SH7786 グループ

SH7786 PCI Express コントローラ(PCIEC)初期化設定例

R01AN0557JJ0100
Rev1.02
2011.04.27

要旨

この資料は，SH7786 の PCI Express コントローラの初期設定に必要な設定例を示します。

動作確認デバイス

SH7786

目次

1. はじめに

1.1 仕様

本アプリケーションノートで説明する PCI Express コントローラ（PCIEC）初期設定例は、パワーオンリセット解除後に、ローカルバスステートコントローラ(LBSC)、DDR3-SDRAM インタフェース(DBSC3)、PCI Express コントローラ（PCIEC）の初期設定を行います。 PCI Express コントローラ（PCIEC）初期設定後には PCI Express Root port または End point として動作し、PCI Express Root port 動作時には PCI Express End point デバイスの VenderID や DeviceID 等をシリアルコンソールへの表示や簡単な DMA 転送を実行します。 PCI Express End point 動作時には VenderID や DeviceID 等を PCI Express コントローラ（PCIEC）にセットします。

1.2 使用機能

- ローカルバスステートコントローラ(LBSC)
- DDR3-SDRAM インタフェース(DBSC3)
- PCI Express コントローラ(PCIEC)
- シリアルコミュニケーションインタフェース(SCIF0)

ローカルバスステートコントローラ(LBSC)、DDR3-SDRAM インタフェース(DBSC3)の初期設定は「SH7786 グループアプリケーションノート SH7786 初期設定例 (R01AN0242JJ0101) 」で説明しています。併せて参照してください。

尚、本アプリケーションノートの参考プログラムは、「SH7786 初期設定例 (R01AN0242JJ0101)」で動作確認していますので、ローカルバスステートコントローラ(LBSC)、DDR3-SDRAM インタフェース(DBSC3)の初期設定の説明は割愛します。

1.3　適用条件

表 1.3.1.1　適用条件

評価ボード	AP-AH4AD-0A（アルファプロジェクト製）（注 1）	
	CPU	SH7786
	動作周波数	内部クロック： 533MHz SuperHyway クロック： 267MHz 周辺クロック： 44MHz DDR3 クロック： 533MHz 外部バスクロック： 89MHz
	クロック動作モード	クロックモード 3 (MD0=High, MD1=High, MD2=Low, MD3=Low)
	エンディアン	リトルエンディアン　(MD8=High)
	アドレスモード	29 ビットアドレスモード　(MD10=Low)
	エリア 0 バス幅	16bit　(MD4=Low, MD5=High, MD6=Low)
	メモリ	NOR 型 Flash メモリ 16M バイト (エリア 0): Spansion 製 S29GL128P90TFIRI DDR3-SDRAM 256M バイト(エリア 2～5): Micron 製 MT41J64M16LA-187E (2 個)
	PCI Express	SH7786 内蔵 PCI Express コントローラ（PCIEC） PCI Express Base Specification Revision1.1 サポート PCI Express Generation1 ：バス周波数： 2.5GHz Root port： PCI Express x4 カードスロット 1 チャネル End point： PCI Express x1 カードエッジ 1 チャネル
	シリアルインタフェース	SH7786 内蔵 SCIF ch0 (115200bps)
	PC-USB-02A(アルファプロジェクト製)（注 2）	
	シリアルコンソール	TTL シリアル⇔USB コンバータ
ツールチェイン	Super-H RISC engine Standard Toolchain Ver9.3.2.0	
	コンパイルオプション(注 3)	-cpu=sh4a -endian=little -include="$(PROJDIR)¥inc","$(PROJDIR)¥inc¥drv" -define=CONFIG_PCIE_ROOT=0 -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo
	アセンブラオプション	-cpu=sh4a -endian=little -round=zero -denormalize=off -include="$(PROJDIR)¥inc" -debug -object="$(CONFIGDIR)¥$(FILELEAF).obj" -literal=pool,branch,jump,return -nolist -nologo -chgincpath -errorpath
	リンカオプション	-noprelink -rom=D=R -nomessage -list= "$(CONFIGDIR)¥$(PROJECTNAME).map" -optimize=safe -start=INTHandler,VECTTBL,INTTBL,IntPRG/0800, PResetPRG/01000,P,C,C$BSEC,C$DSEC,D/02000, RSTHandler,PnonCACHE/0A0000000,B,R/0ADF00000, S/0ADFF0000 -nologo

(注 1) AP-SH4AD-0A の使用方法等の詳細は「AP-SH4AD-0A Hardware Manual」を参照してください。

(注 2) PC-USB-02A の使用方法等の詳細は「AP-SH4AD-0A Hardware Manual」を参照してください。

(注 3) PCI Express コントローラ（PCIEC）を PCI Express Root port で動作させる場合のマクロ定義は CONFIG_PCIE_ROOT=0。また、PCI Express コントローラ(PCIEC)を PCI Express End point として動作させる場合のマクロ定義は CONFIG_PCIE_END=1。

表 1.3.1.2 に本応用例でのセクション配置を示します。

表 1.3.1.2　セクション配置

セクション名	セクション用途	領域	配置アドレス(仮想アドレス)	
INTHandler	例外/割込みハンドラ	ROM	0x00000800	P0 領域 (キャッシング可能，MMU アドレス変換不可)
VECTTBL	リセットベクタテーブル 割込みベクタテーブル	ROM		
INTTBL	割込みマスクテーブル	ROM		
IntPRG	割込み関数	ROM		
PResetPRG	リセットプログラム	ROM	0x00001000	
P	プログラム領域	ROM	0x00002000	
C	定数領域	ROM		
C$BSEC	未初期化データ領域用アドレス構造	ROM		
C$DSEC	初期化データ領域用アドレス構造	ROM		
D	初期化データ	ROM		
RSTHandler	リセットハンドラ	ROM	0xA0000000	P2 領域 (キャッシング不可，MMU アドレス変換不可)
PnonCACHE	プログラム領域 (キャッシュ無効アクセス)	ROM		
B	未初期化データ領域	RAM	0xADF00000	
R	初期化データ領域	RAM		
S	スタック領域	RAM	0xADFF0000	

1.4　本アプリケーションノートで使用する用語の説明

- PCI Express

PCI Express は、PCI-SIG によって策定された PCI バスに代わるシリアル転送インタフェースで、32bit パラレルインタフェースの PCI バス物理層との互換性はありませんが、共通の通信プロトコルを使用しているためソフトウェア資産を継承可能です。

PCI Express で使用される伝送線路（レーン）は、双方向通信を行うために、送受信の 1 組の差動ペアで構成されています（双対単方向伝送デュアル・シンプレックス）。PCI Express Base Specification Revision1.1 (一般に Gen 1 と呼称) において、1 レーンあたりの転送レートは片方向では 2.5Gbps、双方向では 5.0Gbps を実現します。2 レーン（x2）、4 レーン（x4）と複数のレーンを組み合わせることによりさらに高いデータ帯域幅を上げることが可能です。

- PCI Express Root port

PCI Express Root port は、PCI Express の全体制御を行うポートで、PCI Express システムに一つ以上必要です。

PCI Express Root port はコンフィグレーションサイクルの発生による PCI Express システムの初期化、エラーメッセージの受信および回復などの、システム全体の統括を行います。また、Root port はリクエストパケットの送信、コンプリージョンパケットの返信、メッセージの送受信などを行えます。

- PCI Express End point

PCI Express End point は、Root port の制御下でデータ通信を行うポートで、PCI Express システムに複数個存在することが可能です。

End point は、コンフィグレーションサイクルによる初期化を受けた後、エラーの検出および Root port への報告などを行います。また、End point はリクエストパケットの送信、コンプリージョンパケットの返信、メッセージの送受信などを行えます。

- I/O アドレス空間

PCI バス互換の I/O アドレス空間です。

- メモリアドレス空間

PCI バス互換のメモリアドレス空間です。

- コンフィグレーションレジスタ空間

PCI バス互換のコンフィグレーションレジスタ空間です。コンフィグレーションレジスタ空間は 4096 バイトあり、下位 256 バイトは従来の PCI バスと互換性のある領域。上位 3840 バイトは、PCI Express 拡張コンフィグレーション空間と呼ばれる PCI Express の領域です。

1.5　本アプリケーションノートの適用範囲

本アプリケーションノートでは、全ての PCI Express コントローラ（PCIEC）機能をサポートしていません。 本アプリケーションノートでは、PCI Express コントローラ（PCIEC）初期設定後、PCI Express Root port または End point として動作し、PCI Express Root port 動作時には PCI Express End point デバイスの VenderID や DeviceID 等をシリアルコンソールへの表示や簡単な DMA 転送、PCI Express End point 動作時には VenderID や DeviceID 等を PCI Express コントローラ（PCIEC）にセットする基本的な使用方法について説明します。

以下の PCI Express コントローラ（PCIEC）機能については，本アプリケーションノートの説明対象外となります。

— メッセージ送受信

— INTx/MSI による割り込み

— リンクパワー制御機能（L0、L0s、L1、L3 ステート）

2.　PCI Expressコントローラ(PCIEC)

PCI Express コントローラ（以下、PCIEC）は、PCI Express の制御を行い、SH7786 内部バス(SuperHyway バス)と PCI Express に接続される PCI デバイス間のデータ転送を行います。

本章では、本アプリケーションノートでサポートする PCIEC 機能を紹介します。尚、PCIEC の詳細については「SH7786 グループ　ユーザーズマニュアル　ハードウェア編(RJJ09B0533)　13 章 PCI Express コントローラ(PCIEC)」を参照してください。

2.1　　サポートする機能

(1)パケット送受信のサポート

表 2.1.1.1 にサポートする PCI Express パケットを示します。PCIEC では、規格で禁止されていないパケットの送受信をサポートしています。

表 2.1.1.1　サポートする PCI Express パケット

パケット種別	Root port		End point	
	送信	受信	送信	受信
メモリリード	○	○	○	○
メモリライト	○	○	○	○
I/O リード	○	○	-	○
I/O ライト	○	○	-	○
ロック	○*	-	-	-
コンフィグレーションリード	○	○	-	○
コンフィグレーションライト	○	○	-	○
メッセージ	○*	○*	○*	○*

【記号説明】
○：PCIEC はサポートしています
－：PCI Express では、規格により使用が禁じられています
＊ ：PCIEC はサポートしていますが、本アプリケーションノートではサポートしていません

(2)メッセージ送受信のサポート

表 2.1.1.2 にサポートする PCI Express メッセージを示します。PCIEC は、Vender Defined Message をサポートしていません。尚、本アプリケーションノートはメッセージ送受信機能をサポートしていません。

表 2.1.1.2　サポートする PCI Express メッセージ

パケット種別	Root port		End point	
	送信	受信	送信	受信
Assert_INTA	ー	○	○	ー
Assert_INTB	ー	○	○	ー
Assert_INTC	ー	○	○	ー
Assert_INTD	ー	○	○	ー
Deassert_INTA	ー	○	○	ー
Deassert_INTB	ー	○	○	ー
Deassert_INTC	ー	○	○	ー
Deassert_INTD	ー	○	○	ー
PME_Active_State_Nak	△	ー	ー	△
PM_PME	ー	△	△	ー
PME_Turn_Off	△	ー	ー	△
PME_To_Ack	ー	△	△	ー
ERR_COR	ー	○	○	ー
ERR_NONFATAL	ー	○	○	ー
ERR_FATAL	ー	○	○	ー
Unlock	○	ー	ー	○
Set_Slot_Power_Limit	○	ー	ー	○
Vender_Define Type0	×	×	×	×
Vender_Define Type1	×	×	×	×

【記号説明】
○：PCIEC はサポートしています
△：送受信は可能ですが、ソフトウェアによる制御が必要です
ー：PCI Express では、規格により使用が禁じられています
×：PCIEC はサポートしていません

(3)コンフィグレーションレジスタのサポート

表 2.1.1.3 にサポートする PCI Express コンフィグレーションレジスタを示します。PCIEC は、BIST、スイッチ、拡張 ROM に関するレジスタをサポートしていません。

表 2.1.1.3　サポートする PCI Express コンフィグレーションレジスタ

コンフィグレーションレジスタ	PCIEC レジスタ	Root port	End point
Vender ID レジスタ	PCICONF0[15:0]	○	○
Device ID レジスタ	PCICONF0[31:16]	○	○
コマンドレジスタ	PCICONF1[15:0]	○	○
ステータスレジスタ	PCICONF1[31:16]	○	○
リビジョン ID レジスタ	PCICONF2[7:0]	○	○
クラスコードレジスタ	PCICONF2[31:8]	○	○
キャッシュラインサイズ	PCICONF3[7:0]	―	―
マスターレイテンシタイマー	PCICONF3[15:8]	―	―
ヘッダータイプレジスタ	PCICONF3[23:16]	○	○
BIST レジスタ	PCICONF3[31:24]	×	×
ベースアドレスレジスタ 0	PCICONF4[31:0]	○	○
ベースアドレスレジスタ 1	PCICONF5[31:0]	○	○
ベースアドレスレジスタ 2	PCICONF6[31:0]	―	○
プライマリバスナンバ	PCICONF6[7:0]	○	―
セカンダリバスナンバ	PCICONF6[15:8]	○	―
サブオーディネートバスナンバ	PCICONF6[23:16]	○	―
セカンダリレイテンシタイマー	PCICONF6[31:24]	―	―
ベースアドレスレジスタ 3	PCICONF7[31:0]	―	○
I/O ベースレジスタ	PCICONF7[7:0]	×	―
I/O リミットレジスタ	PCICONF7[15:8]	×	―
セカンダリステータスレジスタ	PCICONF7[23:16]	○	―
ベースアドレスレジスタ 4	PCICONF8[31:0]	―	○
メモリベース	PCICONF8[15:0]	×	―
メモリリミット	PCICONF8[31:16]	×	―
ベースアドレスレジスタ 5	PCICONF9[31:0]	―	○
プリフェッチャブルメモリベース	PCICONF9[15:0]	×	―
プリフェッチャブルメモリリミット	PCICONF9[31:16]	×	―
カードバス CIS ポインタ	PCICONF10[31:0]	―	×
プリフェッチャブルベース（上位 32 ビット）	PCICONF10[31:0]	×	―
サブシステム ID レジスタ	PCICONF11[31:0]	―	○
プリフェッチャブルリミット（上位 32 ビット）	PCICONF11[31:0]	×	―
サブシステムベンダ ID レジスタ	PCICONF12[31:0]	―	○
I/O ベース（上位 16 ビット）	PCICONF12[15:0]	×	―
I/O リミット（下位 16 ビット）	PCICONF12[31:16]	×	―

コンフィグレーションレジスタ	PCIEC レジスタ	Root port	End point
ケイパビリティポインタ	PCICONF13[31:0]	○	○
拡張 ROM ベースアドレスレジスタ	PCICONF14[31:16]	ー	×
インタラプトライン	PCICONF15[7:0]	○	○
インタラプトピン	PCICONF15[15:8]	○	○
最小グラント	PCICONF15[23:16]	ー	ー
最大レイテンシ	PCICONF15[31:24]	ー	ー
ブリッジコントロールレジスタ	PCICONF15[31:16]	○	ー

【記号説明】
○：PCIEC はサポートしています
ー：PCI Express では、規格により使用されません
×：PCIEC はサポートしていません

(4)ケイパビリティストラクチャのサポート

表 2.1.1.4 にサポートする PCI Express ケイパビリティストラクチャを示します。PCIEC では、これらのケイパビリティストラクチャをサポートします。尚、本アプリケーションノートは PCI Express ケイパビリティストラクチャをサポートしていません。

表 2.1.1.4　サポートする PCI Express ケイパビリティストラクチャ

Capability Structure	サポート	先頭アドレス
PCI パワーマネジメント	○	H'040
MSI	○	H'050
PCI Express	○	H'070
Advanced Error Reporting	×	―
Virtual Channel	○	H'100
Device Serial Number	○*	H'1B0
PCI Express Link Complex Declaration	×	―
PCI Express Root Complex Internal Link Control	×	―
Power Budgeting	×	―
PCI Express Root Complex Event Collector Endpoint Association	×	―
Multi-Function Vaitual Channel	×	―
Vender-Specific	×	―
RCRB Header	×	―

【記号説明】
○:ハードウェアによりサポートします
×:PCIEC はサポートしていません
*:PCIEC は、デバイスシリアルナンバケイパビリティストラクチャを保持していますが、ハードウェアによるシリアルナンバの付与をおこなっていません。デバイスシリアルナンバケイパビリティストラクチャを使用する場合、ソフトウェアにより、シリアルナンバを付与してください。また、初期状態ではデバイスシリアルナンバケイパビリティストラクチャはケイパビリティリストのチェーンに入っていません。使用する場合は、ケイパビリティリストのチェーンへの追加を行ってください。

2.2　端子設定

PCIEC は、PCI Express の規格で規定されている Root port または End point として動作します。この動作モードはモードピンにより指定します。本アプリケーションノートではモードピンの設定をディップスイッチにて指定します。ディップスイッチの詳細は「4.1 章　SH7786 評価ボード AP-SH4AD-0A」を参照してください。

PCIEC は、PCI Express の規格で規定されているレガシーエンドポイント、ルートコンプレックスインテグレーテッドエンドポイント、スイッチ、ルートコンプレックスイベントコントローラとして動作しません。

(1)Root port

Root port は、PCI Express の全体制御を行うポートで、PCI Express システムに一つ以上必要です。PCIEC は、SH プロセッサをホストプロセッサとした Root port として動作可能です。

Root port はコンフィグレーションサイクルの発生による PCI Express システムの初期化、エラーメッセージの受信および回復などの、システム全体の統括を行います。また、Root port はリクエストパケットの送信、コンプリージョンパケットの返信、メッセージの送受信などを行えます。

(2)End point

End point は、Root port の制御下でデータ通信を行うポートで、PCI Express システムに複数個存在することが可能です。PCIEC は、End point として動作可能です。

End point は、コンフィグレーションサイクルによる初期化を受けた後、エラーの検出および Root port への報告などを行います。また、End point はリクエストパケットの送信、コンプリージョンパケットの返信、メッセージの送受信などを行えます。

2.3　PCIECモジュール初期化

PCIEC による PCI Express パケット通信を行うためには、（1）PCI Express バスと SH7786 内部バス（SuperHyway バス）とをつなぐブリッジ機能の設定、及び、（2）PCI Express バスと PCIEC とのコネクションの確立　を行う必要があります。

(1) PCI Express バスと SuperHyway バスとをつなぐブリッジ機能の設定

PCI Express バスと SuperHyway バスとをつなぐブリッジ機能を設定するには以下のレジスタに転送情報をセットします。セットする転送情報の内容については、「2.6 ターゲット転送」を参照して下さい。

- PCIELAR0～5
- PCIELAMR0～5

(2) PCI Express バスと PCIEC とのコネクションの確立

上記の転送制御レジスタに転送情報をセットした後に、PCIETCTLR[0].CFINIT ビットを 1 にセットし、PCI Express バスと PCIEC とのコネクションの確立開始を指示します。（上記の転送制御レジスタは、CFINIT を 1 とした後には、値を変更することが出来ません。

PCIETCTLR[0].CFINIT を 1 にセットすることにより、データリンク層の初期化が開始され、接続先の PCI Express デバイスと通信を行う準備を開始します。

データリンク層の初期化が完了すると、DL_Active の状態となり、VC0 による通信を行う準備が整います。以下のいずれかの方法により DL_Active であることが確認できた段階で初期化が完了します。

VC0 による通信の確立

- PCIETSTR[0].DLLACT が 1 となっていること
- VCCAP6[17].VC NeGotiation PenDing が 1 となっていること
- DL_Active を示す INTDL 割り込みが発生したこと

DL_Active により INTDL 割り込みを発生させるためには、事前に以下の設定を行う必要があります。

- PCIEINTER[14].INTDLE を 1 にセット
- DLINTENR[31].Data Link Layer ACTive Enable を 1 にセット

PCIEC は、複数の仮想的なバーチャルチャネル（VC : Virtual channel）には対応していません。バーチャルチャネル（VC0）のみにより通信を行います。

2.4　コンフィグレーションサイクル（PCI Express初期化）

PCIEC を Root port として使用する場合、コンフィグレーションサイクルを発生させ、接続先のデバイスのコンフィグレーションを行います。コンフィグレーションサイクルは、コンフィグレーションアクセスを用いて接続先の End point のコンフィグレーションレジスタの状態を調査し、その結果に応じて Root port 自身と End point のコンフィグレーションレジスタに値をセットすることを指します。Root port が自身の持つコンフィグレーションレジスタにアクセスする場合は、通常の SuperHyway バスを介したアクセスを用います。

(1)コンフィグレーションアクセスの発生

PCIEC を Root port として使用する場合には、コンフィグレーションのアクセスを行いコンフィグレーションサイクルを発生させ、各所初期設定を行います。

PCIEC からのコンフィグレーションアクセスによる外部デバイスのコンフィグレーションレジスタへのアクセスは、以下の手順により行います。

PCIEC が Root port として動作している際に、PCIEC 自身のコンフィグレーションレジスタにアクセスする場合には、以下手順ではなく、SuperHyway バスのアドレス空間にマッピングされているレジスタに、SuperHyway バスを経由してアクセスしてください。

(a) PCIEPAR のセット

PCIEPAR に、アクセス先のコンフィグレーションレジスタのレジスタ番号、拡張レジスタ番号、およびアクセス先のデバイスのバス／デバイス／ファンクション番号を指定します。

(b) PCIEPCTLR のセット

PCIEPCTLR に発生させるコンフィグレーションアクセスのタイプ及び、アクセス許可をセットします。

(c)PCIEPDR のセット

PCIEPDR へリードアクセスを行うことによりコンフィグレーションリードが、ライトのアクセスを行うことにより、コンフィグレーションライトが発生します。リード時には、コンフィグレーションリードの結果が読み出されます。

(d) PCIEPCTLR の確認

PCIEPCTLR[16].CRS ビットを確認し、CRS（Configuration Request Retry Status）が返されたかを確認します。

本ビットが 1 の場合、接続先デバイスが立ち上がっていないため、コンフィグレーションリクエストに対する正しい応答が出来ていないことを示します。本ビットが 1 であった場合、本ビットに 1 を書込みクリアした後に、再度(c)の処理から再開してください。

一度コンフィグレーションアクセスが成功したデバイスに対しては、本ビットの確認を行う必要はありません。

(2)コンフィグレーションアクセスの受信

PCIEC を End point として使用する場合には、Root port からのコンフィグレーションアクセスを受信し、初期化処理を受け付けます。

PCIEC によるコンフィグレーションアクセスの受信は、ハードウェアにより自動処理されるため、ソフトウェアによる制御は不要です。

ただし、PMCAP1[1:0].PowerState フィールドに対するコンフィグレーションライトアクセスによる、パワーステートの変更はソフトウェアにより処理が必要です。

正常なコンフィグレーションライトを受信した場合、受信パケット中のバスナンバ、デバイスナンバを取り込み、TLCTLR[31:24].BusNumber、TLCTLR[23:19].DeviceNumber、TLCTLR[18:16].FunctionNumber に書き込まれます。これらの値は PCIEC が生成するパケットのリクエスタ ID として使用されます。

(3)設定内容

Root port として PCIEC を使用する場合、コンフィグレーションリクエストを発行し、PCI Express の初期化として、以下の設定を行ってください。以下のレジスタ設定は、Root port が、Root port/End point の両者のレジスタに対して行います。

下記の内容は、接続先が単一の PCI Express デバイスの場合の説明です。接続先がスイッチあるいはブリッジの場合、別途設定が必要になります。

(a)MPS（Max Payload Size）の設定

Root port、Endpoint を含む、PCI Express システム中に存在する全ての PCI Express デバイスのコンフィグレーションレジスタ中の MPSS(Max Payload Size Supported)を調べ、最も小さな値をシステムの MPS として決定します。決定した MPS の値を、Root port・End point を含む全てのデバイスのコンフィグレーションレジスタに設定します。

(b)MRRS（Max Read Request Size）の設定

PCIEC では、MRRS の値は MPS の値と同じ値とします。MPS と同じ値を Root port･End point を含む全てのデバイスのコンフィグレーションレジスタに設定します。

(c)PCI アドレス空間の設定（BAR の設定）

各デバイスに PCI アドレス空間の割り当てを行います。

PCI Express の規格に則り、アドレス空間の割り当てを行い、各デバイスの BAR のその結果を設定します。

(d)動作モードの設定

PCI Express の動作モードを規定する、以下のコンフィグレーションレジスタの値をセットします。初期値のまま使用する場合には、セットする必要はありません。これらのレジスタは、コンフィグレーションサイクルの完了後には値を変更しないで下さい。　尚、レジスタの詳細は「SH7786 グループ　ユーザーズマニュアル　ハードウェア編　13.4.5 コンフィグレーションレジスタ」を参照して下さい。

PCICONF1[10].Interrupt Disable

PCICONF1[8].SERR Enable

PCICONF1[6].Parity Error Response

PCICONF15[17].SERR Enable (Root port のみ)

PCICONF15[15:8].Interrupt Pin　（End point のみ）

PCICONF15[7:0].Interrupt Line (Root port のみ)

EXPCAP2[11].Enable No Snoop

EXPCAP2[4].Enable Relaxed Ordering

EXPCAP2[3].Unsupported Request Reporting Enable

EXPCAP2[2].Fatal Error Reporting Enable

EXPCAP2[2].Non Fatal Error Reporting Enable

EXPCAP2[2].Correctable Error Reporting Enable

EXPCAP3[20].Data Link Layer Active Reporting Capable (Root port のみ)

EXPCAP7[4].CRS Software Visibility Enable

EXPCAP7[3].PME Interrupt Enable

EXPCAP7[2].System Error on Fatal Error Enable

EXPCAP7[1].System Error on Non-Fatal Error Enable

EXPCAP7[0].System Error on Correctable Error Enable

(e)INTx/MSI 割込みの設定

システムで使用する割込み（INTx 又は MSI）を決定し、各デバイスに設定します。

(f)マスターイネーブルの設定

初期化後に行う転送に応じて、PCICONF1[2].Bus Master Enable、PCICONF1[1].Memory Space Enable、PCICONF1[0].I/O Space Enable を設定します。

Root Port が End point からのリクエストを受信する場合には、まず、Root Port の Bus Master Enable ビットを 1 にセットします。同時に、メモリアクセスを受け付ける場合には Memory Space Enable を、Ｉ／Ｏアクセスを受け付ける場合には、I/O Space Enable を 1 にセットします。この設定を行わないと、Root Port はリクエストを受け付けません。次に、End point の Bus Master Enable ビットを 1 にセットします。この設定を行わないと、End point はリクエストを発行できません。

End point に対してメモリアクセス、I/O アクセスを行う場合には、End point の Memory Space Enable, I/O Space Enable を 1 にセットします。この設定を行わないと、End point はリクエストを受信しません。

2.5　PI/O転送（PCIEC→外部デバイスへのデータ転送）

ここでの PI/O 転送とは、内部バス経由で PCIEC のメモリ空間にアクセスすることにより、PCI Express パケットを生成することにより行う転送を指します。

(1)　概要

PI/O 転送は、CPU などが SuperHyway を経由して PCIEC のメモリ空間にアクセスすることにより、PCI Express パケットを生成し、送信する転送を指します。PI/O 転送により、外部 PCI Express デバイスに対してメモリリード／ライト、IO リード／ライトを行うことができます。

PI/O 転送により、PCI メモリ空間へのアクセスにより、簡易に PCI Express パケットを生成することができます。リードアクセスにより、PCI Express 上でのリードパケットが、ライトアクセスにより PCI-Express 上でのライトパケットが生成されます。

通常の PI/O 転送では、ひとつの PCI メモリ空間へのアクセスから、ひとつの PCI Express パケットが生成されます。生成される PCI Express パケットのデータ長は、PCI メモリ空間へのアクセスサイズと同じとなります。そのため、CPU による４バイトのアクセスでは、データ長が４バイトの短い PCI Express パケットしか生成できず、大量のデータを転送する場合の転送効率は良くありません。

