86	led_disp,tmrc
dig_1	LED2 の表示データを格納 (1 byte)	H'FB87	led_disp,tmrc
cnt	LED1～LED2 の表示切換えのための 8 ビットカウンタ (1 byte)	H'FB88	tmrc
i	ループカウンタを格納 (2 byte)	H'FB80	code_decision, delay
ptr	LED1～LED2 の表示切換え時に使用するポインタ (2 byte)	H'FB82	tmrc
dcnt	受信時のビットデータカウンタ (2 byte)	H'FB84	main, tmrf
data	受信時にビットデータを格納 (700 byte)	H'FB89	code_decision, tmrf
leddata	ビットデータから抽出したコード部分を格納 (100 byte)	H'FE45	code_decision, led_disp
led_cnt	leddata の表示桁を格納(1 byte)	H'FEAA	led_disp
bit_cnt	ON/OFF を行なうビット位置を格納 (1 byte)	H'FEAB	code_decision
pulse_cnt	1 パルスにおける High をカウント(1 byte)	H'FEAC	code_decision
byte_cnt	leddata 数をカウント(1 byte)	H'FEAD	code_decision, led_disp

5. フローチャート

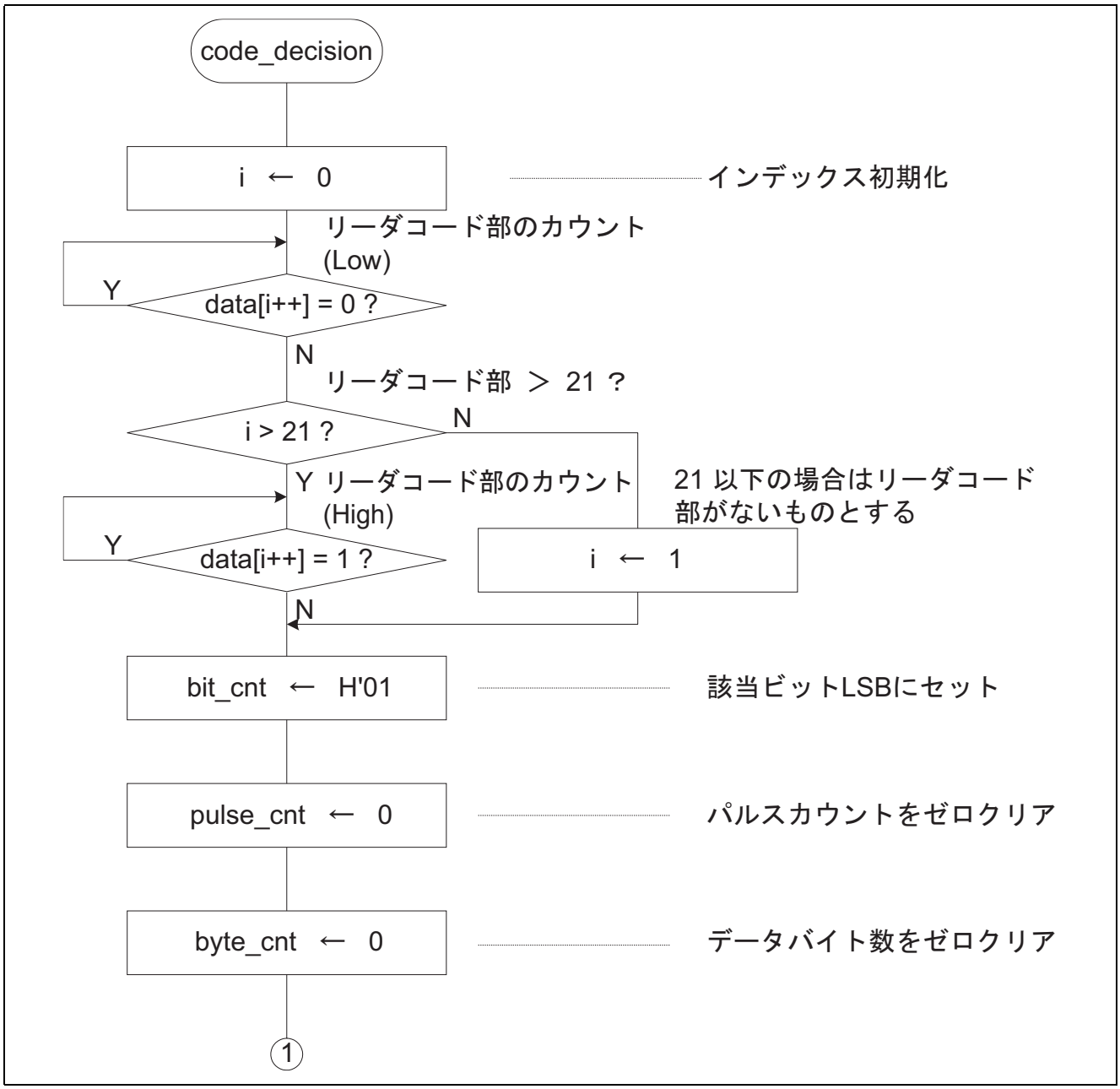
5.1 メインルーチン (main)

