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。

マルチマスタ IIC･ドライバソフト
使用法説明書（uPD780024AY）

ご注意

本ソフトウェアはあくまで参考用のソフトであり、当社がこの動作を保証するものではありません。本ソフトウェアを使用する場合、お客様のセット上で十分な評価の上ご使用いただきますようにお願いいたします。

NEC Corporation
April 2002

改版履歴

版数	作成年月日	記事
第一版	平成13年10月11日	第一版作成

NEC Corporation
April 2002

目次

NEC Corporation
April 2002

1．概要

このドキュメントは、NEC製マイコン（uPD780024AY）上で動作するIICBusのドライバソフトの仕様説明について示したものです。

NEC Corporation
April 2002

2. ソフトウェア構成

本章では、本ソフトウェアに関するファイル構成、使用リソースについて示します。

2．1 ファイル構成

本ソフトウェアは、以下のファイルで構成されます。

ファイル名	機能	種別
IIC－BUS．C	IICBus関連処理	ソース（ユーザ向け）
DEFINE．H	各種基本的な型及び定数定義	ヘッダ（デバック向け）
IIC－BUS．H	IICBusユーザ向け変数／定数定義	ヘッダ（ユーザ向け）

上記のうち、ソースファイルをコンパイル・リンクの対象として下さい。ヘッダファイルはソースと同じディレクトリ、
又はサーチパスの通ったディレクトリに置いて下さい。

2．2 使用リソース

本ソフトウェアでは、uPD780024AYの以下のリソースを使用します。

リソース	内容		備考
RAM	IICBus関連	24byte＋2bit	仕様により増減
	DIP関連	34byte＋3bit	
	タイマ割込み関連	2byte＋3bit	
ROM	IICBus関連	約980byte	コンパイル条件によって多少増減します。
	DIP関連	約760byte	
	タイマ割込み関連	約50byte	
	メイン処理関連	約40byte	
	SFR初期化関連	約185byte	
IICBus関連ハードウェア	IICBus通信		
タイマ50	デバック用1msベース・カウント割込み		
タイマ51	デバック用タイムアウト割込み		
I／O PORT	PORT0	デバック用パラメータ・スイッチ	
	PORT1	未使用	
	PORT2	デバック用パラメータ・スイッチ	
	PORT3	デバック用パラメータ・スイッチ SDA／SCLとしてP32／P33を使用	
	PORT4	DIPスイッチ・スキャン入力	
	PORT5	デバック用LED出力	
	PORT6	未使用	
	PORT7	DIPスイッチ・ストローブ出力	
割込み	IICBus転送終了割込み（INTIIC0）及び、デバック用のイベントタイムアウト検出割込み（INTTM50、INTTM51）の計3本		

3. IICBus関連の関数

本章では、IICBusの制御用の関数を示します。（全てIICーBUS．Cに含まれています）

3．1 IICBusのユーザ向け関数

以下にIICBusのユーザ向け関数（インターフェース関数）を示します。

・AddrInit()

	内容	備考
宣言	byte AddrInit(void)	
説明	自局アドレスを設定するための関数です。 この関数内で設定された値が、そのデバイスの自局アドレスとなります。 IICBusの初期化処理の中で呼び出され、この関数の返す値がスレーブ・アドレス・レジスタ0（SVA0）に格納されます。	
戻り値	自局アドレス値	
引数	なし	

・IICBusINIT()

	内容	備考
宣言	void IICBusINIT(void)	
説明	IICBus関連の初期化関数です。 リセット時など、ハードウェアを初期化する必要があるときに呼び出して下さい。 （※この関数を呼び出す場合は自局アドレスを再設定する必要があります）	
戻り値	なし	
引数	なし	

・IICBusReset()

	内容	備考
宣言	void IICBusReset(void)	
説明	IICBus関連の初期化関数です。 この関数では、IICBus関連を初期化しますが、自局アドレスを設定しないのでソフトウェアリセット時などで呼び出して下さい。 状況に応じて、上記のIICBusINIT()と使い分けて下さい。	
戻り値	なし	
引数	なし	

・IICPut()

	内容	備考
宣言	boolean IICPut(void)	
説明	マスタ送信開始用の関数です。 マスタ送信を開始するときに呼び出して下さい。 送信されるアドレス（転送方向）、データはSetBusData()に設定されたものとなります。 デバイスがスレーブ動作中のときは、この関数を呼び出してもマスタ送信を開始せず、送信開始失敗として上位関数に“偽（0）”を返します。 逆に、デバイスがスレーブ動作中ではなく、スターとコンディションを発生出来た場合は送信開始成功として上位関数に“真（1）”を返します。	
戻り値	0：送信開始失敗（デバイスはスレーブ動作中） 1：送信開始成功（スタートコンディション発行）	
引数	なし	

・SetBusData()

	内容	備考
宣言	void SetBusData(byte ＊DataBuff)	
説明	マスタ送信に使用される、送信先アドレスと送信データを設定する関数です。 マスタ送信開始処理を呼び出す前に、この関数で送信先アドレスと送信データを設定して下さい。 送信先アドレスを変数IICTxAddrに、送信データを送信バッファIICTxData[]にそれぞれ設定して下さい。	
戻り値	なし	
引数	＊DataBuff(Dipスイッチの設定値) (※Dipスイッチ値とはデバック用ソフトウェアで使用している値です、実際この関数を用いる時は仕様に合わせて変更して下さい)	仕様により変化

3．2 IICBusの内部関数

以下にIICBusの内部で使用される関数を示します。

・INTIICBus()

	内容	備考
宣言	static void INTIICBus(void)	
説明	IICBusのメイン制御を行っている関数です。 INTIIC0割込みハンドラとして動作しています。IICBusの割込み発生時毎にこの関数が呼ばれIICBusの制御を行います。	
戻り値	なし	
引数	なし	

・doIICBusMaster()

	内容	備考
宣言	static void doIICBusMaster(void)	
説明	IICBusマスタ動作中シーケンス処理の制御を行う関数です。 マスタ動作ステータスによって、各シーケンス処理を呼び出します。	
戻り値	なし	
引数	なし	

・wait46Clock()

	内容	備考
宣言	static void wait46Clock(void)	
説明	IICBusが、通信予約として動作するかどうかを判別するためのウェイト処理を行う関数です。 スタート・コンディション発行後に呼び出します。	
戻り値	なし	
引数	なし	

・revMaster()

	内容	備考
宣言	static void ervMaster(void)	
説明	IICBusの通信予約の登録を行う関数です。 バス未解放時やアービトレーション負け発生時などの、通信予約を行う必要があるときに呼び出します。	
戻り値	なし	
引数	なし	

・doMasterInit()

	内容	備考
宣言	static void doMasterInit(void)	
説明	マスタ送信開始を行う関数です。 バスの解放をチェックし、解放されていたらスタート・コンディションを発行し、アドレスを送信します。未開放の場合は通信予約処理を呼び出します。 ユーザ向け関数IICPut()により呼び出されます。	
戻り値	なし	
引数	なし	

・chkArb()

	内容	備考
宣言	static void chkArb(void)	
説明	アービトレーション負け発生時の処理を行う関数です。 自分への送信でアービトレーション負けが発生した場合は、通信予約処理を呼び出した後、スレーブ動作を行います。 自分以外の送信に対してアービトレーション負けが発生した場合は通信予約処理を呼び出します。	
戻り値	なし	
引数	なし	

・doMasterDataTx()

	内容	備考
宣言	static void doMasterDataTx(void)	
説明	マスタ動作時のデータ送信シーケンス処理を行う関数です。 アービトレーション負けが未発生で、ACKが検出されているときに送信バッファに格納されているデータをIICシフト・レジスタ0(IIC0)に設定します。 ACK未検出時、全データ送信時にマスタ動作終了処理を呼び出します。	
戻り値	なし	
引数	なし	

・doMasterDataRx()

