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。

M32C/83、85グループ
簡易I2Cバスの応用

RJJ05B0179-0100Z
REV.1.00
2003.08.08

1.0 要約

M32C/83、85グループは、シリアルI/O回路（UART）にI²Cバス回路が搭載されています。I²Cバス回路をソフトウエアと組み合わせて使用することにより、I²Cバスインターフェースの制御が可能となります。本アプリケーションノートでは、I²Cバス仕様の概要を説明し、I²Cバス機能でI²Cバスインターフェースを実現するための各機能の使い方およびプログラムをご紹介致します。

2.0 はじめに

この資料は、ルネサスCMOSマイクロコンピュータM32C/83、85グループに内蔵されているI²Cバスを制御していただくための参考資料です。なお当資料はI²Cバスの通信動作を保証するものではありませんので、ご使用の際には十分に評価を行ってください。
M32C/83、85グループの命令体系については、「M32C/80シリーズ ソフトウェアマニュアル」を合わせてご利用ください。M32C/83、85グループのハードウェアにつきましてはご使用品種のユーザーズマニュアルを、開発サポートツールにつきましては各ツールの操作説明書をご利用ください。
この資料で説明する応用例は次のマイコン、条件での利用に適用されます。

・マイコン　　：M32C/83、85グループ（M3083XXXXP/M3085XXXXP）

本書を使用するにあたって、電気回路、論理回路、およびマイクロコンピュータの基本的な知識が必要です。
本書は3つの章から構成されています。以下に目的に応じた参照先（章、項）を示します。

○M32C/83、85グループのシリアルI/O の構成を知りたい
→ 第1章 UARTの機能 1.1 UARTの有する機能
○M32C/83、85グループのI²Cバスのブロック図およびレジスタ構成を知りたい
→ 第1章 UARTの機能 1.2 ～1.4
○M32C/83、85グループの簡易I²Cバスの各機能の使い方を理解したい
→ 第2章 I²Cバスの各機能
○I²Cバスモードを使用する際に注意することを知りたい
→ 第3章 I²Cバスモードの注意事項
○M32C/83グループを使用したI2Cバスインターフェース部のプログラムを参考にしたい
→ 付録 参考プログラム

3.0 応用例の説明

第一章　M32C/83、85 UARTの機能

第二章　簡易I²Cモードの各機能

第三章　簡易I²Cバスモードの注意事項

付録

第一章
M32C/83、85 UARTの機能

M32C/83、85のシリアルI／Oは、UART0～4の5チャネルで構成されています。これらはそれぞれ専用の転送クロック発生タイマを持ち独立して動作します。本章では、これらのUARTの機能の一つである簡易I²Cバスモードの設定について詳しく説明します。

1.1 シリアルI／Oの有する機能

UART0～4は、同一の機能を持ちます。どのチャネルでもIE-BusインターフェイスおよびI²CバスインタフェースがUse可能です。

UART0～4は、クロック同期形シリアルI／Oモード、もしくは、クロック非同期形シリアルI／Oモード、もしくは、I²CインタフェイスモードのいずれかをSelectして使用します。

M32C/83、85は、IEBusインターフェイスの実現の為に、バス衝突検出等の機能を持ちます。この機能の詳細はM32C/83、85のデータシートをご覧下さい。

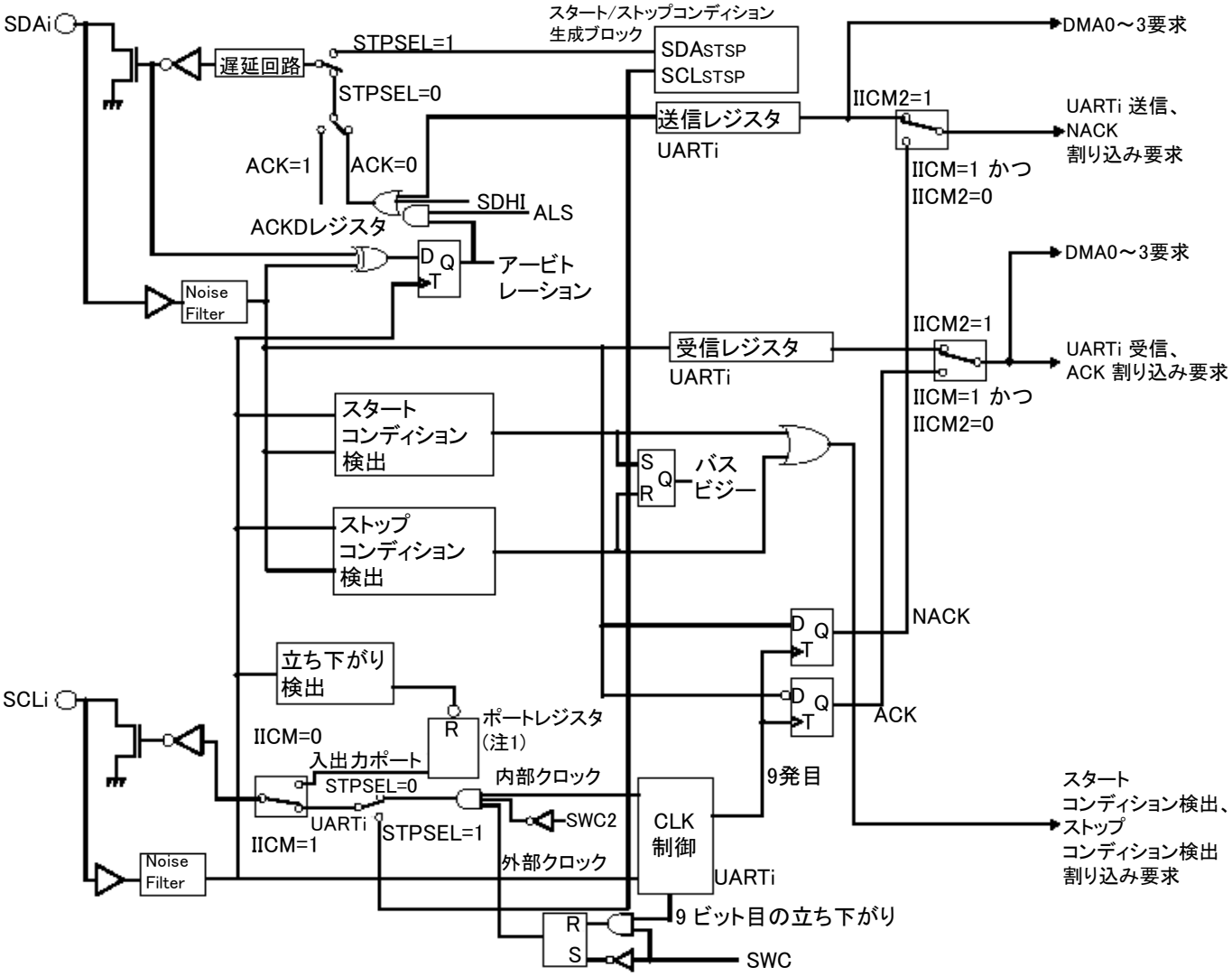
本章ではM32C/83、85の簡易I²Cのブロック図や各種関連レジスタを説明し、次章では、M32C/83、85で、I²Cバスインタフェースを実現するための各種機能を説明します。

M32C/83、85のシリアルI／Oの構成

- UART0
  - クロック同期形シリアルI／O
  - **I²Cインタフェース対応**
  - クロック非同期形シリアルI／O
  - IE-Busインターフェイス対応
- UART1
  - クロック同期形シリアルI／O
  - **I²Cインタフェース対応**
  - クロック非同期形シリアルI／O
  - IE-Busインターフェイス対応
- UART2
  - クロック同期形シリアルI／O
  - **I²Cインタフェース対応**
  - クロック非同期形シリアルI／O
  - IE-Busインターフェイス対応
- UART3
  - クロック同期形シリアルI／O
  - **I²Cインタフェース対応**
  - クロック非同期形シリアルI／O
  - IE-Busインターフェイス対応
- UART4
  - クロック同期形シリアルI／O
  - **I²Cインタフェース対応**
  - クロック非同期形シリアルI／O
  - IE-Busインターフェイス対応

1.2 簡易I²Cバスモードブロック図

I²Cバスインターフェースを実現するには、簡易I²Cバスモードを使用します。
簡易I²C バスモードは、SMD0～2を010Bと設定し、I²Cモード選択ビット[IICM] を"1"に設定することで
I²Cバスインターフェースを実現するための回路を有効にします。
以下に簡易I²Cバスモードのブロック図を示します。



i=0～4
この図は、UiMR レジスタのSMD2～0 ビット=010_2 、UiSMR レジスタのIICM ビット=1 の場合です。
IICM ：UiSMR レジスタのビット
IICM2 ：UiSMR2 レジスタのビット
(注1)IICM ビットが"1 "の場合は、方向ビットが"1 "(出力)であっても、端子が読めます。

1.3 簡易I²Cバスモードで変化する端子の機能および割り込み要因

簡易I²Cバスモード選択時の各機能を示します。

端子の機能

	簡易I²C バスモード	通常モード
P6_3端子機能	SDA0（入出力）	TxD0（出力）
P6_3出力の初期値	シリアルI/O 無効時、P6_2 に設定した値	H レベル(CLK 極性選択ビット=0時)
P6_2端子機能	SCL0（入出力）	RxD0（入力）
P6_2端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ=0 の時、端子をリード
P6_1端子機能	ポートP6_1	CLK0
P6_7端子機能	SDA1（入出力）	TxD1（出力）
P6_7出力の初期値	シリアルI/O 無効時、P6_7 に設定した値	H レベル(CLK 極性選択ビット=0時)
P6_6端子機能	SCL1（入出力）	RxD1（入力）
P6_6端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ=0 の時、端子をリード
P6_5端子機能	ポートP6_5	CLK1
P7_0端子機能	SDA2（入出力）	TxD2（出力）
P7_0出力の初期値	シリアルI/O 無効時、P7_0 に設定した値	H レベル(CLK 極性選択ビット=0時)
P7_1端子機能	SCL2（入出力）	RxD2（入力）
P7_1端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ=0 の時、端子をリード
P7_2端子機能	ポートP7_2	CLK2
P9_2端子機能	SDA3（入出力）	TxD3（出力）
P9_2出力の初期値	シリアルI/O 無効時、P9_2 に設定した値	H レベル(CLK 極性選択ビット=0時)
P9_1端子機能	SCL3（入出力）	RxD3（入力）
P9_1端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ=0 の時、端子をリード
P9_0端子機能	ポートP9_0	CLK3
P9_6端子機能	SDA4（入出力）	TxD4（出力）
P9_6出力の初期値	シリアルI/O 無効時、P9_6 に設定した値	H レベル(CLK 極性選択ビット=0時)
P9_7端子機能	SCL4（入出力）	RxD4（入力）
P9_7端子のリード	方向レジスタに関係なく端子をリード(*)	方向レジスタ=0 の時、端子をリード
P9_5端子機能	ポートP9_5	CLK4

ポートに対するビット処理命令の注意事項

入出力ポートのデータレジスタ（ポートラッチ）をビット処理命令を用いて書き替える場合、
指定していないビットの値が変化することがあります。
（理由）ビット処理命令はリード・モディファイ・ライト形式の命令で、バイト単位で読み出し
及び書き込みを行います。従って、入出力ポートのデータレジスタのあるビットに
対してこの命令を実行した場合、そのデータレジスタの全ビットに対して以下の処理が
行われます。
・入力に設定されているビット：
端子の値がCPU に読み込まれ、ビット処理後、このビットに書き込まれる。
・出力に設定されているビット：
データレジスタのビットの値がCPU に読み込まれ、ビット処理後、このビットに書き込まれる。
(*)ポートに対するリード・モディファイ・ライト命令を行った場合、SCL 、SDA の出力値を
変化させてしまう場合がありますのでご注意ください。

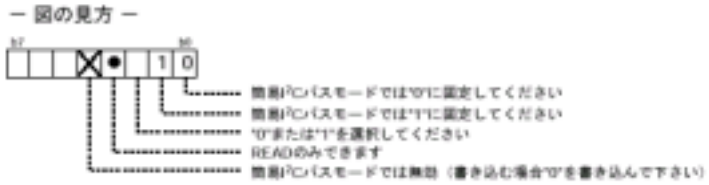
割込要因

(i=0～4)

機能	簡易I²Cバスモード(IICM=1)		通常モード(IICM=0)
	[IICM2] =0	[IICM2] =1	
割込番号39、40、41の要因注1	スタート・ストップコンディション検出	スタート・ストップコンディション検出	バス衝突検出
割り込み番号17,19,33,35,37の要因	アクノリッジ未検出	UARTi送信	UARTi 送信
割り込み番号18,20,34,36,38の要因	アクノリッジ検出	UARTi受信	UARTi 受信
DMA要因	アクノリッジ検出	UARTi受信	UARTi 受信

注1　割込番号40、41の要因は、それぞれUART0/3およびUART1/4を選択して使用します。

1.4 簡易I²Cバスモード時のレジスタ設定



UARTi送信バッファレジスタ(注1)　(i=0～4)

(b15)　(b8)　b7　b0　b7　b0

シンボル	アドレス	リセット時
U0TB	$036B_{16}$、$036A_{16}$番地	不定
U1TB	$02EB_{16}$、$02EA_{16}$番地	不定
U2TB	$033B_{16}$、$033A_{16}$番地	不定
U3TB	$032B_{16}$、$032A_{16}$番地	不定
U4TB	$02FB_{16}$、$02FA_{16}$番地	不定

ビットシンボル	機能	R	W
――	送信データ(bit8はACKとなります)	--	○
--	何も配置されていない 書き込む場合、"0 "を書き込んでください。読み出した場合、その値は不定	--	--

注1 このレジスタの書き込みはMOV命令を使用して下さい。

UARTi受信バッファレジスタ　(i=0～4)

(b15)　(b8)　b7　b0　b7　b0

シンボル	アドレス	リセット時
U0RB	$036F_{16}$、$036E_{16}$番地	不定
U1RB	$02EF_{16}$、$02EE_{16}$番地	不定
U2RB	$033F_{16}$、$033E_{16}$番地	不定
U3RB	$032F_{16}$、$032E_{16}$番地	不定
U4RB	$02FF_{16}$、$02FE_{16}$番地	不定

ビットシンボル	ビット名	機能	R	W
――	受信データ(bit8はACKもしくは、R/Wビットとなります。)		○	--
--	何も配置されていない 書き込む場合、"0 "を書き込んでください。読み出した場合、その値は"0"		--	--
ABT	アビトレーションロストフラグ (注1)	0：未検出(勝) 1：検出(負)	○	○
OER	オーバーランエラーフラグ (注2)	0：オーバーランエラー無し 1：オーバーランエラー有り	○	--
FER	フレーミングエラーフラグ	簡易I²Cモードでは無効	○	--
PER	パリティーエラーフラグ	簡易I²Cモードでは無効	○	--
SUM	エラーサムフラグ	簡易I²Cモードでは無効	○	--

注1 "0"のみ書き込み可能です。
注2 OERはシリアルI/Oモード選択ビット(0368_{16},$02E8_{16}$,0338_{16},0328_{16},$02F8_{16}$番地のbit2～0)を
"000_2"にしたとき、または受信許可ビットを"0"にした時に"0"になります。

UARTi転送速度レジスタ(注1、2)　(i=0～4)

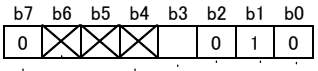
b7　b0

シンボル	アドレス	リセット時
U0BRG	0369_{16}番地	不定
U1BRG	$02E9_{16}$番地	不定
U2BRG	0339_{16}番地	不定
U3BRG	0329_{16}番地	不定
U4BRG	$02F9_{16}$番地	不定

ビットシンボル	機能	設定可能範囲	R	W
――	設定値をnとすると、BRGはカウントソースをn+1分周する	00_{16}～FF_{16}	--	○

注1 このレジスタへの書き込みはMOV命令を使用してください。
注2 送信停止中に書き込んでください。

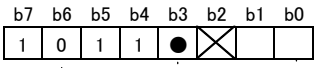
UARTi送受信モードレジスタ (i=0～4)



シンボル	アドレス	リセット時
U0MR	0368_{16}番地	00_{16}
U1MR	$02E8_{16}$番地	00_{16}
U2MR	0338_{16}番地	00_{16}
U3MR	0328_{16}番地	00_{16}
U4MR	$02F8_{16}$番地	00_{16}

ビットシンボル	ビット名	機能	R	W
SMD0	シリアルI/Oモード選択ビット	000：シリアルI／O無効(ポート制御)	○	○
SMD1		010：簡易I²Cモード	○	○
SMD2		I²C使用時は"010"に設定してください	○	○
CKDIR	内/外部クロック選択ビット	0：内部クロック 1：外部クロック	○	○
STPS	ストップビット長選択ビット	簡易I²Cバスモードでは無効	○	○
PRY	パリティ奇/偶選択ビット	簡易I²Cバスモードでは無効	○	○
PRYE	パリティ許可ビット	簡易I²Cバスモードでは無効	○	○
IOPOL	TxD,RxD 入出力極性切り替えビット	0：反転あり 1：反転なし I²C使用時は"0"に設定してください	○	○

UARTi送受信制御レジスタ0 (i=0～4)

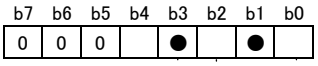


シンボル	アドレス	リセット時
U0C0	$036C_{16}$番地	08_{16}
U1C0	$02EC_{16}$番地	08_{16}
U2C0	$033C_{16}$番地	08_{16}
U3C0	$032C_{16}$番地	08_{16}
U4C0	$02FC_{16}$番地	08_{16}

ビットシンボル	ビット名	機能	R	W
CLK0	BRGカウントソース選択ビット	00：f1を選択 01：f8を選択 10：f2nを選択 11：設定しないでください	○	○
CLK1			○	○
CRS	$\overline{CTS}/\overline{RTS}$機能選択ビット	簡易I²Cバスモードでは無効	○	○
TXEPT	送信レジスタ空フラグ	0：送信レジスタにデータあり(送信中) 1：送信レジスタにデータなし(送信完了)	○	--
CRD	$\overline{CTS}/\overline{RTS}$禁止ビット	0：$\overline{CTS}/\overline{RTS}$機能許可 1：$\overline{CTS}/\overline{RTS}$機能禁止 簡易I²Cバスモードでは"1"を設定してください	○	○
NCH	データ出力選択ビット(注1)	0：TXD端子はCMOS出力 1：TXD端子はNchオープンドレイン出力 簡易I²Cバスモードでは"1"を設定してください	○	○
CKPOL	クロック極性ビット	0：転送クロックの立ち下がりで送信データ出力、立ち上がりで受信データ入力 1：転送クロックの立ち上がりで送信データ出力、立ち下がりで受信データ入力 簡易I²Cバスモードでは"0"を設定してください	○	○
UFORM	転送フォーマット選択ビット	0：LSBファースト 1：MSBファースト 簡易I²Cバスモードでは"1"を設定してください	○	○

注1 UART2の送信端子(TXD2：P70)はNchオープンドレイン端子です。CMOS出力は設定出来ません。

UARTi送受信制御レジスタ1 (i=0～4)

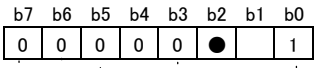


シンボル	アドレス	リセット時
U0C1	$036D_{16}$番地	02_{16}
U1C1	$02ED_{16}$番地	02_{16}
U2C1	$033D_{16}$番地	02_{16}
U3C1	$032D_{16}$番地	02_{16}
U4C1	$02FD_{16}$番地	02_{16}

ビットシンボル	ビット名	機能	R	W
TE	送信許可ビット	0:送信禁止 1:送信許可	○	○
TI	送信レジスタ空フラグ	0:送信バッファレジスタにデータあり 1:送信バッファレジスタにデータなし	○	--
RE	受信許可ビット	0:受信禁止 1:受信許可	○	○
RI	受信完了フラグ	0:受信バッファレジスタにデータなし 1:受信バッファレジスタにデータあり	○	--
UiIRS	UARTi送信 割り込み要因ビット(注1)	0:送信バッファ空 (TI=1) 1:送信完了(TXEPT=1)	○	○
UiRRM	UARTi連続受信モード 許可ビット	0:連続受信モード禁止 1:連続受信モード許可 簡易I²Cバスモードでは"0"を設定してください	○	○
UiLCH	データ論理選択ビット	0:反転なし 1:反転あり 簡易I²Cバスモードでは"0"を設定してください	○	○
UiERE	クロック分周比 同期化停止ビット	0:同期化停止 1:同期化開始 簡易I²Cバスモードでは"0"を設定してください	○	○

注1 [IICM]=1 かつ [IICM2] =0 のときは [UiIRS] は 無効となります。

UARTi特殊モードレジスタ (i=0～4)

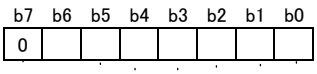


シンボル	アドレス	リセット時
U0SMR	0367_{16}番地	00_{16}
U1SMR	$02E7_{16}$番地	00_{16}
U2SMR	0337_{16}番地	00_{16}
U3SMR	0327_{16}番地	00_{16}
U4SMR	$02F7_{16}$番地	00_{16}

ビットシンボル	ビット名	機能	R	W	説明章
IICM	IICモード選択ビット(注1)	0:通常モード 1:簡易I²Cモード	○	○	1.3
ABC	アービトレーション ロスト検出フラグ制御	0:ビット毎に更新 1:バイト毎に更新	○	○	2.5
BBS	バスビジーフラグ(注2)	0:ストップコンディション検出 1:スタートコンディション検出	○	○	2.2
LSYN	SCL同期出力許可ビット	0:禁止 1:許可 簡易I²Cバスモードでは"0"を設定してください	○	○	----
ABSCS	バス衝突検出サンプリング クロック選択ビット	"0"を設定してください	○	○	----
ACSE	送信許可ビット自動クリア 機能選択ビット	"0"を設定してください	○	○	----
SSS	送信開始条件選択ビット	"0"を設定してください	○	○	----
SCLKDIV	クロック分周設定ビット	"0"を設定してください	○	○	----

注1 簡易I²Cバスモード選択時はシリアルI/O モード選択ビットを"010_2"にしてください。
注2 "0"のみ書き込みできます。

UARTi特殊モードレジスタ2 (i=0～4)



シンボル	アドレス	リセット時
U0SMR2	0366_{16}番地	00_{16}
U1SMR2	$02E6_{16}$番地	00_{16}
U2SMR2	0336_{16}番地	00_{16}
U3SMR2	0326_{16}番地	00_{16}
U4SMR2	$02F6_{16}$番地	00_{16}