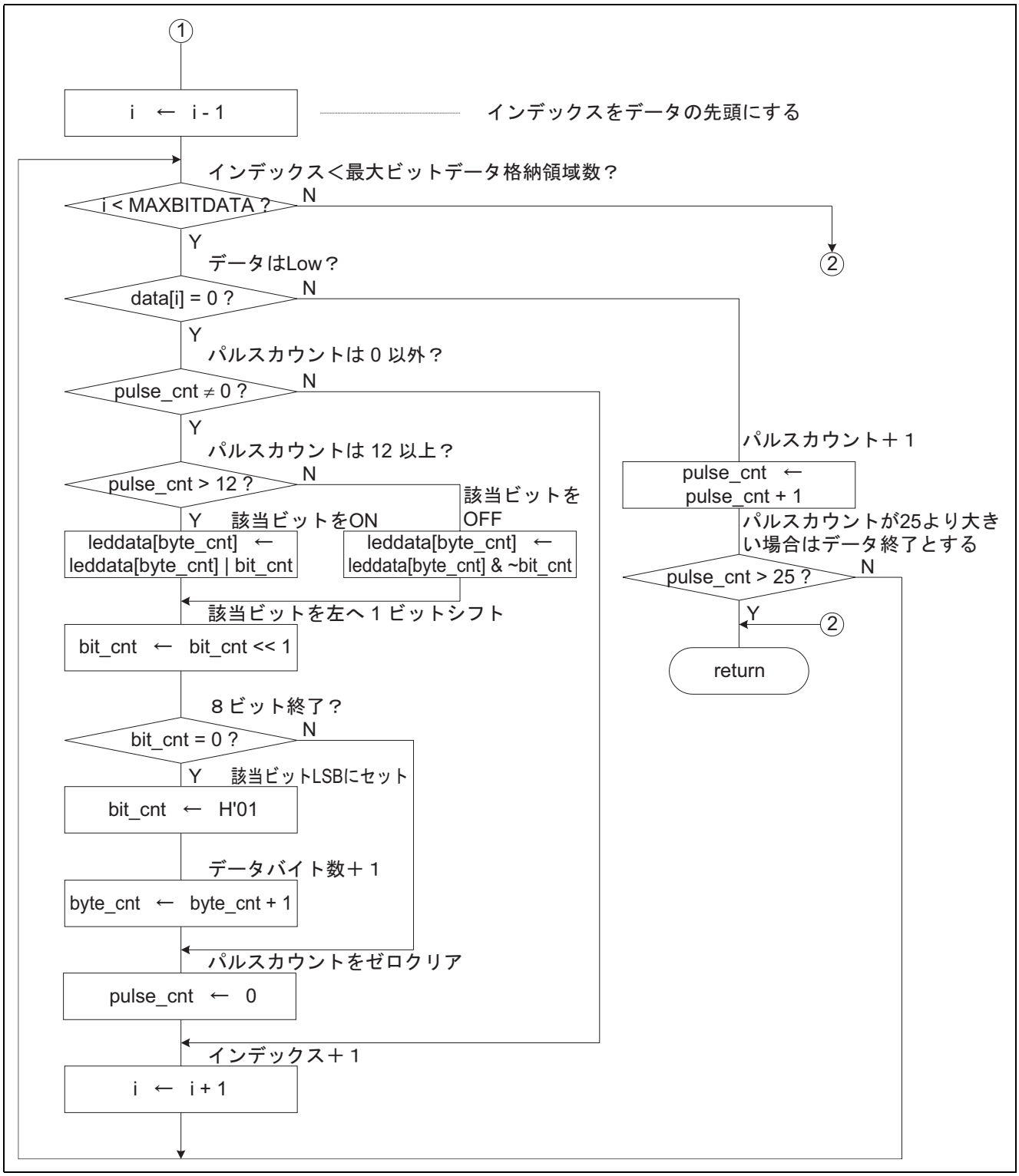


【注】*1：本タスク例ではスタックポインタの設定は INIT.SRC (アセンブリ言語) で行なっています。

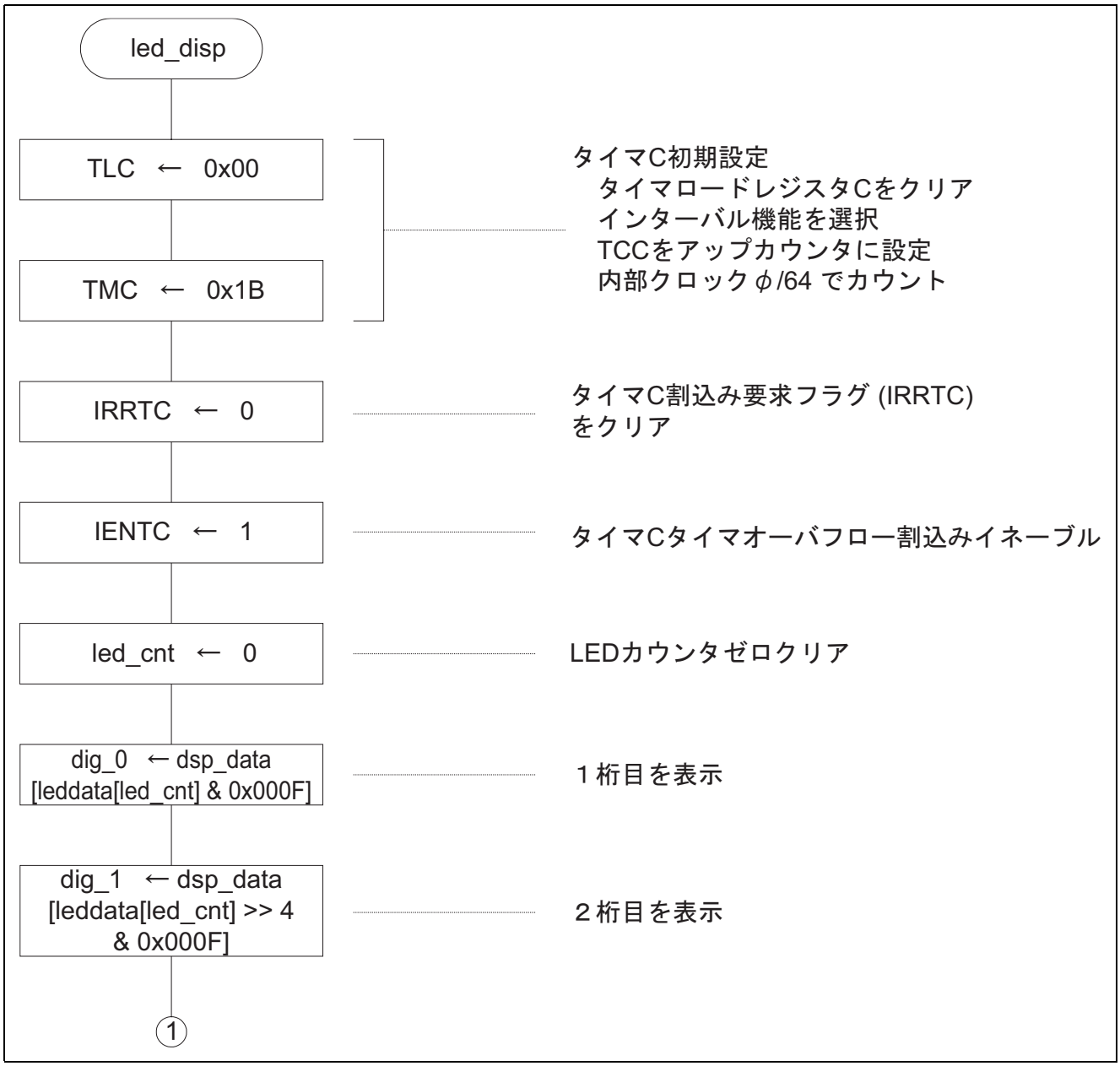

①
TCRF ← H'66
クロック設定 (φ/4)
8ビットモード
TCSRF ← H'01
コンペアマッチによるTCFL のクリアを許可
TCFL ← H'00
タイマカウンタＦＬを0クリア
OCRFL ← H'7D
アウトプットコンペアレジスタのセット
IENTFL ← 0
タイマFL割込み禁止
IENTC ← 0