	内容	備考
宣言	static void doMasterDataRx(void)	
説明	マスタ動作時のデータ受信シーケンス処理を行う関数です。 アービトレーション負けが未発生で、ACKが検出されているときに、IICシフト・レジスタ0(IIC0)の値を受信バッファに格納します。 データ受信後ACKを許可し(ACKE0＝1)、ウェイトを解除(WREL0＝1)します。 全データ受信時にマスタ動作終了処理を呼び出します。	
戻り値	なし	
引数	なし	

・termMaster()

	内容	備考
宣言	static void termMaster(void)	
説明	マスタ動作終了処理を行う関数です。 ストップ・コンディションを発行し、各マスターステータスを初期化します。	
戻り値	なし	
引数	なし	

・doIICBusSlave()

	内容	備考
宣言	static void IICBusSlave(void)	
説明	ICBusスレーブ動作中シーケンス処理の制御を行う関数です。 拡張コード検出時、ストップ・コンディション未検出時に、スレーブ動作ステータスによって、シーケンス処理を呼び出します。	
戻り値	なし	
引数	なし	

・doSlaveADRWai()

	内容	備考
宣言	static void doSlaveADRWai(void)	
説明	スレーブ動作時のアドレス待ちシーケンス処理を行う関数です。 アドレス一致を検出した場合、転送方向に応じてスレーブ動作のステータスとウェイト及び割込み要求発生(WTIM0)を設定します。 データ転送方向がスレーブ送信の場合は、送信バッファの1byte目のデータをIICシフト・レジスタ0(IIC0)に設定します。	
戻り値	なし	
引数	なし	

・doSlaveDataTx()

	内容	備考
宣言	static void doSlaveDataTx(void)	
説明	スレーブ動作時のデータ送信シーケンス処理を行う関数です。 ACKが検出されているときに送信バッファに格納されているデータをIICシフト・レジスタ0(IIC0)に設定します。	
戻り値	なし	
引数	なし	

・doSlaveDataRx()

	内容	備考
宣言	static void doSlaveDataRx(void)	
説明	スレーブ動作時のデータ受信シーケンス処理を行う関数です。 IICシフト・レジスタ0(IIC0)の値を受信バッファに格納します。 データ受信後ACKを許可し(ACKE0＝1)、ウェイトを解除(WREL0＝1)します。	
戻り値	なし	
引数	なし	

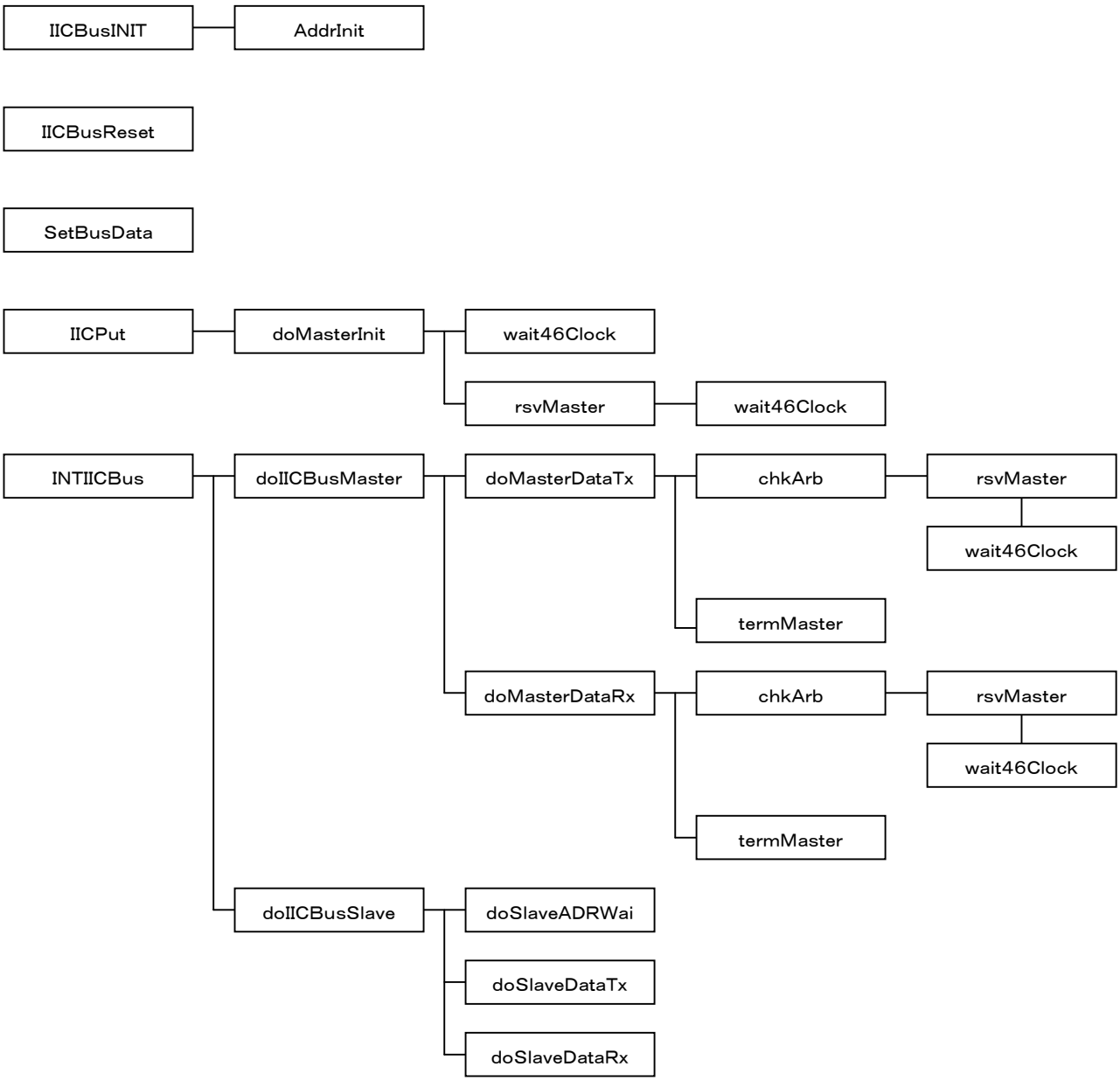
3．3 IICBus関数一覧

以下にIICBusで使用されている関数の一覧を示します。

	名前	機能	戻り値	引数
ユーザ向け関数	AddrInit()	自局アドレス設定処理	byte 自局アドレス	なし
	IICBusINIT()	IICBus初期化処理 （ハードウェアリセット時）	なし	なし
	IICBusReset()	IICBus初期化処理 （ソフトウェアリセット時）	なし	なし
	IICPut()	IICBus転送開始処理	boolean 転送：成功／失敗	なし
	SetBusData()	送信先アドレス／送信データ設定処理	なし	byte ＊DataBuff （仕様により変化）
内部関数	INTIICBus()	IICBus割込みハンドラ （IICBusメイン処理）	なし	なし
	doIICBusMaster()	マスタシーケンス処理	なし	なし
	wait46Clock()	46クロック・ウェイト処理	なし	なし
	rsvMaster()	通信予約処理	なし	なし
	doMasterInit()	マスタ初期状態処理	なし	なし
	chkArb()	アービトレーション負け発生時の処理	なし	なし
	doMasterDataTx()	マスタデータ送信処理	なし	なし
	doMasterDataRx()	マスタデータ受信処理	なし	なし
	termMaster()	マスタ動作終了処理	なし	なし
	doIICBusSlave()	スレーブシーケンス処理	なし	なし
	doSlaveADRWai()	スレーブアドレス待ち処理	なし	なし
	doSlaveDataTx()	スレーブデータ送信処理	なし	なし
	doSlaveDataRx()	スレーブデータ受信処理	なし	なし

