EPSON
EXCEED YOUR VISION

S2S65A30
USB2.0 Device

セイコーエプソン株式会社

Rev.1.10

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 本資料に掲載されている製品のうち「外国為替及び外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。

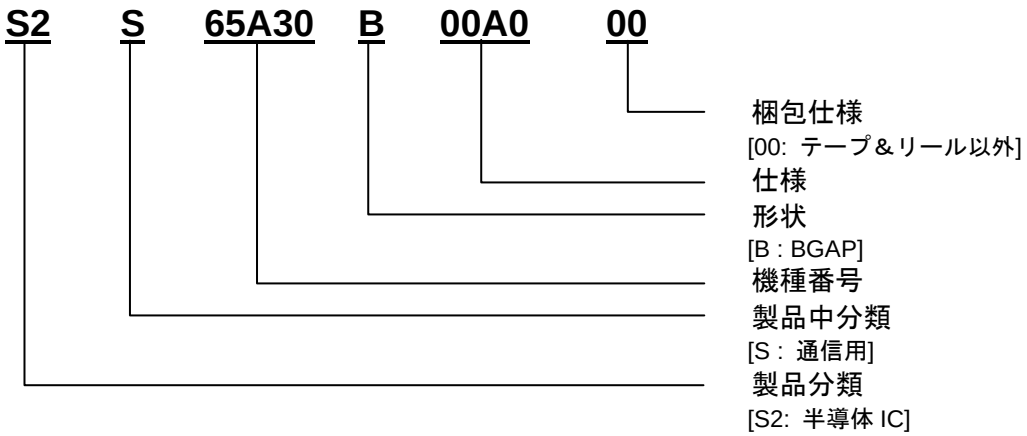
5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

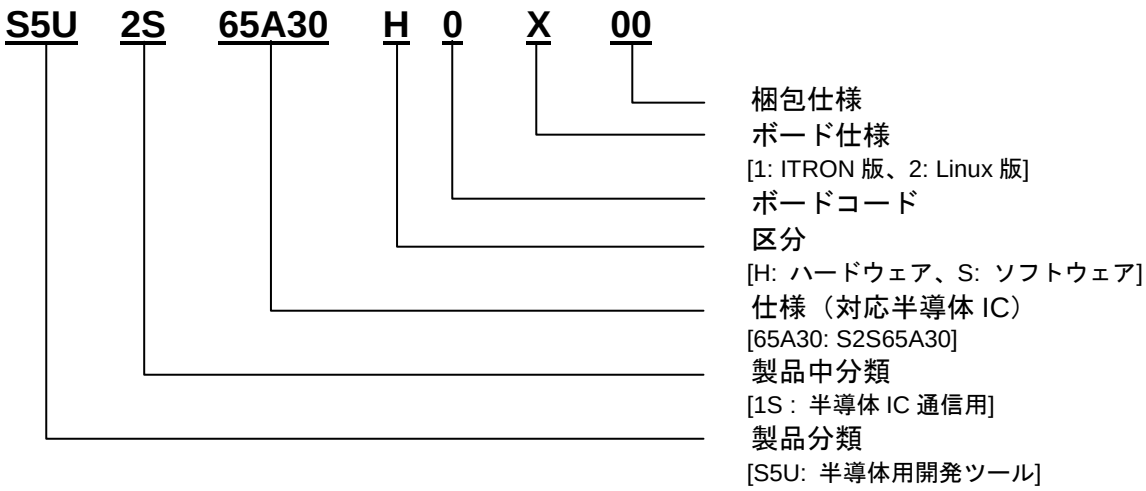

は、ARM 社の登録商標です。
CompactFlash は Sandisk 社の登録商標です。
その他記載の会社名、製品名は、それぞれの会社の商標もしくは登録商標です。

製品型番体系

●デバイス



●開発ツール



使用上の注意事項

本書のレジスタの記述に関しては、以下のことにご注意ください。

本書のレジスタに関する記述については、以下のような省略を用いることがあります。

R/W：	リードおよびライト
RO：	リードオンリ
WO：	ライトオンリ
RSV：	予約ビット／レジスタ（特に指定のない場合は“0”を書き込んでください。）
n/a：	not available（特に指定のない場合は“0”を書き込んでください。）

特に指定のない場合、レジスタの予約ビットには“0”をセットしてください。予約ビットに書き込みを行うと予想できない結果になることがあります。“n/a”と記載してあるビットはハードウェアに影響をあたえません。

あるレジスタは特定の条件のときのみアクセスできるようになっています。アクセス不可のレジスタへのリード／ライトは無効です。

< *Table of Contents* >

<List of Figures>

<List of Tables>

Appendix 2　USB Device コントローラ

A2.1　Scope

本仕様書は、セイコーエプソン㈱製 S2S65A30*に内蔵されている USB HS Device Controller の機能仕様について説明します。

A2.2　Overview

S2S65A30*には、セイコーエプソン製の USB コントローラ「S1R72V05*」のデバイス機能と互換性のある USB コントローラを搭載しています。

- HS（High Speed＝480Mbps）及び FS（Full Speed＝12Mbps）転送サポート
- FS/HS ターミネーション内蔵（外付け回路不要）
- VBUS 5V I/F（外付け保護回路は必要）
- コントロール/バルク/インタラプト転送をサポート
- コントロール転送用（EP0）、Bulk 転送専用 1 本、Bulk/Interrupt 転送用 4 本の Endpoint をサポート　S2S65A30 では、EPa-e を使用します。マニュアル中の f-h は使用できません。
- 4.5KB の Endpoint 用 FIFO
- DMA I/F（DMAC3 と組み合わせて使用することが可能）
- 12MHz / 24MHz の水晶発振子によるクロック入力（帰還抵抗 1MΩ 内蔵）

A2.3　Block Diagram

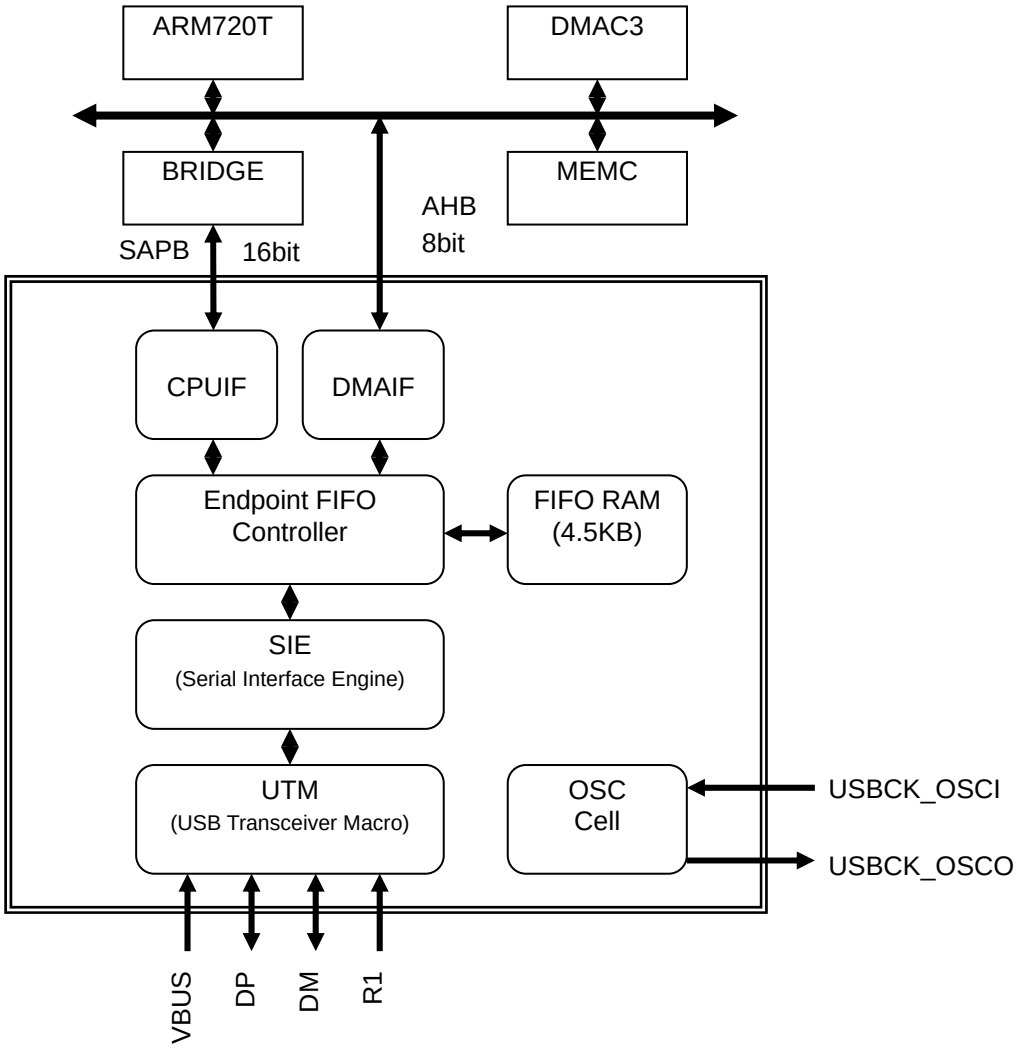


Figure A2.3.1　S2S65A30 USB Controller Block Diagram

A2.4　Pin List

USB インタフェース関連の外部端子は以下のようになっています。

Figure A2.4.2　USB 外部端子

端子名	入出力	端子機能
VBUS	I	USB バス検出信号
DP	I/O	USB データライン Data+
DM	I/O	USB データライン Data-
R1	I	内部動作設定 6.2KΩ±1%の抵抗を VSS 間に接続
USBCK_OSCI	I	USB 内部発振回路入力（12/24MHz）
USBCK_OSCO	O	USB 内部発振回路出力

A2.5　Register

A2.5.1　Register 一覧

本ブロック内にあるコントロール・レジスタのレジスタ・マップをTable A2.5.1～Table A2.5.4 に示します。表中のアドレスはAPBバスにおけるアドレスとなります。これらのベースアドレスは、**0xFFFD_F000**です。

Table A2.5.1　USB レジスター一覧（1/4）

アドレス オフセット	レジスタ名称	説明	R/W	初期値
割込み制御レジスタ				
0x000	MainIntStat	割込みステータスレジスタ	R/(W)	0x00
0x003	CPU_IntStat	CPUIF 関連割込みステータスレジスタ	R/(W)	0x00
0x007	DMA_IntStat	DMA 関連割込みステータスレジスタ	R/(W)	0x00
0x010	MainIntEnb	割込みイネーブルレジスタ	R/W	0x00
0x013	CPU_IntEnb	CPUIF 関連割込みイネーブルレジスタ	R/W	0x00
0x017	DMA_IntEnb	DMA 関連割込みイネーブルレジスタ	R/W	0x00
Power Management レジスタ				
0x020	PM_Control0	Power Management 制御レジスタ	R/W	0x00
0x021	PM_Control1	Power Management ステータスレジスタ	R	0x00
0x022	WakeupTim_L	Wakeup Time 設定レジスタ	R/W	0x00
0x023	WakeupTim_H		R/W	0x00
MISC レジスタ				
0x031	MacroReset	ソフトウェアリセットレジスタ	R/W	0x00
0x033	ModeProtect	設定プロテクトレジスタ	R/W	0x56
0x035	MacroConfig0	クロック設定レジスタ	R/W	0x41
0x037	MacroConfig1	マクロ動作モード設定レジスタ	R/W	0x06
0x039	(fix_CPU_Swap)		R	
0x03A	MacroType0	マクロタイプレジスタ 0	R	0x44
0x03B	MacroType1	マクロタイプレジスタ 1	R	0x08
0x03C	MacroType2	マクロタイプレジスタ 2	R	0x01
0x03D	MacroType3	マクロタイプレジスタ 3	R	0x10
0x03E	FIFO_CapacityL	FIFO RAM 容量レジスタ	R	0x00
0x03F	FIFO_CapacityH		R	0x12
DMA 制御レジスタ				
0x041	DMA0_Config	DMA0 設定レジスタ	R/W	0x00
0x042	DMA0_Control	DMA0 制御レジスタ	R/W	0x00
0x044	DMA0_Remain_L	DMA0 FIFO 残量レジスタ	R	0x00
0x045	DMA0_Remain_H		R	0x00
0x048	DMA0_Count_HL	DMA0 転送カウンタレジスタ	R/W	0x00
0x049	DMA0_Count_HH		R/W	0x00
0x04A	DMA0_Count_LL		R/W	0x00
0x04B	DMA0_Count_LH		R/W	0x00

Table A2.5.2　USB レジスター覧（2/4）

アドレス オフセット	レジスタ名称	説明	R/W	初期値
USB 制御レジスタ				
0x080	DeviceIntStat	USB 割込みステータスレジスタ	R/(W)	0x00
0x081	EPrIntStat	Endpoint 割込みステータスレジスタ	R	0x00
0x082	SIE_IntStat	SIE 割込みステータスレジスタ	R/(W)	0x00
0x084	FIFO_IntStat	FIFO 割込みステータスレジスタ	R/(W)	0x00
0x085	BulkIntStat	Bulk 転送サポート割込みステータスレジスタ	R/(W)	0x00
0x087	EP0IntStat	EP0 割込みステータスレジスタ	R/(W)	0x00
0x088	EPaIntStat	EPa 割込みステータスレジスタ	R/(W)	0x00
0x089	EPbIntStat	EPb 割込みステータスレジスタ	R/(W)	0x00
0x08A	EPcIntStat	EPc 割込みステータスレジスタ	R/(W)	0x00
0x08B	EPdIntStat	EPd 割込みステータスレジスタ	R/(W)	0x00
0x08C	EPeIntStat	EPe 割込みステータスレジスタ	R/(W)	0x00
0x090	DeviceIntEnb	USB 割込み制御レジスタ	R/W	0x00
0x091	EPrIntEnb	Endpoint 割込み制御レジスタ	R/W	0x00
0x092	SIE_IntEnb	SIE 割込み制御レジスタ	R/W	0x00
0x094	FIFO_IntEnb	FIFO 割込み制御レジスタ	R/W	0x00
0x095	BulkIntEnb	Bulk 転送サポート割込み制御レジスタ	R/W	0x00
0x097	EP0IntEnb	EP0 割込み制御レジスタ	R/W	0x00
0x098	EPaIntEnb	EPa 割込み制御レジスタ	R/W	0x00
0x099	EPbIntEnb	EPb 割込み制御レジスタ	R/W	0x00
0x09A	EPcIntEnb	EPc 割込み制御レジスタ	R/W	0x00
0x09B	EPdIntEnb	EPd 割込み制御レジスタ	R/W	0x00
0x09C	EPeIntEnb	EPe 割込み制御レジスタ	R/W	0x00
0x0A0	ResetDTM	トランシーバマクロリセットレジスタ	R/W	0x01
0x0A2	NegoControl	ネゴシエーション制御レジスタ	R/W	0x00
0x0A4	USB_Status	USB バスステータスレジスタ	R/W	0xXX
0x0A5	XcvrControl	トランシーバマクロ制御レジスタ	R/W	0x41
0x0A6	USB_Test	USB テストレジスタ	R/W	0x00
0x0A8	EPnControl	EP 共通制御レジスタ	W	0xXX
0x0A9	EPrFIFO_Clr	FIFO クリアレジスタ	W	0xXX
0x0AA	ClrAllEPnJoin	EP ジョインクリアレジスタ	W	0xXX
0x0AC	BulkOnlyControl	Bulk 転送サポート制御レジスタ	R/W	0x00
0x0AD	BulkOnlyConfig	Bulk 転送サポート設定レジスタ	R/W	0x00
0x0B0	EP0SETUP_0	EP0 セットアップデータレジスタ	R	0x00
0x0B1	EP0SETUP_1		R	0x00
0x0B2	EP0SETUP_2		R	0x00
0x0B3	EP0SETUP_3		R	0x00
0x0B4	EP0SETUP_4		R	0x00
0x0B5	EP0SETUP_5		R	0x00
0x0B6	EP0SETUP_6		R	0x00
0x0B7	EP0SETUP_7		R	0x00
0x0B8	USB_Address	USB アドレスレジスタ	R/(W)	0x00
0x0BA	SETUP_Control	セットアップステージ制御レジスタ	R/W	0x00
0x0BE	FrameNumber_L	フレーム番号レジスタ	R	0x00
0x0BF	FrameNumber_H		R	0x80

Table A2.5.3　USB レジスター覧（3/4）

アドレス オフセット	レジスタ名称	説明	R/W	初期値
USB 制御レジスタ				
0x0C0	EP0MaxSize	EP0 マックスパケットサイズ設定レジスタ	R/W	0x40
0x0C1	EP0Control	EP0 制御レジスタ	R/W	0x00
0x0C2	EP0ControlIN	EP0 IN 転送制御レジスタ	R/W	0x00
0x0C3	EP0ControlOUT	EP0 OUT 転送制御レジスタ	R/W	0x00
0x0C5	EP0JOin	EP0 ジョイン設定レジスタ	R/W	0x00
0x0D0	EPaMaxSize_L	EPa マックスパケットサイズ設定レジスタ	R/W	0x00
0x0D1	EPaMaxSize_H		R/W	0x00
0x0D2	EPaConfig	EPa 設定レジスタ	R/W	0x00
0x0D4	EPaControl	EPa 制御レジスタ	R/W	0x00
0x0D5	EPaJoin	EPa ジョイン設定レジスタ	R/W	0x00
0x0D8	EPbMaxSize_L	EPb マックスパケットサイズ設定レジスタ	R/W	0x00
0x0D9	EPbMaxSize_H		R/W	0x00
0x0DA	EPbConfig	EPb 設定レジスタ	R/W	0x00
0x0DC	EPbControl	EPb 制御レジスタ	R/W	0x00
0x0DD	EPbJoin	EPb ジョイン設定レジスタ	R/W	0x00
0x0E0	EPcMaxSize_L	EPc マックスパケットサイズ設定レジスタ	R/W	0x00
0x0E1	EPcMaxSize_H		R/W	0x00
0x0E2	EPcConfig	EPc 設定レジスタ	R/W	0x00
0x0E4	EPcControl	EPc 制御レジスタ	R/W	0x00
0x0E5	EPcJoin	EPc ジョイン設定レジスタ	R/W	0x00
0x0E8	EPdMaxSize_L	EPd マックスパケットサイズ設定レジスタ	R/W	0x00
0x0E9	EPdMaxSize_H		R/W	0x00
0x0EA	EPdConfig	EPd 設定レジスタ	R/W	0x00
0x0EC	EPdControl	EPd 制御レジスタ	R/W	0x00
0x0ED	EPdJoin	EPd ジョイン設定レジスタ	R/W	0x00
0x0F0	EPeMaxSize_L	EPe マックスパケットサイズ設定レジスタ	R/W	0x00
0x0F1	EPeMaxSize_H		R/W	0x00
0x0F2	EPeConfig	EPe 設定レジスタ	R/W	0x00
0x0F4	EPeControl	EPe 制御レジスタ	R/W	0x00
0x0F5	EPeJoin	EPe ジョイン設定レジスタ	R/W	0x00

Table A2.5.4　USB レジスター覧（4/4）

アドレス オフセット	レジスタ名称	説明	R/W	初期値
USB FIFO 設定レジスタ				
0x110	EPaStartAdrs_L	EPa FIFO スタートアドレス設定レジスタ	R/W	0x00
0x111	EPaStartAdrs_H		R/W	0x00
0x112	EPbStartAdrs_L	EPb FIFO スタートアドレス設定レジスタ	R/W	0x00
0x113	EPbStartAdrs_H		R/W	0x00
0x114	EPcStartAdrs_L	EPc FIFO スタートアドレス設定レジスタ	R/W	0x00
0x115	EPcStartAdrs_H		R/W	0x00
0x116	EPdStartAdrs_L	EPd FIFO スタートアドレス設定レジスタ	R/W	0x00
0x117	EPdStartAdrs_H		R/W	0x00
0x118	EPeStartAdrs_L	EPe FIFO スタートアドレス設定レジスタ	R/W	0x00
0x119	EPeStartAdrs_H		R/W	0x00
0x128	DMA0_FIFO_Control	DMA0-FIFO 制御レジスタ	R/W	0x00
0x12A				
0x130	FIFO_Rd_0	FIFO リードレジスタ	R	0xXX
0x131	FIFO_Rd_1		R	0xXX
0x132	FIFO_Wr_0	FIFO ライトレジスタ	W	0xXX
0x133	FIFO_Wr_1		W	0xXX
0x134	FIFO_RdRemain_L	FIFO リード残量レジスタ	R	0x00
0x135	FIFO_RdRemain_H		R	0x00
0x136	FIFO_WrRemain_L	FIFO ライト残量レジスタ	R	0x00
0x137	FIFO_WrRemain_H		R	0x00
0x138	FIFO_ByteRd	FIFO バイトリードレジスタ	R	0xXX
0x140	RAM_RdAdrs_L	RAM リードアドレス設定レジスタ	R/W	0x00
0x141	RAM_RdAdrs_H		R/W	0x00
0x142	RAM_RdControl	RAM リード制御レジスタ	R/W	0x00
0x143	RAM_RdCount	RAM リードカウント設定レジスタ	R/W	0x00
0x144	RAM_WrAdrs_L	RAM ライトアドレス設定レジスタ	R/W	0x00
0x145	RAM_WrAdrs_H		R/W	0x00
0x146	RAM_WrDoor_0	RAM ライトレジスタ	W	0xXX
0x147	RAM_WrDoor_1		W	0xXX
0x150 ～ 0x16F	RAM_Rd_00 ～ RAM_Rd_1F	RAM リードデータレジスタ	R	0xXX
テストレジスタ（Reserved）				
0x170 ～ 0x17F	Reserved	テスト用レジスタ	ー	

A2.5.2　Register 詳細説明

A2.5.2.1　Register Access 上の注意事項

(1) Register が割り付けられていない Address に対しては書き込みを行わないでください。
(2) Register が割り付けられていない bit に対しては“0x0”の書き込みのみとしてください。
(3) SLEEP / SNOOZE 時にでも読み書きできる Register は*太字斜体*で示しています。
(4) それ以外の Register は、ACTIVE60 / ACT_DEVICE 時に読み書きができます。

A2.5.2.2　割込み制御レジスタ

Main Interrupt Status USB[0x000]　　初期値 = 0x00							
Device IntStat(R)	ー	CPU_ IntStat(R)	ー	ー	DMA_ IntStat(R)	ー	***Finished PM***
7	6	5	4	3	2	1	0

USB マクロの割り込み要因を示します。
このレジスタには割り込み要因を間接指示するビットと直接指示するビットがあります。割り込み要因を間接指示するビットは、それぞれに対応する割り込みステータスレジスタをリードすることにより、割り込み要因を直接指示するビットまで辿ることができます。割り込み要因を間接指示するビットは、リードオンリーであり、大元の割り込み要因を直接指示するビットをクリアすることにより、自動的にクリアされます。割り込み要因を直接指示しているビットは、書き込み可能であり、該当ビットに“1”を書き込むことにより、割り込み要因をクリアすることができます。MainInterruptEnb レジスタにより、対応するビットの割り込みがイネーブルにされているときに、割り込み要因が “1” にセットされると INTC に対して割り込みが発生します。

Bit 7 :　**DeviceIntStat**
割り込み要因を間接指示します。
Device_IntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する SIE_DeviceIntEnb レジスタのビットがイネーブルにされているときに“1”にセットされます。このビットは、SLEEP / SNOOZE 中もリード有効です。

Bit 6 :　**Reserved**

Bit 5 :　**CPU_IntStat**
割り込み要因を間接指示します。
CPU_IntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する CPU_IntEnb レジスタのビットがイネーブルにされているときに “1” にセットされます。

Bit [4:3] :　**Reserved**

Bit 2 :　**DMAIntStat**
割り込み要因を間接指示します。
DMAIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する DMAIntEnb レジスタのビットがイネーブルにされているときに“1”にセットされます。

Bit 1 :　**Reserved**

Bit 0 :　**FinishedPM**
割り込み要因を直接指示します。
PM_Control レジスタで、GoSLEEP,GoSNOOZE,GoActive60,GoActDevice を設定した場合、指示したそれぞれの状態に達したら、このビットは“1”にセットされます。このビットは SLEEP / SNOOZE 中も有効です。

CPU Interrupt Status USB[0x003]　初期値 = 0x00							
RAM_ RdCmp	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0

CPU インタフェース関連の割り込みを表示します。
全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit 7　**RAM_RdCmp**
割り込み要因を直接指示します。
RAM_Rd 機能において、RAM からデータを読み出し、RAM_Rd_XX のデータが有効になったところで、“1”にセットされます。

Bit [6:0] :　**Reserved**

DMA Interrupt Status USB[0x007]　初期値 = 0x00							
–	–	–	–	–	–	DMA0_ CountUp	DMA0_ Cmp
7	6	5	4	3	2	1	0

DMA 関連の割り込みを表示します。
全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit [7:2] :　**Reserved**

Bit 1 :　**DMA0_CountUp**
割り込み要因を直接指示します。
転送モードがフリーランモードで動作しているときに、DMA0_Count_HH,HL,LH,LL の値がオーバーフローしたときに、“1”にセットされます。DMA0_Count_HH,HL,LH,LL の値は 0 に戻り、DMA 動作は継続します。

Bit 0 :　**DMA0_Cmp**
割り込み要因を直接指示します。
DMA 転送が停止されるか、或いは、指定された転送数が終了し、終了処理が完了したときに“1”にセットされます。

Main Interrupt Enable USB[0x010]　　初期値 = 0x00							
EnDevice IntStat	–	EnCPU IntStat	–	–	EnDMA_ IntStat	–	***EnFinished PM***
7	6	5	4	3	2	1	0

MainIntStat レジスタの割り込み要因による INTC モジュールに対する割り込み信号のアサートを許可／禁止するレジスタです。対応するビットを“1”にセットすることで割り込みを許可します。
EnDeviceIntStat, EnFinishedPM ビットは SLEEP / SNOOZE 中も有効です。

CPU Interrupt Enable USB[0x013]　　初期値 = 0x00							
EnRAM_ RdCmp	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0

CPU_IntStat レジスタの割り込み要因による、MainIntStat レジスタの CPU_IntStat ビットのアサートを許可／禁止します。

DMA Interrupt Enable USB[0x017]　　初期値 = 0x00							
–	–	–	–	–	–	EnDMA0_ CountUp	EnDMA0_ Cmp
7	6	5	4	3	2	1	0

DMAIntStat レジスタの割り込み要因による、MainIntStat レジスタの DMAIntStat ビットのアサートを許可／禁止します。

A2.5.2.3　Power Management 制御レジスタ

Power Management Control 0 USB[0x020]　　初期値 = 0x00							
Go SLEEP 7	***Go SNOOZE*** 6	***Go Active60*** 5	***Go ActDevice*** 4	－ 3	－ 2	－ 1	－ 0

本 LSI のパワーマネージメント関連の動作設定を行います。
このレジスタは SLEEP / SNOOZE 中も有効です。

Bit 7 :　**GoSLEEP**

SLEEP ステート以外のステートから、SLEEP ステートへの移行を開始します。
SNOOZE ステート時に本ビットを“1”にセットすると、オシレータの発振を停止し SLEEP ステートに移行します。
ACTIVE60 ステート時に本ビットを“1”にセットすると、まず PLL60 の発振を停止し、その後オシレータの発振を停止し SLEEP ステートに移行します。
ACT_DEVICE ステート時に本ビットを“1”にセットすると、最初に DevicePLL480 の発振を停止し、次に PLL60 の発振を停止し、その後オシレータの発振を停止し SLEEP ステートに移行します。
どのステートからの移行であっても、移行が完了し次第、本ビットは自動的にクリアされ、同時に MainIntStat.FinishedPM ビットがセットされます。

Bit 6 :　**GoSNOOZE**

SNOOZE ステート以外のステートから、SNOOZE ステートへの移行を開始します。
SLEEP ステート時に本ビットを“1”にセットすると、オシレータの発振を開始し、オシレータ発振安定時間（WakeupTim_H, L に設定された時間）経過後、SNOOZE ステートに移行します。
ACTIVE60 ステート時に本ビットを“1”にセットすると、PLL60 の発振を停止し、SNOOZE ステートに移行します。
ACT_DEVICE ステート時に本ビットを“1”にセットすると、最初に DevicePLL480 の発振を停止し、次に PLL60 の発振を停止し、SNOOZE ステートに移行します。
どのステートからの移行であっても、移行が完了し次第、本ビットは自動的にクリアされ、同時に MainIntStat.FinishedPM ビットがセットされます。

Bit 5 :　**GoActive60**

ACTIVE60 ステート以外のステートから、ACTIVE60 ステートへの移行を開始します。
SLEEP ステート時に本ビットを“1”にセットすると、オシレータの発振を開始し、オシレータ発振安定時間（WakeupTim_H, L に設定された時間）経過後、PLL60 の発振を開始し、PLL60 発振安定時間（約 250µs）経過後、ACTIVE60 ステートに移行します。
SNOOZE ステート時に本ビットを“1”にセットすると、PLL60 の発振を開始し、PLL 発振安定時間（約 250µs）経過後、ACTIVE60 ステートに移行します。
ACT_DEVICE ステート時に本ビットを“1”にセットすると、DevicePLL480 の発振を停止し、ACTIVE60 ステートに移行します。
どのステートからの移行であっても、移行が完了し次第、本ビットは自動的にクリアされ、同時に MainIntStat.FinishedPM ビットがセットされます。

Bit 4 :　**GoActDevice**

ACT_DEVICE ステート以外のステートから、ACT_DEVICE ステートへの移行を開始します。
SLEEP・ステート時に本ビットを“1”にセットすると、オシレータの発振を開始し、オシレータ発振安定時間（WakeupTim_H, L に設定された時間）経過後、PLL60 の発振を開始し、PLL60 発振安定時間（約 250µs）経過後、DevicePLL480 の発振を開始し、PLL480 発振安定時間（約 250µs）経過後、ACT_DEVICE ステートに移行します。
SNOOZE ステート時に本ビットを“1”にセットすると、PLL60 の発振を開始し、PLL 発振安定時間（約 250µs）経過後、DevicePLL480 の発振を開始し、PLL480 発振安定時間（約 250µs）経過後、ACT_DEVICE・ステートに移行します。
ACTIVE60 ステート時に本ビットを“1”にセットすると、DevicePLL480 の発振を開始し、PLL480 発振安定時間（約 250µs）経過後、ACT_DEVICE ステートに移行します。
どのステートからの移行であっても、移行が完了し次第、本ビットは自動的にクリアされ、同時に MainIntStat.FinishedPM ビットがセットされます。

※本 LSI は SLEEP / SNOOZE 中にアクセスできない割り込みステータス（以下、同期ステータス）により、SNOOZE 中に XINT 信号がアサートされないようマスクしておりますが、SNOOZE 解除と同時に XINT 端子がアサートされるのを回避するため、F/W により以下の処理を行って下さい。
＜SLEEP / SNOOZE 開始前＞
同期ステータスを処理し、クリアする。（~IntStat）
同期ステータスをディスエーブルにする。（~IntEnb）
＜SLEEP / SNOOZE 解除後＞
同期ステータスをクリアする。（~IntStat）
同期ステータスをイネーブルにする。（~IntEnb）

Bit [3:0] :　**Reserved**

Power Management Control 1
USB[0x021]　**初期値** = 0x00

–	–	–	–	***PM_State[3:0]***(R)			
7	6	5	4	3	2	1	0

本 LSI のパワーマネージメント関連の動作状態をモニタできます。
このレジスタは SLEEP / SNOOZE 中も有効です。

Bit [7:4] :　**Reserved**

Bit [3:0] :　**PM_State[3:0]**

電力モードのステートを表します。
0000 : SLEEPステート　（OSCオフ、PLL60 オフ、DevicePLL480 オフ）
0001 : SNOOZEステート　（OSCオン、PLL60 オフ、DevicePLL480 オフ）
0011 : ACTIVE60 ステート　（OSCオン、PLL60 オン、DevicePLL480 オフ）
0111 : ACT_DEVICEステート　（OSCオン、PLL60 オン、DevicePLL480 オン）
その他 : 未使用
なお本ステートは、PM_Contorl0.GoXXXX をセットしてから MainIntStat.FinishedPM 割り込みステータスがセットされ、PM_Control0.GoXXXX ビットがクリアされるまでは、該当ステートに向かい逐次変化致しますので、参照しないでください。

Wakeup Time[7:0]							
USB[0x022]　初期値 = 0x00							
WakeupTim[7:0]							
7	6	5	4	3	2	1	0

Wakeup Time[15:8]							
USB[0x023]　初期値 = 0x00							
WakeupTim[15:8]							
7	6	5	4	3	2	1	0

SLEEP ステートから SNOOZE ステートへ復帰する際のオシレータ発振安定時間を設定します。本レジスタは SLEEP 時にもアクセスが可能です。
SLEEP ステート時に、PM_Control0.GoActDevice、PM_Control0.GoActive60、PM_Control0. GoSNOOZE ビットに“1”が書き込まれた際に、発振セルをイネーブルにし、オシレータの発振を開始します。この時、カウンタにこの Wakeup_Tim の設定値をロードし、OSC の立ち上がりにてカウントダウンを始めます。カウントダウン終了後、内部 OSCCLK のゲートを開き、PLL 等の回路に CLK を送出開始します。このオシレータ発振安定時間は、発振子、発振セル、基板、負荷容量等により変化します。もし、デバイス動作時において、USB の SUSPEND 時に SLEEP ステートまで落とす場合は、USB の RESET 検出から 5.1ms 以内に 60MHz±10%に内部 SCLK が安定していなければなりません。
したがって、オシレータ発振安定時間 ＋ PLL60 安定時間（250µs 未満）＋ PLL480 安定時間（250µs 未満）の合計が 5.1ms 以下とならなければなりません。

A2.5.2.4　MISC レジスタ

Macro Reset							
USB[0x031]　初期値 = 0xXX							
–	–	–	–	–	–	–	***AllReset*** (W)
7	6	5	4	3	2	1	0

本 LSI をリセットします。
SLEEP / SNOOZE 時でもアクセス有効です。

Bit [7:1] :　**Reserved**

Bit 0 :　**AllReset**
本 LSI の全回路をリセットします。外部リセット端子(XRST)と同義です。
なお、このレジスタにリセット用途以外で、書き込みしないで下さい。
AC スペックに違反してこのレジスタにリセット用途以外の書き込みを行った場合､誤動作の原因となりますのでご注意下さい。

Mode Protect							
USB[0x033]　初期値 = 0x56							
ModeProtect[7:0]							
7	6	5	4	3	2	1	0

Bit [7:0] :　**ModeProtect[7:0]**
MacroConfig0 レジスタ及び ClkSelect.ClkSelect ビットの値を保護します｡このレジスタに 56h を書き込むと MacroConfig0 レジスタ及び ClkSelect.ClkSelect ビットへのライトアクセスが有効になります。
通常使用においては、MacroConfig0 レジスタ及び ClkSelect.ClkSelect ビットを任意に設定した後、このレジスタに 56h 以外の値（例えば 00h）を設定して、MacroConfig0 レジスタ及び ClkSelect.ClkSelect ビットの設定を保護して下さい。
このビットは、SLEEP / SNOOZE 中もアクセス有効です。

Macro Configuration 0 USB[0x035]　　初期値 = 0x41							
–	–	–	–	–	–	–	***ClkSelect***
7	6	5	4	3	2	1	0

Bit [7:1] :　**Reserved**

Bit 0 :　**ClkSelect**
本 LSI で使用するクロックを選択します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： 12MHz
1： 24MHz

Macro Configuration 1 USB[0x037]　　初期値 = 0x06							
IntLevel	***IntMode***	***DREQ_ Level***	***DACK_ Level***	***CS_Mode***	***CPU_Swap***	***BusMode***	***Bus8x16***
7	6	5	4	3	2	1	0

本LSIの動作モードを設定します。本レジスタの初期値はS2S65A30 用に最適化されていますので、特に必要のない限り設定を変更しないでください。

Bit 7 :　**IntLevel**
XINT の論理レベルを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： 負論理
1： 正論理

Bit 6 :　**IntMode**
XINT の出力モードを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： 1／0 モード
1： Hi-z／0 モード