タイマC割込み禁止
IEN0 ← 1
IRQ0 端子の割込み要求を許可
dcnt ← 0
データ読込みカウンタの初期化
CCR I-bit ← 0
CCRの割込みマスクビット(I)を0にクリア
sleep命令実行
スリープ命令実行 (スタンバイモード)
IENTFL ← 1
タイマFL の割込み要求を許可
dcnt < MAXBITDATA ?
Y
N
読込みデータ数＜MAXBITDATA？
code_decision
入力コード決定処理
led_disp
LED表示処理

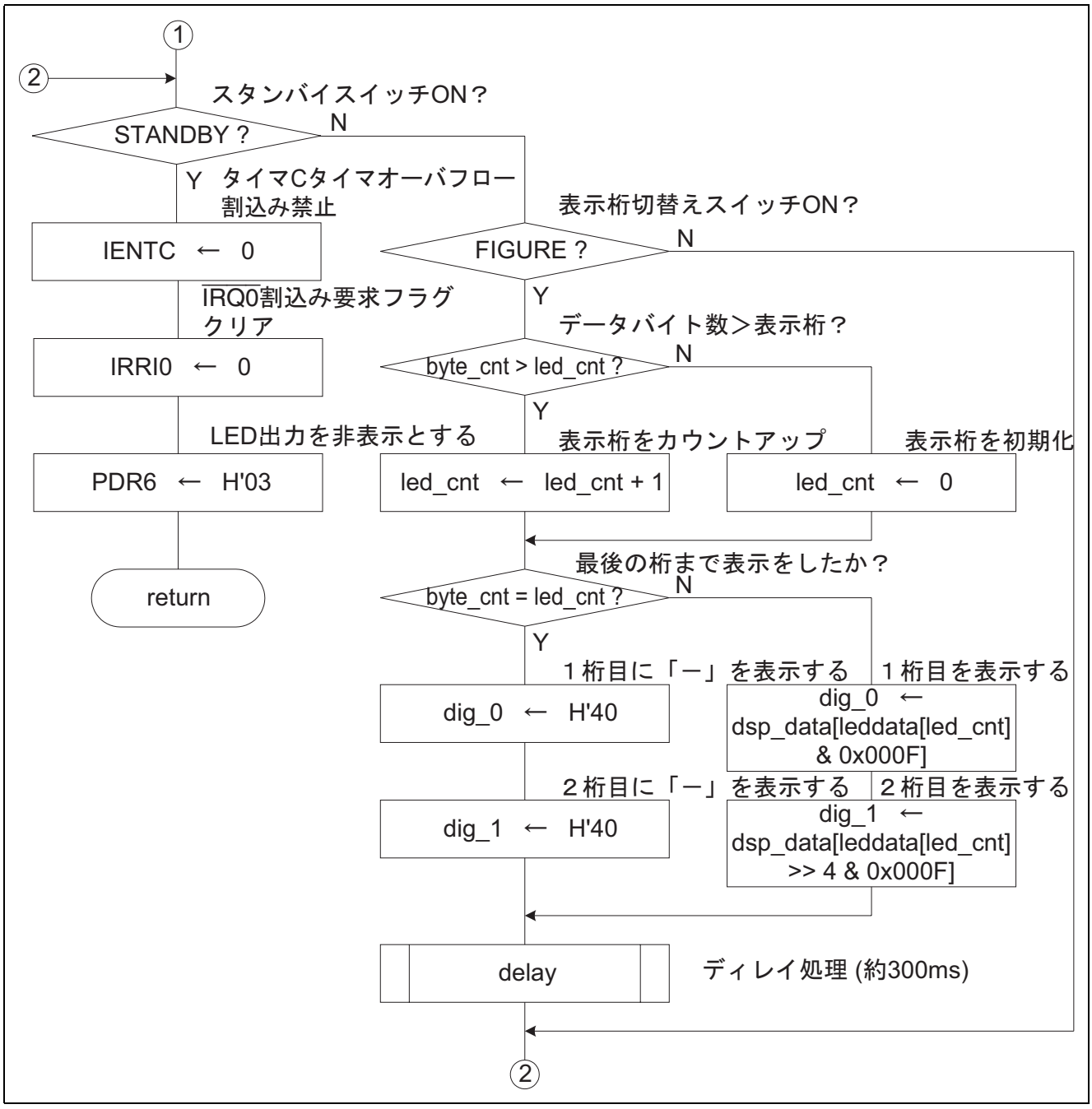
5.2 コード決定処理 (code_decision)