3．4 サブルーチン構成図

以下にIICBusで使用されているサブルーチンの構成を示します。



4. IICBus関連の変数

本章では、IICBusの制御用の関数を示します。（全てIICーBUS．Cに含まれています）

4．1 IICBusのユーザ向け変数

以下にIICBusのユーザ向け変数（インターフェース変数）を示します。

・IICTxAddr

	内容	備考
内容説明	送信先アドレスのエリアです。 関数“SetBusData()”内でセットして下さい。 IICTxAddrの最下位bitが転送方向になります。 設定値が“1”のときにマスタ受信、“0”のときにマスタ送信が設定されます。 例） アドレス10Hのデバイスに対してマスタ送信 IICTxAddr ＝ 10H アドレス10Hのデバイスに対してマスタ受信 IICTxAddr ＝ 11H	
サイズ	1byte	
初期値	00H	
シンボル定義	なし	

・IICTxData[]

	内容	備考
内容説明	IICBusの送信データバッファです。 デバイスがマスタ／スレーブに関らず、送信の場合はこのエリアに設定されているデータが、もう一方へ送信されます。 データは関数“SetBusData()”内でセットして下さい。	
サイズ	1byte × 送信データバッファのサイズ （※デバック用プログラムでは、初期値が8となっております）	仕様により変化
初期値	TX__BUFF__SIZE	
シンボル定義	TX__BUFF__SIZE	

・IICRxData[]

	内容	備考
内容説明	IICBusの受信データバッファです。 デバイスがマスタ／スレーブに関らず、受信の場合はこのエリアに送信データが格納されます。 受信されたデータに関してはユーザが任意に使用して下さい。	
サイズ	1byte × 受信データバッファのサイズ （※デバック用プログラムでは、初期値が8となっております）	仕様により変化
初期値	RX__BUFF__SIZE	
シンボル定義	RX__BUFF__SIZE	

4．2 IICBusの内部変数

以下にIICBusの内部で使用される変数を示します。

・IICBuff

	内容	備考
内容説明	IICシフト・レジスタ0（IIC0）値のエリアです。 INTIIC0割込みが発生するごとにIICシフト・レジスタ0の値を格納します。	
サイズ	1byte	
初期値	00H	
シンボル定義	なし	

・IICBusStatus IICBst

	内容	備考
内容説明	IIC状態レジスタ0（IICS0）値を格納する構造体ビットフィールドです。 INTIIC0割込みが発生するごとにIIC状態レジスタ0の値を格納します。	
サイズ	1byte	
初期値	00H	
シンボル説明	格データは以下の通りなっています。 stMSTS：マスタの状態（MSTS0） stALD：アビトレーション負け検出（ALD0） stEXC：拡張コード受信検出（EXC0） stCOI：アドレス一致検出（COI0） stTRC：送信／受信状態検出（TRC0） stACKD：アクノリッジ検出（ACKD0） stSTD：スタート・コンディション検出（STD0） stSPD：ストップ・コンディション検出（SPD0）	

・isMaster

	内容	備考
内容説明	マスタ動作中の通知として使用されるフラグです。 “真（1）”のときはデバイスがマスタ動作中、“偽（0）”のときはデバイスがマスタ動作中以外を示します。 マスタ送信開始時にセット、送信終了時とアービトレーション負け発生時にクリアされます。	
サイズ	1bit	
初期値	0H	
シンボル定義	なし	

・masterPending

	内容	備考
内容説明	マスタ動作予約の通知として使用されるフラグです。 “真（1）”のときはマスタ送信予約あり、“偽（0）”のときはマスタ動作予約なしを示します。 通信予約処理でセット、スタート・コンディション発行後にクリアされます。	
サイズ	1bit	
初期値	0H	
シンボル定義	なし	

・masterStatus

	内容	備考
内容説明	マスタ動作のステータスのエリアです。 このステータスよって、マスタシーケンス処理が行われます。	
サイズ	1byte	
初期値	MST__INIT	
シンボル定義	MST__INIT(0):マスタ初期状態 MST__DATA__TX(1):マスタ送信シーケンス動作中 MST__DATA__RX(2):マスタ受信シーケンス動作中	

・masterCount

	内容	備考
内容説明	マスタ通信時の送受信データバッファ数のカウンタです。 このカウンタに、送受信動作中はバッファ数がセットされ、そのバッファ数分の送受信処理を行います。	
サイズ	1byte	
初期値	00H	
シンボル定義	なし	

・slaveStatus

	内容	備考
内容説明	スレーブ動作のステータスのエリアです。 このステータスよって、マスタシーケンス処理が行われます。	
サイズ	1byte	
初期値	00H	
シンボル定義	SLV__STP__WAI(0):ストップ・コンディション待ち SLV__ADR__WAI(1):アドレス待ち SLV__DATA__TX(2):スレーブ送信シーケンス動作中 SLV__DATA__RX(3):スレーブ受信シーケンス動作中	

4．3 IICBus変数一覧

以下にIICBusで使用されている変数の一覧を示します。

	名前	機能	サイズ	備考
ユーザ向け変数	IICTxAddr	送信先アドレスのエリア （最下位ビットが転送方向を表す1：受信 0：送信）	1byte	
	IICTxData[]	IICBusの送信データバッファ	1byte × バッファ数	仕様により変化
	IICRxData[]	IICBusの受信データバッファ	1byte × バッファ数	仕様により変化
内部変数	IICBuff	IICシフトレジスタ0（IIC0）格納エリア	1byte	
	IICBusStatus IICBst	IIC状態レジスタ0（IICS0）格納ビットフィールド	1byte	
	isMaster	マスタ動作中フラグ	1bit	
	masterPending	マスタ通信予約フラグ	1bit	
	masterStatus	マスタ動作ステータス格納エリア （シーケンスを表す）	1byte	
	masterCount	マスタ送受信のバッファカウンタ	1byte	
	slaveStatus	スレーブ動作ステータス格納エリア （シーケンスを表す）	1byte	
	slaveCount	スレーブ送受信のバッファカウンタ	1byte	

5.　ユーザ向けI／Fの使用方・使用例

本章では、ユーザ向けI／F（ユーザ向け関数・変数）の使用方法と、使用例について示します。

以下にIICBusのユーザ向けI／F（ユーザ向け関数・変数）の使用方法と使用例を示します。

・AddrInit()

自局アドレスの値を決定する関数です。

このアドレスの戻り値がIICBus初期化処理（IICBusINIT()）内でスレーブ・アドレス・レジスタ0（SVA0）に設定されます。

本ソフトウェアでは、DIPスイッチに設定された値を自局アドレスとするため、キースキャンを行った後、値を上位関数に返しています。

尚、IICBus初期化処理（IICBusINIT()）内でスレーブ・アドレス・レジスタ0に直値で自局アドレスを設定する場合はこの関数を呼ぶ必要はありません。

使用例は下記のIICBus初期化処理を参考にして下さい。

＜処理内容＞

```
static byte AddrInit(void)
{
        unsigned char Addr;             /* アドレス値 */
        unsigned char LastAddr;         /* 一回前の獲得アドレス値 */
        unsigned char Loop;             /* ウェイト */
        unsigned char Count=3;          /* チャタリングカウンタ */

        do{
            P7.0 = low;                 /* Dip Switch(アドレス)ストローブ出力 */
            for(Loop = 0; Loop < 3 ; Loop++){       /* ウェイト */
            }
            Addr = P4^0xff;     /* Dip Switch データ取得 */
            P7.0 = high;        /* Dip Switch(アドレス)ストローブ出力停止 */

            if(LastAddr != Addr){       /* アドレスは前回と一致？ */
            LastAddr = Addr;            /* 比較元アドレスを再設定 */
            Count = 3;                  /* チャタリングカウンタ再設定 */
            }else{
                Count--;                /* チャタリング除去カウント */
            }
        }while( Count != 0 );

        return Addr;                    /* 自局アドレス値を返す */
}
```