ビットシンボル	ビット名	機能	R	W	説明章
IICM2	IICモード選択ビット2	表1参照	○	○	――――
CSC	クロック同期化ビット	0：禁止 1：許可	○	○	2.5
SWC	SCLウエイト出力ビット	0：禁止 1：許可	○	○	2.4
ALS	SDA出力停止ビット	0：禁止 1：許可	○	○	2.5
STAC	UARTi初期化ビット	0：禁止 1：許可	○	○	2.6
SWC2	SCLウエイト出力ビット2	0：UARTiクロック 1：0出力	○	○	2.2
SDHI	SDA出力禁止ビット	0：許可 1：禁止(ハイインピーダンス)	○	○	2.6
SU1HIM	クロック分周 同期化有効ビット	0：クロック同期化無効 1：クロック同期化有効 簡易I²Cバスモードでは"0"を設定してください	○	○	――――

表1　簡易I²Cバスモード(IICM=1)時の機能

機能	[IICM2] =0	[IICM2] =1
割込番号39、40、41の要因注1	スタート・ストップコンディション検出	スタート・ストップコンディション検出
割り込み番号17,19,33,35,37の要因	アクノリッジ未検出	UARTi送信
割り込み番号18,20,34,36,38の要因	アクノリッジ検出	UARTi受信
DMA要因	アクノリッジ検出	UARTi受信
UARTi受信シフトレジスタから受信 バッファへのデータ転送タイミング	受信クロックの最終ビットの 立ち上がり	受信クロックの最終ビットの 立ち下がり
UARTi受信／アクノリッジ検出 割り込み要求発生タイミング	受信クロックの最終ビットの 立ち上がり(アクノリッジ検出)	受信クロックの最終ビットの 立ち下がり(UARTi受信)

注1　割込番号40、41の要因は、それぞれUART0/3およびUART1/4を選択して使用します。

UARTi特殊モードレジスタ3 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0		0

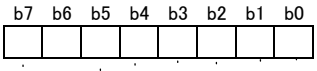
シンボル	アドレス	リセット時
U0SMR3	0365_{16}番地	00_{16}
U1SMR3	$02E5_{16}$番地	00_{16}
U2SMR3	0335_{16}番地	00_{16}
U3SMR3	0325_{16}番地	00_{16}
U4SMR3	$02F5_{16}$番地	00_{16}

ビットシンボル	ビット名	機能	R	W	説明章
SSE	$\overline{SS}$端子機能許可ビット	0：$\overline{SS}$機能禁止 1：$\overline{SS}$機能許可 簡易I²Cバスモードでは"0"を設定してください	○	○	----
CKPH	クロック位相設定ビット	0：クロック遅れなし 1：クロック遅れあり	○	○	2.6
DINC	シリアル入力端子設定ビット	0：TxDi,RxDiを選択(マスタモード) 1：STxDi,SRxDiを選択(スレーブモード) 簡易I²Cバスモードでは"0"を設定してください	○	○	----
NODC	クロック出力選択ビット	0：CLKiはCMOS出力 1：CLKiはNchオープンドレイン出力 簡易I²Cバスモードでは"0"を設定してください	○	○	----
ERR	障害エラーフラグ	0：障害エラーなし 1：障害エラーあり 簡易I²Cバスモードでは"0"を設定してください	○	○	----
DL0	SDAi(TxDi)ディジタル遅延値設定ビット(注1)(注2)	000：遅延なし 001：BRGカウントソースの2サイクル 010：BRGカウントソースの3サイクル 011：BRGカウントソースの4サイクル 100：BRGカウントソースの5サイクル 101：BRGカウントソースの6サイクル 110：BRGカウントソースの7サイクル 111：BRGカウントソースの8サイクル	○	○	2.6
DL1			○	○	
DL2			○	○	

注1　本ビットはIIC インタフェースとしてUARTi を使用する際、SDAi(TxDi)出力にディジタル的に遅延を発生させるも のです。
それ以外の場合は、必ず"000_2"に設定してください。

注2　外部クロックを選択した場合、+100ns程度遅延が大きくなります。

UARTi特殊モードレジスタ4 (i=0～4)



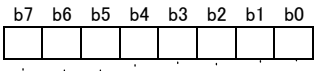
シンボル	アドレス	リセット時
U0SMR4	0364_{16}番地	00_{16}
U1SMR4	$02E4_{16}$番地	00_{16}
U2SMR4	0334_{16}番地	00_{16}
U3SMR4	0324_{16}番地	00_{16}
U4SMR4	$02F4_{16}$番地	00_{16}

ビットシンボル	ビット名	機能	R	W	説明章
STAREQ	スタートコンディション生成ビット(注1)	0:クリア 1:スタート	○	○	2.2
RSTREQ	リスタートコンディション生成ビット(注1)	0:クリア 1:スタート	○	○	2.2
STPREQ	ストップコンディション生成ビット(注1)	0:クリア 1:スタート	○	○	2.2
STSPSEL	SCL、SDA出力選択ビット	0:従来ブロック 1:スタート/ストップコンディション生成ブロック	○	○	2.2
ACKD	ACKデータビット	0:ACK 1:NACK	○	○	2.3
ACKC	ACKデータ出力許可ビット	0:SI/Oデータ出力 1:ACKデータビット(ACKD)出力	○	○	2.3
SCLH	SCL出力停止許可ビット	0:禁止 1:許可	○	○	2.6
SWC9	SCLウェイト出力ビット3	0:SCL″L″ホールド禁止 1:SCL″L″ホールド許可	○	○	2.6

注1 各コンディションが生成された場合、自動的に″0″になります。

外部割り込み要因選択レジスタ

シンボル	アドレス	リセット時
IFSR	$031F_{16}$番地	00_{16}



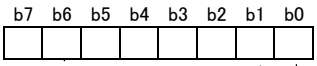
ビットシンボル	ビット名	機能	R	W
IFSR0	INT0割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR1	INT1割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR2	INT2割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR3	INT3割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR4	INT4割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR5	INT5割り込み極性選択ビット(注1)(注2)	0 : 片エッジ 1 : 両エッジ	○	○
IFSR6	UART0/3 割り込み要因選択ビット	0:UART3のバス衝突/スタートストップ検出/障害エラー検出要因選択 1:UART0のバス衝突/スタートストップ検出/障害エラー検出要因選択	○	○
IFSR7	UART1/4 割り込み要因選択ビット	0:UART4のバス衝突/スタートストップ検出/障害エラー検出要因選択 1:UART1のバス衝突/スタートストップ検出/障害エラー検出要因選択	○	○

注1 これらのビットは、簡易I²Cバスに関係ありません。

注2 レベルセンスを選択した場合、このビットは"0"に設定してください。
両エッジを選択する場合、対応するINT 割り込み制御レジスタの極性切り替えビット(bit4)は、「立ち下りエッジ」0に設定してください。

機能選択レジスタA0

シンボル	アドレス	リセット時
PS0	$03B0_{16}$番地	00_{16}

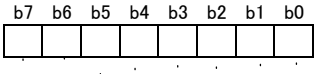


ビットシンボル	ビット名	機能	R	W
PS0_0	ポートP60出力機能選択ビット(注1)	0:入出力ポート 1:UART0出力($\overline{RTS0}$)	○	○
PS0_1	ポートP61出力機能選択ビット(注1)	0:入出力ポート 1:UART0出力(CLK0出力)	○	○
PS0_2	ポートP62出力機能選択ビット	0:入出力ポート 1:PSL0_2で選択された機能	○	○
PS0_3	ポートP63出力機能選択ビット	0:入出力ポート 1:UART0出力(TXD0/SDA0)	○	○
PS0_4	ポートP64出力機能選択ビット(注1)	0:入出力ポート 1:PSL0_4で選択された機能	○	○
PS0_5	ポートP65出力機能選択ビット(注1)	0:入出力ポート 1:UART1出力(CLK1出力)	○	○
PS0_6	ポートP66出力機能選択ビット	0:入出力ポート 1:PSL0_6で選択された機能	○	○
PS0_7	ポートP67出力機能選択ビット	0:入出力ポート 1:UART1出力(TXD1/SDA1)	○	○

注1 これらのビットは、簡易I²Cバスに関係ありません。

機能選択レジスタA1

シンボル	アドレス	リセット時
PS1	$03B1_{16}$番地	00_{16}



ビットシンボル	ビット名	機能	R	W
PS1_0	ポートP70出力機能選択ビット	0:入出力ポート 1:PSL1_0で選択された機能	○	○
PS1_1	ポートP71出力機能選択ビット	0:入出力ポート 1:PSL1_1で選択された機能	○	○
PS1_2	ポートP72出力機能選択ビット(注1)	0:入出力ポート 1:PSL1_2で選択された機能	○	○
PS1_3	ポートP73出力機能選択ビット	0:入出力ポート 1:PSL1_3で選択された機能	○	○
PS1_4	ポートP74出力機能選択ビット(注1)	0:入出力ポート 1:PSL1_4で選択された機能	○	○
PS1_5	ポートP75出力機能選択ビット(注1)	0:入出力ポート 1:PSL1_5で選択された機能	○	○
PS1_6	ポートP76出力機能選択ビット(注1)	0:入出力ポート 1:PSL1_6で選択された機能	○	○
PS1_7	ポートP77出力機能選択ビット(注1)	0:入出力ポート 1:インテリジェントI/O グループ0出力(OUTC01/ISCLK0)	○	○

注1 これらのビットは、簡易I²Cバスに関係ありません。

機能選択レジスタA3(注1)

シンボル	アドレス	リセット時
PS3	$03B5_{16}$番地	00_{16}

b7	b6	b5	b4	b3	b2	b1	b0

ビットシンボル	ビット名	機能	R	W
PS3_0	ポートP90出力機能選択ビット(注2)	0:入出力ポート 1:UART3 出力(CLK3)	○	○
PS3_1	ポートP91出力機能選択ビット	0:入出力ポート 1:PSL3_1で設定された機能	○	○
PS3_2	ポートP92出力機能選択ビット	0:入出力ポート 1:PSL3_2で設定された機能	○	○
PS3_3	ポートP93出力機能選択ビット(注2)	0:入出力ポート 1:UART3出力($\overline{RTS3}$)	○	○
PS3_4	ポートP94出力機能選択ビット(注2)	0:入出力ポート 1:UART4出力($\overline{RTS4}$)	○	○
PS3_5	ポートP95出力機能選択ビット(注2)	0:入出力ポート 1:UART4出力(CLK4)	○	○
PS3_6	ポートP96出力機能選択ビット	0:入出力ポート 1:UART4出力(TxD4/SDA4)	○	○
PS3_7	ポートP97出力機能選択ビット	0:入出力ポート 1:PSL3_7で設定された機能	○	○

注1 このレジスタを書き替える場合、PRCR レジスタのPRC2 ビット を"1"(書き込み許可)にしてください。
注2 これらのビットは、簡易I²Cバスに関係ありません。

機能選択レジスタB0

シンボル	アドレス	リセット時
PSL0	$03B2_{16}$番地	00_{16}

b7	b6	b5	b4	b3	b2	b1	b0
0		0		0		0	0

ビットシンボル	ビット名	機能	R	W
---	予約ビット	必ず'0'を設定してください。	○	○
---			○	○
PSL0_2	ポートP62出力周辺機能選択ビット	0:UART0出力(SCL0) 1:UART0 出力(STxD0)	○	○
---	予約ビット	必ず'0'を設定してください。	○	○
PSL0_4	ポートP64出力周辺機能選択ビット(注1)	0:UART1出力($\overline{RTS1}$) 1:インテリジェントI/O グループ2出力(OUTC21/ISCLK2)	○	○
---	予約ビット	必ず'0'を設定してください。	○	○
PSL0_6	ポートP66出力周辺機能選択ビット	0:UART1出力(SCL1) 1:UART1出力(STxD1)	○	○
---	予約ビット	必ず'0'を設定してください。	○	○

注1 このビットは、簡易I²Cバスに関係ありません。

機能選択レジスタB1

シンボル	アドレス	リセット時
PSL1	$03B3_{16}$番地	00_{16}

b7	b6	b5	b4	b3	b2	b1	b0
0							

ビットシンボル	ビット名	機能	R	W
PSL1_0	ポートP70出力機能選択ビット	0:PSC1_0で選択された機能 1:タイマ出力(TA0OUT)	○	○
PSL1_1	ポートP71出力機能選択ビット	0:PSC1_1で選択された機能 1:UART2出力(STxD2)	○	○
PSL1_2	ポートP72出力機能選択ビット(注1)	0:PSC1_2で選択された機能 1:タイマ出力(TA1OUT)	○	○
PSL1_3	ポートP73出力機能選択ビット(注1)	0:PSC1_3で選択された機能 1:三相PWM出力(V)	○	○
PSL1_4	ポートP74出力機能選択ビット(注1)	0:PSC1_4で選択された機能 1:三相PWM出力(W)	○	○
PSL1_5	ポートP75出力機能選択ビット(注1)	0:PSC1_5で選択された機能 1:インテリジェントI/O グループ1出力(OUTC12)	○	○
PSL1_6	ポートP76出力機能選択ビット(注1)	0:PSC1_6で選択された機能 1:タイマ出力(TA3OUT)	○	○
---	予約ビット	必ず"0"を設定してください。	○	○

注1 これらのビットは、簡易I²Cバスに関係ありません。

機能選択レジスタB3

シンボル	アドレス	リセット時
PSL3	$03B3_{16}$番地	00_{16}

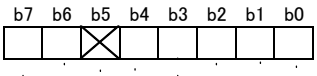
b7	b6	b5	b4	b3	b2	b1	b0
							0

ビットシンボル	ビット名	機能	R	W
---	予約ビット	必ず"0"を設定してください。	○	○
PSL3_1	ポートP91出力機能選択ビット	0:UART3出力(SCL3) 1:UART3出力(STxD3)	○	○
PSL3_2	ポートP92出力機能選択ビット	0:UART3出力(TxD3/SDA3) 1:インテリジェントI/O グループ2出力(OUTC20/IEOUT)	○	○
PSL3_3	ポートP93出力機能選択ビット(注1)(注2)	0:入力周辺機能有効(DA0出力以外) 1:入力周辺機能禁止(DA0出力時)	○	○
PSL3_4	ポートP94出力機能選択ビット(注1)(注2)	0:入力周辺機能有効(DA1出力以外) 1:入力周辺機能禁止(DA1出力時)	○	○
PSL3_5	ポートP95出力機能選択ビット(注1)(注2)	0:入力周辺機能有効(ANEX0使用以外) 1:入力周辺機能禁止(ANEX0使用時)	○	○
PSL3_6	ポートP96出力機能選択ビット	0:入力周辺機能有効(ANEX1使用以外) 1:入力周辺機能禁止(ANEX1使用時)	○	○
PSL3_7	ポートP97出力機能選択ビット(注2)	0:UART4出力(SCL4) 1:UART4出力(STxD4)	○	○

注1 これらのビットは、簡易I²Cバスに関係ありません。
注2 このビットに"0"を設定してもDA0,DA1,ANEX0,ANEX1 は使用できますが、電源電流が増加する場合があります。

機能選択レジスタC

シンボル	アドレス	リセット時
PSC	$03AF_{16}$番地	$00X0\ 0000_2$



ビットシンボル	ビット名	機能	R	W
PSC_0	ポートP70出力周辺機能選択ビット	0 :UART2 出力(TxD2/SDA2) 1 :インテリジェントI/O グループ2 出力(OUTC20/ ISTxD2/IEOUT)	○	○
PSC_1	ポートP71出力周辺機能選択ビット	0 :UART2 出力(SCL2) 1 :インテリジェントI/O グループ2 出力(OUTC22)	○	○
PSC_2	ポートP72出力周辺機能選択ビット(注1)	0 :UART2 出力(CLK2) 1 :三相PWM 出力(V)	○	○
PSC_3	ポートP73出力周辺機能選択ビット(注1)	0 :UART2 出力(RTS2) 1 :インテリジェントI/O グループ1 出力(OUTC10/ ISTxD1/BE1OUT)	○	○
PSC_4	ポートP74出力周辺機能選択ビット(注1)	0 :タイマ出力(TA2OUT) 1 :インテリジェントI/O グループ1 出力(OUTC11/ ISCLK1)	○	○
---	何も配置されていない。書く場合'0'を書き込んでください。読む場合、その値は不定		--	--
PSC_6	ポートP76出力周辺機能選択ビット(注1)	0:インテリジェントI/Oグループ0出力(OUTC00/ISTxD0/BE0OUT) 1:CAN出力(CANOUT)	○	○
PSC_7	ポートP77出力周辺機能選択ビット(注1)	0:キー入力割込信号許可 1:キー入力割込信号禁止	--	○

注1 これらのビットは、簡易I²Cバスに関係ありません。

簡易I2Cモードに使用するシリアルI/O	機能選択レジスタ ビット設定
UART0	PS0_2=1 PS0_3=1 PSL0_2=0
UART1	PS0_6=1 PS0_7=1 PSL0_6=0
UART2	PS1_0=1 PS1_1=1 PSL1_0=0 PSL1_1=0 PSC_0=0 PSC_1=0
UART3	PS3_1=1 PS3_2=1 PSL3_1=0 PSL3_2=0
UART4	PS3_6=1 PS3_7=1 PSL3_6=0

第二章

簡易I2Cバスモードの各機能

本章では、I2Cバスインターフェースを実現するためにM32C/83、85 を簡易I2Cバスモードで使用する場合の、各ハードウエア機能の使用方法について説明します。

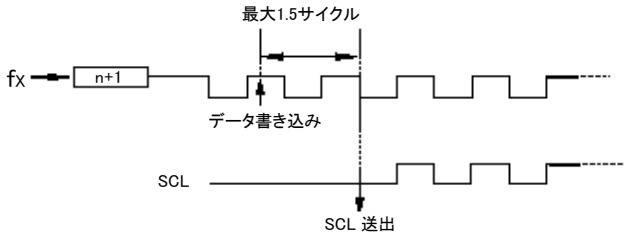
2.1 バイトデータの送信／受信の方法

M32C/83、85をマスタとして使用し簡易I²CバスモードのSCLを送出する設定方法、
また1バイトのデータを送信、または受信する場合の設定方法を説明します。

SCL生成方法（マスタ時）

M32C/83、85をマスタとして使用する場合、送出クロック(SCL)の速度を設定する必要があります。
それらは通常のシリアルI/O 送信と同様に、以下のレジスタで設定します。
SCL は、送信バッファにデータを書き込み後、SCL の1.5 サイクル以内に送出されます。

[SCL送出タイミング]
fx ：BRG カウントソース
n ：UiBRG 設定値(i=0～4)
SCL：CKPH=1の時の波形



[CKPH]の詳細については、2.6項　その他機能　クロック遅延機能を参照してください。

[関連レジスタ]

UARTi送受信モードレジスタ　(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	✕	✕	✕	0	0	1	0

U0MR：0368_{16}番地　U1MR：$02E8_{16}$番地　U2MR：0338_{16}番地
U3MR：0328_{16}番地　U4MR：$02F8_{16}$番地

[SMD]010：簡易I²Cモード
[CKDIR] 0：内部クロックを選択（マスタ時。スレーブ時は 1：外部クロック を選択してください）
[IOPOL] 0：極性反転なし

UARTi特殊モードレジスタ　(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR：0367_{16}番地　U1SMR：$02E7_{16}$番地　U2SMR：0337_{16}番地
U3SMR：0327_{16}番地　U4SMR：02F716番地

[IICM]1：簡易I²Cモード
[ABC] アービトレーションロストを　0：ビット毎に更新　1：バイト毎に更新

UARTi送受信制御レジスタ0　(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
1	0	1	1	●	✕		

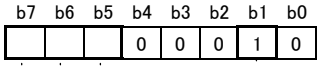
U0C0：$036C_{16}$番地　U1C0：$02EC_{16}$番地　U2C0：$033C_{16}$番地
U3C0：$032C_{16}$番地　U4C0：$02FC_{16}$番地

[CLK1] [CLK0] BRG カウントソースを選択
　0 0 :f1 を選択、0 1 :f8 を選択、1 0 :f2n を選択、1 1 :使用禁止
[CRD] 1 :CTS/RTS 機能禁止
[NCH] 1：SCL、SDA端子はNchオープンドレイン出力(注1)
[CKPOL] 0 :CLK 極性はSCL たち下がりで送信データ出力、立ち上がりで入力
[UFORM] 転送フォーマット 0 ：LSB ファースト 1 ：MSB ファースト

注1　UART2のSDA、SCL端子はNchオープンドレイン端子です。CMOS出力は設定出来ません。
　　この時、bit5は"0"を設定してください。

SCL生成方法（マスタ時）（つづき）

UARTi特殊モードレジスタ3 (i=0～4)

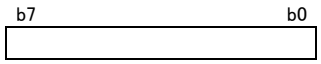


U0SMR3：0365_{16}番地　U1SMR3：$02E5_{16}$番地　U2SMR3：0335_{16}番地
U3SMR3：0325_{16}番地　U4SMR3：$02F5_{16}$番地

[CKPH] 0：クロック遅延なし　1：クロック遅延あり
[DL2][DL1][DL0]デジタル遅延値を設定
000：遅延なし　001：BRG カウントソースの2サイクル
010：BRG カウントソースの3サイクル　011：BRGカウントソースの4サイクル
100：BRG カウントソースの5サイクル　101：BRG カウントソースの6サイクル
110：BRG カウントソースの7サイクル　111：BRG カウントソースの8サイクル

[CKPH]の詳細については、2.6項　その他機能　クロック遅延機能を参照してください。
[DL2][DL1][DL0]の詳細については、2.6項　その他機能　デジタル出力遅延機能を参照してください。

UARTi転送速度レジスタ (i=0～4)



U0BRG：0369_{16}番地　U1BRG：$02E9_{16}$番地　U2BRG：0339_{16}番地
U3BRG：0329_{16}番地　U4BRG：$02F9_{16}$番地

カウントソースをn+1分周

[設定例]
源発振10MHz で使用している場合、送信速度を100kbps に設定する場合は、
・ UiMR =00000010_2（IICモード、内部クロック選択）
・ UiC0 =10010000_2（BRGカウントソースにf1を選択）
・ UiBRG =49