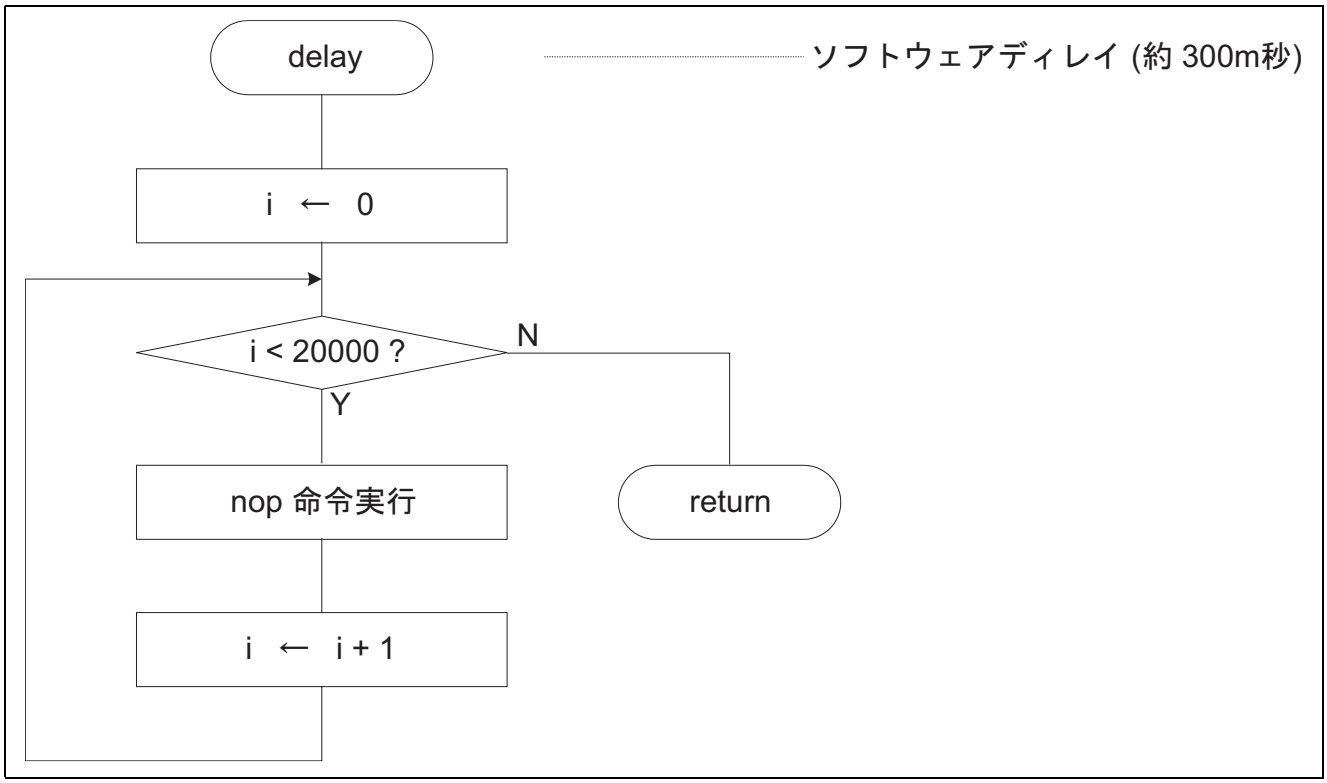

①
i ← i - 1
インデックスをデータの先頭にする
インデックス＜最大ビットデータ格納領域数？
i < MAXBITDATA ?
N
Y
②
データはLow？
data[i] = 0 ?
N
Y
パルスカウントは 0 以外？
pulse_cnt ≠ 0 ?
N
Y
パルスカウントは 12 以上？
pulse_cnt > 12 ?
N
該当ビットを OFF
Y 該当ビットをON
leddata[byte_cnt] ← leddata[byte_cnt] | bit_cnt
leddata[byte_cnt] ← leddata[byte_cnt] & ~bit_cnt
該当ビットを左へ 1 ビットシフト
bit_cnt ← bit_cnt << 1
８ビット終了？
bit_cnt = 0 ?
N
Y 該当ビットLSBにセット
bit_cnt ← H'01
データバイト数＋ 1
byte_cnt ← byte_cnt + 1
パルスカウントをゼロクリア
pulse_cnt ← 0
インデックス＋ 1
i ← i + 1
パルスカウント＋ 1
pulse_cnt ← pulse_cnt + 1
パルスカウントが25より大きい場合はデータ終了とする
pulse_cnt > 25 ?
N
Y
②
return

