お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

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

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

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

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

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

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

RENESAS

アプリケーション・ノート

保守／廃止

μPD72611

PC/AT™ 互換（EISA バス）

デバイス・ドライバ編

SCSI-2 コントローラ

資料番号　S14325JJ1V0AN00（第 1 版）
（旧資料番号　IEA-747）
発行年月　June 1999　N　CP(K)

保守／廃止

〔メ　モ〕

保守／廃止

保守／廃止

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

保守／廃止

PC/ATは米国IBM社の商標です。

MS-DOSは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

- 本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。
- 文書による当社の承諾なしに本資料の転載複製を禁じます。
- 本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。
- 本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。
- 当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「特定水準」に分類しております。また，各品質水準は以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認の上ご使用願います。

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

  特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート／データ・ブック等の資料で，特に品質水準の表示がない場合は標準水準製品であることを表します。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

保守／廃止

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このアプリケーション・ノートは，μPD72611の機能を理解し，それを用いたアプリケーション・システムを設計するユーザのエンジニアを対象とします。

目　的　このアプリケーション・ノートは，μPD72611を用いたPC/AT互換機用評価システムを例に，μPD72611の持つソフトウエア，ハードウエア機能をユーザに理解していただくことを目的としています。

構　成　このアプリケーション・ノートは，大きく分けて次の内容で構成しています。

- システム構成
- μPD72611の使い方
- プログラム機能
- フォーマット・コマンド
- SCSI-HDデバイス・ドライバ
- ASPI（Advanced SCSI Programming Interface）
- SCSIフェーズ処理

読み方　このアプリケーション・ノートの読者は，電気，論理回路，マイクロコンピュータの一般知識を必要とします。

μPD72611を初めて使用される方

→**第3章　μPD72611の使い方**を先に読んでください。

一通りμPD72611の機能を理解しようとするとき

→目次に従って読んでください。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子名称，信号名称の上側に線）
注	：本文中に付けた注の説明
注意	：気を付けて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	：2進数…××××または××××B
	10進数…××××
	16進数…××××H

関連資料　μPD72611に関する資料

- データ・シート（S11157J）
- ユーザーズ・マニュアル（IEU-773）

保守／廃止

保守／廃止

目　　次

図の目次（1/2）

図の目次（2/2）

表の目次（1/3）

表の目次（2/3）

表の目次（3/3）

第1章　概　　要

μPD72611をイニシエータとして動作させる応用例として以下に示すものを製作，作成しました。

- PC/AT互換機（EISAバス）用ホスト・アダプタ・ボード
  （以降μPD72611ホスト・アダプタ・ボードと記します）
- SCSI-HDデバイス・ドライバ（DOS/V用）
- フォーマット・コマンド（DOS/V用）
- ASPIマネージャ（DOS/V用）

なお，ASPIマネージャは米国アダプテック社が推奨するSCSIドライバです（SCSIコントローラおよびフェーズの制御を行います）。SCSI-HDデバイス・ドライバ，フォーマット・コマンドはASPIマネージャを介してターゲットにアクセスします。

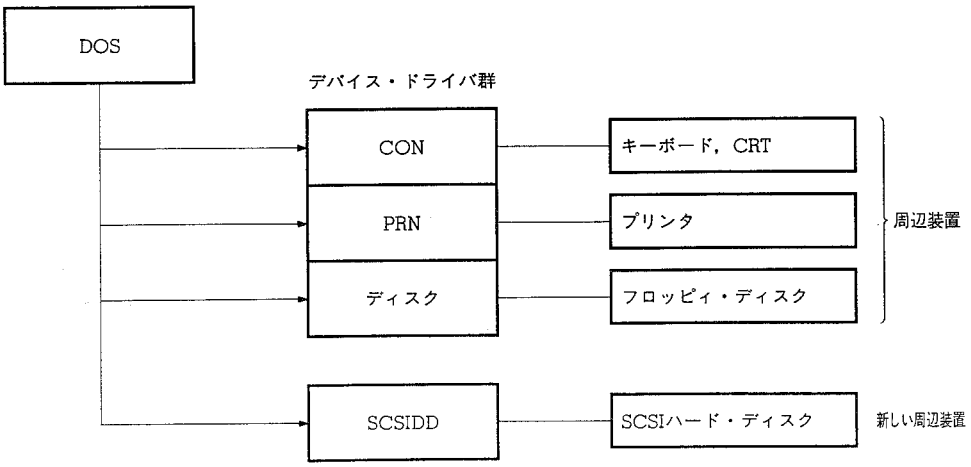
1.1　デバイス・ドライバとは？

デバイス・ドライバとは，図1－1に示すようにDOSとハードウエアを橋渡しする一種のインタフェース・プログラムです。

MS-DOS™が周辺装置を利用する場合はデバイス・ドライバを介して行います。新しい周辺装置用のデバイス・ドライバを追加すれば，DOSはその周辺装置を利用することができます。

今回は，SCSI仕様のハード・ディスクを利用するために，μPD72611デバイス・ドライバを追加しました。

図1－1　DOSとデバイス・ドライバの関係



保守／廃止

第２章　システム構成

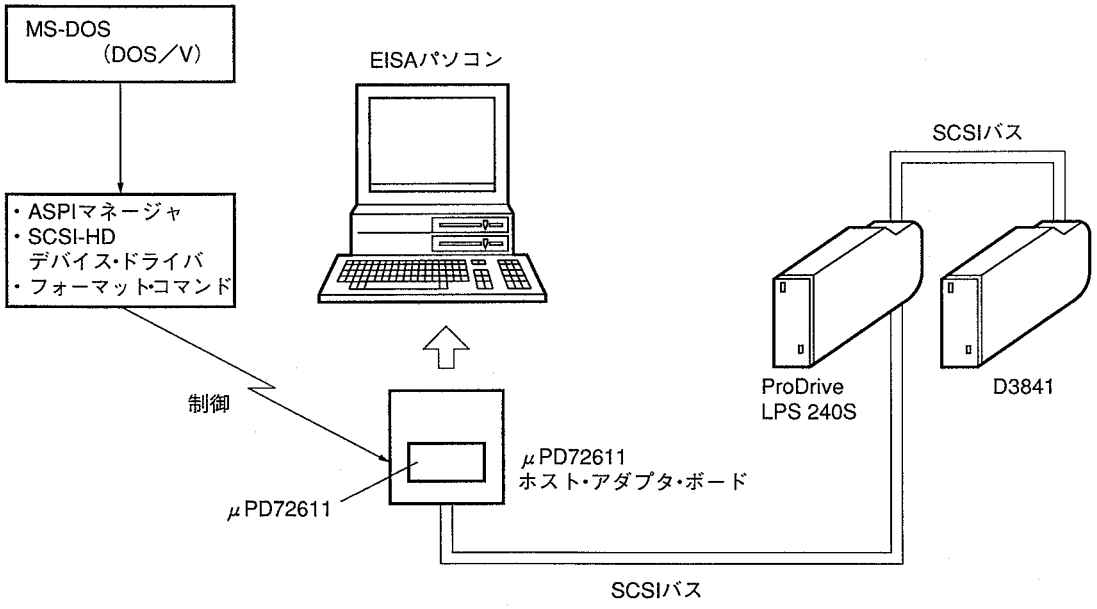
この章ではシステム構成例を説明します。システム構成例を図２－１に示します。

ホスト・システムには拡張バスにEISAバスを搭載したPC/AT互換機(以降EISAパソコンと記します), SCSIハード・ディスクには当社のD3841とQuantum社のProDrive LPS 240Sを使用します。以下に各ハード・ディスクの容量を示します。

D3841　　：40 Mバイト
ProDrive LPS 240S：240 Mバイト

μPD72611を搭載したμPD72611ホスト・アダプタ・ボードによってEISAパソコンとD3841, ProDrive LPS 240Sを接続します。

図２－１　システム構成例



2.1　μPD72611ホスト・アダプタ・ボード

ここでは，μPD72611を使用したホスト・アダプタ・ボードの説明をします。

μPD72611ホスト・アダプタ・ボードは，割り込み処理およびDMA処理にEISAパソコン内にある割り込みコントローラ，DMAコントローラを使用します。

なお，このボードの回路図を**付録Ｂ　回路図**に添付します。

2.1.1　μPD72611

表２－１にμPD72611のI/Oアドレスを示します。LA15-LA4はディップ・スイッチによって決定します。

なお，ディップ・スイッチについては**2.1.5　ディップ・スイッチおよびジャンパ**を参照してください。

表２－１　μPD72611のI/Oアドレス

LA3	LA2	R/W	D31-D24 ($\overline{BE3}$)	D23-D16 ($\overline{BE2}$)	D15-D8 ($\overline{BE1}$)	D7-D0 ($\overline{BE0}$)
0	0	R	ADR	CST	不定	不定
0	0	W	ADR	無効	無効	無効
0	1	R	IST	TP	WIN2	WIN1
0	1	W	CMD	DID	WIN2	WIN1
1	0	R	不定	不定	不定	EXST
1	0	W	無効	無効	無効	無効
1	1	R	DF2			
1	1	W	DF2			

2.1.2　割り込みコントローラ

EISAパソコンの割り込みコントローラの構成を図２－２に示します。μPD72611アダプタ・ボードは割り込みレベル（IRQ3，IRQ4，IRQ5，IRQ7，IRQ10，IRQ11，IRQ12，IRQ15）をディップ・スイッチによって選択できます。ディップ・スイッチについては**2.1.5　ディップ・スイッチおよびジャンパ**を参照してください。

表２－２に割り込みレベルと割り込み要因の関係を示し，表２－３に割り込みコントローラのI/Oアドレスを示します。

なお，表２－３の初期値はBIOSによってイニシャライズされる値です。

保守／廃止

図２－２　割り込みコントローラの構成

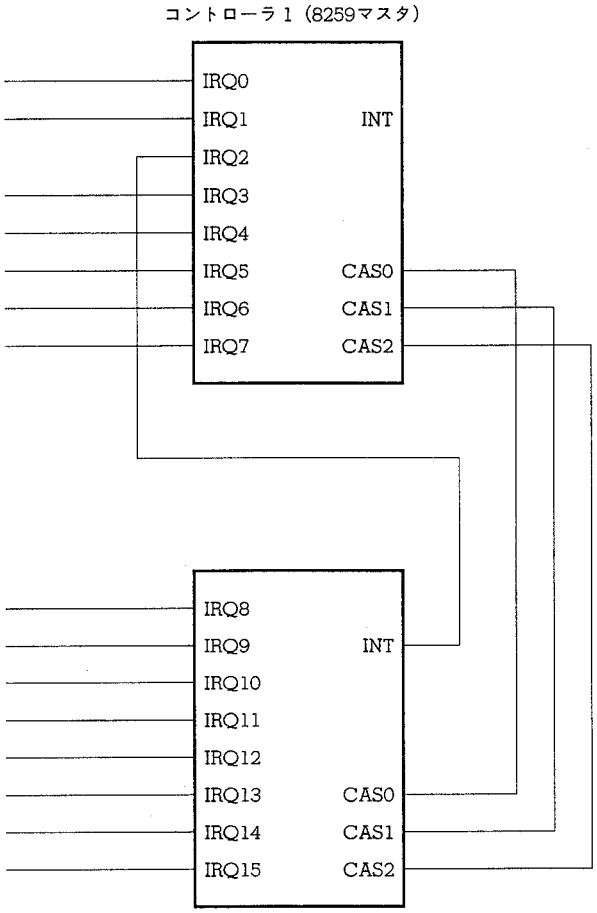


表２－２　割り込みレベルと割り込み要因

プライオリティ	割り込みレベル	コントローラ番号	割り込み要因
1	IRQ0	1	インターバル・タイマ１，カウンタ０出力
2	IRQ1	1	キーボード
3-10	IRQ2	1	コントロール２とのカスケード接続
3	IRQ8	2	リアルタイム・クロック
4	IRQ9	2	拡張バスのB04
5	IRQ10	2	拡張バスのD03
6	IRQ11	2	拡張バスのD04
7	IRQ12	2	拡張バスのD05
8	IRQ13	2	コプロセッサ・エラー，チェイニング
9	IRQ14	2	固定ディスク，拡張バスのD07
10	IRQ15	2	拡張バスのD06
11	IRQ3	1	シリアル・ポート２，拡張バスのB25
12	IRQ4	1	シリアル・ポート１，拡張バスのB24
13	IRQ5	1	パラレル・ポート２，拡張バスのB23
14	IRQ6	1	ディスケット，拡張バスのB22
15	IRQ7	1	パラレル・ポート１，拡張バスのB21

表２－３　割り込みコントローラのI/Oアドレス

I/Oアドレス	初期値	レ　ジ　ス　タ　名
0020H	11H	コントロール１，ICW1
0021H	08H	コントロール１，ICW2
0021H	04H	コントロール１，ICW3
0021H	01H	コントロール１，ICW4
0021H	B8H	コントロール１割り込みマスク
04D0H	00H	コントロール１エッジ／レベル・コントロール
00A0H	11H	コントロール２，ICW1
00A1H	70H	コントロール２，ICW2
00A1H	02H	コントロール２，ICW3
00A1H	01H	コントロール２，ICW4
00A1H	BDH	コントロール２割り込みマスク
04D1H	00H	コントロール２エッジ／レベル・コントロール

保守／廃止

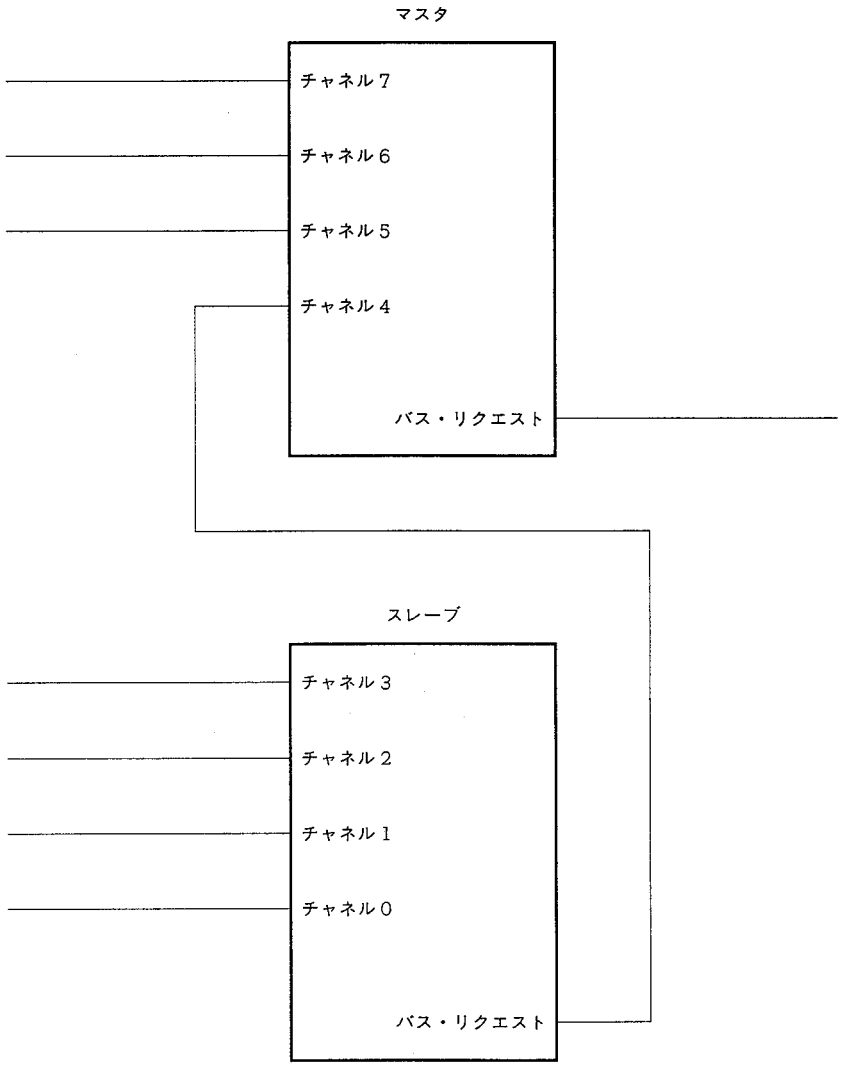
2.1.3　DMAコントローラ

EISAパソコンのDMAコントローラの構成を図２－３に示します。μPD72611アダプタ・ボードはDMAチャネル（チャネル0，チャネル3，チャネル5，チャネル7）をディップ・スイッチによって選択できます。ディップ・スイッチについては**2.1.5　ディップ・スイッチおよびジャンパ**を参照してください。

表２－４および表２－５にDMAコントローラのI/Oアドレスを示します。

なお，表２－４の初期値はリセット後のディフォールト値です。

図２－３　DMAコントローラの構成



保守／廃止

表2－4　DMAコントローラのI/Oアドレス（1/3）

レジスタ／ソフトウエア・コマンド	R/W	チャネル	I/Oアドレス	初期値
コマンド・レジスタ	W	0-3	0008H	00H
		4-7	00D0H	00H
モード・レジスタ	W	0-3	000BH	00H
		4-7	00D6H	00H
拡張モード・レジスタ	W	0-3	040BH	00H
		4-7	04D6H	04H
チェイニング・モード・レジスタ	W	0-3	040AH	00H
		4-7	04D4H	00H
チェイニング・モード・ステータス・レジスタ	R	0-7	04D4H	00H
チャネル・インタラプト・ステータス・レジスタ	R	0-7	040AH	00H
CPU/EISAマスタ・コントロール・レジスタ	R	0-7	040CH	00H
リクエスト・レジスタ	W	0-3	0009H	00H
		4-7	00D2H	00H
ライト・シングル・マスタ・ビット・レジスタ	W	0-3	000AH	不定
		4-7	00D4H	
オール・マスク・ビット・レジスタ	R/W	0-3	000FH	11H
		4-7	00DEH	11H
ステータス・レジスタ	R	0-3	0008H	不定
		4-7	00D0H	
ベース8237コンパチブル・アドレス・レジスタ	W	0	0000H	
		1	0002H	
		2	0004H	
		3	0006H	
		5	00C4H	
		6	00C8H	
		7	00CCH	
ベース・ロウ・ページ・レジスタ	W	0	0087H	
		1	0083H	
		2	0081H	
		3	0082H	
		5	008BH	
		6	0089H	
		7	008AH	

表2-4　DMAコントローラのI/Oアドレス（2/3）

レジスタ／ソフトウエア・コマンド	R/W	チャネル	I/Oアドレス	初期値
ベース・ハイ・ページ・レジスタ	W	0	0487H	不定
		1	0483H	
		2	0481H	
		3	0482H	
		5	048BH	
		6	0489H	
		7	048AH	
カレント8237コンパチブル・アドレス・レジスタ	R/W	0	0000H	
		1	0002H	
		2	0004H	
		3	0006H	
		5	00C4H	
		6	00C8H	
		7	00CCH	
カレント・ベース・ロウ・ページ・レジスタ	R/W	0	0087H	
		1	0083H	
		2	0081H	
		3	0082H	
		5	008BH	
		6	0089H	
		7	008AH	
カレント・ハイ・ページ・レジスタ	R/W	0	0487H	
		1	0483H	
		2	0481H	
		3	0482H	
		5	048BH	
		6	0489H	
		7	048AH	
ベース8237コンパチブル・ワード・カウント・レジスタ	W	0	0001H	
		1	0003H	
		2	0005H	
		3	0007H	
		5	00C6H	
		6	00CAH	
		7	00CEH	

表2－4　DMAコントローラのI/Oアドレス（3/3）

レジスタ／ソフトウエア・コマンド	R/W	チャネル	I/Oアドレス	初期値
ベース・ハイ・ワード・カウント・レジスタ	W	0	0401H	不定
		1	0403H	
		2	0405H	
		3	0407H	
		5	04C6H	
		6	04CAH	
		7	04CEH	
カレント8237コンパチブル・ワード・カウント・レジスタ	R/W	0	0001H	
		1	0003H	
		2	0005H	
		3	0007H	
		5	00C6H	
		6	00CAH	
		7	00CEH	
カレント・ハイ・ワード・カウント・レジスタ	R/W	0	0401H	
		1	0403H	
		2	0405H	
		3	0407H	
		5	04C6H	
		6	04CAH	
		7	04CEH	
ストップ・レジスタ	R/W	0-7	表2－5参照	
クリア・バイト・ポインタ・コマンド	W	0-3	000CH	
		4-7	00D8H	
マスク・クリア・コマンド	W	0-3	000DH	
		4-7	00DAH	
クリア・マスク・レジスタ・コマンド	W	0-3	000EH	
		4-7	00DCH	

表２－５　ストップ・レジスタのI/Oアドレス

DMAチャネル	A23-A16	A15-A8	A7-A0
0	04E2H	04E1H	04E0H
1	04E6H	04E5H	04E4H
2	04EAH	04E9H	04E8H
3	04EEH	04EDH	04ECH
5	04F6H	04F5H	04F4H
6	04FAH	04F9H	04F8H
7	04FEH	04FDH	04FCH

2.1.4　Product IDレジスタ

Product IDレジスタはEISA規格で定められているレジスタです。このレジスタは製造者番号，製品番号，バージョンを指定するものでホスト・アダプタ・ボード上に搭載します。このボードはレジスタの内容をディップ･スイッチによって指定します。表２－６にProduct IDレジスタのI/Oアドレスを示します。

なお，LA15-LA12はディップ・スイッチによって選択できます。ディップ・スイッチについては**2.1.5　ディップ・スイッチおよびジャンパ**を参照してください。

表２－６　Product IDレジスタのI/Oアドレス

I/Oアドレス (LA11-LA2, $\overline{BE3}$-$\overline{BE0}$)	レジスタ名
C80H	Product ID 1stバイト
C81H	Product ID 2ndバイト
C82H	Product ID 3rdバイト
C83H	Product ID 4thバイト

保守／廃止

2.1.5　ディップ・スイッチおよびジャンパ

このボード上に搭載されるディップ・スイッチおよびジャンパについて説明します。

(1) I/Oアドレスの設定

μPD72611のI/Oアドレス設定を表２－７に，Product IDレジスタのI/Oアドレス設定を表２－８に示します。

各表に示すようにμPD72611のI/OアドレスはSW8（LA15-LA12），SW7（LA11-LA4）によって行い，Product IDレジスタのI/OアドレスはSW8（LA15-LA12）によって行います。したがって，μPD72611のI/Oアドレス(LA15-LA12)を決定するとProduct IDのI/Oアドレス(LA15-LA12）も決定されます。

表２－７　μPD72611のI/Oアドレス設定

SW8				SW7								μPD72611のI/Oアドレス	
4	3	2	1	8	7	6	5	4	3	2	1	LA15-LA12	LA11-LA4[注1]
ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	0H[注2]	00H[注2]
ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	0H	01H
ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	ON	0H	02H
ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	0H	03H
…				…								・	…
…				…								・	…
…				…								・	…
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	FH	FDH
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	FH	FEH
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	FH	FFH

注1．LA3，LA2，($\overline{BE3}$-$\overline{BE0}$) はμPD72611の各レジスタに割り当てられます（**表２－１　μPD72611のI/Oアドレス**参照）。

2．16進表記。

保守／廃止

表2－8　Product IDレジスタのI/Oアドレス設定

SW8				Product IDのI/Oアドレス
4	3	2	1	LA15-LA12[注1]
ON	ON	ON	ON	0H[注2]
ON	ON	ON	OFF	1H
ON	ON	OFF	ON	2H
ON	ON	OFF	OFF	3H
…				…
…				…
…				…
OFF	OFF	ON	OFF	DH
OFF	OFF	OFF	ON	EH
OFF	OFF	OFF	OFF	FH

注1．LA11-LA2，($\overline{BE3}$-$\overline{BE0}$)はProduct IDレジスタに割り当てられます（**表2－6　Product IDレジスタのI/Oアドレス**参照）。

2．16進表記。

(2) 割り込みレベルの設定

割り込みレベルの設定を表2－9に示します。

表2－9に示すように割り込みレベルの設定はSW2によって行います。

表2－9　割り込みレベル設定

SW2								割り込みレベル
8	7	6	5	4	3	2	1	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	割り込み未使用
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	IRQ3
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	IRQ4
OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	IRQ5
OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	IRQ7
OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	IRQ10
OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	IRQ11
OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	IRQ12
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	IRQ15

(3) DMAチャネルの設定

DMAチャネルの設定を表２－10に示します。

表２－10に示すようにDMAチャネルの設定はSW1によって行います。

表２－10　DMAチャネル設定

SW1								DMAチャネル
$\overline{DAK}$				DRQ				
8	7	6	5	4	3	2	1	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	DMA未使用
OFF	OFF	OFF	ON	OFF	OFF	OFF	ON	チャネル０
OFF	OFF	ON	OFF	OFF	OFF	ON	OFF	チャネル３
OFF	ON	OFF	OFF	OFF	ON	OFF	OFF	チャネル５
ON	OFF	OFF	OFF	ON	OFF	OFF	OFF	チャネル７

(4) Product IDレジスタの内容設定

Product IDレジスタの内容はSW3-SW6によって設定します。

表２－11に各レジスタとディップ・スイッチの対応を示します。

なお，各ビットはONのとき０，OFFのとき１を示します。

表２－11　Product ID／ディップ・スイッチ対応

ディップ・スイッチ	レジスタ名
SW6	Product ID 1stバイト
SW5	Product ID 2ndバイト
SW4	Product ID 3rdバイト
SW3	Product ID 4thバイト

(5) SCSIバスへの電源供給の設定

SCSIバスのTERMPWR線へ電源供給を行うか否かをJP1によって設定します。

表２－12にジャンパの設定を示します。

表２－12　ジャンパ設定

JP1	内　　容
ショート	SCSIバスのTERMPWR線へ電源供給します
オープン	SCSIバスのTERMPWR線へ電源供給しません

保守／廃止

2.2　D3841ハード・ディスク装置

ここでは，このアプリケーション・ノートで使用した40 Mバイト・ハード・ディスクD3841の仕様（メッセージ，ステータス，コマンド）について説明します。

2.2.1　メッセージ

D3841でサポートしているメッセージを表2－13に示します。

なお，詳細は**D.1　SCSIメッセージ**を参照してください。

表2－13　D3841でサポートしているメッセージ

コード	メッセージ名	転送方向
00H	Command Complete	イニシエータ　←　ターゲット
02H	Save Data Pointer	イニシエータ　←　ターゲット
04H	Disconnect	イニシエータ　←　ターゲット
06H	Abort	イニシエータ　→　ターゲット
07H	Message Reject	イニシエータ　↔　ターゲット
08H	No Operation	イニシーエタ　→　ターゲット
09H	Message Parity Error	イニシエータ　→　ターゲット
0CH	Bus Device	イニシエータ　→　ターゲット
80H	Identify	イニシエータ　↔　ターゲット

2.2.2　ステータス

D3841でサポートしているステータスを表2－14に示します。

なお，詳細は**D.2　SCSIステータス**を参照してください。

表2－14　D3841でサポートしているステータス

コード	ステータス名
00H	Good
02H	Check Condition
08H	Busy
18H	Reservation Conflict

2.2.3　コマンド

D3841でサポートしているコマンドを表2-15に示します。

なお，詳細は**D.3　SCSIコマンド**を参照してください。

表2-15　D3841でサポートしているコマンド

コード	コマンド名	コード	コマンド名
00H	Test Unit Ready[注]	1BH	Start/Stop Unit
01H	Rezero Unit	1CH	Receive Diagnostic
03H	Request Sense[注]	1DH	Send Diagnostic
04H	Format Unit	25H	Read Capacity
07H	Reassign Blocks	28H	Read Extended[注]
08H	Read	2AH	Write Extended[注]
0AH	Write	2BH	Seek Extended
0BH	Seek	2EH	Write & Verify[注]
12H	Inquiry	2FH	Verify
15H	Mode Select[注]	37H	Read Defect Data
16H	Reserve Unit	3BH	Write Data Buffer
17H	Release Unit	3CH	Read Data Buffer
1AH	Mode Sense[注]		

注　μPD72611デバイス・ドライバで使用したコマンド

2.3　ProDrive LPS 240Sハード・ディスク装置

ここでは，このアプリケーション・ノートで使用した240 Mバイト・ハード・ディスク　ProDrive LPS 240Sの仕様（メッセージ，ステータス，コマンド）について説明します。

2.3.1　メッセージ

ProDrive LPS 240Sでサポートしているメッセージを表2－16に示します。

表2－16　ProDrive LPS 240Sでサポートしているメッセージ

コード	メッセージ名	転送方向
00H	Command Complete	イニシエータ　←　ターゲット
02H	Save Data Pointer	イニシエータ　←　ターゲット
04H	Disconnect	イニシエータ　←　ターゲット
06H	Abort	イニシエータ　→　ターゲット
07H	Message Reject	イニシエータ　↔　ターゲット
08H	No Operation	イニシーエタ　→　ターゲット
09H	Message Parity Error	イニシエータ　→　ターゲット
0CH	Bus Device	イニシエータ　→　ターゲット
80H	Identify	イニシエータ　↔　ターゲット

2.3.2　ステータス

ProDrive LPS 240Sでサポートしているステータスを表2－17に示します。

表2－17　ProDrive LPS 240Sでサポートしているステータス

コード	ステータス名
00H	Good
02H	Check Condition
08H	Busy
18H	Reservation Conflict

2.3.3　コマンド

ProDrive LPS 240Sでサポートしているコマンドを表2-18に示します。

表2-18　ProDrive LPS 240Sでサポートしているコマンド

コード	コマンド名	コード	コマンド名
00H	Test Unit Ready	1BH	Start/Stop Unit
01H	Rezero Unit Command	1DH	Send Diagnostic
03H	Request Sense	25H	Read Capacity
04H	Format Unit	28H	Read Extended
07H	Reassign Block	2AH	Write Extended
08H	Read	2BH	Seek Extended
0AH	Write	2EH	Write And Verify
0BH	Seek	2FH	Verify
12H	Inquiry	37H	Read Defect Data
15H	Mode Select	3BH	Write Buffer
16H	Reserve	3CH	Read Buffer
17H	Release	3EH	Read Long
1AH	Mode Sense	3FH	Write Long

第３章　μPD72611の使い方

この章では，μPD72611をイニシエータとして使用した場合の基本的（エラー処理などの対処を除く）な事柄を説明します。今回はモード・セレクト・コマンドを実行する場合を例にとって説明します。

なお，システム構成は，**第２章　システム構成**のうちProDrive LPS 240Sハード・ディスク装置を除いた構成として考えます。

モード・セレクト・コマンドのフェーズ・シーケンスを図３－１に示します。

このフェーズ・シーケンスを処理するときのフロー・チャートを図３－２に示します。

図３－１　モード・セレクト・コマンドのフェーズ・シーケンス

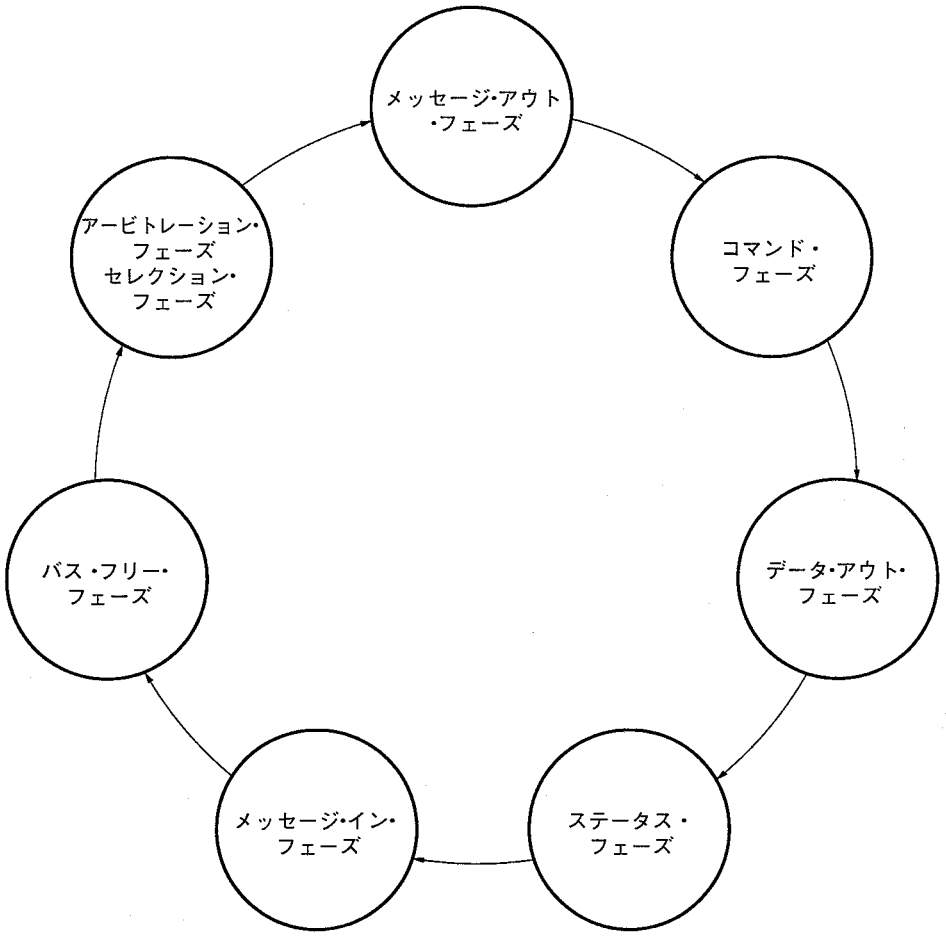
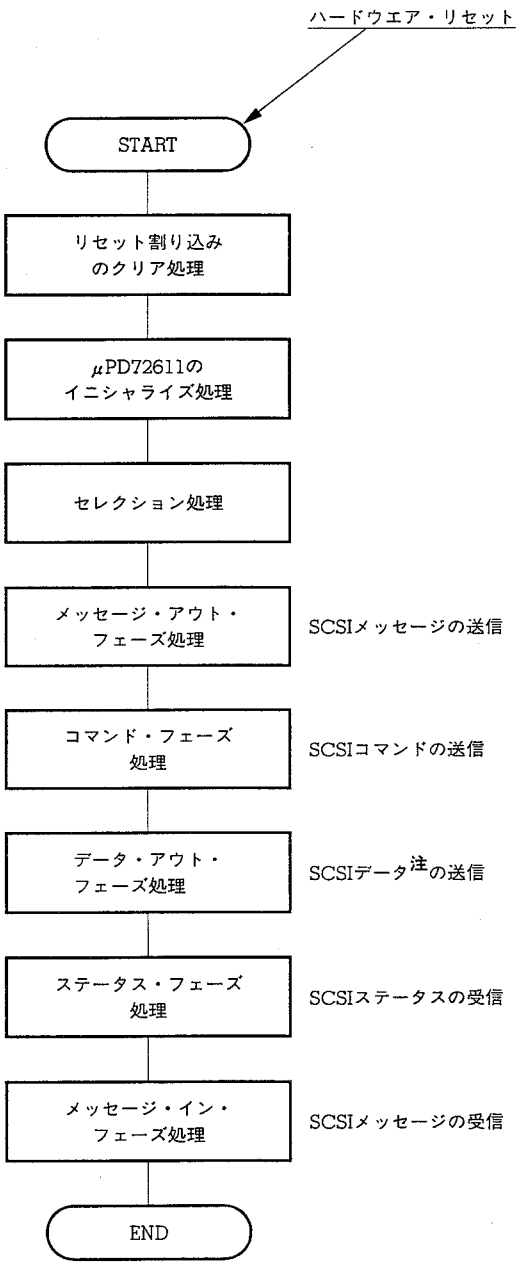


図３-２　モード・セレクト・コマンドの実行処理



注　モード・セレクト・コマンドの場合，データ・イン・フェーズ処理がありませんが，データ方向（インまたはアウト）が違うだけで処理はほとんど変わりません。

詳細は，**3.6　データ・アウト・フェーズ処理**で説明します。

注意　この処理はμPD72611が８ビット・バス・モードになっているものとして考えています。

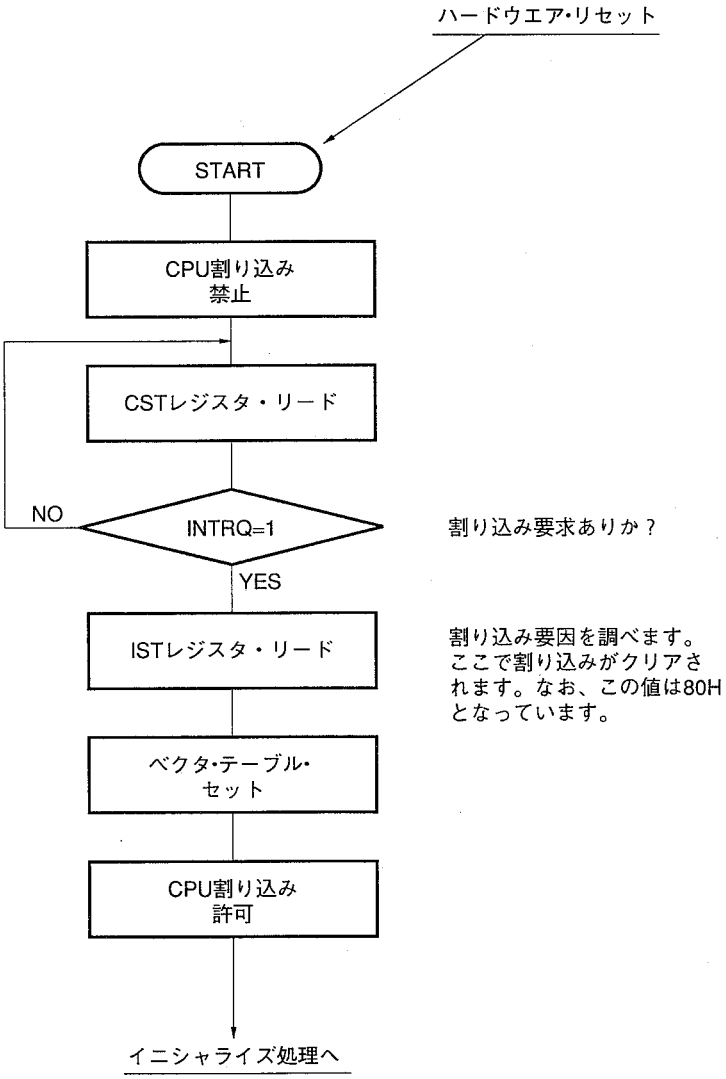
３.１　リセット割り込みのクリア処理

図３－３にリセット割り込みのクリア[注]処理を示します。

この処理では，ハードウエア・リセットによる割り込みのクリアとベクタ・テーブルのセットを行います。

注　割り込みのクリアとは，CSTレジスタのINTRQビットが０になるか，INT端子がインアクティブになることを示します。

図３－３　リセット割り込みのクリア処理



次にリセット割り込みのクリア処理について説明します。

μPD72611はハードウエア・リセット後，リセットの要因によるハードウエア割り込みが発生します。ここでISTレジスタをリードしてハードウエア割り込みをクリアしなければなりません(ハードウエア割り込みをクリアしないと次からの割り込み要因が受け付けられなくなります)。

しかし，ハードウエア割り込みでは処理できないので，ステータス・ポーリングによって処理します。

ハードウエア処理できない理由を以下に示します。

(1) ハードウエア割り込みが発生したときに，8259がイニシャライズされていない。

(2) 割り込みベクタがセットされていない。

なお，これ以降の処理は，ハードウエア割り込みを使用します。

3.2　μD72611のイニシャライズ処理

ここでは，μPD72611のイニシャライズ方法を説明します。

図3－4にイニシャライズ処理を示し，表3－1にイニシャライズの内容を説明します。

なお，参考にレジスタのディフォールト値（リセット後の値）を表3－1に示します。

図3－4　イニシャライズ処理

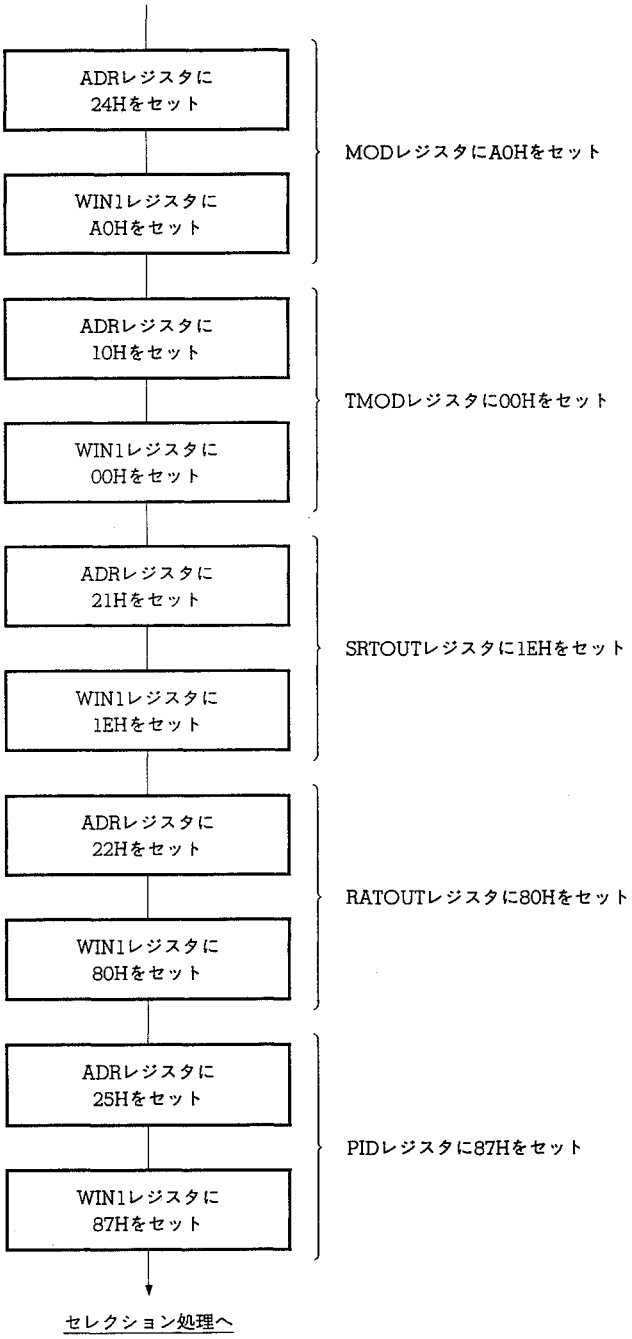


表３－１　イニシャライズの内容

レジスタ	設定値	内　　容
TMOD	00H	情報転送には，非同期転送モードを使用します。
SRTOUT	1EH	セレクト／リセレクト・タイム・アウト時間はANSIで推奨している250 msとします。
RATOUT	80H	$\overline{\text{REQ}}/\overline{\text{ACK}}$タイム・アウト時間を16384 μsとします。
MOD	A0H	このレジスタでは以下に示す項目を設定します。 (1) データ・フェーズにおけるデータ転送はDMAを使用します。 (2) CPUバスに付加するパリティ・チェックはディスエーブルにします。 (3) D3841がSCSIバスのパリティ・チェックをサポートしているので，SCSIバスに付加するパリティ・チェックはイネーブルにします。 (4) アービトレーション・モードをサポートします。 (5) ディスコネクトをサポートしないのでリセレクトされても応答しないようにします。
PID	87H	このレジスタでは次の項目を設定します。 (1) SCSIバス・コントローラとして動作します。 (2) ID番号を７に設定します（ディップ・スイッチより読み出した値)。

表３－２　レジスタのディフォールト値

レジスタ名	値	レジスタ名	値
DF0　DF1	00H	CST	42H
ADR	00H	WIN1，WIN2	00H
TP	00H	DID	00H
IST	80H	CMD	00H
すべての間接レジスタ	00H		

保守／廃止

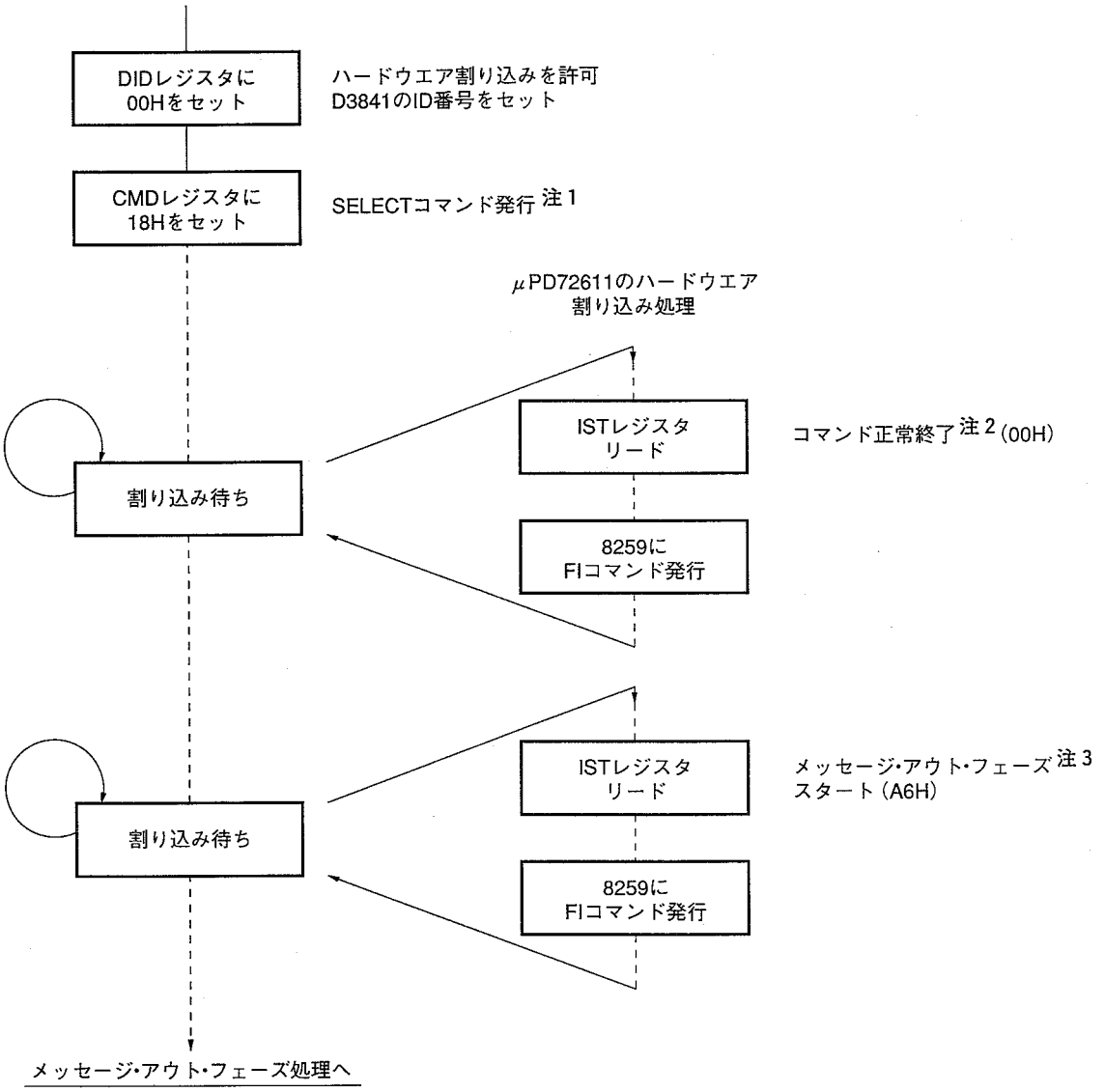
3.3　セレクション処理

ここでは，セレクションについて説明します。

この処理によって目的のターゲット（D3841）をセレクトします。

図3-5にセレクション処理を示します。

図3-5　セレクション処理



注1. SELECTコマンドは，次のフェーズがメッセージ・アウト・フェーズに移行するようにATビットをセットします。

2. SELECTコマンドが成功したことを示します。

3. D3841が次にメッセージ・アウト・フェーズに移行することを示します。この割り込み要因を確認してメッセージ・アウト・フェーズ処理へ移行します。

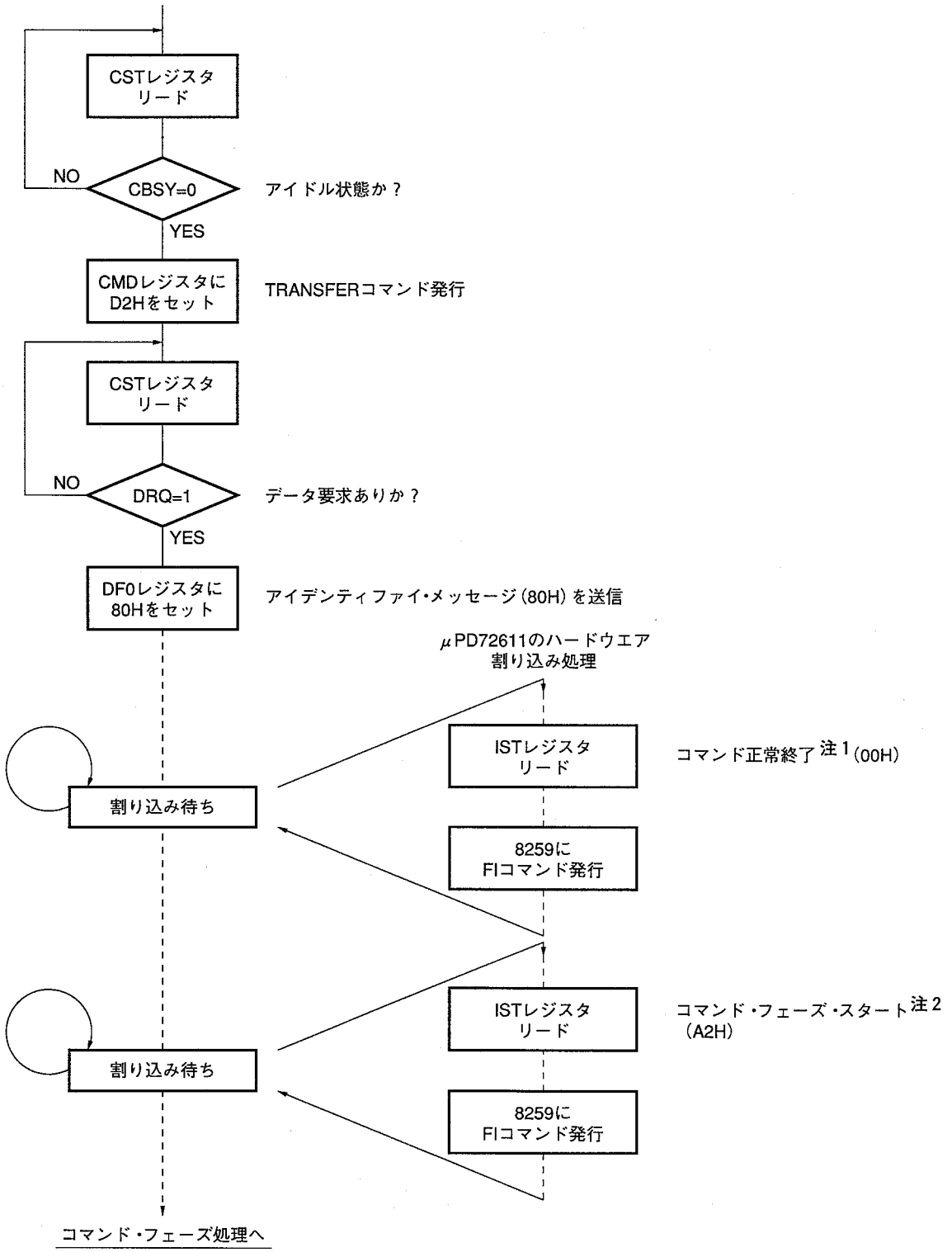
3.4　メッセージ・アウト・フェーズ処理

ここでは，メッセージ・アウト・フェーズ処理を説明します。

この処理では，D3841にSCSIメッセージ（アイデンティファイ・メッセージ80H）を送信します。アイデンティファイ・メッセージの内容については**付録D　SCSIハード・ディスク仕様**を参照してください。

図3-6にメッセージ・アウト・フェーズ処理を示します。

図３－６　メッセージ・アウト・フェーズ処理



注1．TRANSFERコマンドが成功したことを示します。

2．D3841が次にコマンド・フェーズに移行することを示します。この割り込み要因を確認してコマンド・フェーズ処理へ移行します。

3.5　コマンド・フェーズ処理

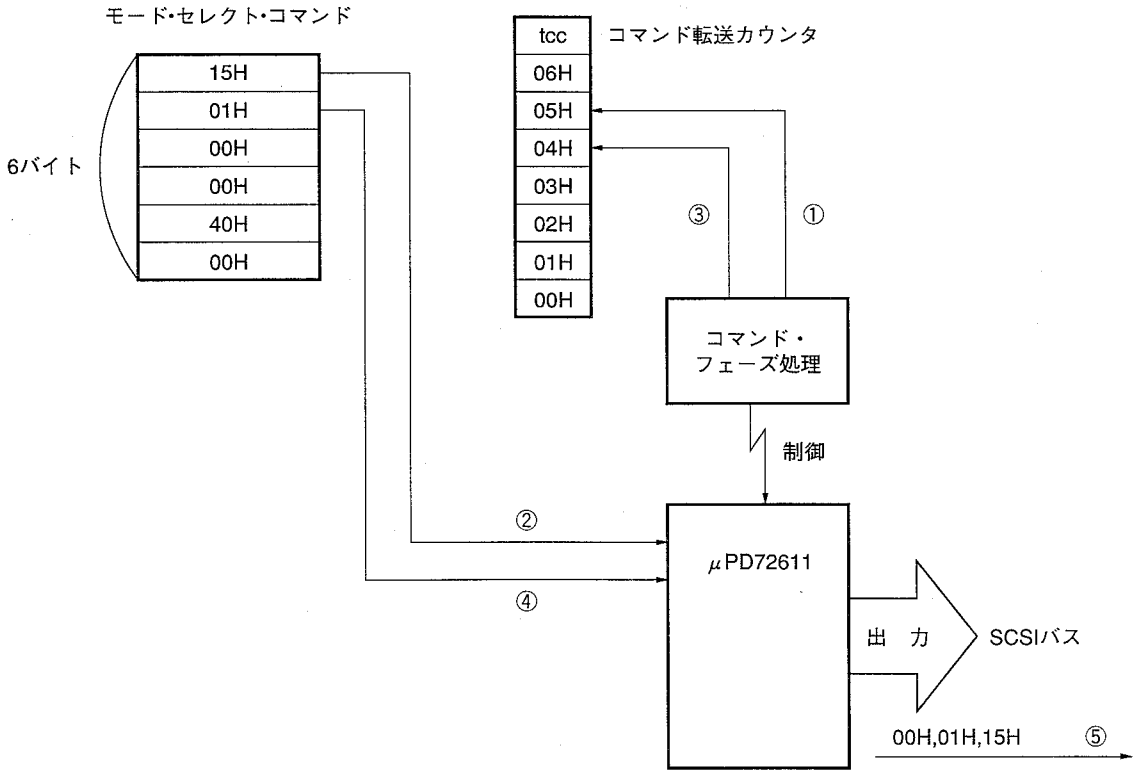
ここでは，コマンド・フェーズ処理を説明します。

この処理ではD3841にSCSIコマンド（モード・セレクト・コマンド）を送信します。

モード・セレクト・コマンドの内容については**付録D　SCSIハード・ディスク仕様**を参照してください。

図３-７にモード・セレクト・コマンドを送信する動作を示し，図３-８にコマンド・フェーズ処理を示します。

図３-７　モード・セレクト・コマンドの送信動作



① tccをデクリメントします。

② コマンド・フェーズ処理は，モード・セレクト・コマンドの１バイト目15Hを出力します。

③ tccをデクリメントします。

④ コマンド・フェーズ処理は，モード・セレクト・コマンドの２バイト目01Hを出力します。

⑤ tccが00Hになるまでモード・セレクト・コマンドを出力します。

図３－８　コマンド・フェーズ処理（1/2）

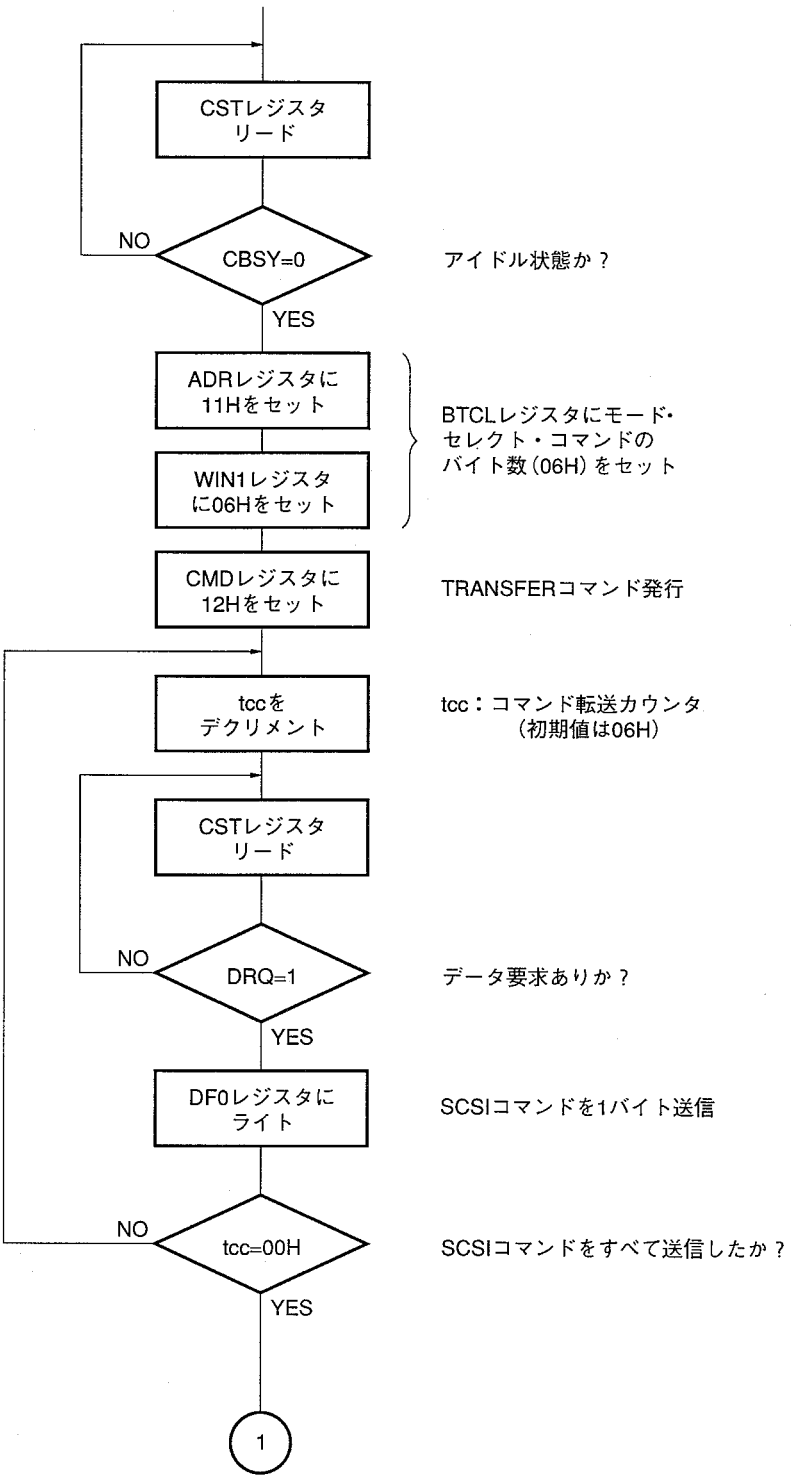
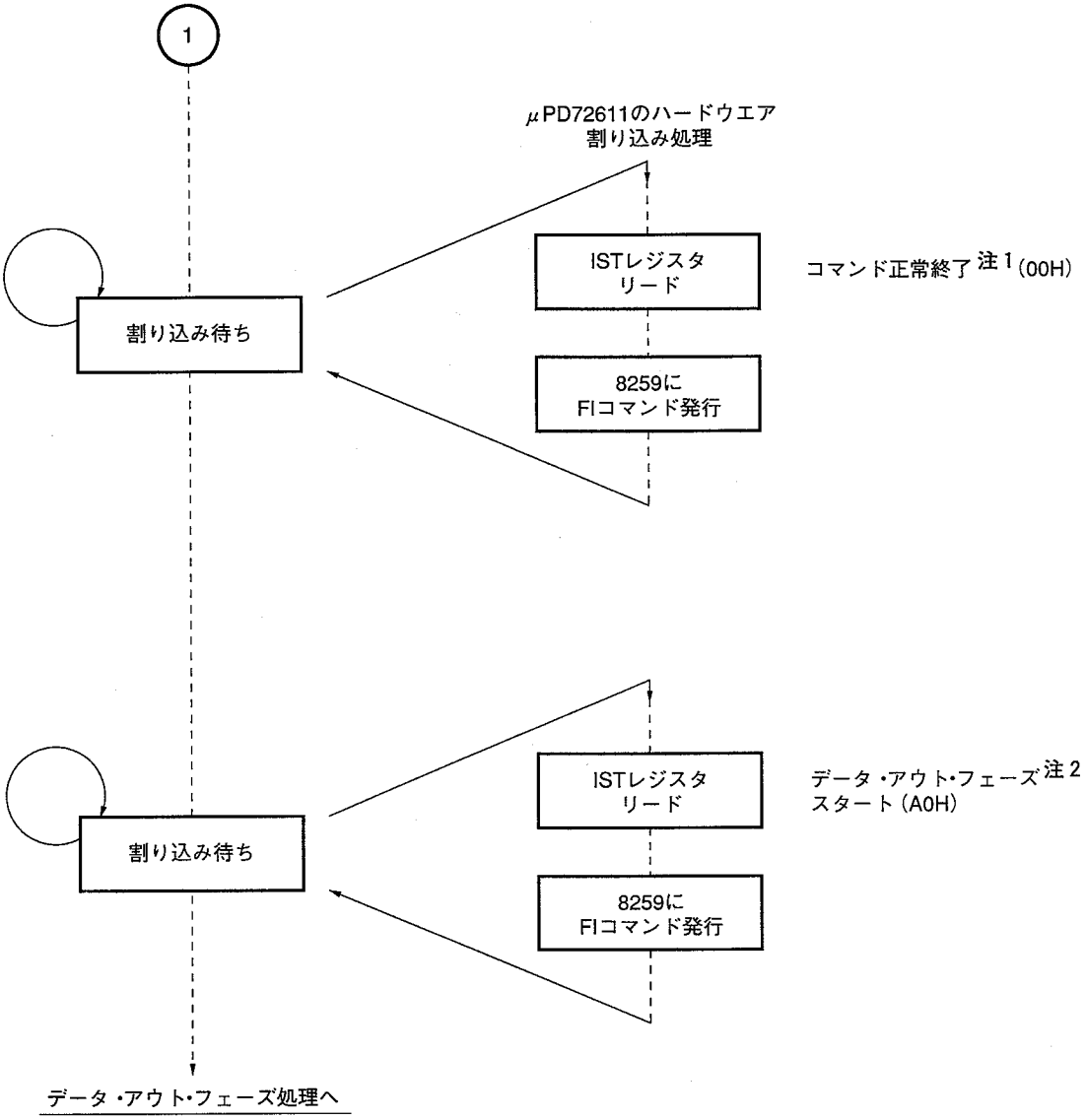


図３-８　コマンド・フェーズ処理（2/2）



注1.　TRANSFERコマンドが成功したことを示します。

2.　D3841が次にデータ・アウト・フェーズに移行することを示します。この割り込み要因を確認してデータ・アウト・フェーズ処理へ移行します。

3.6　データ・アウト・フェーズ処理

ここでは，データ・アウト・フェーズ処理を説明します。

この処理では，D3841にSCSIデータ（モード・セレクト・データ）を送信します。

モード・セレクト・データの転送には，DMAC（8237）を使用します。

モード・セレクト・データの内容については，**付録D　SCSIハード・ディスク仕様**を参照してください。

図3-9にモード・セレクト・データを送信する動作を示します。図3-10にデータ・アウト・フェーズ処理を示します。

図3-9　モード・セレクト・データの送信動作

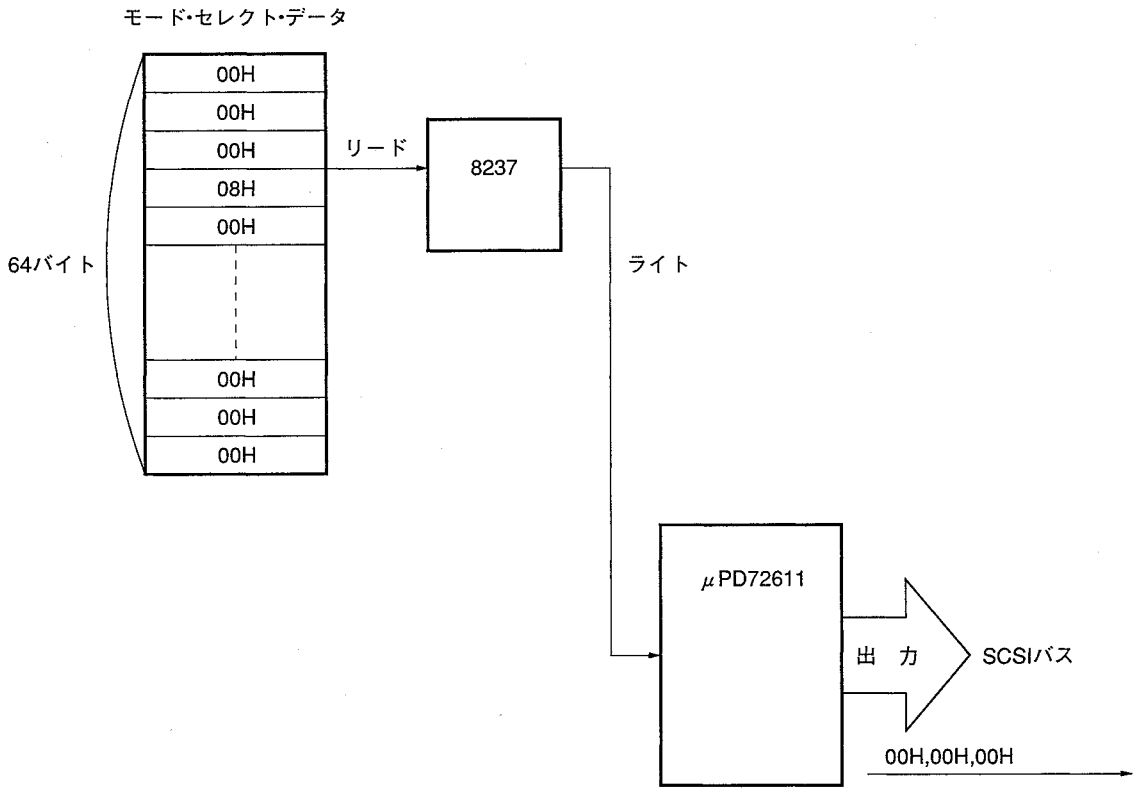
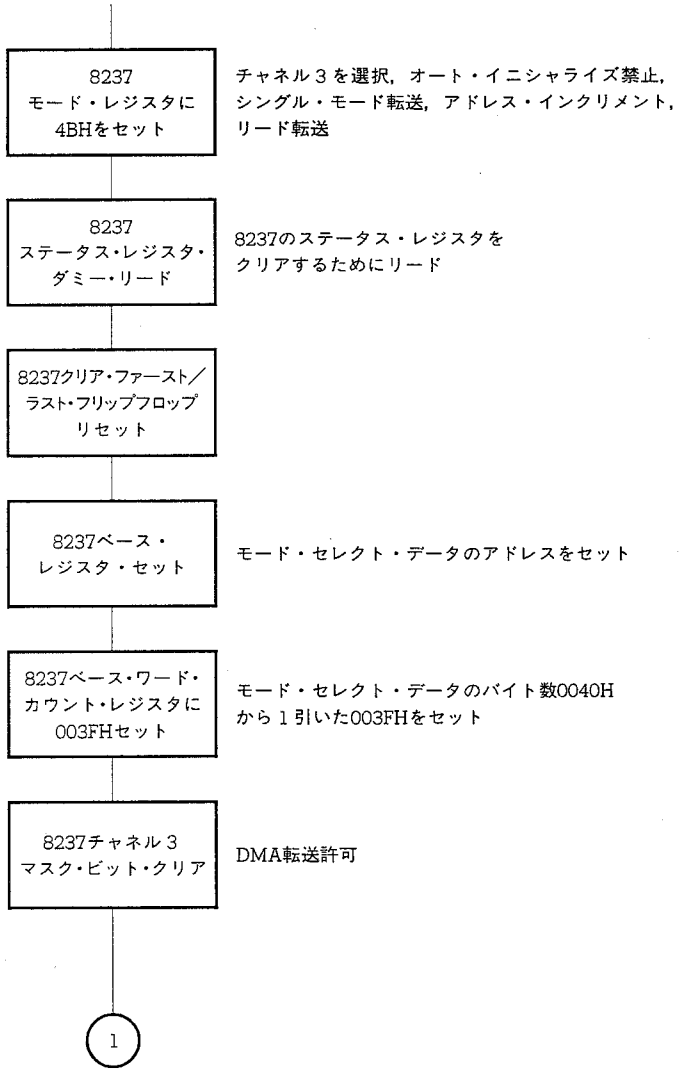


図３－10　データ・アウト・フェーズ処理（1/3）



注　データ・イン・フェーズ処理のときは，8237モード・レジスタに47Hをセットします。なお，データ・イン・フェーズ処理は，8237モード・レジスタをセットするところ以外データ・アウト・フェーズ処理と同じです。

図３－10　データ・アウト・フェーズ処理（2/3）

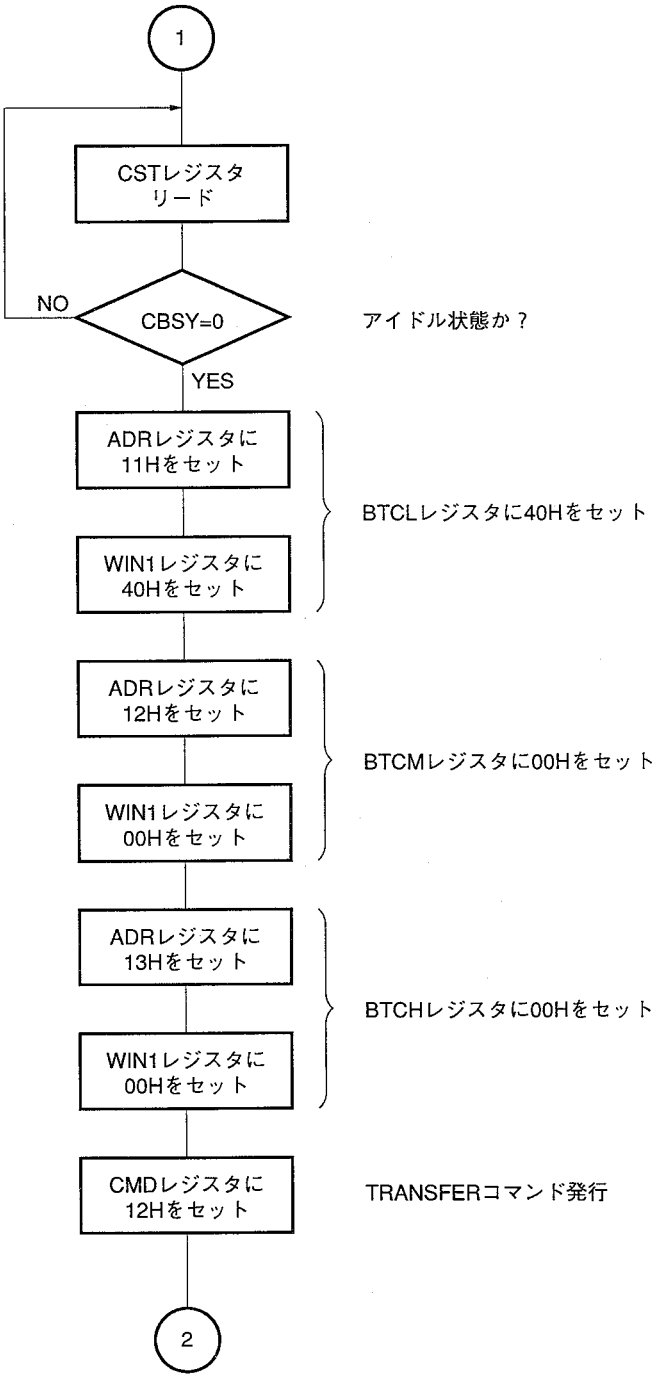
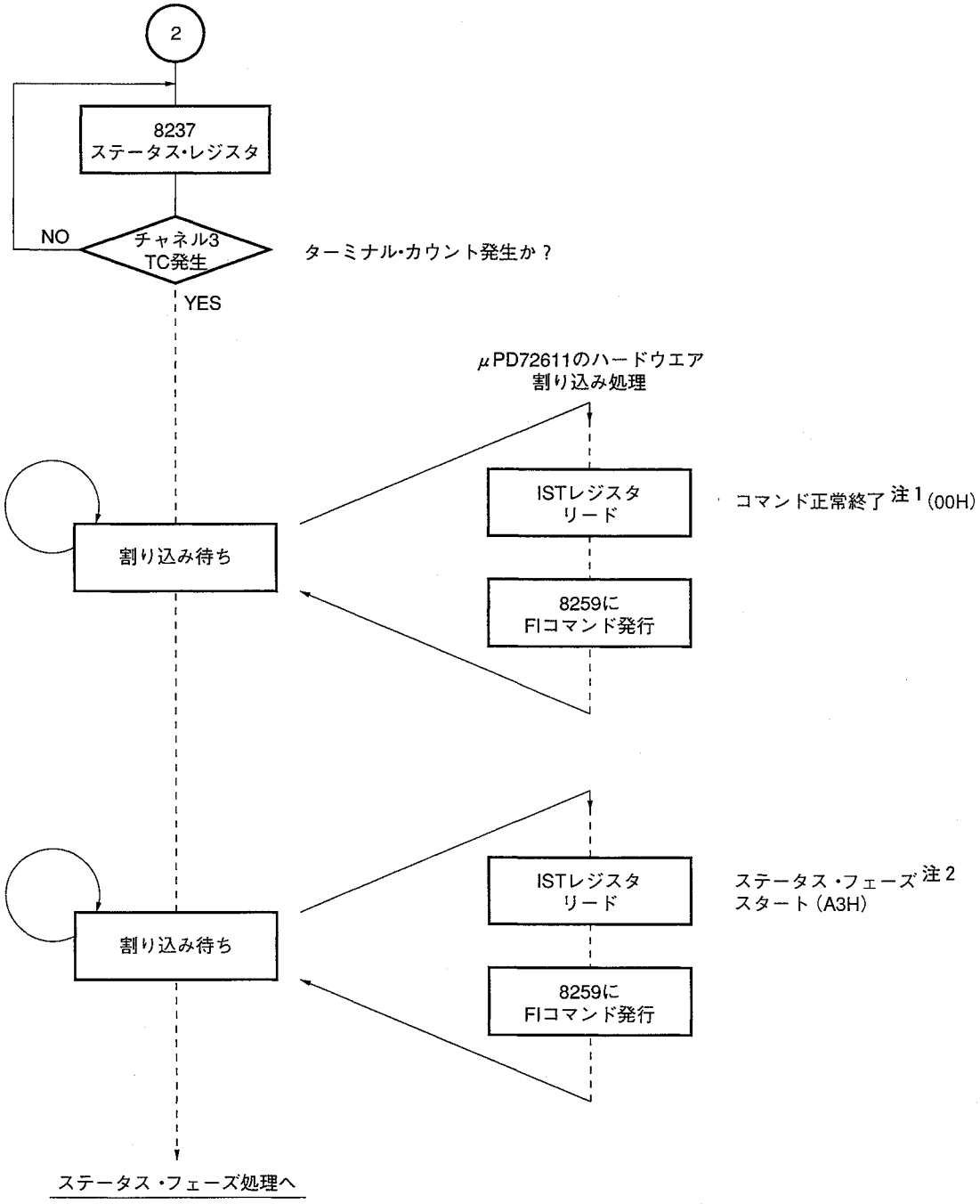


図３－10　データ・アウト・フェーズ処理（3/3）



注1．TRANSFERコマンドが成功したことを示します。

2．D3841が次にステータス・フェーズに移行することを示します。この割り込み要因を確認してステータス・フェーズ処理に移行します。

3.7　ステータス・フェーズ処理

ここでは，ステータス・フェーズ処理を説明します。

この処理では，D3841からSCSIステータスを受信します。

SCSIステータスの内容については，**付録D　SCSIハード・ディスク仕様**を参照してください。

図３－11にステータス・フェーズ処理を示します。

図３－11　ステータス・フェーズ処理（1/2）

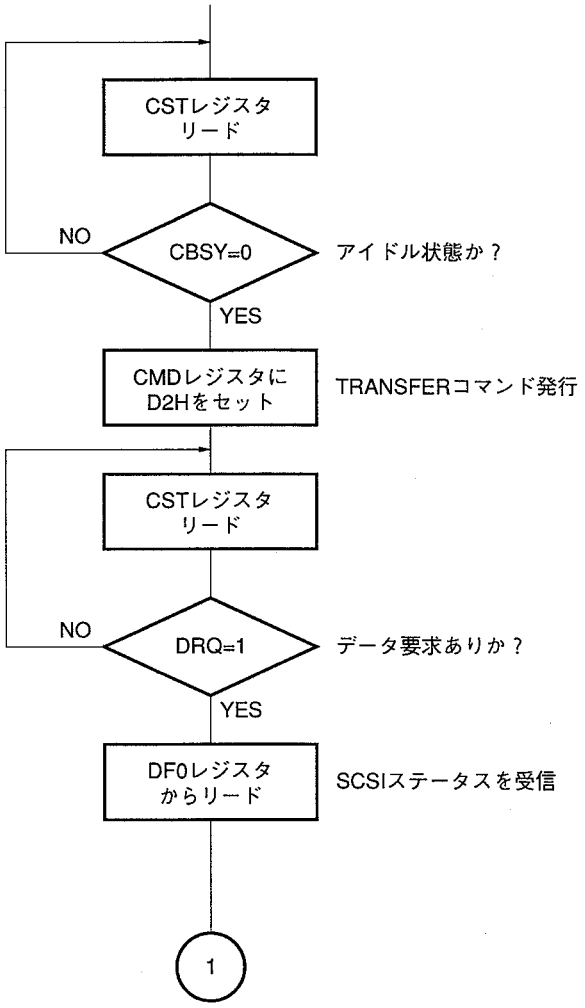
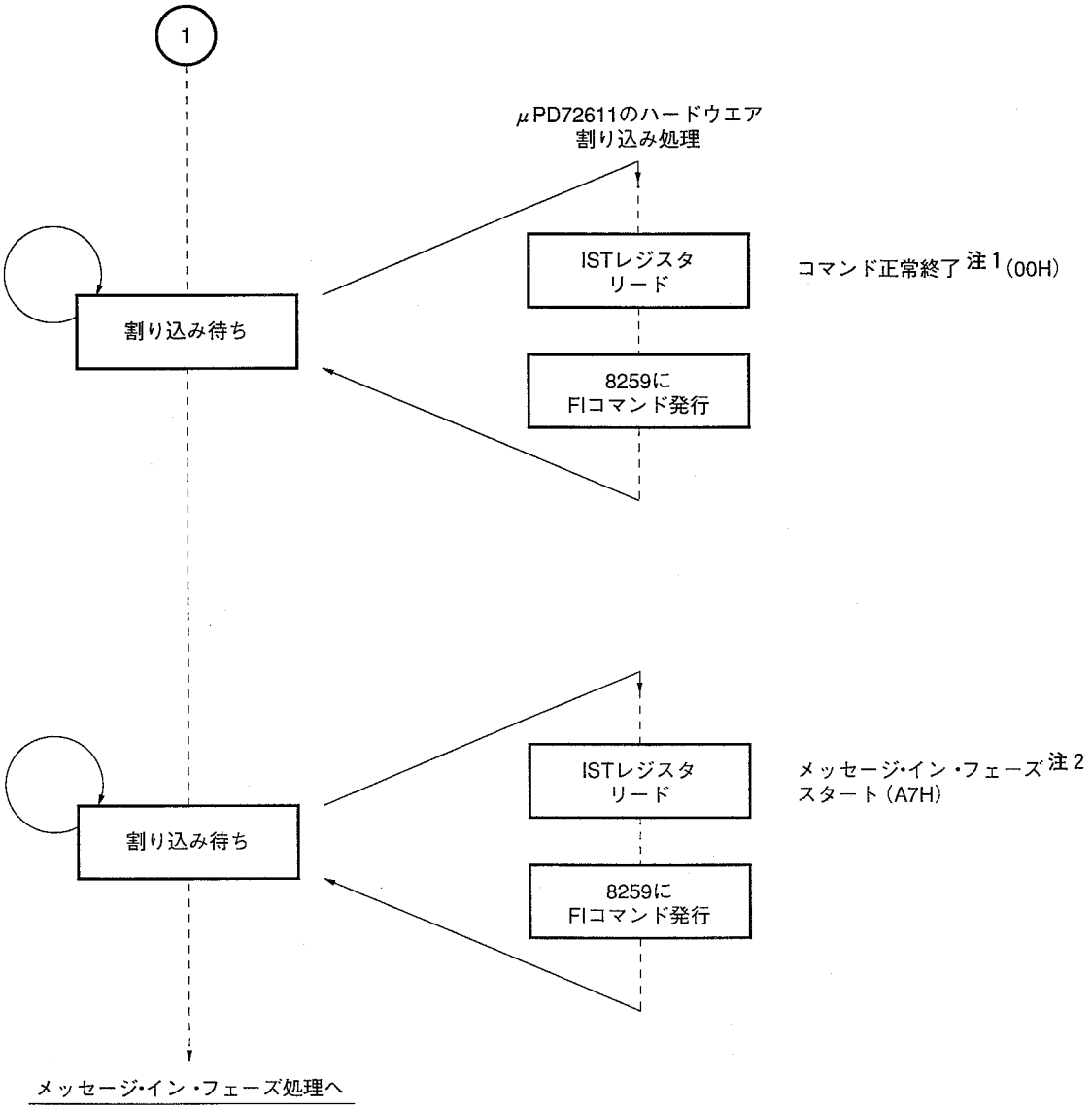


図３－１１　ステータス・フェーズ処理（2/2）



注1．TRANSFERコマンドが成功したことを示します。

2．D3841が次にメッセージ・イン・フェーズに移行することを示します。この割り込み要因を確認してメッセージ・イン・フェーズ処理に移行します。

3.8　メッセージ・イン・フェーズ処理

ここでは，メッセージ・イン・フェーズ処理を説明します。

この処理では，D3841からSCSIメッセージを受信します。

SCSIメッセージを受信することによりバス・フリー・フェーズに移行し，モード・セレクト・コマンドのフェーズ・シーケンスを終了します。

SCSIメッセージの内容については，**付録D　SCSIハード・ディスク仕様**を参照してください。

図3-12にメッセージ・イン・フェーズ処理を示します。

図3-12　メッセージ・イン・フェーズ処理（1/2）

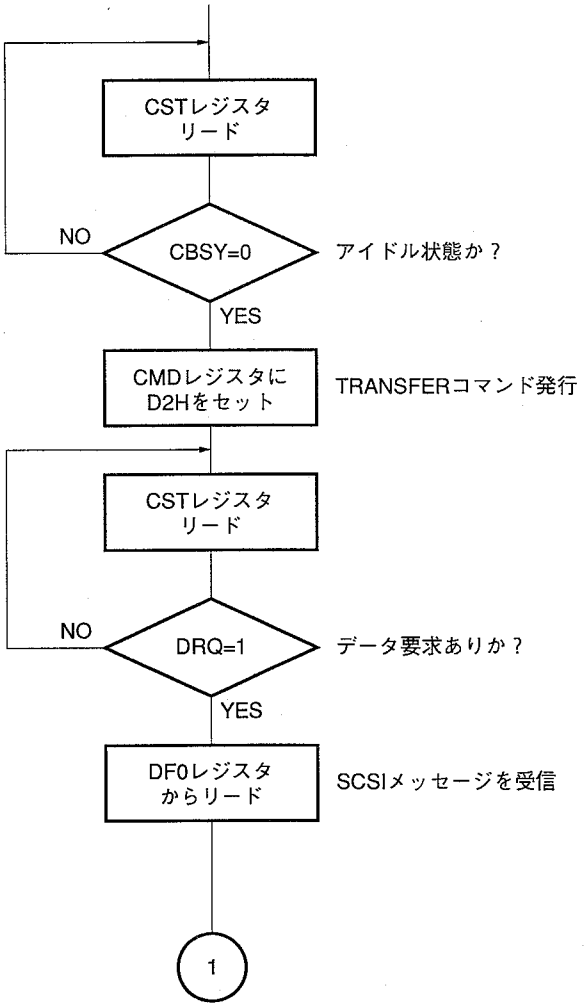
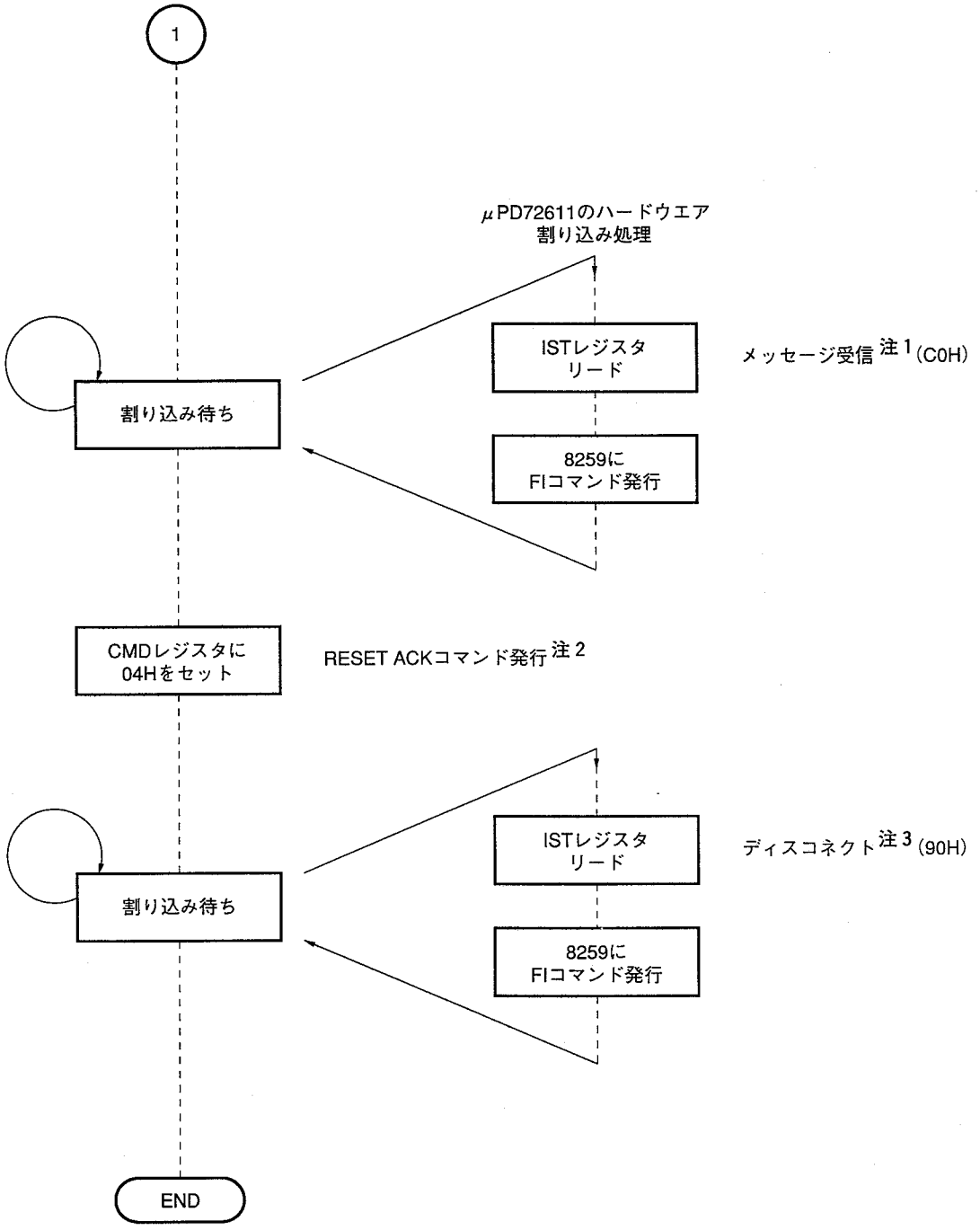


図３－12　メッセージ・イン・フェーズ処理（2/2）



注1．SCSIメッセージを受信した状態でSCSIバスが停止していることを示します。

2．RESET ACKコマンドを発行することで，SCSIバスがバス・フリー・フェーズに移行します。

3．バス・フリー・フェーズとなったことを示します。

第４章　プログラム機能

この章では３つのプログラム機能について説明します。プログラム機能を表４－１に示します。

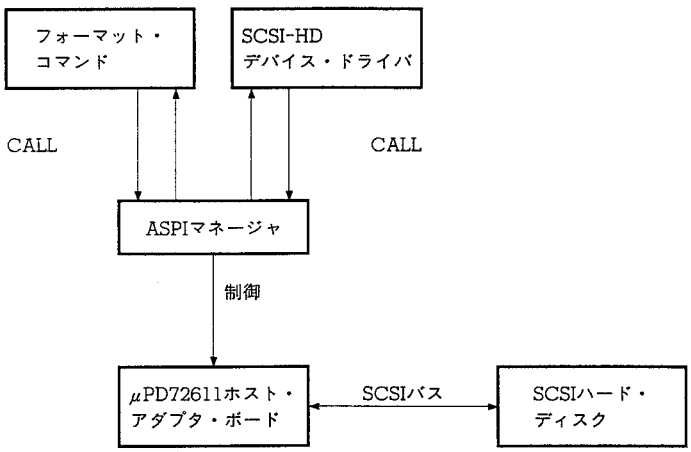
表４－１　プログラム機能

プログラム名	機　　能
フォーマット・コマンド （FORMATS.EXE）	このプログラムはMS-DOS用のフォーマット・プログラムです。
SCSI-HDデバイス・ドライバ （SCSIDD.SYS）	このプログラムはMS-DOSがSCSIハード・ディスクにファイル・アクセスするためのブロック型デバイス・ドライバです。
ASPIマネージャ （ASPIDD.SYS）	このプログラムはASPI（μPD72611ホスト・アダプタ・ボードおよびSCSIフェーズの制御を行うプログラム）をMS-DOSに常駐するためのキャラクタ型デバイス・ドライバです。

４.１　プログラム構成

プログラムの構成を図４－１に示します。図４－１に示すようにフォーマット・コマンドとSCSI-HDデバイス・ドライバはASPIをコールすることによりSCSIハード・ディスクにアクセスします。ここでASPIとASPIマネージャの説明をします。ASPIはμPD72611ホスト・アダプタ・ボードとSCSIフェーズの制御を行うプログラムです。ASPIマネージャはASPIをMS-DOSに常駐するためのキャラクタ型のデバイス・ドライバです。したがってフォーマット・コマンドおよびSCSI-HDデバイス・ドライバはASPIマネージャからASPIのエントリ・ポイントを受け取って実際にはASPIをコールします。

図４－１　プログラム構成



4.2　プログラムのファイル構成

表４－２にプログラムのファイル構成を示します。各プログラムは構成されるファイルをリンクして作成されます（リンク方法は，**第11章　コンパイル方法**を参照）。

なお，章番号の欄は各ファイル内のモジュールを説明している場所を示しています。

また，各ファイルの内容を表４－３に示します。

表４－２　ファイル構成

プログラム名	ファイル名	章　番　号
フォーマット・コマンド （FORMATS.EXE）	SFORMAT.CPP	第５章　フォーマット・コマンド
	SCOM.CPP[注]	第７章　SCSIコマンド実行ブロック
SCSI-HDデバイス・ドライバ （SCSIDD.SYS）	SDSUP.ASM SD_REQ.CPP	第６章　SCSI-HDデバイス・ドライバ
	SCOM.CPP[注]	第７章　SCSIコマンド実行ブロック
ASPIマネージャ （ASPIDD.SYS）	ASSUP.ASM AS_REQ.CPP INITIAL.CPP	第８章　ASPIマネージャ
	ASPI.CPP	第９章　ASPI
	S_PHASE.CPP IO_PORT.ASM	第10章　SCSIフェーズ処理