大量のデータを転送する場合には、パケット結合か、PCIEC 内蔵の DMAC を使用してください。

尚、本アプリケーションノートでは、パケット結合の機能をサポートしていません。

(2)アドレスマップ（SuperHyway 空間）

表 2.5.1.1 に SuperHyway 空間のアドレスマップを示します。

PCIEC には、3 種類(物理的には 8 種類)のアドレス領域があります。PCI メモリ領域(6 種類)、制御レジスタ領域、及びコンフィグレーションレジスタ領域です。このうち、PCI メモリ領域にアクセスすることにより、PCI Express パケットの生成が行われます。PCI メモリ領域と PCI-Express のアドレス空間とのマッピングについては、次節で示します。

表 2.5.1.1　SuperHyway 空間のアドレスマップ

メモリ領域	PCIEC0	PCIEC1	PCIEC2	物理アドレスサイズ
PCI 領域 0	H'FD00_0000 ～ H'FD7F_FFFF	H'FD80_0000 ～ H'FDFF_FFFF	H'FC80_0000 ～ H'FCBF_FFFF	PCIEC0/1: 8MB PCIEC2: 4MB
PCI 領域 1	使用不可	使用不可	使用不可	512MB
PCI 領域 2	H'1000_0000 ～ H'13FF_FFFF （メモリ空間設定 1/2/5/6 選択時のみ)	使用不可	使用不可	64MB
PCI 領域 3	H'FE10_0000 ～ H'FE1F_FFFF	H'FE30_0000 ～ H'FE3F_FFFF	H'FCD0_0000 ～ H'FCDF_FFFF	1MB
制御レジスタ 領域(1)	H'FE00_0000 ～ H'FE03_FFFF	H'FE20_0000 ～ H'FE23_FFFF	H'FCC0_0000 ～ H'FCC3_FFFF	256kB
コンフィグレーション レジスタ	H'FE04_0000 ～ H'FE04_0FFF	H'FE24_0000 ～ H'FE24_0FFF	H'FCC4_0000 ～ H'FCC4_0FFF	4kB
制御レジスタ 領域(2)	H'FE04_1000 ～ H'FE07_FFFF	H'FE24_1000 ～ H'FE27_FFFF	H'FCC4_1000 ～ H'FCC7_FFFF	252kB

【注】
(1)29 ビットアドレッシングモード時でのアドレスマップ
(2)本アプリケーションノートでは、PCIEC0,PCIEC1 を使用

(3)PCI メモリ空間、PCI I/O 空間へのアクセス

図 2.5.1.2 に SuperHyway アドレス空間から PCI アドレス空間へのマッピングを示します。図 2.5.1.2 に示すように、SuperHyway アドレス空間中の PCI 領域へのアクセスは、PCI アドレス空間又は PCI I/O 空間のどちらかにマッピングされます。どちらの空間にマッピングされるか、あるいは個々の空間個々のどのアドレスにマッピングされるかは、PI/O 転送の転送制御レジスタ（後述）により指定します。PCI 空間にマッピングされている SuperHyway 上の空間（PCI 領域）にアクセスすることにより、PCI メモリ空間や PCI I/O 空間にアクセスすることができます。

PCI 領域へのリードのアクセスからは、PCI メモリ空間又は PCI I/O 空間のリードパケットの生成が、PCI 領域へのライトアクセスからは、PCI メモリ空間又は PCI I/O 空間のライトパケットが生成されます。

PCI メモリ空間へのアクセス時は、PCI 領域へのアクセスサイズにより、そのパケット長が決定されます。つまり、4Byte アクセスにより PCI 領域にアクセスした場合、PCI メモリ空間に 4Byte(1DW)のサイズのリード／ライトパケットが生成されます。

PCI I/O 空間は、4Byte(1DW)のアクセスのみが許されます。PCI 領域を PCI I/O 空間にマッピングする場合、その PCI 領域にはアクセスサイズを 4Byte としてアクセスして下さい。

転送先の空間（PCI メモリ、I/O 空間の選択）、各空間での先頭アドレス、転送先空間のサイズ、転送パケットの属性は、PIO 転送の転送制御レジスタにより指定します。

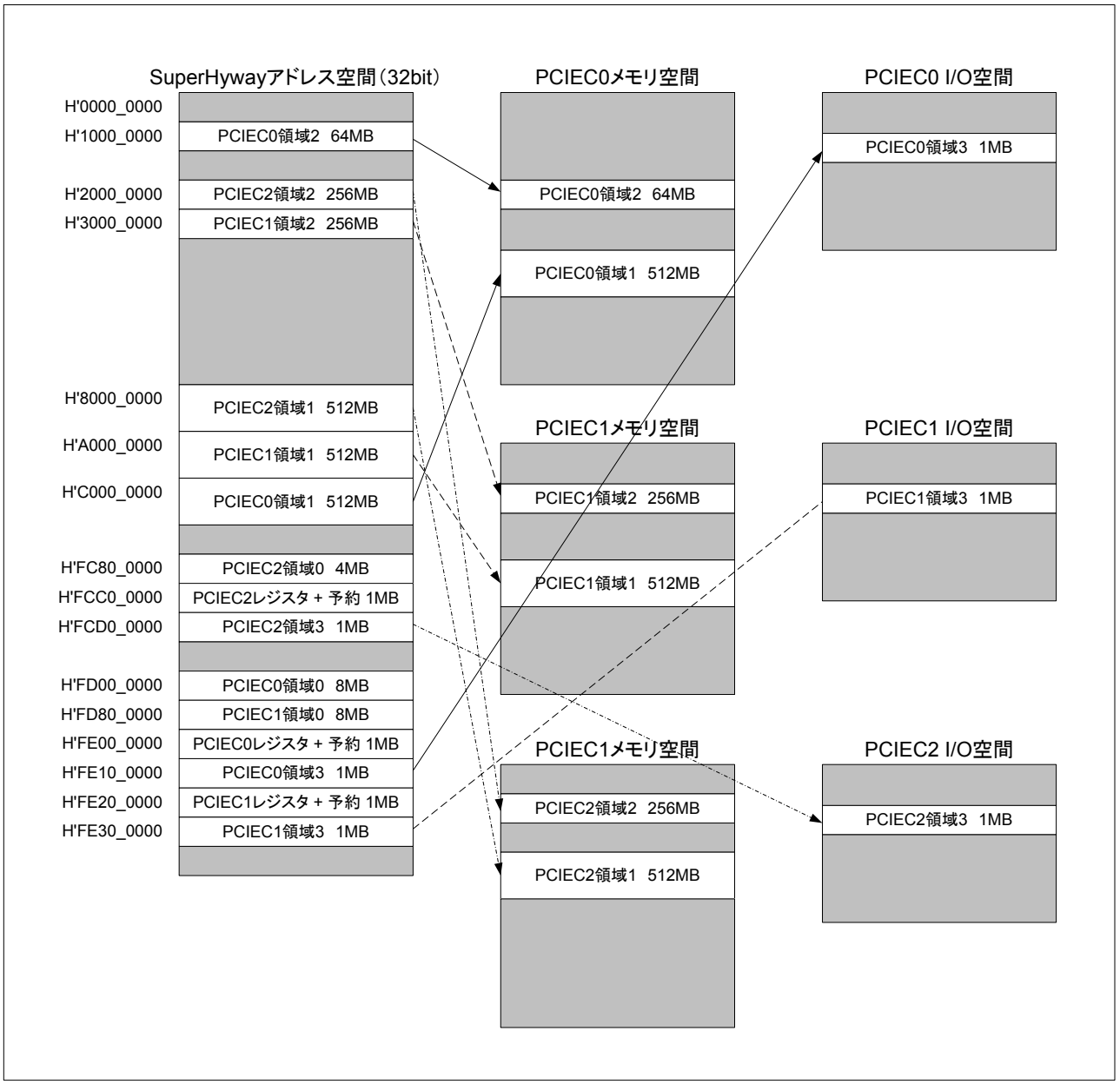


図 2.5.1.2　SuperHyway アドレス空間から PCI アドレス空間へのマッピング

(4)PI/O 転送のレジスタ設定

表 2.5.1.3 に PI/O 転送の転送制御レジスタを示します。PCI 領域 0～3 へのアクセスは、これらのレジスタが指定する内容に従って、PCI メモリ又は I/O 空間にマッピングされます。これらのレジスタの役割を、表 2.5.1.3 に示します。

表 2.5.1.3　PI/O 転送の転送制御レジスタ

PCIEPALR0～3	PCI 領域 0～3 がマッピングされる PCI アドレス空間の先頭アドレス（下位 32 ビット）
PCIEPAHR0～3	PCI 領域 0～3 がマッピングされる PCI アドレス空間の先頭アドレス（上位 32 ビット）
PCIEPAMR0～3	PCI 領域 0～3 中の、PCI アドレス空間へマッピングさせるサイズを指定
PCIEPTCTLR0～3	PCI 領域 0～3 の有効／無効を指定 転送先の空間（PCI メモリ空間、PCI I/O 空間）を指定 変換時の属性（Lock、EP、No Snoop、Relax Ordering）を指定

PCIEPALRn、PCIEPAHRn（n=0～3）により、PCI 領域 n がマッピングされる PCI Express 空間上でのアドレスを指定します。

PCIEPAMRn により、PCI 領域のサイズを指定します。表 2.5.1.1　SuperHyway 空間のアドレスマップに記載される PCI 領域のサイズより大きなサイズは指定できません。

PCIEPTCTLRn により、各領域の有効／無効の設定、転送先の空間、転送時のパケットの属性を指定します。本レジスタで、PCI 領域 n が有効であると指定しないと（初期値は無効）、該当する PCI 領域へのアクセスは無効となります。ロック転送を行う場合、あるいは他の属性を設定する場合には、PCI 領域へのアクセス前に本レジスタに設定を行います。

(5)SuperHyway バスから PCI へのアドレス変換

PCI 領域へのアクセスによる PCI 空間へのアクセス時のアドレスは、アクセスした PCI 領域のアドレスと、転送制御レジスタの設定により決定されます。アドレス変換の詳細は下記の通りであり、その内容を図 2.5.1.4 に示します。（図中及び下記の説明文中の n は、0～3 の値をとり、PCI 領域 0～3 に対応します。）

PCI アドレスの下位 16 ビット（[17:2]）は、SuperHyway アドレスの下位ビットから生成されます。

PCI アドレスの中間の 11 ビット（[28:18]）は、転送制御レジスタ（PCIEPAMRn）の値により、SuperHyway アドレスまたは PCIEPALRn の該当するビットのどちらかが選択されます。（PCIEPAMRn の該当するビットが 1 のとき、SuperHyway アドレスが使用され、0 のとき PCIEPALRn が使用されます。）

PCI アドレスの上位 35 ビット（[63:29]）は、PCIEPAHRn と PCIEPALRn の上位 3 ビットが使用されます。

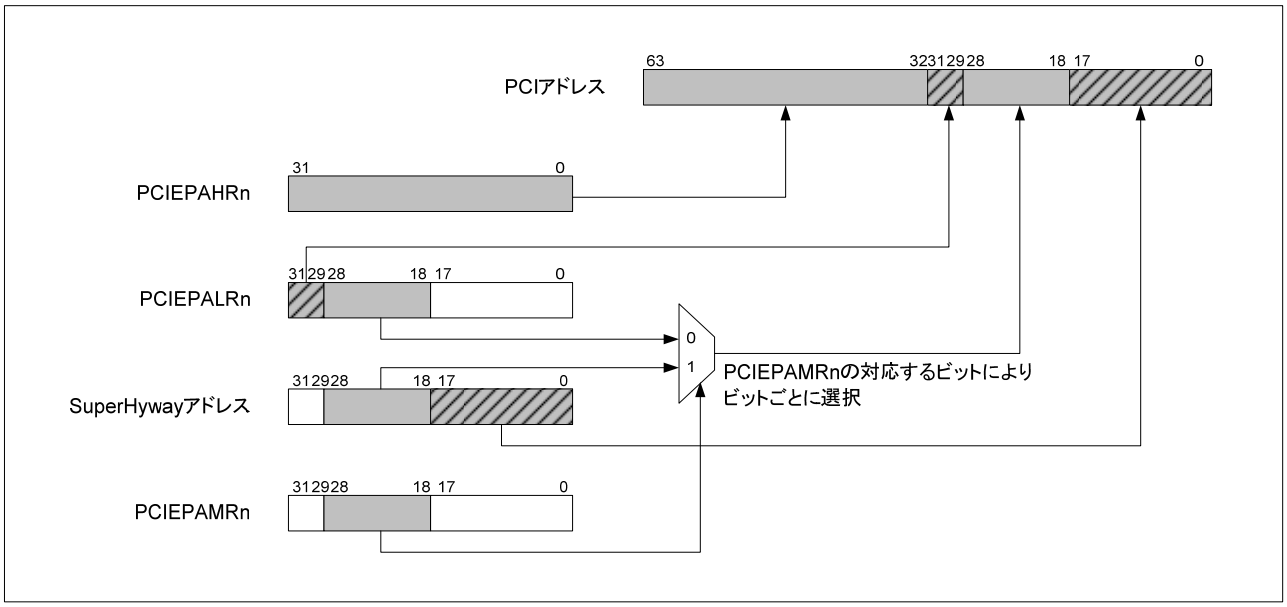


図 2.5.1.4　PCI 空間へのアドレス変換

2.6　ターゲット転送（外部デバイス→PCIECへのデータ転送）

本節ではターゲット転送について説明します。ここでのターゲット転送とは、外部デバイスからの PCI Express パケットを PCIEC が受信し、SuperHyway バス経由で SH7786 内の他モジュールにデータを転送することを指します。

(1)概要

ターゲット転送は、外部デバイスが PCI Express パケットより本モジュールにアクセスすることにより、SuperHyway バスへのリクエストを生成し、他モジュールに送信する転送を指します。ターゲット転送により、外部デバイスがメモリリード・ライト、I/O リード・ライトのパケット送信することにより、SH7786 内部の他モジュール、あるいは DRAM 等の SH7786 に接続している外部メモリにリード、ライトを行うことができます。

ターゲット転送では、MPS(Max Payload Size)で指定されるサイズ以下であれば、任意のサイズのデータ長のパケットを受信することができます。SuperHyway がサポートするサイズより大きなサイズの転送が指定された場合、PCIEC がパケットの分割を行い、複数の内部バスへのリクエストを生成します。

(2)アドレスマップ（PCI Express 空間）

図 2.6.1.1 に PCI 空間の SuperHyway 空間へのマッピングを示します。

PCI Express 空間中のアドレスの割り当ては、初期化時のレジスタ設定を元に、コンフィグレーションサイクル中に Root port により動的に決定されます。初期化時のレジスタ設定では、各領域のサイズと、確保する領域の種類（メモリ空間、I/O 空間の種別など）を指定します。CFINIT に 1 をセットし初期化を完了すると、コンフィグレーションレジスタ中の BARn(Base Address Register n)の値や R/W 属性に初期化内容が反映されます。ここで n は BAR のレジスタ番号を示し、Root port の時は、n=0-1、End point の時は n=0-5 となります。その後のコンフィグレーションサイクル中に、Root port がこれらの設定を参照し、アドレスマップを決定し、その結果を各デバイスのコンフィグレーションレジスタの BARn にセットします。この BARn が指すアドレスが、個々のデバイスに割り当てられた PCI Express 空間中の先頭アドレスとなります。

PCIEC は、メモリ空間を確保するエリアとして、PCI の 64 ビットアドレス空間または 32 ビット空間（64 ビット空間の先頭の 4G の領域）をサポートします。32 ビットアドレス空間に領域を確保する場合は BARn をひとつ使用し、64 ビットアドレス空間に領域を確保する場合は連続する二つの BARn レジスタ(BARn+1/BARn)を使用します。そのため、Root port 時には最大ひとつの 64 ビットアドレス空間の領域を、End point 時には最大 3 つの 64 ビットアドレス空間の領域を確保することができます。

I/O 空間は、ひとつの BAR レジスタにより領域を確保します。

PCI Express からの BARn へのアクセスは、PCIEC が受信し、SuperHyway バスへのアクセスに変換されます。変換先のアドレスは、PCIELARn により指定します。

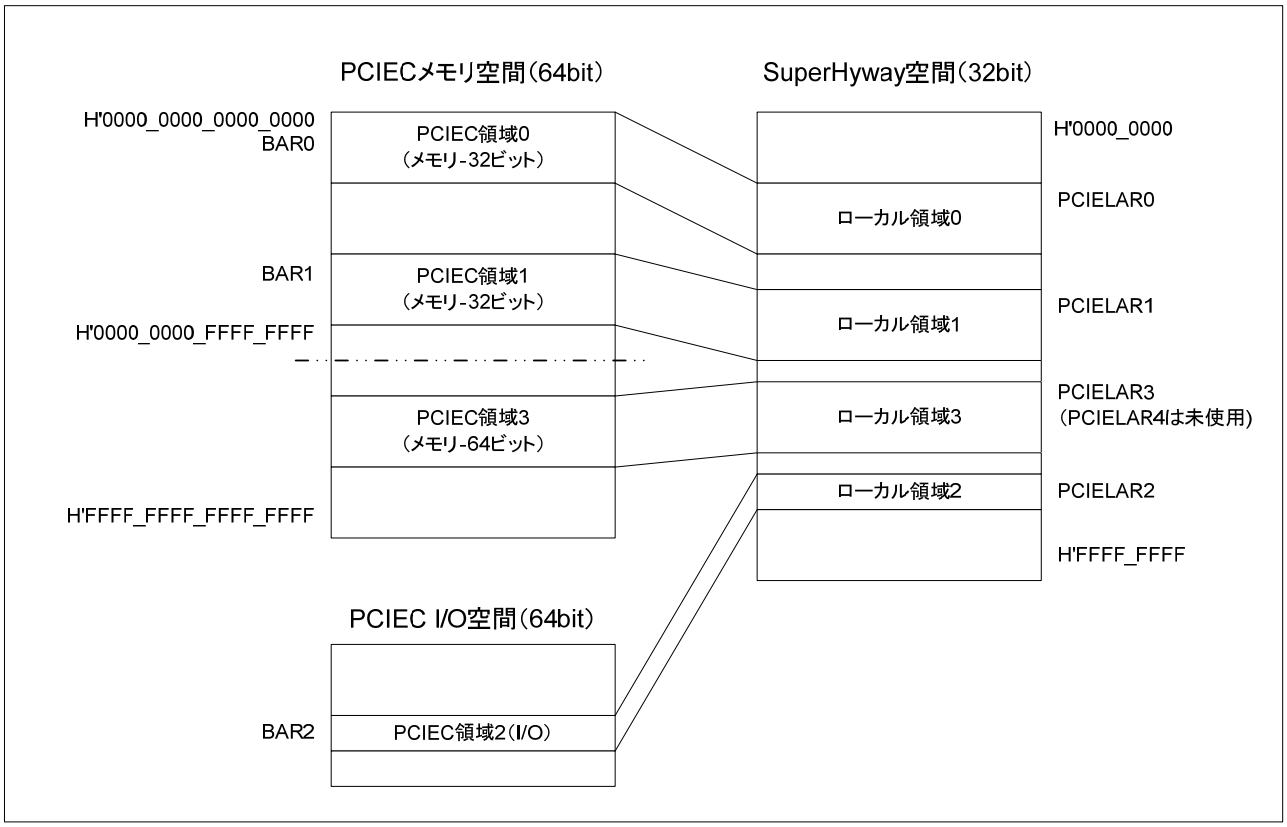


図 2.6.1.1　PCI 空間の SuperHyway 空間へのマッピング

(3)ターゲット転送のレジスタ設定

表 2.6.1.2 にターゲット転送の転送制御レジスタを示します。PCI 空間に確保する領域及び確保した領域へのアクセスからの内部バスへのアクセスは、これらのレジスタにより制御されます。

PCIEC は、6 セットのターゲット転送レジスタを持ち、Root port として使用する場合は最大 2 個、End point として使用する場合は最大 6 個の PCI 領域を PCI 空間上に確保することができます。また、PCIEC は、PCI 空間に確保するメモリ空間として、64 ビット空間と 32 ビット空間をサポートします。32 ビット空間を使用する場合は、1 セットのターゲット転送レジスタにより一つの空間を確保し、64 ビット空間を使用する場合は、連続する 2 セットのターゲット転送レジスタにより一つの空間を確保します。

表 2.6.1.2　ターゲット転送の転送制御レジスタ

PCIELARLn	PCI 領域 n がマッピングされるローカルバス（SuperHyway）空間の先頭アドレス
PCIELAMRn	PCI 領域 n のサイズを指定

【注】 n は Root port のときには 0、1、End point のときには 0～5

PCIELARLn により、BAR 領域 n がマッピングされる SuperHyway 上でのアドレスを指定します。n は Root port の時には 0 または 1 を、End point の時には 0～5 の値をとります。

PCIELAMRn により、PCI 空間上に確保する PCI 領域のサイズ、領域の種類（メモリ空間、I/O 空間など）、領域の有効／無効を指定します。このレジスタで、領域を有効としないと、PCI 空間上での領域の確保が行われず、内部バスへの転送も行われません。（リセット後の初期値は、全領域が無効となっています）。

(4)PCI から SuperHyway バスへのアドレス変換

図 2.6.1.3 に PCI 空間のアドレスデコードを、図 2.6.1.4 に PCI アドレスから SuperHyway アドレスへの変換を示します。

受信した PCI Express パケットは、まずアドレスのデコードを行います。アドレスデコードは、受信したパケットのアドレス幅が 32 ビットか 64 ビットかにより異なります。アドレス幅が 32 ビットの場合、受信パケット中のアドレスと BARn とを比較し、マッチする n の値を決定します。その後、対応する PCIELARn, PCIELAMRn を用いて SuperHyway バスのアドレスへの変換を行います。受信したパケットのアドレス幅が 64 ビットの場合、BARn+1/BARn を組み合わせた 64 ビットのアドレスと受信パケットの 64 ビットアドレスを比較し、マッチする n の値を決定します。その後 PCIELARn, PCIELAMRn を用いて SuperHyway バスのアドレスへの変換を行います。

このとき、PCIELARn+1, PCIELAMRn+1 は使用されません。

変換後の SuperHyway バスアドレスの下位ビット（ビット[17:0]）は、受信 PCI パケットのアドレスの下位ビットから生成されます。中位のビット（ビット[28:18]）は PCIELAMRn のビットにより、受信パケットのアドレス又は PCIELARn の該当ビットが使用され、上位ビット（ビット[31:29]）は PCIELARn のビット[31:29]がそのまま使用されます。

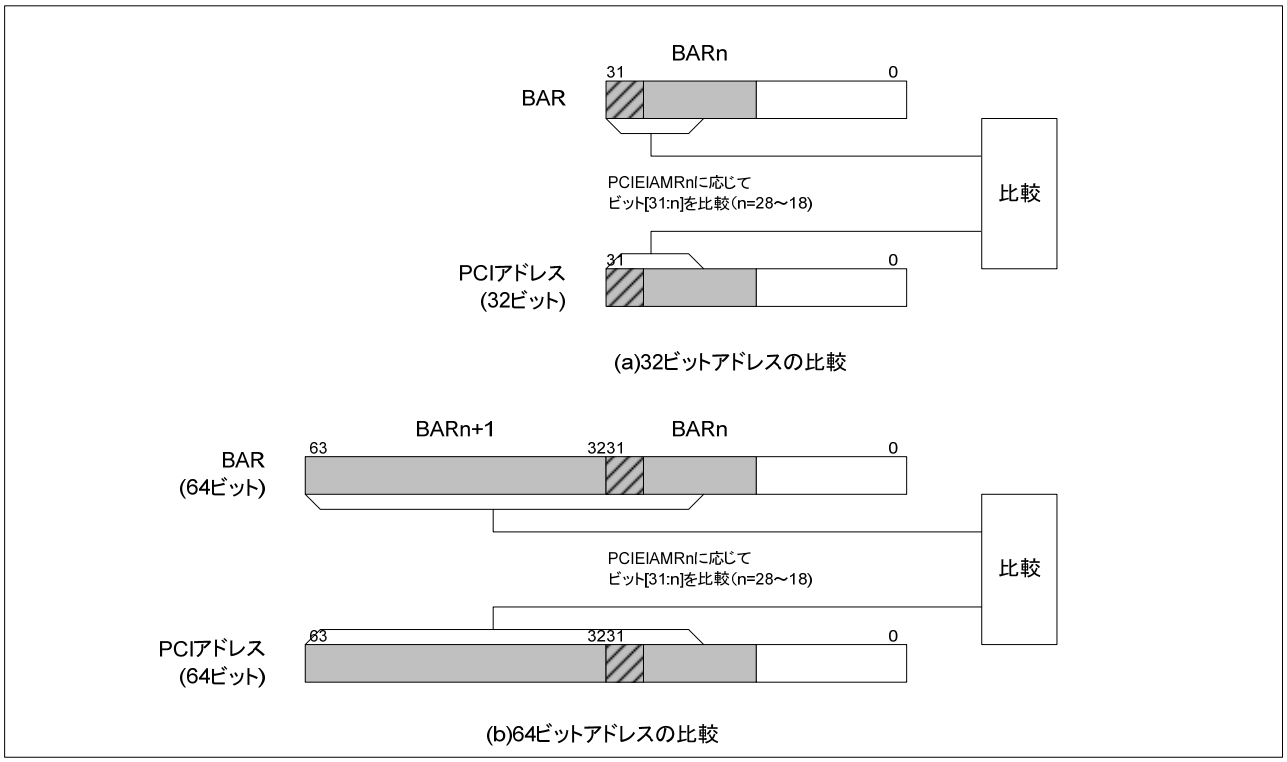


図 2.6.1.3　PCI 空間のアドレスデコード方式

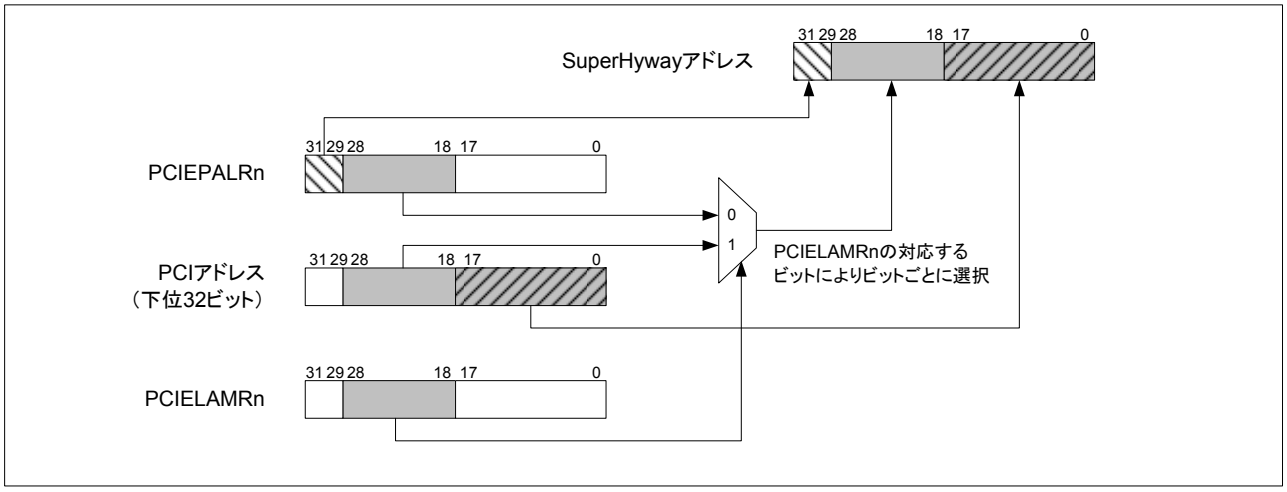


図 2.6.1.4　PCI アドレスから SuperHyway アドレスへの変換方式

(5)PCI Express からの SuperHyway バスへのアクセス

PCIEC を通じて、PCI Express よりアクセス可能な内部バスでの空間は、CS2#、CS3#、DBSC 空間、および他の PCIEC モジュールです。ここで、転送先として指定可能な他の PCIEC モジュールとは、PCIEC0 からは PCIEC1/2、PCIEC1 からは PCIEC0/2、PCIEC2 からは PCIEC0/1 となります。