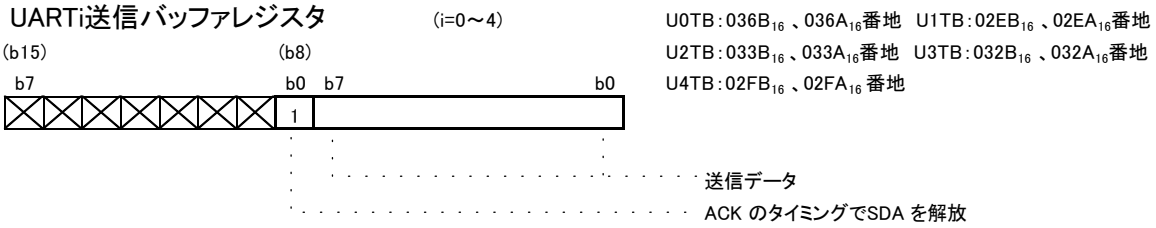
[スレーブ時の設定]
スレーブとして使用する際には、UARTi送受信モードレジスタ(UiMR)のビット3[CKDIR]を
″1″に設定し、外部クロックを選択してください。
その時、BRG カウントソース選択ビット[CLK0] [CLK1]
およびUARTi転送速度レジスタ(UiBRG)の設定は無効となります。

バイトデータの送信方法

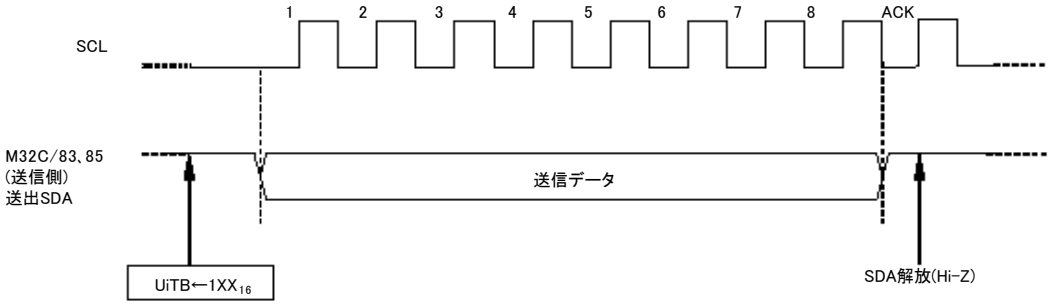
M32C/83、85を送信装置として使用する場合、SDA 端子から8ビットの送信データを送出しますが、送信クロックの9ビット目ではアクノリッジを受信するためにM32C/83、85のSDA端子を解放する必要があります。これは、送信バッファに設定するデータで操作できます。

送信バッファには、9ビットのデータを設定します。I²Cバスでは、MSBファーストでデータを送信します。M32C/83、85では、転送フォーマットをMSBファーストにして9ビットの設定した時、データは b7 → b6 → ・・・→ b0 → b8 の順で送出されます。従って最上位ビットを送出した時が、アクノリッジを受信するタイミングとなります。このときにSDA 端子を解放するためには、最上位ビットに ″1″を設定してM32C/83、85のSDA をハイインピーダンス状態にします。
以上が、バイトデータの送信方法です。

[関連レジスタ]



[タイミング図]



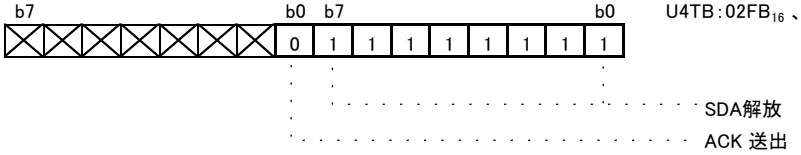
バイトデータの受信方法

M32C/83、85を受信装置として使用する場合、SDA端子から8ビットのデータを受信する間はM32C/83、85のSDA 端子を解放する必要があります。またクロックの9 ビット目では、SDA端子を″L″にしてアクノリッジを生成する必要があります。これは、すでにマスタ側が指定した受信装置のアドレス判定が終了し、自装置に対して送信が行われていることが確定している場合には、送信バッファに書き込む値でアクノリッジの送出を簡単に操作できます。M32C/83、85ではデータ受信時も、送信バッファに9 ビットのデータをダミーデータとして設定します。下位8 ビットを送出している間SDA 端子を解放するためには、下位8 ビットに″1″を設定します。また、アクノリッジ生成のためには、最後に送出するビット(b8)に ″0″を設定します。以上が、バイトデータの受信方法です。

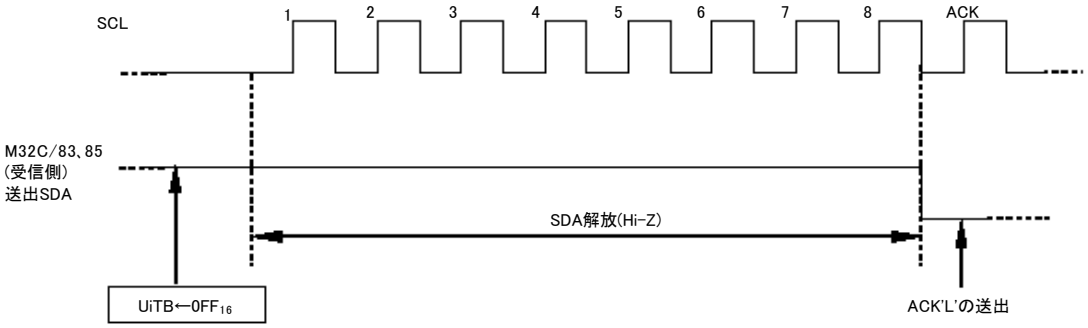
[関連レジスタ]

UARTi送信バッファレジスタ (i=0～4)

U0TB:$036B_{16}$、$036A_{16}$番地 U1TB:$02EB_{16}$、$02EA_{16}$番地
U2TB:$033B_{16}$、$033A_{16}$番地 U3TB:$032B_{16}$、$032A_{16}$番地
U4TB:$02FB_{16}$、$02FA_{16}$番地



[タイミング図]



送信割り込み／受信割り込み

M32C/83、85を送信装置として使用する場合、データ送出の完了は「UARTi送信割込」で検出できます。また、M32C/83、85を受信装置として使用する場合、データ受信の完了は「UARTi受信割込」で検出できます。これらの割込は、割込番号17～20 および割込番号33～38に割り当てられており、I²Cモード選択ビット2[IICM2] を″1″に設定することで割込要因がUARTi送信、およびUARTi受信になります。この場合の送信割込発生タイミングは、UARTi送信割り込み要因選択ビット[UiIRS] が ″0″のときは送信クロックの開始ビットの立ち下がりとなり、[UiIRS] が ″1″のときは次データの1ビット目の立ち下がり([CKPH]=″1″の時)となります。また受信割込発生タイミングは、受信クロックの最終ビットの立ち下がりとなります。
(1.4項「簡易I²Cバスモード時のレジスタ設定」 UARTi特殊モードレジスタ2
表1簡易I²Cバスモード((IICM1=1)時の機能参照)
なお、受信クロックの最終ビットの立ち上がり前(簡易I²Cバスモードの受信完了割込中等)に受信バッファを読み出すと、受信データはビットの位置が変化した状態で読み出されますのでご注意ください。(後述のタイミング図参照ください。)

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0							1

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] 1:UARTi送信／受信割込有効
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:UARTiクロック 1:″L″出力
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
I2Cモード選択時は、“0”に設定してください

UARTi特殊モードレジスタ3 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0	1	0

U0SMR3:0365$_{16}$番地 U1SMR3:02E5$_{16}$番地 U2SMR3:0335$_{16}$番地
U3SMR3:0325$_{16}$番地 U4SMR3:02F5$_{16}$番地

[CKPH] 0:クロック遅延なし 1:クロック遅延あり
[DL2][DL1][DL0]デジタル遅延値を設定
000:遅延なし 001:BRG カウントソースの2サイクル
010:BRG カウントソースの3サイクル 011:BRGカウントソースの4サイクル
100:BRG カウントソースの5サイクル 101:BRG カウントソースの6サイクル
110:BRG カウントソースの7サイクル 111:BRG カウントソースの8サイクル

[CKPH]の詳細については、2.6項 その他機能 クロック遅延機能を参照してください。
[DL2][DL1][DL0]の詳細については、2.6項 その他機能 デジタル出力遅延機能を参照してください。

送信割り込み／受信割り込み（つづき）

UARTi送受信制御レジスタ1 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0		●	1	●	1

U0C1：$036D_{16}$番地　U1C1：$02ED_{16}$番地　U2C1：$033D_{16}$番地
U3C1：$032D_{16}$番地　U4C1：$02FD_{16}$番地

[TE] 1：送信許可
[RI] 1：受信許可
[UiIRS] UARTi送信割込要因選択
　0：送信バッファ空（開始ビットの立ち下がり）1：送信完了（最終ビットの立ち上がり）
[UiRRM] 0：UARTi連続受信 禁止
[UiLCH] 0：データ論理 反転なし
[UiERE] 0：エラー信号出力禁止

UARTi送信割り込み制御レジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
✕	✕	✕	✕	0			

S0TIC：0090_{16}番地　S1TIC：0092_{16}番地　S2TIC：0089_{16}番地
S3TIC：$008B_{16}$番地　S4TIC：$008D_{16}$番地

[ILVL] 割り込み優先レベル
1～7：割り込みレベル設定
（割り込みを使用しないときは"0"(禁止)にしてください）
[IR]　0：割り込み要求発生時"1"にセットされます

UARTi受信割り込み制御レジスタ (i=0～4)

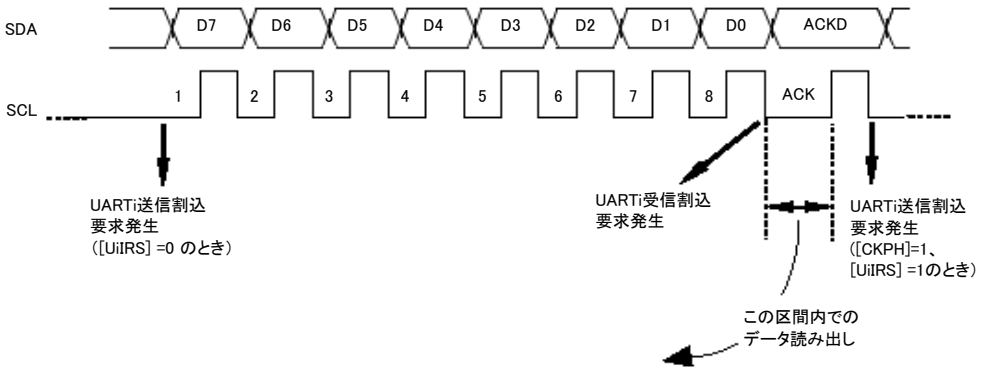
b7	b6	b5	b4	b3	b2	b1	b0
✕	✕	✕	✕	0			

S0RIC：007216番地　S1RIC：007416番地　S2RIC：006B16番地
S3RIC：006D16番地　S4RIC：006F16番地

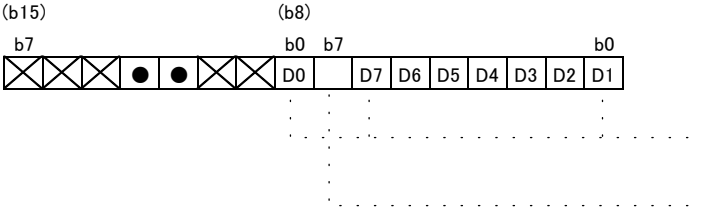
[ILVL] 割り込み優先レベル
1～7：割り込みレベル設定
（割り込みを使用しないときは"0"(禁止)にしてください）
[IR]　0：割り込み要求発生時"1"にセットされます

割り込み要求により割り込みを発生させる場合は、I フラグ を'1'に設定してください。

［タイミング図］（IICM2=1の時）



UARTi受信バッファレジスタ



(i=0～4)
U0RB:036F$_{16}$、036E$_{16}$番地　U1RB:02EF$_{16}$、02EE$_{16}$番地
U2RB:033F$_{16}$、033E$_{16}$番地　U3RB:032F$_{16}$、032E$_{16}$番地
U4RB:02FF$_{16}$、02FE$_{16}$ 番地

受信クロックの8ビット目から受信クロックの最終ビットの立ち上がり前
（簡易I2Cバスモードの受信完了割込中等）にデータを読み出すと
図の様にD0,--,D7～D1と格納されます。
この後、受信クロックの最終ビットの立ち上がり後
（簡易I2Cバスモードの送信完了割込中等）にデータを読み出すと
ACKD、D7～D0とデータが読み出せます。

UARTi受信バッファレジスタ(上記区間後)

(i=0～4)
U0RB:036F$_{16}$、036E$_{16}$番地　U1RB:02EF$_{16}$、02EE$_{16}$番地
U2RB:033F$_{16}$、033E$_{16}$番地　U3RB:032F$_{16}$、032E$_{16}$番地
U4RB:02FF$_{16}$、02FE$_{16}$ 番地

2.2 スタートコンディション／ストップコンディション

データ送受信の開始時には、開始条件（スタートコンディション）、データ送受信の終了時には、
停止条件（ストップコンディション）が発生します。
M32C/83、85をスレーブとして使用する場合、マスタの生成するスタートコンディション
および、ストップコンディションを検出するために、「スタートコンディション／ストップコンディション
検出割込」機能をハードウエアで備えています。
M32C/83、85をマスタとする場合は、スタートコンディションおよびストップコンディションを
生成し送出する機能をハードウェアで備えています。また、スタートコンディション生成時に
バスの使用状態を検知するための機能として、「バスビジー検出」機能を、
スタートコンディション送出後から通信開始までの間に、他デバイスからクロック送出を禁止するための
機能として、SCL から強制的に"L"を出力する「SCL 端子L出力機能2」機能を
ハードウエアで備えています。

スタートコンディション／ストップコンデション検出

SCL がHigh のときにSDA がHigh からLow に変化するスタートコンディション、
およびSCL がHigh のときにSDA がLow からHigh に変化するストップコンディションは、
M32C/83、85では「スタートコンディション／ストップコンディション検出割込」で検出することが
できます。本割込は「ソフトウエア割込番号39～41」に割り当てられ、
I²Cモード選択時（[IICM]="1"）は割り込み番号39～41の割込要因が「スタートコンディション／
ストップコンディション検出割込」に変化します。この割込を検出した場合は、バスビジーフラグ(BBS)の状態で
スタートコンディションまたはストップコンディションのどちらが発生したのかを判断してください。
なお、スタートコンディションおよびストップコンディション検出のセットアップタイム、
ホールドタイムは、I²Cバス規格と異なる場合がありますのでご注意ください
（3.1項「スタート／ストップコンディションのセットアップタイム・ホールドタイム」参照）。

[関連レジスタ]

UARTi特殊モードレジスタ　(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地　U1SMR:02E7$_{16}$番地　U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地　U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを　0:ビット毎に更新　1:バイト毎に更新

UARTiバス衝突検出割り込み制御レジスタ　(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
☒	☒	☒	☒	0			

BCN2IC:008F$_{16}$番地　BCN0IC/BCN3IC:0071$_{16}$番地　BCN1IC/BCN4IC:0091$_{16}$番地

[ILVL] 割り込み優先レベル
1～7:割り込みレベル設定
（割り込みを使用しないときは"0"(禁止)にしてください）

[IR]　0:割り込み要求発生時"1"にセットされます

外部割り込み要因選択レジスタ

シンボル	アドレス
IFSR	031F$_{16}$ 番地

b7	b6	b5	b4	b3	b2	b1	b0

I²Cバスでは、使用しません。

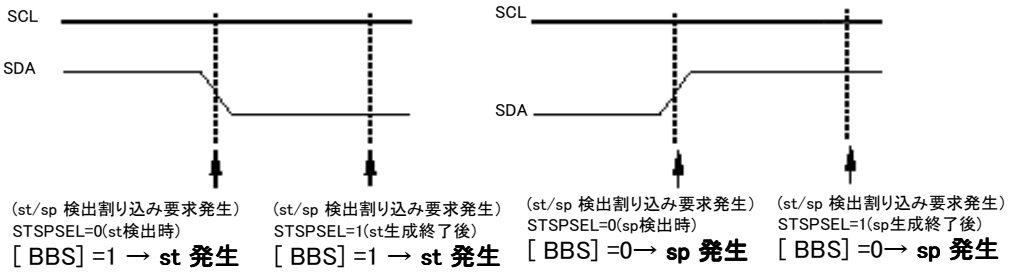
UART0/3 割り込み要因選択ビット
0:UART3のバス衝突/スタートストップ検出/障害エラー検出要因選択
1:UART0のバス衝突/スタートストップ検出/障害エラー検出要因選択

UART1/4 割り込み要因選択ビット
0:UART4のバス衝突/スタートストップ検出/障害エラー検出要因選択
1:UART1のバス衝突/スタートストップ検出/障害エラー検出要因選択

割り込み要求により割り込みを発生させる場合は、I フラグ を'1'に設定してください。

スタートコンディション／ストップコンデション検出(つづき)
[タイミング図]

st :スタートコンディション、sp :ストップコンディション

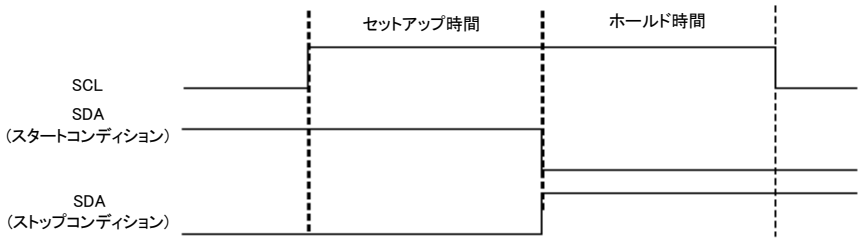


(st/sp 検出割り込み要求発生)
STSPSEL=0(st検出時)
[BBS] =1 → **st 発生**

(st/sp 検出割り込み要求発生)
STSPSEL=1(st生成終了後)
[BBS] =1 → **st 発生**

(st/sp 検出割り込み要求発生)
STSPSEL=0(sp検出時)
[BBS] =0→ **sp 発生**

(st/sp 検出割り込み要求発生)
STSPSEL=1(sp生成終了後)
[BBS] =0→ **sp 発生**



3～6サイクル<セットアップ／ホールド時間
サイクル数はメイクロック入力周波数f(Xin)のサイクル数を示します。

スタートコンディション/ストップコンディション/リスタートコンディション送出

M32C/83、85をマスタとして使用する場合のスタートコンディション、ストップコンディション、
およびリスタートコンディションの生成は、ハードウェアで行えます。
STAREQビットを"1"(スタート)にするとスタートコンディションを生成します。
STPREQビットを"1"(スタート)にすると(SCLが'L'の時は、SCLの解放を待って)ストップコンディションを
生成します。
RSTREQビットを"1"(スタート)にすると(SCLが'L'の時は、SCLの解放を待って)リスタートコンディションを
生成します。
STSPSELビットを"1"にすると、上記ビットで生成した各コンディションの出力を行います。
※各コンディション生成する際には、必ずREQビットを"1"(スタート)してから、STSPSELを"1"にしてください。
[関連レジスタ]

UARTi送受信モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	×	×	×	0	0	1	0

U0MR:0368$_{16}$番地 U1MR:02E8$_{16}$番地 U2MR:0338$_{16}$番地
U3MR:0328$_{16}$番地 U4MR:02F8$_{16}$番地

[SMD]010:簡易I²Cモード
[CKDIR] 0:内部クロックを選択
[IOPOL] 0:極性反転なし

UARTi送受信制御レジスタ0 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
1	0	1	1	●	×		

U0C0:036C$_{16}$番地 U1C0:02EC$_{16}$番地 U2C0:033C$_{16}$番地
U3C0:032C$_{16}$番地 U4C0:02FC$_{16}$番地

[CLK1] [CLK0] BRG カウントソースを選択
0 0 :f1 を選択、0 1 :f8 を選択、1 0 :f32 を選択、1 1 :使用禁止
[CRD] 1 :CTS/RTS 機能禁止
[NCH] 1:SCL、SDA端子はNchオープンドレイン出力(注1)
[CKPOL] 0 :CLK 極性はSCL たち下がりで送信データ出力、立ち上がりで入力
[UFORM] 転送フォーマット 1 :MSB ファースト

注1 UART2のSCL、SDA端子はNchオープンドレイン端子です。"0"を設定してください。

UARTi転送速度レジスタ (i=0～4)

b7							b0

U0BRG:0369$_{16}$番地 U1BRG:02E9$_{16}$番地 U2BRG:0339$_{16}$番地
U3BRG:0329$_{16}$番地 U4BRG:02F9$_{16}$番地

カウントソースをn+1分周

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新

スタートコンディション/ストップコンディション/リスタートコンディション送出（つづき）

UARTi特殊モードレジスタ2

(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0		0				

U0SMR2：0366$_{16}$番地　U1SMR2：02E6$_{16}$番地　U2SMR2：0336$_{16}$番地
U3SMR2：0326$_{16}$番地　U4SMR2：02F6$_{16}$番地

[IICM2] （1.4項「簡易I2Cバスモード時のレジスタ設定」UARTi特殊モードレジスタ2
表1簡易I²Cバスモード((IICM1=1)時の機能参照）
[CSC] 0：クロック同期化禁止　1：クロック同期化許可
[SWC] 0：SCL ウエイト出力禁止　1：SCL クロック出力許可
[ALS] 0：SDA 出力停止禁止　1：SDA 出力停止許可
[STAC] 0：UARTi初期化禁止
[SWC2] 0：UARTiクロック　1：″L″出力
[SDHI] 0：SDA出力許可
I²Cモード選択時は、“0”に設定してください

UARTi特殊モードレジスタ3

(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0		0

U0SMR3：0365$_{16}$番地　U1SMR3：02E5$_{16}$番地　U2SMR3：0335$_{16}$番地
U3SMR3：0325$_{16}$番地　U4SMR3：02F5$_{16}$番地

[CKPH] 0：クロック遅延なし　1：クロック遅延あり
[DL2][DL1][DL0]デジタル遅延値を設定
000：遅延なし　001：BRG カウントソースの2サイクル
010：BRG カウントソースの3サイクル　011：BRGカウントソースの4サイクル
100：BRG カウントソースの5サイクル　101：BRG カウントソースの6サイクル
110：BRG カウントソースの7サイクル　111：BRG カウントソースの8サイクル