注　フォーマット・コマンドとSCSI-HDデバイス・ドライバは共にSCOM.CPPとリンクします。

表４－３　ファイルの内容

ファイル名	内　　容
SFORMAT.CPP	主にフォーマット・コマンドのメイン・ルーチン
SDSUP.ASM	SCSI-HDデバイス・ドライバとMS-DOSのインタフェース部分
SD_REQ.CPP	SCSI-HDデバイス・ドライバのI/Oリクエスト処理ブロック
SCOM.CPP	SCSIコマンド（リード，ライトなど）を実行するモジュール・ブロック（各ブロックはASPIをコールする）
ASSUP.ASM	ASPIマネージャとMS-DOSのインタフェース部分
AS_REQ.CPP	ASPIマネージャのI/Oリクエスト処理ブロック
INITIAL.CPP	変数，μPD72611および周辺デバイス，ターゲットの初期化をするモジュール・ブロック
ASPI.CPP	ASPIのモジュール・ブロック
S_PHASE.CPP	μPD72611とSCSIフェーズの制御を実際に実行するモジュール・ブロック
IO_PORT.ASM	32ビットI/Oの入出力を実行するモジュール・ブロック

第5章　フォーマット・コマンド

この章ではASPIを使用したMS-DOS用フォーマット・コマンドを説明します。
なお，このフォーマット・コマンドはパーティションをサポートしていません。

5.1　フォーマット・コマンドの仕様

フォーマット・コマンドの仕様を表5－1に，フォーマット後のSCSIハード・ディスクの状態を図5－1に示します。また，予約ブロックの内容を表5－2に示します。

表5－1　フォーマット仕様

項　　目	フォーマット仕様
バイト数／ブロック	512
ブロック数／クラスタ	SPC
予約ブロック数	1
FAT数	02
ルート・ディレクトリのエントリ数	512
総論理ブロック数	TC
メディアディスクプリタFAT ID	F8H
ブロック数／FAT	SPF
ルート・ディレクトリのブロック数	32
隠れブロック数	32
FAT長（ビット数）	16

SPC，SPFの計算方法は次のとおりです。

SPC：TC/FFEEH（小数点以下切り上げ）

上記の計算結果より大きい2の累乗の値がSPCの値です。

例　計算結果が5であればSPCの値は8になり，計算結果が3であればSPCの値は4になります。

SPF：SPF＝(TC－33)/(2＋(256×SPF))

図５－１　フォーマット後のSCSIハード・ディスクの状態

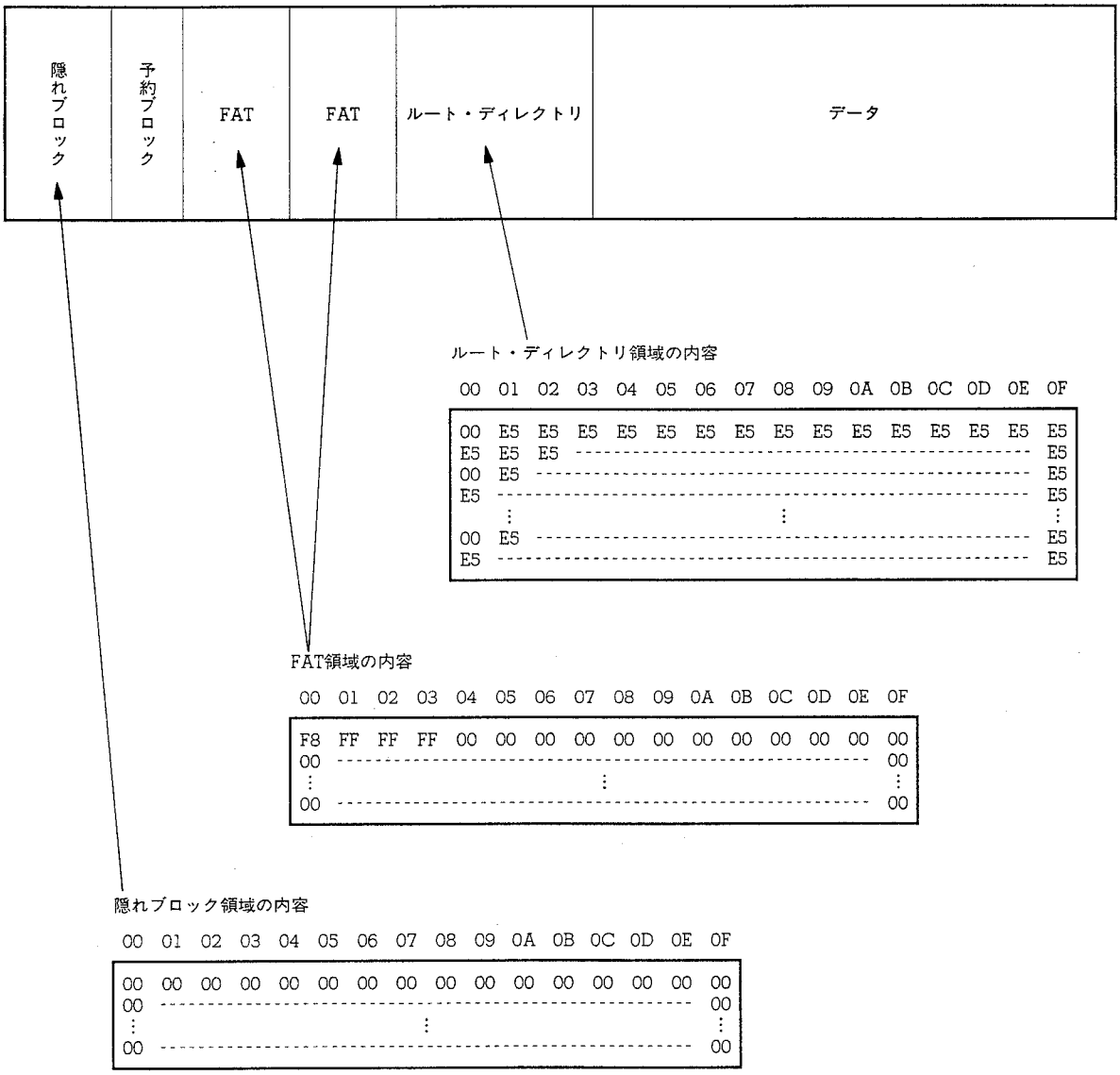


表5－2　予約ブロック

	内　　容	長　さ	オフセット	
	SHORTブランチ	3バイト	00H	← IPLがないのですべて00Hをセットします。
	OEMメーカ名	8バイト	03H	← ASCIIコードで"UPD72611"をセットします。
BPB注	バイト数／ブロック	1ワード	0BH	
BPB注	ブロック数／クラスタ	1バイト	0DH	
BPB注	予約ブロック数	1ワード	0EH	
BPB注	FAT数	1バイト	10H	
BPB注	ルート・ディレクトリのエントリ数	1ワード	11H	
BPB注	総論理ブロック数	1ワード	13H	← 拡張BPBなので0000Hをセットします。
BPB注	メディア・ディスクリプタ	1バイト	15H	
BPB注	ブロック数／FAT	1ワード	16H	
BPB注	ブロック数／トラック	1ワード	18H	
BPB注	ヘッド数	1ワード	1AH	
BPB注	隠れたブロック数	2ワード	1CH	
BPB注	拡張総論理ブロック数	2ワード	20H	
	00 …… 00		24H	

注　BPBはBIOSパラメータ・ブロックの略称です。

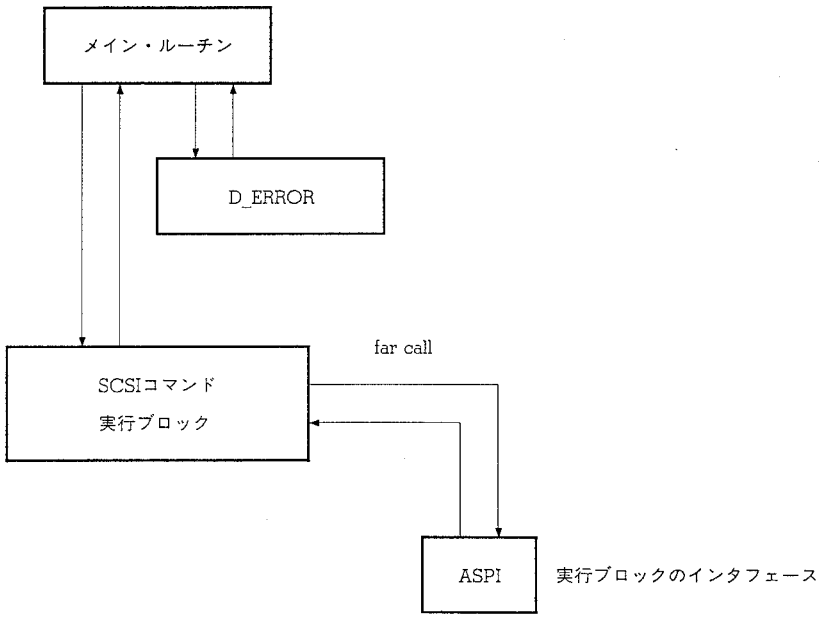
なお，フォーマット・コマンドでセットした予約ブロックのBPBはSCSI-HDデバイス・ドライバがリードし，MS-DOSに渡します。

5.2　フォーマット・コマンドのプログラム構成

図5-2にプログラム構成を示します。D_ERRORはフォーマット・コマンド実行中に発生したエラー内容を表示するモジュールです。

なお，ASPIへのアクセスはSCSIコマンド実行ブロックが行います。

図5-2　フォーマット・コマンド・プログラム構成



保守／廃止

5.2.1　フォーマット・コマンドとSCSIコマンド実行ブロックのインタフェース

フォーマット・コマンドとSCSIコマンド実行ブロックのインタフェースの説明は**7.2.1　SCSIコマンド実行ブロックと上位モジュールのインタフェース**を参照してください。

表5-3にフォーマット・コマンドが使用しているSCSIコマンド実行ブロック内のモジュールを示します。

表5-3　フォーマット・コマンド使用モジュール

モジュール名	機　　能
WRITE_DATA	データのライト処理
VERIFY	ベリファイ処理
RESERVE	リザーブ処理
RELEASE	リリース処理
MODE_SELECT	モード・セレクト処理
READ_CAPACITY	リード・キャパシティ処理
FORMT	論理フォーマット処理
T_INITIAL	ターゲットの初期化処理

5.3　フォーマット・コマンド・モジュール説明

フォーマット･コマンドの各モジュールの説明をします。表５－４に示す内容によって説明し，SPDチャートを示します。

表５－４　各モジュールの説明内容

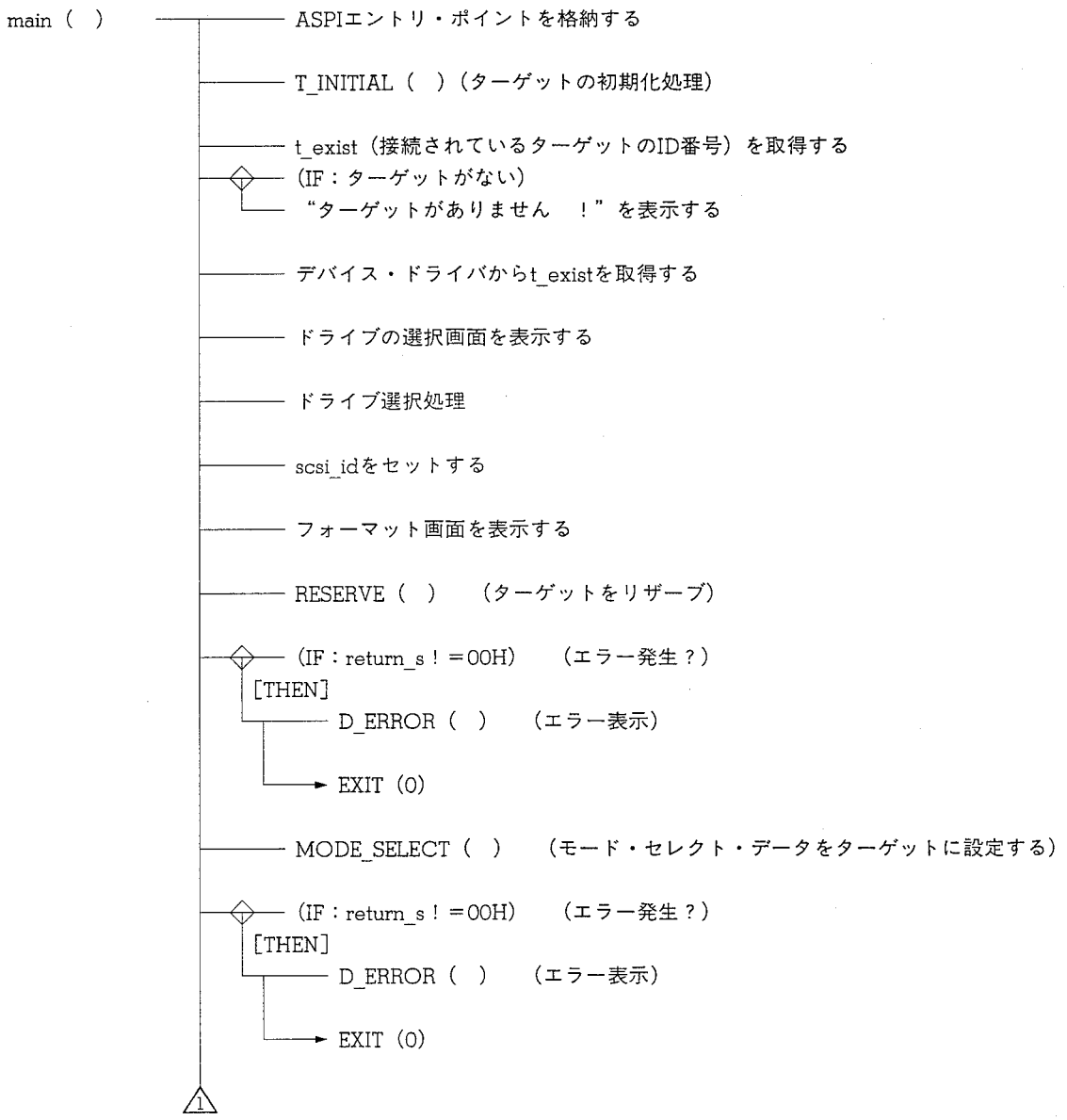
モジュール名	モジュール名を示します。
ファイル名	各モジュールのあるファイル名を示します。
言語	関数がどの言語で作成されているかを示します。
入力	入力時の変数および，レジスタを示します。
出力	出力時の変数および，レジスタを示します。
機能	関数の処理内容を示します。

保守／廃止

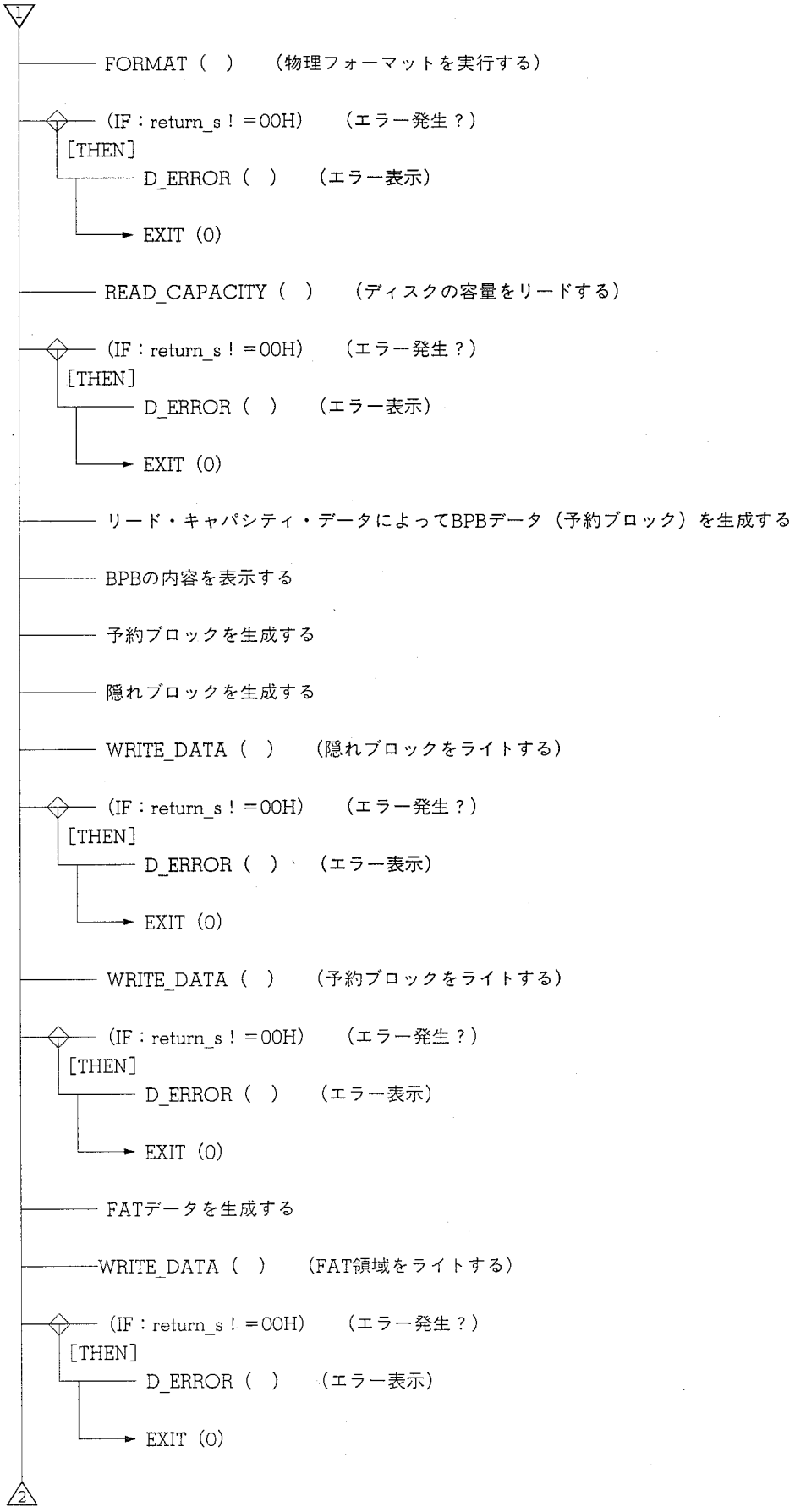
(1) フォーマット・コマンド・メイン・ルーチン

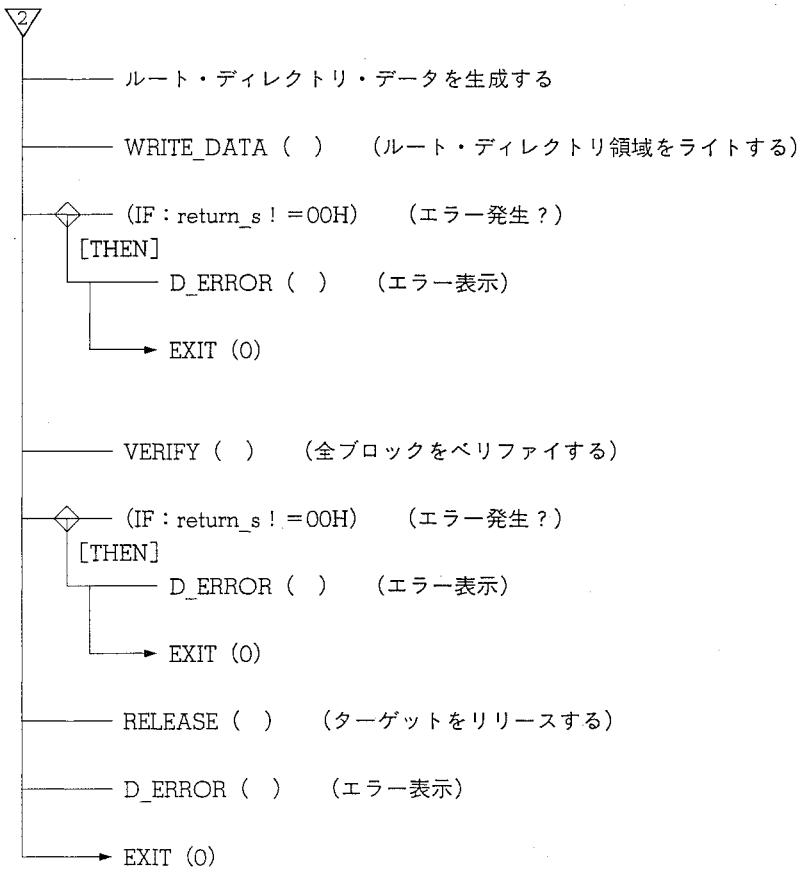
モジュール名	main（メイン・ルーチン）
ファイル名	SFORMAT.CPP
言語	C言語
入力	なし
出力	なし
機能	MS-DOS用のフォーマットを実行するモジュールです。

SPDチャート



保守／廃止


1
FORMAT（　）　（物理フォーマットを実行する）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
READ_CAPACITY（　）　（ディスクの容量をリードする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
リード・キャパシティ・データによってBPBデータ（予約ブロック）を生成する
BPBの内容を表示する
予約ブロックを生成する
隠れブロックを生成する
WRITE_DATA（　）　（隠れブロックをライトする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
WRITE_DATA（　）　（予約ブロックをライトする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
FATデータを生成する
WRITE_DATA（　）　（FAT領域をライトする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
2

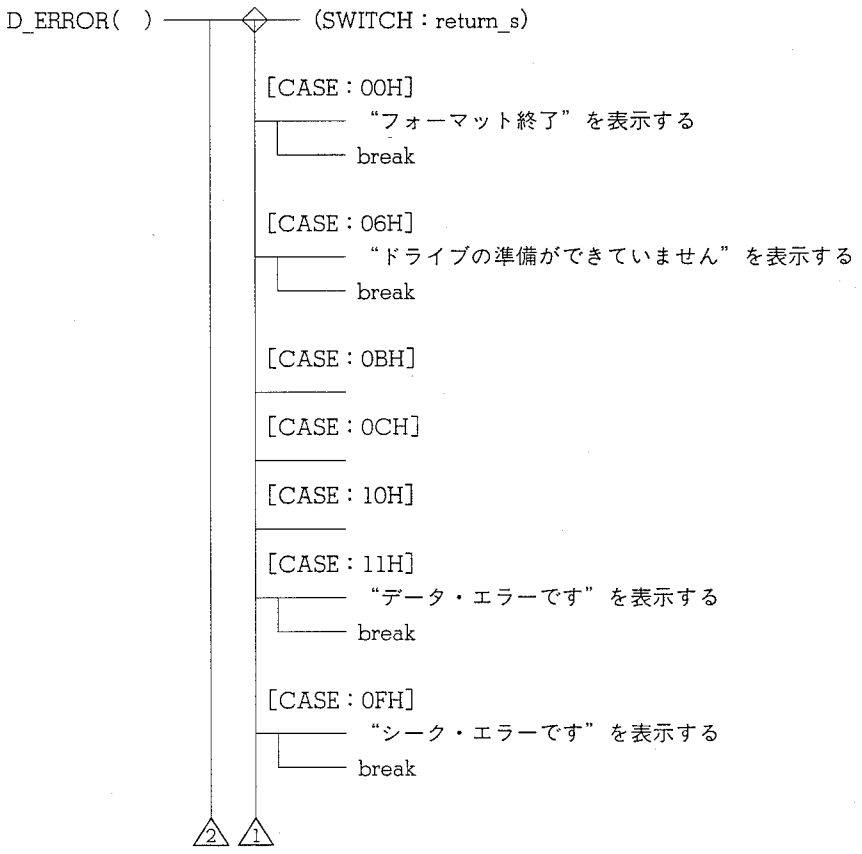

2
ルート・ディレクトリ・データを生成する
WRITE_DATA（　）　（ルート・ディレクトリ領域をライトする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
VERIFY（　）　（全ブロックをベリファイする）
（IF：return_s！＝00H）　（エラー発生？）
[THEN]
D_ERROR（　）　（エラー表示）
EXIT（0）
RELEASE（　）　（ターゲットをリリースする）
D_ERROR（　）　（エラー表示）
EXIT（0）

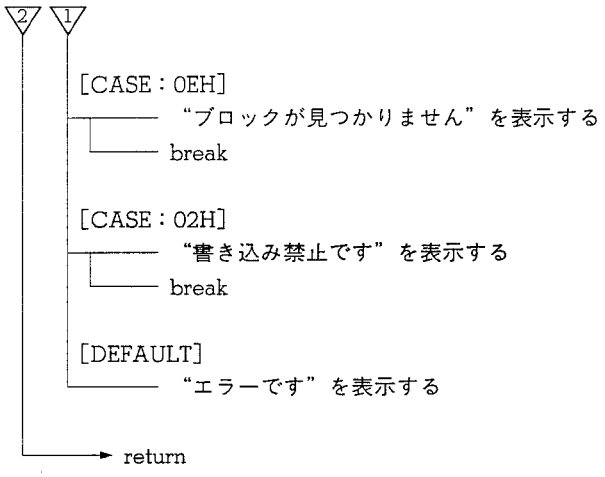
(2) エラー表示処理

モジュール名	D_ERROR
ファイル名	SFORMAT.CPP
言語	C言語
入力	return_s[注]
出力	なし
機能	return_s（SCSIコマンド実行ブロックの実行結果）によりメッセージを表示します。

注　7.3　SCSIコマンド実行ブロックの変数を参照してください。

SPDチャート




2
1
[CASE：0EH]
“ブロックが見つかりません”を表示する
break
[CASE：02H]
“書き込み禁止です”を表示する
break
[DEFAULT]
“エラーです”を表示する
return

保守／廃止

第６章　SCSI-HDデバイス・ドライバ

この章ではASPIを使用したSCSI-HDデバイス・ドライバを説明します。

6.1　SCSI-HDデバイス・ドライバの仕様

このデバイス・ドライバは，**第５章**で説明したフォーマット・コマンドでフォーマットされたSCSIハード・ディスクをアクセスするためのデバイス・ドライバです。表６－１にデバイス・ヘッダの構成を示します。このデバイス・ドライバは７台までのSCSIハード・ディスクをコントロールできます。

BIOSパラメータ・ブロック（BPB）は拡張BPBをサポートしています。BPBの値はSCSIハード・ディスクの予約ブロックからリードしMS-DOSに渡します。

表６－１　デバイス・ヘッダ

値	意　　味
－1	リンク情報
2002H	デバイス属性
STRATEGY	ストラテジ・エントリ・ポイント
INTERRUPT	割り込みエントリ・ポイント
7	ユニット数

６.２　SCSI-HDデバイス・ドライバのプログラム構成

図６-１にプログラム構成を示します。またMS-DOSがデバイス・ドライバをアクセスする動作を以下に示します。

なお，ASPIへのアクセスはSCSI実行ブロックが行います。

このデバイス・ドライバがサポートするコマンド・コードに対応する処理を表６-２に示します。

・① MS-DOSはコマンド・パケットのポインタをES，BXレジスタにセットしてストラテジ・ルーチンをコールします。ストラテジ・ルーチンはコマンド・パケットのポインタをドライバ内にセーブしてMS-DOSへリターンします。

② MS-DOSは割り込みエントリ・ルーチンをコールします。

③ 割り込みエントリ・ルーチンはI/Oリクエスト・コマンド処理をコールします。

④ I/Oリクエスト・コマンド処理はコマンド・パケット内のコマンド・コードに従って各処理（コマンド・コード対応処理）をコールします。

⑤ コマンド・コード対応処理は実行後I/Oリクエスト・コマンド処理へリターンします。

⑥ I/Oリクエスト・コマンド処理は割り込みエントリ・ルーチンにリターンします。
割り込みエントリ・ルーチンはMS-DOSにリターンします。

保守／廃止

図６－１　SCSI-HDデバイス・ドライバ・プログラム構成

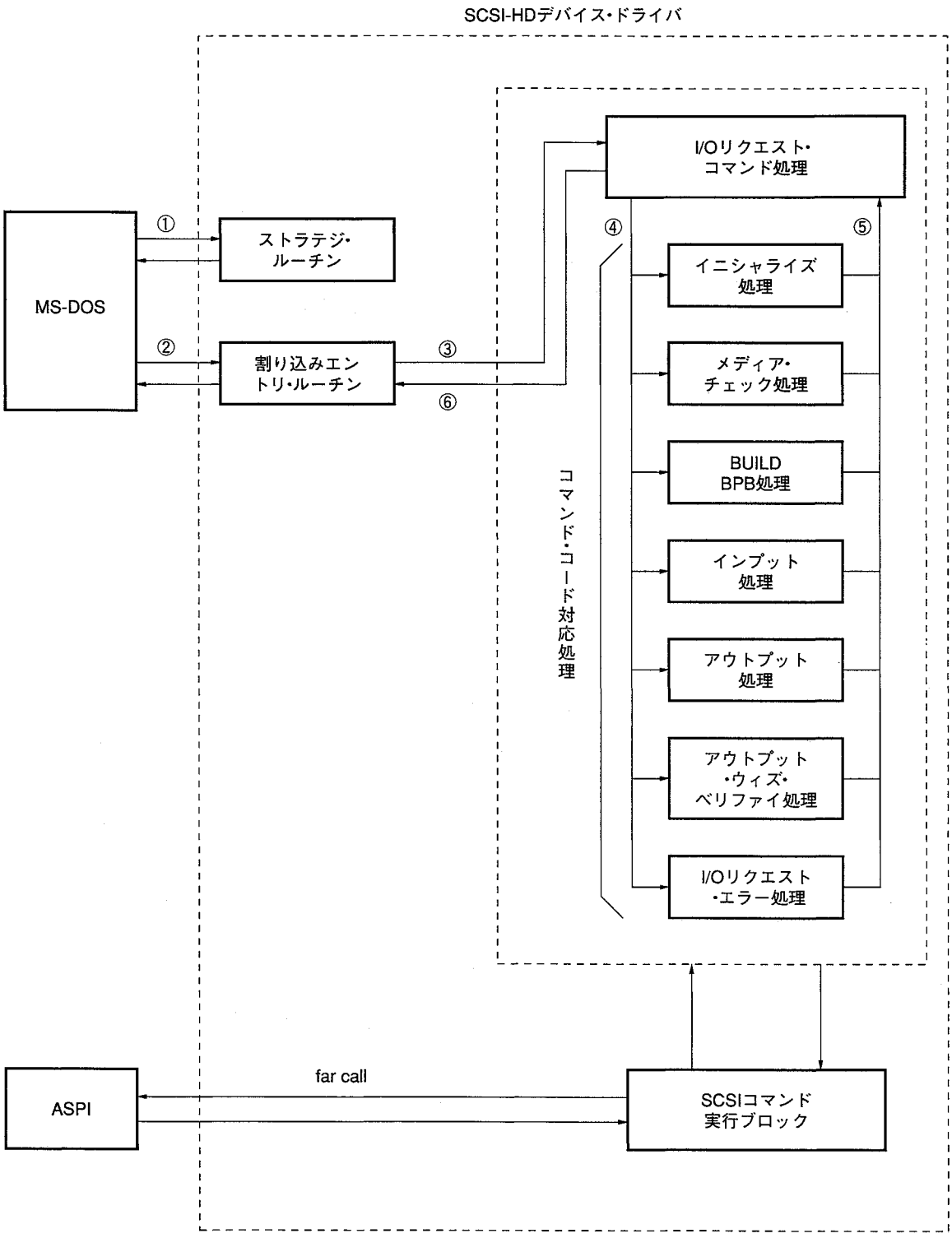


表６－２　コマンド・コード対応処理

I/Oリクエスト・コード	コマンド・コード対応処理
00H	イニシャライズ処理
01H	メディア・チェック処理
02H	BUILD BPB処理
04H	インプット処理
08H	アウトプット処理
09H	アウトプット・ウィズ・ベリファイ処理
その他	I/Oリクエスト・エラー処理

6.2.1　コマンド・コード対応処理とSCSIコマンド実行ブロックのインタフェース

コマンド・コード対応処理とSCSIコマンド実行ブロックのインタフェースの説明は，**7.2.1　SCSIコマンド実行ブロックと上位モジュールのインタフェース**を参照してください。

表６－３にコマンド・コード対応処理が使用しているSCSIコマンド実行ブロック内のモジュールを示します。

表６－３　コマンド・コード対応処理の使用モジュール

モジュール名	機　　能
WRITE_DATA	データのライト処理
READ_DATA	データのリード処理
WRITE_VERIFY	データのライトとベリファイ処理
MODE_SENSE	モード・センス処理
INQUIRY	インクワイアリ処理
T_U_READY	テスト・ユニット・レディ処理
T_INITIAL	ターゲットの初期化処理

6.3　SCSI-HDデバイス・ドライバの変数

表６－４にSCSI-HDデバイス・ドライバの入出力変数を示します。また表６－５にコマンド・パケット内のエラー・コードとリターン・ステータス（SCSIコマンド実行ブロック）の対応を示します。

保守／廃止

表６－４　SCSI-HDデバイス・ドライバの変数

変　数　名	意　　味
packet（32ビット）	コマンド・パケットのポインタ（far pointer）
prog_end（32ビット）	SCSI-HDデバイス・ドライバの終了アドレス（far pointer）
unit_n []（８ビット）	MS-DOSからのユニット番号に対応するSCSI ID番号を格納する変数

表６－５　エラー・コード／リターン・ステータス対応

エラー・コード	内　　容	リターン・ステータス
00H	ライト・プロテクト	02H
01H	無効なユニット	14H
02H	ノット・レディ	06H
03H	無効なコマンド・コード	なし
04H	CRCエラー	0BH
06H	シーク・エラー	0FH
08H	セクタが存在しない	03H，0EH
0AH	書き込みエラー	07H，10H，11H ライト実行時
0BH	読み出しエラー	0CH，10H，11H リード実行時
0CH	一般的なエラー	04H，05H，08H，09H，0AH，0DH， 12H，13H

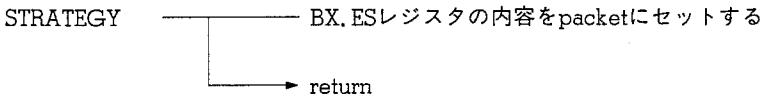
6.4　SCSI-HDデバイス・ドライバ・モジュール説明

SCSI-HDデバイス・ドライバの各モジュールの説明をします。**第５章　フォーマット・コマンド**の表５－４に示す内容によって説明し，SPDチャートを示します。

(1) ストラテジ・ルーチン

モジュール名	STRATEGY
ファイル名	SDSUP.ASM
言語	アセンブラ
入力	BXレジスタ：コマンド・パケットのオフセット・アドレス ESレジスタ：コマンド・パケットのセグメント・アドレス
出力	packet
機能	MS-DOSからBX，ESレジスタを使用して渡してくるコマンド・パケットのポインタをpacketに格納します。

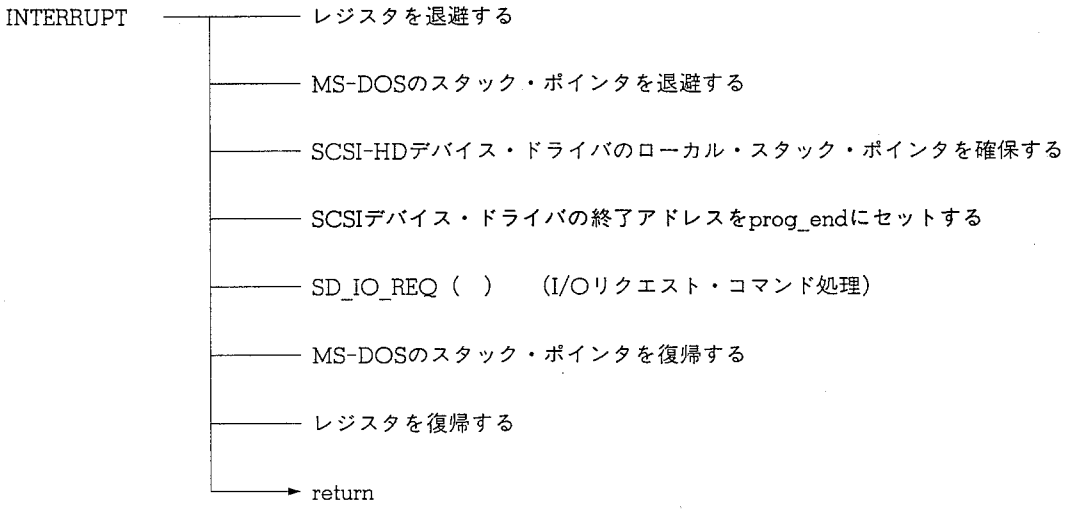
SPDチャート



(2) 割り込みエントリ・ルーチン

モジュール名	INTERRUPT
ファイル名	SDSUP. ASM
言語	アセンブラ
入力	なし
出力	prog_end
機能	SCSI-HDデバイス・ドライバの終了アドレスをprog_endにセットしI/Oリクエスト・コマンド処理をコールします。

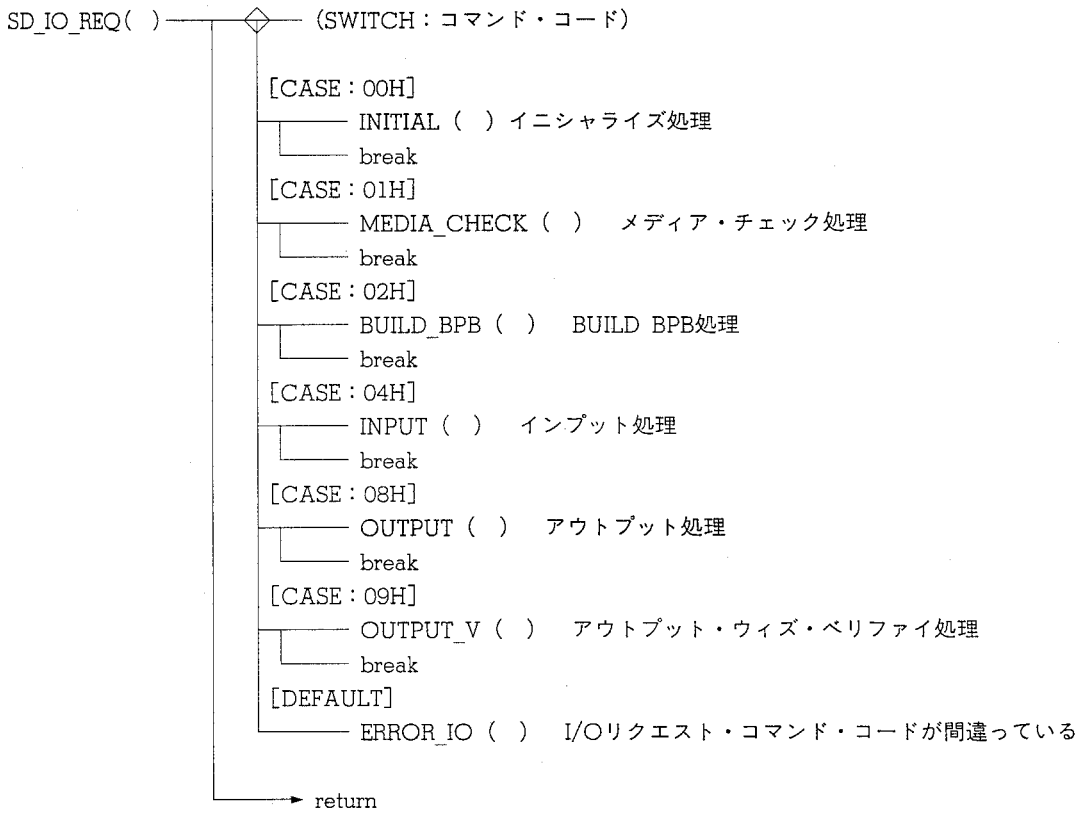
SPDチャート



(3) I/Oリクエスト・コマンド処理

モジュール名	SD_IO_REQ
ファイル名	SD_REQ.CPP
言語	C言語
入力	packet
出力	なし
機能	I/Oリクエスト・コマンド・コードに対応した処理をコールします。

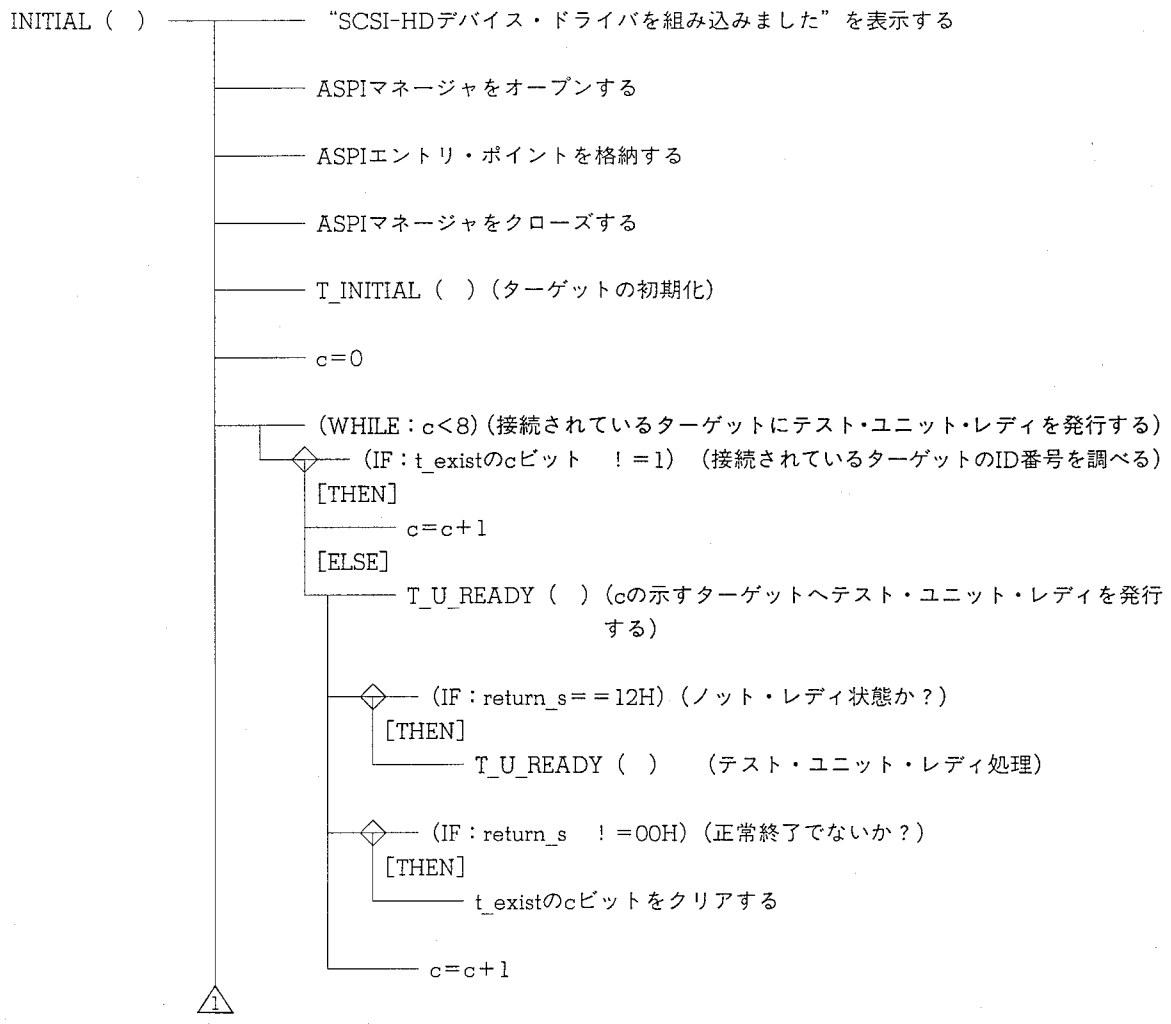
SPDチャート



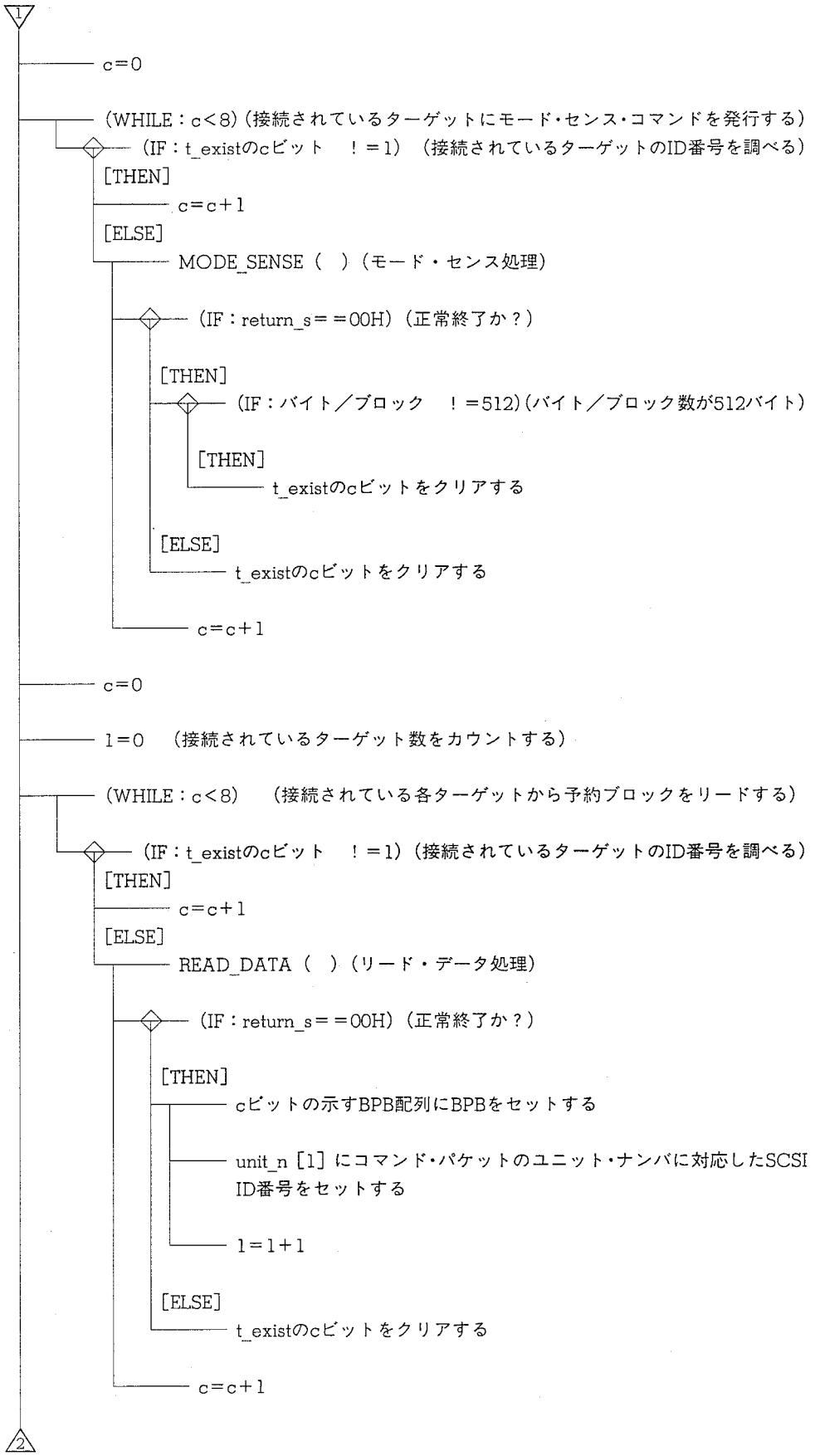
(4) イニシャライズ処理

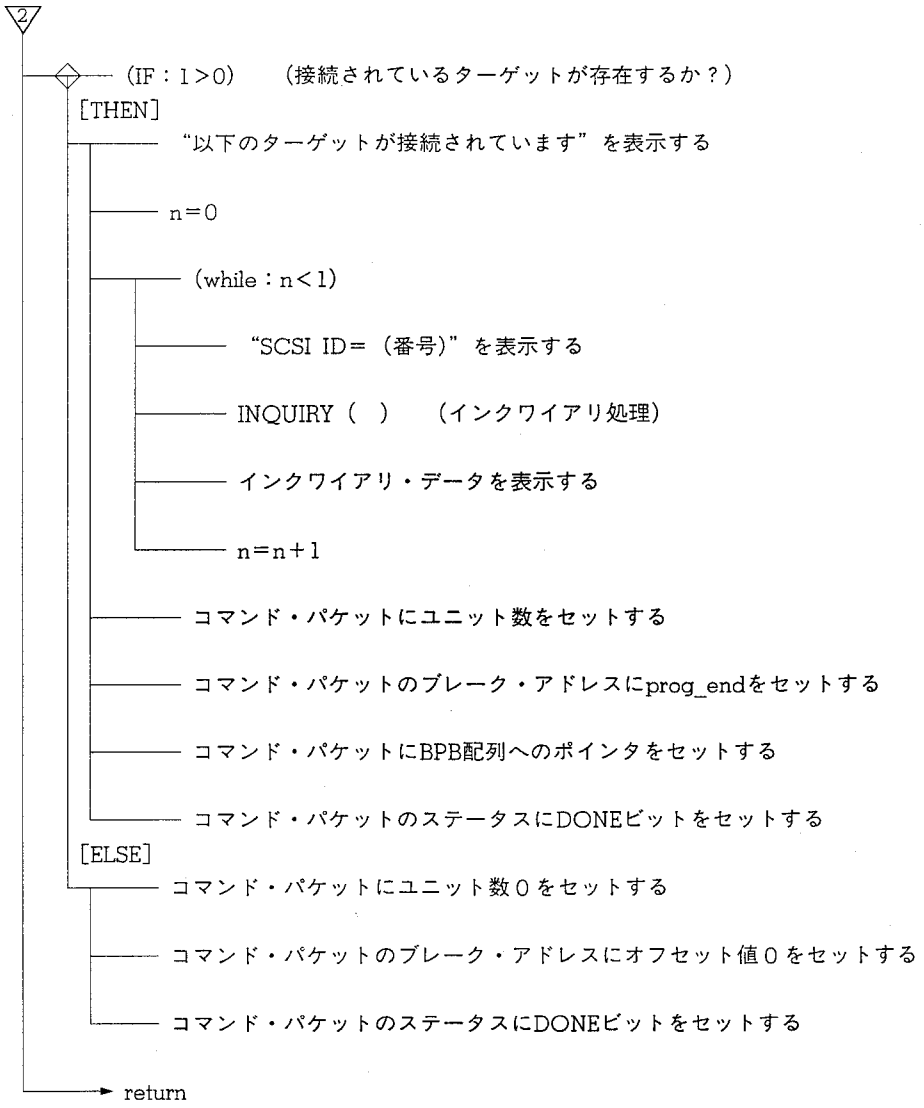
モジュール名	INITIAL
ファイル名	SD_REQ. CPP
言語	C言語
入力	packet, prog_end
出力	packet, unit_n []
機能	ASPIエントリ・ポイントを格納します。接続されているターゲットを調べ各ターゲットを初期化します。ターゲットからBPBをリードし, BPB配列のポインタをpacketにセットします。

SPDチャート



保守／廃止

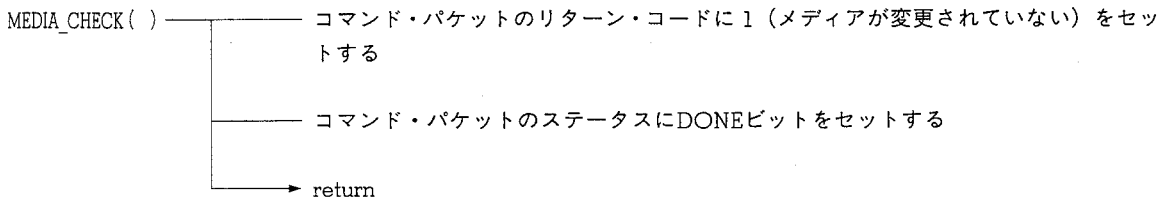

1
c=0
(WHILE：c<8)(接続されているターゲットにモード・センス・コマンドを発行する)
(IF：t_existのcビット　！=1)（接続されているターゲットのID番号を調べる）
[THEN]
c=c+1
[ELSE]
MODE_SENSE（　）(モード・センス処理)
(IF：return_s==00H)（正常終了か？）
[THEN]
(IF：バイト／ブロック　！=512)(バイト／ブロック数が512バイト)
[THEN]
t_existのcビットをクリアする
[ELSE]
t_existのcビットをクリアする
c=c+1
c=0
l=0　(接続されているターゲット数をカウントする)
(WHILE：c<8)　(接続されている各ターゲットから予約ブロックをリードする)
(IF：t_existのcビット　！=1)（接続されているターゲットのID番号を調べる）
[THEN]
c=c+1
[ELSE]
READ_DATA（　）(リード・データ処理)
(IF：return_s==00H)（正常終了か？）
[THEN]
cビットの示すBPB配列にBPBをセットする
unit_n [l] にコマンド・パケットのユニット・ナンバに対応したSCSI ID番号をセットする
l=l+1
[ELSE]
t_existのcビットをクリアする
c=c+1
2


2
(IF：l>0)　(接続されているターゲットが存在するか？)
[THEN]
“以下のターゲットが接続されています”を表示する
n=0
(while：n<l)
“SCSI ID=（番号)”を表示する
INQUIRY（ ）　(インクワイアリ処理)
インクワイアリ・データを表示する
n=n+l
コマンド・パケットにユニット数をセットする
コマンド・パケットのブレーク・アドレスにprog_endをセットする
コマンド・パケットにBPB配列へのポインタをセットする
コマンド・パケットのステータスにDONEビットをセットする
[ELSE]
コマンド・パケットにユニット数０をセットする
コマンド・パケットのブレーク・アドレスにオフセット値０をセットする
コマンド・パケットのステータスにDONEビットをセットする
return

(5) メディア・チェック処理

モジュール名	MEDIA_CHECK
ファイル名	SD_REQ.CPP
言語	C言語
入力	なし
出力	packet
機能	packetに“メディアが変更されていない”ことをセットします。

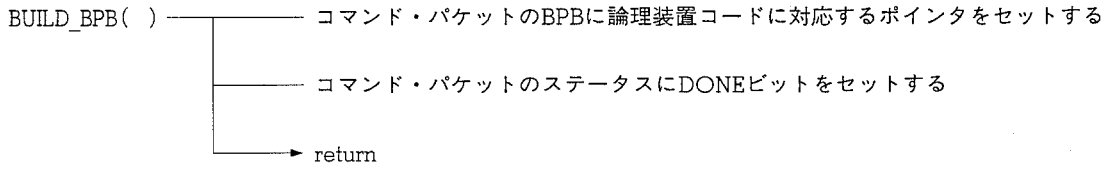
SPDチャート



(6) BUILD BPB処理

モジュール名	BUILD_BPB
ファイル名	SD_REQ.CPP
言語	C言語
入力	なし
出力	packet
機能	packetにBPBのポインタをセットします。

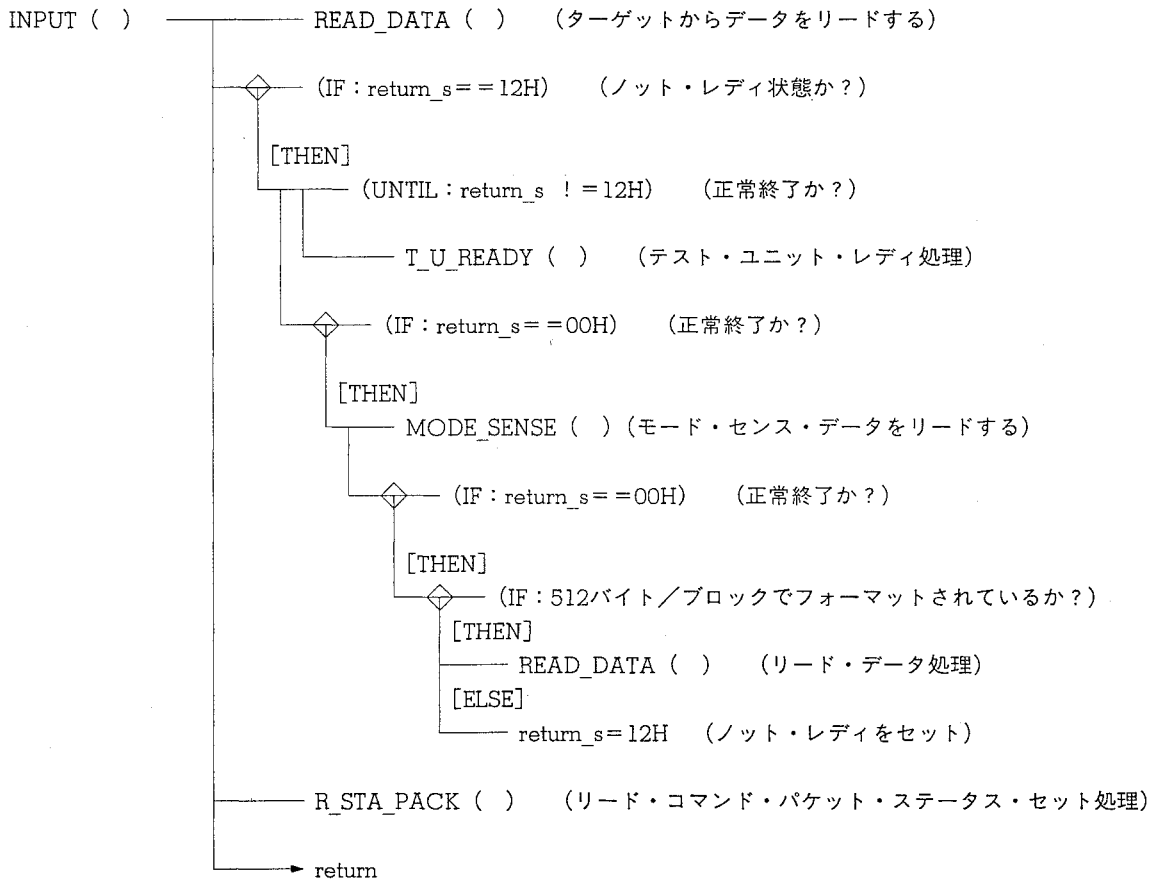
SPDチャート



(7) インプット処理

モジュール名	INPUT
ファイル名	SD_REQ.CPP
言語	C言語
入力	packet, unit_n []
出力	packet
機能	ターゲットからデータをリードし，packetで指定される領域へセットします。

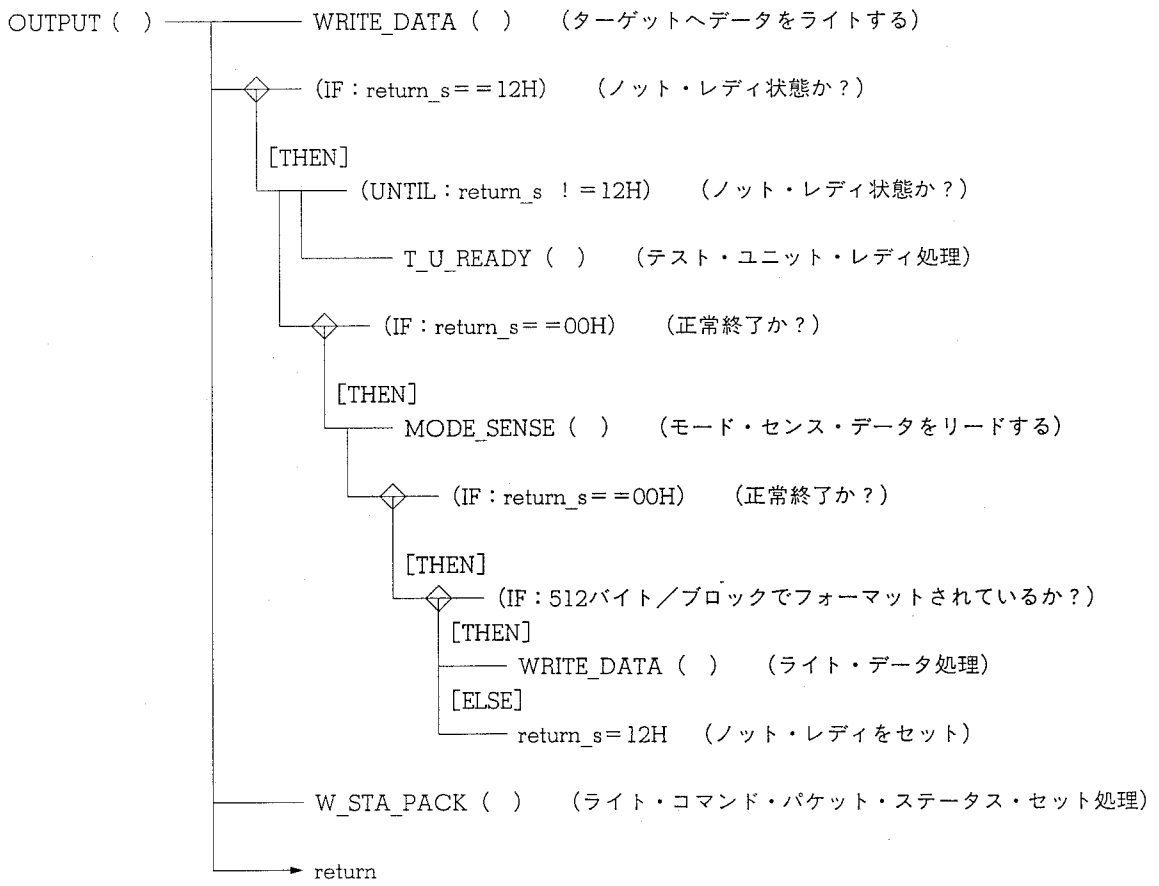
SPDチャート



(8) アウトプット処理

モジュール名	OUTPUT
ファイル名	SD_REQ.CPP
言語	C言語
入力	packet, unit_n []
出力	packet
機能	packetで指定する領域のデータをターゲットにライトします。

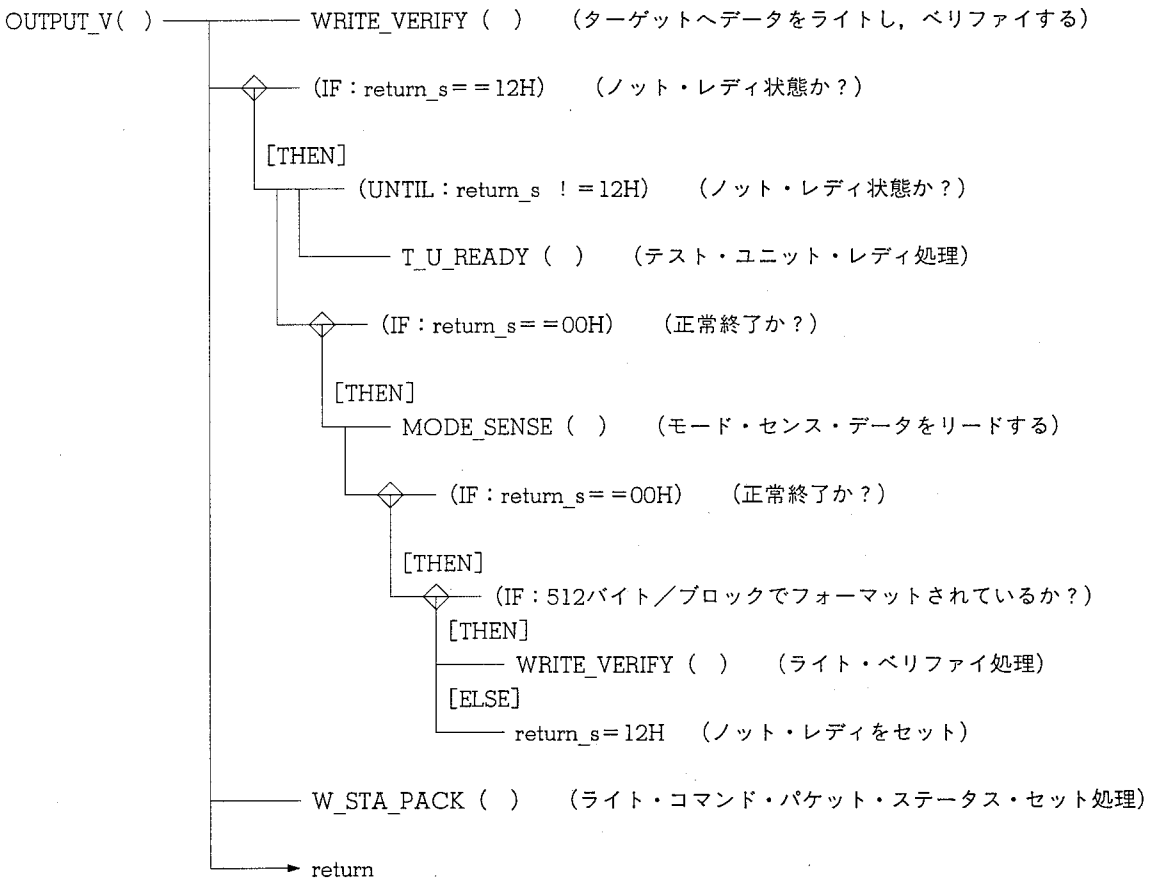
SPDチャート



(9) アウトプット・ウィズ・ベリファイ処理

モジュール名	OUTPUT_V
ファイル名	SD_REQ.CPP
言語	C言語
入力	packet, unit_n []
出力	packet
機能	packetで指定される領域のデータをターゲットにライトし，ベリファイします。

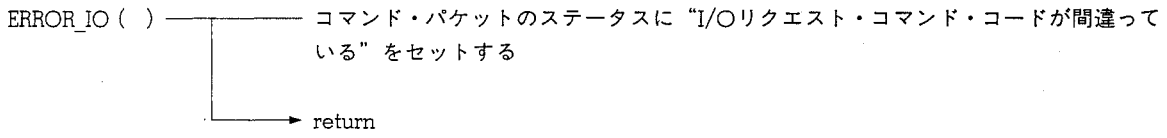
SPDチャート



(10) I/Oリクエスト・エラー処理

モジュール名	ERROR_IO
ファイル名	SD_REQ.CPP
言語	C言語
入力	なし
出力	packet
機能	packetに“I/Oリクエスト・コマンド・コードが間違っている”をセットします。

SPDチャート

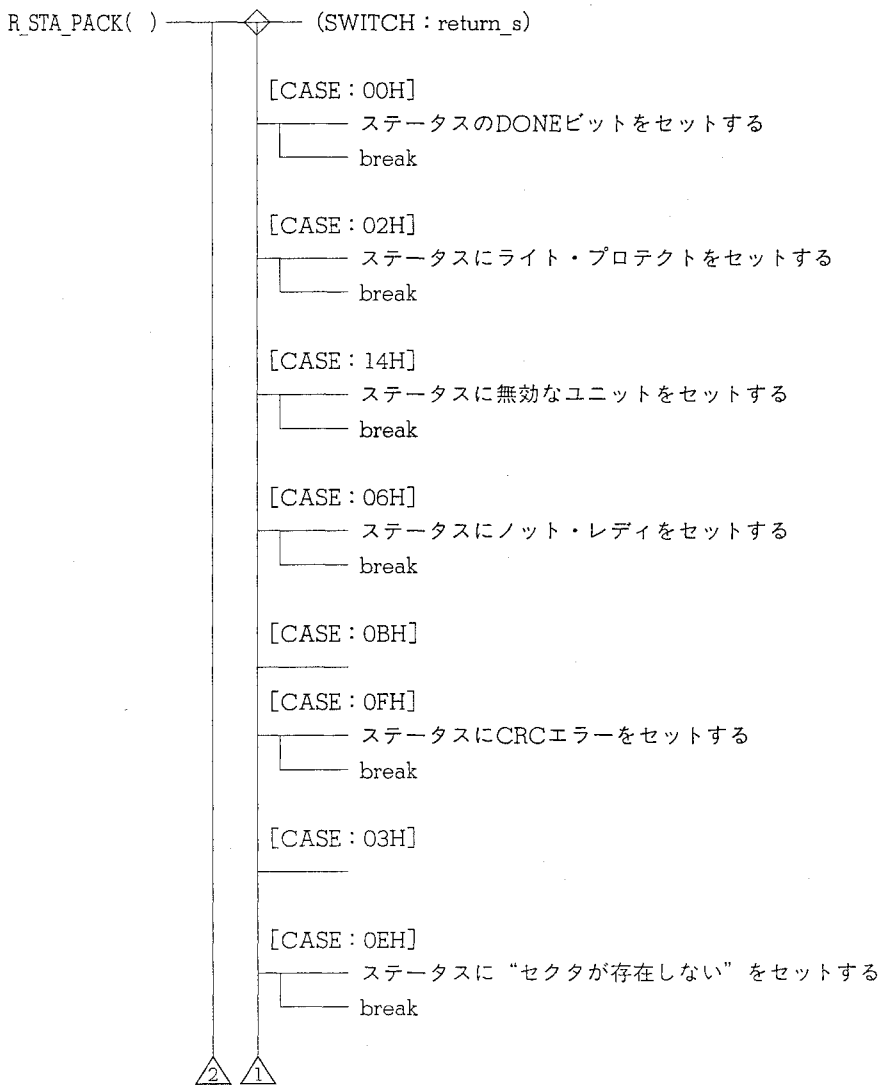


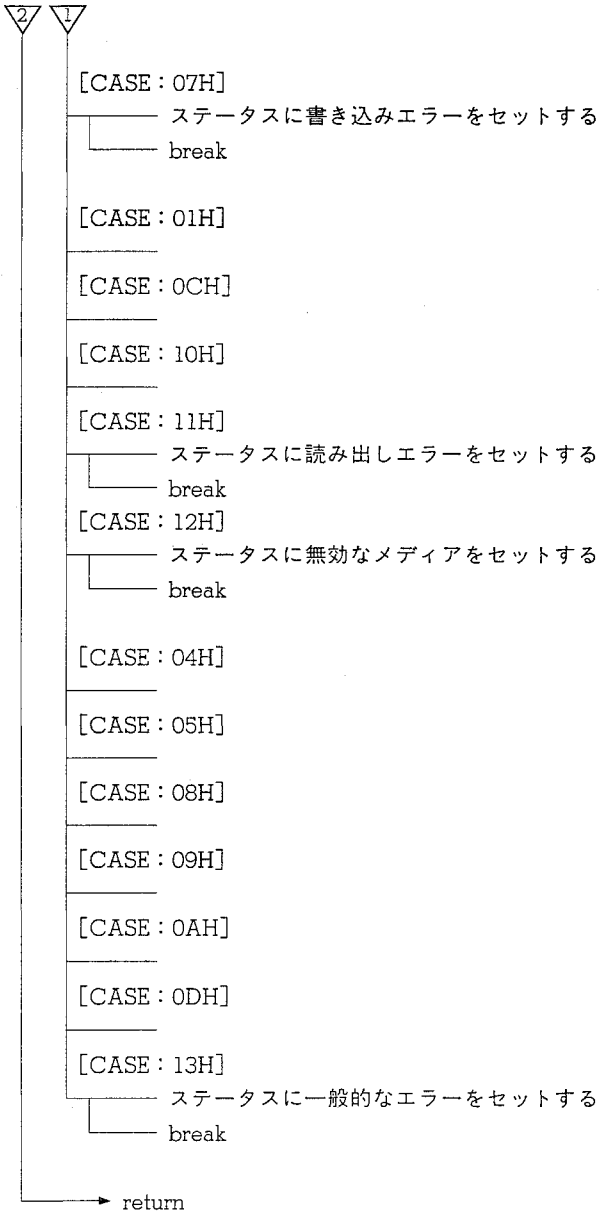
(11) リード・コマンド・パケット・ステータス・セット処理

モジュール名	R_STA_PACK
ファイル名	SD_REQ.CPP
言語	C言語
入力	return_s[注]
出力	packet
機能	return_s(SCSIコマンド実行ブロックの実行結果)に従ってパケットのステータスをセットします。

注　7.3　SCSIコマンド実行ブロックの変数を参照してください。

SPDチャート



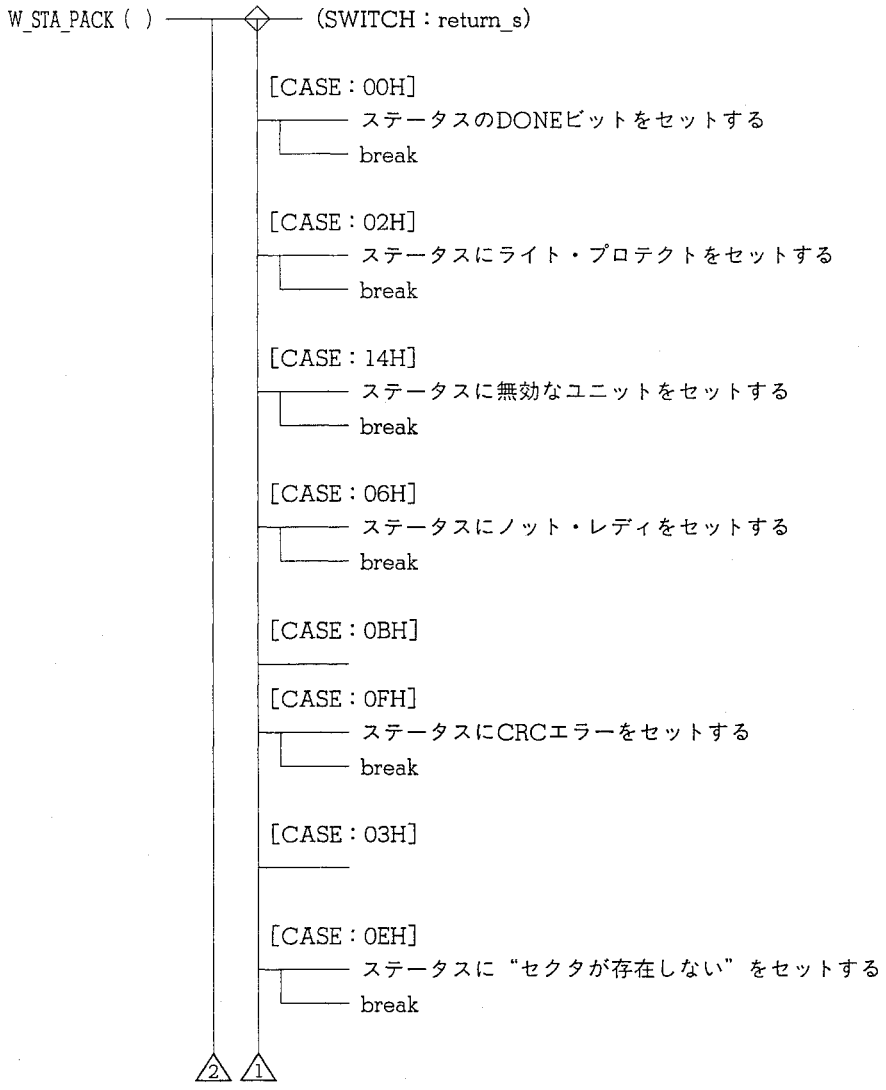

2
1
[CASE：07H]
ステータスに書き込みエラーをセットする
break
[CASE：01H]
[CASE：0CH]
[CASE：10H]
[CASE：11H]
ステータスに読み出しエラーをセットする
break
[CASE：12H]
ステータスに無効なメディアをセットする
break
[CASE：04H]
[CASE：05H]
[CASE：08H]
[CASE：09H]
[CASE：0AH]
[CASE：0DH]
[CASE：13H]
ステータスに一般的なエラーをセットする
break
return

(12) ライト・コマンド・パケット・ステータス・セット処理

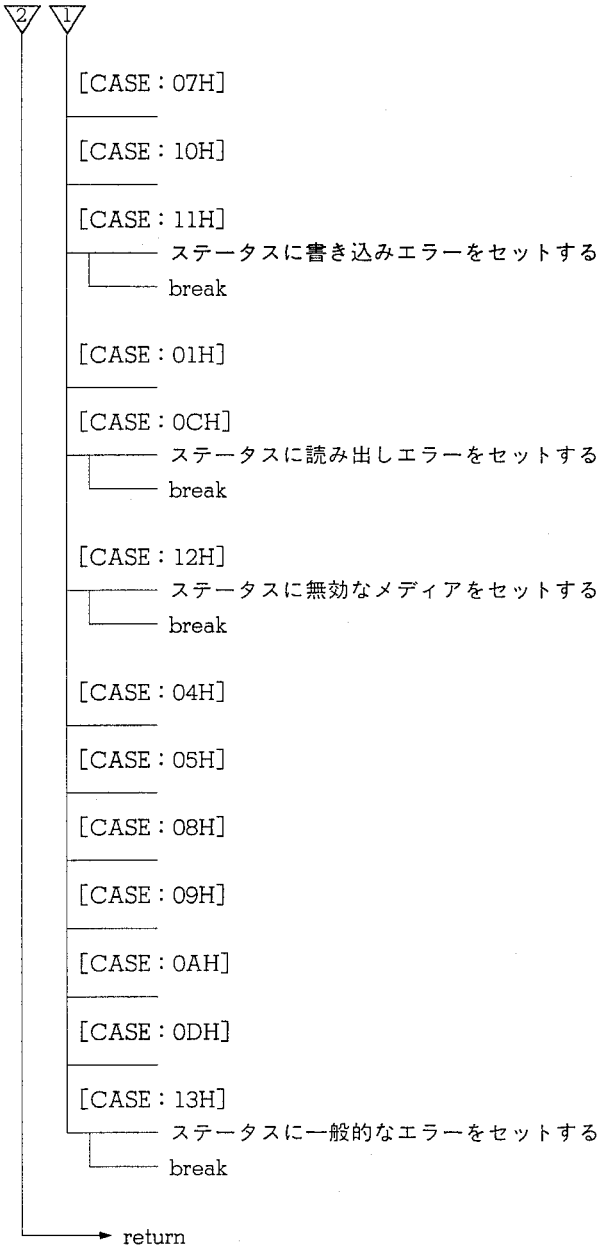
モジュール名	W_STA_PACK
ファイル名	SD_REQ.CPP
言語	C言語
入力	return_s[注]
出力	packet
機能	return_s(SCSIコマンド実行ブロックの実行結果)に従ってパケットのステータスをセットします。

注　**7.3　SCSIコマンド実行ブロックの変数**を参照してください。

SPDチャート



保守／廃止


2
1
[CASE：07H]
[CASE：10H]
[CASE：11H]
ステータスに書き込みエラーをセットする
break
[CASE：01H]
[CASE：0CH]
ステータスに読み出しエラーをセットする
break
[CASE：12H]
ステータスに無効なメディアをセットする
break
[CASE：04H]
[CASE：05H]
[CASE：08H]
[CASE：09H]
[CASE：0AH]
[CASE：0DH]
[CASE：13H]
ステータスに一般的なエラーをセットする
break
return

保守／廃止

第7章　SCSIコマンド実行ブロック

この章では，SCSIコマンド実行ブロックを説明します。SCSIコマンド実行ブロックはASPIによってSCSIコマンド（リード，ライト，ベリファイなど）を実行するモジュール・ブロックです。

7.1　SCSIコマンド実行ブロックの仕様

SCSIコマンド実行ブロックがサポートするモジュールを表7－1に示します。

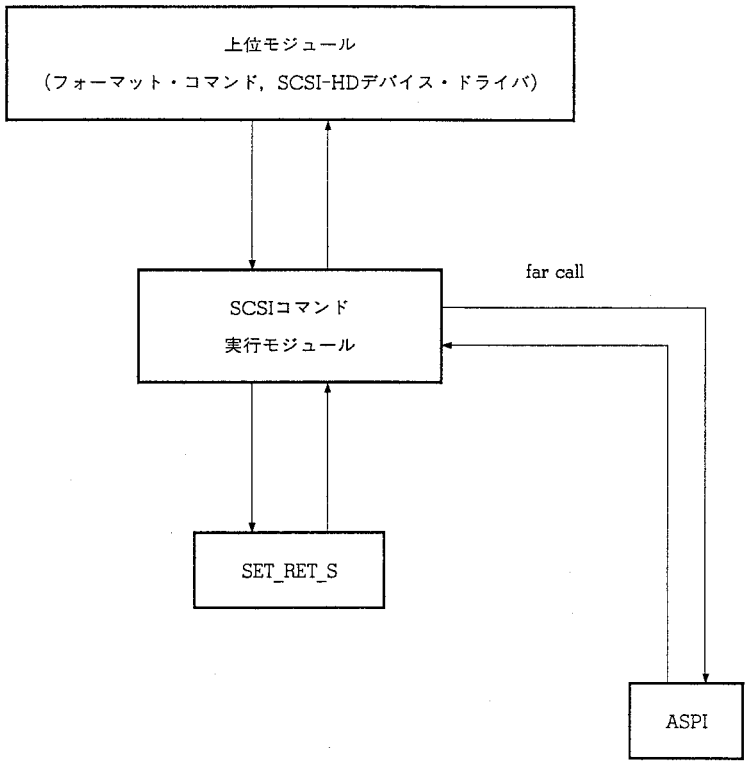
表7－1　SCSIコマンド実行ブロック・モジュール

モジュール名	処　　理
READ_DATA	ターゲットからデータをリードします。
WRITE_DATA	ターゲットへデータをライトします。
VERIFY	ターゲット内のデータをベリファイします。
WRITE_VERIFY	ターゲットへデータをライトし，ベリファイします。
SEEK	目的のブロックへヘッドをシークします。
RECALIBRATE	ヘッドをシリンダ0へシークします。
RETRACT	ヘッドをシッピング・ゾーンへシークします。
RESERVE	ターゲットをリザーブし，ほかのイニシエータからのアクセスを禁止します。
RELEASE	ターゲットをリリースし，ほかのイニシエータからのアクセスを許可します。
SEND_DIAG	ターゲットへ自己診断を要求します。
MODE_SENSE	ターゲットからデバイス・パラメータをリードします。
MODE_SELECT	ターゲットへデバイス・パラメータをライトします。
READ_CAPACITY	ターゲットから総ブロック数とブロック長をリードします。
FORMAT	ターゲットをフォーマットします。
INQUIRY	ターゲットからインクワイアリ・データをリードします。
T_U_READY	ターゲットがレディ状態であることを調べます。
T_INITIAL	接続されているターゲットを調べ，初期化します。

7.2　SCSIコマンド実行ブロックのプログラム構成

SCSIコマンド実行ブロックのプログラム構成を図７－１に示します。図に示すようにSCSIコマンド実行ブロック内のモジュールがMS-DOS上に常駐されているASPIをコールします。SET_RET_SはASPIの実行結果（SRB）に従って実行結果（return_s）をセットするモジュールです。

図７－１　SCSIコマンド実行ブロックのプログラム構成



保守／廃止

7.2.1　SCSIコマンド実行ブロックと上位モジュールのインタフェース

図７－２に上位モジュール（フォーマット・コマンド，SCSI-HDデバイス・ドライバ）とSCSIコマンド実行ブロックのインタフェースを示します。また，以下にインタフェース動作について説明します。

① 上位モジュールは必要な入力変数をセットします。

② 上位モジュールはSCSIコマンド実行ブロック内の必要なモジュールをコールします。

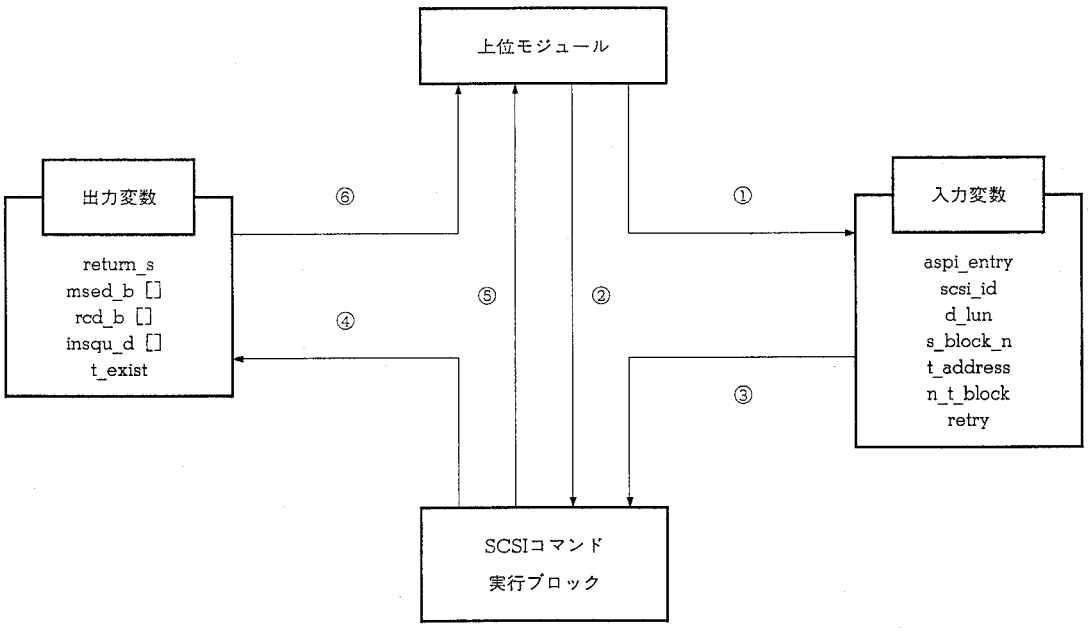
③ SCSIコマンド実行ブロック内のモジュールは入力変数を参照し，ASPIをコールすることによりコマンドを実行します。

④ SCSIコマンド実行ブロック内のモジュールは実行結果を出力変数にセットします。

⑤ SCSIコマンド実行ブロック内のモジュールは上位モジュールにリターンします。

⑥ 上位モジュールは出力変数を参照します。

図７－２　上位モジュールとのインタフェース



7.2.2　SCSIコマンド実行ブロックとASPIのインタフェース

SCSIコマンド実行ブロック内のモジュールはMS-DOS上に常駐されているASPIをコールします。SCSIコマンド実行ブロックとASPIのインタフェースの説明は，**9.2.1　ASPIとアプリケーション・プログラムのインタフェース**を参照してください。

7.3　SCSIコマンド実行ブロックの変数

表7-2にSCSIコマンド実行ブロックの入出力変数を示します。表7-3にリターン・ステータスの内容を示し，表7-4にASPIの実行結果（SRB）とリターン・ステータスの対応を示します。

表7-2　SCSIコマンド実行ブロック入出力変数

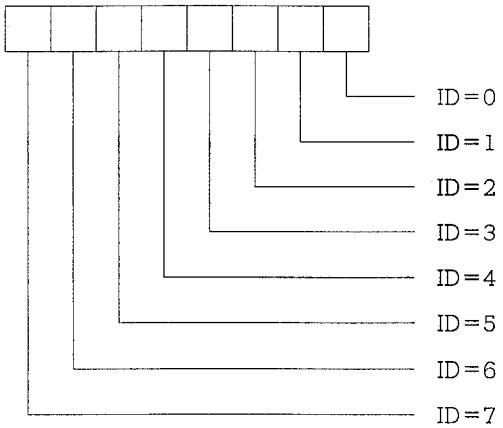
変　数		内　容
aspi_entry		ASPIのエントリ・ポイント
s_block_n	（32ビット）	開始ブロック番号
t_address	（32ビット）	転送アドレス
d_lun	（8ビット）	論理ユニット・ナンバ
n_t_block	（16ビット）	転送ブロック数
retry	（8ビット）	リトライ回数
msed_b []	（8ビット）	モード・センス・データ・バッファ
msld_b []	（8ビット）	モード・セレクト・データ・バッファ
rcd_b []	（8ビット）	リード・キャパシティ・データ・バッファ
scsi_id	（8ビット）	ターゲットのID番号
t_exist	（8ビット）	存在しているターゲットのID番号を示す 
return_s	（8ビット）	リターン・ステータス（SCSIコマンド実行ブロックの実行結果）

表7-3　リターン・ステータス

リターン・ステータス	内　　容
00H	正常終了
01H	エラー訂正されたデータをリードした
02H	ライト・プロテクト
03H	最大ブロック・アドレスを越えてアクセスしようとした
04H	装置に異常があった
05H	オーバラン・エラー，アンダラン・エラー
06H	ノット・レディ
07H	書き込み不可能状態を検出した
08H	イリーガル・コマンド
09H	一般的なエラー
0AH	タイム・アウト・エラー
0BH	データ・エラー（IDフィールド）
0CH	データ・エラー（データ・フィールド）
0DH	代替えブロックが読めない
0EH	目的のブロックが見つからない
0FH	シーク・エラー
10H	アドレス・マークが見つからない（IDフィールド）
11H	アドレス・マークが見つからない（データ・フィールド）
12H	モード・セレクト・データが変更された，リセットが発生した
13H	フォーマット・エラー
14H	デバイスがインストールされていない

表7－4　リターン・ステータス／SRB対応

リターン・ステータス	センス・コード	ASPI_Mステータス
00H	00H, 17H	No Target Status SCSI Request Completed without error
01H	18H, 1EH	
02H	27H	
03H	21H	
04H	01H, 08H, 19H, 1BH, 1CH, 43H, 44H, 46H, 47H, 48H	Unexpected bus free Target bus phase sequence failure SCSI request aborted by host
05H	1AH	Data over-run/under-run
06H	04H	
07H	03H, 09H	
08H	22H, 24H, 25H, 26H	
09H	05H, 07H, 20H, 40H, 41H, 42H, 49H	Specified Target/LUN is busy Reservation conflict
0AH		Selection timeout
0BH	10H	
0CH	11H, 16H	
0DH	32H	
0EH	14H	
0FH	02H, 06H, 15H	
10H	12H	
11H	13H	
12H	29H, 2AH	
13H	31H	
14H		SCSI device not installed

7.4　SCSIコマンド実行ブロック・モジュール説明

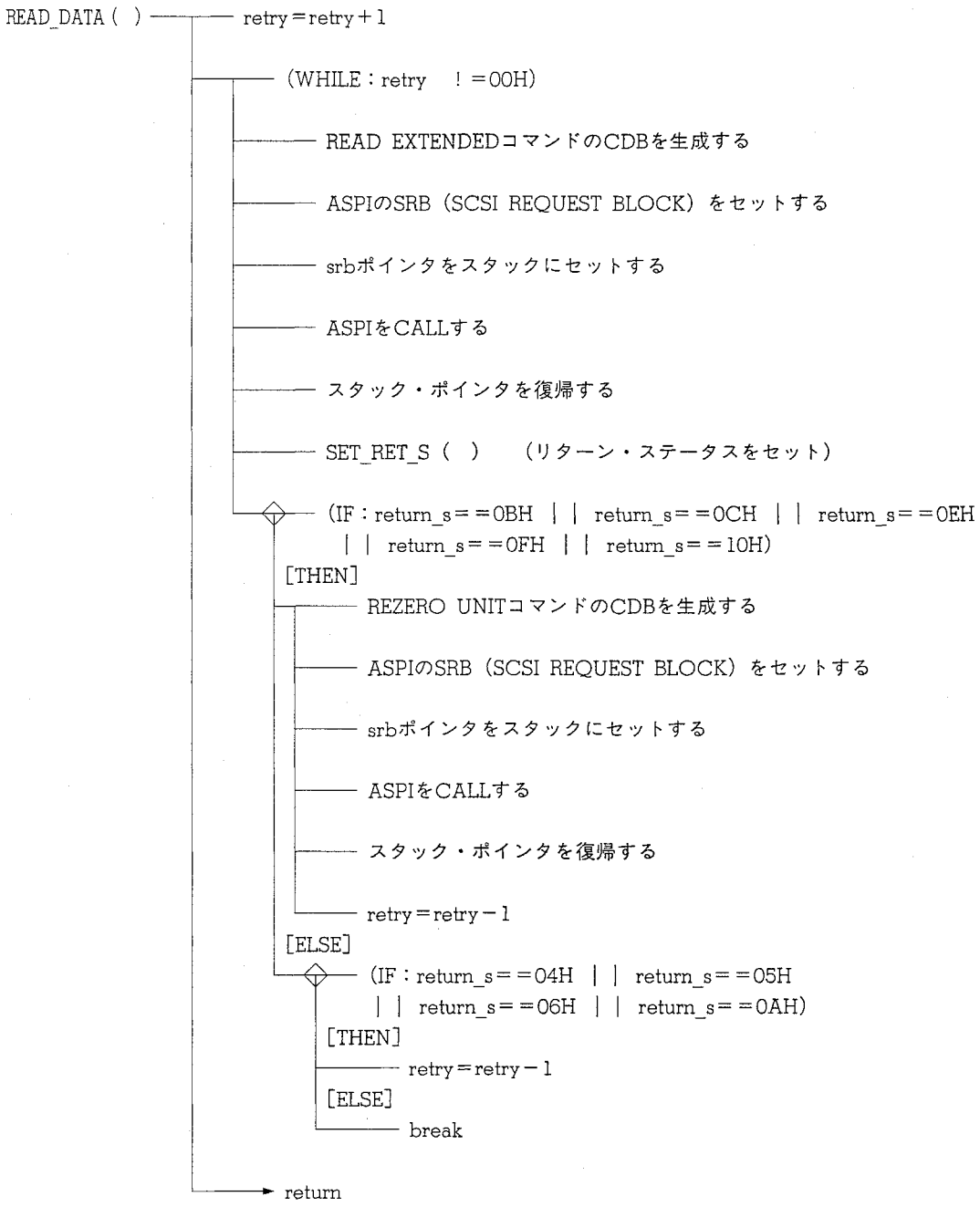
SCSIコマンド実行ブロックの各モジュールの説明をします。**第５章　フォーマット・コマンド**の表５－４に示す内容に沿って説明し，SPDチャートを示します。