Bit 5 :　**DREQ_Level**
XDREQ0,1 の論理レベルを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： 負論理
1： 正論理

Bit 4 :　**DACK_Level**
XDACK0,1 の論理レベルを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： 負論理
1： 正論理

Bit 3 :　**CS_Mode**
DMA0 の動作モードを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
0： XDACK0,1 がアサートされているとき有効な DMA アクセスとして動作します。
1： XCS 且つ XDACK0,1 がアサートされているとき有効な DMA アクセスとして動作します。

Bit 2 :　**CPU_Swap**
16bit mode 時の CPU バスを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。
8bit mode 時はこのビットをセットしないで下さい。
0： 偶数アドレスを上位側、奇数アドレスを下位側とします。
1： 偶数アドレスを下位側、奇数アドレスを上位側とします。
このビットの設定は、レジスタ書き込み後、039h 番地をリードすることにより有効になります。
ChipReset.ResetAll ビットにて、回路のリセットを行った場合、レジスタの値は初期化されますが、その設定が有効になるのは、上述同様に 039h 番地をリードした後になります。

Bit [1:0] :　**BusMode, Bus8x16**

CPU の動作モードを設定します。このビットは、SLEEP / SNOOZE 中もアクセス有効です。

動作モード	bit1.BusMode	bit0.Bus8x16
16bit Strobe mode	0	0
16bit BE mode	1	*
8bit mode	0	1

Macro Type 0
USB[0x03A]　初期値 = 0x44

Device_Host[7:0](R)							
7	6	5	4	3	2	1	0

本マクロのマクロタイプを表します。

Bit [7:0] :　**Device/Host[7:0]**

Device か Host かを表します。
Device : 0x44
Host : 0x48

Macro Type 1
USB[0x03B]　初期値 = 0x05

–	–	–	***Exist_IDE***(R)	***EP_Num[3:0]*** (R)			
7	6	5	4	3	2	1	0

本マクロのマクロタイプを表します。

Bit [7:5] :　**Reserved**

Bit 4 :　**Exist_IDE**

IDE が実装されているかを表します。
Not mount : 0x0
Mount : 0x1

Bit [3:0] :　**EP_Num[3:0]**

End Point 数を表します。
16 進数で表します。

Macro Type 2
USB[0x03C]　初期値 = 0x01

–	–	–	–	***DMAch_Count[3:0]*** (R)			
7	6	5	4	3	2	1	0

本マクロのマクロタイプを表します。

Bit [7:4] :　**Reserved**

Bit [3:0] :　**DMAch_Count[3:0]**

DMA チャンネル数の総数を表します。
16 進数でチャンネル数を表します

Macro Type 3							
USB[0x03D]　初期値 = 0x10							
PortDMA_Mount[3:0] (R)				***CPUDMA_Mount[3:0]*** (R)			
7	6	5	4	3	2	1	0

本マクロのマクロタイプを表します。

Bit [7:4] :　**PortDMA_Mount[3:0]**
実装されている Sync DMA のチャンネルを表します。
2 進数で実装されているチャンネルを表します。

Bit [3:0] :　**CPUDMA_Mount[3:0]**
実装されている CPU DMA のチャンネルを表します。
2 進数で実装されているチャンネルを表します。

FIFO Capacity[7:0]							
USB[0x03E]　初期値 = 0x00							
FIFO_Capacity[7:0]							
7	6	5	4	3	2	1	0

FIFO Capacity[15:8]							
USB[0x03F]　初期値 = 0x12							
FIFO_Capacity[15:8]							
7	6	5	4	3	2	1	0

実装されている FIFO RAM の容量を表します。

A2.5.2.5　DMA 制御レジスタ

DMA0 Configuration							
USB[0x041]　初期値 = 0x00							
FreeRun	−	−	−	ActiveDMA	−	−	−
7	6	5	4	3	2	1	0

DMA0 の動作モードを設定します。

Bit 7 :　**FreeRun**
DMA0 の動作モードを設定します。
0 : カウントモード
1 : フリーランモード

Bit [6:4] :　**Reserved**

Bit 3 :　**ActiveDMA**
PortDMA0 を有効にします。転送中にこの bit をクリアすると即座に転送を中止します。再転送する際は USB FIFO、PortDMA の初期化を行ってください。通常の停止では DMA_Stop を使用してください。
なおこの bit は USBFIFO が Clear されたり、Join が切り替えられた場合、自動的に Clear されます。
1： PortDMA0 有効
0： PortDMA0 無効

Bit [2:0] :　**Reserved**

DMA0 Control USB[0x042]　初期値 = 0x00							
Running (R) 7	– 6	– 5	Counter Clr（W） 4	Dir 3	– 2	Stop (W) 1	Go (W) 0

DMA0 の制御、及び状態を表示します。

Bit 7 :　**Running**
DMA0 の転送中、このビットが“1”にセットされます。このビットが“1”である間は、EPx{x=0,a~h}Join.JoinDMA0 を書き換えることはできません。

Bit [6:5] :　**Reserved**

Bit 4 :　**CounterClr**
このビットに“1”をセットすると、Count_HH,HL,LH,LL レジスタが 0x00 にクリアされます。Running ビットが“1”であるときは、このビットへの書き込みは無視されます。

Bit 3 :　**Dir**
DMA0 の転送方向を設定します。
0： Port → FIFO RAM（DMA ライト）
1： Port ← FIFO RAM（DMA リード）

Bit 2 :　**Reserved**

Bit 1 :　**Stop**
このビットに“1”をセットすると、DMA0 の転送を終了します。DMA0 の転送を停止すると、DMA_Running ビットを“0”にクリアします。また、CPU_IntStat レジスタの DMA0_Cmp ビットに“1”をセットします。DMA の転送を再開する場合、Running ビットまたは DMA0_Cmp ビットを確認し、DMA が終了するのを待って行って下さい。

Bit 0 :　**Go**
このビットを“1”にセットすると、DMA0 の転送を開始します。

DMA0 Remain[7:0]
USB[0x044]　　初期値 = 0x00

Remain[7:0](R)							
7	6	5	4	3	2	1	0

DMA0 Remain[12:8]
USB[0x045]　　初期値 = 0x00

–	–	–	Remain[12:8](R)				
7	6	5	4	3	2	1	0

Bit [7:0] :　**Remain[7:0]**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**Remain[12:8]**

読み出しの場合、EPx{x=0,a~h}Join.JoinDMA0 ビットによって DMA に接続しているエンドポイントの FIFO 内の残りデータ数を示します。
書き込みの場合、EPx{x=0,a~h}Join.JoinDMA0 ビットによって DMA に接続しているエンドポイントの FIFO の空き容量を示します。DMA 書き込みを行った直後にはこのレジスタで正しい FIFO の空き容量を参照することが出来ません。1CPU サイクル以上の間隔を空けて FIFO の空き容量を確認してください。
このレジスタを読み出す場合は上位バイト、下位バイトの順に読み出してください。

DMA0 Count[23:16]
USB[0x048]　　初期値 = 0x00

Count[23:16]							
7	6	5	4	3	2	1	0

DMA0 Count[31:24]
USB[0x049]　　初期値 = 0x00

Count[31:24]							
7	6	5	4	3	2	1	0

DMA0 Count[7:0]
USB[0x04A]　　初期値 = 0x00

Count[7:0]							
7	6	5	4	3	2	1	0

DMA0 Count[15:8]
USB[0x04B]　　初期値 = 0x00

Count[15:8]							
7	6	5	4	3	2	1	0

Bit [7:0] :　　**Count[23:16]**

Bit [7:0] :　　**Count[31:24]**

Bit [7:0] :　　**Count[7:0]**

Bit [7:0] :　　**Count[15:8]**

カウントモード時に DMA0 の、転送データ長をバイト単位で設定します。最大 0xFFFF_FFFF バイトまで設定可能です。設定された値からダウンカウントします。本レジスタに転送数を設定した後、DMA0_Control.Go ビットに“1”をセットして DMA 転送を起動してください。本レジスタに設定された転送 Byte 数の転送が終了すると、DMA 転送は終了します。
フリーランモードの場合、設定された値からカウントアップします。DMA0_Count レジスタの値がオーバーフローすると、DMAIntStat レジスタの DMA0_CountUp ビットに“1”をセットします。オーバーフロー後もカウントは継続されます。このモードでは、DMA の転送数が参照できます。
DMA 書き込みを行った直後は、このレジスタで正確なカウント数を確認することは出来ません。1CPU サイクル以上の間隔をあけて、カウント数を確認してください。このレジスタをリードする場合は上位バイトから順に読み出してください。

A2.5.2.6　USB 制御レジスタ

Device Interrupt Status USB[0x080]　　初期値 = 0x00							
VBUS_Changed 7	Descriptor Cmp 6	***SIE_IntStat***(R) 5	Bulk_ IntStat(R) 4	RcvEP0 SETUP 3	FIFO_ IntStat(R) 2	EP0_ IntStat(R) 1	EPr_ IntStat(R) 0

USB デバイス関連の割り込みを表示します。
このレジスタには割り込み要因を間接指示するビットと直接指示するビットがあります。割り込み要因を間接指示するビットは、それぞれに対応する割り込みステータスレジスタをリードすることにより、割り込み要因を直接指示するビットまで辿ることができます。割り込み要因を間接指示するビットは、リードオンリーであり、大元の割り込み要因を直接指示するビットをクリアすることにより、自動的にクリアされます。割り込み要因を直接指示しているビットは、書き込み可能であり、該当ビットに“1”を書き込むことにより、割り込み要因をクリアすることができます。

Bit 7 :　**VBUS_Changed**
割り込み要因を直接指示します。
VBUS 端子の状態が変化したときに “1” にセットされます。
USB_Status レジスタの VBUS ビットによって VBUS の状態を確認して下さい。VBUS が“0”であれば、ケーブルが抜かれたことを示します。このビットは SLEEP / SNOOZE 中も有効です。

Bit 6 :　**DescriptorCmp**
割り込み要因を直接指示します。
Descriptor 返信機能において、DescriptorSize レジスタの設定数のデータを返信し終えると、“1”にセットされます。
また、DescriptorSize レジスタの設定数まで送信する前にステータスステージへ移行（OUT トークンを受信）した場合には EP0IntStat レジスタの OUT_TranNAK ビットと共に、“1”にセットされます。

Bit 5 :　**SIE_IntStat**
割り込み要因を間接指示します。
SIE_IntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する SIE_IntEnb レジスタのビットがイネーブルにされているときに“1”にセットされます。このビットは、SLEEP / SNOOZE 中もリード有効です。

Bit 4 :　**BulkIntStat**
割り込み要因を間接指示します。
BulkIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する BulkIntEnb レジスタのビットがイネーブルにされているときに “1” にセットされます。

Bit 3 :　**RcvEP0SETUP**
割り込み要因を直接指示します。
コントロール転送のセットアップステージが終了し、受信したデータが EP0SETUP_0～EP0SETUP_7 レジスタに格納されると“1”にセットされます。同時に EP0ControlIN, EP0ControlOUT レジスタの ForceSTALL ビットが“0”に EP0ControlIN, EP0ControlOUT レジスタの ForceNAK ビット、ToggleStat ビット、SETUP_Control レジスタの ProtectEP0 ビットが“1”に、自動的に設定されます。
SetAddress()リクエストに対しては、AutoSetAddress機能が自動応答し、このステータスはセットされません。

Bit 2 :　**FIFO_IntStat**
割り込み要因を間接指示します。
FIFO_IntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する FIFO_IntEnb レジスタのビットがイネーブルにされているときに “1” にセットされます。

Bit 1 :　**EP0IntStat**
割り込み要因を間接指示します。
EP0IntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EP0IntEnb レジスタのビットがイネーブルにされているときに “1” にセットされます。

Bit 0 :　**EPrIntStat**
割り込み要因を間接指示します。
EPrIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPaIntEnb レジスタのビットがイネーブルにされているとき、“1”にセットされます。

EPr Interrupt Status
USB[0x081]　初期値 = 0x00

EP h IntStat(R) 7	EPg IntStat(R) 6	EPf IntStat(R) 5	EPe IntStat(R) 4	EPd IntStat(R) 3	EPc IntStat(R) 2	EPb IntStat(R) 1	EPa IntStat(R) 0

エンドポイント EPr の割り込みを表示します。

Bit 7 :　**EPhIntStat**
割り込み要因を間接指示します。
EPhIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPhIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 6 :　**EPgIntStat**
割り込み要因を間接指示します。
EPgIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPgIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 5 :　**EPfIntStat**
割り込み要因を間接指示します。
EPfIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPfIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 4 :　**EPeIntStat**
割り込み要因を間接指示します。
EPeIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPeIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 3 :　**EPdIntStat**
割り込み要因を間接指示します。
EPdIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPdIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 2 :　**EPcIntStat**
割り込み要因を間接指示します。
EPcIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPcIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 1 :　**EPbIntStat**
割り込み要因を間接指示します。
EPbIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPbIntEnb レジスタのビットがイネーブルにされているとき、“1” にセットされます。

Bit 0 :　**EPaIntStat**
割り込み要因を間接指示します。
EPaIntStat レジスタに割り込み要因があり、かつその割り込み要因に対応する EPaIntEnb レジスタのビットがイネーブルにされているとき、“1”にセットされます。

SIE Interrupt Status USB[0x082]　　初期値 = 0x00							
–	***NonJ***	RcvSOF	Detect Reset	Detect Suspend	Chirp Cmp	Restore Cmp	SetAddress Cmp
7	6	5	4	3	2	1	0

デバイス SIE 関連の割り込みを表示します。
全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit 7 :　**Reserved**

Bit 6 :　**NonJ**
割り込み要因を直接指示します。
USB バス上で J ステート以外の状態を検出すると“1”にセットされます。このビットは、本 LSI が SNOOZE 状態（PM_Control レジスタの InSnooze ビットが“1”）のとき、及び AutoNegotiation 機能使用時に USB_Control レジスタの InSUSPEND ビットが“1”にセットされているときに有効です。

Bit 5 :　**RcvSOF**
割り込み要因を直接指示します。
SOF トークンを受信すると“1”にセットされます。

Bit 4 :　**DetectReset**
割り込み要因を直接指示します。
USB のリセットステートを検出すると“1”にセットされます。このビットがセットされている間は USB のサスペンドステートの検出ができません（DetectSUSPEND がセットされません）。
このリセット検出は、NegoControl レジスタの ActiveUSB ビットが “1” にセットされているときに有効です。
“HS”動作モードの場合は、バス・アクティビティが一定時間無くなると、USB のリセット／サスペンド検出のために FS ターミネーションを自動的に設定し、SE0 が検出されるとリセットと判断して、このビットが“1”にセットされます。
AutoNegotiation 機能を使用しない場合には、このビットが“1”にセットされた場合、継続するリセットを誤検出しないよう、NegoControl レジスタの DisBusDetect ビットを“1”にセットして USB のリセット／サスペンドステートの検出を無効にして下さい。リセットに対する処理終了後に DisBusDetect ビットを“0”にクリアして USB のリセット／サスペンドステートの検出を有効にして下さい。
リセット検出時、NegoControl レジスタの GoChirp ビットにより、“HS Detection Handshake” を開始することができます。
AutoNegotiation 機能については、NegoControl レジスタの EnAutoNego ビットの項を参照して下さい。

Bit 3 :　**DetectSuspend**
割り込み要因を直接指示します。
USB のサスペンドステートを検出すると “1” にセットされます。このビットがセットされている間は USB のリセットステートの検出ができません（DetectRESET がセットされません）。
“HS” 動作モード の場合は、バス・アクティビティが一定時間無くなると、USB のリセット／サスペンド検出のために“FS”動作モードに自動的に設定されます。USB のサスペンドステートの検出後は、PM_Control0 レジスタの GoSnooze ビットを “1” にセットすることにより、本 LSI をスヌーズモード(内蔵 PLL の発振を停止)にすることが出来ます。

Bit 2 :　**ChirpCmp**
割り込み要因を直接指示します。
NegoControl レジスタの GoChirp ビットにより開始された“HS Detection Handshake”が完了すると “1” にセットされます。
割り込み発生後に USB_Status レジスタの FSxHS ビットをリードすることで、現在の動作モード(FS or HS)の判定をすることができます。

Bit 1 :　**RestoreCmp**
割り込み要因を直接指示します。
NegoControl レジスタの RestoreUSB ビットにより開始された Restore 処理が終了すると“1”にセットされます。このビットが“1” にセットされると動作モード（FS or HS）が Suspend する前の状態に戻ります。

Bit 0 :　**SetAddressCmp**
割り込み要因を直接指示します。
SetAddress()リクエストを受信すると、AutoSetAddress 機能(USB_Address レジスタ参照)が、そのコントロール転送の処理を自動的に行います。ステータスステージを行って SetAddress()リクエストに関わるコントロール転送が完了したときに、このステータスが“1”にセットされます。また、同時に USB_Address レジスタにアドレスがセットされます。

同期ビット（Bit5～0）は ACTIVE60 ステートのとき、読み出しはできますが、書き込み（割り込み要因クリア）できません。
ACT_DEVICE ステートを出る際には、これらの割り込みステータスにより割り込み信号 XINT がアサートされないよう、F/W にて以下の処理を行って下さい。
＜ACT_DEVICE ステートを出るとき＞
1)　割り込みステータスを処理し、クリアする（SIE_IntStat.Bit5～0）
2)　割り込みステータスをディスエーブルにする（SIE_IntEnb.Bit5～0）
＜ACT_DEVICE ステートに入るとき＞
1)　割り込みステータスをクリアする（SIE_IntStat.Bit5～0）
2)　割り込みステータスをイネーブルにする（SIE_IntEnb.Bit5～0）

FIFO Interrupt Status
USB[0x084]　　初期値 = 0x00

–	–	–	FIFO_ DMA0_Cmp	–	FIFO_ NotEmpty	FIFO_ Full	FIFO_ Empty
7	6	5	4	3	2	1	0

デバイス FIFO 関連の割り込みステータスを表示します。
全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit [7:5] :　**Reserved**

Bit 4 :　**FIFO_DMA0_Cmp**
割り込み要因を直接指示します。
DMA0 にジョインされているエンドポイントが IN 方向の場合、DMA0 の転送が終了した後に FIFO が空になると、“1”にセットされます。DMA0 にジョインされているエンドポイントが OUT 方向の場合、DMA0 の転送が終了すると、“1”にセットされます。

Bit 3 :　**Reserved**

Bit 2 :　**FIFO_NotEmpty**
割り込み要因を直接指示します。
EPx{x=0,a~h}Join.JoinFIFO_Stat ビットが“1”にセットされているときに、該当するエンドポイントの FIFO 領域が Empty でないときに、“1”にセットされます。

Bit 1 :　**FIFO_Full**
割り込み要因を直接指示します。
EPx{x=0,a~h}Join.JoinFIFO_Stat ビットが“1”にセットされているときに、該当するエンドポイントの FIFO 領域が Full になると、“1”にセットされます。

Bit 0 :　**FIFO_Empty**
割り込み要因を直接指示します。
EPx{x=0,a~h}Join.JoinFIFO_Stat ビットが“1”にセットされているときに、該当するエンドポイントの FIFO 領域が Empty になると、“1”にセットされます。

Bulk Interrupt Status
USB[0x085]　　初期値 = 0x00

CBW_ Cmp 7	CBW_ LengthErr 6	CBW_ Err 5	– 4	CSW_ Cmp 3	CSW_ Err 2	– 1	– 0

Bulk 転送機能関連の割り込みステータスを表示します。全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit 7 :　**CBW_Comp**
割り込み要因を直接指示します。
CBW の 31 バイトを正常に受信できたときに“1”にセットされます。

Bit 6 :　**CBW_LengthErr**
割り込み要因を直接指示します。
受信した CBW のパケット長が 31 バイト以外であったときに“1”にセットされます。

Bit 5 :　**CBW_Err**
割り込み要因を直接指示します。
受信した CBW に CRC エラー等のトランザクションエラーを検出したときに“1”にセットされます。

Bit 4 :　**Reserved**

Bit 3 :　**CSW_Cmp**
割り込み要因を直接指示します。
CSW の 13 バイトを正常に送信できたときに“1”にセットされます。

Bit 2 :　**CSW_Err**
割り込み要因を直接指示します。
CSW の送信にエラーがあったとき（ACK が返ってこなかったとき）に“1”にセットされます。

Bit [1:0] :　**Reserved**

EP0 Interrupt Status USB[0x087]　　初期値 = 0x00							
– 7	OUT_ ShortACK 6	IN_ TranACK 5	OUT_ TranACK 4	IN_ TranNAK 3	OUT_ TranNAK 2	IN_ TranErr 1	OUT_ TranErr 0

エンドポイント EP0 の割り込みステータスを表示します。全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit 7 :　**Reserved**

Bit 6 :　**OUT_ShortACK**
割り込み要因を直接指示します。
OUT トランザクションでショートパケットを受信し、ACK を返信したとき、 OUT_TranACK と同時に“1” にセットされます。

Bit 5 :　**IN_TranACK**
割り込み要因を直接指示します。
IN トランザクションで ACK を受信したとき、“1”にセットされます。

Bit 4 :　**OUT_TranACK**
割り込み要因を直接指示します。
OUT トランザクションで ACK を返信したとき、“1”にセットされます。

Bit 3 :　**IN_TranNAK**
割り込み要因を直接指示します。
IN トランザクションで NAK を返信したとき、“1”にセットされます。

Bit 2 :　**OUT_TranNAK**
割り込み要因を直接指示します。
OUT トランザクション及び PING トランザクションに対して NAK を返信したとき、“1”にセットされます。

Bit 1 :　**IN_TranErr**
割り込み要因を直接指示します。
IN トランザクションにおいて STALL を返した場合、パケットにエラーがあった場合、及びハンドシェイクがタイムアウトになった場合に、“1”にセットされます。

Bit 0 :　**OUT_TranErr**
割り込み要因を直接指示します。
OUT トランザクションにおいて STALL を返信した場合、及び、パケットにエラーがあった場合に、“1”にセットされます。

EPa Interrupt Status
USB[0x088]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPb Interrupt Status
USB[0x089]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPc Interrupt Status
USB[0x08A]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPd Interrupt Status
USB[0x08B]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPe Interrupt Stauts
USB[0x08C]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPf Interrupt Status
USB[0x08D]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPg Interrupt Status
USB[0x08E]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

EPh Interrupt Status
USB[0x08F]　　初期値 = 0x00

–	OUT_ ShortACK	IN_ TranACK	OUT_ TranACK	IN_ TranNAK	OUT_ TranNAK	IN_ TranErr	OUT_ TranErr
7	6	5	4	3	2	1	0

エンドポイント EPx｛a～h｝の割り込みステータスを表示します。全てのビットは“1”を書き込むことで割り込み要因をクリアできます。

Bit 7 :　**Reserved**

Bit 6 :　**OUT_ShortACK**
割り込み要因を直接指示します。
OUT トランザクションでショートパケットを受信し、ACK を返信したとき、 OUT_TranACK と同時に“1” にセットされます。

Bit 5 :　**IN_TranACK**
割り込み要因を直接指示します。
IN トランザクションで ACK を受信したとき、“1”にセットされます。

Bit 4 :　**OUT_TranACK**
割り込み要因を直接指示します。
OUT トランザクションで ACK を返信したとき、“1”にセットされます。

Bit 3 :　**IN_TranNAK**
割り込み要因を直接指示します。
IN トランザクションで NAK を返信したとき、“1”にセットされます。

Bit 2 :　**OUT_TranNAK**
割り込み要因を直接指示します。
OUT トランザクション及び PING トランザクションに対して NAK を返信したとき、“1”にセットされます。

Bit 1 :　**IN_TranErr**
割り込み要因を直接指示します。
IN トランザクションにおいて STALL を返した場合、パケットにエラーがあった場合、及びハンドシェイクがタイムアウトになった場合に、“1”にセットされます。

Bit 0 :　**OUT_TranErr**
割り込み要因を直接指示します。
OUT トランザクションにおいて STALL を返した場合、及び、パケットにエラーがあった場合に、“1”にセットされます。

Device Interrupt Enable
USB[0x090]　初期値 = 0x00

EnVBUS_ Changed	EnDescriptor Cmp	***EnSIE_ IntStat***	EnBulk IntStat	EnRcvEP0 SETUP	EnFIFO_ IntStat	EnEP0 IntStat	EnEPr IntStat
7	6	5	4	3	2	1	0

DeviceIntStat レジスタの割り込み要因による、MainIntStat レジスタの DeviceIntStat ビットのアサートを許可／禁止します。
EnVBUS_Changed, EnSIE_IntStat ビットは SLEEP / SNOOZE 中も有効です。

EPr Interrupt Enable
USB[0x091]　初期値 = 0x00

EnEPh IntStat	EnEPg IntStat	EnEPf IntStat	EnEPe IntStat	EnEPd IntStat	EnEPc IntStat	EnEPb IntStat	EnEPa IntStat
7	6	5	4	3	2	1	0

EPrIntStat レジスタの割り込み要因による、DeviceIntStat レジスタの EPrIntStat ビットのアサートを許可／禁止します。

SIE Interrupt Enable
USB[0x092]　　初期値 = 0x00

–	***EnNonJ***	EnRcvSOF	EnDetect RESET	EnDetect SUSPEND	EnChirp Cmp	EnRestore Cmp	EnSet AddressCmp
7	6	5	4	3	2	1	0

SIE_IntStat レジスタの割り込み要因による、DeviceIntStat レジスタの SIE_IntStat ビットのアサートを許可／禁止します。
EnNonJ ビットは SLEEP / SNOOZE 中も有効です。
同期ビット（Bit5～0）は ACTIVE60 ステートのとき、読み出しはできますが、書き込みできません。これら同期ビットの ACT_DEVICE ステートを出る際の処理に関しては、SIE_IntStat レジスタの説明をご参照下さい。

FIFO Interrupt Enable
USB[0x094]　　初期値 = 0x00

–	–	–	EnFIFO_ DMA0_Cmp	–	EnFIFO_ NotEmpty	EnFIFO_ Full	EnFIFO_ Empty
7	6	5	4	3	2	1	0

FIFO_IntStat レジスタの割り込み要因による、DeviceIntStat レジスタの FIFO_IntStat ビットのアサートを許可／禁止します。

Bulk Interrupt Enable
USB[0x095]　　初期値 = 0x00

EnCBW_ Cmp	EnCBW_ LengthErr	EnCBW_ Err	–	EnCSW_ Cmp	EnCSW_Err	–	–
7	6	5	4	3	2	1	0

BulkIntStat レジスタの割り込み要因による、DeviceIntStat レジスタの BulkIntStat ビットのアサートを許可／禁止します。

EP0 Interrupt Enable
USB[0x097]　　初期値 = 0x00

–	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EP0IntStat レジスタの割り込み要因による、DeviceIntStat レジスタの EP0IntStat ビットのアサートを許可／禁止します。

EPa Interrupt Enable
USB[0x098]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPb Interrupt Enable
USB[0x099]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPc Interrupt Enable
USB[0x09A]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPd Interrupt Enable
USB[0x09B]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPe Interrupt Enable
USB[0x09C]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPf Interrupt Enalbe
USB[0x09D]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPg Interrupt Enable
USB[0x09E]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPh Interrupt Enable
USB[0x09F]　　初期値 = 0x00

_	EnOUT_ ShortACK	EnIN_ TranACK	EnOUT_ TranACK	EnIN_ TranNAK	EnOUT_ TranNAK	EnIN_ TranErr	EnOUT_ TranErr
7	6	5	4	3	2	1	0

EPx{a-h}IntStat レジスタの割り込み要因による、EPrIntStat レジスタの EPx{a-h}IntStat ビットのアサートを許可／禁止します。

Reset DTM USB[0x0A0]　　初期値 = 0x01							
–	–	–	–	–	–	–	***ResetDTM***
7	6	5	4	3	2	1	0

デバイスのトランシーバマクロをリセットします。
SLEEP / SNOOZE 時でもアクセス有効です。

Bit [7:1] :　**Reserved**

Bit 0 :　**ResetDTM**
このビットに“1”をセットすると、本 LSI のデバイストランシーバマクロを初期化します。
リセットを解除するには、このビットを“0”にクリアして下さい。

Negotiation Control USB[0x0A2]　　初期値 = 0x00							
DisBus Detect	EnAuto Nego	In SUSPEND	Disable HS	Send Wakeup	Restore USB	GoChirp	Active USB
7	6	5	4	3	2	1	0

デバイスのネゴシエーション に関する動作設定を行います。

Bit 7 :　**DisBusDetect**
このビットを“1”にセットすると、USB のリセット／サスペンドステートの自動検出を無効にします。このビットが“0”にクリアされている場合、USB のリセット／サスペンドステートの検出のため、USB バス上のバス・アクティビティを監視します。
“HS”モード時は、バス・アクティビティが 3ms の期間検出されない場合、自動的に“FS”モードに切り替えたのち、USB のリセットあるいはサスペンドステートの判定を行い、その後該当する割り込み要因（DetectReset、DetectSuspend）をセットします。“FS”モード時はバス・アクティビティが 3ms の期間検出されないと USB のサスペンドステートと判定し、また、2.5μs 以上の“SE0”を検出するとリセットと判断し、該当する割り込み要因をセットします。
DetectReset、DetectSuspend のビットが“1”にセットされたら DisBusDetect ビットを“1”にセットして USB のリセット／サスペンドステートが継続している間、検出を無効にして下さい。
AutoNegotiation 機能を使用する場合、このビットに“1”をセットしないようにして下さい。

Bit 6 :　**EnAutoNego**
AutoNegotiation 機能を有効にします。AutoNegotiation 機能は、リセット検出時に、スピードネゴシエーションが終了してスピードモードが決定するまでのシーケンスを自動化します。
AutoNegotiation 機能の詳細は、動作説明の章を参照して下さい。

Bit 5 :　**InSUSPEND**
AutoNegotiation 機能使用時に、USB のサスペンドステートを検出すると自動的に“1”にセットされ NonJ ステートの検出機能を有効にします。USB のサスペンドステートから復帰する場合には、このビットを“0”にクリアして下さい。
AutoNegotiation 機能を使用する場合の説明は、“機能説明　オート・ネゴシエーション機能”をご参照下さい。

Bit 4 :　**DisableHS**
GoChirp が“1”にセットされたときに、このビットが“1”にセットされているときには、DeviceChirp を送出せずに強制的に FS モードとなり、ChirpCmp 割り込みを発生します。

Bit 3 :　**SendWakeup**
このビットを“1”にセットすると、USB ポートに RemoteWakeup 信号(K)を出力します。
RemoteWakeup 信号の送出開始から 1ms 以上 15ms 以内経過後、このビットを“0”にクリアして送出を停止して下さい。

Bit 2 :　**RestoreUSB**
USB のサスペンドステートからリジュームする際に、このビットを“1”にセットすると、USB のサスペンド前に保存された動作モード（FS or HS）に自動的に切り替えられ、該当する割り込み要因（RestoreCmp）がセットされます。
このビットは、動作終了後自動的に“0”にクリアされます。
AutoNegotiation 機能を使用する場合、このビットの機能は自動的に制御されますので、このビットをセット／クリアしないで下さい。

Bit 1 :　**GoChirp**
USB バスがリセット状態である場合に、このビットに“1”をセットすると、ホスト／ハブとの間で“HS Detection Handshake”を行い、XcvrControl レジスタの TermSelect ビット、XcvrSelect ビット及び USB_Status レジスタの FSxHS ビットが自動的に設定されます。動作終了と同時に割り込み要因（ChirpCmp）がセットされます。
このビットは､動作終了後自動的に“0”にクリアされます｡動作終了後 USBStauts レジスタの FSxHS ビットを参照することで、“HS Detection Handshake”の結果が確認できます。
AutoNegotiation 機能を使用する場合、このビットの機能は自動的に制御されますので、このビットをセット／クリアしないで下さい。

Bit 0 :　**ActiveUSB**
本 LSI では、このビットがハードリセット後“0”にクリアされているため､USB デバイスの全機能を停止しています。本 LSI の設定終了後に、本ビットを“1”にセットすることで、USB デバイスとしての動作が可能となります。

USB Status
USB[0x0A4]　初期値 = 0xXX

VBUS(R)	FSxHS	–	–	–	–	***LineState[1:0]*** (R)	
7	6	5	4	3	2	1	0

デバイスに関するステータスを表示します。

Bit 7 :　**VBUS**
VBUS 端子の状態が表示されます。このビットは SLEEP / SNOOZE 中でも有効です。

Bit 6 :　**FSxHS**
現在の動作モードを示します。NegoControl.GoChirp ビットにより“HS Detection Handshake”（機能説明参照）を実行すると、自動的に設定されます。このビットを書き込むことにより動作モードを強制的に変更することも可能ですが、シミュレーション等で“HS Detection Handshake”を行わずに動作モードを切り替えたい場合にのみ、このビットを操作してください。
ケーブルアタッチ時に、“FS（1）”にセットして下さい。
このビットは ACTIVE60 / ACT_DEVICE 時に読むことができ、ACT_DEVICE 時に書くことができます。

Bit [5:2] :　**Reserved**

Bit [1:0] :　**LineState [1:0]**
USB ケーブル上の信号状態を示します。このビットは SLEEP / SNOOZE 中でも有効です。
XcvrControl レジスタの TermSelect ビットが“1”であるとき（FS ターミネーション選択時）、XcvrSelect ビットが “1”（FS トランシーバ選択時）であれば､DP/DM の FS レシーバの受信値を、XcvrSelect が“0”（HS トランシーバ選択時）であれば、HS レシーバの受信値を示します。
TermSelect が “0” であるときは、USB のバス・アクティビティを示します。

LineState		
TermSelect	DP / DM	LineState [1:0]
0	Don't Care	バス・アクティビティ
1	SE0	0b00
1	J	0b01
1	K	0b10
1	SE1	0b11

Xcvr Control
USB[0x0A5]　　初期値 = 0x41

Term Select	XcvrSelect	–	–	–	–	OpMode[1:0]	
7	6	5	4	3	2	1	0

デバイスのトランシーバマクロに関する設定を行います。

Bit 7 :　**TermSelect**
FS または HS いずれかのターミネーションを選択して有効にします。USB_Control レジスタの GoChirp ビットによって"HS detection handshake"を実行した場合、または、D_NegoControl レジスタの EnAutoNego ビットがセットされ、AutoNegotiation 機能が実行された場合、このビットは自動的に設定されます。

Bit 6 :　**XcvrSelect**
FS または HS いずれかのトランシーバを選択して有効にします。D_NegoControl レジスタの GoChirp ビットによって"HS detection handshake"を実行した場合、または、D_NegoControl レジスタの EnAutoNego ビットがセットされ、AutoNegotiation 機能が実行された場合、このビットは自動的に設定されます。

Bit [5:2] :　**Reserved**

Bit [1:0] :　**OpMode**
UTM のオペレーションモードを設定します。
USB ケーブルが抜かれているとき（※）、USB のサスペンド状態になるとき、またはテストモード時以外には、通常設定する必要がありません。

OpMode		
00	"Normal Operation"	通常使用状態
01	"Non-Driving"	USB ケーブルが抜かれているときにはこの状態にして下さい。
10	"Disable Bitstuffing and NRZI encoding"	USB テストモード時にはこの状態にしてください。
11	"Power-Down"	USB のサスペンド時にはこの状態にしてください。

※USB ケーブルが抜けているときには、このレジスタを"41h"にセットすることを推奨します。

USB Test
USB[0x0A6]　　初期値 = 0x00

EnHS_ Test	–	–	–	Test_ SE0_NAK	Test_J	Test_K	Test_ Packet
7	6	5	4	3	2	1	0

デバイスの USB 2.0 のテストモードに関する動作設定を行います。SetFeature リクエストで指定されたテストモードに対応するビットを設定し、ステータスステージ終了後に EnHS_Test ビットに"1"をセットすることにより、USB2.0 の規格で定義されたテストモードの動作を行うようにして下さい。