・IICBusINIT()
IICBus関連のレジスタ、内部ステータスの初期化処理です。
主にハードウェアリセット時に使用して下さい。

＜処理内容＞

```
void IICBusINIT(void)
{
        IICE0     = false;              /* IIC 動作停止 */
        SVA0      = AddrInit();         /* 自局アドレス設定 */
        IICCL0    = 0b00000001;
        IICC0     = 0b10011000;

        slaveStatus = SLV_ADR_WAI;
        isMaster = false;

        masterStatus = MST_INIT;
        masterPending = false;

        IICB_STP();                     /* リセットスタート時のバス解放のため */
}
```

・IICBusReset()
IICBus関連のレジスタ、内部ステータスの初期化処理です。
基本的には上記のIICBus初期化処理(IICBusINIT())と処理はほぼ同じですが、この関数ではスレーブ・アドレス・レジスタ0に値を設定しません。
本ソフトウェアでは、アドレス値がDIPスイッチによって度々変更されるのでこの処理を作る必要がありました。
自局アドレスがハードウェアリセット以降変らない場合は、この処理を使用する必要はありません。

＜使用例＞　:IIBBusが一定時間の間動作しなかった場合、ソフトウェアリセットを行う。

```
void BusStopTime(void)
{
        TM_BusOverFlow--;
        if(TM_BusOverFlow == 0){
          IICBusReset();        /* 通信不能時のバスリセット処理 */
                                /* オーバーフロー割込みが発生した場合は, */
                                /* バスを初期状態に戻す */

          TM_BusOverFlow = TM_BUS_STP;
        }
}
```

・IICPut()

IICBusマスタ送信を開始(スタートコンディション発行＆アドレス送信)を行う関数です。
この関数を使用する場合は使用前にIICBus関連の割込みを禁止して下さい。
この関数を呼び出す前に必ず、送信先アドレス／送信データ設定処理(SetBusData())で各値を設定しておいて下さい。
デバイスがスレーブ動作中のときは、この関数を呼び出してもマスタ送信を開始せず、送信開始失敗として上位関数に“偽(0)”を返します。
逆に、デバイスがスレーブ動作中ではなく、スターとコンディションを発生出来た場合は送信開始成功として上位関数に“真(1)”を返します。
上位関数はこの戻り値を利用して、再送信を行う等の対応をとって下さい。

＜使用例＞ :IICBusマスタ送信要求があった場合送信を開始する。送信結果をRAMに格納する。

```
void RepeatSendData(void)
{
        if(F_SendRequest){              /* IIICBusマスタ送信リクエストあり？ */
            F_SendRequst = false;       /* 送信要求フラグクリア */

            DI();                       /* 割込み禁止 */
            SendResult = IICPut();      /* IICBusマスタ送信開始処理 */
            EI();                       /* 割込み許可 */
        }
}
```

・SetBusData()

マスタ送信に使用される、送信先アドレスと送信データを設定する関数です。
マスタ送信開始処理(IICPut())を呼び出す前に、この関数内で送信先アドレスと送信データを設定して下さい。
送信先アドレスを変数(IICTxAddr)に、送信データを送信バッファ(IICTxData[])にそれぞれ設定して下さい。
本ソフトウェアでは引数を使用してデータを設定しておりますが、仕様を合わせる必要はありません。自由に変更して下さい。

＜使用例＞ :DIPスイッチスキャンにより設定されたデータ(DataBuff[])を送信先アドレス、送信バッファに設定する。

```
void SetBusData(byte *DataBuff)
{
        IICTxAddr = DataBuff[0];        /* アドレス & 転送方向 */

        IICTxData[0] = DataBuff[1];     /* 送信データ設定 */
        IICTxData[1] = DataBuff[2];
        IICTxData[2] = DataBuff[3];
        IICTxData[3] = DataBuff[4];
        IICTxData[4] = DataBuff[5];
        IICTxData[5] = DataBuff[6];
        IICTxData[6] = DataBuff[7];
        IICTxData[7] = DataBuff[8];
}
```

・IICTxAddr
送信先アドレスのエリアです。
送信先アドレス／送信データ設定処理（SetBusData()）内でセットして下さい。
この変数の上位7bitを送信先アドレス値として、最下位bitを転送方向として使用しています。
設定値が“1”のときにマスタ受信、“0”のときにマスタ送信処理となります。

＜使用例＞
1：アドレス10Hのデバイスに対してマスタ送信を行う場合。
IICTxAddr＝10H

最上位bit							最下位bit
0	0	0	1	0	0	0	0

2：アドレス10Hのデバイスに対してマスタ受信行う場合。
IICTxAddr＝11H

最上位bit							最下位bit
0	0	0	1	0	0	0	1

・IICTxData[]
IICBus送信データ用のバッファです。
デバイスがマスタ／スレーブに関らず、転送方向が送信の場合はこのエリアに設定されているデータが、もう一方のデバイスへ送信されます。
データは送信先アドレス／送信データ設定処理（SetBusData()）内でセットして下さい。
本ソフトウェアでは、送受信バッファカウンタに、送信バッファサイズ定数（TX__BUFF__SIZE）を使用しているので、送信先アドレス／送信データ設定処理（SetBusData()）では、全バッファにデータを設定して下さい。
※送信終了なのでバッファのクリアはしていないので、つねにデータが上書きされるため不使用バッファには00H等を設定しておくなどして対応して下さい。

・IICRxData[]
IICBus受信データ用のバッファです。
デバイスがマスタ／スレーブに関らず、転送方向が受信の場合はこのエリアにもう一方のデバイスから送信されたデータが格納されます。
本ソフトウェアでは、送受信バッファカウンタに、受信バッファサイズ定数（RX__BUFF__SIZE）を使用しているので、IICBus受信を行うごとに全受信バッファのデータの受信を行います。
受信されたデータに関してはユーザが任意にIICBus受信バッファ（IICRxData[]）の中から取り出して使用して下さい。

6. IICBus転送におけるSFR切り替えタイミング

本章では、IICBus転送処理中でのSFRの切り替え（設定）のタイミングを示します。

6．1 マスタ送信時のSFR切り替えタイミング

以下にマスタ送信時におけるSFRの設定と設定順を、本ソフトウェア（8byteのデータ送信）を元に示します。

INTIIC0割込み（※▲発生箇所です）	IICBusの動作状態	SFRの設定
	スタート・コンディション発行 アドレス発行（転送方向）	1：STT0＝1（スタート・コンディション発行） 2：WTIM0＝1（9クロックウェイト設定） 3：IIC0＝IICTxAddr（アドレス・転送方向送信）
▲	データ送信（1byte目）	1：IIC0＝IICTxData[]（データ送信）
▲	データ送信（2byte目）	1：IIC0＝IICTxData[]（データ送信）
	⋮	
▲	データ送信（7byte目）	1：IIC0＝IICTxData[]（データ送信）
▲	データ送信（8byte目）	1：IIC0＝IICTxData[]（データ送信）
▲	ストップ・コンディション発行	1：SPT0＝1（ストップ・コンディション発行）
▲	通信予約が設定されている場合はスタート・コンディション発行に戻る	

6．2 マスタ受信時のSFR切り替えタイミング

以下にマスタ受信時におけるSFRの設定と設定順を、本ソフトウェア（8byteのデータ受信）を元に示します。

INTIIC0割込み（※▲発生箇所です）	IICBusの動作状態	SFRの設定
	スタート・コンディション発行 アドレス発行（転送方向）	1：STT0＝1（スタート・コンディション発行） 2：WTIM0＝1（9クロックウェイト設定） 3：IIC0＝IICTxAddr（アドレス・転送方向送信）
▲	データ受信（1byte目）	1：IICRxData[]＝IIC0（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除）
▲	データ受信（7byte目）	1：IIC0＝IICRxData[]（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除）
▲	データ受信（8byte目） ストップ・コンディション発行	1：IIC0＝IICRxData[]（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除） 4：SPT0＝1（ストップ・コンディション発行）
▲	通信予約が設定されている場合はスタート・コンディション発行に戻る	