2.7　DMA転送

本節では、PCIEC に内蔵する DMAC（PCIEC-DMAC）を用いた DMA 転送について説明します。

（1）概要

PCIEC-DMAC は、PCI Express と SuperHyway バスを経由して接続する他モジュールや外部メモリとのデータ転送を効率的に行うための DMAC です。PCIEC-DMAC は、最大で 1024byte* のデータ長を持つパケットを PCI Express に対して発行できるよう設計されており、PCI Express の高い転送性能を生かした高速データ転送を可能にします。

【注】*　PCI Express 側に発行するパケットのデータ長は、Max Payload Size が上限となります。

また、PCIEC-DMAC は、不連続な領域のデータを転送するためのストライド転送に対応し、複数の転送コマンドを連続して実行するための機能としてコマンドチェーンに対応しています。ストライド転送では、一定回数の転送を行った後に転送元／転送先のアドレスにオフセットを加える機能により、不連続領域を転送元／転送先とした転送に対応します。コマンドチェーンでは、転送元／先のアドレスや転送サイズなどの DMAC 設定の集合をコマンドとみなし、メモリ上に格納したコマンドを逐次読み出し実行する機能により、CPU を介さないでの複数の転送の連続実行に対応します。

（2）特長

- チャネル数：4 チャネル
- アドレス空間：PCI Express＝64 ビット、SuperHyway バス＝32 ビット
- 転送データ長：PCI Express=4 バイト～1K バイト、SuperHyway バス＝4 バイト～32 バイト
- 最大転送回数：536,870,912 回（2^{29} 回）
- アドレスモード：デュアルアドレスモード
- 転送要求：オートリクエスト（レジスタ制御による起動）
- データ転送：通常モード（連続転送）、ストライド転送、コマンドチェーン
- 優先順位：チャネル優先順位固定モードとラウンドロビンモードから選択可能
- 割り込み要求：データ転送終了時、またはエラー発生時に INTC へ割り込み要求を発生可能

（3）DMAC 転送要求

PCIEC-DMAC は、オートリクエストモードに対応しています。PCIEC-DMAC の起動は、CPU などからの PCIEC-DMAC のレジスタへの書き込みにより行います。

(4) チャネルの優先順位

PCIEC-DMAC では、同時に複数のチャネルに対して転送要求があった場合には、決められた優先順位に従って転送を行います。チャネルの優先順位は固定、ラウンドロビンの 2 種類のモードから選択できます。モードの選択は PCIEDMAOR の ABT ビットにより行います。

PCIEC-DMAC では、転送効率を上げるため、なるべく大きなサイズの PCI Express パケットを転送に使用します。いったん送信・受信処理を開始したパケットは、そのパケットの処理が完了するまで中断されません。そのため、より高い優先順位の転送が実行可能になっても、その段階で実行中の転送でのパケット送信が完了するまでチャネルの切り替えは行われず、最大で 4k バイトの転送が完了するまでチャネルの切り替えが行われない可能性があります。

チャネルの切り替えは、実行中のチャネルでの 1 セットのデータ転送が完了したタイミングで起こります。ここで、1 セットのデータ転送完了とは、SuperHyway バスと PCI Express の両者の転送が同時に完了したタイミングを意味します。

(a) 固定モード

固定モードではチャネルの優先順位は変化しません。優先順位は、以下の通りとなります。

- CH0＞CH1＞CH2＞CH3

(b) ラウンドロビンモード

ラウンドロビンモードでは、1 つのチャネルで 1 セットの転送が完了すると、そのチャネルの優先順位が一番低くなるように優先順位を変更します。

(5) 通常モードの転送

通常モードでの転送では、指定された転送元のアドレスから、指定された転送先のアドレスへのデータ転送を行います。転送方向は、PCI→SuperHyway バスまたは SuperHyway バス→PCI のどちらかを選択できます。

PCIEC-DMAC による通常モードでの転送は、以下の手順により行います。各レジスタの詳細仕様は、「SH7786 グループ ユーザーズマニュアル ハードウェア編(RJJ09B0533) 13 章 PCI Express コントローラ(PCIEC)の PCIEC-DMAC 制御レジスタ」を参照して下さい。

(a) PCIEC-DMAC の全体設定

PCIEDMAOR に、DMA_Enable とアービトレーションの設定を行います。

(b) 転送設定

PCI/SuperHyway のアドレス、バイトカウントの設定、及び転送終了割込みの設定を行います。

PCIEDMPALRn/ PCIEDMPAHRn、PCIEDMSALRn、PCIEDMBCNTRn に、転送元／転送先のアドレスを指定します。ここで、n はチャネル番号（０～３）を示します。指定するアドレスは、転送の方向によらず、PCI 側のアドレスを PCIEDMPALRn/ PCIEDMPAHRn に、SuperHyway バス側のアドレスを PCIEDMSALRn に指定します。

転送終了時に割り込みを発生させる場合、PCIEDMCHSRn に割り込み設定を行います。

ストライド転送を行わない場合には､PCIEDMSBCNTRn 及び PCIEDMSTRRn には０をセットして下さい。

コマンドチェーンを使用しない場合には、PCIEDMCCARn には０をセットして下さい。

(c) DMAC の起動

PCIEDMCHCRn に、転送方向の指定を行うと同時に、チャネルをイネーブルとすることにより、転送を起動します。

ストライド転送を行わない場合には、PCIEDMCHCRn[24].SARE 及び PCIEDMCHCRn[25].PARE には０をセットして下さい。

コマンドチェーンを使用しない場合には、PCIEDMCHCRn[29].CCRE には０をセットして下さい。

(d) 転送終了待ち

PCIEDMCHSRn[0].TE が１となることを確認、あるいは転送終了割込みを検出することにより転送終了を検知します。

(e) 終了処理

PCIEDMCHCRn[31].CHE を０として転送を完了します。また、PCIEDMCHSRn[0].TE に１を書き込み、このビットをクリアします。

この終了処理を行わないと、次回の DMA 転送が起動しません。

(6) ストライド転送

ストライド転送では、一定のバイト数の転送を行った後に、ストライド、つまり転送元／転送先のアドレスへのオフセットの加算を行います。転送先アドレスにストライドを行うことによりスキャッター転送、転送元のアドレスにストライドを行うことにより、ギャザー転送を行えます。転送元／転送先の両者にストライドを行うことにより、非連続領域の転送が行えます。

ストライド転送を行う場合には、転送設定の際に、ストライドを行う間隔（ストライドカウンタ）を PCIEDMSBCNTRn に、ストライド幅を PCIEDMSTRRn にセットして下さい。PCI 側または SuperHyway 側のみにストライドを行う場合には、ストライドを行わない側のストライド幅（PCIEDMSTRRn の SS または PS フィールド）を 0 としてください。

また、DMAC の起動の際に、PCIEDMCHCRn[24].SARE または、PCIEDMCHCRn[25].PARE に１をセットして下さい。

その他の設定は、通常モードの転送と同じです。

(7) コマンドチェーン

コマンドチェーンでは、複数の DMAC コマンドを連続して実行することができます。ここで、DMAC コマンドとは、PCIEC-DMAC の転送を指示する情報の集合を示し、PCIEDMPALRn、PCIEDMSALRn、PCIEDMBCNTRn、PCIEDMSBCNTRn、PCIEDMSTRRn、PCIEDMCCARn、PCIEDMCHCRn により指定される情報を指します。これらの情報は、PCIEC-DMAC 制御レジスタに対して設定するほかに、メモリ上に図 2.7.1.1 に示す形式で設定することが可能です。（PCI 側アドレスの上位 32 ビットは、DMAC コマンドにより指定することは出来ません。PCIEC-DMAC 制御レジスタに指定したものが有効となります）。コマンドチェーンの使用により、DMAC コマンドの実行終了後に、次の DMAC コマンドをメモリから読出し、PCIEC-DMAC 制御レジスタに DMAC コマンドの内容を書き込み、その DMAC コマンドを実行することが出来ます。読み出す DMAC コマンド内に次の DMAC コマンドを指定することにより、DMAC コマンドのチェーンを構築し、転送を連続して行うことが出来ます。

コマンドチェーン使用時には、まず PCIEC-DMAC 制御レジスタの各チャネルのレジスタにより設定される DMAC コマンドを実行します。この DMAC コマンドの実行を終了した後に、PCIEDMCCARn が示すアドレスから次の DMAC コマンドをメモリから読出し、コマンドの内容を PCIEC-DMAC の該当するチャネルのレジスタにその内容を書き込み、実行します。新たに読出した DMAC コマンド中の CCRE ビットが 1 となっていた場合には、そのコマンドの終了後に再度メモリから次コマンドを読出し、実行します。読出した DMAC の CCRE ビットが 0 の場合、そのコマンドの実行が完了した段階で、一連のコマンドチェーンの実行が完了します。

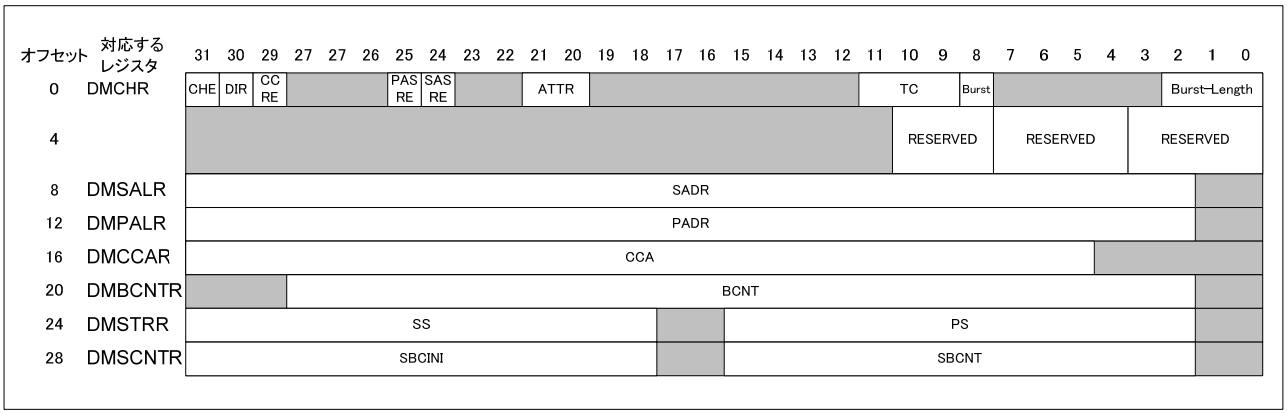


図 2.7.1.1　PCIEC-DMAC コマンドフォーマット

コマンドチェーンは、PCIEDMCHCRn[29].CCRE を 1 とした状態でチャネルをイネーブルとすることにより起動します。コマンドチェーンを起動する場合には、事前に SuperHyway バスからアクセス可能な DDR3 SDRAM、LBSC、IL メモリ、OL メモリ、L2CR で指定する共有メモリ上に DMAC コマンドのチェーンを格納した上で、最初の DMAC コマンドのアドレスを PCIEDMCCARn にセットして下さい。

メモリ上に格納する DMAC コマンドは、以下の条件を満たすものを格納してください。尚、DMAC コマンドは DDR、LBSC、LRMA 等の共有メモリ上に置いてください。

- CHE フィールド

  常に 1 を指定してください。

- ATTR フィールド

  PCIEC-DMAC 制御レジスタの ATTR フィールドにより指定される ATTR を、メモリに格納する DMAC コマンドの ATTR フィールドに指定してください。コマンドのロードにより、ATTR フィールドの内容を変更することは出来ません。

- TC フィールド

  PCIEC-DMAC 制御レジスタの TC（トラフィッククラス）フィールドに指定するバーチャルチャネル VC0 と、同一のバーチャルチャネル VC0 をメモリに格納する DMAC コマンドの TC フィールドに指定してください。

- RESERVED フィールド

  常に８を設定してください。

- CCA フィールド

  最後に実行するコマンドの CCA フィールドは、0 を指定してください。

（8）PCIE-DMAC の割込み要因

PCIEC-DMAC は、チャネルごとに転送終了を示す割込み、全チャネル共通でエラー終了を示す割り込みを発生します。

3.　シリアルコンコミュニケーションインタフェース(SCIF0)

シリアルコミュニケーションインタフェース(以下、SCIF0)は、FIFO バッファを内蔵しており調歩同期式とクロック同期式の 2 方式でシリアル通信ができます。

尚、本アプリケーションノートでは SCIF0 を調歩同期式のシリアルコンソールとして使用します。

SCIF0 の詳細については「SH7786 グループ　ユーザーズマニュアル　ハードウェア編(RJJ09B0533)　24 章 FIFO 内蔵シリアルコミュニケーションインタフェース(SCIF)」を参照してください。

4. 応用例の説明

4.1 SH7786 評価ボードAP-SH4AD-0A

本アプリケーションノートでは、アルファプロジェクト製 SH7786 評価ボード AP-SH4AD-0A（以下、AP-SH4AD-0A）を 2 台使用して、それぞれの PCIEC を PCI Express Root port 及び PCI Express End point の 2 つのモードで動作させます。AP-SH4AD-0A の詳細は「AP-SH4AD-0A Hardware Manual」を参照してください。

4.1.1 メモリマップ

表 4.1.1.1 に AP-SH4AD-0A のメモリマップを示します。

表 4.1.1.1　AP-SH4AD-0A メモリマップ

エリア	アドレス	接続デバイス	バス幅
0	H'0000_0000 - H'00FF_FFFF	S29GL128P90TFIR20 (16MB)	16 ビット
	H'0100_0000 - H'03FF_FFFF	シャドウ	
1	H'0400_0000 - H'0400_0FFF	LAN9221 (512B)	16 ビット
	H'0400_1000 - H'07FF_FFFF	シャドウ	
2	H'0800_0000 - H'0BFF_FFFF	MT41J64M16LA-187E (256MB)	32 ビット
3	H'0C00_0000 - H'0FFF_FFFF		
4	H'1000_0000 - H'13FF_FFFF		
5	H'1400_0000 - H'17FF_FFFF		
6	H'1800_0000 - H'17FF_FFFF	ユーザ開放	32 ビット

4.1.2　PCI Express Root portモードの設定

AP-SH4AD-0A を PCI Express Root port モードに設定するには、ディップスイッチを以下の設定としてください。ディップスイッチの詳細は「AP-SH4AD-0A Hardware Manual　2 章　機能」を参照してください。

- PCI Express モード設定

SW2-2	
MODE11	PCI Express モード
ON	Root port モード

- PCI Express PHY モード設定

SW2-3	
MODE12	PCI Express PHY モード
ON	4 レーン＋1 レーン

4.1.3　PCI Express End pointモードの設定

AP-SH4AD-0A を PCI Express End point モードに設定するには、ディップスイッチを以下の設定としてください。ディップスイッチの詳細は「AP-SH4AD-0A Hardware Manual　2 章　機能」を参照してください。

- PCI Express モード設定

SW2-2	
MODE11	PCI Express モード
OFF	End point モード

- PCI Express PHY モード設定

SW2-3	
MODE12	PCI Express PHY モード
ON	4 レーン＋1 レーン

(注 3) AP-SH4AD-0A を PCI Express End point で動作させ且つ、PCI Express カードエッジから電源を供給する場合、基板上はんだジャンパ JP1 は Open としてください。電源供給の詳細は「AP-SH4AD-0A Hardware Manual　3.7.1 章　電源の供給例」を参照してください。

4.1.4　　シリアルコンソールの設定

AP-SH4AD-0A のシリアルインタフェースには SCIF0 を使用し、以下の設定としています。

また、シリアルコンソールには PC-USB-02A を使用します。PC-USB-02A は SCIF0 の TTL シリアルレベルを USB に変換し、PC との通信を行います。

シリアルインタフェース及び、コンソールの詳細は「AP-SH4AD-0A Hardware Manual　3.7 章　シリアルインタフェース」を参照してください。

表 4.1.4.1　シリアルコンソール設定

項目	仕様
SCIF0	調歩同期式
ボーレート	115200bps
データ	8 ビット
パリティビット	無し
ストップビット	1 ビット
フロー制御	無し

4.2　参考プログラムの説明

本アプリケーションノートの参考プログラムは2台のAP-SH4AD-0Aを使用し、1台はPCI Express Root port、もう1台はPCI Express End pointとし、PCIECの初期設定後にPCI Express Root port側からPCI Express End pointのコンフィグレーションレジスタをシリアルコンソールに表示するPCIECの基本的な使用方法について説明します。

4.2.1　参考プログラムのシステム構成

PCI Express Root portとPCI Express End point設定としたAP-SH4AD-0Aを以下のように接続し、シリアルコンソールPC-USB-02Aを使用してコンソールPCにPCI Express End pointのコンフィグレーションレジスタ（VenderIDやDeviceID等）を表示します。

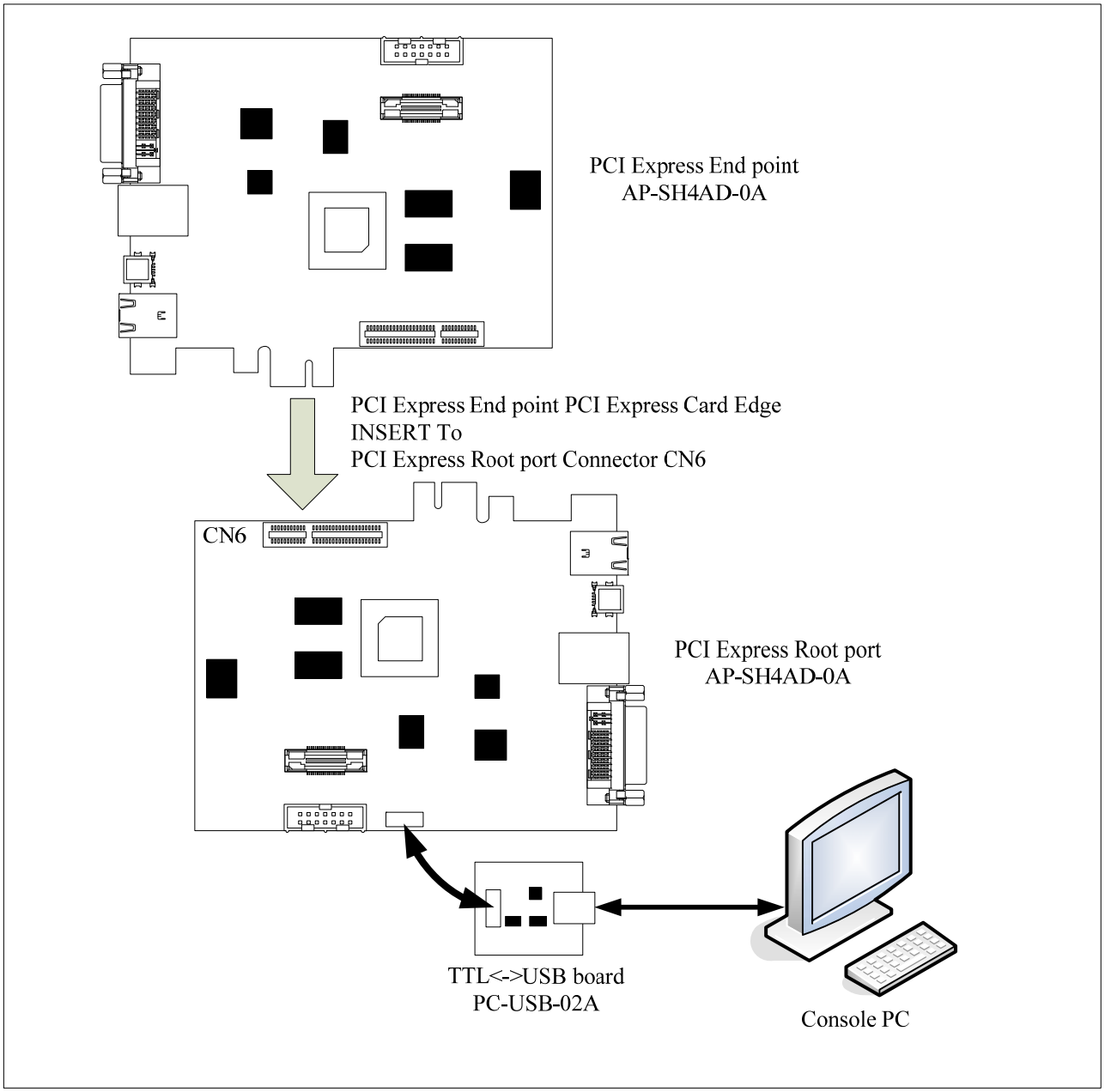


図 4.2.1.1　システム構成

4.2.2　参考プログラムの仕様

- シリアルコンソールの初期設定
- PCIEC の初期設定（PCI Express Root port モード、PCI Express End point モード設定）
- PCI Express End point の VenderID、DeviceID の表示
- PCI Express End point のメモリ空間、IO 空間へのデータ転送（送受信）
- 内蔵 DMA を使用した PCI Express End point のメモリ空間へのデータ転送（送受信）
- 以下は未サポート