(1) リード・データ処理

モジュール名	READ_DATA
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, s_block_n, t_address, n_t_block, retry
出力	return_s
機能	ターゲットからデータをリードします。

保守／廃止

SPDチャート

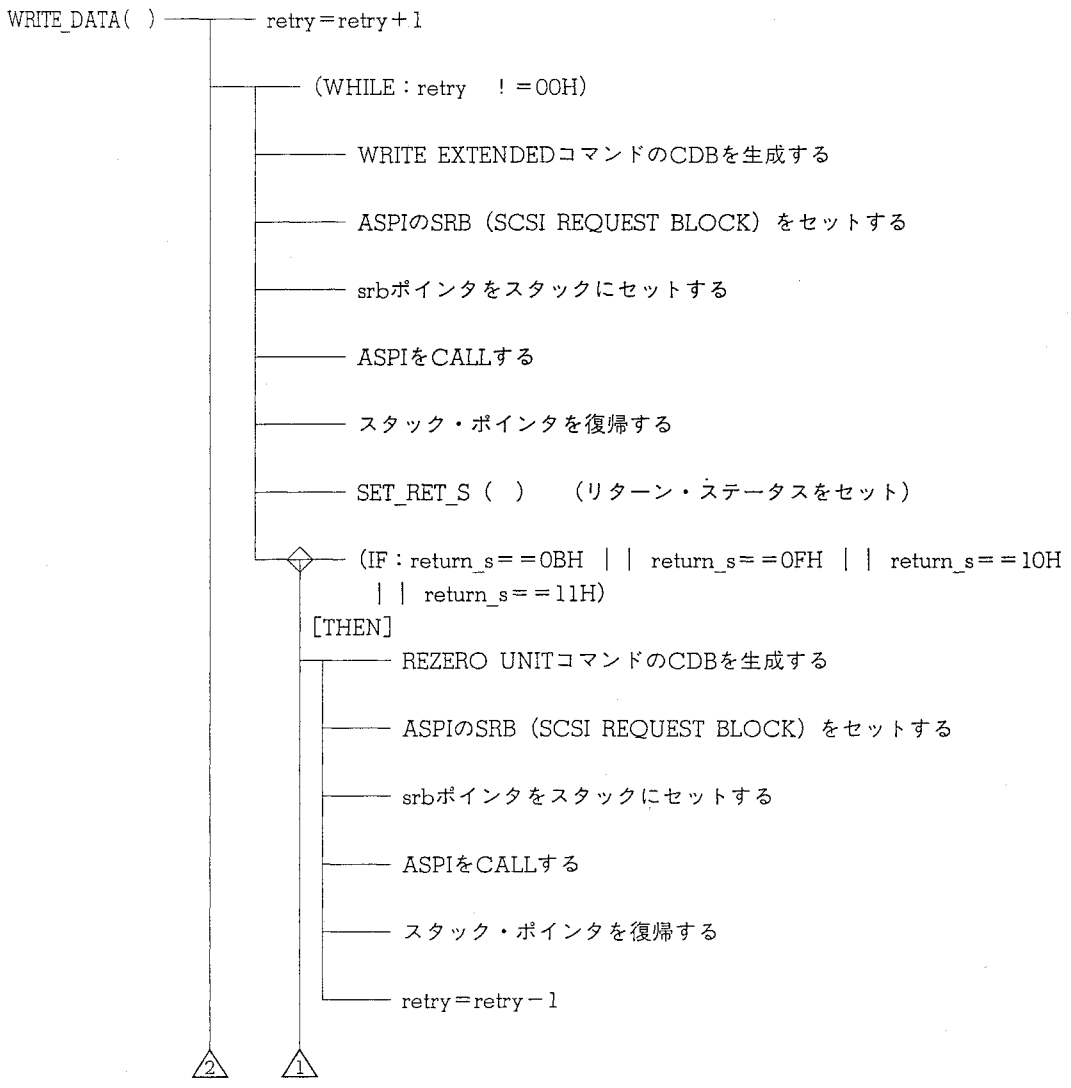


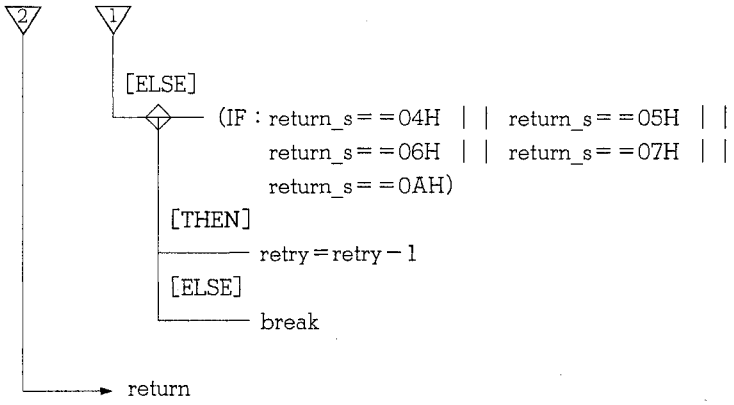
保守／廃止

(2) ライト・データ処理

モジュール名	WRITE_DATA
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, s_block_n, t_address, n_t_block, retry
出力	return_s
機能	ターゲットへデータをライトします。

SPDチャート

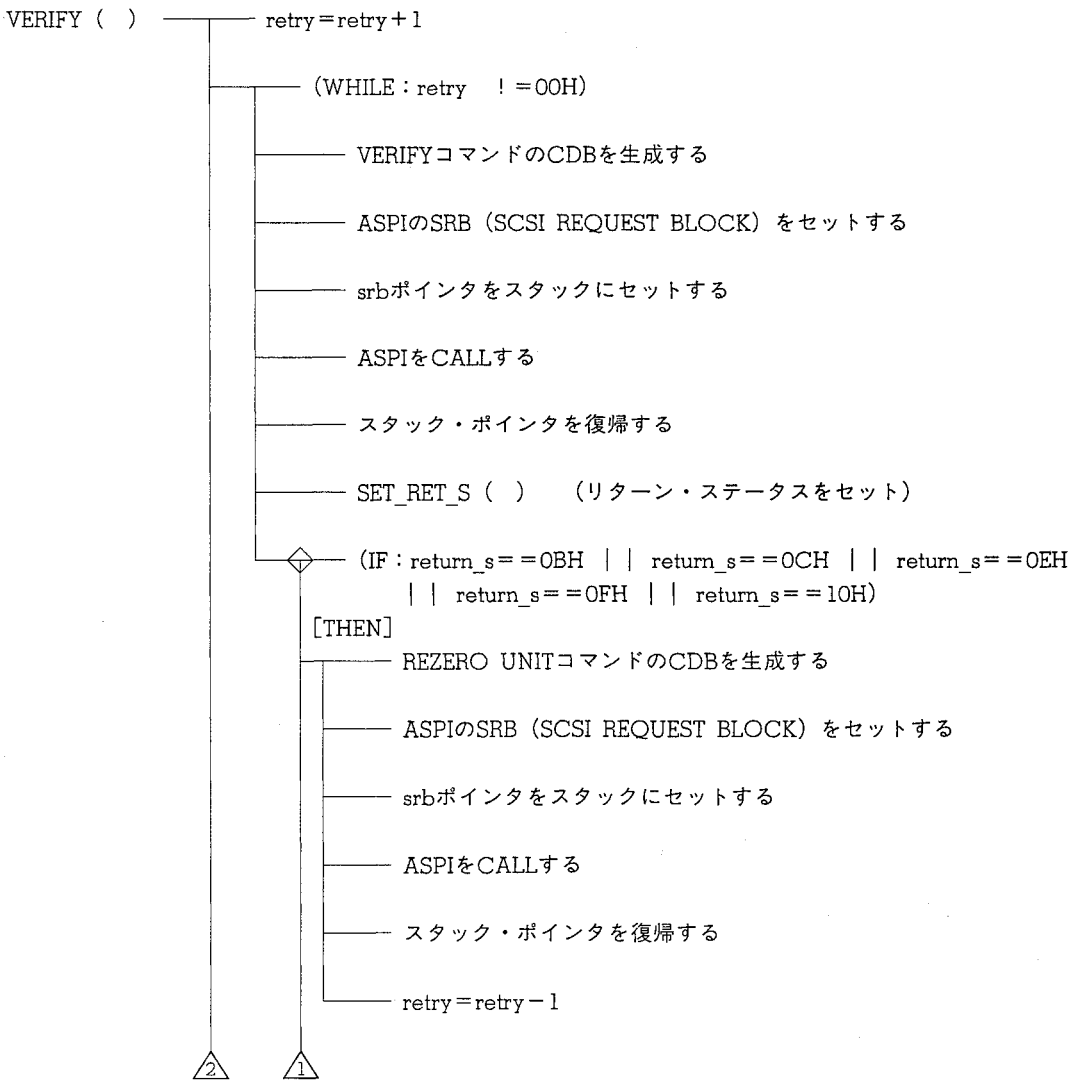


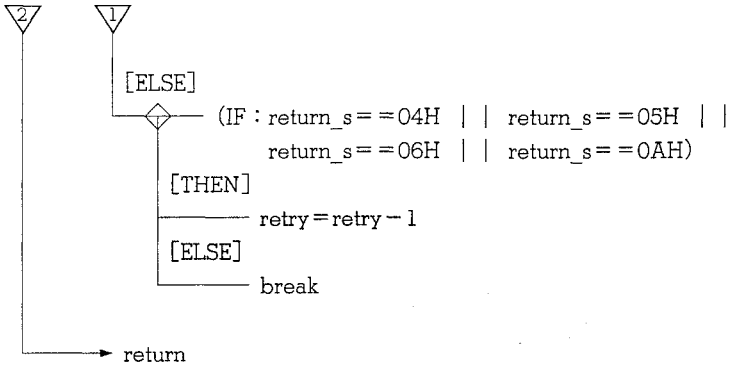

2
1
[ELSE]
(IF : return_s = =04H | | return_s = =05H | |
return_s = =06H | | return_s = =07H | |
return_s = =0AH)
[THEN]
retry = retry − 1
[ELSE]
break
return

(3) ベリファイ処理

モジュール名	VERIFY
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, s_block_n, n_t_block, retry
出力	return_s
機能	ターゲット内のデータをベリファイします。

SPDチャート

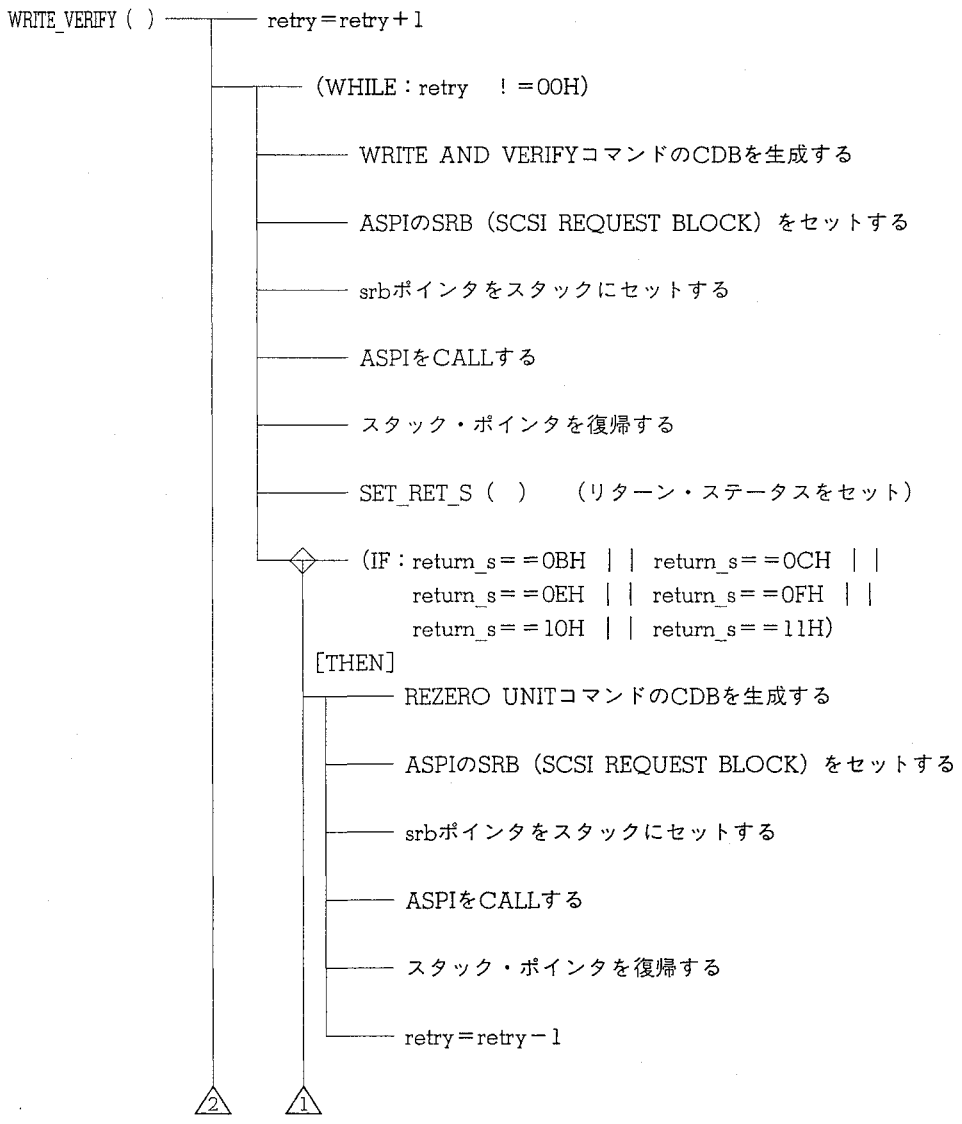


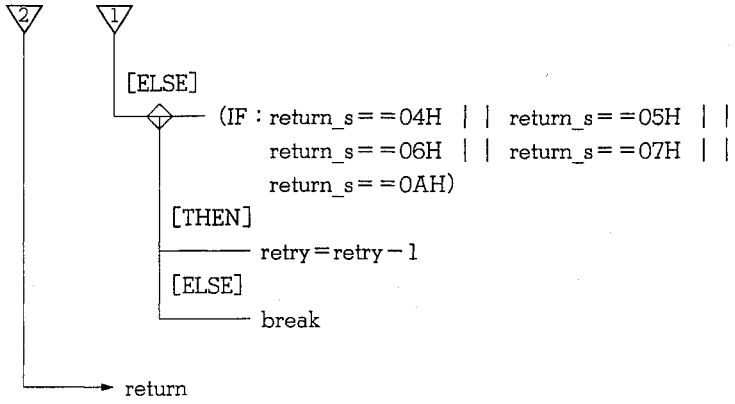

2
1
[ELSE]
(IF：return_s＝＝04H ｜｜ return_s＝＝05H ｜｜
return_s＝＝06H ｜｜ return_s＝＝0AH)
[THEN]
retry＝retry－1
[ELSE]
break
return

(4) ライト・アンド・ベリファイ処理

モジュール名	WRITE_VERIFY
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, s_block_n, t_address, n_t_block, retry
出力	return_s
機能	ターゲットへデータをライトし，ライトしたデータをベリファイします。

SPDチャート

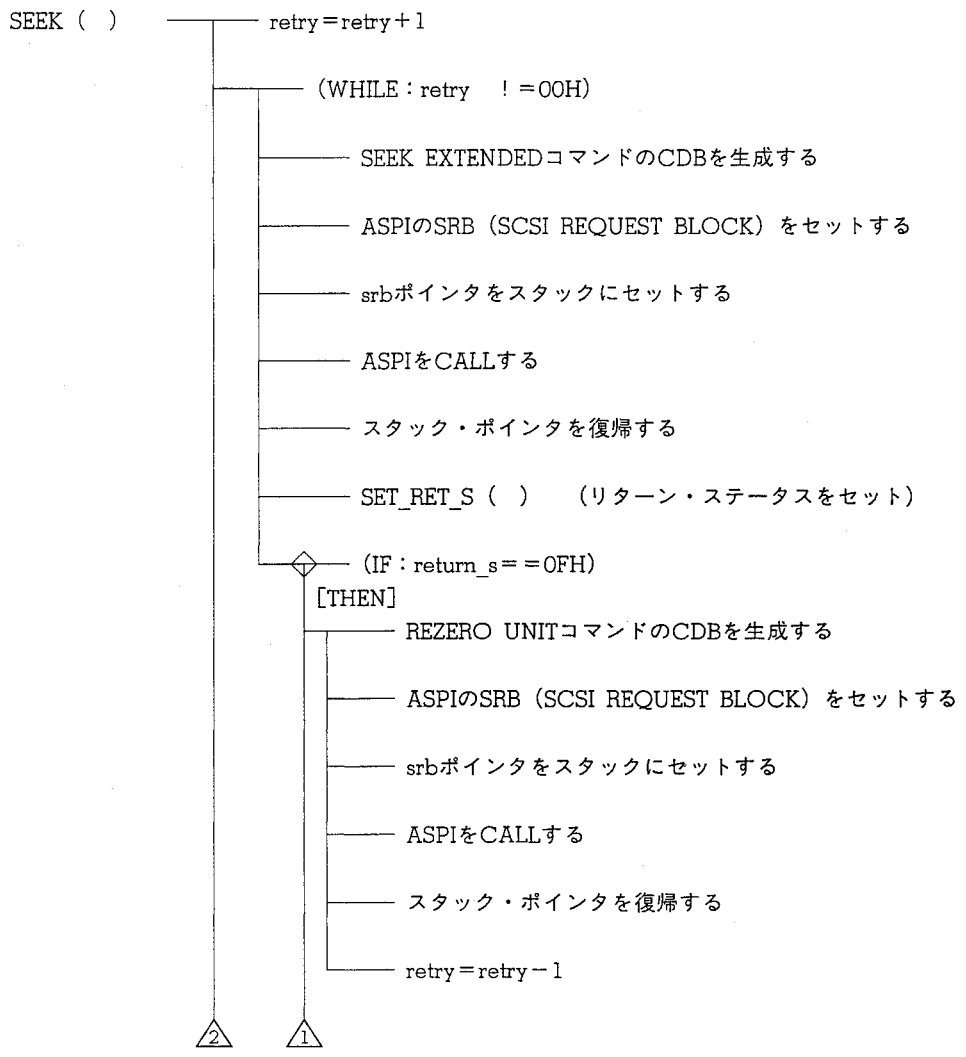


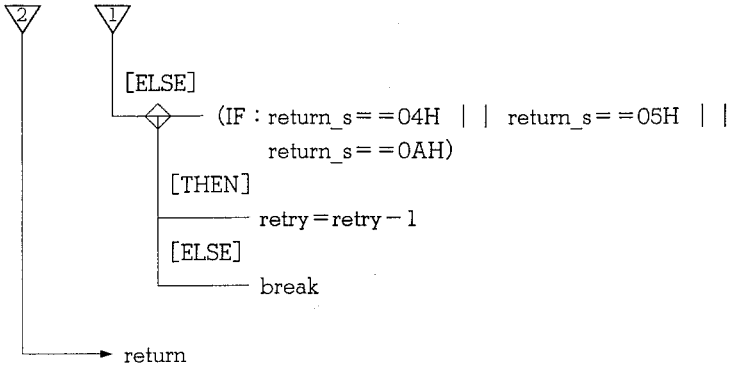

2
1
[ELSE]
(IF : return_s==04H || return_s==05H ||
return_s==06H || return_s==07H ||
return_s==0AH)
[THEN]
retry=retry−1
[ELSE]
break
return

(5) シーク処理

モジュール名	SEEK
ファイル名	SCOM. CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, s_block_n, retry
出力	return_s
機能	目的ブロック・アドレスへヘッドをシークします。

SPDチャート

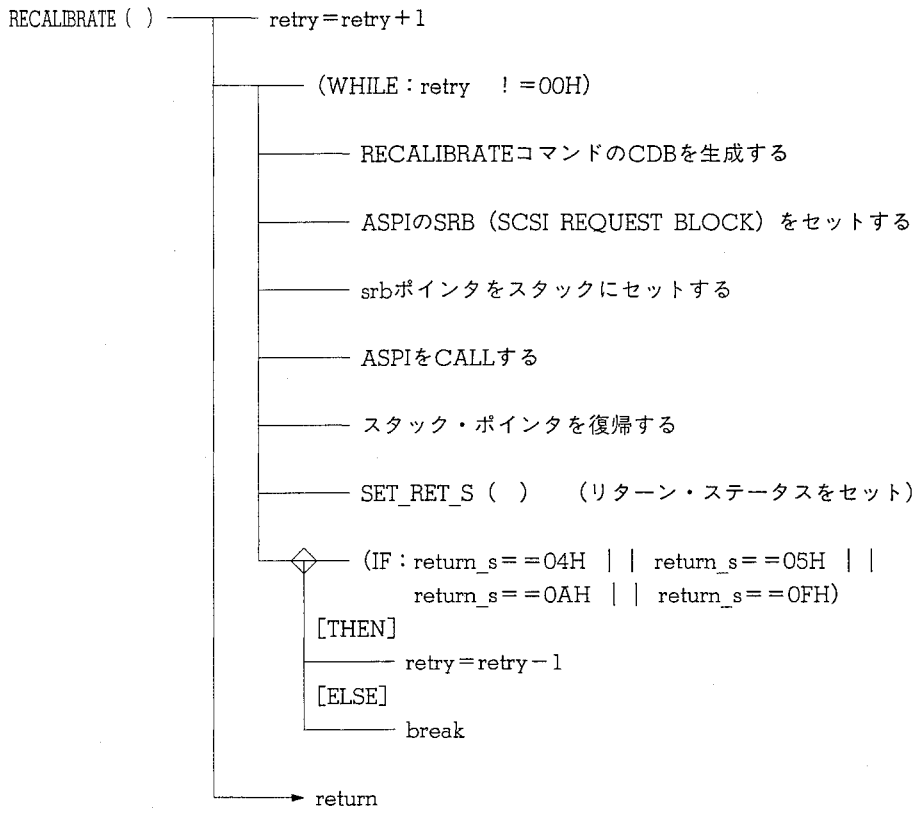



2
1
[ELSE]
(IF：return_s＝＝04H ｜｜ return_s＝＝05H ｜｜
return_s＝＝0AH)
[THEN]
retry＝retry－1
[ELSE]
break
return

(6) リキャリブレイト処理

モジュール名	RECALIBRATE
ファイル名	SCOM. CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ヘッドをシリンダ0へシークします。

SPDチャート



(7) リトラクト処理

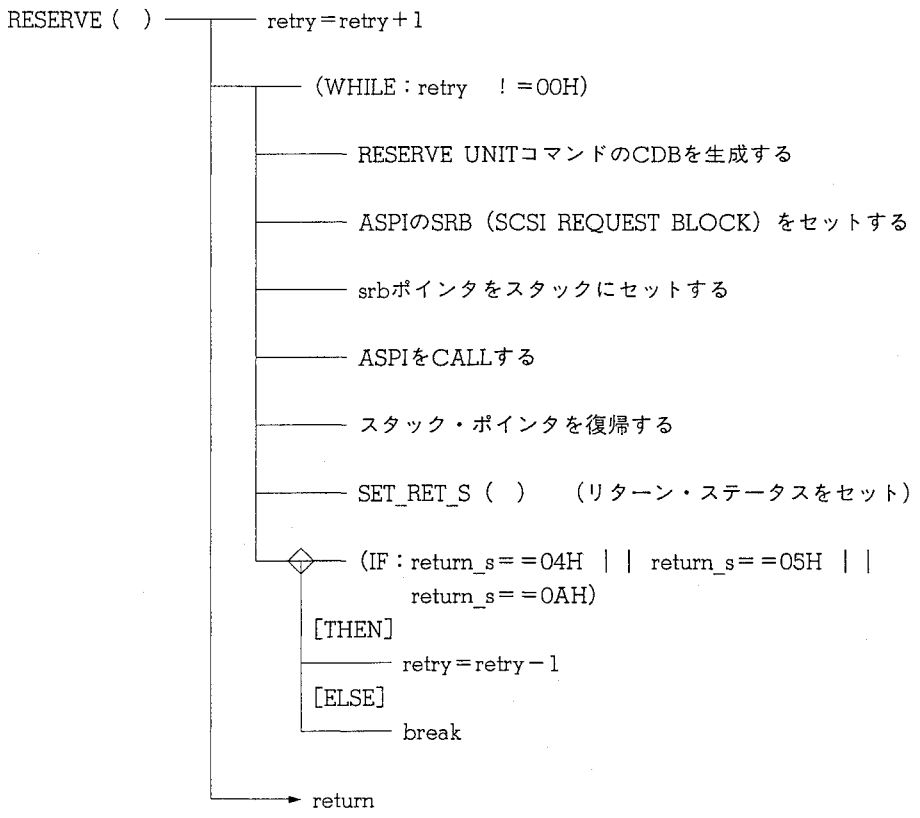
モジュール名	RETRACT
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ヘッドをシッピング・ゾーンへシークします。

SPDチャート

(8) リザーブ処理

モジュール名	RESERVE
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ターゲットをリザーブし，ほかのイニシエータからのアクセスを禁止します。

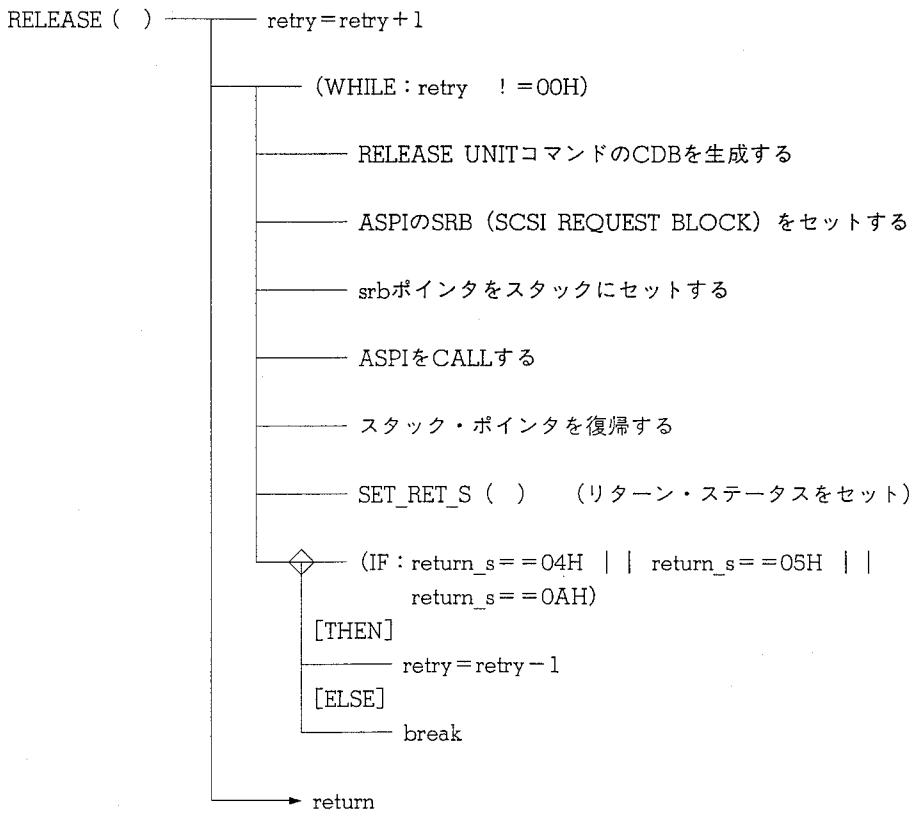
SPDチャート



(9) リリース処理

モジュール名	RELEASE
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ターゲットをリリースし，ほかのイニシエータからのアクセスを許可します。

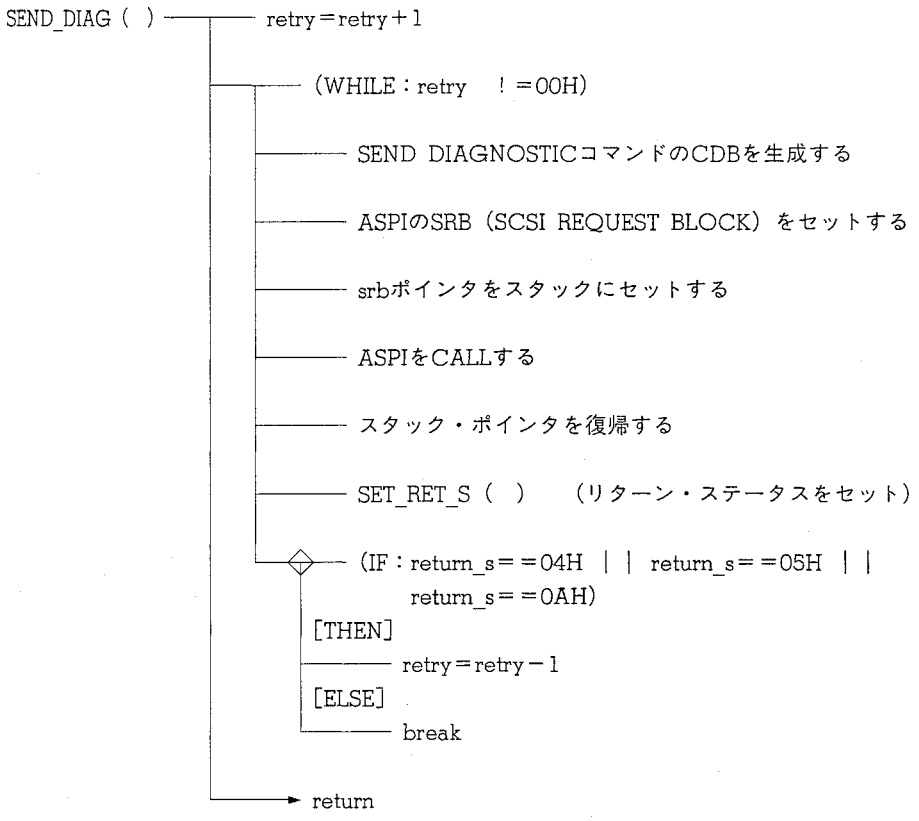
SPDチャート



(10) センド・ダイアグノスティック処理

モジュール名	SEND_DIAG
ファイル名	SCOM. CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ターゲットへ自己診断を要求します。

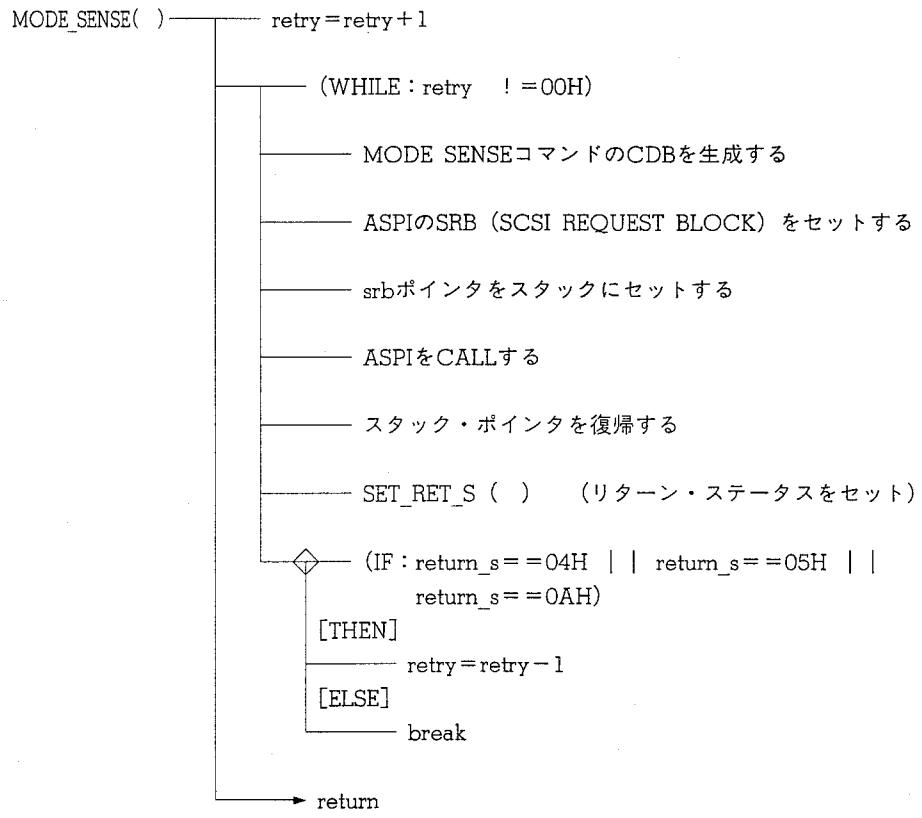
SPDチャート



(11) モード・センス処理

モジュール名	MODE_SENSE
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s, msed_b []
機能	ターゲットからデバイス・パラメータをリードします。

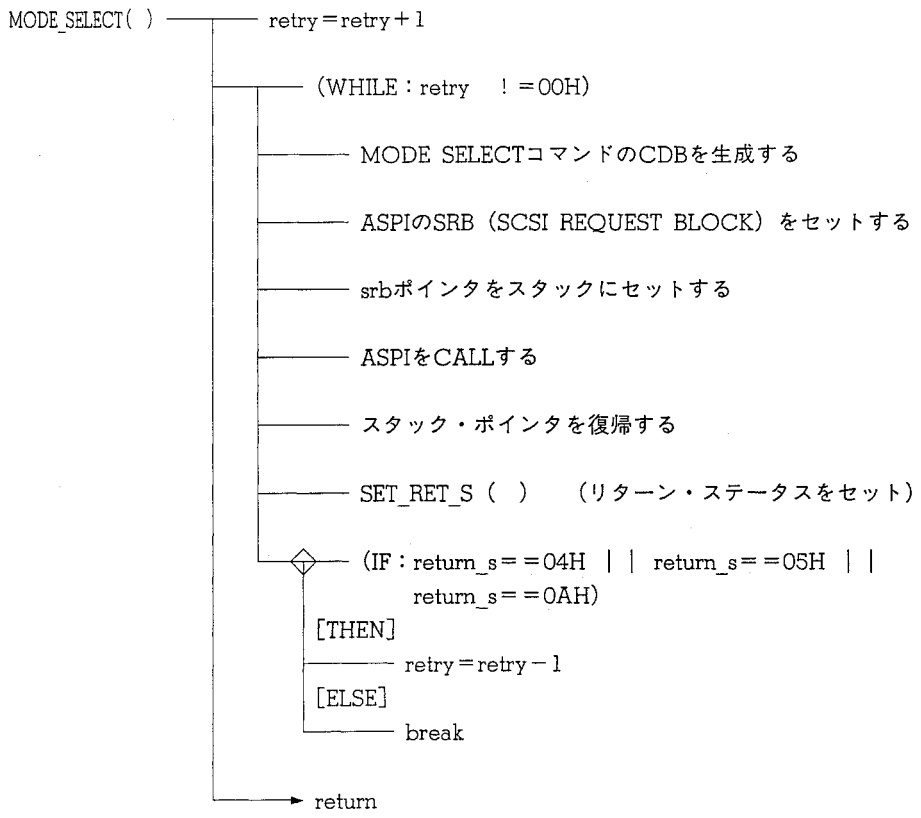
SPDチャート



(12) モード・セレクト処理

モジュール名	MODE_SELECT
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ターゲットへデバイス・パラメータをセットします。

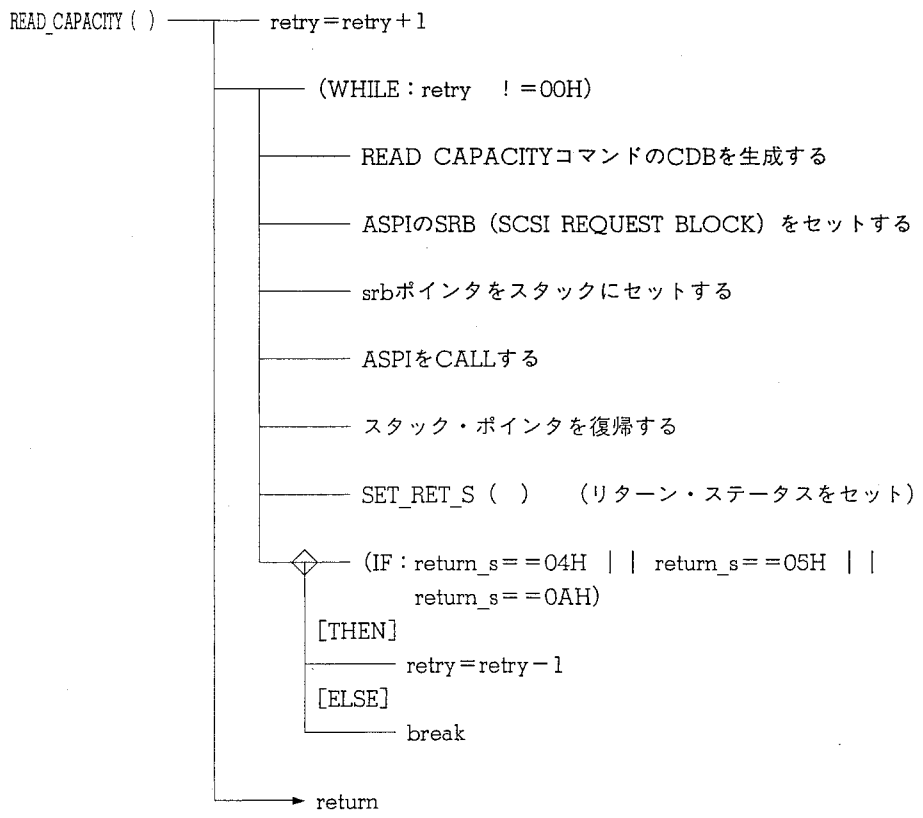
SPDチャート



(13) リード・キャパシティ処理

モジュール名	READ_CAPACITY
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s, rcd_b []
機能	ターゲットから総ブロック数とブロック長をリードします。

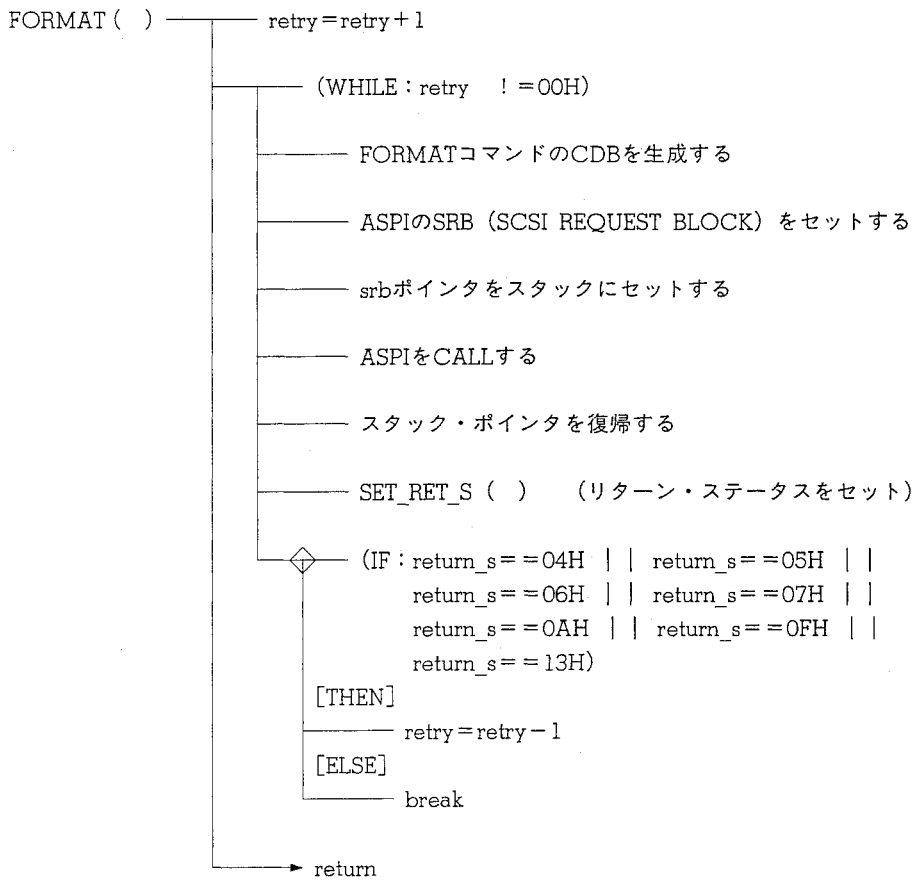
SPDチャート



(14) フォーマット処理

モジュール名	FORMAT
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s
機能	ターゲットをフォーマットします。

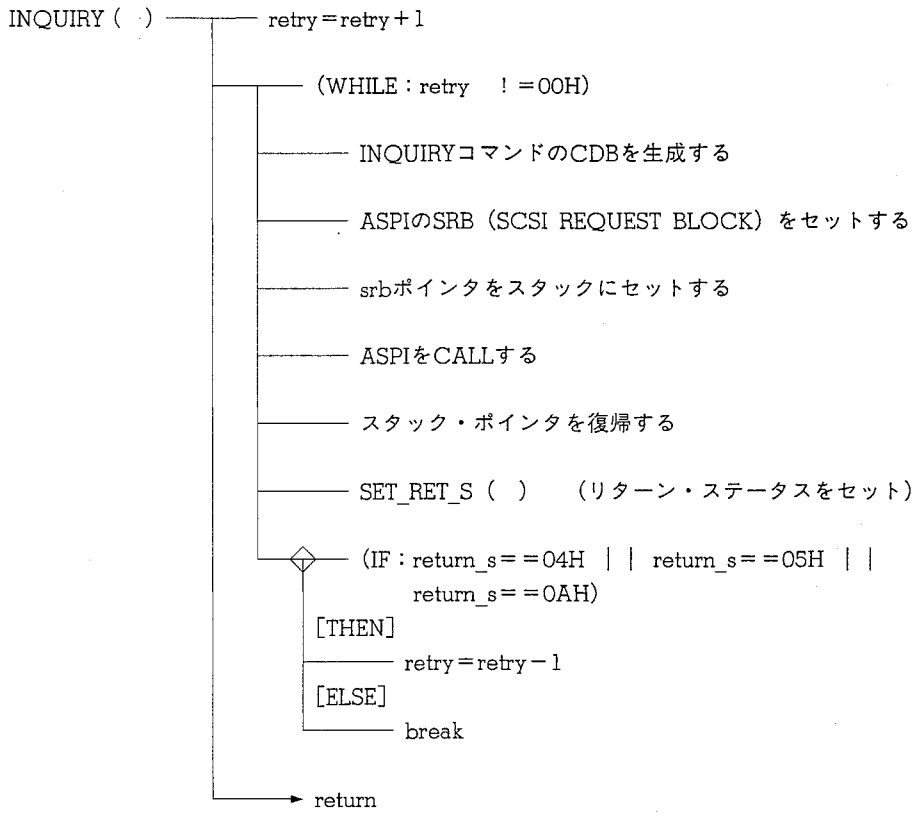
SPDチャート



(15) インクワイアリ処理

モジュール名	INQUIRY
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun, retry
出力	return_s, inqu_d []
機能	ターゲットからインクワイアリ・データをリードします。

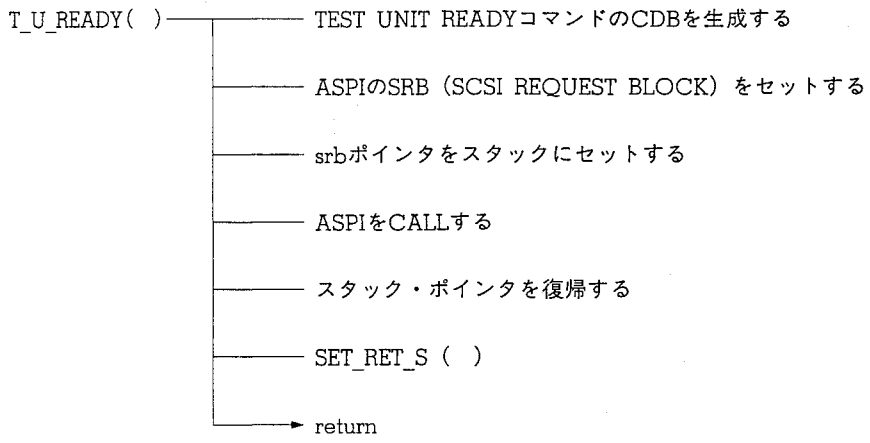
SPDチャート



(16) テスト・ユニット・レディ処理

モジュール名	T_U_READY
ファイル名	SCOM.CPP
言語	C言語
入力	aspi_entry, scsi_id, d_lun
出力	return_s
機能	ターゲットがレディ状態であることを調べます。

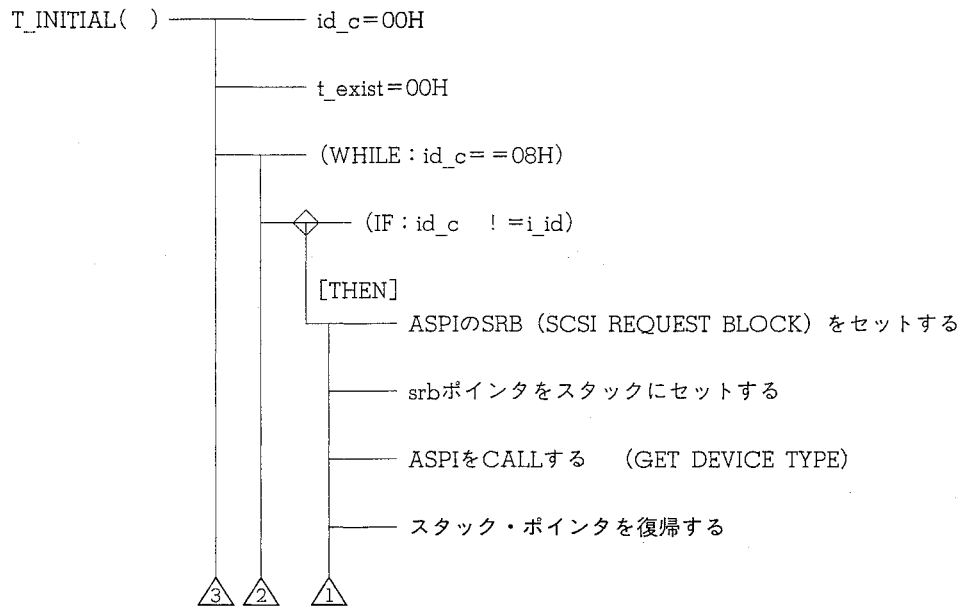
SPDチャート

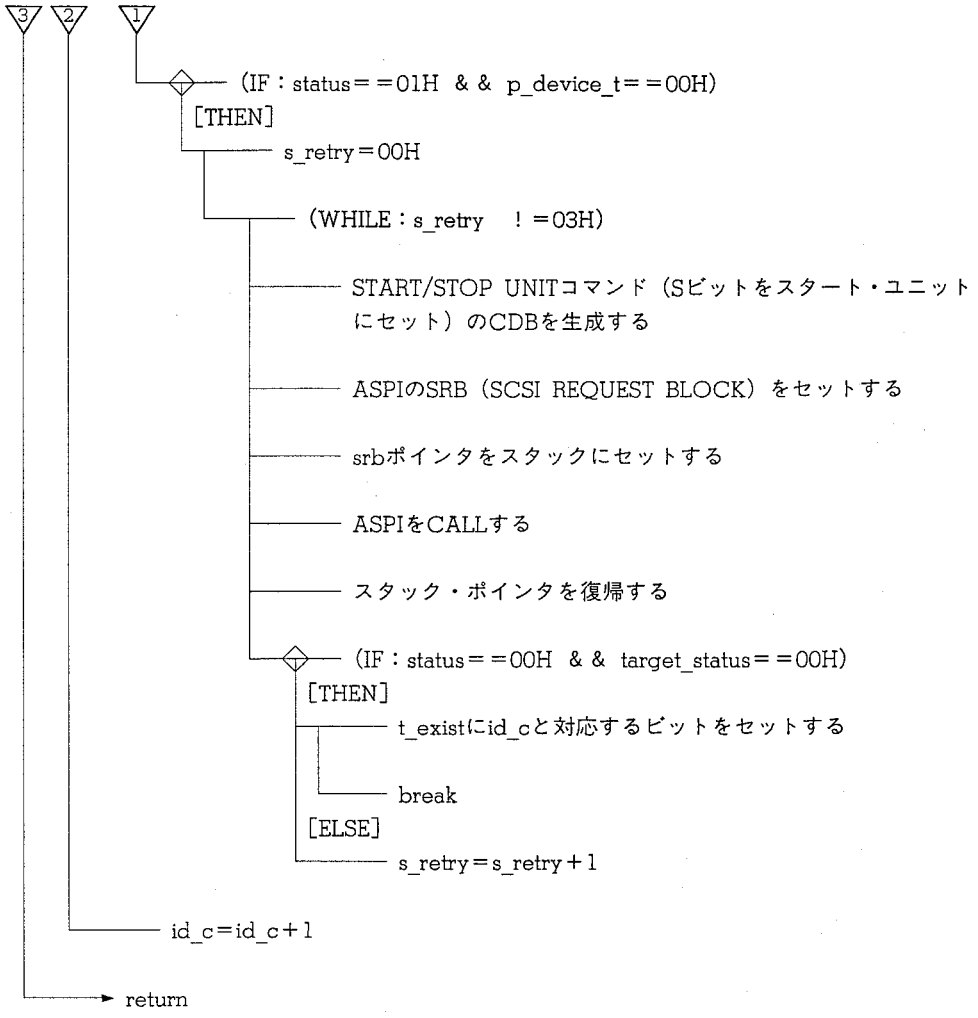


(17) ターゲットの初期化処理

モジュール名	T_INITIAL
ファイル名	SCOM. CPP
言語	C言語
入力	aspi_entry，d_lun
出力	t_exist
機能	接続されているターゲットを調べ，初期化します。

SPDチャート

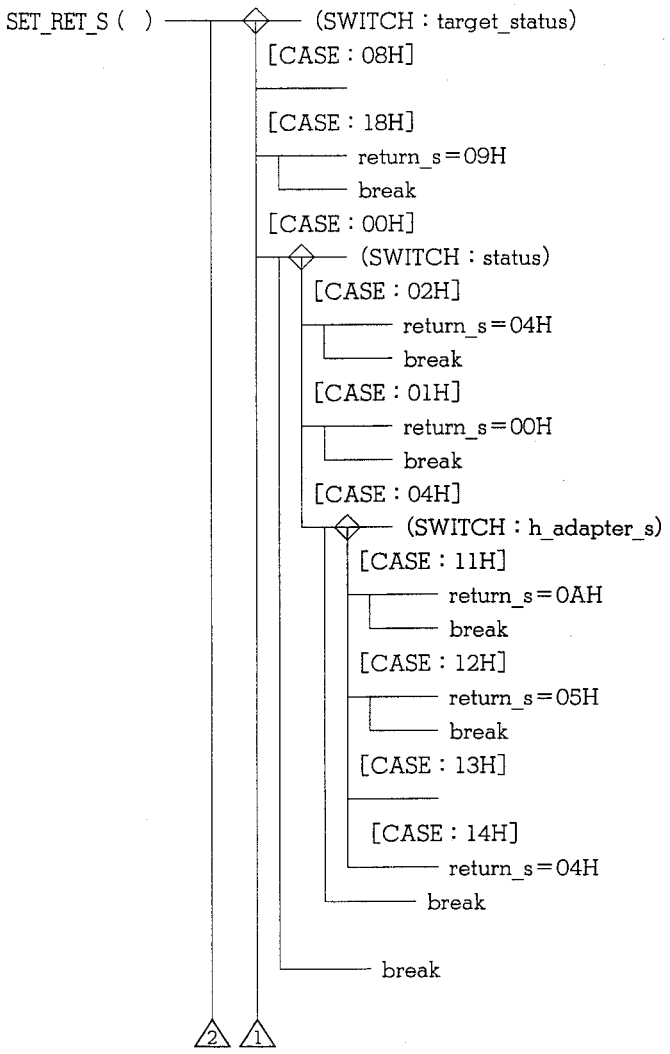


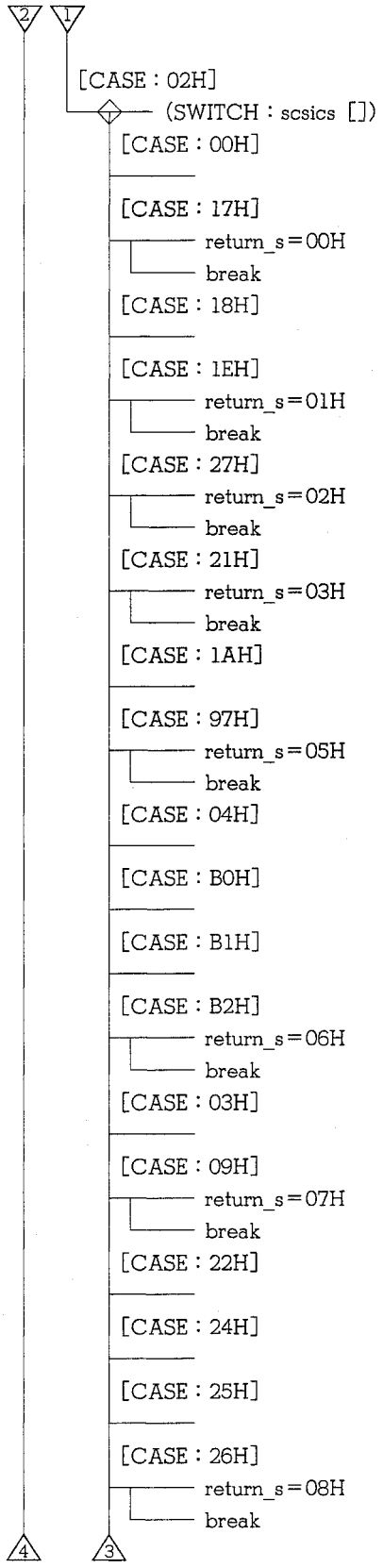

3
2
1
(IF：status＝＝01H　& &　p_device_t＝＝00H)
[THEN]
s_retry＝00H
(WHILE：s_retry　！＝03H)
START/STOP UNITコマンド（Sビットをスタート・ユニットにセット）のCDBを生成する
ASPIのSRB（SCSI REQUEST BLOCK）をセットする
srbポインタをスタックにセットする
ASPIをCALLする
スタック・ポインタを復帰する
(IF：status＝＝00H　& &　target_status＝＝00H)
[THEN]
t_existにid_cと対応するビットをセットする
break
[ELSE]
s_retry＝s_retry＋1
id_c＝id_c＋1
return

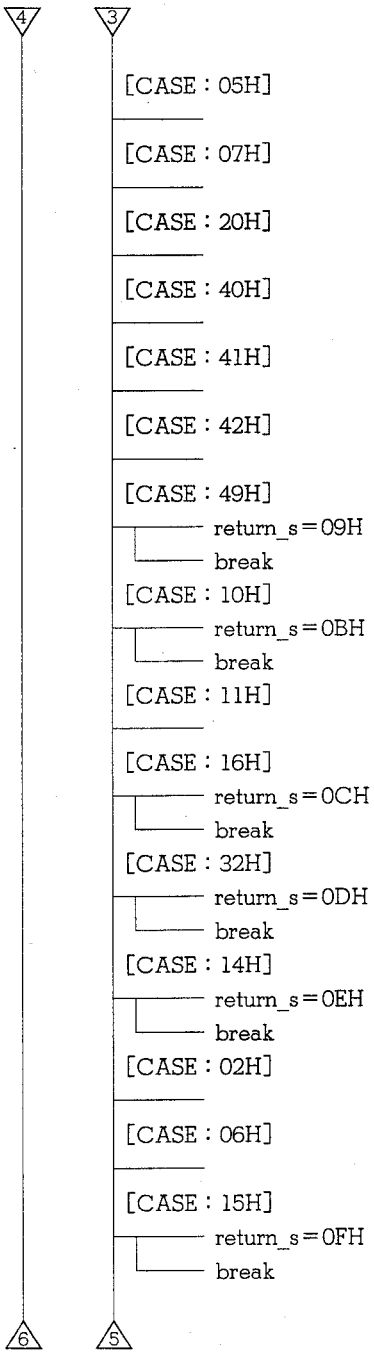
(18) リターン・ステータス・セット処理

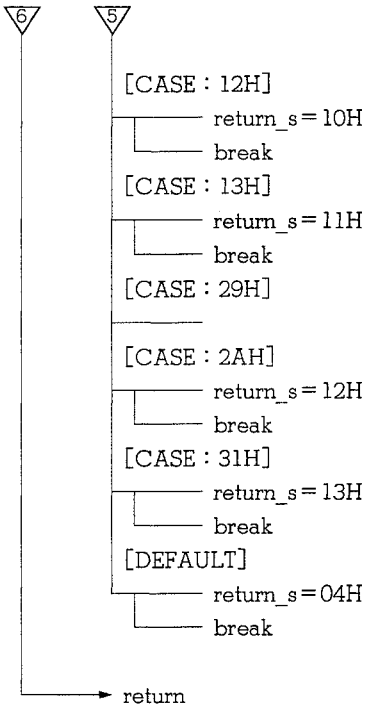
モジュール名	SET_RET_S
ファイル名	SCOM.CPP
言語	C言語
入力	srb
出力	return_s
機能	srbの結果からリターン・ステータスをセットします。

SPDチャート




2
1
[CASE : 02H]
(SWITCH : scsics [])
[CASE : 00H]
[CASE : 17H]
return_s=00H
break
[CASE : 18H]
[CASE : 1EH]
return_s=01H
break
[CASE : 27H]
return_s=02H
break
[CASE : 21H]
return_s=03H
break
[CASE : 1AH]
[CASE : 97H]
return_s=05H
break
[CASE : 04H]
[CASE : B0H]
[CASE : B1H]
[CASE : B2H]
return_s=06H
break
[CASE : 03H]
[CASE : 09H]
return_s=07H
break
[CASE : 22H]
[CASE : 24H]
[CASE : 25H]
[CASE : 26H]
return_s=08H
break
4
3


4
3
[CASE : 05H]
[CASE : 07H]
[CASE : 20H]
[CASE : 40H]
[CASE : 41H]
[CASE : 42H]
[CASE : 49H]
return_s＝09H
break
[CASE : 10H]
return_s＝0BH
break
[CASE : 11H]
[CASE : 16H]
return_s＝0CH
break
[CASE : 32H]
return_s＝0DH
break
[CASE : 14H]
return_s＝0EH
break
[CASE : 02H]
[CASE : 06H]
[CASE : 15H]
return_s＝0FH
break
6
5


6
5
[CASE : 12H]
return_s＝10H
break
[CASE : 13H]
return_s＝11H
break
[CASE : 29H]
[CASE : 2AH]
return_s＝12H
break
[CASE : 31H]
return_s＝13H
break
[DEFAULT]
return_s＝04H
break
return

保守／廃止

第８章　ASPIマネージャ

この章ではASPIマネージャを説明します。ASPIマネージャはASPIをMS-DOS上に常駐させるためのキャラクタ型のデバイス・ドライバです。

8.1　ASPIマネージャの仕様

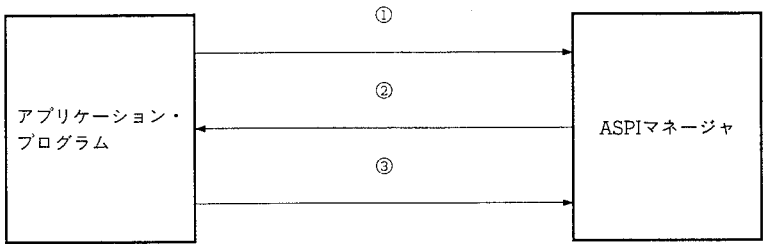
表８－１にデバイス・ヘッダを示します。

ASPIマネージャはターゲットおよびデバイス（μPD72611，割り込みコントローラ）のイニシャライズとASPIのエントリ・ポイントをASPIを利用するアプリケーション・プログラム（フォーマット・コマンド，SCSIデバイス・ドライバなど）にMS-DOSを介して渡します。図８－１にASPIエントリ・ポイントの受け渡し方法を示します。

表８－１　デバイス・ヘッダ

値	意　　味
－1	リンク情報
C000H	デバイス属性
STRATEGY	ストラテジ・エントリ・ポイント
INTERRUPT	割り込みエントリ・ポイント
“SCSIMGR$”	デバイス名

図８－１　ASPIエントリ・ポイントの受け渡し方法



① アプリケーション・プログラムはASPIマネージャをファイル・オープンします。

② アプリケーション・プログラムはIOCTLインプットによってASPIエントリ・ポイントをASPIマネージャから受け取ります。

③ アプリケーション・プログラムはASPIマネージャをファイル・クローズします。

保守／廃止

8.2　ASPIマネージャのプログラム構成

図８－２にプログラム構成を示します。またMS-DOSがデバイス・ドライバをアクセスする動作を以下に示します。図８－２に示すようにアプリケーション・ソフトはMS-DOSを介してアクセスするのではなく，ASPIエントリ・ポイントを得て直接ASPIをコールすることによりアクセスします。

このデバイス・ドライバがサポートするコマンド・コードに対応する処理を表８－２に示します。

① MS-DOSはコマンド・パケットのポインタをES，BXレジスタにセットしてストラテジ・ルーチンをコールします。ストラテジ・ルーチンはコマンド・パケットのポインタをドライバ内にセーブしてMS-DOSへリターンします。

② MS-DOSは割り込みエントリ・ルーチンをコールします。

③ 割り込みエントリ・ルーチンはI/Oリクエスト・コマンド処理をコールします。

④ I/Oリクエスト・コマンド処理はコマンド・パケット内のコマンド・コードに従って各処理（コマンド・コード対応処理）をコールします。

⑤ コマンド・コード対応処理は実行後I/Oリクエスト・コマンド処理へリターンします。

⑥ I/Oリクエスト・コマンド処理は割り込みエントリ・ルーチンにリターンします。割り込みエントリ・ルーチンはMS-DOSにリターンします。

保守／廃止

図８－２　ASPIマネージャ・プログラム構成

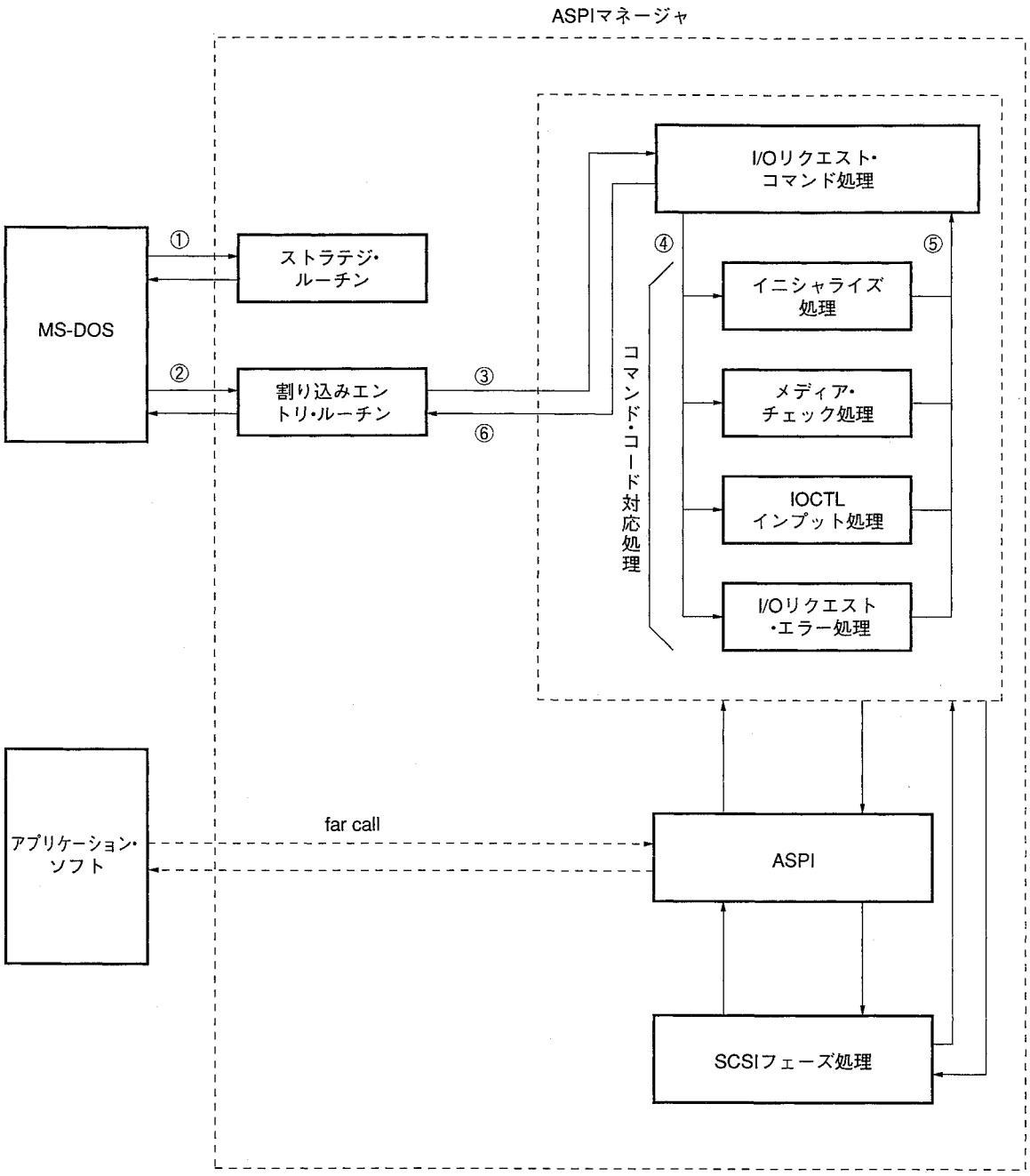


表8－2　コマンド・コード対応処理

I/Oリクエスト・コード	コマンド・コード対応処理
00H	イニシャライズ処理
03H	IOCTLインプット処理
その他	I/Oリクエスト・エラー処理

8.2.1　I/Oコマンド・コード対応処理とSCSIフェーズ処理のインタフェース

I/Oコマンド・コード対応処理とSCSIフェーズ処理のインタフェースの説明は，**10.2.3　SCSIフェーズ処理と上位モジュールのインタフェース**を参照してください。

8.3　ASPIマネージャの変数

表8－3にASPIマネージャの入出力変数を示します。

表8－3　ASPIマネージャの変数

変　数　名	意　　味
packet（32ビット）	コマンド・パケットのポインタ（far pointer）
prog_end（32ビット）	SCSI-HDデバイス・ドライバの終了アドレス（far pointer）

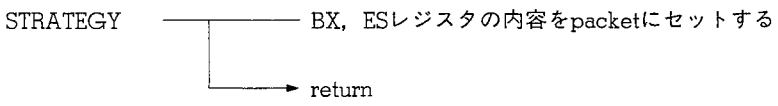
8.4　ASPIマネージャ・モジュール説明

ASPIマネージャの各モジュールの説明をします。**第5章　フォーマット・コマンド**の表5－4に示す内容によって説明し，SPDチャートを示します。

(1) ストラテジ・ルーチン

モジュール名	STRATEGY
ファイル名	ASSUP.ASM
言語	アセンブラ
入力	BXレジスタ：コマンド・パケットのオフセット・アドレス ESレジスタ：コマンド・パケットのセグメント・アドレス
出力	packet
機能	MS-DOSからBX，ESレジスタを使用して渡してくるコマンド・パケットのポインタをpacketに格納します。

SPDチャート

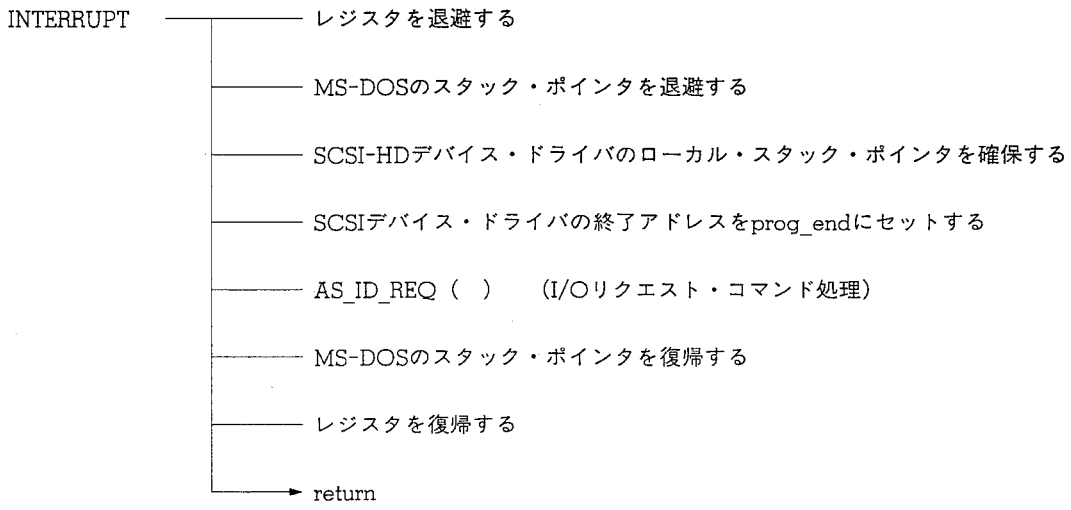


保守／廃止

(2) 割り込みエントリ・ルーチン

モジュール名	INTERRUPT
ファイル名	ASSUP.ASM
言語	アセンブラ
入力	なし
出力	prog_end
機能	SCSI-HDデバイス・ドライバの終了アドレスをprog_endにセットしI/Oリクエスト・コマンド処理をコールします。

SPDチャート

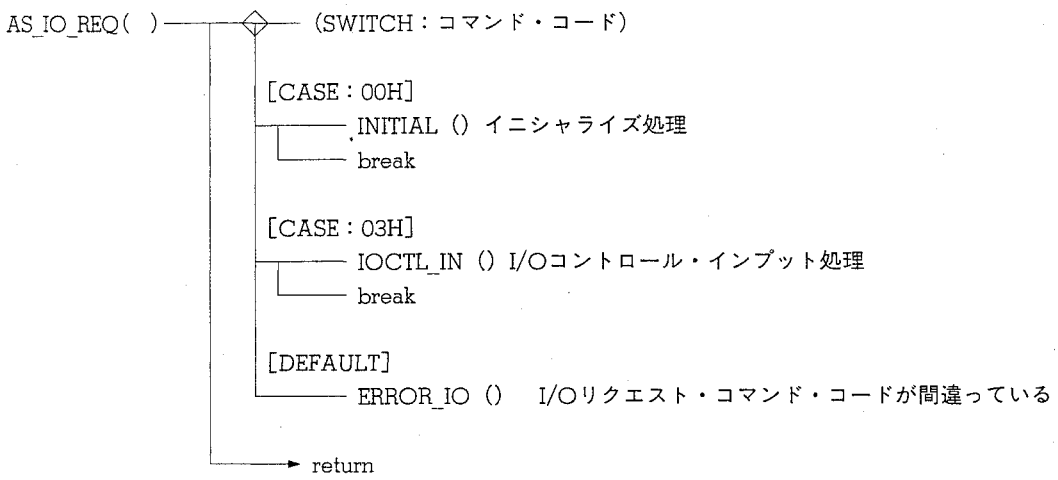


保守／廃止

(3) I/Oリクエスト・コマンド処理

モジュール名	AS_IO_REQ
ファイル名	AS_REQ.CPP
言語	C言語
入力	packet
出力	なし
機能	I/Oリクエスト・コマンド・コードに対応した処理をコールします。

SPDチャート

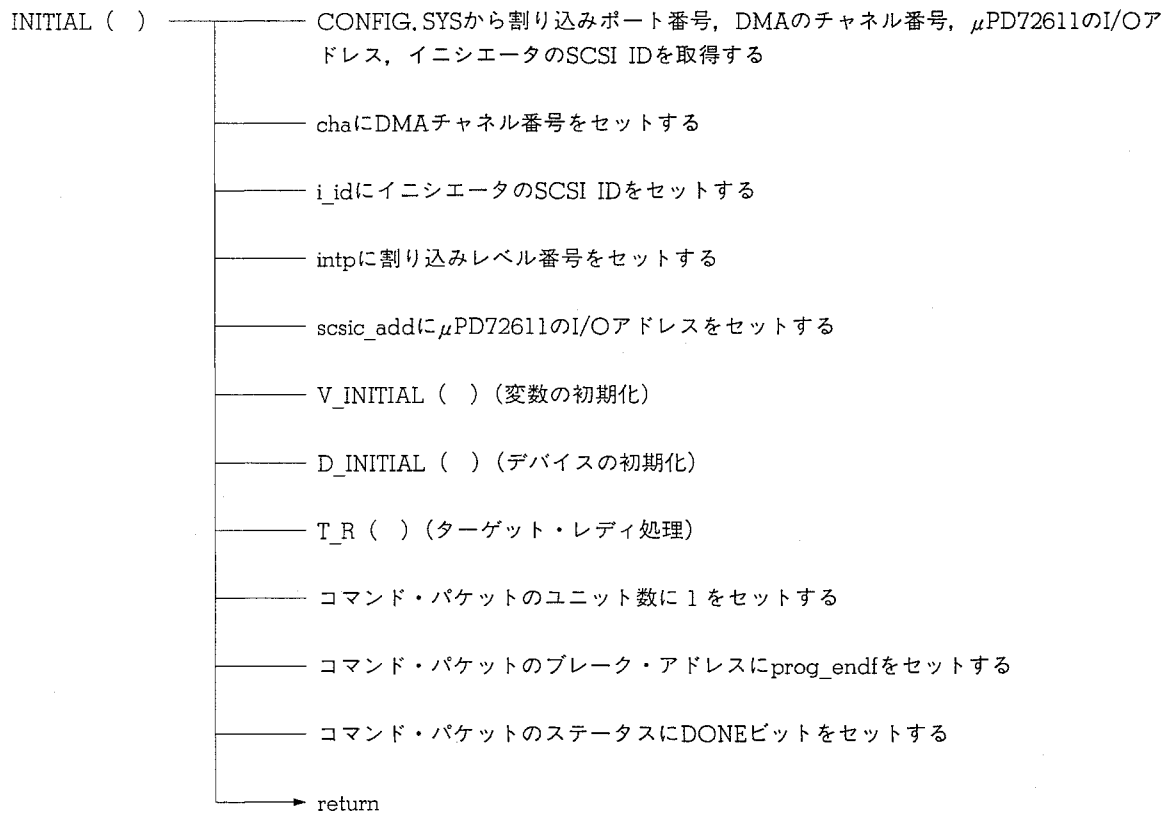


保守／廃止

(4) イニシャライズ処理

モジュール名	INITIAL
ファイル名	AS_REQ.CPP
言語	C言語
入力	packet, prog_end
出力	packet
機能	μPD72611, 割り込みコントローラ, DMAコントローラのイニシャライズおよびターゲットをレディ状態にします。

SPDチャート

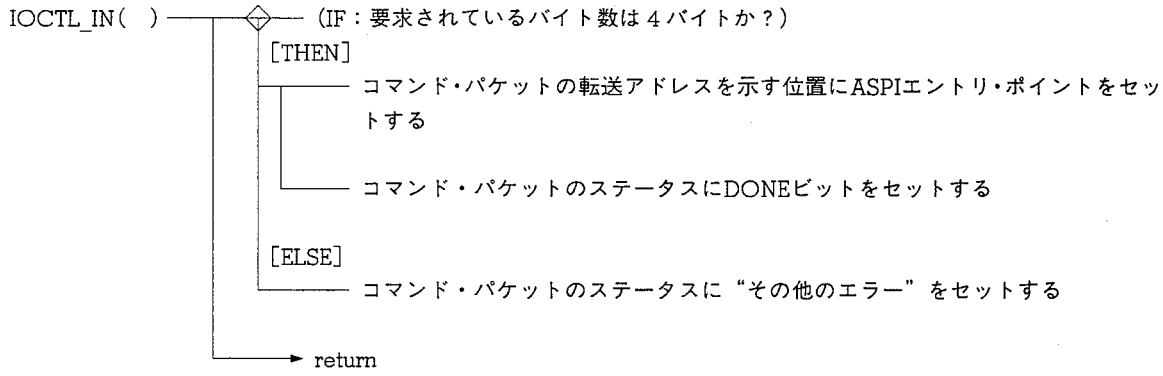


保守／廃止

(5) IOCTLインプット処理

モジュール名	IOCTL_IN
ファイル名	AS_REQ.CPP
言語	C言語
入力	packet
出力	packet
機能	MS-DOSにASPIエントリ・ポイントを返します。

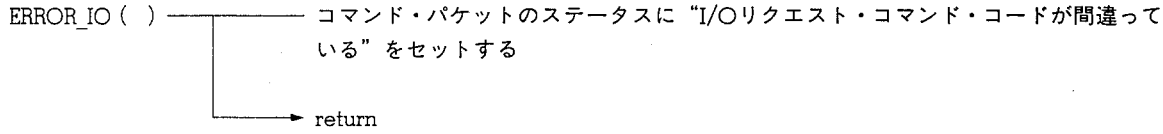
SPDチャート



(6) I/Oリクエスト・エラー処理

モジュール名	ERROR_IO
ファイル名	AS_REQ.CPP
言語	C言語
入力	なし
出力	packet
機能	packetに“I/Oリクエスト・コマンド・コードが間違っている”をセットします。

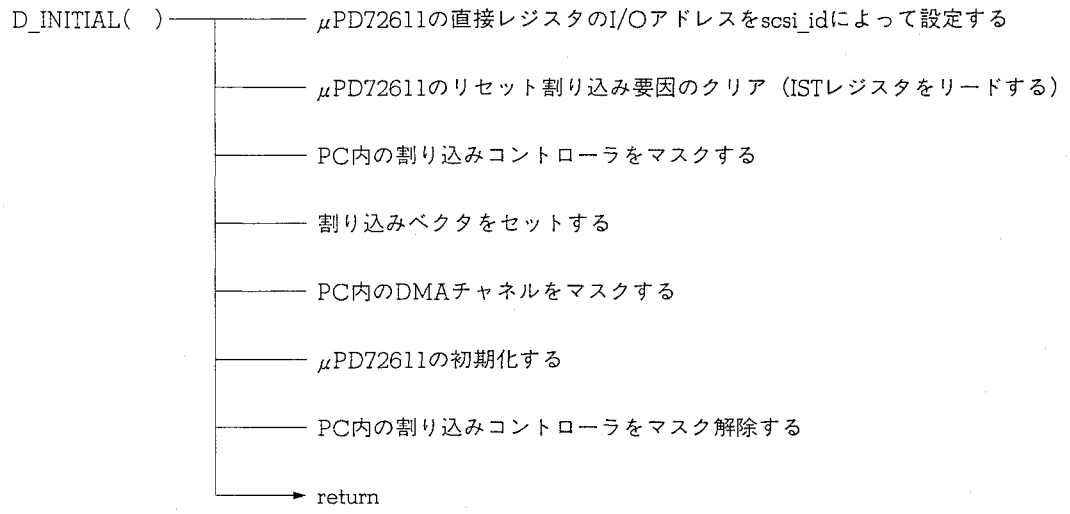
SPDチャート



(7) デバイスの初期化処理

モジュール名	D_INITIAL
ファイル名	INITIAL. CPP
言語	C言語
入力	i_id, scsi_add, intp, cha
出力	なし
機能	デバイスの初期化を実行します。

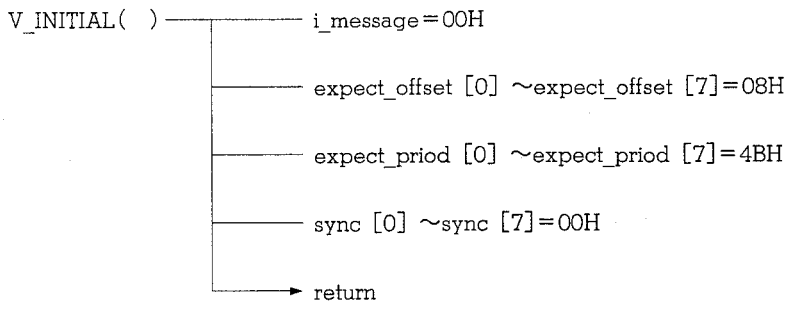
SPDチャート



(8) 変数の初期化処理

モジュール名	V_INITIAL
ファイル名	INITIAL. CPP
言語	C言語
入力	なし
出力	i_message, expect_offset [], expect_priod [], sync []
機能	SCSIフェーズ処理の変数を初期化します。

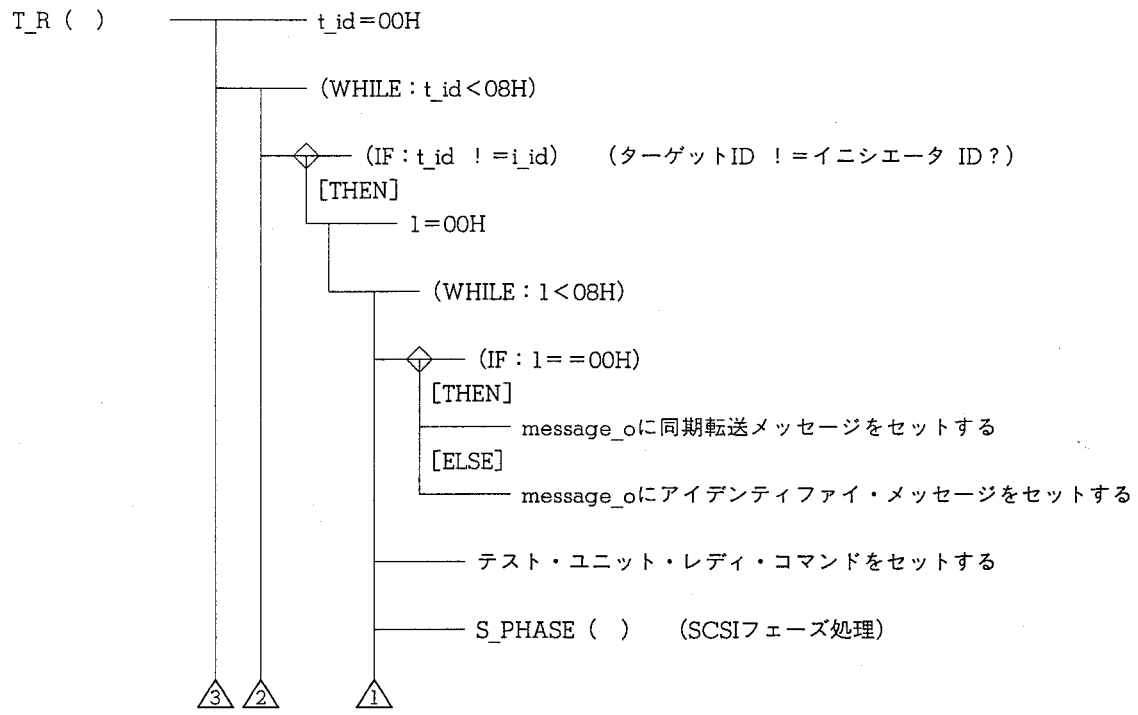
SPDチャート



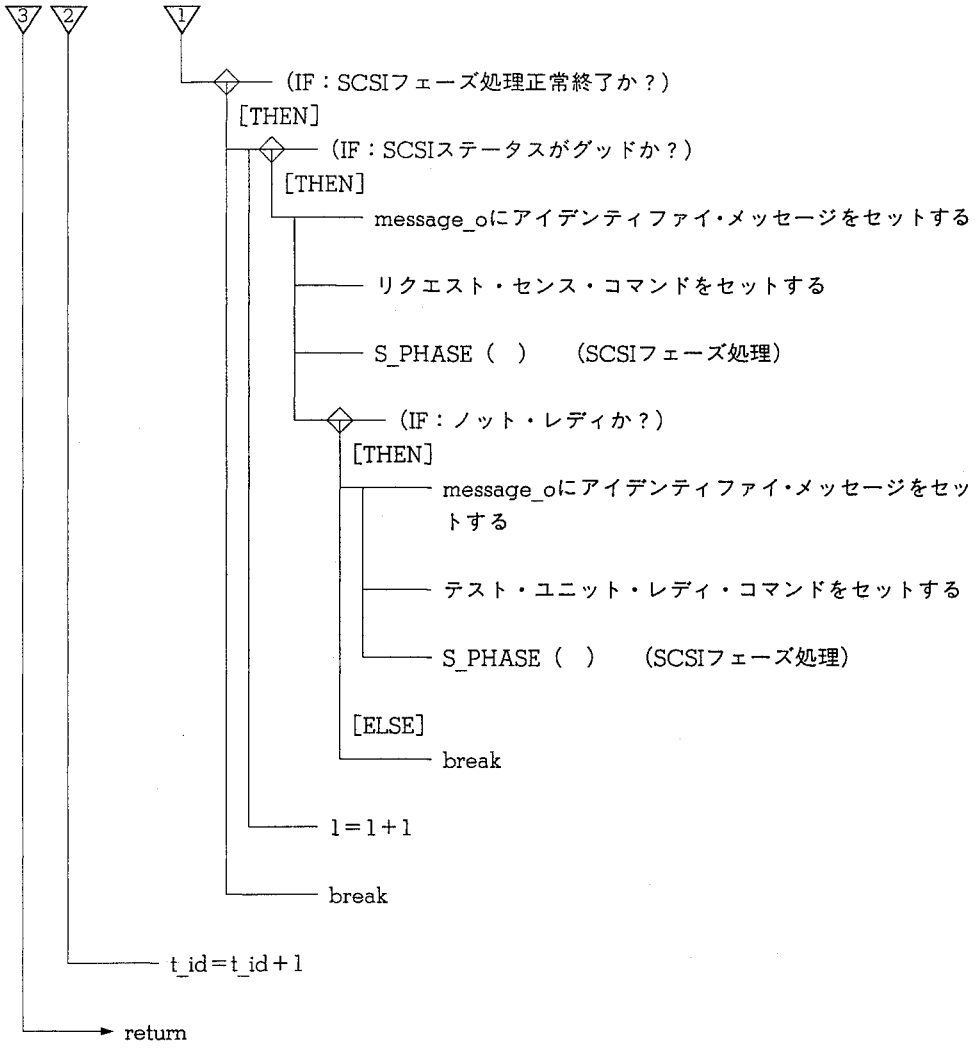
(9) ターゲット・レディ処理

モジュール名	T_R
ファイル名	INITIAL. CPP
言語	C言語
入力	なし
出力	なし
機能	接続されているターゲット（ロジカル・ユニットを含める）をレディ状態にします。

SPDチャート



保守／廃止


3
2
1
(IF：SCSIフェーズ処理正常終了か？)
[THEN]
(IF：SCSIステータスがグッドか？)
[THEN]
message_oにアイデンティファイ・メッセージをセットする
リクエスト・センス・コマンドをセットする
S_PHASE（ ）　(SCSIフェーズ処理)
(IF：ノット・レディか？)
[THEN]
message_oにアイデンティファイ・メッセージをセットする
テスト・ユニット・レディ・コマンドをセットする
S_PHASE（ ）　(SCSIフェーズ処理)
[ELSE]
break
l＝l＋1
break
t_id＝t_id＋1
return

保守／廃止

第９章　ASPI

この章ではASPIを説明します。

９.１　ASPIの仕様

このASPIがサポートしたASPIコマンドを表９－１に示します。なお，ASPI仕様の詳細は**付録C ASPI仕様**を参照してください。

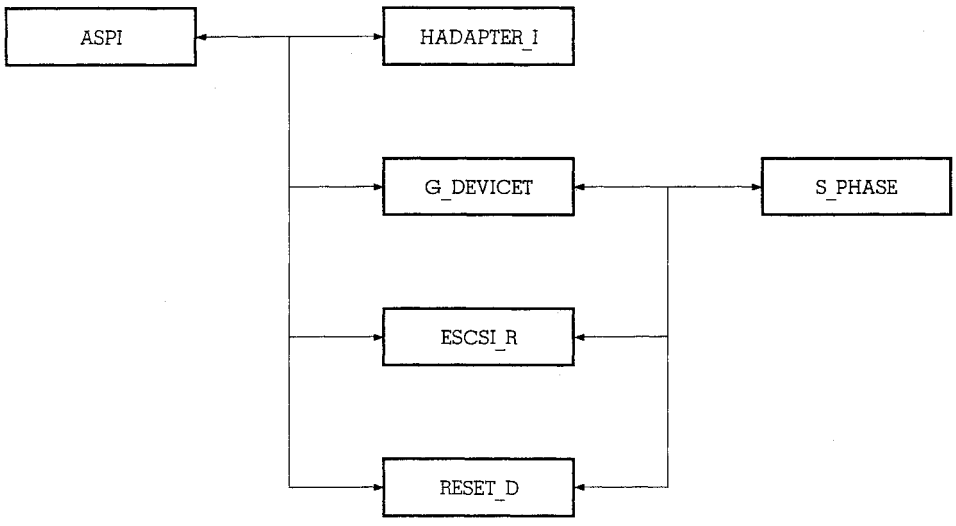
表９－１　ASPIサポート・コマンド

コマンド・コード	コマンド名
00H	Host Adapter Inquiry
01H	Get Device Type
02H	Execute SCSI I/O Request
04H	Reset SCSI Device

９.２　ASPIのプログラム構成

ASPIのプログラム構成を図９－１に示します。

図９－１　ASPIプログラム構成



保守／廃止

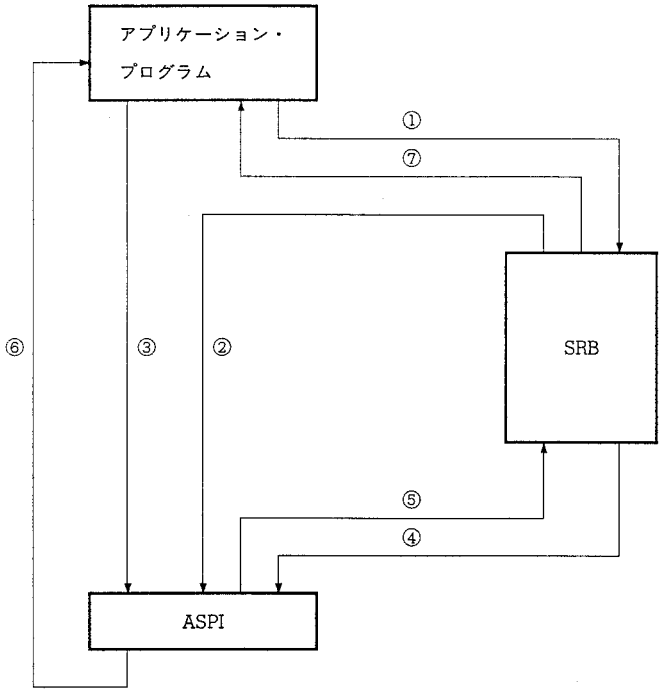
９.２.１　ASPIとアプリケーション・プログラムのインタフェース

アプリケーション・プログラムはMS-DOS上に常駐されているASPIをコールします。

インタフェースにはSCSI REQUEST BLOCK（SRB）と呼ばれるものを使用します。図９－２にASPIとアプリケーションのインタフェースを示します。ASPIを利用するアプリケーション・プログラムはMS-DOSを介してASPIエントリ・ポイントを受け取ります(**８.１　ASPIマネージャの仕様**を参照)。以下にアプリケーション・プログラムがASPIをアクセスする方法を示します。

① アプリケーション・プログラムはSRBをセットします。
② アプリケーション・プログラムはSRBのポインタをASPIに渡します。
③ アプリケーション・プログラムはASPIをコールします。
④ ASPIはSRBを参照しターゲットにアクセスします。
⑤ ASPIは実行結果をSRBにセットします。
⑥ ASPIはSCSIコマンド実行ブロック内のモジュールをリターンします。
⑦ アプリケーション・プログラムはSRBを参照しASPIの実行結果を調べます。