UARTi特殊モードレジスタ4

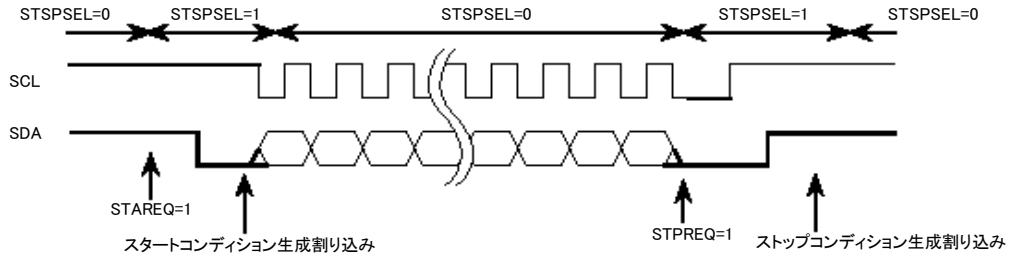
(i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
		0		1			

U0SMR4：0364$_{16}$番地　U1SMR4：02E4$_{16}$番地　U2SMR4：0334$_{16}$番地
U3SMR4：0324$_{16}$番地　U4SMR4：02F4$_{16}$番地

[STAREQ] 0：クリア　1：スタート
[RSTAREQ] 0：クリア　1：スタート
[STPREQ] 0：クリア　1：スタート
[STSPSEL] 1：スタート/ストップコンディション生成ブロック選択
[ACKD] 0：ACK　1：NACK
[ACKC] 0：SI/O データ出力
[SCLHI] 0：禁止　1：許可
[SWC9] 0：SCL″L″ホールド禁止　1：SCL″L″ホールド許可

スタートコンディション/ストップコンディション/リスタートコンディション送出（つづき）
［タイミング図］



バスビジー検出

スタートコンディションを送出する前には、バスが解放されていることを確認する必要があります。M32C/83、85の簡易I2Cバスモードでは、バスの状態はバスビジーフラグ[BBS] で検出できます。

スタートコンディションを検出したとき[BBS] は"1"にセットされ、ストップコンディションを検出したとき[BBS] は"0"にクリアされます。従ってスタートコンディションを送出しようとするときに[BBS] ="1"であったときバスは使用状態ですので、[BBS] ="0"にクリアされるまで待ってから送出を開始します。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367_{16}番地 U1SMR:$02E7_{16}$番地 U2SMR:0337_{16}番地
U3SMR:0327_{16}番地 U4SMR:02F716番地

[IICM]1:簡易I2Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新
[BBS] 0:バスは解放状態 1:バスは使用状態 ("0"のみ書き込み可)

SCL 端子L 出力機能2

下記のビットずれ現象は、クロック遅延機能（2.6項 その他機能参照）の使用により回避することができますが、新機能を使用していない従来方式の簡易I²Cバスファームウェアの流用時を前提に説明します。
M32C/83、85のシリアルI/Oは、送信データを送信バッファに書き込んでから転送クロック（簡易I²CバスモードではSCL）送出までに、最大で転送クロック(SCL)の1.5 サイクルの期間を要します。
また、M32C/83、85のSCL 同期化機能（2.5項 通信調整 参照）は1 ビット目のSCL送出時から有効になりますので、スタートコンディション発生後クロックライン(SCL)同期化機能が有効になるまでの期間内に他デバイスが1 ビット目の送出を行った場合、ビットずれを起こす可能性があります（タイミング図上図）。このため、M32C/83、85ではスタートコンディション送出後に他デバイスからのクロック送出を禁止するためのSCL 端子L出力機能を持っています。この機能を使用することにより、送信バッファにデータを書き込むと同時にSCL 端子から"L"を出力し、他デバイスを待ち状態することができます（タイミング図下図）。本機能はウエイト出力ビット2 [SWC2] を "1"にすることで動作が許可され（SCLから'L'出力）、'0'にすることで解除されます。

[関連レジスタ]

UARTi特殊モードレジスタ2 (i=0～4)

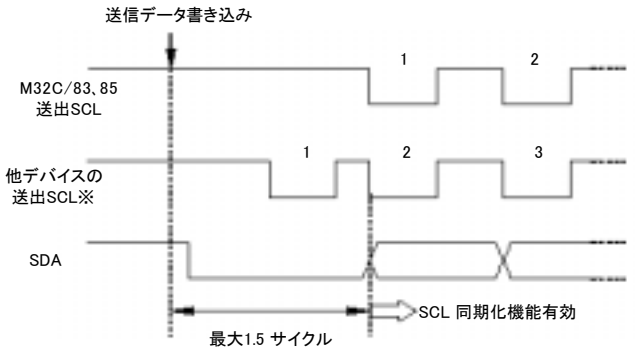
b7	b6	b5	b4	b3	b2	b1	b0
0		1					

U0SMR2：0366_{16}番地　U1SMR2：$02E6_{16}$番地　U2SMR2：0336_{16}番地
U3SMR2：0326_{16}番地　U4SMR2：$02F6_{16}$番地

- [IICM2] （1.4項「簡易I2Cバスモード時のレジスタ設定」 UARTi特殊モードレジスタ2 表1簡易I²Cバスモード((IICM1=1)時の機能参照）
- [CSC] 0：クロック同期化禁止　1：クロック同期化許可
- [SWC] 0：SCL ウエイト出力禁止　1：SCL クロック出力許可
- [ALS] 0：SDA 出力停止禁止　1：SDA 出力停止許可
- [STAC] 0：UARTi初期化禁止　1：UARTi初期化許可
- [SWC2] 0：UARTiクロック　1："L"出力
- [SDHI] 0：SDA出力許可
- I²Cモード選択時は、"0"に設定してください

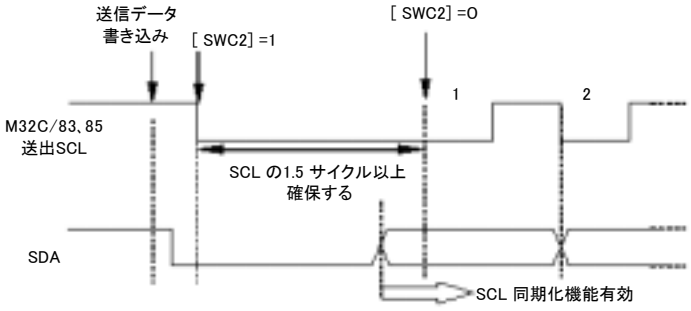
[タイミング図]

SCL 端子L 出力機能を使用しない場合



※ 通常、ビットずれが生じる。

SCL 端子L 出力機能を使用する場合



2.3 アクノリッジ

データの送受信には、1 バイト毎に確認応答（アクノリッジ）が付加されます。
M32C/83、85を送信装置として使用する場合は、1バイト毎に受信装置からのアクノリッジ
の有無を検出する必要があります。そのための機能として、「アクノリッジ検出割込」および
「アクノリッジ未検出割込」をハードウエアで備えています。またM32C/83、85を受信装置として使用し、
一対一通信を行う場合は、送信データの9 ビット目に'0'を設定することにより
容易にアクノリッジの生成ができます。(2.1 章「バイトデータの受信方法」参照)
「アクノリッジ検出割込」および「アクノリッジ未検出割込」を使用する場合には、
I2Cモード選択ビット2(IICM2)を"0"に設定することが必要です。
この設定により、割り込み番号17,19,33,35,37の要因、および割り込み番号18,20,34,36,38の要因が、
それぞれ「アクノリッジ未検出割込」、「アクノリッジ検出割込」となります。
その場合、UARTi受信レジスタから受信バッファレジスタへのデータ転送タイミングは、
受信クロックの最終ビットの立ち上がりとなります。
(1.4章「簡易I2Cバスモード時のレジスタ設定」 UARTi特殊モードレジスタ2表1
簡易I2Cバスモード((IICM1=1)時の機能参照)

アクノリッジ検出
送信クロックの9 ビット目の立ち上がり時に、送信装置側では解放しているSDA のラインのレベルが "L"になっている場合、受信装置側がアクノリッジを生成したと判断できます。
M32C/83、85の場合、これは「アクノリッジ検出割込」機能で検出できます。本割込は「ソフトウエア割込番号18,20,34,36,38」に割り当てられ、I²Cモード選択時([IICM] ="1")で、かつI²Cモード選択ビット2[IICM2] ="0"の時のみ、割り込み番号18,20,34,36,38の割込要因が「アクノリッジ検出割込」となります。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0							0

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] 0:アクノリッジ検出／アクノリッジ未検出割込有効
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:SCL はUARTiクロックを出力 1:SCL から"L"を出力(SCL 端子L 出力機能有効)
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
I²Cモード選択時は、"0"に設定してください

UARTi受信割り込み制御レジスタ (i=0～4) －アクノリッジ検出割込使用時－

b7	b6	b5	b4	b3	b2	b1	b0
☒	☒	☒	☒	0			

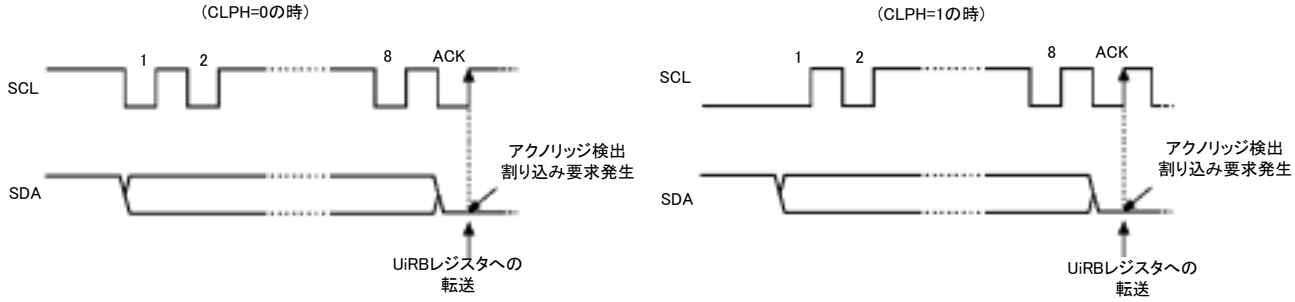
S0RIC:007216番地 S1RIC:007416番地 S2RIC:006B16番地
S3RIC:006D16番地 S4RIC:006F16番地

[ILVL] 割り込み優先レベル
1～7:割り込みレベル設定
(割り込みを使用しないときは"0"(禁止)にしてください)

[IR] 0:割り込み要求発生時"1"にセットされます

割り込み要求により割り込みを発生させる場合は、I フラグ を'1'に設定してください。

[タイミング図]



アクノリッジ未検出
送信クロックの9 ビット目の立ち上がり時に、送信装置側で解放しているSDA のラインのレベルが、
"H"になっている場合、受信装置側がアクノリッジを生成していないと判断します。
M32C/83、85の場合、これは「アクノリッジ未検出割込」機能で検出できます。本割込は
「ソフトウエア割込番号17,19,33,35,37」に割り当てられ、I²Cモード選択時([IICM] ="1")で、
かつI²Cモード選択ビット[IICM2] ="0"の時のみ割り込み番号17,19,33,35,37の割込要因が、
「アクノリッジ未検出割込」となります。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0							0

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] 0:アクノリッジ検出／アクノリッジ未検出割込有効
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:SCL はUARTiクロックを出力 1:SCL から"L"を出力(SCL 端子L 出力機能有効)
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
I²Cモード選択時は、"0"に設定してください

UARTi送信割り込み制御レジスタ (i=0～4) ーアクノリッジ未検出割込使用時ー

b7	b6	b5	b4	b3	b2	b1	b0
☒	☒	☒	☒	0			

S0TIC:0090$_{16}$番地 S1TIC:0092$_{16}$番地 S2TIC:0089$_{16}$番地
S3TIC:008B$_{16}$番地 S4TIC:008D$_{16}$番地

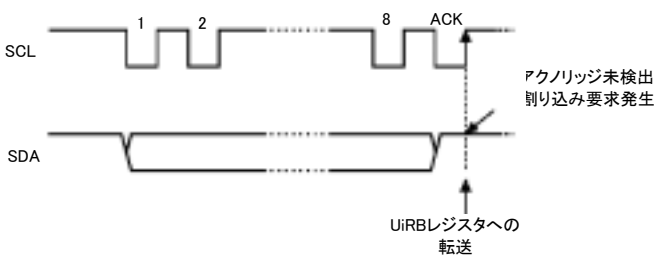
[ILVL] 割り込み優先レベル
1～7:割り込みレベル設定
(割り込みを使用しないときは"0"(禁止)にしてください)

[IR] 0:割り込み要求発生時"1"にセットされます

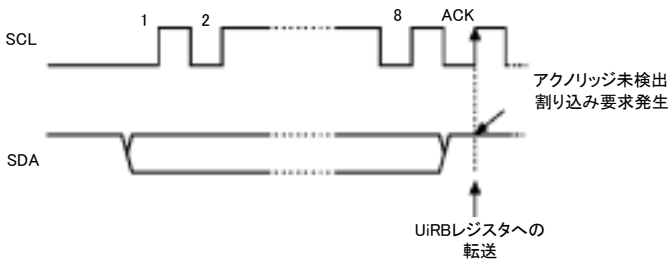
割り込み要求により割り込みを発生させる場合は、I フラグ を'1'に設定してください。

[タイミング図]

(CLPH=0の時)



(CLPH=1の時)



2.4 自己アドレス指定の判定

M32C/83、85をスレーブとして使用している場合、マスタから送信されたアドレスが自己のアドレスと一致するか比較し、一致したときスレーブ(M32C/83、85)はマスタに対しアクノリッジを送出します。M32C/83、85の簡易I2Cバスモードでは、自己アドレスとの比較およびアクノリッジの送出をソフトウエアで行いますが、その間SCL端子をL ホールドしマスタを待ち状態にする「SCL 端子L 出力機能」および、「ACK/NACK送出機能」をハードウエアで備えています。

ACK/NACK送出機能

M32C/83、85では、ACK、NACKの送出を送信データの9ビット目に設定するほかに
ACKCを"1"に設定して、ACKDビットを制御することでも行えます。
M32C/83、85をスレーブとして使用している場合、マスタから送信されたアドレスが自己のアドレスと一致するか比較し、一致したときスレーブ(M32C/83、85)はマスタに対しアクノリッジを送出します。
M32C/83、85の簡易I²Cバスモードでは、自己アドレスとの比較およびアクノリッジの送出をソフトウエアで行いますが、その際に、ACKCを"1"に設定して、ACK/NACK送出を行います。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0							1

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] 1:UARTi送信／受信割込有効
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:SCL はUARTiクロックを出力 1:SCL から"L"を出力(SCL 端子L 出力機能有効)
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
I²Cモード選択時は、"0"に設定してください

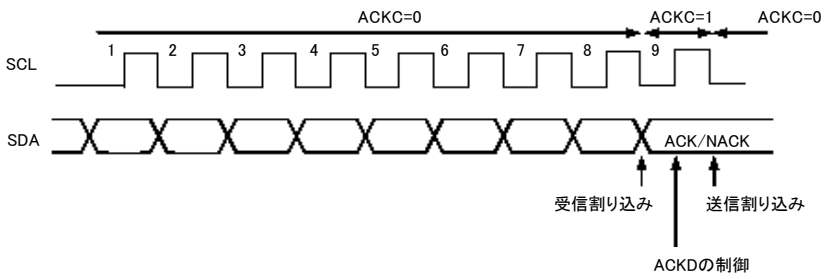
UARTi特殊モードレジスタ4 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
		1		0			

U0SMR4:0364$_{16}$番地 U1SMR4:02E4$_{16}$番地 U2SMR4:0334$_{16}$番地
U3SMR4:0324$_{16}$番地 U4SMR4:02F4$_{16}$番地

[STAREQ] 0:クリア 1:スタート
[RSTAREQ] 0:クリア 1:スタート
[STPREQ] 0:クリア 1:スタート
[STSPSEL]0:シリアルI/Oブロック選択
[ACKD] 0:ACK 1:NACK
[ACKC] 1:ACKデータビット(ACKD)出力
[SCLHI] 0:禁止 1:許可
[SWC9] 0:SCL"L"ホールド禁止 1:SCL"L"ホールド許可

[タイミング図]



SCL 端子L 出力機能

I²Cバスでは、スタートコンディション検出後の第一バイトに、指定されるスレーブのアドレスが送出されます（7 ビットアドレスモード時）。スレーブは、第一バイトに他のマスタから送出されるクロックのはじめの7 ビットの受信データを自己アドレスと比較し、クロックの9 ビット目に同期してアクノリッジを生成（または未生成）する処理が必要です。
本処理のための機能としてM32C/83、85では「SCL 端子L 出力機能」を持っています。この機能を用いると、初めの8 ビットのデータを受信後、9 ビット目のSCL が"L"になるタイミングでM32C/83、85のSCL 端子に "L"を出力して、強制的にマスタを待ち状態にします。そしてソフトウエアでのアドレス比較処理が終了後に、ACK/NACK生成機能(本項説明参照)でアクノリッジの生成／未生成を行うことができます。（一対一通信で使用している場合等は「2.1 バイトデータの受信方法」で説明した方法で、アドレスの受信、アクノリッジの生成が行えます）
本機能は、ウエイト出力ビット[SWC] を"1"にすることにより動作が許可され、"0"にすることにより禁止されます。また、本機能でSCL 端子が"L"レベルになったときは、[SWC] を"0"にすることで解除できます。なお、本機能によりアドレス比較処理を行う場合、最終ビットのクロックの立ち上がり前に受信バッファレジスタの内容を読み出すことになりますので、読み出した受信データはビットの位置が変化していることにご注意ください。（1.1項「送信割込／受信割込」のタイミング図参照）

[関連レジスタ]

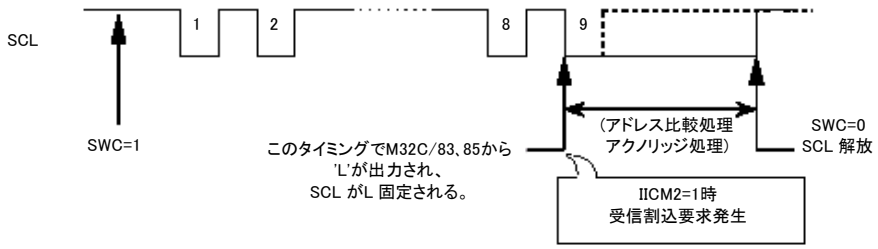
UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0					1		1

U0SMR2：0366$_{16}$番地　U1SMR2：02E6$_{16}$番地　U2SMR2：0336$_{16}$番地
U3SMR2：0326$_{16}$番地　U4SMR2：02F6$_{16}$番地

[IICM2] 1：UARTi送信／受信割込有効
[CSC] 0：クロック同期化禁止　1：クロック同期化許可
[SWC] 0：SCL ウエイト出力禁止　1：SCL クロック出力許可
[ALS] 0：SDA 出力停止禁止　1：SDA 出力停止許可
[STAC] 0：UARTi初期化禁止　1：UARTi初期化許可
[SWC2] 0：SCL はUARTiクロックを出力　1：SCL から"L"を出力（SCL 端子L 出力機能有効）
[SDHI] 0：SDA出力許可　1：SDA 出力禁止（ハイインピーダンス）
I²Cモード選択時は、"0"に設定してください

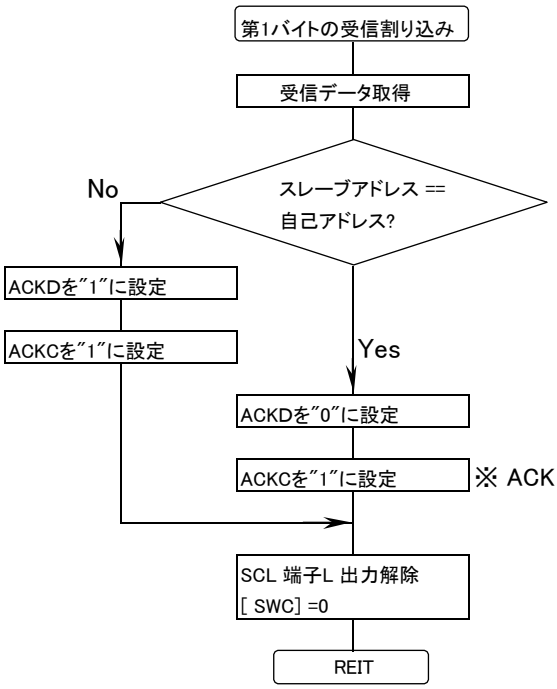
[タイミング図]



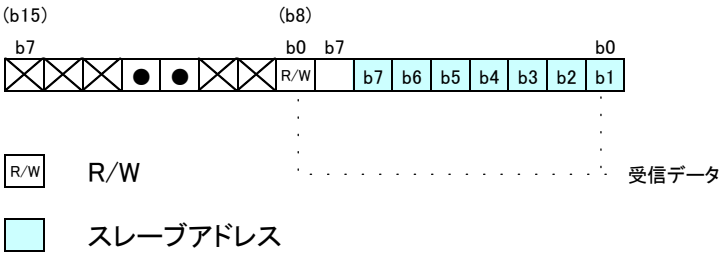
自己アドレス判定の一例
アドレス指定のフォーマットは、7ビットアドレスと10ビットアドレスの2通りあります。
ここでは、7ビットアドレスの判定の一例をご紹介しますが、10ビットアドレスのフォーマットでも同様にして制御できます。ここでは、スタートビット受信後、第一バイトの受信割り込みが発生したときの制御の一例をご紹介します。ここでは、第一バイト受信前に[SWC] =″1″(SCL 端子L 出力機能許可)をすでに設定しているものとし、受信割り込みルーチンの一部のみ記述します。

[フローチャート]

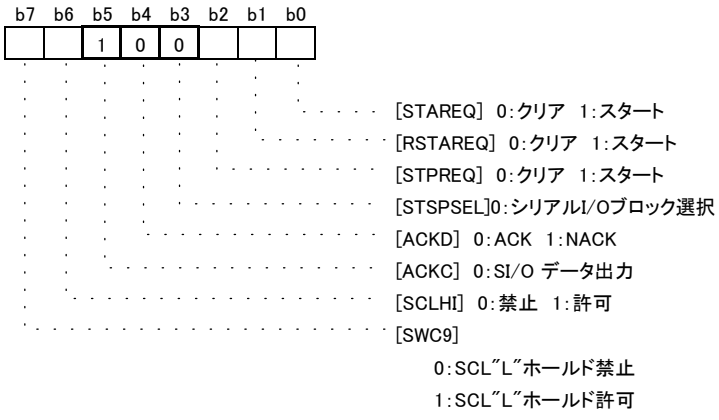
※メインルーチンでの設定(例)
・swc=1 (SCL 端子L 出力機能許可)
・IICM2=1(割込番号18,20,34,36,38の割込要因:受信割込、割込発生タイミングは最終ビットの立ち上がり)
・S2RIC=7 (受信割込許可)
・I フラグ=1 (割込許可)