  内蔵 DMA を使用したストライド転送、コマンドチェーン

  メッセージ送受信

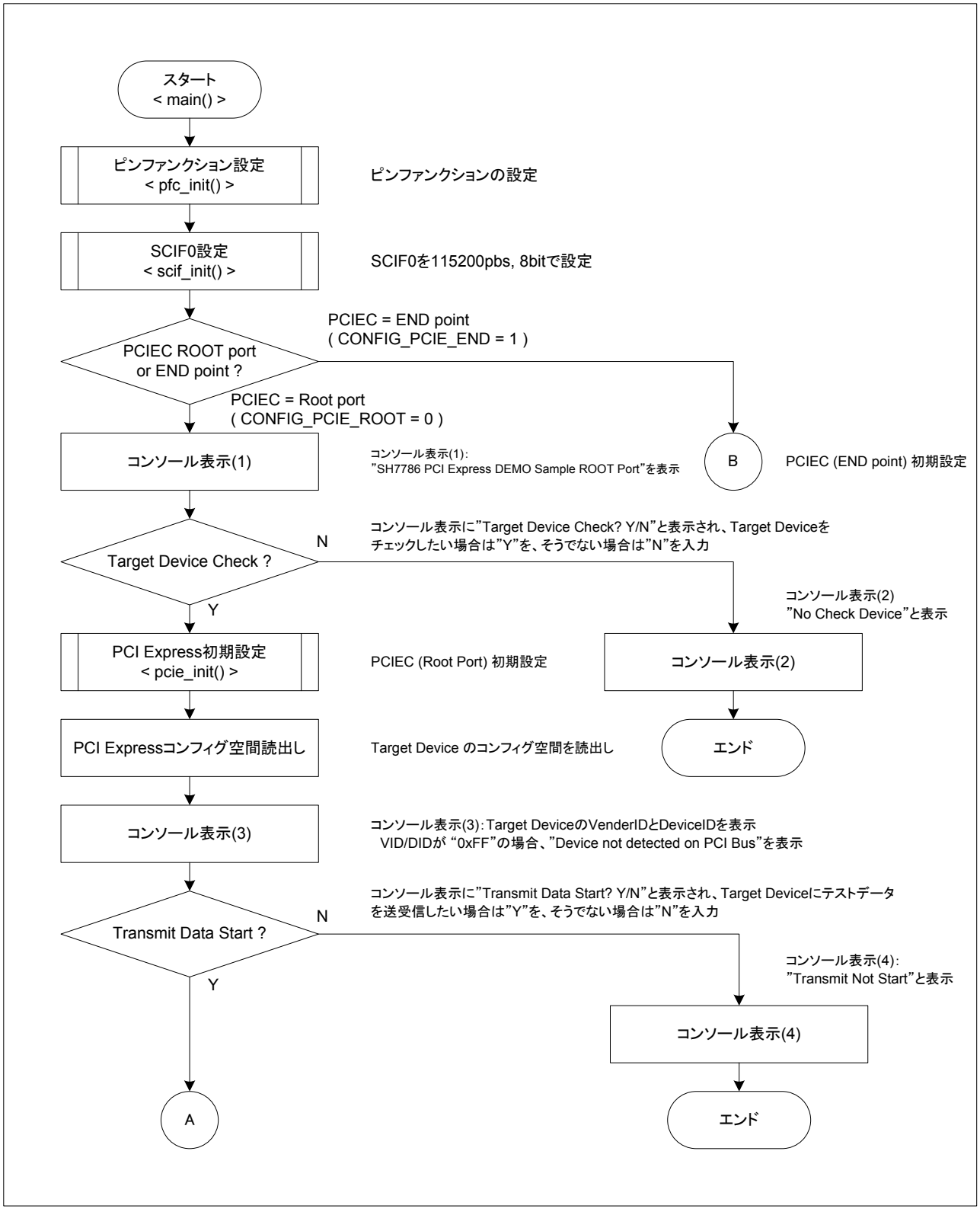
  INTx/MSI による割り込み

  リンクパワー制御機能（L0、L0s、L1、L3 ステート）

4.2.3　参考プログラムのフローチャート

(1)　main()関数フローチャート

パワーオンリセットから LBSC、DBSC3 の初期設定を実行した後、main()関数からの処理フローを示します。



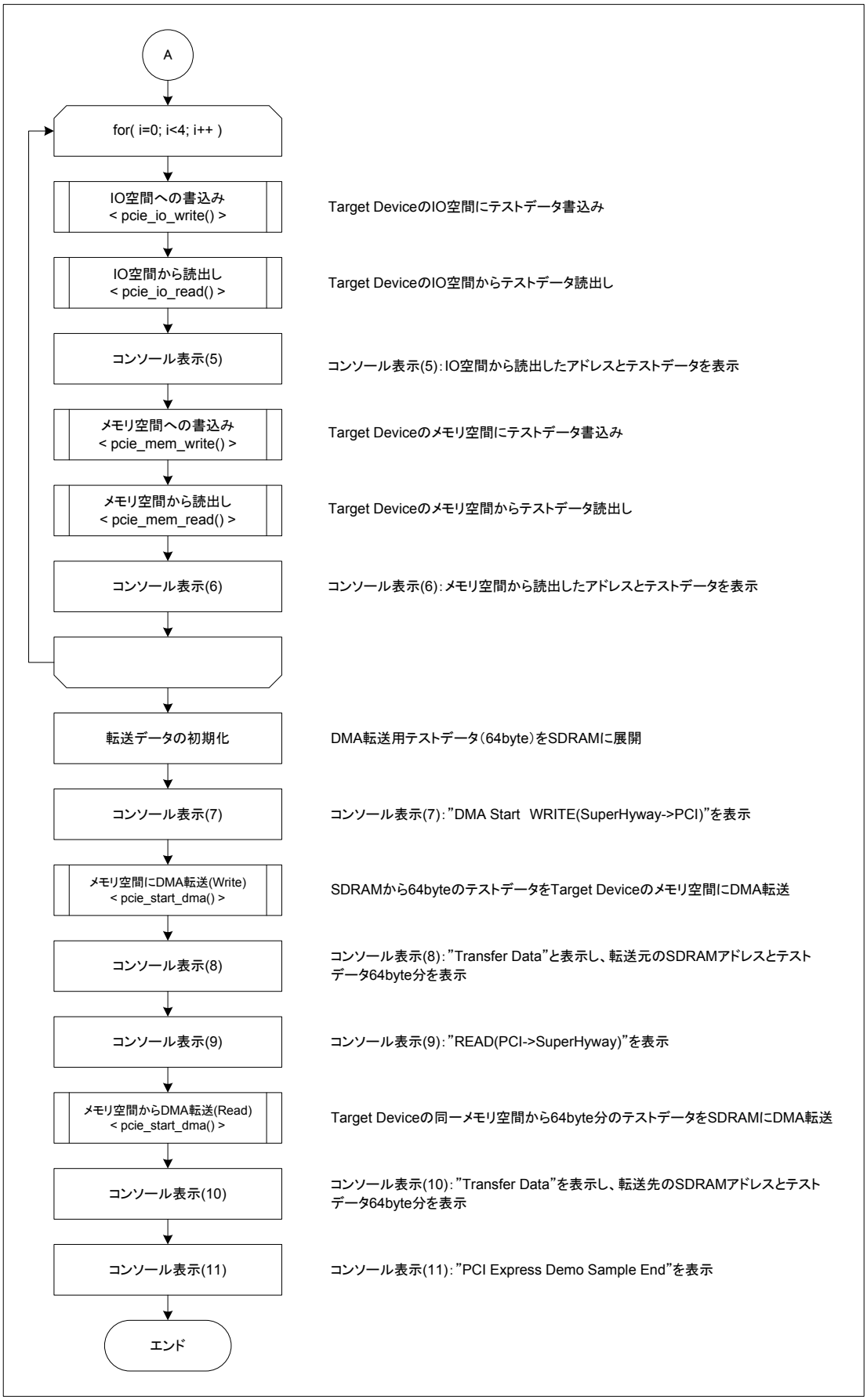


図 4.2.4.1　main フロー（PCI Express Root port）

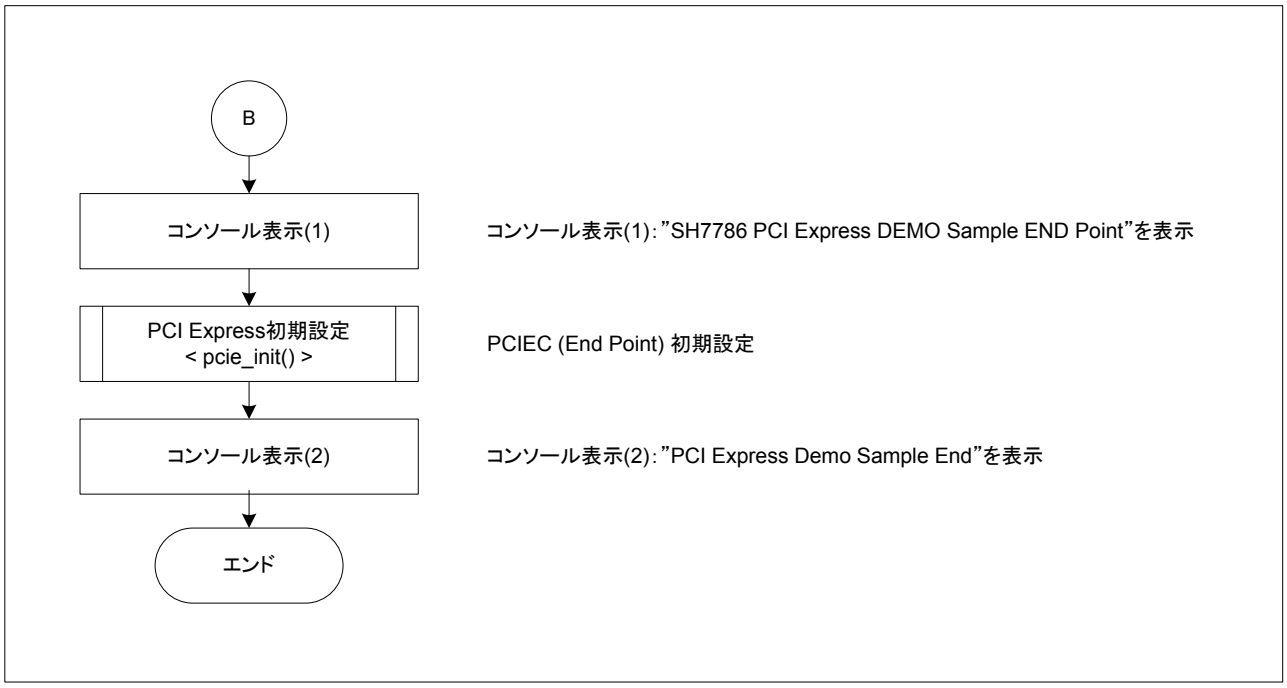


図 4.2.4.2　main フロー（PCI Express End point）

(2)　ピンファンクション設定フローチャート

ピンファンクション設定の処理フローを示します。

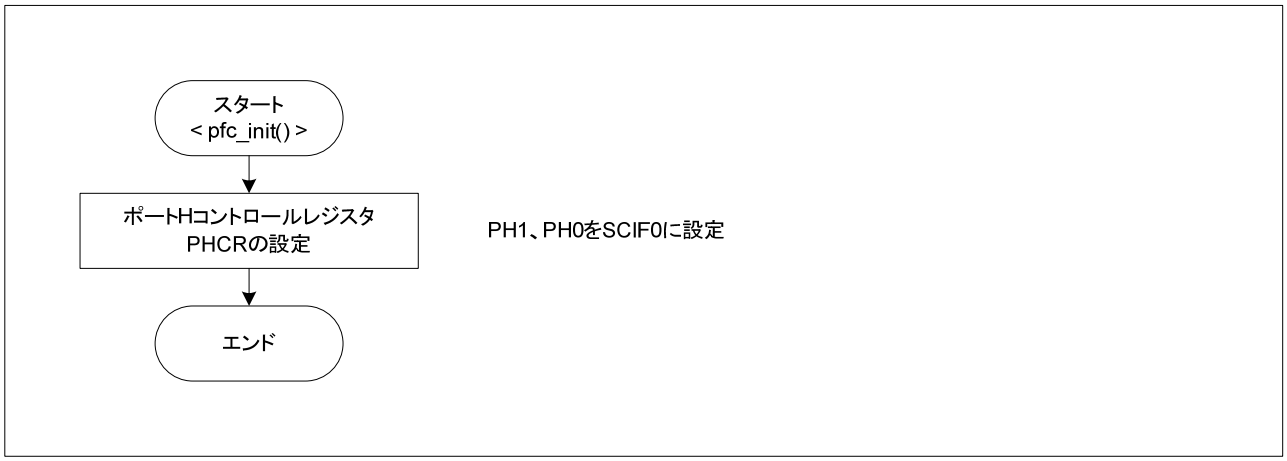


図 4.2.4.3　ピンファンクション設定フロー

(3)　SCIF0 初期設定フローチャート

シリアルコンソールとして使用する SCIF0 初期設定の処理フローを示します。

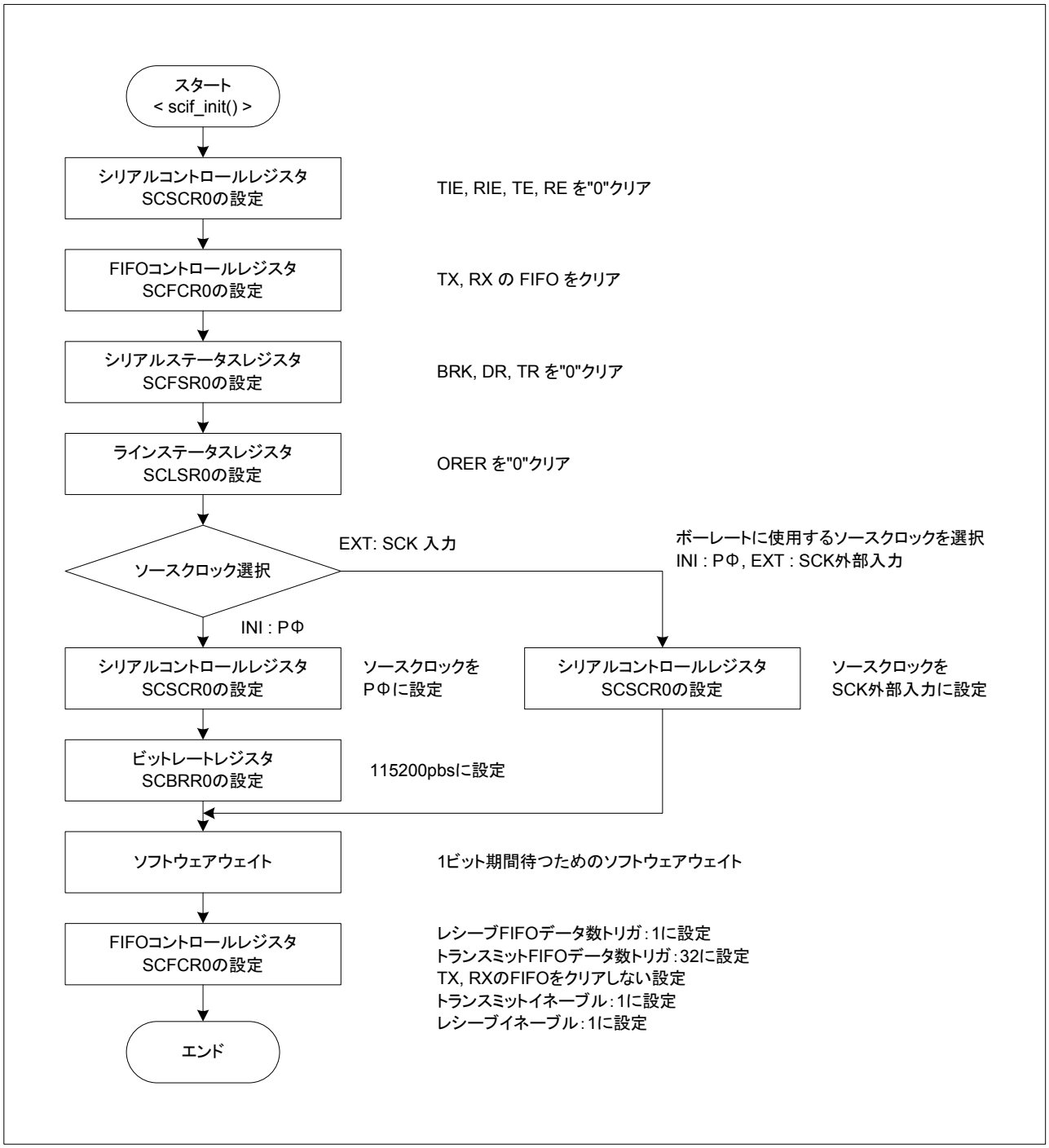


図 4.2.4.4　SCIF0 初期設定フロー

(4)　PCI Express バス初期設定フローチャート

PCI Express バス初期設定の処理フローを示します。

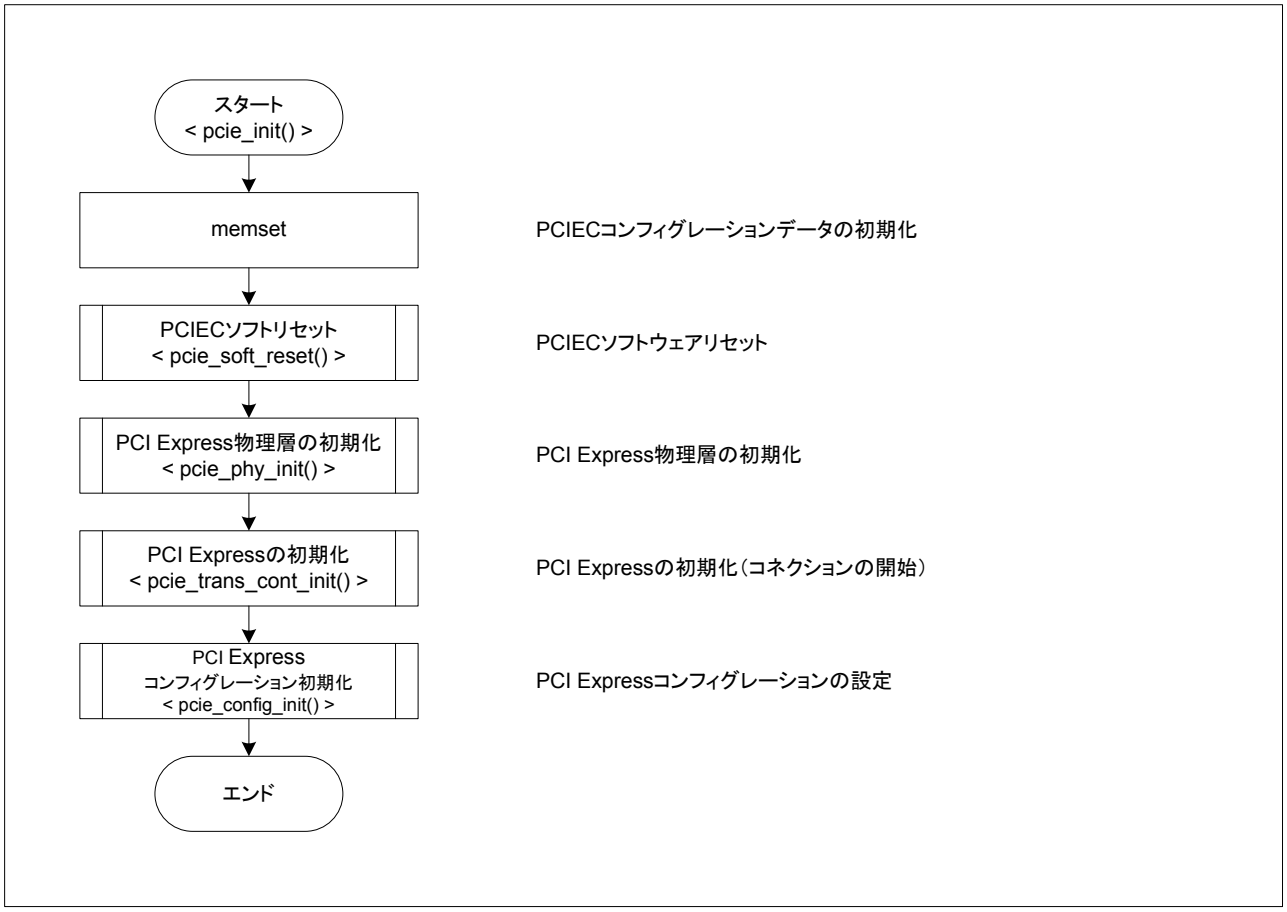


図 4.2.4.5　　PCI Express バス初期設定フロー

(5)　PCIECソフトウェアリセットフローチャート

PCIEC ソフトウェアリセット処理フローを示します。

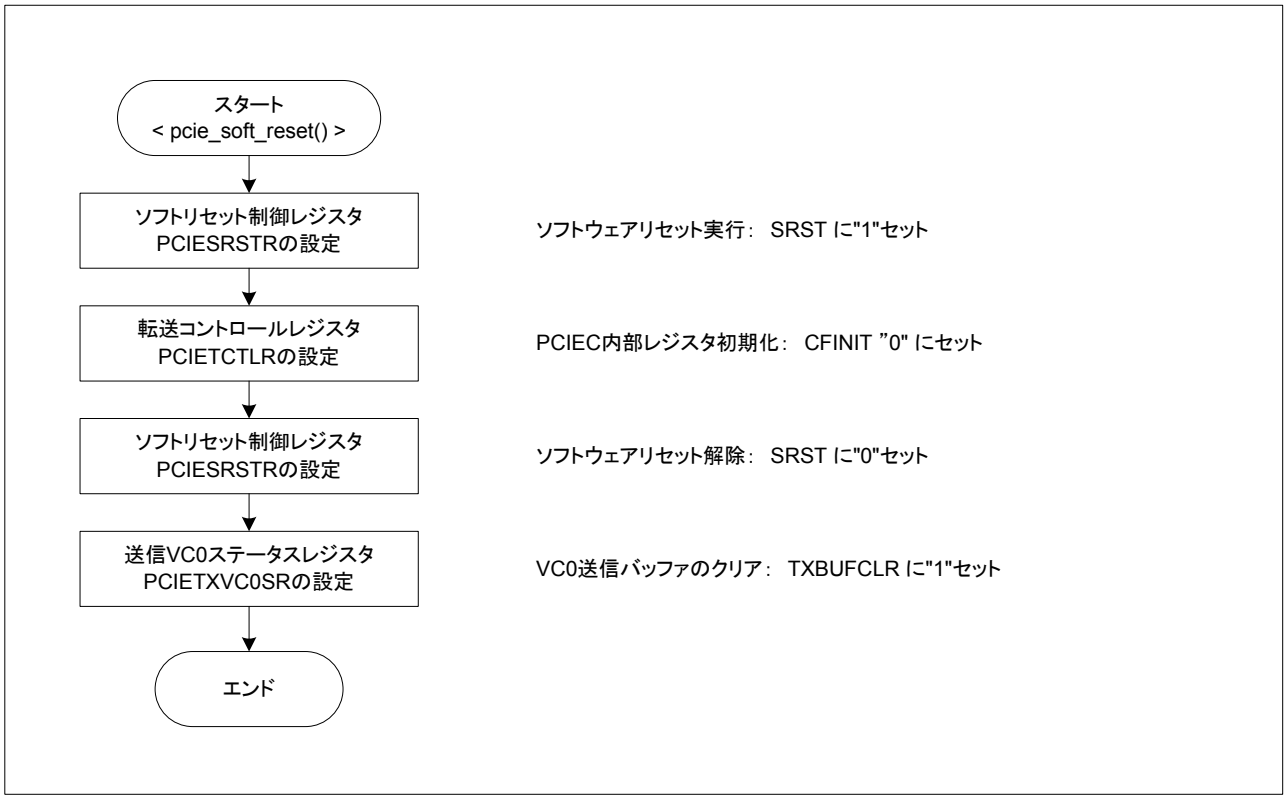


図 4.2.4.6　PCIEC ソフトウェアリセットフロー

(6)　PCIEC物理層初期化フローチャート

PCIEC 物理層初期化の処理フローを示します。

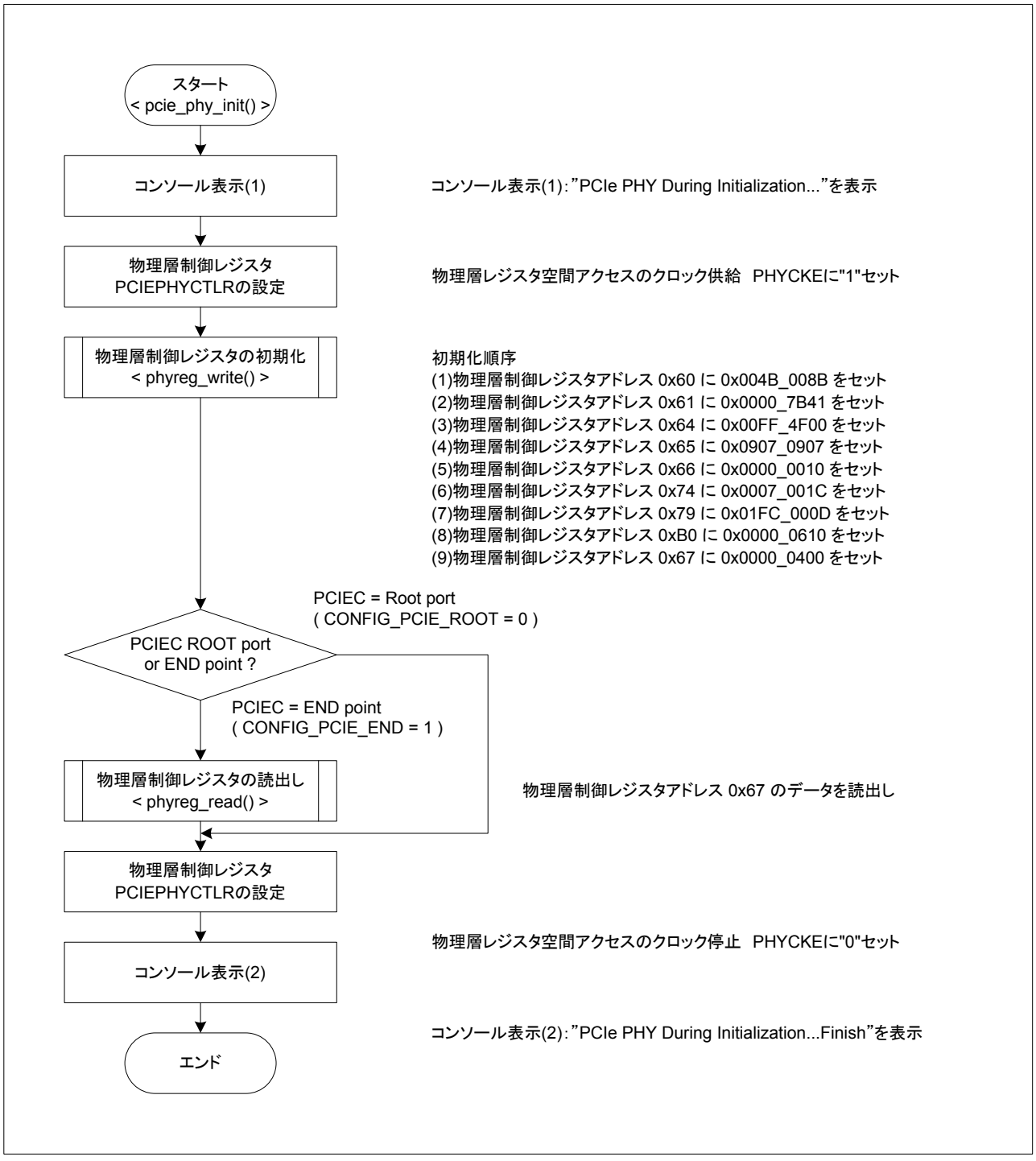


図 4.2.4.7　PCIEC 物理層初期化フロー

(7)　PCIEC初期化フローチャート

PCIEC 初期化の処理フローを示します。

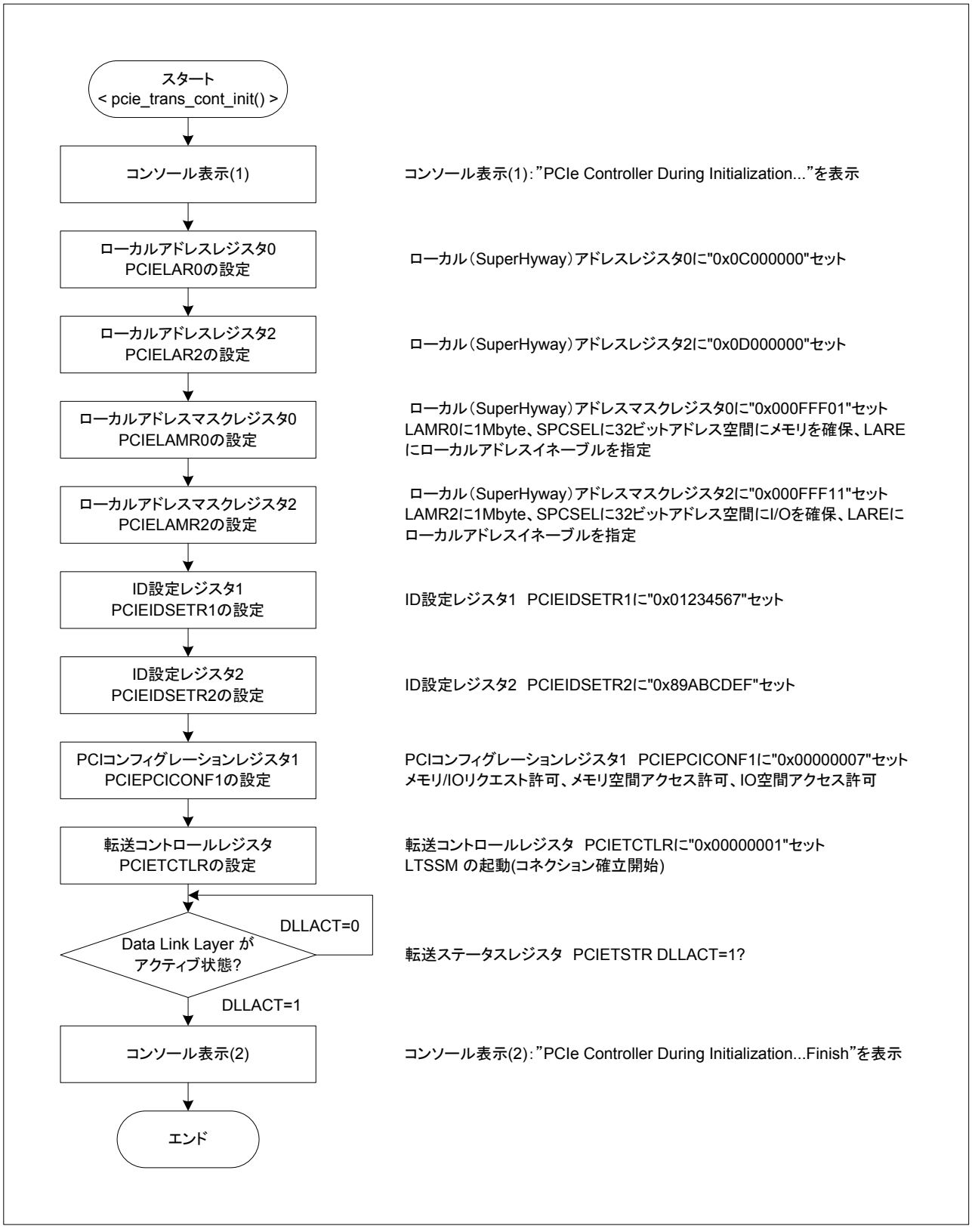
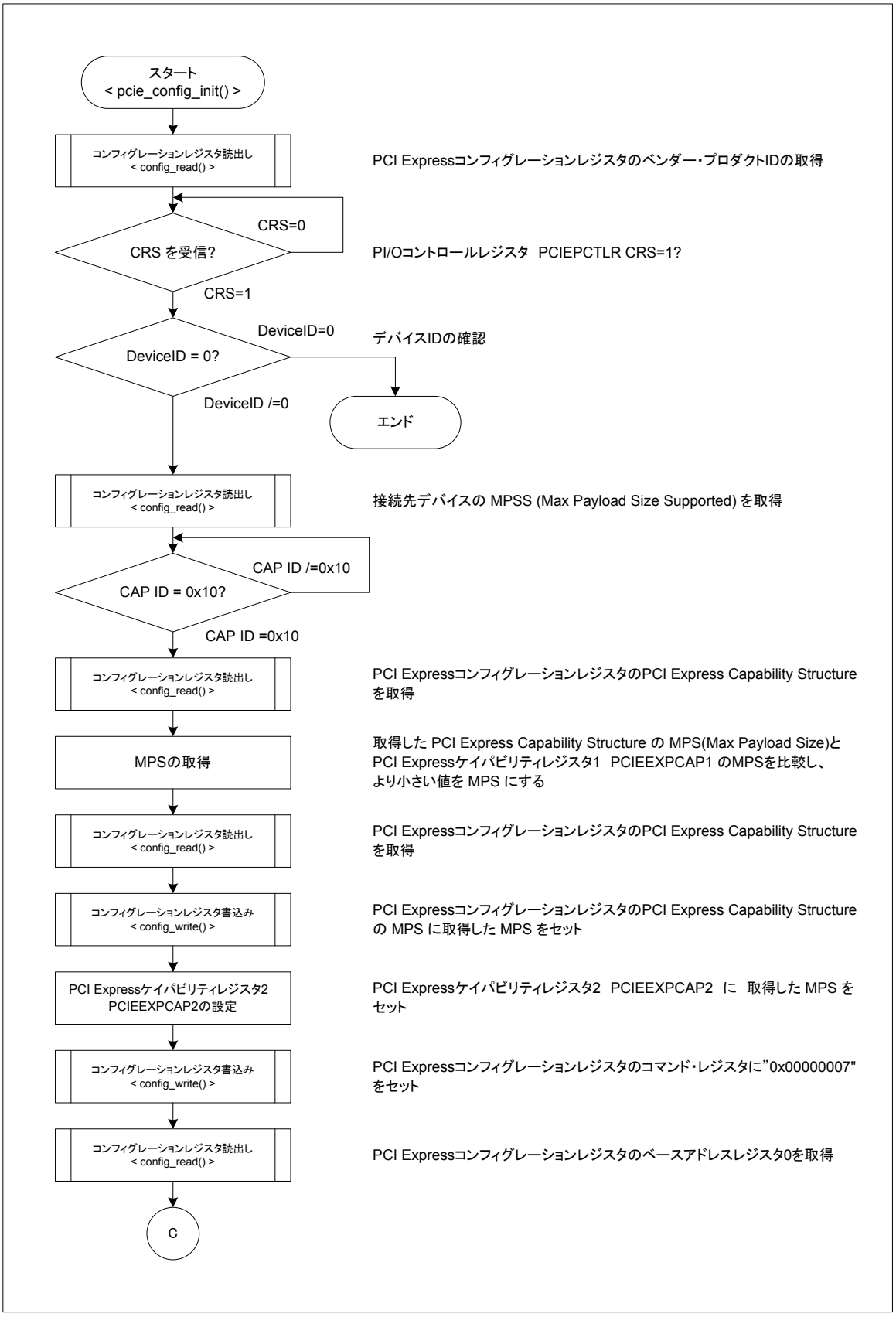