図９－２　アプリケーション／ASPIインタフェース



9.2.2 ASPIとSCSIフェーズ処理のインタフェース

ASPIとSCSIフェーズ処理のインタフェースの説明は，**10.2.3 SCSIフェーズ処理と上位モジュールのインタフェース**を参照してください。

9.3 ASPIの変数

ASPIの入出力変数にSCSI REQUEST BLOCK（SRB）を使用します。このプログラムにおけるSRBの構造体を図9-3に示します。SRBの内容は**付録C ASPI仕様**を参照してください。

図9-3 SRBの構造体

```
struct aspi_t                                   /*scsi request block(SRB)*/
    {
    unsigned char command_code;                 /*command code              (w)*/
    unsigned char status;                       /*status                    (r)*/
    unsigned char h_adapter_n;                  /*host adapter number       (w)*/
    unsigned char scsi_request_f;               /*scsi request flags        (w)*/
    unsigned long reserved;                     /*reserved                     */
    union {
        struct
            {
            unsigned char n_host_a;             /*number of host adapters   (r)*/
            unsigned char id_host_a;            /*id of host adapter        (r)*/
            unsigned char scsi_m_id[16];        /*scsi manager id           (r)*/
            unsigned char h_adapter_id[16];     /*host adapter id           (r)*/
            unsigned char h_adapterp[16];       /*host adapter parameters   (r)*/
            }   hadap;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned char p_device_t;           /*peripheral device type    (r)*/
            }   gdevice;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned long data_a_l;             /*data length               (w)*/
            unsigned char sense_a_l;            /*sense length              (w)*/
            unsigned char far *data_bp;         /*data buff pointer         (w)*/
            unsigned char far *srb_lp;          /*srb link pointer          (w)*/
            unsigned char scsicdb_l;            /*scsi cdb length           (w)*/
            unsigned char h_adapter_s;          /*host adapter status       (r)*/
            unsigned char target_status;        /*target status             (r)*/
            unsigned char far *post_ra;         /*post routine address      (w)*/
            unsigned char warkspace[34];        /*aspi warkspace               */
            unsigned char scsics[256];          /*scsi cdb and sense data (w/r)*/
            }   escsirq;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned char reserved[14];         /*reserved                     */
            unsigned char h_adapter_s;          /*host adapter status       (r)*/
            unsigned char target_status;        /*target status             (r)*/
            unsigned char far *post_ra;         /*post routine address      (w)*/
            unsigned char warkspace[34];        /*aspi warkspace               */
            }   resetsd;
        }   u_srb;
    };
```

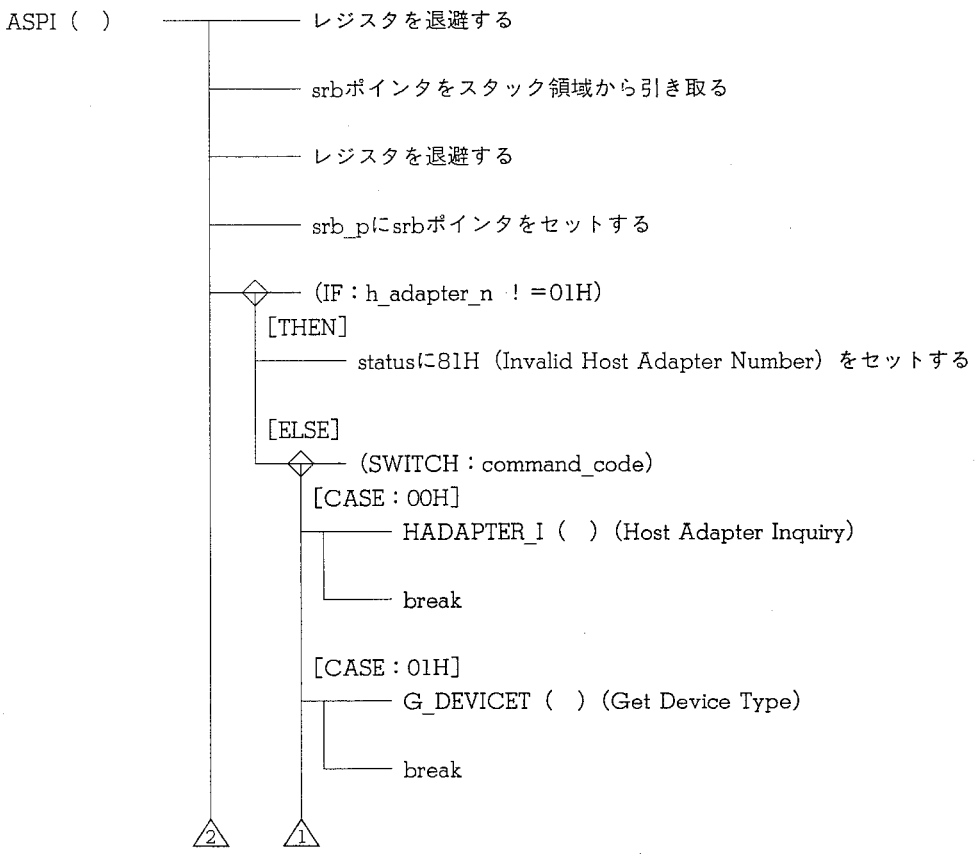
９.４　ASPIモジュール説明

ASPIの各モジュールの説明をします。**第５章　フォーマット・コマンド**の表５－４に示す内容に沿って説明し，SPDチャートを示します。

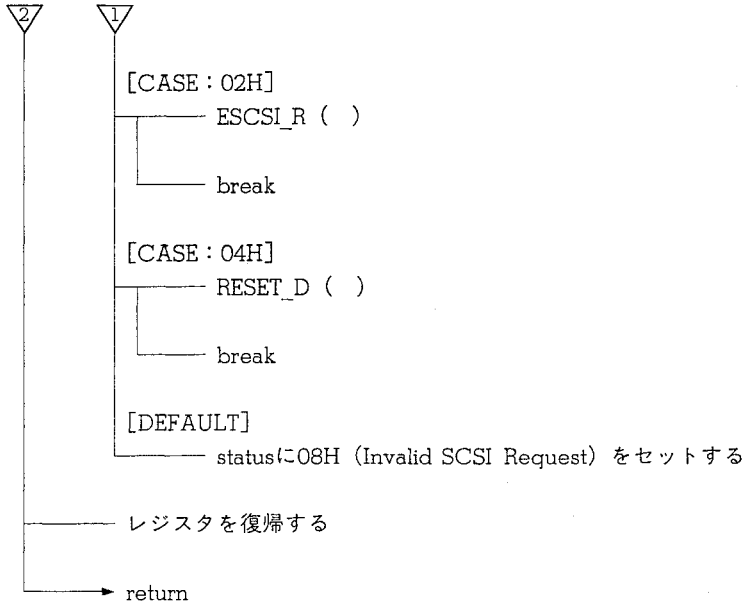
(1) ASPI処理

モジュール名	ASPI
ファイル名	ASPI. CPP
言語	C言語
入力	srb
出力	srb
機能	srbのcommand codeを参照し，各モジュールをコールします。

SPDチャート



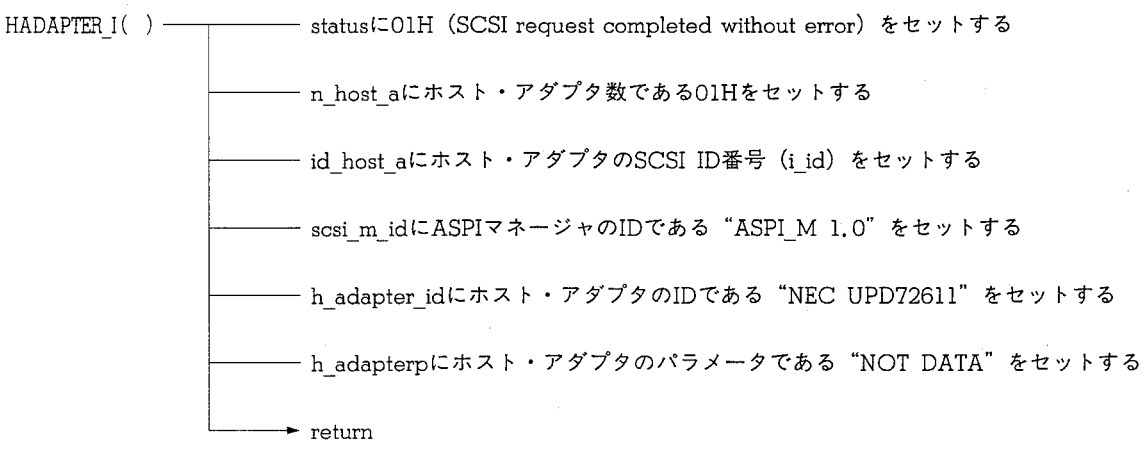
保守／廃止


2
1
[CASE : 02H]
ESCSI_R ( )
break
[CASE : 04H]
RESET_D ( )
break
[DEFAULT]
statusに08H (Invalid SCSI Request) をセットする
レジスタを復帰する
return

(2) HOST ADAPTER INQUIRY処理

モジュール名	HADAPTER_I
ファイル名	ASPI.CPP
言語	C言語
入力	srb
出力	srb
機能	ホスト・アダプタの情報をアプリケーション・プログラムに渡します。

SPDチャート

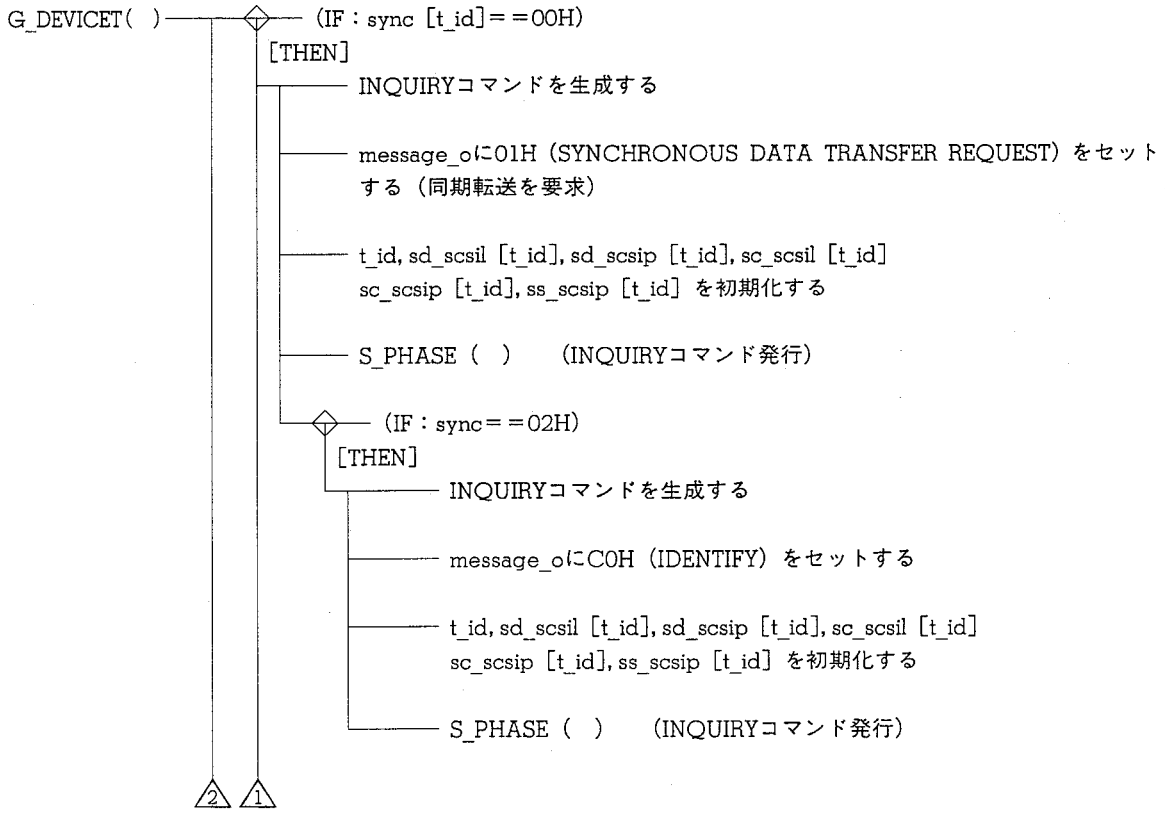


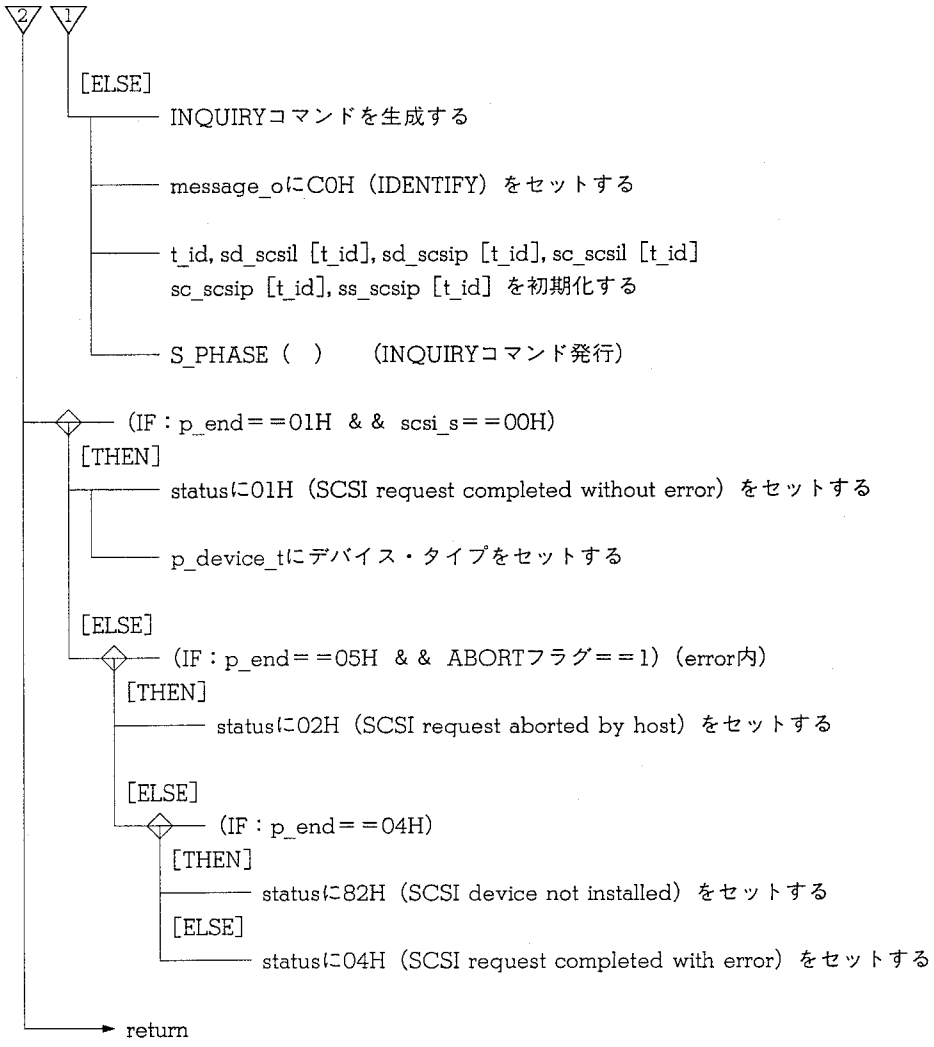
保守／廃止

(3) GET DEVICE TYPE処理

モジュール名	G_DEVICET
ファイル名	ASPI. CPP
言語	C言語
入力	srb
出力	srb
機能	アプリケーション・プログラムがターゲットの種類を確認するためINQUIRYデータをリードし，アプリケーション・プログラムに渡します。

SPDチャート

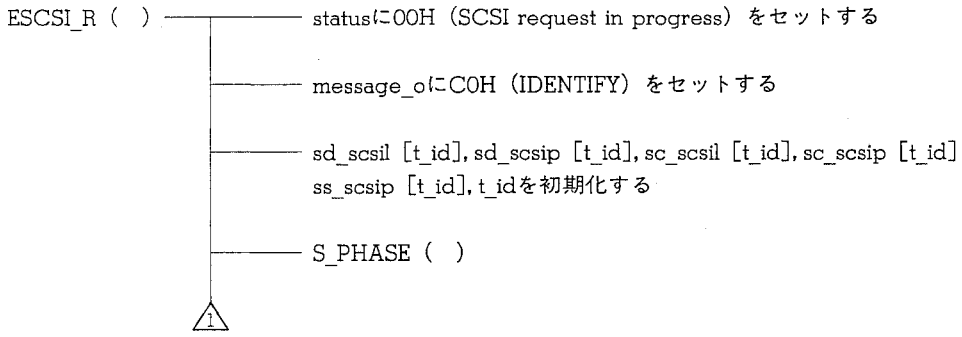



2
1
[ELSE]
INQUIRYコマンドを生成する
message_oにC0H（IDENTIFY）をセットする
t_id, sd_scsil [t_id], sd_scsip [t_id], sc_scsil [t_id]
sc_scsip [t_id], ss_scsip [t_id] を初期化する
S_PHASE（ ）　（INQUIRYコマンド発行）
（IF：p_end＝＝01H ＆＆ scsi_s＝＝00H）
[THEN]
statusに01H（SCSI request completed without error）をセットする
p_device_tにデバイス・タイプをセットする
[ELSE]
（IF：p_end＝＝05H ＆＆ ABORTフラグ＝＝1）（error内）
[THEN]
statusに02H（SCSI request aborted by host）をセットする
[ELSE]
（IF：p_end＝＝04H）
[THEN]
statusに82H（SCSI device not installed）をセットする
[ELSE]
statusに04H（SCSI request completed with error）をセットする
return

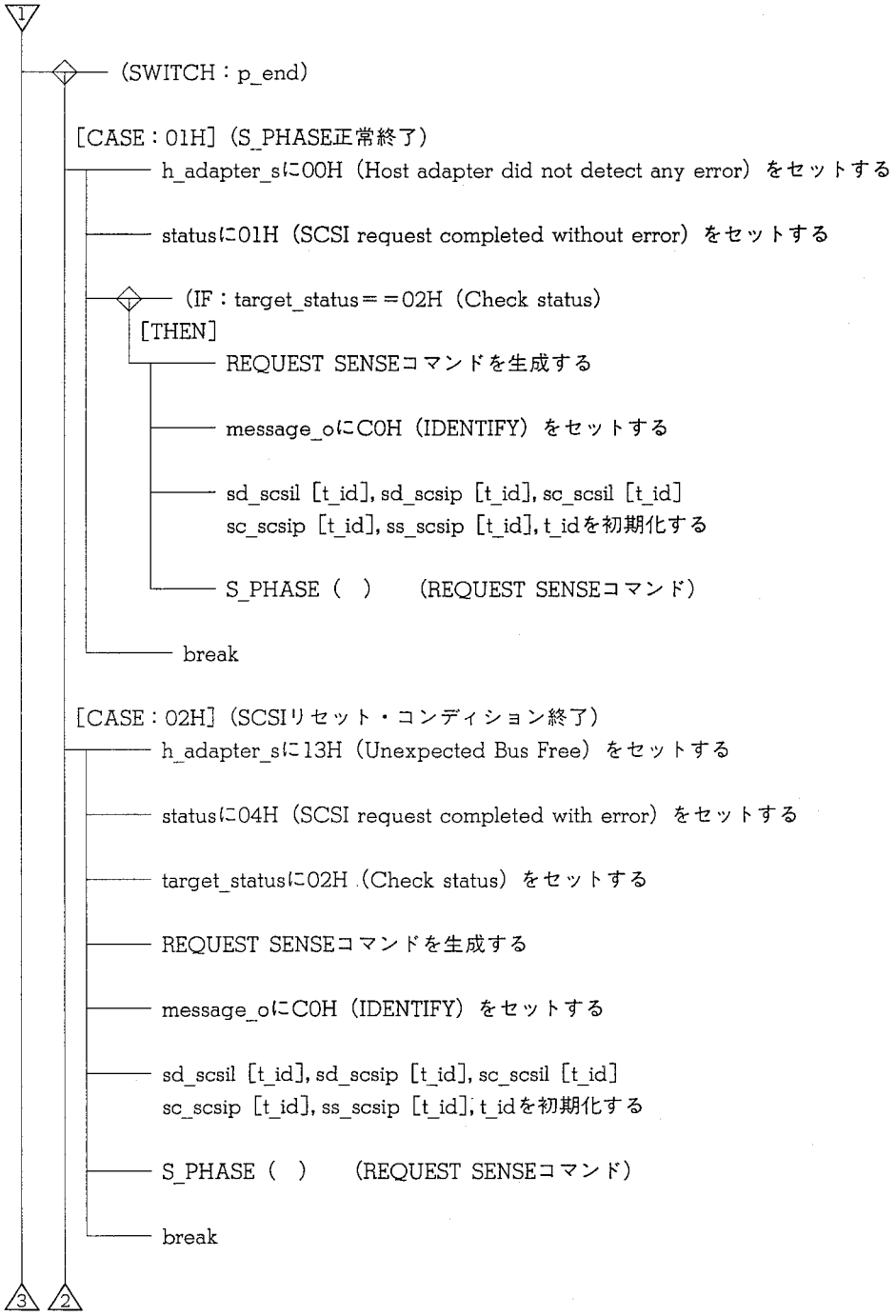
(4) EXECUTE SCSI I/O REQUEST処理

モジュール名	ESCSI_R
ファイル名	ASPI. CPP
言語	C言語
入力	srb
出力	srb
機能	アプリケーション・プログラムの要求によりSCSIコマンドを実行し，実行結果をアプリケーション・プログラムに渡します。

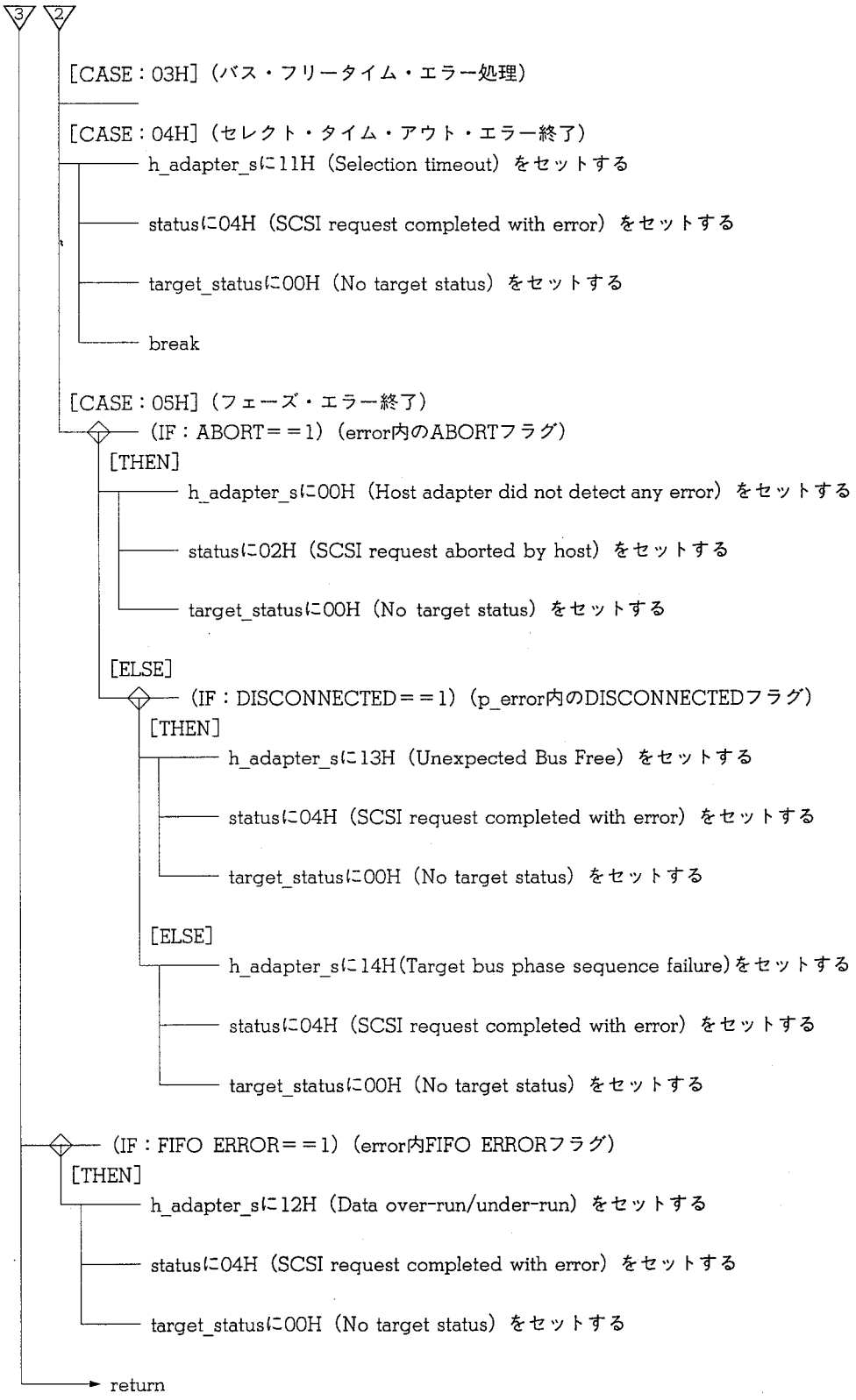
SPDチャート



保守／廃止


1
(SWITCH：p_end)
[CASE：01H]（S_PHASE正常終了）
h_adapter_sに00H（Host adapter did not detect any error）をセットする
statusに01H（SCSI request completed without error）をセットする
(IF：target_status＝＝02H（Check status)
[THEN]
REQUEST SENSEコマンドを生成する
message_oにC0H（IDENTIFY）をセットする
sd_scsil [t_id], sd_scsip [t_id], sc_scsil [t_id]
sc_scsip [t_id], ss_scsip [t_id], t_idを初期化する
S_PHASE（ ）　（REQUEST SENSEコマンド）
break
[CASE：02H]（SCSIリセット・コンディション終了）
h_adapter_sに13H（Unexpected Bus Free）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに02H（Check status）をセットする
REQUEST SENSEコマンドを生成する
message_oにC0H（IDENTIFY）をセットする
sd_scsil [t_id], sd_scsip [t_id], sc_scsil [t_id]
sc_scsip [t_id], ss_scsip [t_id], t_idを初期化する
S_PHASE（ ）　（REQUEST SENSEコマンド）
break
3
2

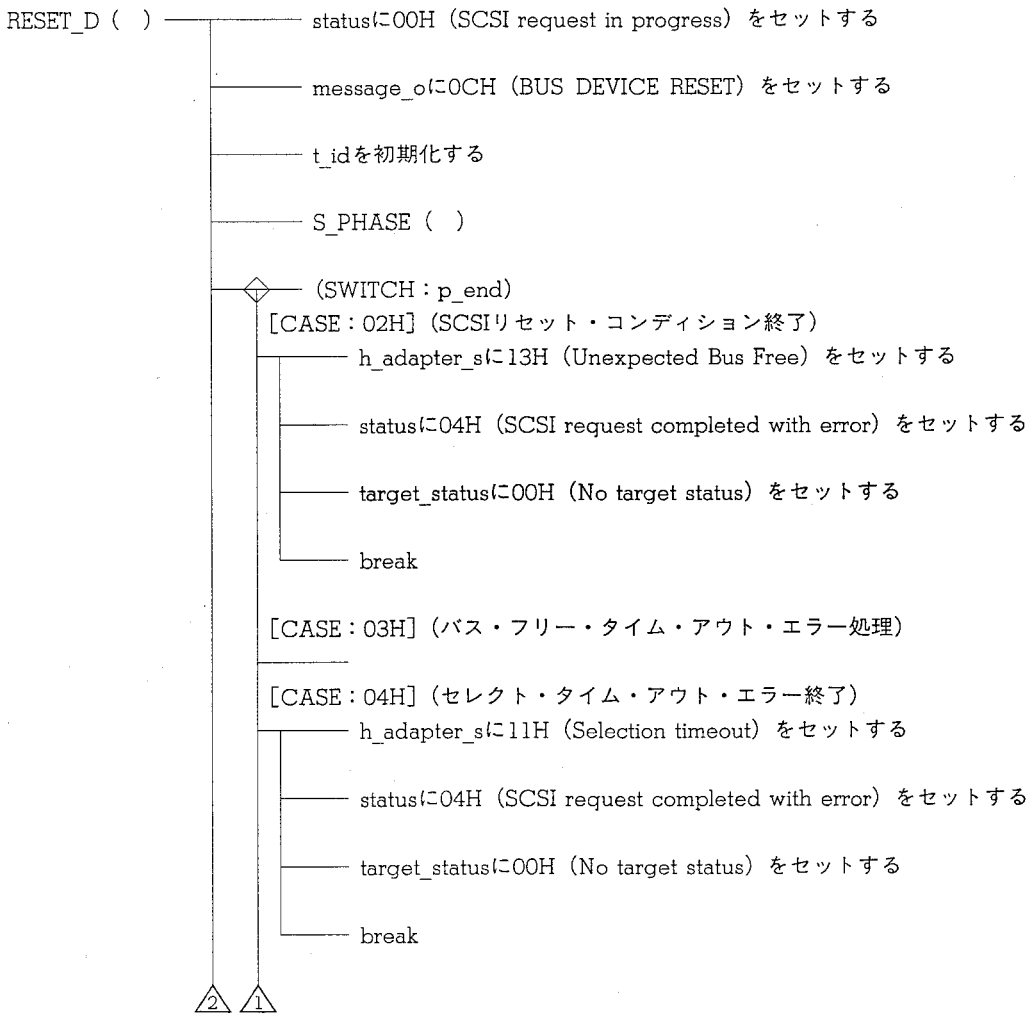
保守／廃止

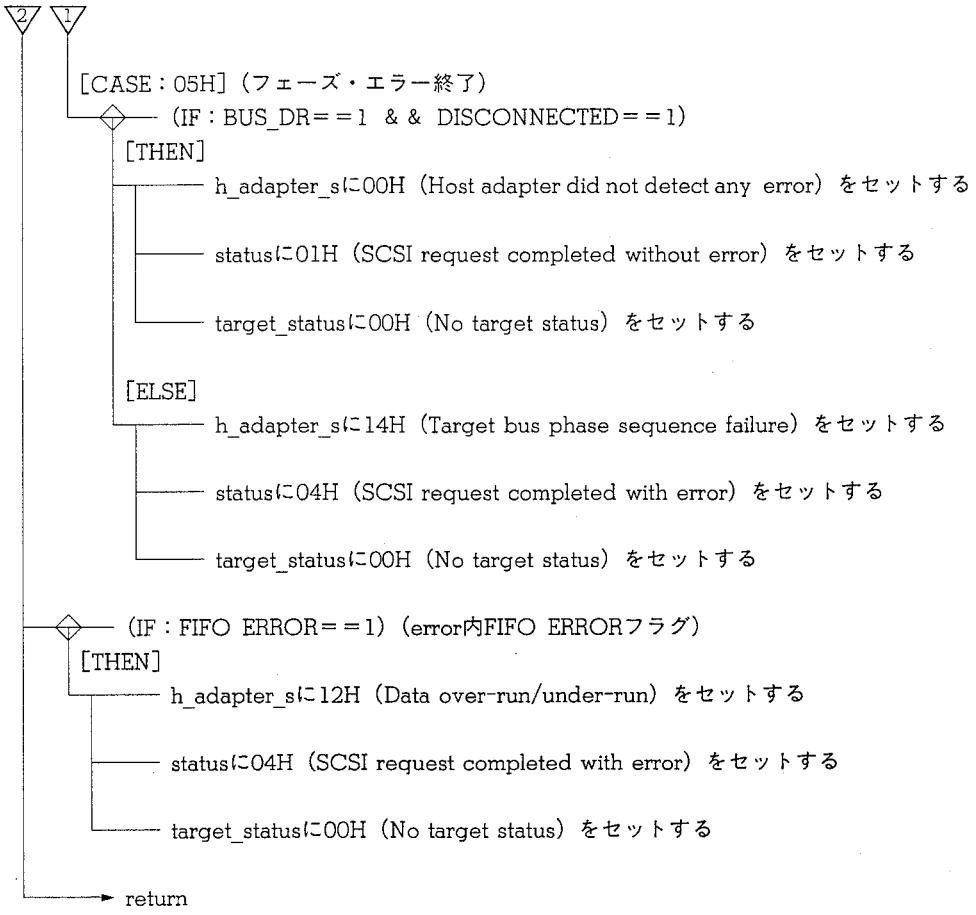

3
2
[CASE：03H]（バス・フリータイム・エラー処理）
[CASE：04H]（セレクト・タイム・アウト・エラー終了）
h_adapter_sに11H（Selection timeout）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
break
[CASE：05H]（フェーズ・エラー終了）
（IF：ABORT＝＝1）（error内のABORTフラグ）
[THEN]
h_adapter_sに00H（Host adapter did not detect any error）をセットする
statusに02H（SCSI request aborted by host）をセットする
target_statusに00H（No target status）をセットする
[ELSE]
（IF：DISCONNECTED＝＝1）（p_error内のDISCONNECTEDフラグ）
[THEN]
h_adapter_sに13H（Unexpected Bus Free）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
[ELSE]
h_adapter_sに14H(Target bus phase sequence failure)をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
（IF：FIFO ERROR＝＝1）（error内FIFO ERRORフラグ）
[THEN]
h_adapter_sに12H（Data over-run/under-run）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
return

(5) RESET SCSI DEVICE処理

モジュール名	RESET_D
ファイル名	ASPI. CPP
言語	C言語
入力	srb
出力	srb
機能	指定されたターゲットをリセットし，実行結果をアプリケーション・プログラムに渡します。

SPDチャート




2
1
[CASE：05H]（フェーズ・エラー終了）
(IF：BUS_DR＝＝1 ＆＆ DISCONNECTED＝＝1)
[THEN]
h_adapter_sに00H（Host adapter did not detect any error）をセットする
statusに01H（SCSI request completed without error）をセットする
target_statusに00H（No target status）をセットする
[ELSE]
h_adapter_sに14H（Target bus phase sequence failure）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
(IF：FIFO ERROR＝＝1)（error内FIFO ERRORフラグ）
[THEN]
h_adapter_sに12H（Data over-run/under-run）をセットする
statusに04H（SCSI request completed with error）をセットする
target_statusに00H（No target status）をセットする
return

保守／廃止

第10章　SCSIフェーズ処理

この章ではSCSIフェーズ処理を説明します。SCSIフェーズ処理はμPD72611ホスト・アダプタ・ボードとSCSIフェーズの制御を行います。

10.1　SCSIフェーズ処理の仕様

SCSIフェーズ処理の仕様を以下に示します。表10－1にμPD72611のイニシャライズ内容を示します。μPD72611のイニシャライズはASPIマネージャのイニシャライズで実行されます。表10－2にDMAのイニシャライズ内容を示します。DMAのイニシャライズはデータ・アウト・フェーズ・スタート処理，データ・イン・フェーズ・スタート処理で実行されます。割り込みコントローラはASPIマネージャのイニシャライズでエッジ／レベル・コントロール・レジスタをエッジにセットします。

① ディスコネクト／リコネクト機能サポート
② μPD72611のハードウエア割り込みをサポート
③ データ・フェーズのDMA転送サポート
④ 同期転送（10 Mbyte/s）サポート

表10－1　μPD72611のイニシャライズ

レジスタ名	値	内　　容
TMOD	—	初期値は非同期転送モードで設定し，SYNCHRONOUS DATA TRANSFER REQUESTメッセージを転送後同期転送モードに設定
EMOD	02H	“キュータグ・メッセージをサポートしない”，“パリティ・スルー・モードを指定しない”，“パリティ・エラーを検出しても転送を続行する”，“ディマンド転送モード”
BFTOUT	FFH	1,671.168 ms
SRTOUT	FFH	1,671.168 ms
RATOUT	00H	検出しない
MOD	E2H	“DMAモード”，“SCSIバス・パリティ・イネーブル”，“CPUバス・パリティ・ディスエーブル”，“アービトレーション・モード”，“イニシエータとしてリセレクトされる”，“ターゲットとしてセレクトされる”

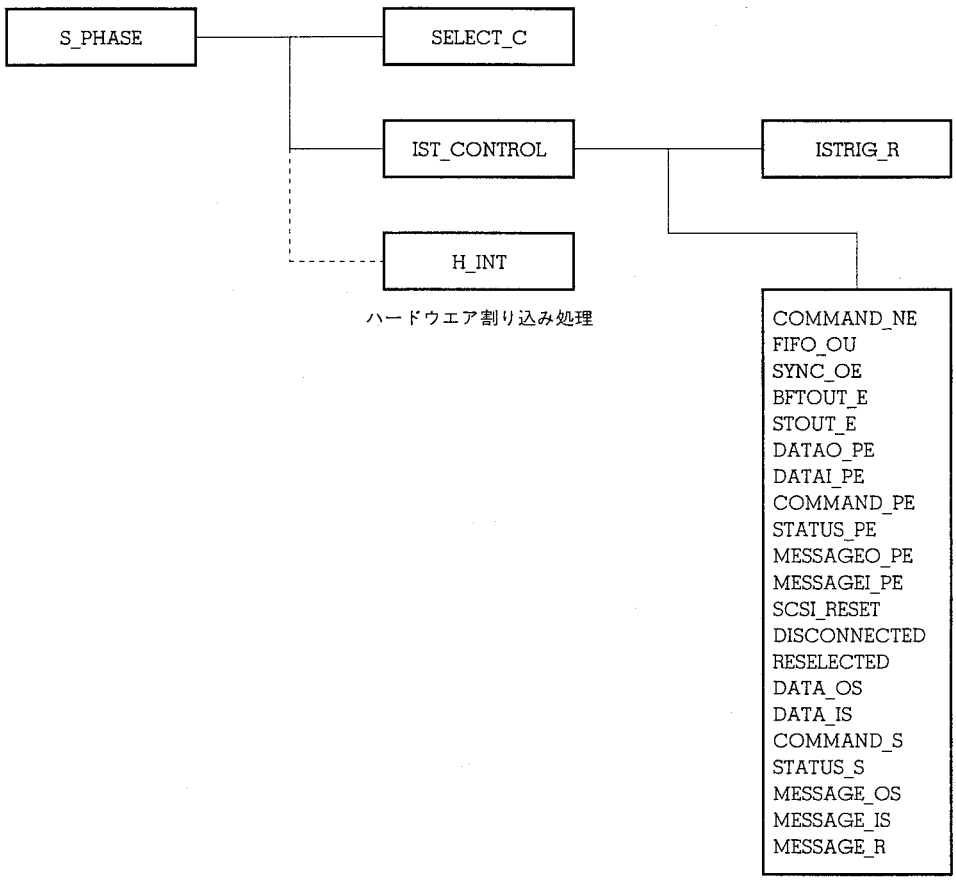
表10－2　DMAコントローラのイニシャライズ

レジスタ名	内　　容
MODE	ディマンド・モード，アドレス・インクリメント，オートイニシャライズ機能ディスエーブル
EXTENDED MODE	32 bit I/O count by byte, TYPE B timing，EOP出力，STOPレジスタ・ディスエーブル

10.2　SCSIフェーズ処理のプログラム構成

図10－1にSCSIフェーズ処理のプログラム構成を示します。

図10－1　SCSIフェーズ処理プログラム構成

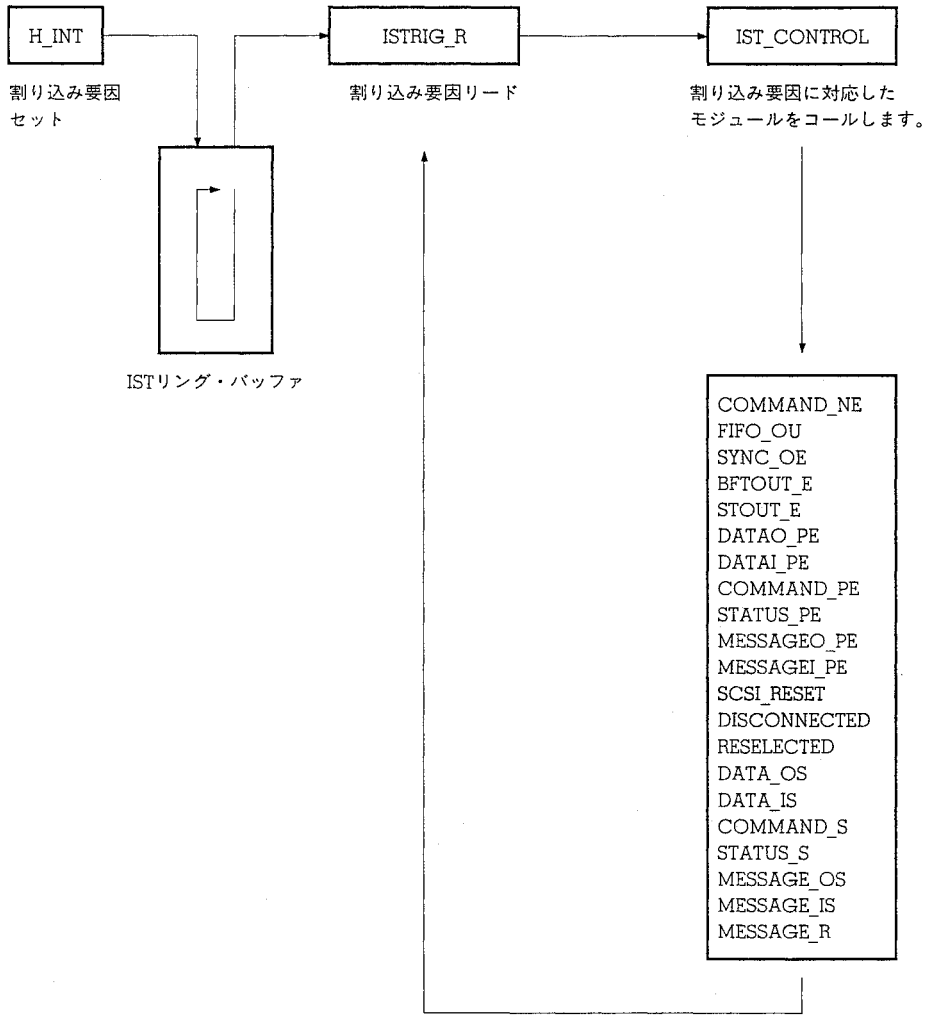


保守／廃止

10.2.1　ハードウエア割り込み処理

μPD72611の割り込み処理を図10－2に示します。ハードウエア割り込み処理はμPD72611のISTレジスタから割り込み要因をリードし，ISTリング・バッファにセットします。ISTリング・バッファから割り込み要因をリードし，割り込み要因に対応したモジュールをコールします。

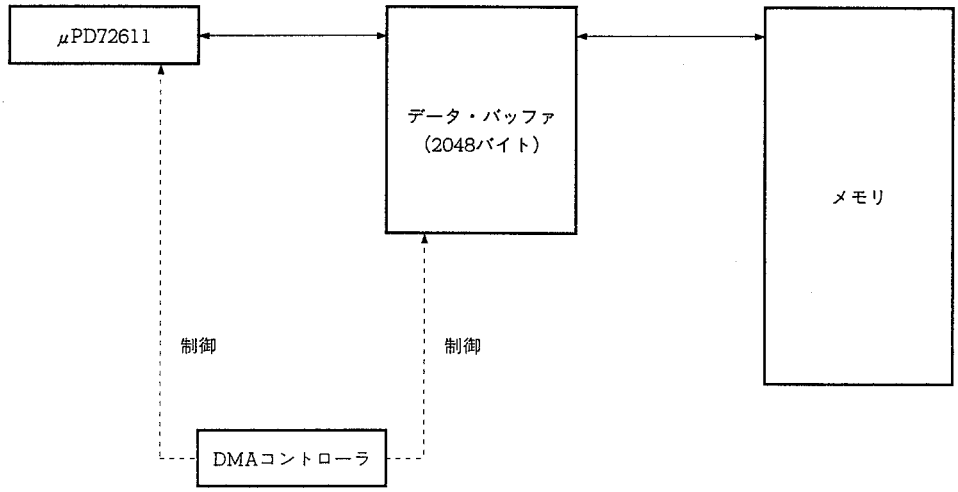
図10－2　ハードウエア割り込み処理



10.2.2　DMAコントローラ処理

図10－3にDMAコントローラ処理を示し，動作を説明します。DMA転送は32ビット転送であるためデータ・バッファはダブルワード（32ビット）境界でアライメントします。C言語ではダブルワードのアライメントができないので，アセンブラ（ファイル名：IO_PORT.ASM）の中でデータ・バッファを確保します。

図10－3　DMAコントローラ処理



μPD72611ライト：

① 目的のアドレスからデータ・バッファへプログラムにより2048バイト転送します（2048バイト以下の転送要求であれば転送要求バイト数で終了します）。

② DMAコントローラはデータ・バッファからμPD72611へ転送します。

③ 次に続く転送データが存在すれば①に戻り，存在しなければ終了します。

μPD72611リード：

① DMAコントローラはμPD72611からデータ・バッファへ2048バイト転送します（2048バイト以下の転送要求であれば転送要求バイト数で終了します）。

② データ・バッファから目的のアドレスへプログラムにより転送します。

③ 次に続く転送データが存在すれば①に戻り，存在しなければ終了します。

保守／廃止

10.2.3　SCSIフェーズ処理と上位モジュールのインタフェース

図10－4に上位モジュールとのインタフェースを示します。以下に動作を示します。

① 上位モジュールはmessage_oにセレクション後のメッセージ・アウト・フェーズにおいて転送するメッセージ，t_idにターゲットのID番号，sd_scsip［t_id］にセーブ・データSCSIポインタ，sd_scsil［t_id］にSCSIデータ長，sc_scsip［t_id］にセーブ・コマンドSCSIポインタ，sc_scsil［t_id］にSCSIコマンド長，ss_scsip［t_id］にセーブ・ステータスSCSIポインタをセットします。

② 上位モジュールはSCSIフェーズ処理をコールする。

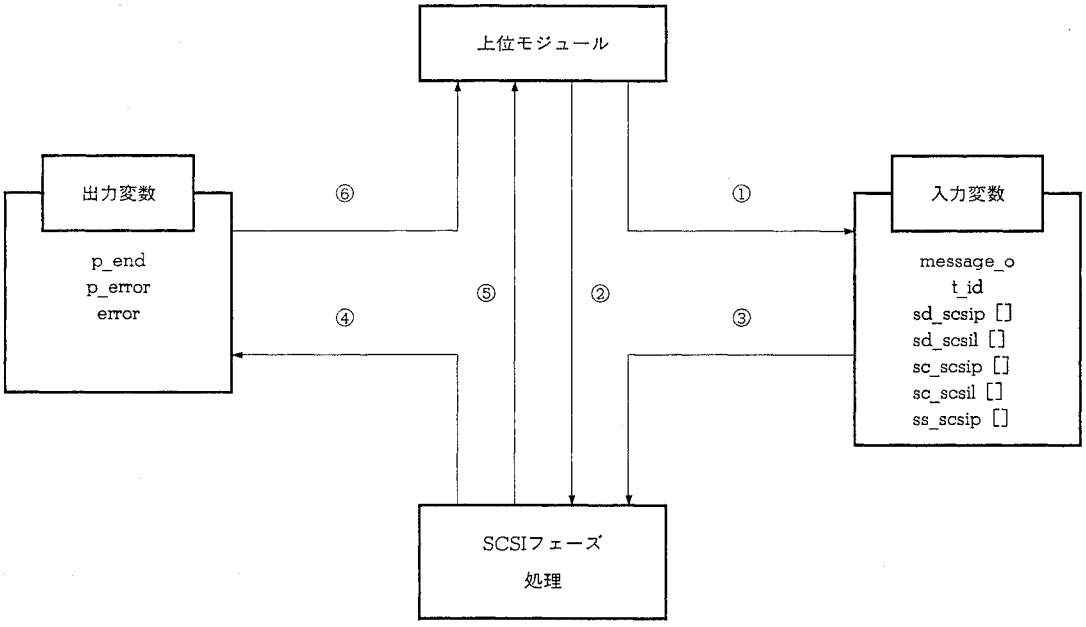
③ SCSIフェーズ処理は入力変数を参照して実行します。

④ SCSIフェーズ処理は実行結果を出力変数にセットします。

⑤ SCSIフェーズ処理は上位モジュールにリターンします。

⑥ 上位モジュールは出力変数の内容を参照し，SCSIフェーズ処理の実行結果を確認します。

図10－4　上位モジュールとのインタフェース



10.3　SCSIフェーズ処理の変数

SCSIフェーズ処理の変数を表10－3に示す内容で説明します。

表10－3　変数説明

変 数 名	変数名を示します。
ビット長	変数のビットの長さを示します。
機　　能	変数の機能を示します。

変 数 名	message_o
ビット長	32ビット
機　　能	

メッセージ・アウト・フェーズ・スタート処理でターゲットへ転送するメッセージをセットします。

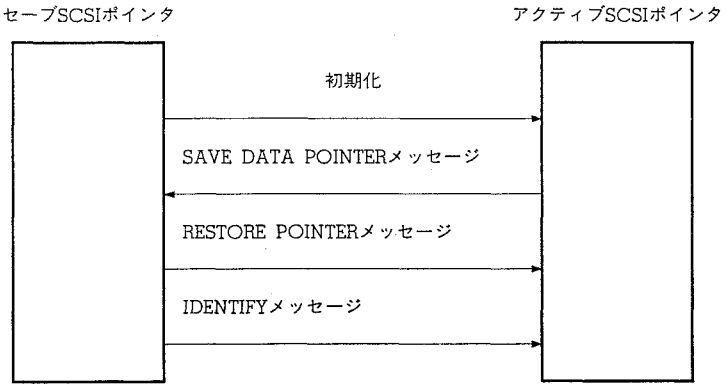
00H ：COMMAND COMPLETE
01H ：SYNCHRONOUS DATA TRANSFER REQUEST
05H ：INITIATOR DETECTED ERROR
06H ：ABORT
07H ：MESSAGE REJECT
08H ：NO OPERATION
09H ：MESSAGE PARITY ERROR
0CH：BUS DEVICE RESET
C0H：IDENTIFY

保守／廃止

変数名	sd_scsil []	sd_scsip []
ビット長	32ビット	8ビット
機　能		

sd_scsil []　：データ数をセーブする領域です。t_idに対応した0〜7の配列です。

sd_scsip []　：セーブ・データSCSIポインタ（データ領域のアドレス）です。t_idに対応した0〜7の配列です。

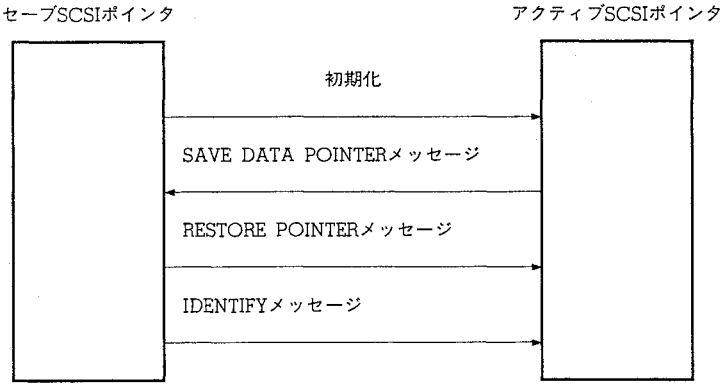


変数名	sc_scsil []	sc_scsip []	ss_scsip []
ビット長	8ビット	8ビット	8ビット
機　能			

sc_scsil []　：コマンド数をセーブする領域であるt_idに対応した0〜7の配列です。

sc_scsip []　：セーブ・コマンドSCSIポインタ（コマンド領域のアドレス）です。t_idに対応した0〜7の配列です。

ss_scsip []　：セーブ・ステータスSCSIポインタである（ステータス領域のアドレスを示します）t_idに対応した0〜7の配列です。

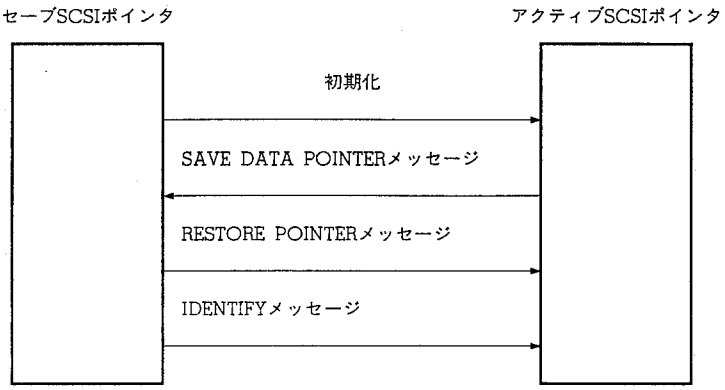


10

変数名	ad_scsil	ad_scsip
ビット長	32ビット	8ビット
機　能		

ad_scsil ：ターゲットへ転送するデータ数を示します。

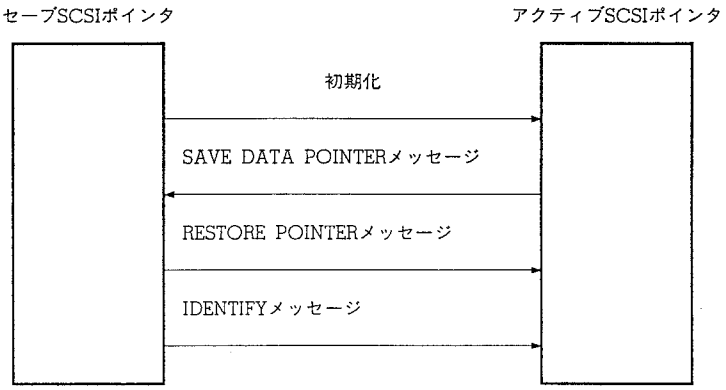
ad_scsip：アクティブ・データSCSIポインタです（ターゲット間で転送するデータ領域のアドレスを示します）。



変数名	ac_scscil	ac_scsip
ビット長	8ビット	8ビット
機　能		

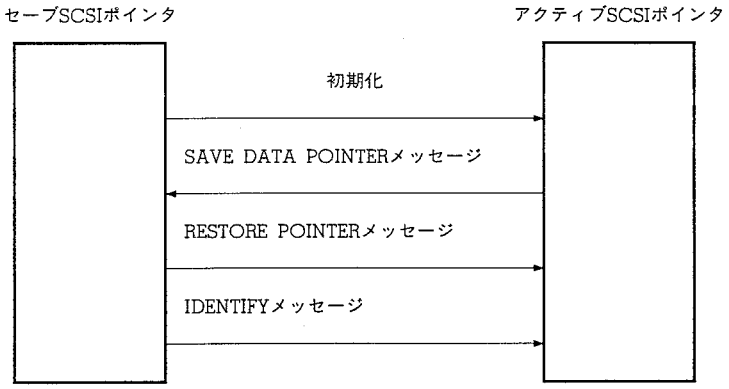
ac_scsil ：ターゲットへ転送するコマンド数を示します。

ac_scsip：アクティブ・コマンドSCSIポインタです（ターゲットへ転送するコマンド領域のアドレスを示します）。



変数名	as_scsip
ビット長	8ビット
機　能	

as_scsip：アクティブ・ステータスSCSIポインタです（ターゲットから転送されるステータス領域のアドレスを示します）。



変数名	t_id
ビット長	8ビット
機　能	

ターゲットのID番号を示します。ターゲットをセレクトする場合およびセーブSCSIポインタの管理に使用します。

変数名	message_i
ビット長	8ビット
機　能	

ターゲットから受信したメッセージを保持する変数です。

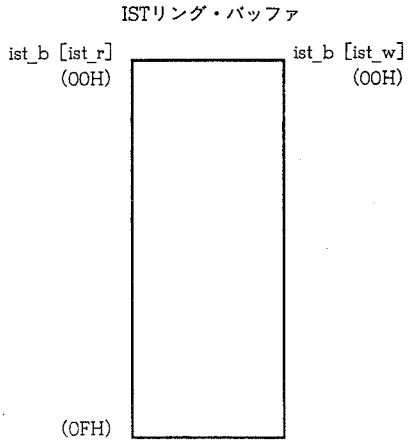
初期値はFFHです。

変数名	ist
ビット長	8ビット
機　能	

μPD72611のISTレジスタの内容を示します。

変数名	ist_b []
ビット長	８ビット
機　　能	

ISTリング・バッファです。



変数名	ist_w
ビット長	８ビット
機　　能	

ISTリング・バッファへ割り込み要因をセットする位置を示します。

00Hから始まり0FHまでインクリメントします。0FHに割り込み要因が書き込まれると00Hにリセットします。

初期値は00Hです。

変数名	ist_r
ビット長	８ビット
機　　能	

ISTリング・バッファから割り込み要因を読み出す位置を示します。

int_r＝int_wになった場合すべての割り込み要因を読み出したことになります。

00Hから始まり0FHまでインクリメントします。0FHから割り込み要因を読み出すと00Hにリセットします。

初期値は00Hです。

保守／廃止

変数名	p_end
ビット長	8ビット
機　能	

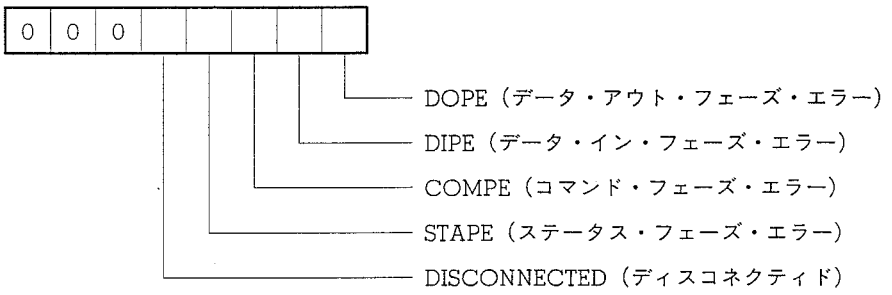
SCSIフェーズ処理の実行結果を示します。

00H：S_PHASE実行中
01H：S_PHASE正常終了
02H：SCSIリセット・コンディション終了
03H：バス・フリー・タイム・アウト・エラー終了
04H：セレクト・タイム・アウト・エラー終了
05H：フェーズ・エラー終了

初期値は00Hです。

変数名	p_error
ビット長	8ビット
機　能	

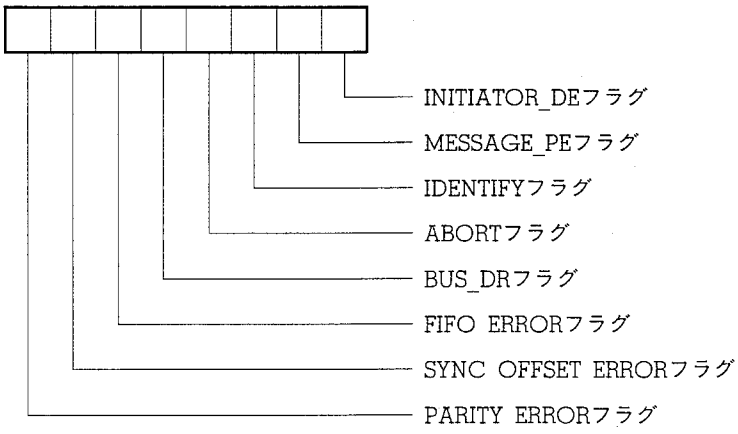
フェーズ・エラー（ターゲットが異常なフェーズ・シーケンスを行った）したときの状態を示します。



初期値は00Hです。

変数名	error
ビット長	8ビット
機　能	

SCSIフェーズ処理中に発生したエラーの状態を示します。初期値は00Hです。



INITIATOR_DEフラグ：INITIATOR DETECTED ERRORメッセージをターゲットがサポートしていないため，このメッセージをイニシエータが転送できなかった。

MESSAGE_PEフラグ：MESSAGE PARITY ERRORメッセージをターゲットがサポートしていないためこのメッセージをイニシエータが転送できなかった。

IDENTIFYフラグ：IDENTIFYメッセージをターゲットがサポートしていないため，このメッセージをイニシエータが転送できなかった。

ABORTフラグ：イニシエータがABORTメッセージをターゲットに転送した。

BUS_DRフラグ：イニシエータがBUS DEVICE RESETメッセージをターゲットに転送した。

FIFO ERRORフラグ：イニシエータにおいてFIFOオーバラン／アンダーラン・エラーが発生した。

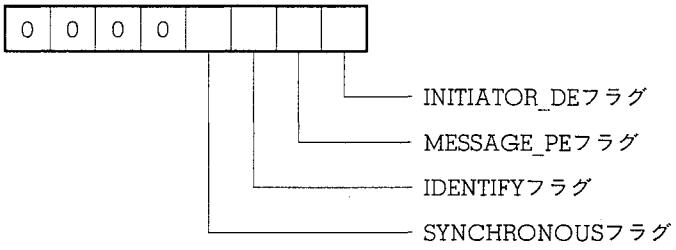
SYNC OFFSET ERRORフラグ：イニシエータにおいて同期転送オフセット・エラーが発生した。

PARITY ERRORフラグ：イニシエータにおいてパリティ・エラーが発生した。

保守／廃止

変数名	i_message
ビット長	8ビット
機　能	

ターゲットがサポートしていないメッセージを示します。



INITIATOR_DEフラグ：ターゲットがINITIATOR DETECTED ERRORメッセージをサポートしていない。
MESSAGE_PEフラグ：ターゲットがMESSAGE PARITY ERRORメッセージをサポートしていない。
IDENTIFYフラグ：ターゲットがIDENTIFYメッセージをサポートしていない。
SYNCHRONOUSフラグ：ターゲットがSYNCHRONOUS DATA TRANSFER REQUESTメッセージをサポートしていない。

変数名	expect_offset []	expect_priod []
ビット長	8ビット	8ビット
機　能		

同期転送をするときのイニシエータの期待値を示します。

expect_offset []　：オフセット値（08H）
expect_priod []　：転送時間　10 Mbyte/s（19H）

変数名	sync []
ビット長	8ビット
機　能	

転送モードの状態を示します。

00H：未定義
01H：同期転送
02H：非同期転送

初期値は00Hです。

変数名	messagepe_c
ビット長	８ビット
機　能	
MESSAGE PARITY ERRORをターゲットに転送した回数をカウントします。 初期値は00Hです。	

変数名	pmessage_c
ビット長	８ビット
機　能	
パリティ・エラーの発生したメッセージのバイト数をカウントします。 初期値は00Hです。	

変数名	emessage_c
ビット長	８ビット
機　能	
拡張メッセージのバイト数をカウントします。 初期値は00Hです。	

保守／廃止

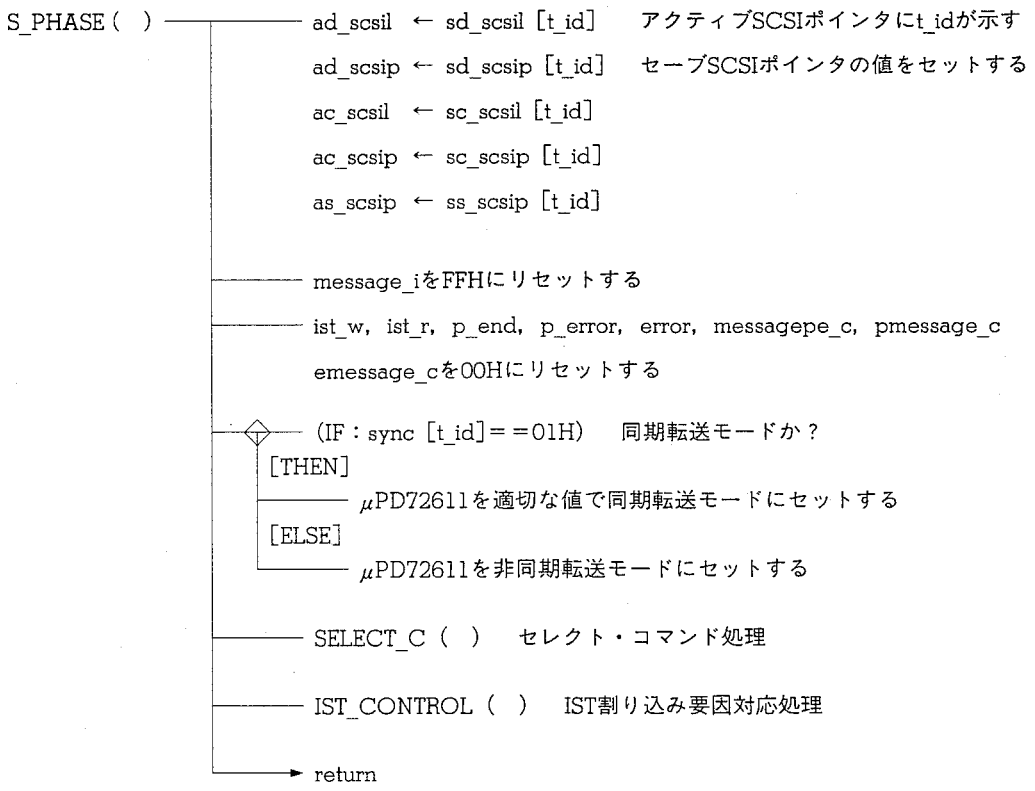
10.4　SCSIフェーズ処理モジュール説明

SCSIフェーズ処理の各モジュールの説明をします。**第5章　フォーマット・コマンド**の表5-4に示す内容に沿って説明し，SPDチャートを示します。

(1) SCSIフェーズ処理

モジュール名	S_PHASE
ファイル名	S_PHASE. CPP
言語	C言語
入力	message_o, sd_scsil [], sd_scsip [], sc_scsil [], sc_scsip [], ss_scsip [] t_id, i_message, expect_priod [], expect_offset [], sync []
出力	p_end, p_error, error, sync []
機能	μPD72611の制御およびSCSIフェーズの管理を行います。

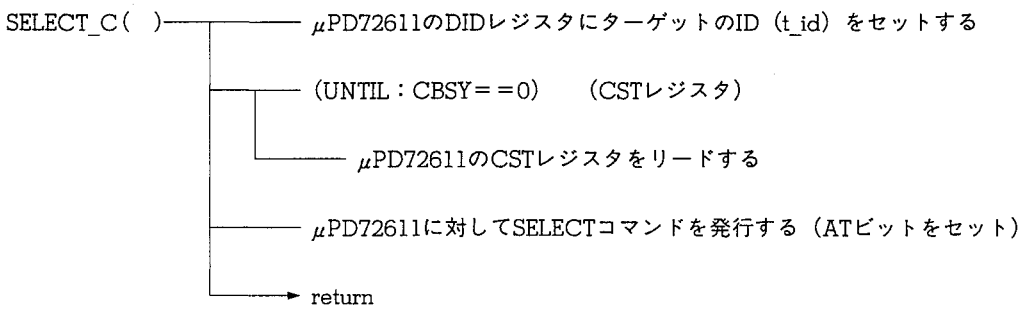
SPDチャート



(2) セレクト・コマンド処理

モジュール名	SELECT_C
ファイル名	S_PHASE. CPP
言語	C言語
入力	t_id
出力	なし
機能	SCSIバスを獲得し，ターゲットをセレクトします。

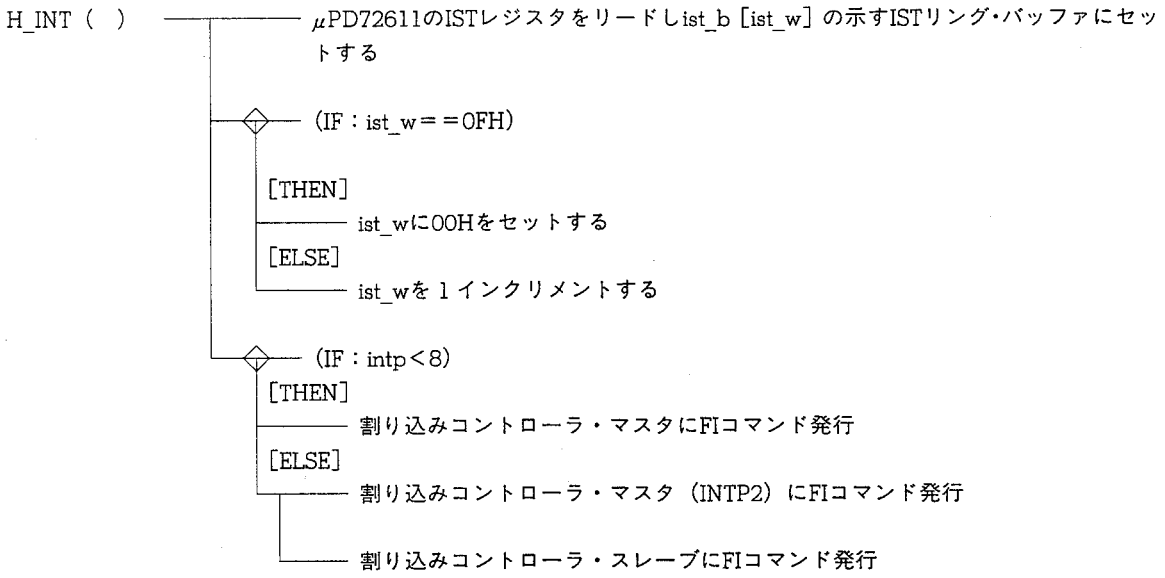
SPDチャート



(3) μPD72611ハードウエア割り込み処理

モジュール名	H_INT
ファイル名	S_PHASE. CPP
言語	C言語
入力	intp
出力	int_b []
機能	μPD72611のISTリング・バッファにセットします。

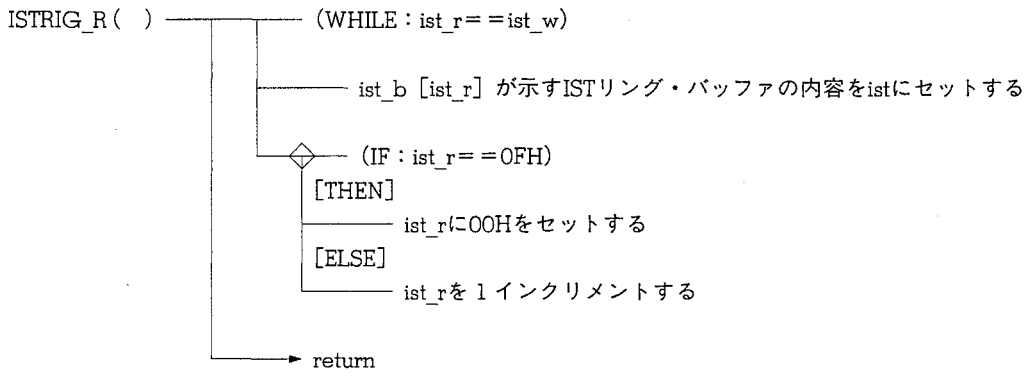
SPDチャート



(4) ISTリング・バッファ・リード処理

モジュール名	ISTRIG_R
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist_b [], ist_w
出力	ist
機能	ISTリング・バッファからμPD72611の割り込み要因をリードします。

SPDチャート

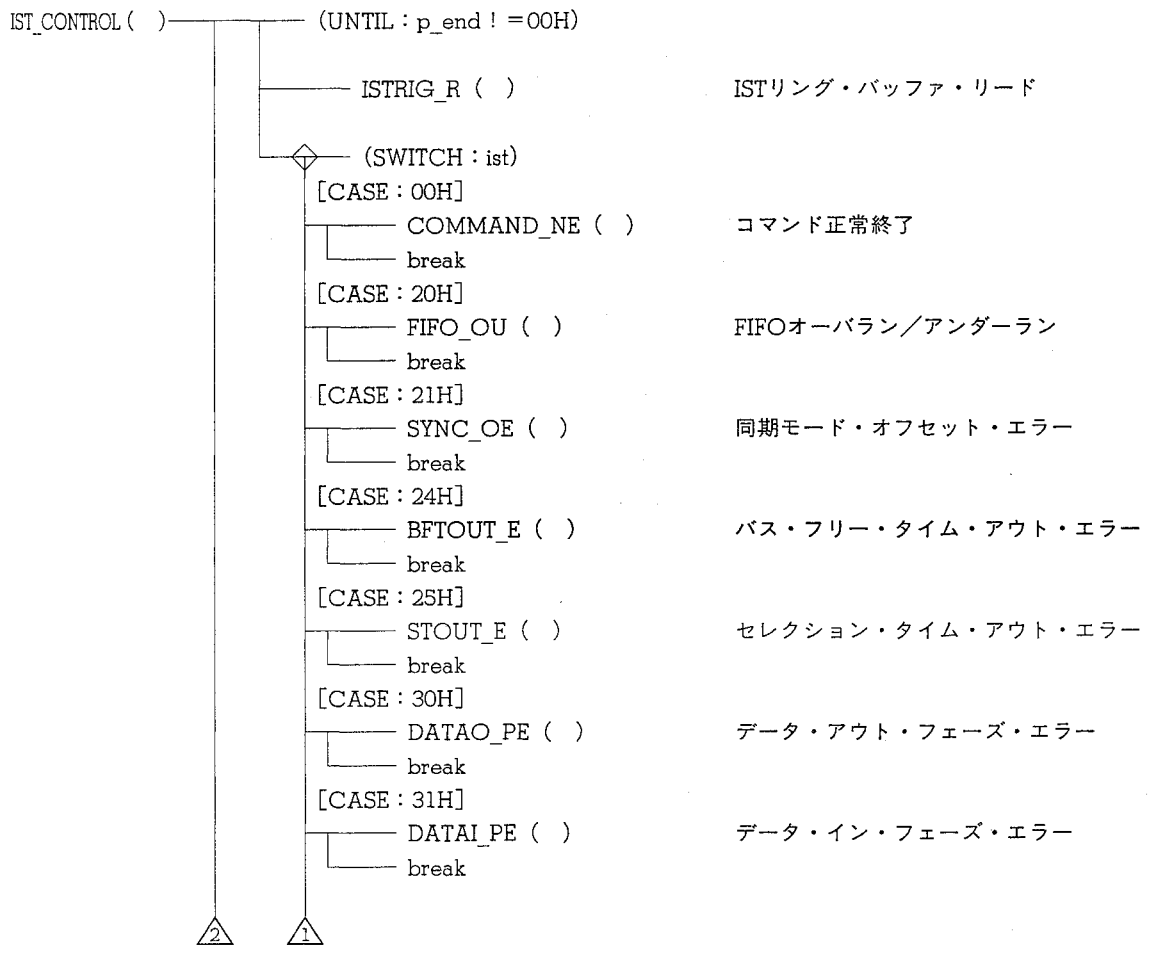


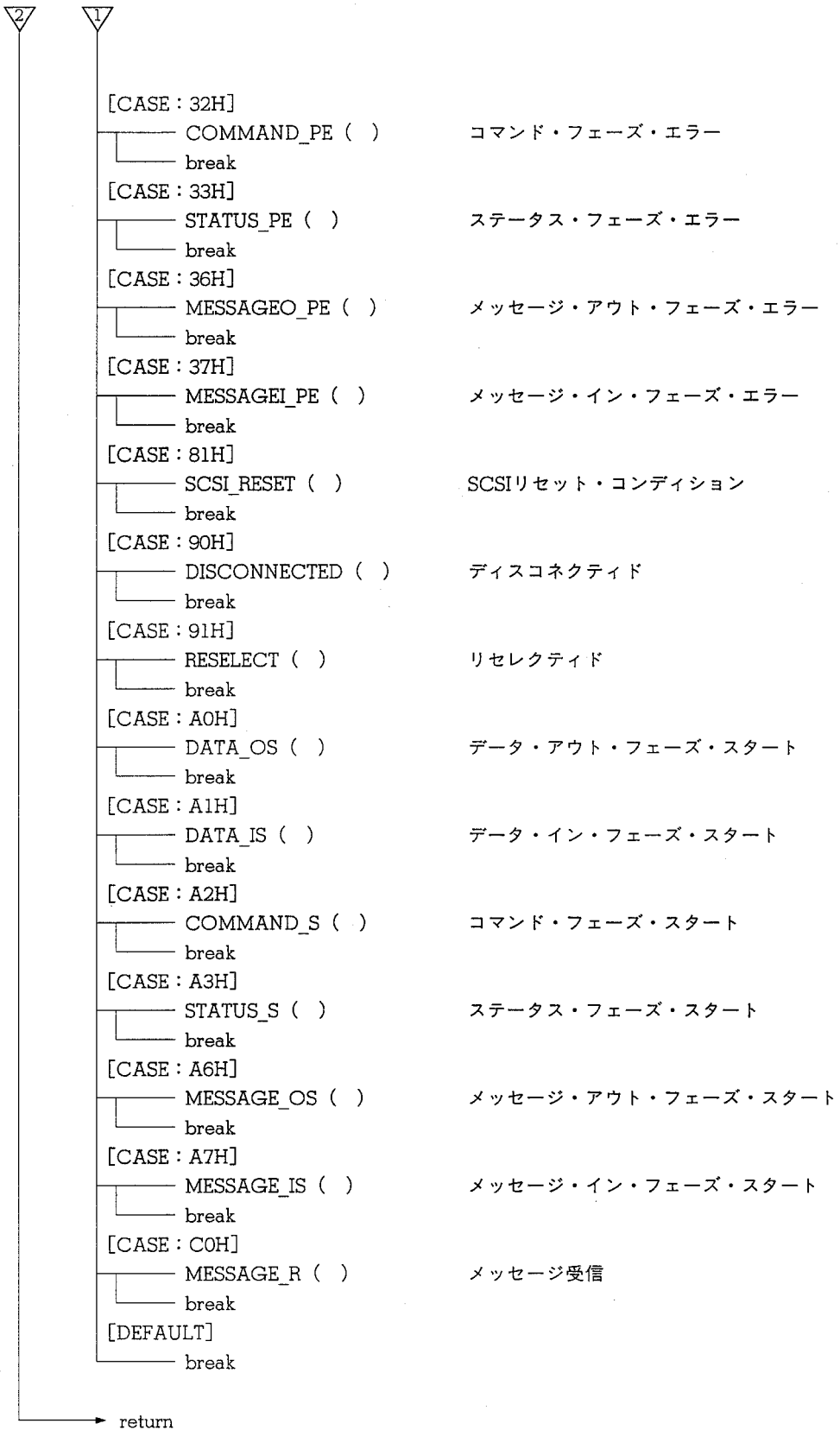
保守／廃止

(5) ISTの割り込み要因対応処理

モジュール名	IST_CONTROL
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist, pend
出力	なし
機能	ISTの割り込み要因に対応したモジュールをコールします。

SPDチャート



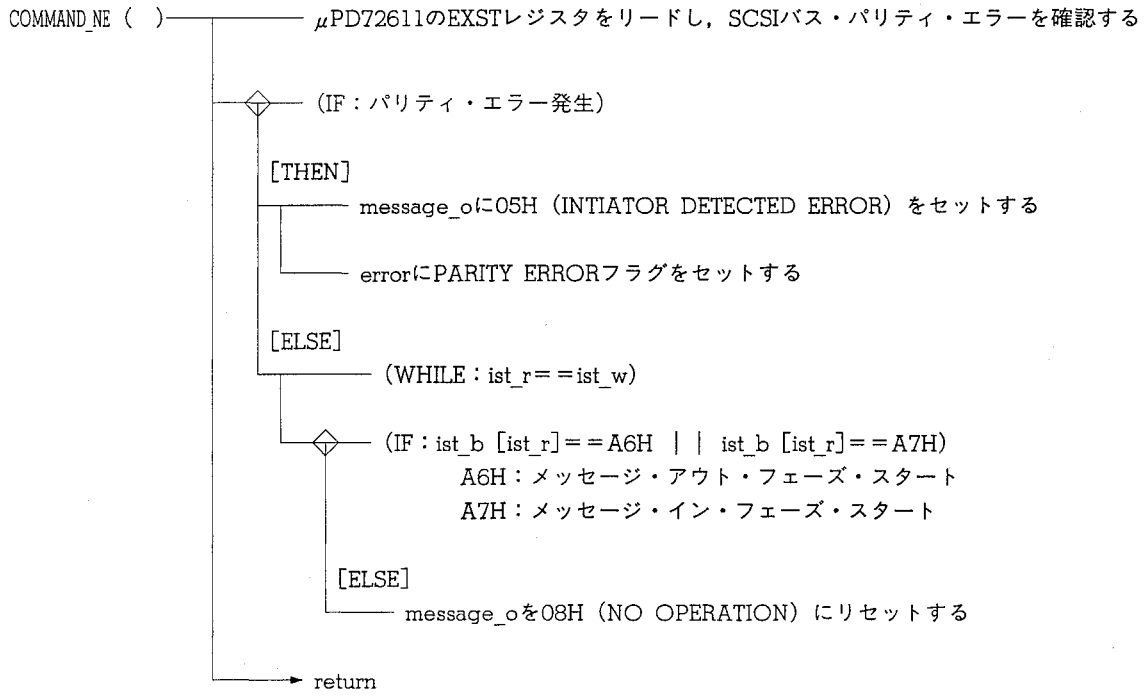

2
1
[CASE : 32H]
COMMAND_PE ( )
コマンド・フェーズ・エラー
break
[CASE : 33H]
STATUS_PE ( )
ステータス・フェーズ・エラー
break
[CASE : 36H]
MESSAGEO_PE ( )
メッセージ・アウト・フェーズ・エラー
break
[CASE : 37H]
MESSAGEI_PE ( )
メッセージ・イン・フェーズ・エラー
break
[CASE : 81H]
SCSI_RESET ( )
SCSIリセット・コンディション
break
[CASE : 90H]
DISCONNECTED ( )
ディスコネクティド
break
[CASE : 91H]
RESELECT ( )
リセレクティド
break
[CASE : A0H]
DATA_OS ( )
データ・アウト・フェーズ・スタート
break
[CASE : A1H]
DATA_IS ( )
データ・イン・フェーズ・スタート
break
[CASE : A2H]
COMMAND_S ( )
コマンド・フェーズ・スタート
break
[CASE : A3H]
STATUS_S ( )
ステータス・フェーズ・スタート
break
[CASE : A6H]
MESSAGE_OS ( )
メッセージ・アウト・フェーズ・スタート
break
[CASE : A7H]
MESSAGE_IS ( )
メッセージ・イン・フェーズ・スタート
break
[CASE : C0H]
MESSAGE_R ( )
メッセージ受信
break
[DEFAULT]
break
return

保守／廃止

(6) μPD72611コマンド正常終了処理

モジュール名	COMMAND_NE
ファイル名	S_PHASE.CPP
言語	C言語
入力	ist_b [], ist_r, ist_w
出力	message_o, error
機能	IST割り込み要因のコマンド正常終了に対応する処理を実行します。

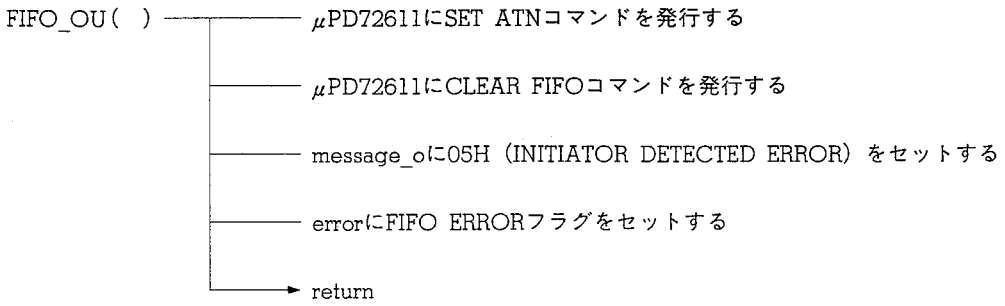
SPDチャート



(7) FIFOオーバラン／アンダーラン処理

モジュール名	FIFO_OU
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	message_o, error
機能	IST割り込み要因のFIFOオーバラン／アンダーランに対応する処理を実行します。

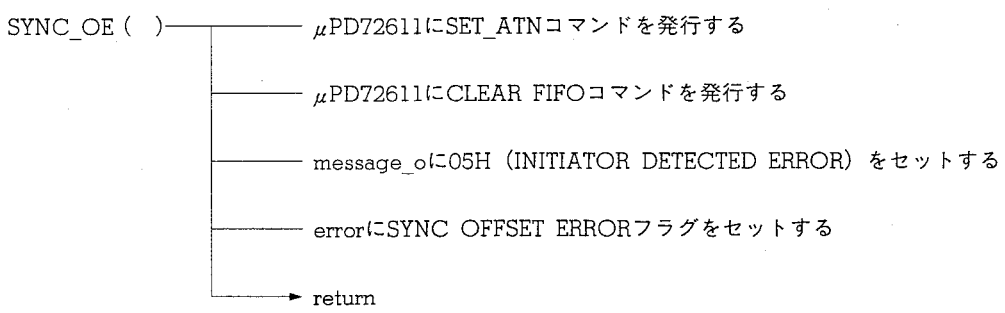
SPDチャート



(8) 同期モード・オフセット・エラー処理

モジュール名	SYNC_OE
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	message_o, error
機能	IST割り込み要因の同期モード・オフセット・エラーに対応する処理を実行します。

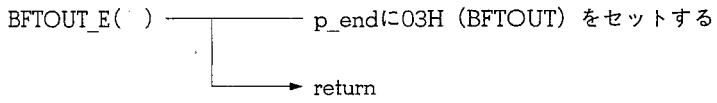
SPDチャート



(9) バス・フリー・タイム・アウト・エラー処理

モジュール名	BFTOUT_E
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	p_end
機能	IST割り込み要因のバス・フリー・タイム・アウト・エラーに対応する処理を実行します。

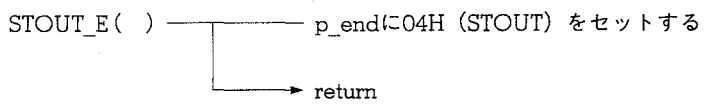
SPDチャート



(10) セレクション・タイム・アウト・エラー処理

モジュール名	STOUT_E
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	p_end
機能	IST割り込み要因のセレクション・タイム・アウト・エラーに対応する処理を実行します。

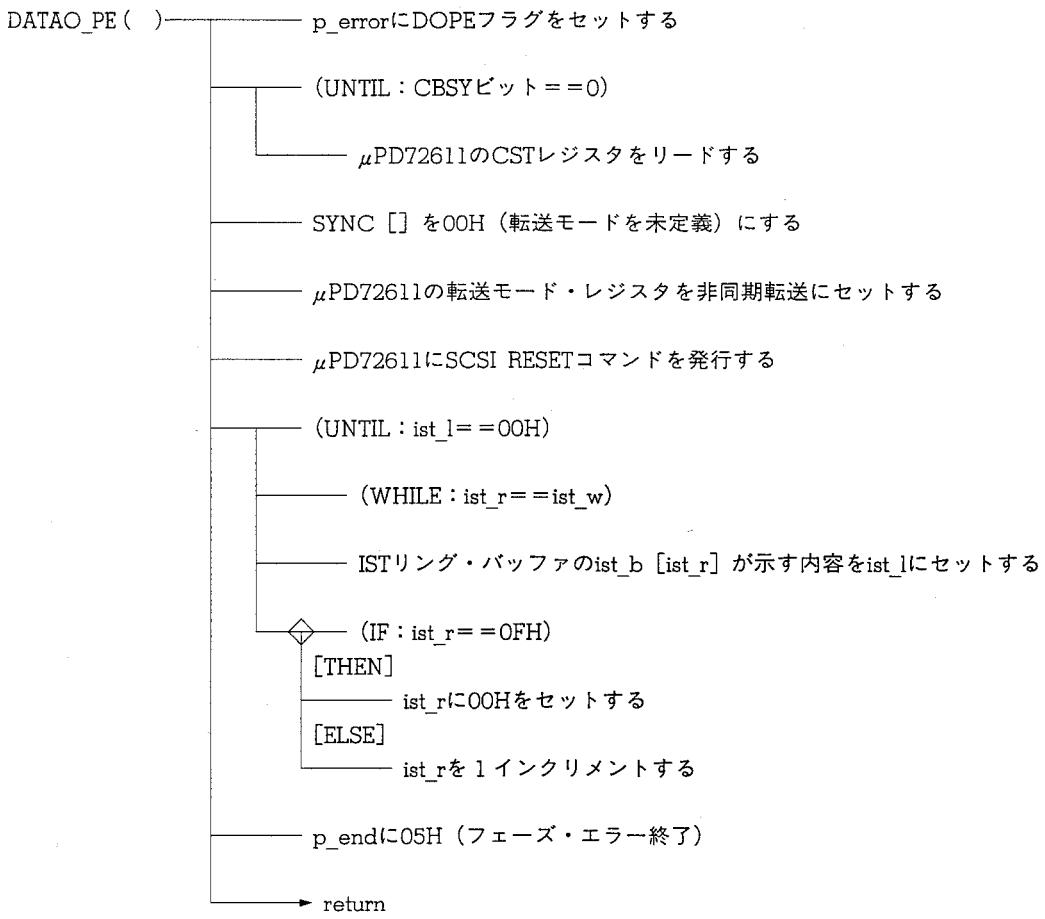
SPDチャート



(11) データ・アウト・フェーズ・エラー処理

モジュール名	DATAO_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist_b [], ist_r, ist_w
出力	p_error, p_end
機能	IST割り込み要因のデータ・アウト・フェーズ・エラーに対応する処理を実行します。

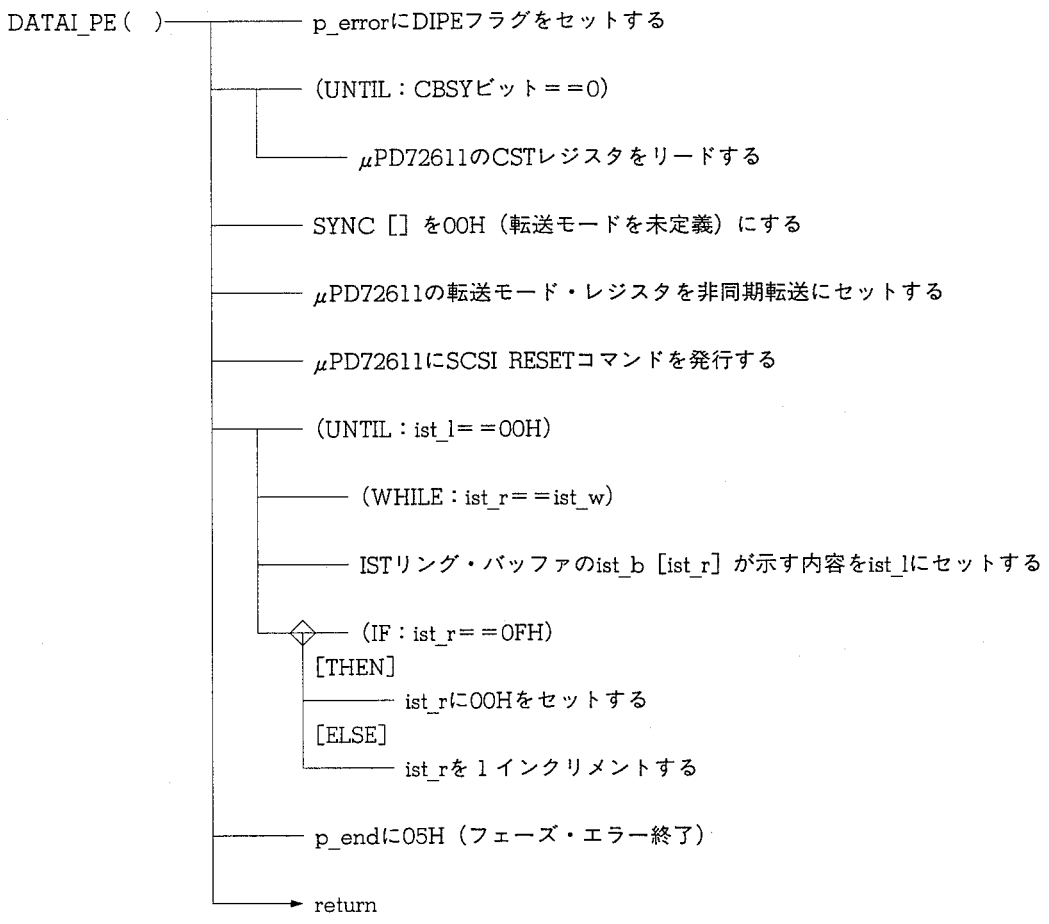
SPDチャート



(12) データ・イン・フェーズ・エラー処理

モジュール名	DATAI_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist_b []，ist_r，ist_w
出力	p_error，p_end
機能	IST割り込み要因のデータ・イン・フェーズ・エラーに対応する処理を実行します。

