H8S/2472, 2463, 2462 グループ
イーサネット送受信設定例

R01AN0645JJ0102
Rev.1.02
2011.07.06

要旨

本資料は，H8S/2472, 2463, 2462 グループのイーサネット設定例を示します。

動作確認デバイス

H8S/2472

目次

1. **はじめに** 2
2. **応用例の説明** 4
3. **参考プログラムリスト** 35

1. はじめに

1.1 仕様

- 本応用例では A~C の 3 種類の処理を選択します。
  A. イーサネットフレームを 1 フレーム送信し，イーサネットフレームを 1 フレーム受信のセットを 2 回します。
  B. イーサネットフレームを 10 フレーム送信します。
  C. イーサネットフレームを 10 フレーム受信します。
- 1 フレームの送信が完了してから次の送信を開始します。
- フレーム送信完了はフレーム送信完了割り込みにより判断します。
- 受信関数をコールすると，1 フレームずつユーザバッファにコピーします。
- PHY 自動交渉結果の取得では, PHY-LSI の自動交渉機能により決定した接続モード (全二重モードまたは半二重モード) を取得します。
- イーサネット PHY-LSI には，SMSC 社の LAN8700 を使用しています。

1.2 使用機能

- イーサネットコントローラ (EtherC)
- イーサネットコントローラ用ダイレクトメモリアクセスコントローラ (E-DMAC)
- 割り込みコントローラ

1.3 適用条件

- マイコン: H8S/2472, H8S/2463, H8S/2462
- 動作周波数: システムクロック　32 MHz
- 総合開発環境: ルネサス エレクトロニクス
  High-performance Embedded Workshop Ver.4.07.00.007
- ツールチェイン: H8S,H8/300 Standard Toolchain (V.6.2.2.0)
- コンパイラオプション: -cpu=2600A:24 -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -nolist -chgincpath -nologo

1.4　PHY-LSIとの接続例

図 1 に SMSC 社の LAN8700 との接続を示します。

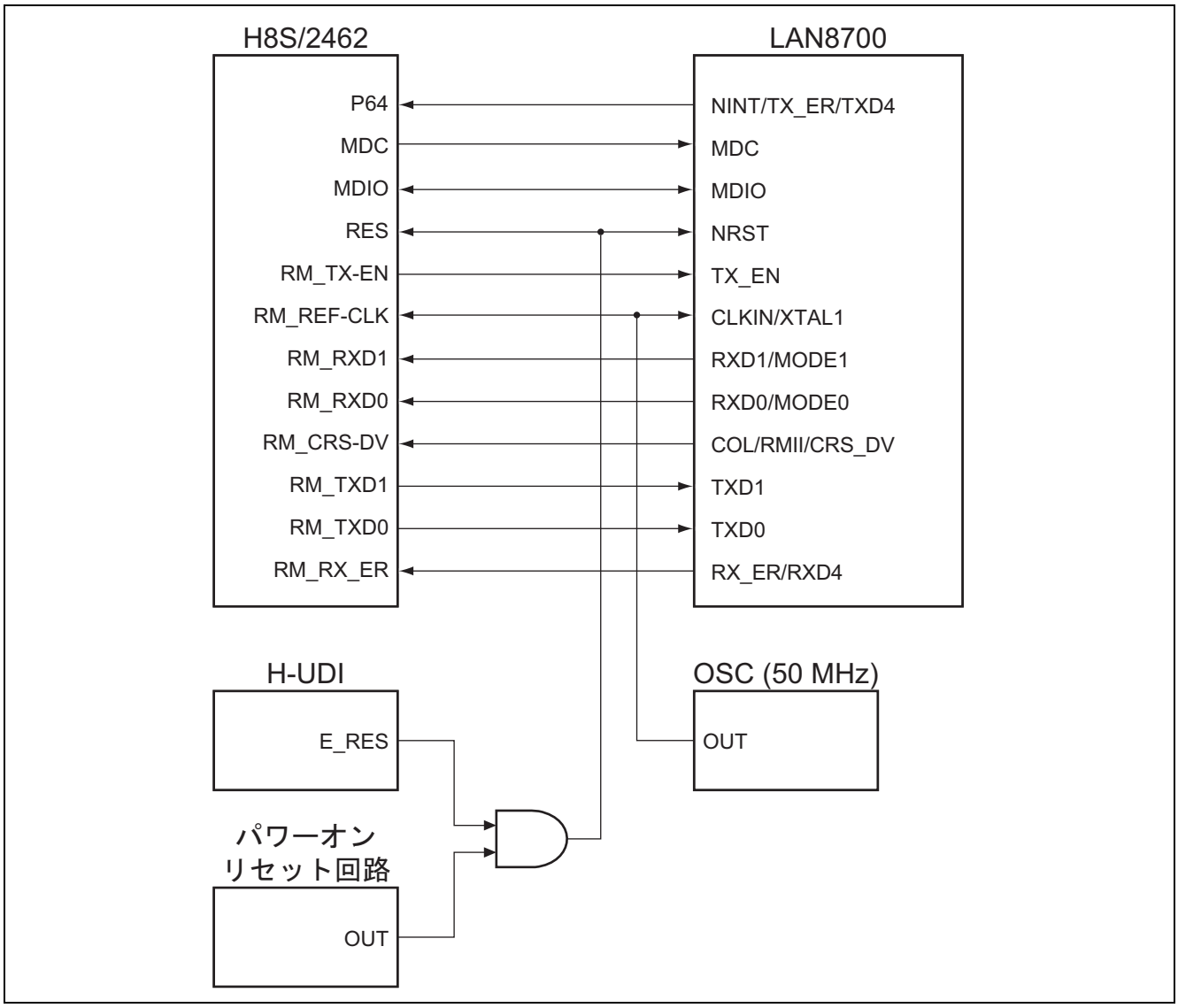


図 1　(RMII **インタフェース**) LAN8700 **との接続例**

2. 応用例の説明

本応用例では，イーサネットコントローラ (EtherC)，およびイーサネットコントローラ用ダイレクトメモリアクセスコントローラ (E-DMAC) を使用します。

本応用例では，イーサネット PHY-LSI により自動交渉を行います。自動交渉結果は EtherC の PHY 部インタフェースレジスタ (PIR) を経由して読み出されます。

2.1 使用機能の動作概要

本 LSI では，イーサネット通信を行う場合かならず EtherC と E-DMAC を使用します。EtherC は送受信制御および MAC 間転送を行います。E-DMAC はその送信/受信 FIFO とユーザが指定するデータ格納先 (バッファ) 間の DMA 転送を専用に行います。

Ethernet PHY-LSI の各種 MII レジスタへのアクセスは，EtherC の PIR を経由して行います。図 2 に MII 管理フレームフォーマット，図 3～図 5 に MII レジスタへのアクセス方法を示します。パルス幅やクロックサイクル時間に制限がありますので，ご注意ください。

アクセス種別	MII管理フレーム							
項目	PRE	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
ビット数	32	2	2	5	5	2	16	—
リード	1‥1	01	10	00001	RRRRR	Z0	D‥D	—
ライト	1‥1	01	01	00001	RRRRR	10	D‥D	X

【記号説明】

PRE: 32個の連続した1
ST: フレームの先頭を表すB'01のライト
OP: アクセスの種類を示すコードのライト
PHYAD: PHY-LSIのアドレスが1の場合，B'00001をライト (MSBから順次ライト)。
このビットは，PHY-LSIのレジスタアドレスによって可変となる
REGAD: PHY-LSIのアドレスが1の場合，B'00001をライト (MSBから順次ライト)。
このビットは，PHY-LSIのレジスタアドレスによって可変となる
TA: MIIインタフェース上でデータの送信元を切り替える時間
(a) リード時は「バス開放」(Z0と表記) を行う
(b) ライト時はB'10をライト
DATA: 16ビットのデータ。MSBから順次ライトあるいはリード
(a) リード時は，16ビットデータのリード
(b) ライト時は，16ビットデータのライト
IDLE: 次のMII管理フォーマット入力までの待機時間
(a) リード時は，すでにTA時にバス開放済みであり制御不要
(b) ライト時は，「単独バス開放」 (Xと表記) を行う

図 2　MII 管理フレームフォーマット

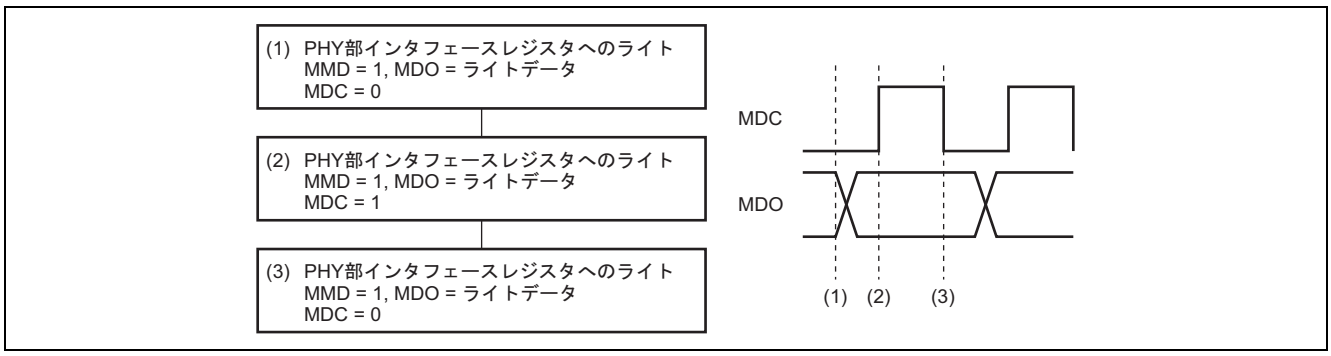


図 3　1 ビットデータのライトフロー

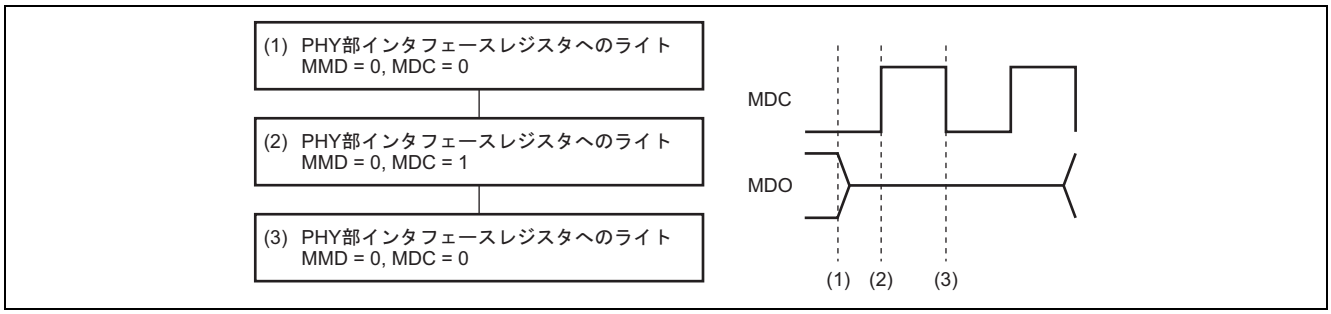


図 4　バス解放フロー

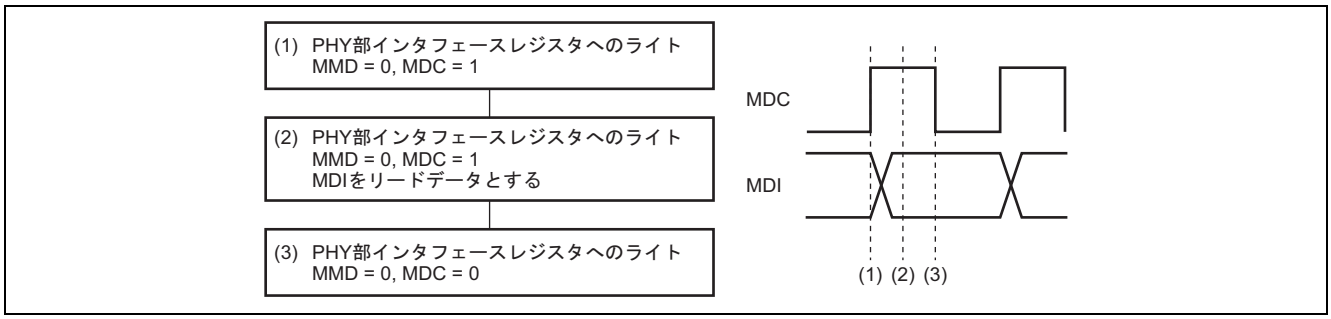


図 5　1 ビットデータのリードフロー

2.1.1　EtherCの概要

本 LSI は，イーサネットあるいは IEEE802.3 の MAC (Media Access Control) 層規格に準拠したイーサネットコントローラ (EtherC) を内蔵しています。EtherC は，同規格に準拠した物理層 LSI (PHY-LSI) と接続することにより，イーサネット/IEEE802.3 フレームの送受信を行うことができます。EtherC は MAC 層インタフェースを 1 系統内蔵しています。また，イーサネットコントローラは，本 LSI 内部でイーサネットコントローラ用ダイレクトメモリアクセスコントローラ (E-DMAC) に接続されており，メモリとの高速アクセスが可能です。

図 6 に EtherC の構成を示します。

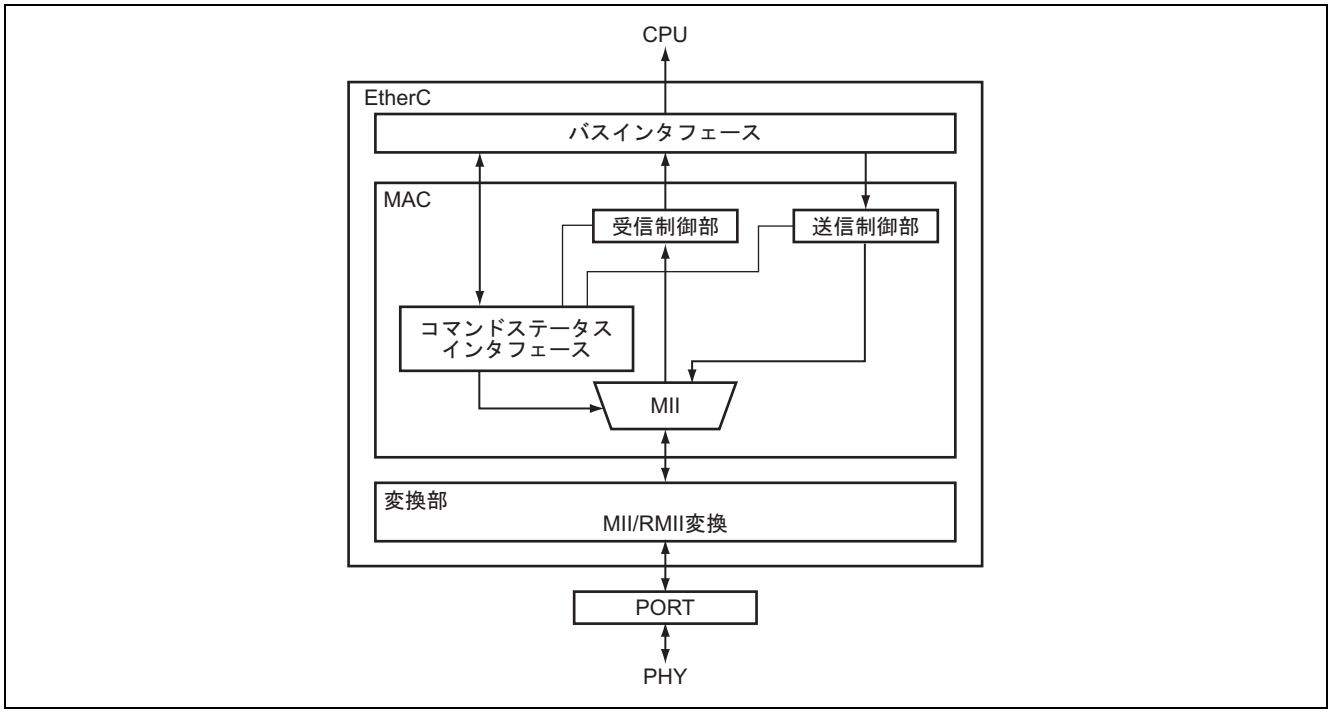
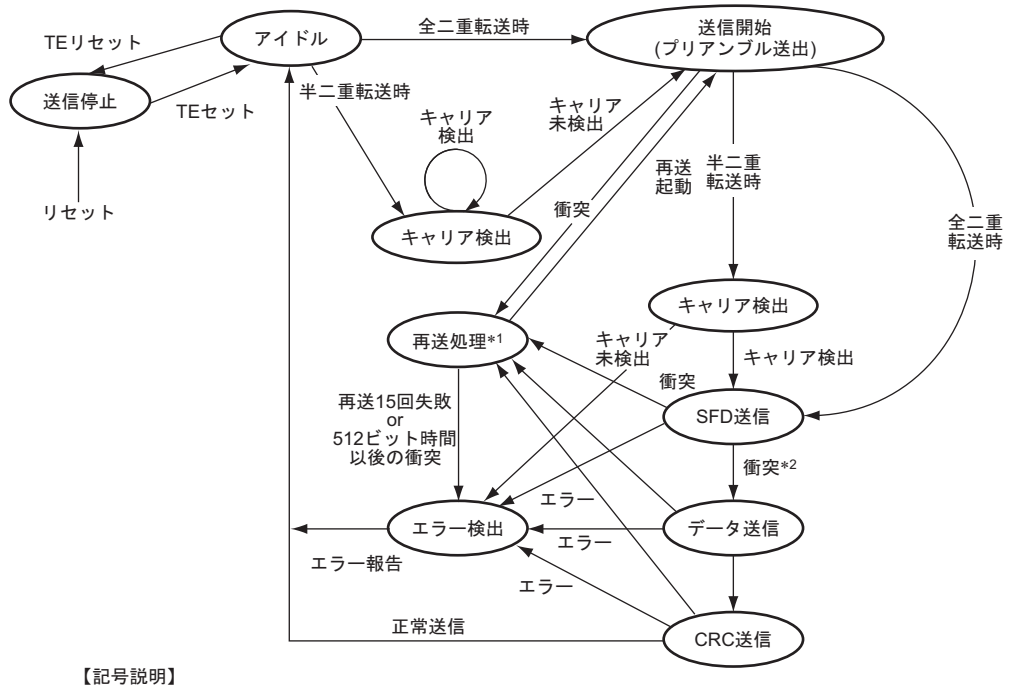


図 6　EtherC の構成

2.1.2　EtherC送信部の概要

EtherC 送信部は，送信 E-DMAC から送信要求があると，送信データをフレームに組み立てて RMII に出力します。RMII を経由した送信データは，PHY-LSI によって回線上に送出されます。Ether-C 送信部の状態遷移図を図 7 に示します。送信動作のフローは以下のようになります。

1. EtherC は, 送信許可ビット (EtherC モードレジスタ (ECMR) の TE ビット) がセットされると送信アイドル状態に遷移します。
2. (A) 半二重転送方式時，E-DMAC 送信部から送信要求があると EtherC はキャリア検出を行い，未検出であればフレーム間隔時間の送信延期を経てプリアンブルを RMII に送出します。キャリアを検出した場合は，キャリアがなくなってからフレーム間隔時間の送信延期を経てプリアンブルを RMII に送出します。
   (B) 全二重転送方式時，キャリア検出を必要とせず，E-DMAC 送信部から送信要求があると即座にプリアンブルを送出します。ただし連続送信時は，直前に送信したフレームから必ずフレーム間隔時間の送信延期を経てプリアンブルを送出します。
3. SFD (Start Frame Delimiter)，データ，CRC (Cyclic Redundancy Check) を順次送信します。送信を終了するとフレーム送信完了割り込み (TC) が発生します。データ送信中に衝突あるいはキャリア未検出状態になるとそれぞれの割り込みが発生します。
4. アイドル状態に遷移し，以後送信データがあれば送信を継続します。



【記号説明】
SFD: Start Frame Delimiter

【注】*1　再送処理には衝突検出にともなうジャムの送信，およびバックオフアルゴリズムによる送信時間間隔の調整が含まれます。
*2　512ビット以下 (プリアンブル，SFD含む) のデータ送出中のみ再送処理を行います。512ビットを超えるデータの送出中に衝突を検出した場合は，ジャムの送出のみを行い，バックオフアルゴリズムによる再送処理を行いません。

図 7　EtherC 送信部状態遷移図

2.1.3　EtherC受信部の概要

EtherC 受信部は，RMII より入力されたフレームをプリアンブル，SFD，データおよび CRC データに分解し，受信 E-DMAC には DA (宛先アドレス) から CRC データまでを出力します。EtherC 受信部の状態遷移図を図 8 に示します。受信動作のフローは以下のようになります。

1. EtherC は EtherC モードレジスタ (ECMR) の受信許可 (RE) ビットがセットされると，受信アイドル状態に遷移します。
2. 受信フレームのプリアンブルに続く SFD を検出すると受信処理を開始します。不当パターンの場合はフレームを破棄します。
3. 通常モードでは，(i) 宛先 MAC アドレスが本 LSI 宛の場合，(ii) ブロードキャストフレームの場合，または(iii) マルチキャストフレームの場合にデータ受信を開始します。プロミスキャスモードでは，フレームの種類にかかわらず受信を開始します。
4. RMII からのフレームを受信後，フレームデータ部の CRC チェックを行います。結果はメモリ上にフレームデータをライトした後，ディスクリプタ内にステータスとして反映されます。異常時は，エラーステータスを EtherC/E-DMAC ステータスレジスタ (EESR) に設定します。
5. 1 フレームを受信後，アイドル状態に遷移し次のフレーム受信に備えます。

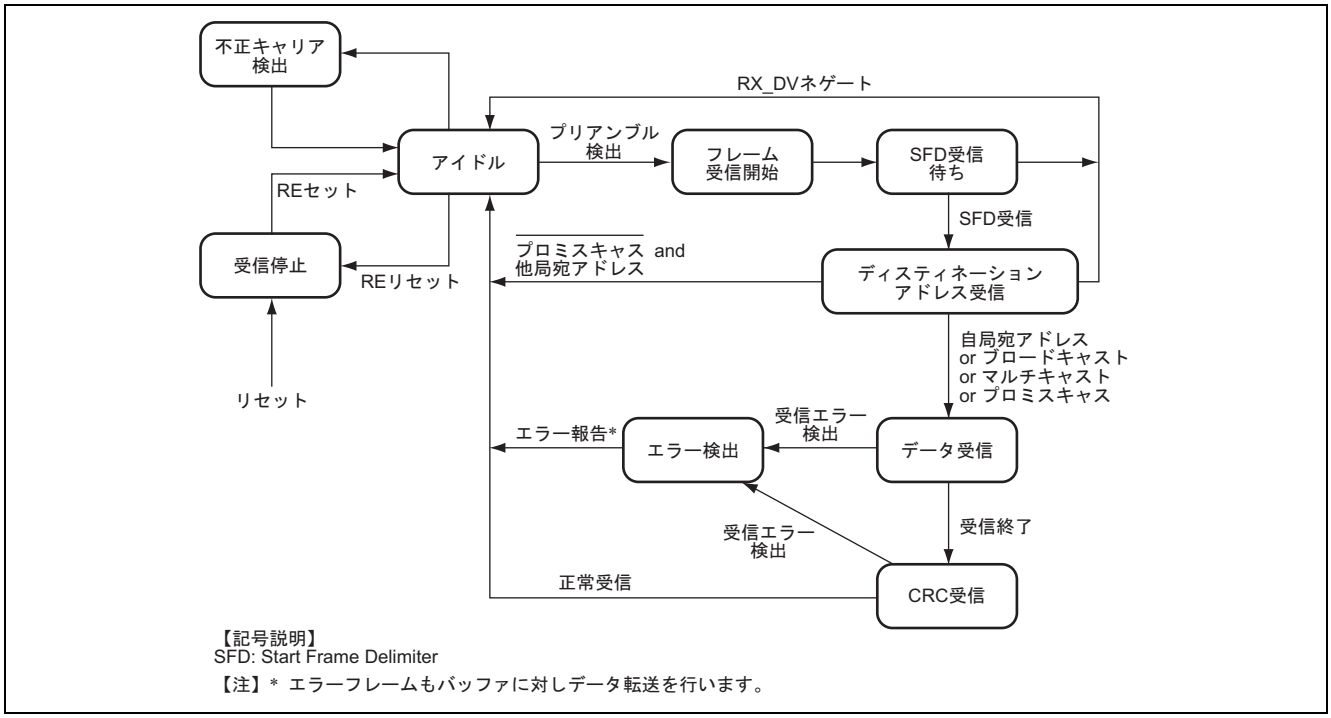


図 8　EtherC 受信部状態遷移図

2.1.4　E-DMACの概要

本 LSI は，イーサネットコントローラ (EtherC) に直結したダイレクトメモリアクセスコントローラ (E-DMAC) を内蔵しています。バッファ管理の多くの部分を E-DMAC がディスクリプタを用いて制御します。このため CPU の負荷を軽減し，効率の良いデータ送受信制御を行うことができます。CPU により直接送信/受信 FIFO のデータを読み書きすることはできません。

この DMA 転送時に，E-DMAC が参照する情報を送信/受信ディスクリプタと呼び，ユーザがメモリ上に配置します。E-DMAC は，イーサネットフレーム送受信に先立ちディスクリプタの情報を読み込み，その内容にしたがって送信データを送信バッファから読み込み，または受信データを受信バッファへ書き込みます。このディスクリプタを複数個並べディスクリプタ列化 (リスト化) することで, 複数のイーサネットフレームの送受信を連続的に行うことができます。

図 9 に E-DMAC とディスクリプタおよびバッファの構成を示します。

E-DMAC の特長は以下のようになります。

特長

- ディスクリプタ管理方式による CPU 負荷の軽減
- 送受信フレームステータスのディスクリプタへの反映
- ブロック転送 (16 バイト単位) によるシステムバスの効率使用
- シングルフレーム・マルチバッファ方式対応可能