図 4.2.4.8　PCIEC 初期化フロー

(8)　PCI Expressコンフィグレーションレジスタ初期設定フローチャート

PCI Express コンフィグレーションレジスタ初期設定の処理フローを示します。



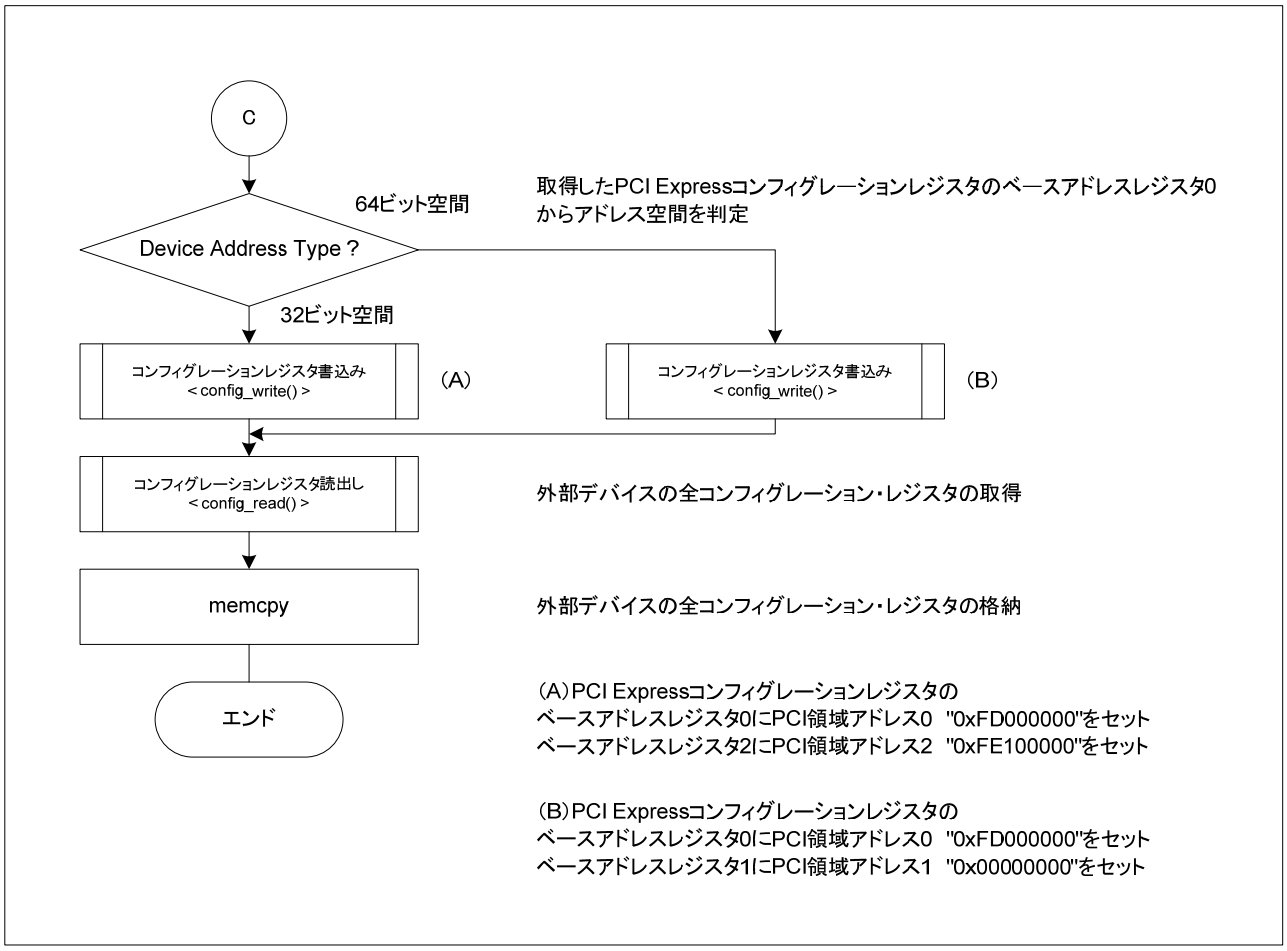


図 4.2.4.9　PCI Express コンフィグレーションレジスタ初期設定フロー

(9)　PCI Expressメモリ転送（Write）フローチャート

PCI Express メモリ転送（Write）の処理フローを示します。

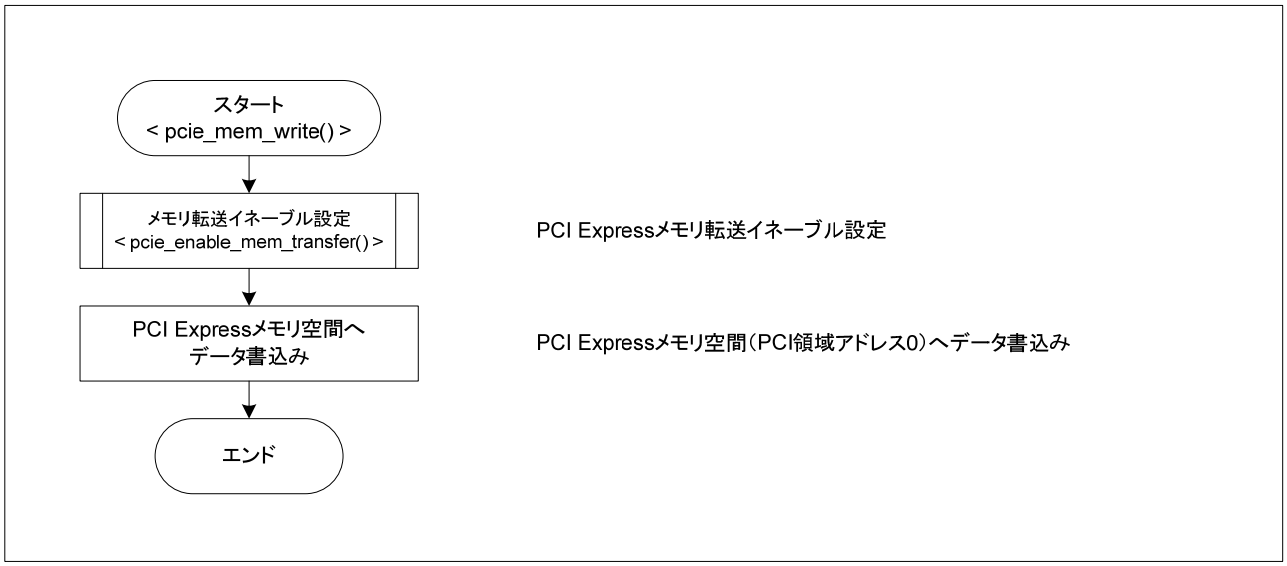


図 4.2.4.10　PCI Express メモリ転送（Write）フロー

(10)　PCI Expressメモリ転送（Read）フローチャート

PCI Express メモリ転送（Read）の処理フローを示します。

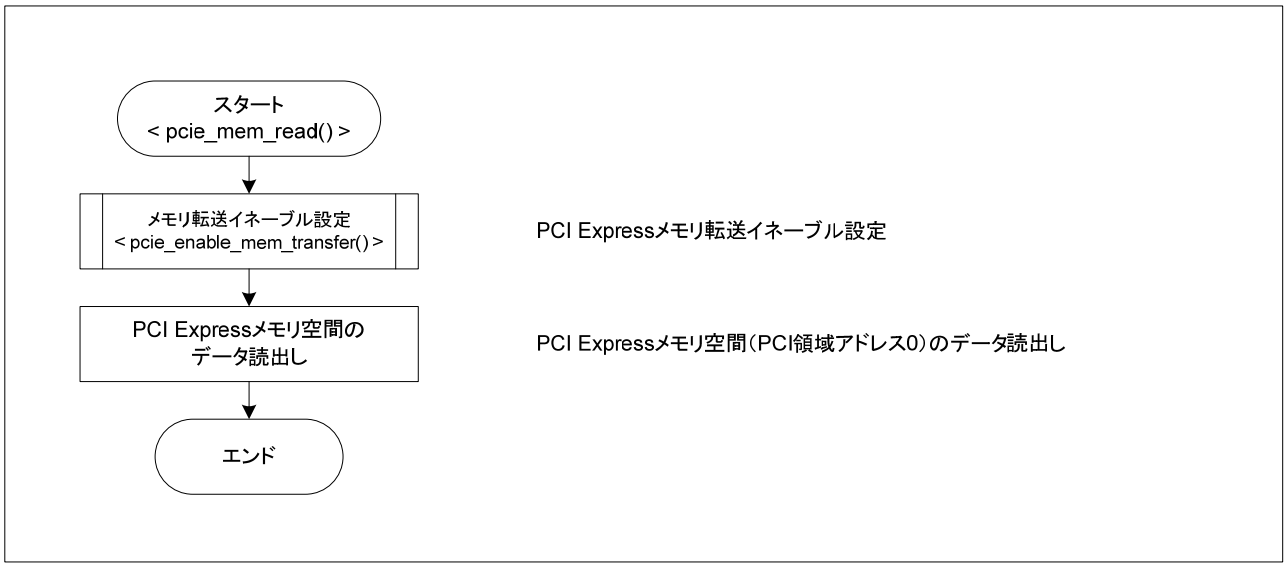


図 4.2.4.11　PCI Express メモリ転送（Read）フロー

(11) PCI Expressメモリ転送イネーブル設定フローチャート

PCI Express メモリ転送イネーブル設定の処理フローを示します。

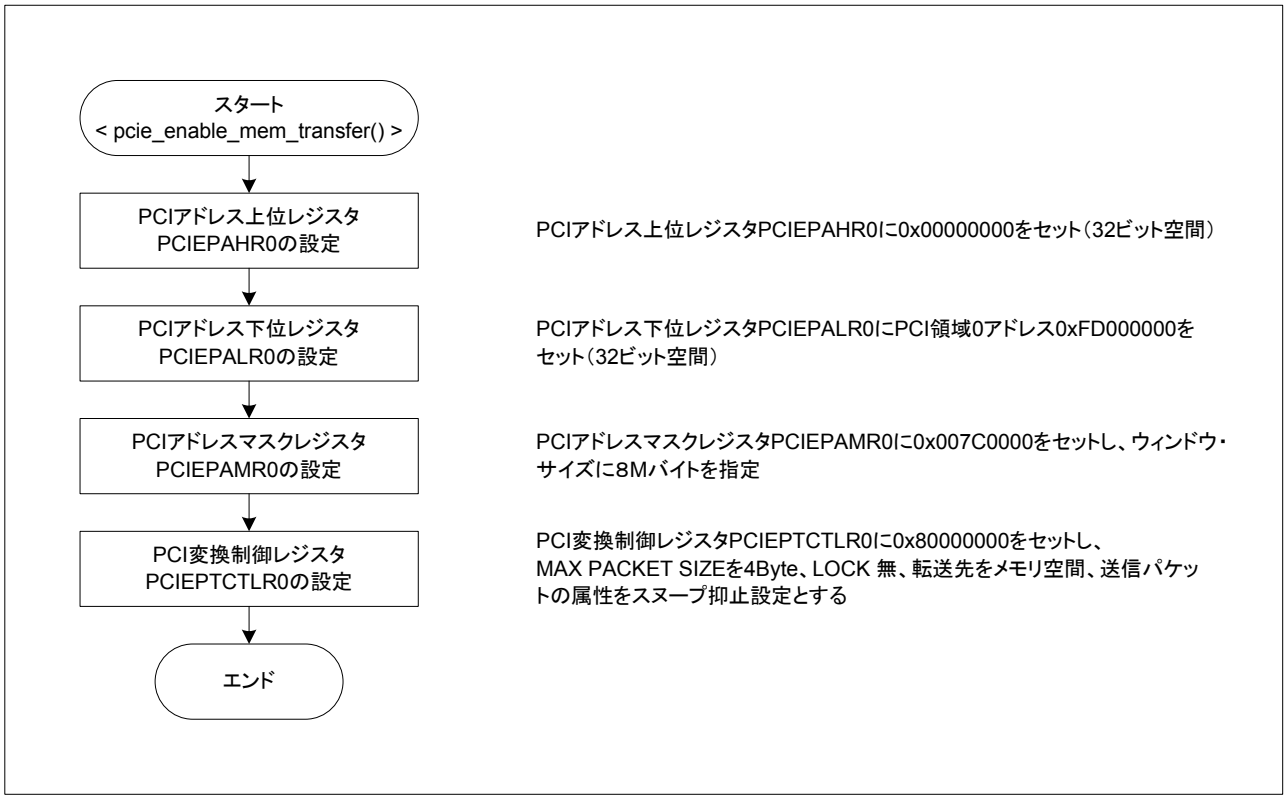


図 4.2.4.12　PCI Express メモリ転送イネーブル設定フロー

(12) PCI Express IO転送（Write）フローチャート

PCI Express IO 転送（Write）の処理フローを示します。

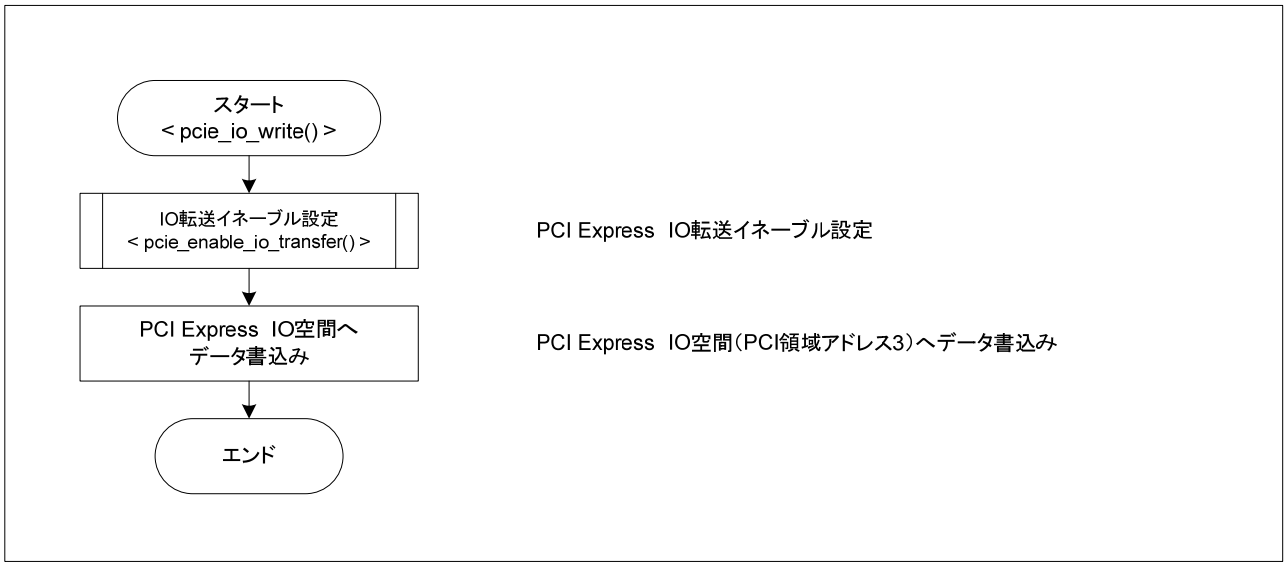


図 4.2.4.13　PCI Express IO 転送（Write）フロー

(13) PCI Express IO転送（Read）フローチャート

PCI Express IO 転送（Read）の処理フローを示します。

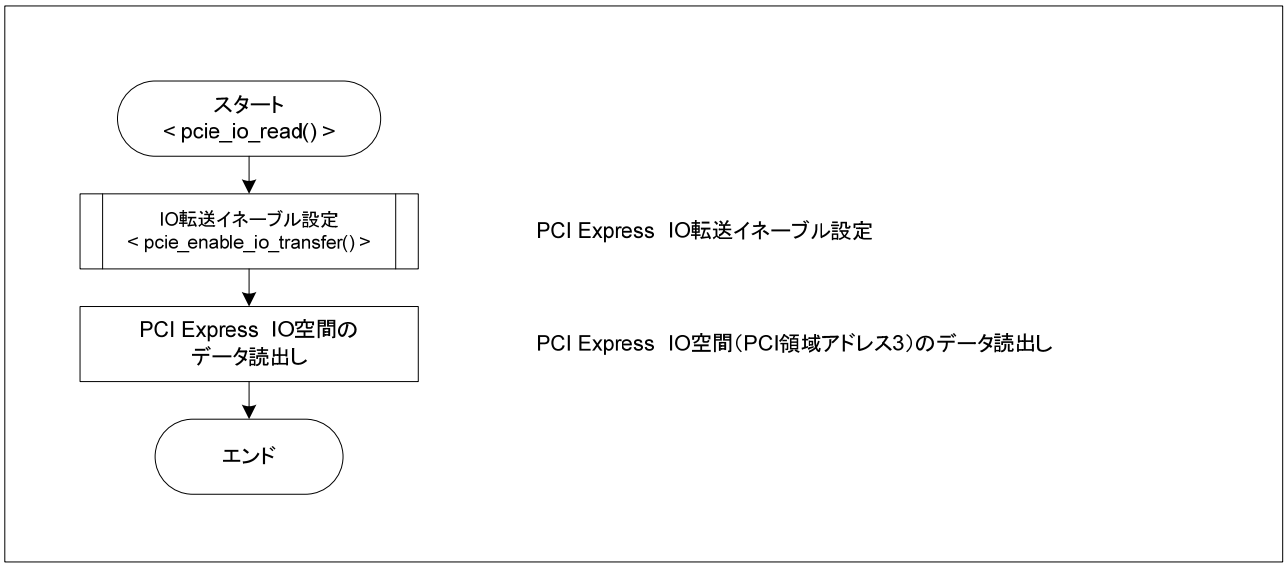


図 4.2.4.14　PCI Express IO 転送（Read）フロー

(14) PCI Express IO転送イネーブル設定フローチャート

PCI Express IO 転送イネーブル設定の処理フローを示します。

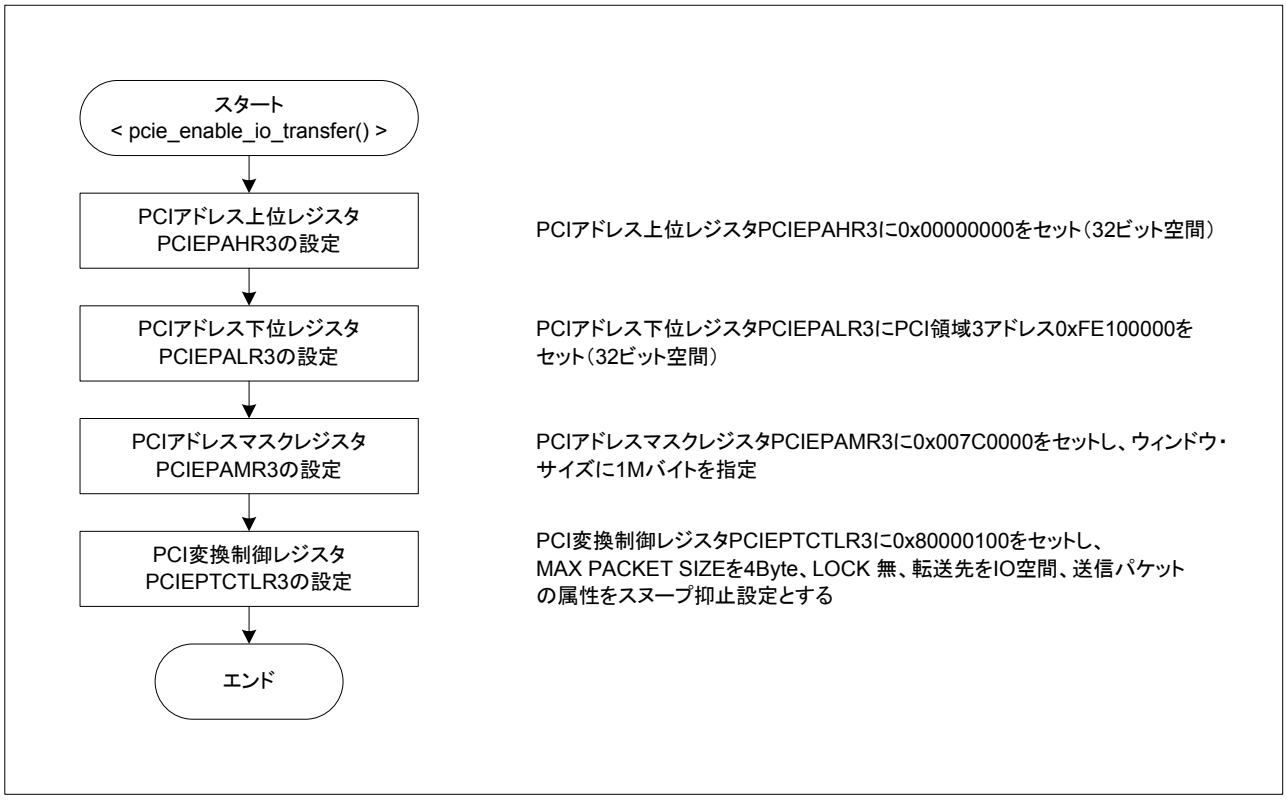


図 4.2.4.15　PCI Express IO 転送イネーブル設定フロー

(15) PCI Express物理層制御レジスタへの書込みフローチャート

PCI Express 物理層制御レジスタへの書込み処理フローを示します。

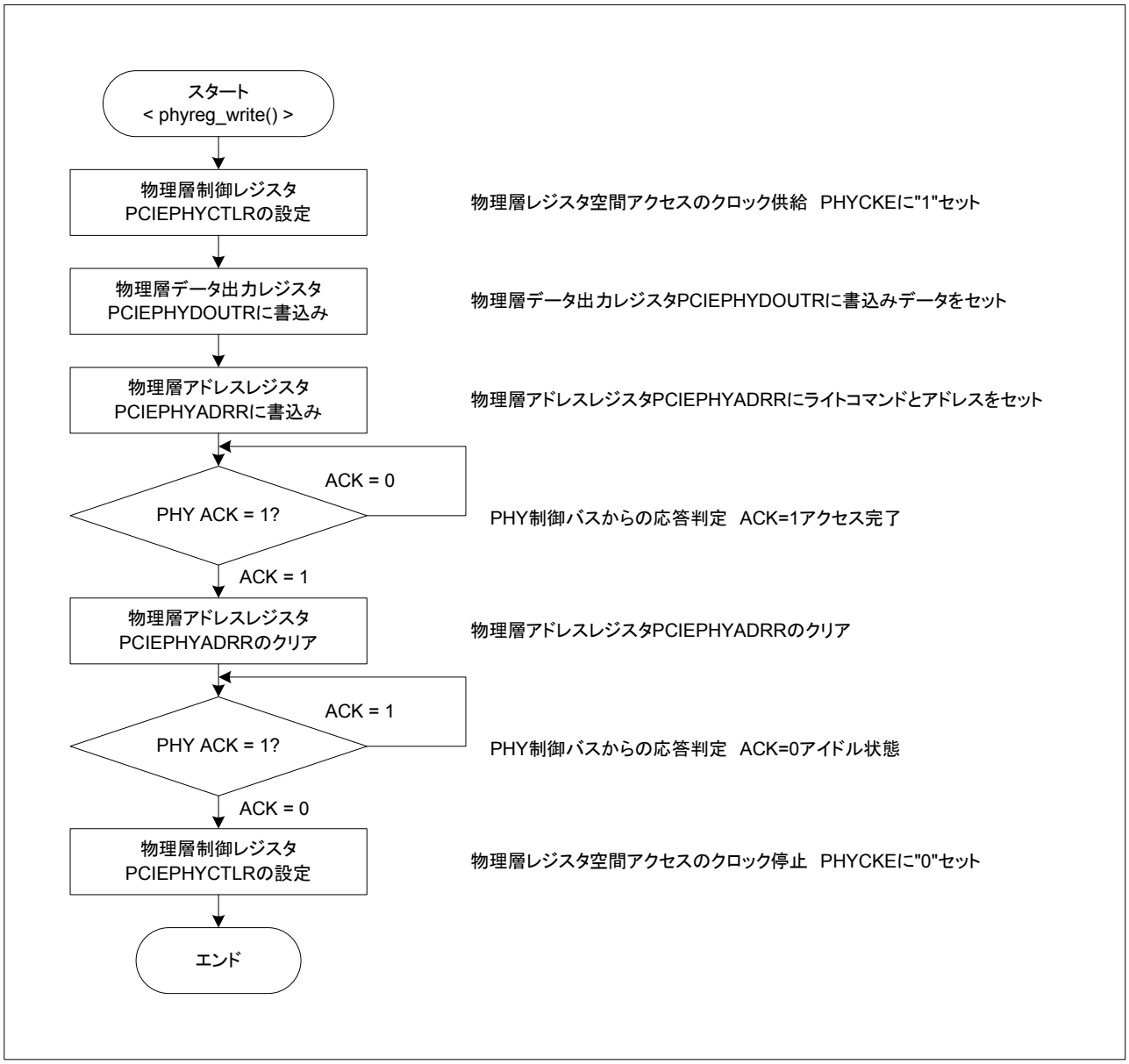


図 4.2.4.16　PCI Express 物理層制御レジスタへの書込みフロー

(16) PCI Expressコンフィグレーションレジスタの読出しフローチャート

PCI Express コンフィグレーションレジスタの読出し処理フローを示します。

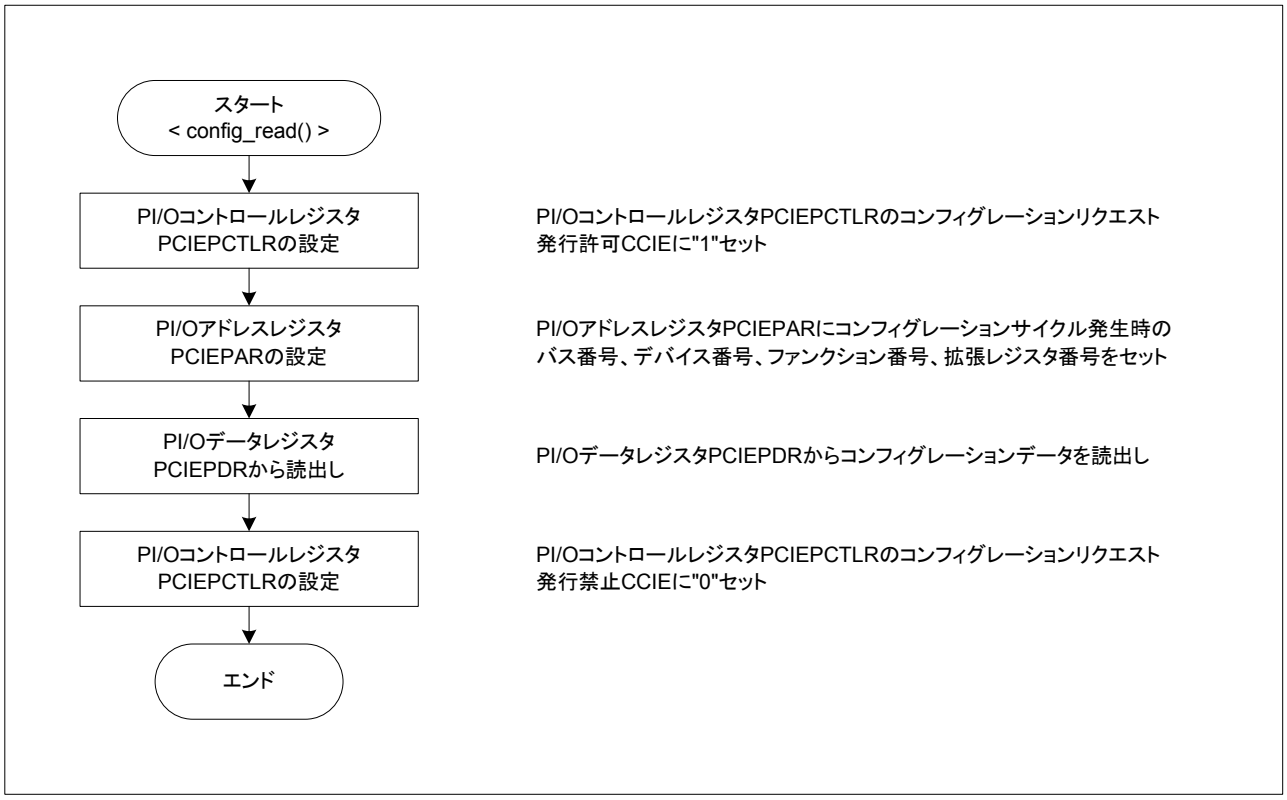


図 4.2.4.17　PCI Express コンフィグレーションレジスタの読出しフロー

(17) PCI Expressコンフィグレーションレジスタへの書込みフローチャート

PCI Express コンフィグレーションレジスタへの書込み処理フローを示します。

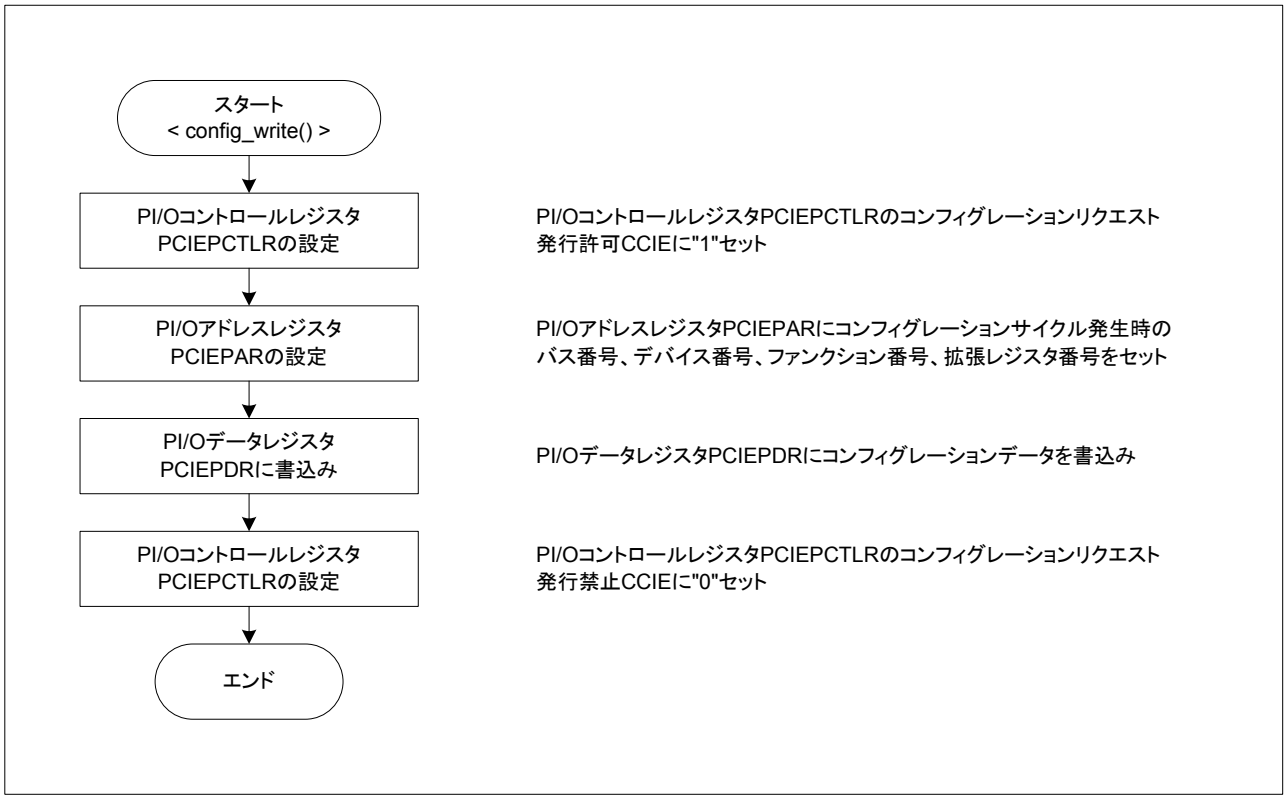
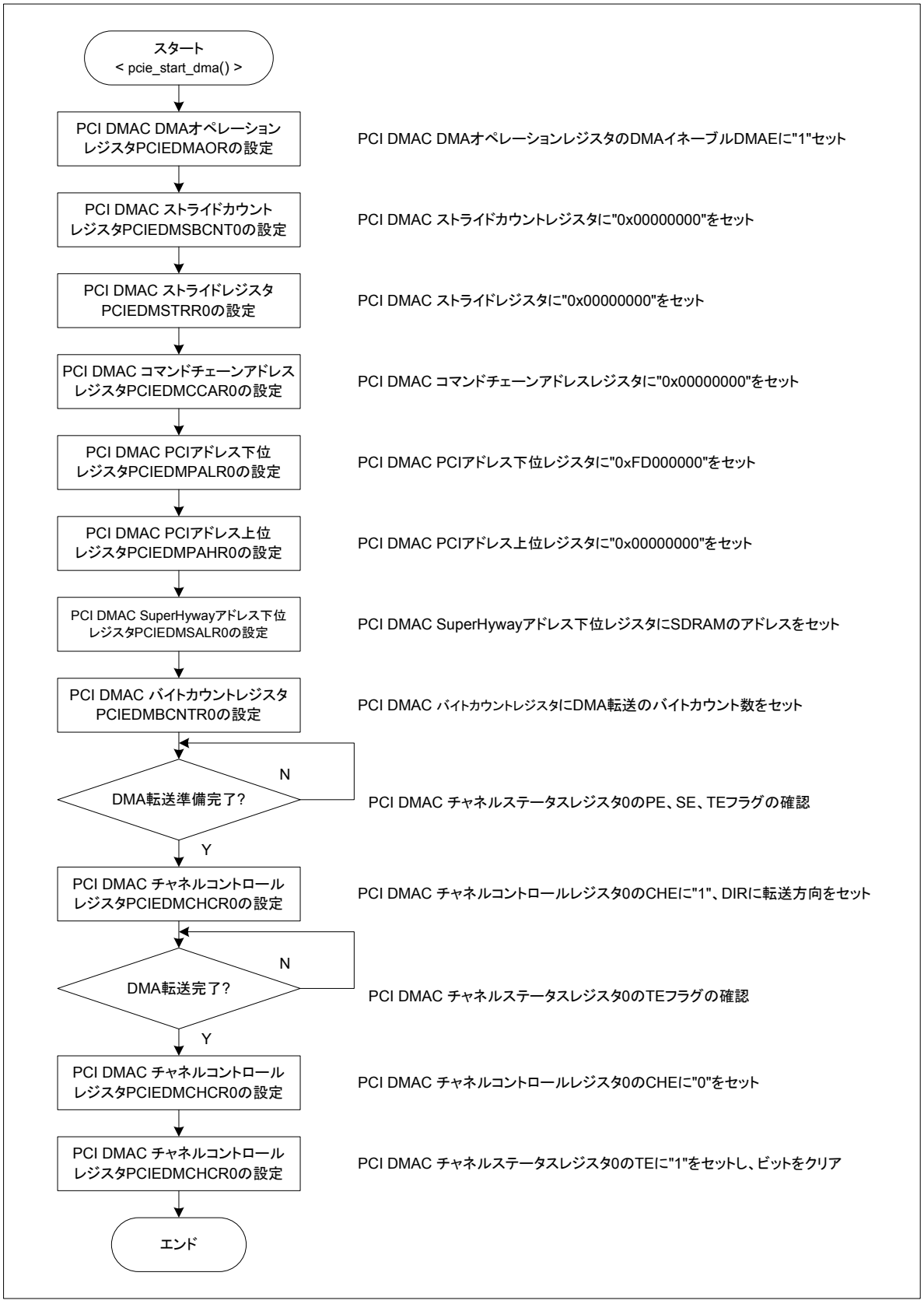


図 4.2.4.18　PCI Express コンフィグレーションレジスタの書込みフロー

(18) PCIEC DMA転送フローチャート

PCIEC 内蔵 DMAC 転送の処理フローを示します。



4.2.4　参考プログラム例

(1)　”PCIe_DemoSample.c”

サンプルプログラムのメイン関数です。

```
001 /***********************************************************************/
002 /*                                                                     */
003 /*    FILE             :PCIe_DemoSample.c                              */
004 /*    DATE             :Wed, Nov 17, 2010                              */
005 /*    DESCRIPTION :Main Program                                        */
006 /*    CPU TYPE         :Other                                          */
007 /*                                                                     */
008 /*    This file is generated by Renesas Project Generator (Ver.4.16).  */
009 /*                                                                     */
010 /***********************************************************************/
011
012
013
014 //#include "typedefine.h"
015 #include "config.h"
016 #include "pcie.h"
017
018 #ifdef __cplusplus
019 //#include <ios>                          // Remove the comment when you use ios
020 //_SINT ios_base::Init::init_cnt;         // Remove the comment when you use ios
021 #endif
022
023 void main(void);
024 #ifdef __cplusplus
025 extern "C" {
026 void abort(void);
027 }
028 #endif
029
030 /* ==== Variable declaration ==== */
031 //extern static PCIE_CONF_DATA conf_data;
032 #define        BUFF_MAX      7
033 #define TransByte      64
034
```

```
035 /* ==== Function declaration ==== */
036 volatile void BuffClear(char *pBuff, int size);
037 volatile void SdramDataInit(int cnt);
038
039
040 void pfc_init(void);
041
042 /*""FUNC COMMENT""************************************************
043 * ID                    :
044 * Outline               : Sample program main
045 *                       :
046 * Include               :
047 * Declaration           : void main(void)
048 * Description           : Main program
049 *                       :
050 *                       :
051 *                       :
052 *                       :
053 *                       :
054 * Limitation            :
055 *                       :
056 * Argument              : none
057 * Return Value          : none
058 * Calling Functions :
059 *""FUNC COMMENT END""*********************************************/
060 void main(void)
061 {
062     volatile static int ret = 0;
063     int i, j;
064     char KeyBuff[BUFF_MAX];
065     unsigned long data;
066     int startadd;
067
068     pfc_init();
069     ret = scif_init();
070
071 #ifdef CONFIG_PCIE_ROOT
072     if( ret == 0 )
073         printf("¥n¥r_/_/_/ SH7786 PCI Express DEMO Sample ROOT Port _/_/_/¥n¥r");
074     BuffClear( KeyBuff, BUFF_MAX );                    // Buffer clear
```

```
075     printf("Target Device Check? Y/N¥n¥r");
076     while( scif_recive_data( KeyBuff ) != 0);
077     switch(KeyBuff[0]) {
078             case 'Y' :
079                                     pcie_init(CONFIG_PCIE_ROOT);
080                                     pcie_check(CONFIG_PCIE_ROOT);
081                                     break;
082             case 'N' :
083                                     printf("Not Check Device¥n¥r");
084                                     break;
085             default :
086                                     break;
087     }
088     delay(1000);
089     BuffClear( KeyBuff, BUFF_MAX );                         // Buffer clear
090     printf("Transmit Data Start? Y/N¥n¥r");
091     while( scif_recive_data( KeyBuff ) != 0);
092     switch(KeyBuff[0]) {
093             case 'Y' :
094                     printf("Transmit Start¥n¥r");
095                     for(i=0;i<4;i++) {
096                             pcie_io_write(CONFIG_PCIE_ROOT, i*4, i+1, Long);
097                             data = pcie_io_read(CONFIG_PCIE_ROOT, i*4, Long);
098                             printf("Addr = %08x, Data = %08x¥n¥r", 0xFE100000 + (i*4), data);
099                             pcie_mem_write(CONFIG_PCIE_ROOT, i*4, i+2, Long);
100                             data = pcie_mem_read(CONFIG_PCIE_ROOT, i*4, Long);
101                             printf("Addr = %08x, Data = %08x¥n¥r", 0xFD000000 + (i*4), data);
102                     }
103                     /* DMA Transfer Test */
104                     SdramDataInit(TransByte);               /* 64Byte data set */
105                     printf("¥r¥nDMA Start ");
106
107                     printf("¥n¥rWRTITE(SuperHyway->PCI)¥n");
108                     ret = pcie_start_dma(CONFIG_PCIE_ROOT, PCIE_AREA_ADDR,
                                sdram_data_area, PCIE_WRITE, TransByte);
109                     if(!ret) {
110                             printf("¥n¥rPCIE DMA Error¥n");
111                             break;
112                     }
113
```

```
114                     startadd = 0xA0000000 | sdram_data_area;
115                     printf("¥r¥nTransfer Data¥n¥r");