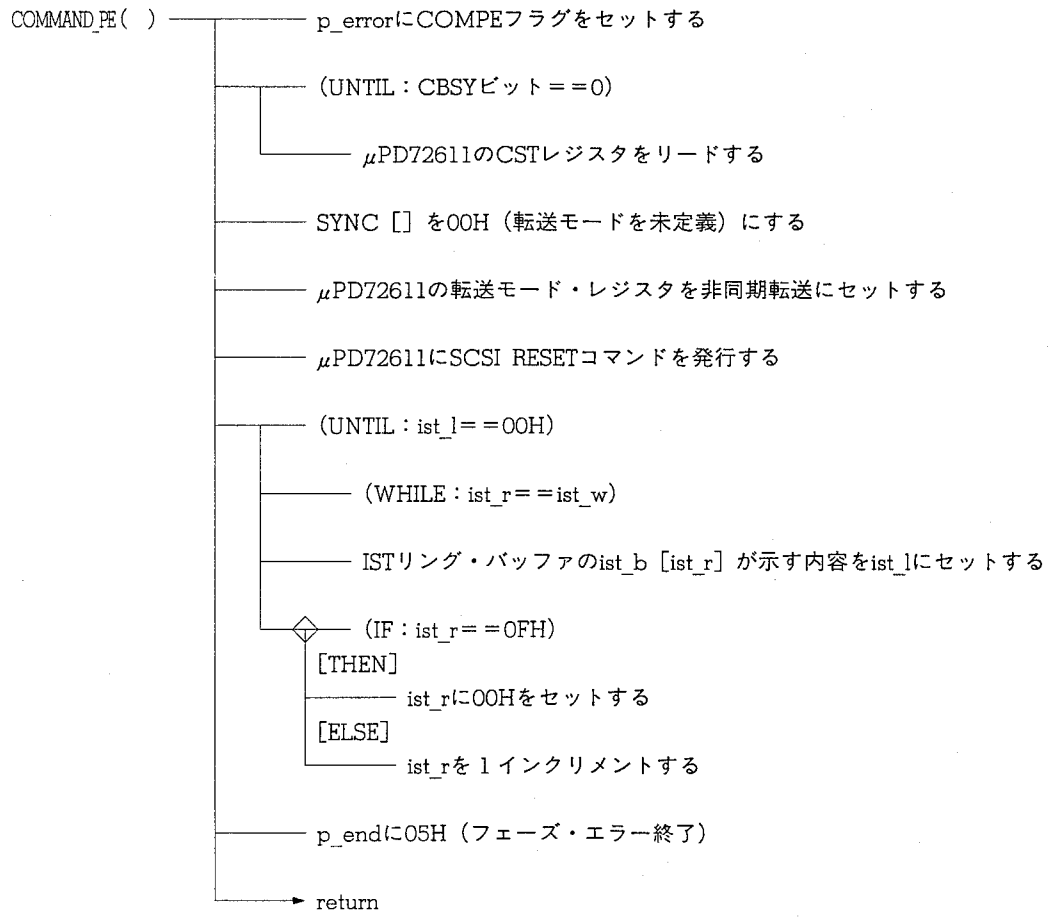
SPDチャート



(13) コマンド・フェーズ・エラー処理

モジュール名	COMMAND_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist_b [], ist_r, ist_w
出力	p_error, p_end
機能	IST割り込み要因のコマンド・フェーズ・エラーに対応する処理を実行します。

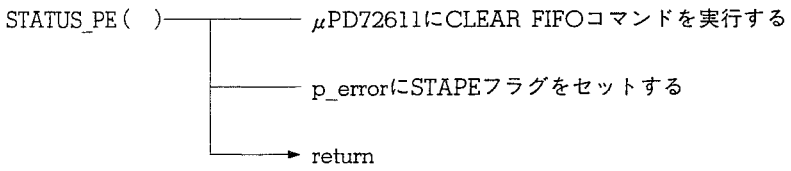
SPDチャート



(14) ステータス・フェーズ・エラー処理

モジュール名	STATUS_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	p_error
機能	IST割り込み要因のステータス・フェーズ・エラーに対応する処理を実行します。

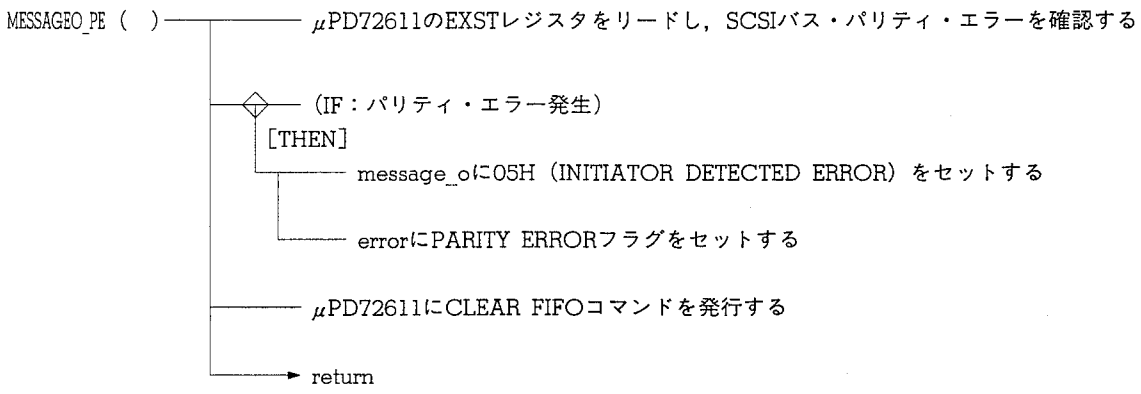
SPDチャート



(15) メッセージ・アウト・フェーズ・エラー処理

モジュール名	MESSAGEO_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	message_o, error
機能	IST割り込み要因のメッセージ・アウト・フェーズ・エラーに対応する処理を実行します。

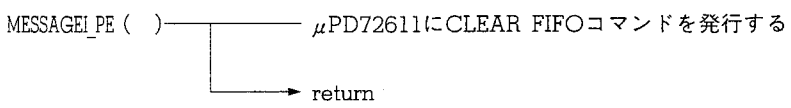
SPDチャート



(16) メッセージ・イン・フェーズ・エラー処理

モジュール名	MESSAGEI_PE
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	なし
機能	IST割り込み要因のメッセージ・イン・フェーズ・エラーに対応する処理を実行します。

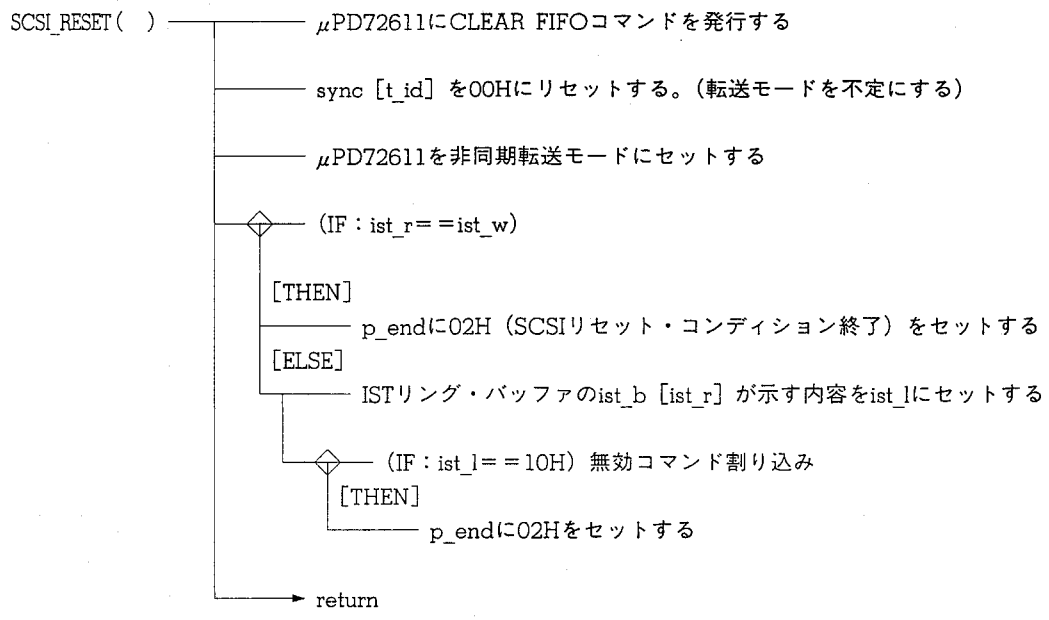
SPDチャート



(17) SCSIリセット・コンディション処理

モジュール名	SCSI_RESET
ファイル名	S_PHASE. CPP
言語	C言語
入力	ist_b [], ist_w, ist_r
出力	p_end, sync []
機能	IST割り込み要因のSCSIリセット・コンディションに対応する処理を実行します。

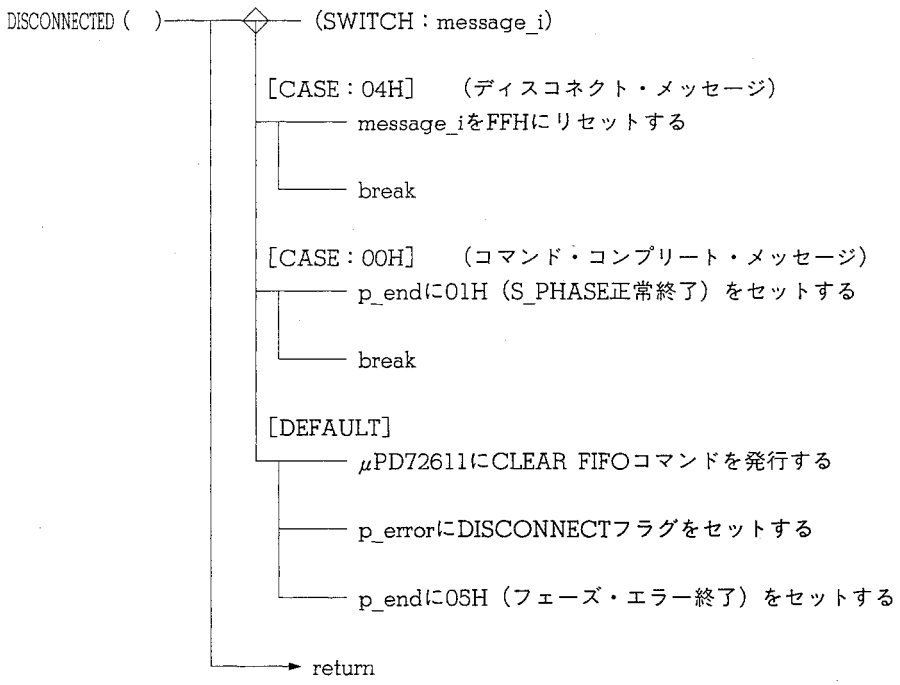
SPDチャート



(18) ディスコネクティド処理

モジュール名	DISCONNECTED
ファイル名	S_PHASE. CPP
言語	C言語
入力	message_i
出力	message_i, p_end, p_error
機能	IST割り込み要因のディスコネクティドに対応する処理を実行します。

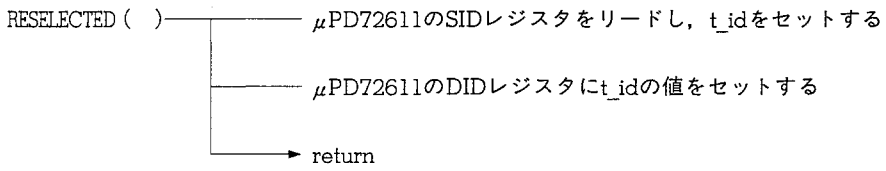
SPDチャート



(19) リセレクティド処理

モジュール名	RESELECTED
ファイル名	S_PHASE.CPP
言語	C言語
入力	なし
出力	t_id
機能	IST割り込み要因のリセレクティドに対応する処理を実行します。

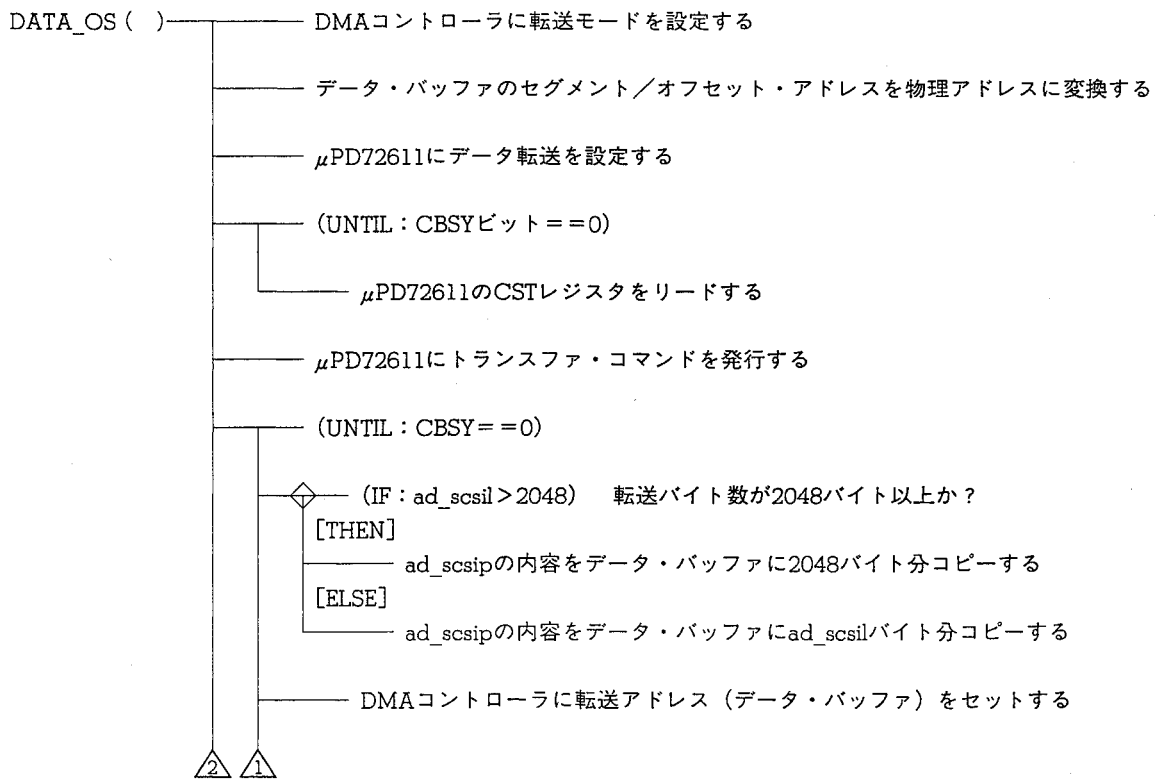
SPDチャート

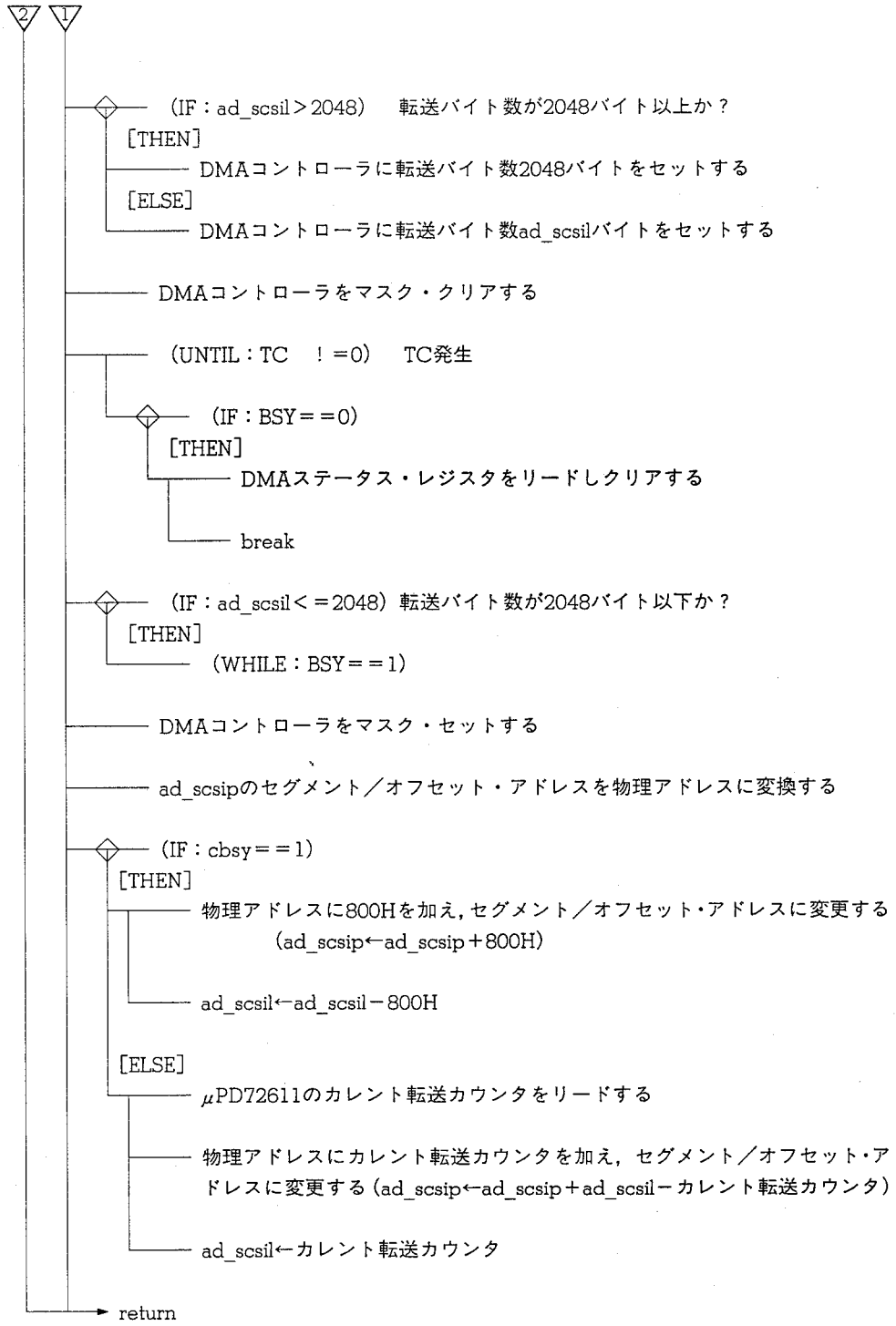


(20) データ・アウト・フェーズ・スタート処理

モジュール名	DATA_OS
ファイル名	S_PHASE. CPP
言語	C言語
入力	ad_scsil, ad_scsip
出力	なし
機能	IST割り込み要因のデータ・アウト・フェーズ・スタートに対応する処理を実行します。

SPDチャート

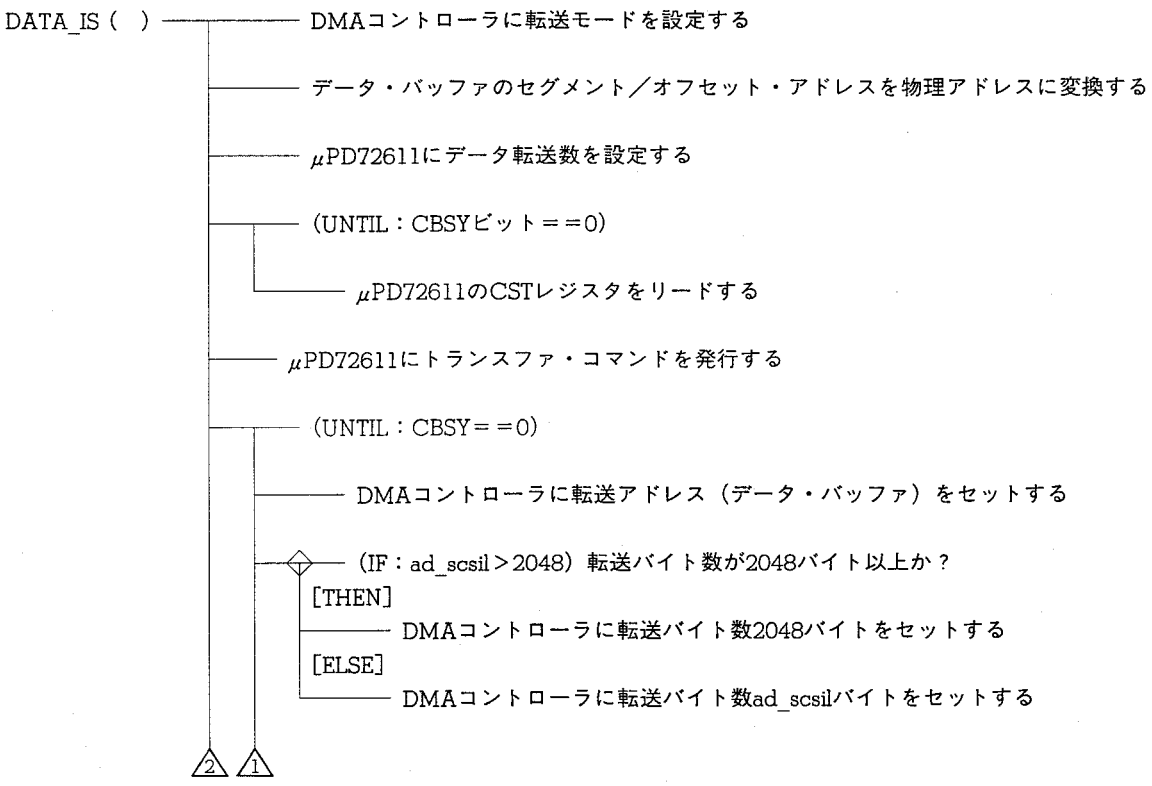



2
1
(IF：ad_scsil＞2048)　転送バイト数が2048バイト以上か？
[THEN]
DMAコントローラに転送バイト数2048バイトをセットする
[ELSE]
DMAコントローラに転送バイト数ad_scsilバイトをセットする
DMAコントローラをマスク・クリアする
(UNTIL：TC　！＝0)　TC発生
(IF：BSY＝＝0)
[THEN]
DMAステータス・レジスタをリードしクリアする
break
(IF：ad_scsil＜＝2048) 転送バイト数が2048バイト以下か？
[THEN]
(WHILE：BSY＝＝1)
DMAコントローラをマスク・セットする
ad_scsipのセグメント／オフセット・アドレスを物理アドレスに変換する
(IF：cbsy＝＝1)
[THEN]
物理アドレスに800Hを加え，セグメント／オフセット・アドレスに変更する
(ad_scsip←ad_scsip＋800H)
ad_scsil←ad_scsil－800H
[ELSE]
μPD72611のカレント転送カウンタをリードする
物理アドレスにカレント転送カウンタを加え，セグメント／オフセット・アドレスに変更する(ad_scsip←ad_scsip＋ad_scsil－カレント転送カウンタ)
ad_scsil←カレント転送カウンタ
return

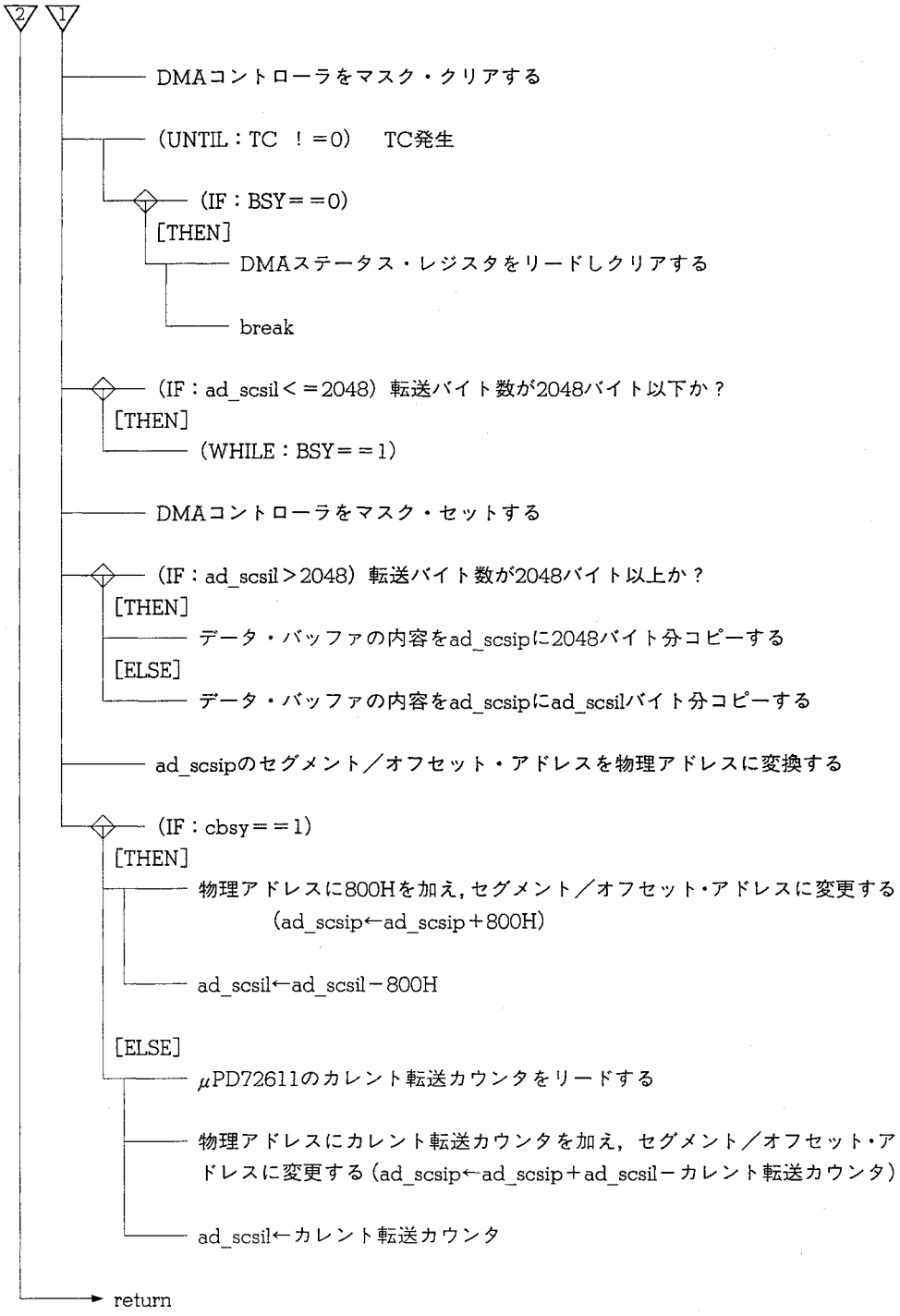
(21) データ・イン・フェーズ・スタート処理

モジュール名	DATA_IS
ファイル名	S_PHASE. CPP
言語	C言語
入力	ad_scsil, ad_scsip
出力	なし
機能	IST割り込み要因のデータ・イン・フェーズ・スタートに対応する処理を実行します。

SPDチャート



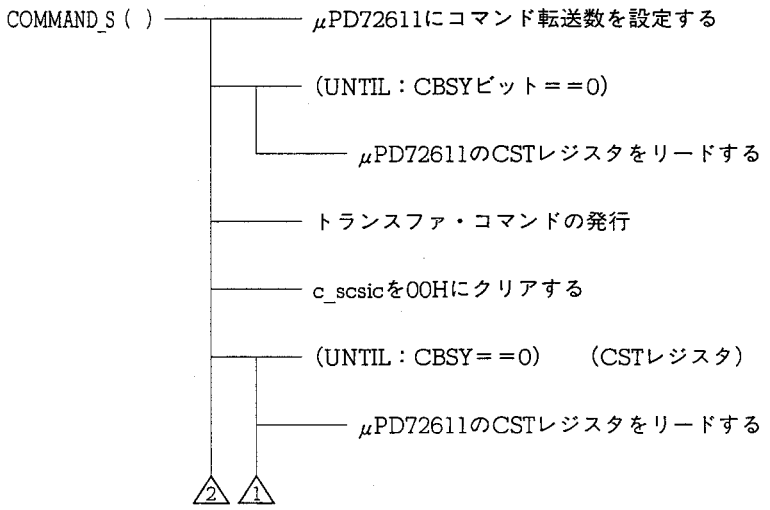
保守／廃止

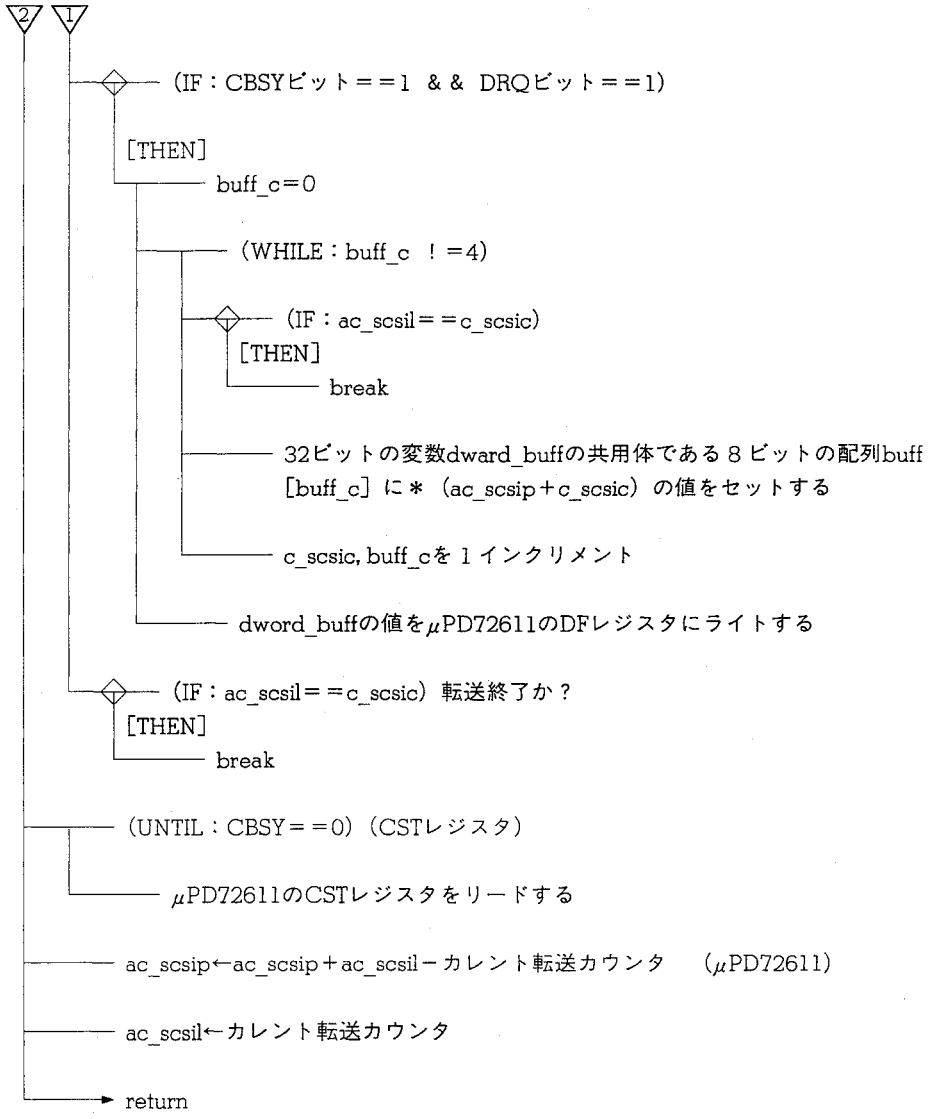

2
1
DMAコントローラをマスク・クリアする
(UNTIL：TC ！＝0)　TC発生
(IF：BSY＝＝0)
[THEN]
DMAステータス・レジスタをリードしクリアする
break
(IF：ad_scsil＜＝2048) 転送バイト数が2048バイト以下か？
[THEN]
(WHILE：BSY＝＝1)
DMAコントローラをマスク・セットする
(IF：ad_scsil＞2048) 転送バイト数が2048バイト以上か？
[THEN]
データ・バッファの内容をad_scsipに2048バイト分コピーする
[ELSE]
データ・バッファの内容をad_scsipにad_scsilバイト分コピーする
ad_scsipのセグメント／オフセット・アドレスを物理アドレスに変換する
(IF：cbsy＝＝1)
[THEN]
物理アドレスに800Hを加え，セグメント／オフセット・アドレスに変更する
(ad_scsip←ad_scsip＋800H)
ad_scsil←ad_scsil－800H
[ELSE]
μPD72611のカレント転送カウンタをリードする
物理アドレスにカレント転送カウンタを加え，セグメント／オフセット・アドレスに変更する(ad_scsip←ad_scsip＋ad_scsil－カレント転送カウンタ)
ad_scsil←カレント転送カウンタ
return

(22) コマンド・フェーズ・スタート処理

モジュール名	COMMAND_S
ファイル名	S_PHASE. CPP
言語	C言語
入力	ac_scsil, ad_scsip
出力	なし
機能	IST割り込み要因のコマンド・フェーズ・スタートに対応する処理を実行します。

SPDチャート

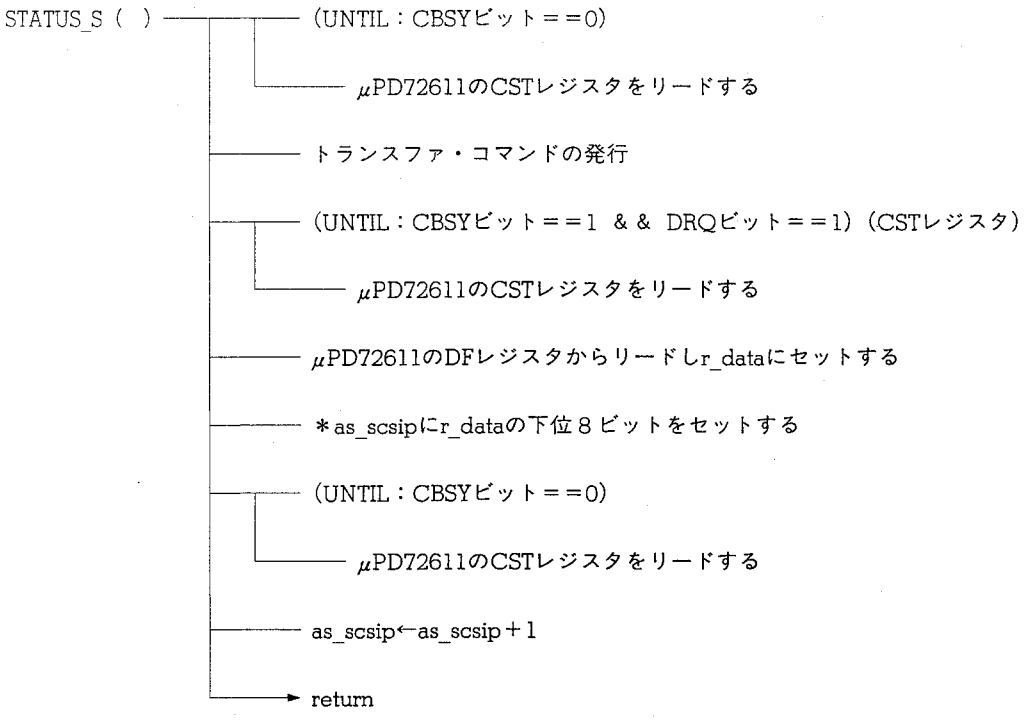



2
1
(IF：CBSYビット＝＝1 ＆＆ DRQビット＝＝1)
[THEN]
buff_c=0
(WHILE：buff_c ！=4)
(IF：ac_scsil＝＝c_scsic)
[THEN]
break
32ビットの変数dward_buffの共用体である８ビットの配列buff
[buff_c] に＊（ac_scsip＋c_scsic）の値をセットする
c_scsic, buff_cを１インクリメント
dword_buffの値をμPD72611のDFレジスタにライトする
(IF：ac_scsil＝＝c_scsic）転送終了か？
[THEN]
break
(UNTIL：CBSY＝＝0)（CSTレジスタ）
μPD72611のCSTレジスタをリードする
ac_scsip←ac_scsip＋ac_scsil－カレント転送カウンタ　（μPD72611）
ac_scsil←カレント転送カウンタ
return

(23) ステータス・フェーズ・スタート処理

モジュール名	STATUS_S
ファイル名	S_PHASE. CPP
言語	C言語
入力	as_scsip
出力	なし
機能	IST割り込み要因のステータス・フェーズ・スタートに対応する処理を実行します。

SPDチャート

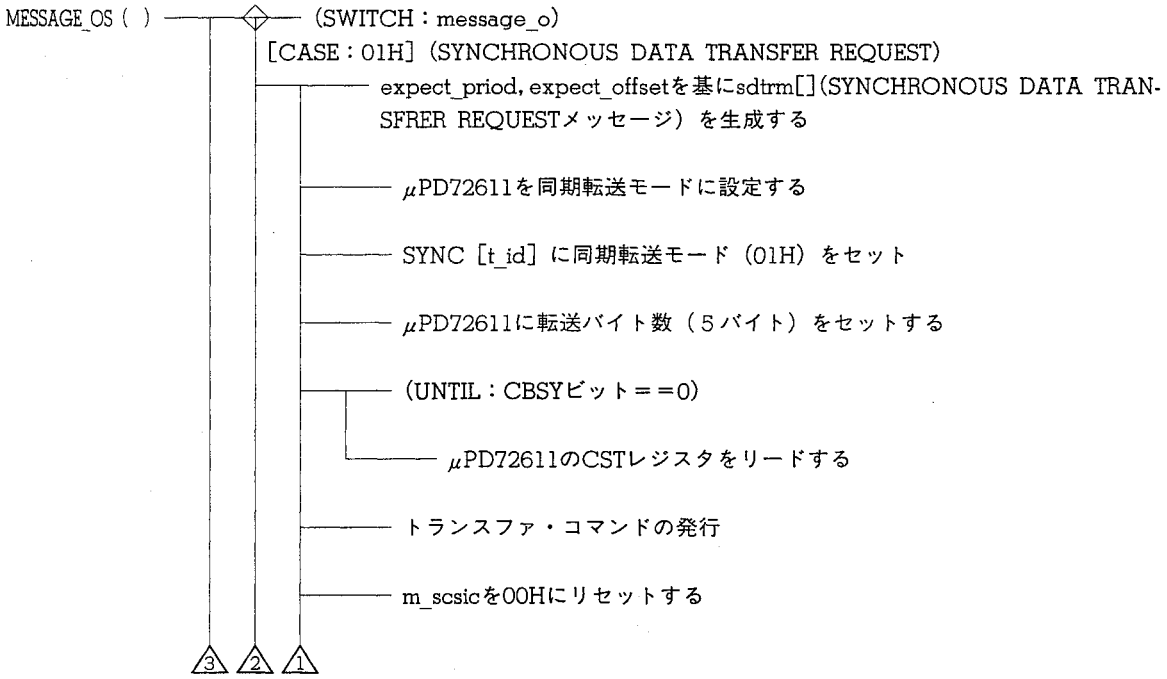


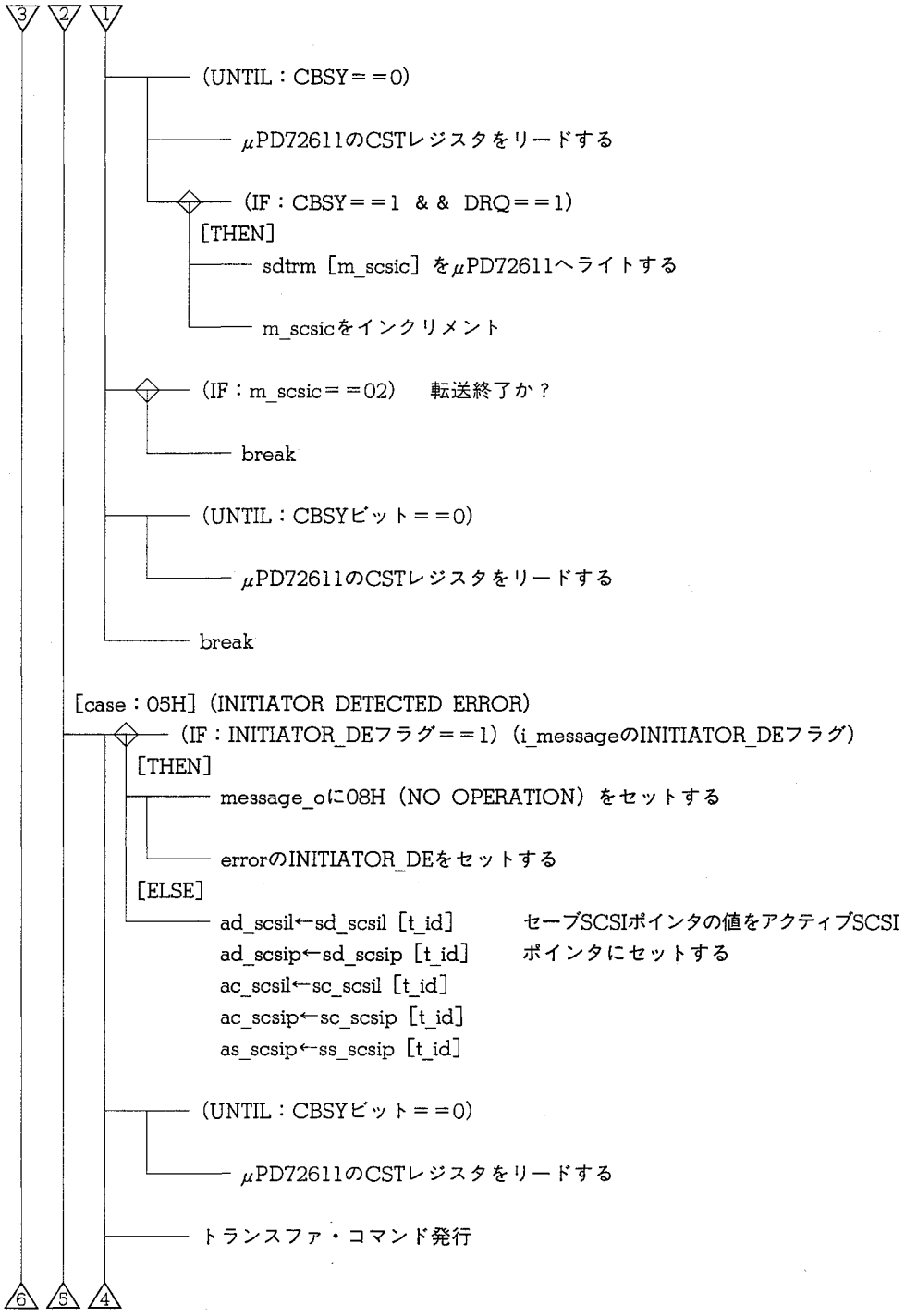
保守／廃止

(24) メッセージ・アウト・フェーズ・スタート処理

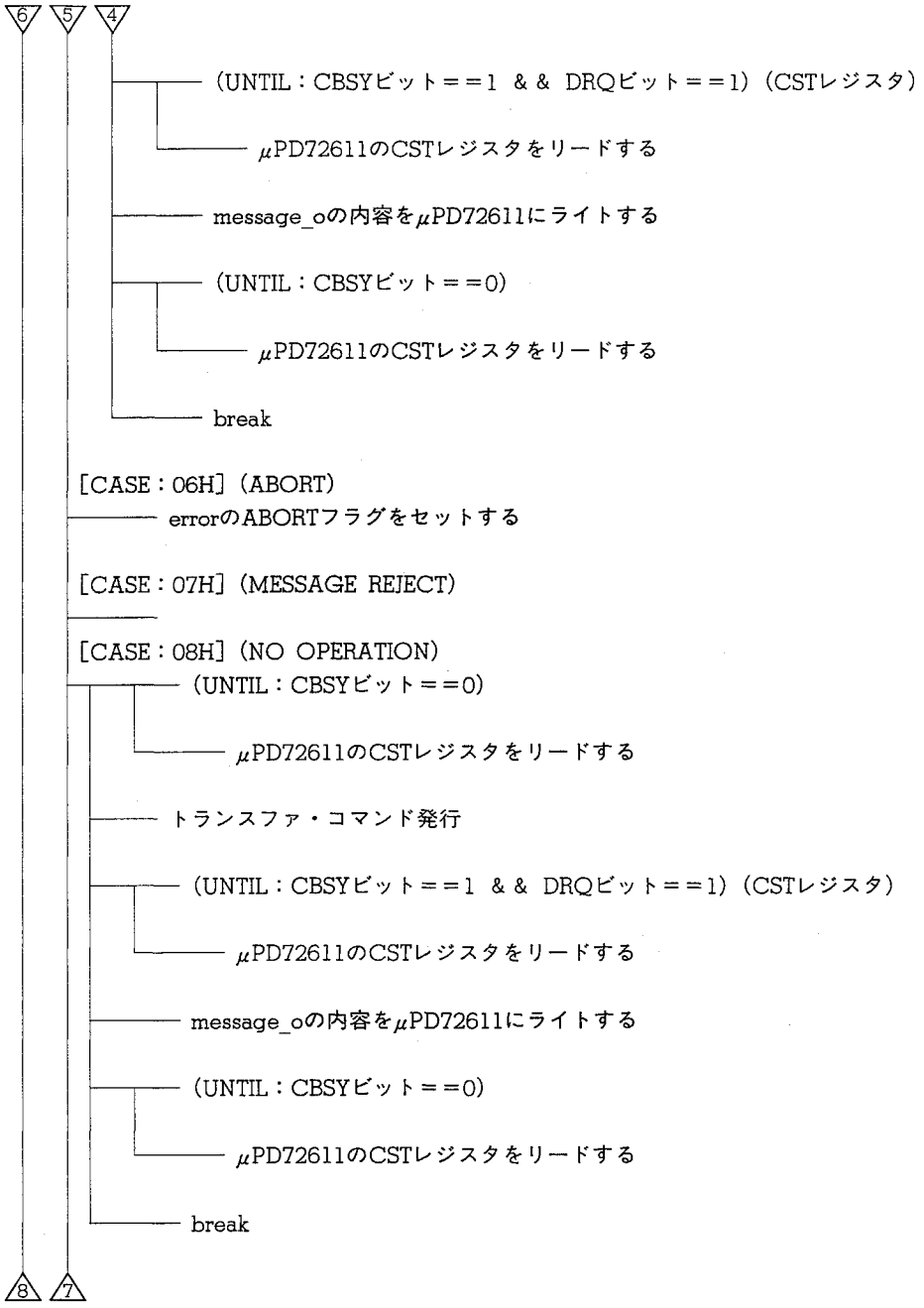
モジュール名	MESSAGE_OS
ファイル名	S_PHASE. CPP
言語	C言語
入力	message_o, i_message, expect_priod [], expect_offset, t_id, sd_scsil [] sd_scsip [], sc_scsil [], sc_scsip [], ss_scsip []
出力	error, messagepe_c, sync [], pmessage_c, ad_scsil, ad_scsip, ac_scsil ac_scsip, as_scsip
機能	IST割り込み要因のメッセージ・アウト・フェーズ・スタートに対応する処理を実行します。

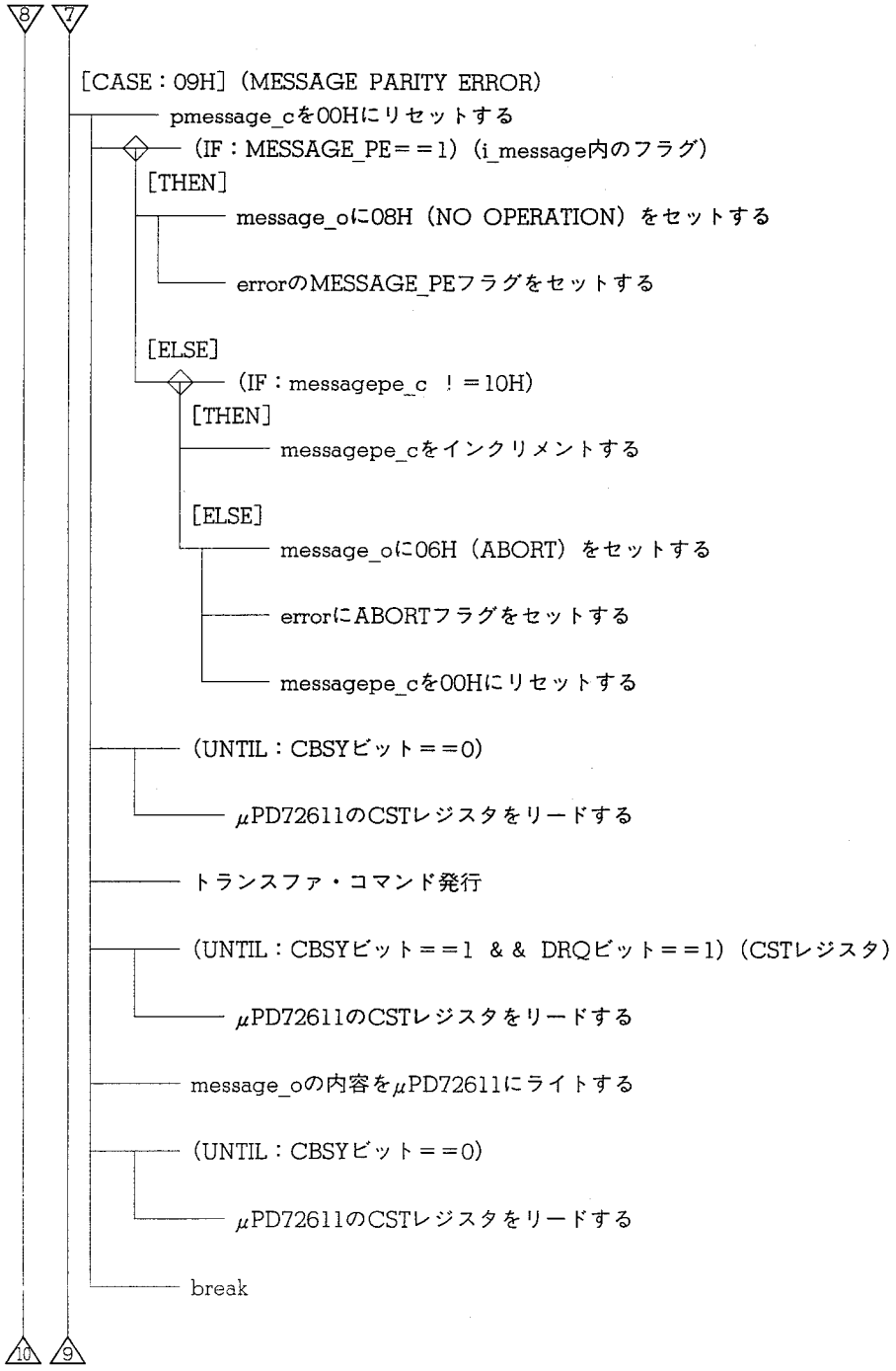
SPDチャート

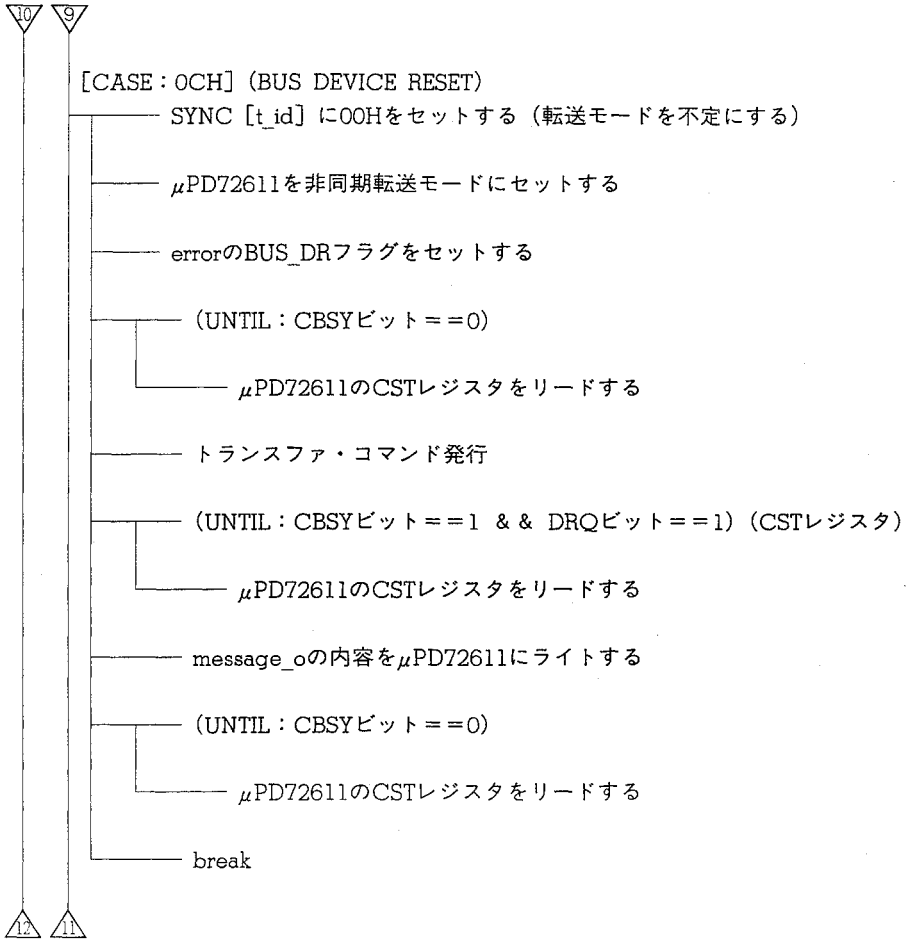


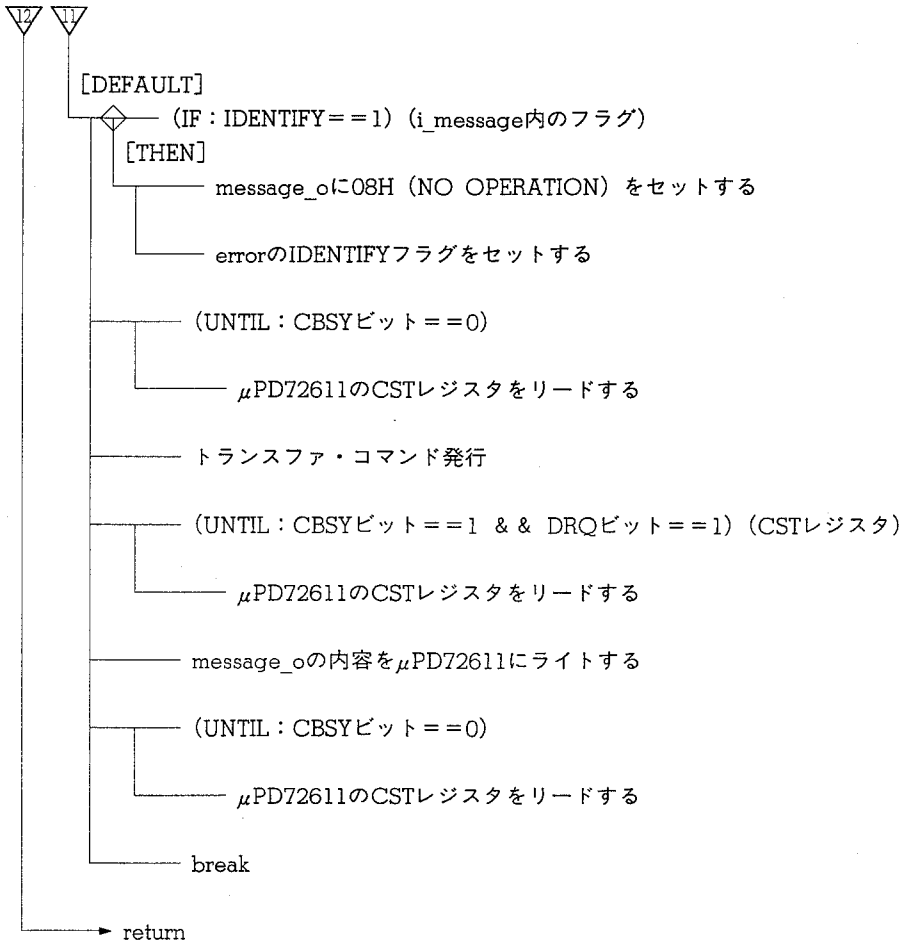

3
2
1
(UNTIL：CBSY＝＝0)
μPD72611のCSTレジスタをリードする
(IF：CBSY＝＝1 ＆＆ DRQ＝＝1)
[THEN]
sdtrm [m_scsic] をμPD72611へライトする
m_scsicをインクリメント
(IF：m_scsic＝＝02)　転送終了か？
break
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
break
[case：05H] (INITIATOR DETECTED ERROR)
(IF：INITIATOR_DEフラグ＝＝1) (i_messageのINITIATOR_DEフラグ)
[THEN]
message_oに08H (NO OPERATION) をセットする
errorのINITIATOR_DEをセットする
[ELSE]
ad_scsil←sd_scsil [t_id]
ad_scsip←sd_scsip [t_id]
ac_scsil←sc_scsil [t_id]
ac_scsip←sc_scsip [t_id]
as_scsip←ss_scsip [t_id]
セーブSCSIポインタの値をアクティブSCSIポインタにセットする
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
トランスファ・コマンド発行
6
5
4

保守／廃止


6
5
4
(UNTIL：CBSYビット＝＝1 ＆＆ DRQビット＝＝1)（CSTレジスタ）
μPD72611のCSTレジスタをリードする
message_oの内容をμPD72611にライトする
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
break
[CASE：06H]（ABORT）
errorのABORTフラグをセットする
[CASE：07H]（MESSAGE REJECT）
[CASE：08H]（NO OPERATION）
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
トランスファ・コマンド発行
(UNTIL：CBSYビット＝＝1 ＆＆ DRQビット＝＝1)（CSTレジスタ）
μPD72611のCSTレジスタをリードする
message_oの内容をμPD72611にライトする
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
break
8
7


8
7
[CASE：09H]（MESSAGE PARITY ERROR）
pmessage_cを00Hにリセットする
（IF：MESSAGE_PE＝＝1）（i_message内のフラグ）
[THEN]
message_oに08H（NO OPERATION）をセットする
errorのMESSAGE_PEフラグをセットする
[ELSE]
（IF：messagepe_c ！＝10H）
[THEN]
messagepe_cをインクリメントする
[ELSE]
message_oに06H（ABORT）をセットする
errorにABORTフラグをセットする
messagepe_cを00Hにリセットする
（UNTIL：CBSYビット＝＝0）
μPD72611のCSTレジスタをリードする
トランスファ・コマンド発行
（UNTIL：CBSYビット＝＝1 ＆＆ DRQビット＝＝1）（CSTレジスタ）
μPD72611のCSTレジスタをリードする
message_oの内容をμPD72611にライトする
（UNTIL：CBSYビット＝＝0）
μPD72611のCSTレジスタをリードする
break
10
9

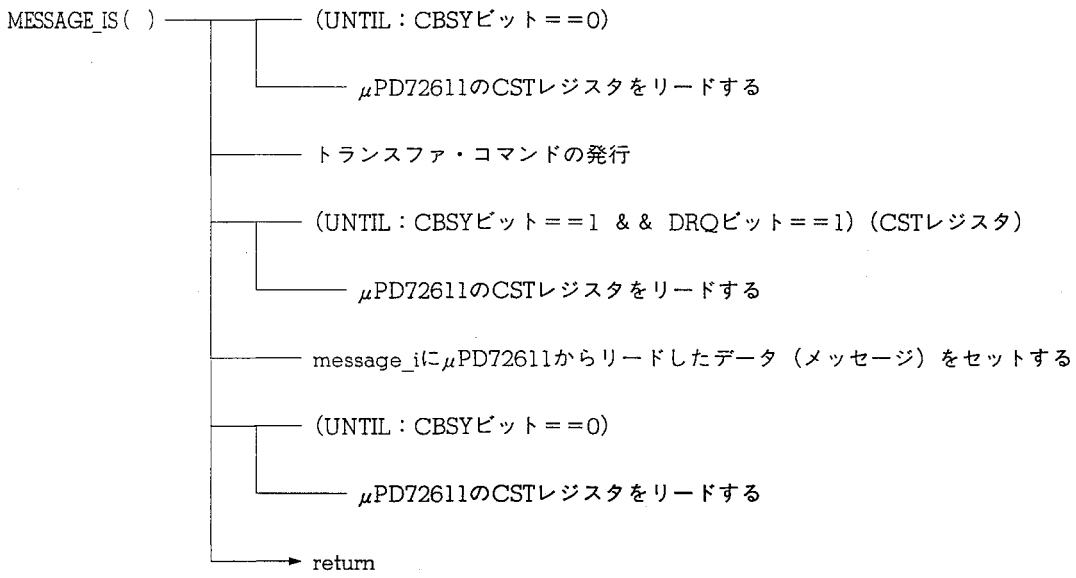

10
9
[CASE：0CH]（BUS DEVICE RESET）
SYNC［t_id］に00Hをセットする（転送モードを不定にする）
μPD72611を非同期転送モードにセットする
errorのBUS_DRフラグをセットする
（UNTIL：CBSYビット＝＝0）
μPD72611のCSTレジスタをリードする
トランスファ・コマンド発行
（UNTIL：CBSYビット＝＝1 ＆＆ DRQビット＝＝1）（CSTレジスタ）
μPD72611のCSTレジスタをリードする
message_oの内容をμPD72611にライトする
（UNTIL：CBSYビット＝＝0）
μPD72611のCSTレジスタをリードする
break
12
11


12
11
[DEFAULT]
(IF：IDENTIFY＝＝1)（i_message内のフラグ）
[THEN]
message_oに08H（NO OPERATION）をセットする
errorのIDENTIFYフラグをセットする
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
トランスファ・コマンド発行
(UNTIL：CBSYビット＝＝1 ＆＆ DRQビット＝＝1)（CSTレジスタ）
μPD72611のCSTレジスタをリードする
message_oの内容をμPD72611にライトする
(UNTIL：CBSYビット＝＝0)
μPD72611のCSTレジスタをリードする
break
return

(25) メッセージ・イン・フェーズ・スタート処理

モジュール名	MESSAGE_IS
ファイル名	S_PHASE. CPP
言語	C言語
入力	なし
出力	message_i
機能	IST割り込み要因のメッセージ・イン・フェーズ・スタートに対応する処理を実行します。

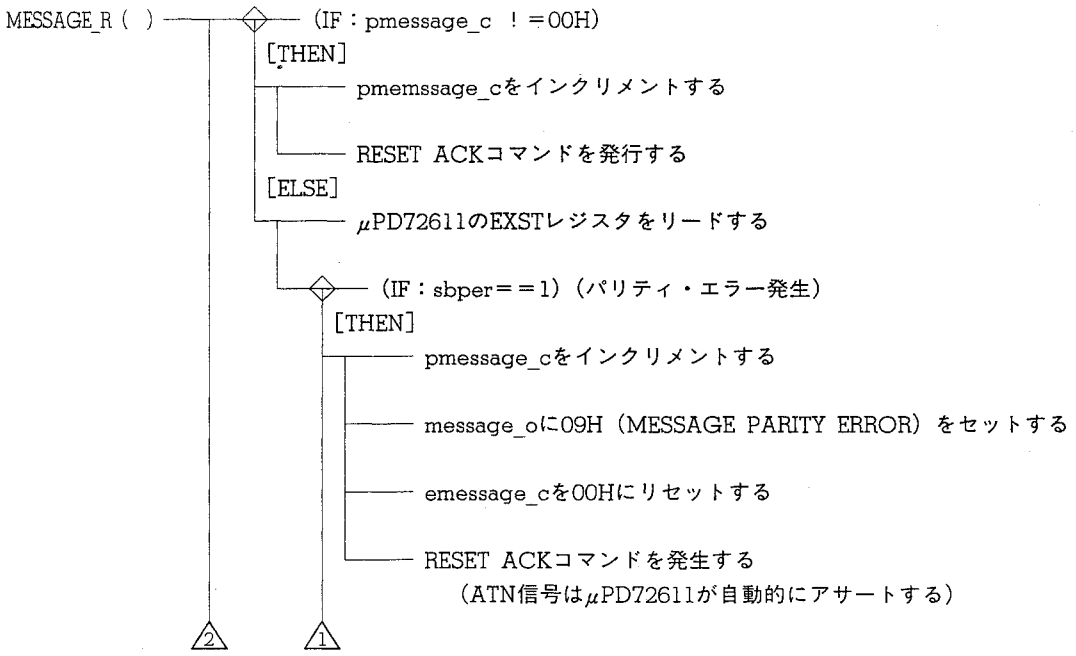
SPDチャート

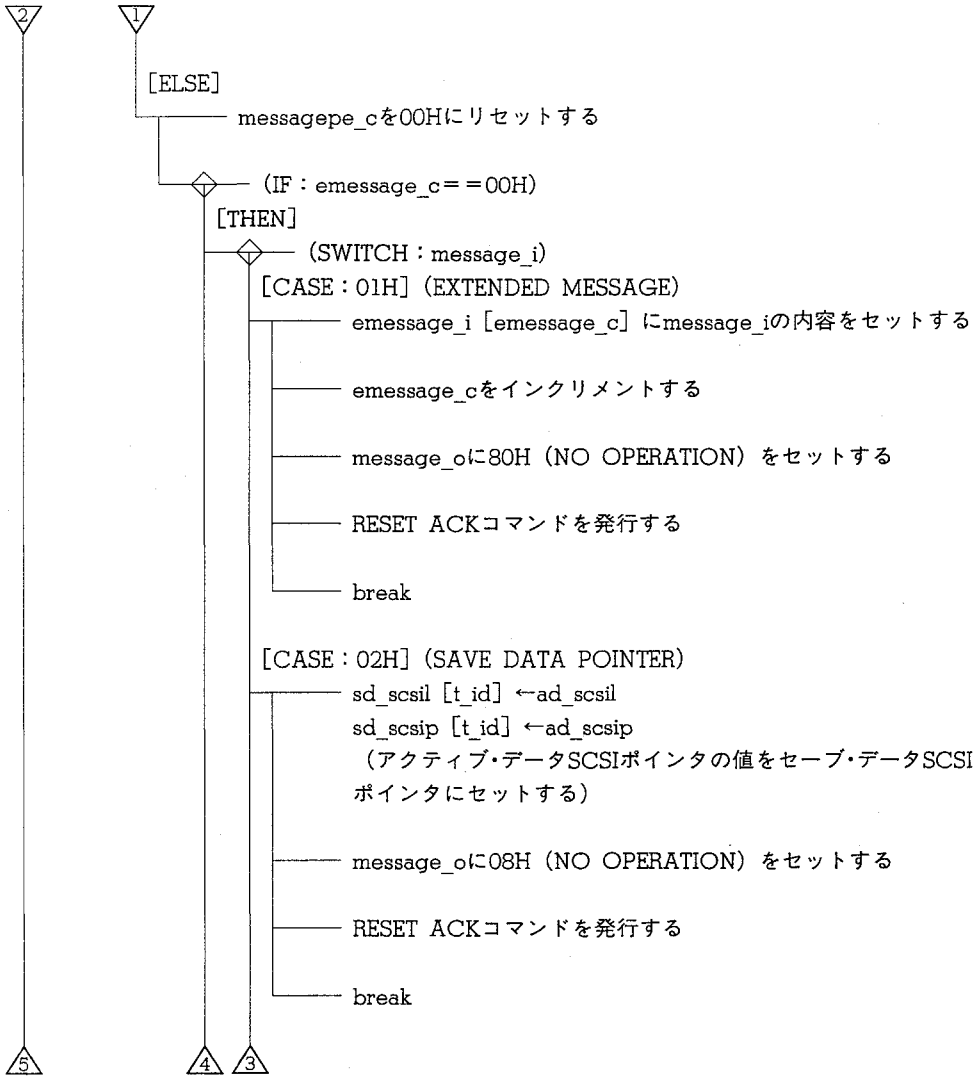


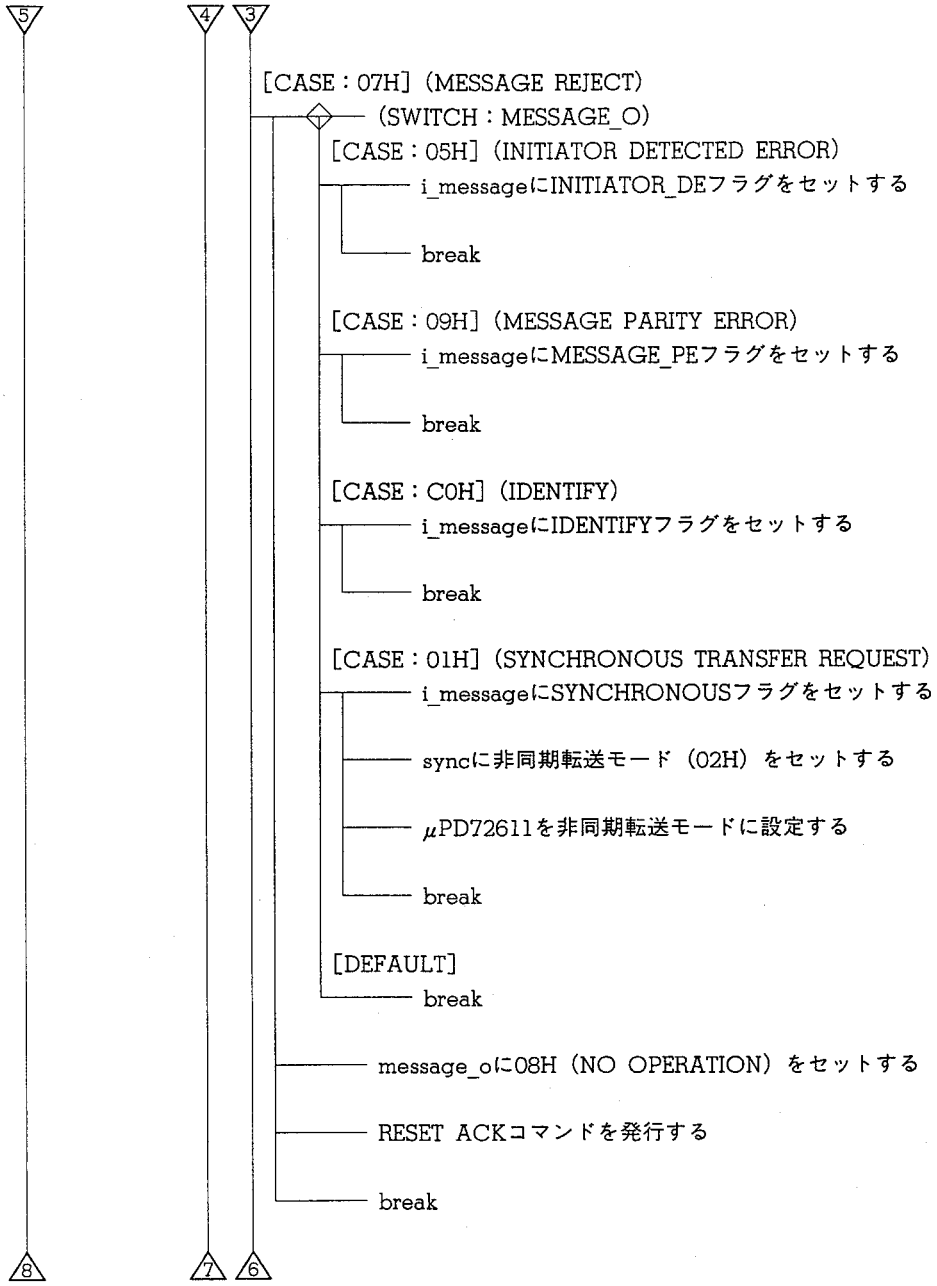
(26) メッセージ受信処理

モジュール名	MESSAGE_R
ファイル名	S_PHASE. CPP
言語	C言語
入力	pmessage_c, message_i, expect_priod [], expect_offset [], message_o t_id, sd_scsil [], sd_scsip [], sc_scsil [], sc_scsip [], ss_scsip [], ad_scsil, ad_scsip, emessage_c
出力	pmessage_c, message_o, sync [], i_message, sd_scsil [], sd_scsip [] ad_scsil, ad_scsip, ac_scsil, ac_scsip, as_scsip, emessage_c
機能	IST割り込み要因のメッセージ受信に対応する処理を実行します。

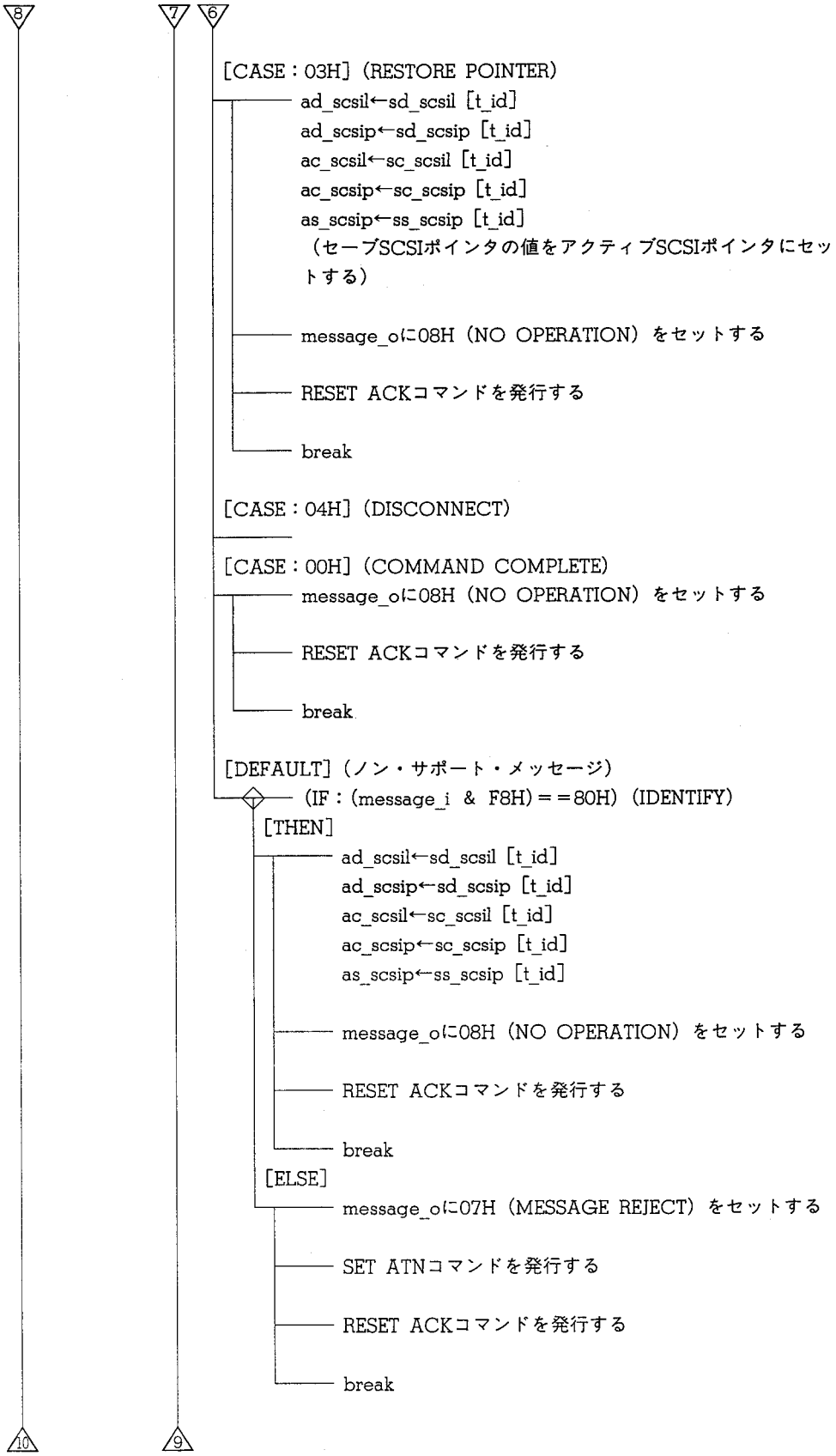
SPDチャート

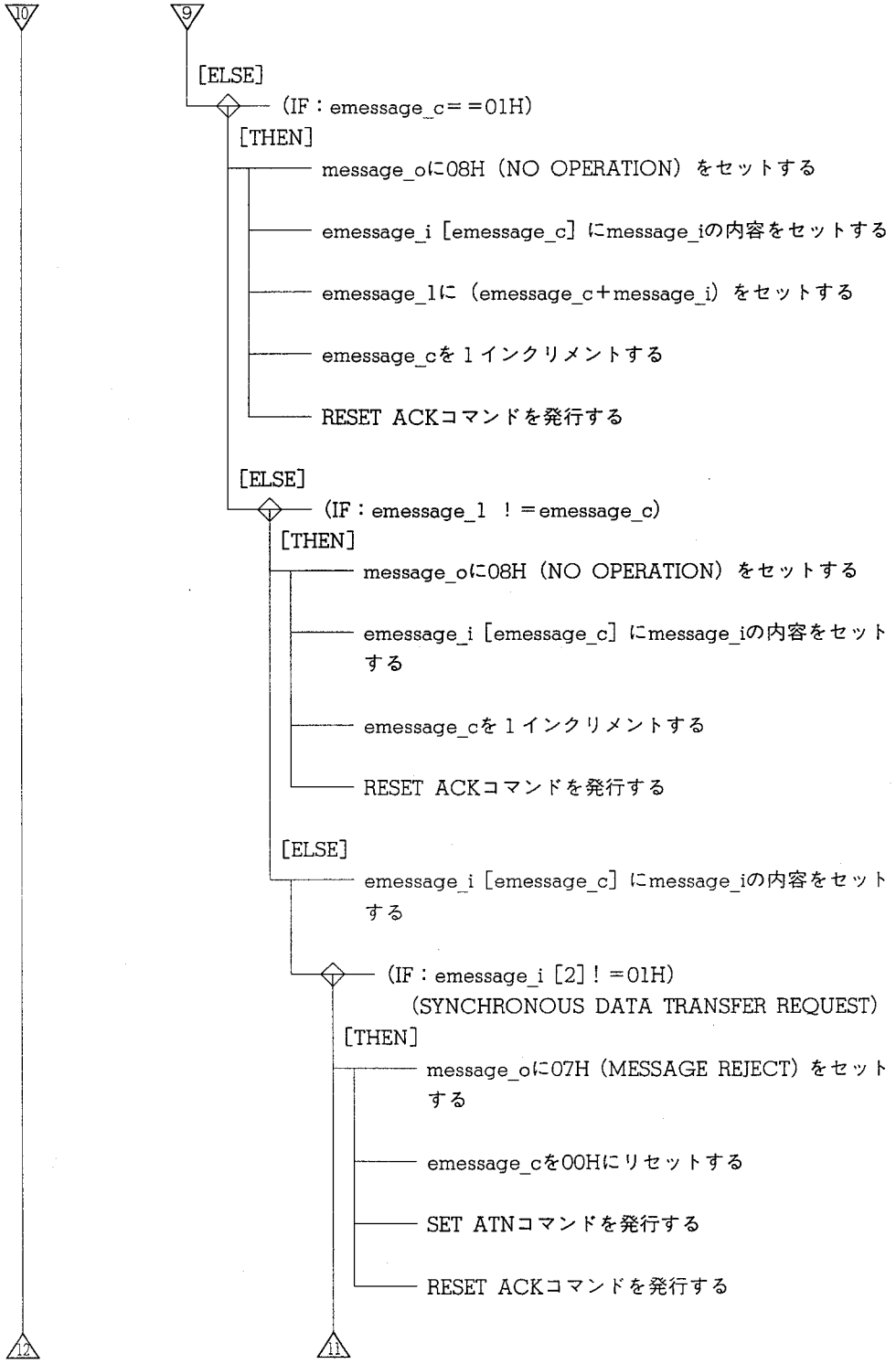



2
1
[ELSE]
messagepe_cを00Hにリセットする
(IF：emessage_c＝＝00H)
[THEN]
(SWITCH：message_i)
[CASE：01H] (EXTENDED MESSAGE)
emessage_i [emessage_c] にmessage_iの内容をセットする
emessage_cをインクリメントする
message_oに80H (NO OPERATION) をセットする
RESET ACKコマンドを発行する
break
[CASE：02H] (SAVE DATA POINTER)
sd_scsil [t_id] ←ad_scsil
sd_scsip [t_id] ←ad_scsip
(アクティブ・データSCSIポインタの値をセーブ・データSCSIポインタにセットする)
message_oに08H (NO OPERATION) をセットする
RESET ACKコマンドを発行する
break
5
4
3

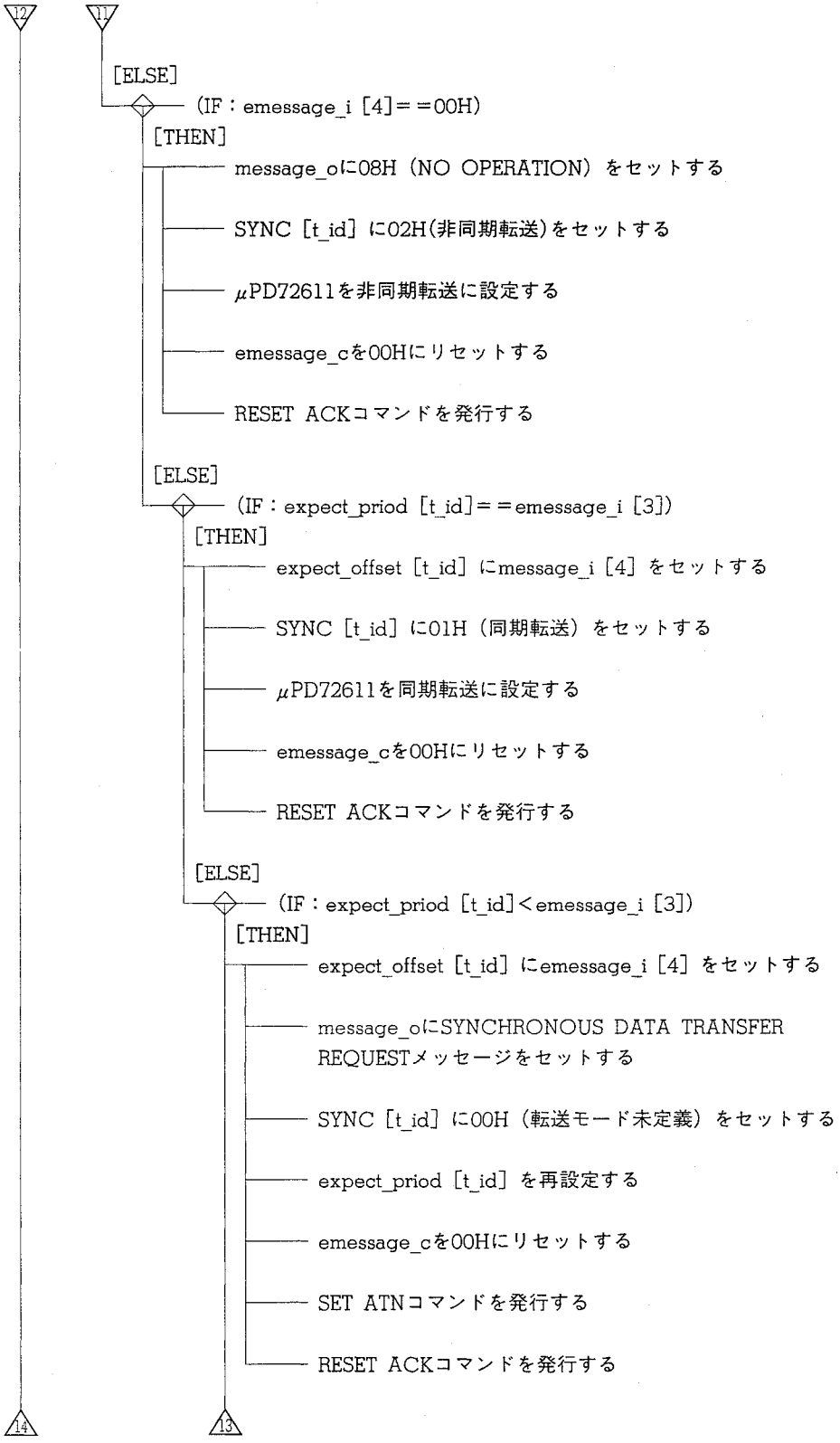

5
4
3
[CASE：07H]（MESSAGE REJECT）
（SWITCH：MESSAGE_O）
[CASE：05H]（INITIATOR DETECTED ERROR）
i_messageにINITIATOR_DEフラグをセットする
break
[CASE：09H]（MESSAGE PARITY ERROR）
i_messageにMESSAGE_PEフラグをセットする
break
[CASE：C0H]（IDENTIFY）
i_messageにIDENTIFYフラグをセットする
break
[CASE：01H]（SYNCHRONOUS TRANSFER REQUEST）
i_messageにSYNCHRONOUSフラグをセットする
syncに非同期転送モード（02H）をセットする
μPD72611を非同期転送モードに設定する
break
[DEFAULT]
break
message_oに08H（NO OPERATION）をセットする
RESET ACKコマンドを発行する
break
8
7
6

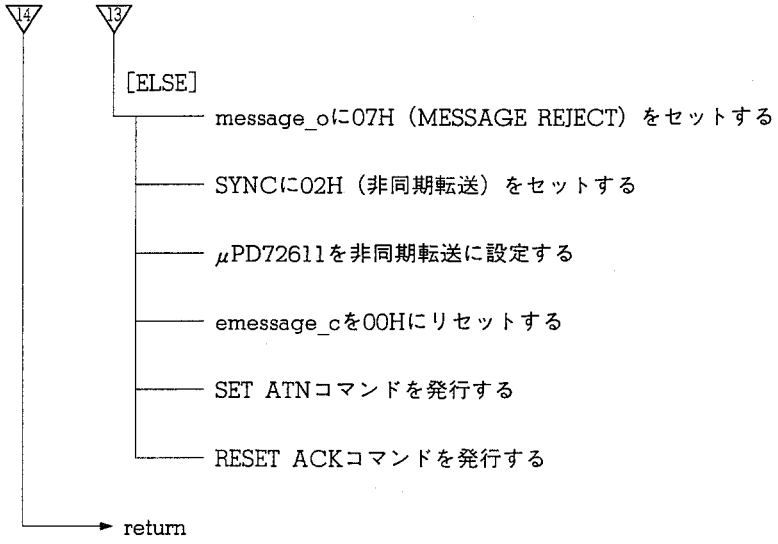
保守／廃止


8
7
6
[CASE：03H]（RESTORE POINTER）
ad_scsil←sd_scsil［t_id］
ad_scsip←sd_scsip［t_id］
ac_scsil←sc_scsil［t_id］
ac_scsip←sc_scsip［t_id］
as_scsip←ss_scsip［t_id］
（セーブSCSIポインタの値をアクティブSCSIポインタにセットする）
message_oに08H（NO OPERATION）をセットする
RESET ACKコマンドを発行する
break
[CASE：04H]（DISCONNECT）
[CASE：00H]（COMMAND COMPLETE）
message_oに08H（NO OPERATION）をセットする
RESET ACKコマンドを発行する
break
[DEFAULT]（ノン・サポート・メッセージ）
（IF：(message_i ＆ F8H)＝＝80H）（IDENTIFY）
[THEN]
ad_scsil←sd_scsil［t_id］
ad_scsip←sd_scsip［t_id］
ac_scsil←sc_scsil［t_id］
ac_scsip←sc_scsip［t_id］
as_scsip←ss_scsip［t_id］
message_oに08H（NO OPERATION）をセットする
RESET ACKコマンドを発行する
break
[ELSE]
message_oに07H（MESSAGE REJECT）をセットする
SET ATNコマンドを発行する
RESET ACKコマンドを発行する
break
10
9


10
9
[ELSE]
(IF：emessage_c＝＝01H)
[THEN]
message_oに08H（NO OPERATION）をセットする
emessage_i [emessage_c] にmessage_iの内容をセットする
emessage_1に（emessage_c＋message_i）をセットする
emessage_cを１インクリメントする
RESET ACKコマンドを発行する
[ELSE]
(IF：emessage_1 ！＝emessage_c)
[THEN]
message_oに08H（NO OPERATION）をセットする
emessage_i [emessage_c] にmessage_iの内容をセットする
emessage_cを１インクリメントする
RESET ACKコマンドを発行する
[ELSE]
emessage_i [emessage_c] にmessage_iの内容をセットする
(IF：emessage_i [2]！＝01H)
(SYNCHRONOUS DATA TRANSFER REQUEST)
[THEN]
message_oに07H（MESSAGE REJECT）をセットする
emessage_cを00Hにリセットする
SET ATNコマンドを発行する
RESET ACKコマンドを発行する
12
11

保守／廃止

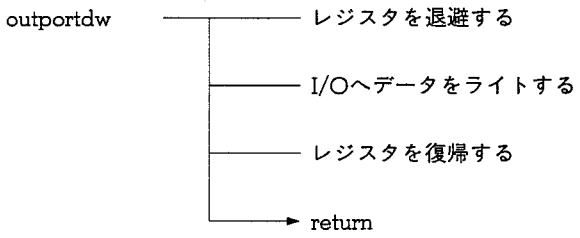

12
11
[ELSE]
(IF：emessage_i [4]＝＝00H)
[THEN]
message_oに08H（NO OPERATION）をセットする
SYNC [t_id] に02H(非同期転送)をセットする
μPD72611を非同期転送に設定する
emessage_cを00Hにリセットする
RESET ACKコマンドを発行する
[ELSE]
(IF：expect_priod [t_id]＝＝emessage_i [3])
[THEN]
expect_offset [t_id] にmessage_i [4] をセットする
SYNC [t_id] に01H（同期転送）をセットする
μPD72611を同期転送に設定する
emessage_cを00Hにリセットする
RESET ACKコマンドを発行する
[ELSE]
(IF：expect_priod [t_id]＜emessage_i [3])
[THEN]
expect_offset [t_id] にemessage_i [4] をセットする
message_oにSYNCHRONOUS DATA TRANSFER REQUESTメッセージをセットする
SYNC [t_id] に00H（転送モード未定義）をセットする
expect_priod [t_id] を再設定する
emessage_cを00Hにリセットする
SET ATNコマンドを発行する
RESET ACKコマンドを発行する
14
13



(27) 32ビット・アウトプット処理

モジュール名	outportdw
ファイル名	IO_PORT.ASM
言語	アセンブラ
入力	w_data, w_port
出力	なし
機能	32ビットのI/Oへデータをライトします。

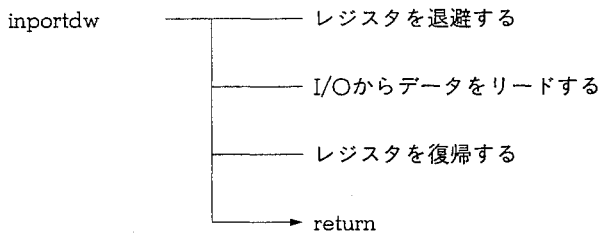
SPDチャート



(28) 32ビット・インプット処理

モジュール名	inportdw
ファイル名	IO_PORT.ASM
言語	アセンブラ
入力	r_port
出力	r_data
機能	32ビットのI/Oからのデータをリードします。

SPDチャート



保守／廃止

第11章　コンパイル方法

この章では，本書で示すプログラムのコンパイル，アセンブル，リンク方法を説明します。
表11－1にプログラムのファイル構成を示します。なお，コンパイラ，アセンブラ，リンカは表11－2に示す製品を使用しました。

表11－1　ファイル構成

プログラム名	ファイル名
フォーマット・コマンド (FORMATS.EXE)	SFORMAT.CPP, SCOM.CPP[注]
SCSI-HDデバイス・ドライバ (SCSIDD.SYS)	SDSUP.ASM, SD_REQ.CPP, SCOM.CPP[注]
ASPIマネージャ (ASPIDD.SYS)	ASSUP.ASM, AS_REQ.CPP, INITIAL.CPP, ASPI.CPP, S_PHASE.CPP IO_PORT.ASM

注　フォーマット・コマンドとSCSI-HDデバイス・ドライバは共にSCOM.CPPとリンクします。

表11－2　ツール

ツール	製　品　名
コンパイラ，リンカ	BORLAND C++3.0　DOS/V用
アセンブラ	TURBO ASSEMBLER 3.0　DOS/V用

①　フォーマット・コマンドのコンパイル方法

1．bcc -c -B -mt sformat
2．bcc -c -B -mt scom
3．tlink /c c0t sformat scom, formats, , cs.lib