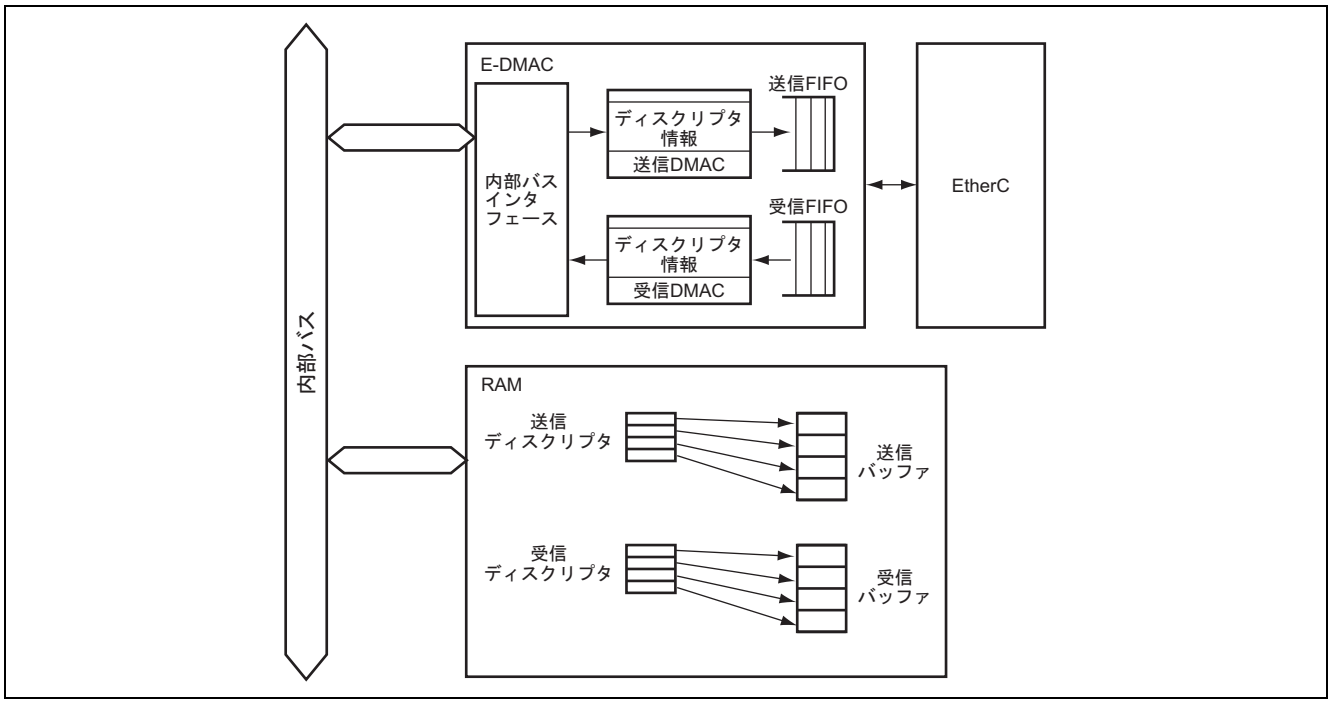


図 9　E-DMAC とディスクリプタおよびバッファの構成

2.1.5　ディスクリプタの概要

E-DMAC が DMA 転送を行うためには，ディスクリプタと呼ばれる送受信データの格納アドレス等が書かれた情報 (データ) が必要になります。ディスクリプタには送信ディスクリプタと受信ディスクリプタの 2 種類があります。E-DMAC は，E-DMAC 送信要求レジスタ (EDTRR) の TR ビットが 1 になると自動的に送信ディスクリプタの読み込みを，E-DMAC 受信要求レジスタ (EDRRR) の RR ビットが 1 になると自動的に受信ディスクリプタの読み込みを開始します。ユーザは送信/受信ディスクリプタにあらかじめ送信/受信データの DMA 転送に関する情報を記述しておく必要があります。イーサネットフレームの送信/受信が完了した後は，E-DMAC がディスクリプタの有効/無効ビット (送信時は TACT ビット，受信時は RACT ビット) を無効にし, 送信/受信結果をステータスビット (送信時は TFS26～TFS0, 受信時は RFS26～RFS0) に反映します。

ディスクリプタは，読み書き可能なメモリ空間に配置し，先頭ディスクリプタ (E-DMAC が最初に読み込むディスクリプタ) のアドレスを送信ディスクリプタリスト先頭アドレスレジスタ (TDLAR) /受信ディスクリプタリスト先頭アドレスレジスタ (RDLAR) に設定します。複数のディスクリプタをディスクリプタ列 (ディスクリプタリスト) として用意する場合には，E-DMAC モードレジスタ (EDMR) の DL0, 1 ビットに設定したディスクリプタ長にしたがって連続したアドレスに配置します。

2.1.6　送信ディスクリプタの概要

図 10 に送信ディスクリプタと送信バッファの関係を示します。

送信ディスクリプタは，データの先頭から 32 ビット単位に TD0, TD1, TD2 およびパディングで構成されます。TD0 は, 送信ディスクリプタの有効/無効, ディスクリプタの構成情報およびステータス情報を示します。TD1 はそのディスクリプタで指示する転送すべき送信バッファのデータ長を示します。TD2 は転送する送信バッファの先頭アドレスを示します。パディングは EDMR レジスタの DL0, 1 ビットで指定するディスクリプタ長に従い長さが決まります。

送信ディスクリプタの設定内容により，ディスクリプタ 1 個で 1 フレームの送信データを指定すること (1 フレーム/1 ディスクリプタ) も，ディスクリプタ複数個で 1 フレームの送信データを指定すること (1 フレーム/マルチディスクリプタ) も可能です。1 フレーム/マルチディスクリプタとしては，たとえばイーサネットフレーム中毎回の送信で固定的に使われるデータ部分を複数のディスクリプタに設定するという方法があります。具体的には，イーサネットフレーム中のあて先アドレス，送信元アドレスのデータを複数のディスクリプタで共有して，残りのデータを各々のバッファに格納するという方法が考えられます。

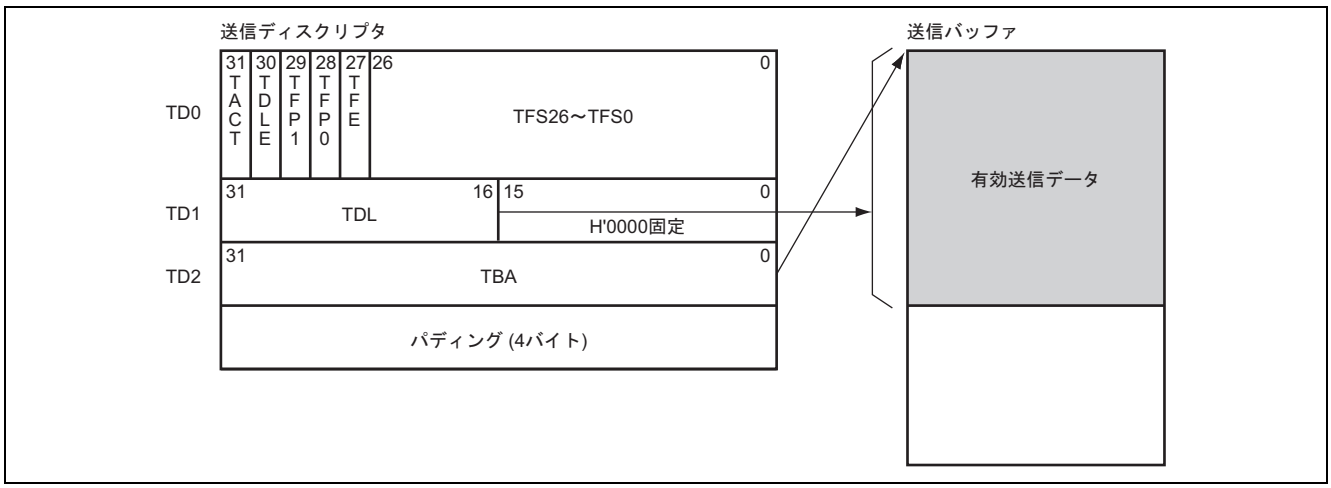


図 10　送信ディスクリプタと送信バッファの関係

2.1.7　受信ディスクリプタの概要

図 11 に受信ディスクリプタと受信バッファの関係を示します。

受信ディスクリプタは, データの先頭から 32 ビット単位に RD0, RD1, RD2 およびパディングで構成されます。RD0 は, 受信ディスクリプタの有効/無効, ディスクリプタの構成情報およびステータス情報を示します。RD1 はそのディスクリプタが参照する受信バッファのサイズ (RBL) と受信したフレームのデータ長 (RDL) を示します。RD2 は受信バッファの先頭アドレスを示します。最後のパディングは EDMR レジスタの DL0, 1 ビットで指定するディスクリプタ長に従い長さが決まります。

受信ディスクリプタの設定内容により, ディスクリプタ 1 個で 1 フレームの受信データ全部を受信バッファに格納すること (1 フレーム/1 ディスクリプタ) も, ディスクリプタ複数個で 1 フレームの受信データを受信バッファに格納すること (1 フレーム/マルチディスクリプタ) も可能です。1 フレーム/マルチディスクリプタでは, あらかじめ複数のディスクリプタ (ディスクリプタリスト) を用意しておきます。E-DMAC は, 受信したフレームがディスクリプタの RBL を超える長さのフレームを受信した場合には, 連続する次のディスクリプタを使用していくことによって受信バッファに転送していきます。たとえば各ディスクリプタの RBL を 500 バイトとしたときに 1514 バイトのイーサネットフレームを受信したとします。受信したイーサネットフレームは最初のディスクリプタから順に 500 バイトずつバッファに転送され, 最後の 14 バイトだけが 4 つ目のバッファに転送されます。

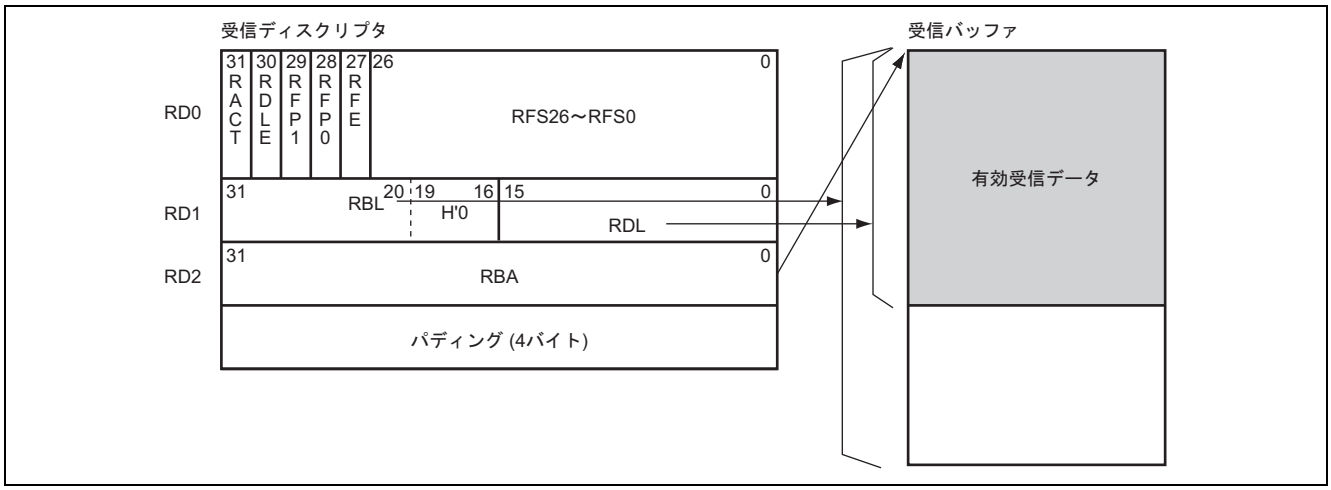


図 11　受信ディスクリプタと受信バッファの関係

2.1.8　送信ディスクリプタの設定例

図12に送信ディスクリプタおよび送信バッファを3面使用した場合の例 (1フレーム/1ディスクリプタ) を示します。ここでは1回の送信要求で1フレームだけ送信するものとします。図では各送信ディスクリプタを TD0 部分のみに簡略して記載しています。図中の番号①，②等は実行順を示します。

設定は以下のようになります。

1. 1フレーム/1ディスクリプタ方式のため, 全ディスクリプタ面の TFP1, TFP0 ビットに B'11 を設定します。
2. 全ディスクリプタ面の TACT ビット，TFE ビット，TFS26～TFS0 ビットには初期値としてすべて 0 を設定します。
3. 第1面と第2面のディスクリプタの TDLE ビットに0を設定します。第3面のディスクリプタの TDLE ビットに1を設定することにより，第3面の処理を終了すると第1面のディスクリプタを読み込みます。このような設定によりディスクリプタをリング構造にすることができます。
4. 図 12 では省略していますが，当該ディスクリプタが参照している送信バッファのデータ長を TDL に，送信バッファの先頭アドレスを TBA に設定します。
5. この例では1回の送信要求で1フレームだけ送信するため，最初の送信では第1面のディスクリプタの TACT ビットにだけ1を設定します。次の送信では第2面のディスクリプタの TACT ビットにだけ1を設定します。

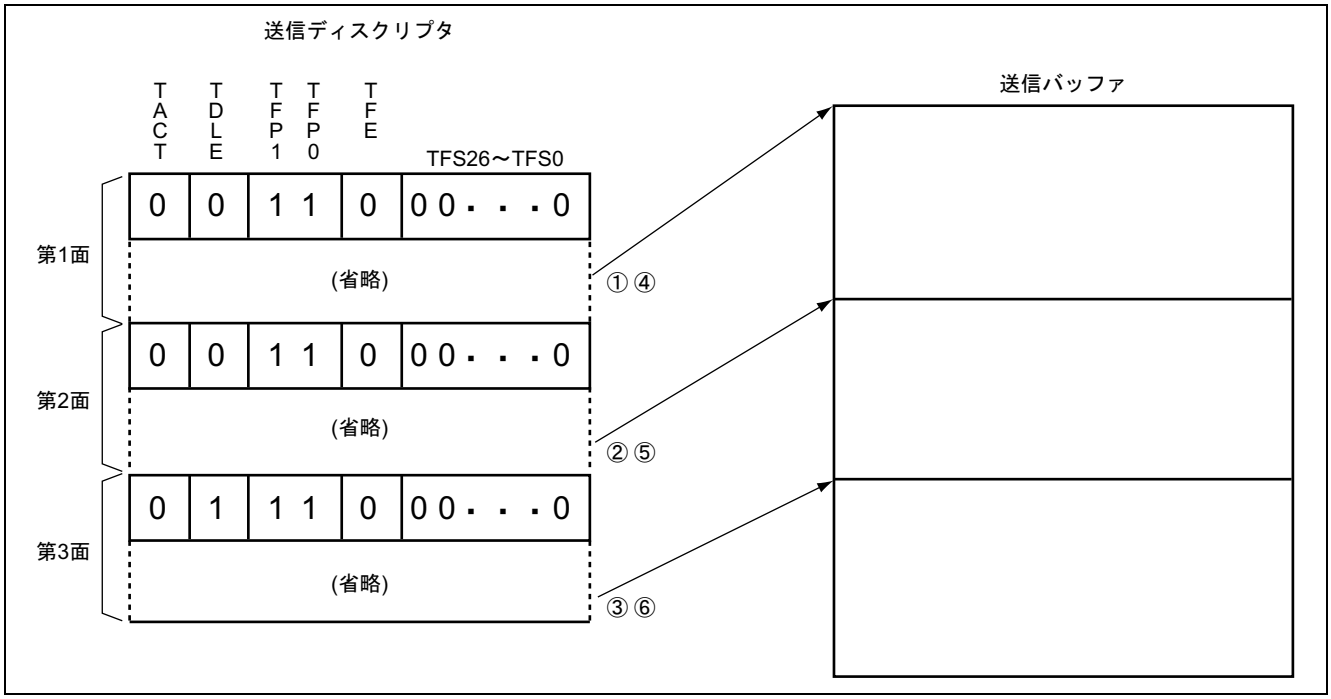


図 12　送信ディスクリプタと送信バッファの関係

2.1.9　受信ディスクリプタの設定例

図 13 に受信ディスクリプタおよび受信バッファを各 3 面使用した場合の例を示します。各受信バッファのサイズを 1520 バイト確保し，1 フレーム/1 ディスクリプタになるようにします。図では各受信ディスクリプタを RD0 部分のみに簡略化して記載しています。図中の番号①，②等は実行順を示します。

設定は以下のようになります。

1. 全ディスクリプタ面の RFP1, RFP0 ビット，RFE ビット，RFS26～RFS0 ビットに 0 を設定します。
2. 第 1 面と第 2 面のディスクリプタの RDLE ビットに 0 を設定します。第 3 面のディスクリプタの RDLE ビットに 1 を設定することにより, 第 3 面のディスクリプタの処理を終了すると第 1 面のディスクリプタを読み込みます。このような設定によりディスクリプタをリング構造にすることができます。
3. 図 13 では省略していますが, 受信開始前に全ディスクリプタ面の RD1 の RBL に受信バッファサイズ 1520 バイトを，RD2 の RBA に対応する受信バッファの先頭アドレスを設定します。
4. 連続受信をさせるため，全ディスクリプタ面の RACT ビットに 1 を設定します。

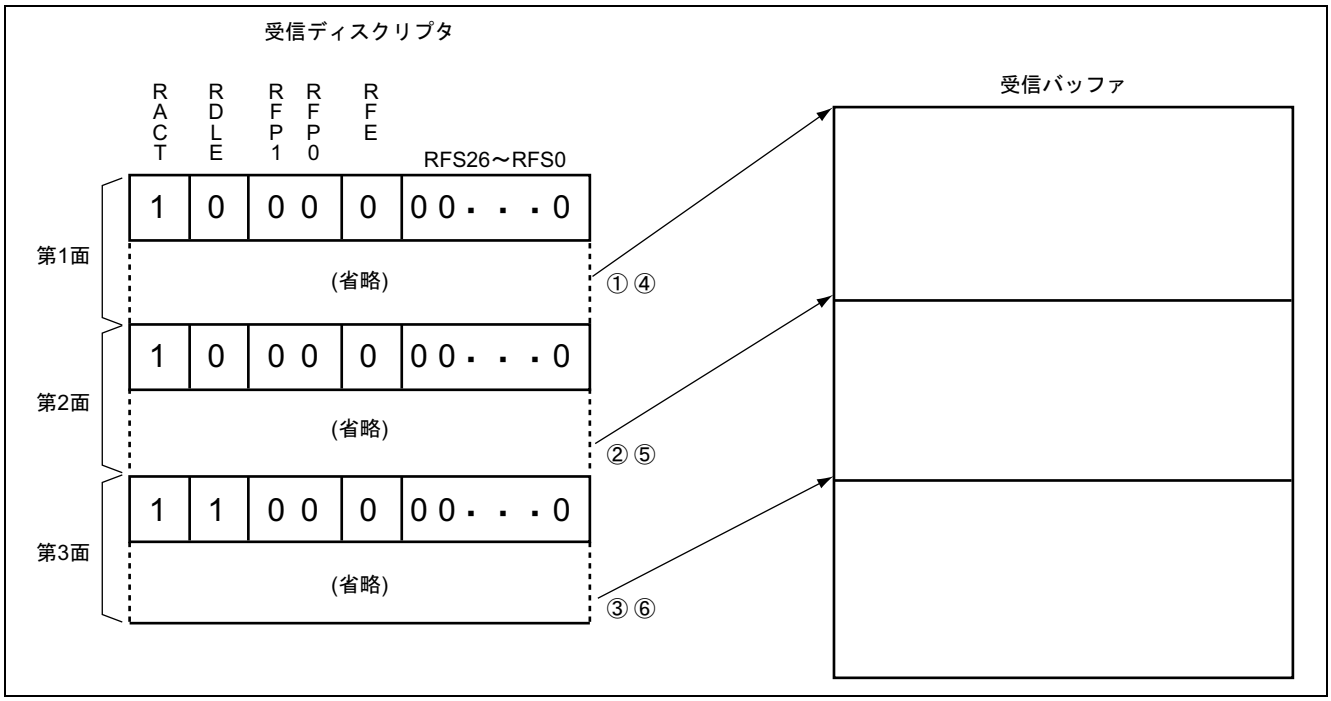


図 13　受信ディスクリプタと受信バッファの関係

2.1.10　使用機能の動作手順 (送信時)

EtherC モードレジスタ (ECMR) の TE ビットが 1 の状態で E-DMAC 送信要求レジスタ (EDTRR) の送信要求ビット (TR) に 1 を書き込むと E-DMAC 送信部が起動します。E-DMAC は EtherC/E-DMAC のソフトウェアリセット後，送信ディスクリプタリスト先頭アドレスレジスタ (TDLAR) で示すディスクリプタを読み込みます。読み込んだディスクリプタの TACT ビットが 1 (有効) の場合は，E-DMAC は送信ディスクリプタの TD2 で指定される送信バッファ先頭アドレスから順次送信フレームデータを読み出して EtherC に転送します。EtherC は送信フレームを作成し RMII に向けて送信を開始します。ディスクリプタ内で指示されるバッファ長分の DMA 転送後，送信ディスクリプタの TFP の値によって以下の処理を行います。

- TFP = B'00 or B'10 (フレーム継続)
  DMA 転送後, ディスクリプタのライトバック (TACT ビットの 0 書き込み) を行います。その後, 次のディスクリプタの TACT ビットを読み込みます。
- TFP = B'01 or B'11 (フレーム終了)
  フレームの送信完了後, ディスクリプタのライトバック (TACT ビットの 0 およびステータスの書き込み) を行います。その後，次のディスクリプタの TACT ビットを読み込みます。

読み込んだディスクリプタの TACT ビットが 1 のときは，フレームの送信を継続し次のディスクリプタを読み込みます。TACT ビットが 0 (無効) のディスクリプタを読み込むと，E-DMAC は EDTRR の TR ビットを 0 にして送信処理を完了します。TR ビットが 0 になった後 TR ビットに 1 を書き込むと再度 E-DMAC 送信部が起動しますが, この場合は最後に送信を行ったディスクリプタの次のディスクリプタを読み込みます。

図 14 に送信フローの例 (1 フレーム/1 ディスクリプタ，複数ディスクリプタ面の場合) を示します。

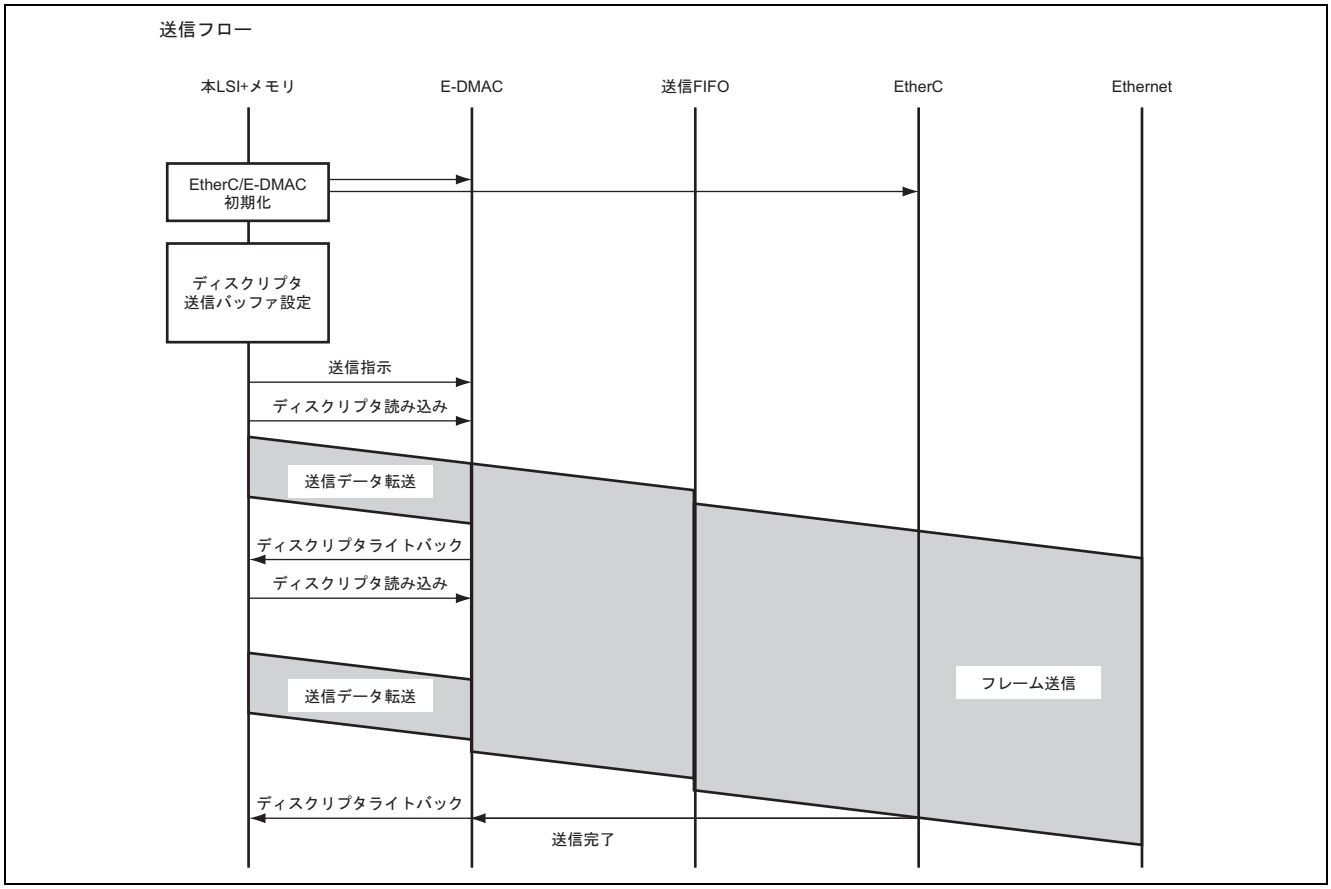


図 14　送信フローの例

2.1.11　使用機能の動作手順 (受信時)

ECMR の RE ビットが 1 の状態で E-DMAC 受信要求レジスタ (EDRRR) の受信要求ビット (RR) に 1 を書き込むと, E-DMAC 受信部が起動します。E-DMAC は, EtherC/E-DMAC のソフトウェアリセット後は受信ディスクリプタ先頭アドレスレジスタ (RDLAR) で示すディスクリプタを読み込み，RACT ビットが 1 (有効) のときに受信待機状態になります。EtherC は自局宛 (自局が受信を許可したアドレス) のフレームを受信すると，受信データを受信 FIFO に格納します。受信ディスクリプタの RACT ビットが 1 のときは，RD2 で指定される受信バッファに転送します (RACT ビットが 0 (無効) の場合は，RR ビットをクリアして E-DMAC の受信動作を停止します)。受信したフレームのデータ長が RD1 で与えられるバッファ長よりも大きい場合は，E-DMAC はバッファが満了となった時点でディスクリプタにライトバック (RFP = B'10 or B'00) を行い次のディスクリプタを読み込みます。フレームの受信が完了した場合，または何らかのエラーでフレーム受信を中断した場合は，当該ディスクリプタにライトバック (RFP = B'11 or B'01) を行います。その後，連続受信方式を選択している場合 (受信方式制御レジスタ (RMCR) 内の受信コントロールビット (RNC) が 1 の場合), E-DMAC は次のディスクリプタを読み込み RACT ビットが 1 のときに受信待機状態になります。連続受信方式を選択していない場合 (RMCR レジスタ内の RNC ビットが 0 の場合) は，EDRRR レジスタの RR ビットを 0 にし E-DMAC は受信処理を終了します。そして再度 RR ビットを 1 に設定すると，E-DMAC は最後に受信を行ったディスクリプタの次のディスクリプタを読み込み受信待機状態になります。