UARTi受信バッファレジスタ (i=0～4)
U0RB:036F$_{16}$、036E$_{16}$番地 U1RB:02EF$_{16}$、02EE$_{16}$番地
U2RB:033F$_{16}$、033E$_{16}$番地 U3RB:032F$_{16}$、032E$_{16}$番地
U4RB:02FF$_{16}$、02FE$_{16}$ 番地



UARTi特殊モードレジスタ4 (i=0～4)
U0SMR4:0364$_{16}$番地 U1SMR4:02E4$_{16}$番地 U2SMR4:0334$_{16}$番地
U3SMR4:0324$_{16}$番地 U4SMR4:02F4$_{16}$番地



2.5 通信調整

I2Cバスをマルチマスタで使用する場合は、複数のマスタが同時にスタートコンディションを生成してデータ送信をスタートしようとすることが考えられます（アービトレーション発生）。この場合、I2Cバスのシステムでは複数のマスタ間で通信調整が行われます。M32C/ 83、85の簡易I2Cバスモードでは、アービトレーション・ロスト発生時に通信を回復するための機能として、「アービトレーション・ロスト検出機能」、「アービトレーション・ロスト発生時のSDA 出力禁止機能」をハードウエアで備えています。またアービトレーション・ロスト以外にも、クロック同期を利用した通信調整のひとつとして「SCL 同期化機能」も備えています。

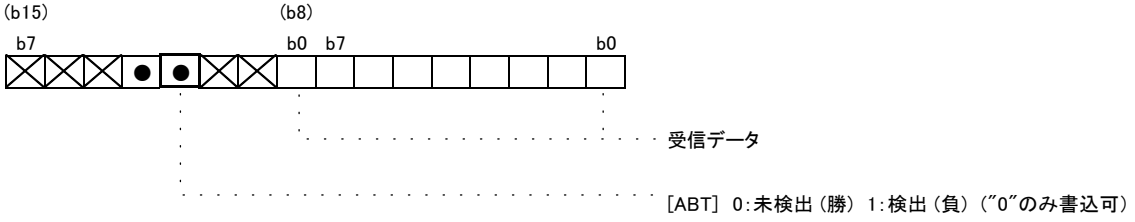
アービトレーション・ロスト検出機能　　(i=0～4)

M32C/83、85の簡易I2Cバスモードは、アービトレーション・ロスト検出フラグ[ABT] を持っています。
この検出フラグはUARTi受信バッファレジスタのビット3に配置されており、
SCL の立ち上がりのタイミングで内部データ・レベルとSDA のレベルの不一致を検出すると
アービトレーションロスト検出フラグ[ABT] は"1"にセットされます。アービトレーションロスト検出フラグの
更新は、アービトレーションロスト検出フラグ制御[ABC] により、ビット毎("0")／バイト毎("1")の
選択が行えます。I2Cモード選択ビット[IICM] 、およびI2Cモード選択ビット2[IICM2] が
共に"1"のときは、[ABC] =0 に固定してください。
アクノリッジ受信のタイミングでは出力"H"、入力"L"となりアービトレーションロスト検出フラグが
セットされてしまいますので、次の送信時にはアービトレーションロスト検出フラグを
"0"にクリアしてから送信を行ってください。

[関連レジスタ]

UARTi受信バッファレジスタ　(i=0～4)

U0RB:036F16 、036E16番地　U1RB:02EF16 、02EE16番地
U2RB:033F16 、033E16番地　U3RB:032F16、032E16番地
U4RB:02FF16 、02FE16 番地



UARTi特殊モードレジスタ　(i=0～4)

U0SMR:0367₁₆番地　U1SMR:02E7₁₆番地　U2SMR:0337₁₆番地
U3SMR:0327₁₆番地　U4SMR:02F716番地

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

[IICM]1:簡易I2Cモード
[ABC] アービトレーションロストを　0:ビット毎に更新　1:バイト毎に更新

アービトレーション・ロスト発生時のSDA出力禁止機能
アービトレーション・ロストの発生を検知したマスタは、その時点でSDA の出力をオフしなくては
なりません。M32C/83、85の簡易I2Cバスモードは、アービトレーション・ロスト発生時に
ハードウエアで自動的にSDA 出力をオフする機能を選択できます。本機能は、
SDA出力停止ビット[ALS] に″1″を設定することで許可され、″0″を設定することで禁止されます。
本機能によりSDA 出力がオフされた場合は、SDA 出力停止ビット[ALS]
またはアービトレーションロスト検出フラグ[ABT]をクリアすることにより解除されます。
ただし本機能を許可している場合はアクノリッジのタイミングでもアービトレーション・ロスト発生と判断して
出力をオフしてしまいますので、次のバイトデータの送出はアービトレーション検出ビット
[ABT] をクリアしてから行ってください。また、アービトレーション検出フラグ制御[ABC]は
″0″に固定してください。

[関連レジスタ]

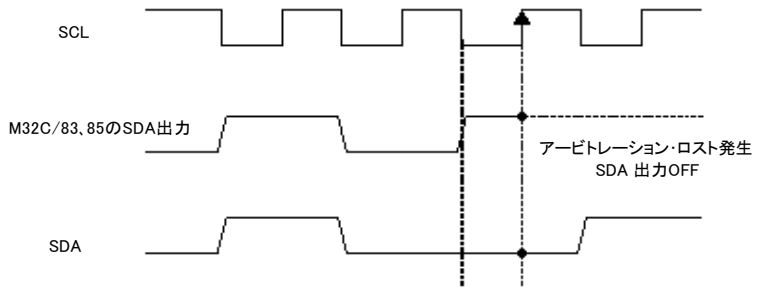
UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0				1			

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] (1.4項「簡易I2Cバスモード時のレジスタ設定」UARTi特殊モードレジスタ2
表1簡易I2Cバスモード((IICM1=1)時の機能参照)
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:SCL はUARTiクロックを出力 1:SCL から″L″を出力(SCL 端子L 出力機能有効)
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
簡易I2Cモード選択時は、“0”に設定してください

[タイミング図]



SCL 同期化機能

M32C/83、85を処理速度の遅いデバイスと接続している場合、他のデバイスがSCLに対しLホールドを行い、マスタから送出するクロックを強制的に待ち状態にすることがあります。
M32C/83、85の簡易I²Cバスモードでは、他デバイスからのL ホールドに対し自動的に待ち状態に入り、Lホールドの解除により待ち状態も解除するSCL 同期化機能を持っています。本機能は、クロック同期化ビット[CSC] を"1"に設定することにより動作が許可され、"0"を設定することにより禁止されます。本機能はM32C/83、85をマスタとして使用する場合（内部クロックモードで）のみご使用ください。

[関連レジスタ]

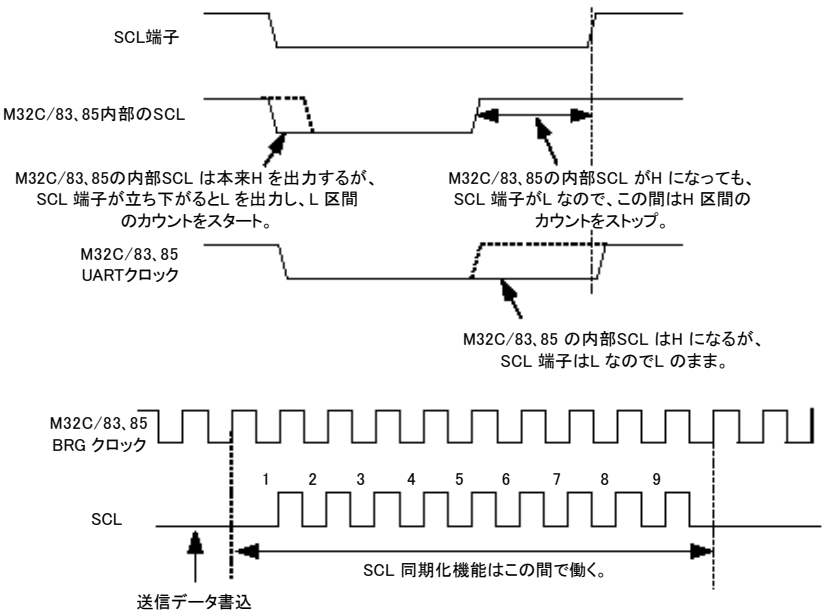
UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0						1	

U0SMR2：0366$_{16}$番地　U1SMR2：02E6$_{16}$番地　U2SMR2：0336$_{16}$番地
U3SMR2：0326$_{16}$番地　U4SMR2：02F6$_{16}$番地

[IICM2] （1.4項「簡易I²Cバスモード時のレジスタ設定」 UARTi特殊モードレジスタ2 表1簡易I²Cバスモード((IICM1=1)時の機能参照）
[CSC] 0：クロック同期化禁止　1：クロック同期化許可
[SWC] 0：SCL ウエイト出力禁止　1：SCL クロック出力許可
[ALS] 0：SDA 出力停止禁止　1：SDA 出力停止許可
[STAC] 0：UARTi初期化禁止　1：UARTi初期化許可
[SWC2] 0：SCL はUARTiクロックを出力　1：SCL から"L"を出力（SCL 端子L 出力機能有効）
[SDHI] 0：SDA出力許可　1：SDA 出力禁止（ハイインピーダンス）
I²Cモード選択時は、"0"に設定してください

[タイミング図]



2.6 その他の機能

M32C/83、85の簡易I2Cバスモードは、2.1～2.5項で説明した他に、I2Cバス制御を容易にする機能として以下の機能もハードウエアで備えています。

SDA 出力禁止機能

M32C/83、85をスレーブとして使用し、スタートコンディション受信後の1 バイト目でアドレス判定を行う際、マスタが指定したアドレスが自アドレスと異なればM32C/83、85はSDAの出力をオフ(ハイインピーダンス)にしなければなりません。その場合、SCL の9 クロック毎(受信割り込み要求が発生する毎)にM32C/83、85の送信バッファレジスタに ″1FFh″を設定することによりSDA 出力をオフにします。また、M32C/83、85が備えているSDA 出力禁止機能を使用することでもSDA 出力をオフすることができます。本機能は、SDA 出力禁止ビット[SDHI] を ″1″にすることで有効となり、送信バッファレジスタに ″1FFh″を設定しなくてもM32C/83、85のSDA 出力をハイインピーダンスにできます。[SDHI] を ″0″にすることで本機能は解除され、次のSCL の入力に同期して送信バッファに設定した値が出力されます。

[関連レジスタ]

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	1						

U0SMR2:0366_{16}番地 U1SMR2:$02E6_{16}$番地 U2SMR2:0336_{16}番地
U3SMR2:0326_{16}番地 U4SMR2:$02F6_{16}$番地

- [IICM2] (1.4項「簡易I2Cバスモード時のレジスタ設定」UARTi特殊モードレジスタ2 表1簡易I2Cバスモード((IICM1=1)時の機能参照)
- [CSC] 0:クロック同期化禁止 1:クロック同期化許可
- [SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
- [ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
- [STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
- [SWC2] 0:SCL はUARTiクロックを出力 1:SCL から″L″を出力(SCL 端子L 出力機能有効)
- [SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
- I2Cモード選択時は、“0”に設定してください

UARTi初期化機能(i=0～4)
M32C/83、85の簡易I²Cバスモードは、スタートコンディション検出のタイミングで自動的に
UARTiを初期化する機能を備えています。本機能はM32C/83、85をスレーブとする際に使用します。
スタートコンディション初期化ビット[STAC] を ″1″にすることにより機能が許可され、
″0″にすることにより禁止されます。スタートコンディションを検出すると、以下の初期化が行われます。
① 送信レジスタは初期化され、送信バッファレジスタの内容が送信レジスタに転送されます。
これにより、データ受信時に送信バッファレジスタにデータを再設定する必要はなく、
次に入力されたクロックを1 ビット目として送信が開始されます。
ただし、その時の送信データは最後に送信していたデータと同じものとなりますので、
SDA出力禁止ビット[SDHI] を″1″として送信データの出力を禁止してください。
② 受信レジスタは初期化され、次に入力されたクロックを1 ビット目として受信が開始されます。
受信バッファレジスタを読み出す前のタイミングで初期化され
受信が開始しても、オーバランエラーは発生しません。
③ウエイト出力ビット[SWC] が″1″にセットされます。これによりSCL端子L出力機能が
有効となり、転送クロックの9 ビット目の立ち下がりでSCL 端子からL が出力されます。
本機能は外部クロック選択でのみ使用してください。また本機能を使用しUARTiの
送受信を開始した場合、送信バッファ空フラグの値は変化しませんのでご注意ください。

[関連レジスタ]

UARTi特殊モードレジスタ2 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0			1				

U0SMR2:0366$_{16}$番地 U1SMR2:02E6$_{16}$番地 U2SMR2:0336$_{16}$番地
U3SMR2:0326$_{16}$番地 U4SMR2:02F6$_{16}$番地

[IICM2] (1.4項「簡易I²Cバスモード時のレジスタ設定」 UARTi特殊モードレジスタ2
表1簡易I²Cバスモード((IICM1=1)時の機能参照)
[CSC] 0:クロック同期化禁止 1:クロック同期化許可
[SWC] 0:SCL ウエイト出力禁止 1:SCL クロック出力許可
[ALS] 0:SDA 出力停止禁止 1:SDA 出力停止許可
[STAC] 0:UARTi初期化禁止 1:UARTi初期化許可
[SWC2] 0:SCL はUARTiクロックを出力 1:SCL から″L″を出力(SCL 端子L 出力機能有効)
[SDHI] 0:SDA出力許可 1:SDA 出力禁止(ハイインピーダンス)
I²Cモード選択時は、“0”に設定してください

SCL出力停止機能

I2Cバスをマルチマスタで使用する場合に、M32C/83、85がマスタとして送受信を行っている際、
他のマスタが、ストップコンディションを生成することが考えられます。
この時、M32C/83、85は、SCLとSDAを解放し通信を終了しなければなりません。
SDAはアービトレーション・ロスト発生時のSDA出力禁止機能
(2.5項 通信調整参照)を使用することにより、解放することが可能です。
SCLに関しては本機能を使用することにより、解放することが可能となります。
本機能は、[SCLHI]を"1"に設定することにより、有効となります。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367$_{16}$番地 U1SMR:02E7$_{16}$番地 U2SMR:0337$_{16}$番地
U3SMR:0327$_{16}$番地 U4SMR:02F716番地

[IICM]1:簡易I2Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新
[BBS] 0:バスは解放状態 1:バスは使用状態 ("0"のみ書き込み可)

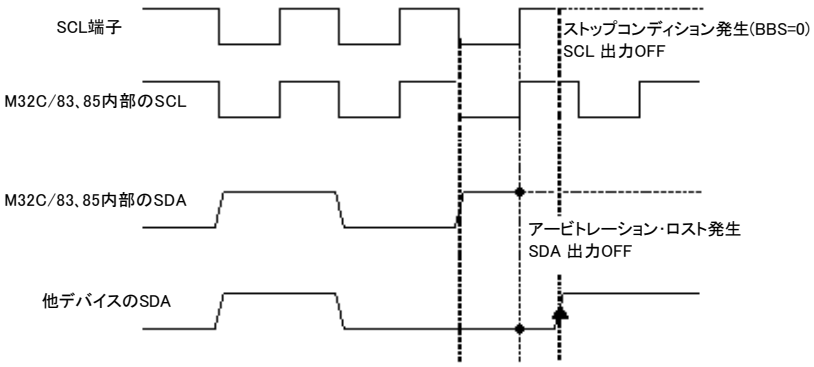
UARTi特殊モードレジスタ4 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
	1						

U0SMR4:0364$_{16}$番地 U1SMR4:02E4$_{16}$番地 U2SMR4:0334$_{16}$番地
U3SMR4:0324$_{16}$番地 U4SMR4:02F4$_{16}$番地

[STAREQ] 0:クリア 1:スタート
[RSTAREQ] 0:クリア 1:スタート
[STPREQ] 0:クリア 1:スタート
[STSPSEL]0:シリアルI/Oブロック選択 1:スタート/ストップコンディション生成ブロック選択
[ACKD] 0:ACK 1:NACK
[ACKC] 0:SI/O データ出力 1:ACKデータビット(ACKD)出力
[SCLHI] 0:禁止 1:許可
[SWC9] 0:SCL"L"ホールド禁止 1:SCL"L"ホールド許可

[タイミング図]



SCL 端子L 出力機能2

I2Cバスでスレーブ送信を行う際、マスタは9 ビット目に同期してアクノリッジの生成（または未生成）を行います。この時、スレーブはアクノリッジ判定を行い。アクノリッジ検出時は、送信の継続(送信次データの設定)を行い。未検出時は送信を終了します。

本処理のための機能としてM32C/83、85では「SCL 端子L 出力機能2」を持っています。この機能を用いると、初めの9 ビット(ACK/NACK)のデータを受信後、最終ビットのSCL が″L″になるタイミングでM32C/83、85のSCL 端子に ″L″を出力して、強制的にマスタを待ち状態にします。

そして、ソフトウエアでのアクノリッジ判定処理が終了後に、送信継続処理、もしくは送信終了処理を行うことができます。

本機能は、ウエイト出力ビット2[SWC9] を″1″にすることにより動作が許可され、″0″にすることにより禁止されます。また、本機能でSCL 端子が″0″になったときは、[SWC9] を″0″にすることで解除できます。

[関連レジスタ]

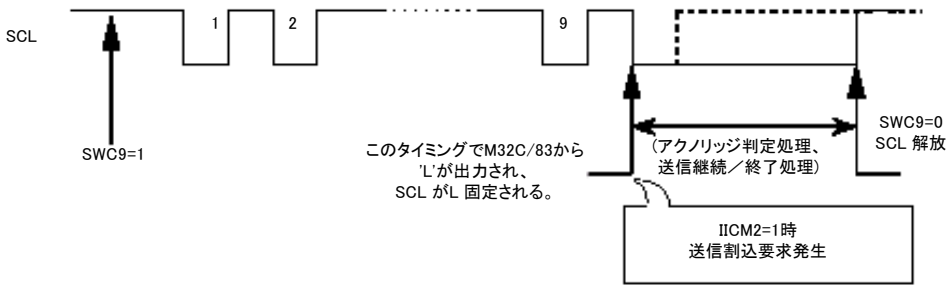
UARTi特殊モードレジスタ4 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
1							

U0SMR4：0364_{16}番地　U1SMR4：$02E4_{16}$番地　U2SMR4：0334_{16}番地
U3SMR4：0324_{16}番地　U4SMR4：$02F4_{16}$番地

[STAREQ] 0：クリア　1：スタート
[RSTAREQ] 0：クリア　1：スタート
[STPREQ] 0：クリア　1：スタート
[STSPSEL]0：シリアルI/Oブロック選択
[ACKD] 0：ACK　1：NACK
[ACKC] 0：SI/O データ出力
[SCLHI] 0：禁止　1：許可
[SWC9] 0：SCL″L″ホールド禁止　1：SCL″L″ホールド許可

[タイミング図]



クロック遅延機能

I²CバスでM32C/83、85を使用する場合、クロック遅延機能を利用してUARTi(i=0～4)を使用します。
この機能を利用する事により、SCL初期値"L"、終了値"L"の波形出力が行えるため
各コンディションとのつなぎがスムーズに行えます。
また、SCLの8bit目の立ち下がりと9bitの立ち上がりにUiRBレジスタへの
書き込みタイミングが発生する様になり、ACK/NACKビットの受信が可能となります。
さらに、IICM2="1"の場合、ACK/NACKビットの受信後に送信割り込みが発生します。
本機能は、[IICM]を1に設定し、[CKPH]を"1"に設定することにより有効となります。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367_{16}番地 U1SMR:$02E7_{16}$番地 U2SMR:0337_{16}番地
U3SMR:0327_{16}番地 U4SMR:02F716番地

[IICM]1:簡易I²Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新
[BBS] 0:バスは解放状態 1:バスは使用状態 ("0"のみ書き込み可)

UARTi特殊モードレジスタ3 (i=0～4)

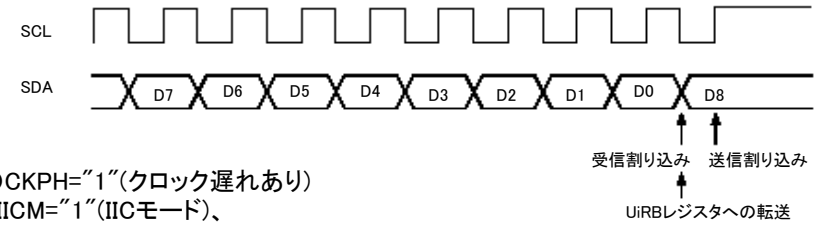
b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0	1	0

U0SMR3:0365_{16}番地 U1SMR3:$02E5_{16}$番地 U2SMR3:0335_{16}番地
U3SMR3:0325_{16}番地 U4SMR3:$02F5_{16}$番地

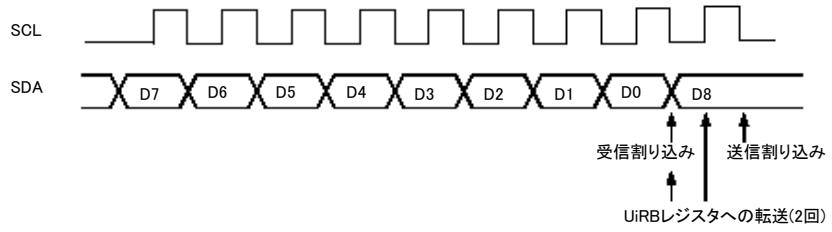
[CKPH] 0:クロック遅延なし 1:クロック遅延あり
[DL2][DL1][DL0]デジタル遅延値を設定
000:遅延なし 001:BRG カウントソースの2サイクル
010:BRG カウントソースの3サイクル 011:BRGカウントソースの4サイクル
100:BRG カウントソースの5サイクル 101:BRG カウントソースの6サイクル
110:BRG カウントソースの7サイクル 111:BRG カウントソースの8サイクル

[タイミング図]