②　SCSI-HDデバイス・ドライバのコンパイル方法

1．tasm /m1 sdsup
2．bcc -c -B -mt sd_req
3．bcc -c -B -mt scom
4．tlink /c sdsup sd_req scom, scsidd, , cs.lib
　exeファイルが作成されます。これをsysファイルに変換します。
5．exe2bin scsidd.exe scsidd.sys

③ ASPIマネージャのコンパイル方法

1. tasm /ml assup
2. tasm /ml io_port
3. bcc -S -mt as_req

   asmファイルが作成されます。このasmファイルの図11－1で示す箇所をエディタによって修正します。

図11－1　修正箇所1

```
;           *packet->up.rw.transrw = (unsigned long)ASPI;    /*ASPIのｴﾝﾄﾘ･ﾎﾟｲﾝﾄをｾｯﾄ*/
;
    les     bx,dword ptr DGROUP:_packet
    les     bx,dword ptr es:[bx+14]
    mov     word ptr es:[bx+2],seg @ASPI$qv
    mov     word ptr es:[bx],offset @ASPI$qv
;                                                  このように修正します

    mov     word ptr es:[bx+2],cs
```

4. tasm /ml as_req
5. bcc -S -mt initial

   asmファイルが作成されます。このasmファイルの図11－2で示す箇所をエディタによって修正します。

図11－2　修正箇所2

```
;
;       u_v.vec = (unsigned long)H_INT;
;
    mov     word ptr [bp-4],seg @H_INT$qv
    mov     word ptr [bp-6],offset @H_INT$qv
;                                              このように修正します

    mov     word ptr [bp-4],cs
```

6. tasm /ml initial
7. bcc -c -B -mt！ aspi
8. bcc -c -mt！ s_phase

9. tlink /3 /c assup as_req initial aspi s_phase io_port, aspidd, , cs. lib
   exeファイルが作成されます。これをsysファイルに変換します。

10. exe2bin aspidd. exe aspidd. sys

なお，図11－1，図11－2のように修正しないとsysファイルに変換できません。

保守／廃止

第12章　CONFIG.SYSへの登録方法

この章ではaspidd.sys（ASPIマネージャ），scsidd.sys（SCSI-HDデバイス・ドライバ）の登録方法を示します。図12－1に登録例を示します。
aspidd.sysのパラメータについて表12－1に示します。

図12－1　CONFIG.SYSの登録

```
DEVICE = aspidd.sys /C=3 /I=7 /P=A /A=1000
DEVICE = scsidd.sys
```

なお，aspi.sysは必ずscsidd.sysより先に登録してください。

表12－1　aspi.sysパラメータ

パラメータ	内　　容
C	DMAコントローラのチャネル番号を示します。0，3，5，7の中から選択してください。
I	イニシエータのSCSI ID番号を示します。0～7の中から選択してください。
P	割り込みコントローラの割り込みレベルを示します。3，4，5，7，A，B，C，F（16進数表記）の中から選択してください。
A	μPD72611のI/Oアドレスの先頭アドレスを示します。(16進数表記)

保守／廃止

付録A　リ　ス　ト

プログラムの全リストを示します。

保守／廃止

```
/*************************************************************************/
/*  フォーマット・コマンド     (SFORMAT.CPP)                              */
/*************************************************************************/


#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

extern unsigned char scsi_id;                       /*ﾀｰｹﾞｯﾄのID番号*/
extern unsigned char d_lun;                         /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰ*/
extern unsigned long s_block_n;                     /*開始ﾌﾞﾛｯｸ番号*/
extern unsigned char far *t_address;                /*転送ｱﾄﾞﾚｽ*/
extern unsigned int n_t_block;                      /*転送ﾌﾞﾛｯｸ数*/
extern unsigned char retry;                         /*ﾘﾄﾗｲ回数*/
extern unsigned char return_s;                      /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ*/
extern unsigned char msed_b[20];                    /*ﾓｰﾄﾞ･ｾﾝｽ･ﾃﾞｰﾀ*/
extern unsigned char rcd_b[8];                      /*ﾘｰﾄﾞ･ｷｬﾊﾟｼﾃｨ･ﾃﾞｰﾀ*/
extern unsigned char t_exist;                       /*ﾀｰｹﾞｯﾄが存在しているID番号を示す*/


extern void WRITE_DATA(void);                       /*ライト・データ処理*/
extern void VERIFY(void);                           /*ベリファイ処理*/
extern void RESERVE(void);                          /*リザーブ処理*/
extern void RELEASE(void);                          /*リリース処理*/
extern void MODE_SELECT(void);                      /*モード・セレクト処理*/
extern void READ_CAPACITY(void);                    /*リード・キャパシティ処理*/
extern void FORMAT(void);                           /*フォーマット処理*/
extern void T_INITIAL(void);                        /*ターゲットの初期化処理*/

unsigned long aspi_entry;                           /*ASPIｴﾝﾄﾘ･ﾎﾟｲﾝﾄを格納する変数*/
                                                    /*ASPIﾏﾈｰｼﾞｬのﾃﾞﾊﾞｲｽ名*/
unsigned char aspiname[10] ={'S','C','S','I','M','G','R','$','¥x0'};

void D_ERROR(void);                                 /*エラー表示処理*/

struct REGPACK reg;

/*************************************************************************/
/*                              メイン・ルーチン                         */
/*                                                                       */
/*          入力：なし                                                   */
/*          出力：なし                                                   */
/*************************************************************************/


void main(void)
    {
    unsigned char buff;
    unsigned char drive_n[8];                       /*ﾄﾞﾗｲﾌﾞ番号に対応するSCSI ID番号をｾｯﾄする*/
    unsigned char n;                                /*接続されているSCSI ﾄﾞﾗｲﾌﾞ数*/
    unsigned char id;                               /*SCSI ID番号*/
    unsigned char f;
    unsigned char c;                                /*ﾙｰﾌﾟ･ｶｳﾝﾀ*/
    unsigned int key_in;                            /*キー入力されたコードを格納する変数*/
             char d_n;
```

```
union bpb_un                                    /*BPBの共用体*/
    {
    unsigned char bpb_b[25];
    struct bpb_st                               /*BPBﾌｫｰﾏｯﾄ*/
        {
        unsigned int b_block;                   /*ﾊﾞｲﾄ/ﾌﾞﾛｯｸ*/
        unsigned char block_c;                  /*ﾌﾞﾛｯｸ/ｸﾗｽﾀ*/
        unsigned int resv_block;                /*予約ﾌﾞﾛｯｸ*/
        unsigned char fat_n;                    /*ﾌｧｯﾄ数*/
        unsigned int dir_n;                     /*ﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘ数*/
        unsigned int tol_block;                 /*総ﾌﾞﾛｯｸ数*/
        unsigned char med_id;                   /*ﾒﾃﾞｨｱ･ﾃﾞｨｽｸﾘﾌﾟﾀ*/
        unsigned int block_fat;                 /*ﾌﾞﾛｯｸ/ﾌｧｯﾄ*/
        unsigned int block_t;                   /*ﾌﾞﾛｯｸ/ﾄﾗｯｸ*/
        unsigned int head_n;                    /*ﾍｯﾄﾞ数*/
        unsigned long hidden_block;             /*隠れたﾌﾞﾛｯｸ数*/
        unsigned long etol_block;               /*拡張総ﾌﾞﾛｯｸ数*/
        } bpb_h;
    }   u_bpb;

unsigned char reser_b[512];                     /*予約ﾌﾞﾛｯｸ･ﾃﾞｰﾀ*/
unsigned int c1;                                /*ｶｳﾝﾀ1*/
unsigned int c2;                                /*ｶｳﾝﾀ2*/
unsigned long c3;                               /*ｶｳﾝﾀ3*/
unsigned long c4;                               /*ｶｳﾝﾀ4*/
unsigned char fat1[512];                        /*FATの先頭の1ﾌﾞﾛｯｸ目*/
unsigned char fat2[2048];                       /*FATの4ﾌﾞﾛｯｸ*/
unsigned char root[2048];                       /*ﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘの4ﾌﾞﾛｯｸ*/
unsigned char hidden[512];                      /*隠れｾｸﾀ*/

asm mov ax,3d00h;                               /*ASPIﾏﾈｰｼﾞｬ･ｵｰﾌﾟﾝ*/
asm lea dx,aspiname;
asm int 21h;

asm mov bx,ax;                                  /*ASPIｴﾝﾄﾘ･ﾎﾟｲﾝﾄを格納する*/
asm push bx;
asm mov ax,4402h;
asm lea dx,aspi_entry;
asm mov cx,4;
asm int 21h;

asm pop bx;                                     /*ASPIﾏﾈｰｼﾞｬ･ｸﾛｰｽﾞ*/
asm mov ah,03eh;
asm int 21h;

printf("¥x1b[2J");
printf("¥x1b[2;8Hμ ＰＤ７２６１１フォーマット・コマンド");

d_lun =0x00;
T_INITIAL();                                    /*ﾀｰｹﾞｯﾄの初期化処理*/

printf("¥x1b[m");
printf("¥x1b[2J");
printf("¥x1b[2;8Hμ ＰＤ７２６１１フォーマット・コマンド");
printf("¥x1b[4;8HフォーマットするターゲットのＩＤ番号を選択して下さい");

buff = t_exist;                                 /*接続されているﾀｰｹﾞｯﾄのID番号を調べる*/
n = 0x00;
id = 0x00;
f = 0x01;
while(id < 8)                                   /*接続されているﾀｰｹﾞｯﾄのID番号をdrive_n[ ]へｾｯﾄする*/
    {
    if((buff&f) != 0x00)
        {
        drive_n[n] = id;
        ++n;
        }
    ++id;
    f = f<<1;
    }
```

```
if(n == 0)
    {
    printf("¥x1b[6;20Hターゲットがありません　!");
    exit(0);
    }

c = 0;                                          /*接続されているﾀｰｹﾞｯﾄを表示*/
while(c < n)
    {
    printf("¥x1b[%d;20H",(6+(c*2)));
    printf("SCSI ID = %c",(0x30 + drive_n[c]));
    ++c;
    }

printf("¥x1b[24;10H↑，↓キーで選択，リターンで決定");

d_n = 0x00;                                     /*キー入力によってﾀｰｹﾞｯﾄを選択する*/
printf("¥x1b[7m");
printf("¥x1b[%d;20H",(6+(d_n*2)));
printf("SCSI ID = %c",(0x30 + drive_n[d_n]));

do
    {
    reg.r_ax = 0x0000;                          /*入力ｷｰ･ｺｰﾄﾞを獲得*/
    intr(0x16,&reg);
    key_in = reg.r_ax;
    switch(key_in)
        {
        case 0x4800:                            /*入力ｷｰ･ｺｰﾄﾞが↑0x4800の場合*/
            printf("¥x1b[m");
            printf("¥x1b[%d;20H",(6+(d_n*2)));
            printf("SCSI ID = %c",(0x30 + drive_n[d_n]));
            --d_n;
            if(d_n < 0)
                d_n = n - 1;
            break;
        case 0x5000:                            /*入力ｷｰ･ｺｰﾄﾞが↓0x5000の場合*/
            printf("¥x1b[m");
            printf("¥x1b[%d;20H",(6+(d_n*2)));
            printf("SCSI ID = %c",(0x30 + drive_n[d_n]));
            ++d_n;
            if(d_n == n)
                d_n = 0;
            break;
        default :
            break;
        }
    printf("¥x1b[7m");
    printf("¥x1b[%d;20H",(6+(d_n*2)));
    printf("SCSI ID = %c",(0x30 + drive_n[d_n]));
    }
while(key_in != 0x1c0d);

printf("¥x1b[m");
printf("¥x1b[2J");
printf("¥x1b[2;8Hμ７２６１１フォーマット・コマンド");
printf("¥x1b[4;20H 選択SCSI ID = %c",(0x30 + drive_n[d_n]));
printf("¥x1b[21;50H フォーマット中　… ");
printf("¥x1b[m");

scsi_id = drive_n[d_n];                         /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                                   /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰ･ｾｯﾄ*/
retry = 3;                                      /*ﾘﾄﾗｲ回数をｾｯﾄ*/
RESERVE();                                      /*ﾀｰｹﾞｯﾄをﾘｻﾞｰﾌﾞ*/
if(return_s != 0x00)                            /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
```

```
scsi_id = drive_n[d_n];                    /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                              /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
retry = 3;                                 /*ﾘﾄﾗｲ回数をｾｯﾄ*/
MODE_SELECT();                             /*ﾓｰﾄﾞ･ｾﾚｸﾄ*/
if(return_s != 0x00)                       /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
scsi_id = drive_n[d_n];                    /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                              /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
retry = 3;                                 /*ﾘﾄﾗｲ回数をｾｯﾄ*/
FORMAT();                                  /*ﾌｫｰﾏｯﾄ実行*/
if(return_s != 0x00)                       /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
scsi_id = drive_n[d_n];                    /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                              /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
retry = 3;                                 /*ﾘﾄﾗｲ回数をｾｯﾄ*/
READ_CAPACITY();                           /*ﾀｰｹﾞｯﾄの全容量をﾘｰﾄﾞ*/
if(return_s != 0x00)                       /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }

u_bpb.bpb_h.b_block = 512;                 /*BPBをｾｯﾄする*/
u_bpb.bpb_h.block_c = 0x00;
u_bpb.bpb_h.resv_block = 1;
u_bpb.bpb_h.fat_n = 2;
u_bpb.bpb_h.dir_n = 512;
u_bpb.bpb_h.tol_block = 0;
u_bpb.bpb_h.med_id = 0xF8;
u_bpb.bpb_h.block_fat = 0;
u_bpb.bpb_h.block_t = 0x0020;
u_bpb.bpb_h.head_n = 0x0008;
u_bpb.bpb_h.hidden_block = 0x20;
u_bpb.bpb_h.etol_block = 0x00L;


u_bpb.bpb_b[24] = rcd_b[0];                /*ﾘｰﾄﾞ･ｷｬﾊﾟｼﾃｨ･ﾃﾞｰﾀから拡張総ﾌﾞﾛｯｸ数をｾｯﾄ*/
u_bpb.bpb_b[23] = rcd_b[1];
u_bpb.bpb_b[22] = rcd_b[2];
u_bpb.bpb_b[21] = rcd_b[3];
++u_bpb.bpb_h.etol_block;
u_bpb.bpb_h.etol_block = u_bpb.bpb_h.etol_block - 0x20;

                                           /*ﾌﾞﾛｯｸ/ｸﾗｽﾀをetol_blockから求める*/
u_bpb.bpb_h.block_c = (unsigned int)(u_bpb.bpb_h.etol_block/0xFFEE);
                                           /*小数点以下切り上げ*/
if((u_bpb.bpb_h.etol_block%0xFFEE) > 0)
    ++u_bpb.bpb_h.block_c;

c = 0x80;                                  /*ﾌﾞﾛｯｸ/ｸﾗｽﾀを２の累乗にする*/
while(c >= u_bpb.bpb_h.block_c)
    {
    c = c >> 1;
    }
u_bpb.bpb_h.block_c = c << 1;
                                           /*ﾌﾞﾛｯｸ/ﾌｧｯﾄ数を求める*/
u_bpb.bpb_h.block_fat = (unsigned int)((u_bpb.bpb_h.etol_block-33)/(2+(256*u_bpb.bpb_h.block_c)));
                                           /*小数点以下切り上げ*/
if(((u_bpb.bpb_h.etol_block-33)%(2+(256*u_bpb.bpb_h.block_c))) > 0)
    ++u_bpb.bpb_h.block_fat;
```

保守／廃止

```
reser_b[0] = 0x00;                                  /*予約ブロックを生成*/
reser_b[1] = 0x00;
reser_b[2] = 0x00;
reser_b[3] = 'U';
reser_b[4] = 'P';
reser_b[5] = 'D';
reser_b[6] = '7';
reser_b[7] = '2';
reser_b[8] = '6';
reser_b[9] = '1';
reser_b[10] = '1';
c1 = 11;
c2 = 0;
while (c2 < 25)
    {
    reser_b[c1] = u_bpb.bpb_b[c2];
    ++c1;
    ++c2;
    }
while(c1 < 512)
    {
    reser_b[c1] = 0x00;
    ++c1;
    }
                                                    /*BPBの内容を表示*/
printf("¥x1b[5;10Hバイト数／ブロック    =%10d",u_bpb.bpb_h.b_block);
printf("¥x1b[7;10Hブロック数／クラスタ  =%10d",u_bpb.bpb_h.block_c);
printf("¥x1b[9;10H予約ブロック数        =%10d",u_bpb.bpb_h.resv_block);
buff = 0x30 + u_bpb.bpb_h.fat_n;
printf("¥x1b[11;10HＦＡＴ数             =%10c",buff);
printf("¥x1b[13;10Hルート・ディレクトリ数=%10d",u_bpb.bpb_h.dir_n);
printf("¥x1b[15;10Hブロック数／ＦＡＴ   =%10d",u_bpb.bpb_h.block_fat);
printf("¥x1b[17;10H総ブロック数         =%10ld",u_bpb.bpb_h.etol_block);
printf("¥x1b[19;10H総バイト数           =%10ld",(u_bpb.bpb_h.etol_block*512));


c1 = 0;                                             /*隠れブロック･データをｾｯﾄ*/
while(c1 < 512)
    {
    hidden[c1] = 0x00;
    ++c1;
    }
scsi_id = drive_n[d_n];                             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
s_block_n = 0x00L;                                  /*隠れブロック"00000000H"をｾｯﾄ*/
t_address = hidden;                                 /*隠れブロック･ﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
n_t_block = 0x0001;                                 /*転送ブロック数に1ブロックをｾｯﾄ*/
retry = 3;                                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
WRITE_DATA();                                       /*ﾀｰｹﾞｯﾄに隠れブロックをﾗｲﾄ*/
if(return_s != 0x00)                                /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
                                                    /*予約ブロック領域をライト*/
scsi_id = drive_n[d_n];                             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
s_block_n = 0x20L;                                  /*予約ブロック"00000020H"をｾｯﾄ*/
t_address = reser_b;                                /*予約ブロック･ﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
n_t_block = 0x0001;                                 /*転送ブロック数に1ブロックをｾｯﾄ*/
retry = 3;                                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
WRITE_DATA();                                       /*ﾀｰｹﾞｯﾄに予約ブロックをﾗｲﾄ*/
if(return_s != 0x00)                                /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
```

```
fat1[0] = 0xF8;                         /*FATﾌﾞﾛｯｸのﾃﾞｰﾀを生成*/
fat1[1] = 0xFF;
fat1[2] = 0xFF;
fat1[3] = 0xFF;
c1 = 4;
while(c1 < 512)
    {
    fat1[c1] = 0x00;
    ++c1;
    }
c1 = 0;
while(c1 < 2048)
    {
    fat2[c1] = 0x00;
    ++c1;
    }
                                        /*１つ目のＦＡＴ領域をライト*/
scsi_id = drive_n[d_n];                 /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                           /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
s_block_n = 0x21L;                      /*FATﾌﾞﾛｯｸをｾｯﾄ*/
t_address = fat1;                       /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
n_t_block = 0x0001;                     /*転送ﾌﾞﾛｯｸ数に1ﾌﾞﾛｯｸをｾｯﾄ*/
retry = 3;                              /*ﾘﾄﾗｲ回数をｾｯﾄ*/
WRITE_DATA();                           /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
c2 = (u_bpb.bpb_h.block_fat-1)/4;
c1 = 0x0022;
while(c2 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = (unsigned long)c1;      /*FATﾌﾞﾛｯｸをｾｯﾄ*/
    t_address = fat2;                   /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
    n_t_block = 0x0004;                 /*転送ﾌﾞﾛｯｸ数に4ﾌﾞﾛｯｸをｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    WRITE_DATA();                       /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        break;
    --c2;
    c1 = c1+4;
    }
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
c2 = (u_bpb.bpb_h.block_fat-1)%4;
if(c2 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = (unsigned long)c1;      /*FATﾌﾞﾛｯｸをｾｯﾄ*/
    t_address = fat2;                   /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
    n_t_block = c2;                     /*転送ﾌﾞﾛｯｸ数に残りのﾌﾞﾛｯｸ数をｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    WRITE_DATA();                       /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        {
        D_ERROR();
        exit(0);
        }
    }
```

```
                                        /*２つ目のＦＡＴ領域をライト*/
c1 = u_bpb.bpb_h.block_fat+0x21;
scsi_id = drive_n[d_n];                 /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                           /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
s_block_n = (unsigned long)c1;          /*FATﾌﾞﾛｯｸをｾｯﾄ*/
t_address = fat1;                       /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
n_t_block = 0x0001;                     /*転送ﾌﾞﾛｯｸ数に1ﾌﾞﾛｯｸをｾｯﾄ*/
retry = 3;                              /*ﾘﾄﾗｲ回数をｾｯﾄ*/
WRITE_DATA();                           /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }

c2 = (u_bpb.bpb_h.block_fat-1)/4;
c1 = c1+1;
while(c2 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = (unsigned long)c1;      /*FATﾌﾞﾛｯｸをｾｯﾄ*/
    t_address = fat2;                   /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
    n_t_block = 0x0004;                 /*転送ﾌﾞﾛｯｸ数に4ﾌﾞﾛｯｸをｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    WRITE_DATA();                       /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        break;
    --c2;
    c1 = c1+4;
    }
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
c2 = (u_bpb.bpb_h.block_fat-1)%4;
if(c2 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = (unsigned long)c1;      /*FATﾌﾞﾛｯｸをｾｯﾄ*/
    t_address = fat2;                   /*FATﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
    n_t_block = c2;                     /*転送ﾌﾞﾛｯｸ数に残りのﾌﾞﾛｯｸ数をｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    WRITE_DATA();                       /*ﾀｰｹﾞｯﾄにFATをﾗｲﾄ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        {
        D_ERROR();
        exit(0);
        }
    }
c1 = 0;                                 /*ﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘ･ﾃﾞｰﾀをｾｯﾄ*/
while(c1 < 2048)
    {
    root[c1] = 0xE5;
    ++c1;
    }
c1 = 0;
while(c1 < 2048)
    {
    root[c1] = 0x00;
    c1 = c1+32;
    }
```

```
                                        /*ルート・ディレクトリ領域をライト*/
c1 = (u_bpb.bpb_h.block_fat*2)+0x21;
c2 = 0;
while(c2 < 32)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = (unsigned long)c1;      /*ﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘ･ﾌﾞﾛｯｸをｾｯﾄ*/
    t_address = root;                   /*ﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘ･ﾃﾞｰﾀの先頭ｱﾄﾞﾚｽをｾｯﾄ*/
    n_t_block = 0x0004;                 /*転送ﾌﾞﾛｯｸ数に4ﾌﾞﾛｯｸをｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    WRITE_DATA();                       /*ﾀｰｹﾞｯﾄにﾙｰﾄ･ﾃﾞｨﾚｸﾄﾘをﾗｲﾄ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        break;
    c2 = c2+4;
    c1 = c1+4;
    }
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }

c3 = u_bpb.bpb_h.etol_block/0xFFFF;
c4 = 0;
while(c3 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = c4;                     /*先頭ﾌﾞﾛｯｸ"をｾｯﾄ*/
    n_t_block = 0xFFFF;                 /*ﾍﾞﾘﾌｧｲ･ﾌﾞﾛｯｸ数をｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    VERIFY();                           /*ﾀｰｹﾞｯﾄの全容量をﾘｰﾄﾞ*/
    if(return_s != 0x00)                /*ｴﾗｰ発生*/
        break;
    --c3;
    c4 = c4 + 0xFFFF;
    }
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
c3 = u_bpb.bpb_h.etol_block%0xFFFF;
if(c3 > 0)
    {
    scsi_id = drive_n[d_n];             /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
    d_lun = 0x00;                       /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
    s_block_n = c4;                     /*先頭ﾌﾞﾛｯｸ"をｾｯﾄ*/
    n_t_block = (unsigned int)c3;       /*ﾍﾞﾘﾌｧｲ･ﾌﾞﾛｯｸ数をｾｯﾄ*/
    retry = 3;                          /*ﾘﾄﾗｲ回数をｾｯﾄ*/
    VERIFY();                           /*ﾀｰｹﾞｯﾄの全容量をﾘｰﾄﾞ*/
    }
if(return_s != 0x00)                    /*ｴﾗｰ発生*/
    {
    D_ERROR();
    exit(0);
    }
scsi_id = drive_n[d_n];                 /*ﾀｰｹﾞｯﾄIDをｾｯﾄ*/
d_lun = 0x00;                           /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰｾｯﾄ*/
retry = 3;                              /*ﾘﾄﾗｲ回数をｾｯﾄ*/
RELEASE();                              /*ﾀｰｹﾞｯﾄをﾘﾘｰｽﾞ*/
D_ERROR();
exit(0);
}
```

保守／廃止

```
/*************************************************************************
 *                          エラー表示処理                               *
 *                                                                       *
 *          入力：return_s                                               *
 *          出力：なし                                                   *
 *************************************************************************/

void D_ERROR(void)
    {
    printf("¥x1b[21;0H");
    printf("¥x1b[2K");
        switch(return_s)
        {
        case    0x00:
            printf("¥x1b[21;30Hフォーマット終了");
            break;
        case    0x06:
            printf("¥x1b[21;30Hドライブ準備ができていません");
            break;
        case    0x0B:
        case    0x0C:
        case    0x10:
        case    0x11:
            printf("¥x1b[21;30Hデータ・エラーです");
            break;
        case    0x0F:
            printf("¥x1b[21;30Hシーク・エラーです");
            break;
        case    0x0E:
            printf("¥x1b[21;30Hブロックが見つかりません");
            break;
        case    0x02:
            printf("¥x1b[21;30H書き込み禁止です");
            break;
        default:
            printf("¥x1b[21;30Hエラーです");
            break;

        }
    return;
    }
```

保守／廃止

```
;************************************************************************
;*   SCSI－HDデバイス・ドライバ(ｽﾀｰﾄ･ｱｯﾌﾟ･ﾙｰﾁﾝ) SDSUP.ASM*
;************************************************************************
;
;
_TEXT SEGMENT DWORD PUBLIC USE16 'CODE'
_TEXT ENDS
;
_DATA SEGMENT DWORD PUBLIC USE16 'DATA'
_DATA ENDS
;
_BSS SEGMENT DWORD PUBLIC USE16 'BSS'
_BSS ENDS
;
_PEND SEGMENT DWORD PUBLIC USE16 'PEND'
_PEND ENDS
;
DGROUP GROUP _TEXT,_DATA,_BSS,_PEND
    ASSUME  CS:_TEXT,DS:DGROUP,SS:DGROUP,ES:DGROUP
;
;
_DATA SEGMENT DWORD PUBLIC USE16 'DATA'
;
_packet     dd  ?                       ;コマンド・パケットのポインタ領域
stacksave   dd  ?                       ;スタック・ポインタのセーブ領域
_prog_end   dd  ?                       ;プログラム終了アドレスのセーブ領域
        db  1024*4 dup(?)
stacktop    label   byte                ;スタック先頭アドレス
_DATA ENDS
;
_PEND SEGMENT DWORD PUBLIC USE16 'PEND'
progend     label   word                ;プログラム終了アドレス
_PEND ENDS
;
public  _packet                         ;コマンド・パケットのポインタ領域
public  _prog_end                       ;プログラム終了アドレスのセーブ領域
;
_TEXT SEGMENT DWORD PUBLIC USE16 'CODE'
;
EXTRN   _SD_IO_REQ:PROC                 ;Ｉ／Ｏリクエスト・コマンド処理
;
org 0
;
;************************************************************************
;*                      デバイス・ヘッダの構成                          *
;************************************************************************
;
header  label   word
    dd  -1                              ;次のデバイスへのポインタ
    dw  2002h                           ;デバイスの属性
    dw  STRATEGY                        ;ストラテジ・エントリ・ポイント
    dw  INTERRUPT                       ;割り込みエントリ・ポイント
    db  7                               ;ユニット数
;
;*************************************************************************
;*                              ストラテジ・ルーチン                     *
;*                                                  入力：bx,es          *
;*コマンド・パケットのポインタを格納する。          出力：packet         *
;*************************************************************************
;
STRATEGY    proc    far
;
    mov word ptr cs:[_packet],bx
    mov word ptr cs:[_packet+2],es
    ret
;
STRATEGY    endp
```

A

保守／廃止

```
;*****************************************************************************
;*                          割り込みエントリ・ルーチン                       *
;*                                                    入力：なし             *
;*ＳＣＳＩデバイス・ドライバの入口と出口。この処理の    出力：prog_end         *
;*中でＩ／Ｏリクエスト・コマンド処理をＣＡＬＬする。                          *
;*****************************************************************************
;
INTERRUPT   proc    far
;
    push    ax                          ;レジスタ退避
    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    bp
    push    ds
    push    es
;
    mov ax,cs                           ;ｄｓ　＝　ｃｓ
    mov ds,ax
    mov word ptr [stacksave],sp         ;ＭＳ－ＤＯＳのスタック・ポインタの退避
    mov word ptr [stacksave+2],ss
;
    cli
    mov ss,ax                           ;ＳＣＳＩ－ＨＤデバイス・ドライバのローカル・スタック・ポインタの確保
    mov sp,offset DGROUP:stacktop
    sti
;
                                        ;プログラムの終了アドレスをprog_endにセット
    mov word ptr [_prog_end],offset DGROUP:progend
    mov word ptr [_prog_end+2],cs
    call    _SD_IO_REQ                  ;Ｉ／Ｏリクエスト・コマンド処理


    cli
    mov sp,word ptr [stacksave]         ;ＭＳ－ＤＯＳのスタック・ポインタの復帰
    mov ss,word ptr [stacksave+2]
    sti
;
    pop es                              ;レジスタ復帰
    pop ds
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
;
INTERRUPT   endp

_TEXT ENDS
;
END
```

保守／廃止

```
/*****************************************************************************
 * SCSI－HDデバイス・ドライバ（I／Oリクエスト・コマンド処理）　SD_REQ.CPP *
 *****************************************************************************/

#include <dos.h>

    /*  コマンド・パケット・ステータス情報  */

#define     ST_ERROR        0x8000              /*ERRORビット*/
#define     ST_DONE         0x0100              /*DONEビット*/
#define     ST_ERR_PRO           0              /*ライト・プロテクト*/
#define     ST_ERR_UNIT          1              /*無効なユニット*/
#define     ST_ERR_NOTRDY        2              /*ドライブの準備ができていない*/
#define     ST_ERR_BADCMD        3              /*コマンド番号が間違っている*/
#define     ST_ERR_CRC           4              /*ＣＲＣエラー*/
#define     ST_ERR_SEEK          6              /*シーク・エラー*/
#define     ST_ERR_MEDIA         7              /*無効なメディア*/
#define     ST_ERR_SECTOR        8              /*セクタが見つかりません*/
#define     ST_ERR_WRITE        10              /*書き込みエラー*/
#define     ST_ERR_READ         11              /*読み出しエラー*/
#define     ST_ERR_GENERAL      12              /*その他のエラー*/

    /*  コマンド・パケットの構造体  */

    struct command_packet
        {
        char command_length;                    /*コマンド・パケットのバイト数*/
        char unit_number;                       /*論理装置コード*/
        char cmd;                               /*Ｉ／Ｏリクエスト・コマンド・コード*/
        int  command_status;                    /*ステータス*/
        long resv1,resv2;                       /*予約域　（８バイト）*/

        union
            {
            /*  build  bpbのコマンド・パケット  */

            struct
                {
                unsigned char mediab;           /*ＤＰＢへのﾒﾃﾞｨｱ･ﾃﾞｨｽｸｨﾌﾟﾀ*/
                int  transb_off;                /*転送アドレス*/
                int  transb_seg;
                unsigned char far *bpb_po;      /*ＢＰＢに対するポインタ*/
                } bb;

            /*  リード，ライトのコマンド・パケット  */

            struct
                {
                unsigned char mediarw;          /*ＤＰＢへのﾒﾃﾞｨｱ･ﾃﾞｨｽｸｨﾌﾟﾀ*/
                unsigned char far *transrw;     /*転送アドレス*/
                unsigned int sector_count;      /*セクタ・カウント*/
                unsigned int start_sector;
                unsigned long id_p;             /*ﾎﾞﾘｭｰﾑIDのﾎﾟｲﾝﾀ*/
                unsigned long start_32;         /*開始セクタ番号*/
                } rw;

            /*  メディア・チェックのコマンド・パケット  */

            struct
                {
                char mediamc;                   /*ＤＰＢへのﾒﾃﾞｨｱ･ﾃﾞｨｽｸｨﾌﾟﾀ*/
                int  retpr;                     /*返す値*/
                } mc;

            /*  イニシャライズのコマンド・パケット  */

            struct
                {
                unsigned char unit;             /*ユニット数*/
                unsigned char far *break_add;   /*ブレーク・アドレス*/
                unsigned char far *bpb_di;      /*ＢＰＢの配列ポインタ*/
                } in;
            } up;
        };
```

A

```
extern struct command_packet far *packet;          /*コマンド・パケットのポインタ*/

extern unsigned char far *prog_end;                /*プログラム・エンド値*/
extern unsigned char scsi_id;                      /*ターゲットのID番号*/
extern unsigned char d_lun;                        /*ロジカル・ユニット・ナンバー*/
extern unsigned long s_block_n;                    /*開始ブロック番号*/
extern unsigned char far *t_address;               /*転送アドレス*/
extern unsigned int n_t_block;                     /*転送ブロック数*/
extern unsigned char retry;                        /*リトライ回数*/
extern unsigned char return_s;                     /*リターン・ステータス*/
extern unsigned char msed_b[20];                   /*モード・センス・データ*/
extern unsigned char rcd_b[8];                     /*リード・キャパシティ・データ*/
extern unsigned char inqu_d[45];                   /*インクワイア・データ*/
extern unsigned char t_exist;                      /*ターゲットが存在しているID番号を示す*/

extern void READ_DATA(void);                       /*リード・データ処理*/
extern void WRITE_DATA(void);                      /*ライト・データ処理*/
extern void WRITE_VERIFY(void);                    /*ライト・アンド・ベリファイ処理*/
extern void MODE_SENSE(void);                      /*モード・センス処理*/
extern void INQUIRY(void);                         /*インクワイアリ処理*/
extern void T_U_READY(void);                       /*テスト・ユニット・レディ処理*/
extern void T_INITIAL(void);                       /*ターゲットの初期化処理*/

unsigned char bpb[8][25];                          /*ＢＰＢ*/
unsigned char *bpb_p[8];                           /*BPBポインタ*/
unsigned char unit_n[8];                           /*ユニット番号に対応するID番号を格納する*/
unsigned long aspi_entry;                          /*ASPIエントリ・ポイントを格納する変数*/
                                                   /*ASPIマネージャのデバイス名*/
unsigned char aspiname[10] ={'S','C','S','I','M','G','R','$','¥x0'};


extern "C" void SD_IO_REQ(void);                   /*I/Oリクエスト・コマンド処理*/

void INITIAL(void);                                /*イニシャライズ処理*/
void MEDIA_CHECK(void);                            /*メディア・チェック処理*/
void BUILD_BPB(void);                              /*build bpb処理*/
void INPUT(void);                                  /*インプット処理*/
void OUTPUT(void);                                 /*アウトプット処理*/
void OUTPUT_V(void);                               /*アウトプット・ベリファイ処理*/
void ERROR_IO(void);                               /*IOリクエスト・エラー処理*/
void R_STA_PACK(void);                             /*リード・コマンド・パケット・ステータス・セット処理*/
void W_STA_PACK(void);                             /*ライト・コマンド・パケット・ステータス・セット処理*/
```

保守／廃止

```
/*************************************************************************
 *                    Ｉ／Ｏリクエスト・コマンド処理                     *
 *I/Oﾘｸｴｽﾄ･ｺﾏﾝﾄﾞ･ｺｰﾄﾞに従って処理をＣＡＬＬする。                      *
 *      入力：packet                                                     *
 *      出力：なし                                                       *
 *************************************************************************/


void SD_IO_REQ(void)
    {

    switch(packet->cmd)
        {
        case    0x00:
            INITIAL();                          /*イニシャライズ処理*/
            break;
        case    0x01:
            MEDIA_CHECK();                      /*メディア・チェック処理*/
            break;
        case    0x02:
            BUILD_BPB();                        /*ＢＵＩＬＤ　ＢＰＢ処理*/
            break;
        case    0x04:
            INPUT();                            /*インプット処理*/
            break;
        case    0x08:
            OUTPUT();                           /*アウトプット処理*/
            break;
        case    0x09:
            OUTPUT_V();                         /*アウトプット・ウィズ・ベリファイ処理*/
            break;
        default:
            ERROR_IO();                         /*IOﾘｸｴｽﾄ･ｴﾗｰ処理*/
        }
    return;
    }


/*************************************************************************
 *                        イニシャライズト処理                           *
 *ターゲットの初期化を実行する。                                         *
 *                                                                       *
 *      入力：packet,prog_end                                            *
 *      出力：packet,unit_n[]                                            *
 *************************************************************************/


void INITIAL(void)
    {
    unsigned char c;                            /*ｶｳﾝﾀ*/
    unsigned char n;
    unsigned char l;
    unsigned char rev_d[512];                   /*予約ﾌﾞﾛｯｸ*/

    union                                       /*ﾊﾞｲﾄ/ﾌﾞﾛｯｸをセットする共用体*/
        {
        unsigned long byte_b;
        unsigned char msed[4];
        }   u_l;

    static unsigned int w[30] = {'Ｓ','Ｃ','Ｓ','Ｉ','ー','Ｈ','Ｄ','デ','バ','イ','ス','ド','ラ','イ','バ','を','組','込',
    'み','ま','し','た','¥x0d¥x0a','$'};

    static unsigned int x[30] = {'以','下','の','タ','ー','ゲ','ッ','ト','Ｉ','Ｄ','が','接','続','さ','れ','て','い','ま',
    'す','¥x0d¥x0a','$'};

    static unsigned char y[20] = {'Ｓ','Ｃ','Ｓ','Ｉ',' ',' ','Ｉ','Ｄ','=','¥x30',' ',' ','|',' ',' ','$'};

    static unsigned int z[30] = {'タ','ー','ゲ','ッ','ト','が','接','続','さ','れ','て','い','ま','せ','ん','¥x0d¥x0a','$'};


    _AH = 0X09;                                 /*" ＳＣＳＩーＨＤデバイス・ドライバを組込みました"と表示する*/
    _DX= (int)w;
    geninterrupt(0x21);
```

A

```
asm mov ax,3d00h;                       /*ASPIﾏﾈｰｼﾞｬ･ｵｰﾌﾟﾝ*/
asm lea dx,aspiname;
asm int 21h;

asm mov bx,ax;                          /*ASPIｴﾝﾄﾘ･ﾎﾟｲﾝﾄを格納する*/
asm push bx;
asm mov ax,4402h;
asm lea dx,aspi_entry;
asm mov cx,4;
asm int 21h;

asm pop bx;                             /*ASPIﾏﾈｰｼﾞｬ･ｸﾛｰｽﾞ*/
asm mov ah,03eh;
asm int 21h;

d_lun = 0x00;
T_INITIAL();                            /*ﾀｰｹﾞｯﾄの初期化*/

c = 0;                                  /*接続されているﾀｰｹﾞｯﾄにﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨを発行*/
while(c < 8)
    {
    if((t_exist&(0x01<<c)) == 0x00)     /*接続されているID番号を調べる*/
        ++c;
    else
        {
        scsi_id = c;
        d_lun = 0x00;
        T_U_READY();                    /*ﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨ処理*/
        if(return_s == 0x12)            /*ﾉｯﾄ･ﾚﾃﾞｨ状態か？*/
            {
            scsi_id = c;
            d_lun = 0x00;
            T_U_READY();                /*ﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨ処理*/
            if(return_s != 0x00)        /*正常状態でないか？*/
                                        /*t_existのcﾋﾞｯﾄをｸﾘｱする*/
                t_exist = t_exist & (0xFF^(0x01<<c));
            }
        ++c;
        }
    }

c = 0;                                  /*接続されているﾀｰｹﾞｯﾄにﾓｰﾄﾞ･ｾﾝｽ･ｺﾏﾝﾄﾞを発行*/
while(c < 8)
    {
    if((t_exist&(0x01<<c)) == 0x00)     /*接続されているID番号を調べる*/
        ++c;
    else
        {
        scsi_id = c;
        d_lun = 0x00;
        retry = 3;
        MODE_SENSE();                   /*ﾓｰﾄﾞ･ｾﾝｽ処理*/
            if(return_s == 0x00)
                {
                u_l.msed[0] = msed_b[11];
                u_l.msed[1] = msed_b[10];
                u_l.msed[2] = msed_b[9];
                u_l.msed[3] = msed_b[8];
                if(u_l.byte_b != 512)   /*512ﾊﾞｲﾄ/ﾌﾞﾛｯｸでﾌｫｰﾏｯﾄされているか？*/
                                        /*t_existのcﾋﾞｯﾄをｸﾘｱする*/
                    t_exist = t_exist & (0xFF^(0x01<<c));
                }
            else
                {
                                        /*t_existのcﾋﾞｯﾄをｸﾘｱする*/
                t_exist = t_exist & (0xFF^(0x01<<c));
                }
        ++c;
        }
    }
```

```
c = 0;
l = 0;                                          /*接続されているターゲット数をカウントする*/
while(c < 8)                                    /*接続されているターゲットから予約ﾌﾞﾛｯｸをリードする*/
    {
    if((t_exist&(0x01<<c)) == 0x00)             /*接続されているID番号を調べる*/
        ++c;
    else
        {
        scsi_id = c;
        d_lun = 0x00;
        retry = 3;
        s_block_n = 0x20L;
        n_t_block = 0x0001;
        t_address = rev_d;
        READ_DATA();                            /*リード・データ処理*/
        if(return_s == 0x00)                    /*正常終了か？*/
            {
            n = 0;
            while(n < 25)
                {
                bpb[l][n] = rev_d[11+n];        /*ＢＰＢをセット*/
                ++n;
                }
            bpb_p[l] = &bpb[l][0];              /*ＢＰＢの配列にBPBﾎﾟｲﾝﾀをセット*/
            unit_n[l] = c;                      /*ｺﾏﾝﾄﾞ･ﾊﾟｹｯﾄのﾕﾆｯﾄ･ﾅﾝﾊﾞに対応したID番号をセット*/
            ++l;
            }
        else
            {
                                                /*t_existのcﾋﾞｯﾄをｸﾘｱする*/
            t_exist = t_exist & (0xFF^(0x01<<c));
            }
        ++c;
        }
    }

if(l > 0)                                       /*接続されているターゲットが存在するか？*/
    {
    _AH = 0x09;                                 /*”以下のターゲットが接続されています”を表示する*/
    _DX= (int)x;
    geninterrupt(0x21);

    n = 0 ;
    while(n < l)
        {
        y[8] = unit_n[n] + 0x30;                /*”ＳＣＳＩ　ＩＤ＝（番号）｜”を表示する*/
        _AH = 0x09;
        _DX= (int)y;
        geninterrupt(0x21);
        scsi_id = unit_n[n];
        d_lun = 0x00;
        retry = 0x03;
        INQUIRY();                              /*インクワイアリ処理*/
        inqu_d[42] = 0x0D;
        inqu_d[43] = 0x0A;
        inqu_d[44] = '$';
        _AH = 0x09;                             /*インクワイアリ・データを表示する*/
        _DX= (int)&inqu_d[8];
        geninterrupt(0x21);
        ++n;
        }
    packet->up.in.unit = l;                             /*ユニット数をセット*/
    packet->up.in.break_add = prog_end;                 /*ブレーク・アドレスをセット*/
    packet->up.in.bpb_di = (unsigned char far *)bpb_p;  /*ＢＰＢの配列へのポインタをセット*/
    packet->command_status = ST_DONE;                   /*ＤＯＮＥビットをセット*/
    }
else
    {
    _AH = 0x09;
    _DX= (int)z;
    geninterrupt(0x21);                                 /*”ターゲットが接続されていません”を表示する*/
    packet->up.in.unit = 0;                             /*ユニット数をセット*/
    u_l.byte_b = (unsigned long)prog_end;               /*ブレーク・アドレスをセット*/
    u_l.byte_b = u_l.byte_b & 0xFFFF0000L;
```

```
        prog_end = (unsigned char far *)u_l.byte_b;
        packet->up.in.break_add = prog_end;
        packet->command_status = ST_DONE;                    /*DONEビットをセット*/
        }

    return;
    }


/************************************************************************
 *                        メディア・チェック処理                        *
 *packetに“メディアが変更されていない”ことをセットする。                *
 *                                                                      *
 *      入力：なし                                                      *
 *      出力：packet                                                    *
 ************************************************************************/


void MEDIA_CHECK(void)
    {
        packet->up.mc.retpr = 1;                    /*返す値に１をセット*/
        packet->command_status = ST_DONE;           /*DONEビットをセット*/
        return;
    }


/************************************************************************
 *                        BUILD  BPB処理                                *
 *packetにＢＰＢのポインタをセットする。                                *
 *                                                                      *
 *      入力：なし                                                      *
 *      出力：packet                                                    *
 ************************************************************************/

void BUILD_BPB(void)
    {
                                                /*ＢＰＢに対するポインタをセット*/
    packet->up.bb.bpb_po = &bpb[packet->unit_number][0];
                                                /*DONEビットをセット*/
    packet->command_status = ST_DONE;
    return;
    }
```

```
/*************************************************************************
 *                          インプット処理                               *
 *ターゲットからデータをリードし，packetが指定する領域にセットする。     *
 *                                                                       *
 *      入力：packet,unit_n[]                                            *
 *      出力：packet                                                     *
 *************************************************************************/


void INPUT(void)
    {
    union                                       /*バイト/ブロックをセットする共用体*/
        {
        unsigned long byte_b;
        unsigned char msed[4];
        }   u_l;

    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
    s_block_n = s_block_n + 0x20L;

    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
    t_address = packet->up.rw.transrw;          /*転送アドレスをセット*/
    retry = 3;                                  /*リトライ回数をセット*/
    READ_DATA();                                /*データ・リード処理*/
    if(return_s == 0x12)                        /*ノット・レディ処理状態か？*/
        {
        do
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            T_U_READY();                            /*テスト・ユニット・レディ処理*/
            }
        while(return_s == 0x12);                    /*ノット・レディ状態か？*/

        if(return_s == 0x00)                        /*正常終了か？*/
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            retry = 3;                              /*リトライ回数をセット*/
            MODE_SENSE();                           /*モード・センス処理*/
            if(return_s == 0x00)                    /*正常終了か？*/
                {
                u_l.msed[0] = msed_b[11];
                u_l.msed[1] = msed_b[10];
                u_l.msed[2] = msed_b[9];
                u_l.msed[3] = msed_b[8];
                if(u_l.byte_b == 512)               /*512バイト/ブロックでフォーマットされているか？*/
                    {
                    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
                    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
                    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
                    s_block_n = s_block_n + 0x20L;
                    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
                    t_address = packet->up.rw.transrw;                  /*転送アドレスをセット*/
                    retry = 3;                                  /*リトライ回数をセット*/
                    READ_DATA();                                /*データ・リード処理*/
                    }
                else
                    return_s = 0x12;                            /*ノット・レディをセット*/
                }
            }
        }
    R_STA_PACK();                                   /*リード・コマンド・パケット・ステータス・セット処理*/
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                          アウトプット処理                             *
 *packetが指定する領域のデータをターゲットにライトする。                  *
 *                                                                       *
 *      入力：packet,unit_n[]                                            *
 *      出力：packet                                                     *
 *************************************************************************/


void OUTPUT(void)
    {
    union                                       /*バイト/ブロックをセットする共用体*/
        {
        unsigned long byte_b;
        unsigned char msed[4];
        }   u_l;

    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
    s_block_n = s_block_n + 0x20L;
    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
    t_address = packet->up.rw.transrw;          /*転送アドレスをセット*/
    retry = 3;                                  /*リトライ回数をセット*/
    WRITE_DATA();                               /*データ・ライト処理*/
    if(return_s == 0x12)                        /*ノット・レディ状態か？*/
        {
        do
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            T_U_READY();                            /*テスト・ユニット・レディ処理*/
            }
        while(return_s == 0x12);                    /*ノット・レディ状態か？*/

        if(return_s == 0x00)                        /*正常終了か？*/
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            retry = 3;                              /*リトライ回数をセット*/
            MODE_SENSE();                           /*モード・センス処理*/
            if(return_s == 0x00)                    /*正常終了か？*/
                {
                u_l.msed[0] = msed_b[11];
                u_l.msed[1] = msed_b[10];
                u_l.msed[2] = msed_b[9];
                u_l.msed[3] = msed_b[8];
                if(u_l.byte_b == 512)               /*512バイト/ブロックでフォーマットされているか？*/
                    {
                    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
                    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
                    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
                    s_block_n = s_block_n + 0x20L;
                    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
                    t_address = packet->up.rw.transrw;          /*転送アドレスをセット*/
                    retry = 3;                                  /*リトライ回数をセット*/
                    WRITE_DATA();                               /*データ・ライト処理*/
                    }
                else
                    return_s = 0x12;                            /*ノット・レディをセット*/
                }
            }
        }
    W_STA_PACK();                                               /*ライト・コマンド・パケット・ステータス・セット処理*/
    return;
    }
```

保守／廃止

```
/*********************************************************************
 *                アウトプット・ウィズ・ベリファイ処理                *
 *packetが指定する領域のデータをターゲットにライトし，ベリファイする。 *
 *                                                                   *
 *      入力：packet,unit_n[]                                        *
 *      出力：packet                                                 *
 *********************************************************************/


void OUTPUT_V(void)
    {
    union                                       /*バイト/ブロックをセットする共用体*/
        {
        unsigned long byte_b;
        unsigned char msed[4];
        }   u_l;

    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
    s_block_n = s_block_n + 0x20L;
    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
    t_address = packet->up.rw.transrw;          /*転送アドレスをセット*/
    retry = 3;                                  /*リトライ回数をセット*/
    WRITE_VERIFY();                             /*ライト・ベリファイ処理*/
    if(return_s == 0x12)                        /*ノット・レディ状態か？*/
        {
        do
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            T_U_READY();                            /*テスト・ユニット・レディ処理*/
            }
        while(return_s == 0x12);                /*ノット・レディ状態か？*/

        if(return_s == 0x00)                    /*正常終了か？*/
            {
            scsi_id = unit_n[packet->unit_number];  /*ターゲットのIDをセット*/
            d_lun = 0x00;                           /*ロジカル・ユニット・ナンバーをセット*/
            retry = 3;                              /*リトライ回数をセット*/
            MODE_SENSE();                           /*モード・センス処理*/
            if(return_s == 0x00)                    /*正常終了か？*/
                {
                u_l.msed[0] = msed_b[11];
                u_l.msed[1] = msed_b[10];
                u_l.msed[2] = msed_b[9];
                u_l.msed[3] = msed_b[8];
                if(u_l.byte_b == 512)               /*512バイト/ブロックでフォーマットされているか？*/
                    {
                    scsi_id = unit_n[packet->unit_number];      /*ターゲットのIDをセット*/
                    d_lun = 0x00;                               /*ロジカル・ユニット・ナンバーをセット*/
                    s_block_n = packet->up.rw.start_32;         /*開始ブロック番号をセット*/
                    s_block_n = s_block_n + 0x20L;
                    n_t_block = packet->up.rw.sector_count;     /*転送ブロック数をセット*/
                    t_address = packet->up.rw.transrw;          /*転送アドレスをセット*/
                    retry = 3;                                  /*リトライ回数をセット*/
                    WRITE_VERIFY();                             /*ライト・ベリファイ処理*/
                    }
                else
                    return_s = 0x12;                            /*ノット・レディをセット*/
                }
            }
        }
    W_STA_PACK();                                               /*ライト・コマンド・パケット・ステータス・セット処理*/
    return;
    }
```

A

保守／廃止

```
/*************************************************************************
 *                        Ｉ／Ｏリクエスト・エラー処理                   *
 *packetに“Ｉ／Ｏリクエスト・コマンド・コードが間違っている”をセットす*
 *る。                                                                   *
 *      入力：なし                                                       *
 *      出力：packet                                                     *
 *************************************************************************/


void ERROR_IO(void)
    {
                                                    /*Ｉ／Ｏリクエスト・コマンド・コードが間違っている*/
    packet->command_status = (ST_ERROR|ST_DONE|ST_ERR_BADCMD);
    return;
    }
```

保守／廃止

```
/*********************************************************************
 *          リード・コマンド・パケット・ステータス・セット処理          *
 *return_sに従ってコマンド・パケットにステータスをセットする。          *
 *                                                                   *
 *      入力：return_s                                               *
 *      出力：packet                                                 *
 *********************************************************************/


void R_STA_PACK(void)
    {
    switch(return_s)
        {
        case    0x00:
            packet->command_status = ST_DONE;       /*DONEﾋﾞｯﾄをセット*/
            break;
        case    0x02:
                                                    /*ﾗｲﾄ･ﾌﾟﾛﾃｸﾄをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_PRO;
            break;
        case    0x14:
                                                    /*無効なﾕﾆｯﾄをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_UNIT;
            break;
        case    0x06:
                                                    /*ﾉｯﾄ･ﾚﾃﾞｨをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_NOTRDY;
            break;
        case    0x0B:
        case    0x0F:
                                                    /*CRCｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_CRC;
            break;
        case    0x03:
        case    0x0E:
                                                    /*ｾｸﾀが存在しないをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_SECTOR;
            break;
        case    0x07:
                                                    /*書き込みｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_WRITE;
            break;
        case    0x01:
        case    0x0C:
        case    0x10:
        case    0x11:
                                                    /*読み出しｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_READ;
            break;
        case    0x12:
                                                    /*無効なﾒﾃﾞｨｱをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_MEDIA;
            break;
        case    0x04:
        case    0x05:
        case    0x08:
        case    0x09:
        case    0x0A:
        case    0x0D:
        case    0x13:
                                                    /*一般的なｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_GENERAL;
            break;
        }
    return;
    }
```

A

保守／廃止

```
/*************************************************************************
 *          ライト・コマンド・パケット・ステータス・セット処理           *
 *return_sに従ってコマンド・パケットにステータスをセットする。           *
 *                                                                       *
 *      入力：return_s                                                   *
 *      出力：packet                                                     *
 *************************************************************************/


void W_STA_PACK(void)
    {
    switch(return_s)
        {
        case    0x00:
            packet->command_status = ST_DONE;       /*DONEﾋﾞｯﾄをセット*/
            break;
        case    0x02:
                                                    /*ﾗｲﾄ･ﾌﾟﾛﾃｸﾄをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_PRO;
            break;
        case    0x14:
                                                    /*無効なﾕﾆｯﾄをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_UNIT;
            break;
        case    0x06:
                                                    /*ﾉｯﾄ･ﾚﾃﾞｨをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_NOTRDY;
            break;
        case    0x0B:
        case    0x0F:
                                                    /*CRCｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_CRC;
            break;
        case    0x03:
        case    0x0E:
                                                    /*ｾｸﾀが存在しないをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_SECTOR;
            break;
        case    0x07:
        case    0x10:
        case    0x11:
                                                    /*書き込みｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_WRITE;
            break;
        case    0x01:
        case    0x0C:
                                                    /*読み出しｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_READ;
            break;
        case    0x12:
                                                    /*無効なﾒﾃﾞｨｱをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_MEDIA;
            break;
        case    0x04:
        case    0x05:
        case    0x08:
        case    0x09:
        case    0x0A:
        case    0x0D:
        case    0x13:
                                                    /*一般的なｴﾗｰをセット*/
            packet->command_status = ST_DONE|ST_ERROR|ST_ERR_GENERAL;
            break;
        }
    return;
    }
```

保守／廃止

```
/*************************************************************
 *  SCSIコマンド実行ブロック    SCOM.CPP          *
 *************************************************************/

#include <dos.h>


extern unsigned long aspi_entry;                        /*ASPIｴﾝﾄﾘ･ﾎﾟｲﾝﾄ*/

unsigned char scsi_id;                                  /*ﾀｰｹﾞｯﾄのID番号*/
unsigned char d_lun;                                    /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞｰ*/
unsigned long s_block_n;                                /*開始ﾌﾞﾛｯｸ番号*/
unsigned char far *t_address;                           /*転送ｱﾄﾞﾚｽ*/
unsigned int n_t_block;                                 /*転送ﾌﾞﾛｯｸ数*/
unsigned char retry;                                    /*ﾘﾄﾗｲ回数*/
unsigned char return_s;                                 /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ*/
unsigned char msed_b[20];                               /*ﾓｰﾄﾞ･ｾﾝｽ･ﾃﾞｰﾀ*/

                                                        /*ﾓｰﾄﾞ･ｾﾚｸﾄ･ﾃﾞｰﾀ*/
unsigned char msld_b[44] = {0x00,0x00,0x00,0x08,
                            0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,
                            0x01,0x06,0x24,0x10,0x08,0x00,0x00,0x00,
                            0x03,0x16,0x00,0x00,0x00,0x00,0x00,0x00,
                            0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x00,
                            0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char rcd_b[8];                                 /*ﾘｰﾄﾞ･ｷｬﾊﾟｼﾃｨ･ﾃﾞｰﾀ*/
unsigned char inqu_d[45];                               /*ｲﾝｸﾜｲｱ･ﾃﾞｰﾀ*/
unsigned char t_exist;                                  /*ﾀｰｹﾞｯﾄが存在しているID番号を示す*/


                                                        /*SCSI REQUEST BLOCK(SRB)の構造体*/
struct aspi_t
    {
    unsigned char command_code;                         /*command code              (w)*/
    unsigned char status;                               /*status                    (r)*/
    unsigned char h_adapter_n;                          /*host adapter number       (w)*/
    unsigned char scsi_request_f;                       /*scsi request flags        (w)*/
    unsigned long reserved;                             /*reserved                    */

    union {
        struct
            {
                                                        /*HOST ADAPTER INQUIRY*/
            unsigned char n_host_a;                     /*number of host adapters   (r)*/
            unsigned char id_host_a;                    /*id of host adapter        (r)*/
            unsigned char scsi_m_id[16];                /*scsi manager id           (r)*/
            unsigned char h_adapter_id[16];             /*host adapter id           (r)*/
            unsigned char h_adapterp[16];               /*host adapter parameters   (r)*/
            }   hadap;
        struct
            {
                                                        /*GET DEVICE TYPE*/
            unsigned char target_id;                    /*target id                 (w)*/
            unsigned char lun;                          /*lun                       (w)*/
            unsigned char p_device_t;                   /*peripheral device type    (r)*/
            }   gdevice;
        struct
            {
                                                        /*EXECUTE SCSI I/O REQUEST*/
            unsigned char target_id;                    /*target id                 (w)*/
            unsigned char lun;                          /*lun                       (w)*/
            unsigned long data_a_l;                     /*data length               (w)*/
            unsigned char sense_a_l;                    /*sense length              (w)*/
            unsigned char far *data_bp;                 /*data buff pointer         (w)*/
            unsigned char far *srb_lp;                  /*srb link pointer          (w)*/
            unsigned char scsicdb_l;                    /*scsi cdb length           (w)*/
            unsigned char h_adapter_s;                  /*host adapter status       (r)*/
            unsigned char target_status;                /*target status             (r)*/
            unsigned char far *post_ra;                 /*post routine address      (w)*/
            unsigned char warkspace[34];                /*aspi warkspace              */
            unsigned char scsics[256];                  /*scsi cdb and sense data (w/r)*/
            }   escsirq;
```

A

```
        struct
            {
                                                    /*RESET SCSI DEVICE*/
            unsigned char target_id;                /*target id             (w)*/
            unsigned char lun;                      /*lun                   (w)*/
            unsigned char reserved[14];             /*reserved                 */
            unsigned char h_adapter_s;              /*host adapter status   (r)*/
            unsigned char target_status;            /*target status         (r)*/
            unsigned char far *post_ra;             /*post routine address  (w)*/
            unsigned char warkspace[34];            /*aspi warkspace           */
            }   resetsd;
        }   u_srb;
    }   srb;

void READ_DATA(void);                               /*リード・データ処理*/
void WRITE_DATA(void);                              /*ライト・データ処理*/
void VERIFY(void);                                  /*ベリファイ処理*/
void WRITE_VERIFY(void);                            /*ライト・アンド・ベリファイ処理*/
void SEEK(void);                                    /*シーク処理*/
void RECALIBRATE(void);                             /*リキャリブレイト処理*/
void RETRACT(void);                                 /*リトラクト処理*/
void RESERVE(void);                                 /*リザーブ処理*/
void RELEASE(void);                                 /*リリース処理*/
void SEND_DIAG(void);                               /*センド・ダイアグノスティック*/
void MODE_SENSE(void);                              /*モード・センス処理*/
void MODE_SELECT(void);                             /*モード・セレクト処理*/
void READ_CAPACITY(void);                           /*リード・キャパシティ処理*/
void FORMAT(void);                                  /*フォーマット処理*/
void INQUIRY(void);                                 /*インクワイアリ処理*/
void T_U_READY(void);                               /*テスト・ユニット・レディ処理*/
void T_INITIAL(void);                               /*ターゲット初期化処理*/
void SET_RET_S(void);                               /*リターン・ステータス・セット処理*/


/************************************************************************
 *                          リード・データ処理                          *
 *  ターゲットからデータをリードする                                    *
 *                                                                      *
 *      入力：aspi_entey, scsi_id, d_lun, s_block_n, t_address, n_t_block,  *
 *            retry                                                     *
 *      出力：return_s                                                  *
 ************************************************************************/

void READ_DATA(void)
    {
    union                                           /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセットする共用体*/
        {
        unsigned long long_d;
        unsigned char b[4];
        }   l_byte;

    union                                           /*転送長をセットする共用体*/
        {
        unsigned int int_d;
        unsigned char b[2];
        }   i_byte;

    unsigned long buff;

    retry = retry + 1;
    while(retry != 0x00)                            /*最後までリトライしたか？*/
        {
        srb.u_srb.escsirq.scsics[0] = 0x28;         /*READ EXTENDEDｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;    /*lunをセット*/

        l_byte.long_d = s_block_n;
        srb.u_srb.escsirq.scsics[2] = l_byte.b[3];  /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセット*/
        srb.u_srb.escsirq.scsics[3] = l_byte.b[2];
        srb.u_srb.escsirq.scsics[4] = l_byte.b[1];
        srb.u_srb.escsirq.scsics[5] = l_byte.b[0];
        srb.u_srb.escsirq.scsics[6] = 0x00;
```

```
    i_byte.int_d = n_t_block;
    srb.u_srb.escsirq.scsics[7] = i_byte.b[1];          /*転送長*/
    srb.u_srb.escsirq.scsics[8] = i_byte.b[0];
    srb.u_srb.escsirq.scsics[9] = 0x00;

    srb.command_code = 0x02;                            /*EXECUTE SCSI I/O REQUEST*/
    srb.h_adapter_n = 0x00;                             /*host adapter numberに１をセット*/
    srb.scsi_request_f = 0x00;                          /*scsi request flagにOOHをセット*/
    srb.u_srb.escsirq.target_id = scsi_id;              /*ﾀｰｹﾞｯﾄのID番号をセット*/
    srb.u_srb.escsirq.lun = d_lun;                      /*lunをセット*/

    buff = (long)n_t_block;
    srb.u_srb.escsirq.data_a_l = buff*512;              /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
    srb.u_srb.escsirq.sense_a_l = 13;                   /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
    srb.u_srb.escsirq.data_bp = t_address;              /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
    srb.u_srb.escsirq.scsicdb_l = 10;                   /*CDB長をセット*/

    asm push ds;                                        /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
    asm lea bx,srb;
    asm push bx;
    asm lea bx,aspi_entry;
    asm call DWORD PTR [bx];                            /*ASPIをCALL*/
    asm add sp,4;

    SET_RET_S();                                        /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
    if((return_s==0x0B)||(return_s==0x0C)||(return_s==0x0E)
        ||(return_s==0x0F)||(return_s==0x10))
        {
        srb.u_srb.escsirq.scsics[0] = 0x01;             /*REZERO UNITｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;        /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに１をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        --retry;
        }
    else
        {
        if((return_s==0x04)||(return_s==0x05)
            ||(return_s==0x06)||(return_s==0x0A))
            --retry;
        else
            break;
        }

    }
return;
}
```

保守／廃止

```
/*************************************************************************
 *                          ライト・データ処理                           *
 *  ターゲットへデータをライトする                                        *
 *                                                                        *
 *      入力：aspi_entry, scsi_id, d_lun, s_block_n, t_address, n_t_block,*
 *            retry                                                       *
 *      出力：return_s                                                    *
 *************************************************************************/

void WRITE_DATA(void)
    {
    union                                           /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセットする共用体*/
        {
        unsigned long long_d;
        unsigned char b[4];
        }   l_byte;

    union                                           /*転送長をセットする共用体*/
        {
        unsigned int int_d;
        unsigned char b[2];
        }   i_byte;

    unsigned long buff;

    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x2A;         /*WRITE EXTENDEDｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;    /*lunをセット*/

        l_byte.long_d = s_block_n;
        srb.u_srb.escsirq.scsics[2] = l_byte.b[3];  /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセット*/
        srb.u_srb.escsirq.scsics[3] = l_byte.b[2];
        srb.u_srb.escsirq.scsics[4] = l_byte.b[1];
        srb.u_srb.escsirq.scsics[5] = l_byte.b[0];
        srb.u_srb.escsirq.scsics[6] = 0x00;

        i_byte.int_d = n_t_block;
        srb.u_srb.escsirq.scsics[7] = i_byte.b[1];  /*転送長*/
        srb.u_srb.escsirq.scsics[8] = i_byte.b[0];
        srb.u_srb.escsirq.scsics[9] = 0x00;

        srb.command_code = 0x02;                    /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                     /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                  /*scsi request flagに00Hをｾｯﾄ*/
        srb.u_srb.escsirq.target_id = scsi_id;      /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;              /*lunをセット*/

        buff = (long)n_t_block;
        srb.u_srb.escsirq.data_a_l = buff*512;      /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;           /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.data_bp = t_address;      /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 10;           /*CDB長をセット*/

        asm push ds;                                /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                    /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x0B)||(return_s==0x0F)
            ||(return_s==0x10)||(return_s==0x11))
            {
            srb.u_srb.escsirq.scsics[0] = 0x01;     /*REZERO UNITｺﾏﾝﾄﾞ*/
            srb.u_srb.escsirq.scsics[1] = d_lun <<5;/*lunをセット*/
            srb.u_srb.escsirq.scsics[2] = 0x00;
            srb.u_srb.escsirq.scsics[3] = 0x00;
            srb.u_srb.escsirq.scsics[4] = 0x00;
            srb.u_srb.escsirq.scsics[5] = 0x00;
```

```
            srb.command_code = 0x02;                    /*EXECUTE SCSI I/O REQUEST*/
            srb.h_adapter_n = 0x00;                     /*host adapter numberに1をセット*/
            srb.scsi_request_f = 0x00;                  /*scsi request flagに00Hをセット*/
            srb.u_srb.escsirq.target_id = scsi_id;      /*ﾀｰｹﾞｯﾄのID番号をセット*/
            srb.u_srb.escsirq.lun = d_lun;              /*lunをセット*/
            srb.u_srb.escsirq.data_a_l = 0x00L;         /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
            srb.u_srb.escsirq.sense_a_l = 13;           /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
            srb.u_srb.escsirq.scsicdb_l = 6;            /*CDB長をセット*/

            asm push ds;                                /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
            asm lea bx,srb;
            asm push bx;
            asm lea bx,aspi_entry;
            asm call DWORD PTR [bx];                    /*ASPIをCALL*/
            asm add sp,4;

            --retry;
            }
        else
            {
            if((return_s==0x04)||(return_s==0x05)||(return_s==0x06)
                ||(return_s==0x07)||(return_s==0x0A))
                --retry;
            else
                break;
            }

        }
    return;
    }


/*************************************************************************
 *                          ベリファイ処理                               *
 *  ターゲット内のデータをベリファイする                                 *
 *                                                                       *
 *      入力 : aspi_entry, scsi_id, d_lun, s_block_n, n_t_block, retry   *
 *      出力 : return_s                                                  *
 *************************************************************************/

void VERIFY(void)
    {
    union                                               /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセットする共用体*/
        {
        unsigned long long_d;
        unsigned char b[4];
        }   l_byte;

    union                                               /*転送長をセットする共用体*/
        {
        unsigned int int_d;
        unsigned char b[2];
        }   i_byte;

    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x2F;             /*VERIFYｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;        /*lunをセット*/

        l_byte.long_d = s_block_n;
        srb.u_srb.escsirq.scsics[2] = l_byte.b[3];      /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセット*/
        srb.u_srb.escsirq.scsics[3] = l_byte.b[2];
        srb.u_srb.escsirq.scsics[4] = l_byte.b[1];
        srb.u_srb.escsirq.scsics[5] = l_byte.b[0];
        srb.u_srb.escsirq.scsics[6] = 0x00;

        i_byte.int_d = n_t_block;
        srb.u_srb.escsirq.scsics[7] = i_byte.b[1];      /*ベリファイ長*/
        srb.u_srb.escsirq.scsics[8] = i_byte.b[0];
        srb.u_srb.escsirq.scsics[9] = 0x00;
```

```
    srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
    srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
    srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
    srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
    srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/

    srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
    srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
    srb.u_srb.escsirq.scsicdb_l = 10;               /*CDB長をセット*/

    asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
    asm lea bx,srb;
    asm push bx;
    asm lea bx,aspi_entry;
    asm call DWORD PTR [bx];                        /*ASPIをCALL*/
    asm add sp,4;

    SET_RET_S();                                    /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
    if((return_s==0x0B)||(return_s==0x0C)||(return_s==0x0E)
        ||(return_s==0x0F)||(return_s==0x10))
        {
        srb.u_srb.escsirq.scsics[0] = 0x01;         /*REZERO UNITｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;    /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                    /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                     /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                  /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;      /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;              /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;         /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;           /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;            /*CDB長をセット*/

        asm push ds;                                /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm push ds;
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                    /*ASPIをCALL*/
        asm add sp,4;

        --retry;
        }
    else
        {
        if((return_s==0x04)||(return_s==0x05)
            ||(return_s==0x06)||(return_s==0x0A))
            --retry;
        else
            break;
        }

    }
return;
}
```

```
/*************************************************************************
 *                    ライト・アンド・ベリファイ処理                     *
 *  ターゲットへデータをライトし，ライトしたデータをベリファイする       *
 *                                                                       *
 *      入力：aspi_entry, scsi_id, d_lun, s_block_n, t_address, n_t_block,  *
 *            retry                                                      *
 *      出力：return_s                                                   *
 *************************************************************************/

void WRITE_VERIFY(void)
    {
    union                                           /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをｾｯﾄする共用体*/
        {
        unsigned long long_d;
        unsigned char b[4];
        }   l_byte;

    union                                           /*転送長をｾｯﾄする共用体*/
        {
        unsigned int int_d;
        unsigned char b[2];
        }   i_byte;

    unsigned long buff;

    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x2E;         /*WRITE AND VERIFYｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;    /*lunをｾｯﾄ*/

        l_byte.long_d = s_block_n;
        srb.u_srb.escsirq.scsics[2] = l_byte.b[3];  /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをｾｯﾄ*/
        srb.u_srb.escsirq.scsics[3] = l_byte.b[2];
        srb.u_srb.escsirq.scsics[4] = l_byte.b[1];
        srb.u_srb.escsirq.scsics[5] = l_byte.b[0];
        srb.u_srb.escsirq.scsics[6] = 0x00;

        i_byte.int_d = n_t_block;
        srb.u_srb.escsirq.scsics[7] = i_byte.b[1];  /*転送長*/
        srb.u_srb.escsirq.scsics[8] = i_byte.b[0];
        srb.u_srb.escsirq.scsics[9] = 0x00;

        srb.command_code = 0x02;                    /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                     /*host adapter numberに1をｾｯﾄ*/
        srb.scsi_request_f = 0x00;                  /*scsi request flagにOOHをｾｯﾄ*/
        srb.u_srb.escsirq.target_id = scsi_id;      /*ﾀｰｹﾞｯﾄのID番号をｾｯﾄ*/
        srb.u_srb.escsirq.lun = d_lun;              /*lunをｾｯﾄ*/

        buff = (long)n_t_block;
        srb.u_srb.escsirq.data_a_l = buff*512;      /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をｾｯﾄ*/
        srb.u_srb.escsirq.sense_a_l = 13;           /*ｾﾝｽ･ﾃﾞｰﾀ長をｾｯﾄ*/
        srb.u_srb.escsirq.data_bp = t_address;      /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 10;           /*CDB長をｾｯﾄ*/

        asm push ds;                                /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにｾｯﾄ*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                    /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x0B)||(return_s==0x0C)||(return_s==0x0E)
            ||(return_s==0x0F)||(return_s==0x10)||(return_s==0x11))
            {
            srb.u_srb.escsirq.scsics[0] = 0x01;     /*REZERO UNITｺﾏﾝﾄﾞ*/
            srb.u_srb.escsirq.scsics[1] = d_lun <<5;/*lunをｾｯﾄ*/
            srb.u_srb.escsirq.scsics[2] = 0x00;
            srb.u_srb.escsirq.scsics[3] = 0x00;
            srb.u_srb.escsirq.scsics[4] = 0x00;
            srb.u_srb.escsirq.scsics[5] = 0x00;
```

```
        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        --retry;
        }
    else
        {
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x06)
            ||(return_s==0x07)||(return_s==0x0A))
            --retry;
        else
            break;
        }

    }
return;
}


/*************************************************************************
 *                          シーク処理                                   *
 *  目的のブロック・アドレスへヘッドをシークする                         *
 *                                                                       *
 *      入力：aspi_entry,scsi_id,d_lun,s_block_n,retry                   *
 *      出力：return_s                                                   *
 *************************************************************************/

void SEEK(void)
    {
    union                                               /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセットする共用体*/
        {
        unsigned long long_d;
        unsigned char b[4];
        }   l_byte;

    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x2B;             /*SEEK EXTENDEDｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;        /*lunをセット*/

        l_byte.long_d = s_block_n;
        srb.u_srb.escsirq.scsics[2] = l_byte.b[3];      /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセット*/
        srb.u_srb.escsirq.scsics[3] = l_byte.b[2];
        srb.u_srb.escsirq.scsics[4] = l_byte.b[1];
        srb.u_srb.escsirq.scsics[5] = l_byte.b[0];
        srb.u_srb.escsirq.scsics[6] = 0x00;
        srb.u_srb.escsirq.scsics[7] = 0x00;
        srb.u_srb.escsirq.scsics[8] = 0x00;
        srb.u_srb.escsirq.scsics[9] = 0x00;

        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/

        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 10;               /*CDB長をセット*/
```

```
        asm push ds;                                    /*SRBのポインタをスタックにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                    /*リターン・ステータス・セット処理*/
        if(return_s == 0x0F)
            {
            srb.u_srb.escsirq.scsics[0] = 0x01;         /*REZERO UNITコマンド*/
            srb.u_srb.escsirq.scsics[1] = d_lun <<5;    /*lunをセット*/
            srb.u_srb.escsirq.scsics[2] = 0x00;
            srb.u_srb.escsirq.scsics[3] = 0x00;
            srb.u_srb.escsirq.scsics[4] = 0x00;
            srb.u_srb.escsirq.scsics[5] = 0x00;

            srb.command_code = 0x02;                    /*EXECUTE SCSI I/O REQUEST*/
            srb.h_adapter_n = 0x00;                     /*host adapter numberに１をセット*/
            srb.scsi_request_f = 0x00;                  /*scsi request flagに00Hをセット*/
            srb.u_srb.escsirq.target_id = scsi_id;      /*ターゲットのID番号をセット*/
            srb.u_srb.escsirq.lun = d_lun;              /*lunをセット*/
            srb.u_srb.escsirq.data_a_l = 0x00L;         /*転送データ・バイト数をセット*/
            srb.u_srb.escsirq.sense_a_l = 13;           /*センス・データ長をセット*/
            srb.u_srb.escsirq.scsicdb_l = 6;            /*CDB長をセット*/

            asm push ds;                                /*SRBのポインタをスタックにセット*/
            asm lea bx,srb;
            asm push bx;
            asm lea bx,aspi_entry;
            asm call DWORD PTR [bx];                    /*ASPIをCALL*/
            asm add sp,4;

            --retry;
            }
        else
            {
            if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
                --retry;
            else
                break;
            }

        }
    return;
    }


/*************************************************************************
 *                      リキャリブレイト処理                             *
 *  シリンダ０へヘッドをシークする                                       *
 *                                                                       *
 *      入力：aspi_entry,scsi_id,d_lun,retry                             *
 *      出力：return_s                                                   *
 *************************************************************************/

void RECALIBRATE(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x01;             /*REZERO UNITコマンド*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;        /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;
```

```
        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                    /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)
            ||(return_s==0x0A)||(return_s==0x0F))
                --retry;
            else
                break;

        }
    return;
    }


/*************************************************************************
 *                          リトラクト処理                               *
 *  ヘッドをシッピング・ゾーンへシークする                               *
 *                                                                       *
 *      入力：aspi_entry,scsi_id,d_lun,retry                             *
 *      出力：return_s                                                   *
 *************************************************************************/

void RETRACT(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x1B;             /*START/STOP UNITｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;        /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;             /*S=0:ｽﾄｯﾌﾟ･ﾕﾆｯﾄ*/
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                    /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)
            ||(return_s==0x0A)||(return_s==0x0F))
                --retry;
            else
                break;

        }
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                          リザーブ処理                                 *
 *  ターゲットをリザーブし，他のイニシエータからのアクセスを禁止する    *
 *                                                                       *
 *      入力：aspi_entry, scsi_id, d_lun, retry                          *
 *      出力：return_s                                                   *
 *************************************************************************/

void RESERVE(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x16;              /*RESERVE UNITｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;         /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                         /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                          /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                       /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;           /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                   /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;              /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;                /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                 /*CDB長をセット*/

        asm push ds;                                     /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                         /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                     /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
                --retry;
            else
                break;

        }
    return;
    }


/*************************************************************************
 *                          リリース処理                                 *
 *  ターゲットのリリースし，他のイニシエータからのアクセスを許可する    *
 *                                                                       *
 *      入力：aspi_entry, scsi_id, d_lun, retry                          *
 *      出力：return_s                                                   *
 *************************************************************************/

void RELEASE(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x17;              /*RELEASE UNITｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;         /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;
```

A

```
        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                    /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
                --retry;
            else
                break;

        }
    return;
    }


/*************************************************************************
 *                  センド・ダイアグノスティック処理                    *
 *  ターゲットへ自己診断を要求する                                       *
 *                                                                       *
 *      入力 : aspi_entry, scsi_id, d_lun, retry                         *
 *      出力 : return_s                                                  *
 *************************************************************************/

void SEND_DIAG(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x1D;             /*SEND DIAGNOSTICｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = (d_lun <<5)|0x04; /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                        /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;          /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                  /*lunをセット*/
        srb.u_srb.escsirq.data_a_l = 0x00L;             /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;               /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                /*CDB長をセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                    /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
                --retry;
            else
                break;

        }
    return;
    }
```

```
/*************************************************************************
 *                          モード・センス処理                           *
 *  ターゲットからデバイス・パラメータをリードする                       *
 *                                                                       *
 *      入力：aspi_entry, scsi_id, d_lun, retry                          *
 *      出力：return_s, msed_b[]                                         *
 *************************************************************************/

void MODE_SENSE(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x1A;              /*MODE SENSEｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;         /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0xC1;              /*REPORT SAVED VALUE/PAGE CODE = 1*/
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x14;              /*ﾓｰﾄﾞ･ｾﾝｽ･ﾃﾞｰﾀ数*/
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                         /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                          /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                       /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;           /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                   /*lunをセット*/

        srb.u_srb.escsirq.data_a_l = 0x14L;              /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;                /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.data_bp = msed_b;              /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 6;                 /*CDB長をセット*/

        asm push ds;                                     /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                         /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                     /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
            --retry;
        else
            break;
        }
    return;
    }


/*************************************************************************
 *                          モード・セレクト処理                         *
 *  ターゲットへデバイス・パラメータをセットする                         *
 *                                                                       *
 *      入力：aspi_entry, scsi_id, d_lun, retry                          *
 *      出力：return_s                                                   *
 *************************************************************************/

void MODE_SELECT(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x15;              /*MODE SELECTｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = ((d_lun <<5)|0x01);/*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;              /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをセット*/
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x2C;              /*ﾓｰﾄﾞ･ｾﾚｸﾄ･ﾃﾞｰﾀ数*/
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                         /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                          /*host adapter numberに1をセット*/
        srb.scsi_request_f = 0x00;                       /*scsi request flagに00Hをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;           /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                   /*lunをセット*/
```

```
        srb.u_srb.escsirq.data_a_l = 0x2CL;                    /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をｾｯﾄ*/
        srb.u_srb.escsirq.sense_a_l = 13;                      /*ｾﾝｽ･ﾃﾞｰﾀ長をｾｯﾄ*/
        srb.u_srb.escsirq.data_bp = msld_b;                    /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 6;                       /*CDB長をｾｯﾄ*/

        asm push ds;                                           /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにｾｯﾄ*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                               /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                           /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
            --retry;
        else
            break;
        }
    return;
    }


/*************************************************************************
 *                        リード・キャパシティ処理                       *
 *  ターゲットから総ブロック数とブロック長をリードする                   *
 *                                                                       *
 *      入力：aspi_entry,scsi_id,d_lun,retry                             *
 *      出力：return_s,rcd_b[]                                           *
 *************************************************************************/

void READ_CAPACITY(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x25;                    /*READ CAPACITYｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;               /*lunをｾｯﾄ*/
        srb.u_srb.escsirq.scsics[2] = 0x00;                    /*論理ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽをｾｯﾄ*/
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;
        srb.u_srb.escsirq.scsics[6] = 0x00;
        srb.u_srb.escsirq.scsics[7] = 0x00;
        srb.u_srb.escsirq.scsics[8] = 0x00;
        srb.u_srb.escsirq.scsics[9] = 0x00;

        srb.command_code = 0x02;                               /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                                /*host adapter numberに1をｾｯﾄ*/
        srb.scsi_request_f = 0x00;                             /*scsi request flagに00Hをｾｯﾄ*/
        srb.u_srb.escsirq.target_id = scsi_id;                 /*ﾀｰｹﾞｯﾄのID番号をｾｯﾄ*/
        srb.u_srb.escsirq.lun = d_lun;                         /*lunをｾｯﾄ*/

        srb.u_srb.escsirq.data_a_l = 0x08L;                    /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をｾｯﾄ*/
        srb.u_srb.escsirq.sense_a_l = 13;                      /*ｾﾝｽ･ﾃﾞｰﾀ長をｾｯﾄ*/
        srb.u_srb.escsirq.data_bp = rcd_b;                     /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 10;                      /*CDB長をｾｯﾄ*/

        asm push ds;                                           /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにｾｯﾄ*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                               /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                           /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
            --retry;
        else
            break;
        }
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                          フォーマット処理                             *
 *  ターゲットをフォーマットする                                         *
 *                                                                       *
 *      入力 : aspi_entry, scsi_id, d_lun, retry                         *
 *      出力 : return_s                                                  *
 *************************************************************************/

void FORMAT(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x04;              /*FORMATｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;         /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x00;
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                         /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                          /*host adapter numberに１をセット*/
        srb.scsi_request_f = 0x00;                       /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;           /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                   /*lunをセット*/

        srb.u_srb.escsirq.data_a_l = 0x00L;              /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
        srb.u_srb.escsirq.sense_a_l = 13;                /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
        srb.u_srb.escsirq.scsicdb_l = 6;                 /*CDB長をセット*/

        asm push ds;                                     /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                         /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                     /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x06)
            ||(return_s==0x07)||(return_s==0x0A)||(return_s==0x0F)||(return_s==0x13))
            --retry;
        else
            break;
        }
    return;
    }


/*************************************************************************
 *                          インクワイアリ処理                           *
 *  ターゲットからインクワイアリ・データをリードする                     *
 *                                                                       *
 *      入力 : aspi_entry, scsi_id, d_lun, retry                         *
 *      出力 : return_s, inqu_d[]                                        *
 *************************************************************************/

void INQUIRY(void)
    {
    retry = retry + 1;
    while(retry != 0x00)
        {
        srb.u_srb.escsirq.scsics[0] = 0x12;              /*INQUIRYｺﾏﾝﾄﾞ*/
        srb.u_srb.escsirq.scsics[1] = d_lun <<5;         /*lunをセット*/
        srb.u_srb.escsirq.scsics[2] = 0x00;
        srb.u_srb.escsirq.scsics[3] = 0x00;
        srb.u_srb.escsirq.scsics[4] = 0x24;              /*ｲﾝｸﾜｲｱﾘ･ﾃﾞｰﾀ数*/
        srb.u_srb.escsirq.scsics[5] = 0x00;

        srb.command_code = 0x02;                         /*EXECUTE SCSI I/O REQUEST*/
        srb.h_adapter_n = 0x00;                          /*host adapter numberに１をセット*/
        srb.scsi_request_f = 0x00;                       /*scsi request flagにOOHをセット*/
        srb.u_srb.escsirq.target_id = scsi_id;           /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.escsirq.lun = d_lun;                   /*lunをセット*/
```

A

```
        srb.u_srb.escsirq.data_a_l = 0x24L;                    /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をｾｯﾄ*/
        srb.u_srb.escsirq.sense_a_l = 13;                      /*ｾﾝｽ･ﾃﾞｰﾀ長をｾｯﾄ*/
        srb.u_srb.escsirq.data_bp = inqu_d;                    /*転送ﾃﾞｰﾀ･ｱﾄﾞﾚｽ*/
        srb.u_srb.escsirq.scsicdb_l = 6;                       /*CDB長をｾｯﾄ*/

        asm push ds;                                           /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにｾｯﾄ*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                               /*ASPIをCALL*/
        asm add sp,4;

        SET_RET_S();                                           /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
        if((return_s==0x04)||(return_s==0x05)||(return_s==0x0A))
            --retry;
        else
            break;
        }
    return;
    }


/*************************************************************************
 *                    テスト・ユニット・レディ処理                       *
 *  ターゲットがレディ状態であることを調べる                             *
 *                                                                       *
 *      入力：aspi_entry,scsi_id,d_lun                                   *
 *      出力：return_s                                                   *
 *************************************************************************/

void T_U_READY(void)
    {
    srb.u_srb.escsirq.scsics[0] = 0x00;                        /*TEST UNIT READYｺﾏﾝﾄﾞ*/
    srb.u_srb.escsirq.scsics[1] = d_lun <<5;                   /*lunをｾｯﾄ*/
    srb.u_srb.escsirq.scsics[2] = 0x00;
    srb.u_srb.escsirq.scsics[3] = 0x00;
    srb.u_srb.escsirq.scsics[4] = 0x00;
    srb.u_srb.escsirq.scsics[5] = 0x00;

    srb.command_code = 0x02;                                   /*EXECUTE SCSI I/O REQUEST*/
    srb.h_adapter_n = 0x00;                                    /*host adapter numberに1をｾｯﾄ*/
    srb.scsi_request_f = 0x00;                                 /*scsi request flagに00Hをｾｯﾄ*/
    srb.u_srb.escsirq.target_id = scsi_id;                     /*ﾀｰｹﾞｯﾄのID番号をｾｯﾄ*/
    srb.u_srb.escsirq.lun = d_lun;                             /*lunをｾｯﾄ*/

    srb.u_srb.escsirq.data_a_l = 0x00L;                        /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をｾｯﾄ*/
    srb.u_srb.escsirq.sense_a_l = 13;                          /*ｾﾝｽ･ﾃﾞｰﾀ長をｾｯﾄ*/
    srb.u_srb.escsirq.scsicdb_l = 6;                           /*CDB長をｾｯﾄ*/

    asm push ds;                                               /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにｾｯﾄ*/
    asm lea bx,srb;
    asm push bx;
    asm lea bx,aspi_entry;
    asm call DWORD PTR [bx];                                   /*ASPIをCALL*/
    asm add sp,4;

    SET_RET_S();                                               /*ﾘﾀｰﾝ･ｽﾃｰﾀｽ･ｾｯﾄ処理*/
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                    ターゲットの初期化処理                             *
 *  接続されているターゲットを調べ，初期化する                           *
 *                                                                       *
 *      入力：aspi_entry,d_lun                                           *
 *      出力：t_exist                                                    *
 *************************************************************************/

void T_INITIAL(void)
    {
    unsigned char id_c;                                 /*ID番号のカウンタ*/
    unsigned char s_retry;                              /*ﾘﾄﾗｲ回数*/

    id_c = 0x00;                                        /*ﾘｾｯﾄ*/
    t_exist = 0x00;                                     /*ﾘｾｯﾄ*/

    while(id_c != 0x08)
        {
        srb.command_code = 0x01;                        /*GET DEVICE TYPE*/
        srb.h_adapter_n = 0x00;                         /*host adapter numberに0をセット*/
        srb.scsi_request_f = 0x00;                      /*scsi request flagに00Hをセット*/
        srb.u_srb.gdevice.target_id = id_c;             /*ﾀｰｹﾞｯﾄのID番号をセット*/
        srb.u_srb.gdevice.lun = d_lun;                  /*lunをセット*/

        asm push ds;                                    /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
        asm lea bx,srb;
        asm push bx;
        asm lea bx,aspi_entry;
        asm call DWORD PTR [bx];                        /*ASPIをCALL*/
        asm add sp,4;
        if((srb.status == 0x01) && (srb.u_srb.gdevice.p_device_t == 0x00))
            {
            s_retry = 0x00;                             /*ﾘﾄﾗｲ･ｶｳﾝﾀ･ｸﾘｱ*/
            while(s_retry != 0x03)                      /*３回までリトライする*/
                {
                srb.u_srb.escsirq.scsics[0] = 0x1B;     /*START/STOP UNITｺﾏﾝﾄﾞ*/
                srb.u_srb.escsirq.scsics[1] = d_lun <<5;/*lunをセット*/
                srb.u_srb.escsirq.scsics[2] = 0x00;
                srb.u_srb.escsirq.scsics[3] = 0x00;
                srb.u_srb.escsirq.scsics[4] = 0x01;     /*ｽﾀｰﾄ･ﾕﾆｯﾄ*/
                srb.u_srb.escsirq.scsics[5] = 0x00;

                srb.command_code = 0x02;                /*EXECUTE SCSI I/O REQUEST*/
                srb.h_adapter_n = 0x00;                 /*host adapter numberに１をセット*/
                srb.scsi_request_f = 0x00;              /*scsi request flagに00Hをセット*/
                srb.u_srb.escsirq.target_id = id_c;     /*ﾀｰｹﾞｯﾄのID番号をセット*/
                srb.u_srb.escsirq.lun = d_lun;          /*lunをセット*/
                srb.u_srb.escsirq.data_a_l = 0x00L;     /*転送ﾃﾞｰﾀ･ﾊﾞｲﾄ数をセット*/
                srb.u_srb.escsirq.sense_a_l = 13;       /*ｾﾝｽ･ﾃﾞｰﾀ長をセット*/
                srb.u_srb.escsirq.scsicdb_l = 6;        /*CDB長をセット*/

                asm push ds;                            /*SRBのﾎﾟｲﾝﾀをｽﾀｯｸにセット*/
                asm lea bx,srb;
                asm push bx;
                asm lea bx,aspi_entry;
                asm call DWORD PTR [bx];                /*ASPIをCALL*/
                asm add sp,4;
                if((srb.status==0x01)&&(srb.u_srb.escsirq.target_status==0x00))
                    {
                    t_exist = t_exist | (0x01 << id_c);/*接続されているﾀｰｹﾞｯﾄID番号と対応するﾋﾞｯﾄをセット*/
                    break;
                    }
                else
                    ++s_retry;
                }
            }
        ++id_c;
        }
    return;
    }
```

```
/*************************************************************************
 *                  リターン・ステータス・セット処理                     *
 *  ＳＲＢの結果からリターン・ステータスをセットする                     *
 *                                                                       *
 *      入力：srb                                                        *
 *      出力：return_s                                                   *
 *************************************************************************/


void SET_RET_S(void)
    {
    switch(srb.u_srb.escsirq.target_status)              /*ﾀｰｹﾞｯﾄ･ｽﾃｰﾀｽ*/
        {
        case    0x08:                                    /*busy*/
        case    0x18:                                    /*reservation conflict*/
            return_s = 0x09;                             /*一般的なエラー*/
            break;
        case    0x00:                                    /*good*/
            switch(srb.status)                           /*ASPIｽﾃｰﾀｽ*/
                {
                case    0x02:                            /*scsi request abort by host*/
                    return_s = 0x04;                     /*装置に異常があった*/
                    break;
                case    0x01:                            /*scsi request completed without error*/
                    return_s = 0x00;                     /*正常終了*/
                    break;
                case    0x04:                            /*scsi request completed with error*/
                    switch(srb.u_srb.escsirq.h_adapter_s)/*host adapter status*/
                        {
                        case    0x11:                    /*selection timeout*/
                            return_s = 0x0A;             /*ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
                            break;
                        case    0x12:                    /*data over-run/under-run*/
                            return_s = 0x05;             /*ｵｰﾊﾞｰ･ﾗﾝ･ｴﾗｰ*/
                            break;
                        case    0x13:                    /*unexpected bus free*/
                        case    0x14:                    /*target bus phase sequence failure*/
                            return_s = 0x04;             /*装置に異常があった*/
                            break;
                        }
                    break;
                }
            break;
        case    0x02:                                    /*check status*/
            switch(srb.u_srb.escsirq.scsics[srb.u_srb.escsirq.scsicdb_l+12])
                                                         /*ｾﾝｽ･ｺｰﾄﾞ*/
                {
                case    0x00:                            /*no additional sense code*/
                case    0x17:                            /*recovered read data with target's read retries*/
                    return_s = 0x00;                     /*正常終了*/
                    break;
                case    0x18:                            /*recovered read data with target's ecc correction*/
                case    0x1E:                            /*recovered id with target's ecc correction*/
                    return_s = 0x01;                     /*エラー訂正されたデータをリードした*/
                    break;
                case    0x27:                            /*write protected*/
                    return_s = 0x02;                     /*ﾗｲﾄ･ﾌﾟﾛﾃｸﾄ*/
                    break;
                case    0X21:                            /*illegal block address*/
                    return_s = 0x03;                     /*最大ﾌﾞﾛｯｸ･ｱﾄﾞﾚｽを越えてアクセスしようとした*/
                    break;
                case    0x1A:                            /*parameter overrun*/
                case    0x97:                            /*parameter overrun*/
                    return_s = 0x05;                     /*ｵｰﾊﾞｰ･ﾗﾝ･ｴﾗｰ*/
                    break;
                case    0x04:                            /*parameter overrun*/
                case    0xB0:                            /*parameter overrun*/
                case    0xB1:                            /*parameter overrun*/
                case    0xB2:                            /*drive not ready*/
                    return_s = 0x06;                     /*ﾉｯﾄ･ﾚﾃﾞｨ*/
                    break;
```

```
            case    0x03:                       /*write fault*/
            case    0x09:                       /*track following error*/
                return_s = 0x07;                /*書き込み不可能状態を検出した*/
                break;
            case    0x22:                       /*illegal function for device type*/
            case    0x24:                       /*illegal field in CDB*/
            case    0x25:                       /*invalid LUN*/
            case    0x26:                       /*invalid field in parameter list*/
                return_s = 0x08;                /*イリーガル・コマンド*/
                break;
            case    0x05:                       /*drive not select*/
            case    0x07:                       /*multiple drive select*/
            case    0x20:                       /*invalid command operation code*/
            case    0x40:                       /*ram failure*/
            case    0x41:                       /*data path diagnostic failure*/
            case    0x42:                       /*power on diagnostic failure*/
            case    0x49:                       /*illegal message*/
                return_s = 0x09;                /*一般的なエラー*/
                break;
            case    0x10:                       /*id crc or ecc error*/
                return_s = 0x0B;                /*データ・エラー（ＩＤフィールド）*/
                break;
            case    0x11:                       /*unrecovered read error of data blocks*/
            case    0x16:                       /*data synchronization mark error*/
                return_s = 0x0C;                /*データ・エラー（データ・フィールド）*/
                break;
            case    0x32:                       /*no defect spare location available*/
                return_s = 0x0D;                /*代替えブロックが読めない*/
                break;
            case    0x14:                       /*no record found*/
                return_s = 0x0E;                /*目的のブロックが見つからない*/
                break;
            case    0x02:                       /*no seek complete*/
            case    0x06:                       /*no track zero found*/
            case    0x15:                       /*seek positioning error*/
                return_s = 0x0F;                /*シーク・エラー*/
                break;
            case    0x12:                       /*no address mark found in id field*/
                return_s = 0x10;                /*アドレス・マークが見つからない（ＩＤフィールド）*/
                break;
            case    0x13:                       /*no address mark found in data field*/
                return_s = 0x11;                /*アドレス・マークが見つからない（データ・フィールド）*/
                break;
            case    0x29:                       /*power on/reset/bus device reset occurred*/
            case    0x2A:                       /*mode select parameters changed*/
                return_s = 0x12;                /*モード・セレクト・データが変更された*/
                break;
            case    0x31:                       /*medium format corrupted*/
                return_s = 0x13;                /*フォーマット・エラー*/
                break;
            default:
                return_s = 0x04;                /*装置に異常があった*/
                break;
            }
    }
return;
}
```