6．3 スレーブ送信時のSFR切り替えタイミング

以下にスレーブ送信時におけるSFRの設定と設定順を、本ソフトウェア（8byteのデータ送信）を元に示します。

INTIIC0割込み（※▲発生箇所です）	IICBusの動作状態	SFRの設定
	アドレス検出待ち	
▲	アドレス一致 データ送信（1byte目）	1：WTIM0＝1（9クロックウェイト設定） 2：IIC0＝IICTxData[]（データ送信）
▲	データ送信（2byte目）	1：IIC0＝IICTxData[]（データ送信）
	｜	
▲	データ送信（7byte目）	1：IIC0＝IICTxData[]（データ送信）
▲	データ送信（8byte目）	1：IIC0＝IICTxData[]（データ送信）
▲	ストップコンディション待ち	1：WREL0＝1（ウェイト解除）

6．4 スレーブ受信時のSFR切り替えタイミング

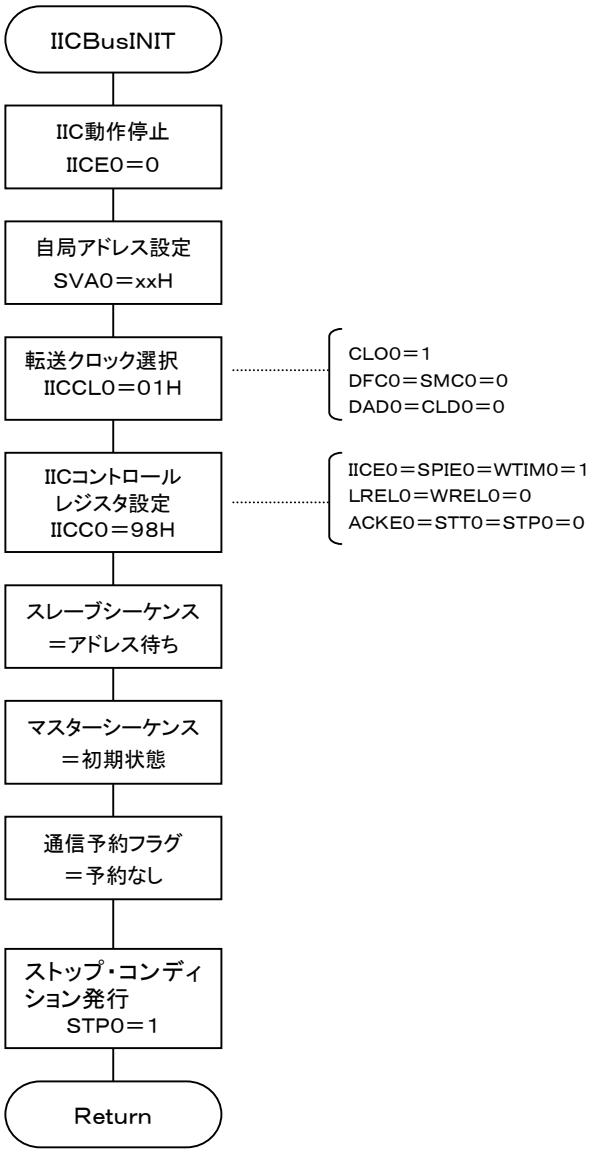
以下にスレーブ受信時におけるSFRの設定と設定順を、本ソフトウェア（8byteのデータ受信）を元に示します。

INTIIC0割込み（※▲発生箇所です）	IICBusの動作状態	SFRの設定
▲	アドレス検出待ち	1：WTIM0＝0（8クロックウェイト設定） 2：WREL0＝1（ウェイト解除）
▲	データ受信（1byte目）	1：IICRxData[]＝IIC0（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除）
▲	データ受信（7byte目）	1：IIC0＝IICRxData[]（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除）
▲	データ受信（8byte目）	1：IIC0＝IICRxData[]（データ受信） 2：ACKE0＝1（ACK許可） 3：WREL0＝1（ウェイト解除）
▲	ストップコンディション待ち	1：WREL0＝1（ウェイト解除）