●CKPH="0"(クロック遅れなし)
IICM="1"(IICモード)、
IICM=2(UART送受信割り込み)の場合



●CKPH="1"(クロック遅れあり)
IICM="1"(IICモード)、
IICM=2(UART送受信割り込み)の場合



UART出力デジタル遅延機能

I2Cバスで送信を行う場合、SCLが"L"区間の時に、データを切り替えなければなりません。
SCLが"H"区間の時、SDAが変化すると各コンディションが誤検出されてしまいます。
M32C/83、85では、UART出力デジタル遅延機能を利用することにより、クロック"L"期間でのデータ変化を確定できます。
本機能は、[IICM]を1に設定し、[DL0～2]を"1"～"7"に設定することで、有効となります。

[関連レジスタ]

UARTi特殊モードレジスタ (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	●		1

U0SMR:0367_{16}番地 U1SMR:$02E7_{16}$番地 U2SMR:0337_{16}番地
U3SMR:0327_{16}番地 U4SMR:02F716番地

[IICM]1:簡易I2Cモード
[ABC] アービトレーションロストを 0:ビット毎に更新 1:バイト毎に更新
[BBS] 0:バスは解放状態 1:バスは使用状態 ("0"のみ書き込み可)

UARTi特殊モードレジスタ3 (i=0～4)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0	1	0

U0SMR3:0365_{16}番地 U1SMR3:$02E5_{16}$番地 U2SMR3:0335_{16}番地
U3SMR3:0325_{16}番地 U4SMR3:$02F5_{16}$番地

[CKPH] 0:クロック遅延なし 1:クロック遅延あり
[DL2][DL1][DL0]デジタル遅延値を設定
000:遅延なし 001:BRG カウントソースの2サイクル
010:BRG カウントソースの3サイクル 011:BRGカウントソースの4サイクル
100:BRG カウントソースの5サイクル 101:BRG カウントソースの6サイクル
110:BRG カウントソースの7サイクル 111:BRG カウントソースの8サイクル

UARTi送受信制御レジスタ0 (i=0～4)

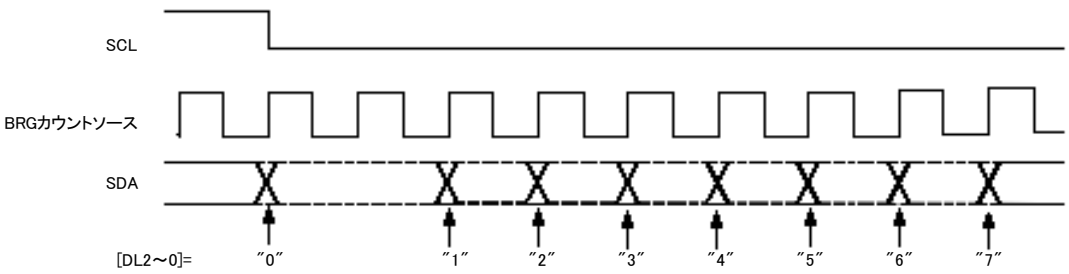
b7	b6	b5	b4	b3	b2	b1	b0
1	0	1	1	●	☒		

U0C0:$036C_{16}$番地 U1C0:$02EC_{16}$番地 U2C0:$033C_{16}$番地
U3C0:$032C_{16}$番地 U4C0:$02FC_{16}$番地

[CLK1] [CLK0] BRG カウントソースを選択
0 0 :f1 を選択、0 1 :f8 を選択、1 0 :f32 を選択、1 1 :使用禁止
[CRD] 1 :CTS/RTS 機能禁止
[NCH] 1:SCL、SDA端子はNchオープンドレイン出力(注1)
[CKPOL] 0 :CLK 極性はSCL たち下がりで送信データ出力、立ち上がりで入力
[UFORM] 転送フォーマット 1 :MSB ファースト

注1 UART2のSCL、SDA端子はNchオープンドレイン端子です。"0"を設定してください。

[タイミング図]



各[DL2～0]設定時のSDA出力切り替えタイミング

第三章
簡易I2Cバスモード使用時の注意事項

3.1 電気的特性

本章では、M32C/83、85の簡易I2Cバスモードを使用し、I2Cバスのプロトコル制御を行う場合の注意事項および制限事項を説明します。

3.1 電気的特性

M32C/83、85の電気的特性は、I²Cバス規格と一部異なる点があります。
以下にI²Cバスの規格を示します。

パラメータ	記号	標準モード		高速モード		単位
		Min.	Max.	Min.	Max.	
LOW レベル入力電圧 : 入力レベルが一定の場合 入力レベルがVDD に応じて変化する場合	VIL	 −0.5 −0.5	 1.5 0.3VDD	 −0.5 −0.5	 1.5 0.3VDD	V
HIGH レベル入力電圧: 入力レベルが一定の場合 入力レベルがVDD に応じて変化する場合	VIH	 3.0 0.7VDD	 *1) *1)	 3.0 0.7VDD	 *1) *1)	V
シュミット(Schmitt)トリガ入力のヒステリシス: 入力レベルが一定の場合 入力レベルがVDD に応じて変化する場合	Vhys	 n/a n/a	 n/a n/a	 0.2 0.05VDD	 — —	V
入力フィルタによって抑制されるスパイクの パルス幅	tSP	n/a	n/a	0	50	ns
LOW レベル出力電圧 (オープン・ドレイン またはオープン・コレクタ): シンク電流3mA 時 シンク電流6mA 時	 VOL1 VOL2	 0 n/a	 0.4 n/a	 0 0	 0.4 0.6	V
バスのキャパシタンスが10pF ～400pF(VOL2 の 並列抵抗を通して最大6mA まで)の場合の VIHmin.からVILmax.への出力立ち下がり時間: VOL1 での最大シンク電流3mA VOL2 での最大シンク電流6mA	tOF	 — n/a	 250 2) n/a	 20+0.1Cb 2)	 250 250 3)	ns
入力電圧0.4V ～0.9VDDmax.時の 各I/O ピンの入力電流	Ii	 −10	 10	 −10 3)	 10 3)	μa
各I/O ピンのキャパシタンス	Ci	—	10	—	10	pF

n/a =該当せず

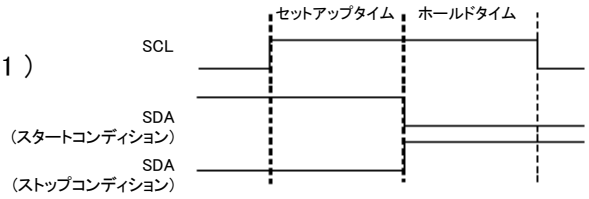
1)最大VIH =VDD max.+0.5V

2)Cb =1つのバス・ラインのキャパシタンス(単位pF)。SDA およびSCL バス・ラインの最大tF(300ns)は出力段での最大tOF(250ns)より大きくなります。
これによって、最大規定tF を超えることなくSDA/SCL ピンとSDA/SCL バス・ラインの間に直列保護抵抗(Rs)を接続することが可能となります。

3)VDD の供給が切れたときに、I/O ピンがSDA およびSCL ラインを妨害しないようにする必要があります。

スタート／ストップコンディションのセットアップタイム・ホールドタイム
M32C/83、85のスタートコンディション／ストップコンディション検出時のセットアップタイム
およびホールドタイムは、I2C バスの規格値と異なる場合があります。
（高速モード時）
M32C/83、85のスタートコンディション／ストップコンディション検出時のセットアップタイム
およびホールドタイムは、以下の値となります。

セットアップ時間 ＞ 3 ～6 サイクル（注1 ）
ホールド時間 ＞ 3 ～6 サイクル（注1）



（注1）サイクル数はメインクロック入力発振周波数 f((XIN)のサイクル数を示します。
高速モードI2Cバスの規格において、スタートコンディションおよびストップコンディションの
セットアップ／ホールドタイムはどちらもMin.600ns です。それに対し、M32C/83、85の
セットアップ／ホールドタイムは Min.6 サイクル(f(XIN)のサイクル数)となります。
従って、メインクロック（f(XIN))を10MHz で使用した場合は、M32C/83、85の簡易I2Cバスのセットアップ
／ホールドタイムは Min.600ns となり高速モードI2Cバス規格にも対応できますが、
メインクロックを10MHz 未満で使用する場合は、セットアップ／ホールドタイムは高速モード
I2Cバス規格を満たすことができなくなります。

LOW レベル／HIGH レベル入力電圧

M32C/83、85の電気的特性は、2.7V ～5.5V 動作時において
H入力電圧(VIH)=min.0.8Vcc（保証値）
L入力電圧(VIL)=max.0.2Vcc（保証値）
となります。従って、I²Cバスの規格値
5V 動作時：VIH =3V 、VIL =1.5V
5V 以外 ：VIH =0.7Vcc 、VIL =0.3Vcc
とは異なります。
またM32C/83、85の"L"出力電圧は、Vcc =5V 、IOL =5mA 時に
L出力電圧(VOL)=max.2.0V（保証値）
となります。I ²Cバスの規格値
L出力電圧(VOL)=max.0.6V（IOL =6mA 時）
とは異なります。
ただしM32C/83、85の標準特性においては、Vcc =5V 、IOL =5mA 時に
L出力電圧(VOL)=0.6V
程度となります。

3.2 BRG カウントソースによる最大転送速度の制限

M32C/83、85がSCL のレベルを認識するまでの時間はサンプリング周期に依存し、
最大BRGカウントソースの3 クロック分を要します。従って、動作周波数およびBRG カウントソース
設定ビットで設定したBRG カウントソースの速度によって、M32C/83、85の簡易I²Cバスに
接続可能なI²Cバスの最大転送速度が制限されます。以下の条件を満たす転送速度で使用しない場合、
ビットずれを起こす可能性があります。
I ²Cバスの最大転送速度 (Hz)＜ **BRG カウントソース (Hz)**／ 3
（例）源発振10MHz 、BRG カウントソースにfc32 を選択した場合
I²Cバスの最大転送速度(Hz)＜**10MHz/32** ／ 3 =104Kbps
この場合のI ²Cバスの最大転送速度は、104Kbps になります。

3.3 最大動作周波数

最大動作周波数は、M32C/83、85の最大動作周波数f(Xin)maxに依存します。

4.0 参考ドキュメント

データシート
M32C/83グループデータシート Rev.1.0
（最新版をルネサス テクノロジホームページから入手してください。）

5.0 ホームページとサポート窓口

ルネサス テクノロジホームページ
http://www.renesas.com/

ルネサス製品全般に関するお問合せ先
カスタマ・サポート・センター：csc@renesas.com

M16Cに関する技術的なお問い合わせ先
M16CファミリMCU技術サポート窓口：support_apl@renesas.com

・付録
ソフトI²Cバスコントローラ仕様書

・本プログラムは参考プログラムです。通信動作を保証するものではありません。
システムへの組み込みの際には、十分検討の上ご使用ください。
また、本ソフトウェア単体ではシステムとしての評価ができないため、
最終システムでの評価を実施してください。

［目次］

1. 概要

本ソフトウェアはM32C/83グループに内蔵された簡易I2CバスモードのH/Wを制御し、
I2Cバスの通信プロトコルを実現するものです。
下記使用条件によりI2Cバスの通信プロトコルに準拠します。

＜動作条件＞

発信周波数：20MHz（ノンウェイト、分周なし）

※20MHz以外で使用される場合は、本アプリケーションノートを参考にUiBRG及び、DL2～0(UiSMR3)の値を選定してください。

2.1 バイトデータの送信/受信の方法、2.6 その他の機能「クロック遅延機能」 3章 簡易I2Cバスモード使用時の注意事項 参照

＜使用上の制限事項＞

・7bitアドレス装置間の通信のみ対応しています。
・特別なアドレス（ジェネラルコール等のアドレス）は使用できません。
・ C-BUS 、M3L-BUS等のI2Cバス互換プロトコルとの混在には対応していません。
・ リスタート条件を用いて、スレーブの切り替えを行う通信フォーマットをバス上に
流さないで下さい。（通信に異常をきたします。）
・バス衝突割り込みベクタが共用の為、UART0とUART3を同時にI2Cバスモードでは、使用できません。
また同様にUART1とUART4を同時使用できません。

2. 機能説明

2.1.アドレス

〈マスタ装置〉
7 ビットアドレスを持つスレーブ装置との送受信
〈スレーブ装置〉
7 ビットアドレスを持っています。
注 ：特別なアドレス（ジェネラルコール等のアドレス）に対する送受信は対応しておりません。
転送速度は、0 ～100Kbps です。よって、高速モードのマスタとの通信はできません。

2.2.転送速度

〈マスタ装置〉

2.3 1～256バイトのデータ送受信が可能です。

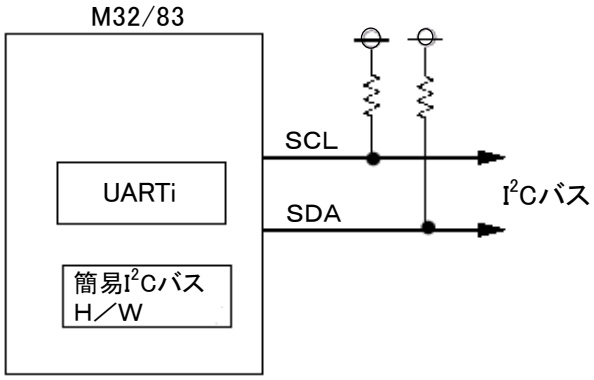
〈スレーブ装置〉
iic0_slave_end()の関数の引数であるiic_indexに与えられます。
その範囲は1～256です。

2.4.マルチマスタ

I2Cバスに接続している、複数の装置が他の装置へのデータ伝送を行うことができます。

3. ハードウェア説明

I2CバスプロトコルをM32C/83のUARTi(i=0～4)+内蔵簡易I2CバスH/W のみを
使用します。プルアップ抵抗は、システムに合わせて適切に選定してください。



4. 使用方法

4.1.組み込み方法

[ncrt.a30]

① near 領域の最後のセクションエントリとして以下の記述を加えてください。
これにより、ソフトI2Cバスで使用するRAM 領域10バイトを確保する場所を確定します 。

.section iicbus,data,align

② 以下の割込ベクタ追加してください。(i=0～4)
ソフトウェア割込番号39、40、41(バス衝突検出割込)
.glb s?s_int
.lword s?s_int (?=0～4)
ソフトウェア割込番号18、20、34、36、38(UARTi 受信割込)
.glb s?r_int
.lword s?r_int (?=0～4)
ソフトウェア割込番号17、19、33、35、37(UARTi 送信割込)
.glb s?t_int
.lword s?t_int (?=0～4)

③ 割り込み許可"fset I"を実行してください。

[アクセス禁止レジスタ]
以下に示すレジスタを変更しないで下さい。
(i=0～4 ただし、I2Cバスとして使用しているUARTのレジスタに限る)

レジスタ名	ビット							
	7	6	5	4	3	2	1	0
UARTiバス衝突検出割込制御レジスタ	×	×	×	×	×	×	×	×
UARTi 送信割込制御レジスタ	×	×	×	×	×	×	×	×
UARTi 受信割込制御レジスタ	×	×	×	×	×	×	×	×
UARTi 特殊モードレジスタ	×	×	×	×	×	×	×	×
UARTi 特殊モードレジスタ2	×	×	×	×	×	×	×	×
UARTi 特殊モードレジスタ3	×	×	×	×	×	×	×	×
UARTi 特殊モードレジスタ4	×	×	×	×	×	×	×	×
UARTi 送受信モードレジスタ	×	×	×	×	×	×	×	×
UARTi 転送速度レジスタ	×	×	×	×	×	×	×	×
UARTi 送信バッファレジスタ	×	×	×	×	×	×	×	×
UARTi 送受信制御レジスタ0	×	×	×	×	×	×	×	×
UARTi 送受信制御レジスタ1	×	×	×	×	×	×	×	×
UARTi 受信バッファ	×	×	×	×	×	×	×	×
ポート6	×1	×1	○	○	×0	×0	○	○
ポート6 方向レジスタ	×1	×1	○	○	×0	×0	○	○
ポート7	○	○	○	○	○	○	×2	×2
ポート7 方向レジスタ	○	○	○	○	○	○	×2	×2
ポート9	×4	×4	○	○	○	×3	×3	○
ポート9 方向レジスタ	×4	×4	○	○	○	×3	×3	○
機能選択レジスタA0	×1	×1	○	○	×0	×0	○	○
機能選択レジスタA1	○	○	○	○	○	○	×2	×2
機能選択レジスタA3	×4	×4	○	○	○	×3	×3	○
機能選択レジスタB0	×1	×1	○	○	×0	×0	○	○
機能選択レジスタB1	○	○	○	○	○	○	×2	×2
機能選択レジスタB3	×4	×4	○	○	○	×3	×3	○
機能選択レジスタC	○	○	○	○	○	○	×2	×2
外部割り込み要因選択レジスタ	×B	×A	○	○	○	○	○	○

×i:UARTiを簡易I2Cバスと使用する場合アクセス禁止(i=0～4)
×A:UARTjを簡易I2Cバスと使用する場合アクセス禁止(j=0、3)
×B:UARTkを簡易I2Cバスと使用する場合アクセス禁止(k=1、4)

4.2.使用メモリ

RAMサイズ データ	10バイト
ROM サイズ	1430バイト

4.3.関数

〇初期化関数:

char iic_ini(char SWITCH);

機能:I²Cバスで送受信するための初期化処理を行います。この処理が終了し、
割込許可の状態であればスレーブ装置として動作します。
また、以下に述べるマスタ送受信を開始する関数をコールすることで、
マスタ装置として動作するようになります。

引数	SWITCH	0: IIC機能 無効
		1:IIC機能 有効
戻り値		0:失敗
		1:成功

その他 :
IIC機能無効時は、次から示す関数を使用することはできない。

〇Master制御開始関数

char iic_master_start(char SLAVE, char RW, char * BUF, char LEN);

機能: マスタ制御開始を開始します。
この関数を使用するには、iic_ini によりI²Cバスを使用できる状態にしておく必要があります。

引数	SLAVE	0x00～0x7f:指定するスレーブ装置アドレス
	RW	0:マスタ送信動作
		1:マスタ受信動作
	*BUF	:送信バッファ or 受信バッファ へのpointer
	LEN	0x00～0xff:通信データ長
戻り値		0:Master制御開始失敗
		1:Master制御開始成功

※Master制御開始関数の先頭で"fclr i"を実行し、割り込みを禁止しています。
これは、バスビジー判定後にスタートコンディションをなるべく早く出力するための処理です。

〇Master E2PROM ランダムリード開始関数

char iic_master_randomread(char SLAVE, char ROM_ADR, char * BUF, char LEN);

機能: E²PROM に対するランダムリードを開始します。
この関数を使用するには、iic_ini によりI²Cバスを使用できる状態にしておく必要があります。

引数	SLAVE	0x00～0x7f:指定するE2PROM装置のアドレス
	ROM_ADR	:読込対象の E2PROM 内のアドレス
	*BUF	:受信バッファ へのpointer
	LEN	0x00～0xff:受信データ数
戻り値		0:Master制御開始失敗
		1:Master制御開始成功

[ユーザが作成するソフトI²Cバス関数]
ソフトI²Cバスでは、送受信のステータスと、そのデータ数を引数とする以下の関数を
呼び出します。
この関数は、ソフトI²Cバスに対してユーザが提供しなければなりません。

○Master 制御完了関数
　void iic_master_end(char STATUS);
機能: Master 制御完了後 F/W がコールする関数です。
　マスタ通信の終了状態を以下に示す引数によりユーザに知らせます。

引数	STATUS	上位4bit	0:マスタ送信
			1:マスタ受信
			2:E²PROM ランダムリード
		下位4bit	0:正常終了
			2:バス競合負け
			3:NACK 終了

戻り値　なし
その他:ソフトI²Cバスの割込処理の中からコールしています。

○Slave check 関数
　＊ char iic_id_chk(char ID, char RW);
機能:1st Byte 受信後 F/W がコールする関数です。マスタからのスレーブへの要求内容を
　次に示す引数によりユーザに知らせます。戻り値にNULLポインタを返すと
　スレーブ指定を拒否し、通信バッファのポインタを返せば、スレーブ動作を開始します。

引数	ID	0x00～0x7f:マスタが指定しているスレーブ装置アドレス
	RW	0:マスタは受信を要求(スレーブ受信する)
		1:マスタは送信を要求(スレーブ送信する)
戻り値	NULL pointer	:スレーブ指定拒否
	pointer	:送信バッファ or 受信バッファへの Pointer

○Slave 制御完了関数
　void iic_slave_end(char STATUS,char IIC_INDEX);
機能:Slave 制御完了後 F/W がコールする関数。
　スレーブ通信の終了状態を以下に示す引数によりユーザに知らせます。

引数	STATUS	上位4bit	0:スレーブ受信
			1:スレーブ送信
		下位4bit	0:正常終了
	IIC_INDEX	0x00～0xff:受信データ数	
戻り値	なし		

4.4.通信方法

4.4.1.準備

I2Cバスの通信を行うためには、以下の例のようにイニシャルルーチンなどのプログラム動作初期段階でiic_ini をコールする必要があります。
iic_ini をコールするときには、1つの引数を渡します。
第1引数は、I2Cバス機能の有効／無効を表します。

```
(システム初期化処理)
char iic_ini(1);                    // I2C-BUS 初期化
asm("fset I");                      // I フラグセット
```

4.4.2.マスタ通信

マスタ通信を開始するには、iic_master_startをコールします。iic_master_startをコールするときには、4つの引数を渡します。
第1引数は、送信相手のアドレスを指定します。送信相手のアドレスには、機能アドレスを指定しないでください。
第2引数は、マスタ通信の送受信の指定を行います。0なら送信を、1なら受信を行います。
第3引数は、データ格納先先頭アドレスを指定します。このデータ格納先は、マスタ通信が終了するまでに解放されなければnear 属性のRAM 領域のどこにでも配置できます。
第4引数は、送信データ長を指定します。0を指定すると最大送信データ長である、256バイトを送信します。
また、iic_master_startは戻り値をもっています。マスタ通信が開始された場合0を 、開始されなかった場合1 を返します。

以下の例では、55_{16} というアドレスを持つスレーブ装置にiic_ram から5 バイトのデータを送信します。

例)

```
if (iic_master_start(0x55, 0, char *iic_ram, 5)!=0 ){
    // マスタ通信失敗確認処理
}else{
    // マスタ通信開始確認処理
}
```