図 15 に受信フローの例 (1 フレーム/1 ディスクリプタ，連続受信方式設定時の場合) を示します。

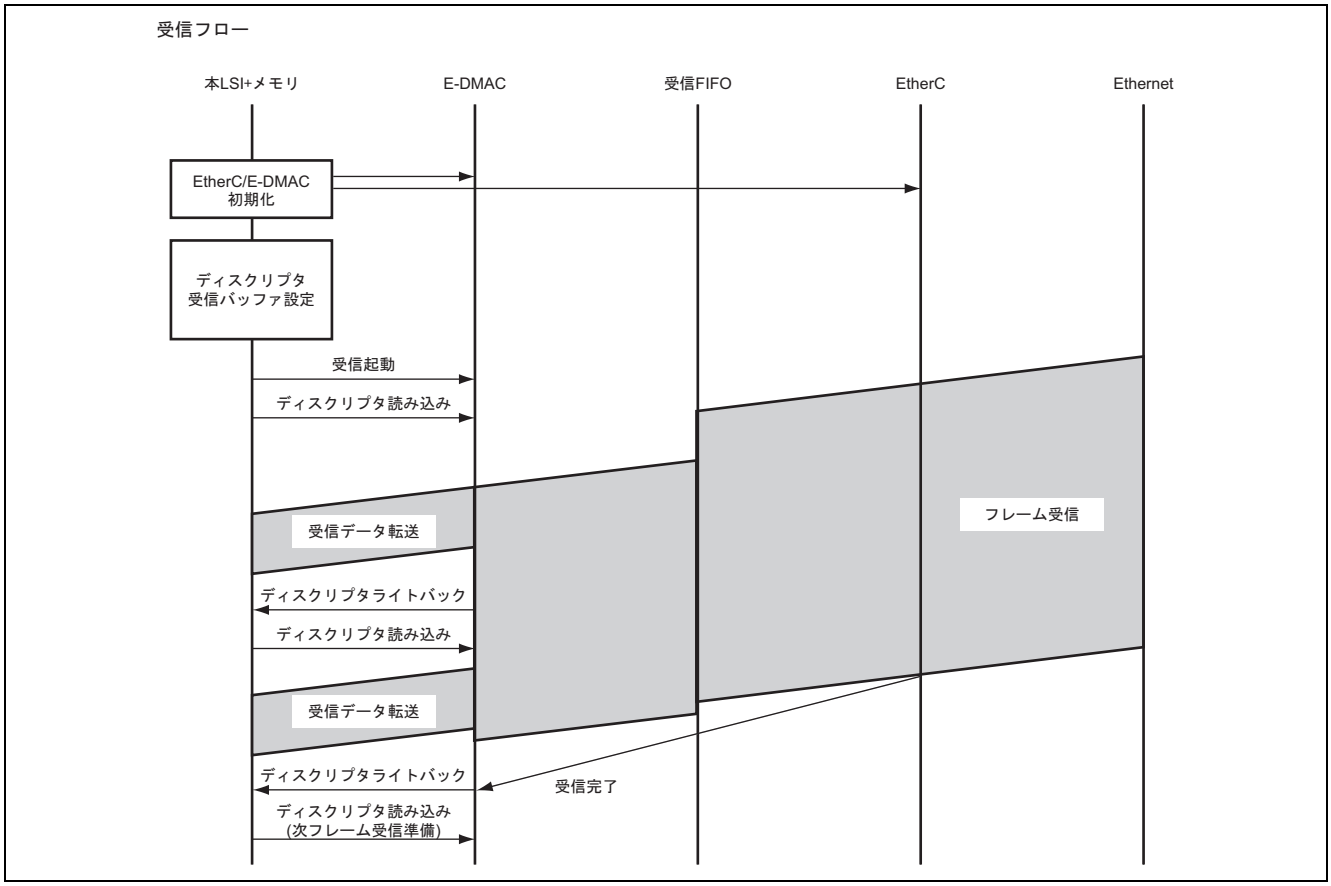


図 15　受信フローの例

2.1.12　使用機能の設定手順 (送受信時)

ここでは，イーサネット送受信するための基本的な設定例について説明します。図 16，図 17 にイーサネット設定フロー例を示します。

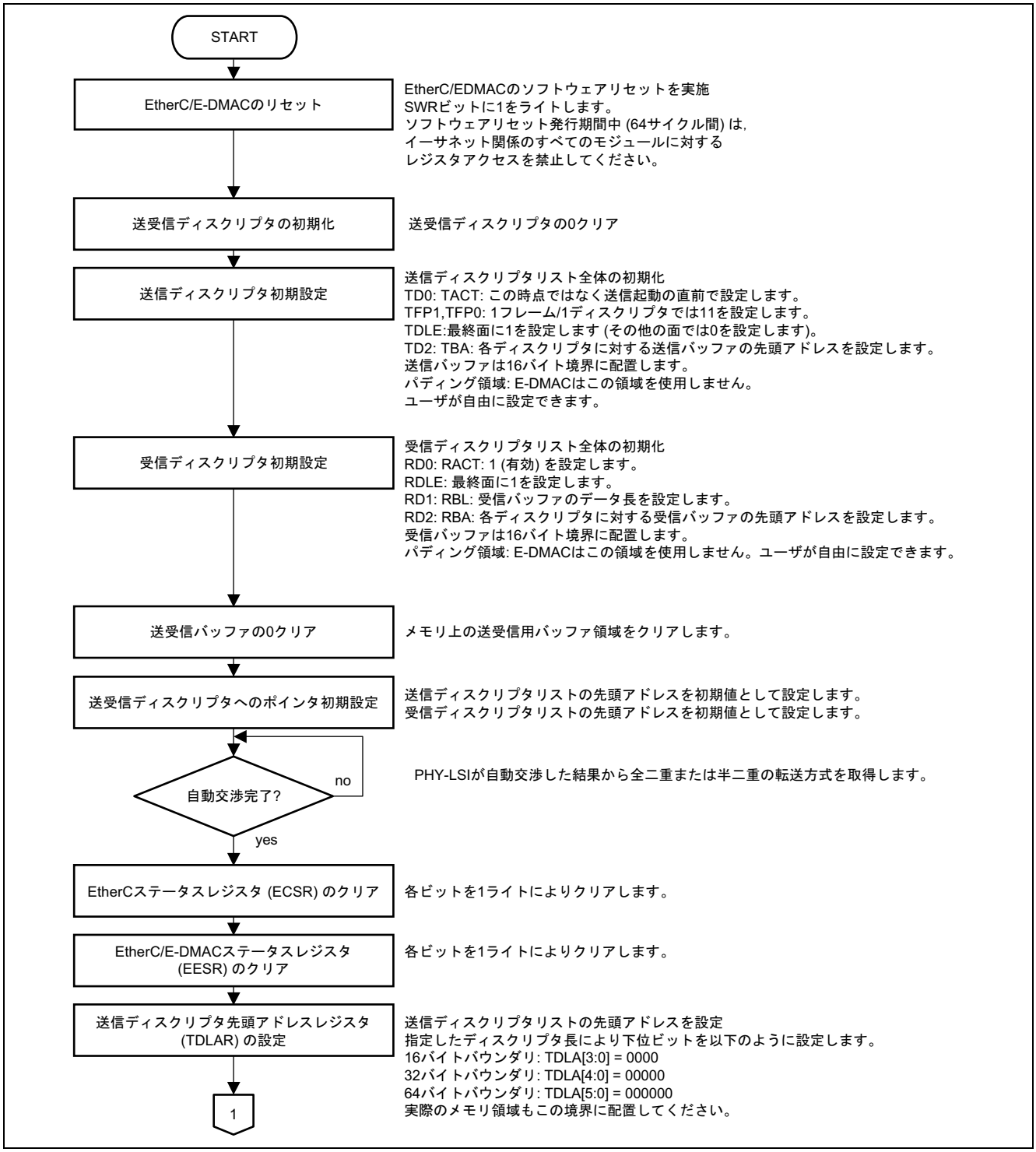


図 16　イーサネット設定フロー例 (1)

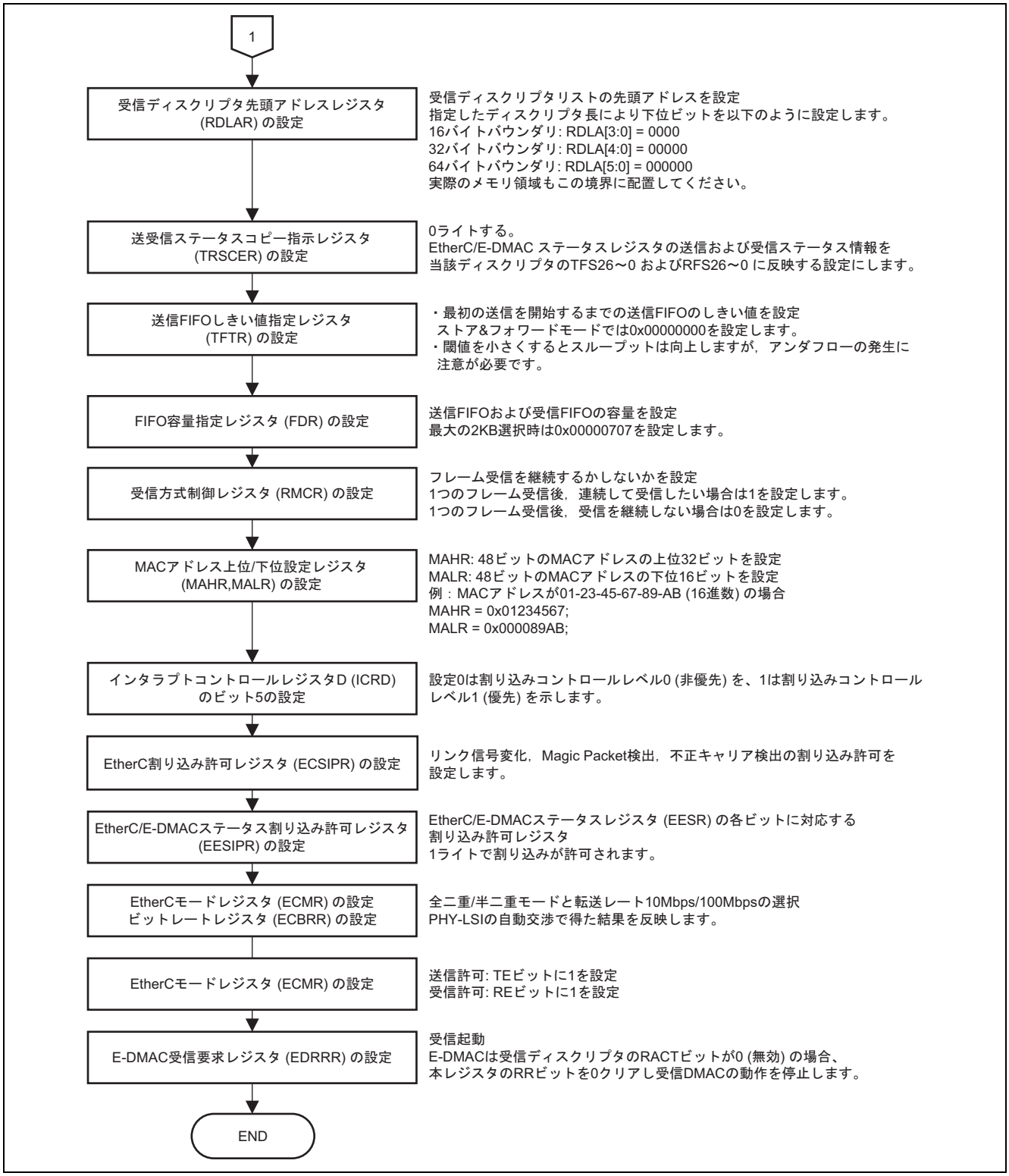


図 17　イーサネット設定フロー例 (2)

2.2　参考プログラムの動作 (送信時)

参考プログラムでは，EtherC および E-DMAC を使用し，対向ホストに向けて 10 フレームを送信します。送信ディスクリプタと 1520 バイトの送信バッファを 4 面用意 (1 フレーム/1 ディスクリプタ) しています。送信ディスクリプタをリング状にして使用しています。フレーム送信完了割り込み (TCIP) により 1 フレームの送信が完了したと判断し，次の送信を開始します。

送信データについては，イーサネットフレームのうちプリアンブル，スタートフレームデリミタ (SFD), および CRC 部を除いた部分を用意する必要があります。ヘッダ部の宛先 MAC アドレスおよび送信元 MAC アドレスは，ご使用になる製品の MAC アドレスに変更していただく必要があります。なお，EtherC は送信元 MAC アドレスのチェックは行いません。

2.3　参考プログラムの動作 (受信時)

参考プログラムでは, EtherC および E-DMAC を使用し, 対向ホストからイーサネットフレームを 10 フレーム受信します。受信ディスクリプタと 1520 バイトの受信バッファを 4 面用意しています。受信方式制御レジスタ (RMCR) 内の受信コントロールビット (RNC) に 1 を設定し，連続受信方式にしています。受信関数がコールされると、受信ディスクリプタの RFE ビット (RD0 のビット 27) をチェックし，エラーがなければ (RFE = 0 の場合) 受信バッファにある 1 フレーム分のデータをユーザバッファにコピーします。その後当該ディスクリプタを初期化し次回の受信に備えます。エラーがあれば (RFE = 1 の場合)，ユーザバッファへのコピーは行わず当該ディスクリプタを初期化するだけにします。

なお，受信バッファにはイーサネットフレームのうちプリアンブル，SFD，および CRC を除いた部分が転送されます。

2.4　参考プログラムの動作 (送受信時)

参考プログラムでは，EtherC および E-DMAC を使用し，対向ホストに向けてイーサネットフレームを 1 フレーム送信，対向ホストからイーサネットフレームを 1 フレーム受信する送受信を 2 回行います。送信ディスクリプタと 1520 バイトの送信バッファと受信ディスクリプタと 1520 バイトの受信バッファを 4 面用意しています。送信の動作は「2.2　参考プログラムの動作 (送信時)」，受信の動作は「2.3　参考プログラムの動作 (受信時)」と同様になります。

2.5　参考プログラムの動作環境

図 18 に参考プログラムの動作環境を示します。

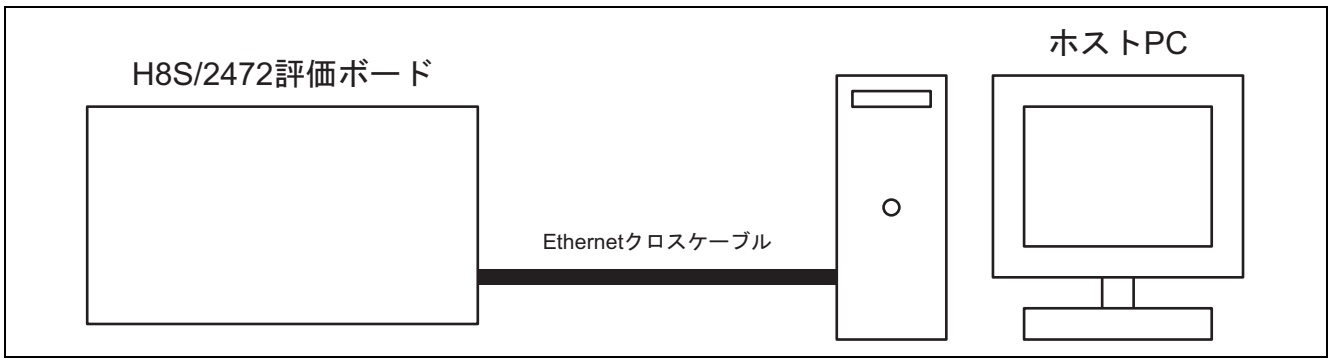


図 18　プログラムの動作環境

2.6　イーサネットフレームフォーマット

図 19 にイーサネットフレームフォーマットを示します。

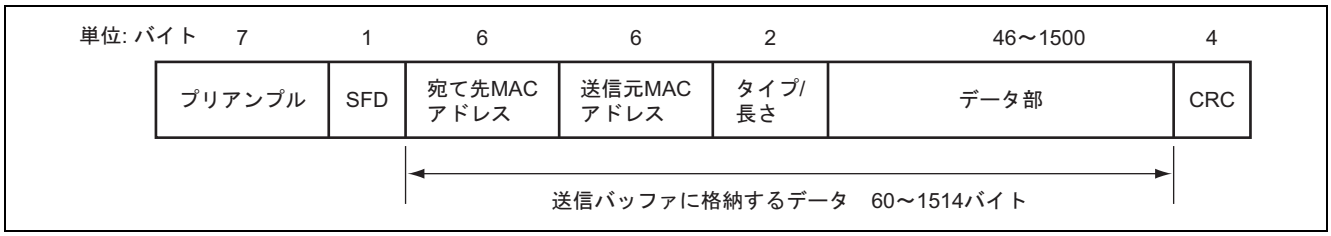


図 19　イーサネットフレームフォーマット

2.7　参考プログラムのディスクリプタ定義

E-DMAC ではディスクリプタのパディング領域を使用しません。ユーザが自由に使用できます。本プログラムではこの領域に次のディスクリプタの先頭アドレスを設定し，ソフトウェアにてもリング構造を実現しています。

図 20 に参考プログラムでの送信ディスクリプタ構造体の定義と送信ディスクリプタ列の使用例，図 21 に参考プログラムでの受信ディスクリプタ構造体の定義と受信ディスクリプタ列の使用例を示します。

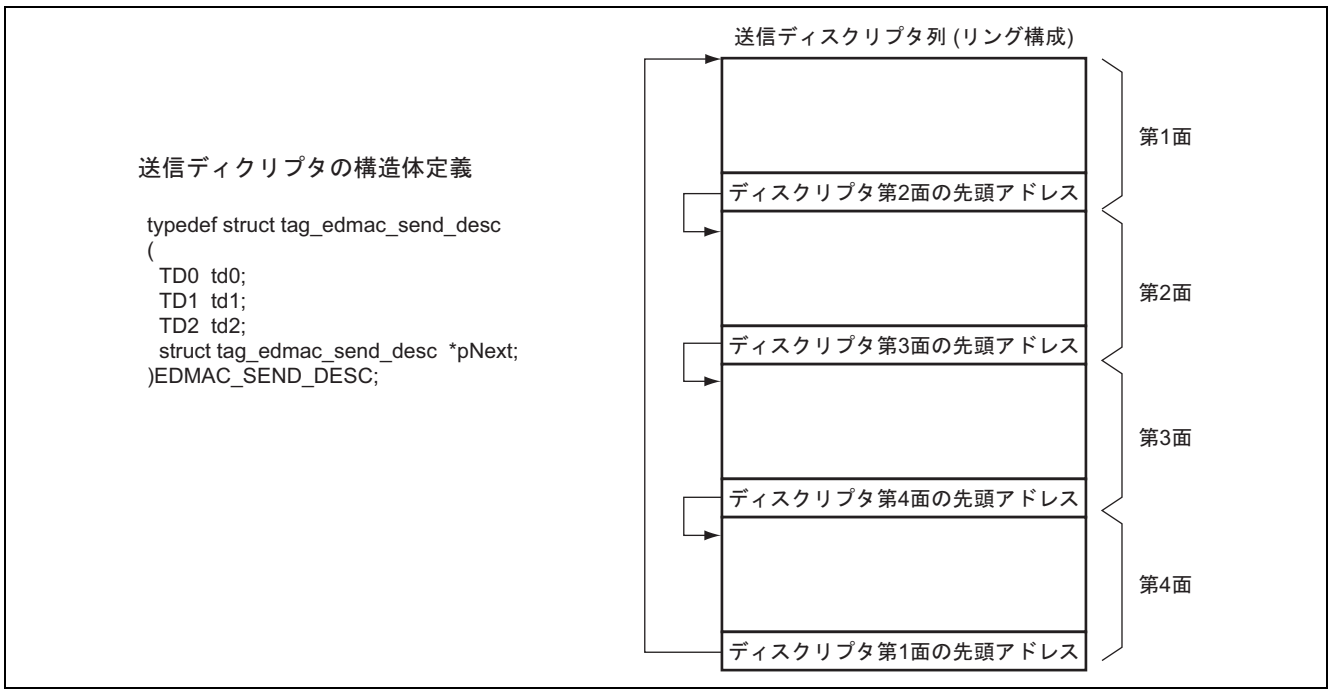


図 20　送信ディスクリプタ構造体の定義と送信ディスクリプタ列の使用例

受信ディクリプタの構造体定義

```
typedef struct tag_edmac_recv_desc
{
    RD0    rd0;
    RD1    rd1;
    RD2    rd2;
    struct  tag_edmac_recv_desc  *pNext;
}EDMAC_RECV_DESC;
```

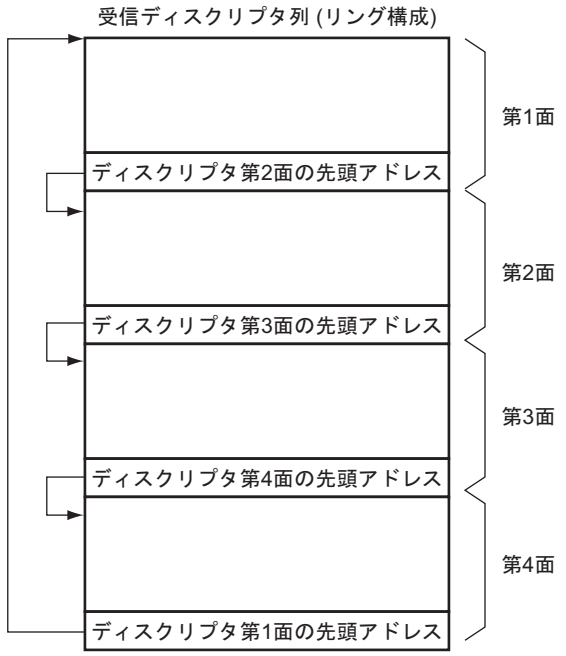


図 21　受信ディスクリプタ構造体の定義と受信ディスクリプタ列の使用例

2.8　参考プログラムの処理手順

図 22～図 31 に参考プログラムの処理フローを示します。

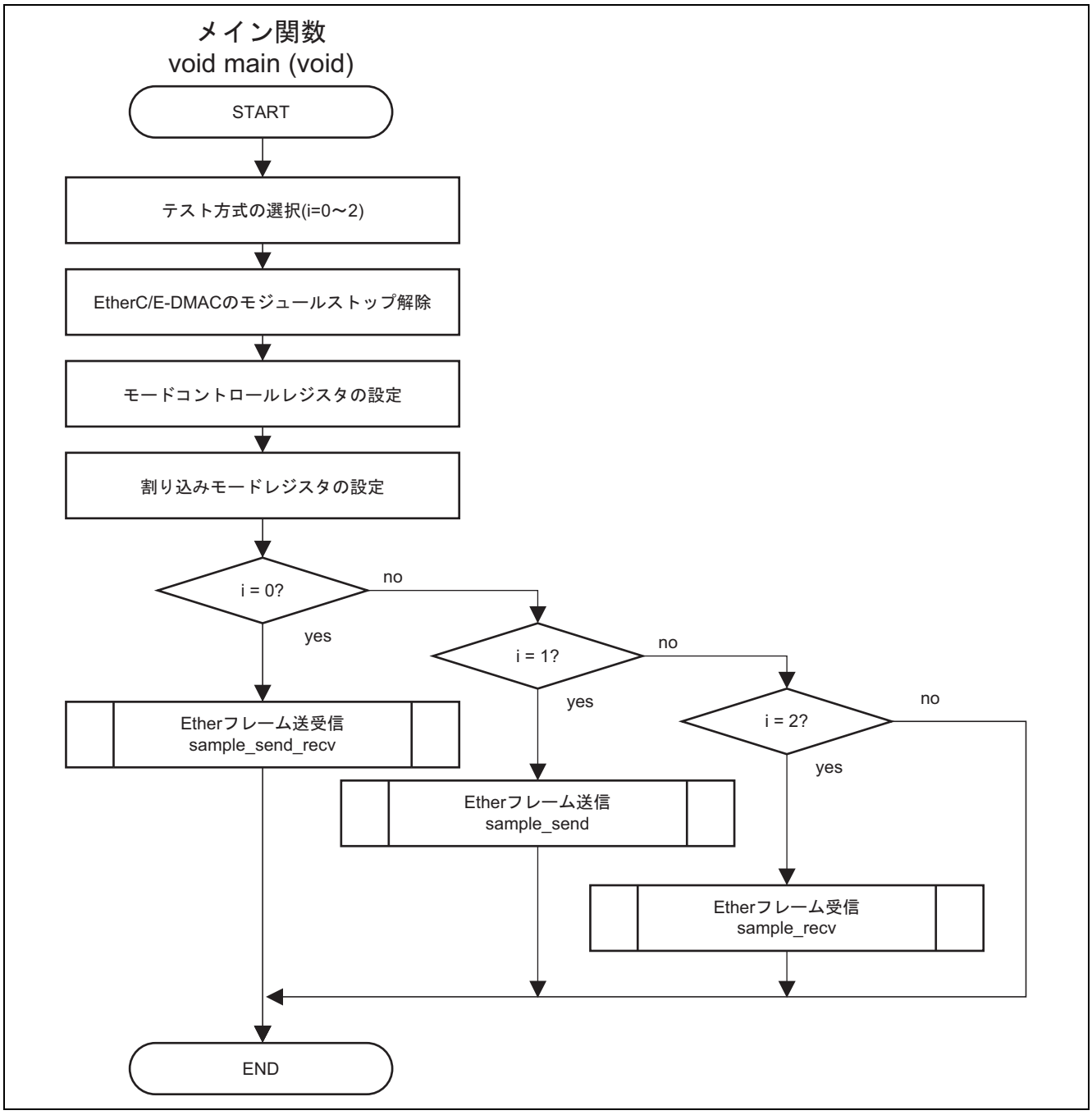


図 22　参考プログラムの処理フロー例 (1)