5.3 LED 表示処理 (led_disp)

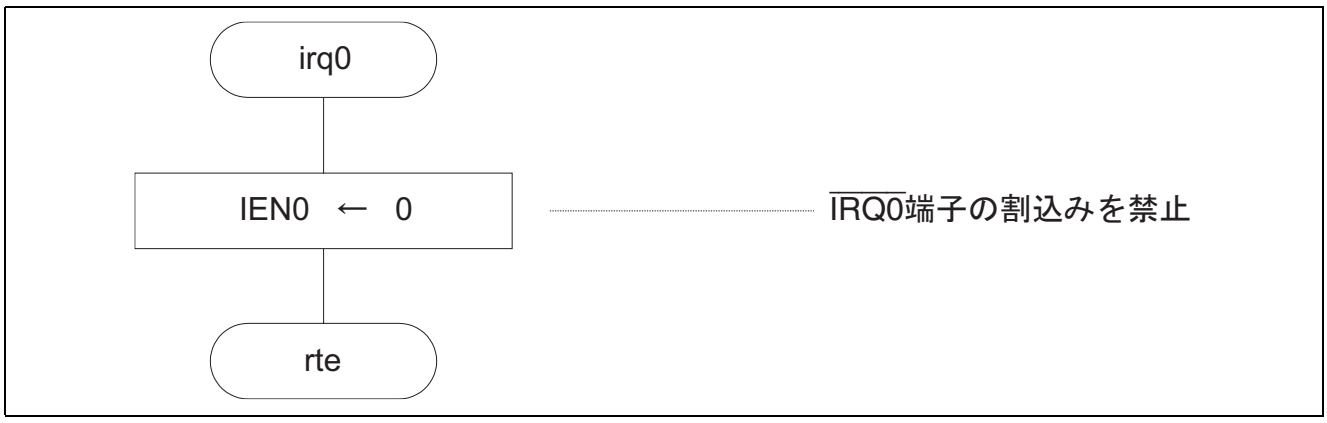



1
2
スタンバイスイッチON？
STANDBY ？
N
Y
タイマCタイマオーバフロー
割込み禁止
IENTC ← 0
IRQ0割込み要求フラグ
クリア
IRRI0 ← 0
LED出力を非表示とする
PDR6 ← H'03
return
表示桁切替えスイッチON？
FIGURE ？
N
Y
データバイト数＞表示桁？
byte_cnt > led_cnt ？
N
Y
表示桁をカウントアップ
led_cnt ← led_cnt + 1
表示桁を初期化
led_cnt ← 0
最後の桁まで表示をしたか？
byte_cnt = led_cnt ？
N
Y
１桁目に「－」を表示する
dig_0 ← H'40
１桁目を表示する
dig_0 ← dsp_data[leddata[led_cnt] & 0x000F]
２桁目に「－」を表示する
dig_1 ← H'40
２桁目を表示する
dig_1 ← dsp_data[leddata[led_cnt] >> 4 & 0x000F]
delay
ディレイ処理 (約300ms)
2

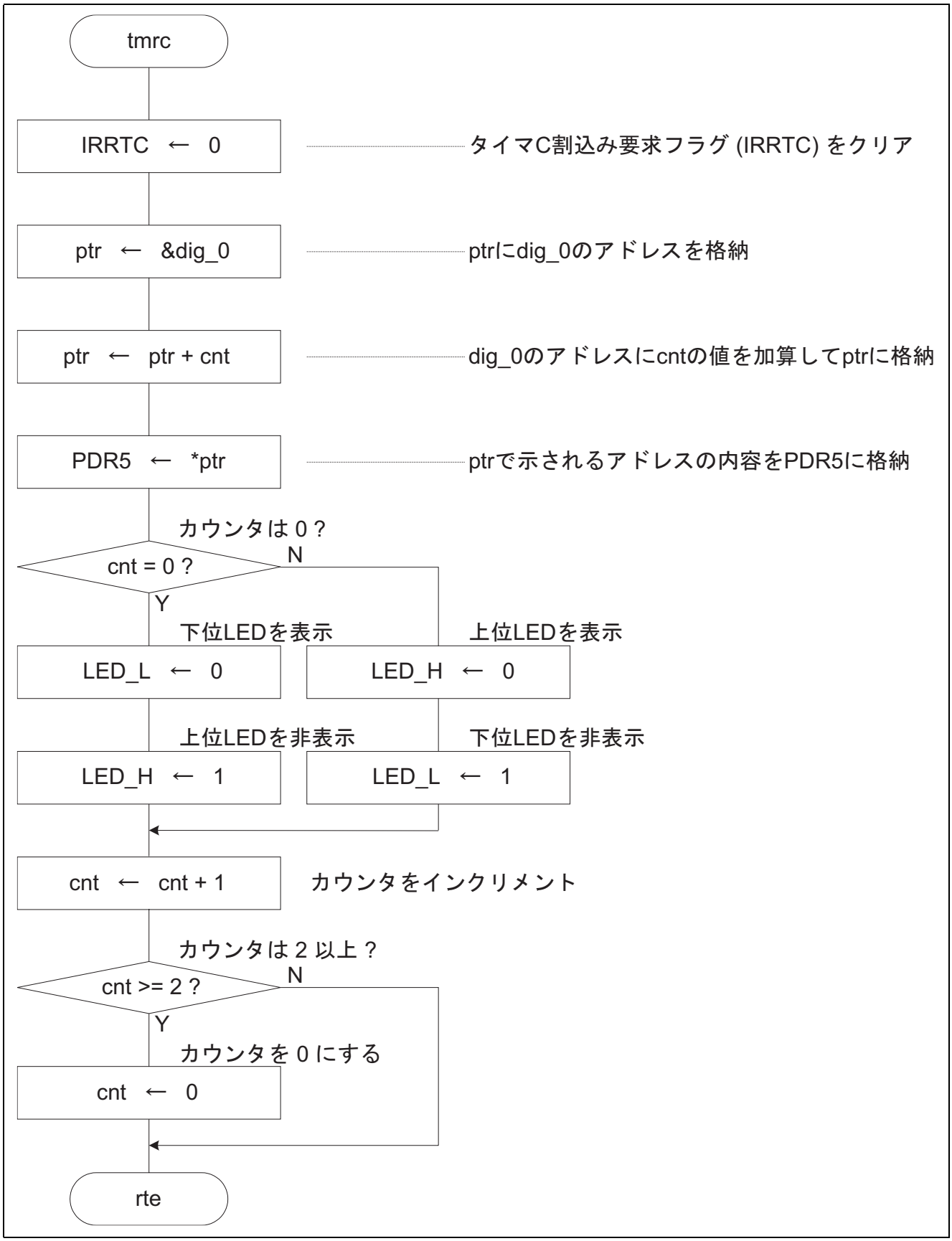
5.4 ソフトウェアディレイ処理 (delay)