マスタ通信が終了すると、ソフトI2Cバスはiic_master_endをコールします。この関数は、
ユーザが作成する必要があります。ソフトI2Cバスがiic_master_endをコールするとき、
1つの引数渡します。引数は、マスタ送信終了ステータスを示しています。
ステータスの内容は4.3 項に示してあります。
次にiic_master_endの例を示します。

```
// プロトタイプ
void iic_master_end(unsigned char);
// マスタ通信終了関数
 void iic_master_end(char status){
    if((status&0xf0)==0x10){        //送信モード
        switch (status&0x0f){       //送信ステータス処理
            case 0:
                // 正常終了
                break;
            case 1:
                // 第1 バイトにNACK
                break;
            case 2:
                // 第2 バイト以降にNACK
                break;
            case 3:
                // BUS 競合負け
                break;
            case 4:
                // 不正スタート条件
                break;
            case 5:
                // 不正ストップ条件
                default:break;
        }
    }else if((status&0xf0)==0x20){     //受信モード
                                       //送信ステータス処理時と同様に受信ステータス処理を行ってください。
    }
    }else if((status&0xf0)==0x30){     //EEPROMモード
                                       //送信ステータス処理時と同様に
    }                                  //EEPROMモードステータス処理を行ってください。
}
```

4.4.3.スレーブ通信

マスタからのデータの1Byte受信時、ソフトI²Cバスは関数iic_id_chkをコールします。
この関数は、ユーザが作成する必要があります。
ソフトI²Cバスは、マスタからのスレーブへの要求内容を次に示す引数に渡します。
第一引数はマスタが指定しているスレーブ装置のアドレスを示します。
第二引数はマスタの要求している通信内容を示します。
戻り値にNULLポインタを返すとスレーブ指定を拒否し、
通信バッファのポインタを返せば、スレーブ動作を開始します。

```
// プロトタイプ
* char iic_id_chk(char, char);

// スレーブチェック関数
unsigned char sw_buf[256]                   //送信バッファ
unsigned char sr_buf[256]                   //受信バッファ
* char iic_id_chk(char ID, char RW){
    if(ID==0x55){                           //自装置IDが0x55の時
        if(R/W==1){
            return(&sw_buf[0]);             //スレーブ送信
        }else{
            return(&sr_buf[0]);             //スレーブ受信
        }
    }else{
        return(0)                           //スレーブ操作無し
    }
}
```

スレーブ通信が終了すると、ソフトI2Cバスはiic_slave_endをコールします。この関数は、
ユーザが作成する必要があります。ソフトI2CバスがIic_slave_endをコールするとき、
2つの引数渡します。
第一引数はスレーブ通信のステータスを示します。
第二引数はスレーブ受信のデータ数を示します。
ステータスの内容は4.3 項に示してあります。
次にiic_slave_endの例を示します。

```
// プロトタイプ
void iic_slave_end(char,char);

// スレーブチェック関数
 void iic_slave_end(char status,char iic_index){
    if((status&0xf0)==0x10){        //送信モード
        switch (status&0x0f){       //送信ステータス処理
            case 0:
                // 正常終了
                break;
            case 1:
                // 第1 バイトにNACK
                break;
            case 2:
                // 第2 バイト以降にNACK
                break;
            case 3:
                // BUS 競合負け
                break;
            case 4:
                // 不正スタート条件
                break;
            case 5:
                // 不正ストップ条件
                default:break;
        }
    }else{                          //受信モード
        switch (status&0x0f){       //受信ステータス処理
            case 0:
                // 正常終了(iic_index分のデータ受信あり)
                break;
            case 1:
                // 第1 バイトにNACK
                break;
            case 2:
                // 第2 バイト以降にNACK
                break;
            case 3:
                // BUS 競合負け
                break;
            case 4:
                // 不正スタート条件
                break;
            case 5:
                // 不正ストップ条件
                default:break;
    }
}
```