116                     for(i=0;i<(TransByte/16);i++) {
117                             for(j=0;j<4;j++) {
118                                     printf("%08x ", (*(unsigned long *)startadd));
119                                     startadd += 4;
120                                     }
121                             printf("¥n¥r");
122                     }
123
124
125                     printf("¥r¥nREAD(PCI->SuperHyway)¥n");
126                     ret = pcie_start_dma(CONFIG_PCIE_ROOT, PCIE_AREA_ADDR,
                                sdram_data_area + 0x1000, PCIE_READ, TransByte);
127                     if(!ret) {
128                             printf("¥n¥rPCIE DMA Error¥n");
129                             break;
130                     }
131                     startadd = 0xA0001000 | sdram_data_area;
132                     printf("¥r¥nTransfer Data¥n¥r");
133                     for(i=0;i<(TransByte/16);i++) {
134                             for(j=0;j<4;j++) {
135                                     printf("%08x ", (*(unsigned long *)startadd));
136                                     startadd += 4;
137                             }
138                     printf("¥n¥r");
139                     }
140
141                     break;
142             case 'N' :
143                     printf("Transmit Not Start¥n¥r");
144                     break;
145             default :
146                     break;
147     }
148
149 #else
150     if( ret == 0 )
151             printf("¥n¥r_/_/_/ SH7786 PCI Express DEMO Sample END Point _/_/_/¥n¥r");
152     pcie_init(CONFIG_PCIE_END);
```

```
153
154 #endif
155       printf("PCI Express Demo Sample End¥n¥r");
156
157 }
158
159 /*""FUNC COMMENT""*****************************************************
160 * ID                    :
161 * Outline               : Sample program main
162 *                       : (PCI Express)
163 * Include               :
164 * Declaration           : void pfc_init( void )
165 * Description           : A set of a pin function
166 *                       :
167 *                       :
168 *                       :
169 *                       :
170 *                       :
171 * Limitation            :
172 *                       :
173 * Argument              : none
174 * Return Value          : none
175 * Calling Functions :
176 *""FUNC COMMENT END""************************************************/
177 void pfc_init(void)
178 {
179       /* SCIF0 */
180       GPIOR.PHCR.WORD = 0xFC30;
181 }
182
183 /*""FUNC COMMENT""*****************************************************
184 * ID                    :
185 * Outline               : Sample program main
186 *                       : (PCI Express)
187 * Include               :
188 * Declaration           : void BuffClear(char *pBuff, int size)
189 * Description           : A initialization of the buffer for serial receive datas
190 *                       :
191 *                       :
192 *                       :
```

```
193 *                           :
194 *                           :
195 * Limitation              :
196 *                           :
197 * Argument                : *pBuff:Buffer,   size:Buffer size
198 * Return Value            : none
199 * Calling Functions :
200 *""FUNC COMMENT END""**************************************************/
201 volatile void BuffClear(char *pBuff, int size)
202 {
203       int i;
204       for( i = 0; i < size; i++ )     /* A clear of a serial data receiving workpiece          */
205       {
206                 *( pBuff + i ) = 0;
207       }
208 }
209
210 /*""FUNC COMMENT""**************************************************
211 * ID                          :
212 * Outline                 : Sample program main
213 *                           : (PCI Express)
214 * Include                 :
215 * Declaration             : volatile void SdramDataInit(int cnt)
216 * Description             : A initialization of the data for sdram
217 *                           :
218 *                           :
219 *                           :
220 *                           :
221 *                           :
222 * Limitation              :
223 *                           :
224 * Argument                : none
225 * Return Value            : none
226 * Calling Functions :
227 *""FUNC COMMENT END""**************************************************/
228 volatile void SdramDataInit(int cnt)
229 {
230       int i;
231       int sdram_add = 0xA0000000 | sdram_data_area;
232       for( i = 0; i < cnt/4; i++ )   /* SDRAM Data Initialize cnt/4 byte          */
```

```
233       {
234                 (*(unsigned long *)sdram_add) = i;
235                 sdram_add += 4;
236       }
237 }
238
239
240 #ifdef __cplusplus
241 void abort(void)
242 {
243
244 }
245 #endif
```

(2)　”config.h”

サンプルプログラムのメイン関数で使用するヘッダファイルです。

```
01 #ifndef _CONFIG_H_
02 #define _CONFIG_H_
03
04 #include <stdarg.h>
05 #include <stdio.h>
06 #include <stdlib.h>
07 #include "iodefine.h"
08
09 #define         Long    4
10 #define         Word    2
11 #define         Byte    1
12
13 /* SCIF */
14 #define         CONFIG_PCLK  44400000
15 #define         CONFIG_SCIF0
16 //#define       CONFIG_SCIF1
17 //#define       CONFIG_SCIF2
18 //#define       CONFIG_SCIF3
19 //#define       CONFIG_SCIF4
20 //#define       CONFIG_SCIF5
21 //#define       CONFIG_SCIF_CLK_EXTERNAL
22 #define         CONFIG_SCIF_CLK_PCLK        CONFIG_PCLK
23 #define         CONFIG_BPS    115200
24
25
26 //#define CONFIG_PCIE_ROOT          0
27 //#define       CONFIG_PCIE_END               1
28
29 #undef printf
30 #define printf  scif_printf
31
32 /*** SCIF ***/
33 extern void delay( int cnt );
34 extern int scif_init(void);
35 extern char     scif_recive_data( char      *Data );
36 extern char     scif_recive_data_byte( char         *Data );
```

```
37 extern void scif_transmit_data( char *Data );
38 extern void scif_transmit_data_byte( char *Data );
39 extern void scif_printf(char* str, ...);
40
41 /*** PCIe ***/
42 extern void pcie_init(int sel);
43 extern void pcie_check(int sel);
44 extern void pcie_enable_mem_transfer(int sel);
45 extern long pcie_mem_write(int sel, unsigned long addr, unsigned long data, unsigned long size);
46 extern unsigned long pcie_mem_read(int sel, unsigned long addr, unsigned long size);
47 extern void pcie_enable_io_transfer(int sel);
48 extern long pcie_io_write(int sel, unsigned long addr, unsigned long data, unsigned long size);
49 extern unsigned long pcie_io_read(int sel, unsigned long addr, unsigned long size);
50
51 #endif /* _CONFIG_H_ */
```

(3)　”pcie.c”