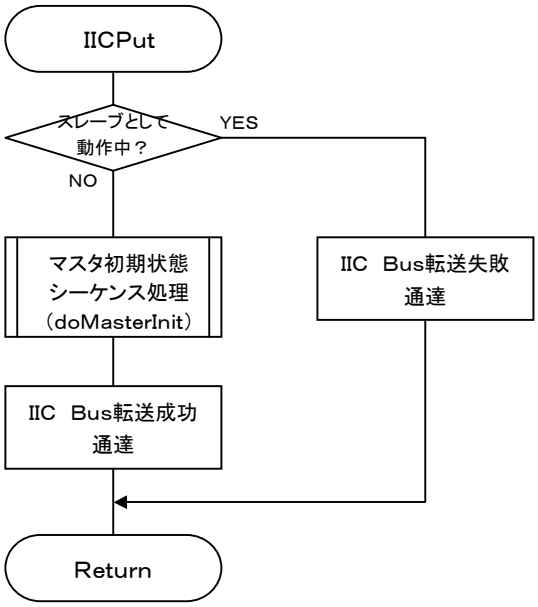
7. IICBusフローチャート

本章では、IICBus処理で使用される処理のフローチャートを示します。

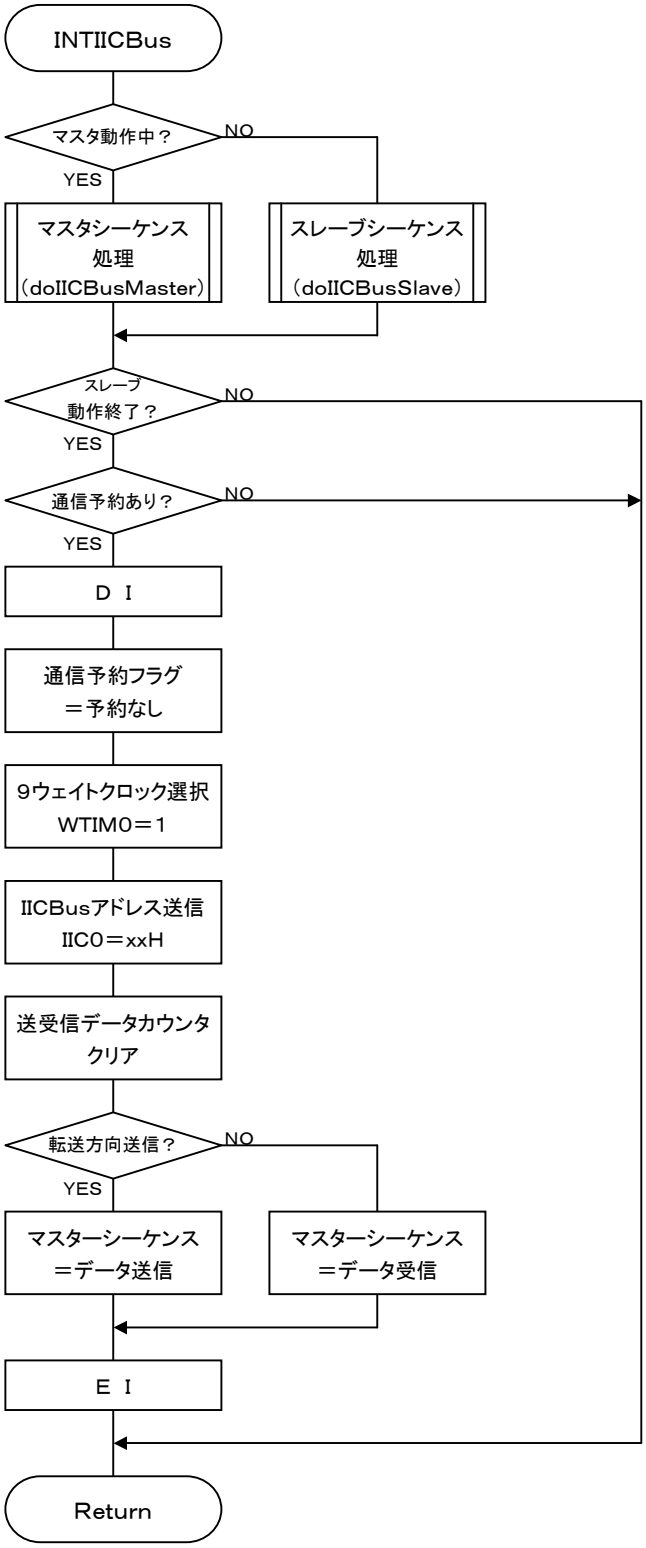
IICBus初期化処理



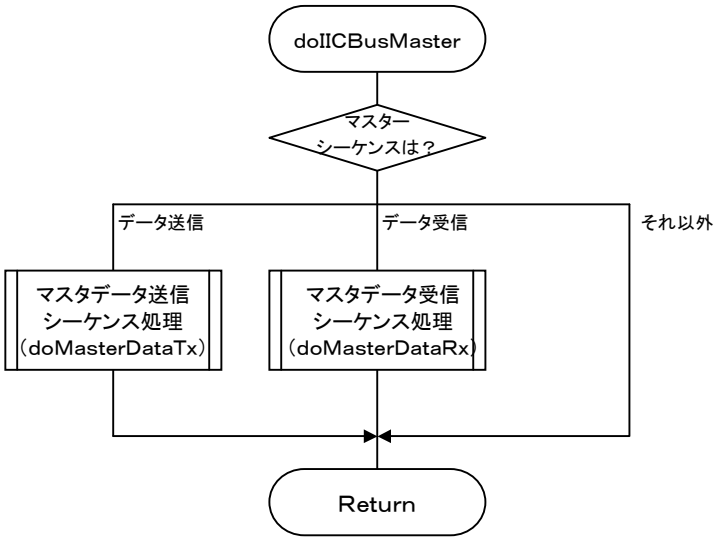
IICBus転送開始処理



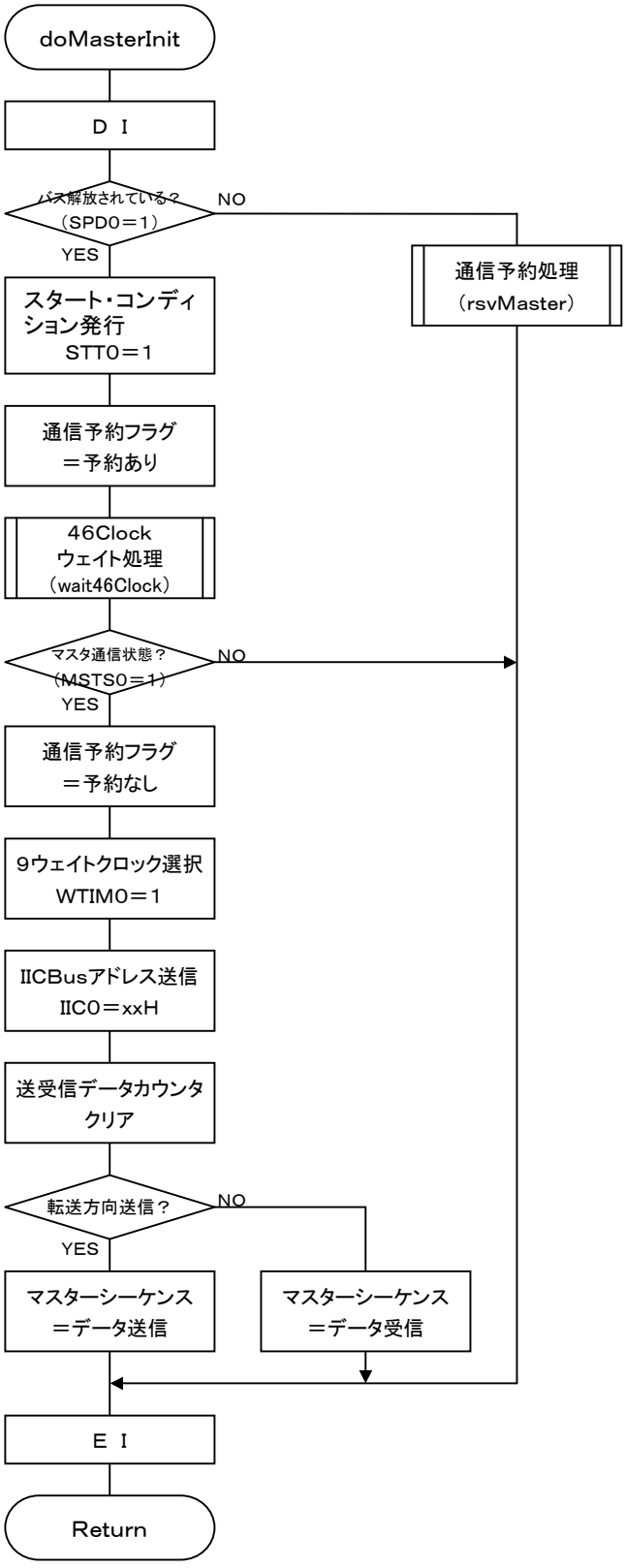
IICBus割込みハンドラ（IICBusメイン処理）



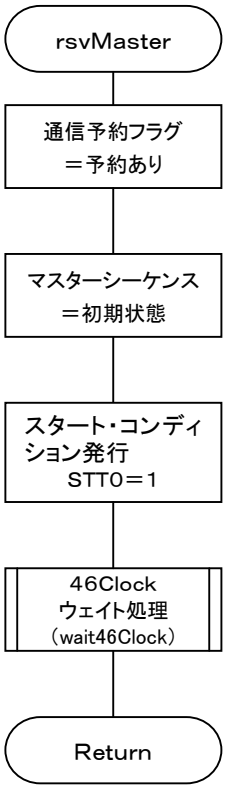
マスタシーケンス処理


doIICBusMaster
マスター
シーケンスは？
データ送信
データ受信
それ以外
マスタデータ送信
シーケンス処理
（doMasterDataTx）
マスタデータ受信
シーケンス処理
（doMasterDataRx）
Return