保守／廃止

```
;************************************************************************
;*  ＡＳＰＩマネージャ(ｽﾀｰﾄ･ｱｯﾌﾟ･ﾙｰﾁﾝ)      ＡＳＳＵＰ．ＡＳＭ          *
;************************************************************************
;
;
_TEXT SEGMENT DWORD PUBLIC USE16 'CODE'
_TEXT ENDS
;
_DATA SEGMENT DWORD PUBLIC USE16 'DATA'
_DATA ENDS
;
_BSS SEGMENT DWORD PUBLIC USE16 'BSS'
_BSS ENDS
;
_PEND SEGMENT DWORD PUBLIC USE16 'PEND'
_PEND ENDS
;
DGROUP GROUP _TEXT, _DATA, _BSS, _PEND
    ASSUME  CS:_TEXT, DS:DGROUP, SS:DGROUP, ES:DGROUP
;
;
_DATA SEGMENT DWORD PUBLIC USE16 'DATA'
;
DGROUP@     dw  ?
_packet     dd  ?                       ;コマンド・パケットのポインタ領域
stacksave   dd  ?                       ;スタック・ポインタのセーブ領域
_prog_end   dd  ?                       ;プログラム終了アドレスのセーブ領域
        db  1024*4 dup(?)
stacktop    label   byte                ;スタック先頭アドレス
_DATA ENDS
;
_PEND SEGMENT DWORD PUBLIC USE16 'PEND'
progend     label   word                ;プログラム終了アドレス
_PEND ENDS
;
public  _packet                         ;コマンド・パケットのポインタ領域
public  _prog_end                       ;プログラム終了アドレスのセーブ領域
public  DGROUP@
;
_TEXT SEGMENT DWORD PUBLIC USE16 'CODE'
;
EXTRN   _AS_IO_REQ:PROC                 ;Ｉ／Ｏリクエスト・コマンド処理
;
org 0
;
;************************************************************************
;*                      デバイス・ヘッダの構成                          *
;************************************************************************
;
header  label   word
    dd  -1                              ;次のデバイスへのポインタ
    dw  0C000h                          ;デバイスの属性
    dw  STRATEGY                        ;ストラテジ・エントリ・ポイント
    dw  INTERRUPT                       ;割り込みエントリ・ポイント
    db  'SCSIMGR$'                      ;ＳＣＳＩマネージャ
;
;************************************************************************
;*                          ストラテジ・ルーチン                        *
;*                                                  入力：bx,es         *
;*コマンド・パケットのポインタを格納する。          出力：packet        *
;************************************************************************
;
STRATEGY    proc    far
;
    mov word ptr cs:[_packet],bx
    mov word ptr cs:[_packet+2],es
    ret
;
STRATEGY    endp
```

保守／廃止

```
;*****************************************************************************
;*                                割り込みルーチン                           *
;*                                                    入力：なし             *
;*ＡＳＰＩマネージャの入口と出口。                    出力：prog_end         *
;*この処理の中でＩ／Ｏリクエスト・コマンド処理をＣＡＬＬする。              *
;*****************************************************************************
;
INTERRUPT   proc    far
;
    push    ax                          ;レジスタ退避
    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    bp
    push    ds
    push    es
;
    mov ax,cs                           ;ｄｓ　＝　ｃｓ
    mov ds,ax
    mov cs:DGROUP@,ax
    mov word ptr [stacksave],sp         ;ＭＳ－ＤＯＳのスタック・ポインタの退避
    mov word ptr [stacksave+2],ss
;
    cli
    mov ss,ax                           ;ＡＳＰＩマネージャのローカル・スタックの確保
    mov sp,offset DGROUP:stacktop
    sti
;
                                        ;プログラムの終了アドレスをprog_endにセット
    mov word ptr [_prog_end],offset DGROUP:progend
    mov word ptr [_prog_end+2],cs
    call    _AS_IO_REQ                  ;Ｉ／Ｏリクエスト・コマンド処理


    cli
    mov sp,word ptr [stacksave]         ;ＭＳ－ＤＯＳのスタック・ポインタの復帰
    mov ss,word ptr [stacksave+2]
    sti
;
    pop es                              ;レジスタ復帰
    pop ds
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
;
INTERRUPT   endp

_TEXT ENDS
;
END
```

A

保守／廃止

```
/*****************************************************************************
 * ＡＳＰＩマネージャ・デバイス・ドライバ（I/Oﾘｸｴｽﾄ･ｺﾏﾝﾄﾞ処理）　ＡＳ＿ＲＥＱ．ＣＰＰ*
 *****************************************************************************/


#include <dos.h>


    /*  コマンド・パケット・ステータス情報  */

#define     ST_ERROR        0x8000                  /*ERRORビット*/
#define     ST_DONE         0x0100                  /*DONEビット*/
#define     ST_ERR_BADCMD      3                    /*コマンド番号が間違っている*/
#define     ST_ERR_GENERAL    12                    /*その他のエラー*/

    /*  コマンド・パケットの構造体  */

    struct command_packet
        {
        char command_length;                        /*コマンド・パケットのバイト数*/
        char unit_number;                           /*論理装置コード*/
        char cmd;                                   /*Ｉ／Ｏリクエスト・コマンド・コード*/
        int  command_status;                        /*ステータス*/
        long resv1,resv2;                           /*予約域　（８バイト*/

        union
            {

            /*  リード，ライトのコマンド・パケット  */

            struct
                {
                unsigned char mediarw;              /*ＤＰＢへのﾒﾃﾞｨｱ･ﾃﾞｨｽｸｨﾌﾟﾀ*/
                unsigned long far *transrw;         /*転送アドレス*/
                unsigned int sector_count;          /*セクタ・カウント*/
                unsigned int start_sector;
                unsigned long id_p;                 /*ﾎﾞﾘｭｰﾑIDのﾎﾟｲﾝﾀ*/
                unsigned long start_32;             /*開始セクタ番号*/
                } rw;

            /*  イニシャライズのコマンド・パケット  */

            struct
                {
                unsigned char unit;                 /*ユニット数*/
                unsigned char far *break_add;       /*ブレーク・アドレス*/
                unsigned char far *bpb_di;          /*ＢＰＢの配列ポインタ*/
                } in;
            } up;
        };


extern struct command_packet far *packet;           /*コマンド・パケットのポインタ*/

extern unsigned char far *prog_end;                 /*プログラム・エンド値 */
extern unsigned char i_id;                          /*ｲﾆｼｴｰﾀのID番号*/
extern unsigned int scsi_add;                       /*μPD72611のI/Oｱﾄﾞﾚｽ*/
extern unsigned char intp;                          /*割り込みポート*/
extern unsigned char cha;                           /*DMAｺﾝﾄﾛｰﾗのﾁｬﾈﾙ番号*/

extern far void ASPI(void);                         /*ASPI処理*/
extern void D_INITIAL(void);                        /*デバイスの初期化処理*/
extern void V_INITIAL(void);                        /*変数の初期化処理*/
extern void T_R(void);                              /*ターゲット・レディ処理*/

extern "C" void AS_IO_REQ(void);                    /*I/Oﾘｸｴｽﾄ処理*/

void INITIAL(void);                                 /*ｲﾆｼｬﾗｲｽﾞ 処理*/
void IOCTL_IN(void);                                /*IOCTLｲﾝﾌﾟｯﾄ処理*/
void ERROR_IO(void);                                /*IOﾘｸｴｽﾄ･ｴﾗｰ処理*/
```

保守／廃止

```
/*************************************************************************
 *                    Ｉ／Ｏリクエスト・コマンド処理                     *
 *I/Oﾘｸｴｽﾄ･ｺﾏﾝﾄﾞ･ｺｰﾄﾞに従って処理をＣＡＬＬする。                      *
 *                                                                       *
 *      入力：packet                                                     *
 *      出力：なし                                                       *
 *************************************************************************/


void AS_IO_REQ(void)
    {
    switch(packet->cmd)
        {
        case    0x00:
            INITIAL();                              /*イニシャライズ処理*/
            break;
        case    0x03:
            IOCTL_IN();                             /*ＩＯＣＴＬインプット処理*/
            break;
        default:
            ERROR_IO();                             /*IOﾘｸｴｽﾄ･ｴﾗｰ処理*/
        }
    return;
    }


/*************************************************************************
 *                          イニシャライズト処理                         *
 *μPD72611,割り込みｺﾝﾄﾛｰﾗのｲﾆｼｬﾗｲｽﾞおよびﾀｰｹﾞｯﾄをレディ状態にする       *
 *                                                                       *
 *      入力：packet,prog_end                                            *
 *      出力：packet                                                     *
 *************************************************************************/


void INITIAL(void)
    {
    unsigned char cong[4][20];                      /*DEVICEｺﾏﾝﾄﾞのﾊﾟﾗﾒｰﾀ文字列*/
    unsigned char c;
    unsigned char n;
    unsigned int buff;
    unsigned char far *p;

    static unsigned int w[30] = {'Ｓ','Ｃ','Ｓ','Ｉ','マ','ネ','ー','ジ','ャ','を','組','込','み','ま','し','た',
¥x0d¥x0a','$'};

    _AH = 0X09;                                     /*"SCSIﾏﾈｰｼﾞｬを組込みました"と表示する*/
    _DX= (int)w;
    geninterrupt(0x21);

    p = packet->up.in.bpb_di;                       /*例に示すようなCONFIG.SYS内のﾃﾞﾊﾞｲｽ･ﾄﾞﾗｲﾊﾞ名に続く*/
    n = 0;                                          /*ﾊﾟﾗﾒｰﾀ文字列をcong[][]にセットする*/
    do                                              /* (例) DEVICE = ASPIDD.SYS /C=3 /I=7 /P=A /A=1000 */
        {
        if(*p == '/')                               /*C:DMAﾁｬﾈﾙ*/
            {                                       /*I:ｲﾆｼｴｰﾀのSCSI ID番号*/
            ++p;                                    /*P:割り込みﾚﾍﾞﾙ*/
            c = 0;                                  /*μPD72611のﾍﾞｰｽ･ｱﾄﾞﾚｽ*/
            do
                {
                cong[n][c] = *p;
                ++p;
                ++c;
                }
            while((*p != '/')&&(*p != 0x0D)&&(*p != 0x0A));
            ++n;
            }
        else
            ++p;

        }
    while((*p != 0x0D)&&(*p != 0x0A));
```

A

```
    scsi_add = 0;
    n = 0;
    while(n < 4)                                    /*cong[][]の内容をcha,i_id,intp,scsi_addにセット*/
        {
        switch(cong[n][0])
            {
            case    'C':                            /*chaにDMAﾁｬﾈﾙをセット*/
                cha = cong[n][2]-0x30;
                break;
            case    'I':                            /*i_idにｲﾆｼｴｰﾀのSCSI ID番号をセット*/
                i_id = cong[n][2]-0x30;
                break;
            case    'P':                            /*intpに割り込みﾚﾍﾞﾙをセット*/
                if(cong[n][2] < 'A')
                    intp = cong[n][2]-0x30;
                else
                    intp = cong[n][2]-0x41+0x0A;
                break;
            case    'A':                            /*scsi_addにμPD72611のﾍﾞｰｽ･ｱﾄﾞﾚｽをセット*/
                c = 0;
                while(c < 4)
                    {
                    if(cong[n][c+2] < 'A')
                        buff = (unsigned int)cong[n][c+2]-0x30;
                    else
                        buff = (unsigned int)cong[n][c+2]-0x41+0x0A;
                    scsi_add = scsi_add | (buff<<(12-(c*4)));
                    ++c;
                    }
                break;
            }
        ++n;
        }

    V_INITIAL();                                    /*変数の初期化*/
    D_INITIAL();                                    /*デバイスの初期化*/
    T_R();                                          /*ターゲット・レディ処理*/


    packet->up.in.unit = 1;                         /*ユニット数をセット*/
    packet->up.in.break_add = prog_end;             /*ブレーク・アドレスをセット*/
    packet->command_status = ST_DONE;               /*ＤＯＮＥビットをセット*/

    return;
    }



/************************************************************************
 *                      ＩＯＣＴＬインプット処理                        *
 *ＭＳ－ＤＯＳにASPIｴﾝﾄﾘ･ﾎﾟｲﾝﾄを返す。                                *
 *                                                                      *
 *      入力：packet                                                    *
 *      出力：packet                                                    *
 ************************************************************************/


void IOCTL_IN(void)
    {
    if(packet->up.rw.sector_count == 4)
        {

        *packet->up.rw.transrw = (unsigned long)ASPI;   /*ASPIのｴﾝﾄﾘ･ﾎﾟｲﾝﾄをｾｯﾄ*/
        packet->command_status = ST_DONE;               /*ＤＯＮＥビットをセット*/
        }
    else
        {
                                                        /*その他のエラーをセット*/
        packet->command_status = ST_DONE|ST_ERROR|ST_ERR_GENERAL;
        }
    return;
    }
```

```
/*************************************************************************
 *                        Ｉ／Ｏリクエスト・エラー処理                   *
 *ＭＳ－ＤＯＳに“Ｉ／Ｏリクエスト・コマンド・コードが間違っている”を返*
 *す。                                                                   *
 *       入力：なし                                                      *
 *       出力：packet                                                    *
 *************************************************************************/


void ERROR_IO(void)
    {
                                                        /*Ｉ／Ｏリクエスト・コマンド・コードが間違っている*/
    packet->command_status = (ST_ERROR|ST_DONE|ST_ERR_BADCMD);
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *     イニシャライズ          INITIAL.CPP                    *
 *************************************************************************/

#include <dos.h>

/*μPD72611の間接ﾚｼﾞｽﾀI/Oｱﾄﾞﾚｽ*/
#define     R_TMOD      0x10                    /*転送ﾓｰﾄﾞ･ﾚｼﾞｽﾀ*/
#define     R_EXMOD     0x16                    /*ｴｸｽﾃﾝﾃﾞｨﾄﾞ･ﾓｰﾄﾞ･ﾚｼﾞｽﾀ*/
#define     R_BFTOUT    0x20                    /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ･ﾚｼﾞｽﾀ*/
#define     R_SRTOUT    0x21                    /*ｾﾚｸｼｮﾝ･ﾀｲﾑ･ｱｳﾄ･ﾚｼﾞｽﾀ*/
#define     R_RATOUT    0x22                    /*REQ/ACKﾀｲﾑ･ｱｳﾄ･ﾚｼﾞｽﾀ*/
#define     R_MOD       0x24                    /*ﾓｰﾄﾞ･ﾚｼﾞｽﾀ*/
#define     R_PID       0x25                    /*物理IDﾚｼﾞｽﾀ*/

/*割り込みｺﾝﾄﾛｰﾗI/Oｱﾄﾞﾚｽ*/
#define     R1_ICW1     0x0020                  /*ｺﾝﾄﾛｰﾙ1 ICW1ﾚｼﾞｽﾀ*/
#define     R1_ICW2     0x0021                  /*ｺﾝﾄﾛｰﾙ1 ICW2ﾚｼﾞｽﾀ*/
#define     R1_ICW3     0x0021                  /*ｺﾝﾄﾛｰﾙ1 ICW3ﾚｼﾞｽﾀ*/
#define     R1_ICW4     0x0021                  /*ｺﾝﾄﾛｰﾙ1 ICW4ﾚｼﾞｽﾀ*/
#define     R1_INTM     0x0021                  /*ｺﾝﾄﾛｰﾙ1割り込みﾏｽｸ･ﾚｼﾞｽﾀ*/
#define     R1_ELC      0x04D0                  /*ｺﾝﾄﾛｰﾙ1ｴｯｼﾞ/ﾚﾍﾞﾙ･ﾚｼﾞｽﾀ*/
#define     R2_ICW1     0x00A0                  /*ｺﾝﾄﾛｰﾙ2 ICW1ﾚｼﾞｽﾀ*/
#define     R2_ICW2     0x00A1                  /*ｺﾝﾄﾛｰﾙ2 ICW2ﾚｼﾞｽﾀ*/
#define     R2_ICW3     0x00A1                  /*ｺﾝﾄﾛｰﾙ2 ICW3ﾚｼﾞｽﾀ*/
#define     R2_ICW4     0x00A1                  /*ｺﾝﾄﾛｰﾙ2 ICW4ﾚｼﾞｽﾀ*/
#define     R2_INTM     0x00A1                  /*ｺﾝﾄﾛｰﾙ2割り込みﾏｽｸ･ﾚｼﾞｽﾀ*/
#define     R2_ELC      0x04D1                  /*ｺﾝﾄﾛｰﾙ2ｴｯｼﾞ/ﾚﾍﾞﾙ･ﾚｼﾞｽﾀ*/

#define     SYNCHRONOUS 0x01L                   /*SYNCHRONOUS DATA TRANSFER REQUEST*/
#define     IDENTIFY1   0xC0L                   /*IDENTIFY*/

extern  void S_PHASE(void);                     /*ﾌｪｰｽﾞ処理*/
extern void interrupt H_INT();                  /*ﾊｰﾄﾞｳｪｱ割り込みﾙｰﾁﾝ*/

extern  unsigned long sd_scsil[8];              /*転送ﾃﾞｰﾀ数*/
extern  unsigned char far *sd_scsip[8];         /*ｾｰﾌﾞ･ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ*/
extern  unsigned char sc_scsil[8];              /*転送ｺﾏﾝﾄﾞ数*/
extern  unsigned char far *sc_scsip[8];         /*ｾｰﾌﾞ･ｺﾏﾝﾄﾞscsiﾎﾟｲﾝﾀ*/
extern  unsigned char far *ss_scsip[8];         /*ｾｰﾌﾞ･ｽﾃｰﾀｽscsiﾎﾟｲﾝﾀ*/
extern  unsigned char p_end;                    /*S_PHASEの終了結果*/
extern  unsigned char t_id;                     /*ﾀｰｹﾞｯﾄ ID*/
extern  unsigned long message_o;                /*ﾀｰｹﾞｯﾄに転送するﾒｯｾｰｼﾞ*/
extern  unsigned char i_message;                /*ﾀｰｹﾞｯﾄがｻﾎﾟｰﾄしていないﾒｯｾｰｼﾞを示す*/
extern  unsigned char sync[8];                  /*転送ﾓｰﾄﾞ*/
extern  unsigned char expect_priod[8];          /*転送時間*/
extern  unsigned char expect_offset[8];         /*転送ｵﾌｾｯﾄ値*/

/*μPD72611直接ﾚｼﾞｽﾀ*/
unsigned int R_ADR;                             /*ｱﾄﾞﾚｽ･ﾚｼﾞｽﾀ*/
unsigned int R_CST;                             /*ｺﾝﾄﾛｰﾗ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ*/
unsigned int R_IST;                             /*割り込みｽﾃｰﾀｽ･ﾚｼﾞｽﾀ*/
unsigned int R_CMD;                             /*ｺﾏﾝﾄﾞ･ﾚｼﾞｽﾀ*/
unsigned int R_DID;                             /*ﾃﾞｨｽﾄﾈｰｼｮﾝIDﾚｼﾞｽﾀ*/
unsigned int R_WIN1;                            /*ｳｲﾝﾄﾞ1*/
unsigned int R_EXST;                            /*ｴｸｽﾃﾝﾃﾞｨﾄﾞ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ*/
unsigned int R_DF2;                             /*ﾃﾞｰﾀFIFO2ﾚｼﾞｽﾀ*/
```

```
/*DMACのI/Oｱﾄﾞﾚｽ*/
/*ﾓｰﾄﾞ･ﾚｼﾞｽﾀ*/
unsigned int    R_DMOD[8] = {0x000B, 0x000B, 0x000B, 0x000B,
                             0x00D6, 0x00D6, 0x00D6, 0x00D6};
/*拡張ﾓｰﾄﾞ ･ﾚｼﾞｽﾀ*/
unsigned int    R_DEMOD[8] = {0x040B, 0x040B, 0x040B, 0x040B,
                              0x04D6, 0x04D6, 0x04D6, 0x04D6};
/*ﾗｲﾄ･ｼﾝｸﾞﾙ･ﾏｽﾀ･ﾋﾞｯﾄ･ﾚｼﾞｽﾀ*/
unsigned int    R_DWSMB[8] = {0x000A, 0x000A, 0x000A, 0x000A,
                              0x00D4, 0x00D4, 0x00D4, 0x00D4};
/*ﾍﾞｰｽ8237ｺﾝﾊﾟﾁﾌﾞﾙ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀ*/
unsigned int    R_DCWC[8] = {0x0001, 0x0003, 0x0005, 0x0007,
                             0xFFFF, 0x00C6, 0x00CA, 0x00CE};
/*ﾍﾞｰｽ･ﾊｲ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀ*/
unsigned int    R_DHWC[8] = {0x0401, 0x0403, 0x0405, 0x0407,
                             0xFFFF, 0x04C6, 0x04CA, 0x04CE};
/*ﾍﾞｰｽ8237ｺﾝﾊﾟﾁﾌﾞﾙ･ｱﾄﾞﾚｽ･ﾚｼﾞｽﾀ*/
unsigned int    R_DBCA[8] = {0x0000, 0x0002, 0x0004, 0x0006,
                             0xFFFF, 0x00C4, 0x00C8, 0x00CC};
/*ﾍﾞｰｽ･ﾛｳ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀ*/
unsigned int    R_DBLP[8] = {0x0087, 0x0083, 0x0081, 0x0082,
                             0xFFFF, 0x008B, 0x0089, 0x008A};
/*ﾍﾞｰｽ･ﾊｲ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀ*/
unsigned int    R_DBHP[8] = {0x0487, 0x0483, 0x0481, 0x0482,
                             0xFFFF, 0x048B, 0x0489, 0x048A};
/*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
unsigned int    R_DCBP[8] = {0x000C, 0x000C, 0x000C, 0x000C,
                             0x00D8, 0x00D8, 0x00D8, 0x00D8};
/*ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ*/
unsigned int    R_DSTA[8] = {0x0008, 0x0008, 0x0008, 0x0008,
                             0x00D0, 0x00D0, 0x00D0, 0x00D0};

unsigned char i_id;                                  /*ｲﾆｼｴｰﾀのID番号*/
unsigned int scsi_add;                               /*μPD72611のI/Oｱﾄﾞﾚｽ*/
unsigned char intp;                                  /*割り込みポート*/
unsigned char cha;                                   /*DMAｺﾝﾄﾛｰﾗのﾁｬﾈﾙ番号*/


void D_INITIAL(void);                                /*デバイスの初期化処理*/
void V_INITIAL(void);                                /*変数の初期化処理*/
void T_R(void);                                      /*ターゲット・レディ処理*/
```

保守／廃止

```
/*************************************************************************
 *                          デバイスの初期化処理                          *
 *  デバイスの初期化を実行する                                            *
 *                                                                        *
 *      入力：i_id,scsi_add,intp,cha                                      *
 *      出力：なし                                                        *
 *************************************************************************/

void D_INITIAL(void)
    {
    unsigned char b;                                    /*I/Oﾘｰﾄﾞのためのﾊﾞｯﾌｧ*/
    union                                               /*割り込みﾍﾞｸﾀをｾｯﾄするための共用体*/
        {
        unsigned long vec;
        struct
            {
            unsigned int vec_off;
            unsigned int vec_seg;
            }   int_v;
        }   u_v;

    R_ADR = scsi_add + 0x03;                            /*μPD72611の直接ﾚｼﾞｽﾀのI/Oｱﾄﾞﾚｽをscsi_addによって設定*/
    R_CST = scsi_add + 0x02;
    R_IST = scsi_add + 0x07;
    R_CMD = scsi_add + 0x07;
    R_DID = scsi_add + 0x06;
    R_WIN1 = scsi_add + 0x04;
    R_EXST = scsi_add + 0x08;
    R_DF2 = scsi_add + 0x0C;

    b = inportb(R_IST);                                 /*μPD72611のﾘｾｯﾄ割り込み要因のｸﾘｱ*/

    u_v.vec = (unsigned long)H_INT;

    if(intp < 8)
        {
        b = inportb(R1_ICW4);                           /*割り込みｺﾝﾄﾛｰﾗ1・ﾏｽｸ・ﾚｼﾞｽﾀ・ﾘｰﾄﾞ*/
        b = b | (0x01<<intp);
        outportb(R1_ICW4,b);                            /*INTﾎﾟｰﾄ・ﾏｽｸ*/
                                                        /*割り込みﾍﾞｸﾀ・ｾｯﾄ*/
        poke(0x0000,0x0020+(unsigned int)(intp*4),u_v.int_v.vec_off);
        poke(0x0000,0x0020+(unsigned int)((intp*4)+2),u_v.int_v.vec_seg);
        b = inportb(R1_ELC);
        b = b & ((0x01<<intp)^0xFF);                    /*INTﾎﾟｰﾄをｴｯｼﾞ・ﾄﾘｶﾞにｾｯﾄ*/
        outportb(R1_ELC,b);
        }
    else
        {
        b = inportb(R2_ICW4);                           /*割り込みｺﾝﾄﾛｰﾗ2・ﾏｽｸ・ﾚｼﾞｽﾀ・ﾘｰﾄﾞ*/
        b = b | (0x01<<(intp&0x07));
        outportb(R2_ICW4,b);                            /*INTﾎﾟｰﾄ・ﾏｽｸ*/
                                                        /*割り込みﾍﾞｸﾀ・ｾｯﾄ*/
        poke(0x0000,0x01C0+(unsigned int)((intp&0x07)*4),u_v.int_v.vec_off);
        poke(0x0000,0x01C0+(unsigned int)(((intp&0x07)*4)+2),u_v.int_v.vec_seg);
        b = inportb(R1_ELC);
        b = b & ((0x01<<(intp&0x07))^0xFF);              /*INTﾎﾟｰﾄをｴｯｼﾞ・ﾄﾘｶﾞにｾｯﾄ*/
        outportb(R1_ELC,b);
        }
```

```
    outportb(R_DWSMB[cha], 0x04|(cha&0x03));                /*DMAﾁｬﾝﾙをﾏｽｸ*/

                                                            /*μPD72611のｲﾆｼｬﾗｲｽﾞ*/

    outportb(R_ADR, R_TMOD);                                /*非同期転送ﾓｰﾄﾞにｾｯﾄ*/
    outportb(R_WIN1, 0x00);
    outportb(R_ADR, R_EXMOD);                               /*ｷｭｰ･ﾀｸﾞ･ﾒｯｾｰｼﾞ･ﾉﾝ･ｻﾎﾟｰﾄ/ﾉﾝ･ﾊﾟﾘﾃｨ･ｽﾙｰ･ﾓｰﾄﾞ*/
    outportb(R_WIN1, 0x02);                                 /*ﾊﾟﾘﾃｨ･ｴﾗｰ続行ﾓｰﾄﾞ/ﾃﾞｨﾏﾝﾄﾞ･ﾓｰﾄﾞ*/
    outportb(R_ADR, R_BFTOUT);                              /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ*/
    outportb(R_WIN1, 0xFF);                                 /*FFH*/
    outportb(R_ADR, R_SRTOUT);                              /*ｾﾚｸﾄ･ﾀｲﾑ･ｱｳﾄ*/
    outportb(R_WIN1, 0xFF);                                 /*FFH*/
    outportb(R_ADR, R_RATOUT);                              /*REQ/ACKﾀｲﾑ･ｱｳﾄ*/
    outportb(R_WIN1, 0x00);                                 /*検出しない*/
    outportb(R_ADR, R_MOD);                                 /*ﾓｰﾄﾞ･ﾚｼﾞｽﾀ*/
    outportb(R_WIN1, 0xE2);
    outportb(R_ADR, R_PID);                                 /*ｲﾆｼｴｰﾀID*/
    outportb(R_WIN1, (0x80|i_id));

    if(intp < 8)
        {
        b = inportb(R1_ICW4);                               /*割り込みｺﾝﾄﾛｰﾗ1･ﾏｽｸ･ﾚｼﾞｽﾀ･ﾘｰﾄﾞ*/
        b = b & ((0x01<<intp)^0xFF);
        outportb(R1_ICW4, b);                               /*INTﾎﾟｰﾄ･ﾏｽｸ解除*/
        }
    else
        {
        b = inportb(R2_ICW4);                               /*割り込みｺﾝﾄﾛｰﾗ2･ﾏｽｸ･ﾚｼﾞｽﾀ･ﾘｰﾄﾞ*/
        b = b & ((0x01<<(intp&0x07))^0xFF);
        outportb(R2_ICW4, b);                               /*INTﾎﾟｰﾄ･ﾏｽｸ解除*/
        }
    return;
    }


/*************************************************************************
 *                          変数の初期化処理                             *
 *  ＳＣＳＩフェーズ処理(S_PHASE)の変数を初期化する                      *
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：i_message, expect_offset[], expect_priod[], sync[]         *
 *************************************************************************/

void V_INITIAL(void)
    {
    i_message = 0x00;
    expect_offset[0] = 0x08;
    expect_offset[1] = 0x08;
    expect_offset[2] = 0x08;
    expect_offset[3] = 0x08;
    expect_offset[4] = 0x08;
    expect_offset[5] = 0x08;
    expect_offset[6] = 0x08;
    expect_offset[7] = 0x08;
    expect_priod[0] = 0x19;
    expect_priod[1] = 0x19;
    expect_priod[2] = 0x19;
    expect_priod[3] = 0x19;
    expect_priod[4] = 0x19;
    expect_priod[5] = 0x19;
    expect_priod[6] = 0x19;
    expect_priod[7] = 0x19;
    sync[0] = 0x00;
    sync[1] = 0x00;
    sync[2] = 0x00;
    sync[3] = 0x00;
    sync[4] = 0x00;
    sync[5] = 0x00;
    sync[6] = 0x00;
    sync[7] = 0x00;

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                      ターゲット・レディ処理                           *
 *  接続されているターゲット(ﾛｼﾞｶﾙ･ﾕﾆｯﾄを含める)をレディ状態にする       *
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：なし                                                       *
 *************************************************************************/

void T_R(void)
    {
    unsigned char tr_c[6] = {0x00,0x00,0x00,0x00,0x00,0x00};    /*ﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨ･ｺﾏﾝﾄﾞ*/
    unsigned char rqs_c[6] = {0x03,0x00,0x00,0x00,0x0D,0x00};   /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ*/
    unsigned char rqs_d[13];                                    /*ﾘｸｴｽﾄ･ｾﾝｽ･ﾃﾞｰﾀ領域*/
    unsigned char s;                                            /*SCSIｽﾃｰﾀｽ領域*/
    unsigned char l;                                            /*ﾛｼﾞｶﾙ･ﾕﾆｯﾄ･ﾅﾝﾊﾞ*/

    t_id = 0x00;
    while(t_id < 0x08)                                          /*各ﾀｰｹﾞｯﾄをレディ状態にする*/
        {
        if(t_id != i_id)                                        /*ﾀｰｹﾞｯﾄ ID != ｲﾆｼｴｰﾀ ID？*/
            {
            l = 0x00;
            while(l < 0x08)                                     /*各ﾛｼﾞｶﾙ･ﾕﾆｯﾄをレディ状態にする*/
                {
                if(l == 0x00)
                    message_o = SYNCHRONOUS;                    /*同期転送ﾒｯｾｰｼﾞ*/
                else
                    {
                    message_o = IDENTIFY1;                      /*ｱｲﾃﾞﾝﾃｨﾌｧｲ･ﾒｯｾｰｼﾞ*/
                    message_o = message_o | l;
                    }
                s = 0x00;                                       /*ﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨ･ｺﾏﾝﾄﾞ実行*/
                tr_c[1] = l << 5;
                sc_scsil[t_id] = 0x06;
                sc_scsip[t_id] = tr_c;
                ss_scsip[t_id] = &s;
                S_PHASE();                                      /*SCSIﾌｪｰｽﾞ処理をCALL*/

                if(p_end == 0x01)                               /*SCSIﾌｪｰｽﾞ処理正常終了か？*/
                    {
                    if(s != 0x00)                               /*SCSIｽﾃｰﾀｽがｸﾞｯﾄﾞか？*/
                        {
                        message_o = IDENTIFY1;                  /*ｱｲﾃﾞﾝﾃｨﾌｧｲ･ﾒｯｾｰｼﾞ*/
                        message_o = message_o | l;
                        s = 0x00;                               /*ﾚｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ実行*/
                        rqs_c[1] = l << 5;
                        sc_scsil[t_id] = 0x06;
                        sc_scsip[t_id] = rqs_c;
                        sd_scsil[t_id] = 0x0D;
                        sd_scsip[t_id] = rqs_d;
                        ss_scsip[t_id] = &s;
                        S_PHASE();                              /*SCSIﾌｪｰｽﾞ処理をCALL*/
```

```
                                                    /*ﾉｯﾄ･ﾚﾃﾞｨか？*/
                if((p_end==0x01)&&(s==0x00)&&(rqs_d[12]==0x29))
                    {
                    message_o = IDENTIFY1;          /*ｱｲﾃﾞﾝﾃｨﾌｧｲ･ﾒｯｾｰｼﾞ*/
                    message_o = message_o | 1;
                    s = 0x00;                       /*ﾃｽﾄ･ﾕﾆｯﾄ･ﾚﾃﾞｨ･ｺﾏﾝﾄﾞ実行*/
                    tr_c[1] = 1 << 5;
                    sc_scsil[t_id] = 0x06;
                    sc_scsip[t_id] = tr_c;
                    ss_scsip[t_id] = &s;
                    S_PHASE();                      /*SCSIﾌｪｰｽﾞ処理をCALL*/
                    }
                else
                    break;
                }
            ++l;
            }
        else
            break;
        }
    }
    ++t_id;
  }
return;
}
```

保守／廃止

```
/*****************************************************************************
 *          ASPI            (ASPI.CPP)                                       *
 *****************************************************************************/


/*変数p_end*/
#define     S_NEND      0x01                    /*S_PHASE正常終了*/
#define     S_RESETC    0x02                    /*SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ終了*/
#define     BFTOUT      0x03                    /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ終了*/
#define     STOUT       0x04                    /*ｾﾚｸｼｮﾝ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ終了*/
#define     P_EEND      0x05                    /*ﾌｪｰｽﾞ･ｴﾗｰ終了*/

/*error*/
#define     INITDE_F        0x01                /*INITIATOR_DEﾌﾗｸﾞ*/
#define     MESSAGEPE_F     0x02                /*MESSAGE_PEﾌﾗｸﾞ*/
#define     IDENTIFY_F      0x04                /*IDENTIFYﾌﾗｸﾞ*/
#define     ABORT_F         0x08                /*ABORTﾌﾗｸﾞ*/
#define     BUSDR_F         0x10                /*BUS_DRﾌﾗｸﾞ*/
#define     FIFOE_F         0x20                /*FIFO ERRORﾌﾗｸﾞ*/
#define     SYNCOF_F        0x40                /*SYNC OFFSET ERRORﾌﾗｸﾞ*/
#define     PERROR_F        0x80                /*PARITY ERRORﾌﾗｸﾞ*/

/*message*/
#define     COMMAND_CP  0x00L                   /*COMMAND COMPLETE*/
#define     EXTEND_ME   0x01L                   /*拡張ﾒｯｾｰｼﾞ*/
#define     SYNCHRONOUS 0x01L                   /*SYNCHRONOUS DATA TRANSFER REQUEST*/
#define     SAVE_DP     0x02L                   /*SAVE DATA POINTER*/
#define     RESTORE_P   0x03L                   /*RESTORE POINTER*/
#define     DISCON      0x04L                   /*DISCONNECT*/
#define     I_DETECTED  0x05L                   /*INITIATOR DETECTED ERROR*/
#define     ABORT       0X06L                   /*ABORT*/
#define     MESS_REJ    0x07L                   /*MESSAGE REJECT*/
#define     NO_OPE      0x08L                   /*NO OPERATION*/
#define     MESS_PARE   0x09L                   /*MESSAGE PARITY ERROR*/
#define     BUS_D_R     0x0CL                   /*BUS DEVICE RESET*/
#define     IDENTIFY1   0xC0L                   /*IDENTIFY*/

/*p_error*/
#define     DOPE_F          0x01                /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     DIPE_F          0x02                /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     COMPE_F         0x04                /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     STAPE_F         0x08                /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     DISCON_F        0x10                /*ﾃﾞｨｽｺﾈｸﾃｨｯﾄﾞ*/
```

保守／廃止

```
struct aspi_t                                   /*scsi request block(SRB)*/
    {
    unsigned char command_code;                 /*command code              (w)*/
    unsigned char status;                       /*status                    (r)*/
    unsigned char h_adapter_n;                  /*host adapter number       (w)*/
    unsigned char scsi_request_f;               /*scsi request flags        (w)*/
    unsigned long reserved;                     /*reserved                    */

    union {
        struct
            {
            unsigned char n_host_a;             /*number of host adapters   (r)*/
            unsigned char id_host_a;            /*id of host adapter        (r)*/
            unsigned char scsi_m_id[16];        /*scsi manager id           (r)*/
            unsigned char h_adapter_id[16];     /*host adapter id           (r)*/
            unsigned char h_adapterp[16];       /*host adapter parameters   (r)*/
            }   hadap;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned char p_device_t;           /*peripheral device type    (r)*/
            }   gdevice;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned long data_a_l;             /*data length               (w)*/
            unsigned char sense_a_l;            /*sense length              (w)*/
            unsigned char far *data_bp;         /*data buff pointer         (w)*/
            unsigned char far *srb_lp;          /*srb link pointer          (w)*/
            unsigned char scsicdb_l;            /*scsi cdb length           (w)*/
            unsigned char h_adapter_s;          /*host adapter status       (r)*/
            unsigned char target_status;        /*target status             (r)*/
            unsigned char far *post_ra;         /*post routine address      (w)*/
            unsigned char warkspace[34];        /*aspi warkspace              */
            unsigned char scsics[256];          /*scsi cdb and sense data (w/r)*/
            }   escsirq;
        struct
            {
            unsigned char target_id;            /*target id                 (w)*/
            unsigned char lun;                  /*lun                       (w)*/
            unsigned char reserved[14];         /*reserved                    */
            unsigned char h_adapter_s;          /*host adapter status       (r)*/
            unsigned char target_status;        /*target status             (r)*/
            unsigned char far *post_ra;         /*post routine address      (w)*/
            unsigned char warkspace[34];        /*aspi warkspace              */
            }   resetsd;
        }   u_srb;
    };

struct aspi_t far *srb_p;                       /*SRB ﾎﾟｲﾝﾀ*/

union long_s                                    /*ｽﾀｯｸ領域からSRBﾎﾟｲﾝﾀを引き取るための共用体*/
    {
    unsigned long srb;
    struct int_s
        {
        unsigned int srb_l;
        unsigned int srb_h;
        }s_b;
    }s_u;

extern  void S_PHASE(void);                     /*ﾌｪｰｽﾞ処理関数*/
extern  unsigned long sd_scsil[8];              /*転送ﾃﾞｰﾀ数*/
extern  unsigned char far *sd_scsip[8];         /*ｾｰﾌﾞ･ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ*/
extern  unsigned char sc_scsil[8];              /*転送ｺﾏﾝﾄﾞ数*/
extern  unsigned char far *sc_scsip[8];         /*ｾｰﾌﾞ･ｺﾏﾝﾄﾞscsiﾎﾟｲﾝﾀ*/
extern  unsigned char far *ss_scsip[8];         /*ｾｰﾌﾞ･ｽﾃｰﾀｽscsiﾎﾟｲﾝﾀ*/
extern  unsigned char t_id;                     /*ﾀｰｹﾞｯﾄ ID*/
extern  unsigned long message_o;                /*ﾀｰｹﾞｯﾄに転送するﾒｯｾｰｼﾞ*/
extern  unsigned char p_end;                    /*S_PHASEの終了結果*/
extern  unsigned char p_error;                  /*ﾌｪｰｽﾞ･ｴﾗｰのﾌｪｰｽﾞを示す*/
extern  unsigned char error;                    /*ｴﾗｰを示す*/
```

A

```
extern  unsigned char sync[8];                          /*転送ﾓｰﾄﾞ*/
extern  unsigned char i_id;                             /*ｲﾆｼｴｰﾀのID番号*/

void far ASPI(void);                                    /*aspi*/
void    HADAPTER_I(void);                               /*host adapter inquiryｺﾏﾝﾄﾞ*/
void    G_DEVICET(void);                                /*get device typeｺﾏﾝﾄﾞ*/
void    ESCSI_R(void);                                  /*execute scsi requestｺﾏﾝﾄﾞ*/
void    RESET_D(void);                                  /*reset scsi deviceｺﾏﾝﾄﾞ*/


/*************************************************************************
 *                      ＡＳＰＩ処理                                     *
 *  ＳＲＢのcommand codeを参照し各モジュールをＣＡＬＬする               *
 *                                                                       *
 *      入力：SRB                                                        *
 *      出力：SRB                                                        *
 *************************************************************************/


void far ASPI(void)
    {
    asm push bp;                                        /*レジスタ退避*/
    asm push ax;
    asm push ds
    asm mov bp,sp;                                      /*BPﾚｼﾞｽﾀ ← SP*/
    asm mov ax,cs;
    asm mov ds,ax;
    asm mov ax,ss:[bp+16];
    asm mov s_u.s_b.srb_l,ax;                           /*SRBﾎﾟｲﾝﾀをｽﾀｯｸ領域から引き取る*/
    asm mov ax,ss:[bp+18];
    asm mov s_u.s_b.srb_h,ax;
    asm push bx;                                        /*ﾚｼﾞｽﾀ退避*/
    asm push cx;
    asm push dx;
    asm push es;
    asm push si;
    asm push di;

    srb_p = (struct aspi_t far *)s_u.srb;

    if(srb_p->h_adapter_n != 0x00)
        srb_p->status = 0x81;                           /*invalid host adapter numberをセット*/
    else
        {
        switch(srb_p->command_code)
            {
            case    0x00:
                HADAPTER_I();                           /*host adapter inquiryｺﾏﾝﾄﾞ*/
                break;
            case    0x01:
                G_DEVICET();                            /*get device typeｺﾏﾝﾄﾞ*/
                break;
            case    0x02:
                ESCSI_R();                              /*execute scsi requestｺﾏﾝﾄﾞ*/
                break;
            case    0x04:
                RESET_D();                              /*reset scsi deviceｺﾏﾝﾄﾞ*/
                break;
            default:
                srb_p->status = 0x80;                   /*invalid scsi requestをセット*/
            }
        }
    asm pop di;                                         /*レジスタ復帰*/
    asm pop si;
    asm pop es;
    asm pop dx;
    asm pop cx;
    asm pop bx;
    asm pop ds;
    asm pop ax;
    asm pop bp;

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *            HOST ADAPTER INQUIRY処理                                   *
 *  ホスト・アダプタの情報をアプリケーション・ソフトに渡す                 *
 *                                                                       *
 *      入力：SRB                                                        *
 *      出力：SRB                                                        *
 *************************************************************************/


void HADAPTER_I(void)
    {
    srb_p->status = 0x01;                           /*scsi request completed without errorをセット*/
    srb_p->u_srb.hadap.n_host_a = 0x01;             /*ﾎｽﾄ･ｱﾀﾞﾌﾟﾀ数をセット*/
    srb_p->u_srb.hadap.id_host_a = i_id;            /*ﾎｽﾄ･ｱﾀﾞﾌﾟﾀのSCSI ID番号をセット*/
    srb_p->u_srb.hadap.scsi_m_id[0] = '¥';          /*ASPIﾏﾈｰｼﾞｬのID(ASPI_M 1.0)をセット*/
    srb_p->u_srb.hadap.scsi_m_id[1] = 'A';
    srb_p->u_srb.hadap.scsi_m_id[2] = 'S';
    srb_p->u_srb.hadap.scsi_m_id[3] = 'P';
    srb_p->u_srb.hadap.scsi_m_id[4] = 'I';
    srb_p->u_srb.hadap.scsi_m_id[5] = '_';
    srb_p->u_srb.hadap.scsi_m_id[6] = 'M';
    srb_p->u_srb.hadap.scsi_m_id[7] = ' ';
    srb_p->u_srb.hadap.scsi_m_id[8] = '1';
    srb_p->u_srb.hadap.scsi_m_id[9] = '.';
    srb_p->u_srb.hadap.scsi_m_id[10] = '0';
    srb_p->u_srb.hadap.scsi_m_id[11] = '¥';
    srb_p->u_srb.hadap.scsi_m_id[12] = 0x00;
    srb_p->u_srb.hadap.scsi_m_id[13] = 0x00;
    srb_p->u_srb.hadap.scsi_m_id[14] = 0x00;
    srb_p->u_srb.hadap.scsi_m_id[15] = 0x00;
    srb_p->u_srb.hadap.h_adapter_id[0] = '¥';       /*ﾎｽﾄ･ｱﾀﾞﾌﾟﾀのIDをセット*/
    srb_p->u_srb.hadap.h_adapter_id[1] = 'N';
    srb_p->u_srb.hadap.h_adapter_id[2] = 'E';
    srb_p->u_srb.hadap.h_adapter_id[3] = 'C';
    srb_p->u_srb.hadap.h_adapter_id[4] = ' ';
    srb_p->u_srb.hadap.h_adapter_id[5] = 'U';
    srb_p->u_srb.hadap.h_adapter_id[6] = 'P';
    srb_p->u_srb.hadap.h_adapter_id[7] = 'D';
    srb_p->u_srb.hadap.h_adapter_id[8] = '7';
    srb_p->u_srb.hadap.h_adapter_id[9] = '2';
    srb_p->u_srb.hadap.h_adapter_id[10] = '6';
    srb_p->u_srb.hadap.h_adapter_id[11] = '1';
    srb_p->u_srb.hadap.h_adapter_id[12] = '1';
    srb_p->u_srb.hadap.h_adapter_id[13] = '¥';
    srb_p->u_srb.hadap.h_adapter_id[14] = 0x00;
    srb_p->u_srb.hadap.h_adapter_id[15] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[0] = '¥';         /*ﾎｽﾄ･ｱﾀﾞﾌﾟﾀのﾊﾟﾗﾒｰﾀをセット*/
    srb_p->u_srb.hadap.h_adapterp[1] = 'N';
    srb_p->u_srb.hadap.h_adapterp[2] = 'O';
    srb_p->u_srb.hadap.h_adapterp[3] = 'T';
    srb_p->u_srb.hadap.h_adapterp[4] = ' ';
    srb_p->u_srb.hadap.h_adapterp[5] = 'D';
    srb_p->u_srb.hadap.h_adapterp[6] = 'A';
    srb_p->u_srb.hadap.h_adapterp[7] = 'T';
    srb_p->u_srb.hadap.h_adapterp[8] = 'A';
    srb_p->u_srb.hadap.h_adapterp[9] = '¥';
    srb_p->u_srb.hadap.h_adapterp[10] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[11] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[12] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[13] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[14] = 0x00;
    srb_p->u_srb.hadap.h_adapterp[15] = 0x00;

    return;
    }
```

A

保守／廃止

```
/*************************************************************************
*                GET  DEVICE  TYPE処理                                  *
*ターゲットの確認のためＩＮＱＵＩＲＹデータをリードしアプリケーション・*
*ソフトに渡す                                                           *
*      入力：SRB                                                        *
*      出力：SRB                                                        *
*************************************************************************/


void G_DEVICET(void)
    {
                                                    /*INQUIRYｺﾏﾝﾄﾞ*/
    unsigned char inquiry_c[6] = {0x12,0x00,0x00,0x00,0x05,0x00};
    unsigned char inquiry_d[5];                     /*INQUIRYﾃﾞｰﾀ*/
    unsigned char scsi_s;                           /*scsiｽﾃｰﾀｽ*/

    if(srb_p->u_srb.gdevice.target_id != i_id)
        {
        t_id = srb_p->u_srb.gdevice.target_id;      /*t_idをセット*/

        if(sync[t_id] == 0x00)                      /*転送モードは不定か？*/
            {
                                                    /*INQUIRYｺﾏﾝﾄﾞにLUNをセット*/
            inquiry_c[1] = srb_p->u_srb.gdevice.lun << 5;
            message_o = SYNCHRONOUS;                /*message_oにSYNCHRONOUS DATA TRANSFER REQUESTﾒｯｾｰｼﾞをｾｯﾄ*/
            sd_scsil[t_id] = 0x05L;                 /*INQUIRYﾃﾞｰﾀ数をセット*/
            sd_scsip[t_id] = inquiry_d;             /*INQUIRYﾃﾞｰﾀのﾎﾟｲﾝﾀをセット*/
            sc_scsil[t_id] = 0x06;                  /*INQUIRYｺﾏﾝﾄﾞ数をセット*/
            sc_scsip[t_id] = inquiry_c;             /*INQUIRYｺﾏﾝﾄﾞのﾎﾟｲﾝﾀをセット*/
            ss_scsip[t_id] = &scsi_s;               /*scsiｽﾃｰﾀｽのﾎﾟｲﾝﾀをセット*/

            S_PHASE();                              /*SCSIﾌｪｰｽﾞ処理*/

            if(sync[t_id] == 0x02)                  /*非同期モードか？*/
                {
                                                    /*INQUIRYｺﾏﾝﾄﾞにLUNをセット*/
                inquiry_c[1] = srb_p->u_srb.gdevice.lun << 5;
                message_o = IDENTIFY1;              /*message_oにIDENTIFYﾒｯｾｰｼﾞをｾｯﾄ*/
                message_o = message_o|(unsigned long)srb_p->u_srb.gdevice.lun;
                sd_scsil[t_id] = 0x05L;             /*INQUIRYﾃﾞｰﾀ数をセット*/
                sd_scsip[t_id] = inquiry_d;         /*INQUIRYﾃﾞｰﾀのﾎﾟｲﾝﾀをセット*/
                sc_scsil[t_id] = 0x06;              /*INQUIRYｺﾏﾝﾄﾞ数をセット*/
                sc_scsip[t_id] = inquiry_c;         /*INQUIRYｺﾏﾝﾄﾞのﾎﾟｲﾝﾀをセット*/
                ss_scsip[t_id] = &scsi_s;           /*scsiｽﾃｰﾀｽのﾎﾟｲﾝﾀをセット*/

                S_PHASE();                          /*SCSIﾌｪｰｽﾞ処理*/
                }
            }
        else
            {
                                                    /*INQUIRYｺﾏﾝﾄﾞにLUNをセット*/
            inquiry_c[1] = srb_p->u_srb.gdevice.lun << 5;
            message_o = IDENTIFY1;                  /*message_oにIDENTIFYﾒｯｾｰｼﾞをｾｯﾄ*/
            message_o = message_o|(unsigned long)srb_p->u_srb.gdevice.lun;
            sd_scsil[t_id] = 0x05L;                 /*INQUIRYﾃﾞｰﾀ数をセット*/
            sd_scsip[t_id] = inquiry_d;             /*INQUIRYﾃﾞｰﾀのﾎﾟｲﾝﾀをセット*/
            sc_scsil[t_id] = 0x06;                  /*INQUIRYｺﾏﾝﾄﾞ数をセット*/
            sc_scsip[t_id] = inquiry_c;             /*INQUIRYｺﾏﾝﾄﾞのﾎﾟｲﾝﾀをセット*/
            ss_scsip[t_id] = &scsi_s;               /*scsiｽﾃｰﾀｽのﾎﾟｲﾝﾀをセット*/

            S_PHASE();                              /*SCSIﾌｪｰｽﾞ処理*/
            }
```

```
        if(p_end == S_NEND && scsi_s == 0x00)              /*S_PHASE()正常終了 and scsiｽﾃｰﾀｽがgood？*/
            {
            srb_p->status = 0x01;                          /*scsi request completed without errorをセット*/
            srb_p->u_srb.gdevice.p_device_t = inquiry_d[0]; /*ﾃﾞﾊﾞｲｽ･ﾀｲﾌﾟをセット*/
            }
        else
            {
            if(p_end == P_EEND && ABORT_F==(error&ABORT_F))
                srb_p->status = 0x02;                      /*scsi request aborted by hostをセット*/
            else
                {
                if(p_end == STOUT)
                    srb_p->status = 0x82;                  /*scsi device not installedをセット*/
                else
                    srb_p->status = 0x04;                  /*scsi request completed with error*/
                }
            }
        }

    else
        srb_p->status = 0x82;                              /*scsi device not installedをセット*/

    return;
    }


/*************************************************************************
 *        EXECUTE  SCSI  I／O  REQUEST処理                *
 *アプリケーション・プログラムの要求によりＳＣＳＩコマンドを実行し，    *
 *実行結果をアプリケーション・プログラムに渡す                          *
 *      入力：SRB                                                        *
 *      出力：SRB                                                        *
 *************************************************************************/


void ESCSI_R(void)
    {
                                                       /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ*/
    unsigned char sense_c[6] = {0x03,0x00,0x00,0x00,0x00,0x00};
    unsigned char scsi_s;                              /*scsiｽﾃｰﾀｽ*/

    if(srb_p->u_srb.escsirq.target_id != i_id)
        {
        srb_p->status = 0x00;                          /*scsi request in progressをセット*/
        message_o = IDENTIFY1;                         /*IDENTIFYﾒｯｾｰｼﾞをセット*/
        message_o = message_o|(unsigned long)srb_p->u_srb.escsirq.lun;
        t_id = srb_p->u_srb.escsirq.target_id;         /*ﾀｰｹﾞｯﾄidをセット*/
        sd_scsil[t_id] = srb_p->u_srb.escsirq.data_a_l;    /*転送ﾃﾞｰﾀ数をセット*/
        sd_scsip[t_id] = srb_p->u_srb.escsirq.data_bp;     /*転送ﾃﾞｰﾀ･ﾎﾟｲﾝﾀをセット*/
        sc_scsil[t_id] = srb_p->u_srb.escsirq.scsicdb_l;   /*転送ｺﾏﾝﾄﾞ数をセット*/
        sc_scsip[t_id] = srb_p->u_srb.escsirq.scsics;      /*転送ｺﾏﾝﾄﾞ･ﾎﾟｲﾝﾀをセット*/
                                                       /*ﾀｰｹﾞｯﾄ･ｽﾃｰﾀｽ･ﾎﾟｲﾝﾀをセット*/
        ss_scsip[t_id] = &(srb_p->u_srb.escsirq.target_status);

        S_PHASE();                                     /*SCSIﾌｪｰｽﾞ処理*/
```

```
switch(p_end)
    {
    case    S_NEND:                                 /*S_PHASE正常終了*/
            srb_p->u_srb.escsirq.h_adapter_s = 0x00;/*host adapter did not detect any errorをｾｯﾄ*/
            srb_p->status = 0x01;                   /*scsi request completed without errorをｾｯﾄ*/
                                                    /*check statusか？*/
            if(srb_p->u_srb.escsirq.target_status == 0x02)
                {
                                                    /*sense_cにlunをセット*/
                sense_c[1] = srb_p->u_srb.escsirq.lun<<5;
                                                    /*sense_cにｾﾝｽ･ﾃﾞｰﾀ数をセット*/
                sense_c[4] = srb_p->u_srb.escsirq.sense_a_l;
                message_o = IDENTIFY1;              /*IDENTIFYﾒｯｾｰｼﾞをセット*/
                message_o = message_o|(unsigned long)srb_p->u_srb.escsirq.lun;
                                                    /*ﾀｰｹﾞｯﾄidをセット*/
                t_id = srb_p->u_srb.escsirq.target_id;
                                                    /*ｾﾝｽ･ﾃﾞｰﾀ数をセット*/
                sd_scsil[t_id] = srb_p->u_srb.escsirq.sense_a_l;
                                                    /*ｾﾝｽ･ﾃﾞｰﾀ･ﾎﾟｲﾝﾀをセット*/
                sd_scsip[t_id] = &(srb_p->u_srb.escsirq.scsics[srb_p->u_srb.escsirq.scsicdb_l]);
                sc_scsil[t_id] = 0x06;              /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ数をセット*/
                sc_scsip[t_id] = sense_c;           /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ･ﾎﾟｲﾝﾀをセット*/
                ss_scsip[t_id] = &scsi_s;           /*scsiｽﾃｰﾀｽ･ﾎﾟｲﾝﾀをセット*/

                S_PHASE();                          /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ実行 (SCSIﾌｪｰｽﾞ処理) */
                }

            break;

    case    S_RESETC:                               /*SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ終了*/
            srb_p->u_srb.escsirq.h_adapter_s = 0x13;/*unexpected bus freeをセット*/
            srb_p->status = 0x04;                   /*scsi request completed with errorをセット*/
                                                    /*check statusをセット*/
            srb_p->u_srb.escsirq.target_status = 0x02;
                                                    /*sense_cにlunをセット*/
            sense_c[1] = srb_p->u_srb.escsirq.lun<<5;
                                                    /*sense_cにｾﾝｽ･ﾃﾞｰﾀ数をセット*/
            sense_c[4] = srb_p->u_srb.escsirq.sense_a_l;
            message_o = IDENTIFY1;                  /*IDENTIFYﾒｯｾｰｼﾞをセット*/
            message_o = message_o|(unsigned long)srb_p->u_srb.escsirq.lun;
            t_id = srb_p->u_srb.escsirq.target_id;  /*ﾀｰｹﾞｯﾄidをセット*/
                                                    /*ｾﾝｽ･ﾃﾞｰﾀ数をセット*/
            sd_scsil[t_id] = srb_p->u_srb.escsirq.sense_a_l;
                                                    /*ｾﾝｽ･ﾃﾞｰﾀ･ﾎﾟｲﾝﾀをセット*/
            sd_scsip[t_id] = &(srb_p->u_srb.escsirq.scsics[srb_p->u_srb.escsirq.scsicdb_l]);
            sc_scsil[t_id] = 0x06;                  /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ数をセット*/
            sc_scsip[t_id] = sense_c;               /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ･ﾎﾟｲﾝﾀをセット*/
            ss_scsip[t_id] = &scsi_s;               /*scsiｽﾃｰﾀｽ･ﾎﾟｲﾝﾀをセット*/

            S_PHASE();                              /*ﾘｸｴｽﾄ･ｾﾝｽ･ｺﾏﾝﾄﾞ実行 (SCSIﾌｪｰｽﾞ処理) */
            break;

    case    BFTOUT:                                 /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
    case    STOUT:                                  /*ｾﾚｸﾄ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
            srb_p->u_srb.escsirq.h_adapter_s = 0x11;/*selection timeoutをセット*/
            srb_p->status = 0x04;                   /*scsi request completed with errorをセット*/
                                                    /*no target statusをセット*/
            srb_p->u_srb.escsirq.target_status = 0x00;
            break;
```

```
        case    P_EEND:                                 /*ﾌｪｰｽﾞ･ｴﾗｰ終了*/
                if(ABORT_F == (error&ABORT_F))          /*ABORTﾌﾗｸﾞ=1?*/
                    {
                                                        /*host adapter did not detect any errorをｾｯﾄ*/
                    srb_p->u_srb.escsirq.h_adapter_s = 0x00;
                    srb_p->status = 0x02;               /*scsi request aborted by hostをｾｯﾄ*/
                                                        /*no target statusをｾｯﾄ*/
                    srb_p->u_srb.escsirq.target_status = 0x00;
                    }
                else
                    {
                    if(DISCON_F == (p_error&DISCON_F))  /*DISCONNECTﾌﾗｸﾞ=1?*/
                        {
                                                        /*unexpected bus freeをｾｯﾄ*/
                        srb_p->u_srb.escsirq.h_adapter_s = 0x13;
                        srb_p->status = 0x04;           /*scsi request completed with errorをｾｯﾄ*/
                                                        /*no target statusをｾｯﾄ*/
                        srb_p->u_srb.escsirq.target_status = 0x00;
                        }
                    else
                        {
                                                        /*target bus phase sequence failureをｾｯﾄ*/
                        srb_p->u_srb.escsirq.h_adapter_s = 0x14;
                        srb_p->status = 0x04;           /*scsi request completed with errorをｾｯﾄ*/
                                                        /*no target statusをｾｯﾄ*/
                        srb_p->u_srb.escsirq.target_status = 0x00;
                        }
                    }
            }

        if(FIFOE_F == (error&FIFOE_F))                  /*FIFO ERRORﾌﾗｸﾞ=1?*/
            {
            srb_p->u_srb.escsirq.h_adapter_s = 0x12;    /*data over-run/under-runｾｯﾄ*/
            srb_p->status = 0x04;                       /*scsi request completed with errorをｾｯﾄ*/
            srb_p->u_srb.escsirq.target_status = 0x00;  /*no target statusをｾｯﾄ*/
            }
        }
    else
        srb_p->status = 0x82;                           /*scsi device not installedをｾｯﾄ*/

    return;
    }
```

保守／廃止

```
/*********************************************************************
 *                RESET SCSI DEVICE処理                              *
 *指定されたＳＣＳＩターゲットをリセットし，実行結果をアプリケーション・*
 *プログラムに渡す                                                   *
 *      入力：SRB                                                    *
 *      出力：SRB                                                    *
 *********************************************************************/


void RESET_D(void)
    {

    if(srb_p->u_srb.resetsd.target_id != i_id)
        {
        srb_p->status = 0x00;                           /*scsi request in progressをセット*/
        message_o = BUS_D_R;                            /*BUS RESET DEVICEﾒｯｾｰｼﾞをセット*/
        t_id = srb_p->u_srb.resetsd.target_id;          /*ﾀｰｹﾞｯﾄidをセット*/

        S_PHASE();                                      /*SCSIﾌｪｰｽﾞ処理*/

        switch(p_end)
            {
            case    S_RESETC:
                    srb_p->u_srb.resetsd.h_adapter_s = 0x13;/*unexpected bus freeをセット*/
                    srb_p->status = 0x04;               /*scsi request completed with errorをセット*/
                                                        /*no target statusをセット*/
                    srb_p->u_srb.resetsd.target_status = 0x00;
                    break;

            case    BFTOUT:                             /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
            case    STOUT:                              /*ｾﾚｸﾄ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
                    srb_p->u_srb.resetsd.h_adapter_s = 0x11;/*selection timeoutをセット*/
                    srb_p->status = 0x04;               /*scsi request completed with errorをセット*/
                                                        /*no target statusをセット*/
                    srb_p->u_srb.resetsd.target_status = 0x00;
                    break;

            case    P_EEND:                             /*ﾌｪｰｽﾞ･ｴﾗｰ終了*/
                                                        /*BUS_DRﾌﾗｸﾞ=1and DISCONNECTﾌﾗｸﾞ?*/
                    if(BUSDR_F==(error&BUSDR_F) && DISCON_F==(p_error&DISCON_F))
                        {
                                                        /*host adapter did not detect any errorをｾｯﾄ*/
                        srb_p->u_srb.resetsd.h_adapter_s = 0x00;
                        srb_p->status = 0x01;           /*scsi request completed without errorをセット*/
                                                        /*no target statusをセット*/
                        srb_p->u_srb.resetsd.target_status = 0x00;
                        }
                    else
                        {
                                                        /*target bus phase sequence failureをセット*/
                        srb_p->u_srb.resetsd.h_adapter_s = 0x14;
                        srb_p->status = 0x04;           /*scsi request completed with errorをセット*/
                                                        /*no target statusをセット*/
                        srb_p->u_srb.resetsd.target_status = 0x00;
                        }
            }

        if(FIFOE_F == (error&FIFOE_F))                  /*FIFO ERRORﾌﾗｸﾞ=1?*/
            {
            srb_p->u_srb.resetsd.h_adapter_s = 0x12;    /*data over-run/under-runｾｯﾄ*/
            srb_p->status = 0x04;                       /*scsi request completed with errorをセット*/
            srb_p->u_srb.resetsd.target_status = 0x00;  /*no target statusをセット*/
            }
        }

    else
        srb_p->status = 0x82;                           /*scsi device not installedをセット*/

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *      SCSIフェーズ処理    (S_PHASE.CPP)                    *
 *************************************************************************/

#include <dos.h>
#include <mem.h>

/*μPD72611の間接ﾚｼﾞｽﾀI/Oｱﾄﾞﾚｽ*/
#define     R_TMOD      0x10                    /*転送モード・レジスタ*/
#define     R_SID       0x02                    /*ソースＩＤレジスタ*/
#define     R_BTCL      0x11                    /*ベース転送カウンタ*/
#define     R_BTCM      0x12
#define     R_BTCH      0x13
#define     R_CTCL      0x11                    /*カレント転送カウンタ*/
#define     R_CTCM      0x12
#define     R_CTCH      0x13

/*μPD72611ｺﾏﾝﾄﾞ*/
#define     SET_ATN     0x03                    /*SET ATN*/
#define     RESET_ACK   0x04                    /*RESET ACK*/
#define     CLR_FIFO    0x05                    /*CLEAR FIFO*/
#define     SCSI_RES    0x08                    /*SCSI RESET*/
#define     CMD_SELECT  0x18                    /*SELECT*/
#define     TRAN_0      0x12                    /*TRANSFER*/
#define     TRAN_2      0x92
#define     TRAN_3      0xD2

/*割り込みｺﾝﾄﾛｰﾗI/Oｱﾄﾞﾚｽ*/
#define     R1_ICW1     0x0020                  /*ｺﾝﾄﾛｰﾙ1 ICW1ﾚｼﾞｽﾀ*/
#define     R2_ICW1     0x00A0                  /*ｺﾝﾄﾛｰﾙ2 ICW1ﾚｼﾞｽﾀ*/

/*割り込みｺﾝﾄﾛｰﾗFIｺﾏﾝﾄﾞ*/
#define     FI_C        0x60                    /*FIｺﾏﾝﾄﾞ/回転なし/割り込みﾚﾍﾞﾙ指定あり*/

/*変数p_end*/
#define     S_NEND      0x01                    /*S_PHASE正常終了*/
#define     S_RESETC    0x02                    /*SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ終了*/
#define     BFTOUT      0x03                    /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ終了*/
#define     STOUT       0x04                    /*ｾﾚｸｼｮﾝ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ終了*/
#define     P_EEND      0x05                    /*ﾌｪｰｽﾞ･ｴﾗｰ終了*/

/*message*/
#define     COMMAND_CP  0x00L                   /*COMMAND COMPLETE*/
#define     EXTEND_ME   0x01L                   /*拡張ﾒｯｾｰｼﾞ*/
#define     SYNCHRONOUS 0x01L                   /*SYNCHRONOUS DATA TRANSFER REQUEST*/
#define     SAVE_DP     0x02L                   /*SAVE DATA POINTER*/
#define     RESTORE_P   0x03L                   /*RESTORE POINTER*/
#define     DISCON      0x04L                   /*DISCONNECT*/
#define     I_DETECTED  0x05L                   /*INITIATOR DETECTED ERROR*/
#define     ABORT       0x06L                   /*ABORT*/
#define     MESS_REJ    0x07L                   /*MESSAGE REJECT*/
#define     NO_OPE      0x08L                   /*NO OPERATION*/
#define     MESS_PARE   0x09L                   /*MESSAGE PARITY ERROR*/
#define     BUS_D_R     0x0CL                   /*BUS DEVICE RESET*/
#define     IDENTIFY1   0xC0L                   /*IDENTIFY*/
#define     IDENTIFY2   0x80L

/*i_message*/
#define     INITIDE_F       0x01                /*INITIATOR_DEﾌﾗｸﾞ*/
#define     MESSPE_F        0x02                /*MESSAGE_PEﾌﾗｸﾞ*/
#define     IDEN_F          0x04                /*IDENTIFYﾌﾗｸﾞ*/
#define     SYNC_F          0x08                /*SYNCHRONOUSﾌﾗｸﾞ*/

/*p_error*/
#define     DOPE_F          0x01                /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     DIPE_F          0x02                /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     COMPE_F         0x04                /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     STAPE_F         0x08                /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｴﾗｰ*/
#define     DISCON_F        0x10                /*ﾃﾞｨｽｺﾈｸﾃｨｯﾄﾞ*/
```

A

```
/*error*/
#define     INITDE_F        0x01                    /*INITIATOR_DEﾌﾗｸﾞ*/
#define     MESSAGEPE_F     0x02                    /*MESSAGE_PEﾌﾗｸﾞ*/
#define     IDENTIFY_F      0x04                    /*IDENTIFYﾌﾗｸﾞ*/
#define     ABORT_F         0x08                    /*ABORTﾌﾗｸﾞ*/
#define     BUSDR_F         0x10                    /*BUS_DRﾌﾗｸﾞ*/
#define     FIFOE_F         0x20                    /*FIFO ERRORﾌﾗｸﾞ*/
#define     SYNCOF_F        0x40                    /*SYNC OFFSET ERRORﾌﾗｸﾞ*/
#define     PERROR_F        0x80                    /*PARITY ERRORﾌﾗｸﾞ*/

/*μPD72611直接ﾚｼﾞｽﾀ*/
extern  unsigned int R_ADR;                         /*ｱﾄﾞﾚｽ・ﾚｼﾞｽﾀ*/
extern  unsigned int R_CST;                         /*ｺﾝﾄﾛｰﾗ・ｽﾃｰﾀｽ・ﾚｼﾞｽﾀ*/
extern  unsigned int R_IST;                         /*割り込みｽﾃｰﾀｽ・ﾚｼﾞｽﾀ*/
extern  unsigned int R_CMD;                         /*ｺﾏﾝﾄﾞ・ﾚｼﾞｽﾀ*/
extern  unsigned int R_DID;                         /*ﾃﾞｨｽﾄﾈｰｼｮﾝIDﾚｼﾞｽﾀ*/
extern  unsigned int R_WIN1;                        /*ｳｲﾝﾄﾞ1*/
extern  unsigned int R_EXST;                        /*ｴｸｽﾃﾝﾃﾞｨﾄﾞ・ｽﾃｰﾀｽ・ﾚｼﾞｽﾀ*/
extern  unsigned int R_DF2;                         /*ﾃﾞｰﾀFIFO2ﾚｼﾞｽﾀ*/

/*割り込みｺﾝﾄﾛｰﾗ・ﾎﾟｰﾄ番号*/
extern  unsigned char intp;

/*outportw(),inportw()*/
extern  "C" void    outportdw(void);                /*32ﾋﾞｯﾄI/OのOUT関数*/
extern  "C" void    inportdw(void);                 /*32ﾋﾞｯﾄI/OのIN関数*/
extern  unsigned int    w_port;                     /*ﾗｲﾄI/Oﾎﾟｰﾄ・ｱﾄﾞﾚｽ*/
extern  unsigned long   w_data;                     /*32ﾋﾞｯﾄ・ﾗｲﾄ・ﾃﾞｰﾀ*/
extern  unsigned int    r_port;                     /*ﾘｰﾄﾞI/Oﾎﾟｰﾄ・ｱﾄﾞﾚｽ*/
extern  unsigned long   r_data;                     /*32ﾋﾞｯﾄ・ﾘｰﾄﾞ・ﾃﾞｰﾀ*/
extern  unsigned char   dataf_b[2048];              /*ﾃﾞｰﾀ・ﾌｪｰｽﾞにおけるﾊﾞｯﾌｧ*/

/*DMAC*/
extern  unsigned char   cha;                        /*DMAのﾁｬﾈﾙ番号*/

extern  unsigned int    R_DMOD[];                   /*ﾓｰﾄﾞ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DEMOD[];                  /*拡張ﾓｰﾄﾞ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DWSMB[];                  /*ﾗｲﾄ・ｼﾝｸﾞﾙ・ﾏｽﾀ・ﾋﾞｯﾄ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DCWC[];                   /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾁﾌﾞﾙ・ﾜｰﾄﾞ・ｶｳﾝﾄ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DHWC[];                   /*ﾍﾞｰｽ・ﾊｲ・ﾜｰﾄﾞ・ｶｳﾝﾄ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DBCA[];                   /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾁﾌﾞﾙ・ｱﾄﾞﾚｽ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DBLP[];                   /*ﾍﾞｰｽ・ﾛｳ・ﾍﾟｰｼﾞ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DBHP[];                   /*ﾍﾞｰｽ・ﾊｲ・ﾍﾟｰｼﾞ・ﾚｼﾞｽﾀ*/
extern  unsigned int    R_DCBP[];                   /*ｸﾘｱ・ﾊﾞｲﾄ・ﾎﾟｲﾝﾀ・ｺﾏﾝﾄﾞ*/
extern  unsigned int    R_DSTA[];                   /*ｽﾃｰﾀｽ・ﾚｼﾞｽﾀ*/

unsigned long   ad_scsil;                           /*転送ﾃﾞｰﾀ数*/
unsigned char far *ad_scsip;                        /*ｱｸﾃｨﾌﾞ・ﾃﾞｰﾀSCSIﾎﾟｲﾝﾀ*/
unsigned char   ac_scsil;                           /*転送ｺﾏﾝﾄﾞ数*/
unsigned char far *ac_scsip;                        /*ｱｸﾃｨﾌﾞ・ｺﾏﾝﾄﾞSCSIﾎﾟｲﾝﾀ*/
unsigned char far *as_scsip;                        /*ｱｸﾃｨﾌﾞ・ｽﾃｰﾀｽ・SCSIﾎﾟｲﾝﾀ*/
unsigned long   sd_scsil[8];                        /*転送ﾃﾞｰﾀ数*/
unsigned char far *sd_scsip[8];                     /*ｾｰﾌﾞ・ﾃﾞｰﾀSCSIﾎﾟｲﾝﾀ*/
unsigned char   sc_scsil[8];                        /*転送ｺﾏﾝﾄﾞ数*/
unsigned char far *sc_scsip[8];                     /*ｾｰﾌﾞ・ｺﾏﾝﾄﾞSCSIﾎﾟｲﾝﾀ*/
unsigned char far *ss_scsip[8];                     /*ｾｰﾌﾞ・ｽﾃｰﾀｽSCSIﾎﾟｲﾝﾀ*/
unsigned char t_id;                                 /*ﾀｰｹﾞｯﾄID*/
unsigned char ist_b[16];                            /*ISTﾘﾝｸﾞ・ﾊﾞｯﾌｧ*/
unsigned char ist_w;                                /*ISTﾘﾝｸﾞ・ﾊﾞｯﾌｧ・ﾗｲﾄ・ﾎﾟｲﾝﾀ*/
unsigned char ist_r;                                /*ISTﾘﾝｸﾞ・ﾊﾞｯﾌｧ・ﾗｲﾄ・ﾎﾟｲﾝﾀ*/
unsigned char ist;                                  /*IST*/
unsigned char message_i;                            /*ﾀｰｹﾞｯﾄからのﾒｯｾｰｼﾞを示す*/
unsigned char p_end;                                /*S_PHASEの終了結果*/
unsigned char p_error;                              /*ﾌｪｰｽﾞ・ｴﾗｰのﾌｪｰｽﾞを示す*/
unsigned char error;                                /*ｴﾗｰを示す*/
unsigned char emessage_c;                           /*拡張ﾒｯｾｰｼﾞのﾊﾞｲﾄ数*/
unsigned char i_message;                            /*ﾀｰｹﾞｯﾄがｻﾎﾟｰﾄしていないﾒｯｾｰｼﾞを示す*/
unsigned char pmessage_c;                           /*ﾊﾟﾘﾃｨ・ｴﾗｰが発生した拡張ﾒｯｾｰｼﾞのﾊﾞｲﾄ数を示す*/
unsigned char messagepe_c;                          /*message parity errorﾒｯｾｰｼﾞの転送回数*/
unsigned long message_o;                            /*ﾀｰｹﾞｯﾄに転送するﾒｯｾｰｼﾞ*/
unsigned char sync[8];                              /*転送ﾓｰﾄﾞ*/
unsigned char expect_priod[8];                      /*転送時間*/
unsigned char expect_offset[8];                     /*転送ｵﾌｾｯﾄ値*/
```

```
unsigned char expect_sync[11][2] = {{0xC8, 0x80},          /*μPD72611を同期転送に設定する*/
                                    {0xAF, 0xF0},          /*値を参照するテーブル*/
                                    {0x96, 0xE0},
                                    {0x7D, 0xD0},
                                    {0x64, 0xC0},
                                    {0x58, 0xF8},
                                    {0x4B, 0xE8},
                                    {0x3F, 0xD8},
                                    {0x32, 0xC8},
                                    {0x26, 0xB8},
                                    {0x19, 0xA8}};

void S_PHASE(void);                                        /*ＳＣＳＩフェーズ処理*/
void interrupt H_INT();                                    /*μPD72611ﾊｰﾄﾞｳｪｱ割り込み処理*/
void SELECT_C(void);                                       /*セレクト・コマンド処理*/
void IST_CONTROL(void);                                    /*ＩＳＴ割り込み要因対応処理*/
void ISTRIG_R(void);                                       /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧ･ﾘｰﾄﾞ*/
void COMMAND_NE(void);                                     /*コマンド正常終了*/
void FIFO_OU(void);                                        /*FIFOｵｰﾊﾞﾗﾝ/ｱﾝﾀﾞｰﾗﾝ･ｴﾗｰ*/
void SYNC_OE(void);                                        /*同期モード・オフセット・エラー*/
void BFTOUT_E(void);                                       /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑｱｳﾄ･ｴﾗｰ*/
void STOUT_E(void);                                        /*ｾﾚｸｼｮﾝ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
void DATAO_PE(void);                                       /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void DATAI_PE(void);                                       /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void COMMAND_PE(void);                                     /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void STATUS_PE(void);                                      /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void MESSAGEO_PE(void);                                    /*ﾒｯｾｰｼﾞ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void MESSAGEI_PE(void);                                    /*ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
void SCSI_RESET(void);                                     /*SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ*/
void DISCONNECTED(void);                                   /*ﾃﾞｨｽｺﾈｸﾃｨﾄﾞ*/
void RESELECTED(void);                                     /*ﾘｾﾚｸﾃｨﾄﾞ*/
void DATA_OS(void);                                        /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void DATA_IS(void);                                        /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void COMMAND_S(void);                                      /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void STATUS_S(void);                                       /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void MESSAGE_OS(void);                                     /*ﾒｯｾｰｼﾞ･ｱｳﾄ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void MESSAGE_IS(void);                                     /*ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
void MESSAGE_R(void);                                      /*メッセージ受信*/
```

```
/*************************************************************************
 *                    ＳＣＳＩフェーズ処理                              *
 *      μＰＤ７２６１１の制御およびＳＣＳＩフェーズの管理を行う      *
 *                                                                       *
 *      入力：message_o, sd_scsil[], sd_scsip[], sc_scsil[], sc_scsip[]  *
 *            ss_scsip[], t_id, i_message, expect_priod, expect_offset, sync *
 *      出力：p_end, p_error, error, sync[]                              *
 *************************************************************************/

void S_PHASE(void)
    {
    unsigned char c;

    ad_scsil = sd_scsil[t_id];                             /*ｱｸﾃｨﾌﾞ SCSIﾎﾟｲﾝﾀにt_idが示すｾｰﾌﾞ*/
    ad_scsip = sd_scsip[t_id];                             /*SCSIﾎﾟｲﾝﾀの値をセットする      */
    ac_scsil = sc_scsil[t_id];
    ac_scsip = sc_scsip[t_id];
    as_scsip = ss_scsip[t_id];

    message_i = 0xFF;                                      /*各変数をリセットする*/
    ist_w = 0x00;
    ist_r = 0x00;
    p_end = 0x00;
    p_error = 0x00;
    error = 0x00;
    messagepe_c = 0x00;
    pmessage_c = 0x00;
    emessage_c = 0x00;
```

```
    if(sync[t_id] == 0x01)
        {
        c = 0;                                          /*μPD72611を同期転送モードにセットする*/
        while(c < 11)
            {
            if(expect_priod[t_id] == expect_sync[c][0])
                break;
            else
                ++c;
            }
        outportb(R_ADR,R_TMOD);
        outportb(R_WIN1,(expect_sync[c][1]|(expect_offset[t_id]&0x07)));
        }
    else
        {
        outportb(R_ADR,R_TMOD);
        outportb(R_WIN1,0x00);
        }

    SELECT_C();                                         /*セレクト・コマンド処理*/

    IST_CONTROL();                                      /*ＩＳＴ割り込み要因対応処理*/

    return;
    }


/*************************************************************************
 *                    セレクト・コマンド処理                             *
 *   ＳＣＳＩバスを獲得し，ターゲットをセレクトする                      *
 *                                                                       *
 *    入力：  t_id                                                       *
 *    出力：  なし                                                       *
 *************************************************************************/


void SELECT_C(void)
    {
    unsigned char cbsy;

    outportb(R_DID,t_id);                               /*μＰＤ７２６１１のＤＩＤレジスタに*/
                                                        /*ターゲットＩＤ番号をセット*/

    do
        {
        cbsy = inportb(R_CST);                          /*μＰＤ７２６１１のＣＳＴレジスタを*/
                                                        /*リードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                                /*CBSYﾋﾞｯﾄ == 0？*/

    outportb(R_CMD,CMD_SELECT);                         /*μＰＤ７２６１１にＳＥＬＥＣＴコマンド*/
                                                        /*発行する*/
    return;
    }
```

```
/*************************************************************************
 *            μPD72611ハードウェア割り込み処理                          *
 *μPD72611のISTレジスタをリードし，ISTリング・バッファに                *
 *セットする                                                             *
 *                                                                       *
 *      入力：intp                                                       *
 *      出力：ist_b[]                                                    *
 *************************************************************************/


void interrupt H_INT()
    {
    ist_b[ist_w] = inportb(R_IST);                  /*μPD72611のISTレジスタから*/
                                                    /*リードしISTリング・バッファにセット*/
                                                    /*する*/
    if(ist_w == 0x0f)
        ist_w = 0x00;                               /*ist_wがOFHを越える場合リセットする*/
    else
        ++ist_w;                                    /*ist_wがOFHを越えない場合インクリメント*/

    if(intp < 8)                                    /*割り込みｺﾝﾄﾛｰﾗにFIｺﾏﾝﾄﾞ発行*/
        outportb(R1_ICW1,(FI_C|(intp&0x07)));       /*ﾏｽﾀｰにFIｺﾏﾝﾄﾞ発行*/
    else
        {
        outportb(R1_ICW1,(FI_C|0x02));              /*ﾏｽﾀｰにFIｺﾏﾝﾄﾞ発行*/
        outportb(R2_ICW1,(FI_C|(intp&0x07)));       /*ｽﾚｰﾌﾞにFIｺﾏﾝﾄﾞ発行*/
        }
    }


/*************************************************************************
 *                  ISTリング・バッファ・リード処理                      *
 * ISTリング・バッファからμPD72611の割り込み要因をリードする           *
 *                                                                       *
 *      入力：ist_b[],ist_w                                              *
 *      出力：ist                                                        *
 *************************************************************************/


void ISTRIG_R(void)
    {
    while(ist_r == ist_w)
        ;                                           /*ist_r≠ist_wになるまで待つ*/
    ist = ist_b[ist_r];                             /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧの内容をistにセットする*/
    if(ist_r == 0x0f)
        ist_r = 0x00;                               /*ist_rがOFHを越える場合リセットする*/
    else
        ++ist_r;                                    /*ist_rがOFHを越えない場合インクリメント*/
                                                    /*する*/
    return;
    }
```

```
/*************************************************************************
 *                    ＩＳＴ割り込み要因対応処理                         *
 *  ＩＳＴの割り込み要因に対応したモジュールを呼び出す                   *
 *                                                                       *
 *      入力：ist,p_end                                                  *
 *      出力：なし                                                       *
 *************************************************************************/


void IST_CONTROL(void)
    {
    do
        {
        ISTRIG_R();                                 /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧ･ﾘｰﾄﾞ*/

        switch(ist)                                 /*ＩＳＴレジスタの内容*/
            {
            case    0x00:
                COMMAND_NE();                       /*コマンド正常終了*/
                break;
            case    0x20:
                FIFO_OU();                          /*FIFOｵｰﾊﾞﾗﾝ/ｱﾝﾀﾞｰﾗﾝ･ｴﾗｰ*/
                break;
            case    0x21:
                SYNC_OE();                          /*同期モード・オフセット・エラー*/
                break;
            case    0x24:
                BFTOUT_E();                         /*ﾊﾞｽ･ﾌﾘｰ･ﾀｲﾑｱｳﾄ･ｴﾗｰ*/
                break;
            case    0x25:
                STOUT_E();                          /*ｾﾚｸｼｮﾝ･ﾀｲﾑ･ｱｳﾄ･ｴﾗｰ*/
                break;
            case    0x30:
                DATAO_PE();                         /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x31:
                DATAI_PE();                         /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x32:
                COMMAND_PE();                       /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x33:
                STATUS_PE();                        /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x36:
                MESSAGEO_PE();                      /*ﾒｯｾｰｼﾞ･ｱｳﾄ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x37:
                MESSAGEI_PE();                      /*ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｴﾗｰ*/
                break;
            case    0x81:
                SCSI_RESET();                       /*SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ*/
                break;
            case    0x90:
                DISCONNECTED();                     /*ﾃﾞｨｽｺﾈｸﾃｨﾄﾞ*/
                break;
            case    0x91:
                RESELECTED();                       /*ﾘｾﾚｸﾃｨﾄﾞ*/
                break;
            case    0xA0:
                DATA_OS();                          /*ﾃﾞｰﾀ･ｱｳﾄ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
            case    0xA1:
                DATA_IS();                          /*ﾃﾞｰﾀ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
            case    0xA2:
                COMMAND_S();                        /*ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
            case    0xA3:
                STATUS_S();                         /*ｽﾃｰﾀｽ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
```

```
            case    0xA6:
                MESSAGE_OS();                          /*ﾒｯｾｰｼﾞ･ｱｳﾄ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
            case    0xA7:
                MESSAGE_IS();                          /*ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                break;
            case    0xC0:
                MESSAGE_R();                           /*メッセージ受信*/
                break;
            default:
                break;
            }
        }
    while(p_end == 0x00);

    return;

    }


/*************************************************************************
 *            μＰＤ７２６１１コマンド正常終了処理                       *
 *ＩＳＴ割り込み要因のコマンド正常終了に対応する処理                       *
 *                                                                       *
 *          入力：ist_b[], ist_r, ist_w                                  *
 *          出力：message_o, error                                       *
 *************************************************************************/


void COMMAND_NE(void)
    {
    unsigned char sbper;
    sbper = inportb(R_EXST);                           /*μＰＤ７２６１１のＥＸＳＴレジスタを*/
    sbper = sbper & 0x01;                              /*ﾘｰﾄﾞしSCSIﾊﾞｽ･ﾊﾟﾘﾃｨ･ｴﾗｰを確認する*/
    if(sbper == 0x01)
        {
        message_o = I_DETECTED;                        /*INITIATOR DETECTED ERROR*/
        error = error | PERROR_F;                      /*errorにPARITY ERRORﾌﾗｸﾞをｾｯﾄ*/
        }
    else
        {
        while(ist_r == ist_w);
        if((ist_b[ist_r]==0xA6)||(ist_b[ist_r]==0xA7))
            ;                                          /*A6H:ﾒｯｾｰｼﾞ･ｱｳﾄ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
                                                       /*A7H:ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ*/
        else
            message_o = NO_OPE;                        /*NO OPERATION*/
        }
    return;
    }


/*************************************************************************
 *            ＦＩＦＯオーバラン／アンダーラン処理                       *
 * ＩＳＴ割り込み要因のＦＩＦＯオーバラン／アンダーランに対応する処理   *
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：message_o, error                                           *
 *************************************************************************/


void FIFO_OU(void)
    {
    outportb(R_CMD, SET_ATN);                          /*SET ATNｺﾏﾝﾄﾞを発行する*/
    outportb(R_CMD, CLR_FIFO);                         /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    message_o = I_DETECTED;                            /*INITIATOR DETECTED ERROR*/
    error = error | FIFOE_F;                           /*errorにFIFO ERRORﾌﾗｸﾞをｾｯﾄ*/

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                    同期モード・オフセット・エラー処理                 *
 * ＩＳＴ割り込み要因の同期モード・オフセット・エラーに対応する処理     *
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：message_o,error                                            *
 *************************************************************************/


void SYNC_OE(void)
    {
    outportb(R_CMD,SET_ATN);                          /*SET ATNｺﾏﾝﾄﾞを発行する*/
    outportb(R_CMD,CLR_FIFO);                         /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    message_o = I_DETECTED;                           /*INITIATOR DETECTED ERROR*/
    error = error | SYNCOF_F;                         /*errorにSYNC OFFSET ERRORﾌﾗｸﾞをｾｯﾄ*/

    return;
    }


/*************************************************************************
 *            バス・フリー・タイム・アウト・エラー処理                   *
 *ＩＳＴ割り込み要因のバス・フリー・タイム・アウト・エラーに対応する処理*
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：p_end                                                      *
 *************************************************************************/


void BFTOUT_E(void)
    {
    p_end = BFTOUT;                                   /*p_endに03H(BFTOUT)をセットする*/
    return;
    }


/*************************************************************************
 *            セレクション・タイム・アウト・エラー処理                   *
 *ＩＳＴ割り込み要因のセレクション・タイム・アウト・エラーに対応する処理*
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：p_end                                                      *
 *************************************************************************/


void STOUT_E(void)
    {
    p_end = STOUT;                                    /*p_endに04H(STOUT)をセットする*/
    return;
    }
```