Bit 7 :　**EnHS_Test**

このビットに“1”をセットすると、USB_Test レジスタの下位 4 ビットのいずれかのビットに“1”が設定されている場合、そのビットに対応するテストモードに入ります。テストモードを行う際には、NegoControl レジスタの DisBusDetect ビットを“1”にして USB のサスペンドとリセットの検出を行わないようにする必要があります。また、NegoControl レジスタの EnAutoNego ビットを“0”にクリアして、AutoNegotiation 機能を無効にして下さい。
また、テストモードへの移行は、SetFeature リクエストにおけるステータスステージの終了後に行うように、ご注意下さい。

Bit [6:4] :　**Reserved**

Bit 3 :　**Test_SE0_NAK**

このビットを“1”に設定し、EnHS_Test ビットに“1”をセットすることにより、Test_SE0_NAK テストモードに入ることができます。

Bit 2 :　**TEST_J**

このビットを“1”に設定し、EnHS_Test ビットに“1”をセットすることにより、Test_J テストモードに入ることができます。なお、このテストモードでは、EnHS_Test ビットを“1”にセットする前に、XcvrControl レジスタの、TermSelect 及び XcvrSelect をスピードに従って設定し、また、OpMode を“10”（Disable Bitstuffing and NRZI encoding）にセットして下さい。

Bit 1 :　**TEST_K**

このビットを“1”に設定し、EnHS_Test ビットに“1”をセットすることにより、Test_K テストモードに入ることができます。なお、このテストモードでは、EnHS_Test ビットを“1”にする前に、XcvrControl レジスタの、TermSelect 及び XcvrSelect をスピードに従って設定し、また、OpMode を“10”（Disable Bitstuffing and NRZI encoding）にセットして下さい。

Bit 0 :　**Test_Packet**

このビットを“1”に設定し、EnHS_Test ビットに “1” をセットすることにより、Test_Packet テストモードに入ることができます。
このテストモードは EP0 以外の任意のエンドポイントで使用できますので、下記の設定を行って下さい。

1) エンドポイント EPx{x=a~h}の MaxPacketSize を 64 以上、転送方向を IN に設定し、EndpointNumber を“0xF”に設定して、使用可能として下さい。また、エンドポイント EPx{x=a~h}の FIFO を 64Byte 以上、割り当てて下さい。
2) エンドポイントの設定を、上記 EPx{x=a~h}の設定と重複しないようにして下さい。
   または、EPx{x=a~h}Config.EnEndpoint ビットをクリアして下さい。
3) EPx{x=a~h}の FIFO をクリアし、下記のテストパケット用のデータをこの FIFO に書き込んで下さい。
   EPx{x=a~h}IntStat レジスタの IN_TranErr ビットを“0”にクリアして下さい。
4) Test Packet の送信完了の毎に、IN_TranErr ステータスが“1”にセットされます。
   パケット送信テストモード時に FIFO に書き込むデータは以下の 53 バイトです。
   00h、00h、00h、00h、00h、00h、00h、00h、
   00h、AAh、AAh、AAh、AAh、AAh、AAh、AAh、
   AAh、EEh、EEh、EEh、EEh、EEh、EEh、EEh、
   EEh、FEh、FFh、FFh、FFh、FFh、FFh、FFh、
   FFh、FFh、FFh、FFh、FFh、7Fh、BFh、DFh、
   EFh、F7h、FBh、FDh、FCh、7Eh、BFh、DFh、
   EFh、F7h、FBh、FDh、7Eh

テストパケット送出時に、SIE が PID と CRC を付加しますので、FIFO に書き込むデータは、USB 規格 Rev.2.0 に記載されているテストパケットデータのうち、DATA0 PID の次のデータから、CRC16 以外のデータまでとなります。

EPn Control USB[0x0A8]　　初期値 = 0xXX							
AllForce NAK(W) 7	EPrForce STALL(W) 6	AllFIFO_ Clr(W) 5	– 4	– 3	– 2	– 1	EP0FIFO_ Clr(W) 0

エンドポイントの動作設定を行います。ライトオンリーのレジスタです。

Bit 7 :　**AllForceNAK**
全てのエンドポイントの ForceNAK ビットを“1”にセットします。

Bit 6 :　**EPrForceSTALL**
エンドポイント EPx{x=a~h}の ForceSTALL ビットを“1”にセットします。

Bit 5 :　**AllFIFO_Clr**
全てのエンドポイントの FIFO がクリアされます。各エンドポイントの領域設定を行ったときは、設定終了後に一度必ずこのビットに“1”をセットして、全てのエンドポイントの FIFO をクリアして下さい。このビットは、FIFO クリア完了後自動的に“0”にクリアされます。
エンドポイントに DMAx{x=0}がジョインされ、かつ、該当する DMA が起動中(Running ビットが“1”の間)に、該当するエンドポイントのビットを“1”にセットしないで下さい。

Bit [4:1] :　**Reserved**

Bit 0 :　**EP0FIFO_Clr**
エンドポイント EP0 の FIFO をクリアします。
このビットは、“1”をセットされると FIFO をクリアする動作のみ行い、セットされた値は保持しません。
エンドポイント EP0 に DMAx{x=0}がジョインされ、かつ、該当する DMA が起動中(Running ビットが“1”の間)に、このビットを“1”にセットしないで下さい。

EPr FIFO Clr USB[0x0A9]　　初期値 = 0xXX							
EPh FIFO_Clr(W) 7	EPg FIFO_Clr(W) 6	EPf FIFO_Clr(W) 5	EPe FIFO_Clr(W) 4	EPd FIFO_Clr(W) 3	EPc FIFO_Clr(W) 2	EPb FIFO_Clr(W) 1	EPa FIFO_Clr(W) 0

該当するエンドポイントの FIFO をクリアします。ライトオンリーのレジスタです。
このレジスタの各ビットは、“1”をセットされると FIFO をクリアする動作のみ行い、セットされた値は保持しません。
エンドポイントに DMAx{x=0}がジョインされ、かつ、該当する DMA が起動中(Running ビットが“1”の間)に、該当するエンドポイントのビットを“1”にセットしないで下さい。

Clr All EPn Join USB[0x0AA]　　初期値 = 0xXX							
– 7	ClrJoin FIFO_Stat(W) 6	– 5	– 4	– 3	ClrJoin DMA0(W) 2	ClrJoin CPU_Rd(W) 1	ClrJoin CPU_Wr(W) 0

該当するポートと各エンドポイントの接続をクリアします。ライトオンリーのレジスタです。
このレジスタのビットは、接続クリア後、自動的に“0”にクリアされます。
エンドポイントがポートに接続（EPx{x=0,a~h}Join レジスタの該当するビットが“1”にセット）され、且つ各ポートの起動中に、このレジスタのビットを“1”にセットしないで下さい。誤動作の原因となります。

Bulk Only Control USB[0x0AC]　初期値 = 0x00							
AutoForce NAK_CBW 7	– 6	– 5	– 4	– 3	GoCBW_ Mode 2	GoCSW_ Mode 1	– 0

バルクオンリーサポート機能を制御します。

Bit 7 :　**AutoForceNAK_CBW**
このビットを“1”にセットすると、CBW サポートによって CBW の受信する OUT トランザクションが完了すると、該当するエンドポイントの ForceNAK ビットを“1”にセットします。

Bit [6:3] :　**Reserved**

Bit 2 :　**GoCBW_Mode**
このビットを“1”にセットすると、該当するエンドポイントで CBW サポートを実行します。CBW サポートを実行するエンドポイントについては、BulkOnlyConfig レジスタの項を参照して下さい。

Bit 1 :　**GoCSW_Mode**
このビットを“1”にセットすると、該当するエンドポイントで CSW サポートを実行します。CSW サポートを実行するエンドポイントについては、BulkOnlyConfig レジスタの項を参照して下さい。

Bit 0 :　**Reserved**

Bulk Only Configuration USB[0x0AD]　初期値 = 0x00							
EPh BulkOnly 7	EPg BulkOnly 6	EPf BulkOnly 5	EPe BulkOnly 4	EPd BulkOnly 3	EPc BulkOnly 2	EPb BulkOnly 1	EPa BulkOnly 0

バルクオンリーサポート機能を有効にします。

Bit [7:0] :　**EPx{a-h}BulkOnly**
このビットを“1”にセットすると、エンドポイント EPx{a-h}でバルクオンリーサポート機能が有効になります。バルクオンリーサポートが有効にされると、エンドポイント EPx{a-h}が OUT のエンドポイントである場合、BulkOnlyControl.GoCBW_Mode ビットをセットすることによって、CBW サポートを行います。また、エンドポイント EPx{a-h}が IN のエンドポイントである場合、BulkOnlyControl. GoCSW_Mode ビットをセットすることによって、CSW サポートを行います。
同時に 2 つ以上の OUT のエンドポイントでバルクオンリーサポート機能を有効にしないで下さい。

EP0 SETUP 0
USB[0x0B0]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 1
USB[0x0B1]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 2
USB[0x0B2]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 3
USB[0x0B3]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 4
USB[0x0B4]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 5
USB[0x0B5]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 6
USB[0x0B6]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

EP0 SETUP 7
USB[0x0B7]　　初期値 = 0x00

EP0 SETUP_0~7[7:0](R)							
7	6	5	4	3	2	1	0

エンドポイント EP0 のセットアップステージで受信した 8 バイトのデータが、EP0SETUP_0 から順に格納されます。

EP0SETUP_0

BmRequestType がセットされます。

EP0SETUP_1

BRequest がセットされます。

EP0SETUP_2

Wvalue の下位 8 ビットがセットされます。

EP0SETUP_3

Wvalue の上位 8 ビットがセットされます。

EP0SETUP_4

WIndex の下位 8 ビットがセットされます。

EP0SETUP_5

WIndex の上位 8 ビットがセットされます。

EP0SETUP_6

WLength の下位 8 ビットがセットされます。

EP0SETUP_7

WLength の上位 8 ビットがセットされます。

USB Address USB[0x0B8]　　初期値 = 0x00							
–	USB Address[6:0]						
7	6	5	4	3	2	1	0

AutoSetAddress 機能により、USB アドレスが設定されます。
SetAddress()リクエストを受信すると、AutoSetAddress 機能がそのコントロール転送を自動的に行います。AutoSetAddress 機能は、SetAddress()リクエストに関わるコントロール転送のステータスステージが完了し、USB_Address をセットした後に、SetAddressCmp ステータスを発行します。

Bit 7 :　　**Reserved**

Bit [6:0] :　　**USB_Address**
USB アドレスが設定されます。
AutoSetAddress 機能によって自動的に書き込まれます。
また、書き込みが可能ですが、SetAddress()リクエストを受信すると、再度自動的に書き換えます。

SETUP Control USB[0x0BA]　　初期値 = 0x00							
–	–	–	–	–	–	–	Protect EP0
7	6	5	4	3	2	1	0

コントロール転送関係の設定をします。

Bit [7:1] :　　**Reserved**

Bit 0 :　　**ProtectEP0**
コントロール転送のセットアップステージが終了し、受信したデータが EP0SETUP_0～EP0SETUP_7 レジスタに格納されると、“1”にセットされます。
同時に EP0ControlIN,EP0ControlOUT レジスタの ForceSTALL ビットが“0”に、ForceNAK ビットが“1”に、ToggleStat ビットが“1”に、自動的に設定されます。
ProtectEP0 ビットは SETUP トランザクションが行われるとセットされます。従って、SetAddress()リクエストに対してもセットされます。
このビットが“1”にセットされていると、EP0 の ForceNAK ビット、ForceSTALL ビットの設定変更ができません。

Frame Number L USB[0x0BE]　　初期値 = 0x00							
FrameNumber[7:0](R)							
7	6	5	4	3	2	1	0

Frame Number H USB[0x0BF]　　初期値 = 0x00							
FN_ Invalid(R)	–	–	–	–	FrameNumber[10:8] (R)		
7	6	5	4	3	2	1	0

SOF トークンを受信する毎に更新される、USB のフレームナンバーが表示されます。フレームナンバーを取得する場合は、FrameNumber_H と FrameNumber_L レジスタを対でアクセスする必要があります。その際に FrameNumber_H レジスタを先にアクセスして下さい。

Bit 7 :　**Fn_Invalid**
受信した SOF パケットにエラーが発生したときに、このビットが “1” にセットされます。

Bit [6:3] :　**Reserved**

Bit [2:0] :　**FrameNumber[10:8]**

Bit [7:0] :　**FrameNumber[7:0]**
受信した SOF パケットの FrameNumber が表示されます。

EP0 Max Size
USB[0x0C0]　初期値 = 0x40

–	EP0 Max Size[6:3]				–	–	–
7	6	5	4	3	2	1	0

エンドポイント EP0 の設定を行います。

Bit 7 :　**Reserved**

Bit [6:3] :　**EP0MaxSize[6:3]**
エンドポイント EP0 の MaxPacketSize を設定します。
このエンドポイントは、以下のサイズから任意のサイズを選択して使用可能です。
FS 時 : 8, 16, 32, 64 バイト
HS 時 : 64 バイト

Bit [2:0] :　**Reserved**

EP0 Control
USB[0x0C1]　初期値 = 0x00

INxOUT	–	–	–	–	–	–	Reply Descriptor
7	6	5	4	3	2	1	0

エンドポイント EP0 の設定を行います。

Bit 7 :　**INxOUT**
エンドポイント EP0 の転送方向を設定します。
セットアップステージで受信したリクエストを判断して、このビットに値を設定して下さい。
データステージがある場合は、このビットにデータステージにおける転送方向をセットして下さい。セットアップステージが完了することにより、EP0ControlIN 及び EP0ControlOUT レジスタの ForceNAK ビットがセットされるので、データステージ及びステータスステージの実行時にクリアして下さい。
データステージが終了したら、ステータスステージの方向に合わせて、このビットを設定しなおして下さい。データステージの転送方向が IN の場合は、ステータスステージは OUT 方向となりますので、このビットに“0”を設定して下さい。また、データステージの転送方向が OUT、またはデータステージがない場合は、ステータスステージは IN 方向となりますので、エンドポイント EP0 の FIFO をクリアして、このビットに“1”を設定して下さい。
このビットの設定値と異なる方向の IN または OUT トランザクションに対しては、NAK 応答します。但し、そのトランザクション方向に対応する EP0ControlIN または EP0ControlOUT レジスタの ForceSTALL ビットがセットされていると STALL 応答します。

Bit [6:1] :　**Reserved**

Bit 0 :　**ReplyDescriptor**
Descriptor 返信機能を実行します。
このビットが“1”にセットされると、エンドポイント EP0 の IN トランザクションに応答して、FIFO から Descriptor データを、MaxPacketSize 分返信します。Descriptor データは、DescAdrs_H,L レジスタの設定値のアドレスを先頭にする、DescSize_H,L レジスタの設定サイズのデータを指します。これらの設定値は、Descriptor 返信機能の実行中に更新されますので、ReplyDescriptor ビットをセットする毎に設定して下さい。
1 つのトランザクション毎に、DescAdrs_H,L レジスタは、送信したデータ数だけインクリメントされ、また、DescSize_H,L レジスタは、送信したデータ数だけデクリメントされます。
DescSize_H,L の設定数のデータを送信して終了した場合、及び、IN トランザクション以外のトランザクションが行われた場合には、Descriptor 返信機能は終了し、ReplyDescriptor ビットは“0”にクリアされ、FIFO_IntStat レジスタの DescriptorCmp ビットと EP0IntStat レジスタの IN_TranACK ビットに“1”がセットされます。
さらに詳細な説明は、動作説明の章を参照して下さい。

EP0 Control IN
USB[0x0C2]　　初期値 = 0x00

–	En ShortPkt	–	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

エンドポイント EP0 の IN トランザクションに関する動作設定及び状態表示を行います。

Bit 7 :　**Reserved**

Bit 6 :　**EnShortPkt**
このビットを“1”にセットすることで、エンドポイント EP0 の IN トランザクションに対して、MaxPacketSize に満たない FIFO 内のデータをショートパケットとして送信することができます。ショートパケットを送信した IN トランザクションが完了すると、自動的にこのビットが“0”にクリアされます。MaxPacketSize のパケットを送信した場合は、このビットはクリアされません。
FIFO 内にデータが無い場合にこのビットを“1”にセットすると、ホストからの IN トークンに対して Zero 長パケットを送信することができます。このビットをセットしてパケットを送信している最中に、該当 FIFO にデータを書き込むと、タイミングによりそのデータも含めて送信されることがあります。パケットの送信が終了し、このビットがクリアされるまで、FIFO へのデータ書き込みは行わないで下さい。

Bit 5 :　**Reserved**

Bit 4 :　**ToggleStat**
エンドポイント EP0 の、IN トランザクションのトグルシーケンスビットの状態を示します。

Bit 3 :　**ToggleSet**
エンドポイント EP0 の、IN トランザクションのトグルシーケンスビットを“1”にセットします。ToggleClr ビットと同時にセットした場合、ToggleClr ビットの機能が優先されます。

Bit 2 :　**ToggleClr**
エンドポイント EP0 の、IN トランザクションのトグルシーケンスビットを“0”にクリアします。ToggleSet ビットと同時にセットした場合、このビットの機能が優先されます。

Bit 1 :　**ForceNAK**
このビットを“1”にセットすると、FIFO のデータ数に関わらずエンドポイント EP0 の IN トランザクションに対して NAK 応答します。
セットアップステージが完了することによって MainIntStat レジスタの RcvEP0SETUP ビットに“1”がセットされると、このビットは“1”にセットされ、RcvEP0SETUP ビットが“1”である間、このビットは“0”にクリアできません。また、ショートパケットを送信した IN トランザクションが完了したとき、このビットは“1”にセットされます。

このビットを“1”にセットする際に、既にトランザクションが実行中である場合においては、そのトランザクションが終了するまでビットはセットされず、終了と同時にこのビットは“1”にセットされます。トランザクションが実行中で無い場合においては、即座に“1”にセットされます。

Bit 0 :　**ForceSTALL**
このビットを“1”にセットすると、エンドポイント EP0 の IN トランザクションに対して STALL 応答します。このビットは、ForceNAK ビットの設定より優先されます。
セットアップステージが完了することによって、DeviceIntStat レジスタの RcvEP0SETUP ビットに“1”がセットされると、このビットは“0”にクリアされ、RcvEP0SETUP ビットが“1”である間はこのビットを“1”にセットできません。
現在実行中のトランザクションがある場合、トランザクション開始から一定時間後のこのビットの設定は、次のトランザクションから有効になります。

EP0 Control OUT
USB[0x0C3]　　初期値 = 0x00

Auto ForceNAK	–	–	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

エンドポイント EP0 の OUT トランザクションに関する動作設定及び状態表示を行います。

Bit 7 :　**AutoForceNAK**
エンドポイント EP0 の OUT トランザクションが正常に完結すると、このレジスタの ForceNAK ビットを“1”にセットします。

Bit [6:5] :　**Reserved**

Bit 4 :　**ToggleStat**
エンドポイント EP0 の、OUT トランザクションのトグルシーケンスビットの状態を示します。

Bit 3 :　**ToggleSet**
エンドポイント EP0 の、OUT トランザクションのトグルシーケンスビットを “1” にセットします。ToggleClr ビットと同時にセットした場合、ToggleClr ビットの機能が優先されます。

Bit 2 :　**ToggleClr**
エンドポイント EP0 の、OUT トランザクションのトグルシーケンスビットを “0” にクリアします。ToggleSet ビットと同時にセットした場合、このビットの機能が優先されます。

Bit 1 :　**ForceNAK**
このビットを“1”にセットすると、FIFO の空き容量に関わらずエンドポイント EP0 の OUT トランザクションに対して NAK 応答します。
セットアップステージが完了することによって DeviceIntStat レジスタの RcvEP0SETUP ビットに“1”がセットされると、このビットは“1”にセットされ、RcvEP0SETUP ビットが“1”である間はこのビットを“0”にクリアすることはできません。
このビットを“1”にセットする際に、既にトランザクションが実行中である場合においては、そのトランザクションが終了するまでビットはセットされず、終了と同時にこのビットは“1”にセットされます。トランザクションが実行中で無い場合においては、即座に“1”にセットされます。

Bit 0 :　**ForceSTALL**
このビットを“1”にセットすると、エンドポイント EP0 の OUT トランザクションに対して STALL 応答します。このビットは、ForceNAK ビットの設定より優先されます。
セットアップステージが完了することによって、DeviceIntStat レジスタの RcvEP0SETUP ビットに“1”がセットされると、このビットは“0”にクリアされ、RcvEP0SETUP ビットが“1”である間はこのビットを“1”にセットすることはできません。
現在実行中のトランザクションがある場合、トランザクション開始から一定時間後のこのビットの設定は、次のトランザクションから有効になります。

EP0 Join USB[0x0C5]　　初期値 = 0x00							
– 7	Join FIFO_Stat 6	– 5	– 4	– 3	Join DMA0 2	Join CPU_Rd 1	Join CPU_Wr 0

エンドポイント 0 とデータ転送を行うポートを指定します。

Bit 7 :　**Reserved**

Bit 6 :　**JoinFIFO_Stat**
エンドポイント EP0 の FIFO の Full 及び Empty の状態を、FIFO_IntStat.FIFO_Full、FIFO_IntStat.FIFO_Empty 及び FIFO_IntStat.FIFO_NotEmpty でモニタできるようにします。

Bit [5:3] :　**Reserved**

Bit 2 :　**JoinDMA0**
エンドポイント EP0 の FIFO で DMA0 の転送を行います。転送の方向は、DMA0_Control.Dir ビットの設定によります。

Bit 1 :　**JoinCPU_Rd**
エンドポイント EP0 の FIFO で CPU レジスタアクセスのリード転送を行います。即ち FIFO_Rd_H,L レジスタ、または、FIFO_ByteRd レジスタのリードが行われると、このエンドポイントの FIFO からデータが読み出されます。

Bit 0 :　**JoinCPU_Wr**
エンドポイント EP0 の FIFO で CPU レジスタアクセスのライト転送を行います。即ち FIFO_Wr_H,L レジスタへのライトが行われると、このエンドポイントの FIFO にデータが書き込まれます。

JoinDMAx{x=0}ビットを設定した場合は、DMAx_Control.Dir ビットが 1 のときは残りデータ数、0 のときは空き容量が、DMAx{x=0}_Remain_H,L レジスタにより、それぞれ参照できます。
JoinCPU_Rd, JoinCPU_Wr ビットを設定した場合は、FIFO_RdRemain_H,L、FIFO_WrRemain_H,L を参照し、FIFO_Rd_H,L、FIFO_ByteRd、FIFO_Wr_H,L レジスタからデータを読み出し、または書き込みできます。
JoinDMA0 ビット、JoinCPU_Rd ビット、JoinCPU_Wr ビットは、同時に 1 ビットのみを“1”にセットすることが可能です。複数のビットに対して、同時に“1”を書きこんだ場合、上位ビットが有効とされます。

EPa Max Size L
USB[0x0D0]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPb Max Size L
USB[0x0D8]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPc Max Size L
USB[0x0E0]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPd Max Size L
USB[0x0E8]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPe Max Size L
USB[0x0F0]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPf Max Size L
USB[0x0F8]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPg Max Size L
USB[0x100]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPh Max Size L
USB[0x108]　　初期値 = 0x00

MaxSize[7:3]					–	–	–
7	6	5	4	3	2	1	0

EPa Max Size H
USB[0x0D1]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPb Max Size H
USB[0x0D9]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPc Max Size H
USB[0x0E1]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPd Max Size H
USB[0x0E9]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPe Max Size H
USB[0x0F1]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPf Max Size H
USB[0x0F9]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPg Max Size H
USB[0x101]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

EPh Max Size H
USB[0x109]　　初期値 = 0x00

–	–	–	–	–	–	MaxSize[9:8]	
7	6	5	4	3	2	1	0

MaxPacketSize を設定します。

Bit [7:3] :　**MaxSize[7:3]**

Bit [2:0] :　**Reserved**

Bit [7:2] :　**Reserved**

Bit [1:0] :　**MaxSize[9:8]**
エンドポイント EPx{a-h}の MaxPacketSize を設定します。
このエンドポイントをバルク転送用として使用する場合には、
　FS 時 : 8, 16, 32, 64 バイト
　HS 時 : 512 バイト
のいずれかに設定して下さい。
このエンドポイントをインタラプト転送用として使用する場合は、
　FS 時 : 64 バイトまで
　HS 時 : 512 バイトまで
の任意の転送数が設定可能です。

EPa Configuration
USB[0x0D2]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPb Configuration
USB[0x0DA]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPc Configuration
USB[0x0E2]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPd Configuration
USB[0x0EA]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPe Configuration
USB[0x0F2]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPf Configuration
USB[0x0FA]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPg Configuration
USB[0x102]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

EPh Configuration
USB[0x10A]　　初期値 = 0x00

INxOUT	IntEP_Mode	EnEndpoint	–	EndpointNumber[3:0]			
7	6	5	4	3	2	1	0

エンドポイント EPx{a-h}の設定を行います。
EndpointNumber と INxOUT の組み合わせが、他のエンドポイントと重複しないように設定して下さい。

Bit 7 :　**INxOUT**
エンドポイントの転送方向を設定します。

Bit 6 :　**IntEP_Mode**
Interrupt 転送に関する設定を行います。
Bulk のエンドポイントでは、このビットに“1”を設定しないで下さい。
このビットの設定は、エンドポイントの方向（IN/OUT）によって異なります（エンドポイントの方向は Bit7“INxOUT”によって設定されます）。
IN 方向（INxOUT = 1）の場合、トグルシーケンスビットの動作モードを設定します。トグルシーケンスの動作モードは、アプリケーションに依存します。Interrupt IN のエンドポイントに対し、どちらかの動作モードを選択して下さい。
0: Normal toggle － 通常のトグルシーケンスを行います。
1: Always toggle － トランザクション毎に常にトグルします。このモードについては、USB2.0 規格書 5.7.5 項をご参照下さい。
OUT 方向(INxOUT = 0)の場合、このエンドポイントにおいて PING フローコントロールを行うか否かを設定します。Interrupt OUT のエンドポイントでは、このビットを“1”にセットして下さい。
0: Bulk OUT － Bulk OUT のエンドポイントはこの設定にして下さい。
1: Interrupt OUT － Interrupt OUT のエンドポイントはこの設定にして下さい。

Bit 5 :　**EnEndpoint**
このビットを“1”にセットすることで、このエンドポイントを有効にします。
このビットが“0”のときは、エンドポイントへのアクセスを無視します。
ホストからの SetConfiguration リクエストに従って設定して下さい。

Bit 4 :　**Reserved**

Bit [3:0] :　**EndpointNumber[3:0]**
0x1～0xF の任意のエンドポイントナンバーを設定します。

EPa Control
USB[0x0D4]　初期値 = 0x00

Auto ForceNAK 7	EnShort Pkt 6	DisAF_ NAK_Short 5	Toggle Stat(R) 4	Toggle Set(W) 3	Toggle Clr(W) 2	Force NAK 1	Force STALL 0

EPb Control
USB[0x0DC]　初期値 = 0x00

Auto ForceNAK 7	EnShort Pkt 6	DisAF_ NAK_Short 5	Toggle Stat(R) 4	Toggle Set(W) 3	Toggle Clr(W) 2	Force NAK 1	Force STALL 0

EPc Control
USB[0x0E4]　初期値 = 0x00

Auto ForceNAK 7	EnShort Pkt 6	DisAF_ NAK_Short 5	Toggle Stat(R) 4	Toggle Set(W) 3	Toggle Clr(W) 2	Force NAK 1	Force STALL 0

EPd Control
USB[0x0EC]　初期値 = 0x00

Auto ForceNAK 7	EnShort Pkt 6	DisAF_ NAK_Short 5	Toggle Stat(R) 4	Toggle Set(W) 3	Toggle Clr(W) 2	Force NAK 1	Force STALL 0

EPe Control							
USB[0x0F4]　初期値 = 0x00							
Auto ForceNAK	EnShort Pkt	DisAF_ NAK_Short	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

EPf Control							
USB[0x0FC]　初期値 = 0x00							
Auto ForceNAK	EnShort Pkt	DisAF_ NAK_Short	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

EPg Control							
USB[0x104]　初期値 = 0x00							
Auto ForceNAK	EnShort Pkt	DisAF_ NAK_Short	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

EPh Control							
USB[0x10C]　初期値 = 0x00							
Auto ForceNAK	EnShort Pkt	DisAF_ NAK_Short	Toggle Stat(R)	Toggle Set(W)	Toggle Clr(W)	Force NAK	Force STALL
7	6	5	4	3	2	1	0

エンドポイント EPx{a-h}の動作設定を行います。

Bit 7 :　**AutoForceNAK**

エンドポイント EP x{a-h}のトランザクションが正常に完結すると、このレジスタの ForceNAK ビットを“1”にセットします。

Bit 6 :　**EnShortPkt**

このビットを“1”にセットすることで､エンドポイント EP x{a-h}の IN トランザクションに対して、MaxPacketSize に満たない FIFO 内のデータをショートパケットとして送信することができます。ショートパケットを送信した IN トランザクションが完了すると、自動的にこのビットが“0”にクリアされます。マックスパケットサイズのパケットを送信した場合は、このビットはクリアされません。

FIFO 内にデータが無い場合にこのビットを“1”にセットすると、ホストからの IN トークンに対して Zero 長パケットを送信することができます。このビットをセットしてパケットを送信している最中に、該当 FIFO にデータを書き込むと、タイミングによりそのデータも含めて送信されることがあります。パケットの送信が終了し、このビットがクリアされるまで、FIFO へのデータ書き込みは行わないで下さい。

Bit 5 :　**DisAF_NAK_Short**

Auto Force NAK Short（以下、AF_NAK_Short※）機能の有効／無効を設定します。

※正常な OUT トランザクション完結時に受信したパケットがショートパケットの場合､自動的に ForceNAK ビットを“1”にセットする。

デフォルトの設定は AF_NAK_Short 機能が有効です。

このビットを“1”にセットすると、AF_NAK_Short 機能が無効になります。

AutoForceNAK ビットが“1”にセットされている場合は、AutoForceNAK ビットが優先されます。

Bit 4 :　**ToggleStat**

エンドポイント EP x{a-h}のトグルシーケンスビットの状態を示します。

Bit 3 :　**ToggleSet**

エンドポイント EP x{a-h}トグルシーケンスビットを“1”にセットします。ToggleClr ビットと同時にセットした場合、ToggleClr ビットの機能が優先されます。

Bit 2 :　**ToggleClr**
エンドポイント EP x{a-h}のトグルシーケンスビットを“0”にクリアします。ToggleSet ビットと同時にセットした場合、このビットの機能が優先されます。

Bit 1 :　**ForceNAK**
このビットを“1”にセットすると、FIFO のデータ数または空き容量に関わらずエンドポイント EP x{a-h}のトランザクションに対して NAK 応答します。
このビットを“1”にセットする際に、既にトランザクションが実行中である場合においては、そのトランザクションが終了するまでビットはセットされず、終了と同時にこのビットは“1”にセットされます。トランザクションが実行中で無い場合においては、即座に“1”にセットされます。

Bit 0 :　**ForceSTALL**
このビットを“1”にセットすると、エンドポイント EP x{a-h}のトランザクションに対して STALL 応答します。このビットは、ForceNAK ビットの設定より優先されます。
現在実行中のトランザクションがある場合、トランザクション開始から一定時間後のこのビットの設定は、次のトランザクションから有効になります。

EPa Join
USB[0x0D5]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPb Join
USB[0x0DD]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPc Join
USB[0x0E5]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPd Join
USB[0x0ED]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPe Join
USB[0x0F5]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPf Join
USB[0x0FD]　初期値 = 0x00

–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPg Join USB[0x105]　　初期値 = 0x00							
–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

EPh Join USB[0x10D]　　初期値 = 0x00							
–	Join FIFO_Stat	–	–	–	Join DMA0	Join CPU_Rd	Join CPU_Wr
7	6	5	4	3	2	1	0

エンドポイント EPx{a-h}とデータ転送を行うポートを指定します。

Bit 7 :　**Reserved**

Bit 6 :　**JoinFIFO_Stat**
エンドポイント EP x{a-h}の FIFO の Full 及び Empty の状態を、FIFO_IntStat.FIFO_Full、FIFO_IntStat.FIFO_Empty 及び FIFO_IntStat.FIFO_NotEmpty でモニタできるようにします。

Bit [5:3] :　**Reserved**

Bit 2 :　**JoinDMA0**
エンドポイント EP x{a-h}の FIFO で DMA0 の転送を行います。転送の方向は、DMA0_Control.Dir ビットの設定によります。

Bit 1 :　**JoinCPU_Rd**
エンドポイント EP x{a-h}の FIFO で CPU レジスタアクセスのリード転送を行います。即ち FIFO_Rd_H,L レジスタ、または、FIFO_ByteRd レジスタのリードが行われると、このエンドポイントの FIFO からデータが読み出されます。

Bit 0 :　**JoinCPU_Wr**
エンドポイント EP x{a-h}の FIFO で CPU レジスタアクセスのライト転送を行います。即ち FIFO_Wr_H,L レジスタへのライトが行われると、このエンドポイントの FIFO にデータが書き込まれます。

JoinDMAx{x=0}ビットを設定した場合は、DMAx_Control.Dir ビットが 1 のときは残りデータ数、0 のときは空き容量が、DMAx{x=0}_Remain_H,L レジスタにより、それぞれ参照できます。
JoinCPU_Rd, JoinCPU_Wr ビットを設定した場合は、FIFO_RdRemain_H,L、FIFO_WrRemain_H,L を参照し、FIFO_Rd_H,L、EPnCHnFIFO_ByteRd、FIFO_Wr_H,L レジスタからデータを読み出し、または書き込みできます。
JoinDMA0 ビット、JoinCPU_Rd ビット、JoinCPU_Wr ビットは、同時に 1 ビットのみを“1”にセットすることが可能です。複数のビットに対して、同時に“1”を書きこんだ場合、上位ビットが有効とされます。

A2.5.2.7　USB FIFO 設定レジスタ

EPa Start Address L
USB[0x110]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPb Start Address L
USB[0x112]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPc Start Address L
USB[0x114]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPd Start Address L
USB[0x116]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPe Start Address L
USB[0x118]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPf Start Address L
USB[0x11A]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPg Start Address L
USB[0x11C]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPh Start Address L
USB[0x11E]　　初期値 = 0x00

Start Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EPa Start Address H
USB[0x111]　　初期値 = 0x00

—	—	—	Start Adrs[12:8]				
7	6	5	4	3	2	1	0

EPb Start Address H
USB[0x113]　　初期値 = 0x00

—	—	—	Start Adrs[12:8]				
7	6	5	4	3	2	1	0

EPc Start Address H
USB[0x115]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPd Start Address H
USB[0x117]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPe Start Address H
USB[0x119]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPf Start Address H
USB[0x11B]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPg Start Address H
USB[0x11D]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPh Start Address H
USB[0x11F]　　初期値 = 0x00

7	6	5	4	3	2	1	0
–	–	–	Start Adrs[12:8]				

EPx{a-h}で使用する FIFO の領域設定を行います。

Bit [7:2] :　**StartAdrs[7:2]**

Bit [1:0] :　**Reserved**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**StartAdrs[12:8]**
エンドポイント EP x{a-h}に割り当てる FIFO の先頭アドレスを設定します。
アドレス値は、上位 12 ビット～2 ビットでの設定のため 4 バイト単位での指定になります。
チャネル EP x{a-g} に割り当てられる領域は、次の EP の先頭アドレスに設定されたアドレスの 1 バイト前までとなります。EPh に割り当てられる領域は、EP_EndAdrs に設定されたアドレスの 1 バイト前までとなります。
EP x{a-h} StartAdrs を設定した後は､必ず EPrFIFO_Clr レジスタの EP x{a-h}FIFO_Clr ビットを“1”にしてエンドポイント EP x{a-h}の FIFO をクリアして下さい。
なお、ここで設定した領域より、EP x{a-h} の MaxSize が大きい場合には、正常に動作しません。
全エンドポイントで確保する FIFO 領域、EP0 エリア、デスクリプタエリア、CBW エリア、CSW エリアの合計が内蔵の RAM の合計を超えないように設定して下さい。