5.Program List

```
/*""file comment""*********************************************************************
;System:IIC-BUS F/W Ver0.81(Sample program)
;Outline description: M32C/83(20 MHz, not divided internally)
;Multi-master/slave communication
;Communication rate: 100 Kbps
;Functional addresses and 10-bit addresses inhibited
;Only C language supported for the interface
;C bus, M3Low, etc. cannot be connected
;None of fail-safe features incorporated
;Date:Sep./5/2002(Thu)
;Object file name:i2cbus.c
;*********************************************************************************
; copyright 2003 Renesas Technology Corporation
; and Renesas Solutions Corporation
;""file comment end""*****************************************************************/

/*********************************************************************************
; Prototype definition
;*********************************************************************************/
#define uarti 0       //Used UARTx when #define uarti x (x=0～4)
#if uarti == 0
unsigned char iic0_ini(unsigned char);
unsigned char iic0_master_start(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
unsigned char iic0_master_randomread(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
void iic0_master_end(unsigned char);
unsigned char* iic0_id_check(unsigned char,
        unsigned char);
void iic0_slave_end(unsigned char,
        unsigned char);
void s0s_int(void);
void s0r_int(void);
void s0t_int(void);
#elif uarti == 1
unsigned char iic1_ini(unsigned char);
unsigned char iic1_master_start(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
unsigned char iic1_master_randomread(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
void iic1_master_end(unsigned char);
unsigned char* iic1_id_check(unsigned char,
        unsigned char);
void iic1_slave_end(unsigned char,
        unsigned char);
void s1s_int(void);
void s1r_int(void);
void s1t_int(void);
```

```
#elif uarti == 2
unsigned char iic2_ini(unsigned char);
unsigned char iic2_master_start(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
unsigned char iic2_master_randomread(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
void iic2_master_end(unsigned char);
unsigned char* iic2_id_check(unsigned char,
        unsigned char);
void iic2_slave_end(unsigned char,
        unsigned char);
void s2s_int(void);
void s2r_int(void);
void s2t_int(void);
#elif uarti == 3
unsigned char iic3_ini(unsigned char);
unsigned char iic3_master_start(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
unsigned char iic3_master_randomread(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
void iic3_master_end(unsigned char);
unsigned char* iic3_id_check(unsigned char,
        unsigned char);
void iic3_slave_end(unsigned char,
        unsigned char);
void s3s_int(void);
void s3r_int(void);
void s3t_int(void);
#elif uarti == 4
unsigned char iic4_ini(unsigned char);
unsigned char iic4_master_start(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
unsigned char iic4_master_randomread(unsigned char,
        unsigned char,
        unsigned char*,
        unsigned char);
void iic4_master_end(unsigned char);
unsigned char* iic4_id_check(unsigned char,
        unsigned char);
void iic4_slave_end(unsigned char,
        unsigned char);
void s4s_int(void);
void s4r_int(void);
void s4t_int(void);
#endif
static void sta_int(void);
static void stp_int(void);
```

```
/*****************************************************************************
; SFR
;*****************************************************************************/
#if uarti == 0
#pragma ADDRESS uismr4 0364H
#pragma ADDRESS uismr3 0365H
#pragma ADDRESS uismr2 0366H
#pragma ADDRESS uismr 0367H
#pragma ADDRESS uimr 0368H
#pragma ADDRESS uibrg 0369H
#pragma ADDRESS uitb 036aH
#pragma ADDRESS uic0 036cH
#pragma ADDRESS uic1 036dH
#pragma ADDRESS uirb 036eH
#pragma ADDRESS bcniic 0071H
#pragma ADDRESS sitic 0090H
#pragma ADDRESS siric 0072H
#pragma ADDRESS ps0  03b0H
#pragma ADDRESS psl0 03b2H
#pragma ADDRESS p6  03c0H
#pragma ADDRESS pd6  03c2H
#pragma ADDRESS ifsr 031fH
#elif uarti == 1
#pragma ADDRESS uismr4 02e4H
#pragma ADDRESS uismr3 02e5H
#pragma ADDRESS uismr2 02e6H
#pragma ADDRESS uismr 02e7H
#pragma ADDRESS uimr 02e8H
#pragma ADDRESS uibrg 02e9H
#pragma ADDRESS uitb 02eaH
#pragma ADDRESS uic0 02ecH
#pragma ADDRESS uic1 02edH
#pragma ADDRESS uirb 02eeH
#pragma ADDRESS bcniic 0091H
#pragma ADDRESS sitic 0092H
#pragma ADDRESS siric 0074H
#pragma ADDRESS ps0  03b0H
#pragma ADDRESS psl0 03b2H
#pragma ADDRESS p6  03c0H
#pragma ADDRESS pd6  03c2H
#pragma ADDRESS ifsr 031fH
#elif uarti == 2
#pragma ADDRESS uismr4 0334H
#pragma ADDRESS uismr3 0335H
#pragma ADDRESS uismr2 0336H
#pragma ADDRESS uismr 0337H
#pragma ADDRESS uimr 0338H
#pragma ADDRESS uibrg 0339H
#pragma ADDRESS uitb 033aH
#pragma ADDRESS uic0 033cH
#pragma ADDRESS uic1 033dH
#pragma ADDRESS uirb 033eH
#pragma ADDRESS bcniic 008fH
#pragma ADDRESS sitic 0089H
#pragma ADDRESS siric 006bH
#pragma ADDRESS ps1  03b1H
#pragma ADDRESS psl1 03b3H
#pragma ADDRESS psc  03afH
#pragma ADDRESS p7  03c1H
#pragma ADDRESS pd7  03c3H
```

```
#elif uarti == 3
#pragma ADDRESS uismr4 0324H
#pragma ADDRESS uismr3 0325H
#pragma ADDRESS uismr2 0326H
#pragma ADDRESS uismr 0327H
#pragma ADDRESS uimr 0328H
#pragma ADDRESS uibrg 0329H
#pragma ADDRESS uitb 032aH
#pragma ADDRESS uic0 032cH
#pragma ADDRESS uic1 032dH
#pragma ADDRESS uirb 032eH
#pragma ADDRESS bcniic 0071H
#pragma ADDRESS sitic 008bH
#pragma ADDRESS siric 006dH
#pragma ADDRESS ps3  03b5H
#pragma ADDRESS psl3 03b7H
#pragma ADDRESS p9  03c5H
#pragma ADDRESS pd9  03c7H
#pragma ADDRESS ifsr 031fH
#elif uarti == 4
#pragma ADDRESS uismr4 02f4H
#pragma ADDRESS uismr3 02f5H
#pragma ADDRESS uismr2 02f6H
#pragma ADDRESS uismr 02f7H
#pragma ADDRESS uimr 02f8H
#pragma ADDRESS uibrg 02f9H
#pragma ADDRESS uitb 02faH
#pragma ADDRESS uic0 02fcH
#pragma ADDRESS uic1 02fdH
#pragma ADDRESS uirb 02feH
#pragma ADDRESS bcniic 0091H
#pragma ADDRESS sitic 008dH
#pragma ADDRESS siric 006fH
#pragma ADDRESS ps3  03b5H
#pragma ADDRESS psl3 03b7H
#pragma ADDRESS p9  03c5H
#pragma ADDRESS pd9  03c7H
#pragma ADDRESS ifsr 031fH
#endif
#pragma ADDRESS prcr 000aH
union{
 struct{
  char b0:1;
  char b1:1;
  char b2:1;
  char b3:1;
  char b4:1;
  char b5:1;
  char b6:1;
  char b7:1;
 }bit;
 char byte;
}bcniic,sitic,siric,prcr,
uimr,uibrg,uic0,uic1,uismr,uismr2,uismr3,uismr4,
#if uarti == 0 || uarti == 1
ps0,psl0,p6,pd6,ifsr;
#elif uarti == 2
ps1,psl1,psc,p7,pd7;
#elif uarti == 3 || uarti == 4
ps3,psl3,p9,pd9,ifsr;
#endif
```

```
union{
 struct{
  char b0:1;
  char b1:1;
  char b2:1;
  char b3:1;
  char b4:1;
  char b5:1;
  char b6:1;
  char b7:1;
  char b8:1;
  char b9:1;
  char b10:1;
  char b11:1;
  char b12:1;
  char b13:1;
  char b14:1;
  char b15:1;
 }bit;
 struct{
  char bytel:8;
  char byteh:8;
 }byte;
 int word;
}uitb,uirb;
#define UiSMR4 uismr4.byte
#define UiSMR3 uismr3.byte
#define UiSMR2 uismr2.byte
#define UiSMR uismr.byte
#define UiMR uimr.byte
#define UiBRG uibrg.byte
#define UiTB uitb.word
#define UiC0 uic0.byte
#define UiC1 uic1.byte
#define UiRB uirb.word
#define SiSIC bcniic.byte
#define SiTIC sitic.byte
#define SiRIC siric.byte
#define stareq uismr4.bit.b0   // Start condition generate bit
#define rstareq uismr4.bit.b1  // Restart condition generate bit
#define stpreq uismr4.bit.b2   // Stop condition generate bit
#define stspsel uismr4.bit.b3  // SCL,SDA output select bit
#define ackd uismr4.bit.b4     // ACK data bit
#define ackc uismr4.bit.b5     // ACK data output enable bit
#define sclhi uismr4.bit.b6    // SCL output stop enable bit
#define swc9 uismr4.bit.b7     // SCL wait output bit3(final pulse)
#define ckdir uimr.bit.b3      // Internal/external clock select bit
#define csc  uismr2.bit.b1     // Clock synchronous bit
#define swc  uismr2.bit.b2     // SCL wait output bit(9th pulse)
#define als  uismr2.bit.b3     // SDA output stop bit
#define stc  uismr2.bit.b4     // UARTi initialize bit
#define abl  uirb.bit.b11      // Arbitration lost detecting flag
#define PRCR prcr.byte         //
#if uarti == 0
#define ps_scl ps0.bit.b2      // Function select A reg. scl bit
#define psl_scl psl0.bit.b2    // Function select B reg. scl bit
#define ps_sda ps0.bit.b3      // Function select A reg. sda bit
#define p_sda p6.bit.b3        // SDA port data bit
#define p_scl p6.bit.b2        // SCL port data bit
#define pd_sda pd6.bit.b3      // SDA port direction bit
#define pd_scl pd6.bit.b2      // SCL port direction bit
#define IFSR ifsr.byte         //
```

```
#elif uarti == 1
#define ps_scl ps0.bit.b6     // Function select A reg. scl bit
#define psl_scl psl0.bit.b6   // Function select B reg. scl bit
#define ps_sda ps0.bit.b7     // Function select A reg. sda bit
#define p_sda p6.bit.b7       // SDA port data bit
#define p_scl p6.bit.b6       // SCL port data bit
#define pd_sda pd6.bit.b7     // SDA port direction bit
#define pd_scl pd6.bit.b6     // SCL port direction bit
#define IFSR ifsr.byte        //
#elif uarti == 2
#define ps_scl ps1.bit.b1     // Function select A reg. scl bit
#define psl_scl psl1.bit.b1   // Function select B reg. scl bit
#define psc_scl psc.bit.b1    // Function select C reg. scl bit
#define ps_sda ps1.bit.b0     // Function select A reg. sda bit
#define psl_sda psl1.bit.b0   // Function select B reg. sda bit
#define psc_sda psc.bit.b0    // Function select C reg. sda bit
#define p_sda p7.bit.b0       // SDA port data bit
#define p_scl p7.bit.b1       // SCL port data bit
#define pd_sda pd7.bit.b0     // SDA port direction bit
#define pd_scl pd7.bit.b1     // SCL port direction bit
#elif uarti == 3
#define ps_scl ps3.bit.b1     // Function select A reg. scl bit
#define psl_scl psl3.bit.b1   // Function select B reg. scl bit
#define ps_sda ps3.bit.b2     // Function select A reg. sda bit
#define psl_sda psl3.bit.b2   // Function select B reg. sda bit
#define p_sda p9.bit.b2       // SDA port data bit
#define p_scl p9.bit.b1       // SCL port data bit
#define pd_sda pd9.bit.b2     // SDA port direction bit
#define pd_scl pd9.bit.b1     // SCL port direction bit
#define IFSR ifsr.byte        //
#elif uarti == 4
#define ps_scl ps3.bit.b7     // Function select A reg. scl bit
#define psl_scl psl3.bit.b7   // Function select B reg. scl bit
#define ps_sda ps3.bit.b6     // Function select A reg. sda bit
#define p_sda p9.bit.b6       // SDA port data bit
#define p_scl p9.bit.b7       // SCL port data bit
#define pd_sda pd9.bit.b6     // SDA port direction bit
#define pd_scl pd9.bit.b7     // SCL port direction bit
#define IFSR ifsr.byte        //
#endif
/*****************************************************************************
; Memories definition
;*****************************************************************************/
typedef union{
 struct{
  unsigned char b0:1;
  unsigned char b1:1;
  unsigned char b2:1;
  unsigned char b3:1;
  unsigned char b4:1;
  unsigned char b5:1;
  unsigned char b6:1;
  unsigned char b7:1;
 }bit;
 unsigned char all;
}byte_dt;
```

```
typedef union{
 struct{
  unsigned char b0:1;
  unsigned char b1:1;
  unsigned char b2:1;
  unsigned char b3:1;
  unsigned char b4:1;
  unsigned char b5:1;
  unsigned char b6:1;
  unsigned char b7:1;
  unsigned char b8:1;
  unsigned char b9:1;
  unsigned char b10:1;
  unsigned char b11:1;
  unsigned char b12:1;
  unsigned char b13:1;
  unsigned char b14:1;
  unsigned char b15:1;
 }bit;
 struct{
  unsigned char byte0:8;
  unsigned char byte1:8;
 }byte;
 unsigned int all;
}word_dt;
static byte_dt iic_md;              // IICbus mode
#define iic_mode iic_md.all
#define f_rw iic_md.bit.b0          //  0:wite  1:read
#define f_ms iic_md.bit.b4          //  0:slave  1:master
#define f_ep iic_md.bit.b5          //  0:slave  1:master
#define f_sr iic_md.bit.b7          //  0:receive 1:send
static byte_dt iic_sl;              // Master 1st byte
#define iic_slave iic_sl.all
#define iic_rw iic_sl.bit.b0        //  0:write  1:read
static unsigned char iic_length;    // Master length
static unsigned char iic_index;
static unsigned char *iic_pointer; // pointer
static unsigned char iic_eeplen;    //
static unsigned char iic_eepadr;    //
/*****************************************************************************
; IICbus initialize function
;""func comment end""*************************************************************/
#if uarti == 0
unsigned char iic0_ini(unsigned char ini){
#elif uarti == 1
unsigned char iic1_ini(unsigned char ini){
#elif uarti == 2
unsigned char iic2_ini(unsigned char ini){
#elif uarti == 3
unsigned char iic3_ini(unsigned char ini){
#elif uarti == 4
unsigned char iic4_ini(unsigned char ini){
#endif
 if(ini == 1){      // IICbus mode initialize(START)
  UiMR = 0x0a;     // 9bit SI/O mode(ext. clock)
  UiBRG = 100-1;     // 100KBPS
  UiC0 = 0xb0;     // MSB first,f1,Nch,CTS disable
  UiSMR = 0x01;     // IICbus mode,Arbitration lost flag Update per byte
  UiSMR2 = 0x11;     // transfer/receive interrupt,disalbe Clock sync,UART initialize enable
  UiSMR3 = 0x62;     // SDA delay = 4-cycle of BRG count source
  UiSMR4 = 0x30;     // ACK data output(SDA="H")
  UiC1 = 0x15;     // Transfer/Receive enable
```

```
#if uarti == 0
  ps_scl = 1;     // 0:port62  1:psl62
  ps_sda = 1;     // 0:port63  1:TXD0/SDA0
  psl_scl = 0;    // 0:SCL0  1:STXD0
  IFSR |= 0x40;    //
#elif uarti == 1
  ps_scl = 1;     // 0:port66  1:psl67
  ps_sda = 1;     // 0:port67  1:TXD1/SDA1
  psl_scl = 0;    // 0:SCL1  1:STXD1
  pd_sda = 0;     // SDA-port input
  pd_scl = 0;     // SCL-port input
  IFSR |= 0x80;    //
#elif uarti == 2
  ps_scl = 1;     // 0:port71  1:psl71
  ps_sda = 1;     // 0:port70  1:psl70
  psl_scl = 0;    // 0:psc71  1:STXD2
  psl_sda = 0;    // 0:psc70  1:TA0out
  psc_scl = 0;    // 0:SCL2  1:OUTC22
  psc_sda = 0;    // 0:SDA2/TXD2 1:IEout/ISTXD2/OUTC20
  pd_sda = 0;     // SDA-port input
  pd_scl = 0;     // SCL-port input
#elif uarti == 3
  PRCR = 0x04;
  ps_scl = 1;     // 0:port91  1:psl91
  PRCR = 0x04;
  ps_sda = 1;     // 0:port92  1:psl92
  psl_scl = 0;    // 0:SCL3  1:STXD3
  psl_sda = 0;    // 0:SDA3/TXD3 1:IEout/OUTC20
  PRCR = 0x04;
  pd_sda = 0;     // SDA-port input
  PRCR = 0x04;
  pd_scl = 0;     // SCL-port input
  IFSR &= 0xbf;    //
#elif uarti == 4
  PRCR = 0x04;
  ps_scl = 1;     // 0:port97  1:psl97
  PRCR = 0x04;
  ps_sda = 1;     // 0:port96  1:TXD4/SDA4
  psl_scl = 0;    // 0:SCL4  1:STXD4
  PRCR = 0x04;
  pd_sda = 0;     // SDA-port input
  PRCR = 0x04;
  pd_scl = 0;     // SCL-port input
  IFSR &= 0x7f;    //
#endif
  iic_mode = 0x00;   // Slave mode
  iic_index = 0x00;   //
  SiSIC = 0x00;
  SiTIC = 0x00;
  SiRIC = 0x01;    // Receive int. enable
 }
 else{       // Invalidate IICbus(Stop sequence)
  SiSIC = 0x00;
  SiTIC = 0x00;
  SiRIC = 0x00;
  UiC1 = 0x10;    // Transfer/Receive disable
  UiSMR2 = 0x01;    // UART initialize disable
 }
 return(1);
}
```

```
/*****************************************************************************
; IICbus master mode starts function
;""func comment end""*********************************************************/
#if uarti == 0
unsigned char iic0_master_start(unsigned char slave,
#elif uarti == 1
unsigned char iic1_master_start(unsigned char slave,
#elif uarti == 2
unsigned char iic2_master_start(unsigned char slave,
#elif uarti == 3
unsigned char iic3_master_start(unsigned char slave,
#elif uarti == 4
unsigned char iic4_master_start(unsigned char slave,
#endif
        unsigned char rw,
        unsigned char *buf,
        unsigned char len){
 if(uismr.bit.b2 == 1){
  return(0);
 }
 else{
  asm("pushc FLG");
  asm("fclr I");
  UiSMR = 0x01;    // All bit clear without bit0
  UiSMR4 = 0x70;    // SCL and SDA is output"H"
  UiMR = 0x00;    //
  UiMR = 0x02;    //
  UiBRG = 100-1;    //
  SiSIC = 0x01;    // Start con. int. enable
  UiC1 = 0x10;    // Transfer/Receive disable
  UiSMR2 = 0x03;    // UART init. disable,Clock sync. enable
  UiSMR4 = 0x71;    // Start condition generate
  UiSMR4 = 0x09;    // STSP output enable
  iic_slave = slave << 1;  //
  iic_length = len;   //
  iic_pointer = buf;   //
  if(rw == 0){    //
   iic_mode = 0x10;  // Master transfer mode
   iic_rw = 0;
  }
  else{
   iic_mode = 0x11;  // Master receive mode
   iic_rw = 1;
  }
  asm("popc FLG");
  return(1);
 }
}
/*****************************************************************************
; IIC master EEPROM randam-read function
;""func comment end""*********************************************************/
#if uarti == 0
unsigned char iic0_master_randomread(unsigned char slave,
#elif uarti == 1
unsigned char iic1_master_randomread(unsigned char slave,
#elif uarti == 2
unsigned char iic2_master_randomread(unsigned char slave,
#elif uarti == 3
unsigned char iic3_master_randomread(unsigned char slave,
#elif uarti == 4
unsigned char iic4_master_randomread(unsigned char slave,
#endif
```

```
        unsigned char rom_adr,
        unsigned char *buf,
        unsigned char len){
 if(uismr.bit.b2 == 1){
  return(0);
 }
 else{
  UiSMR = 0x01;    // All bit clear without bit0
  UiSMR4 = 0x70;    // SCL and SDA is output"H"
  UiMR = 0x00;    //
  UiMR = 0x02;    //
  UiBRG = 100-1;    //
  SiSIC = 0x01;    // Start int. enable
  UiC1 = 0x10;    // Transfer/Receive disable
  UiSMR2 = 0x03;    // UART init. disable,Clock sync. enable
  UiSMR4 = 0x71;    // Start condition generate
  UiSMR4 = 0x09;    // STSPoutput enable
  iic_slave = slave << 1;  //
  iic_length = 1;    //
  iic_eeplen = len;   //
  iic_eepadr = rom_adr;  //
  iic_pointer = buf;   //
  iic_mode = 0x30;   // Master transfer mode
  iic_rw = 0;
  return(1);
 }
}
/*****************************************************************************
; IIC start/stop condition interrupt function
;""func comment end""*************************************************************/
#if uarti == 0
#pragma INTERRUPT s0s_int
void s0s_int(void){
#elif uarti == 1
#pragma INTERRUPT s1s_int
void s1s_int(void){
#elif uarti == 2
#pragma INTERRUPT s2s_int
void s2s_int(void){
#elif uarti == 3
#pragma INTERRUPT s3s_int
void s3s_int(void){
#elif uarti == 4
#pragma INTERRUPT s4s_int
void s4s_int(void){
#endif
 if(uismr.bit.b2 == 1){   // Start condition interrupt
  sta_int();
 }
 else{       // Stop condition interrupt
  stp_int();
 }
}
```

```
/*****************************************************************************
; IIC start condition function
;""func comment end""*********************************************************/
static void sta_int(void){
 word_dt temp;
 UiC1 = 0x10;       // Transfer/Receive disable
 UiC1 = 0x15;       // Transfer/Receive enable
 UiMR = 0x02;
 UiSMR4 = 0x00;       // STSPSEL = 0 (SI/O output sel)
 temp.byte.byte0 = iic_slave;  // Slave address set
 temp.byte.byte1 = 0x01;     // NACK data set
 UiTB = temp.all;      // Start 1st byte transfer
 UiRB = 0x00;       // Arbitration lost flag clear
 UiSMR2 = 0x1f;       // UART init. enable,Clock sync. enable
 SiSIC = 0x01;       // Stop int. enable
 SiRIC = 0x01;       // /Receive int. enable
 iic_index = 0x00;      //
}
/*****************************************************************************
; IIC stop condition function
;""func comment end""*********************************************************/
static void stp_int(void){
 PRCR = 0x04;
 ps_scl = 0;
 PRCR = 0x04;
 ps_sda = 0;
 UiMR = 0x00;       // port set(Purpose:TXFUL,TBFUL flag must be cleared when slave receive)
 UiMR = 0x0a;       // ext. clock sel
 UiSMR2 = 0x11;       //
 UiSMR4 = 0x30;       // ACK data output"H"
 UiC1 = 0x15;       // transfer/receive enable
 PRCR = 0x04;
 ps_scl = 1;
 PRCR = 0x04;
 ps_sda = 1;
 SiRIC = 0x01;       // receive int. enable
 SiTIC = 0x00;       // transfer int. disable
 SiSIC = 0x00;       // start/stop int. disable
 if(f_ms == 0 && f_sr == 0){   // slave receive
  --iic_index;
#if uarti == 0
  iic0_slave_end(0x00,iic_index);// slave receive complete
#elif uarti == 1
  iic1_slave_end(0x00,iic_index);// slave receive complete
#elif uarti == 2
  iic2_slave_end(0x00,iic_index);// slave receive complete
#elif uarti == 3
  iic3_slave_end(0x00,iic_index);// slave receive complete
#elif uarti == 4
  iic4_slave_end(0x00,iic_index);// slave receive complete
#endif
 }
 iic_mode = 0x00;      // slave mode
 iic_index = 0x00;      //
}
```

```
/*****************************************************************************
; IIC receive(Falling edge of 9th pulse)interrupt function
;""func comment end""*********************************************************/
#if uarti == 0
#pragma INTERRUPT s0r_int
void s0r_int(void){
#elif uarti == 1
#pragma INTERRUPT s1r_int
void s1r_int(void){
#elif uarti == 2
#pragma INTERRUPT s2r_int
void s2r_int(void){
#elif uarti == 3
#pragma INTERRUPT s3r_int
void s3r_int(void){
#elif uarti == 4
#pragma INTERRUPT s4r_int
void s4r_int(void){
#endif
 word_dt temp;
 temp.all = UiRB;      // Read receive buffer
 if(iic_index == 0x00){     // = On the 1st time =
  f_sr = f_ms ^ temp.bit.b8;   // Saved transfer/receive flags
  if(f_ms == 1){      // = When master=
   if(abl == 1){     // = When Arbitration lost =
#if uarti == 0
    iic0_master_end(0x03);  // Arbitration lost error found
#elif uarti == 1
    iic1_master_end(0x03);  // Arbitration lost error found
#elif uarti == 2
    iic2_master_end(0x03);  // Arbitration lost error found
#elif uarti == 3
    iic3_master_end(0x03);  // Arbitration lost error found
#elif uarti == 4
    iic4_master_end(0x03);  // Arbitration lost error found
#endif
    f_ms = 0;     // Changed slave mode
    f_sr = ~f_sr;    // Reversed transfer/receive mode
    UiMR = 0x0a;
    goto r1_slave;    // Go to slave function
   }
   else{       // = When Arbitration win =
    SiTIC = 0x01;    // Transfer int. enable
    UiSMR2 = 0x1b;    // Released SCL line
   }
  }
  else{        // = When Slave =
r1_slave:
   temp.bit.b7 = 0;    // Masked bit7
   if(f_sr == 1){
#if uarti == 0
    iic_pointer = iic0_id_check(temp.byte.byte0,1); // Slave transfer request
#elif uarti == 1
    iic_pointer = iic1_id_check(temp.byte.byte0,1); // Slave transfer request
#elif uarti == 2
    iic_pointer = iic2_id_check(temp.byte.byte0,1); // Slave transfer request
#elif uarti == 3
    iic_pointer = iic3_id_check(temp.byte.byte0,1); // Slave transfer request
#elif uarti == 4
    iic_pointer = iic4_id_check(temp.byte.byte0,1); // Slave transfer request
#endif
   }
```

```
   else{
#if uarti == 0
    iic_pointer = iic0_id_check(temp.byte.byte0,0); // Slave receive request
#elif uarti == 1
    iic_pointer = iic1_id_check(temp.byte.byte0,0); // Slave receive request
#elif uarti == 2
    iic_pointer = iic2_id_check(temp.byte.byte0,0); // Slave receive request
#elif uarti == 3
    iic_pointer = iic3_id_check(temp.byte.byte0,0); // Slave receive request
#elif uarti == 4
    iic_pointer = iic4_id_check(temp.byte.byte0,0); // Slave receive request
#endif
   }
   if(iic_pointer != 0){        // Agreed address
    UiSMR4 = 0xa0;          // ACK-data output enable
                // When Falling edge of last pulse,
                // "L"hold enable
    SiTIC = 0x01;          // Transfer int. enable
    SiSIC = 0x01;          // Start/stop int. enable
    if(f_sr == 1){
     temp.byte.byte0 = *iic_pointer;    // send-data set
     temp.byte.byte1 = 0x01;      // NACK-send set
     UiTB = temp.all;       // Data1 transfer start
    }
    else{
     UiTB = 0x00ff;         // dummy data (with ACK)transfer start
    }
   }
   else{            // disagreed address
    UiSMR4 = 0x30;          // NACK-data output enable
    UiMR = 0x0a;          //
    stc = 1;           // UART initialize enable
    SiRIC = 0x01;          //
    iic_mode = 0x00;
    iic_index = 0x00;
   }
   UiSMR2 = 0x11;           // ALS clear, CSC clear(for Arbitration lost)
  }
 }
 else{           // = On and after the 2nd time =
  if(f_ms == 1){        // = When master=
   if(f_sr == 1){       // = When transfer =
    if(abl == 1){     // = When Arbitration lost =
#if uarti == 0
     iic0_master_end(0x02);  // Arbitration lost error found
#elif uarti == 1
     iic1_master_end(0x02);  // Arbitration lost error found
#elif uarti == 2
     iic2_master_end(0x02);  // Arbitration lost error found
#elif uarti == 3
     iic3_master_end(0x02);  // Arbitration lost error found
#elif uarti == 4
     iic4_master_end(0x02);  // Arbitration lost error found
#endif
     UiMR = 0x0a;    // Ext. clock sel
     UiSMR4 = 0x30;    //
     UiSMR2 = 0x11;    //
     iic_mode = 0x00;   // Slave mode set
     iic_index = 0x00;   //
    }
```

```
    else{       // = When Arbitration win =
     als = 0;     // When Arbitration lost,SDA "HiZ" disable
     SiTIC = 0x01;    // Transfer int. enable
    }
   }
   else{        // = When receive =
    abl = 0;      // Arbitration lost flag clear
    SiTIC = 0x01;     // Transfer int. enable
   }
   swc = 0;       // Released SCL line
  }
  else{         // = When slave =
   if(f_sr == 1){      // = When transfer =
    temp.byte.byte0 = *iic_pointer; // send-data set
    temp.byte.byte1 = 0x01;   // NACK-data set
    UiTB = temp.all;    // Data1 transfer start
   }
   else{        // = When receive =
    UiTB = 0x00ff;     //
   }
   SiTIC = 0x01;      // Transfer int. enable
   swc = 0;       //
   swc9 = 1;       //
  }
 }
}
/*****************************************************************************
; IIC transfer(Falling edge of last pulse)interrupt function
;""func comment end""*********************************************************/
#if uarti == 0
#pragma INTERRUPT s0t_int
void s0t_int(void){
#elif uarti == 1
#pragma INTERRUPT s1t_int
void s1t_int(void){
#elif uarti == 2
#pragma INTERRUPT s2t_int
void s2t_int(void){
#elif uarti == 3
#pragma INTERRUPT s3t_int
void s3t_int(void){
#elif uarti == 4
#pragma INTERRUPT s4t_int
void s4t_int(void){
#endif
 word_dt temp;
 temp.all = UiRB;
 if(iic_index == 0x00){       // = On the 1st time =
  if(f_ms == 1){        // = When master =
   if(temp.bit.b8 == 1){     // = When NACK found =
    if(f_ep == 0){
     if(f_sr == 1){
#if uarti == 0
      iic0_master_end(0x03);  // When the 1st byte , NACK found finish
#elif uarti == 1
      iic1_master_end(0x03);  // When the 1st byte , NACK found finish
#elif uarti == 2
      iic2_master_end(0x03);  // When the 1st byte , NACK found finish
#elif uarti == 3
      iic3_master_end(0x03);  // When the 1st byte , NACK found finish
#elif uarti == 4
      iic4_master_end(0x03);  // When the 1st byte , NACK found finish
#endif
     }
```

```
     else{
#if uarti == 0
      iic0_master_end(0x13);  // When the 1st byte , NACK found finish
#elif uarti == 1
      iic1_master_end(0x13);  // When the 1st byte , NACK found finish
#elif uarti == 2
      iic2_master_end(0x13);  // When the 1st byte , NACK found finish
#elif uarti == 3
      iic3_master_end(0x13);  // When the 1st byte , NACK found finish
#elif uarti == 4
      iic4_master_end(0x13);  // When the 1st byte , NACK found finish
#endif
     }
    }
    else{
#if uarti == 0
      iic0_master_end(0x23);  // When the 1st byte , NACK found finish
#elif uarti == 1
      iic1_master_end(0x23);  // When the 1st byte , NACK found finish
#elif uarti == 2
      iic2_master_end(0x23);  // When the 1st byte , NACK found finish
#elif uarti == 3
      iic3_master_end(0x23);  // When the 1st byte , NACK found finish
#elif uarti == 4
      iic4_master_end(0x23);  // When the 1st byte , NACK found finish
#endif
    }
    als = 0;       // When Arbitration lost,SDA "HiZ" disable
    UiSMR4 = 0x04;      // Stop condition generate
    UiSMR4 = 0x3c;      // ST/SP output enable
    UiSMR2 = 0x01;      //
    SiRIC = 0x01;      // receive int. enable

   }
   else{         // = When ACK found =
    if(f_sr == 0){      // = When receive =
     als = 0;      // When Arbitration lost,SDA "HiZ" disable
     if(iic_length == 1){   // = When receive at the last byte=
      UiTB = 0x01ff;    // Send NACK
     }
     else{       // = When continuous receive =
      UiTB = 0x00ff;    // Send ACK
     }
    }
    else{        // = When transfer =
     if(f_ep == 0){
      temp.byte.byte0 = *iic_pointer;
     }
     else{
      temp.byte.byte0 = iic_eepadr;
     }
      temp.byte.byte1 = 0x01;  // NACK-data set
      UiTB = temp.all;
      abl = 0;     // Arbitration lost flag clear
      als = 1;     // When Arbitration lost,SDA "HiZ" disable
     }
   else{         // = When ACK found =
    if(f_sr == 0){      // = When receive =
     als = 0;      // When Arbitration lost,SDA "HiZ" disable
     if(iic_length == 1){   // = When receive at the last byte=
      UiTB = 0x01ff;    // Send NACK
     }
     else{       // = When continuous receive =
      UiTB = 0x00ff;    // Send ACK
```

```
      else{
       temp.byte.byte0 = iic_eepadr;
      }
       temp.byte.byte1 = 0x01;  // NACK-data set
       UiTB = temp.all;
       abl = 0;      // Arbitration lost flag clear
       als = 1;      // When Arbitration lost,SDA "HiZ" disable
      }
      swc = 1;       // SCL"L"Hold enable
      SiRIC = 0x01;      // Receive int. enable
      if(f_ep == 0 || f_sr == 0){  // When EEPROM read mode address set,Pointer no touch
       ++iic_pointer;    // Pointer moved
     }
     ++iic_index;
    }
   }
    else{          // = When slave =
     UiMR = 0x0a;       // Ext. clock sel.
     stspsel = 0;       // SI/O output enable
     ackc = 0;        // ACKoutput disable
     swc9 = 0;        // SCL"L"Hold3 disable
     swc = 1;        // SCL"L"Hold enable
     ++iic_pointer;       // Pointer moved
    ++iic_index;
    SiRIC = 0x01;
   }
  }
   else{           // = On and after the 2nd time =
    if(f_ms == 1){        // = When master =
     if(iic_length == iic_index){  // = When last data =
      if(f_sr == 0){      // = When receive =
      --iic_pointer;
      *iic_pointer = temp.byte.byte0;
      ++iic_pointer;
      if(abl == 1){      // Arbitration lost found(Self-uint send NACK < other-master send ACK)
       ackd = 1;      // Other-master still transmitting,So transmit finish and stop condition
                      // no generate
       ackc = 1;
       UiMR = 0x0a;    // Ext. clock sel
       if(f_ep == 0){
#if uarti == 0
        iic0_master_end(0x10); // Master normally finish
#elif uarti == 1
        iic1_master_end(0x10); // Master normally finish
#elif uarti == 2
        iic2_master_end(0x10); // Master normally finish
#elif uarti == 3
        iic3_master_end(0x10); // Master normally finish
#elif uarti == 4
        iic4_master_end(0x10); // Master normally finish
#endif
      }
```

```
      else{
#if uarti == 0
       iic0_master_end(0x20); // Master normally finish
#elif uarti == 1
       iic1_master_end(0x20); // Master normally finish
#elif uarti == 2
       iic2_master_end(0x20); // Master normally finish
#elif uarti == 3
       iic3_master_end(0x20); // Master normally finish
#elif uarti == 4
       iic4_master_end(0x20); // Master normally finish
#endif
      }
      iic_mode = 0x00;
      iic_index = 0x00;
     }
     else{
      UiSMR4 = 0x04;
      UiSMR4 = 0x3c;    // Stop condition generate
      if(f_ep == 0){
#if uarti == 0
       iic0_master_end(0x10); // Master normally finish
#elif uarti == 1
       iic1_master_end(0x10); // Master normally finish
#elif uarti == 2
       iic2_master_end(0x10); // Master normally finish
#elif uarti == 3
       iic3_master_end(0x10); // Master normally finish
#elif uarti == 4
       iic4_master_end(0x10); // Master normally finish
#endif
      }
      else{
#if uarti == 0
       iic0_master_end(0x20); // Master normally finish
#elif uarti == 1
       iic1_master_end(0x20); // Master normally finish
#elif uarti == 2
       iic2_master_end(0x20); // Master normally finish
#elif uarti == 3
       iic3_master_end(0x20); // Master normally finish
#elif uarti == 4
       iic4_master_end(0x20); // Master normally finish
#endif
      }

     }
    }
```

```
    else{
     if(f_ep == 0){
#if uarti == 0
      iic0_master_end(0x00);  //
#elif uarti == 1
      iic1_master_end(0x00);  //
#elif uarti == 2
      iic2_master_end(0x00);  //
#elif uarti == 3
      iic3_master_end(0x00);  //
#elif uarti == 4
      iic4_master_end(0x00);  //
#endif
      UiSMR4 = 0x04;    // Stop condition generate
      UiSMR4 = 0x3c;

     }
     else{
      iic_mode = 0x31;
      iic_length = iic_eeplen;
      iic_rw = 1;
      UiSMR4 = 0x02;    // Restart condition generate
      UiSMR4 = 0x3a;

     }
    }
    UiSMR2 = 0x03;      //
    SiRIC = 0x01;      // receive int. enable
    SiTIC = 0x00;      // transfer int. disable
   }
   else{         // = When continue =
    if(f_sr == 0){      // = When receive =
     --iic_pointer;
     *iic_pointer = temp.byte.byte0;
     ++iic_pointer;
     ++iic_pointer;     // Pointer moved
     ++iic_index;
     if(iic_length == iic_index){
      UiTB = 0x01ff;    // Send NACK
     }
     else{
      UiTB = 0x00ff;    // Send ACK
     }
     swc = 1;
     SiRIC = 0x01;     // receive int. enable
    }
    else{        // = When transfer =
     if(temp.bit.b8 == 1){   // = When NACK found =
#if uarti == 0
      iic0_master_end(0x03);  // When N byte , NACK found finish
#elif uarti == 1
      iic1_master_end(0x03);  // When N byte , NACK found finish
#elif uarti == 2
      iic2_master_end(0x03);  // When N byte , NACK found finish
#elif uarti == 3
      iic3_master_end(0x03);  // When N byte , NACK found finish
#elif uarti == 4
      iic4_master_end(0x03);  // When N byte , NACK found finish
#endif
```

```
        als = 0;
        UiSMR4 = 0x04;    // Stop condition generate
        UiSMR4 = 0x3c;    //
        UiSMR2 = 0x01;
        SiRIC = 0x01;    // receive int. enable

       }
       else{       // = When ACK found =
        temp.byte.byte0 = *iic_pointer;
        temp.byte.byte1 = 0x01;  // NACK-data set
        UiTB = temp.all;
        abl = 0;     // Arbitration lost flag clear
        als = 1;     // When Arbitration lost,SDA "HiZ" disable
        swc = 1;
        SiRIC = 0x01;    // receive int. enable
        ++iic_pointer;    // Pointer moved
        ++iic_index;
       }
      }
     }
    }
    else{          // = When slave =
     UiMR = 0x0a;
     if(f_sr == 1){       // = When transfer =
      if(temp.bit.b8 == 1){    // = When NACK found =
       ackd = 1;      // Output NACK-data
       ackc = 1;      // NACK-data output enable
       swc9 = 0;      // SCL"L"Hold3 disable
       SiSIC = 0x00;     // stop int. disable
       SiRIC = 0x01;     // Receive int. enable
#if uarti == 0
       iic0_slave_end(0x10,iic_index); // Slave transfer complete
#elif uarti == 1
       iic1_slave_end(0x10,iic_index); // Slave transfer complete
#elif uarti == 2
       iic2_slave_end(0x10,iic_index); // Slave transfer complete
#elif uarti == 3
       iic3_slave_end(0x10,iic_index); // Slave transfer complete
#elif uarti == 4
       iic4_slave_end(0x10,iic_index); // Slave transfer complete
#endif
       iic_index = 0x00;    //
       iic_mode = 0x00;    // Slave mode set
      }
      else{        // = When ACK =
       swc9 = 0;      // SCL"L"HOLD3 disable
       swc = 1;      // SCL"L"HOLD enable
       ++iic_pointer;     // Pointer moved
       ++iic_index;
       SiRIC = 0x01;
      }
     }
```

```
    else{          // = When receive =
     --iic_pointer;
     *iic_pointer = temp.byte.byte0;
     ++iic_pointer;
     UiTB = 0x00ff;       // Send ACK
     swc9 = 0;        // SCL"L"HOLD3 disable
     swc = 1;        // SCL"L"HOLD enable
     ++iic_pointer;       // Pointer moved
     ++iic_index;
     SiRIC = 0x01;
    }
   }
  }
 }
```

安全設計に関するお願い

・ 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

・ 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。
・ 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。
・ 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。
・ 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。
・ 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。
・ 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。
・ 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。
・ 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。