PCIEC 初期化関数、PCI Express 制御系関数、DMAC 制御関数のサンプルプログラムです。

```
001 /***********************************************************************
002 * DISCLAIMER
003
004 * This software is supplied by Renesas Electronics    Corporation. and is only
005 * intended for use with Renesas products. No other uses are authorized.
006
007 * This software is owned by Renesas Electronics Corporation. and is protected under
008 * all applicable laws, including copyright laws.
009
010 * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
011 * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
012 * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
013 * PARTICULAR PURPOSE AND NON-INFRINGEMENT.    ALL SUCH WARRANTIES ARE EXPRESSLY
014 * DISCLAIMED.
015
016 * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
017 * ELECTRONICS CORPORATION. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
018 * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
019 * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
020 * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
021
022 * Renesas reserves the right, without notice, to make changes to this
023 * software and to discontinue the availability of this software.
024 * By using this software, you agree to the additional terms and
025 * conditions found by accessing the following link:
026 * http://www.renesas.com/disclaimer
027 ***********************************************************************/
028 /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
029 /*""FILE COMMENT""************ Technical reference data ****************
030 * System Name    : SH7786 Sample Program
031 * File Name      : scif.c
032 * Abstract       : The example of a set of PCI Express Sample Program
033 * Version        : Ver 1.00
034 * Device         : SH7786
035 * Tool-Chain     : High-performance Embedded Workshop (Version 4.07.00.007)
036 *                  : C/C++ Compiler Package for SuperH Family (V.9.3.2.0)
```

```
037 * OS                  : None
038 * H/W Platform : SH-4A Board P/N:AP-SH4AD-3A (Manufacturer:ALPHA PROJECT)
039 * Description    : It is an example program of the example of a SH7786 PCI Express set.
040 *                       :
041 * Operation       :
042 * Limitation      :
043 *                       :
044 ***********************************************************************
045 * History          : 01.Sep.2010 Ver. 1.00 First Release
046 *""FILE COMMENT END""*****************************************************/
047
048
049 #include "pcie.h"
050
051 static PCIE_CONF_DATA conf_data;
052
053 /*""FUNC COMMENT""*******************************************************
054 * ID                          :
055 * Outline                  : Sample Program Main
056 *                             : (PCI Express)
057 * Include                  :
058 * Declaration            : void pcie_enable_mem_transfer(int sel)
059 * Description            : Memory transfer significance
060 *                             :
061 *                             :
062 *                             :
063 *                             :
064 *                             :
065 * Limitation             :
066 *                             :
067 * Argument                : none
068 * Return Value          : none
069 * Calling Functions :
070 *""FUNC COMMENT END""*************************************************/
071 void pcie_enable_mem_transfer(int sel)
072 {
073       /* A set of a register（A set of a window) */
074       PCIE_REG(sel, PAHR0) = 0x00000000;              /* 32bit of a upper address */
075       PCIE_REG(sel, PALR0) = PCIE_AREA_ADDR;     /* 32 bits of a lower address */
076       PCIE_REG(sel, PAMR0) = 0x007C0000;              /* A window size is specified(8M) */
```

```
077       PCIE_REG(sel, PTCTLR0) = 0x80000000;  /* A transmitting packet property is specified */
078 }
079
080 /*""FUNC COMMENT""************************************************
081 * ID                    :
082 * Outline               : Sample Program Main
083 *                       : (PCI Express)
084 * Include               :
085 * Declaration           : long pcie_mem_write(int sel, unsigned long addr, unsigned long data, unsigned long size)
086 * Description           : It writes in a memory space
087 *                       :
088 *                       :
089 *                       :
090 *                       :
091 *                       :
092 * Limitation            :
093 *                       :
094 * Argument              : none
095 * Return Value          : -1:size error、0:Normal
096 * Calling Functions :
097 *""FUNC COMMENT END""************************************************/
098 long pcie_mem_write(int sel, unsigned long addr, unsigned long data, unsigned long size)
099 {
100       unsigned long pcie_addr;
101
102       /* Memory transfer significance */
103       pcie_enable_mem_transfer(sel);
104
105       /* The write to a memory space */
106       pcie_addr = PCIE_AREA_ADDR + addr;
107       switch(size){
108                 case 1:
109                           PCIE_WRITEB(pcie_addr, data);
110                           break;
111                 case 2:
112                           PCIE_WRITEW(pcie_addr, data);
113                           break;
114                 case 4:
115                           PCIE_WRITEL(pcie_addr, data);
116                           break;
```

```
117                 default:
118                         return -1;
119         }
120
121         return 0;
122 }
123
124 /*""FUNC COMMENT""************************************************
125 * ID                    :
126 * Outline               : Sample Program Main
127 *                       : (PCI Express)
128 * Include               :
129 * Declaration           : long pcie_endpoint_mem_burst_write(int sel, unsigned long addr, unsigned long *data,
unsigned long size)
130 * Description           : It is a burst write to a memory space
131 *                       :
132 *                       :
133 *                       :
134 *                       :
135 *                       :
136 * Limitation            :
137 *                       :
138 * Argument              : none
139 * Return Value          : -1:size error、0:Normal
140 * Calling Functions :
141 *""FUNC COMMENT END""*****************************************************/
142 long pcie_mem_burst_write(int sel, unsigned long addr, unsigned long *data, unsigned long size)
143 {
144         unsigned long pcie_addr;
145         int i;
146
147         /* Memory transfer is validated */
148         pcie_enable_mem_transfer(sel);
149
150         /* Validation of a packet joining（MAX 4096 byte） */
151         PCIE_REG(sel, PTCTLR0) = 0x9B000000;
152
153         /* The 16byte write to a memory space */
154         pcie_addr = PCIE_AREA_ADDR + addr;
155
```

```
156       for(i = 0; i < (size/4); i++) {
157                 PCIE_WRITEL(pcie_addr + i * 4, data[i]);
158       }
159
160       /* A run of a packet joining */
161       PCIE_REG(sel, PCCTLR) = 0x00000001;
162
163       return 0;
164 }
165 /*""FUNC COMMENT""*******************************************************
166 * ID                    :
167 * Outline               : Sample Program Main
168 *                       : (PCI Express)
169 * Include               :
170 * Declaration           : long pcie_mem_read(unsigned long addr, unsigned long *data, unsigned long size)
171 * Description           : Read of a memory space
172 *                       :
173 *                       :
174 *                       :
175 *                       :
176 *                       :
177 * Limitation            :
178 *                       :
179 * Argument              : none
180 * Return Value          : -1:size error、0:Normal
181 * Calling Functions :
182 *""FUNC COMMENT END""*************************************************/
183 unsigned long pcie_mem_read(int sel, unsigned long addr, unsigned long size)
184 {
185       unsigned long pcie_addr;
186       unsigned short wdata;
187       unsigned char bdata;
188       unsigned long data;
189
190       /* Memory transfer is validated */
191       pcie_enable_mem_transfer(sel);
192
193       /* From a memory space to read */
194       pcie_addr = PCIE_AREA_ADDR + addr;
195       switch(size){
```

```
196                 case 1:
197                         bdata = PCIE_READB(pcie_addr);
198                         data = (unsigned long) bdata;
199                         break;
200                 case 2:
201                         wdata = PCIE_READW(pcie_addr);
202                         data = (unsigned long) wdata;
203                         break;
204                 case 4:
205                         data = PCIE_READL(pcie_addr);
206                         break;
207                 default:
208                         data = 0;
209                         break;;
210         }
211         return data;
212 }
213
214
215 /*""FUNC COMMENT""*****************************************************
216 * ID                    :
217 * Outline               : Sample Program Main
218 *                       : (PCI Express)
219 * Include               :
220 * Declaration           : void pcie_enable_io_transfer(int sel)
221 * Description           : An I/O transmission is validated
222 *                       :
223 *                       :
224 *                       :
225 *                       :
226 *                       :
227 * Limitation            :
228 *                       :
229 * Argument              : none
230 * Return Value          : none
231 * Calling Functions :
232 *""FUNC COMMENT END""*************************************************/
233 void pcie_enable_io_transfer(int sel)
234 {
235         /* A set of a register（A set of a window) */
```

```
236     PCIE_REG(sel, PAHR3) = 0x00000000;              /* 32bit of a upper address */
237     PCIE_REG(sel, PALR3) = PCIE_AREA_IO_ADDR;       /* 32 bits of a lower address */
238     PCIE_REG(sel, PAMR3) = 0x000C0000;              /* A window size is specified(1M) */
239     PCIE_REG(sel, PTCTLR3) = 0x80000100;            /* A transmitting packet property is specified */
240 }
241
242 /*""FUNC COMMENT""*****************************************************
243 * ID                    :
244 * Outline               : Sample Program Main
245 *                       : (PCI Express)
246 * Include               :
247 * Declaration           : long pcie_io_write(unsigned long addr, unsigned long data, unsigned long size)
248 * Description           : It writes in an I/O field
249 *                       :
250 *                       :
251 *                       :
252 *                       :
253 *                       :
254 * Limitation            :
255 *                       :
256 * Argument              : none
257 * Return Value          : -1:size error、0:Normal
258 * Calling Functions :
259 *""FUNC COMMENT END""*************************************************/
260 long pcie_io_write(int sel, unsigned long addr, unsigned long data, unsigned long size)
261 {
262     unsigned long pcie_addr;
263
264     /* Validation of an I/O transmission */
265     pcie_enable_io_transfer(sel);
266
267     /* The write to an I/O field */
268     pcie_addr = PCIE_AREA_IO_ADDR + addr;
269     switch(size){
270         case 1:
271             PCIE_WRITEB(pcie_addr, data);
272             break;
273         case 2:
274             PCIE_WRITEW(pcie_addr, data);
275             break;
```

```
276                 case 4:
277                         PCIE_WRITEL(pcie_addr, data);
278                         break;
279                 default:
280                         return -1;
281         }
282
283         return 0;
284 }
285
286 /*""FUNC COMMENT""*****************************************************
287 * ID                    :
288 * Outline               : Sample Program Main
289 *                       : (PCI Express)
290 * Include               :
291 * Declaration           : long pcie_io_read(int sel, unsigned long addr, unsigned long *data, unsigned long size)
292 * Description           : Read of an input-output field
293 *                       :
294 *                       :
295 *                       :
296 *                       :
297 *                       :
298 * Limitation            :
299 *                       :
300 * Argument              : none
301 * Return Value          : -1:size error、0:Normal
302 * Calling Functions :
303 *""FUNC COMMENT END""*************************************************/
304 unsigned long pcie_io_read(int sel, unsigned long addr, unsigned long size)
305 {
306         unsigned long pcie_addr;
307         unsigned short wdata;
308         unsigned char bdata;
309         unsigned long data;
310
311         /* Validation of an I/O transmission */
312         pcie_enable_io_transfer(sel);
313
314         /* From an I/O field to read */
315         pcie_addr = PCIE_AREA_IO_ADDR + addr;
```

```
316     switch(size){
317             case 1:
318                     bdata = PCIE_READB(pcie_addr);
319                     data = (unsigned long) bdata;
320                     break;
321             case 2:
322                     wdata = PCIE_READW(pcie_addr);
323                     data = (unsigned long) wdata;
324                     break;
325             case 4:
326                     data = PCIE_READL(pcie_addr);
327                     break;
328             default:
329                     data = 0;
330                     break;
331     }
332
333     return data;
334
335 }
336
337
338 /*""FUNC COMMENT""************************************************
339 * ID                    :
340 * Outline               : Sample Program Main
341 *                       : (PCI Express)
342 * Include               :
343 * Declaration           : static int phyreg_write(int sel, int addr, int lane, unsigned long data)
344 * Description           : The write to a physical-layer control register
345 *                       :
346 *                       :
347 *                       :
348 *                       :
349 *                       :
350 * Limitation            :
351 *                       :
352 * Argument              : none
353 * Return Value          : 0
354 * Calling Functions :
355 *""FUNC COMMENT END""*************************************************/
```

```
356 static int phyreg_write(int sel, int addr, int lane, unsigned long data)
357 {
358       unsigned long wdata;
359
360       /*The write to a physical-layer control register */
361       wdata = 0x00010000 + ((lane & 0xf) << 8) + (addr & 0xff);
362       PCIE_REG(sel, PHYCTLR) = 0x00000001; /* clock enable */
363       PCIE_REG(sel, PHYDOUTR) = data;                 /* A set of a writed data */
364       PCIE_REG(sel, PHYADRR) = wdata;                 /* A set of a command/address */
365       while( (PCIE_REG(sel, PHYADRR) & 0x01000000) == 0 );
366
367       /* Waiting for ACK */
368       PCIE_REG(sel, PHYADRR) = 0x00000000;            /* Command clear */
369       while( (PCIE_REG(sel, PHYADRR) & 0x01000000) != 0 );
370
371       PCIE_REG(sel, PHYCTLR) = 0x00000000; /* Clock disabling */
372
373       return 0;
374 }
375
376 /*""FUNC COMMENT""************************************************
377 * ID                    :
378 * Outline               : Sample Program Main
379 *                       : (PCI Express)
380 * Include               :
381 * Declaration           : static int phyreg_read(int sel, int addr, int lane, unsigned long *data)
382 * Description           : Read of a physical-layer control register
383 *                       :
384 *                       :
385 *                       :
386 *                       :
387 *                       :
388 * Limitation            :
389 *                       :
390 * Argument              : none
391 * Return Value          : 0
392 * Calling Functions :
393 *""FUNC COMMENT END""*********************************************/
394 static int phyreg_read(int sel, int addr, int lane, unsigned long *data)
395 {
```

```
396     unsigned long wdata;
397
398     /*Read of a physical-layer control register */
399     wdata = 0x00020000 + ((lane & 0xf) << 8) + (addr & 0xff);
400     PCIE_REG(sel, PHYCTLR) = 0x00000001; /* clock enable */
401     PCIE_REG(sel, PHYADRR) = wdata;              /* A set of a command/address */
402     while( (PCIE_REG(sel, PHYADRR) & 0x01000000) == 0 );
403
404     /* ACK 待ち  */
405     *data = PCIE_REG(sel, PHYDINR);              /* Read data */
406     PCIE_REG(sel, PHYADRR) = 0x00000000;         /* Command clear */
407     while( (PCIE_REG(sel, PHYADRR) & 0x01000000) != 0 );
408
409     PCIE_REG(sel, PHYCTLR) = 0x00000000; /* Clock disabling */
410
411     return 0;
412 }
413
414 /*""FUNC COMMENT""*****************************************************
415 * ID                    :
416 * Outline               : Sample Program Main
417 *                       : (PCI Express)
418 * Include               :
419 * Declaration           : static int config_read(int sel, int bus, int dev, int func, int regno, unsigned long *data)
420 * Description           : Read of a configuration register
421 *                       :
422 *                       :
423 *                       :
424 *                       :
425 *                       :
426 * Limitation            :
427 *                       :
428 * Argument              : none
429 * Return Value          : 0
430 * Calling Functions :
431 *""FUNC COMMENT END""*************************************************/
432 static int config_read(int sel, int bus, int dev, int func, int regno, unsigned long *data)
433 {
434     unsigned long wdata;
435
```

```
436     PCIE_REG(sel, PCTLR) = 0x80000000;      /* An issue Clearance of a configuration request */
437     wdata = (bus << 24) + (dev << 19) + (func << 16) + regno;
438     PCIE_REG(sel, PAR) = wdata;
439
440     /* Set of addr, bus/dev/func */
441     *data = PCIE_REG(sel, PDR);                     /* Issue of configuration read */
442     PCIE_REG(sel, PCTLR) = 0x00000000;      /* Issue of a configuration request is forbidden */
443
444     return 0;
445 }
446
447 /*""FUNC COMMENT""*****************************************************
448 * ID                    :
449 * Outline               : Sample Program Main
450 *                       : (PCI Express)
451 * Include               :
452 * Declaration           : static int config_write(int sel, int bus, int dev, int func, int regno, unsigned long data)
453 * Description           : Write of a configuration register
454 *                       :
455 *                       :
456 *                       :
457 *                       :
458 *                       :
459 * Limitation            :
460 *                       :
461 * Argument              : none
462 * Return Value          : 0
463 * Calling Functions :
464 *""FUNC COMMENT END""*************************************************/
465 static int config_write(int sel, int bus, int dev, int func, int regno, unsigned long data)
466 {
467     unsigned long wdata;
468
469     PCIE_REG(sel, PCTLR) = 0x80000000;      /* An issue Clearance of a configuration request */
470     wdata = (bus << 24) + (dev << 19) + (func << 16) + regno;
471     PCIE_REG(sel, PAR) = wdata;
472
473     /* Set of addr, bus/dev/func */
474     PCIE_REG(sel, PDR) = data;                      /* Issue of configuration read */
475     PCIE_REG(sel, PCTLR) = 0x00000000;      /* Issue of a configuration request is forbidden */
```

```
476
477         return 0;
478 }
479
480 /*""FUNC COMMENT""*****************************************************
481 * ID                    :
482 * Outline               : Sample Program Main
483 *                       : (PCI Express)
484 * Include               :
485 * Declaration           : static void pcie_soft_reset(int sel)
486 * Description           : Software reset of a PCIE controller
487 *                       :
488 *                       :
489 *                       :
490 *                       :
491 *                       :
492 * Limitation            :
493 *                       :
494 * Argument              : none
495 * Return Value          : none
496 * Calling Functions :
497 *""FUNC COMMENT END""*************************************************/
498 static void pcie_soft_reset(int sel)
499 {
500         /* A run of software reset */
501         PCIE_REG(sel, SRSTR) = 0x00000001;
502
503         /* A reset of a PCIE internal register */
504         PCIE_REG(sel, TCTLR) = 0x00000000;
505
506         /* A reset of software reset */
507         PCIE_REG(sel, SRSTR) = 0x00000000;
508
509         /* A clear of a VCO transmitter buffer */
510         PCIE_REG(sel, TXVC0SR) = 0x80000000;
511 }
512
513 /*""FUNC COMMENT""*****************************************************
514 * ID                    :
515 * Outline               : Sample Program Main
```

```
516 *                           : (PCI Express)
517 * Include                  :
518 * Declaration              : static int pcie_phy_init(int sel)
519 * Description              : The initialization of a PCIE controller transfer control register
520 *                           :
521 *                           :
522 *                           :
523 *                           :
524 *                           :
525 * Limitation               :
526 *                           :
527 * Argument                 : none
528 * Return Value             : -1:Time out, 0:Normal
529 * Calling Functions :
530 *""FUNC COMMENT END""**************************************************/
531 static int pcie_phy_init(int sel)
532 {
533         unsigned long stime;
534         static unsigned long data;
535
536         printf("PCI Express PHY During Initialization...");
537         /* A clock supply of a physical-layer register space accessing */
538         PCIE_REG(sel, PHYCTLR) = 0x00000001;
539
540         /* A physical layer's initialization */
541         phyreg_write(sel, 0x60, 0xf, 0x004B008B);
542         phyreg_write(sel, 0x61, 0xf, 0x00007B41);
543         phyreg_write(sel, 0x64, 0xf, 0x00FF4F00);
544         phyreg_write(sel, 0x65, 0xf, 0x09070907);
545         phyreg_write(sel, 0x66, 0xf, 0x00000010);
546         phyreg_write(sel, 0x74, 0xf, 0x0007001C);
547         phyreg_write(sel, 0x79, 0xf, 0x01FC000D);
548         phyreg_write(sel, 0xB0, 0xf, 0x00000610);
549
550         /* A boot of a physical layer */
551         phyreg_write(sel, 0x67, 0x1, 0x00000400);
552
553         if(sel)
554                 phyreg_read(sel, 0x67, 0x1, &data);
555
```

```
556     /*The clock of a physical-layer register space accessing is suspended */
557     PCIE_REG(sel, PHYCTLR) = 0x00000000;
558
559     /* Waiting for set a physical module */
560     stime = 1000;
561     while(stime--) {
562             /* It waits until a physical module will be in a ready state */
563             if( (PCIE_REG(sel, PHYSR) & 0x00000001) != 0 ) {
564                     break;
565             }
566             delay(1000);
567     }
568     if(!stime)
569             return -1;
570
571     printf("        Finish¥n¥r");
572
573     return 0;
574 }
575
576 /*""FUNC COMMENT""************************************************
577 * ID                    :
578 * Outline               : Sample Program Main
579 *                       : (PCI Express)
580 * Include               :
581 * Declaration           : static int pcie_trans_cont_init(int sel)
582 * Description           : The initialization of a PCIE controller transfer control register
583 *                       :
584 *                       :
585 *                       :
586 *                       :
587 *                       :
588 * Limitation            :
589 *                       :
590 * Argument              : none
591 * Return Value          : -1:Time out, 0:Normal
592 * Calling Functions :
593 *""FUNC COMMENT END""************************************************/
594 static int pcie_trans_cont_init(int sel)
595 {
```

```
596     unsigned long stime;
597
598     printf("PCI Express Controller During Initialization...");
599
600     /* A set of a bridge facility */
601     PCIE_REG(sel, LAR0) = 0x0C000000;       /* A local address 0 is specified */
602     PCIE_REG(sel, LAR2) = 0x0D000000;       /* A local address 1 is specified */
603
604     PCIE_REG(sel, LAMR0) = 0x000FFF01;      /* Local(SHwy)space register 0 */
605                                             /* 1MB */
606                                             /* A memory is secured in 32 bit address space */
607                                             /* Local address enabling is specified */
608     PCIE_REG(sel, LAMR2) = 0x000FFF11;      /* Local(SHwy)space register 1 */
609                                             /* 1MB */
610                                             /* A memory is secured in 32 bit address space */
611                                             /* Local address enabling is specified */
612
613     /* A set of a configuration register */
614     if(sel) {
615             PCIE_REG(sel, IDSETR1) = 0x01234567;
616             PCIE_REG(sel, IDSETR2) = 0x89ABCDEF;
617     }
618
619     PCIE_REG(sel, PCICONF1) = 0x00000007;
620
621     /* A wake-up of LTSSM(A settlement of a connection is started) */
622     PCIE_REG(sel, TCTLR) = 0x00000001;
623
624     /* A connection's settlement waiting */
625     stime = 10000;
626     while(stime--) {
627             /* Data Link Layer Active check */
628             if(PCIE_REG(sel, TSTR) != 0) {
629                     break;
630             }
631             delay(1000);
632     }
633     if(!stime)
634             return -1;
635     printf("        Finish¥n¥r");
```

```
636     return 0;
637 }
638
639 /*""FUNC COMMENT""*****************************************************
640 * ID                    :
641 * Outline               : Sample Program Main
642 *                       : (PCI Express)
643 * Include               :
644 * Declaration           : static int pcie_config_init(void)
645 * Description           : A set of the configuration of a PCIE controller
646 *                       :
647 *                       :
648 *                       :
649 *                       :
650 *                       :
651 * Limitation            :
652 *                       :
653 * Argument              : none
654 * Return Value          : -1:Inaccurate ID, 0:Normal
655 * Calling Functions :
656 *""FUNC COMMENT END""*************************************************/
657 static int pcie_config_init(int sel)
658 {
659     unsigned long config_data[(PCIE_MAX_CONFREG_SIZE/4)];
660     unsigned long regno;
661     unsigned long data;
662     unsigned long dev_id;
663     unsigned long dev_cap;
664     unsigned long dev_ctrl;
665     unsigned long dev_bar;
666     unsigned long dev_type;
667     unsigned char cap_ptr, next_ptr, cap_id, mpss1, mpss2, mps;
668     int stime = 1000;
669
670     /* A confirm of a vender and product ID */
671     while(stime--) {
672         config_read(sel, 0, 1, 0, PCIE_CONF_DEVICE_ID, &data);
673         if(!(PCIE_REG(sel, PCTLR) & 0x00010000))
674             break;
675         delay(1000);
```

```
676     }
677     if(!stime)
678             return -1;
679
680     dev_id = data;
681     if(dev_id == 0x00000000) {
682             return -1;
683     }
684
685     /* Max Payload Size Supported of a splicing place device is acquired. */
686     config_read(sel, 0, 1, 0, PCIE_CONF_CAP_PTR, &data);
687     next_ptr = data & 0xff;
688     cap_id = 0xff;
689     while(1) {
690             if (cap_id == 0x10) {
691                     /* PCI Express Capability Structure */
692                     config_read(sel, 0, 1, 0, (unsigned long)(cap_ptr+0x04), &dev_cap);
693                     mpss1 = dev_cap & 0x07;
694                     break;
695             } else {
696                     /* other capability list */
697                     cap_ptr = next_ptr;
698                     config_read(sel, 0, 1, 0, (unsigned long)cap_ptr, &data);
699                     cap_id = data & 0xff;
700                     next_ptr = (data >> 8) & 0xff;
701             }
702     }
703
704     /* MPSS of a self-device */
705     mpss2 = PCIE_REG(sel, EXPCAP1) & 0x07;
706     mps = (mpss1 < mpss2) ? mpss1 : mpss2; /* Both smallest value is taken. */
707
708     /* Set of MPS  : Splicing place device */
709     config_read(sel, 0,1,0,(unsigned long)(cap_ptr + 0x08), &dev_ctrl);
710     dev_ctrl = (dev_ctrl & 0xFFFFFF1F) | (mps << 5); /* bit7-5 is substituted for MPS */
711     config_write(sel, 0,1,0,(unsigned long)(cap_ptr + 0x08), dev_ctrl);
712
713     /* Set of MPS : self-device */
714     dev_ctrl = PCIE_REG(sel, EXPCAP2);
715     dev_ctrl = (dev_ctrl & 0xFFFFFF1F) | (mps << 5); /* bit7-5 is substituted for MPS */
```

```
716     PCIE_REG(sel, EXPCAP1) = dev_ctrl;
717
718     /* A set of a command and a register */
719     config_write(sel, 0, 1, 0, PCIE_CONF_COMMAND, 0x00000007);      /* External device */
720
721     /* Set of BAR : Splicing place device */
722     config_read(sel, 0, 1, 0, PCIE_CONF_BASE_ADDRESS_0, &dev_bar);
723     dev_type = dev_bar & 0x06;
724     if(dev_type == 0x00) {
725             /* 32 bit space */
726             config_write(sel, 0, 1, 0, PCIE_CONF_BASE_ADDRESS_0, PCIE_AREA_ADDR);
727             config_write(sel, 0, 1, 0, PCIE_CONF_BASE_ADDRESS_2, PCIE_AREA_IO_ADDR);
728     } else if(dev_type == 0x04) {
729             /* 64 bit space */
730             config_write(sel, 0, 1, 0, PCIE_CONF_BASE_ADDRESS_0, PCIE_AREA_ADDR);
731             config_write(sel, 0, 1, 0, PCIE_CONF_BASE_ADDRESS_1, 0x00000000);
732     }
733
734     /* An obtaining of all the configuration registers of an external device */
735     for (regno = 0; regno < (PCIE_MAX_CONFREG_SIZE/4); regno++) {
736             config_read(sel, 0, 1, 0, regno * 4, &data);
737             config_data[regno] = data;
738     }
739     memcpy( &conf_data, config_data, sizeof( PCIE_CONF_DATA ) );
740
741     return 0;
742 }
743
744 /*""FUNC COMMENT""*****************************************************
745 * ID                    :
746 * Outline               : Sample Program Main
747 *                       : (PCI Express)
748 * Include               :
749 * Declaration           : static unsigned long pcie_link_lane(void)
750 * Description           : An obtaining of an effective lane
751 *                       :
752 *                       :
753 *                       :
754 *                       :
755 *                       :
```

```
756 * Limitation             :
757 *                         :
758 * Argument                : none
759 * Return Value            : An obtaining of a link data
760 * Calling Functions :
761 *""FUNC COMMENT END""**************************************************/
762 static unsigned long pcie_link_lane(int sel)
763 {
764         unsigned long data;
765
766         /* The read of a link status */
767         data = PCIE_REG(sel, EXPCAP4);
768         data >>= 20;
769         data &= 0x3f;
770
771         return data;
772 }
773
774 /*""FUNC COMMENT""*******************************************************
775 * ID                      :
776 * Outline                 : Sample Program Main
777 *                         : (PCI Express)
778 * Include                 :
779 * Declaration             : void pcie_init(int sel)
780 * Description             : The initialization of a PCIE controller
781 *                         :
782 *                         :
783 *                         :
784 *                         :
785 *                         :
786 * Limitation              :
787 *                         :
788 * Argument                : none
789 * Return Value            : none
790 * Calling Functions :
791 *""FUNC COMMENT END""**************************************************/
792 void pcie_init(int sel)
793 {
794         /* The initialization of a PCIE configuration data */
795         memset( &conf_data, 0xFF, sizeof( PCIE_CONF_DATA ) );
```

```
796
797       /* Software reset */
798       pcie_soft_reset(sel);
799
800       /* A physical layer's initialization */
801       if( pcie_phy_init(sel) < 0 ) {
802                 return;
803       }
804
805       /* The initialization of PCIE（A connection start）  */
806       if( pcie_trans_cont_init(sel) < 0 ) {
807                 return;
808       }
809
810       /* A set of a configuration */
811       if( pcie_config_init(sel) < 0 ) {
812                 return;
813       }
814
815 }
816
817 /*""FUNC COMMENT""*****************************************************
818 * ID                        :
819 * Outline                   : Sample Program Main
820 *                           : (PCI Express)
821 * Include                   :
822 * Declaration               : void pcie_check(int sel)
823 * Description               : The check of a PCIE controller device
824 *                           :
825 *                           :
826 *                           :
827 *                           :
828 *                           :
829 * Limitation                :
830 *                           :
831 * Argument                  : none
832 * Return Value              : none
833 * Calling Functions :
834 *""FUNC COMMENT END""*************************************************/
835 void pcie_check(int sel)
```

```
836 {
837     unsigned long lane;
838
839     /* A view of a configuration register */
840     if( (conf_data.VenderID != 0xFFFF) && (conf_data.DeviceID != 0xFFFF) ) {
841             scif_printf(" Enable lane : %d LANE¥n¥r", pcie_link_lane(sel));
842             scif_printf("¥n¥r");
843
844             scif_printf(" Vender ID      : %04x¥n¥r", conf_data.VenderID);
845             scif_printf(" Device ID      : %04x¥n¥r", conf_data.DeviceID);
846             scif_printf(" Command          : %04x¥n¥r", conf_data.Command );
847             scif_printf(" Status           : %04x¥n¥r", conf_data.Status    );
848             scif_printf(" Revision ID : %02x¥n¥r", conf_data.RevisionID);
849             scif_printf(" ProgrammingInterface : %02x¥n¥r", conf_data.ProgrammingInterface);
850             scif_printf(" SubClass       : %02x¥n¥r", conf_data.SubClass);
851             scif_printf(" BaseClass      : %02x¥n¥r", conf_data.BaseClass);
852             scif_printf(" CachLineSize: %02x¥n¥r", conf_data.CachLineSize);
853             scif_printf(" LatencyTimer: %02x¥n¥r", conf_data.LatencyTimer);
854             scif_printf(" HeaderType    : %02x¥n¥r", conf_data.HeaderType);
855             scif_printf(" BIST             : %02x¥n¥r", conf_data.BIST);
856             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[0]);
857             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[1]);
858             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[2]);
859             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[3]);
860             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[4]);
861             scif_printf(" BaseAdrsREG : %08x¥n¥r", conf_data.BaseAddressRegisters[5]);
862             scif_printf(" CardbusCISpointer : %08x¥n¥r", conf_data.CardbusCISpointer);
863             scif_printf(" SubsystemVenderID : %04x¥n¥r", conf_data.SubsystemVenderID);
864             scif_printf(" SubsystemID : %04x¥n¥r", conf_data.SubsystemID);
865             scif_printf(" ExpantionROMbaseAddress : %08x¥n¥r", conf_data.ExpantionROMbaseAddress);
866             scif_printf(" CAP_PTR        : %02x¥n¥r", conf_data.CAP_PTR);
867             scif_printf(" Int Line       : %02x¥n¥r", conf_data.InttreuptLine);
868             scif_printf(" Int Pin        : %02x¥n¥r", conf_data.InttreuptPin);
869             scif_printf(" Min_Gnt        : %02x¥n¥r", conf_data.Min_Gnt);
870             scif_printf(" Max_Lat        : %02x¥n¥r", conf_data.Max_Lat);
871     } else {
872             /* no device */
873             scif_printf("Device not detected on PCI Bus!¥r¥n");
874     }
875 }
```

```
876
877 /*""FUNC COMMENT""************************************************
878 * ID                    :
879 * Outline               : Sample Program Main
880 *                       : (PCI Express)
881 * Include               :
882 * Declaration           : int pcie_start_dma(int sel, int pciadd, int shadd, int dir, int cnt)
883 * Description           : PCIEC-DMAC Setting and start
884 *                       :
885 *                       :
886 *                       :
887 *                       :
888 *                       :
889 * Limitation            :
890 *                       :
891 * Argument              : none
892 * Return Value          : none
893 * Calling Functions :
894 *""FUNC COMMENT END""************************************************/
895 int pcie_start_dma(int sel, int pciadd, int shadd, int dir, int cnt)
896 {
897       int stime, status;
898       /* DMA Nomal Transfer */
899       PCIE_REG(sel, DMAOR) = DMAE;                          /* DMAC Enable */
900
901       PCIE_REG(sel, DMSBCNTR0) = 0x00000000;          /* No Stride Transfer */
902       PCIE_REG(sel, DMSTRR0) = 0x00000000;
903       PCIE_REG(sel, DMCCAR0) = 0x00000000;          /* No DMAC Command chain Transfer */
904
905       PCIE_REG(sel, DMPALR0) = pciadd;                /* 32bit of a lower address for PCI */
906       PCIE_REG(sel, DMPAHR0) = 0x00000000;          /* 32bit of a upper address for PCI */
907
908       PCIE_REG(sel, DMSALR0) = shadd;                       /* 32bit of a address for SuperHyway
*/
909
910       PCIE_REG(sel, DMBCNTR0) = cnt;            /* Transfer Count */
911
912       /* DMA Transfer Raedy settlement waiting */
913       stime = 10000;
914       while(stime--) {
915             /* Channel Status PE & SE & TE = 0 */
```

```
916                 status = PCIE_REG(sel, DMCHSR0);
917                 if(((status & PE) != PE) |
918                     ((status & SE) != SE) |
919                     ((status & TE) != TE)) {
920                         break;
921                 }
922                 delay(1000);
923         }
924         if(!stime)
925                 return -1;
926
927         PCIE_REG(sel, DMCHCR0) = CHE | DIR(dir);        /* Transfer Start / Direction */
928
929         while((PCIE_REG(sel, DMCHSR0) & TE) != TE); /* Transfer End Wait */
930         PCIE_REG(sel, DMCHCR0) &= !CHE;                                 /* Transfer Disable */
931         PCIE_REG(sel, DMCHSR0) = TE;
932
933 }
```

(4)　”pcie.h”