EP End Address L
USB[0x120]　　初期値 = 0x00

End Adrs[7:2]						—	—
7	6	5	4	3	2	1	0

EP End Address H
USB[0x121]　　初期値 = 0x12

—	—	—	End Adrs[12:8]				
7	6	5	4	3	2	1	0

エンドポイント FIFO の領域設定を行います。

Bit [7:2] :　**EndAdrs[7:2]**

Bit [1:0] :　**Reserved**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**EndAdrs[12:8]**
エンドポイント EPh に割り当てる FIFO の終端アドレスを設定します。
アドレス値は、上位 12 ビット～2 ビットでの設定のため 4 バイト単位での指定になります。
リセット時の初期値は搭載している FIFO RAM の最終アドレスが設定されています。
ここで設定した領域より、エンドポイントの MaxSize が大きい場合には、正常に動作しません。
全エンドポイントで確保する FIFO 領域、EP0 エリア、デスクリプタエリア、CBW エリア、CSW エリアの合計が内蔵の RAM の合計を超えないように設定して下さい。
エンドポイント数の変化による本レジスタの割り付けアドレスの変更はありません。
リセット時の初期値は FIFO 領域の大きさを示すアドレス値になりますが値の変更は可能です。

Descriptor Address L
USB[0x124]　　初期値 = 0x00

Desc Adrs[7:0]							
7	6	5	4	3	2	1	0

Descriptor Address H
USB[0x125]　　初期値 = 0x00

—	—	—	—	Desc Adrs[11:8]			
7	6	5	4	3	2	1	0

Descriptor Adress を指定します。

Bit [7:0] :　**DescAdrs[7:0]**

Bit [7:4] :　**Reserved**

Bit [3:0] :　**DescAdrs[11:8]**
Descriptor 返信機能における、Descriptor 返信動作開始時の FIFO の先頭アドレスを指定します。
Descriptor Address は、Descriptor 返信機能に対して FIFO 領域を割り当てるものではありません。
Descriptor Address は、FIFO RAM の全領域を指定することが出来ます。
Descriptor 返信時には、エンドポイント EP0 における IN トランザクション完了毎に、送信データ数の分だけ DescAdrs は更新されます。Descriptor 返信機能については、EP0Control レジスタの ReplyDescriptor の項を参照して下さい。
Descriptor 返信機能用の FIFO 領域は、明示的には割り当てませんので、DescAdrs_H,L レジスタと DescSize_H,L レジスタの指定によって、他のエンドポイントの FIFO との重複を避けて下さい。
エンドポイント EP0 の予約された領域の終了アドレス(0x040)から CBW 領域の先頭アドレス(0x190)までの間が適切です。
Desriptor Address を参照する場合は、DescAdrs_H,DescAdrs_L の順に読み出して下さい。

Descriptor Size L							
USB[0x126]　初期値 = 0x00							
Desc Size[7:0]							
7	6	5	4	3	2	1	0

Descriptor Size H							
USB[0x127]　初期値 = 0x00							
–	–	–	Desc Size[12:8]				
7	6	5	4	3	2	1	0

Descriptor Size を指定します。

Bit [7:0] :　**DescSize[7:0]**

Bit [7:5] :　**Reserved**

Bit [1:0] :　**DescSize[12:8]**

Descriptor Size には、Descriptor 返信機能において、返信する総データ数を指定します。Descriptor 返信機能については、EP0Control レジスタの ReplyDescriptor ビットの項を参照して下さい。
Descriptor Size には、FIFO のサイズ及び領域設定に関わらず、0x000 から 0x1FFF までの値を指定することが出来ます。Descriptor 返信時には、エンドポイント EP0 における IN トランザクション完了毎に、送信データ数の分だけ DescSize は更新されます。
Descriptor 返信機能用の FIFO 領域は、明示的には割り当てませんので、DescAdrs_H,L レジスタと DescSize_H,L レジスタの指定によって、他のエンドポイントの FIFO との重複を避けて下さい。
エンドポイント EP0 の予約された領域の終了アドレス(0x040)から CBW 領域の先頭アドレス(0x190)までの間を使用するようにして下さい。
Desriptor Size を参照する場合は、DescSize_H,DescSize_L の順に読み出して下さい。

DMA0 FIFO Control							
USB[0x128]　初期値 = 0x00							
FIFO_ Running(R)	AutoEn Short	–	–	–	–	–	–
7	6	5	4	3	2	1	0

DMA0 転送時の、FIFO の状態の表示及び設定を行います。

Bit 7 :　**FIFO_Running**

DMA0 に接続されたエンドポイントの FIFO が動作中であることを示します。DMA0 を起動すると“1”にセットされ、DMA0 が終了した後、FIFO が空になると“0”にクリアされます。

Bit 6 :　**AutoEnShort**

DMA0 の終了時に、マックスパケットサイズに満たないデータ数が FIFO に残る場合に、そのエンドポイントの EnShortPkt ビットを“1”にセットします。
DMA0 に接続されたエンドポイントが IN 方向である場合に有効です。

Bit [5:0] :　**Reserved**

FIFO Rd 0
USB[0x130]　　初期値 = 0xXX

FIFO Rd0[7:0](R)							
7	6	5	4	3	2	1	0

FIFO Rd 1
USB[0x131]　　初期値 = 0xXX

FIFO Rd1[7:0] (R)							
7	6	5	4	3	2	1	0

Bit [7:0] :　　**FIFO_Rd0[7:0]**

Bit [7:0] :　　**FIFO_Rd1[7:0]**
EPx{x=0,a~h}Join.JoinCPU_Rd ビットがセットされているエンドポイントの FIFO データを読み出せます。
8bit mode 時には FIFO_Rd_H,L いずれのレジスタにアクセスしても FIFO のデータを読み出せます。
16bit mode 時の FIFO にバイト境界がある場合に、このレジスタを読み出した場合は片側にのみ有効なデータが出力されます。詳細は機能説明“FIFO アクセスの端数処理”をご参照ください。
このレジスタを用いて FIFO のデータを読み出す場合は、必ず FIFO_RdRemain_H,L レジスタにより読み出し可能データ数を確認した後、リードを行ってください。

FIFO Wr 0
USB[0x132]　　初期値 = 0xXX

FIFO Wr0[7:0](W)							
7	6	5	4	3	2	1	0

FIFO Wr 1
USB[0x133]　　初期値 = 0xXX

FIFO Wr1[7:0] (W)							
7	6	5	4	3	2	1	0

Bit [7:0] :　　**FIFO_Wr0[7:0]**

Bit [7:0] :　　**FIFO_Wr1[7:0]**
EPx{x=0,a~h}Join.JoinCPU_Wr ビットがセットされているエンドポイントの FIFO へデータを書き込めます。
8bit mode 時には FIFO_Wr_H,L いずれのレジスタにアクセスしても FIFO へデータを書き込めます。
16bit mode 時の FIFO にバイト境界がある場合に、このレジスタへ書き込んだ場合は片側のみ書き込みが行われます。詳細は機能説明“FIFO アクセスの端数処理”をご参照ください。
このレジスタを用いて FIFO へデータを書き込む場合は、必ず FIFO_WrRemain_H,L レジスタにより、書き込み可能データ数を確認した後、ライトを行ってください。

FIFO Rd Remain L
USB[0x134]　　初期値 = 0x00

RdRemain[7:0](R)							
7	6	5	4	3	2	1	0

FIFO Rd Remain H
USB[0x135]　　初期値 = 0x00

RdRemain Valid(R)	–	–	RdRemain[12:8] (R)				
7	6	5	4	3	2	1	0

Bit [7:0] :　　**RdRemain[7:0]**

Bit 7 :　**RdRemainValid**

EPx{x=0,a~h}Join.JoinCPU_Rd ビットによって、CPU I/F にエンドポイントがジョインされかつ、FIFO_RdRemain の値が有効な場合に“1”にセットされます。このビットがクリアされている場合の RdRemain の値は無効です。

Bit [6:5] :　**Reserved**

Bit [4:0] :　**RdRemain[12:8]**

EPx{x=0,a~h}Join.JoinCPU_Rd ビットによって CPU I/F に接続しているエンドポイントの FIFO 内のリード可能なデータ数を示します。FIFO のリード可能なデータ数を取得する場合は、FIFO_RdRemain_H レジスタと FIFO_RdRemain_L レジスタを対でアクセスする必要があります。その際に、FIFO_RdRemain_H レジスタを先にアクセスして下さい。

FIFO Wr Remain L
USB[0x136]　初期値 = 0x00

WrRemain[7:0] (R)							
7	6	5	4	3	2	1	0

FIFO Wr Remain H
USB[0x137]　初期値 = 0x00

–	–	–	WrRemain[12:8] (R)				
7	6	5	4	3	2	1	0

Bit [7:0] :　**WrRemain[7:0]**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**WrRemain[12:8]**

EPx{x=0,a~h}Join.JoinCPU_Wr ビットにより CPU I/F に接続しているエンドポイント FIFO により CPU I/F に接続しているチャネル FIFO の空き容量を示します。ただし、FIFO への書き込み直後には正確な FIFO の空き容量を確認することは出来ません。1CPU サイクル以上の間隔を開けて FIFO の空き容量を確認して下さい。FIFO の空き容量を取得する場合は、FIFO_WrRemain_H レジスタと FIFO_WrRemain_L レジスタを対でアクセスする必要があります。その際に、FIFO_WrRemain_H レジスタを先にアクセスして下さい。

FIFO Byte Rd
USB[0x138]　初期値 = 0xXX

FIFO ByteRd[7:0] (R)							
7	6	5	4	3	2	1	0

Bit [7:0] :　**FIFO_ByteRd[7:0]**

EPx{x=0,a~h}Join.JoinCPU_Rd ビットがセットされているエンドポイント FIFO のデータをバイト単位で読み出せます。このレジスタを用いて FIFO のデータを読み出す場合は、必ず FIFO_RdRemain_H,L レジスタにより読み出し可能データ数を確認した後、リードを行ってください。

RAM Rd Adrs L							
USB[0x140]　初期値 = 0x00							
RAM Read Address[7:2]						–	–
7	6	5	4	3	2	1	0

RAM Rd Adrs H							
USB[0x141]　初期値 = 0x00							
–	–	–	RAM Read Address[12:8]				
7	6	5	4	3	2	1	0

Bit [7:2] :　**RAM_RdAdrs[8:2]**

Bit [1:0] :　**Reserved**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**RAM_RdAdrs[12:8]**
RAM_Rd を行う先頭アドレスを設定します。このレジスタを設定した後、RAM_RdCount レジスタを設定し、RAM_RdControl レジスタのビットをセットして下さい。RAM_Rd 機能が開始します。このレジスタの値は、RAM_Rd 機能作動中は内部動作に応じて変化します。従って、一旦 RAM_RdControl レジスタのビットをセットして、RAM_Rd 機能を開始させた後は CPU_IntStat.RAM_RdCmp ビットがセットされるまでこのレジスタの値を読み出さないで下さい。RAM_Rd 機能作動中にこのレジスタを読み出した場合の値は保証されません。また、RAM_Rd 機能作動中にこのレジスタに書き込んだ場合、誤動作の原因となりますのでご注意ください。

RAM Rd Control							
USB[0x142]　初期値 = 0x00							
RAM_GoRd CBW_CSW	RAM_GoRd	–	–	–	–	–	–
7	6	5	4	3	2	1	0

Bit 7 :　**RAM_GoRdCBW_CSW**
CBW エリアに受信したデータを読み出すために RAM_Rd 機能を開始するビットです。
このビットに“1”を書き込むと、RAM_Rd 機能を開始し、CBW エリアからデータをリードします。
RAM_Rd_00～RAM_Rd_1E レジスタの値が有効になると、CPU_IntStat.RAM_RdCmp ビットが“1”にセットされ、このビットが自動的にクリアされます。
RAM_RdAdrs_H,L レジスタ、RAM_RdCount レジスタの設定は必要ありません。
RAM_GoRd ビットと同時にセットすると、本ビットの機能が優先されます。

Bit 6 :　**RAM_GoRd**
RAM_Rd 機能を開始するビットです。
RAM_RdAdrs_H,L レジスタに RAM_Rd を行う先頭アドレスを設定した後、RAM_RdCount レジスタを設定し、このビットに“1”を書き込むと RAM_Rd 機能を開始します。指定された先頭アドレスから、指定されたカウント数分のデータをリードし RAM_Rd_xx{xx=00~1F}レジスタの値が有効になると CPU_IntStat.RAM_RdCmp ビットが“1”にセットされ、このビットが自動的にクリアされます。
RAM_GoRdCBW_CSW ビットと同時にセットすると、RAM_GoRdCBW_CSW ビットの機能が優先されます。

Bit [5:0] :　**Reserved**

RAM Rd Count
USB[0x143]　　初期値 = 0x00

–	–	RAM Rd Count[5:2]				–	–
7	6	5	4	3	2	1	0

Bit [7:6] :　**Reserved**

Bit [5:2] :　**RAM_RdCount [5:2]**
RAM_Rd 機能を用いて RAM_Rd_xx{xx=00~1F}レジスタにリードするデータ数を設定します。RAM_RdAdrs_H,L レジスタを設定した後、このレジスタをセットし、RAM_RdControl レジスタのビットをセットして RAM_Rd 機能を開始して下さい。このレジスタの値は、RAM_Rd 機能作動中は内部動作に応じて変化します。従って、一旦 RAM_RdControl レジスタのビットをセットして、RAM_Rd 機能を開始させた後は CPU_IntStat.RAM_RdCmp ビットがセットされるまでこのレジスタの値を読み出さないで下さい。RAM_Rd 機能作動中にこのレジスタを読み出した場合の値は保証されません。また、RAM_Rd 機能作動中にこのレジスタに書き込んだ場合、誤動作の原因となりますのでご注意ください。
このレジスタの最大設定数は 32 バイトです。32 バイトを超えるデータ数の設定は誤動作の原因となりますのでご注意ください。

Bit [1:0] :　**Reserved**

RAM Wr Address L
USB[0x144]　　初期値 = 0x00

RAM WrAdrs[7:0]							
7	6	5	4	3	2	1	0

RAM Wr Address H
USB[0x144]　　初期値 = 0x00

–	–	–	RAM WrAdrs[12:8]				
7	6	5	4	3	2	1	0

RAM_WrDoor_H,L レジスタによる RAM へのライトを行うアドレスを指定します。

Bit [7:0] :　**RAM_WrAdrs[7:0]**

Bit [7:5] :　**Reserved**

Bit [4:0] :　**RAM_WrAdrs[12:8]**
RAM への書き込みを行う際のアドレスを指定します。RAM_WrDoor_H,L レジスタへの書き込みバイト数に応じてアドレスはインクリメントされます。RAM_WrDoor_H,L レジスタ書き込み直後には正確な RAM_WrAdrs を確認することは出来ませんので、1CPU サイクル以上の間隔を開けて、RAM_WrAdrs を確認して下さい。データの書き込みに関しては RAM_WrDoor_H,L レジスタの項を参照して下さい。
RAM_WrAdrs を参照する場合は、RAM_WrAdrs_H,RAM_WrAdrs_L の順に読み出してください。

RAM Wr Door 0
USB[0x146]　　初期値 = 0xXX

RAM WrDoor0[7:0](W)							
7	6	5	4	3	2	1	0

RAM Wr Door 1
USB[0x147]　　初期値 = 0xXX

RAM WrDoor1[7:0] (W)							
7	6	5	4	3	2	1	0

Bit [7:0] :　　**RAM_WrDoor0[7:0]**

Bit [7:0] :　　**RAM_WrDoor1[7:0]**
RAM へのライトを行う際のアクセスレジスタです。ライトオンリーのレジスタです。
書き込み開始前に、RAM_WrAdrs_H,L レジスタに、RAM のデータを書き込む先頭アドレスを設定して下さい。その後、このレジスタに書き込みを行えば、RAM_WrAdrs_H,L が自動的に書き込みバイト数に応じてインクリメントされ、順次書き込みが行えます。
デバイス・モードの場合には、RAM_WrDoor_H,L レジスタにより、デスクリプタエリアおよび CSW エリアへのデータの書き込みを行えます。RAM_WrDoor_H,L レジスタによりデスクリプタエリアへ書き込んだデータは、ReplyDescriptor の機能によって何度でも使用できます。即ち、このデータは Descriptor 返信機能によって、消される、または上書きされることは有りません。但し、Descriptor データを書き込んだ領域が、他のエンドポイントで確保されている領域と重なる場合には、データは上書きされることがあります。

RAM Rd 00～RAM Rd 1F
USB[0x150]~USB[0x16F]　　初期値 = 0x00

RAM Rd 00[7:0](R) ～ RAM Rd 1F[7:0] (R)							
7	6	5	4	3	2	1	0

Bit [7:0] :　　**RAM_Rd_xx[7:0]**
RAM_Rd 機能を用いて RAM からリードしたデータを格納するレジスタです。RAM_RdAdrs_H,L レジスタ、RAM_RdCount レジスタを設定し、RAM_RdControl レジスタのビットを用いて RAM_Rd 機能を開始して下さい。本レジスタの値が有効になると FIFO_IntaStat.RAM_RdCmp ビットが“1”にセットされます。RAM_RdCount レジスタに設定した値が 32 バイト未満の場合、RAM からリードしたデータは RAM_Rd_00 から順に格納されます。RAM_RdCount レジスタに設定したカウント数以降のレジスタの値（例えば、カウント設定が“16”の場合、RAM_Rd_10～RAM_Rd_1F）は無効となります。

A2.6　機能説明

A2.6.1　初期設定

USB コントローラを使用する際には、下記の初期設定を行ってください。

A2.6.1.1　USB コントローラへのアクセス設定

本 USB コントローラへアクセスするには以下の初期設定が必要です。

① APB バス Wait の選択
CPU から USB コントローラへのアクセスウエイトを APB ブリッジの“APB WAIT2 レジスタ（APBWAIT2）”の bit[31:30]に+2wait 以上で設定してください。
アドレス 0xFFFE_0008/bit[31:30] = 10

② I/O クロックの On　（CPU⇔USB コントローラ）
システムコントローラの“IO　Clock Control Register （IOCLKCTL）”の bit1 に“1”を書き込みます。
アドレス　：0xFFFF_D014/bit[1] = 1

③ USB コントローラ内部のクロック　On
USB コントローラ内部で使用するクロックを有効にします。
システムコントローラの“IO　Clock Control Register （MISC）”の bit1 に“1”を書き込むことで、USB コントローラ内のレジスタは USB コントローラ内蔵の PLL480 の 8 分の 1 分周の 60MHz クロックで動作します。
アドレス　：0xFFFF_D06C/bit[1] = 1
bit[1]　：　1　60MHz ON
0　60MHz OFF

A2.6.1.2　割り込みの設定

本 USB コントローラからの割り込みは、割り込みコントローラ（INTC）の通常割り込み IRQ29 に割り付け割れています。USB コントローラからの割り込み信号は Low アクティブのレベル信号で出力されますが、IC の内部において INTC の入力時に論理反転されているため、INTC の設定としては High アクティブのレベル信号として扱います。

① レベルレジスタの設定
割り込みコントローラの“IRQxx レベルレジスタ”の bit[29]に 0 を設定してください。
アドレス　：0xFFFF_F0A0/bit[29] = 0

② 極性レジスタの設定
割り込みコントローラの“IRQxx 極性レジスタ”の bit[29]に 1 を設定してください。
アドレス　：0xFFFF_F0A4/bit[29] = 1

A2.6.1.3　Macro Config1[0x37]に関する注意事項

本レジスタは本 LSI 用に最適化されています。必要のない限り、設定を変更しないでください。

bit[7]	0：負論理
bit[6]	0：1/0 モード
bit[5]	0：負論理
bit[4]	0：負論理
bit[3]	0：XDACK0,1 がアサートされているとき有効な DMA アクセスとして動作します。
bit[2]	1：偶数アドレスを下位側、奇数アドレスを上位側とします。
bit[1:0]	1x：16bit BE mode （固定）

本レジスタの設定（bit[2]）により、初期状態では Little Endian 形式となっています。

A2.6.2　USB デバイス制御

ここでは、USB デバイス機能について説明します。

A2.6.2.1　エンドポイント

本 LSI はコントロール転送用のエンドポイント（EP0）と、8 本の汎用エンドポイント（EPa-h）を持ちます。エンドポイント EPa-h は、それぞれに、バルクまたはインタラプト転送用のエンドポイントとして使用できます。
LSI のハードウェアは、エンドポイントを提供し、トランザクションの管理を行います。一方、USB に定義されるインタフェース（以下、USB 定義インタフェース）の管理機能を提供しません。USB 定義インタフェースは､ファームウェアで実装して下さい｡デバイス固有のデスクリプタ定義に沿って、エンドポイントを適宜設定し、組み合わせて、USB 定義インタフェースを構成してください。
各エンドポイントには、USB 定義インタフェースによって決定される固定の基本設定項目と、転送毎に制御を行う可変の制御項目及びステータスがあります｡基本設定項目は､チップ初期化時､または､USB 定義インタフェースの切り替え時等に設定して下さい。
Table A2.6.1 にエンドポイント EP0（デフォルトコントロールパイプ）の基本設定項目を示します。
エンドポイント EP0 は、IN 方向と OUT 方向とで、レジスタセットや FIFO 領域を共有します。エンドポイント EP0 におけるデータステージ及びステータスステージでは、その実行に際して、ファームウェアによって、適宜データトランザクションの方向を設定して下さい。

Table A2.6.1　エンドポイント EP0 の基本設定項目

項目	レジスタ／ビット	説明
マックスパケットサイズ	EP0MaxSize	マックスパケットサイズを、FS 動作時には 8,16,32,64 のいずれかの値に設定します。また HS 動作時には 64 に設定します。 エンドポイント EP0 には FIFO の 0 番地から 64 バイトの領域が割当てられます。

Table A2.6.2 に汎用エンドポイント（EPa-h）の基本設定項目を示します。エンドポイント EPa-h は、トランザクション方向とエンドポイントナンバーを任意に設定出来ますので、8 本までの独立したエンドポイントを使用できます。USB 定義インタフェースの定義内容に合わせて適宜設定し、また有効にすることによって、USB 定義インタフェースを構成して下さい。
エンドポイント EPa-h の FIFO 領域はスタートアドレスおよびエンドアドレスにより設定します。

Table A2.6.2　汎用エンドポイントの基本設定項目

項目	レジスタ／ビット	説明
トランザクション方向	EPx(x=a-h)Config.INxOUT	各エンドポイントの転送方向を設定します。
マックスパケットサイズ	EPx(x=a-h)MaxSize_H EPx(x=a-h)MaxSize_L	各エンドポイントのマックスパケットサイズを 8,16,32,64,512 バイトのいずれかの値に設定します。 但しバルク転送を行うエンドポイントでは､FS モード時は 8/16/32/64 バイトのいずれか、HS モード時は 512 バイトに設定してください。
エンドポイントナンバー	EPx(x=a-h)Config.EndpointNumber	各エンドポイントのエンドポイントナンバーを 0x1~0xF の間の任意の値に設定します。
トグルモード	EPx(x=a0h)Config.IntEP_Mode	インタラプト転送の動作モードを設定します。バルク転送を行うエンドポイントでは、方向に関わらず“0”に設定してください。 IN 方向のエンドポイントでは､トグルシーケンスのモード設定をします｡OUT 転送のエンドポイントでは、インタラプト転送を行う場合に“0”にセットしてください。
エンドポイント有効	EPx(x=a-h)Config.EnEndpoint	各エンドポイントを有効にします。 そのエンドポイントを使用する USB 定義インタフェースが有効にされたときに設定してください。
FIFO 領域	EPx(x=a-h)StartAdrs_H EPx(x=a-h)StartAdrs_L EP_EndAdrs_H EP_EndAdrs_L	各エンドポイントに割り当てる領域を FIFO のアドレスで設定します。 FIFO 領域は、各チャネルのマックスパケットサイズ以上の領域を割り当ててください。また、FIFO 領域のサイズがデータ転送のスループットに影響します。 FIFO の領域割り当ての詳細は､機能説明の FIFO の項を参照してください。

A2.6.2.2　トランザクション

LSI は H/W でトランザクション実行機能と、ファームウェアに対するトランザクション実行のためのインタフェースを提供します。ファームウェアに対するインタフェースは、制御レジスタとステータスレジスタ、及び、ステータスによりアサートされる割り込み信号として実装されています。ステータスにより割り込みをアサートする設定については、レジスタ説明の章を参照して下さい。
LSI は個々のトランザクション毎に、ファームウェアに対してステータスを発行します。しかしながら、ファームウェアは必ずしも、個々のトランザクションを管理する必要はありません。LSI はトランザクションへの応答を行うときに、FIFO を参照し、そのデータ数または空き数によって、データ転送を行えるか否かを判断して自動的に処理を行います。
例えば、OUT のエンドポイントであれば、ファームウェアは、CPU インタフェース（DMA リードまたはレジスタリード）により、FIFO からデータを読み出して FIFO に空き領域を作り出すことによって、OUT トランザクションを自動的に連続して実行させることが出来ます。また、IN のエンドポイントであれば、ファームウェアは、CPU インタフェース（DMA ライトまたはレジスタライト）により、FIFO にデータを書き込んで FIFO に有効データを作り出すことによって、IN トランザクションを自動的に連続して実行させることが出来ます。

Table A2.6.3 にエンドポイント EP0 のトランザクション制御に関する制御項目及びステータスを示します。

Table A2.6.3　エンドポイント EP0 の制御項目及びステータス

項目	レジスタ／ビット	説明
トランザクション方向	EP0Control.INxOUT	データステージ及びステータスステージにおいて、転送方向を設定します。
デスクリプタ返信イネーブル	EP0Control.ReplyDescriptor	デスクリプタの自動応答を起動します。
デスクリプタ返信アドレス	DescAdrs_H DescAdrs_L	デスクリプタの自動応答によって、返信を行うデータの FIFO 上の先頭アドレスを指定します。
デスクリプタサイズ	DescSize_H DescSize_L	デスクリプタの自動応答によって、返信を行うデータ数を指定します。
制御禁止	SETUP_Control.ProtectEP0	このビットがセットされていると、EP0ControlIN 及び EP0ControlOUT レジスタの ForceNAK ビットと ForceSTALL ビットへのアクセスが行えません。 このビットは、RcvEP0SETUP ステータスが立つと、LSI の H/W によってセットされ、CPU によるレジスタアクセスでクリアできます。
ショートパケット送信イネーブル	EP0ControlIN.EnShortPkt	マックスパケットサイズに満たない、ショートパケットの送信を有効にします。ショートパケットを送信した IN トランザクションが完結すると､クリアされます。
トグルシーケンスビット	EP0ControlIN.ToggleStat EP0ControlOUT.ToggleStat	トグルシーケンスビットの状態を示します。SETUP ステージにより、自動的に初期化されます。
トグルセット	EP0ControlIN.ToggleSet EP0ControlOUT.ToggleSet	トグルシーケンスビットをセットします。
トグルクリア	EP0ControlIN.ToggleClr EP0ControlOUT.ToggleClr	トグルシーケンスビットをクリアします。
強制 NAK 応答	EP0ControlIN.ForceNAK EP0ControlOUT.ForceNAK	FIFO のデータ数/空き数に関わらず、IN または OUT（PING を含む）トランザクションに NAK 応答します。
STALL 応答	EP0ControlIN.ForceSTALL EP0ControlOUT.ForceSTALL	IN または OUT（PING を含む）トランザクションに STALL 応答します。
自動 ForceNAK セット	EP0ControlOUT.AutoForceNAK	OUT トランザクションの完結ごとに、EP0ControlOUT.ForceNAK ビットをセットします。
SETUP 受信ステータス	DeviceIntStat.RcvEP0SETUP	SETUP トランザクションが実行されたことを示します。
トランザクションステータス	EP0IntStat.OUT_ShortACK EP0IntStat.IN_TranACK EP0IntStat.OUT_TranACK EP0IntStat.IN_TranNAK EP0IntStat.OUT_TranNAK EP0IntStat.IN_TranErr EP0IntStat.OUT_TranErr	トランザクションの結果を示します。
デスクリプタ返信データステージ終了ステータス	FIFO_IntStat.DescriptorCmp	デスクリプタ j 同応答のデータステージが終了したことを示します。

Table A2.6.4 に汎用エンドポイント EPa-h のトランザクション処理に関する制御項目とステータスを示します。

Table A2.6.4　汎用エンドポイントの制御項目とステータス

項目	レジスタ／ビット	説明
自動 ForceNAK セット	EPx{x=a-h}Control.AutoForceNAK	OUT トランザクションの完結ごとに、そのエンドポイントの EPx{x=a-h}Control.ForceNAK ビットをセットします。
ショートパケット送信イネーブル	EP x{x=a-h}Control.EnShortPkt	IN トランザクションに対し、マックスパケットサイズに満たない、ショートパケットの送信を有効にします。ショートパケットを送信した IN トランザクションが完結すると、クリアされます。
ショートパケット受信による自動 ForceNAK セットの禁止	EPx{x=a-h}Control.DisAF_NAK_Short	OUT トランザクションにおいて、ショートパケットを受信すると、自動的にそのエンドポイントの EPx{a-h}Control.ForceNAK ビットをセットする機能（※）を禁止します。 ※：このビットにより禁止しない場合は有効になっています。
トグルシーケンスビット	EP x{x=a-h}Control.ToggleStat	トグルシーケンスビットの状態を示します。
トグルセット	EP x{x=a-h}Control.ToggleSet	トグルシーケンスビットをセットします。
トグルクリア	EP x{x=a-h}Control.ToggleClr	トグルシーケンスビットをクリアします。
強制 NAK 応答	EP x{x=a-h}Control.ForceNAK	FIFO のデータ数/空き数に関わらず、トランザクションに NAK 応答します。
STALL 応答	EP x{x=a-h}Control.ForceSTALL	トランザクションに STALL 応答します。
トランザクションステータス	EP x{x=a-h}IntStat.OUT_ShortACK EP x{x=a-h}IntStat.IN_TranACK EP x{x=a-h}IntStat.OUT_TranACK EP x{x=a-h}IntStat.IN_TranNAK EP x{x=a-h}IntStat.OUT_TranNAK EP x{x=a-h}IntStat.IN_TranErr EP x{x=a-h}IntStat.OUT_TranErr	トランザクションの結果を示します。

A2.6.2.2.1　SETUP トランザクション

自ノードのエンドポイント EP0 宛ての SETUP トランザクションは、無条件に実施します。（NegoControl.ActiveUSB ビットによって USB 機能は有効にされている必要が有ります。）
SETUP トランザクションが発行されると、データパケット（8Byte）の全ての内容を EP0SETUP_0～EP0SETUP_7 レジスタに格納し、ACK 応答します。また、SetAddress()リクエストを除き、ファームウェアに対し RcvEP0SETUP ステータスを発行します。
SETUP トランザクション中にエラーが発生した場合には、応答せず、ステータスを発行しません。
SETUP トランザクションが完結すると、EP0ControlIN レジスタ及び EP0ControlOUT レジスタの ForceNAK ビットをセットし、ForceSTALL ビットをクリアします。また、ToggleStat ビットをセットします。また、SETUP_Control.ProtectEP0 ビットをセットします。ファームウェアは、エンドポイント EP0 の設定を終え、次のステージに移行可能になったら、SETUP_Control.ProtectEP0 ビットをクリアし、EP0ControlIN レジスタまたは EP0ControlOUT レジスタにおいて、該当する方向の ForceNAK ビットをクリアして下さい。

Figure A2.6.1 に SETUP トランザクションの様子を図示します。(a) ホストが、このノードのエンドポイント 0 に宛てた SETUP トークンを発行します。(b) ホストは続けて、8Byte 長のデータパケットを送信します。LSI はこのデータを EP0SETUP_0～EP0SETUP_7 レジスタに書き込みます。(c) LSI は自動的に ACK 応答します。また、自動設定するレジスタを設定し、ファームウェアに対しステータスを発行します。

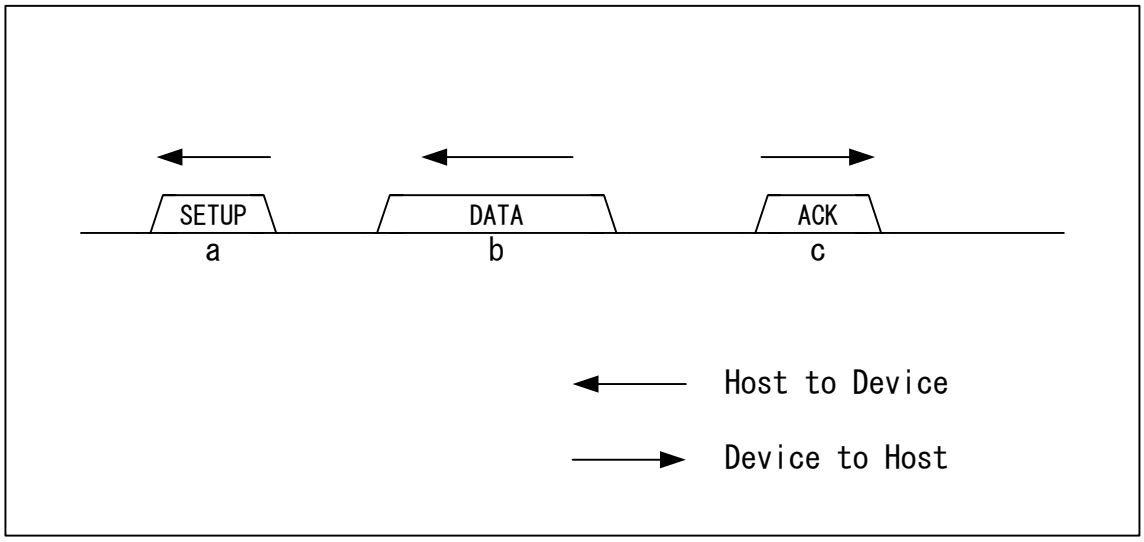


Figure A2.6.1　SETUP トランザクション

A2.6.2.2.2　バルク/インタラプト OUT トランザクション

バルク及びインタラプト OUT トランザクションでは、FIFO の空き容量がマックスパケットサイズ以上有ると、データの受信を開始します。
バルク及びインタラプト OUT トランザクションにおいて、全てのデータが正常に受信できると、トランザクションを完結し、ACK 応答します。また、ファームウェアに対し、該当するエンドポイントの OUT_TranACK ステータス（EPx{x=0,a-h}IntStat.OUT_TranACK ビット）を発行します。また、FIFO を更新して、データを受信済みとして、領域を確保します。
また、バルク及びインタラプト OUT トランザクションにおいて、ショートパケットの全てのデータを受信できると、上記のトランザクション完結処理に加え、OUT_ShortACK ステータス (EPx{x=0,a-h}IntStat.OUT_ShortACK ビット)を発行します。また、EPx{x=a-h}Control.DisAF_NAK_Short ビットがクリアされていると、そのエンドポイントの、EPx{x=a-h}ForceNAK ビットをセットします。
バルク及びインタラプト OUT トランザクションにおいて、トグルミスマッチが発生した場合、トランザクションに ACK 応答しますが、ステータスを発行しません。FIFO は更新されません。
バルク及びインタラプト OUT トランザクションにエラーが発生した場合、トランザクションに応答しません。また、OUT_TranErr ステータス（EPx{x=0,a-h}IntStat.OUT_TranErr ビット）を発行します。FIFO は更新されません。
バルク及びインタラプト OUT トランザクションにおいて、全てのデータを受信できなかった場合、トランザクションに NAK 応答します。また、OUT_TranNAK ステータス（EPx{x=0,a-h}IntStat.OUT_TranNAK ビット）を発行します。FIFO は更新されません。
Figure A2.6.2 に、完結する場合のバルクまたはインタラプト OUT トランザクションの様子を図示します。(a) ホストが、このノードに存在する OUT 方向のエンドポイントに宛てた OUT トークンを発行します。(b) ホストは続けて、マックスパケットサイズ以内のデータパケットを送信します。LSI はこのデータを、該当するエンドポイントの FIFO に書き込みます。(c) LSI はデータを受信できると、自動的に ACK 応答します。また、自動設定するレジスタを設定し、ファームウェアに対しステータスを発行します。