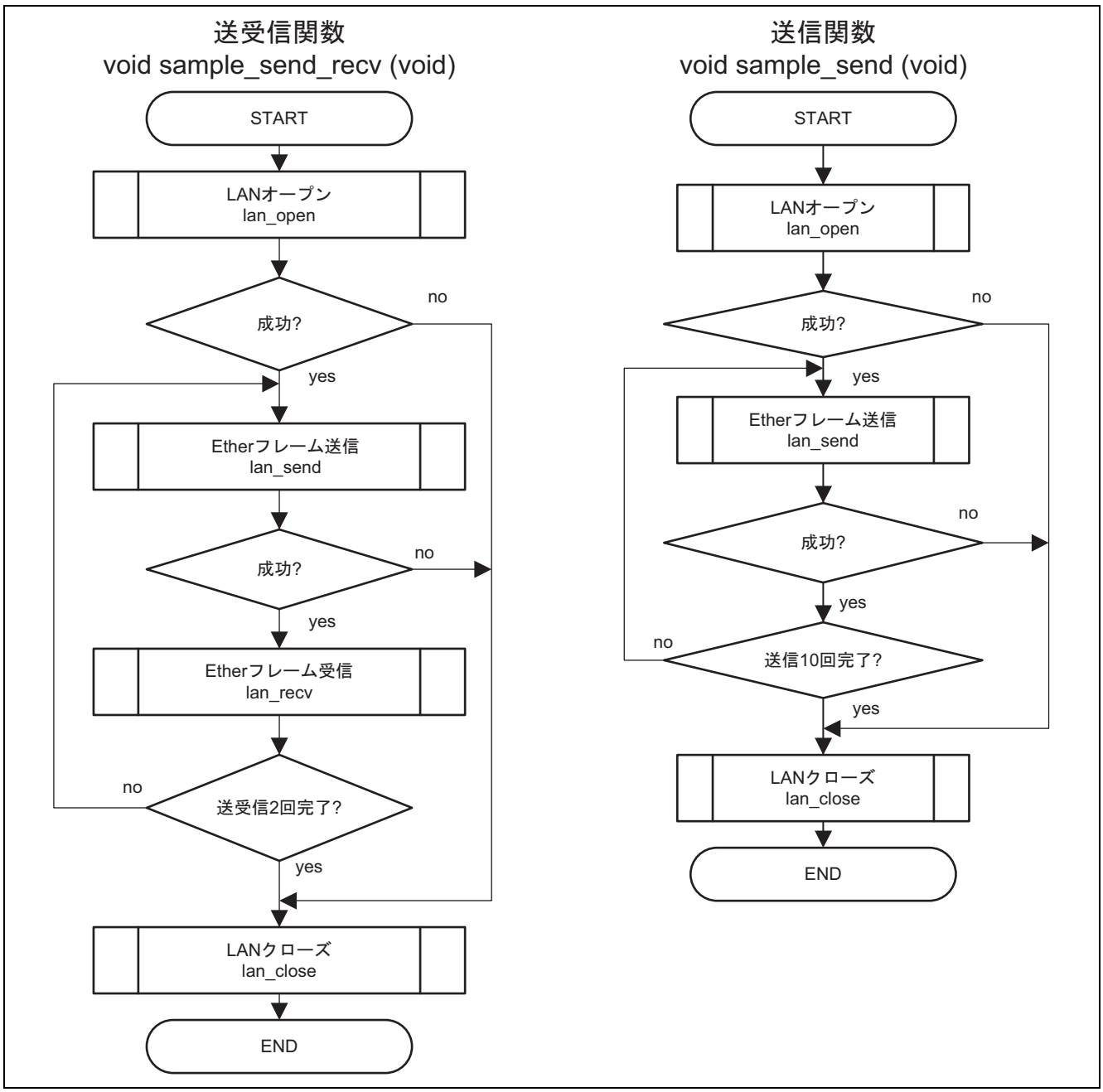


図 23　参考プログラムの処理フロー例 (2)

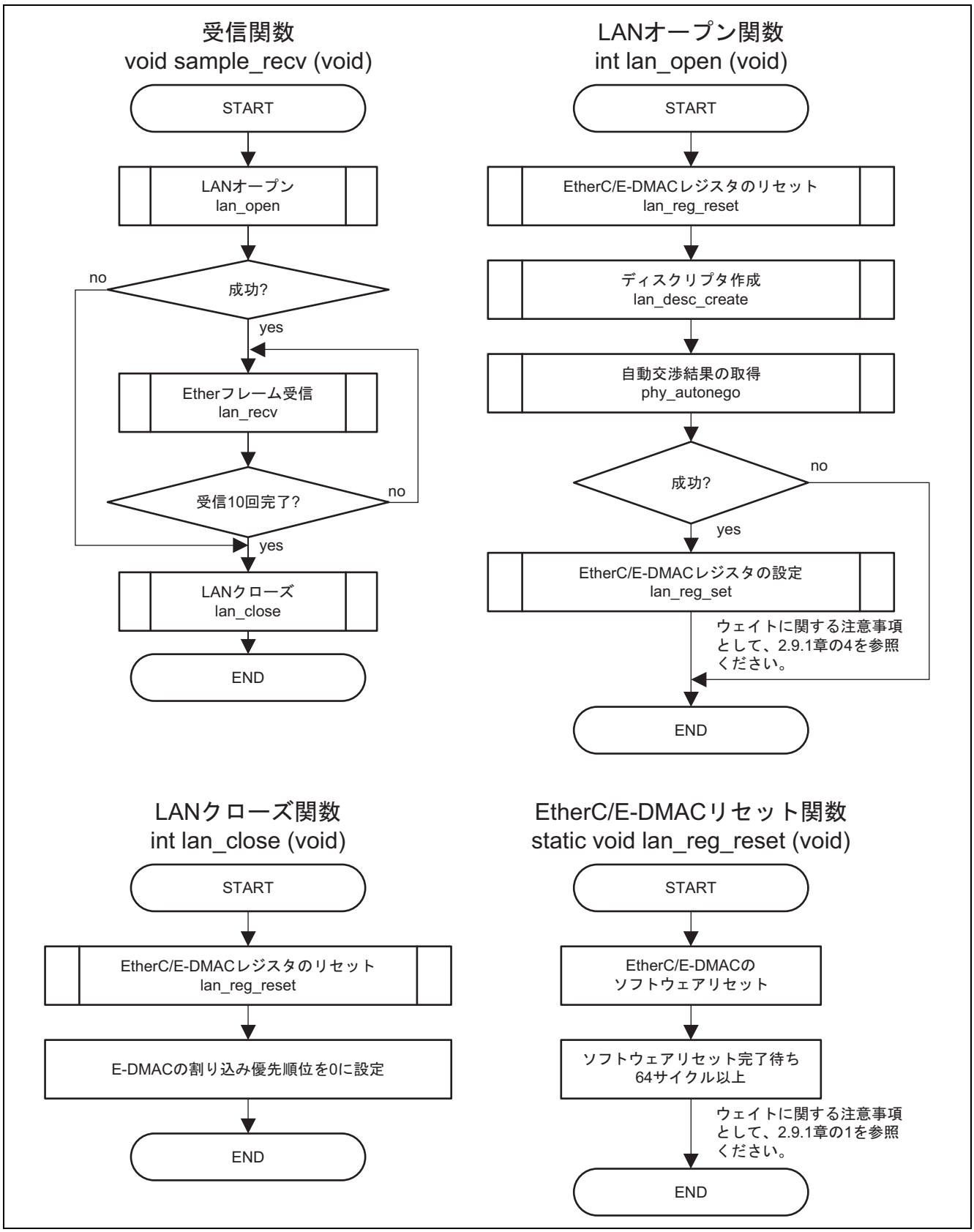


図 24　参考プログラムの処理フロー例 (3)

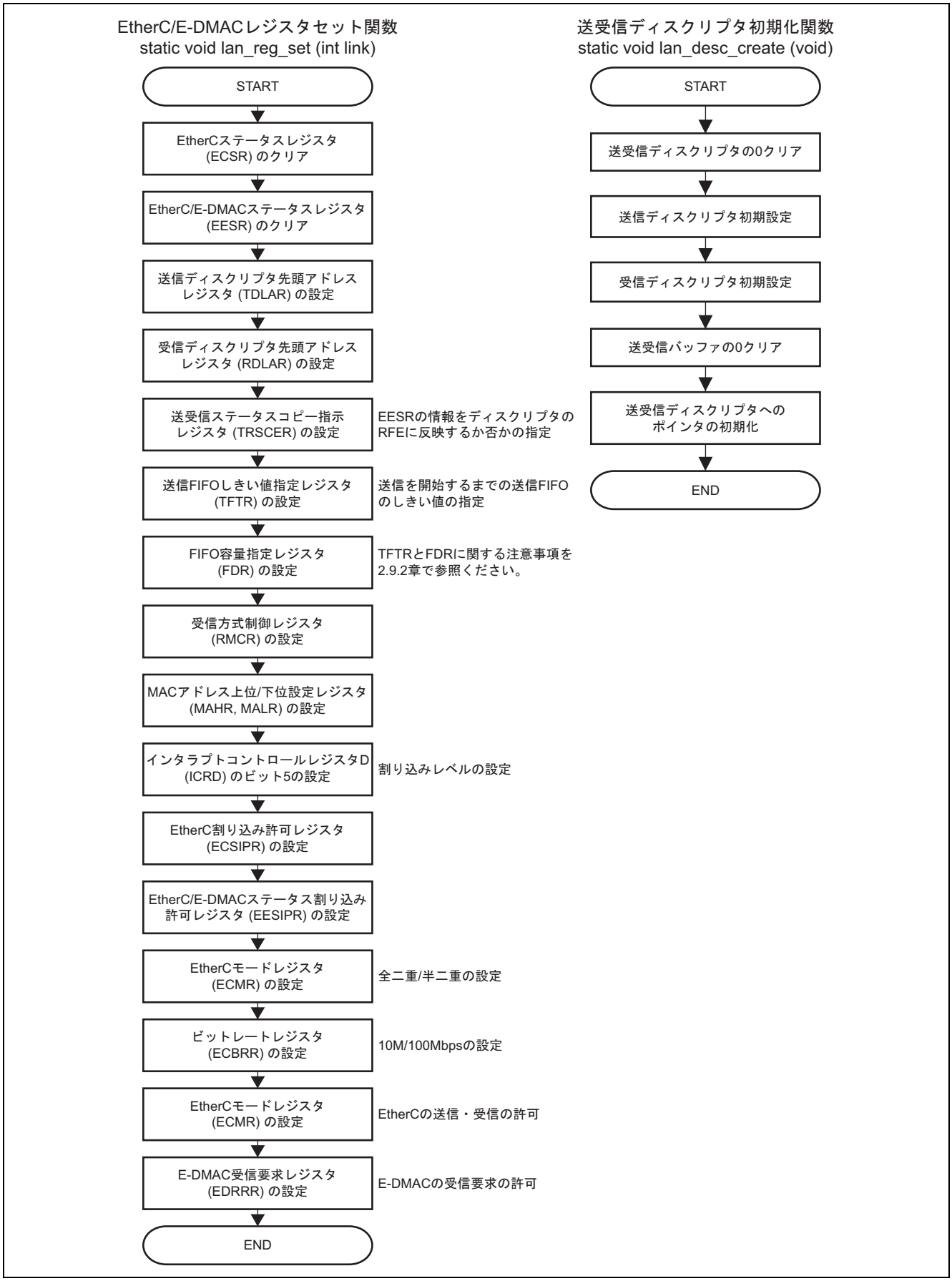


図 25　参考プログラムの処理フロー例 (4)

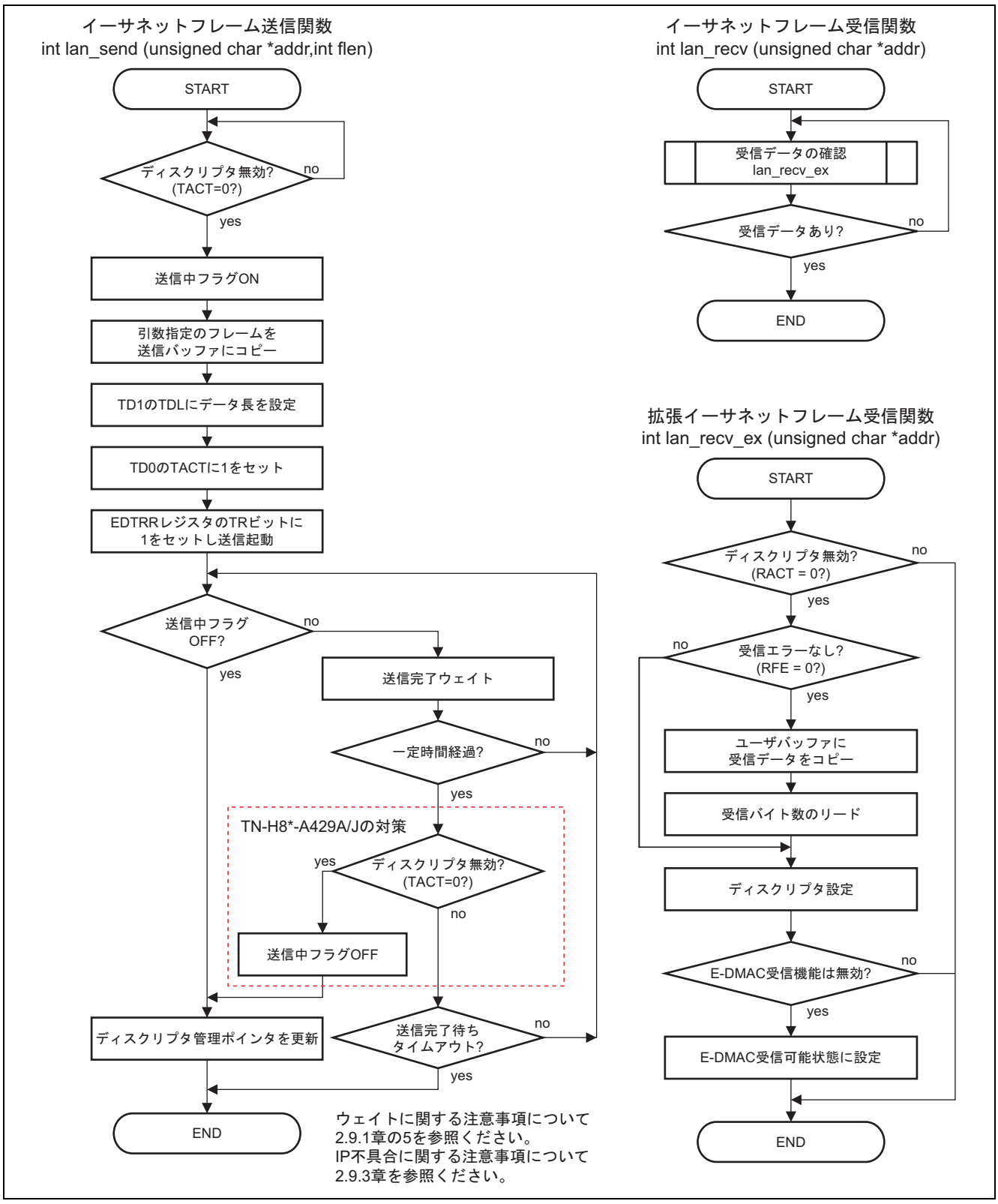


図 26　参考プログラムの処理フロー例 (5)

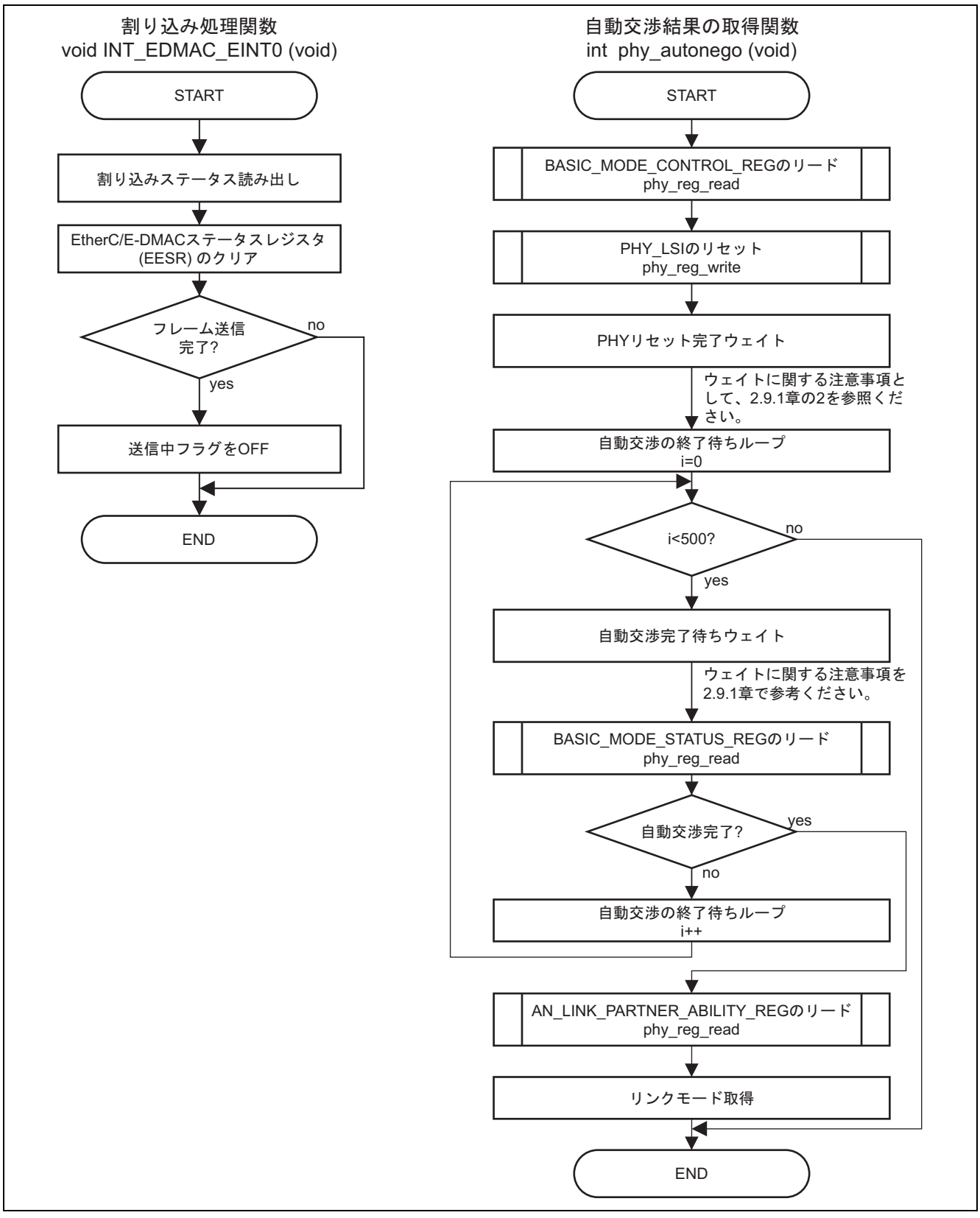


図 27　参考プログラムの処理フロー例 (6)

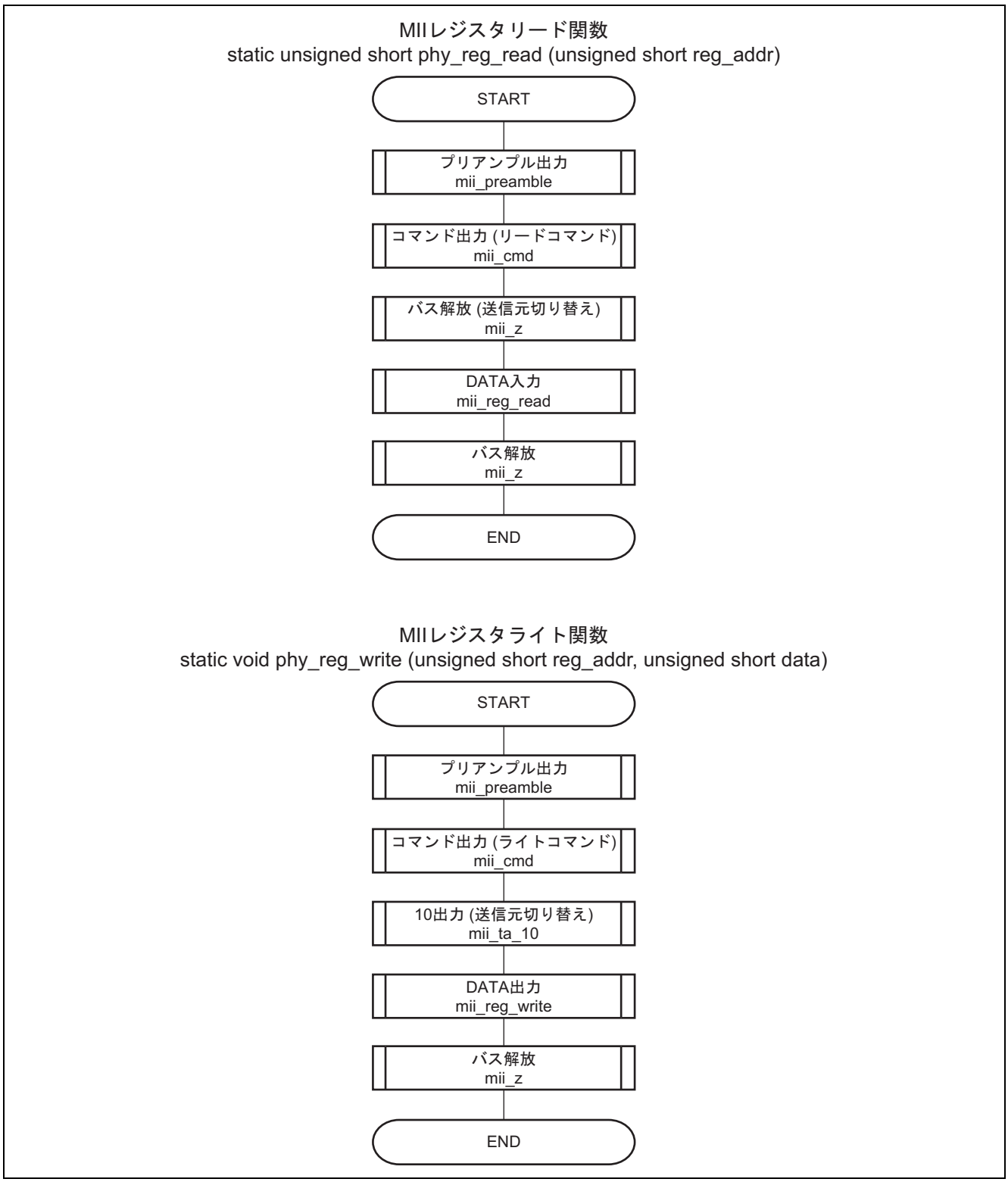


図 28　参考プログラムの処理フロー例 (7)

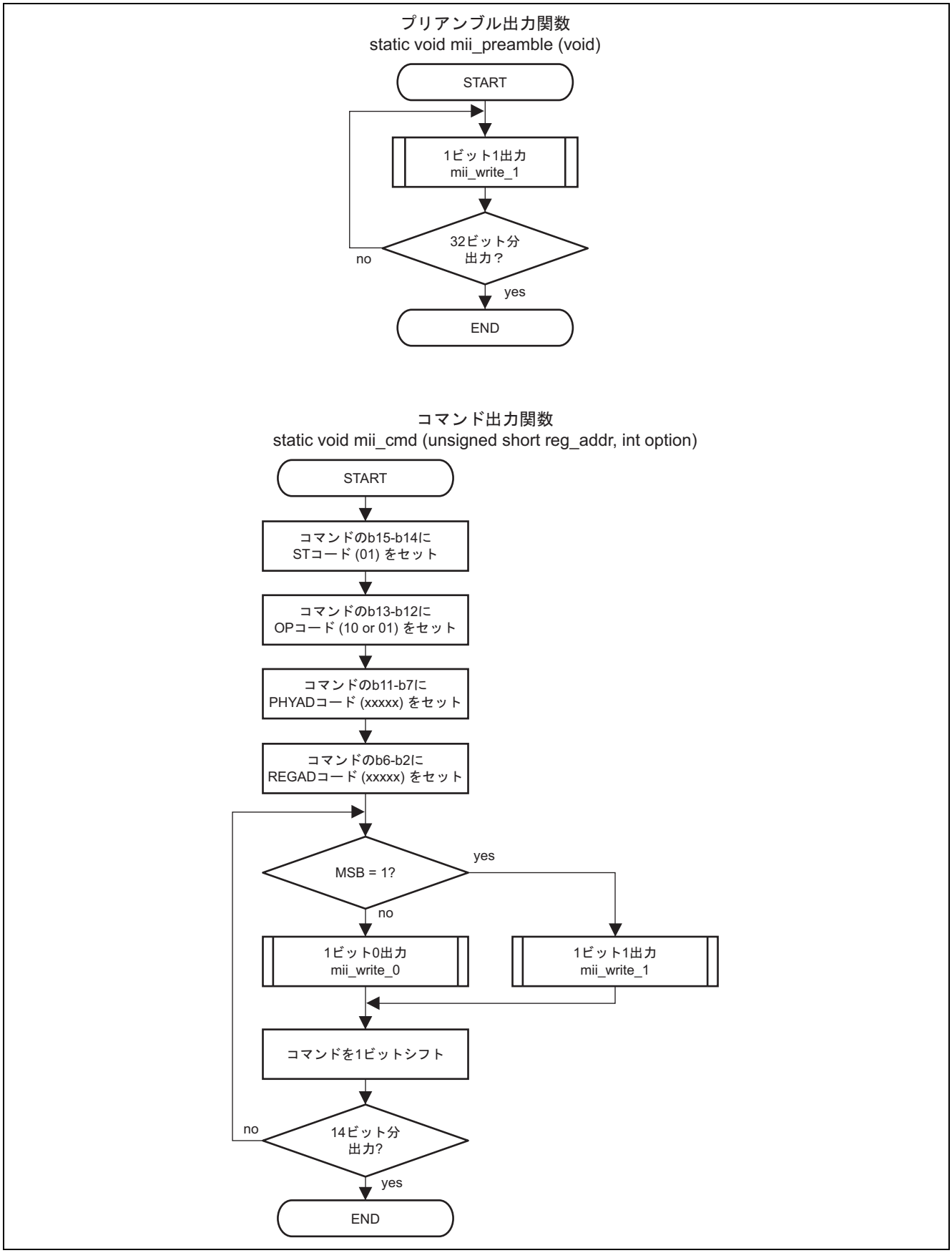


図 29　参考プログラムの処理フロー例 (8)

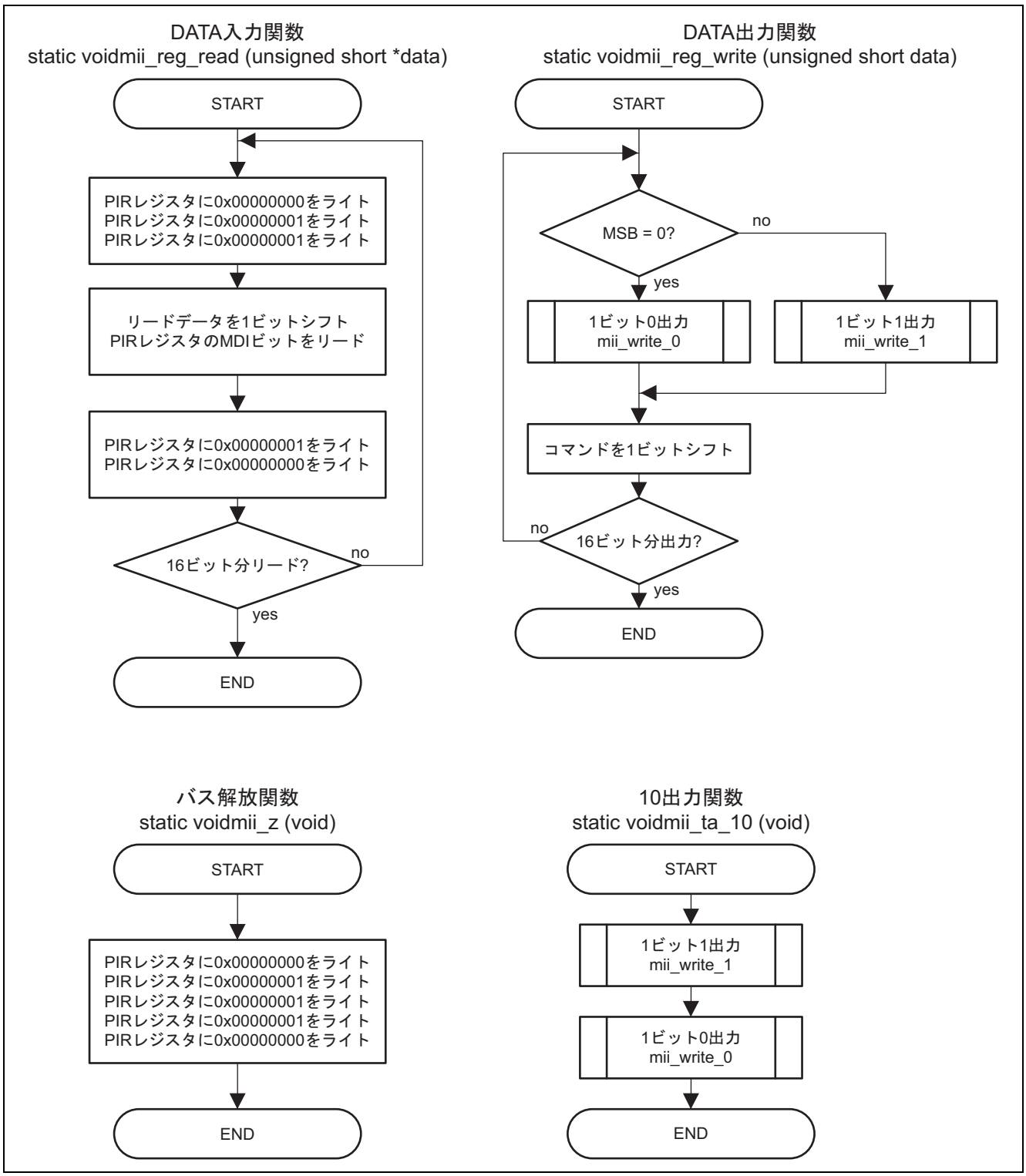


図 30　参考プログラムの処理フロー例 (9)