サンプルプログラムのメイン関数、PCIEC 初期化関数、PCI Express 制御系関数、DMA 制御関数で使用するヘッダファイルです。

```
001 #ifndef        _PCIE_H_
002 #define        _PCIE_H_
003
004 #include "config.h"
005
006 #define        PCIE_BASE       0xFE000000
007 #define        PCIE_REG(p, x) (*(volatile unsigned long    *)(PCIE_BASE | (p << 21) | x))
008
009 #define ENBLR            0x00008
010 #define ECR              0x0000C
011 #define PAR              0x00010
012 #define PCTLR            0x00018
013 #define PDR              0x00020
014 #define MSGALR           0x00030
015 #define MSGAHR           0x00034
016 #define MSGCTLR          0x00038
017 #define UNLOCKCR         0x00048
018 #define IDR              0x00060
019 #define DBGCTLR          0x00100
020 #define INTXR            0x04000
021 #define RMSGR            0x04010
022 #define RMSGIER          0x04040
023 #define RSTR0            0x08000
024 #define RSTR1            0x08004
025 #define RSTR2            0x08008
026 #define RSTR3            0x0800C
027 #define SRSTR            0x08040
028 #define PHYCTLR          0x10000
029 #define PHYADRR          0x10004
030 #define PHYDINR          0x10008
031 #define PHYDOUTR         0x1000C
032 #define PHYSR            0x10010
033 #define TCTLR            0x20000
034 #define TSTR             0x20004
035 #define INTR             0x20008
```

```
036 #define INTER       0x2000C
037 #define EH0R        0x20010
038 #define EH1R        0x20014
039 #define EH2R        0x20018
040 #define EH3R        0x2001C
041 #define ERRFR       0x20020
042 #define ERRFER      0x20024
043 #define ERRFR2      0x20028
044 #define MSIR        0x20040
045 #define MSIFR       0x20044
046 #define PWRCTLR     0x20100
047 #define PCCTLR      0x20180
048 #define LAR0        0x20200
049 #define LAMR0       0x20208
050 #define LAR1        0x20220
051 #define LAMR1       0x20228
052 #define LAR2        0x20240
053 #define LAMR2       0x20248
054 #define LAR3        0x20260
055 #define LAMR3       0x20268
056 #define LAR4        0x20280
057 #define LAMR4       0x20288
058 #define LAR5        0x202A0
059 #define LAMR5       0x202A8
060 #define PALR0       0x20400
061 #define PAHR0       0x20404
062 #define PAMR0       0x20408
063 #define PTCTLR0     0x2040C
064 #define PALR1       0x20420
065 #define PAHR1       0x20424
066 #define PAMR1       0x20428
067 #define PTCTLR1     0x2042C
068 #define PALR2       0x20440
069 #define PAHR2       0x20444
070 #define PAMR2       0x20448
071 #define PTCTLR2     0x2044C
072 #define PALR3       0x20460
073 #define PAHR3       0x20464
074 #define PAMR3       0x20468
075 #define PTCTLR3     0x2046C
```

```
076 #define DMAOR           0x21000
077 #define DMPALR0         0x21100
078 #define DMPAHR0         0x21104
079 #define DMSALR0         0x21108
080 #define DMBCNTR0        0x21110
081 #define DMSBCNTR0       0x21114
082 #define DMSTRR0         0x21118
083 #define DMCCAR0         0x21120
084 #define DMCHCR0         0x21128
085 #define DMCHSR0         0x2112C
086 #define DMPALR1         0x21140
087 #define DMPAHR1         0x21144
088 #define DMSALR1         0x21148
089 #define DMBCNTR1        0x21150
090 #define DMSBCNTR1       0x21154
091 #define DMSTRR1         0x21158
092 #define DMCCAR1         0x21160
093 #define DMCHCR1         0x21168
094 #define DMCHSR1         0x2116C
095 #define DMPALR2         0x21180
096 #define DMPAHR2         0x21184
097 #define DMSALR2         0x21188
098 #define DMBCNTR2        0x21190
099 #define DMSBCNTR2       0x21194
100 #define DMSTRR2         0x21198
101 #define DMCCAR2         0x211A0
102 #define DMCHCR2         0x211A8
103 #define DMCHSR2         0x211AC
104 #define DMPALR3         0x211C0
105 #define DMPAHR3         0x211C4
106 #define DMSALR3         0x211C8
107 #define DMBCNTR3        0x211D0
108 #define DMSBCNTR3       0x211D4
109 #define DMSTRR3         0x211D8
110 #define DMCCAR3         0x211E0
111 #define DMCHCR3         0x211E8
112 #define DMCHSR3         0x211EC
113 #define PCICONF0        0x40000
114 #define PCICONF1        0x40004
115 #define PCICONF2        0x40008
```

```
116 #define PCICONF3        0x4000C
117 #define PCICONF4        0x40010
118 #define PCICONF5        0x40014
119 #define PCICONF6        0x40018
120 #define PCICONF7        0x4001C
121 #define PCICONF8        0x40020
122 #define PCICONF9        0x40024
123 #define PCICONF10       0x40028
124 #define PCICONF11       0x4002C
125 #define PCICONF12       0x40030
126 #define PCICONF13       0x40034
127 #define PCICONF14       0x40038
128 #define PCICONF15       0x4003C
129 #define PMCAP0          0x40040
130 #define PMCAP1          0x40044
131 #define MSICAP0         0x40050
132 #define MSICAP1         0x40054
133 #define MSICAP2         0x40058
134 #define MSICAP3         0x4005C
135 #define MSICAP4         0x40060
136 #define MSICAP5         0x40064
137 #define EXPCAP0         0x40070
138 #define EXPCAP1         0x40074
139 #define EXPCAP2         0x40078
140 #define EXPCAP3         0x4007C
141 #define EXPCAP4         0x40080
142 #define EXPCAP5         0x40084
143 #define EXPCAP6         0x40088
144 #define EXPCAP7         0x4008C
145 #define EXPCAP8         0x40090
146 #define VCCAP0          0x40100
147 #define VCCAP1          0x40104
148 #define VCCAP2          0x40108
149 #define VCCAP3          0x4010C
150 #define VCCAP4          0x40110
151 #define VCCAP5          0x40114
152 #define VCCAP6          0x40118
153 #define VCCAP7          0x4011C
154 #define VCCAP8          0x40120
155 #define VCCAP9          0x40124
```

```
156 #define NUMCAP0         0x401B0
157 #define NUMCAP1         0x401B4
158 #define NUMCAP2         0x401B8
159 #define IDSETR1         0x41004
160 #define IDSETR2         0x41024
161 #define DSERSETR0       0x4102C
162 #define DSERSETR1       0x41030
163 #define TLSR            0x41044
164 #define TLCTLR          0x41048
165 #define DLSR            0x4104C
166 #define DLCTLR          0x41050
167 #define MACSR           0x41054
168 #define MACCTLR         0x41058
169 #define PMSR            0x4105C
170 #define PMCTLR          0x41060
171 #define TLINTENR        0x41064
172 #define DLINTENR        0x41068
173 #define MACINTENR       0x4106C
174 #define PMINTENR        0x41070
175 #define TXSR            0x44028
176 #define TXVC0SR         0x44108
177
178
179 /*
180    * PCIE DMAC
181    */
182 /* DMAOR */
183 #define         DMAE            (1 << 31)
184
185 /* CHCR */
186 #define         CHE             (1 << 31)
187 #define         DIR(x)          (x << 30)
188
189 #define PCIE_WRITE          1
190 #define PCIE_READ           0
191
192 /* CHSR */
193 #define         PE              (1 << 11)
194 #define         SE              (1 << 9)
195 #define         TE              (1 << 0)
```

```
196
197
198 /*
199    * PCI area
200    */
201 #ifdef CONFIG_PCIE_ROOT
202 #define PCIE_AREA_ADDR              0xFD000000      /* PCI area 0 address */
203 #define PCIE_AREA_IO_ADDR           0xFE100000      /* PCI area 3 address */
204 #else
205 #define PCIE_AREA_ADDR              0xFD800000      /* PCI area 0 address */
206 #define PCIE_AREA_IO_ADDR           0xFE300000      /* PCI area 3 address */
207 #endif
208
209 /*
210    * SDRAM DATA area
211    */
212 #define sdram_data_area                 0x0C000000      /* SDRAM DATA AREA */
213
214 /*
215    *  PCIE data macroinstruction
216    */
217 #define         PCIE_MAX_CONFREG_SIZE     0x1000              /* Configuration register size 256 byte */
218
219 /*
220    * PCIE Configuration register
221    */
222 #define PCIE_CONF_DEVICE_ID             0x00
223 #define PCIE_CONF_VENDER_ID             0x00
224 #define PCIE_CONF_STATUS                0x04
225 #define PCIE_CONF_COMMAND               0x04
226 #define PCIE_CONF_CLASS_CODE            0x08
227 #define PCIE_CONF_REVISION_ID           0x08
228 #define PCIE_CONF_BIST                  0x0C
229 #define PCIE_CONF_HEADER_TYPE           0x0C
230 #define PCIE_CONF_LATENCY_TIMER         0x0C
231 #define PCIE_CONF_CACHE_LINE_SIZE       0x0C
232 #define PCIE_CONF_BASE_ADDRESS_0        0x10
233 #define PCIE_CONF_BASE_ADDRESS_1        0x14
234 #define PCIE_CONF_BASE_ADDRESS_2        0x18
235 #define PCIE_CONF_BASE_ADDRESS_3        0x1C
```

```
236 #define PCIE_CONF_BASE_ADDRESS_4        0x20
237 #define PCIE_CONF_BASE_ADDRESS_5        0x24
238 #define PCIE_CONF_SUB_SYSTEM_ID         0x2C
239 #define PCIE_CONF_SUB_VENDER_ID         0x2C
240 #define PCIE_CONF_EXP_ROM_BASE          0x30
241 #define PCIE_CONF_CAP_PTR               0x34
242 #define PCIE_CONF_MAX_LAT               0x3C
243 #define PCIE_CONF_MIN_GNT               0x3C
244 #define PCIE_CONF_INTERRUPT_PIN         0x3C
245 #define PCIE_CONF_INTERRUPT_LINE        0x3C
246
247 /*    Register accessing            */
248 #define       PCIE_WRITEL(addr,data)        (*((volatile unsigned long    *)(addr)) = (unsigned long)(data))
249 #define       PCIE_READL(addr)              (*((volatile unsigned long    *)(addr)))
250 #define       PCIE_WRITEW(addr,data)        (*((volatile unsigned short *)(addr)) = (unsigned short)(data))
251 #define       PCIE_READW(addr)              (*((volatile unsigned short *)(addr)))
252 #define       PCIE_WRITEB(addr,data)        (*((volatile unsigned char    *)(addr)) = (unsigned char)(data))
253 #define       PCIE_READB(addr)              (*((volatile unsigned char    *)(addr)))
254
255
256 /*
257    *  A configuration space register's structure
258    */
259
260 /* Header Type 0 */
261 typedef struct _PCIE_CONF_DATA {
262 #if defined(_BIG)
263       unsigned short    DeviceID;                   /* Device ID */
264       unsigned short    VenderID;                   /* Vender ID */
265
266       unsigned short    Status;                     /* Device Status */
267       unsigned short    Command;                    /* Device Control */
268
269       unsigned char     BaseClass;                  /* Base Class */
270       unsigned char     SubClass;                   /* Sub Class */
271       unsigned char     ProgrammingInterface;       /* Programming Interface */
272       unsigned char     RevisionID;                 /* Revision ID */
273
274       unsigned char     BIST;                       /* BIST */
275       unsigned char     HeaderType;                 /* Header Type */
```

```
276     unsigned char   LatencyTimer;              /* Master latency timer */
277     unsigned char   CachLineSize;              /* Cach Line Size */
278
279     unsigned long   BaseAddressRegisters[6];   /* Base Address Registers space */
280
281     unsigned long   CardbusCISpointer;         /* Card Bus CIS pointer */
282
283     unsigned short  SubsystemID;               /* Subsystem ID */
284     unsigned short  SubsystemVenderID;         /* Subsystem Vender ID */
285
286     unsigned long   ExpantionROMbaseAddress;   /* Expantion ROM base Address */
287
288     unsigned char   reserve0[3];               /* Reserve */
289     unsigned char   CAP_PTR;                   /* New facility pointer */
290
291     unsigned long   reserve1;                  /* Reserve */
292
293     unsigned char   Max_Lat;                   /* MAX latency */
294     unsigned char   Min_Gnt;                   /* MIN Grant */
295     unsigned char   InttreuptPin;              /* PCI Inttreupt pin */
296     unsigned char   InttreuptLine;             /* Interrupt line */
297 #else
298     unsigned short  VenderID;                  /* Vender ID */
299     unsigned short  DeviceID;                  /* Device ID */
300
301     unsigned short  Command;                   /* Device Control */
302     unsigned short  Status;                    /* Device Status */
303
304     unsigned char   RevisionID;                /* Revision ID */
305     unsigned char   ProgrammingInterface;      /* Programming Interface */
306     unsigned char   SubClass;                  /* Sub Class */
307     unsigned char   BaseClass;                 /* Base Class */
308
309     unsigned char   CachLineSize;              /* Cach Line Size */
310     unsigned char   LatencyTimer;              /* Master latency timer */
311     unsigned char   HeaderType;                /* Header Type */
312     unsigned char   BIST;                      /* BIST */
313
314     unsigned long   BaseAddressRegisters[6];   /* Base Address Registers space */
315
```

```
316     unsigned long   CardbusCISpointer;          /* Card Bus CIS pointer */
317
318     unsigned short  SubsystemVenderID;          /* Subsystem Vender ID */
319     unsigned short  SubsystemID;                /* Subsystem ID */
320
321     unsigned long   ExpantionROMbaseAddress;    /* Expantion ROM base Address */
322
323     unsigned char   CAP_PTR;                    /* New facility pointer */
324     unsigned char   reserve0[3];                /* Reserve */
325
326     unsigned long   reserve1;                   /* Reserve */
327
328     unsigned char   InttreuptLine;              /* Interrupt line */
329     unsigned char   InttreuptPin;               /* PCI Inttreupt pin */
330     unsigned char   Min_Gnt;                    /* MIN Grant */
331     unsigned char   Max_Lat;                    /* MAX latency */
332
333 #endif
334 } PCIE_CONF_DATA;
335
336
337 #endif /* _PCIE_H_ */
```

(5)　”scif.c”

SCIF0 初期設定、シリアルドライバ関数のサンプルドライバです。

```
001 /***********************************************************************
002 * DISCLAIMER
003
004 * This software is supplied by Renesas Electronics    Corporation. and is only
005 * intended for use with Renesas products. No other uses are authorized.
006
007 * This software is owned by Renesas Electronics    Corporation. and is protected under
008 * all applicable laws, including copyright laws.
009
010 * THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
011 * REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
012 * INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
013 * PARTICULAR PURPOSE AND NON-INFRINGEMENT.    ALL SUCH WARRANTIES ARE EXPRESSLY
014 * DISCLAIMED.
015
016 * TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
017 * ELECTRONICS CORPORATION. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
018 * FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
019 * FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
020 * AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
021
022 * Renesas reserves the right, without notice, to make changes to this
023 * software and to discontinue the availability of this software.
024 * By using this software, you agree to the additional terms and
025 * conditions found by accessing the following link:
026 * http://www.renesas.com/disclaimer
027 ***********************************************************************/
028 /* Copyright (C) 2010. Renesas Electronics Corporation., All Rights Reserved.*/
029 /*""FILE COMMENT""************ Technical reference data ****************
030 * System Name    : SH7786 Sample Program
031 * File Name       : scif.c
032 * Abstract        : The example of a set of SCIF Sample Program
033 * Version         : Ver 1.00
034 * Device          : SH7786
035 * Tool-Chain      : High-performance Embedded Workshop (Version 4.07.00.007)
036 *                   : C/C++ Compiler Package for SuperH Family (V.9.3.2.0)
```

```
037 * OS                  : None
038 * H/W Platform : SH-4A Board P/N:AP-SH4AD-3A (Manufacturer:ALPHA PROJECT)
039 * Description    : It is an example program of the example of a SH7786 SCIF set.
040 *                       :
041 * Operation       :
042 * Limitation      :
043 *                       :
044 ***********************************************************************
045 * History          : 01.Sep.2010 Ver. 1.00 First Release
046 *""FILE COMMENT END""*****************************************************/
047
048
049 #include       "scif.h"
050
051 /*""FUNC COMMENT""*******************************************************
052 * ID                          :
053 * Outline                  : Sample Program Main
054 *                             :
055 * Include                  :
056 * Declaration            : int delay( int cnt )
057 * Description            : Software weight
058 *                             : A part for the count of "cnt" and a "for" are repeated.
059 *                             :
060 *                             :
061 *                             :
062 *                             :
063 * Limitation             :
064 *                             :
065 * Argument                : cnt
066 * Return Value          : none
067 * Calling Functions :
068 *""FUNC COMMENT END""*****************************************************/
069 void delay( int cnt )
070 {
071       int i;
072       for(i=0;i<cnt;i++);
073 }
074
075 /*""FUNC COMMENT""*******************************************************
076 * ID                          :
```

```
077 * Outline                : Sample Program Main
078 *                          :
079 * Include                :
080 * Declaration            : int scif_init(void)
081 * Description            : The initialization of SCIF
082 *                          :
083 *                          :
084 *                          :
085 *                          :
086 *                          :
087 * Limitation             :
088 *                          :
089 * Argument                 : none
090 * Return Value           : -1 : Baud rate clock count error
091 * Calling Functions :
092 *""FUNC COMMENT END""**************************************************/
093 int scif_init(void)
094 {
095         unsigned short data;
096         int t = -1, cnt = 0;
097
098         SCIF.SCSCR.WORD = 0x0000;     /* TIE, RIE, TE, RE Clear */
099
100         SCIF.SCFCR.BIT.TFCL = 1;          /* Tx FIFO Clear */
101         SCIF.SCFCR.BIT.RFCL = 1;          /* Rx FIFO Clear */
102
103         SCIF.SCFSR.WORD = 0x0000;     /* BRK, DR, TR Clear */
104         SCIF.SCLSR.BIT.ORER = 0;          /* ORER Clear */
105
106 #if defined(CONFIG_SCIF_CLK_EXTERNAL)
107         SCIF.SCSCR.BIT.CKE = 2;           /* Clock source : SCK */
108 #elif    defined(CONFIG_SCIF_CLK_PCLK)
109         SCIF.SCSCR.BIT.CKE = 0;           /*    Clock source : PCLK */
110         t = SCBRR_VALUE(CONFIG_BPS, CONFIG_SCIF_CLK_PCLK);
111 #endif /* CONFIG_SCIF_CLK */
112
113         if(t > 0) {
114                 while(t >= 256) {
115                         cnt++;
116                         t >> 2;
```

```
117                 }
118                 if(cnt > 3)
119                         return -1;
120
121                 SCIF.SCSMR.BIT.CKS = cnt;
122                 SCIF.SCBRR = t;
123         }
124         delay(1000);
125
126         SCIF.SCFCR.BIT.RTRG = 0;
127         SCIF.SCFCR.BIT.TTRG = 0;
128         SCIF.SCFCR.BIT.TFCL = 1;        /* Tx FIFO Clear */
129         SCIF.SCFCR.BIT.RFCL = 1;        /* Rx FIFO Clear */
130
131         SCIF.SCFCR.BIT.TFCL = 0;        /* Tx FIFO Not Clear */
132         SCIF.SCFCR.BIT.RFCL = 0;        /* Rx FIFO Not Clear */
133         SCIF.SCSCR.BIT.TE   = 1;
134         SCIF.SCSCR.BIT.RE   = 1;
135         return 0;
136 }
137
138 /*""FUNC COMMENT""*****************************************************
139 * ID                    :
140 * Outline               : Sample Program Main
141 *                       :
142 * Include               :
143 * Declaration           : void scif_transmit_data( char *Data )
144 * Description           : A transmission of two or more byte data of SCIF.
145 *                       :
146 *                       :
147 *                       :
148 *                       :
149 *                       :
150 * Limitation            :
151 *                       :
152 * Argument              : *Data：A send data is stored.
153 * Return Value          : none
154 * Calling Functions :
155 *""FUNC COMMENT END""*************************************************/
156 voidscif_transmit_data( char    *Data )
```

```
157 {
158         while( *Data )
159         {
160                     while(!(SCIF.SCFSR.BIT.TDFE)); /* Weight is carried out until the write of a send data will be in an
authorized state.  */
161                     SCIF.SCFTDR = *Data;                                        /* A set of a send data       */
162                     Data++;
163                     while(!(SCIF.SCFSR.BIT.TEND));/* Waiting for the quit of transmitting          */
164                     SCIF.SCFSR.BIT.TDFE = 0;
165                     SCIF.SCFSR.BIT.TEND = 0;
166         }
167 }
168
169 /*""FUNC COMMENT""*****************************************************
170 * ID                        :
171 * Outline                   : Sample Program Main
172 *                           : (PCIe)
173 * Include                   :
174 * Declaration               : void scif_transmit_byte_data( char *Data )
175 * Description               : A transmission of the single byte data of SCIF
176 *                           :
177 *                           :
178 *                           :
179 *                           :
180 *                           :
181 * Limitation                :
182 *                           :
183 * Argument                  : *Data：A send data is stored.
184 * Return Value              : none
185 * Calling Functions :
186 *""FUNC COMMENT END""*************************************************/
187 voidscif_transmit_data_byte( char         *Data )
188 {
189                     while(!(SCIF.SCFSR.BIT.TDFE)); /* Weight is carried out until the write of a send data will be in an
authorized state.  */
190                     SCIF.SCFTDR = *Data;                                        /* A set of a send data       */
191                     while(!(SCIF.SCFSR.BIT.TEND));/* Waiting for the quit of transmitting          */
192                     SCIF.SCFSR.BIT.TDFE = 0;
193                     SCIF.SCFSR.BIT.TEND = 0;
194 }
195
```

```
196
197 /*""FUNC COMMENT""************************************************
198 * ID                    :
199 * Outline               : Sample Program Main
200 *                       : (PCIe)
201 * Include               :
202 * Declaration           : void sci_printf(char* str, ...)
203 * Description           : A text with a format is outputted.
204 *                       :
205 *                       :
206 *                       :
207 *                       :
208 *                       :
209 * Limitation            :
210 *                       :
211 * Argument              : *Data：A send data is stored.
212 * Return Value          : none
213 * Calling Functions :
214 *""FUNC COMMENT END""*************************************************/
215 /***********************************/
216 /*    A text with a format is outputted                              */
217 /***********************************/
218 #define PRINTF_SIZE           1024
219 static char printf_str[PRINTF_SIZE];
220
221 void scif_printf(char* str, ...)
222 {
223       va_list args;
224       size_t size;
225
226       size = strlen(str);
227
228       if( size > PRINTF_SIZE ) {
229                 return;
230       }
231
232       va_start(args, str);
233       vsprintf(printf_str, str, args);
234       va_end(args);
235
```

```
236       scif_transmit_data(printf_str);
237 }
238
239 /*""FUNC COMMENT""*****************************************************
240 * ID                    :
241 * Outline               : Sample Program Main
242 *                       :
243 * Include               :
244 * Declaration           : char scif_recive_data( char          *Data )
245 * Description           : The data of SCIF is received.
246 *                       :
247 *                       :
248 *                       :
249 *                       :
250 *                       :
251 * Limitation            :
252 *                       :
253 * Argument              : *Data：A receive data is stored.
254 * Return Value          : -1：A receive data error
255 * Calling Functions :
256 *""FUNC COMMENT END""*************************************************/
257 char scif_recive_data( char       *Data )
258 {
259       unsigned char     ReadData, i = 0;
260       char      ret_cd = 0;
261
262       for(;;)
263       {
264                 if(( SCIF.SCFSR.BIT.ER    ) ||
265                     ( SCIF.SCFSR.BIT.BRK ) ||
266                     ( SCIF.SCFSR.BIT.DR    ))              /* An error occurs? */
267                 {
268                           ReadData = SCIF.SCFRDR;         /* Read of a data dummy   */
269                           ret_cd = -1;        /* A set of a reception error          */
270                           SCIF.SCFSR.WORD &= 0x0000;  /* A clear of an error       */
271                           SCIF.SCLSR.WORD &= 0x0000;
272                 }
273                 else if( SCIF.SCFSR.BIT.RDF )     /* A data was received? */
274                 {
275                           *Data = SCIF.SCFRDR;   /* A data is acquired         */
```

```
276                     SCIF.SCFSR.BIT.RDF = 0;         /* A clear of a receive data sign */
277                     SCIF.SCFSR.BIT.DR  = 0;         /* A clear of a reception sign */
278                     scif_transmit_data_byte( Data );
279                     if( *Data == '¥n' )/* An obtaining data is CR? */
280                     {
281                             break;  /* A processing is completed. */
282                     }
283                     if( *Data == 0x0d )         /* An obtaining data is CR? */
284                     {
285                             break;  /* A processing is completed. */
286                     }
287                     Data++; /* The following set of the one-plus-one address which the data acquired */
288                     if( ++i == 4 )
289                     {
290                             ret_cd = -1;
291                     }
292             }
293             if( ret_cd == -1 )
294             {
295                     break;
296             }
297     }
298     return( ret_cd );
299 }
300
301 /*""FUNC COMMENT""************************************************
302 * ID                    :
303 * Outline               : Sample Program Main
304 *                       :
305 * Include               :
306 * Declaration           : char scif_recive_data_byte( char   *Data )
307 * Description           : A data reception of SCIF
308 *                       :
309 *                       :
310 *                       :
311 *                       :
312 *                       :
313 * Limitation            :
314 *                       :
315 * Argument              : *Data：A receive data is stored.
```

```
316 * Return Value          : -1 : A receive data error
317 * Calling Functions :
318 *""FUNC COMMENT END""**************************************************/
319 char scif_recive_data_byte( char           *Data )
320 {
321     unsigned char     ReadData, i = 0;
322     char      ret_cd = 0;
323
324     for(;;)
325     {
326         if(( SCIF.SCFSR.BIT.ER    ) ||
327            ( SCIF.SCFSR.BIT.BRK ) ||
328            ( SCIF.SCFSR.BIT.DR    ))          /* An error occurs? */
329         {
330             ReadData = SCIF.SCFRDR;          /* Read of a data dummy   */
331             ret_cd = -1;        /* A set of a reception error          */
332             SCIF.SCFSR.WORD &= 0x0000;  /* A clear of an error        */
333             SCIF.SCLSR.WORD &= 0x0000;
334         }
335         else if( SCIF.SCFSR.BIT.RDF )     /* A data was received?     */
336         {
337             *Data = SCIF.SCFRDR;   /* A data is acquired          */
338             SCIF.SCFSR.BIT.RDF = 0;          /* A clear of a receive data sign */
339             SCIF.SCFSR.BIT.DR    = 0;          /* A clear of a receive data sign */
340 //          scif_transmit_data_byte( Data );
341             break;    /* A processing is completed. */
342         }
343     }
344     return( ret_cd );
345 }
346
```

(6)　”scif.h”

SCIF0 初期設定、シリアルドライバ関数で使用するヘッダファイルです。

```
01
02 #ifndef _SCIF_H
03 #define _SCIF_H
04
05 #include "config.h"
06
07 #if defined(CONFIG_SCIF0)
08 #define SCIF     (*(volatile struct st_scif      *)0xFFEA0000) /* SCIF0     Address */
09 #elif defined(CONFIG_SCIF1)
10 #define SCIF     (*(volatile struct st_scif      *)0xFFEB0000) /* SCIF1     Address */
11 #elif defined(CONFIG_SCIF2)
12 #define SCIF     (*(volatile struct st_scif      *)0xFFEC0000) /* SCIF2     Address */
13 #elif defined(CONFIG_SCIF3)
14 #define SCIF     (*(volatile struct st_scif      *)0xFFED0000) /* SCIF3     Address */
15 #elif defined(CONFIG_SCIF4)
16 #define SCIF     (*(volatile struct st_scif      *)0xFFEE0000) /* SCIF4     Address */
17 #elif defined(CONFIG_SCIF5)
18 #define SCIF     (*(volatile struct st_scif      *)0xFFEF0000) /* SCIF5     Address */
19 #endif /* CONFIG_SCIFn */
20
21 //#define SCBRR_VALUE(bps, clk) ((clk+16*bps)/(16*bps)-1)
22 #define SCBRR_VALUE(bps, clk) ((clk)/(32*bps)-1)
23
24 /* SCFCR */
25 #define         RTRG1  0
26 #define         RTRG16          1
27 #define         RTRG32          2
28 #define         RTRG48          3
29 #define         TTRG32          0
30 #define         TTRG16          1
31 #define         TTRG2  2
32 #define         TTRG0  3
33
34
35
36 #endif /* _SCIF_H */
```

5.　参考ドキュメント

- ソフトウェアマニュアル
  SH4-A ソフトウェアマニュアル(RJJ09B0090)
  (最新版をルネサスエレクトロニクスホームページから入手してください)

- ハードウェアマニュアル
  SH7786 グループハードウェアマニュアル(RJJ09B0533)
  (最新版をルネサスエレクトロニクスホームページから入手してください)

- 評価ボードマニュアル
  SH-4A Multi SH7786 CPU ボード AP-SH4AD-0A ハードウェアマニュアル
  (最新版は株式会社アルファプロジェクトにお問い合わせ下さい)

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	11.4.1	—	初版発行
1.01	11.4.5	3	適用条件内クロック説明修正（四捨五入切捨）
		3	適用条件内 PCIEC 仕様追記
		5	用語の説明追記（PCI Express, Root port, End point）
		6	本アプリケーションノートの適用範囲の説明修正
		13	ブリッジ機能、コネクションの確立説明追記
		28	DMA 転送説明追記
		37	参考プログラムの仕様追記
		38	参考プログラムの仕様の説明修正
		39	main 関数フローDMA 転送追記
		56	DMA 転送関数フロー追加
		57-63	プロラム例 PCIe_DemoSample.c 更新
		66-89	プロラム例 pcie.c 更新
		90-98	プロラム例 pcie.h 更新
		109	PCIe 商標追記
1.02	11.4.27	13	バーチャルチャネルについての説明更新
		28	DMA 転送サイズ長最大値 128byte→1024byte に修正
		28	DMA 転送データ長最大値 4K バイト→1K バイトに修正
		31	VCX 転送について説明削除
		31	LD32 についての説明削除
		32	TC フィールドについての説明更新

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

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