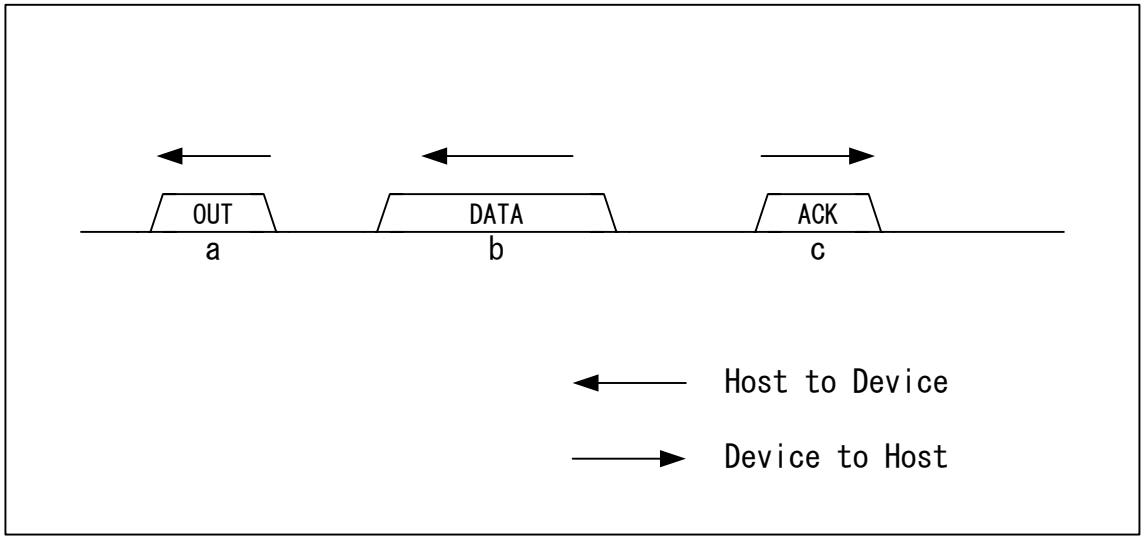


Figure A2.6.2　OUT トランザクション

A2.6.2.2.3　バルク/インタラプト IN トランザクション

IN 方向のバルク及びインタラプトのエンドポイントにおいて、FIFO にマックスパケットサイズ分のデータが有るか、または、ファームウェアによってショートパケットの送信が許可されていると、IN トランザクションに応答して、データパケットを返信します。
ショートパケット（データ長ゼロのパケットを含む）の送信許可は、EP0ControlIN.EnShortPkt ビットまたは EPx{x=a-h}Control.EnShortPkt ビットをセットすることで行います。ショートパケットを送信する場合、送信許可後、トランザクションが完結するまでの間、新たなデータをそのエンドポイントの FIFO に書き込まないようにして下さい。
エンドポイント EP0 では、ショートパケットを送信する IN トランザクションが完結すると、EP0ControlIN. ForceNAK ビットがセットされます。
データ返信した IN トランザクションで、ACK を受信すると、トランザクションを完結し、ファームウェアに対し、IN_TranACK ステータス（EPx{x=0,a-h}IntStat.IN_TranACK ビット）を発行します。また、FIFO を更新して、送信したデータを送信済みとして領域を開放します。
データ返信した IN トランザクションで、ACK を受信しないと、トランザクションを失敗と見なし、ファームウェアに対し、IN_TranErr ステータス（EPx{x=0,a-h}IntStat.IN_TranErr ビット）を発行します。また、FIFO を更新せず、領域を開放しません。
バルク及びインタラプトの IN 方向のエンドポイントにおいて、FIFO にマックスパケットサイズ分のデータが無く、かつ、ショートパケットの送信が許可されていないと、IN トランザクションに NAK 応答し、ファームウェアに対し、IN_TranNAK ステータス（EPx{x=0,a-h}IntStat.IN_TranNAK ビット）を発行します。また、FIFO を更新せず、領域を開放しません。
Figure A2.6.3 に、完結する場合のバルクまたはインタラプト IN トランザクションの様子を図示します。
（a）ホストが、このノードに存在する IN 方向のエンドポイントに宛てた IN トークンを発行します。
（b）LSI は、この IN トランザクションに応答できる場合、マックスパケットサイズ以内のデータパケットを送信します。（c）ホストは ACK 応答します。LSI は ACK を受信すると、自動設定するレジスタを設定し、ファームウェアに対しステータスを発行します。

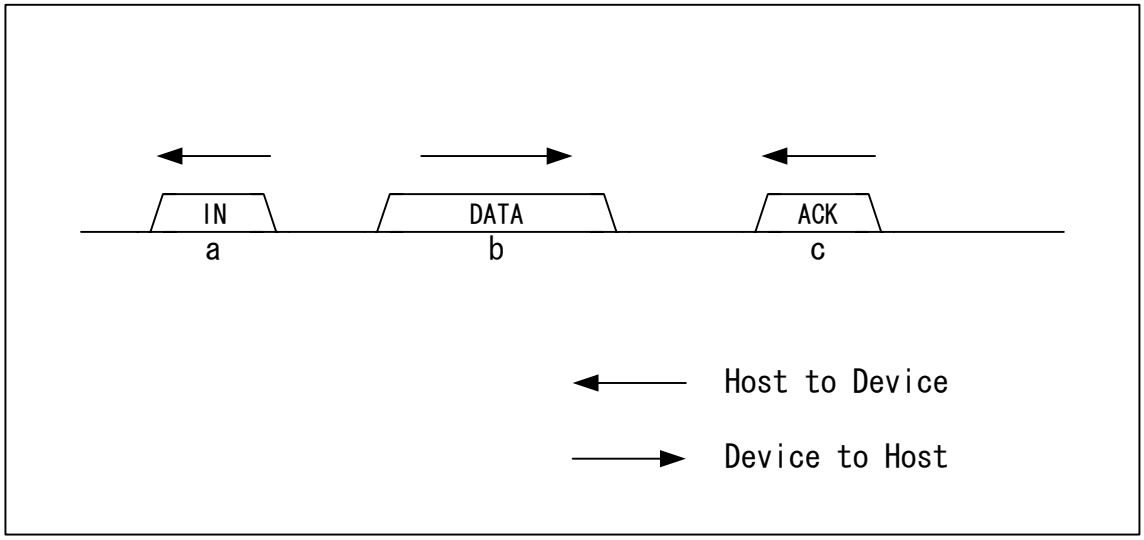


Figure A2.6.3　IN トランザクション

A2.6.2.2.4　PING トランザクション

バルクの OUT 方向のエンドポイントでは、HS 動作時に、PING トランザクションを実行します。
該当するエンドポイントの FIFO 空き容量がマックスパケットサイズ以上であった場合に、PING トランザクションに対して ACK 応答します。また、ファームウェアに対して、ステータスを発行しません。
該当するエンドポイントの FIFO の空き容量がマックスパケットサイズ未満であった場合に、PING トランザクションに対して NAK 応答します。また、ファームウェアに対し、OUT_TranNAK ステータス（EPx{x=0,a-h}IntStat.OUT_TranNAK ビット）を発行します。
PING トランザクションにおいては、FIFO が更新されることはありません。
Figure A2.6.4 に、PING トランザクションに対して ACK 応答する様子を図示します。(a) ホストが、このノードに存在する OUT 方向のエンドポイントに宛てた PING トークンを発行します。(b) LSI は、FIFO にマックスパケットサイズ分の空きがある場合、この PING トランザクションに対し ACK 応答します。また、ファームウェアに対しステータスを発行します。

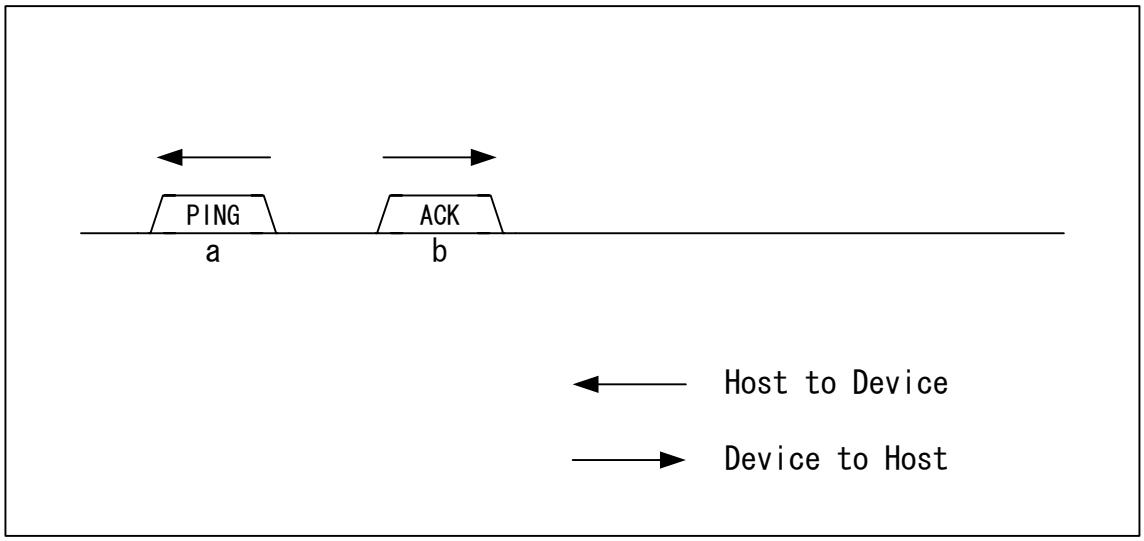


Figure A2.6.4　PING トランザクション

A2.6.2.3　コントロール転送

エンドポイント EP0 におけるコントロール転送は、SetAddress()リクエストを除き、個々のトランザクションの組み合わせとして制御します。SetAddress()リクエストは、後述の自動アドレス設定機能により、自動的に処理されます。
Figure A2.6.5 にデータステージが OUT 方向である場合のコントロール転送の様子を図示します。(a) ホストは、SETUP トランザクションによって、コントロール転送を開始します。デバイスのファームウェアはリクエストの内容を解析して、データステージに応答する準備をします。(b) ホストは OUT トランザクションを発行して、データステージを行い、デバイスはデータを受信します。(c) ホストは IN トランザクションを発行して、ステータスステージを行い、デバイスはデータ長ゼロのパケットを返信します。
データステージの無いコントロール転送は、この例におけるデータステージが無い状態で実施されます。
ステータスステージへの移行は、ホストがデータステージと逆方向のトランザクションを発行することによってなされます。ファームウェアは、IN_TranNAK ステータス（EP0IntStat.IN_TranNAK ビット）を監視して、データステージからステータスステージに移行するきっかけとして下さい。

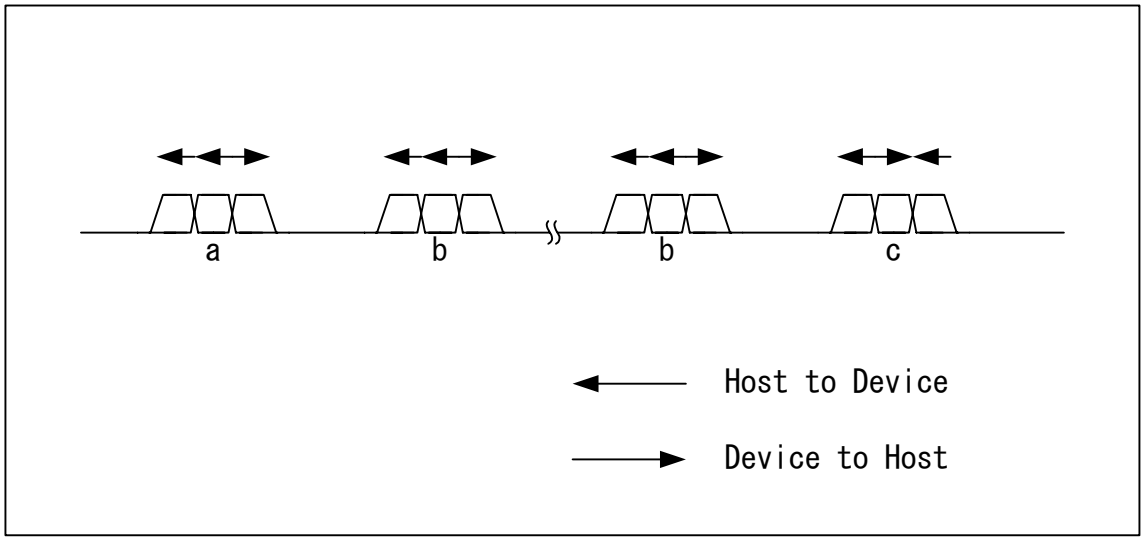


Figure A2.6.5　データステージが OUT 方向のコントロール転送

Figure A2.6.6 にデータステージが IN 方向である場合のコントロール転送の様子を図示します。(a) ホストは、SETUP トランザクションによって、コントロール転送を開始します。デバイスのファームウェアはリクエストの内容を解析して、データステージに応答する準備をします。(b) ホストは IN トランザクションを発行して、データステージを行い、デバイスはデータを送信します。(c) ホストは OUT トランザクションを発行して、ステータスステージを行い、デバイスは ACK 応答します。
ステータスステージへの移行は、ホストがデータステージと逆方向のトランザクションを発行することによってなされます。ファームウェアは、OUT_TranNAK ステータス（EP0IntStat.OUT_TranNAK ビット）を監視して、データステージからステータスステージに移行するきっかけとして下さい。
コントロール転送のデータステージ及びステータスステージは、通常の OUT 及び IN トランザクションを行いますので、NAK によるフロー制御が有効です。デバイスは定められた時間内に、応答する準備をすることが許されています。

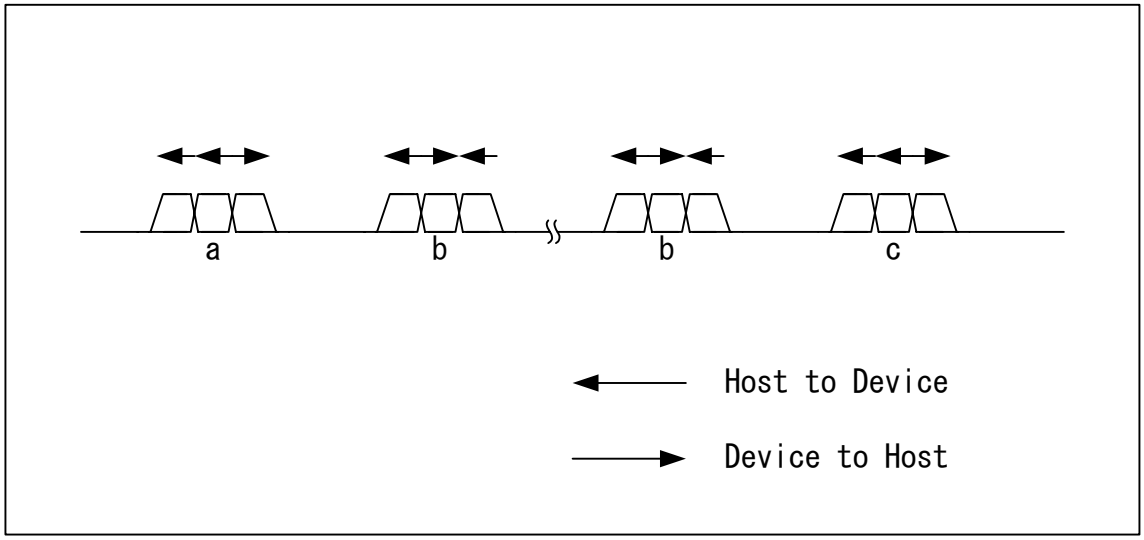


Figure A2.6.6　データステージが IN 方向のコントロール転送

A2.6.2.3.1　セットアップステージ

自ノードに宛てられた SETUP トークンを受信すると、自動的にセットアップトランザクションを実行します。
ファームウェアは、RcvEP0SETUP ステータスをモニタし、EP0SETUP_0～EP0SETUP_7 レジスタによりリクエストを解析して、コントロール転送を制御して下さい。
受信したリクエストが、OUT 方向のデータステージが有るものであった場合、データステージに移行するため、EP0Control レジスタの INxOUT ビットをクリアして、エンドポイント EP0 を OUT 方向に設定して下さい。
受信したリクエストが、IN 方向のデータステージが有るものであった場合、データステージに移行するため、EP0Control レジスタの INxOUT ビットをセットして、エンドポイント EP0 を IN 方向に設定して下さい。
受信したリクエストが、データステージの無いものであった場合、ステータスステージに移行するため、EP0Control レジスタの INxOUT ビットをセットして、エンドポイント EP0 を IN 方向に設定して下さい。

A2.6.2.3.2　データステージ／ステータスステージ

EP0SETUP_0～EP0SETUP_7 レジスタを読み出してリクエストを解析した内容に従って、次のステージに移行して下さい。
そのステージが OUT 方向である場合、EP0Control レジスタの INxOUT をクリアして OUT 方向に設定し、EP0ControlOUT レジスタを適宜設定して、ステージをコントロールして下さい。SETUP ステージ終了時は、ForceNAK ビットがセットされています。また、SETUP_Control.ProtectEP0 ビットがセットされています。
そのステージが IN 方向である場合、EP0Control レジスタの INxOUT をセットして IN 方向に設定し、EP0ControlIN レジスタを適宜設定して、ステージをコントロールして下さい。SETUP ステージ終了時は、ForceNAK ビットがセットされています。また、SETUP_Control.ProtectEP0 ビットがセットされています。

A2.6.2.3.3　自動アドレス設定機能

本 LSI には、エンドポイント EP0 におけるコントロール転送において、SetAddress()リクエストの処理を自動化する機能があります。
LSI の H/W は EP0SETUP_0～EP0SETUP_7 レジスタによってリクエストの内容を確認し、有効な

SetAddress()リクエストで有った場合には、ファームウェアに通知することなく、このリクエストのステータスステージの処理に移行します。ステータスステージが完了すると、USB_Address レジスタにアドレスを設定し、ファームウェアに対し SetAddressCmp ステータス(SIE_IntStat.SetAddressCmp ビット)を発行します。
ファームウェアは SetAddressCmp ステータスを監視し、これが発行されたら USB_Address レジスタにより、アドレスを確認できます。

A2.6.2.3.4　デスクリプタ返信機能

本 LSI には、エンドポイント EP0 におけるコントロール転送において、GetDescriptor()等の複数回発行されデータを要求するリクエストに有効な、デスクリプタ返信機能があります。
データステージが IN 転送であるリクエストにおいて、ファームウェアはこの機能を使用することが出来ます。
EP0ControlIN.ForceNAK ビットをクリアして、データステージへの応答を開始する前に、DescAdrs_H,L レジスタに FIFO のデスクリプタ領域内の、返信するデータの先頭アドレスを、また、DescSize_H,L レジスタに返信するデータの総バイト数を設定し、EP0Control.ReplyDescriptor ビットをセットして下さい。
デスクリプタ返信機能は、設定数のデータを送り終わるまで、データステージの IN トランザクションに応答してデータパケットを返信し、IN トランザクションを実行します。設定数のデータを送り終えた後に IN トランザクションが発行されると、NAK 応答します。マックスパケットサイズに対し端数のデータが存在すると、デスクリプタ返信機能は、EP0ControlIN.EnShortPkt をセットし、全てのデータを返信するまで、IN トランザクションに応答出来るようにします。
OUT トークンを受信し、ステータスステージへの移行を検知すると、EP0Control.ReplyDescriptor ビットをクリアし、ファームウェアに対し DescriptorCmp ステータス（FIFO_IntStat.DescriptorCmp ビット）を発行します。DescriptorCmp ステータスを検知したら、ファームウェアはステータスステージを行って下さい。
デスクリプタ領域については、機能説明の FIFO の項を参照して下さい。

A2.6.2.4　バルク転送/インタラプト転送

汎用エンドポイント EPa-h におけるバルク転送、インタラプト転送は、データフロー（A2.6.2.5 参照）としても、連続する個々のトランザクション（A2.6.2.2 参照）としても制御できます。

A2.6.2.5　データフロー

OUT 転送及び IN 転送の一般的なデータフローの制御について、説明します。

A2.6.2.5.1　OUT 転送

OUT 転送によって受信したデータは、各エンドポイントの FIFO 上に書き込まれます。FIFO のデータを読み出すには、CPU インタフェースによるレジスタ読み出し、DMA による読み出し方法があります。
CPU インタフェースのレジスタリードにより、FIFO のデータを読み出すには、EPx{x=0,a-h}Join.JoinCPU_Rdビットにより、ただ一つのエンドポイントを選択して下さい。選択したエンドポイントのFIFOは、FIFO_Rdレジスタ、または、FIFO_ByteRdレジスタにより、受信順に読み出すことが出来ます。また、読み出し可能なFIFOのデータ数を、FIFO_RdRemain_H,Lレジスタにより参照できます。空のFIFOを読み出すことはできませんので、必ずFIFO_RdRemain_H,Lレジスタによりデータ数を確認し、その数を超えないように読み出して下さい。
DMA リードにより、FIFO のデータを読み出すには、EPx{x=0,a-h}Join.JoinDMAx{x=0}ビットにより、DMA のチャネル毎にただ一つのエンドポイントを選択し、DMAx{x=0}_Control.Dir ビットに 1 を設定して下さい。選択したエンドポイントの FIFO は、外部 DMA コントローラ等により DMA 転送を実行することにより、受信順に読み出されます。また、FIFO の残りデータ数を、DMAx{x=0}_Remain_H,L レジスタで参照できます。FIFO が空になると、自動的に DMA を一時停止してフロー制御を行います。
FIFO にデータパケットを受信できる空き領域が有れば、OUT トランザクションに自動的に応答して、

データを受信できます。従って、ファームウェアによって、個々のトランザクションについての制御を行うことなく、OUT 転送を行うことができます。但し、EPx{x=a-h}Control.DisAF_NAK_Short ビットがクリアされている場合(初期値)、ショートパケット（データ長ゼロのパケットを含む）を受信した場合、そのエンドポイントの EPx{x=a-h}Control.ForceNAK ビットをセットしますので、次のデータ転送を行う準備が出来たら、EPx{x=a-h}Control.ForceNAK ビットをクリアして下さい。

A2.6.2.5.2　IN 転送

IN 転送により送信するデータを、各エンドポイントの FIFO 上に書き込んで下さい。FIFO にデータを書き込むには、CPU インタフェースによるレジスタ書き込みと、DMA による書き込み方法があります。
CPU インタフェースのレジスタライトにより、FIFO にデータを書き込むには、EPx{x=0,a-h}Join.JoinCPU_Wr ビットにより、ただ一つのエンドポイントを選択して下さい。選択したエンドポイントの FIFO には、FIFO_Wr レジスタにより書き込むことができ、書き込み順にデータパケットで送信されます。また、FIFO の空き容量を、FIFO_WrRemain_H,L レジスタにより参照できます。フル状態の FIFO へ書き込むことは出来ません。必ず FIFO_WrRemain_H,L レジスタにより空き数を確認し、その数を超えないように書き込んで下さい。
DMA ライトにより、FIFO にデータを書き込むには、EPx{x=0,a-h}Join.JoinDMAx{x=0}ビットにより、DMA のチャネル毎にただ一つのエンドポイントを選択し、DMAx{x=0}_Control.Dir ビットに 0 を設定して下さい。選択したエンドポイントの FIFO は、外部 DMA コントローラ等により DMA 転送を実行することにより書き込まれ、書き込み順にデータパケットで送信されます。FIFO がフルになると、自動的に DMA を一時停止してフロー制御を行います。
FIFO にマックスパケットサイズ以上のデータが有れば、IN トランザクションに自動的に応答して、データを送信できます。従って、ファームウェアによって、個々のトランザクションについての制御を行うことなく、IN 転送を行うことが出来ます。但し、データ転送の最後にショートパケットを送信する必要がある場合、EnShortPkt ビットをセットして下さい。このビットはショートパケットを送信した IN トランザクションが完結することによってクリアされます。FIFO へのデータ書き込みが終了した時点でセットすることが可能です。また、DMAx{x=0}_FIFO_Control.AutoEnShort ビットがセットされていると、DMA 書き込みが終了したときに、マックスパケットサイズに満たない端数データが FIFO に有ると、そのエンドポイントの EnShortPkt ビットを自動的にセットします。

A2.6.2.6　バルクオンリーサポート

本 LSI には、汎用エンドポイント EPa-h におけるバルク転送において、USB Mass Storage Class(BulkOnly Transport Protocol)に固有の Command Block Wrapper(CBW)の受信及び Command Status Wrapper(CSW)の送信を補助する、バルクオンリーサポート機能があります。
BulkOnlyConfig.EPx{x=a-h}BulkOnly ビットをセットすると、対象となるエンドポイントで、バルクオンリーサポート機能が有効になります。
バルクオンリーサポート機能の CBW サポート、または CSW サポートが実行されている間、エンドポイントに通常割り当てられた FIFO 領域ではなく、CBW 領域または CSW 領域として割り当てられている領域を使用して、パケットの受信(CBW)または送信(CSW)を行います。

A2.6.2.6.1　CBW サポート

ファームウェアは BulkOnly Transport Protocol のコマンドトランスポートを行うときに、CBW サポートを使用することが出来ます。BunlOnlyConfig.EPx{x=a-h}BulkOnly ビットがセットされると、対応する OUT のエンドポイントで CBW サポートが有効になります。CBW サポートは、ただひとつのエンドポイントで有効になるように制御して下さい。CBW サポートが有効であるときに、BulkOnlyContol.GoCBW_Mode ビットをセットすると、CBW サポートが実行され、対象となるエンドポイントにおける OUT トランザクションで受信したデータを CBW として扱います。
データパケットのデータ長が CBW として期待される 31 バイト長であった場合には、データを CBW 領域に保存し、ファームウェアに対し CBW 完了ステータス(BulkIntStat.CBW_Cmp ビット)を発行します。また、BulkOnlyControl.GoCBW_Mode ビットを自動的にクリアし、CBW サポートの実行が終了します。また、このとき BulkOnlyControl.GoCSW_Mode ビットがセットされていると、同時にクリアします。

データパケットのデータ長が、31 バイト長未満か、または、31 バイト長を超えた場合には、データを受信せず、ファームウェアに対し CBW データ長エラーステータス(BulkIntStat.CBW_LengthErr ビット)を発行します。また、BulkOnlyControl.GoCBW_Mode ビットを自動的にクリアし、CBW サポートの実行を終了します。また、このとき BulkOnlyControl.GoCSW_Mode ビットがセットされていると、同時にクリアします。CBW_Err ステータスが発行された場合、BulkOnly Transport Protocol でフェーズミスマッチが発生していますので、ファームウェアはエンドポイントを STALL するなどして、通信の復旧を行ってください。
対象となるエンドポイントで EPx{x=a-h}Control.ForceSTALL がセットされ、OUT トランザクションに STALL 応答した場合には、ファームウェアに対し CBW エラーステータス(BulkIntStat.CBW_Err ビット)を発行し、BulkOnlyControl.GoCBW_Mode ビットをクリアし、CBW サポートの実行が終了します。また、このとき BulkOnlyControl.GoCSW_Mode ビットがセットされていると、同時にクリアします。
OUT トランザクションに CRC エラーなどのトランザクションエラーが発生した場合は、データを受信せずファームウェアに対し CBW トランザクションエラーステータス(BulkIntStat.CBW_TranErr ビット)ステータスを発行します。この場合には、BulkOnlyControl.GoCBW_Mode ビットがクリアされず、CBW サポートの実行が継続します。また、このとき BulkOnlyControl.GoCSW_Mode ビットがセットされていてもクリアされません。
CBW 領域に受信したデータは、RAM_Rd 機能を用いて読み出すことが出来ます。

A2.6.2.6.2　CSW サポート

ファームウェアは BulkOnly Transport Protocol のステータストランスポートを行うときに、CSW サポートを使用することが出来ます。BulkOnlyConfig.EPx{x=a-h}BulkOnly ビットがセットされると、対応する IN のエンドポイントで CSW サポートが有効になります。CSW サポートは、ただひとつのエンドポイントで有効になるように制御して下さい。CSW サポートが有効であるときに、D_BulkOnlyControl.GoCSW_Mode ビットをセットすると、CSW サポートが実行され、対象となるエンドポイントにおける IN トランザクションにおいて送信するデータを CSW とします。
IN トランザクションにおいて、13 バイトの CSW データをホストへ返信した後に、ホストからの ACK を受信してトランザクションを完結した場合には、ファームウェアに対し CSW 完了ステータス(BulkIntStat.CSW_Cmp ビット)を発行します。また、BulkOnlyControl.GoCSW_Mode ビットを自動的にクリアして CSW サポートの実行を終了します。また、同時に BulkOnlyControl.GoCBW_Mode ビットをセットして CBW サポートの実行を開始します。
IN トランザクションにおいて、13 バイトのデータをホストへ返信した後に、ホストからの ACK が受信できなかった場合は、ファームウェアに対し CSW エラーステータス(BulkIntStat.CSW_Err ビット)を発行します。このとき、BulkOnlyControl.GoCSW_Mode ビットをクリアせずに CSW サポートの実行を継続します。また、同時にハードウェアが BulkOnlyControl.GoCBW_Mode ビットをセットして CBW サポートの実行を開始します。即ち、この場合には、CSW サポートの実行と、CBW サポートの実行が同時に行われている状態となります。もし、ホストが CSW を受信できずエラーとなっていた場合には、CSW のリトライが行われますが、CSW サポートが実行中なので応答することができます。また、デバイスが ACK を受信できずにエラーとなった場合には、次の CBW が行われますが、CBW サポートが実行中なので、応答することができ、また、CBW サポートが行われることによって CSW サポートの実行が終了されます。
CSW 領域へは、RAM_WrDoor 機能を用いてデータを書き込むことが出来ます。

A2.6.2.7　オート・ネゴシエーション機能

サスペンド検出、リセット検出、HS Detection Handshake 実行、レジューム検出、リストア実行を USB バスの状態を逐次チェックしながら自動的に行います。実際に何が行われたかは、各割込み（DetectRESET、DetectSUSPEND、ChirpCmp、RestoreCmp）をチェックすることにより確認することができます。

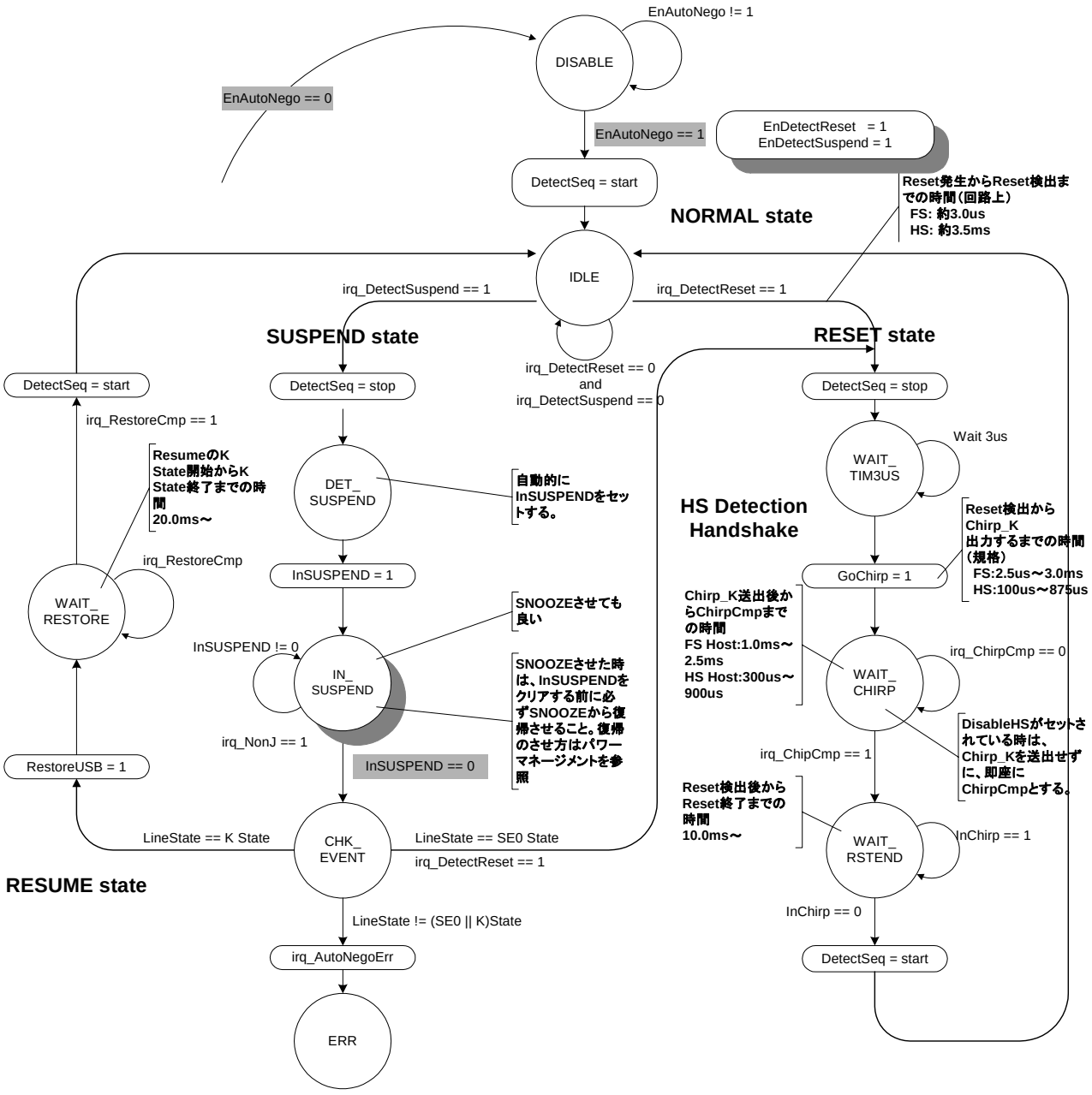


Figure A2.6.7　オート・ネゴシエータ

A2.6.2.7.1　DISABLE

NegoControl.EnAutoNego ビットをクリアしているときに、このステートに入ります。
オート・ネゴシエーション機能を有効にするときには、NegoControl.EnAutoNego ビットをセットする前にリセット検出割り込み許可ビット（SIE_IntEnb.EnDetectRESET）、サスペンド検出割り込み許可ビット（SIE_IntEnb.EnDetectSUSPEND）をセットし、両イベント検出割り込みを許可してください。

オート・ネゴシエーション機能を有効にすると、内部イベント検出機能を有効にします。オート・ネゴシエーション機能を有効にしている間は、NegoControl.DisBusDetect ビットを絶対にセットしないでください。

A2.6.2.7.2　IDLE

リセット検出待ち、サスペンド検出待ちをするステートです。
現在の USB スピードが HS のときには、USB バス上にバス・アクティビティが 3ms 以上検出できなかった場合に、一旦 FS のターミネーションを有効にし、FS-J が検出された場合はサスペンド、SE0 が検出された場合はリセットと判断します。現在のスピードが FS のときには、2.5µs 以上の SE0 が検出された場合はリセット、3ms 以上バス・アクティビティが検出できなかった場合はサスペンドと判断します。これらの判断と同時に、リセット検出割り込み、またはサスペンド検出割り込みが発生し、SIE_IntStat.DetectRESET ビット、または SIE_IntStat.DetectSUSPEND ビットがセットされます。
サスペンドと判断した場合、一旦イベント検出機能を停止し、DET_SUSPEND ステートに入ります。
リセットと判断した場合、一旦イベント検出機能を停止し、WAIT_TIM3US ステートに入ります。

A2.6.2.7.3　WAIT_TIM3US

リセット検出後、HS Detection Handshake を実行するまでの時間を調整しています。一定時間経過後（約 3µs 後）、WAIT_CHIRP ステートに入ります。

A2.6.2.7.4　WAIT_CHIRP

NegoControl.GoChirp ビットを自動的にセットし、HS Detection Handshake を実行します。HS Detection Handshake が終了すると、Chirp 終了割り込みステータス（SIE_IntStat.ChirpCmp）がセットされ、WAIT_RSTEND ステートに入ります。HS Detection Handshake の詳細については、0 を参照してください。
また、NegoControl.DisableHS ビットをセットしている場合は、HS Detection Handshake を実行せずに、Chirp 終了割り込みステータス（SIE_IntStat.ChirpCmp）がセットされ、WAIT_RSTEND ステートに入ります。
なお、このステート終了後は、USB_Status.FSxHS ビットに設定された転送スピードにて動作することとなります。転送スピードが変化したことを検出する必要がある場合は、前述の Chirp 終了割り込みを有効にするために、SIE_IntEnb.EnChirpCmp ビットをセットしてください。