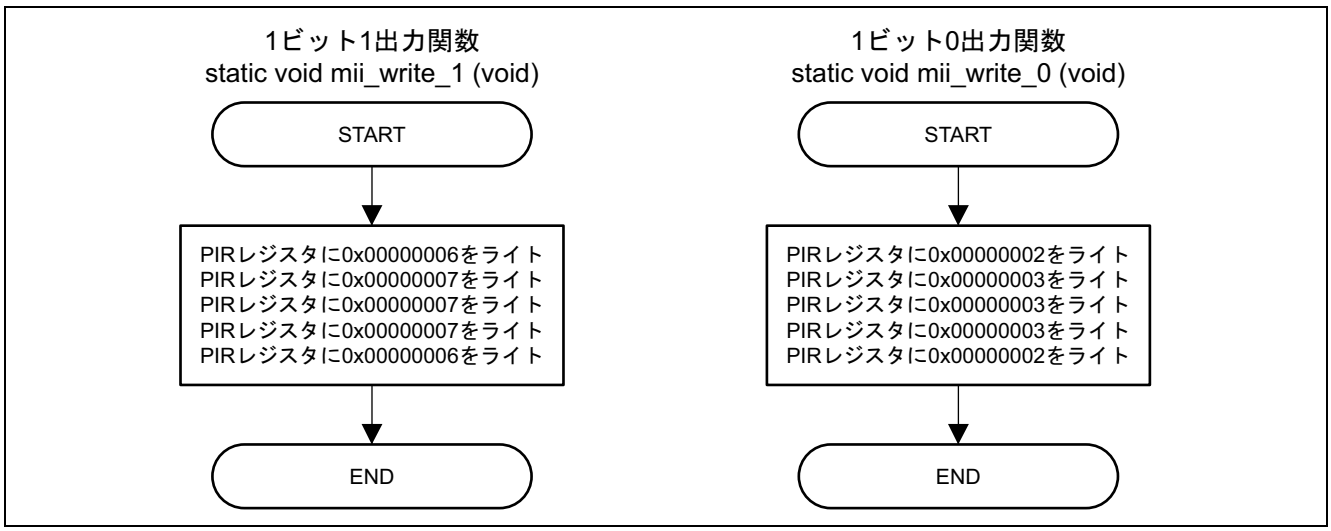


図 31　参考プログラムの処理フロー例 (10)

2.9　参考プログラムの注意事項

2.9.1　ウェイト処理に関する注意点

参考プログラムのウェイト時間の値は参考値となっております。また、ウェイト処理のコードは C ソースで記述しているため、動作周波数やコンパイルオプションまたはコンパイラのバージョンによってウェイト時間が変化します。

ウェイト時間はお客様のシステム合わせて十分に評価していただく必要がありますのでご注意ください。

参考プログラムは 32MHz で動作させた場合の設定値となっております。

1. ソフトウェアリセット完了待ちのウェイト処理
   EtherC、E-DMAC を対象としたソフトウェアリセットの発行期間中はイーサネット関係の全てのモジュールに対するレジスタアクセスが禁止されています。ソフトウェアリセット完了までの間 (64 ステート期間) は、イーサネット関係の全てのモジュールに対するレジスタアクセスを行わないようにする必要があるため、参考プログラムではソフトウェアリセットの完了待ちを行っています。ウェイト処理は"ether.c"ファイルの lan_reg_reset 関数内 (194 行目～197 行目) の下記コードで行っています。ウェイト時間を修正する場合はローカル変数 t の設定値を修正してください。
   参考プログラムは 32MHz で動作させた場合の設定値となっております。設定値は H8S/2472 の要求スペックよりも十分なマージンを持った値となっております。

```
/* ==== Wait for 64 cycles at φ (approx. 2 us@φ = 32 MHz) is required. ==== */
while(--t){
  /* wait */
}
```

2. PHY-LSI リセット完了待ちのウェイト処理
   参考プログラムは phy_autonego 関数内で PHY-LSI をリセットしているのでリセットが完了するまでウェイトを行っています。ウェイト処理は"phy.c"ファイルの phy_autonego 関数内 (130 行目～136 行目) の下記コードで行っています。ウェイト時間を修正する場合はローカル変数 t, i の設定値を修正してください。
   また、PHY-LSI リセット完了待ちのウェイト処理には後述の「4. システム動作を安定させるウェイト」が含まれております。このため、ウェイト時間を短くした場合には自動交渉が成功していても通信に失敗することがありますので、お客様のシステムに合わせて十分に評価して頂く必要があります。

```
/* ---- Wait ---- */
for(i=0;i<1000;i++){
  t=0x27C0;
  while( --t){              /* approx. 2.9-ms wait counting@32 MHz */
     ;
  }
}
```

3. 自動交渉完了待ちのウェイト処理
   参考プログラムは通信方式の選択に自動交渉機能を用いているので自動交渉が完了するまでウェイトを行っています。ウェイト処理は"phy.c"ファイルの phy_autonego 関数内 (168 行目～175 行目) にある下記コードで行っています。ウェイト時間を修正する場合はローカル変数 t, i の設定値を修正してください。

```
/* ==== Wait loop for end of automatic negotiation ==== */
for( i=0; i<500; i++){

  /* ---- approx. 10-ms wait ---- */
  t=36000;
  while( --t){            /* approx. 10-ms wait counting@32 MHz */
     ;
  }
```

4. システム動作を安定させるためのウェイト処理
   参考プログラムは通信方式の選択に自動交渉機能を用いています。H8S/2472 と H8S/2472 の接続相手の自動交渉完了までの時間差が大きい場合、自動交渉が成功していても通信に失敗することがあります。
   H8S/2472 側の PHY が自動交渉に成功していても H8S/2472 の接続相手の受信準備が完了していない場合、接続相手が受信可能な状態になるまで H8S/2472 側でウェイト処理を挿入してください。
   接続相手が受信可能な状態になるまでの時間はシステムによって異なるため、ウェイト時間はお客様側で十分に評価して頂く必要があります。
   前述の 2.9.1 の「2. PHY-LSI リセット完了待ちのウェイト処理」には、システム動作を安定させるためのウェイト処理が含まれております。
   システム動作を安定させるためのウェイト時間は"ether.c"ファイルの lan_open 関数内 (139 行目～143 行目) の下記コードを有効にして修正してください。または、前述の「2. PHY-LSI リセット完了待ちのウェイト処理」の部分を修正してください。

```
#if 0
  /* Delay to stabilize */
  /* Set the count according to the system */
  for( i = 0 ; i < 0x00100000 ; i++ );
#endif
```

5. 送信完了待ちのウェイト処理
参考プログラムの送信関数 (lan_send 関数) は関数内 ("ether.c"ファイルの 392 行目～415 行目) の下記コードで送信完了を確認しています。

```
 /* ==== Confirmation of completion of data transmission  ==== */
 while(tx_flag0 == TX_FLAG_ON){  /* A flag indicating transmission in progress is turned
ON */

   /* ==== approx. 10us wait ==== */
   for(w=0;w<16;w++){
      ;
   }

   /* ==== Workaround of Technical Update "TN-H8*-A429A/J" or "TN-H8*-A429A/E" ==== */
   if((--t1ms) <= 0){
      t1ms = 100;   /*  */
      if(psenddesc0->td0.BIT.TACT == 0){
         tx_flag0 = TX_FLAG_OFF;
         break;
      }
      /* ==== Workaround of Technical Update "TN-H8*-A428A/J" or "TN-H8*-A428A/E" ==== */
      else if((--t400ms) <= 0){
         return SEND_NG;
      }
      else{
         /* DO NOTHING */
      }
   }
 }
```

参考プログラムでは for 文で記述したウェイト時間ごとにグローバル変数の tx_flag0 を確認することで送信完了を判断しています。tx_flag0 は送信完了のステータスを示しており、割り込み関数内でステータスが更新されます。
また、送信完了の確認には IP の不具合対策とタイムアウト処理が組み込まれています。不具合の詳細はテクニカルアップデート TN-H8*-A429A/J と TN-H8*-A428A/J を参照ください。IP の不具合対策として、ローカル変数の t1ms で設定したタイミングで TACT ビットを確認しています。TACT ビットを確認するタイミングは任意の時間となります。送信完了のチェックのタイムアウト時間は、テクニカルアップデート TN-H8*-A428A/J に記載された最大規定時間 (1 フレーム送信にかかる最大時間または、フロー制御の最大時間) を参考にして、ローカル変数 t400ms で設定しています。
ウェイト時間を修正する場合は 3 つのローカル変数 w、t1ms、t400ms の設定値を修正ください。
ウェイトする時間は、送信パフォーマンスに影響しますのでお客様側で十分に評価していただく必要があります。

2.9.2 FIFO容量指定レジスタ (FDR)、送信FIFOしきい値指定レジスタ (TFTR) の設定値を変更する場合の注意点

参考プログラムは H8S/2472 の IP 不具合を回避するため、FIFO 容量指定レジスタ (FDR) に 2048 バイトと送信 FIFO しきい値指定レジスタ (TFTR) にストア&フォワードモードを設定しています。

FIFO 容量指定レジスタ (FDR) と送信 FIFO しきい値指定レジスタ (TFTR) を参考プログラムの設定値以外で使用した場合はテクニカルアップデート TN-H8*-A428A/J の不具合が発生する可能性がありますので、参考プログラムの設定値から変更しないでください。

FDR と TFTR の設定値を変更される場合、お客様側でテクニカルアップデート TN-H8*-A428A/J の不具合対策を行っていただく必要がありますのでご注意ください。

2.9.3 H8S/2472 のIP不具合に関する注意点

H8S/2472 の IP には 2 つの不具合があります。不具合の詳細はテクニカルアップデート TN-H8*-A429A/J と TN-H8*-A428A/J を参照ください。

参考プログラムでは、「2.9.1 ウェイト処理に関する注意点の 5 送信完了待ちのウェイト処理」と「2.9.2 FIFO 容量指定レジスタ (FDR)、送信 FIFO しきい値指定レジスタ (TFTR) の設定値を変更する場合の注意点」で述べた内容により、不具合の対策を施してあります。

3. 参考プログラムリスト

3.1 サンプルプログラムリスト "LAN_2472.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************
28    * Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29    ***********************************************************************
30    * File Name    : LAN_2472.c
31    * Version      : 1.01
32    * Device(s)    : H8S/2472
33    * Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
34    * OS          : None
35    * H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
36    * Description  : Main Program
37    * Operation    :
38    * Limitations  : None
39    ***********************************************************************
40    * History : DD.MM.YYYY Version Description
41    *         : 18.10.2007 1.00    First Release
42    *         : 26.04.2010 1.01    Modification of test program
43    ***********************************************************************/
44
45
46    /***********************************************************************
47    Includes   <System Includes> , "Project Includes"
48    ***********************************************************************/
49    #include <machine.h>
50    #include "ether¥iodefine2472.h"
51    #include "ether¥ether.h"
52
53    /***********************************************************************
54    Typedef definitions
55    ***********************************************************************/
56
57    /***********************************************************************
58    Macro definitions
59    ***********************************************************************/
```

```
60
61    /***********************************************************************
62    Imported global variables and functions (from other files)
63    ***********************************************************************/
64
65    /***********************************************************************
66    Exported global variables and functions (to be accessed by other files)
67    ***********************************************************************/
68
69    /***********************************************************************
70    Private global variables and functions
71    ***********************************************************************/
72    #ifdef __cplusplus
73    extern "C" {
74    void abort(void);
75    #endif
76    void main(void);
77    #ifdef __cplusplus
78    }
79    #endif
80
81    static unsigned char frame[] =
82    {
83        /* MAC header */
84        0x00,0x0e,0x35,0x18,0x34,0xfa,  /* Destination MAC Address(00-0E-35-18-34-FA) */
85        0x00,0x11,0x25,0xbc,0xfd,0x0a,  /* Source MAC Address(00-11-25-BC-FD-0A) */
86        0x08,0x00,                  /* Type(IP) */
87        /* IP header */
88        0x45,0x00,0x00,0x2e,          /* Version(IPv4), IHL(20byte), TOS, Total Leneght(46byte), */
89        0x02,0xf6,0x00,0x00,          /* Identification, Flags, Fragment Offset */
90        0x80,0x01,0x43,0x67,          /* TTL, Protocol(ICMP), Header Checksum */
91        0xac,0x1e,0x4e,0x24,          /* Source Address(172.30.78.36) */
92        0xac,0x1e,0x4e,0x22,          /* Destination Address(172.30.78.34) */
93        /* ICMP header(Echo request) */
94        0x08,0x00,0xb1,0xff,          /* Type, Code, Checksum */
95        0x02,0x00,0x04,0x00,          /* Identifier, Sequence Number */
96        /* Data */
97        0x40,0x00,0x00,0x00,0x00,0x00,
98        0x00,0x00,0x00,0x00,0x00,0x00,
99        0x00,0x00,0x00,0x00,0x00,0x00
100   };
101
102   /* ==== Declaration of global variables ==== */
103   unsigned charuser_buffer[10][1520];
104   unsigned short   receive_size[10];
105
106
107   /*""FUNC COMMENT""*****************************************************
108   * Outline    : Sample program "main"
109   *-----------------------------------------------------------------------
110   * Declaration: void sample_send_recv(void)
111   *-----------------------------------------------------------------------
112   * Description: Two rounds of transmitting one frame and receiving one frame proceed.
113   *-----------------------------------------------------------------------
114   * Argument       : None
115   *-----------------------------------------------------------------------
116   * Return Value   : None
117   *-----------------------------------------------------------------------
118   * Note           :
119   *""FUNC COMMENT END""*************************************************/
120   void sample_send_recv(void)
121   {
122       OPEN_STATUS      opensts;
123       SEND_STATUS      sendsts;
```

```
124       CLOSE_STATUS closests;
125
126      static int i = 10;
127
128      /* ==== Initialization of the EtherC/E-DMAC, PHY, and buffer memory  ==== */
129      opensts = lan_open();    /*  ch0 is selected */
130
131      /* ==== Transmission if open  ==== */
132       if(opensts == OPEN_OK){
133         /* ==== Packet transmission  ==== */
134         for(i=0;i<2;i++){
135             sendsts = lan_send(frame,sizeof(frame));
136             if(sendsts == SEND_NG){    /* Transmission error */
137                 break;
138             }
139             receive_size[i] = lan_recv(&user_buffer[i][0]);
140         }
141       }
142
143       /* ==== LAN close ==== */
144       closests = lan_close();
145       if(closests == CLOSE_NG){
146         ;   /* waiting */
147       }
148   }
149
150   /*""FUNC COMMENT""*****************************************************************
151   * Outline     : Sample program "main"
152   *-------------------------------------------------------------------------------
153   * Declaration: void sample_main(void)
154   *-------------------------------------------------------------------------------
155   * Description: 10 frames are transmitted from the Ethernet.
156   *-------------------------------------------------------------------------------
157   * Argument       : None
158   *-------------------------------------------------------------------------------
159   * Return Value   : None
160   *-------------------------------------------------------------------------------
161   * Note           :
162   *""FUNC COMMENT END""*************************************************************/
163   void sample_send(void)
164   {
165       OPEN_STATUS      opensts;
166       SEND_STATUS      sendsts;
167       CLOSE_STATUS closests;
168
169      int i = 10;
170
171      /* ==== Initialization of the EtherC/E-DMAC, PHY, and buffer memory  ==== */
172      opensts = lan_open();    /*  ch0 is selected */
173
174      /* ==== Transmission if open  ==== */
175       if(opensts == OPEN_OK){
176
177         /* ==== Packet transmission  ==== */
178         for(i=0;i<10;i++){
179             sendsts = lan_send(frame,sizeof(frame));
180             if(sendsts == SEND_NG){    /* Transmission error */
181                 break;
182             }
183         }
184       }
185
186       /* ==== LAN close ==== */
187       closests = lan_close();
```

```
188       while(closests == CLOSE_NG){
189         ;   /* waiting */
190       }
191   }
192
193   /*""FUNC COMMENT""*****************************************************************
194   * Outline     : Sample program "main"
195   *-------------------------------------------------------------------------------
196   * Declaration: void sample_recv(void)
197   *-------------------------------------------------------------------------------
198   * Description: 10 frames are received from the Ethernet.
199   *-------------------------------------------------------------------------------
200   * Argument       : None
201   *-------------------------------------------------------------------------------
202   * Return Value   : None
203   *-------------------------------------------------------------------------------
204   * Note           :
205   *""FUNC COMMENT END""*************************************************************/
206   void sample_recv(void)
207   {
208      OPEN_STATUS    opensts;
209      CLOSE_STATUS   closests;
210
211       int i;
212
213      /* ==== Initialization of the EtherC/E-DMAC, PHY, and buffer memory  ==== */
214      opensts = lan_open();     /*  Ch0 is selected */
215
216      /* ==== Reception if open  ==== */
217      if(opensts == OPEN_OK){
218
219         /* ==== Packet reception  ==== */
220         for(i=0;i<10;i++){
221            receive_size[i] = lan_recv(&user_buffer[i][0]);
222         }
223      }
224
225      /* ==== LAN close ==== */
226      closests = lan_close();
227      while(closests == CLOSE_NG){
228         ;   /* waiting */
229      }
230   }
231
232
233   void main(void)
234   {
235       int   i;
236       unsigned char dummy;
237
238       i = 0;
239
240       dummy = SYSTEM.MDCR.BYTE;
241       SYSTEM.SUBMSTPBH.BIT.EtherC = 0;
242       SYSTEM.SUBMSTPBH.BIT.EDMAC = 0;
243       SYSTEM.MDCR.BIT.EXPE = 0;       // 0: Single-chip mode   1: Extended mode
244       SYSTEM.SYSCR.BIT.INTM = 1;      // Interrupt control mode1
245
246       set_imask_exr(0);
247       and_ccr(0x3F); // Interrupt level 0
248
249       switch(i){
250         case 0:
251             sample_send_recv();
```

```
252             while(1);
253             break;
254         case 1:
255             sample_send();
256             while(1);
257             break;
258         case 2:
259             sample_recv();
260             while(1);
261             break;
262         default:
263             break;
264       }
265   }
266
267
268   #ifdef __cplusplus
269   void abort(void)
270   {
271
272   }
273   #endif
```

3.2　サンプルプログラムリスト "ether.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************
28    * Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29    ***********************************************************************
30    * File Name    : ether.c
31    * Version      : 1.02
32    * Device(s)    : H8S/2472
33    * Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
34    * OS          : None
35    * H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
36    * Description  : Ethernet transmission processing
37    * Limitations  : None
38    ***********************************************************************
39    * History : DD.MM.YYYY Version Description
40    *         : 15.05.2007 1.00    First Release
41    *         : 19.04.2010 1.01    Bug fix of reception
42    *         : 29.07.2010 1.02    Modification of wait counter
43    *         :                 Modification of send complete check
44    ***********************************************************************/
45
46
47    /***********************************************************************
48    Includes   <System Includes> , "Project Includes"
49    ***********************************************************************/
50    #include <machine.h>
51    #include <string.h>
52    #include "iodefine2472.h"
53    #include "ether.h"
54    #include "phy.h"
55
56    /***********************************************************************
57    Typedef definitions
58    ***********************************************************************/
59
60    /***********************************************************************
61    Macro definitions
62    ***********************************************************************/
```

```
63    //#define ETHER_LOOP_BACK
64
65    /******************************************************************************
66    Imported global variables and functions (from other files)
67    ******************************************************************************/
68
69    /******************************************************************************
70    Exported global variables and functions (to be accessed by other files)
71    ******************************************************************************/
72
73    /******************************************************************************
74    Private global variables and functions
75    ******************************************************************************/
76    /* ==== Declaration of prototype ==== */
77    static void lan_reg_reset(void);
78    static void lan_reg_set(int link);
79    static void lan_desc_create(void);
80
81    /* ==== Declaration of variables ==== */
82    volatile static int tx_flag0;
83    static volatile  EDMAC_SEND_DESC  *psenddesc0;
84    static volatile  EDMAC_RECV_DESC *precvdesc0;
85
86    /* Since descriptors need to be placed on 16-byte boundaries,
87     a section on a 16-byte boundary is reserved for the descriptors. */
88    #pragma section TXRXBUFFDESC
89    static   TXRX_BUFFER_SET buffer0;
90    static  TXRX_DESCRIPTOR_SET   descriptor0;
91    #pragma section
92
93    /*""FUNC COMMENT""*****************************************************************
94    * Outline    : LAN Open Functions
95    *-------------------------------------------------------------------------------
96    * Declaration: int lan_open(void)
97    *-------------------------------------------------------------------------------
98    * Description: The E-DMAC, EtherC, transmit/receive descriptors, and memory
99    *              for the transmission/receiving buffers are initialized.
100   *             : The result of automatic negotiation by PHY-LSI is obtained,
101   *              and an error is returned if automatic negotiation is failed.
102   *-------------------------------------------------------------------------------
103   * Argument       : None
104   *-------------------------------------------------------------------------------
105   * Return Values : 0(OPEN_OK) : Success in opening
106   *             : -1(OPEN_NG) : Failure in opening
107   *-------------------------------------------------------------------------------
108   * Note           :
109   *""FUNC COMMENT END""*************************************************************/
110   int lan_open(void)
111   {
112       unsigned int physts;
113       // volatile unsigned long i;
114
115       /* ==== Reset of the EtherC/E-DMAC ==== */
116       lan_reg_reset();
117
118       /* ==== Initialization of transmit/receive descriptors ==== */
119       lan_desc_create();
120
121   #ifdef ETHER_LOOP_BACK
122       MAC0.ECMR.BIT.ILB =1;
123       physts = FULL_TX;
124   #else
125       /* ==== Obtaining result of automatic negotiation by PHY-LSI ==== */
126       physts = phy_autonego();
```

```
#endif

    /* ==== In case of success in automatic negotiation ==== */
    if(physts != NEGO_FAIL){

      /* ==== Setting of EtherC/E-DMAC registers ==== */
      lan_reg_set(physts);
    }
    else{
      return OPEN_NG;
    }

#if 0
    /* Delay to stabilize */
    /* Set the count according to the system */
    for( i = 0 ; i < 0x00100000 ; i++ );
#endif

    return OPEN_OK;
}


/*""FUNC COMMENT""*****************************************************************
* Module Outline : LAN Close Function
*---------------------------------------------------------------------------------
* Declaration    : int lan_close(void)
*---------------------------------------------------------------------------------
* Description    : The Ether function is halted, and transmission and reception are prohibited.
*---------------------------------------------------------------------------------
* Argument           : void
*---------------------------------------------------------------------------------
* Return Value       : 0(CLOSE_OK) : Success in closing
*                : -1(CLOSE_NG) : Failure in closing
*---------------------------------------------------------------------------------
* Note               :
*""FUNC COMMENT END""*************************************************************/
int lan_close(void)
{
    /* ==== Reset of registers related to the EtherC and E-DMAC ==== */
    lan_reg_reset();

    /* ==== Setting of the interrupt control register (ICRD5) ==== */
    INT.ICRD.BIT.ICRD5 = 0;

    return CLOSE_OK;
}

/*""FUNC COMMENT""*****************************************************************
* Outline    : EtherC/E-DMAC Reset Function
*---------------------------------------------------------------------------------
* Declaration: static void lan_reg_reset(void)
*---------------------------------------------------------------------------------
* Description: Software of the E-DMAC and EtherC is reset.
*---------------------------------------------------------------------------------
* Argument       : None
*---------------------------------------------------------------------------------
* Return Value   : None
*---------------------------------------------------------------------------------
* Note           : -
*""FUNC COMMENT END""*************************************************************/
static void lan_reg_reset(void)
{
    volatile int t = 10; /* For approx. 3-us wait @32 MHz */