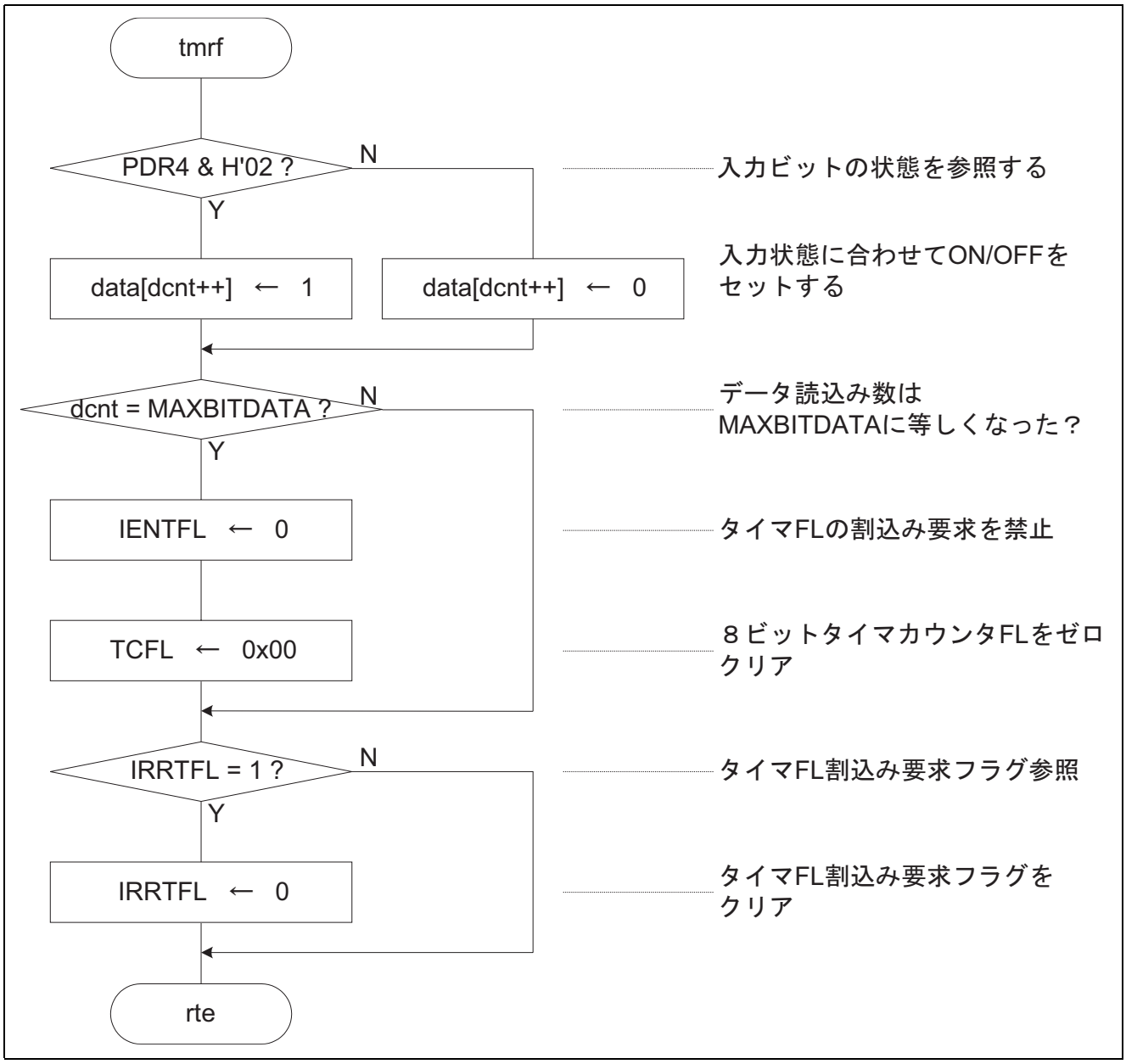
5.5 $\overline{\text{IRQ0}}$ 割込み処理 (irq0)