A2.6.2.7.5　WAIT_RSTEND

リセット期間が終了するまで、このステートにて待ちます。HS 時はホストからの Chirp 送信（この IC にとっては受信）が終了したこと、FS 時は SE0 から J に遷移したことをもって、リセット期間の終了と判断します。
リセット期間終了と判断した後、イベント検出機能を有効にし、再度 IDLE ステートに入ります。

A2.6.2.7.6　DET_SUSPEND

サスペンドと判断された場合に、自動的に NegoControl.InSUSPEND ビットがセットされ、IN_SUSPEND ステートに入ります。この NegoControl.InSUSPEND ビットによって、FS-J からのバスの遷移を検出する機能を有効にし、ホストからのレジューム及びリセットを検出するようになります。
サスペンド中に実際に消費電流を軽減するかどうかは、アプリケーションに依存します。本 LSI では、２段階の消費電流軽減策（スヌーズ、スリープ）を持っています。詳しい内容、さらには制御方法につきましては、パワーマネージメント機能（0）を参照してください。
また、このときサスペンド終了指示であるレジューム（FS-K）を検出するために、F/W にて SIE_IntEnb.EnNonJ ビットをセットし、NonJ 割り込みを許可してください。

A2.6.2.7.7　IN_SUSPEND

NonJ 割り込みステータス（SIE_IntStat.NonJ）がセットされた場合サスペンドからの復帰指示であると判断し、NegoControl.InSUSPEND ビットを F/W にてクリアすると、CHK_EVENT ステートに入ります。リモート・ウェイクアップ機能を有効にしているアプリケーションで、自発的にサスペンドから復帰する場合には、このステートの中で NegoControl.SendWakeup ビットをセットし、1ms 以上、15ms 以下の間 FS-K を出力してください。

A2.6.2.7.8　CHK_EVENT

USB ケーブル上をチェックし、FS-K を検出した場合レジュームであると判断し、SE0 を検出した場合リセットであると判断します。レジュームと判断した場合は、NegoControl.RestoreUSB ビットをセットし、サスペンド前の転送スピード（USB_Status.FSxHS の値に従う）に戻ります。リセットと判断した場合は、IDLE ステートからの遷移と同じく、一旦イベント検出機能を停止し、WAIT_TIM3US ステートに入ります。
もし、FS-K でも SE0 でも無いステートを検出した場合には、オート・ネゴシエーション・エラー割り込みステータス（SIE_IntStat.AutoNegoErr）ビットをセットし、ERR ステートに入ります。

A2.6.2.7.9　WAIT_RESTORE

SIE_IntStat.RestoreCmp ビットがセットされると、イベント検出機能を有効にし、IDLE ステートに入ります。

A2.6.2.7.10　ERR

一旦このステートに突入すると、オート・ネゴシエーション機能を停止させない限り、このステートから抜けません。このステートは、USB 規格上在りえません。

なお、どのステートにおいても、USB ケーブルが抜かれたことによる判断を行っていないため、もし USB ケーブルが抜かれた場合には、すぐにオート・ネゴシエーション機能を停止してください。

A2.6.2.7.11　各ネゴシエーション機能の単体説明

A2.6.2.7.11.1　サスペンド検出（HS モード）

本 LSI が HS モードで動作しているときに、3ms 以上送受信が何も検出されなかった場合（T1）、FS モードに自動的に移行します（HS のターミネーションを無効にし、FS のターミネーション（Rpu）を有効にします）。この動作により DP は“H”になり、USB_Status.LineState[1:0]ビットで“J”を確認することができます（もし、“SE0”を検出した場合は、リセット（後述）となることに注意）。その後 T2 の時点で依然“J”が検出された場合、SIE_IntStat.DetectSUSPEND ビットがセットされます。このとき、SIE_IntEnb.EnDetectSUSPEND ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされますので、USB のサスペンドステートであると判断します。以下の図では、スヌーズを行ったときの動作を表しています）。

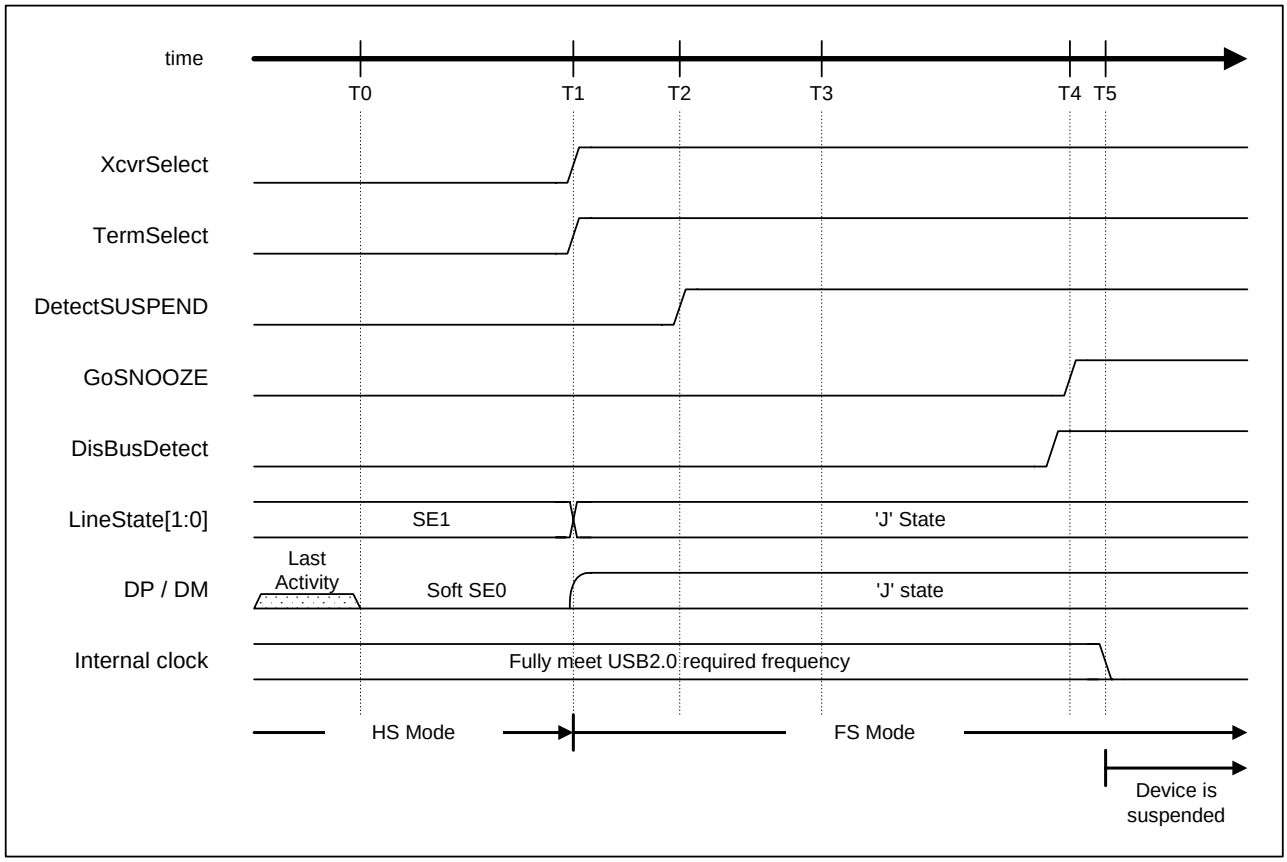


Figure A2.6.8　Suspend Timing (HS mode)

Table A2.6.5　Suspend Timing Values (HS mode)

Timing Parameter	Description	Values
T0	最後のバス・アクティビティ	0(reference)
T1	この時点で依然バス・アクティビティが無い場合、XcvrSelect、TermSelect を“1”にセットし、HS モードから FS モードに切り替える。	HS Reset T0 + 3.0ms < T1 {TWTREV} < HS Reset T0 + 3.125ms
T2	LineState[1:0]をサンプリングする。このとき“J”なら、DetectSUSPEND が“1”になり、USB のサスペンドステートと判断する。	T1 + 100µs < T2 {TWTWRSTHS} < T1 + 875µs
T3	これより前では、RESUME を発行してはいけない。	HS Reset T0 + 5ms {TWTRSM}
T4	完全にスヌーズに移行。これ以降は VBUS から USB で規定されたサスペンド電流以上を引っ張ってはいけない。（スヌーズ移行前に、DisBusDetect を“1”にセット）	HS Reset T0 + 10ms {T2SUSP}
T5	内部クロックが完全停止	T5 < T4 + 10µs

注：　{ }は、USB2.0 規格書で規格されている名称。

A2.6.2.7.11.2　サスペンド検出（FS モード）

本 LSI が FS モードで動作しているときに、3ms 以上送受信が何も検出されなかった場合、または USB_Status.LineState[1:0]ビットに“J”を検出し続け（T1）、さらに T2 の時点で依然“J”が検出された場合、USB のサスペンドステートであると判断し、SIE_IntStat.DetectSUSPEND ビットがセットされます。
このとき、SIE_IntEnb.EnDetectSUSPEND ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされます。以下の図では、スヌーズを行ったときの動作を表しています。

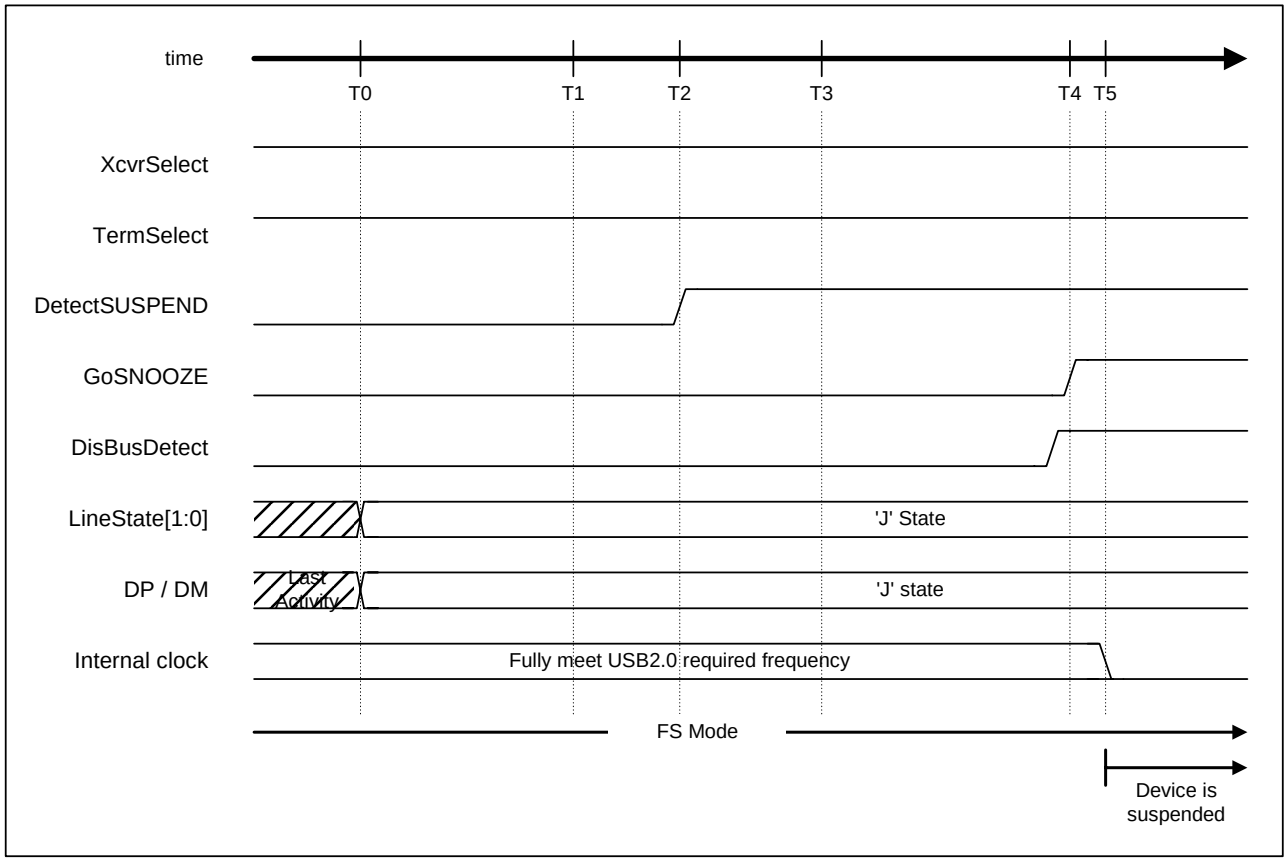


Figure A2.6.9　Suspend Timing (FS mode)

Table A2.6.6　Suspend Timing Values (FS mode)

Timing Parameter	Description	Values
T0	最後のバス・アクティビティ	0(reference)
T1	この時点で依然バス・アクティビティが無い。	T0 + 3.0ms < T1 {TWTREV} < T0 + 3.125ms
T2	LineState[1:0]をサンプリングする。このとき“J”なら、DetectSUSPEND が“1”になり、USB のサスペンドステートと判断する。	T1 + 100µs < T2 {TWTWRSTHS} < T1 + 875µs
T3	これより前では、RESUME を発行してはいけない。	T0 + 5ms {TWTRSM}
T4	完全にスヌーズに移行。これ以降は VBUS から USB で規定されたサスペンド電流以上を引っ張ってはいけない。（スヌーズ移行前に、DisBusDetect を“1”にセット）	T0 + 10ms {T2SUSP}
T5	内部クロックが完全停止。	T5 < T4 + 10µs

注：{ }は、USB2.0 規格書で規格されている名称。

A2.6.2.7.11.3　リセット検出（HS モード）

本 LSI が HS モードで動作しているときに、3ms 以上送受信が何も検出されなかった場合、FS モードに自動的に移行します（HS のターミネーションを無効にし、FS のターミネーション（Rpu）を有効にします)。この動作が行われても DP ラインは“L”になったままで、この結果 USB_Status.LineState[1:0] ビットでも“SE0”を検出することができます。T2 の時点で依然“SE0”が検出された場合には、SIE_IntStat.DetectRESET ビットがセットされます。
このとき、SIE_IntEnb.EnDetectRESET ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされますので、リセットの指示であると判断し、以降は、NegoControl.DisBusDetect ビットをセットした後に、HS Detection Handshake（後述）を行ってください。

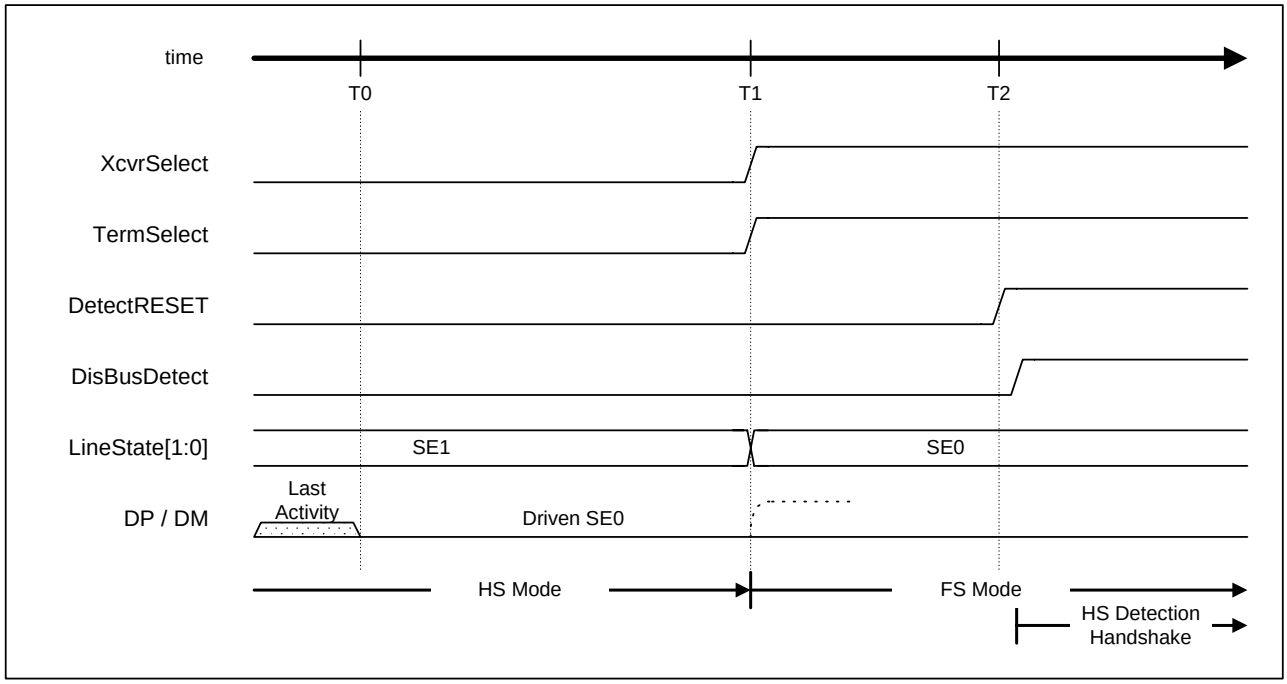


Figure A2.6.10　Reset Timing (HS mode)

Table A2.6.7　Reset Timing Values (HS mode)

Timing Parameter	Description	Values
T0	最後のバス・アクティビティ	0(reference)
T1	この時点で依然バス・アクティビティが無い場合、XcvrSelect、TermSelect を“1”にセットし、HS モードから FS モードに切り替える。	HS Reset T0 + 3.0ms < T1 {TWTREV} < HS Reset T0 + 3.125ms
T2	LineState[1:0]をサンプリングする。このとき“SE0”なら、DetectRESET が“1”になり、リセットの移行と判断する。リセット指示の検出後、DisBusDetect を“1”にセットし、以降 HS Detection Handshake を行う。	T1 + 100µs < T2 {TWTWRSTHS} < T1 + 875µs

注：{ }は、USB2.0 規格書で規格されている名称。

リセット検出（FS モード）

本 LSI が FS モードで動作しているときに、2.5µs 以上 USB_Status.LineState[1:0]ビットに“SE0”を検出し続けた場合には（T1）、SIE_IntStat.DetectRESET ビットがセットされます。
このとき、SIE_IntEnb.EnDetectRESET ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされますので、リセットの指示であると判断し、以降は NegoControl.DisBusDetect ビットをセットした後に、HS Detection Handshake（後述）を行ってください。

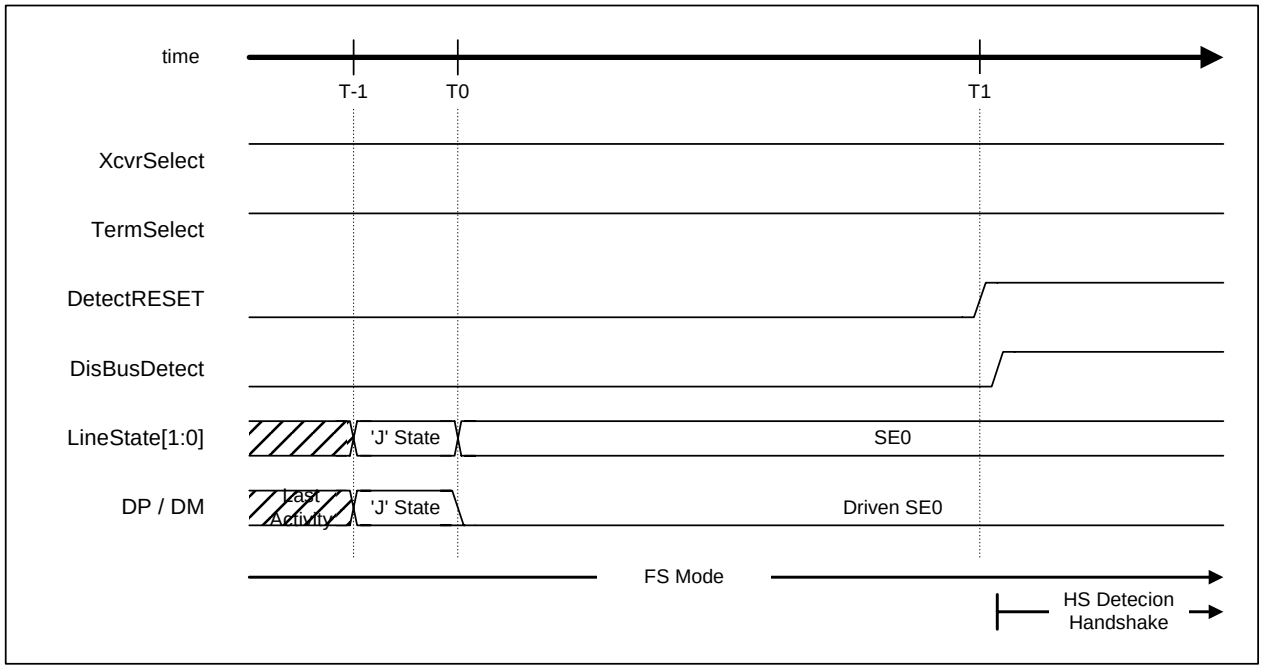


Figure A2.6.11　Reset Timing (FS mode)

Table A2.6.8　Reset Timing Values (FS mode)

Timing Parameter	Description	Values
T-1	最後のバス・アクティビティ	
T0	downstream port からのリセットの指示開始	0(reference)
T1	“SE0”が継続されている場合、DetectRESET が“1”になり、リセットへの移行と判断する。 リセット指示の検出後、DisBusDetect を“1”にセットし、以降 HS Detection Handshake を行う。	HS Reset T0 + 2.5µs < T1 {TWTREV}

注：{ }は、USB2.0 規格書で規格されている名称

A2.6.2.7.11.4　HS Detection Handshake

HS Detection Handshake は、サスペンド中、FS 動作中、或いは HS 動作中の 3 状態のいずれかから、downstream port からの“SE0”のアサートにより開始されます（上記状態からのリセットが開始されたとき）。詳細は、USB2.0 規格書を参照してください。
ここで、上記 3 状態から HS Detection Handshake に移行する方法について説明します。
本 LSI がサスペンド状態では、バス上に“SE0”を検出後直ちに HS Detection Handshake に移行してください。
本 LSI が FS モードで動作している状態では、2.5μs 以上の“SE0”を検出後、HS Detection Handshake に移行してください。
本 LSI が HS モードで動作している状態では、3.0ms 以上の“SE0”を検出後、まず USB のサスペンドステートなのかリセットなのかを判断しなければならない為、一旦 FS モードに切り替わります。このとき動作としては、XcvrControl.XcvrSelect、XcvrControl.TermSelect の両ビットを FS モードに切り替え、HS ターミネーションを無効にし、FS ターミネーションを有効にします。これらのモード切り替えは、3.125ms 以内に行われなければなりません。このモード切り替えから 100μs 以上 875μs 以内に USB_Status.LineState[1:0]ビットをチェックし、“J”なら USB のサスペンドステートとして判断し、“SE0”ならリセットと判断します。このとき、リセットと判断された場合には、その後 HS Detection Handshake に移行してください。
いずれの場合も、リセットは最小 10ms 存在しますが、移行する前の状態（HS もしくは FS）により、タイミングが多少異なります。ここでは、リセットが開始された時間を“HS Reset T0”として定義し、以降は、この“HS Reset T0”からの動作について説明します。
動作中の場合は内部クロックも充分静定しており問題ありませんが、サスペンド中にスリープ／スヌーズさせていた場合には、リセット検出時には内部クロックが出力されていません。このため HS Detection Handshake が行うために、必ず PM_Control_0.GoActDevice ビットを“1”にセットし、内部クロックを動作させてください。この動作の詳細は、パワーマネージメント機能（0）を参照してください。

A2.6.2.7.11.5　FS のダウンストリームポートに繋がれた場合

本 LSI が、HS をサポートしていない downstream port に接続されたときの動作を示します。HS Detection Handshake の開始時（T0）では、XcvrControl.XcvrSelect と XcvrControl.TermSelect は両ビットともに FS モードでなければいけません（FS ターミネーション、即ち DP のプルアップ抵抗（Rpu）を有効にし、HS ターミネーションを無効にします）。
まず、NegoControl.GoChirp ビットをセットします。すると XcvrControl.OpMode[1:0]ビットが“Disable Bit Stuffing and NRZI encoding”になり、“0”で埋め尽くされたデータが準備されます（T1）。これは、バス上に“HS K”（chirp）を送出するためのものです。また同時に、XcvrControl.XcvrSelect ビットが HS モードに設定され、かつ送信可能状態に設定されることで、downstream port に“HS K”（chirp）が送出されます。送出終了後、downstream port からの chirp を待ちます（T2）。通常、HS をサポートしている downstream port は、T3 から“HS K”“HS J”を連続的に送出してきますが（後述）、downstream port が HS をサポートしていない場合（今回の場合）は、T4 の時点でも chirp を送出してこないため、XcvrControl.XcvrSelect ビットを FS モードに自動的に切り替え、NegoControl.GoChirp ビットがクリアされるとともに USB_Status.FSxHS ビットがセットされ、さらに SIE_IntStat.ChirpCmp ビットがセットされます。
このとき、SIE_IntEnb.EnChirpCmp ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされますので、HS Detection Handshake が終了したと判断してください。

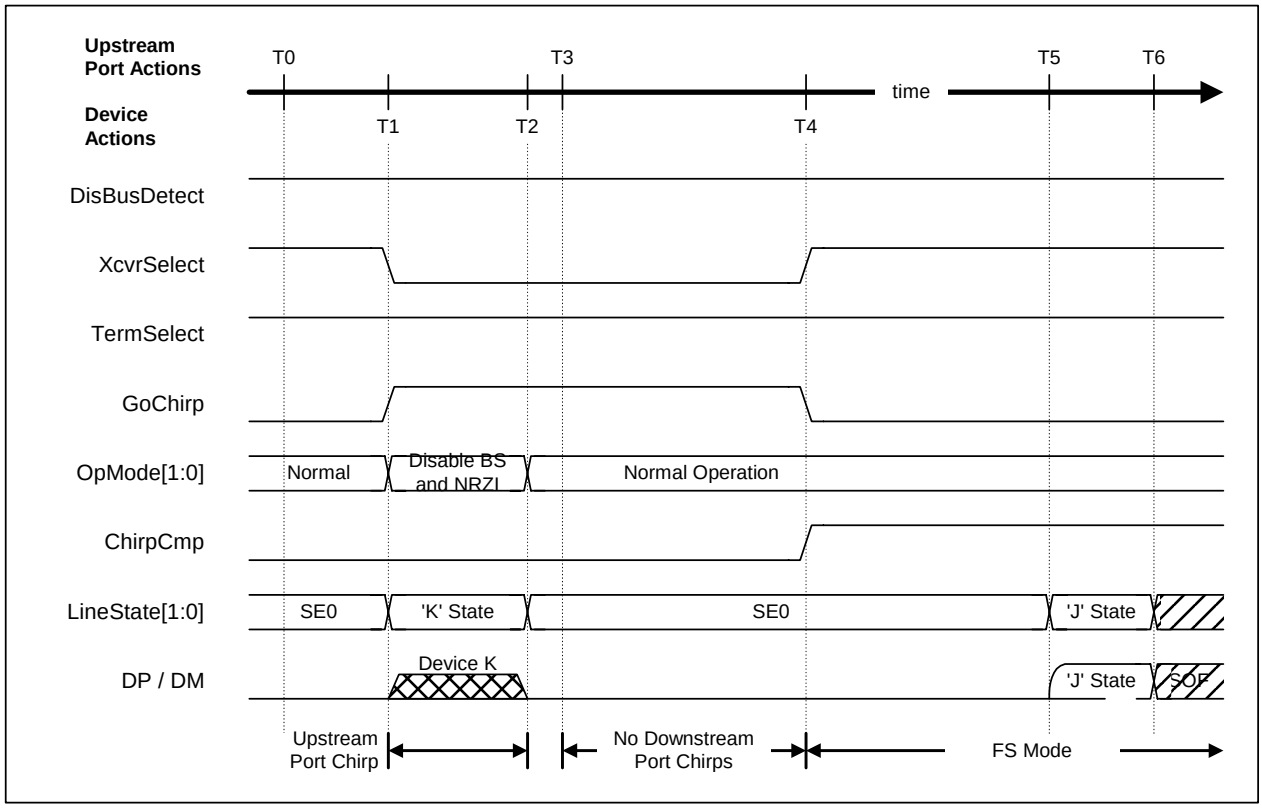


Figure A2.6.12　HS Detection Handshake (FS mode)

Table A2.6.9　HS Detection Handshake Timing Values (FS mode)

Timing Parameter	Description	Values
T0	HS Detection Handshake 開始	0(reference)
T1	HS トランシーバをイネーブルにし、GoChirp を“1”にセットして Chirp K を送出開始。	T0 < T1 < HS Reset T0 + 6.0ms
T2	Chirp K 送出完了。最小 1ms の間は送出しなければならない。	T1 + 1.0ms {TUCH} < T2 HS Reset T0+7.0ms {TUCHEND}
T3	downstread port が HS をサポートしている場合、ここから Chirp K を送出開始する。	T2 < T3 < T2 + 100μs {TWTDCH}
T4	Chirp を検出できない場合、この時点で FS モードに戻り、ChirpCmp が“1”にセットされ、リセットシーケンスが終了するのを待つ。	T2 + 1.0ms < T4 {TWTFS} < T2 + 2.5ms
T5	リセットシーケンスの終了	HS Rest T0 + 10ms {TDRST(MIN)}
T6	FS モードでの通常動作。	T6

注：　{ }は、USB2.0 規格書で規格されている名称。
注：　最小 1ms の Chirp K を生成するために、66000 サイクル（内部クロック 60MHz）で判断する。

A2.6.2.7.11.6　HS のダウンストリームポートに繋がれた場合

本 LSI が、HS をサポートしている downstream port に接続されたときの動作を示します。HS Detection Handshake の開始時（T0）では、XcvrControl.XcvrSelect と XcvrControl.TermSelect は両ビットともに FS モードでなければなりません（FS ターミネーション、即ち DP のプルアップ抵抗（Rpu）を有効にし、HS ターミネーションを無効にします)。
まずは NegoControl.GoChirp ビットをセットします。すると XcvrControl.OpMode[1:0]ビットが“Disable Bit Stuffing and NRZI encoding”になり、“0”で埋め尽くされたデータが準備されます（T1)。これは、バス上に“HS K”（chirp）を送出するためのものです。また同時に、XcvrControl.XcvrSelect ビットが HS モードに設定され、かつ送信可能状態に設定されることで、downstream port に“HS K”（chirp）が送出されます。送出終了後、downstream port からの chirp を待ちます（T2)。ここでは downstream port は HS をサポートしているので、“HS K”（Chirp K)、“HS J”（Chirp J）を交互に連続して送出してきます（T3)。この状態を USB_Status.LineState[1:0]ビットで Chirp K-J-K-J-K-J と最低 6 回検出したところで（T6)、XcvrControl.TermSelect ビットを HS モードに自動的に切り替え（T7)、完全な HS モードに移行します。これと同時に、NegoControl.GoChirp ビットがクリアされるとともに NegoStatus.FSxHS ビットがクリアされ、さらに SIE_IntStat.ChirpCmp ビットがセットされます。
このとき、SIE_IntEnb.EnChirpCmp ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat がセットされている場合には、同時に割込み信号がアサートされますので、HS Detection Handshake が終了したと判断してください。
この downstream port からの Chirp K、Chirp J はバス・アクティビティとして認識し、USB のサスペンドステートと判断しないようにしなければなりません。そこで、HS モードでは、この Chirp K、Chirp J を逐次検出し、内部の Suspend Timer に取り込んでいます。
なお、Chirp K-J-K-J-K-J を検出する為に、USB_Status.LineState[1:0]ビットを使用しています。通常の HS パケットと違い、Chirp K、Chirp J は非常に遅い為、USB_Status.LineState[1:0]ビットを使用できます。しかし、本来のパケット受信時に USB_Status.LineState[1:0]ビットにバスの信号を載せると非常にノイジーな為、XcvrControl.TermSelect ビットが HS モードのとき、バス・アクティビティがあると判断される場合には、USB_Status.LineState[1:0]ビットは“J”を、バス・アクティビティが無いと判断される場合には“SE0”を出力します。
次図で、T6 の時点から Chirp の高さが変わっているのは、XcvrControl.TermSelect ビットによりデバイス側の HS ターミネーションが有効になったことを表しています。通常、XcvrControl.TermSelect が FS モード時の Chirp は約 800mV、XcvrControl.TermSelect ビットが HS モード時の Chirp（HS の通常送受信パケットも同様）では、約 400mV となります。

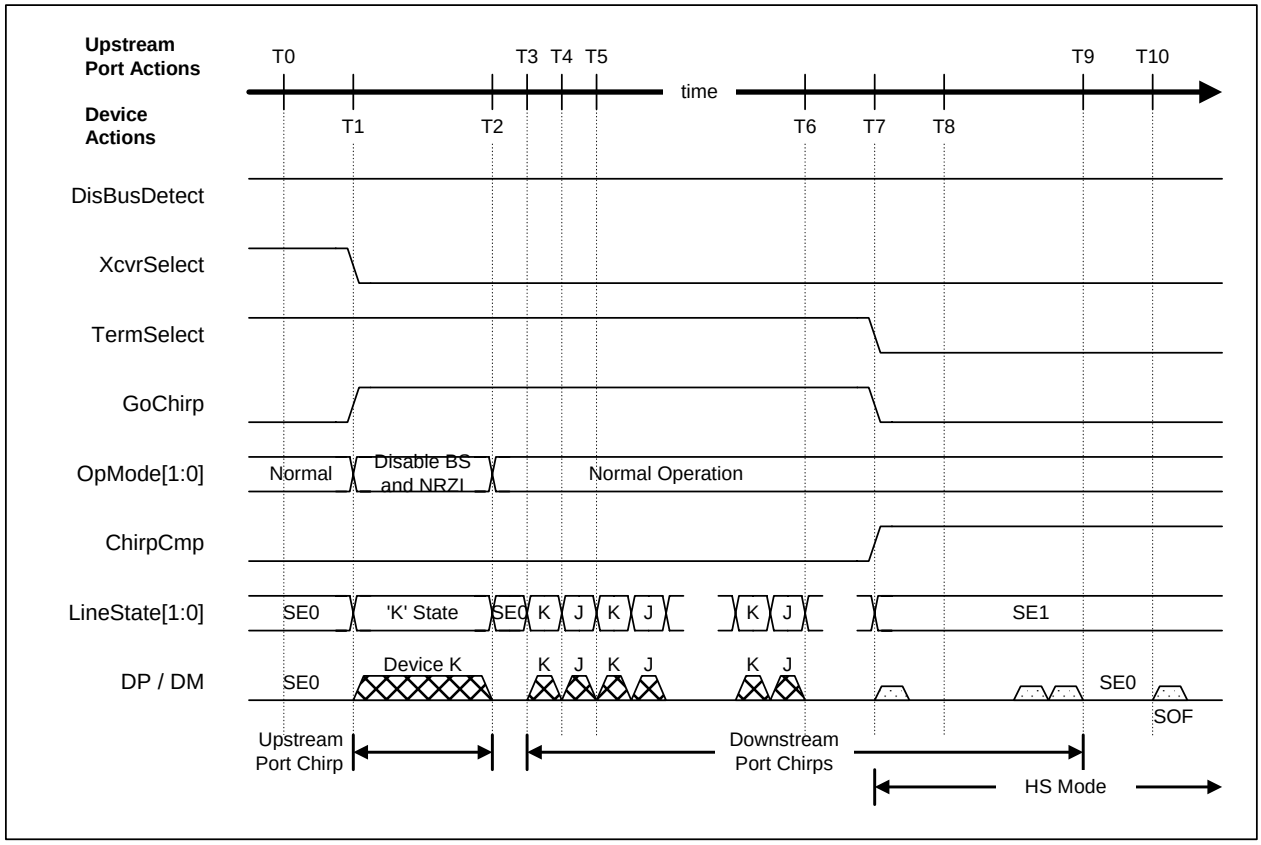


Figure A2.6.13　HS Detection Handshake Timing (HS mode)