```

```
191       /* ==== Setting of the E-DMAC mode register (EDMR) ==== */
192       EDMAC0.EDMR.BIT.SWR =1;
193
194       /* ==== Wait for 64 cycles at φ (approx. 2 us@φ = 32 MHz) is required. ==== */
195       while(--t){
196         /* wait */
197       }
198   }
199
200   /*""FUNC COMMENT""*****************************************************************
201   * Outline     : Initialization of the EtherC/E-DMAC Registers
202   *-------------------------------------------------------------------------------
203   * Declaration: static void lan_reg_set(int link)
204   *-------------------------------------------------------------------------------
205   * Description: Registers of the E-DMAC and EtherC are set so that
206   *              transmission and receiving operations are allowed.
207   *            : Settings for reception have been made but receiving
208   *              operations are not initiated.
209   *            : The initial value of TSU_FWSLC is changed so that signals
210   *              from pins CAMSEN0 and CAMSEN1 are not referred to during
211   *              receiving operations.
212   *-------------------------------------------------------------------------------
213   * Argument       : int link: I : Result of automatic negotiation by PHY
214   *            : HALF_10M(1), FULL_10M(2), HALF_TX(3), FULL_TX(4)
215   *-------------------------------------------------------------------------------
216   * Return Value   : None
217   *-------------------------------------------------------------------------------
218   * Note           : This function is based on the assumption that an external
219   *              CAM is not used.
220   *""FUNC COMMENT END""*************************************************************/
221   static void lan_reg_set(int link)
222   {
223       /* ==== Clear of the EtherC status register (ECSR) ==== */
224       MAC0.ECSR.LONG = 0x00000007; /* Clear of 1 write */
225
226       /* ==== Clear of the EtherC/E-DMAC status register (EESR) ==== */
227       EDMAC0.EESR.LONG = 0x47FF0F9F;   /* Clear of 1 write */
228
229       /* ==== Setting of the transmit descriptor list address register (TDLAR) ==== */
230       EDMAC0.TDLAR = descriptor0.send_desc;
231
232       /* ==== Setting of the receive descriptor list address register (RDLAR) ==== */
233       EDMAC0.RDLAR = descriptor0.recv_desc;
234
235       /* ==== Setting of the transmit/receive status copy enable register (TRSCER) ==== */
236       EDMAC0.TRSCER.LONG = 0x00000000;
237
238       /* ==== Setting of transmit FIFO threshold register (TFTR) ==== */
239       EDMAC0.TFTR.LONG = 0x00000000;  /* Store and forward mode */
240
241       /* ==== Setting of the FIFO depth register (FDR) ==== */
242       EDMAC0.FDR.LONG = 0x00000707;   /* Capacity of transmit/receive FIFO is set to 2 KB. */
243
244       /* ==== Setting of the receiving method control register (RMCR) ==== */
245       EDMAC0.RMCR.BIT.RNC = 0x1;  /* Continuous reception */
246
247       /* ==== Setting of the MAC address high register/MAC address low register (MAHR, MALR) ====
248   */
249       MAC0.MAHR.LONG = MAC_ADDRESS_HIGH0;
250       MAC0.MALR.LONG = MAC_ADDRESS_LOW0;
251
252       /* ==== Setting of the interrupt control register (ICRD5) ==== */
253       INT.ICRD.BIT.ICRD5 = 1;
254
```

```
255       /* ==== Setting of the EtherC interrupt permission register (ECSIPR) ==== */
256       MAC0.ECSIPR.LONG = 0x0000;  /* All disabled (change of LINK signals,
257                                   Magic Packet detection,
258                                   illegal carrier detection) */
259
260       /* ==== Setting of the EtherC/E-DMAC status interrupt permission register (EESIPR) ==== */
261       EDMAC0.EESIPR.LONG = 0x073c039f;    /* Transmission/reception related sources are only enabled.
262   */
263   //  EDMAC0.EESIPR.LONG = 0x04380300;    /* Transmission related sources are only enabled. */
264   //  EDMAC0.EESIPR.LONG = 0x0304009f;    /* Reception related sources are only enabled. */
265                                         /* However, receive descriptor empty interrupt and receive FIFO
266   overflow bits are excluded */
267
268       /* ==== Setting of the EtherC mode register (ECMR) ==== */
269       if(link == FULL_TX || link == FULL_10M){
270         MAC0.ECMR.BIT.DM =1;   /* Full-duplex transfer */
271       }
272       else {
273         MAC0.ECMR.BIT.DM =0;   /* Half-duplex transfer */
274       }
275
276       if(link == FULL_TX || link == HALF_TX){
277         EDMAC0.ECBRR.BIT.RTM =1;   /* 100Mbps */
278       }
279       else {
280         EDMAC0.ECBRR.BIT.RTM =0;   /* 10Mbps */
281       }
282
283       MAC0.ECMR.BIT.RE = 1;    /* Reception enabled */
284       MAC0.ECMR.BIT.TE = 1;    /* Transmission enabled */
285
286       /* ==== Setting of the E-DMAC receive request register (EDRRR) ==== */
287       EDMAC0.EDRRR.LONG = 0x00000001; /* The E-DMAC is ready to receive. */
288   }
289
290   /*""FUNC COMMENT""*****************************************************************
291   * Outline     : Initialization of Transmit/Receive Descriptors
292   *-------------------------------------------------------------------------------
293   * Declaration: static void lan_desc_create(void)
294   *-------------------------------------------------------------------------------
295   * Description: Transmit/receive descriptors and transmission/receiving
296   *              buffers are initialized.
297   *             : Descriptors are arranged in a ring structure.
298   *-------------------------------------------------------------------------------
299   * Argument       : None
300   *-------------------------------------------------------------------------------
301   * Return Value   : None
302   *-------------------------------------------------------------------------------
303   * Note           : The TACT and TDL bits are set to 0 for initialization.
304   *              These bits are set by the lan_send function.
305   *""FUNC COMMENT END""*************************************************************/
306   static void lan_desc_create(void)
307   {
308       int i;
309       EDMAC_SEND_DESC *psnd;
310       EDMAC_RECV_DESC *prcv;
311
312       /* ==== Clear of transmit/receive descriptors to 0 ==== */
313       memset(&descriptor0,0x0,sizeof(descriptor0));
314
315       /* ==== Initialization of transmit descriptors ==== */
316       psnd = descriptor0.send_desc;
317       for(i = 0; i<NUM_OF_TX_DESCRIPTOR ;i++){
318         psnd->td2.TBA = &buffer0.send_buf[i][0];
```

```
319         psnd->td0.BIT.TFP =0x3;    /* 1 frame/1 descriptor */
320         psnd->pNext = psnd +1;
321         psnd++;
322       }
323       psnd--;
324       psnd->td0.BIT.TDLE = 1;
325       psnd->pNext = descriptor0.send_desc;
326
327       /* ==== Initialization of receive descriptors ==== */
328       prcv = descriptor0.recv_desc;
329       for(i = 0; i<NUM_OF_RX_DESCRIPTOR ;i++){
330         prcv->rd0.BIT.RACT =0x1; /* Restore the descriptor to the state where reception is possible
331   */
332         prcv->rd1.RBL = 0x05f0;
333         prcv->rd2.RBA = &buffer0.recv_buf[i][0];
334         prcv->pNext = prcv +1;
335         prcv++;
336       }
337       prcv--;
338       prcv->rd0.BIT.RDLE = 1;
339       prcv->pNext = descriptor0.recv_desc;
340
341       /* ==== Clear of transmission and receiving buffers to 0 ==== */
342       memset(&buffer0,0x0,sizeof(buffer0));
343
344       /* ==== Initialization of pointers to transmit & receive descriptors ==== */
345       psenddesc0 = descriptor0.send_desc;
346       precvdesc0 = descriptor0.recv_desc;
347
348   }
349
350   /*""FUNC COMMENT""*****************************************************************
351   * Outline     : Ethernet Frame Transmission Function
352   *-------------------------------------------------------------------------------
353   * Declaration     : int lan_send(unsigned char *addr, int flen)
354   *-------------------------------------------------------------------------------
355   * Description     : Data specified by the arguments are copied to the transmission buffer.
356   *             : Transmit descriptors are set and transmission operation is initiated.
357   *             : A flag indicating "transmission in progress" is checked.
358   *               If the flag is OFF, transmission is judged to have been completed.
359   *-------------------------------------------------------------------------------
360   * Argument        : unsigned char *addr : I : Start address of the Ethernet frame for
361   *               transmission
362   *             : int flen : I : Frame size (number of bytes)
363   *-------------------------------------------------------------------------------
364   * Return value    : 0(SEND_OK) : Success in transmission
365   *             : -1(SEND_NG) : Failure in transmission
366   *-------------------------------------------------------------------------------
367   * Note            :
368   *""FUNC COMMENT END""*************************************************************/
369   int lan_send(unsigned char *addr, int flen)
370   {
371       volatile int    w;
372       volatile int t1ms = 100; /* approx. 1-ms counter */
373       volatile int t400ms = 400;  /* approx. 400-ms counter */
374       int           value;
375
376       /* ==== Wait until the TACT bit of the transmit descriptor becomes 0 ==== */
377       while(psenddesc0->td0.BIT.TACT == 1){
378         /* wait */
379       }
380
381       /* ==== Turn ON the flag to indicate transmission in progress. ==== */
382       tx_flag0 = TX_FLAG_ON;
```

```
383
384       /* ==== Copy data for transmission indicated by arguments to the transmission buffer ==== */
385       memcpy(psenddesc0->td2.TBA,addr,flen);
386
387       /* ==== Setting of transmit descriptors==== */
388       psenddesc0->td1.TDL = flen;
389       psenddesc0->td0.BIT.TACT = 1;
390
391       /* ==== Initiating transmission ==== */
392       if(EDMAC0.EDTRR.BIT.TR == 0){
393         EDMAC0.EDTRR.BIT.TR = 1;
394       }
395
396       /* ==== Confirmation of completion of data transmission  ==== */
397       while(tx_flag0 == TX_FLAG_ON){  /* A flag indicating transmission in progress is turned ON */
398
399         /* ==== approx. 10us wait ==== */
400         for(w=0;w<16;w++){
401             ;
402         }
403
404         /* ==== Workaround of Technical Update "TN-H8*-A429A/J" or "TN-H8*-A429A/E" ==== */
405         if((--t1ms) <= 0){
406             t1ms = 100; /*  */
407             if(psenddesc0->td0.BIT.TACT == 0){
408                 tx_flag0 = TX_FLAG_OFF;
409                 break;
410             }
411             /* ==== Workaround of Technical Update "TN-H8*-A428A/J" or "TN-H8*-A428A/E" ==== */
412             else if((--t400ms) <= 0){
413                 return SEND_NG;
414             }
415             else{
416                 /* DO NOTHING */
417             }
418         }
419       }
420
421       /* ==== Setting of transmit descriptors==== */
422       psenddesc0 = psenddesc0->pNext; /* Update to a pointer for descriptor management */
423
424       return SEND_OK;
425   }
426
427   /*""FUNC COMMENT""************************************************************
428   * Outline        : Ethernet Frame Reception Function
429   *-----------------------------------------------------------------------------
430   * Declaration    : int lan_recv(unsigned char *addr)
431   *-----------------------------------------------------------------------------
432   * Description    : Ethernet frame of one frame is only received.
433   *             : If there are no errors in the received frame, data are copied
434   *               to the user buffer specified by an argument.
435   *-----------------------------------------------------------------------------
436   * Argument       : unsigned char *addr : 0 : Start address to which the received
437   *              Ethernet frame is copied
438   *-----------------------------------------------------------------------------
439   * Return Value   : Number of bytes of the received frame : In case that receiving
440   *              operation is succeeded
441   *-----------------------------------------------------------------------------
442   * Note           :
443   *""FUNC COMMENT END""*************************************************************/
444   int lan_recv(unsigned char *addr)
445   {
446       int ret;
```

```
447
448       do {
449         ret = lan_recv_ex(addr);
450       } while (ret < 0);
451
452       return ret;
453   }
454
455
456   /*""FUNC COMMENT""*****************************************************************
457   * Outline        : Ethernet Frame Reception Function
458   *-------------------------------------------------------------------------------
459   * Declaration    : int lan_recv_ex(unsigned char *addr)
460   *-------------------------------------------------------------------------------
461   * Description    : Ethernet frame of one frame is only received.
462   *             : If there are no errors in the received frame, data are copied
463   *               to the user buffer specified by an argument.
464   *-------------------------------------------------------------------------------
465   * Argument       : unsigned char *addr : 0 : Start address to which the received
466   *               Ethernet frame is copied
467   *-------------------------------------------------------------------------------
468   * Return Value   : Number of bytes of the received frame : In case that receiving
469   *               operation is succeeded
470   *             : -1 : No receive data
471   *-------------------------------------------------------------------------------
472   * Note           :
473   *""FUNC COMMENT END""*************************************************************/
474   int lan_recv_ex(unsigned char *addr)
475   {
476       int i;
477       int dsize = 0;   /* Number of received data bytes */
478
479       /* ==== Check whether receive data remains ==== */
480       if(precvdesc0->rd0.BIT.RACT == 0x1)
481       { /* No receive data */
482          return -1;
483       }
484       else if(precvdesc0->rd0.BIT.RACT == 0x0)
485       { /* Receive data remains */
486
487         /* ==== Confirmation of received frame error ==== */
488         if(precvdesc0->rd0.BIT.RFE == 0)
489         { /* Case where no received frame errors occur */
490             memcpy(addr,precvdesc0->rd2.RBA,precvdesc0->rd1.RDL);
491             dsize = precvdesc0->rd1.RDL;
492         }
493
494         /* ==== Initialization of receive descriptors ==== */
495         precvdesc0->rd0.LONG &= 0x40000000;   /* Bits other than RDLE are cleared to 0 */
496         precvdesc0->rd0.BIT.RACT=0x1; /*Restore the descriptor to the state where reception is possible
497   */
498         precvdesc0->rd1.RDL = 0x0000;
499         precvdesc0 = precvdesc0->pNext;
500
501         /* ==== Initiating data reception ==== */
502         if(EDMAC0.EDRRR.BIT.RR == 0)
503         {
504             EDMAC0.EDRRR.BIT.RR = 1;
505         }
506
507         return dsize;
508       }
509   }
510
```

```
/*""FUNC COMMENT""*************************************************************
* Outline        : Interrupt Handling for Completion of E-DMAC Transmission (ch0)
*-------------------------------------------------------------------------------
* Declaration    : void INT_EDMAC_EINT0(void)
*-------------------------------------------------------------------------------
* Description    : Interrupt handling for completion of transmission and
*            : reception of frames When transmission of frames is completed,
*            : the flag to indicate transmission in progress is turned OFF.
*-------------------------------------------------------------------------------
* Argument       : None
*-------------------------------------------------------------------------------
* Return Value   : None
*-------------------------------------------------------------------------------
* Note           :
*""FUNC COMMENT END""*********************************************************/
#pragma section IntPRG
__interrupt(vect=119) void INT_EDMAC_EINT0(void)
{
    unsigned long status;

    /* ==== Reading of interrupt status ==== */
    status = EDMAC0.EESR.LONG & EDMAC0.EESIPR.LONG;

    /* ==== Clear of interrupt sources ==== */
    EDMAC0.EESR.LONG = status;  /* Clear of 1 write */

    /* ==== At the time frame transmission has been completed ==== */
    if(status & FRAME_TRANSMIT_COMPLETE){
      tx_flag0 = TX_FLAG_OFF;
    }
}
#pragma section
/* End of File */
```

3.3　サンプルプログラムリスト "ether.h"

```
1     /***********************************************************************
2     * DISCLAIMER
3     * Please refer to http://www.renesas.com/disclaimer
4     ***********************************************************************
5       Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
6     ***********************************************************************
7     * File Name    : ether.h
8     * Version      : 1.01
9     * Device(s)    : H8S/2472
10    * Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
11    * OS          : None
12    * H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
13    * Description  : This is a sample program for setting the transmit/receive
14    *              descriptors and transmission/receiving buffers.
15    * Limitations  : None
16    ***********************************************************************
17    * History : DD.MM.YYYY Version Description
18    *         : 19.02.2007 1.00    First Release
19    *         : 19.04.2010 1.01    Unnecessary code deleted
20    ***********************************************************************/
21
22    #ifndef ETHER_H
23    #define ETHER_H
24
25    /***********************************************************************
26    Includes   <System Includes> , "Project Includes"
27    ***********************************************************************/
28
29    /***********************************************************************
30    Macro definitions
31    ***********************************************************************/
32    #define NUM_OF_TX_DESCRIPTOR     4     /* Number of descriptors for data transmission */
33    #define NUM_OF_RX_DESCRIPTOR     4     /* Number of descriptors for data reception */
34    #define   SIZE_OF_TX_BUFFER        1520     /* Transmission buffer size must be an integer multiple
35    of 16 bytes. */
36    #define   SIZE_OF_RX_BUFFER        1520     /* Receiving buffer size must be integer multiple of 16
37    bytes. */
38    #define   MAC_ADDRESS_HIGH0  0x001125bc    /* In case that the MAC address is 00-11-25-bc-fd-0a
39    (hexadecimal) */
40    #define   MAC_ADDRESS_LOW0   0x0000fd0a
41    #define   TX_FLAG_ON  1
42    #define   TX_FLAG_OFF 0
43    #define FRAME_TRANSMIT_COMPLETE 0x00200000
44
45    /***********************************************************************
46    Typedef definitions
47    ***********************************************************************/
48    /* ==== Definition of enumeration constant of return value for lan_open() ==== */
49    typedef enum{OPEN_OK= 0,OPEN_NG= -1}OPEN_STATUS;
50
51    /* ==== Definition of enumeration constant of return value for lan_send() ==== */
52    typedef enum{SEND_OK = 0,SEND_NG =-1}SEND_STATUS;
53
54    /* ==== Definition of enumeration constant of return value for lan_close() ==== */
55    typedef enum{CLOSE_OK= 0,CLOSE_NG= -1}CLOSE_STATUS;
56
57    /* ==== Definition of structure of transmit descriptors ==== */
58    typedef union
59    {
60        unsigned long LONG;
61        struct{
62          unsigned int   TACT:1;        /* Transmit descriptor active bit */
```

```
63          unsigned int   TDLE:1;        /* Transmit descriptor list end */
64          unsigned int   TFP:2;         /* Transmit frame position */
65          unsigned int   TFE:1;         /* Occurrence of the transmit frame error (refer to TFSx for
66    error source.) */
67          unsigned int   reserved1:11;  /* Not in use */
68          unsigned int   reserved2:7;   /* Not in use */
69          unsigned int   TFS8:1;        /* Transmit abort */
70          unsigned int   reserved3:4;   /* Not in use */
71          unsigned int   TFS3:1;        /* Carrier not detected at the time of initiating data
72    transmission */
73          unsigned int   TFS2:1;        /* Detect loss of carrier during transmission */
74          unsigned int   TFS1:1;        /* Delayed collision detect */
75          unsigned int   TFS0:1;        /* Transmit retry over */
76        }BIT;
77    }TD0;
78    typedef struct
79    {
80        unsigned short   TDL;           /* Size of transmission buffer (number of bytes) */
81        unsigned short   reserved;
82    }TD1;
83    typedef struct
84    {
85        unsigned char    *TBA;          /* Address of transmission buffer */
86    }TD2;
87
88    typedef struct tag_edmac_send_desc
89    {
90        TD0    td0;
91        TD1    td1;
92        TD2    td2;
93       structtag_edmac_send_desc    *pNext;
94    }EDMAC_SEND_DESC;
95
96    /* ==== Definition of structure of receive descriptors ==== */
97    typedef union
98    {
99        unsigned long LONG;
100       struct{
101         unsigned int   RACT:1;        /* Receive descriptor active */
102         unsigned int   RDLE:1;        /* Receive descriptor list end */
103         unsigned int   RFP:2;         /* Received frame position */
104         unsigned int   RFE:1;         /* Occurrence of received frame error (refer to TFSx for error
105   source.) */
106         unsigned int   reserved1:3;   /* Not in use */
107         unsigned int   reserved2:8;   /* Not in use */
108         unsigned int   reserved3:6;   /* Not in use */
109         unsigned int   RFS9:1;        /* Receive FIFO overflow */
110         unsigned int   RFS8:1;        /* Abort detection during data reception */
111         unsigned int   RFS7:1;        /* Multicast address frame received */
112         unsigned int   reserved4:2;   /* Not in use */
113         unsigned int   RFS4:1;        /* Fraction of bits for frame received error */
114         unsigned int   RFS3:1;        /* Receive too-long frame error */
115         unsigned int   RFS2:1;        /* Receive too-short frame error */
116         unsigned int   RFS1:1;        /* PHY-LSI receive error */
117         unsigned int   RFS0:1;        /* CRC error in received frame */
118       }BIT;
119   }RD0;
120   typedef struct
121   {
122       unsigned short   RBL;           /* Data length of receiving buffer (unit: bytes, specified for
123   16-byte boundaries) */
124       unsigned short   RDL;           /* Length of received data (this is set when the last frame
125   is received.) */
126   }RD1;
```

```
127   typedef struct
128   {
129       unsigned char    *RBA;          /* Start address of receiving buffer, 16-byte boundary in case
130   of SDRAM */
131   }RD2;
132
133   typedef struct tag_edmac_recv_desc
134   {
135       RD0    rd0;
136       RD1    rd1;
137       RD2    rd2;
138      structtag_edmac_recv_desc  *pNext;
139   }EDMAC_RECV_DESC;
140
141   /* ==== Definition of structure of transmission and receiving buffers ==== */
142   typedef struct
143   {
144       /* Area of transmission buffers (this must be aligned with a 16-byte boundary.) */
145       unsigned char    send_buf[NUM_OF_TX_DESCRIPTOR][SIZE_OF_TX_BUFFER];
146
147       /* Area of receiving buffers (this must be aligned with a 16-byte boundary.) */
148       unsigned char    recv_buf[NUM_OF_RX_DESCRIPTOR][SIZE_OF_RX_BUFFER];
149   }TXRX_BUFFER_SET;
150
151   /* ==== Definition of structure of transmit and receive descriptors ==== */
152   typedef struct
153   {
154       /* Transmit descriptor (it must be aligned on 16-byte boundary.) */
155       EDMAC_SEND_DESC    send_desc[NUM_OF_TX_DESCRIPTOR];
156
157       /* Receive descriptor (it must be aligned on 16-byte boundary.) */
158       EDMAC_RECV_DESC    recv_desc[NUM_OF_RX_DESCRIPTOR];
159   }TXRX_DESCRIPTOR_SET;
160
161   /***************************************************************************
162   Variable Externs
163   ***************************************************************************/
164
165   /***************************************************************************
166   Functions Prototypes
167   ***************************************************************************/
168   int lan_open(void);
169   int lan_recv(unsigned char *addr);
170   int lan_recv_ex(unsigned char *addr);
171   int lan_send(unsigned char *addr, int flen);
172   int lan_close(void);
173
174
175
176
177
178   #endif /* ETHER_H */
```