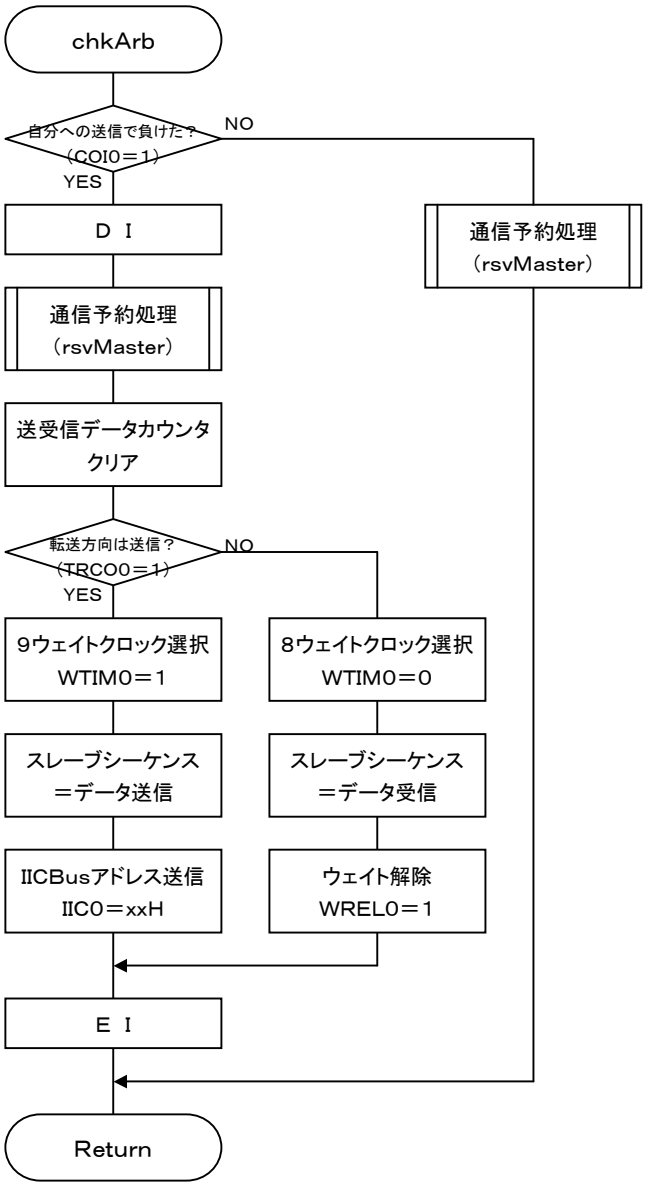
マスタ初期状態処理


doMasterInit
D I
バス解放されている？
（SPD0＝1）
NO
YES
通信予約処理
（rsvMaster）
スタート・コンディション発行
STT0＝1
通信予約フラグ
＝予約あり
46Clock
ウェイト処理
（wait46Clock）
マスタ通信状態？
（MSTS0＝1）
NO
YES
通信予約フラグ
＝予約なし
9ウェイトクロック選択
WTIM0＝1
IICBusアドレス送信
IIC0＝xxH
送受信データカウンタ
クリア
転送方向送信？
NO
YES
マスターシーケンス
＝データ送信
マスターシーケンス
＝データ受信
E I
Return

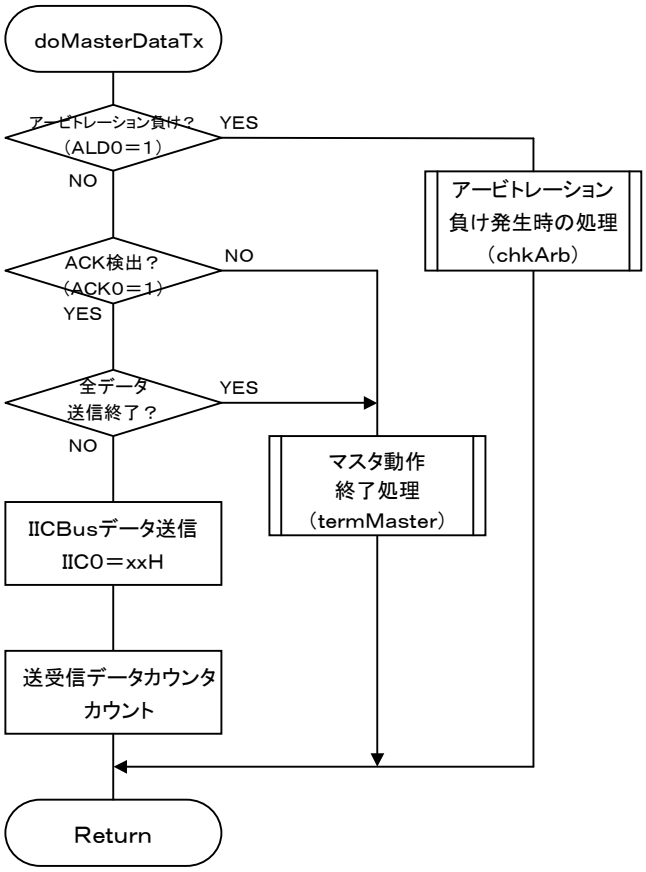
通信予約処理


rsvMaster
通信予約フラグ
＝予約あり
マスターシーケンス
＝初期状態
スタート・コンディション発行
STT0＝1
46Clock
ウェイト処理
（wait46Clock）
Return

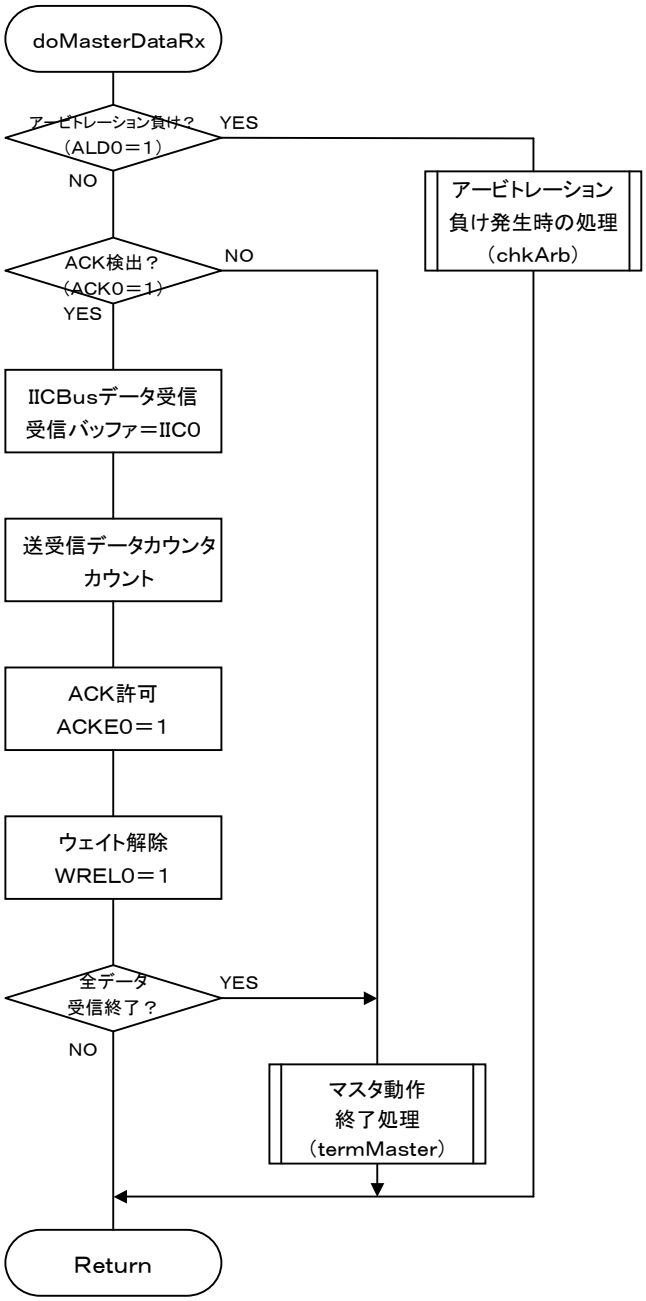
アービトレーション負け発生時の処理



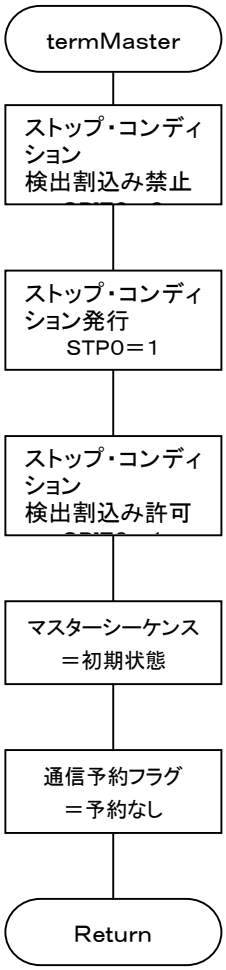
マスタデータ送信処理


doMasterDataTx
アービトレーション負け？
（ALD0＝1）
YES
NO
アービトレーション
負け発生時の処理
（chkArb）
ACK検出？
（ACK0＝1）
NO
YES
全データ
送信終了？
YES
NO
マスタ動作
終了処理
（termMaster）
IICBusデータ送信
IIC0＝xxH
送受信データカウンタ
カウント
Return