Table A2.6.10　HS Detection Handshake Timing Values (HS mode)

Timing Parameter	Description	Values
T0	HS Detection Handshake 開始	0(reference)
T1	HS トランシーバをイネーブルにし、GoChirp を “1” にセットして Chirp K を送出開始。	T0 < T1 < HS Reset T0 + 6.0ms
T2	Chirp K 送出完了。最小 1ms の間は送出しなければならない。	T1 + 1.0ms {TUCH} < T2 HS Reset T0 + 7.0ms {TUCHEND}
T3	downstream port が最初の Chirp K をバスに送出。	T2 < T3 < T2 + 100µs {TWTDCH}
T4	downstream port が Chirp K から Chirp J に切り替え送出。	T3 + 40µs {TDCHBIT(MIN)} < T4 < T3 + 60µs {TDCHBIT(MAX)}
T5	downstream port が Chirp J から Chirp K に切り替え送出。	T4 + 40µs {TDCHBIT(MIN)} < T5 < T4 + 60µs {TDCHBIT(MAX)}
T6	Chirp K-J-K-J-K-J を検出。	T6
T7	Chirp K-J-K-J-K-J を検出したことを受けて、FS ターミネーションを無効に、HS ターミネーションを有効にする。ChirpCmp が “1” にセットされ、リセットの終了を待つ。	T6 < T7 < t6 + 500µs
T8	Chirp K、Chirp J によりバス・アクティビティと認識される。ただし SYNC が検出できないため、パケット受信中と認識されることは無い。	T8
T9	downstream port からの Chirp K、Chirp J の送出終了。	T10 – 500µs {TDCHSE0(MAX)} < T9 < T10 – 100µs {TDCHSE0(MIN)}
T10	リセットシーケンスの終了	HS Rest T0 + 10ms {TDRST(MIN)}

注： { }は、USB2.0 規格書で規格されている名称。
注： 最小 1ms の Chirp K を生成するために、66000 サイクル（内部クロック 60MHz）で判断する。

A2.6.2.7.11.7　スヌーズ中にリセットされた場合

本 LSI は、スヌーズ状態では、内部クロックは出力されていません。ここでは、発振回路は動作しているものとして（スリープ状態ではなく、スヌーズ状態）、そのときの動作を説明します。
スヌーズ状態で、リセットが検出された場合（T0）、SIE_IntStat.NonJ ビットがセットされます。さらに、SIE_IntEnb.EnNonJ ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat ビットがセットされている場合には、同時に割込み信号がアサートされます。このとき、すぐさまスヌーズから復帰しリセットシーケンスに移行させる為に、PM_Control_0.GoActDevice ビットを“1”にセットしてください（T1）。PLL パワーアップ時間経過後（T2）、PM_Control_1.PM_State[2:0]が“ACT_DEVICE”になり、内部クロックが出力され始めます。この後 HS Detection Handshake（前述）を行ってください。
このとき、発振回路を停めていなければ（スリープ状態からの復帰でなければ）、内部クロックは USB2.0 規格に則った周波数精度で出力されています。

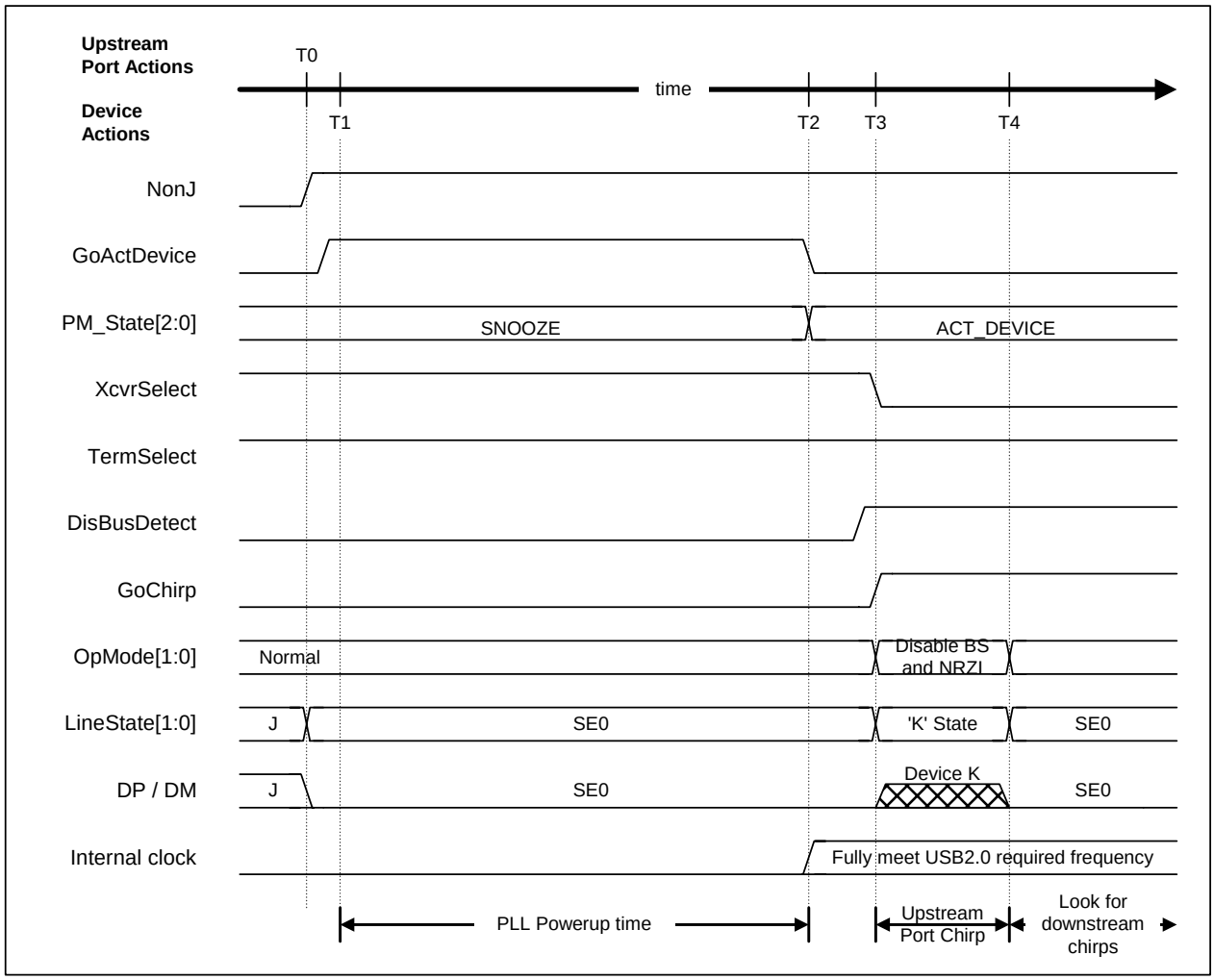


Figure A2.6.14　HS Detection Handshake Timing from Suspend

Table A2.6.11　HS Detection Handshake Timing Values from Suspend

Timing Parameter	Description	Values
T0	NonJ が“1”にセットされ、LineState[1:0]で“SE0”を確認すると、スヌーズ中のリセットを検出。	0（HS Reset T0）
T1	リセット検出後、GoActDevice を“1”にセット。	T1
T2	PM_State が“ACT_DEVICE”になる。内部クロック出力安定。	T1 + 250μs < T2
T3	GoChirp を“1”にセットし、Chirp K をバスに送出。 （Chirp K 送出前には DisBusDetect を“1”にセットする）	T2 < T3 < HS Reset T0 + 5.8ms
T4	Chirp K 送出を終了。	T3 + 1.0ms {TUCH} < T4 < HS Reset T0 + 7.0ms {TUCHEND}

注： { }は、USB2.0 規格書で規格されている名称。
注： 最小 1ms の Chirp K を生成するために、66000 サイクル（内部クロック 60MHz）で判断する。
注： 発振回路も停止していた場合（スリープ状態）については後述する（PLL パワーアップ時間の他に、OSC パワーアップ時間が必要）。

A2.6.2.7.11.8　レジュームの発行

ここでは、リモート・ウェイクアップがサポートされていて、かつホストからこのリモート・ウェイクアップを有効にされているときに、何らかの要因で、自らレジュームする方法を説明します。ただしリモート・ウェイクアップを行うことが可能なのは、バスがアイドルになってから少なくとも 5ms 経過してからでなくてはなりません。さらに、レジューム信号を出力してから 10ms 経過以前は、USB のサスペンド状態に入る前の電流を VBUS から引っ張ることはできません。
デバイスは、リモート・ウェイクアップする為に、まずスリープ／スヌーズから復帰します。SIE_IntEnb.EnNonJ ビットをクリアし、PM_Control_0.GoActDevice ビットをセットし（T0）、PLL パワーアップ時間経過後（T1）、PM_Control_1.PM_State[2:0]ビットが“ACT_DEVICE”になると同時に内部クロックが出力され始めます。このとき、発振回路を停めていなければ、この内部クロックは、USB2.0 規格に則った周波数精度で出力されています。
その後、NegoControl.SendWakeup ビットをセットし、レジューム信号を送出します（T2）。このとき、内部では、XcvrControl.OpMode[1:0]を“Disable Bit Stuffing and NRZI encoding”に設定し、送信データとして“0”を準備し、パケット送信状態にして、“K”（Resume 信号）を送出します。downstream port は、このレジューム信号を検出し、バス上に“K”（レジューム信号）を返してきます（T3）。
レジューム信号を送出し始めてから約 1ms 後、NegoControl.SendWakeup ビットをクリアすることによってバスに送出していたレジューム信号が停止されますが（T4）、この時点では downstream port が依然バスをレジューム信号にホールドしています。
そこで、NegoControl.RestoreUSB ビットをセットします。一定時間経過後、downstream port はレジューム信号の送出を停止し（T5）、2 ビットの LS-EOP（2*SE0）を送出し、USB のサスペンド以前のスピードモードに切り替わります。これを検出した（“K”でなくなった）ところで、XcvrControl.XcvrSelect、XcvrControl.TermSelect の両ビットが所望のモード（今回の場合 HS モード）に切り替えられ、NegoControl.RestoreUSB ビットがクリアされるとともに SIE_IntStat.RestoreCmp ビットがセットされます。このとき、SIE_IntEnb.EnRestoreCmp ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat ビットがセットされている場合には、同時に割込み信号がアサートされます。
ここで、USB のサスペンド開始時には、スピードモード（HS 又は FS）を、USB_Status.FSxHS ビットにて保存しており、レジュームにて復帰する場合には、この USB_Status.FSxHS ビットが示すモードに戻ります。このとき、レジュームごとに HS Detection Handshake の必要はありません。ここでは USB のサスペンド以前のモードが HS モードであった場合についてのみ説明していることに注意してください。実際、FS モードのときには、T5 以降が通常の FS モードとなり、特に大きなシーケンスの変化はありません。
本 LSI が、スヌーズ状態（PM_Control_1.PM_State[2:0]ビットが“SNOOZE”）では、内部クロックは出力されていません。ここでは、発振回路は動作しているものとして（スリープ状態ではなく、スヌーズ状態）、そのときの動作を説明しています。

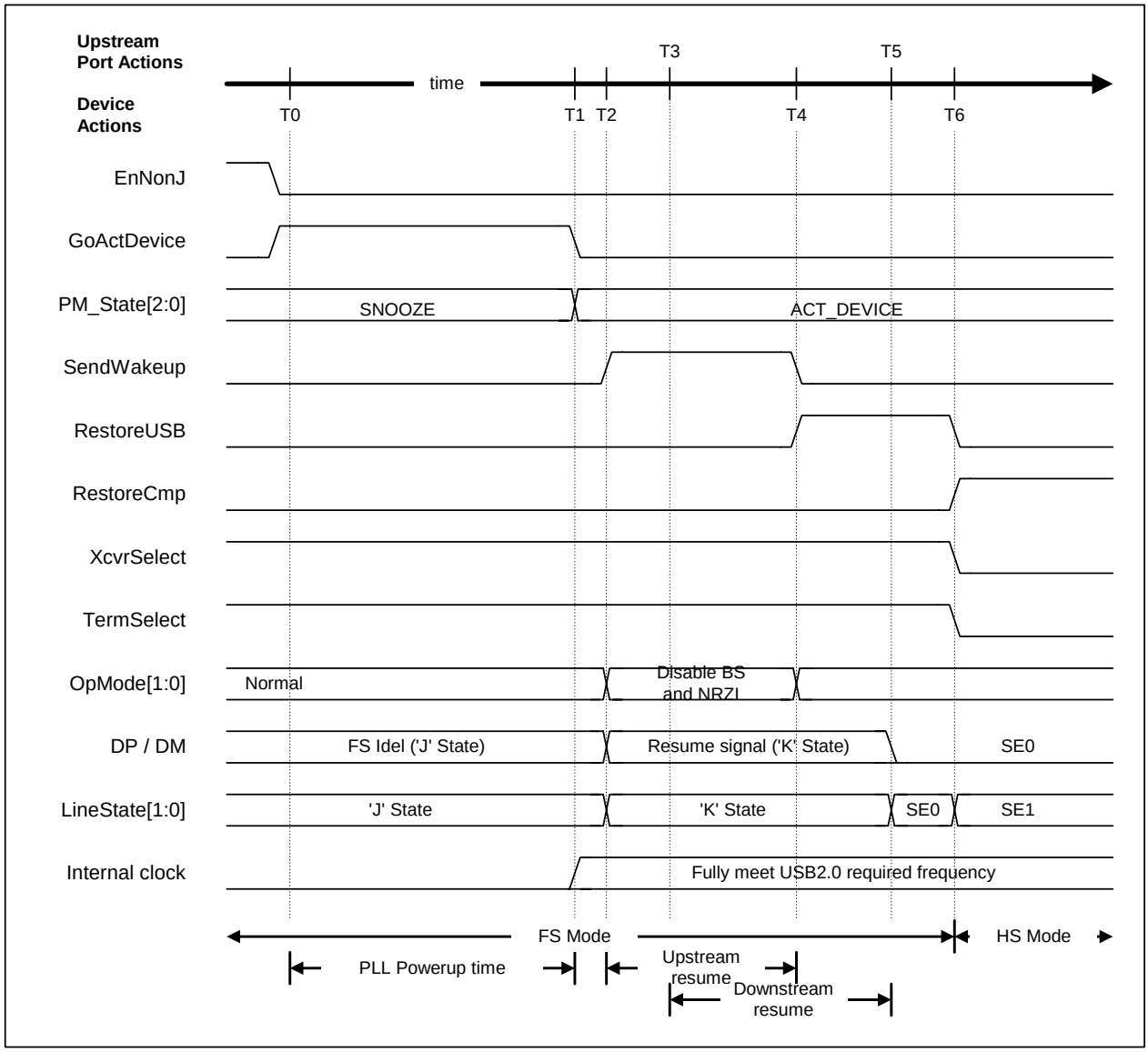


Figure A2.6.15　Assert Resume Timing (HS mode)

Table A2.6.12　Assert Resume Timing Values (HS mode)

Timing Parameter	Description	Values
T0	レジューム開始。GoActDevice を“1”にセット。 （レジューム開始前に EnNonJ を“0”にクリアすること）	0（reference）
T1	PM_State が“ACT_DEVICE”になる。内部クロック出力の安定。	T0 + 250μs < T1
T2	SendWakeup を“1”にセットし、FS の“K”を送出開始。ここで、10ms 以内は USB のサスペンド以前の電流を引っ張ってはいけない。	T0 < T2 < T0 + 10ms
T3	downstream port が FS の“K”を返す。	T2 < T3 < T2 + 1.0ms
T4	SendWakeup を“0”にクリアし、FS の“K”送出を終了。LineState[1:0]により“K”を確認後、RestoreUSB を“1”にセットする。	T2 + 1.0ms {TDRSMUP(MIN)} < T4 < T2 + 15ms {TDRSMUP(MAX)}
T5	downstream port が FS の“K”送出を終了。	T2 + 20ms {TDRSMDN}
T6	RestoreCmp が“1”になる。USB のサスペンド以前が HS モードであった場合、自動的に HS モードに移行。	T5 + 1.33μs {2 Low-speed bit time}

注：{ }は、USB2.0 規格書で規格されている名称。

A2.6.2.7.11.9　レジュームの検出

本 LSI がスヌーズしているとき、バス上には“J”（USB_Status.LineState[1:0]は“J”）が観測されます。バス上に“K”が観測されたときは、downstream port からのウェイクアップの指示（レジューム指示）を受け取ったことになります（T0）。このとき、発振回路が動作を停止していなければ（スリープ状態でなければ）、SIE_IntStat.NonJ ビットがセットされます。このとき、SIE_IntEnb.EnNonJ ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat ビットがセットされている場合には、同時に割込み信号がアサートされます。
まず、PM_Control_0.GoActDevice ビットを“1”にセットし（T1）、PLL パワーアップ時間経過後（T2）、PM_Control_1.PM_State[2:0]が“ACT_DEVICE”になると同時に内部クロックが出力され始めます。このとき、発振回路を停めていなければ、この内部クロックは USB2.0 規格に則った周波数精度で出力されています。
そこで、NegoControl.RestoreUSB をセットします。一定時間経過後、downstream port はレジューム信号の送出を停止し（T3）、USB のサスペンド以前のスピードモードに切り替わります。これを検出した（“K”でなくなった）ところで、XcvrControl.XcvrSelect、XcvrControl.TermSelect の両ビットが所望のモード（今回の場合 HS モード）に切り替えられ、NegoControl.RestoreUSB ビットがクリアされるとともに SIE_IntStat.RestoreCmp ビットがセットされます。このとき、SIE_IntEnb.EnRestoreCmp ビットおよび DeviceIntEnb.EnSIE_IntStat ビットがセットされ、MainIntEnb.EnDeviceIntStat ビットがセットされている場合には、同時に割込み信号がアサートされます。
ここでは、発振回路は動作しているものとして（スリープ状態ではなく、スヌーズ状態）、そのときの動作を説明しています。

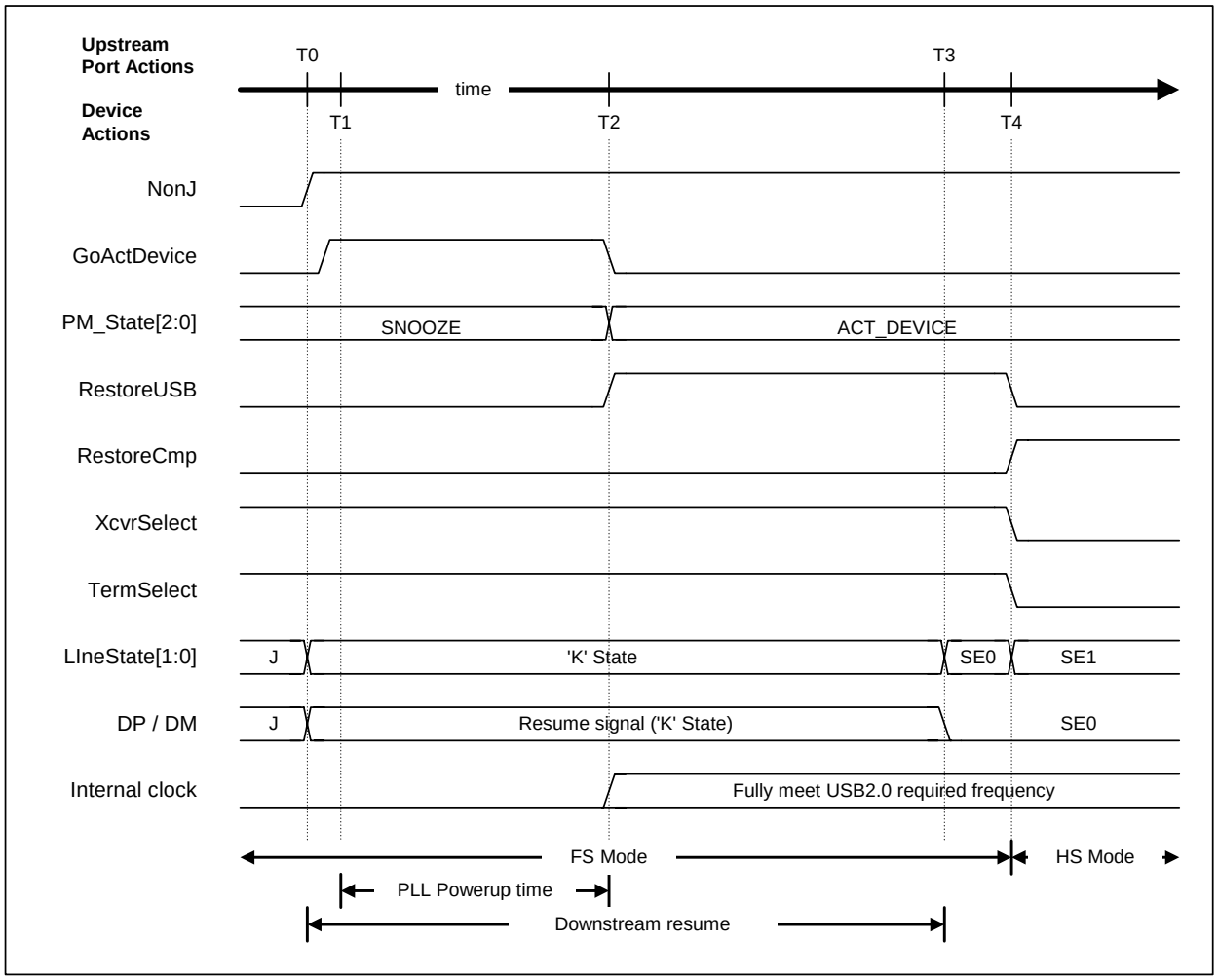


Figure A2.6.16　Detect Resume Timing (HS mode)

Table A2.6.13　Detect Resume Timing Values (HS mode)

Timing Parameter	Description	Values
T0	downstream port が FS の“K”を送出。NonJ が“1”になる。	0（reference）
T1	GoActDevice を“1”にセット。	T1
T2	PM_State が“ACT_DEVICE”になる.内部クロック出力安定。LineState[1:0]で“K”を確認後、RestoreUSB を“1”にセットする。	T1 + 250µs < T2
T3	downstream port が FS の“K”送出を終了。同時に downstream port は USB のサスペンド以前の HS モードに移行。	T2 + 20ms {TDRSMDN}
T4	USB のサスペンド以前が HS モードであった場合、自動的に HS モードに移行。	T3 + 1.33µs {2 Low-speed bit time}

注：{ }は、USB2.0 規格書で規格されている名称。

A2.6.2.7.11.10　ケーブル挿入

ここでは、ハブ又はホストに接続された場合、すなわちケーブルが挿入された場合を説明します。
ケーブルが抜かれている状態、もしくは繋がっていない状態に故意にしているときには、XcvrControl.XcvrSelect ビットは FS モード、XcvrControl.TermSelect ビットは HS モードを初期値としてください。
ケーブルが接続されていない状態（T0）でケーブルが接続されると、V_{BUS}が“H”になり、同時に USB_Status.VBUS ビットがセットされます（T1）。このときスヌーズ状態にしていた場合は、PM_Control_0. GoActDevice ビットを“1”にセットし（T2）、PLL パワーアップ時間経過後（T3）、PM_Control_1.PM_State[2:0]が“ACT_DEVICE”になると同時に内部クロックが出力され始めます。その後、まずはFSデバイスが接続されたことにしなければならないため、一旦はFSモードになるために、XcvrControl.TermSelect ビットをFSモードにしてください（T4）。
その後、downstream port はリセットを送出し（T5）、ここから HS Detection Handshake が開始されます。

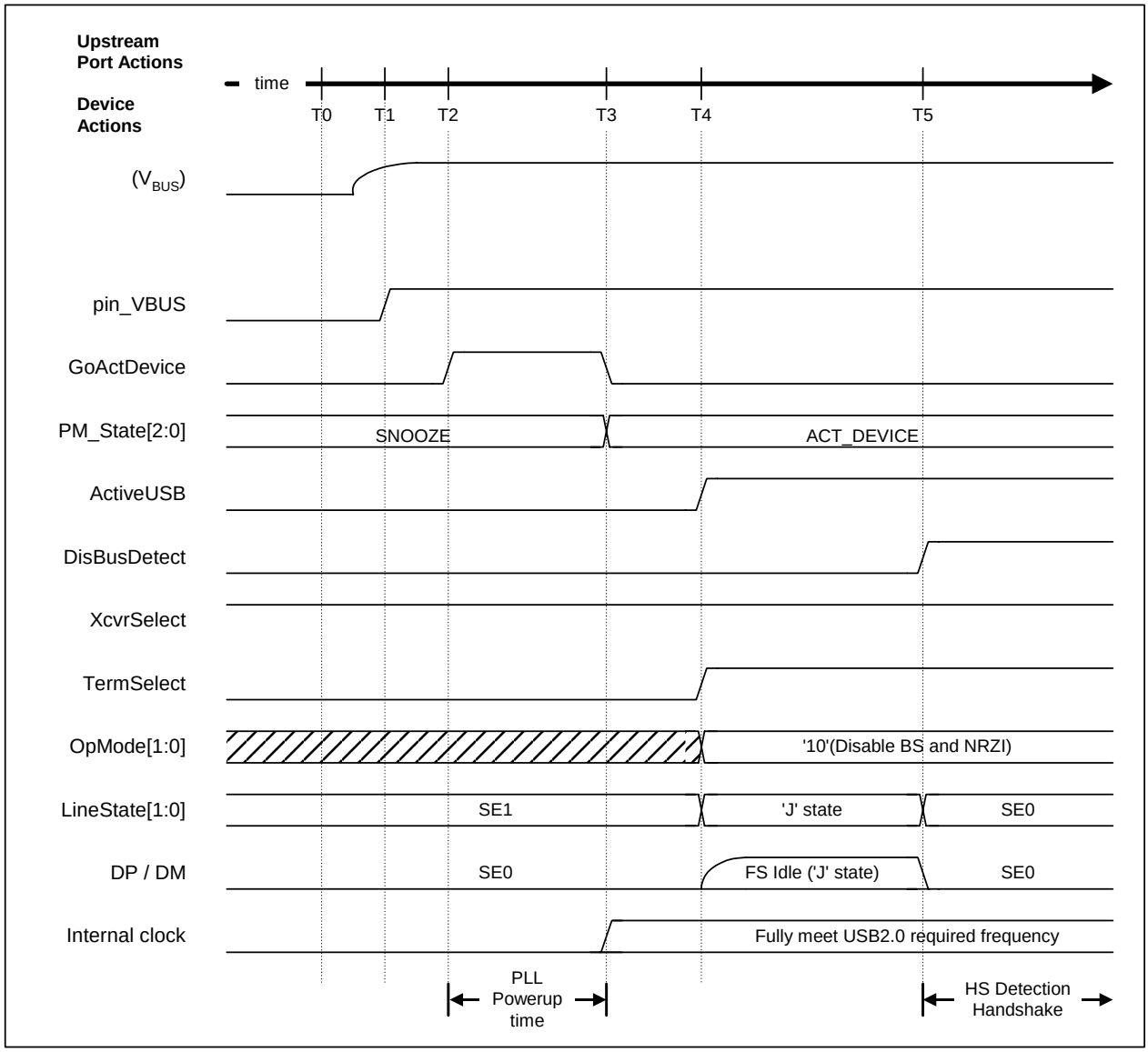


Figure A2.6.17　Device Attach Timing

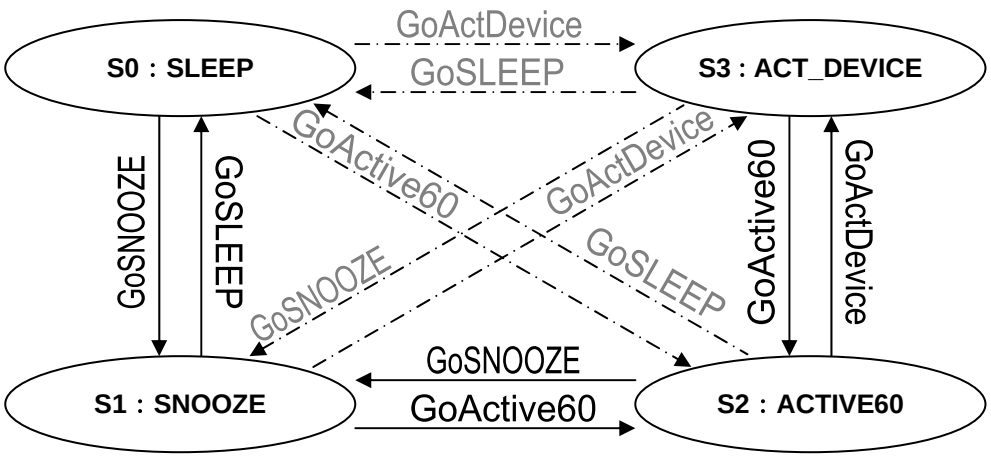
Table A2.6.14　Device Attach Timing Values

Timing Parameter	Description	Values
T0	ケーブルは挿入されていない。	0（reference）
T1	ケーブルが挿入され、入力ピン VBUS が“H”になる。	T1
T2	GoActDevice を“1”にセット。	T2
T3	PM_State が“ACT_DEVICE”になる。内部クロック出力の安定。	T2 + 250μs < T3
T4	ActiveUSB を“1”にセット。TermSelect を“1”にセット。 OpMode[1:0]を“00”に設定。 FS モードに移行。FS ターミネーションが有効。	T1 + 100ms {TSIGATT} < T4
T5	downstream port からリセットが送出される。DisBusDetect を“1”にセット。	T4 + 100ms {TATTDB} < T5

注：　{ }は、USB2.0 規格書で規格されている名称。

A2.6.3　パワーマネージメント機能

オシレータ、PLL（DevicePLL480）の動作を制御し、スリープ、スヌーズ、アクティブ 60、アクトデバイスの４ステートを遷移します。他のステートに遷移する場合には、PM_Control_0.GoSLEEP、PM_Control_0.GoSNOOZE、PM_Control_0.GoActive60、PM_Control_0.GoActDevice ビットをセットすることで開始され、任意の処理が行われた後終了します。現在、どのステートにいるかを確認するには、PM_Control_1.PM_State[3:0]を確認してください。また遷移終了時、MainIntStat.FinishedPM イベントを発生します。このとき、MainIntEnb.EnFinishedPM ビットをセットし、かつ MainIntEnb.EnSIE_IntStat ビットをセットしていれば、割り込みが発生します。
全てのステートから他のステートへ遷移することが可能で、アクトデバイス・ステート時に PM_Control.GoSLEEP ビットをセットした場合は、アクティブ 60・ステート、スヌーズ・ステートを経由し、スリープ・ステートに遷移し、完全に遷移終了した時点で SIE_IntStat.FinishedPM イベントを発生します。またスリープ・ステート時に PM_Control.GoActDevice ビットをセットした場合も、スヌーズ・ステート、アクティブ 60・ステートを経由し、アクトデバイス・ステートに遷移し、完全に遷移終了した時点で MainIntStat.FinishedPM イベントを発生します。同様に、アクティブ 60・ステート時に PM_Control.GoSLEEP ビットをセットした場合も、スヌーズ・ステートを経由し、スリープ・ステートに遷移し、完全に遷移終了した時点で MainIntStat.FinishedPM イベントを発生します。またスリープ・ステート時に PM_Control_0.GoActive60 ビットをセットした場合も、スヌーズ・ステートを経由し、アクティブ 60・ステートに遷移し、完全に遷移終了した時点で MainIntStat.FinishedPM イベントを発生します。



※一点鎖線の遷移は、実際には実線を通って遷移する。

Figure A2.6.18　パワーマネージメント

注）S2S65A30 においては、60MHz 専用の PLL を搭載しておりません。パワーマネージメントの機能として「ACTIVE60」というステートが存在しますが、内部の状態としては「ACTIVE60」ステートと「SNOOZE」ステートとは同じ状態にあります。

A2.6.3.1　SLEEP（スリープ）

オシレータが発振していないステートです。したがいまして、このステートでは、PLL も発振していません。
スヌーズ・ステート、アクティブ 60・ステート、アクトデバイス・ステート中に PM_Control_0.GoSLEEP ビットをセットし、スリープに遷移する場合は、動作している PLL、OSC を停止し、最終的に、OSCCLK の出力を停めた後に発振を停止します。
逆に、スリープ・ステート中に PM_Control_0.GoSNOOZE、PM_Control_0.GoActve60 ビット、PM_Control.GoActDevice ビットをセットし、スリープからスヌーズへ離脱するときには、オシレータが安定して発振するまで内部回路には OSCCLK を与えないように、発振安定時間ゲートしています。この発振安定時間は、発振セル、発振子、周辺回路及び基板によって変化しますので、WakeUpTim_H,L レジスタを用いて設定してください。

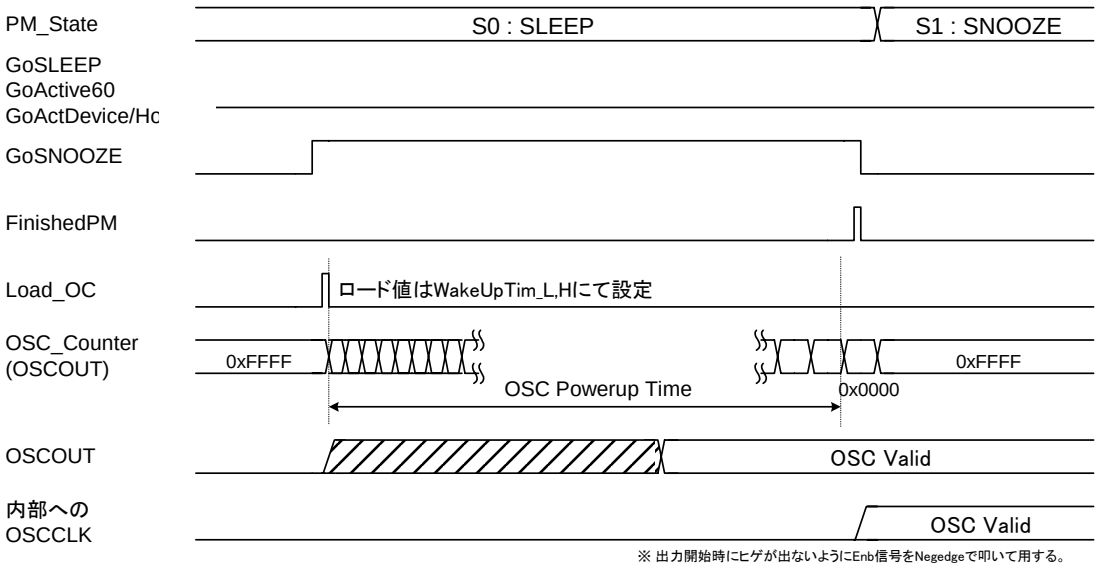


Figure A2.6.19　SLEEP ステートからの離脱（GoSNOOZE 時）

A2.6.3.2　SNOOZE（スヌーズ）

オシレータは発振している状態で、PLL が発振していないステートです。
アクティブ 60・ステート、アクトデバイス・ステート中に PM_Control_0.GoSNOOZE ビットをセットし、スヌーズに遷移する場合は、出力しているクロックを停止した後、DevicePLL480 を停止します。しかし､スヌーズ･ステート中に PM_Control_0.GoActDevice ビット及び PM_Control_0.GoActive60 ビットをセットし、スヌーズからアクティブへ離脱するときには、PLL が安定して発振するまで内部回路には SCLK を与えないように、PLL 安定時間（約 250μs）ゲートします。