3.4　サンプルプログラムリスト "phy.c"

```
1     /***********************************************************************
2     * DISCLAIMER
3
4     * This software is supplied by Renesas Electronics Corporation and is only
5     * intended for use with Renesas products. No other uses are authorized.
6
7     * This software is owned by Renesas Electronics Corporation and is protected under
8     * all applicable laws, including copyright laws.
9
10    * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    * PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
14    * DISCLAIMED.
15
16    * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    * ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    * FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21
22    * Renesas reserves the right, without notice, to make changes to this
23    * software and to discontinue the availability of this software.
24    * By using this software, you agree to the additional terms and
25    * conditions found by accessing the following link:
26    * http://www.renesas.com/disclaimer
27    ***********************************************************************
28    * Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29    ***********************************************************************
30    * File Name    : phy.c
31    * Version      : 1.02
32    * Device(s)    : H8S/2472
33    * Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
34    * OS          : None
35    * H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
36    * Description  : Initialization Example of the PHY-LSI Automatic Negotiation Function
37    * Limitations  : None
38    ***********************************************************************
39    * History : DD.MM.YYYY Version Description
40    *         : 01.01.2009 1.00    First Release
41    *         : 19.04.2010 1.01    Unnecessary code deleted
42    *         : 29.07.2010 1.02    Modification of wait counter
43    ***********************************************************************/
44
45
46    /***********************************************************************
47    Includes   <System Includes> , "Project Includes"
48    ***********************************************************************/
49    #include "iodefine2472.h"
50    #include "phy.h"
51
52    /***********************************************************************
53    Typedef definitions
54    ***********************************************************************/
55
56    /***********************************************************************
57    Macro definitions
58    ***********************************************************************/
59    #define   NO_WAIT
60    //#undef  NO_WAIT
61
62    //#define PHY_LOOP_BACK
```

```
63
64    /* The MII register of the PHY-LSI */
65    #define   PHY_ADD                     0x03c0
66    #define   BASIC_MODE_CONTROL_REG      PHY_ADD+(0x0000)
67    #define   BASIC_MODE_STATUS_REG       PHY_ADD+(0x0001)
68    #define   PHY_IDENTIFIER1_REG             PHY_ADD+(0x0002)
69    #define   PHY_IDENTIFIER2_REG             PHY_ADD+(0x0003)
70    #define   AN_ADVERTISEMENT_REG        PHY_ADD+(0x0004)
71    #define   AN_LINK_PARTNER_ABILITY_REG    PHY_ADD+(0x0005)
72    #define   AN_EXPANSION_REG            PHY_ADD+(0x0006)
73    /* For accessing the MII */
74    #define PHY_ST                        1
75    #define PHY_WRITE                     1
76    #define PHY_READ                2
77    #define PHY_ADDR0                     1
78
79    /*************************************************************************
80    Imported global variables and functions (from other files)
81    *************************************************************************/
82
83    /*************************************************************************
84    Exported global variables and functions (to be accessed by other files)
85    *************************************************************************/
86
87    /*************************************************************************
88    Private global variables and functions
89    *************************************************************************/
90    static unsigned short    phy_reg_read(unsigned short reg_addr);
91    static void  phy_reg_write(unsigned short reg_addr, unsigned short data);
92    static void  mii_preamble(void);
93    static void  mii_cmd(unsigned short reg_addr, int option);
94    static void  mii_reg_read(unsigned short *data);
95    static void  mii_reg_write(unsigned short data);
96    static void  mii_z(void);
97    static void  mii_ta_10(void);
98    static void  mii_write_1(void);
99    static void  mii_write_0(void);
100
101   /*""FUNC COMMENT""*****************************************************
102   * Outline        : Detection of Negotiation Result of the PHY Link
103   *-------------------------------------------------------------------------
104   * Declaration    : int phy_autonego(void);
105   *-------------------------------------------------------------------------
106   * Description    : Result of automatic negotiation is read and returned as a return value.
107   *            : This function waits up to 5 seconds for automatic negotiation to end.
108   *-------------------------------------------------------------------------
109   * Argument       : None
110   *-------------------------------------------------------------------------
111   * Return Value  : 4(FULL_TX)   :100 Mbps, full-duplex transfer
112   *             : 3(HALF_TX)   :100 Mbps, half-duplex transfer
113   *             : 2(FULL_10M)  :10 Mbps, full-duplex transfer
114   *             : 1(HALF_10M)  :10 Mbps, half-duplex transfer
115   *             : 0(NEGO_FAIL) :Negotiation failed
116   *-------------------------------------------------------------------------
117   * Notice     :
118   *""FUNC COMMENT END""*************************************************/
119   int phy_autonego(void)
120   {
121       unsigned short data0,data1;
122       int link = NEGO_FAIL;
123       volatile int t;
124       int i;
125
126       data0 = phy_reg_read(BASIC_MODE_CONTROL_REG);
```

```
127       data0 = data0 | 0x8000;
128       phy_reg_write(BASIC_MODE_CONTROL_REG, data0);
129
130       /* ---- Wait ---- */
131       for(i=0;i<1000;i++){
132         t=0x27C0;
133         while( --t){                  /* approx. 2.9-ms wait counting@32 MHz */
134             ;
135         }
136       }
137
138   #ifdef PHY_LOOP_BACK
139       data0 = phy_reg_read(BASIC_MODE_CONTROL_REG);
140       data0 = data0 & ~0x1000;
141       phy_reg_write(BASIC_MODE_CONTROL_REG, data0);
142       /* ---- approx. 10-ms wait ---- */
143       t=36000;
144       while( --t){                /* approx. 10-ms wait counting@32 MHz */
145         ;
146       }
147       data0 = phy_reg_read(BASIC_MODE_CONTROL_REG);
148       data0 = data0 | 0x4000;
149       phy_reg_write(BASIC_MODE_CONTROL_REG, data0);
150       /* ---- approx. 10-ms wait ---- */
151       t=36000;
152       while( --t){                /* approx. 10-ms wait counting@32 MHz */
153         ;
154       }
155       data0 = phy_reg_read(BASIC_MODE_CONTROL_REG);
156       data0 = data0 | 0x0100;
157       phy_reg_write(BASIC_MODE_CONTROL_REG, data0);
158       /* ---- approx. 10-ms wait ---- */
159       t=36000;
160       while( --t){                /* approx. 10-ms wait counting@32 MHz */
161         ;
162       }
163       data0 = phy_reg_read(BASIC_MODE_CONTROL_REG);
164       return FULL_TX;
165   //  return FULL_10M;
166   #endif
167
168       /* ==== Wait loop for end of automatic negotiation ==== */
169       for( i=0; i<500; i++){
170
171         /* ---- approx. 10-ms wait ---- */
172         t=36000;
173         while( --t){                  /* approx. 10-ms wait counting@32 MHz */
174             ;
175         }
176         data0 = phy_reg_read(BASIC_MODE_STATUS_REG);
177         if(data0 & 0x0020){     /* Automatic negotiation is completed */
178
179             /* ---- Result of automatic negotiation is obtained. ---- */
180             data1 = phy_reg_read(AN_LINK_PARTNER_ABILITY_REG);
181                                 /* Result of automatic negotiation AN_LINK_PARTNER_ABILITY_REG *
182                                  * If the device at the other end does not support automatic negotiation,
183                                  parallel detection is performed, but the result is indicated in this
184                                  register.                */
185             /* ---- Judgment of result -> break at the end of negotiation. ---- */
186             if( data1&0x0100 ){
187                 link = FULL_TX;
188             }
189             else if( data1&0x0080 ){
190                 link = HALF_TX;
```

```
            }
            else if( data1&0x0040 ){
                link = FULL_10M;
            }
            else if( data1&0x0020 ){
                link = HALF_10M;
            }
            else{
                link = NEGO_FAIL;
            }
            break;
        }
      }
      return link;
  }


  /*""FUNC COMMENT""*****************************************************************
  * Outline        : Reading of All MII Registers in the PHY-LSI
  *-------------------------------------------------------------------------------
  * Declaration    : static unsigned short phy_reg_read(unsigned short reg_addr)
  *-------------------------------------------------------------------------------
  * Description: Values of all MII registers in the PHY-LSI are obtained.
  *-------------------------------------------------------------------------------
  * Argument       : unsigned short reg_addr : I : Address of the PHY-LSI register
  *              from which a value is read
  *-------------------------------------------------------------------------------
  * Return Value   : Obtained register values
  *-------------------------------------------------------------------------------
  * Notice     :
  *""FUNC COMMENT END""*************************************************************/
  static unsigned short    phy_reg_read(unsigned short reg_addr)
  {
      unsigned short   data;

      mii_preamble();
      mii_cmd(reg_addr, PHY_READ);
      mii_z();
      mii_reg_read(&data);
      mii_z();

      return data;
  }

  /*""FUNC COMMENT""*****************************************************************
  * Outline        : Writing of All MII Registers in the PHY-LSI
  *-------------------------------------------------------------------------------
  * Declaration    : static void phy_reg_write(unsigned short reg_addr, unsigned short data )
  *-------------------------------------------------------------------------------
  * Description    : Values are set in all MII registers in the PHY-LSI.
  *-------------------------------------------------------------------------------
  * Argument       : unsigned short reg_addr : I : The PHY-LSI register address
  *              to which values are written
  *             : unsigned short data : I : Values set in registers of the PHY-LSI
  *-------------------------------------------------------------------------------
  * Return Value   : None
  *-------------------------------------------------------------------------------
  * Notice     :
  *""FUNC COMMENT END""*************************************************************/
  static void  phy_reg_write(unsigned short reg_addr, unsigned short data)
  {
      mii_preamble();
      mii_cmd(reg_addr, PHY_WRITE);
      mii_ta_10();
```

```
255       mii_reg_write(data);
256       mii_z();
257
258   }
259
260   /*""FUNC COMMENT""*************************************************************
261   * Outline        : Preparation for Accessing All MII Registers in the PHY-LSI
262   *-------------------------------------------------------------------------------
263   * Declaration    : static void mii_preamble(void)
264   *-------------------------------------------------------------------------------
265   * Description        : As advance preparation for access to PHY-LSI registers,
266   *            : one of 32 bits is output to the MII block.
267   *-------------------------------------------------------------------------------
268   * Argument       : None
269   *-------------------------------------------------------------------------------
270   * Return Value   : None
271   *-------------------------------------------------------------------------------
272   * Notice     :
273   *""FUNC COMMENT END""*********************************************************/
274   static void  mii_preamble(void)
275   {
276       short i;
277
278       i = 32;
279       while( i > 0 ) {
280         /* 1 is output to the MII (Media Independent Interface) block. */
281         mii_write_1();
282         i--;
283       }
284   }
285
286   /*""FUNC COMMENT""*************************************************************
287   * Outline        : Setting Modes of All MII Registers in the PHY-LSI
288   *-------------------------------------------------------------------------------
289   * Declaration    : static void mii_cmd(unsigned short reg_addr, int option)
290   *-------------------------------------------------------------------------------
291   * Description    : R/W mode of all MII registers in the PHY-LSI is set.
292   *-------------------------------------------------------------------------------
293   * Argument       : unsigned short reg_addr : I : Register address of the PHY-LSI
294   *            : int option : I : Specification of R/W mode
295   *-------------------------------------------------------------------------------
296   * Return Value   : None
297   *-------------------------------------------------------------------------------
298   * Notice     :
299   *""FUNC COMMENT END""*********************************************************/
300   static void  mii_cmd(unsigned short reg_addr, int option)
301   {
302       int   i;
303       unsigned short   data;
304
305       data = 0;
306       data = (PHY_ST << 14);              /* ST code     */
307       if( option == PHY_READ ) {
308         data |= (PHY_READ << 12);      /* OP code(RD) */
309       }
310       else {
311         data |= (PHY_WRITE << 12);     /* OP code(WT) */
312       }
313
314       data |= (PHY_ADDR0 << 7);       /* PHY Address */
315       data |= (reg_addr << 2);        /* Reg Address */
316
317       for(i=14; i>0; i--){
318         if( (data & 0x8000) == 0 ) {
```

```
319             mii_write_0();
320         }
321         else {
322             mii_write_1();
323         }
324         data <<= 1;
325       }
326   }
327
328   /*""FUNC COMMENT""*****************************************************************
329   * Outline        : Obtaining Value in All MII Registers in the PHY-LSI
330   *-------------------------------------------------------------------------------
331   * Declaration    : static void mii_reg_read (unsigned short *data)
332   *-------------------------------------------------------------------------------
333   * Description    : Acquires the values of all MII registers in the PHY-LSI,
334   *              one bit at a time.
335   *-------------------------------------------------------------------------------
336   * Argument       : unsigned short *data : O : Destination address for storage
337   *              of the acquired value
338   *-------------------------------------------------------------------------------
339   * Return Value   : None
340   *-------------------------------------------------------------------------------
341   * Notice     :
342   *""FUNC COMMENT END""*************************************************************/
343   static void  mii_reg_read(unsigned short *data)
344   {
345       int   i;
346       unsigned short   reg_data;
347
348   #ifdef    NO_WAIT
349       /* One-bit-unit data is read. */
350       reg_data = 0;
351       for(i=16; i>0; i--){
352         MAC0.PIR.LONG = 0x00000000;
353         MAC0.PIR.LONG = 0x00000001;
354         MAC0.PIR.LONG = 0x00000001;
355         reg_data <<= 1;
356         reg_data |= (MAC0.PIR.LONG & 0x00000008) >> 3;    /* MDI read*/
357         MAC0.PIR.LONG = 0x00000001;
358         MAC0.PIR.LONG = 0x00000000;
359       }
360       *data = reg_data;
361   #else
362       /* Data are read one bit at a time. */
363       int   j;
364       reg_data = 0;
365       for(i=16; i>0; i--){
366         for(j=0; j<3; j++) MAC0.PIR.LONG = 0x00000000;
367         for(j=0; j<3; j++) MAC0.PIR.LONG = 0x00000001;
368         for(j=0; j<3; j++) MAC0.PIR.LONG = 0x00000001;
369         reg_data <<= 1;
370         reg_data |= (MAC0.PIR.LONG & 0x00000008) >> 3;    /* MDI read*/
371         for(j=0; j<3; j++) MAC0.PIR.LONG = 0x00000001;
372         for(j=0; j<3; j++) MAC0.PIR.LONG = 0x00000000;
373       }
374       *data = reg_data;
375   #endif
376   }
377
378   /*""FUNC COMMENT""*****************************************************************
379   * Outline        : Setting Value in All MII Registers in the PHY -LSI
380   *-------------------------------------------------------------------------------
381   * Declaration    : static void mii_reg_write (unsigned short data)
382   *-------------------------------------------------------------------------------
```

```
383   * Description    : One-bit-unit data is set for value in all MII registers in the PHY-LSI.
384   *------------------------------------------------------------------------------
385   * Argument       : unsigned short data : I : Value set in a register
386   *------------------------------------------------------------------------------
387   * Return Value   : None
388   *------------------------------------------------------------------------------
389   * Notice     :
390   *""FUNC COMMENT END""*************************************************************/
391   static void  mii_reg_write(unsigned short data)
392   {
393       int   i;
394
395       /* One-bit-unit data is written. */
396       for(i=16; i>0; i--){
397         if( (data & 0x8000) == 0 ) {
398             mii_write_0();
399         }
400         else {
401             mii_write_1();
402         }
403         data <<= 1;
404       }
405   }
406
407   /*""FUNC COMMENT""*************************************************************
408   * Outline        : Release of the MII Bus
409   *------------------------------------------------------------------------------
410   * Declaration    : static void mii_z(void)
411   *------------------------------------------------------------------------------
412   * Description    : Settings to release the bus from access to the MII.
413   *------------------------------------------------------------------------------
414   * Argument       : None
415   *------------------------------------------------------------------------------
416   * Return Value   : None
417   *------------------------------------------------------------------------------
418   * Notice     :
419   *""FUNC COMMENT END""*************************************************************/
420   static void  mii_z(void)
421   {
422   #ifdef    NO_WAIT
423       MAC0.PIR.LONG = 0x00000000;
424       MAC0.PIR.LONG = 0x00000001;
425       MAC0.PIR.LONG = 0x00000001;
426       MAC0.PIR.LONG = 0x00000001;
427       MAC0.PIR.LONG = 0x00000000;
428   #else
429       int   j;
430       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000000;
431       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000001;
432       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000001;
433       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000001;
434       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000000;
435   #endif
436   }
437
438   /*""FUNC COMMENT""*************************************************************
439   * Outline        : Output of the MII TA (1 or 0) Bit
440   *------------------------------------------------------------------------------
441   * Declaration    : static void mii_ta_10(void)
442   *------------------------------------------------------------------------------
443   * Description        : 1 or 0 is output to the MII.
444   *------------------------------------------------------------------------------
445   * Argument       : None
446   *------------------------------------------------------------------------------
```

```
447   * Return Value   : None
448   *-------------------------------------------------------------------------
449   * Notice     :
450   *""FUNC COMMENT END""*****************************************************/
451   static void  mii_ta_10(void)
452   {
453       mii_write_1();
454       mii_write_0();
455   }
456
457   /*""FUNC COMMENT""*************************************************************
458   * Outline        : Output of One Bit (1) to the MII
459   *-------------------------------------------------------------------------
460   * Declaration    : static void mii_write_1(void)
461   *-------------------------------------------------------------------------
462   * Description    : 1 is output to the MII.
463   *-------------------------------------------------------------------------
464   * Argument       : None
465   *-------------------------------------------------------------------------
466   * Return Value   : None
467   *-------------------------------------------------------------------------
468   * Notice     :
469   *""FUNC COMMENT END""*****************************************************/
470   static void  mii_write_1(void)
471   {
472   #ifdef   NO_WAIT
473       MAC0.PIR.LONG = 0x00000006;
474       MAC0.PIR.LONG = 0x00000007;
475       MAC0.PIR.LONG = 0x00000007;
476       MAC0.PIR.LONG = 0x00000007;
477       MAC0.PIR.LONG = 0x00000006;
478   #else
479       int   j;
480       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000006;
481       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000007;
482       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000007;
483       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000007;
484       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000006;
485   #endif
486   }
487
488   /*""FUNC COMMENT""*************************************************************
489   * Outline        : Output of One Bit (0) to the MII
490   *-------------------------------------------------------------------------
491   * Declaration    : static void mii_write_0(void)
492   *-------------------------------------------------------------------------
493   * Description    : 0 is output to the MII.
494   *-------------------------------------------------------------------------
495   * Argument       : None
496   *-------------------------------------------------------------------------
497   * Return Value   : None
498   *-------------------------------------------------------------------------
499   * Notice     :
500   *""FUNC COMMENT END""*****************************************************/
501   static void  mii_write_0(void)
502   {
503   #ifdef   NO_WAIT
504       MAC0.PIR.LONG = 0x00000002;
505       MAC0.PIR.LONG = 0x00000003;
506       MAC0.PIR.LONG = 0x00000003;
507       MAC0.PIR.LONG = 0x00000003;
508       MAC0.PIR.LONG = 0x00000002;
509   #else
510       int   j;
```

```
511       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000002;
512       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000003;
513       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000003;
514       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000003;
515       for(j=0; j<3; j++)   MAC0.PIR.LONG = 0x00000002;
516   #endif
517   }
518
519   /* End of File */
```

3.5　サンプルプログラムリスト "phy.h"

```
1     /***********************************************************************************
2     * DISCLAIMER
3     * Please refer to http://www.renesas.com/disclaimer
4     ***********************************************************************************
5       Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
6     ***********************************************************************************
7     * File Name    : phy.h
8     * Version      : 1.00
9     * Device(s)    : H8S/2472
10    * Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
11    * OS          : None
12    * H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
13    * Description  : Header program of the PHY
14    * Limitations  : None
15    ***********************************************************************************
16    * History : DD.MM.YYYY Version Description
17    *         : 19.02.2007 1.00    First Release
18    ***********************************************************************************/
19
20    #ifndef PHY_H
21    #define PHY_H
22
23    /***********************************************************************************
24    Includes   <System Includes> , "Project Includes"
25    ***********************************************************************************/
26
27    /***********************************************************************************
28    Typedef definitions
29    ***********************************************************************************/
30
31    /***********************************************************************************
32    Macro definitions
33    ***********************************************************************************/
34    /* Link result */
35    #define NEGO_FAIL            0
36    #define HALF_10M         1
37    #define FULL_10M         2
38    #define HALF_TX              3
39    #define FULL_TX              4
40
41    /***********************************************************************************
42    Variable Externs
43    ***********************************************************************************/
44
45    /***********************************************************************************
46    Functions Prototypes
47    ***********************************************************************************/
48    int phy_autonego(void);
49
50
51
52
53    #endif /* PHY_H */
```