5.6 タイマ C 割込み処理 (tmrc)



5.7 タイマ F 割込み処理 (tmrf)



6. プログラムリスト

INIT.SRC (プログラムリスト)

```
    .export _INIT
    .import _main
;
    .section    P,CODE
_INIT:
    mov.w   #h'ff80,r7
    ldc.b   #b'10000000,ccr
    jmp @_main
;
    .end
```

```
/* Super Low Power Series -H8/38024- Application note */
/* 応用編 */
/* リモコン */

#include <machine.h>

/* Symbol Definition */
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 H   1                                       /* High Level */
#define L   0                                       /* Low Level */
#define MAXBITDATA 700                              /* max bit data size */
#define MAXLEDDATA 100                              /* max led data size */

#define PDR4    *(volatile unsigned char *)0xFFD7   /* Port data register 4 */
#define PCR4    *(volatile unsigned char *)0xFFE7   /* Port control register 4 */
#define PMR2    *(volatile unsigned char *)0xFFC9   /* Port mode register 2 */

#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 PDR6    *(volatile unsigned char *)0xFFD9   /* Port data register 6 */
#define PDR6_BIT (*(struct BIT *)0xFFD9)
#define STANDBY    PDR6_BIT.b3                          /* standby switch */
#define FIGURE PDR6_BIT.b2                          /* LED figure switch */
#define LED_H  PDR6_BIT.b1                          /* LED(HIGH) ON/OFF */
#define LED_L  PDR6_BIT.b0                          /* LED(LOW) ON/OFF */
#define PCR6    *(volatile unsigned char *)0xFFE9   /* Port control register 6 */

#define TCRF    *(volatile unsigned char *)0xFFB6   /* timer control register F */
#define TCSRF   *(volatile unsigned char *)0xFFB7   /* timer control status register F */
```

```
#define TCSRF_BIT (*(struct BIT *)0xFFB7)
#define CMFL   TCSRF_BIT.b2                          /* Compare-Match Flag L */
#define TCFL   *(volatile unsigned char *)0xFFB9     /* 8 bit timer counter F(LOW)  */
#define OCRFL  *(volatile unsigned char *)0xFFBB     /* 8 bit output compare register F(LOW)  */

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

#define IENR1  *(volatile unsigned char *)0xFFF3     /* Interrupt enable register 1 */
#define IENR1_BIT (*(struct BIT *)0xFFF3)
#define IEN0   IENR1_BIT.b0                          /* IRQ0 interrupt enable */
#define IRR1   *(volatile unsigned char *)0xFFF6     /* Interrupt request register 1 */
#define IRR1_BIT (*(struct BIT *)0xFFF6)
#define IRRI0  IRR1_BIT.b0                           /* IRQ0 interrupt request flag */

#define IENR2  *(volatile unsigned char *)0xFFF4     /* interrupt enable register 2 */
#define IENR2_BIT (*(struct BIT *)0xFFF4)
#define IENTFL IENR2_BIT.b2                          /* Timer FL interrupt enable */
#define IENTC  IENR2_BIT.b1                          /* Timer C interrupt enable */
#define IRR2   *(volatile unsigned char *)0xFFF7     /* interrupt request register 2 */
#define IRR2_BIT (*(struct BIT *)0xFFF7)
#define IRRTFL IRR2_BIT.b2                           /* Timer FL interrupt enable */
#define IRRTC  IRR2_BIT.b1                           /* Timer C interrupt request flag */

#define SYSCR1 *(volatile unsigned char *)0xFFF0     /* system control register 1 */
#define SYSCR2 *(volatile unsigned char *)0xFFF1     /* system control register 2 */


#pragma interrupt (irq0)
#pragma interrupt (tmrc)
#pragma interrupt (tmrf)

/* Function define */
extern void INIT(void);                                 /* Stack pointer set */
void main(void);                                     /* main routine */
void code_decision(void);                            /* code decision routine */
void led_disp(void);                                 /* LED display routine */
void delay(void);                                    /* delay routine */
void irq0(void);                                     /* IRQ0 routine */
void tmrc(void);                                     /* Timer C interrupt routine */
void tmrf(void);                                     /* Timer F interrupt routine */

/* 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 cnt;                              /* LED enable counter */
int i;                                          /* loop counter    */
unsigned char *ptr;                                 /* Pointer set */
int dcnt;                                       /* read data counter */
unsigned char data[MAXBITDATA];                     /* read data(bit data) */
unsigned char leddata[MAXLEDDATA];                  /* read data(led data) */
unsigned char led_cnt;                          /* led display counter */
unsigned char bit_cnt;                          /* 8bit counter */
unsigned char pulse_cnt;                        /* pulse counter */
unsigned char byte_cnt;                             /* byte counter */

/* 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) = {
    irq0                                        /* H'0008 IRQ0 vector */
};
#pragma section V3                              /* Vector section set */
void (*const VEC_TBL3[])(void) = {
    tmrc                                        /* H'001a Timer C interrupt vector */
};
#pragma section V4                              /* Vector section set */
void (*const VEC_TBL4[])(void) = {
    tmrf                                        /* H'001c Timer F interrupt vector */
};
#pragma section                                     /* P */

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


    PCR5 = 0xFF;                                /* Initialize : output LED */
    PDR5 = 0x00;                                /* LED clear */
    PCR6 = 0x03;                                /* Initialize : input SW & LED control*/
    PDR6 = 0x03;                                /* LED OFF */

    SYSCR1 = 0xF7;                              /* standby mode, 16 state */

    PMR2 = 0xD9;                                /* Initialize : use IRQ0 */
    IRRI0 = 0;                                  /* clear IRQ0 interrupt request flag */
```

```
    TCRF = 0x66;                                    /* Set intrernal clock : φ/4 */
    TCSRF = 0x01;                                   /* Enable TCFL clear */
    TCFL = 0x00;                                    /* clear Timer Counter FL to 0       */
    OCRFL = 0x7D;                                   /* set interrupt interval to 0.1msec  */

    IENTFL = 0;                                     /* Timer FL interrupt disable */