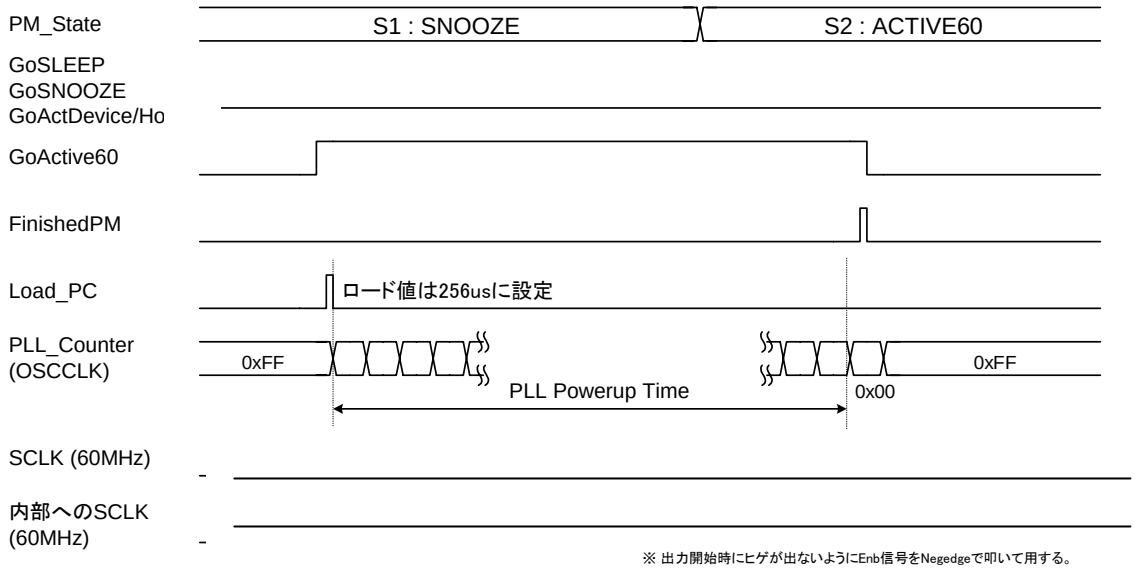


Figure A2.6.20　SNOOZE ステートからの離脱（GoActive60 時）

A2.6.3.3　ACTIVE60（アクティブ 60）

S2S65A30 においては、60MHz 専用の PLL を搭載していないため、「SNOOZE」ステートと同じ状態にあります。ただし、スヌーズ・ステートからアクティブ 60・ステートに遷移するには、
に示すように約 256μs の時間がかかります。

A2.6.3.4　ACT_DEVICE（アクトデバイス）

オシレータ、DevicePLL480 が動作しているステートです。レジスタマップの***斜体太字***のレジスタ及びビットはスヌーズ及びスリープ時でも読み書き可能なレジスタです。***斜体文字***以外のレジスタについては、本ステートにおいてのみ読み書きすることができます。
また、USB デバイス回路はアクトデバイス・ステートでのみ動作します。

A2.6.4　FIFO 管理

A2.6.4.1　FIFO メモリマップ

FIFO のメモリマップを以下に示します。

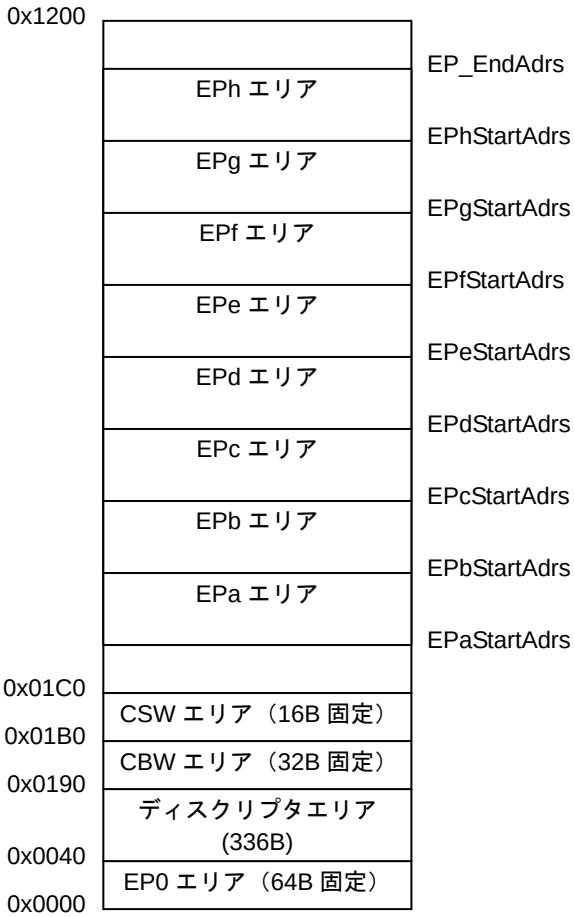


Figure A2.6.21　FIFO メモリマップ例

FIFO のメモリは、EP0 エリア、デスクリプタエリア、CBW エリア、CSW エリア、EPa~h エリアの最大 12 エリアに分割して使用できます。EP0 エリア、デスクリプタエリア、CBW エリア、CSW エリアについては Figure A2.6.21 のように固定領域が割り当てられます。一方その他の EPx{x=a-h}エリアについては FIFO エリア設定レジスタ（EPx{x=a-h}StartAdrs_H,L, EP_EndAdrs_H,L）で領域を柔軟に設定することが可能です。
EP0 エリアは、USB で必須のエンドポイント 0 に使用するエリアで、IN 及び OUT の両方向に使用されます。このエリアには、64 バイトが確保されていますが、そのうち使用できるのは、アドレス 0x000 から始まる、エンドポイント 0 のマックスパケットサイズ分の領域です。したがって、エンドポイント 0 は常にシングルバッファとなります。
デスクリプタエリアは、デスクリプタ返信機能で使用される領域です。336 バイトが確保されており、そのどの位置からでも使用することが出来ます。実際の使用方法については、A2.6.4.2 に後述します。実際には全ての FIFO 領域をデスクリプタ返信機能に使用設定することが可能となっていますが、競合を避けるために、ここで表したエリアをデスクリプタエリアと定め、使用を推奨します。
CBW エリアは、バルクオンリーサポート機能の CBW サポートで使用される領域です。32 バイト確保されていますが、そのうち、アドレス 0x190 から始まる 31 バイトの領域を使用します。実際の使用方法については、A2.6.4.3 に後述します。
CSW エリアは、バルクオンリーサポート機能の CSW サポートで使用される領域です。16 バイト確保されていますが、そのうち、アドレス 0x1B0 から始まる 13 バイトの領域を使用します。実際の使用方法については、A2.6.4.4 に後述します。

EPa~h エリアは、エンドポイント番号と、IN または OUT の方向を任意に設定できる、汎用エンドポイントのエリアです。
EP0 エリア、EPa~h エリアは、FIFO として制御されており、データ格納数が保持されています。この保持された状態をクリアするためには、EPnControl.AllFIFO_Clr ビットか EPnControl.EP0FIFO_Clr ビット、または、各領域に対応した EPrFIFO_Clr.EPx{x=a-h}FIFO_Clr の各ビットをセットして下さい。
なお、これらの状態クリアは、データ保持情報の初期化動作のみを行い、データは書き込みまたはクリアをしません。従って、このビットにより RAM 上のデータがクリアされることはありませんので、デスクリプタエリア内に記録された情報は消えることは無く、クリア後に再度データを書き込み直す必要はありません。

A2.6.4.2　デスクリプタエリアの使用方法

デスクリプタエリアは、デスクリプタ返信機能に使用するエリアです。デスクリプタ返信機能は、エンドポイント 0 において、データステージが IN 転送で行われる場合に使用することが出来ます。
IN 方向のデータステージを行う場合に、この領域内に書き込まれたデータの先頭アドレスと、返信するデータサイズを設定し、デスクリプタ返信機能を実行すると、自動的にデータステージが行われます。
デバイスデスクリプタ等、機器で一意に決定されるような内容を、電源投入後の初期設定時等にこのエリアに書き込んでおくことにより、リクエストを受け付けるとこのエリアのデータを返信するよう指示することが出来ます。リクエスト毎に EP0 エリアにデータを書き込む必要が無く、高速に応答することができます。

A2.6.4.2.1　デスクリプタエリアへのデータの書込み

デスクリプタエリアへデータを書き込むには、RAM_WrDoor 機能を使用します。書き込み開始アドレスを RAM_WrAdrs_H,L レジスタに設定し、RAM_WrDoor_0,1 レジスタにデータを書き込むことによって行います。RAM_WrAdrs_H,L レジスタの値は、一回の書き込み毎に、書き込みデータ数づつ更新されますので、連続したアドレスにデータを書き込む場合には、RAM_WrDoor_0,1 レジスタに連続して書き込むことが出来ます。
なお、RAM_WrDoor_0,1 レジスタは書き込みのみとなっています。

A2.6.4.2.2　デスクリプタエリアでのデータステージ（IN）の実行

書き込まれたデータを、デスクリプタ返信機能にて使用する場合には、DescAdrs_H,L レジスタに、データステージに送信するデータの先頭アドレスを設定し、返信するデータサイズを DescSize_H,L レジスタに設定した後、EP0Control.ReplyDescriptor ビットに 1 をセットします。また、EP0Control.INxOUT ビットに“1”をセットして、IN トランザクションを行えるようにします。また、データステージの IN トランザクションにデータパケットを返信出来るよう、SETUP_Control.ProtectEP0 をクリアしてから、EP0Control_IN.ForceNAK ビットをクリアするようご注意下さい。
設定後、ホストからの IN トランザクションに応答し、マックスパケットサイズ（EP0MaxSize にて設定）に自動的に分割しながら、ホストに DescSize_H,L レジスタに設定されたデータ数までデータパケットを返信します。また、DescSize_H,L レジスタの値がマックスパケットサイズに満たない場合や、分割した後の残りのデータ数がマックスパケットサイズに満たない場合には、自動的にショートパケットとして送信します。
ホストから OUT トランザクションが発行されると、EP0Control.ReplyDescriptor をクリアし、FIFO_IntStat. DescriptorCmp がセットされます。ファームウェアはステータスステージの処理に移行して下さい。

A2.6.4.3　CBW エリアの使用方法

CBW エリアは、バルクオンリーサポート機能の CBW サポートに使用するエリアです。Bulk OUT のエンドポイントで、Bulk Only Transport Protocol のコマンドトランスポートを行うときに、このエリアに受信することが出来ます。こうすることにより、エンドポイントの FIFO にはデータだけを受信することが出来、DMA 等の転送の制御を容易にします。

A2.6.4.3.1　CBW エリアへの受信

CBW サポートを実行しているときに､対象となるエンドポイントで OUT トランザクションが行われ、データサイズが 31 バイトであると、CBW エリアに受信します。データ長が 31 バイト未満で無かった場合は、エラーステータスを発行し、データは破棄します。

A2.6.4.3.2　CBW エリアからのデータの読み出し

CBW エリアに受信したデータを読み出すには、RAM_Rd 機能を使用します。RAM_RdControl.RAM_GoRdCBW_CSW ビットをセットすると、CBW エリアのデータが読み出されて RAM_Rd_00~RAM_Rd_1F レジスタにコピーされ、完了ステータス(CPU_IntStat.RAM_RdCmp ビット)が発行されます。

A2.6.4.4　CSW エリアの使用方法

CSW エリアは、バルクオンリーサポート機能の CSW サポートに使用するエリアです。Bulk IN のエンドポイントで、Bulk Only Transport Protocol のステータストランスポートを行うときに、このエリアから送信することが出来ます。こうすることにより、エンドポイントの FIFO からはデータだけを送信することが出来、DMA 等の転送の制御を容易にします。

A2.6.4.4.1　CSW エリアからの送信

CSW サポートを実行しているときに、対象となるエンドポイントで IN トランザクションが行われると、CSW エリアから 13 バイトのデータをデータパケットとして送信します。

A2.6.4.4.2　CSW エリアへのデータの書き込み

CSW エリアにデータを書き込むには、RAM_WrDoor 機能を使用します。RAM_WrAdrs_H,L レジスタに、CSW エリアの先頭アドレス(0x1B0)を書き、RAM_WrDoor_0,1 レジスタを介して 13 バイトの有効データを書き込みます。CSW エリアは 16Byte 確保されていますので、ワードアクセスで 14 バイト書き込んでも、他の領域を侵すことはなく問題ありません。

A2.6.4.5　FIFO へのアクセス方法

FIFO へのアクセス要因には、CPU(レジスタ), DMA, USB が有ります。

A2.6.4.5.1　FIFO へのアクセス方法（RAM_Rd）

FIFO に CPUIF の RAM_Rd レジスタによってリードアクセスする場合には､読み出しを行いたい FIFO 領域の先頭アドレスとデータサイズを RAM_RdAdrs_H,L レジスタ､RAM_RdCount レジスタに設定し、RAM_RdControl.RAM_GoRd ビットを設定して下さい。指定された FIFO 領域のデータを RAM_Rd レジスタから読み出し可能になると CPU_IntStat.RAM_RdCmp ビットが“1”にセットされます。RAM_RdCmp ビットを確認した後 RAM_Rd レジスタからデータを読み出してください。RAM_Rd レジスタのデータは RAM_Rd_00 から順に格納されます。RAM_RdCount レジスタに設定したサイズが 32 バイトより小さい場合は、設定サイズ以降の RAM_Rd レジスタの値は無効となります。
RAM_Rd レジスタによる FIFO データの読み出しは、チャネルの FIFO 領域の設定に関係なく、任意に行うことが出来ます。
RAM_Rd 機能動作中は、RAM_RdAdrs_H,L レジスタ、RAM_RdCount レジスタの値が逐次更新されます。RAM_Rd 機能を開始した後は CPU_IntStat.RAM_RdCmp ビットがセットされるまで、これらのレジスタへアクセスしないで下さい。RAM_Rd 機能動作中にこれらのレジスタを読み出した場合の値は保証されません。また、これらのレジスタに書き込みを行った場合、誤動作の原因となります。

A2.6.4.5.2　FIFO へのアクセス方法（RAM_WrDoor）

FIFO に CPUIF の RAM_WrDoor_0,1 レジスタによってライトアクセスする場合には、書き込み開始アドレスを RAM_WrAdrs_H,L レジスタに設定し、RAM_WrDoor_0,1 レジスタにてデータを書き込むことによって行います。一回のライトアクセス毎に、自動的に RAM_WrAdrs_H,L レジスタは書き込み数ずつインクリメントされますので、連続したアドレスにデータを書き込む場合には、RAM_WrDoor_0,1 レジスタに連続して書き込むことが出来ます。
RAM_WrDoor_0,1 レジスタによる FIFO への書き込みは、チャネルの FIFO 領域の設定に関係なく、任意に行うことが出来ます。

A2.6.4.5.3　FIFO へのアクセス方法（レジスタアクセス）

FIFO に CPU のレジスタアクセスによってリードアクセスする場合には、いずれか 1 つのチャネルに対して、D_EPx{x=0,a-h}Join.JoinCPU_Rd に 1 をセットし、FIFO_Rd_0,1 レジスタ、または FIFO_ByteRd レジスタにて読み出しを行います。
また、FIFO に CPU のレジスタアクセスによってライトアクセスする場合には、いずれか 1 つのチャネルに対して、D_EPx{x=0,a-h}Join.JoinCPU _Wr に 1 をセットし、FIFO_Wr_0,1 レジスタに書き込みを行います。
FIFO_RdRemain_H,L レジスタは、EPx{x=0,a-h}Join.JoinCPU_Rd にてただ 1 つ設定されたチャネルにおいて、FIFO から読み出し可能なデータの残り数を表しています。また、FIFO_WrRemain_H,L レジスタは、D_EPx{x=0,a-h}Join.JoinCPU_Wr にてただ 1 つ設定されたチャネルにおいて、FIFO に書き込み可能なエリアの残り数を表しています。
ICE 等を使用してファームウェアのデバッグを行うに際し、レジスタのダンプ等を行う場合に、D_EPx{x=0,a-h}Join.JoinCPU_Rd レジスタのいずれかがセットされていると、レジスタのダンプ時に FIFO からデータを読み出されてしまうことに注意してください。

A2.6.4.5.3.1　FIFO アクセス（ライト）

FIFO ライトアクセスは FIFO_Wr_0,1 レジスタへの書き込みを意味します。

FIFO アクセス（ライト）には以下の制限事項があります。

- EPx{x=0,a-h}Join.JoinCPU_Wr ビットを設定した後、FIFO_WrRemain_H,L レジスタで書き込み可能なデータ数を確認した上でアクセスを行って下さい。基本的にワード（2 バイト）単位でアクセスを行って下さい。端数（奇数）バイトの書き込みを行う場合は FIFO のバイト境界を意識してアクセスサイズを制御して下さい。詳細は“FIFO アクセスの端数処理”をご参照下さい。
- FIFO_Wr_0,1 レジスタへのライト直後に FIFO_WrRemain_H,L レジスタを確認しても正確な FIFO の空き領域を確認することが出来ません。必ず 1CPU サイクル以上の間隔を空けて確認してください。

A2.6.4.5.3.2　FIFO アクセス（リード）

FIFO リードアクセスは FIFO_Rd_0,1 レジスタ、FIFO_ByteRd レジスタの読み出しを意味します。

FIFO リードアクセスには、以下の制限事項があります。

- EPx{x=0,a-h}Join.JoinCPU_Wr ビットを設定した後、FIFO_RdRemain_H,L レジスタで読み出し可能データ数及び RdRemainValid ビットを確認した上でアクセスを行って下さい。
- ワード読みを行う場合は FIFO_Rd_0,1 レジスタを用いて行って下さい。バイト読みを行う場合は FIFO_ByteRd レジスタを用いて行って下さい。バイト境界が存在する場合はバイト読みを行って下さい。この場合に FIFO_Rd_0,1 レジスタを用いてワード読みを行った場合は片側にのみ有効なデータが出力されます。詳細は“FIFO アクセスの端数処理”をご参照下さい。

A2.6.4.5.3.3　FIFO アクセスの端数処理

端数（奇数）データを扱う場合の FIFO へのデータの格納状態と FIFO アクセスの関係を説明します。実際の FIFO は 4byte 幅ですが、この章の説明では簡易化のため 2byte 幅で表記致します。4byte/2byte による動作の相違はありません。
【ライト動作】
基本的にはバイト境界の存在しない状態から書き込み動作を行うことを推奨致します。
EPnControl.EP0FIFO_Clr ビットや EPrFIFO_Clr.EPx{x=a-h}FIFO_Clr ビットをセットする等行って、バイト境界の存在しない状態からワード書き込みを行い、奇数データが存在する場合は、連続するデータの最終バイト（データ Z）のみ High 側に書き込んで下さい。この状態を Figure A2.6.22 の（1）に示します。USB 等からはデータ A, B, C, D, ・・・X, Y, Z の順に出力されます。
FIFO にバイト境界がある状態から書き込みを行う場合は、最初に Low 側にデータを書き込み（データ K の書き込み)、バイト境界を解消した後、ワード書き込み（データ L, M）を行って下さい。この状態を Figure A2.6.22 の（2）に示します。
以上は、正常な書き込み動作です。

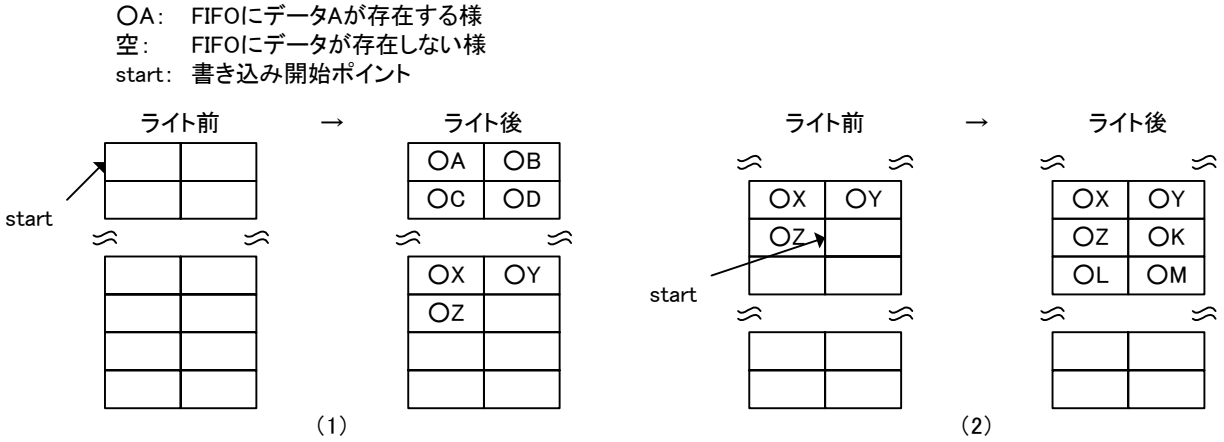


Figure A2.6.22　FIFO ライト処理（正常動作）

以下は注意が必要な書き込みを行った場合の動作です。
FIFO にバイト境界がある状態からワード書き込みを行った場合、High 側の書き込みは無視され Low 側のみ書き込みが行われます（Figure A2.6.23 の（3)）。つまり、Low 側にバイト書き込みを行ったときと同じ動作を行います。また、FIFO にバイト境界のある状態から High 側にのみ書き込みを行った場合、その書き込みは無視されます（Figure A2.6.23 の（4)）。
FIFO にバイト境界が無い状態から Low 側にのみ書き込みを行った場合、その書き込みは無視されます（Figure A2.6.23 の（5)）。また、FIFO にバイト境界が無く、且つ書き込み可能数が“1”の状態からワード書き込みを行った場合、Low 側の書き込みは無視され、High 側のみ書き込みが行われます（Figure A2.6.23 の（6)）。つまり、High 側にバイト書き込みを行ったときと同じ動作を行います。

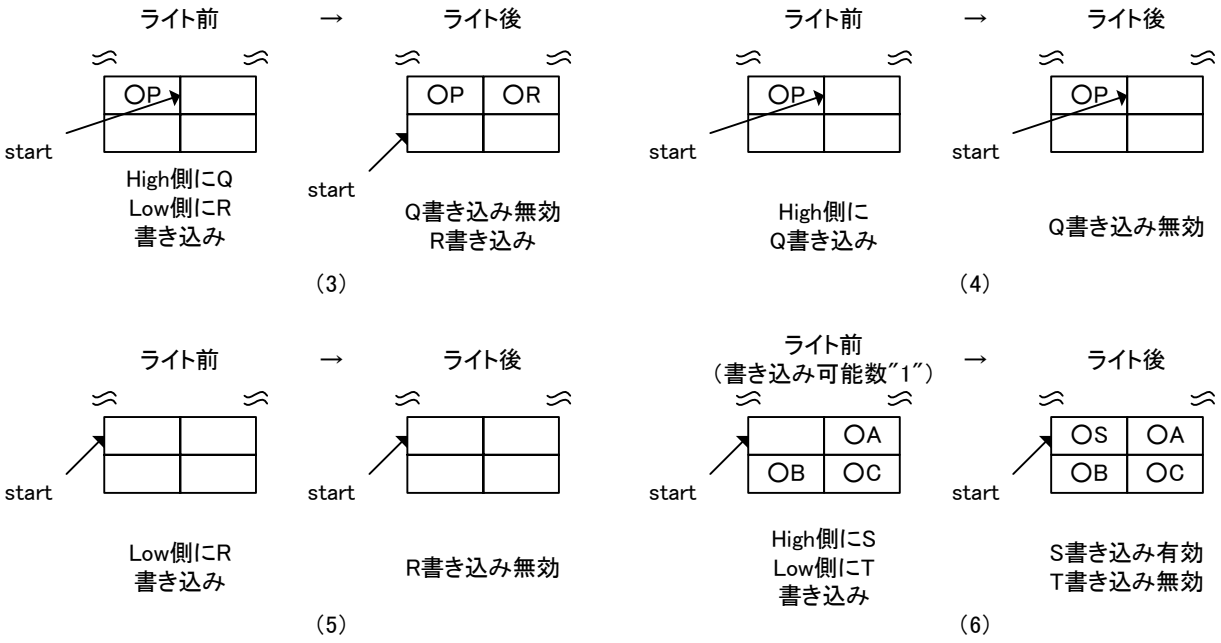


Figure A2.6.23　FIFO ライト処理（注意が必要な動作）

【リード動作】
バイト境界が無い場合は、FIFO_Rd_0,1 レジスタを用いたワード読み出し、FIFO_ByteRd レジスタを用いたバイト読み出し何れを行っても問題ありません。バイト境界がある場合は、FIFO_ByteRd レジスタを用いたバイト読み出しを行って下さい。一旦バイト境界を解消した後はワード読み出し、バイト読み出し何れを行っても問題ありません。
バイト境界が無い状態からのワード読み出しの様子を Figure A2.6.24 の（1）に示します。アクセス毎にデータ A, B→データ C, D と読み出されます。また、バイト読み出しの様子を Figure A2.6.24 の（2）に示します。アクセス毎にデータ A→データ B→データ C→データ D と読み出されます。以上は正常な読み出し動作です。

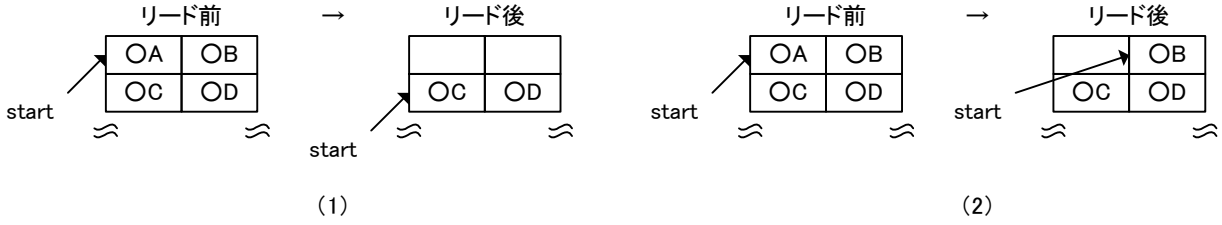


Figure A2.6.24　FIFO リード処理（正常動作）

以下は注意が必要な読み出しを行った場合の動作です。
Figure A2.6.25 の（3）はバイト境界がある状態から、FIFO_Rd_0,1 レジスタを用いてワード読み出しを行った場合の動作です。High 側には不定のデータが出力され、Low 側にデータ J が出力されます。リードのポインタは 1 バイト分のみ進みます。Figure A2.6.25 の（4）はバイト境界が無いが残りデータが 1 バイトの状態から、FIFO_Rd_0,1 レジスタを用いてワード読み出しを行った場合の動作です。High 側にはデータ X が出力され、Low 側には不定データが出力されます。リードのポインタは 1 バイト分のみ進みます。

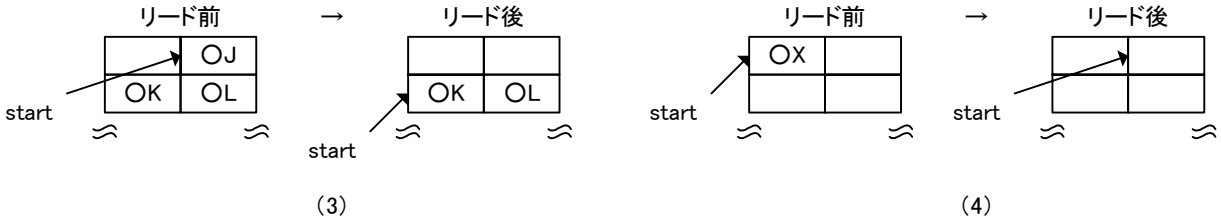


Figure A2.6.25　FIFO リード処理（注意が必要な動作）

上記より、端数処理のリード動作を行う場合の例を説明します。

1） USB から 64 バイト送られてきたデータを 31 バイト→33 バイトで読み出したい場合。
   ① H/W が 64 バイトのレディをラッチして一連のリードシーケンスを開始する。
   ② 30 バイト分のデータを FIFO_Rd_0,1 レジスタでワード読み、もしくは FIFO_ByteRd レジスタでバイト読みする。
   ③ 31 バイト目のデータを FIFO_ByteRd レジスタでバイト読みする。→バイト境界発生。
   ④ 32 バイト目のデータをバイト読みする。この場合 FIFO_ByteRd レジスタでのバイト読みを推奨します。FIFO_Rd_0,1 レジスタでのワード読みを行った場合は Low 側にデータが出力されます。→バイト境界解消。
   ⑤ 残り 32 バイトのデータを FIFO_Rd_0,1 レジスタでワード読み、もしくは FIFO_ByteRd でバイト読みする。
2） JoinCPU_Rd がセットされている状態で USB から 31 バイト、33 バイトと送られてきたデータ 64 バイトを全て FIFO_Rd_0,1 レジスタでワード読みしたい場合。
   ① USB から 31 バイトのデータを受信した時点で H/W は 31 バイトのレディをラッチして一連の動作シーケンスを開始する。
   ② 30 バイト分のデータをワード読みする。
   ③ キャッシュされている 31 バイト目のデータ（バイト境界）を解消するため、一旦ジョインを切り離す。
   ④ 33 バイトのデータが USB から送られてきた後、再度ジョインする。（1+33 バイト）
   ⑤ CPUIF は 34 バイトのレディをラッチして一連の動作シーケンスを開始する。
   ⑥ 34 バイト分のデータをワード読みする。

A2.6.4.5.4　FIFO へのアクセス方法（DMA）

FIFO に DMA アクセスによってリードアクセスする場合には、DMA の各チャネルにつき、EPx{x=0,a-h}Join. JoinDMAx{x=0} ビットによりただ一つのエンドポイントを選択し、DMAx{x=0}_Control.Dir ビットに 1 を設定し、DMA 手順を実行して読み出しを行います。
また、FIFO に DMA アクセスによってライトアクセスする場合には、DMA の各チャネルに付き、EPx{x=0,a-h} Join.JoinDMAx{x=0} ビットによりただ一つのエンドポイントを選択し、DMAx{x=0}_Control.Dir ビットに 0 を設定し、DMA 手順を実行して書き込みを行います。
DMAx{x=0}_Remain_H,L レジスタは、DMA の各チャネルについて、EPx{x=0,a-h}Join. JoinDMAx{x=0} ビットによりただ一つ選択されたエンドポイントにおいて、FIFO から読み出し可能なデータの残り数を表しています。また、DMA の各チャネルについて、EPx{x=0,a-h}Join.JoinDMAx{x=0}ビットによりただ一つ選択されたエンドポイントにおいて、FIFO に書き込み可能なエリアの残り数を表しています。

A2.6.4.5.5　FIFO へのアクセス制限

本 LSI の FIFO には、USB との送受信、CPU バスからのレジスタ書込み、読出し、また DMA による書き込み、読み出しが同時に行われます。また、CPU バスからの読み出しについては、先読み処理を行っています。
これらのことから、それぞれのチャネルにおける FIFO へのアクセス設定方法(Join)に対し、下記の排他ルールがあります。

- 一つのエンドポイントには、JoinCPU_Wr,JoinCPU_Rd,JoinDMAx{x=0}のうちの一つしか設定できない。
- JoinCPU_Wr, JoinCPU_Rd, JoinDMAx{x=0,は、それぞれ、同時に一つのエンドポイントにしか設

定できない。

また、USB からのアクセスに関して、下記の禁止事項があります。
- USB から書き込み中の FIFO エリアに対し、他の要因から書き込みを行ってはいけない。
- USB への読み出し中の FIFO エリアに対し、他の要因から読み出しを行ってはいけない。

例えば、OUT のエンドポイントの FIFO エリアに対し、JoinCPU_Wr を設定して書き込むことが出来ますが、必ず OUT トランザクションが行われない状況で、CPU からの書き込みを行う必要があります。また、IN のエンドポイントの FIFO エリアに対し、JoinCPU_Rd を設定して読み出すことが出来ますが、必ず IN トランザクションが行われない状況で、CPU からの読み出しを行う必要があります。トランザクションが行われない状況には、ActiveUSB ビットがクリアされている場合、各エンドポイントの EnEndpoint がクリアされている場合や、ForceNAK がセットされている場合などがあります。

A2.6.5　DMA

A2.6.5.1　概要

本 USB コントローラには、AHB バス(1)に直接接続されている 8bit の DMA I/F を 1ch.搭載しております。DMA コントローラ 3 と連動させることにより、USB コントローラとメモリ間での DMA 転送を行うことができます。
DMA I/F のポートは、S2S65A30 では AHB バス(1)上の以下のアドレスにマッピングされています。

DMA0　　0xE000_0000（固定）
※USB コントローラの DMA ポートに対しては、アドレス固定モードで転送してください。

A2.6.5.2　基本機能

USB コントローラの DMA I/F には、2 つの動作モード（カウントモード、フリーランモード）があります。
DMA の転送方向に関しては、本仕様書では以下として説明いたします。
DMA Write（ライト）:　S2S65A30 内部/外部メモリ　⇒　USB コントローラ
DMA Read（リード）:　USB コントローラ　⇒　S2S65A30 内部/外部メモリ

A2.6.5.2.1　カウントモード

DMA0_Count[31:0]に設定されたカウント数（バイト数）分の DMA 転送を行います。設定されたカウント数分の DMA 転送が終了すると、DMA_IntStat.DMA0_Cmp の割込み要因が発生し、DMA 回路は停止します。
DMA0_Count レジスタにカウント数をセットした後、DMA0_Control.Go ビットに“1”をセットすると、USB コントローラ内部の FIFO の状態を確認し、DMA コントローラに対してデータ転送要求を発行します。データ転送を行う度に DMA0_Count レジスタの内容をカウントダウン（デクリメント）していき、カウント数が“0”になったところで DMA 転送を停止し、DMA_IntStat.DMA0_Cmp 割り込み要因を発生します。
DMA0_Control.Stop ビットに“1”をセットすることで DMA 転送を停止（中断）させることができます。DMA0_Control.Stop ビットに“1”をセットした直後でも、DMA 転送が完全に停止するまでは DMA0/1_Control.Running ビットに“1”を示します。DMA 転送が完全に停止（中断）すると、Running ビットが“0”になり、DMA_IntStat.DMA0_Cmp 割込み要因が発生しますので、これらのビットを確認してから次の処理を行うようにしてください。

A2.6.5.2.2　フリーランモード

このモードで DMA 転送を実行すると、USB コントローラ内部の FIFO 状態のみを確認し DMA 転送を永久的に継続します。外部の DMA コントローラ側でデータ転送数を管理するような場合、本モードを用いることで設定手順を省略することができます。
DMA0_Config.FreeRun ビットを“1”にセットし、DMA0_Control.Go ビットに“1”をセットすると、USB コントローラ内部の FIFO の状態を確認し、DMA コントローラに対してデータ転送要求を発行します。データ転送を行うたびに DMA0_Count レジスタの内容をカウントアップ（インクリメント）していき、カウント数が“0xFFFF_FFFF”から“0x0000_0000”になったときに DMA_IntStat.DMA0_CountUp 割込み要因を発生します。しかし DMA0/1_CountUp 割込み要因が発生した後も、DMA 転送は継続されます。
DMA0_Control.Stop ビットに“1”をセットすることで DMA 転送を停止（中断）させることができます。DMA0_Control.Stop ビットに“1”をセットした直後でも、DMA 転送が完全に停止するまでは DMA0_Control.Running ビットに“1”を示します。DMA 転送が完全に停止（中断）すると、Running ビットが“0”になり、DMA_IntStat.DMA0_Cmp 割込み要因が発生しますので、これらのビットを確認してから次の処理を行うようにしてください。

DMA コントローラ 3 を利用した DMA 転送手順は、ソフトウェアマニュアルを参照下さい。

A2.6.5.3　強制終了

DMA 転送を強制的に終了したい場合は、DMA0_Config.ActiveDMA ビットを“0”にクリアすることで行うことができます。ただしこの場合には、転送途中の DMA データの連続性が保証されませんので、通常の停止（中断）には DMA0_Control.Stop ビットにより行ってください。強制終了した後に、再度 DMA 転送を行う場合には、USB FIFO の初期化、Join の設定および DMA の設定を再度行ってください。
DMA 転送最中に USB FIFO が初期化されたり、Join の設定が変更された場合には、同様に DMA 転送が一旦強制終了され、DMA0_Config.ActiveDMA が自動的に“0”にクリアされます。

改訂履歴

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.0	2009/08/xx	全ページ	新規	新規制定
Rev 1.1	2010/4/2	p.3 p14,p15	更新	マクロタイプレジスタ 2 および 3 の初期値の誤記を修正

セイコーエプソン株式会社
半導体事業部　IC 営業部

<IC 国内営業グループ>

東京　〒191-8501　東京都日野市日野 421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

大阪　〒541-0059　大阪市中央区博労町 3-5-1　エプソン大阪ビル 15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

ドキュメントコード：411736801
2009 年 8 月 作成 (H)
2010 年 4 月 改訂