3.6　サンプルプログラムリスト "iodefine2472.h"

```
/***********************************************************************************
* DISCLAIMER
* Please refer to http://www.renesas.com/disclaimer
************************************************************************************
  Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
************************************************************************************
* File Name    : iodefine2472.h
* Version      : 1.01
* Device(s)    : H8S/2472
* Tool-Chain   : HEW, H8S,H8/300 Standard Toolchain (V.6.2.2.0)
* OS           : None
* H/W Platform : R0K402472D000BR,R0K402472D001BR,R0K402472D002BR
* Description  : register definition
* Limitations  : None
************************************************************************************
* History : DD.MM.YYYY Version Description
*         : 18.10.2007 1.00    First Release
*         : 19.04.2010 1.01    Unnecessary code deleted
***********************************************************************************/

#ifndef IODEFINE2472_H
#define IODEFINE2472_H

/***********************************************************************************
Includes   <System Includes> , "Project Includes"
***********************************************************************************/

/***********************************************************************************
Typedef definitions
***********************************************************************************/
struct st_ether {                                   /* struct ETHER */
      union {                                      /* ECMR        */
            unsigned long LONG;                     /*  Long Access */
            struct {                               /*  Word Access */
                  unsigned short H;                 /*   High      */
                  unsigned short L;                 /*   Low       */
                  } WORD;                          /*            */
            struct {                               /*  Bit Access  */
                  unsigned long :12;                /*            */
                  unsigned long ZPF:1;              /*   ZPF       */
                  unsigned long PFR:1;              /*   PFR       */
                  unsigned long RXF:1;              /*   RXF       */
                  unsigned long TXF:1;              /*   TXF       */
                  unsigned long :3;                 /*            */
                  unsigned long PRCEF:1;            /*   PRCEF     */
                  unsigned long :2;                 /*            */
                  unsigned long MPDE:1;             /*   MPDE      */
                  unsigned long :2;                 /*            */
                  unsigned long RE:1;               /*   RE        */
                  unsigned long TE:1;               /*   TE        */
                  unsigned long :1;                 /*            */
                  unsigned long ILB:1;              /*   ILB       */
                  unsigned long ELB:1;              /*   ELB       */
                  unsigned long DM:1;               /*   DM        */
                  unsigned long PRM:1;              /*   PRM       */
                  } BIT;                           /*            */
            } ECMR;                                /*            */
      union {                                      /* ECSR        */
            unsigned long LONG;                     /*  Long Access */
            struct {                               /*  Word Access */
                  unsigned short H;                 /*   High      */
                  unsigned short L;                 /*   Low       */
```

```
63                    } WORD;                          /*           */
64               struct {                              /*  Bit Access  */
65                    unsigned long :27;                /*           */
66                    unsigned long PSRTO:1;            /*   PSRTO     */
67                    unsigned long :1;                 /*           */
68                    unsigned long LCHNG:1;            /*   LCHNG     */
69                    unsigned long MPD:1;              /*   MPD       */
70                    unsigned long ICD:1;              /*   ICD       */
71                    } BIT;                           /*           */
72               } ECSR;                               /*           */
73          union {                                    /* ECSIPR      */
74               unsigned long LONG;                    /*  Long Access */
75               struct {                              /*  Word Access */
76                    unsigned short H;                 /*   High      */
77                    unsigned short L;                 /*   Low       */
78                    } WORD;                          /*           */
79               struct {                              /*  Bit Access  */
80                    unsigned long :27;                /*           */
81                    unsigned long PSRTOIP:1;          /*   PSRTOIP   */
82                    unsigned long :1;                 /*           */
83                    unsigned long LCHNGIP:1;          /*   LCHNGIP   */
84                    unsigned long MPDIP:1;            /*   MPDIP     */
85                    unsigned long ICDIP:1;            /*   ICDIP     */
86                    } BIT;                           /*           */
87               } ECSIPR;                             /*           */
88          union {                                    /* PIR         */
89               unsigned long LONG;                    /*  Long Access */
90               struct {                              /*  Word Access */
91                    unsigned short H;                 /*   High      */
92                    unsigned short L;                 /*   Low       */
93                    } WORD;                          /*           */
94               struct {                              /*  Bit Access  */
95                    unsigned long :28;                /*           */
96                    unsigned long MDI:1;              /*   MDI       */
97                    unsigned long MDO:1;              /*   MDO       */
98                    unsigned long MMD:1;              /*   MMD       */
99                    unsigned long MDC:1;              /*   MDC       */
100                   } BIT;                           /*           */
101              } PIR;                                /*           */
102         union {                                    /* MAHR        */
103              unsigned long LONG;                    /*  Long Access */
104              struct {                              /*  Word Access */
105                   unsigned short H;                 /*   High      */
106                   unsigned short L;                 /*   Low       */
107                   } WORD;                          /*           */
108              struct {                              /*  Bit Access  */
109                   unsigned long MA47:1;             /*   MA47      */
110                   unsigned long MA46:1;             /*   MA46      */
111                   unsigned long MA45:1;             /*   MA45      */
112                   unsigned long MA44:1;             /*   MA44      */
113                   unsigned long MA43:1;             /*   MA43      */
114                   unsigned long MA42:1;             /*   MA42      */
115                   unsigned long MA41:1;             /*   MA41      */
116                   unsigned long MA40:1;             /*   MA40      */
117                   unsigned long MA39:1;             /*   MA39      */
118                   unsigned long MA38:1;             /*   MA38      */
119                   unsigned long MA37:1;             /*   MA37      */
120                   unsigned long MA36:1;             /*   MA36      */
121                   unsigned long MA35:1;             /*   MA35      */
122                   unsigned long MA34:1;             /*   MA34      */
123                   unsigned long MA33:1;             /*   MA33      */
124                   unsigned long MA32:1;             /*   MA32      */
125                   unsigned long MA31:1;             /*   MA31      */
126                   unsigned long MA30:1;             /*   MA30      */
```

```
127                   unsigned long MA29:1;           /*   MA29      */
128                   unsigned long MA28:1;           /*   MA28      */
129                   unsigned long MA27:1;           /*   MA27      */
130                   unsigned long MA26:1;           /*   MA26      */
131                   unsigned long MA25:1;           /*   MA25      */
132                   unsigned long MA24:1;           /*   MA24      */
133                   unsigned long MA23:1;           /*   MA23      */
134                   unsigned long MA22:1;           /*   MA22      */
135                   unsigned long MA21:1;           /*   MA21      */
136                   unsigned long MA20:1;           /*   MA20      */
137                   unsigned long MA19:1;           /*   MA19      */
138                   unsigned long MA18:1;           /*   MA18      */
139                   unsigned long MA17:1;           /*   MA17      */
140                   unsigned long MA16:1;           /*   MA16      */
141                   } BIT;                         /*             */
142             } MAHR;                              /*             */
143       union {                                    /* MALR        */
144             unsigned long LONG;                   /*  Long Access */
145             struct {                             /*  Bit Access  */
146                   unsigned long :16;              /*              */
147                   unsigned long MA:16;            /*   MA         */
148                   } BIT;                         /*             */
149             } MALR;                              /*             */
150       union {                                    /* RFLR        */
151             unsigned long LONG;                   /*  Long Access */
152             struct {                             /*  Word Access */
153                   unsigned short H;               /*   High      */
154                   unsigned short L;               /*   Low       */
155                   } WORD;                        /*             */
156             struct {                             /*  Bit Access  */
157                   unsigned long :20;              /*              */
158                   unsigned long RFL:12;           /*   RFL       */
159                   } BIT;                         /*             */
160             } RFLR;                              /*             */
161       union {                                    /* PSR         */
162             unsigned long LONG;                   /*  Long Access */
163             struct {                             /*  Word Access */
164                   unsigned short H;               /*   High      */
165                   unsigned short L;               /*   Low       */
166                   } WORD;                        /*             */
167             struct {                             /*  Bit Access  */
168                   unsigned long :31;              /*              */
169                   unsigned long LMON:1;           /*   LMON      */
170                   } BIT;                         /*             */
171             } PSR;                               /*             */
172       unsigned long TROCR;                        /* TROCR       */
173       unsigned long CDCR;                         /* CDCR        */
174       unsigned long LCCR;                         /* LCCR        */
175       unsigned long CNDCR;                        /* CNDCR       */
176       unsigned char wk0[4];                       /*             */
177       unsigned long CEFCR;                        /* CEFCR       */
178       unsigned long FRECR;                        /* FRECR       */
179       unsigned long TSFRCR;                       /* TSFRCR      */
180       unsigned long TLFRCR;                       /* TLFRCR      */
181       unsigned long RFCR;                         /* RFCR        */
182       unsigned long MAFCR;                        /* MAFCR       */
183       unsigned char wk1[8];                       /*             */
184       union {                                    /* IPGR        */
185             unsigned long LONG;                   /*  Long Access */
186             struct {                             /*  Word Access */
187                   unsigned short H;               /*   High      */
188                   unsigned short L;               /*   Low       */
189                   } WORD;                        /*             */
190             struct {                             /*  Bit Access  */
```

```
191                   unsigned long :27;              /*            */
192                   unsigned long IPG:5;            /*   IPG      */
193                   } BIT;                         /*            */
194             } IPGR;                              /*            */
195         union {                                  /* APR        */
196             unsigned long LONG;                  /*  Long Access */
197             struct {                             /*  Bit Access  */
198                   unsigned long :16;              /*            */
199                   unsigned long AP:16;            /*   AP       */
200                   } BIT;                         /*            */
201             } APR;                               /*            */
202         union {                                  /* MPR        */
203             unsigned long LONG;                  /*  Long Access */
204             struct {                             /*  Bit Access  */
205                   unsigned long :16;              /*            */
206                   unsigned long MP:16;            /*   MP       */
207                   } BIT;                         /*            */
208             } MPR;                               /*            */
209         unsigned char wk2[4];                     /*            */
210         union {                                  /* TPAUSER    */
211             unsigned long LONG;                  /*  Long Access */
212             struct {                             /*  Bit Access  */
213                   unsigned long :16;              /*            */
214                   unsigned long TPAUSE:16;        /*   TPAUSE    */
215                   } BIT;                         /*            */
216             } TPAUSER;                           /*            */
217   };                                            /*            */
218   struct st_edmac {                               /* struct EDMAC */
219         union {                                  /* EDMR       */
220             unsigned long LONG;                  /*  Long Access */
221             struct {                             /*  Word Access */
222                   unsigned short H;               /*   High      */
223                   unsigned short L;               /*   Low       */
224                   } WORD;                        /*            */
225             struct {                             /*  Bit Access  */
226                   unsigned long :25;              /*            */
227                   unsigned long DE:1;             /*   DE       */
228                   unsigned long DL:2;             /*   DL       */
229                   unsigned long :3;               /*            */
230                   unsigned long SWR:1;            /*   SWR      */
231                   } BIT;                         /*            */
232             } EDMR;                              /*            */
233         union {                                  /* EDTRR      */
234             unsigned long LONG;                  /*  Long Access */
235             struct {                             /*  Bit Access  */
236                   unsigned long :31;              /*            */
237                   unsigned long TR:1;             /*   TR       */
238                   } BIT;                         /*            */
239             } EDTRR;                             /*            */
240         union {                                  /* EDRRR      */
241             unsigned long LONG;                  /*  Long Access */
242             struct {                             /*  Bit Access  */
243                   unsigned long :31;              /*            */
244                   unsigned long RR:1;             /*   RR       */
245                   } BIT;                         /*            */
246             } EDRRR;                             /*            */
247         void          *TDLAR;                    /* TDLAR      */
248         void          *RDLAR;                    /* RDLAR      */
249         union {                                  /* EESR       */
250             unsigned long LONG;                  /*  Long Access */
251             struct {                             /*  Word Access */
252                   unsigned short H;               /*   High      */
253                   unsigned short L;               /*   Low       */
254                   } WORD;                        /*            */
```

```
255             struct {                            /*  Bit Access  */
256                   unsigned long :1;               /*              */
257                   unsigned long TWB:1;            /*   TWB        */
258                   unsigned long :3;               /*              */
259                   unsigned long TABT:1;           /*   TABT       */
260                   unsigned long RABT:1;           /*   RABT       */
261                   unsigned long RFCOF:1;          /*   RFCOF      */
262                   unsigned long ADE:1;            /*   ADE        */
263                   unsigned long ECI:1;            /*   ECI        */
264                   unsigned long TC:1;             /*   TC         */
265                   unsigned long TDE:1;            /*   TDE        */
266                   unsigned long TFUF:1;           /*   TFUF       */
267                   unsigned long FR:1;             /*   FR         */
268                   unsigned long RDE:1;            /*   RDE        */
269                   unsigned long RFOF:1;           /*   RFOF       */
270                   unsigned long :4;               /*              */
271                   unsigned long CND:1;            /*   CND        */
272                   unsigned long DLC:1;            /*   DLC        */
273                   unsigned long CD:1;             /*   CD         */
274                   unsigned long TRO:1;            /*   TRO        */
275                   unsigned long RMAF:1;           /*   RMAF       */
276                   unsigned long :2;               /*              */
277                   unsigned long RRF:1;            /*   RRF        */
278                   unsigned long RTLF:1;           /*   RTLF       */
279                   unsigned long RTSF:1;           /*   RTSF       */
280                   unsigned long PRE:1;            /*   PRE        */
281                   unsigned long CERF:1;           /*   CERF       */
282                   } BIT;                         /*              */
283             } EESR;                               /*              */
284       union {                                     /* EESIPR       */
285             unsigned long LONG;                     /*  Long Access */
286             struct {                              /*  Word Access */
287                   unsigned short H;               /*   High       */
288                   unsigned short L;               /*   Low        */
289                   } WORD;                         /*              */
290             struct {                              /*  Bit Access  */
291                   unsigned long :1;               /*              */
292                   unsigned long TWBIP:1;          /*   TWBIP      */
293                   unsigned long :3;               /*              */
294                   unsigned long TABTIP:1;         /*   TABTIP     */
295                   unsigned long RABTIP:1;         /*   RABTIP     */
296                   unsigned long RFCOFIP:1;        /*   RFCOFIP    */
297                   unsigned long ADEIP:1;          /*   ADEIP      */
298                   unsigned long ECIIP:1;          /*   ECIIP      */
299                   unsigned long TCIP:1;           /*   TCIP       */
300                   unsigned long TDEIP:1;          /*   TDEIP      */
301                   unsigned long TFUFIP:1;         /*   TFUFIP     */
302                   unsigned long FRIP:1;           /*   FRIP       */
303                   unsigned long RDEIP:1;          /*   RDEIP      */
304                   unsigned long RFOFIP:1;         /*   RFOFIP     */
305                   unsigned long :4;               /*              */
306                   unsigned long CNDIP:1;          /*   CNDIP      */
307                   unsigned long DLCIP:1;          /*   DLCIP      */
308                   unsigned long CDIP:1;           /*   CDIP       */
309                   unsigned long TROIP:1;          /*   TROIP      */
310                   unsigned long RMAFIP:1;         /*   RMAFIP     */
311                   unsigned long :2;               /*              */
312                   unsigned long RRFIP:1;          /*   RRFIP      */
313                   unsigned long RTLFIP:1;         /*   RTLFIP     */
314                   unsigned long RTSFIP:1;         /*   RTSFIP     */
315                   unsigned long PREIP:1;          /*   PREIP      */
316                   unsigned long CERFIP:1;         /*   CERFIP     */
317                   } BIT;                         /*              */
318             } EESIPR;                             /*              */
```

```
319         union {                                 /* TRSCER      */
320              unsigned long LONG;                  /*  Long Access */
321              struct {                           /*  Word Access */
322                   unsigned short H;               /*   High      */
323                   unsigned short L;               /*   Low       */
324                   } WORD;                        /*            */
325              struct {                           /*  Bit Access  */
326                   unsigned long :20;              /*            */
327                   unsigned long CNDCE:1;           /*   CNDCE     */
328                   unsigned long DLCCE:1;           /*   DLCCE     */
329                   unsigned long CDCE:1;            /*   CDCE      */
330                   unsigned long TROCE:1;           /*   TROCE     */
331                   unsigned long RMAFCE:1;          /*   RMAFCE    */
332                   unsigned long :2;               /*            */
333                   unsigned long RRFCE:1;           /*   RRFCE     */
334                   unsigned long RTLFCE:1;          /*   RTLFCE    */
335                   unsigned long RTSFCE:1;          /*   RTSFCE    */
336                   unsigned long PRECE:1;           /*   PRECE     */
337                   unsigned long CERFCE:1;          /*   CERFCE    */
338                   } BIT;                         /*            */
339              } TRSCER;                           /*            */
340         union {                                 /* RMFCR       */
341              unsigned long LONG;                  /*  Long Access */
342              struct {                           /*  Bit Access  */
343                   unsigned long :16;              /*            */
344                   unsigned long MFC:16;            /*   MFC       */
345                   } BIT;                         /*            */
346              } RMFCR;                            /*            */
347         union {                                 /* TFTR        */
348              unsigned long LONG;                  /*  Long Access */
349              struct {                           /*  Bit Access  */
350                   unsigned long :21;              /*            */
351                   unsigned long TFT:11;            /*   TFT       */
352                   } BIT;                         /*            */
353              } TFTR;                             /*            */
354         union {                                 /* FDR         */
355              unsigned long LONG;                  /*  Long Access */
356              struct {                           /*  Word Access */
357                   unsigned short H;               /*   High      */
358                   unsigned short L;               /*   Low       */
359                   } WORD;                        /*            */
360              struct {                           /*  Bit Access  */
361                   unsigned long :21;              /*            */
362                   unsigned long TFD:3;             /*   TFD       */
363                   unsigned long :5;               /*            */
364                   unsigned long RFD:3;             /*   RFD       */
365                   } BIT;                         /*            */
366              } FDR;                              /*            */
367         union {                                 /* RMCR        */
368              unsigned long LONG;                  /*  Long Access */
369              struct {                           /*  Bit Access  */
370                   unsigned long :31;              /*            */
371                   unsigned long RNC:1;             /*   RNC       */
372                   } BIT;                         /*            */
373              } RMCR;                             /*            */
374         unsigned char wk0[4];                     /*            */
375         union {                                 /* FCFTR       */
376              unsigned long LONG;                  /*  Long Access */
377              struct {                           /*  Word Access */
378                   unsigned short H;               /*   High      */
379                   unsigned short L;               /*   Low       */
380                   } WORD;                        /*            */
381              struct {                           /*  Bit Access  */
382                   unsigned long :13;              /*            */
```

```
383                   unsigned long RFF:3;            /*   RFF       */
384                   unsigned long :13;              /*             */
385                   unsigned long RFD:3;            /*   RFD       */
386                   } BIT;                         /*             */
387             } FCFTR;                             /*             */
388         unsigned char wk1[8];                      /*             */
389         unsigned long RBWAR;                       /* RBWAR       */
390         unsigned long RDFAR;                       /* RDFAR       */
391         unsigned char wk2[4];                      /*             */
392         unsigned long TBRAR;                       /* TBRAR       */
393         unsigned long TDFAR;                       /* TDFAR       */
394         union {                                   /* ECBRR       */
395               unsigned char BYTE;                  /*  Byte Access */
396               struct {                            /*  Bit Access  */
397                     unsigned char :7;              /*             */
398                     unsigned char RTM:1;           /*   RTM       */
399                     } BIT;                        /*             */
400               } ECBRR;                            /*             */
401   };                                              /*             */
402   struct st_system {                                /* struct SYSTEM */
403         union {                                   /* SUBMSTPAH    */
404               unsigned char BYTE;                  /*  Byte Access */
405               struct {                            /*  Bit Access  */
406                     unsigned char SMSTPA15:1;       /*   SMSTPA15   */
407                     unsigned char EtherC:1;         /*   EtherC     */
408                     unsigned char EDMAC:1;          /*   EDMAC      */
409                     unsigned char USB:1;            /*   USB        */
410                     unsigned char SMSTPA11:1;       /*   SMSTPA11   */
411                     unsigned char SMSTPA10:1;       /*   SMSTPA10   */
412                     unsigned char SMSTPA9:1;        /*   SMSTPA9    */
413                     unsigned char SMSTPA8:1;        /*   SMSTPA8    */
414                     } BIT;                        /*             */
415               } SUBMSTPAH;                         /*             */
416         union {                                   /* SUBMSTPAL    */
417               unsigned char BYTE;                  /*  Byte Access */
418               struct {                            /*  Bit Access  */
419                     unsigned char SMSTPA7:1;        /*   SMSTPA7    */
420                     unsigned char SMSTPA6:1;        /*   SMSTPA6    */
421                     unsigned char SMSTPA5:1;        /*   SMSTPA5    */
422                     unsigned char PECI:1;           /*   PECI       */
423                     unsigned char SCIF:1;           /*   SCIF       */
424                     unsigned char SSU:1;            /*   SSU        */
425                     unsigned char LPC:1;            /*   LPC        */
426                     unsigned char SMSTPA0:1;        /*   SMSTPA0    */
427                     } BIT;                        /*             */
428               } SUBMSTPAL;                         /*             */
429         union {                                   /* SUBMSTPBH    */
430               unsigned char BYTE;                  /*  Byte Access */
431               struct {                            /*  Bit Access  */
432                     unsigned char SMSTPB15:1;       /*   SMSTPB15   */
433                     unsigned char EtherC:1;         /*   EtherC     */
434                     unsigned char EDMAC:1;          /*   EDMAC      */
435                     unsigned char USB:1;            /*   USB        */
436                     unsigned char SMSTPB11:1;       /*   SMSTPB11   */
437                     unsigned char SMSTPB10:1;       /*   SMSTPB10   */
438                     unsigned char SMSTPB9:1;        /*   SMSTPB9    */
439                     unsigned char SMSTPB8:1;        /*   SMSTPB8    */
440                     } BIT;                        /*             */
441               } SUBMSTPBH;                         /*             */
442         union {                                   /* SUBMSTPBL    */
443               unsigned char BYTE;                  /*  Byte Access */
444               struct {                            /*  Bit Access  */
445                     unsigned char SMSTPB7:1;        /*   SMSTPB7    */
446                     unsigned char SMSTPB6:1;        /*   SMSTPB6    */
```

```
447                   unsigned char SMSTPB5:1;          /*   SMSTPB5    */
448                   unsigned char PECI:1;             /*   PECI       */
449                   unsigned char SCIF:1;             /*   SCIF       */
450                   unsigned char SSU:1;              /*   SSU        */
451                   unsigned char LPC:1;              /*   LPC        */
452                   unsigned char SMSTPB0:1;          /*   SMSTPB0    */
453                   } BIT;                           /*              */
454              } SUBMSTPBL;                           /*              */
455         unsigned char wk0[3];                       /*              */
456         unsigned char MSTPCRA;                      /* MSTPCRA      */
457         unsigned char wk1[318];                     /*              */
458         union {                                    /* PCSR         */
459              unsigned char BYTE;                    /*  Byte Access */
460              struct {                              /*  Bit Access  */
461                   unsigned char PWCKX1B:1;          /*   PWCKX1B    */
462                   unsigned char PWCKX1A:1;          /*   PWCKX1A    */
463                   unsigned char PWCKX0B:1;          /*   PWCKX0B    */
464                   unsigned char PWCKX0A:1;          /*   PWCKX0A    */
465                   unsigned char PWCKX1C:1;          /*   PWCKX1C    */
466                   unsigned char PWCKB:1;            /*   PWCKB      */
467                   unsigned char PWCKA:1;            /*   PWCKA      */
468                   unsigned char PWCKX0C:1;          /*   PWCKX0C    */
469                   } BIT;                           /*              */
470              } PCSR;                               /*              */
471         union {                                    /* SYSCR2       */
472              unsigned char BYTE;                    /*  Byte Access */
473              struct {                              /*  Bit Access  */
474                   unsigned char :4;                /*              */
475                   unsigned char ADMXE:1;           /*   ADMXE      */
476                   unsigned char :3;                /*              */
477                   } BIT;                           /*              */
478              } SYSCR2;                             /*              */
479         union {                                    /* SBYCR        */
480              unsigned char BYTE;                    /*  Byte Access */
481              struct {                              /*  Bit Access  */
482                   unsigned char SSBY:1;            /*   SSBY       */
483                   unsigned char STS2:1;            /*   STS2       */
484                   unsigned char STS0:2;            /*   STS0       */
485                   unsigned char DTSPEED:1;         /*   DTSPEED    */
486                   unsigned char SCK:3;             /*   SCK        */
487                   } BIT;                           /*              */
488              } SBYCR;                              /*              */
489         union {                                    /* LPWRCR       */
490              unsigned char BYTE;                    /*  Byte Access */
491              struct {                              /*  Bit Access  */
492                   unsigned char DTON:1;            /*   DTON       */
493                   unsigned char LSON:1;            /*   LSON       */
494                   unsigned char NESEL:1;           /*   NESEL      */
495                   unsigned char EXCLE:1;           /*   EXCLE      */
496                   unsigned char :1;                /*              */
497                   unsigned char PNCCS:1;           /*   PNCCS      */
498                   unsigned char PNCAH:1;           /*   PNCAH      */
499                   unsigned char :1;                /*              */
500                   } BIT;                           /*              */
501              } LPWRCR;                             /*              */
502         union {                                    /* MSTPCRH      */
503              unsigned char BYTE;                    /*  Byte Access */
504              struct {                              /*  Bit Access  */
505                   unsigned char MSTP15:1;          /*   MSTP15     */
506                   unsigned char MSTP14:1;          /*   MSTP14     */
507                   unsigned char MSTP13:1;          /*   MSTP13     */
508                   unsigned char MSTP12:1;          /*   MSTP12     */
509                   unsigned char MSTP11:1;          /*   MSTP11     */
510                   unsigned char MSTP10:1;          /*   MSTP10     */
```

```
511                   unsigned char MSTP9:1;          /*   MSTP9     */
512                   unsigned char MSTP8:1;          /*   MSTP8     */
513                   } BIT;                         /*            */
514             } MSTPCRH;                           /*            */
515       unsigned char MSTPCRL;                     /* MSTPCRL     */
516       unsigned char wk2[59];                     /*             */
517       union {                                    /* STCR        */
518             unsigned char BYTE;                  /*  Byte Access */
519             struct {                             /*  Bit Access  */
520                   unsigned char IICX:3;          /*   IICX      */
521                   unsigned char IICE:1;          /*   IICE      */
522                   unsigned char FLSHE:1;         /*   FLSHE     */
523                   unsigned char :1;              /*             */
524                   unsigned char ICKS:2;          /*   ICKS      */
525                   } BIT;                         /*            */
526             } STCR;                              /*            */
527       union {                                    /* SYSCR       */
528             unsigned char BYTE;                  /*  Byte Access */
529             struct {                             /*  Bit Access  */
530                   unsigned char CS256E:1;        /*   CS256E    */
531                   unsigned char IOSE:1;          /*   IOSE      */
532                   unsigned char INTM:2;          /*   INTM      */
533                   unsigned char XRST:1;          /*   XRST      */
534                   unsigned char NMIEG:1;         /*   NMIEG     */
535                   unsigned char :1;              /*             */
536                   unsigned char RAME:1;          /*   RAME      */
537                   } BIT;                         /*            */
538             } SYSCR;                             /*            */
539       union {                                    /* MDCR        */
540             unsigned char BYTE;                  /*  Byte Access */
541             struct {                             /*  Bit Access  */
542                   unsigned char EXPE:1;          /*   EXPE      */
543                   unsigned char :4;              /*             */
544                   unsigned char MDS:3;           /*   MDS       */
545                   } BIT;                         /*            */
546             } MDCR;                              /*            */
547   };                                             /*            */
548   struct st_int {                                /* struct INT  */
549       union {                                    /* ICRD        */
550             unsigned char BYTE;                  /*  Byte Access */
551             struct {                             /*  Bit Access  */
552                   unsigned char ICRD7:1;         /*   ICRD7     */
553                   unsigned char ICRD6:1;         /*   ICRD6     */
554                   unsigned char ICRD5:1;         /*   ICRD6     */
555                   unsigned char :5;              /*             */
556                   } BIT;                         /*            */
557             } ICRD;                              /*            */
558       unsigned char ICRA;                        /* ICRA        */
559       union {                                    /* ICRB        */
560             unsigned char BYTE;                  /*  Byte Access */
561             struct {                             /*  Bit Access  */
562                   unsigned char ICRB7:1;         /*   ICRB7     */
563                   unsigned char ICRB6:1;         /*   ICRB6     */
564                   unsigned char :1;              /*             */
565                   unsigned char ICRB:5;          /*   ICRB      */
566                   } BIT;                         /*            */
567             } ICRB;                              /*            */
568       union {                                    /* ICRC        */
569             unsigned char BYTE;                  /*  Byte Access */
570             struct {                             /*  Bit Access  */
571                   unsigned char ICRC7:1;         /*   ICRC7     */
572                   unsigned char ICRC6:1;         /*   ICRC6     */
573                   unsigned char ICRC5:1;         /*   ICRC5     */
574                   unsigned char ICRC4:1;         /*   ICRC4     */
```

```
575                   unsigned char ICRC3:1;           /*   ICRC3     */
576                   unsigned char ICRC2:1;           /*   ICRC2     */
577                   unsigned char ICRC1:1;           /*   ICRC1     */
578                   unsigned char :1;               /*            */
579                   } BIT;                          /*            */
580             } ICRC;                               /*            */
581         union {                                   /* ISR        */
582             unsigned char BYTE;                     /*  Byte Access */
583             struct {                              /*  Bit Access  */
584                   unsigned char IRQ7F:1;           /*   IRQ7F     */
585                   unsigned char IRQ6F:1;           /*   IRQ6F     */
586                   unsigned char IRQ5F:1;           /*   IRQ5F     */
587                   unsigned char IRQ4F:1;           /*   IRQ4F     */
588                   unsigned char IRQ3F:1;           /*   IRQ3F     */
589                   unsigned char IRQ2F:1;           /*   IRQ2F     */
590                   unsigned char IRQ1F:1;           /*   IRQ1F     */
591                   unsigned char IRQ0F:1;           /*   IRQ0F     */
592                   } BIT;                          /*            */
593             } ISR;                                /*            */
594         union {                                   /* ISCRH       */
595             unsigned char BYTE;                     /*  Byte Access */
596             struct {                              /*  Bit Access  */
597                   unsigned char IRQ7SCB:1;         /*   IRQ7SCB    */
598                   unsigned char IRQ7SCA:1;         /*   IRQ7SCA    */
599                   unsigned char IRQ6SCB:1;         /*   IRQ6SCB    */
600                   unsigned char IRQ6SCA:1;         /*   IRQ6SCA    */
601                   unsigned char IRQ5SCB:1;         /*   IRQ5SCB    */
602                   unsigned char IRQ5SCA:1;         /*   IRQ5SCA    */
603                   unsigned char IRQ4SCB:1;         /*   IRQ4SCB    */
604                   unsigned char IRQ4SCA:1;         /*   IRQ4SCA    */
605                   } BIT;                          /*            */
606             } ISCRH;                              /*            */
607         union {                                   /* ISCRL       */
608             unsigned char BYTE;                     /*  Byte Access */
609             struct {                              /*  Bit Access  */
610                   unsigned char IRQ3SCB:1;         /*   IRQ3SCB    */
611                   unsigned char IRQ3SCA:1;         /*   IRQ3SCA    */
612                   unsigned char IRQ2SCB:1;         /*   IRQ2SCB    */
613                   unsigned char IRQ2SCA:1;         /*   IRQ2SCA    */
614                   unsigned char IRQ1SCB:1;         /*   IRQ1SCB    */
615                   unsigned char IRQ1SCA:1;         /*   IRQ1SCA    */
616                   unsigned char IRQ0SCB:1;         /*   IRQ0SCB    */
617                   unsigned char IRQ0SCA:1;         /*   IRQ0SCA    */
618                   } BIT;                          /*            */
619             } ISCRL;                              /*            */
620         unsigned char wk0[6];                       /*            */
621         union {                                   /* ABRKCR      */
622             unsigned char BYTE;                     /*  Byte Access */
623             struct {                              /*  Bit Access  */
624                   unsigned char CMF:1;             /*   CMF       */
625                   unsigned char :4;               /*            */
626                   unsigned char TESTSEL:2;         /*   TESTSEL    */
627                   unsigned char BIE:1;             /*   BIE       */
628                   } BIT;                          /*            */
629             } ABRKCR;                             /*            */
630         union {                                   /* BARA        */
631             unsigned char BYTE;                     /*  Byte Access */
632             struct {                              /*  Bit Access  */
633                   unsigned char A23:1;             /*   A23       */
634                   unsigned char A22:1;             /*   A22       */
635                   unsigned char A21:1;             /*   A21       */
636                   unsigned char A20:1;             /*   A20       */
637                   unsigned char A19:1;             /*   A19       */
638                   unsigned char A18:1;             /*   A18       */
```

```
639                   unsigned char A17:1;              /*   A17       */
640                   unsigned char A16:1;              /*   A16       */
641                   } BIT;                           /*             */
642             } BARA;                                /*             */
643         union {                                    /* BARB        */
644             unsigned char BYTE;                      /*  Byte Access */
645             struct {                               /*  Bit Access  */
646                   unsigned char A15:1;              /*   A15       */
647                   unsigned char A14:1;              /*   A14       */
648                   unsigned char A13:1;              /*   A13       */
649                   unsigned char A12:1;              /*   A12       */
650                   unsigned char A11:1;              /*   A11       */
651                   unsigned char A10:1;              /*   A10       */
652                   unsigned char A9:1;               /*   A9        */
653                   unsigned char A8:1;               /*   A8        */
654                   } BIT;                           /*             */
655             } BARB;                                /*             */
656         union {                                    /* BARC        */
657             unsigned char BYTE;                      /*  Byte Access */
658             struct {                               /*  Bit Access  */
659                   unsigned char A7:1;               /*   A7        */
660                   unsigned char A6:1;               /*   A6        */
661                   unsigned char A5:1;               /*   A5        */
662                   unsigned char A4:1;               /*   A4        */
663                   unsigned char A3:1;               /*   A3        */
664                   unsigned char A2:1;               /*   A2        */
665                   unsigned char A1:1;               /*   A1        */
666                   unsigned char :1;                /*             */
667                   } BIT;                           /*             */
668             } BARC;                                /*             */
669         union {                                    /* IER16       */
670             unsigned char BYTE;                      /*  Byte Access */
671             struct {                               /*  Bit Access  */
672                   unsigned char IRQ15E:1;           /*   IRQ15E     */
673                   unsigned char IRQ14E:1;           /*   IRQ14E     */
674                   unsigned char IRQ13E:1;           /*   IRQ13E     */
675                   unsigned char IRQ12E:1;           /*   IRQ12E     */
676                   unsigned char IRQ11E:1;           /*   IRQ11E     */
677                   unsigned char IRQ10E:1;           /*   IRQ10E     */
678                   unsigned char IRQ9E:1;            /*   IRQ9E      */
679                   unsigned char IRQ8E:1;            /*   IRQ8E      */
680                   } BIT;                           /*             */
681             } IER16;                               /*             */
682         union {                                    /* ISR16       */
683             unsigned char BYTE;                      /*  Byte Access */
684             struct {                               /*  Bit Access  */
685                   unsigned char IRQ15F:1;           /*   IRQ15F     */
686                   unsigned char IRQ14F:1;           /*   IRQ14F     */
687                   unsigned char IRQ13F:1;           /*   IRQ13F     */
688                   unsigned char IRQ12F:1;           /*   IRQ12F     */
689                   unsigned char IRQ11F:1;           /*   IRQ11F     */
690                   unsigned char IRQ10F:1;           /*   IRQ10F     */
691                   unsigned char IRQ9F:1;            /*   IRQ9F      */
692                   unsigned char IRQ8F:1;            /*   IRQ8F      */
693                   } BIT;                           /*             */
694             } ISR16;                               /*             */
695         union {                                    /* ISCR16H     */
696             unsigned char BYTE;                      /*  Byte Access */
697             struct {                               /*  Bit Access  */
698                   unsigned char IRQ15SCB:1;         /*   IRQ15SCB   */
699                   unsigned char IRQ15SCA:1;         /*   IRQ15SCA   */
700                   unsigned char IRQ14SCB:1;         /*   IRQ14SCB   */
701                   unsigned char IRQ14SCA:1;         /*   IRQ14SCA   */
702                   unsigned char IRQ13SCB:1;         /*   IRQ13SCB   */
```

```
703                   unsigned char IRQ13SCA:1;         /*   IRQ13SCA   */
704                   unsigned char IRQ12SCB:1;         /*   IRQ12SCB   */
705                   unsigned char IRQ12SCA:1;         /*   IRQ12SCA   */
706                   } BIT;                           /*              */
707             } ISCR16H;                              /*              */
708         union {                                     /* ISCR16L      */
709             unsigned char BYTE;                       /*  Byte Access */
710             struct {                                /*  Bit Access  */
711                   unsigned char IRQ11SCB:1;         /*   IRQ11SCB   */
712                   unsigned char IRQ11SCA:1;         /*   IRQ11SCA   */
713                   unsigned char IRQ10SCB:1;         /*   IRQ10SCB   */
714                   unsigned char IRQ10SCA:1;         /*   IRQ10SCA   */
715                   unsigned char IRQ9SCB:1;          /*   IRQ9SCB    */
716                   unsigned char IRQ9SCA:1;          /*   IRQ9SCA    */
717                   unsigned char IRQ8SCB:1;          /*   IRQ8SCB    */
718                   unsigned char IRQ8SCA:1;          /*   IRQ8SCA    */
719                   } BIT;                           /*              */
720             } ISCR16L;                              /*              */
721         unsigned char wk1[198];                       /*              */
722         union {                                     /* IER          */
723             unsigned char BYTE;                       /*  Byte Access */
724             struct {                                /*  Bit Access  */
725                   unsigned char IRQ7E:1;            /*   IRQ7E      */
726                   unsigned char IRQ6E:1;            /*   IRQ6E      */
727                   unsigned char IRQ5E:1;            /*   IRQ5E      */
728                   unsigned char IRQ4E:1;            /*   IRQ4E      */
729                   unsigned char IRQ3E:1;            /*   IRQ3E      */
730                   unsigned char IRQ2E:1;            /*   IRQ2E      */
731                   unsigned char IRQ1E:1;            /*   IRQ1E      */
732                   unsigned char IRQ0E:1;            /*   IRQ0E      */
733                   } BIT;                           /*              */
734             } IER;                                  /*              */
735   };                                               /*              */
736
737   /************************************************************************
738   Macro definitions
739   ************************************************************************/
740   #define MAC0 (*(volatile struct st_ether __evenaccess *)0xFFF900)    /* ETHER Address */
741   #define EDMAC0 (*(volatile struct st_edmac __evenaccess *)0xFFF980)    /* EDMAC Address */
742   #define SYSTEM (*(volatile struct st_system *)0xFFFE3C)  /* SYSTEM Address */
743   #define INT (*(volatile struct st_int *)0xFFFEE7)       /* INT Address  */
744
745   /************************************************************************
746   Variable Externs
747   ************************************************************************/
748
749   /************************************************************************
750   Functions Prototypes
751   ************************************************************************/
752
753
754
755
756   #endif /* IODEFINE2472_H */
```

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2008.07.29	—	初版発行
1.01	2009.02.10	3	図 1 にオシレータを追加
1.02	2011.07.06		「ドライバ改訂に伴う修正」
		2	仕様、適用条件の修正
		18	参考プログラムの動作説明の修正
		22	図 23 の修正
		23	図 24 の修正
		24	図 25 の修正
		25	図 26 の修正
		26	図 27 の修正
		31～61	サンプルプログラムリストのコードの修正と追加
		32	2.9 章の追加
			「アプリケーションノート誤記の修正」
		1	動作確認デバイスの修正
		3	図 1 の修正
		17	図 17 の修正
		20	図 21 の修正
		28	図 29 の修正
		29	図 30 の修正

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. **未使用端子の処理**
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. **電源投入時の処置**
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. **リザーブアドレスのアクセス禁止**
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. **クロックについて**
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. **製品間の相違について**
   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。
   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

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、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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0