    IENTC = 0;                                      /* Timer C interrupt disable */

    while(1){
        IEN0 = 1;                                   /* IRQ0 enable */

        dcnt = 0;                                   /* clear read data count */

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

        IENTFL = 1;                                 /* enable interrupt request FL */

        while(dcnt < MAXBITDATA);

        code_decision();                            /* code decision routine */

        led_disp();                                 /* LED display routine */
    }
}

/*********************************************************/
/*   code decision routine                                */
/*********************************************************/
void code_decision(void)
{
    i = 0;
    while(data[i++] == 0);                          /* Leader Code */
    if(i > 21)                                      /* then leader cord follows */
        while(data[i++]);                           /* Leader Code */
    else                                            /* else data code */
        i = 1;                                      /* initialize index */

    bit_cnt = 0x01;                                     /* set bit counter to LSB */
    pulse_cnt = 0;                                  /* pulse counter to zero clear */
    byte_cnt = 0;                                   /* byte counter to zero clear */
    for(i=i-1;i<MAXBITDATA;i++){
        if(data[i] == L){                           /* Low ? */
            if(pulse_cnt){
                if(pulse_cnt > 12)
                    leddata[byte_cnt] |= bit_cnt;   /* bit on */
                else
                    leddata[byte_cnt] &= ~bit_cnt;      /* bit off */
                bit_cnt <<= 1;                      /* bit shift */
                if(!bit_cnt){                       /* next byte ? */
                    bit_cnt = 0x01;                     /* set bit counter to LSB */
                    byte_cnt++;                     /* count up */
                }
                pulse_cnt = 0;                      /* pulse counter to zero clear */
            }
        } else {                                    /* High */
            pulse_cnt++;                            /* count up */
```

```
            if(pulse_cnt > 25)
                break;
        }
    }
}

/*********************************************************/
/*   LED display routine                                 */
/*********************************************************/
void led_disp(void)
{
    TLC   = 0x00;                                   /* Clear Timer Load register C to 0 */
    TMC   = 0x1B;                                   /* Timer C initialize */
    IRRTC = 0;                                      /* Clear IRRTC to 0 */
    IENTC = 1;                                      /* Timer C interrupt enable */

    led_cnt = 0;
    dig_0 = dsp_data[leddata[led_cnt]  & 0x0F];          /* Dig-0 LED display data set */
    dig_1 = dsp_data[leddata[led_cnt]  >> 4 & 0x0F]; /* Dig-1 LED display data set */
    while(1){
        if(STANDBY){                                /* standby switch ON ? */
            delay();
            IENTC = 0;                              /* Timer C interrupt disable */
            IRRI0 = 0;                              /* clear IRQ0 interrupt request flag */
            PDR6 = 0x03;                            /* LED OFF */
            break;
        } else if(FIGURE){                          /* LED figure switch ON ? */
            if(byte_cnt > led_cnt)
                led_cnt++;                          /* next byte */
            else
                led_cnt = 0;                        /* start byte */
            if(byte_cnt == led_cnt){
                dig_0 = 0x40;                                   /* Dig-0 LED display data set(-) */
            dig_1 = 0x40;                                      /* Dig-1 LED display data set(-) */
            } else {
                dig_0 = dsp_data[leddata[led_cnt]  & 0x0F];          /* Dig-0 LED display data set */
            dig_1 = dsp_data[leddata[led_cnt]  >> 4 & 0x0F]; /* Dig-1 LED display data set */
            }
            delay();
        }
    }
}

/*********************************************************/
/*   delay routine(about 300msec)                        */
/*********************************************************/
void delay(void)
{
    long i;

    for(i = 0;i < 20000;i++)
        nop();
}

/*********************************************************/
/* Interrupt Request 0                                   */
/*********************************************************/
```

```
void irq0(void)
{
    IEN0 = 0;                                   /* IRQ0 disable */
}

/*********************************************************/
/* Timer C Interrupt(in order to light LED in turn)     */
/*********************************************************/
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 */
    if(!cnt){
        LED_L = 0;                              /* LED(LOW) ON */
        LED_H = 1;                              /* LED(HIGH) OFF */
    } else {
        LED_H = 0;                              /* LED(HIGH) ON */
        LED_L = 1;                              /* LED(LOW) OFF */
    }
   cnt++;                                  /* "cnt" increment */
   if (cnt >= 2){                               /* 2 times end ? */
    cnt = 0;                                  /* "cnt" initialize */
  }
}

/*********************************************************/
/* Timer F Interrupt(every 0.1msec)                      */
/*********************************************************/
void tmrf(void)
{
    if(PDR4 & 0x02)
        data[dcnt++] = 1;                       /* bit ON */
    else
        data[dcnt++] = 0;                       /* bit OFF */

    if(dcnt == MAXBITDATA){                         /* end ? */
        IENTFL = 0;                             /* disable interrupt request FL */
        TCFL = 0x00;                            /* clear Timer Counter FL to 0       */
    }
    if ( IRRTFL == 1 ) {
        IRRTFL = 0;                             /* Clear Compare match flag A */
    }
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.07.28	—	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。