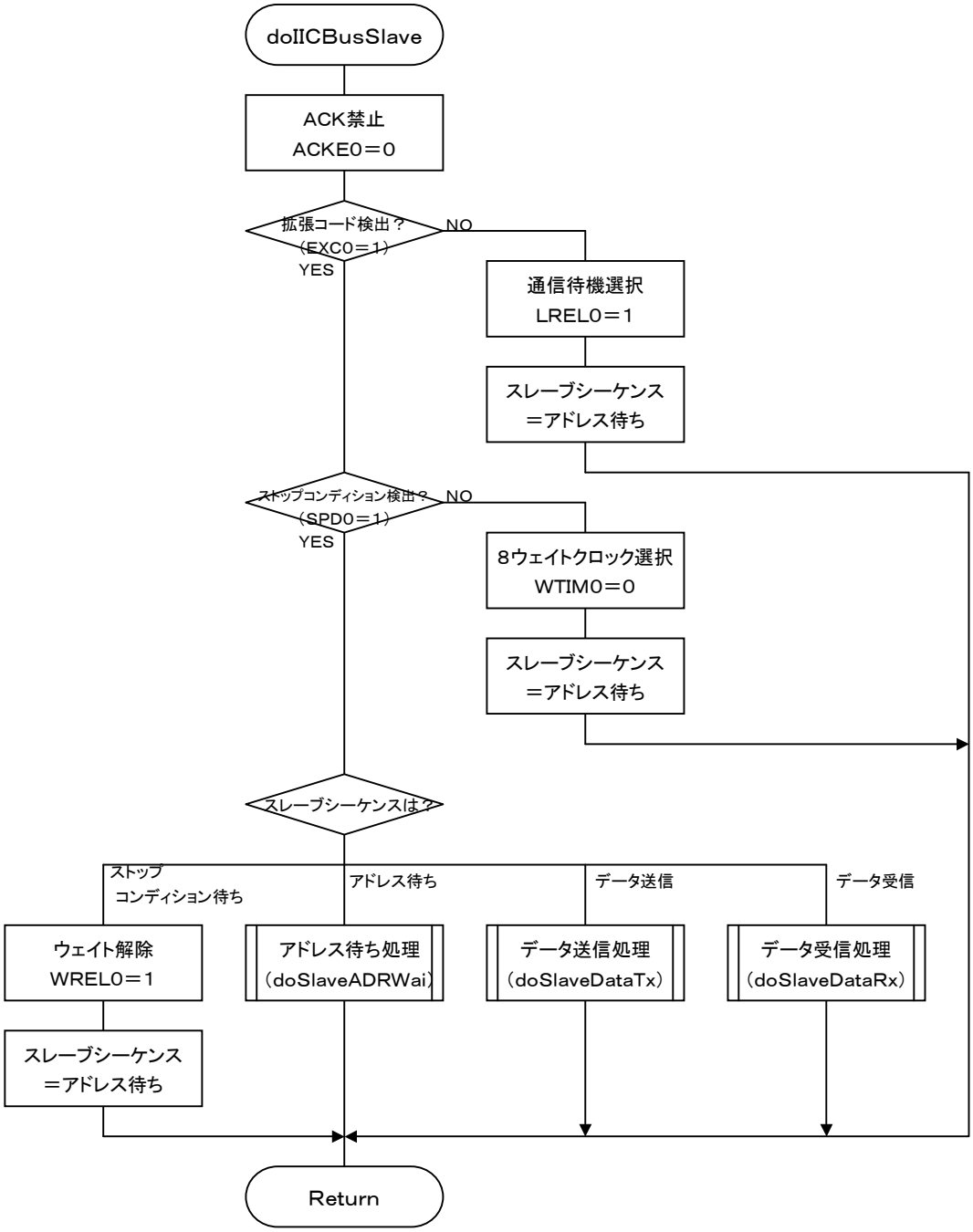
マスタデータ受信処理


doMasterDataRx
アービトレーション負け？
（ALD0＝1）
YES
NO
アービトレーション
負け発生時の処理
（chkArb）
ACK検出？
（ACK0＝1）
NO
YES
IICBusデータ受信
受信バッファ＝IIC0
送受信データカウンタ
カウント
ACK許可
ACKE0＝1
ウェイト解除
WREL0＝1
全データ
受信終了？
YES
NO
マスタ動作
終了処理
（termMaster）
Return

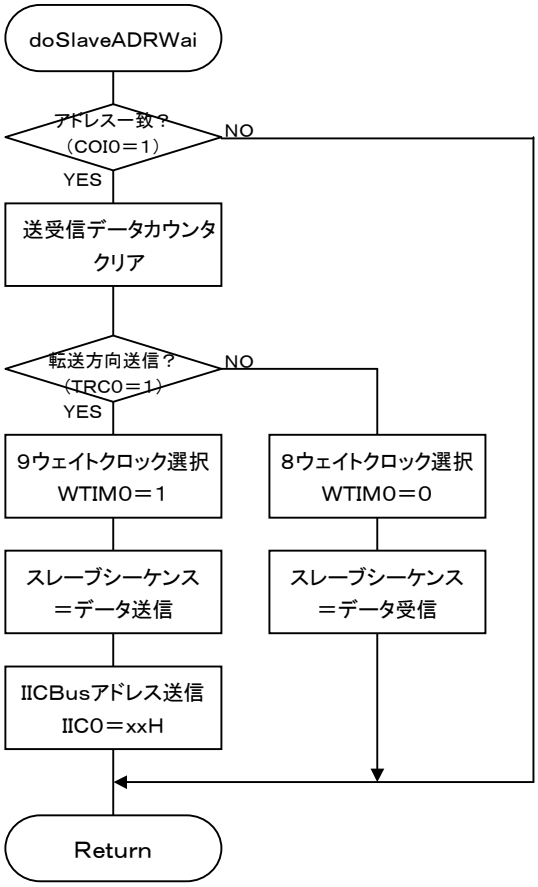
マスタ動作終了処理


termMaster
ストップ・コンディション
検出割込み禁止
ストップ・コンディション発行
STP0＝1
ストップ・コンディション
検出割込み許可
マスターシーケンス
＝初期状態
通信予約フラグ
＝予約なし
Return

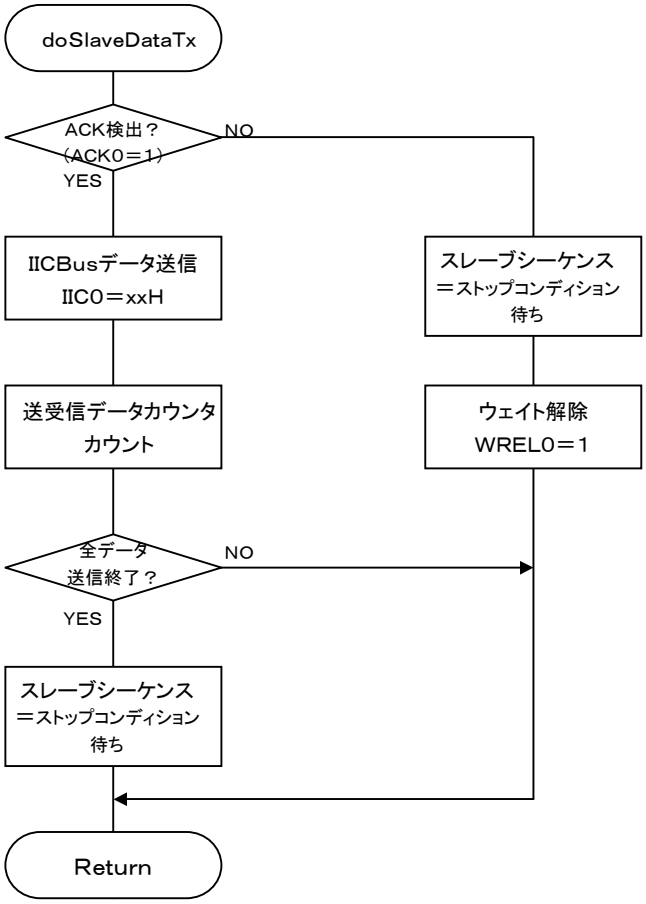
スレーブシーケンス処理



スレーブアドレス待ち処理



スレーブデータ送信処理



スレーブデータ受信処理