保守／廃止

```
/*************************************************************************
 *              データ・アウト・フェーズ・エラー処理                    *
 *ＩＳＴ割り込み要因のデータ・アウト・フェーズ・エラーに対応する処理    *
 *                                                                       *
 *      入力：ist_b[],ist_r,ist_w                                        *
 *      出力：p_error,p_end                                              *
 *************************************************************************/

void DATAO_PE(void)
    {
    unsigned char cbsy;
    unsigned char ist_l;
    p_error = p_error | DOPE_F;                    /*p_errorにDOPEﾌﾗｸﾞをセットする*/

    do
        {
        cbsy = inportb(R_CST);                     /*CSTﾚｼﾞｽﾀをリードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                           /*CBSYﾋﾞｯﾄ == 0?*/

    sync[t_id] = 0x00;                             /*syncをＯＯＨ（転送モードを不定）にする*/
    outportb(R_ADR,R_TMOD);
    outportb(R_WIN1,0x00);                         /*転送モード・レジスタのSYNCﾋﾞｯﾄに0をセットする*/

    outportb(R_CMD,SCSI_RES);                      /*μPD72611にSCSI RESETｺﾏﾝﾄﾞを発行する*/

    do
        {
        while(ist_r == ist_w);
        ist_l = ist_b[ist_r];                      /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧのist_b[ist_r]が示す内容を*/
                                                   /*ist_lにセットする*/
        if(ist_r == 0x0F)
            ist_r = 0x00;
        else
            ++ist_r;
        }
    while(ist_l != 0x00);

    p_end = P_EEND;                                /*p_endに05H(ﾌｪｰｽﾞ･ｴﾗｰ終了)をセットする*/

    return;
    }


/*************************************************************************
 *              データ・イン・フェーズ・エラー処理                      *
 *ＩＳＴ割り込み要因のデータ・イン・フェーズ・エラーに対応する処理      *
 *                                                                       *
 *      入力：ist_b[],ist_r,ist_w                                        *
 *      出力：p_error,p_end                                              *
 *************************************************************************/

void DATAI_PE(void)
    {
    unsigned char cbsy;
    unsigned char ist_l;
    p_error = p_error | DIPE_F;                    /*p_errorにDIPEﾌﾗｸﾞをセットする*/

    do
        {
        cbsy = inportb(R_CST);                     /*CSTﾚｼﾞｽﾀをリードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                           /*CBSYﾋﾞｯﾄ == 0?*/

    sync[t_id] = 0x00;                             /*syncをＯＯＨ（転送モードを不定）にする*/
    outportb(R_ADR,R_TMOD);
    outportb(R_WIN1,0x00);                         /*転送モード・レジスタのSYNCﾋﾞｯﾄに0をセットする*/

    outportb(R_CMD,SCSI_RES);                      /*μPD72611にSCSI RESETｺﾏﾝﾄﾞを発行する*/
```

A

保守／廃止

```
    do
        {
        while(ist_r == ist_w);
        ist_l = ist_b[ist_r];                    /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧのist_b[ist_r]が示す内容を*/
                                                 /*ist_lにセットする*/
        if(ist_r == 0x0F)
            ist_r = 0x00;
        else
            ++ist_r;
        }
    while(ist_l != 0x00);

    p_end = P_EEND;                              /*p_endに05H(ﾌｪｰｽﾞ･ｴﾗｰ終了)をセットする*/

    return;
    }


/*************************************************************************
 *                  コマンド・フェーズ・エラー処理                       *
 *ＩＳＴ割り込み要因のコマンド・フェーズ・エラーに対応する処理           *
 *                                                                       *
 *      入力：ist_b[],ist_r,ist_w                                        *
 *      出力：p_error,p_end                                              *
 *************************************************************************/


void COMMAND_PE(void)
    {
    unsigned char cbsy;
    unsigned char ist_l;
    p_error = p_error | COMPE_F;                 /*p_errorにCOMPE_Fﾌﾗｸﾞをセットする*/

    do
        {
        cbsy = inportb(R_CST);                   /*CSTﾚｼﾞｽﾀをリードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                         /*CBSYﾋﾞｯﾄ == 0?*/

    sync[t_id] = 0x00;                           /*syncをＯＯＨ（転送モードを不定）にする*/
    outportb(R_ADR,R_TMOD);
    outportb(R_WIN1,0x00);                       /*転送モード・レジスタのSYNCﾋﾞｯﾄに0をセットする*/

    outportb(R_CMD,SCSI_RES);                    /*μPD72611にSCSI RESETｺﾏﾝﾄﾞを発行する*/

    do
        {
        while(ist_r == ist_w);
        ist_l = ist_b[ist_r];                    /*ISTﾘﾝｸﾞ･ﾊﾞｯﾌｧのist_b[ist_r]が示す内容を*/
                                                 /*ist_lにセットする*/
        if(ist_r == 0x0F)
            ist_r = 0x00;
        else
            ++ist_r;
        }
    while(ist_l != 0x00);

    p_end = P_EEND;                              /*p_endに05H(ﾌｪｰｽﾞ･ｴﾗｰ終了)をセットする*/

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                    ステータス・フェーズ・エラー処理                   *
 *ＩＳＴ割り込み要因のステータス・フェーズ・エラーに対応する処理        *
 *                                                                       *
 *       入力：なし                                                      *
 *       出力：p_error                                                   *
 *************************************************************************/


void STATUS_PE(void)
    {
    outportb(R_CMD,CLR_FIFO);                    /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    p_error = p_error | STAPE_F;                 /*p_errorにSTAPE_Fﾌﾗｸﾞをセットする*/
    return;
    }


/*************************************************************************
 *              メッセージ・アウト・フェーズ・エラー処理                 *
 *ＩＳＴ割り込み要因のメッセージ・アウト・フェーズ・エラーに対応する処理*
 *                                                                       *
 *          入力：なし                                                   *
 *          出力：message_o,error                                        *
 *************************************************************************/


void MESSAGEO_PE(void)
    {
    unsigned char sbper;
    sbper = inportb(R_EXST);                     /*μＰＤ７２６１１のＥＸＳＴレジスタを*/
    sbper = sbper & 0x01;                        /*リードしSCSIﾊﾞｽ･ﾊﾟﾘﾃｨ･ｴﾗｰを確認する*/
    if(sbper == 0x01)
        {                                        /*ﾊﾟﾘﾃｨ･ｴﾗｰ発生*/
        message_o = I_DETECTED;                  /*INITIATOR DETECTED ERROR*/
        error = error | PERROR_F;                /*errorにPARITY ERRORﾌﾗｸﾞをｾｯﾄ*/
        }
    outportb(R_CMD,CLR_FIFO);                    /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    return;
    }


/*************************************************************************
 *              メッセージ・イン・フェーズ・エラー処理                   *
 *ＩＳＴ割り込み要因のメッセージ・イン・フェーズ・エラーに対応する処理  *
 *                                                                       *
 *          入力：なし                                                   *
 *          出力：なし                                                   *
 *************************************************************************/


void MESSAGEI_PE(void)
    {
    outportb(R_CMD,CLR_FIFO);                    /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    return;
    }
```

A

保守／廃止

```
/*************************************************************************
 *              ＳＣＳＩリセット・コンディション処理                    *
 *ＩＳＴ割り込み要因のＳＣＳＩリセット・コンディションに対応する処理    *
 *                                                                       *
 *          入力：ist_r, ist_w, ist_b[]                                  *
 *          出力：p_end, sync[]                                          *
 *************************************************************************/


void SCSI_RESET(void)
    {
    unsigned char ist_l;
    outportb(R_CMD, CLR_FIFO);                          /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
    sync[t_id] = 0x00;                                  /*syncをＯＯＨ（転送モードを不定）にする*/
    outportb(R_ADR, R_TMOD);
    outportb(R_WIN1, 0x00);                             /*転送モード・レジスタのSYNCﾋﾞｯﾄに0をセットする*/
    if(ist_r == ist_w)
        p_end = S_RESETC;                               /*p_endに02H（SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ終了）をセットする*/
    else
        {
        ist_l = ist_b[ist_r];
        if(ist_l == 0x10)                               /*無効コマンド割り込み（１０Ｈ）*/
            p_end = S_RESETC;                           /*p_endに02H（SCSIﾘｾｯﾄ･ｺﾝﾃﾞｨｼｮﾝ終了）をセットする*/
        }
    return;
    }


/*************************************************************************
 *                      ディスコネクティド処理                           *
 *ＩＳＴ割り込み要因のディスコネクティドに対応する処理                   *
 *                                                                       *
 *          入力：message_i                                              *
 *          出力：message_i, p_end, p_error                              *
 *************************************************************************/


void DISCONNECTED(void)
    {
    switch(message_i)
        {
        case    DISCON:                                 /*ﾃﾞｨｽｺﾈｸﾄ･ﾒｯｾｰｼﾞ*/
            message_i = 0xFF;                           /*リセレクト直後にバス・フリーになる場合を考慮するため*/
            break;
        case    COMMAND_CP:                             /*ｺﾏﾝﾄﾞ･ｺﾝﾌﾟﾘｰﾄ･ﾒｯｾｰｼﾞ*/
            p_end = S_NEND;                             /*p_endに01H（S_PHASE正常終了）をセットする*/
            break;
        default:
            outportb(R_CMD, CLR_FIFO);                  /*CLEAR FIFOｺﾏﾝﾄﾞを発行する*/
            p_error = p_error|DISCON_F;                 /*p_errorにDISCON_Fﾌﾗｸﾞをセットする*/
            p_end = P_EEND;                             /*p_endに05H（ﾌｪｰｽﾞ･ｴﾗｰ終了）をセットする*/
        }
    return;
    }


/*************************************************************************
 *                      リセレクティド処理                               *
 *ＩＳＴ割り込み要因のリセレクティドに対応する処理                       *
 *                                                                       *
 *          入力：なし                                                   *
 *          出力：t_id                                                   *
 *************************************************************************/


void RESELECTED(void)
    {
    outportb(R_ADR, R_SID);                             /*μPD72611のSIDﾚｼﾞｽﾀをリードし，t_idにセットする*/
    t_id = inportb(R_WIN1);
    t_id = t_id & 0x07;                                 /*SIDﾚｼﾞｽﾀのS/Rﾋﾞｯﾄ(ﾋﾞｯﾄ7)をクリアする*/
    outportb(R_DID, t_id);                              /*μPD72611のDIDﾚｼﾞｽﾀにt_idの内容をセットする*/
    return;
    }
```

保守／廃止

```
/*********************************************************************
 *                データ・アウト・フェーズ・スタート処理               *
 *ＩＳＴ割り込み要因のデータ・アウト・フェーズ・スタートに対応する処理  *
 *                                                                   *
 *      入力：ad_scsil,ad_scsip                                      *
 *      出力：なし                                                   *
 *********************************************************************/


void DATA_OS(void)
    {
    union   b_count                             /*ad_scsil（データ・バイト数）をμPD72611，DMACにセット*/
            {                                   /*するための共用体*/
            unsigned long count_l;
            struct
                {
                unsigned char count_1;          /*D7～D0*/
                unsigned char count_2;          /*D15～D8*/
                unsigned char count_3;          /*D23～D16*/
                unsigned char count_4;          /*D31～D24*/
                }   byte_count;
            }   u_count;

    unsigned long change;
    unsigned long addr;

    union   b_addr                              /*ad_scsip（転送アドレス）をDMACにセットするための共用体*/
            {
            unsigned long  addr_l;              /*物理アドレス（A31～A0）*/
            struct
                {
                unsigned char addr_1;           /*A7～A0*/
                unsigned char addr_2;           /*A15～A8*/
                unsigned char addr_3;           /*A23～A16*/
                unsigned char addr_4;           /*A31～A24*/
                }   byte_addr;
            }   u_addr;

    unsigned char cbsy;

    outportb(R_DMOD[cha],(0x08|(cha&0x03)));    /*モード・レジスタ*/
                                                /*ﾃﾞｨﾏﾝﾄﾞ･ﾓｰﾄﾞ，ｱﾄﾞﾚｽ･ｲﾝｸﾘﾒﾝﾄ*/
                                                /*ｵｰﾄｲﾆｼｬﾗｲｽﾞ機能ﾃﾞｨｾｰﾌﾞﾙ，ﾒﾓﾘ→I/O転送*/
    outportb(R_DEMOD[cha],(0x28|(cha&0x03)));   /*拡張モード・レジスタ*/
                                                /*ｽﾄｯﾌﾟ･ﾚｼﾞｽﾀ･ﾃﾞｨｾｰﾌﾞﾙ，T-C出力，Type Bｻｲｸﾙ*/
                                                /*32bit I/O count by byte*/
    change = (unsigned long)dataf_b;            /*ｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽを物理ｱﾄﾞﾚｽに変換する*/
    u_addr.addr_l = change & 0x0000FFFFL;
    change = change & 0xFFFF0000L;
    u_addr.addr_l = u_addr.addr_l + (change >> 12);

    u_count.count_l = ad_scsil;                 /*μPD72611のﾍﾞｰｽ転送ｶｳﾝﾀをセットする*/
    outportb(R_ADR,R_BTCL);                     /*BTCLﾚｼﾞｽﾀにD7～D0をセット*/
    outportb(R_WIN1,u_count.byte_count.count_1);
    outportb(R_ADR,R_BTCM);                     /*BTCMﾚｼﾞｽﾀにD15～D8をセット*/
    outportb(R_WIN1,u_count.byte_count.count_2);
    outportb(R_ADR,R_BTCH);                     /*BTCHﾚｼﾞｽﾀにD23～D16をセット*/
    outportb(R_WIN1,u_count.byte_count.count_3);

    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをリードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                        /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_0);                     /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/
```

A

```
do
    {
    if(ad_scsil > 2048)
        _fmemcpy(dataf_b,ad_scsip,0x800);
    else
        _fmemcpy(dataf_b,ad_scsip,(unsigned int)ad_scsil);

    inportb(R_DSTA[cha]);                                   /*DMA STATUS REGISTER ｸﾘｱ*/
    outportb(R_DCBP[cha],0x00);                             /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
    outportb(R_DBCA[cha],u_addr.byte_addr.addr_1);          /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ｱﾄﾞﾚｽ･ﾚｼﾞｽﾀにA15～A0*/
    outportb(R_DBCA[cha],u_addr.byte_addr.addr_2);          /*をｾｯﾄ*/
    outportb(R_DBLP[cha],u_addr.byte_addr.addr_3);          /*ﾍﾞｰｽ･ﾛｳ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀにA23～A16をｾｯﾄ*/
    outportb(R_DBHP[cha],u_addr.byte_addr.addr_4);          /*ﾍﾞｰｽ･ﾊｲ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀにA31～A24をｾｯﾄ*/

    if(ad_scsil > 2048)
        {
        outportb(R_DCBP[cha],0x00);                         /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
        outportb(R_DCWC[cha],0xFF);                         /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD15～D0*/
        outportb(R_DCWC[cha],0x07);                         /*をｾｯﾄ*/
        outportb(R_DHWC[cha],0x00);                         /*ﾍﾞｰｽ･ﾊｲ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD23～D16をｾｯﾄ*/
        }
    else
        {
        u_count.count_1 = ad_scsil - 1;
        outportb(R_DCBP[cha],0x00);                         /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
                                                            /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD15～D0をｾｯﾄ*/
        outportb(R_DCWC[cha],u_count.byte_count.count_1);
        outportb(R_DCWC[cha],u_count.byte_count.count_2);
                                                            /*ﾍﾞｰｽ･ﾊｲ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD23～D16をｾｯﾄ*/
        outportb(R_DHWC[cha],u_count.byte_count.count_3);
        }
    outportb(R_DWSMB[cha],(0x00|(cha&0x03)));               /*DMAﾏｽｸ･ｸﾘｱ*/

    do
        {
        if((inportb(R_CST)&0x80) != 0x80)                   /*cbsy = 0?*/
            {
            inportb(R_DSTA[cha]);                           /*DMA STATUS REGISTER ｸﾘｱ*/
            break;
            }
        }
                                                            /*TC発生?*/
    while((inportb(R_DSTA[cha])&(0x01<<(cha&0x03)))==0x00);

    if(ad_scsil <= 2048)
        {
        while((inportb(R_CST)&0x80) == 0x80);               /*cbsy = 0?*/
        }

    outportb(R_DWSMB[cha],(0x04|(cha&0x03)));               /*DMAﾏｽｸ･ｾｯﾄ*/

    change = (unsigned long)ad_scsip;                       /*ｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽを物理ｱﾄﾞﾚｽに変換する*/
    addr = change & 0x0000FFFFL;
    change = change & 0xFFFF0000L;
    addr = addr + (change >> 12);
```

```
        if((inportb(R_CST)&0x80)==0x80)
            {
            addr = addr + 0x800;                        /*物理ｱﾄﾞﾚｽをｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽに変更*/
            change = addr & 0x0000000FL;
            addr = addr & 0xFFFFFFF0L;
            change = change | (addr << 12);
            ad_scsip = (unsigned char far *)change;     /*ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ更新*/
            ad_scsil = ad_scsil - 0x800;                /*ﾃﾞｰﾀ長更新*/
            }
        else
            {
            outportb(R_ADR,R_CTCL);                     /*CTCLﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_1 = inportb(R_WIN1);
            outportb(R_ADR,R_CTCM);                     /*CTCMﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_2 = inportb(R_WIN1);
            outportb(R_ADR,R_CTCH);                     /*CTCHﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_3 = inportb(R_WIN1);
                                                        /*物理ｱﾄﾞﾚｽをｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽに変更*/
            addr = addr + (ad_scsil - u_count.count_1);
            change = addr & 0x0000000FL;
            addr = addr & 0xFFFFFFF0L;
            change = change | (addr << 12);
            ad_scsip = (unsigned char far *)change;     /*ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ更新*/
            ad_scsil = u_count.count_1;                 /*ﾃﾞｰﾀ長更新*/
            }
        }
    while((inportb(R_CST)&0x80) == 0x80);

    return;
    }


/*************************************************************************
 *              データ・イン・フェーズ・スタート処理                     *
 *ＩＳＴ割り込み要因のデータ・イン・フェーズ・スタートに対応する処理     *
 *                                                                       *
 *      入力：ad_scsil,ad_scsip                                          *
 *      出力：なし                                                       *
 *************************************************************************/


void DATA_IS(void)
    {
    union   b_count                                     /*ad_scsil（データ・バイト数）をμPD72611，DMACにセット*/
            {                                           /*するための共用体*/
            unsigned long count_1;
            struct
                {
                unsigned char count_1;                  /*D7～D0*/
                unsigned char count_2;                  /*D15～D8*/
                unsigned char count_3;                  /*D23～D16*/
                unsigned char count_4;                  /*D31～D24*/
                }   byte_count;
            }   u_count;

    unsigned long change;
    unsigned long addr;

    union   b_addr                                      /*ad_scsip（転送アドレス）をDMACにセットするための共用体*/
            {
            unsigned long  addr_1;                      /*物理アドレス（A31～A0）*/
            struct
                {
                unsigned char addr_1;                   /*A7～A0*/
                unsigned char addr_2;                   /*A15～A8*/
                unsigned char addr_3;                   /*A23～A16*/
                unsigned char addr_4;                   /*A31～A24*/
                }   byte_addr;
            }   u_addr;


    unsigned char cbsy;
```

```
outportb(R_DMOD[cha], (0x04|(cha&0x03)));                /*モード・レジスタ*/
                                                         /*ｼﾝｸﾞﾙ･ﾓｰﾄﾞ, ｱﾄﾞﾚｽ･ｲﾝｸﾘﾒﾝﾄ*/
                                                         /*ｵｰﾄｲﾆｼｬﾗｲｽﾞ 機能ﾃﾞｨｾｰﾌﾞﾙ, I/O→ﾒﾓﾘ転送*/
outportb(R_DEMOD[cha], (0x28|(cha&0x03)));               /*拡張モード・レジスタ*/
                                                         /*ｽﾄｯﾌﾟ･ﾚｼﾞｽﾀ･ﾃﾞｨｾｰﾌﾞﾙ, T-C出力, Type Bｻｲｸﾙ*/
                                                         /*32bit I/O count by byte*/
change = (unsigned long)dataf_b;                         /*ｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽを物理ｱﾄﾞﾚｽに変換する*/
u_addr.addr_l = change & 0x0000FFFFL;
change = change & 0xFFFF0000L;
u_addr.addr_l = u_addr.addr_l + (change >> 12);

u_count.count_l = ad_scsil;                              /*μPD72611のﾍﾞｰｽ転送ｶｳﾝﾀをセットする*/
outportb(R_ADR,R_BTCL);                                  /*BTCLﾚｼﾞｽﾀにD7～D0をセット*/
outportb(R_WIN1,u_count.byte_count.count_1);
outportb(R_ADR,R_BTCM);                                  /*BTCMﾚｼﾞｽﾀにD15～D8をセット*/
outportb(R_WIN1,u_count.byte_count.count_2);
outportb(R_ADR,R_BTCH);                                  /*BTCHﾚｼﾞｽﾀにD23～D16をセット*/
outportb(R_WIN1,u_count.byte_count.count_3);

do
    {
    cbsy = inportb(R_CST);                               /*CSTﾚｼﾞｽﾀをリードする*/
    cbsy = cbsy & 0x80;
    }
while(cbsy == 0x80);                                     /*CBSYﾋﾞｯﾄ == 0?*/

outportb(R_CMD,TRAN_0);                                  /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

do
    {
    inportb(R_DSTA[cha]);                                /*DMA STATUS REGISTER ｸﾘｱ*/
    outportb(R_DCBP[cha],0x00);                          /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
    outportb(R_DBCA[cha],u_addr.byte_addr.addr_1);       /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ｱﾄﾞﾚｽ･ﾚｼﾞｽﾀにA15～A0*/
    outportb(R_DBCA[cha],u_addr.byte_addr.addr_2);       /*をセット*/
    outportb(R_DBLP[cha],u_addr.byte_addr.addr_3);       /*ﾍﾞｰｽ･ﾛｳ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀにA23～A16をセット*/
    outportb(R_DBHP[cha],u_addr.byte_addr.addr_4);       /*ﾍﾞｰｽ･ﾊｲ･ﾍﾟｰｼﾞ･ﾚｼﾞｽﾀにA31～A24をセット*/

    if(ad_scsil > 2048)
        {
        outportb(R_DCBP[cha],0x00);                      /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
        outportb(R_DCWC[cha],0xFF);                      /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD15～D0*/
        outportb(R_DCWC[cha],0x07);                      /*をセット*/
        outportb(R_DHWC[cha],0x00);                      /*ﾍﾞｰｽ･ﾊｲ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD23～D16をセット*/
        }
    else
        {
        u_count.count_l = ad_scsil - 1;
        outportb(R_DCBP[cha],0x00);                      /*ｸﾘｱ･ﾊﾞｲﾄ･ﾎﾟｲﾝﾀ･ｺﾏﾝﾄﾞ*/
                                                         /*ﾍﾞｰｽ8237ｺﾝﾊﾟﾃｨﾌﾞﾙ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD15～D0をセット*/
        outportb(R_DCWC[cha],u_count.byte_count.count_1);
        outportb(R_DCWC[cha],u_count.byte_count.count_2);
                                                         /*ﾍﾞｰｽ･ﾊｲ･ﾜｰﾄﾞ･ｶｳﾝﾄ･ﾚｼﾞｽﾀにD23～D16をセット*/
        outportb(R_DHWC[cha],u_count.byte_count.count_3);
        }
    outportb(R_DWSMB[cha], (0x00|(cha&0x03)));           /*DMAﾏｽｸ･ｸﾘｱ*/

    do
        {
        if((inportb(R_CST)&0x80) != 0x80)                /*cbsy = 0?*/
            {
            inportb(R_DSTA[cha]);                        /*DMA STATUS REGISTER ｸﾘｱ*/
            break;
            }
        }
                                                         /*TC発生?*/
    while((inportb(R_DSTA[cha])&(0x01<<(cha&0x03)))==0x00);

    if(ad_scsil <= 2048)
        {
        while((inportb(R_CST)&0x80) == 0x80);            /*cbsy = 0?*/
        }

    outportb(R_DWSMB[cha], (0x04|(cha&0x03)));           /*DMAﾏｽｸ･ｾｯﾄ*/
```

```
        if(ad_scsil > 2048)
            _fmemcpy(ad_scsip,dataf_b,0x800);
        else
            _fmemcpy(ad_scsip,dataf_b,(unsigned int)ad_scsil);

        change = (unsigned long)ad_scsip;                    /*ｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽを物理ｱﾄﾞﾚｽに変換する*/
        addr = change & 0x0000FFFFL;
        change = change & 0xFFFF0000L;
        addr = addr + (change >> 12);

        if((inportb(R_CST)&0x80)==0x80)
            {
            addr = addr + 0x800;                             /*物理ｱﾄﾞﾚｽをｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽに変更*/
            change = addr & 0x0000000FL;
            addr = addr & 0xFFFFFFF0L;
            change = change | (addr << 12);
            ad_scsip = (unsigned char far *)change;          /*ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ更新*/
            ad_scsil = ad_scsil - 0x800;                     /*ﾃﾞｰﾀ長更新*/
            }
        else
            {
            outportb(R_ADR,R_CTCL);                          /*CTCLﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_1 = inportb(R_WIN1);
            outportb(R_ADR,R_CTCM);                          /*CTCMﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_2 = inportb(R_WIN1);
            outportb(R_ADR,R_CTCH);                          /*CTCHﾚｼﾞｽﾀをﾘｰﾄﾞ*/
            u_count.byte_count.count_3 = inportb(R_WIN1);
                                                             /*物理ｱﾄﾞﾚｽをｾｸﾞﾒﾝﾄ/ｵﾌｾｯﾄ･ｱﾄﾞﾚｽに変更*/
            addr = addr + (ad_scsil- u_count.count_1);
            change = addr & 0x0000000FL;
            addr = addr & 0xFFFFFFF0L;
            change = change | (addr << 12);
            ad_scsip = (unsigned char far *)change;          /*ﾃﾞｰﾀscsiﾎﾟｲﾝﾀ更新*/
            ad_scsil = u_count.count_1;                      /*ﾃﾞｰﾀ長更新*/
            }
        }
    while((inportb(R_CST)&0x80) == 0x80);

    return;
    }


/*************************************************************************
 *                    ｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｽﾀｰﾄ処理                          *
 *ＩＳＴ割り込み要因のｺﾏﾝﾄﾞ･ﾌｪｰｽﾞ･ｽﾀｰﾄに対応する処理                *
 *                                                                       *
 *      入力：ac_scsil,ac_scsip                                          *
 *      出力：なし                                                       *
 *************************************************************************/


void COMMAND_S(void)
    {
    unsigned char c_scsic;                                   /*scsiｺﾏﾝﾄﾞ･ｶｳﾝﾀ*/
    unsigned char cbsy;
    unsigned char buff_c;                                    /*dward_buffのｶｳﾝﾀ*/

    union   command_buff                                     /*μPD72611に32ﾋﾞｯﾄでﾗｲﾄするための共用体*/
        {
        unsigned long dword_buff;
        unsigned char buff[4];
        }   u_buff;

    outportb(R_ADR,R_BTCL);                                  /*μPD72611のﾍﾞｰｽ転送ｶｳﾝﾀをセットする*/
    outportb(R_WIN1,ac_scsil);                               /*BTCLﾚｼﾞｽﾀにコマンド長をセット*/

    do
        {
        cbsy = inportb(R_CST);                               /*CSTﾚｼﾞｽﾀをリードする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                                     /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_2);                                  /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/
```

```
    c_scsic = 0x00;                                     /*c_scsicをﾘｾｯﾄする*/
    do
        {
        cbsy = inportb(R_CST);                          /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        if((cbsy & 0x81) == 0x81)
            {
            buff_c = 0x00;                              /*buff_cをﾘｾｯﾄする*/
            while(buff_c != 4)                          /*μPD72611のDF2ﾚｼﾞｽﾀにﾗｲﾄするため8ﾋﾞｯﾄで構成されている*/
                {                                       /*SCSIｺﾏﾝﾄﾞを32ﾋﾞｯﾄに変換している*/
                if(ac_scsil == c_scsic)
                    break;
                u_buff.buff[buff_c] = *(ac_scsip + c_scsic);
                ++c_scsic;
                ++buff_c;
                }
            w_port = R_DF2;                             /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
            w_data = u_buff.dword_buff;                 /*DFﾚｼﾞｽﾀにﾗｲﾄするｺﾏﾝﾄﾞをｾｯﾄ*/
            outportdw();                                /*DFﾚｼﾞｽﾀにﾗｲﾄ*/
            }
        if(ac_scsil == c_scsic)
            break;
        }
    while((cbsy & 0x80) == 0x80);                       /*CBSYﾋﾞｯﾄ == 0?*/

    do
        {
        cbsy = inportb(R_CST);                          /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x80) == 0x80);                       /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_ADR,R_CTCL);                             /*CTCLﾚｼﾞｽﾀをﾘｰﾄﾞ*/
    ac_scsip = ac_scsip + (ac_scsil - inportb(R_WIN1));  /*ｺﾏﾝﾄﾞscsiﾎﾟｲﾝﾀ更新*/
    outportb(R_ADR,R_CTCL);                             /*CTCLﾚｼﾞｽﾀをﾘｰﾄﾞ*/
    ac_scsil =  inportb(R_WIN1);                        /*ﾃﾞｰﾀ長更新*/

    return;
    }


/*************************************************************************
 *                  ステータス・フェーズ・スタート処理                   *
 *ＩＳＴ割り込み要因のステータス・フェーズ・スタートに対応する処理       *
 *                                                                       *
 *      入力：as_scsip                                                   *
 *      出力：なし                                                       *
 *************************************************************************/


void STATUS_S(void)
    {
    unsigned char cbsy;

    do
        {
        cbsy = inportb(R_CST);                          /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                                /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_3);                             /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

    do
        {
        cbsy = inportb(R_CST);                          /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x81) != 0x81);

    r_port = R_DF2;                                     /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
    inportdw();                                         /*DFﾚｼﾞｽﾀをﾘｰﾄﾞ*/
    *as_scsip = (char)r_data;                           /*as_scsipにﾘｰﾄﾞしたｽﾃｰﾀｽをｾｯﾄ*/
```

```
    do
        {
        cbsy = inportb(R_CST);                          /*CSTﾚｼﾞｽﾀをリードする*/
        }
    while((cbsy & 0x80) == 0x80);                       /*CBSYﾋﾞｯﾄ == 0?*/

    ++as_scsip;                                         /*ｽﾃｰﾀｽscsiﾎﾟｲﾝﾀ更新*/

    return;
    }

/*********************************************************************
 *              メッセージ・アウト・フェーズ・スタート処理           *
 *ＩＳＴ割り込み要因のメッセージ・アウト・フェーズ・スタートに対応する *
 *処理                                                               *
 *      入力：message_o, i_message, expect_priod, expect_offset      *
 *            t_id, sd_scsi[], sd_scsip[], sc_scsil[], sc_scsip[], ss_scsip[]*
 *      出力：error, messagepe_c, sync[], pmessage_c, ad_scsil, ad_scsip *
 *            ac_scsil, ac_scsip, as_scsip                           *
 *********************************************************************/


void MESSAGE_OS(void)
    {
    unsigned char cbsy;
    unsigned long sdtrm[2];

    union   s_m                                         /*syncﾒｯｾｰｼﾞを32ﾋﾞｯﾄ単位にするための共用体*/
        {
        unsigned long   dword_s;
        unsigned char   byte_s[4];
        }   u_sm;
    unsigned char m_scsic;
    unsigned char c;

    switch(message_o)                                   /*出力するメッセージ*/
        {
        case    SYNCHRONOUS:                            /*SYNCHRONOUS DATA TRANSFER REQUEST*/
            u_sm.byte_s[0] = 0x01;                      /*expect_priod, expect_offsetを基にsdtrm[]*/
            u_sm.byte_s[1] = 0x03;                      /*(SYNCHRONOUS DATA TRANSFER REQUEST)を生成する*/
            u_sm.byte_s[2] = 0x01;
            u_sm.byte_s[3] = expect_priod[t_id];
            sdtrm[0] = u_sm.dword_s;
            u_sm.byte_s[0] = expect_offset[t_id];
            sdtrm[1] = u_sm.dword_s;

            c = 0;                                      /*μPD72611を同期転送モードにセットする*/
            while(c < 11)
                {
                if(expect_priod[t_id] == expect_sync[c][0])
                    break;
                else
                    ++c;
                }
            outportb(R_ADR, R_TMOD);
            outportb(R_WIN1, (expect_sync[c][1]|(expect_offset[t_id]&0x07)));
            sync[t_id] = 0x01;                          /*syncを同期モードにｾｯﾄ*/

            outportb(R_ADR, R_BTCL);                    /*μPD72611に転送数（５バイト）をセットする*/
            outportb(R_WIN1, 0x05);
            do
                {
                cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをリードする*/
                cbsy = cbsy & 0x80;
                }
            while(cbsy == 0x80);                        /*CBSYﾋﾞｯﾄ == 0?*/

            outportb(R_CMD, TRAN_2);                    /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/
```

```
    m_scsic = 0x00;                          /*m_scsicｶｳﾝﾀをﾘｾｯﾄする*/
    do
        {
        cbsy = inportb(R_CST);               /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        if((cbsy & 0x81) == 0x81);
            {
            w_port = R_DF2;                  /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
            w_data = sdtrm[m_scsic];         /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
            outportdw();                     /*DFﾚｼﾞｽﾀにﾗｲﾄ*/
            ++m_scsic;
            }
        if(m_scsic == 0x02)                  /*転送終了か？*/
            break;
        }
    while((cbsy & 0x80) == 0x80);            /*CBSYﾋﾞｯﾄ == 0?*/
    do
        {
        cbsy = inportb(R_CST);               /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x80) == 0x80);            /*CBSYﾋﾞｯﾄ == 0?*/

    break;

case    I_DETECTED:                          /*INITIATOR DETECTED ERROR*/
    if((i_message & INITIDE_F) == 0x01)      /*INITIATOR_DEﾌﾗｸﾞ == 1?*/
        {
        message_o = NO_OPE;                  /*NO OPERATION*/
        error = error|INITDE_F;              /*errorにINITIATOR_DEﾌﾗｸﾞをｾｯﾄ*/
        }
    else
        {
        ad_scsil = sd_scsil[t_id];           /*ｾｰﾌﾞscsiﾎﾟｲﾝﾀの内容をｱｸﾃｨﾌﾞscsiﾎﾟｲﾝﾀにｾｯﾄする*/
        ad_scsip = sd_scsip[t_id];
        ac_scsil = sc_scsil[t_id];
        ac_scsip = sc_scsip[t_id];
        as_scsip = ss_scsip[t_id];
        }

    do
        {
        cbsy = inportb(R_CST);               /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                     /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_3);                  /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

    do
        {
        cbsy = inportb(R_CST);               /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x81) != 0x81);

    w_port = R_DF2;                          /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
    w_data = message_o;                      /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
    outportdw();                             /*DFﾚｼﾞｽﾀにﾗｲﾄ*/

    do
        {
        cbsy = inportb(R_CST);               /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x80) == 0x80);            /*CBSYﾋﾞｯﾄ == 0?*/

    break;
```

```
case    ABORT:                                  /*ABORT*/
    error = error | ABORT_F;                    /*errorにABORTﾌﾗｸﾞをｾｯﾄする*/

case    MESS_REJ:                               /*MESSAGE REJECT*/

case    NO_OPE:                                 /*NO OPERATION*/
    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                        /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_3);                     /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x81) != 0x81);

    w_port = R_DF2;                             /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
    w_data = message_o;                         /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
    outportdw();                                /*DFﾚｼﾞｽﾀにﾗｲﾄ*/

    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x80) == 0x80);               /*CBSYﾋﾞｯﾄ == 0?*/

    break;

case    MESS_PARE:                              /*MESSAGE PARITY ERROR*/
    pmessage_c = 0x00;                          /*pmessage_cをﾘｾｯﾄする*/
    if((i_message&MESSPE_F) == 0x02)            /*MESSAGE_PEﾌﾗｸﾞ == 1?*/
        {
        message_o = NO_OPE;                     /*NO OPERATION*/
        error = error|MESSAGEPE_F;              /*errorにMESSAGE_PEﾌﾗｸﾞをｾｯﾄ*/
        }
    else
        {
        if(messagepe_c != 0x10)                 /*MESSAGE PARITY ERRORの転送回数をカウント*/
            ++messagepe_c;
        else
            {
            message_o = ABORT;                  /*message_oにABORTﾒｯｾｰｼﾞをｾｯﾄ*/
            error = error | ABORT_F;            /*errorにABORTﾌﾗｸﾞをｾｯﾄする*/
            messagepe_c = 0x00;                 /*messagepe_cをﾘｾｯﾄ*/
            }
        }
    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                        /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_3);                     /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

    do
        {
        cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x81) != 0x81);

    w_port = R_DF2;                             /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
    w_data = message_o;                         /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
    outportdw();                                /*DFﾚｼﾞｽﾀにﾗｲﾄ*/
```

```
        do
            {
            cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
            }
        while((cbsy & 0x80) == 0x80);               /*CBSYﾋﾞｯﾄ == 0?*/

        break;

    case    BUS_D_R:                                /*BUS DEVICE RESET*/
        sync[t_id] = 0x00;                          /*転送ﾓｰﾄﾞを不定にする*/
        outportb(R_ADR,R_TMOD);                     /*μPD72611を非同期ﾓｰﾄﾞにｾｯﾄする*/
        outportb(R_WIN1,0x00);
        error = error | BUSDR_F;                    /*errorにDUS_DRﾌﾗｸﾞをｾｯﾄする*/

        do
            {
            cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
            cbsy = cbsy & 0x80;
            }
        while(cbsy == 0x80);                        /*CBSYﾋﾞｯﾄ == 0?*/

        outportb(R_CMD,TRAN_3);                     /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

        do
            {
            cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
            }
        while((cbsy & 0x81) != 0x81);

        w_port = R_DF2;                             /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
        w_data = message_o;                         /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
        outportdw();                                /*DFﾚｼﾞｽﾀにﾗｲﾄ*/

        do
            {
            cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
            }
        while((cbsy & 0x80) == 0x80);               /*CBSYﾋﾞｯﾄ == 0?*/

        break;

    default:
        if((message_o & 0xF8) == IDENTIFY1)         /*IDENTIFY*/
            {
            if((i_message&IDEN_F) == 0x04)          /*IDENTIFYﾌﾗｸﾞ == 1?*/
                {
                message_o = NO_OPE;                 /*NO OPERATION*/
                error = error|IDENTIFY_F;           /*errorにIDENTIFYﾌﾗｸﾞをｾｯﾄ*/
                }

            do
                {
                cbsy = inportb(R_CST);              /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
                cbsy = cbsy & 0x80;
                }
            while(cbsy == 0x80);                    /*CBSYﾋﾞｯﾄ == 0?*/

            outportb(R_CMD,TRAN_3);                 /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

            do
                {
                cbsy = inportb(R_CST);              /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
                }
            while((cbsy & 0x81) != 0x81);

            w_port = R_DF2;                         /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
            w_data = message_o;                     /*DFﾚｼﾞｽﾀにﾗｲﾄするﾒｯｾｰｼﾞをｾｯﾄ*/
            outportdw();                            /*DFﾚｼﾞｽﾀにﾗｲﾄ*/
```

```
                do
                    {
                    cbsy = inportb(R_CST);                  /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
                    }
                while((cbsy & 0x80) == 0x80);               /*CBSYﾋﾞｯﾄ == 0?*/
                }

            break;

        }
    return;
    }


/*************************************************************************
 *              ﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄ処理                              *
 *ＩＳＴ割り込み要因のﾒｯｾｰｼﾞ･ｲﾝ･ﾌｪｰｽﾞ･ｽﾀｰﾄに対応する処理*
 *                                                                       *
 *      入力：なし                                                       *
 *      出力：message_i                                                  *
 *************************************************************************/


void MESSAGE_IS(void)
    {
    unsigned char cbsy;

    do
        {
        cbsy = inportb(R_CST);                              /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        cbsy = cbsy & 0x80;
        }
    while(cbsy == 0x80);                                    /*CBSYﾋﾞｯﾄ == 0?*/

    outportb(R_CMD,TRAN_3);                                 /*μPD72611にTRANSFERｺﾏﾝﾄﾞを発行する*/

    do
        {
        cbsy = inportb(R_CST);                              /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x81) != 0x81);

    r_port = R_DF2;                                         /*μPD72611のDFﾚｼﾞｽﾀのｱﾄﾞﾚｽをｾｯﾄ*/
    inportdw();                                             /*DFﾚｼﾞｽﾀをﾘｰﾄﾞ*/
    message_i = (unsigned char)r_data;                      /*as_scsipにﾘｰﾄﾞしたｽﾃｰﾀｽをｾｯﾄ*/

    do
        {
        cbsy = inportb(R_CST);                              /*CSTﾚｼﾞｽﾀをﾘｰﾄﾞする*/
        }
    while((cbsy & 0x80) == 0x80);                           /*CBSYﾋﾞｯﾄ == 0?*/

    return;
    }
```

保守／廃止

```
/*************************************************************************
 *                    メッセージ受信処理                                 *
 *ＩＳＴ割り込み要因のメッセージ受信に対応した処理                       *
 *      入力：pmessage_c, message_i, expect_priod, expect_offset, message_o  *
 *            t_id, sd_scsi[], sd_scsip[], sc_scsil[], sc_scsip[], ss_scsip[] *
 *            ad_scsil, ad_scsip, emessage_c                               *
 *      出力：pmessage_c, message_o, sync[], i_message, sd_scsil[], sd_scsip[] *
 *            ad_scsil, ad_scsip, ac_scsil, ac_scsip, as_scsip, emessage_c *
 *************************************************************************/


void MESSAGE_R(void)
    {
    unsigned char sbper;
    unsigned static char emessage_i[10];                    /*拡張ﾒｯｾｰｼﾞの領域*/
    unsigned static char emessage_l;
    unsigned char c;

    if(pmessage_c != 0x00)                                  /*ﾊﾟﾘﾃｨ･ｴﾗｰが発生したメッセージを受信中か？*/
        {
        ++pmessage_c;                                       /*ﾊﾟﾘﾃｨ･ｴﾗｰ･ﾒｯｾｰｼﾞ･ｶｳﾝﾀをｲﾝｸﾘﾒﾝﾄ*/
        outportb(R_CMD, RESET_ACK);                         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
        }
    else
        {
        sbper = inportb(R_EXST);                            /*μＰＤ７２６１１のＥＸＳＴレジスタを*/
        sbper = sbper & 0x01;                               /*ﾘｰﾄﾞしSCSIﾊﾞｽ･ﾊﾟﾘﾃｨ･ｴﾗｰを確認する*/
        if(sbper == 0x01)
            {
            ++pmessage_c;                                   /*ﾊﾟﾘﾃｨ･ｴﾗｰ･ﾒｯｾｰｼﾞ･ｶｳﾝﾀをｲﾝｸﾘﾒﾝﾄ*/
            message_o = MESS_PARE;                          /*MESSAGE PARITY ERRORをｾｯﾄ*/
            emessage_c = 0x00;                              /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄする*/
            outportb(R_CMD, RESET_ACK);                     /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
            }                                               /*ATN信号はμPD72611が自動的にｱｻｰﾄする*/
        else
            {
            messagepe_c = 0x00;                             /*ﾒｯｾｰｼﾞ･ﾊﾟﾘﾃｨ･ｴﾗｰ･ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄする*/
            if(emessage_c == 0x00)                          /*最初に受信したメッセージか？*/
                {
                switch(message_i)
                    {
                    case    EXTEND_ME:                      /*EXTENDED MESSAGE*/
                        emessage_i[emessage_c] = message_i; /*拡張ﾒｯｾｰｼﾞ領域にｾｯﾄ*/
                        ++emessage_c;                       /*emessage_cｲﾝｸﾘﾒﾝﾄ*/
                        message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
                        outportb(R_CMD, RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                        break;

                    case    SAVE_DP:                        /*SAVE DATA POINTER*/
                        sd_scsil[t_id] = ad_scsil;          /*ｱｸﾃｨﾌﾞ･ﾃﾞｰﾀscsiﾎﾟｲﾝﾀの内容をｾｰﾌﾞ･ﾃﾞｰﾀscsi*/
                        sd_scsip[t_id] = ad_scsip;          /*ﾎﾟｲﾝﾀにｾｯﾄする*/
                        message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
                        outportb(R_CMD, RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                        break;

                    case    MESS_REJ:                       /*MESSAGE REJECT*/
                        switch(message_o)
                            {
                            case    I_DETECTED:             /*INITIATOR DETECTED ERROR*/
                                                            /*INITIATOR_DEﾌﾗｸﾞをｾｯﾄ*/
                                i_message = i_message|INITIDE_F;
                                break;

                            case    MESS_PARE:              /*MESSAGE PARITY ERROR*/
                                                            /*MESSAGE_PEﾌﾗｸﾞをｾｯﾄ*/
                                i_message = i_message|MESSPE_F;
                                break;

                            case    IDENTIFY1:              /*IDENTIFY*/
                                                            /*IDENTIFYﾌﾗｸﾞをｾｯﾄ*/
                                i_message = i_message|IDEN_F;
                                break;
```

```
                case    SYNCHRONOUS:            /*SYNCHRONOUS TRANSFER*/
                                                /*SYNCHRONOUSﾌﾗｸﾞをｾｯﾄ*/
                    i_message = i_message|SYNC_F;
                    sync[t_id] = 0x02;          /*syncに非同期転送をｾｯﾄ*/
                    outportb(R_ADR, R_TMOD);    /*μPD72611を非同期モードにｾｯﾄする*/
                    outportb(R_WIN1, 0x00);
                    break;

                default:
                    break;
                }
            message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
            outportb(R_CMD, RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
            break;

        case    RESTORE_P:                      /*RESTORE POINTER*/
            ad_scsil = sd_scsil[t_id];          /*ｾｰﾌﾞscsiﾎﾟｲﾝﾀの内容を*/
            ad_scsip = sd_scsip[t_id];          /*ｱｸﾃｨﾌﾞscsiﾎﾟｲﾝﾀにｾｯﾄする*/
            ac_scsil = sc_scsil[t_id];
            ac_scsip = sc_scsip[t_id];
            as_scsip = ss_scsip[t_id];
            message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
            outportb(R_CMD, RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
            break;

        case    DISCON:                         /*DISCONNECT*/
        case    COMMAND_CP:                     /*COMMAND COMPLETE*/
            message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
            outportb(R_CMD, RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
            break;

        default:
            if((message_i & 0xF8) == IDENTIFY2)
                {
                ad_scsil = sd_scsil[t_id];      /*ｾｰﾌﾞscsiﾎﾟｲﾝﾀの内容を*/
                ad_scsip = sd_scsip[t_id];      /*ｱｸﾃｨﾌﾞscsiﾎﾟｲﾝﾀにｾｯﾄする*/
                ac_scsil = sc_scsil[t_id];
                ac_scsip = sc_scsip[t_id];
                as_scsip = ss_scsip[t_id];
                message_o = NO_OPE;             /*NO OPERATIONをｾｯﾄ*/
                outportb(R_CMD, RESET_ACK);     /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                break;
                }
            else                                /*ﾉﾝ･ｻﾎﾟｰﾄ･ﾒｯｾｰｼﾞ*/
                {
                message_o = MESS_REJ;           /*MESSAGE REJECT*/
                outportb(R_CMD, SET_ATN);       /*μPD72611にSET ATNｺﾏﾝﾄﾞを発行する*/
                outportb(R_CMD, RESET_ACK);     /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                break;
                }
        }
    }
else
    {
    if(emessage_c == 0x01)
        {
        message_o = NO_OPE;                     /*NO OPERATIONをｾｯﾄ*/
        emessage_i[emessage_c] = message_i;     /*拡張ﾒｯｾｰｼﾞ領域にｾｯﾄ*/
        emessage_l = emessage_c + message_i;    /*拡張ﾒｯｾｰｼﾞ長をｾｯﾄ*/
        ++emessage_c;
        outportb(R_CMD, RESET_ACK);             /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
        }
```

保守／廃止

```
else
    {
    if(emessage_l != emessage_c)
        {
        message_o = NO_OPE;                 /*NO OPERATIONをｾｯﾄ*/
        emessage_i[emessage_c] = message_i; /*拡張ﾒｯｾｰｼﾞ領域にｾｯﾄ*/
        ++emessage_c;
        outportb(R_CMD,RESET_ACK);          /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
        }
    else
        {
        emessage_i[emessage_c] = message_i; /*拡張ﾒｯｾｰｼﾞ領域にｾｯﾄ*/
        if(emessage_i[2] != 0x01)           /*SYNCHRONOUS DATA TRANSFER REQUEST?*/
            {
            message_o = MESS_REJ;           /*MESSAGE REJECT*/
            emessage_c = 0x00;              /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄ*/
            outportb(R_CMD,SET_ATN);        /*μPD72611にSET ATNｺﾏﾝﾄﾞを発行する*/
            outportb(R_CMD,RESET_ACK);      /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
            }
        else
            {
            if(emessage_i[4] == 0x00)
                {
                message_o = NO_OPE;         /*NO OPERATIONをｾｯﾄ*/
                sync[t_id] = 0x02;          /*syncを非同期ﾓｰﾄﾞにｾｯﾄ*/
                outportb(R_ADR,R_TMOD);     /*μPD72611を非同期転送にｾｯﾄ*/
                outportb(R_WIN1,0x00);
                emessage_c = 0x00;          /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄ*/
                outportb(R_CMD,RESET_ACK);  /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                }
            else
                {
                if(expect_priod[t_id] == emessage_i[3])
                    {
                    expect_offset[t_id] = emessage_i[4];
                    sync[t_id] = 0x01;      /*syncを同期ﾓｰﾄﾞにｾｯﾄ*/
                    c = 0;                  /*μPD72611を同期転送ﾓｰﾄﾞにｾｯﾄする*/
                    while(c < 11)
                        {
                        if(expect_priod[t_id] == expect_sync[c][0])
                            break;
                        else
                            ++c;
                        }
                    outportb(R_ADR,R_TMOD);
                    outportb(R_WIN1,(expect_sync[c][1]|(expect_offset[t_id]&0x07)));
                    emessage_c = 0x00;      /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄ*/
                                            /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                    outportb(R_CMD,RESET_ACK);
                    }
                else
                    {
                    if(expect_priod[t_id] < emessage_i[3])
                        {
                        expect_offset[t_id] = emessage_i[4];
                                            /*SYNCHRONOUSをｾｯﾄ*/
                        message_o = SYNCHRONOUS;
                        sync[t_id] = 0x00;  /*syncを不定ﾓｰﾄﾞにｾｯﾄ*/
                        c = 0;
                        while(c < 11)
                            {
                            if(emessage_i[3] <= expect_sync[c][0])
                                ++c;
                            else
                                break;
                            }
                        expect_priod[t_id] = expect_sync[c-1][0];
                                            /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄ*/
                        emessage_c = 0x00;
                                            /*μPD72611にSET ATNｺﾏﾝﾄﾞを発行する*/
                        outportb(R_CMD,SET_ATN);
                                            /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                        outportb(R_CMD,RESET_ACK);
                        }
```

保守／廃止

```
                                else
                                    {
                                    message_o = MESS_REJ;              /*MESSAGE REJECT*/
                                    sync[t_id] = 0x02;                 /*syncを非同期モードにｾｯﾄ*/
                                    outportb(R_ADR,R_TMOD);            /*μPD72611を非同期転送にｾｯﾄ*/
                                    outportb(R_WIN1,0x00);
                                    emessage_c = 0x00;                 /*拡張ﾒｯｾｰｼﾞ･ｶｳﾝﾀをﾘｾｯﾄ*/
                                    outportb(R_CMD,SET_ATN);           /*μPD72611にSET ATNｺﾏﾝﾄﾞを発行する*/
                                    outportb(R_CMD,RESET_ACK);         /*μPD72611にRESET ACKｺﾏﾝﾄﾞを発行する*/
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
return;
}
```

保守／廃止

```
;*************************************************************************
;*      ３２ビット入出力ポート処理      （ＩＯ_ＰＯＲＴ．ＡＳＭ）        *
;*************************************************************************

.386

DGROUP GROUP _TEXT,_DATA
    ASSUME  CS:_TEXT,DS:DGROUP


_DATA   SEGMENT DWORD PUBLIC USE16 'DATA'

PUBLIC  _dataf_b
PUBLIC  _w_data
PUBLIC  _w_port
PUBLIC  _r_data
PUBLIC  _r_port

ALIGN 4
_dataf_b    DB  512*4   dup(?)                  ;ﾃﾞｰﾀ･ﾌｪｰｽﾞにおけるﾊﾞｯﾌｧ
_w_data     DD  ?                               ;32ﾋﾞｯﾄ･ﾗｲﾄ･ﾃﾞｰﾀ
_w_port     DW  ?                               ;ﾗｲﾄ･I/Oﾎﾟｰﾄ･ｱﾄﾞﾚｽ
_r_data     DD  ?                               ;32ﾋﾞｯﾄ･ﾘｰﾄﾞ･ﾃﾞｰﾀ
_r_port     DW  ?                               ;ﾘｰﾄﾞ・I/Oﾎﾟｰﾄ･ｱﾄﾞﾚｽ

_DATA   ENDS


_TEXT SEGMENT DWORD PUBLIC  USE16 'CODE'

PUBLIC  _outportdw
PUBLIC  _inportdw


_outportdw  PROC    NEAR                        ;32ﾋﾞｯﾄｱｳﾄﾌﾟｯﾄ処理

            push    dx
            push    eax
            mov     dx,DGROUP:_w_port
            mov     eax,DGROUP:_w_data
            out     dx,eax
            pop     eax
            pop     dx
            ret

_outportdw  ENDP



_inportdw   PROC    NEAR                        ;32ﾋﾞｯﾄｲﾝﾌﾟｯﾄ処理

            push    dx
            push    eax
            mov     dx,DGROUP:_r_port
            in      eax,dx
            mov     DGROUP:_r_data,eax
            pop     eax
            pop     dx
            ret

_inportdw   ENDP

_TEXT   ENDS

END
```

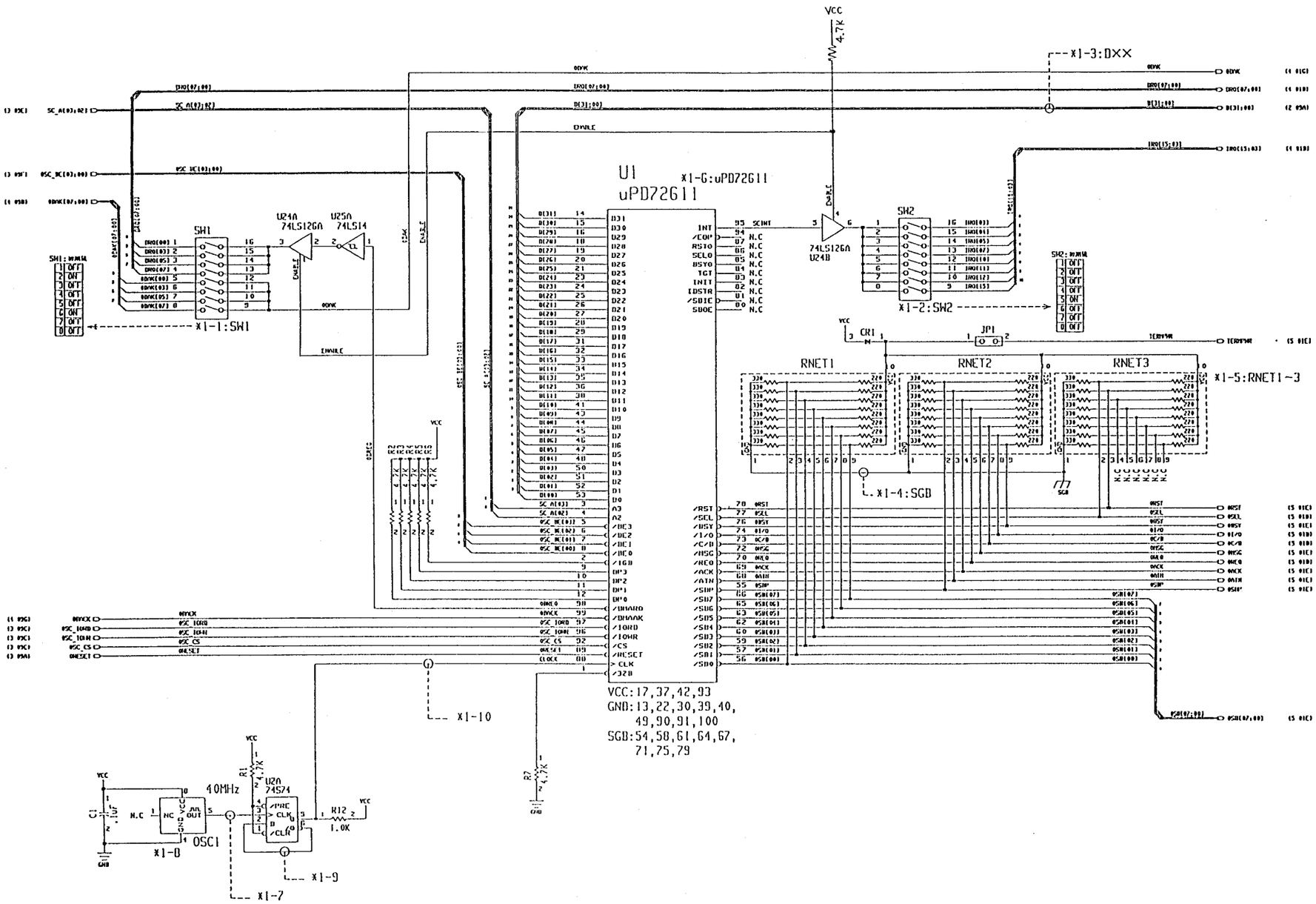
付録B　回 路 図

μPD72611ホスト・アダプタ・ボードの回路図，PLDソース・リスト，PLD内部の回路図を示します。

保守／廃止

図B－1　μPD72611ホスト・アダプタ・ボードの回路図（1/5）

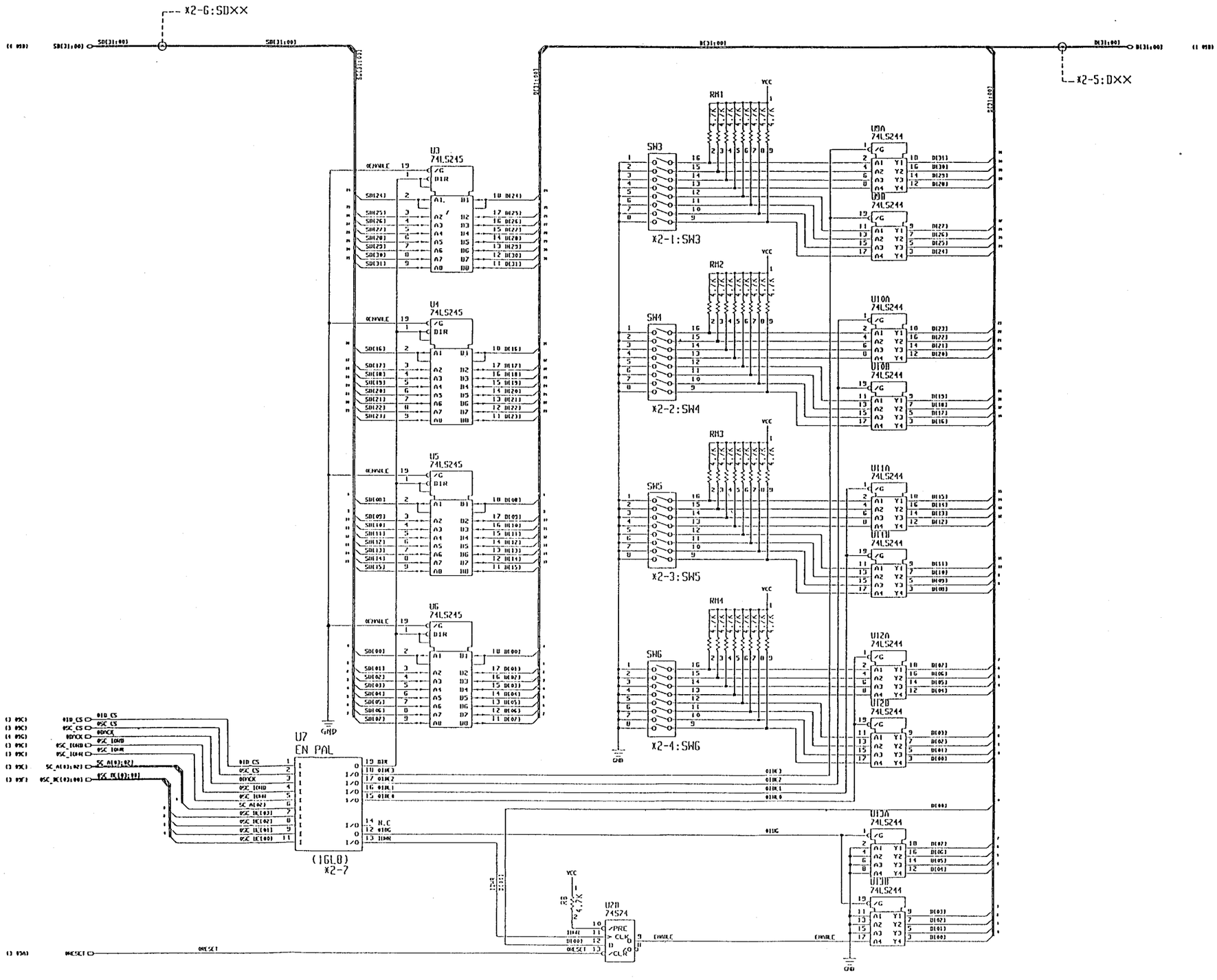
（図B－1　μPD72611ホスト・アダプタ・ボードの回路図（1/5））



B

図B－1　μPD72611ホスト・アダプタ・ボードの回路図（2/5）

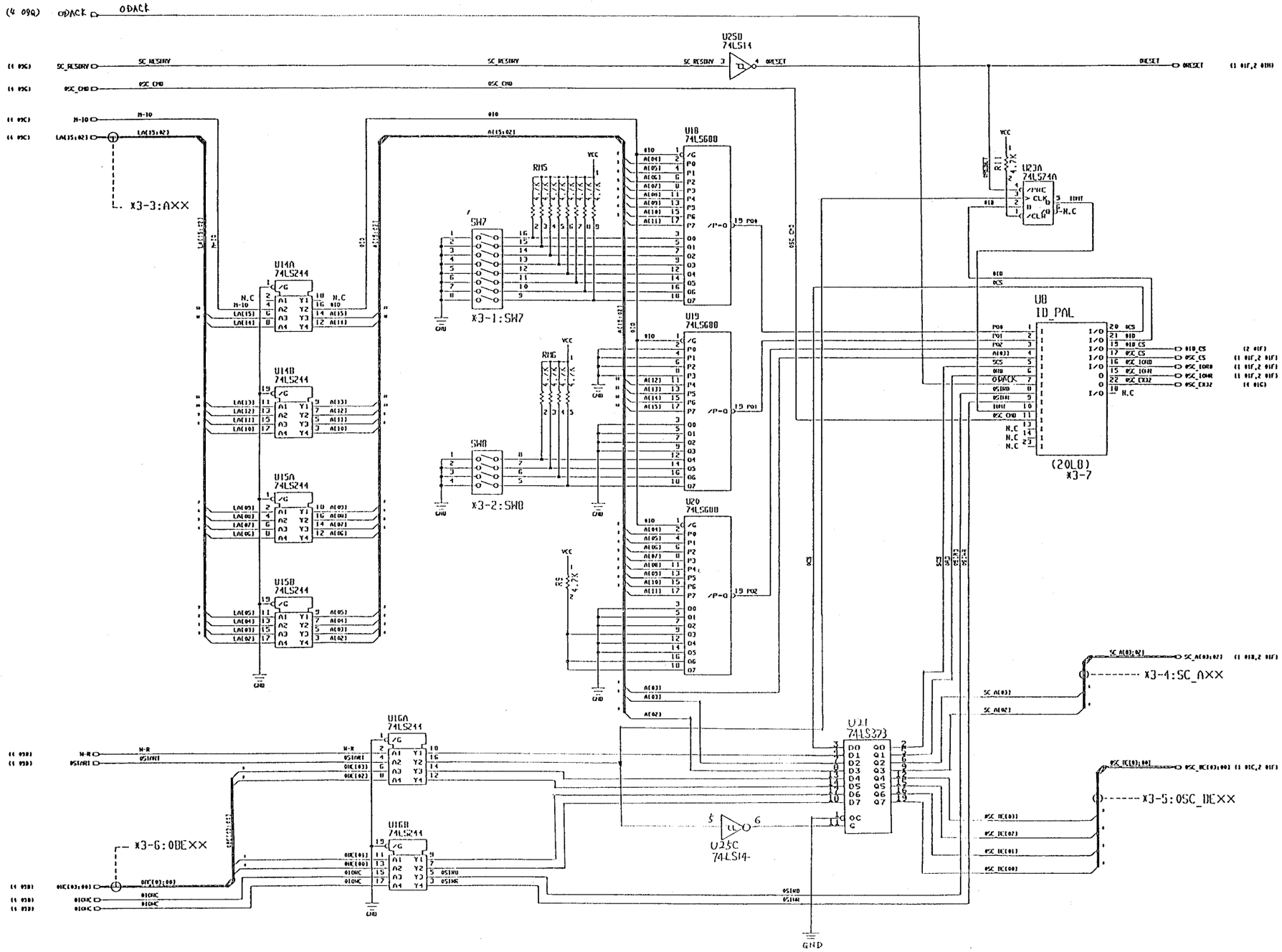
（図B－1　μPD72611ホスト・アダプタ・ボードの回路図（2/5））



保守／廃止

図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（3/5）

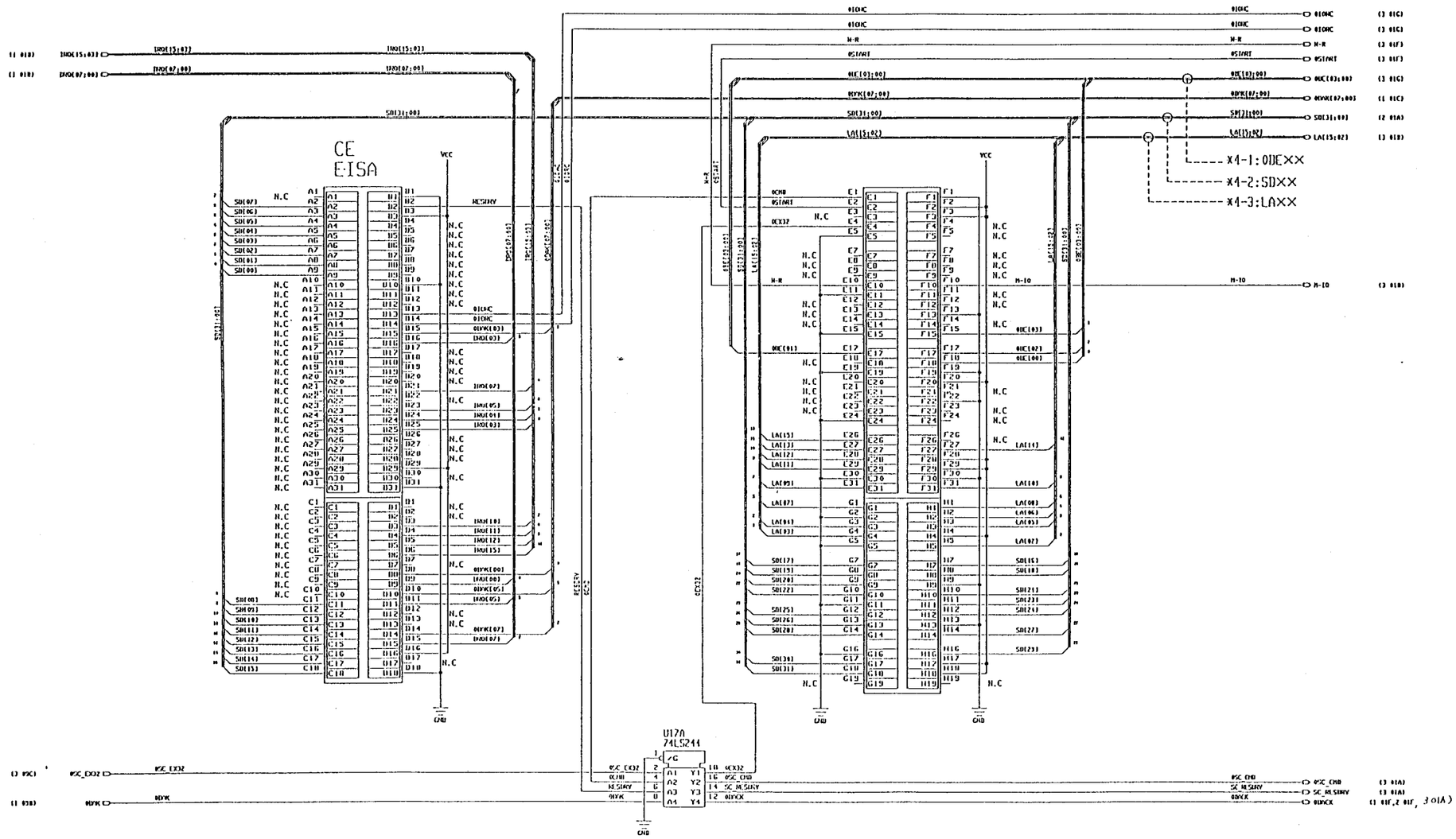
（図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（3/5））



B

図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（4/5）

（図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（4/5））

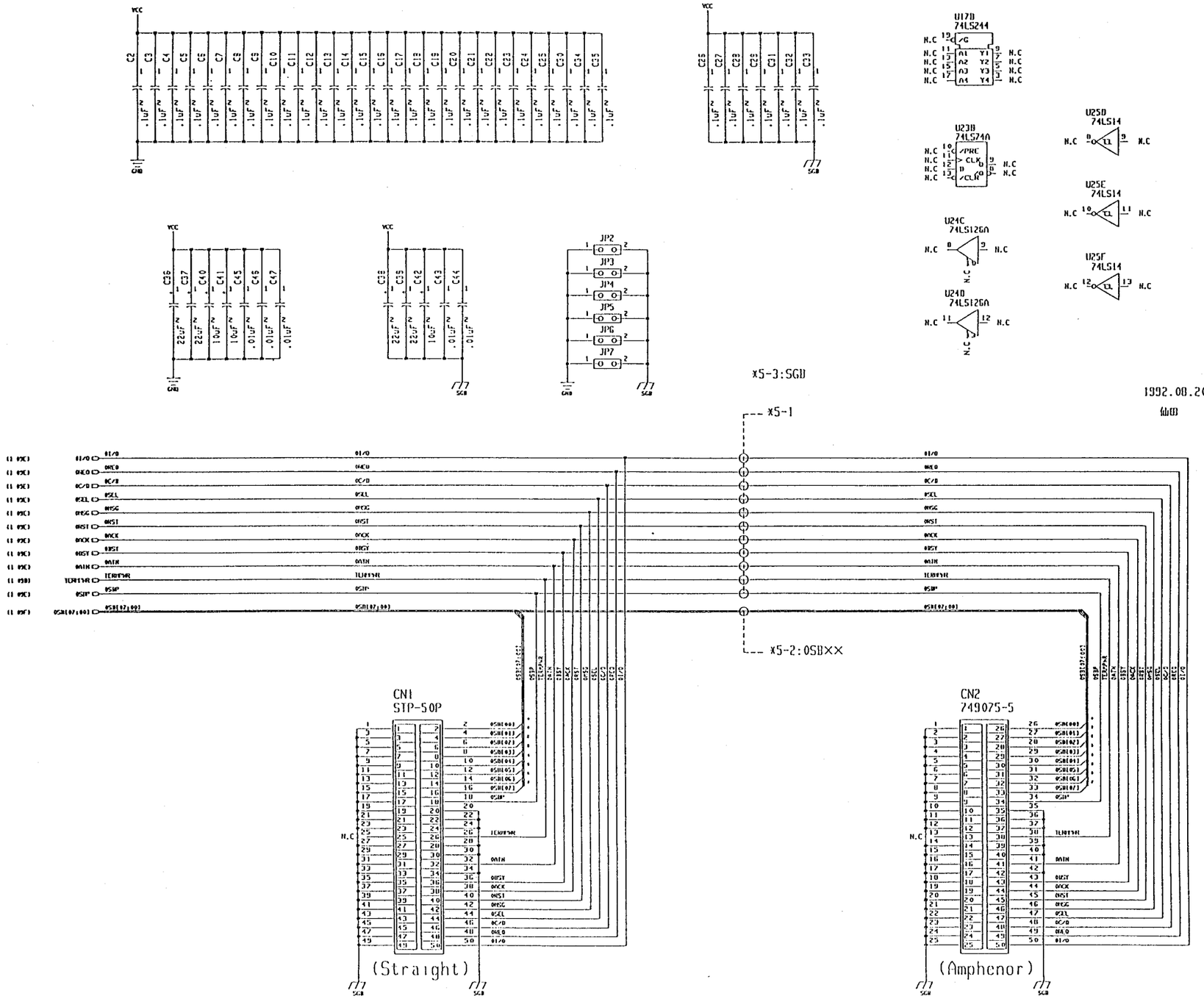


B

保守／廃止

図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（5/5）

（図B－1　μPD72611 ホスト・アダプタ・ボードの回路図（5/5））



B

保守／廃止

PDL ソース・リスト EN_PAL

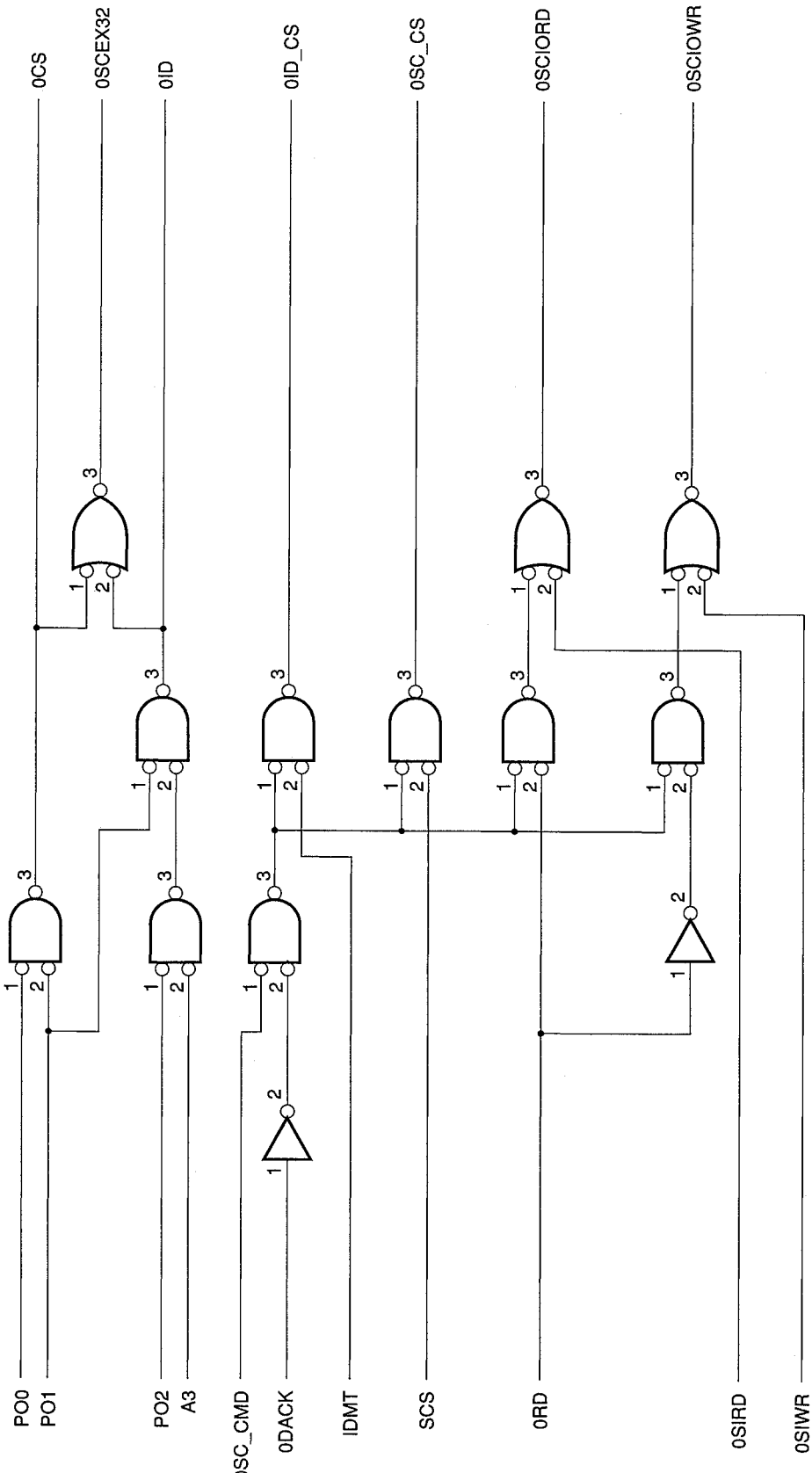
```
module en_pal
title'NEC 21-4-1992'

EN_PAL  device'P16V8C';

"Define input
        OID_CS          pin 1;
        OSC_CS          pin 2;
        ODACK           pin 3;
        OSCIORD         pin 4;
        OSCIOWR         pin 5;
        SC_A02          pin 6;
        OSC_BE3         pin 7;
        OSC_BE2         pin 8;
        OSC_BE1         pin 9;
        OSC_BE0         pin 11;

"Define output

        OIDG            pin 12;
        IDWR            pin 13;
        OI_BE0          pin 15;
        OI_BE1          pin 16;
        OI_BE2          pin 17;
        OI_BE3          pin 18;
        DIR             pin 19;

"Define
        H,L,X   = 1,0,.X.;

Equations

        OIDG            =  OSCIORD      #  OID_CS       # !SC_A02       #  OSC_BE0;
        IDWR            =  OSCIOWR      #  OID_CS       # !SC_A02       #  OSC_BE0;
        OI_BE0          =  OSCIORD      #  OID_CS       #  SC_A02       #  OSC_BE0;
        OI_BE1          =  OSCIORD      #  OID_CS       #  SC_A02       #  OSC_BE1;
        OI_BE2          =  OSCIORD      #  OID_CS       #  SC_A02       #  OSC_BE2;
        OI_BE3          =  OSCIORD      #  OID_CS       #  SC_A02       #  OSC_BE3;
        DIR             =  OID_CS        & OSC_CS       &ODACK
                        #  OSCIORD;

end en_pal
```

PLDソース・リスト ID_PAL

```
module id_pal
title'NEC 21-4-1992'

ID_PAL  device'P20V8C';

"Define input
        PQ0             pin 1;
        PQ1             pin 2;
        PQ2             pin 3;
        A3              pin 4;
        SCS             pin 5;
        ORD             pin 6;
        ODACK           pin 7;
        OSIRD           pin 8;
        OSIWR           pin 9;
        IDMT            pin 10;
        OSC_CMD         pin 11;

"Define output
        OSCIOWR         pin 15;
        OSCIORD         pin 16;
        OSC_CS          pin 17;
        OID_CS          pin 19;
        OCS             pin 20;
        OID             pin 21;
        OSCEX32         pin 22;

"Define
        H,L,X   = 1,0,.X.;


Equations

        OSCIOWR         =  ( OSC_CMD    # !ODACK       # !ORD )       &  OSIWR;
        OSCIORD         =  ( OSC_CMD    # !ODACK       #  ORD )       &  OSIRD;
        OSC_CS          =    OSC_CMD    # !ODACK       #  SCS ;
        OID_CS          =    OSC_CMD    # !ODACK       #  IDMT;
        OCS             =    PQ0        #  PQ1;
        OID             =    PQ1        #  PQ2          #  A3;
        OSCEX32         =  ( PQ0        #  PQ1 )
                        &  ( PQ1        #  PQ2          #  A3 );

end id_pal
```

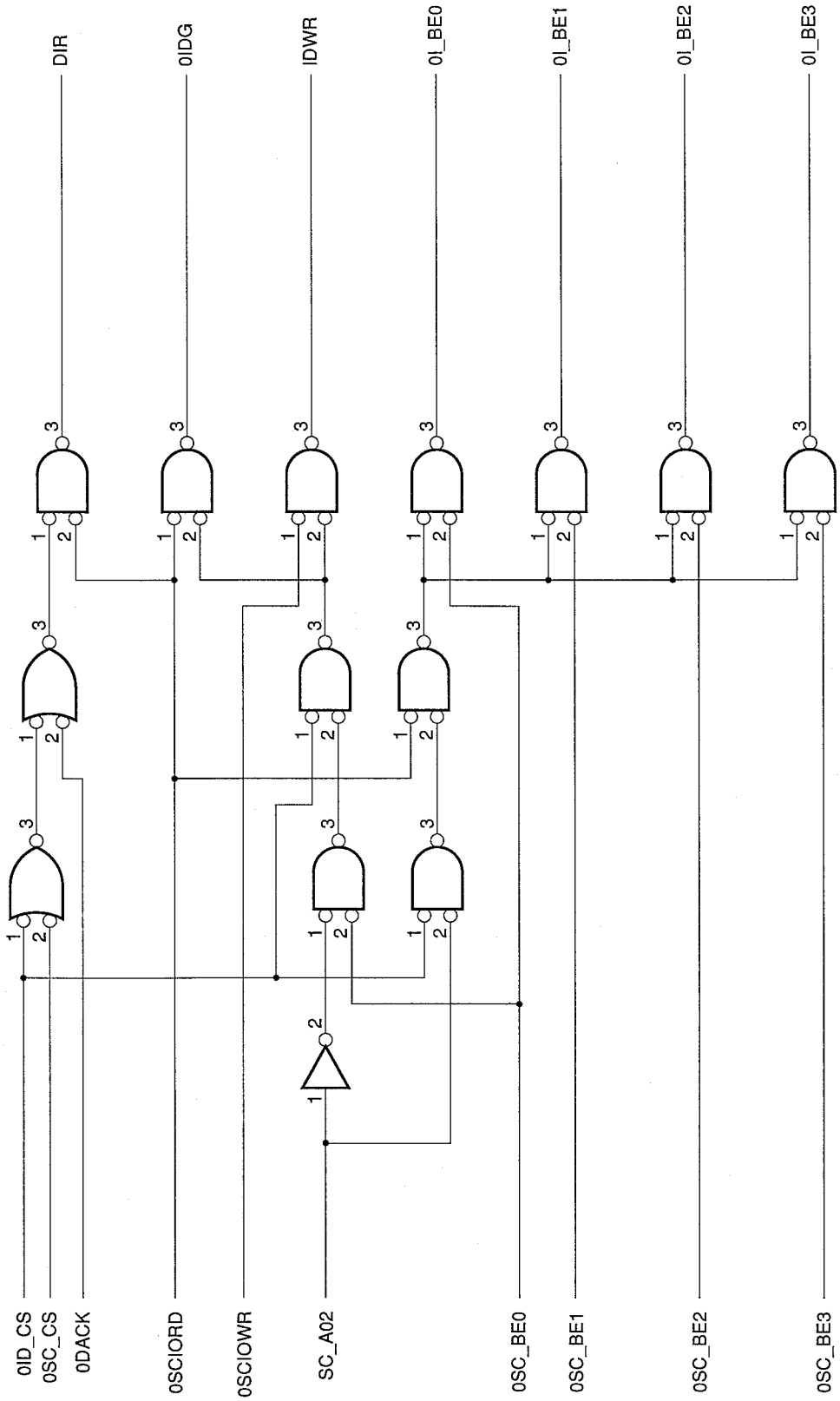
保守／廃止

図B-2　PLD内部の回路図（ID_PAL）



B

図B－3　PLD内部の回路図（EN_PAL）



保守／廃止

付録C　ASPI仕様

ASPIとはAdvanced SCSI Programming Interfaceの略であり，米国アダプテック社が推奨するSCSIの制御プログラムです。

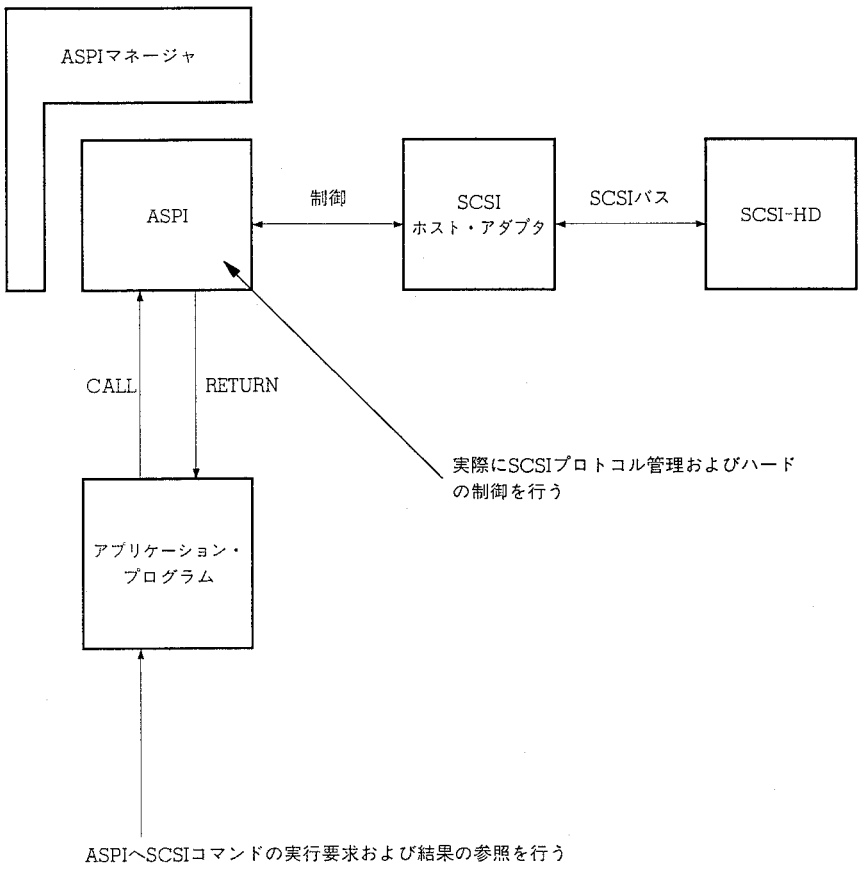
C.1　ASPIの概要

図C－1にASPIの概要を示します。ASPIはハードウエアおよびSCSIプロトコルに依存する部分を制御するプログラムです。ASPIマネージャはASPIをMS-DOS上に常駐するためのキャラクタ型のデバイス・ドライバです。

アプリケーション・プログラムがASPIをアクセスするには，以下の手続きを行います。

1. ASPIマネージャからASPIのエントリ・ポイントを受け取ります。
2. ASPIをコールします。

図C－1　ASPIの概要



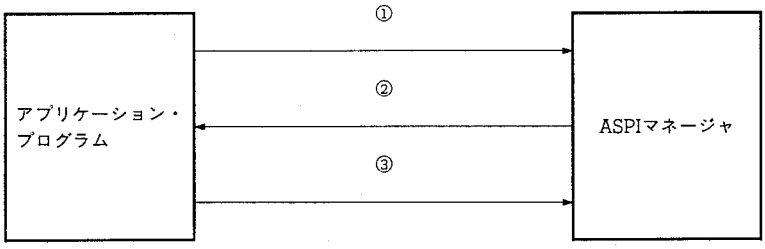
C

保守／廃止

C.2　ASPIエントリ・ポイントの獲得方法

アプリケーション・プログラムはASPIマネージャからMS-DOSを介してASPIエントリ・ポイントを獲得します。図C－2にASPIエントリ・ポイントの獲得方法を示します。

図C－2　ASPIエントリ・ポイントの獲得方法



① アプリケーション・プログラムはASPIマネージャをファイル・オープンします。

② アプリケーション・プログラムはIOCTLインプットによってASPIエントリ・ポイントをASPIマネージャから受け取ります。

③ アプリケーション・プログラムはASPIマネージャをファイル・クローズします。

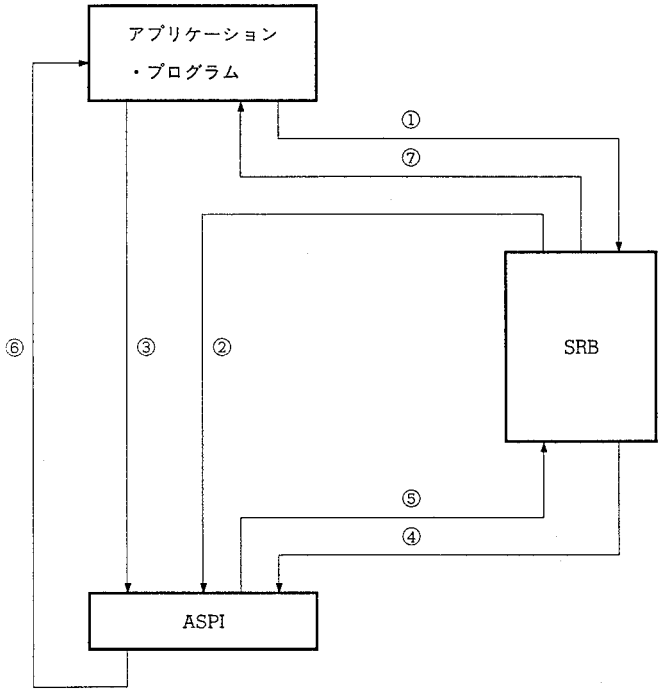
保守／廃止

C.3　ASPIのアクセス方法

図C－3にアプリケーション・プログラムからMS-DOS上に常駐されているASPIへのアクセス方法を示します。インタフェースにはSCSI REQUEST BLOCK（SRB）と呼ばれるものを使用します。

なお，ASPIへのアクセス方法のプログラム例を図C－4に示します。

図C－3　ASPIのアクセス方法



① アプリケーション・プログラムはSRBをセットします。

② アプリケーション・プログラムはSRBのポインタをASPIに渡します。

③ アプリケーション・プログラムはASPIをコールします。

④ ASPIはSRBを参照しターゲットにアクセスします。

⑤ ASPIは実行結果をSRBにセットします。

⑥ ASPIはSCSIコマンド実行ブロック内のモジュールをリターンします。

⑦ アプリケーション・プログラムはSRBを参照しASPIの実行結果を調べます。

C

図C－4　プログラム例

```
                .MODEL  SMALL
                .STACK  100h                ;100h byte stack

                .DATA
SCSIMgrString   db   "SCSIMGR$"
                dw   0                      ;NULL-terminate string
ASPI_Entry          db   4 dup (?)
SRB             db   58 dup (0)             ;Initialize SRB for Host
                                            ;  Adapter Inquiry
                .CODE
start:          mov  ax,@DATA
                mov  ds,ax                  ;Init DS
                mov  ax,03D00h
                lea  dx,SCSIMgrString
                int  21h                    ;Open ASPI Manager
                jc   NoASPIManager          ;Branch if none found
                push ax                     ;Save ASPI File Handle

                mov  bx,ax                  ;BX = File Handle
                mov  ax,4402h
                lea  dx,ASPI_Entry          ;Store entry point here
                mov  cx,4                   ;Four bytes to transfer
                int  21h                    ;Get ASPI entry point

                mov  ah,03Eh
                pop  bx                     ;BX = ASPI File Handle
                int  21h                    ;Close ASPI Manager

                push ds                     ;Push SRB's segment
                lea  bx,SRB
                push bx                     ;Push SRB's offset
                lea  bx,ASPI_Entry
                call DWORD PTR [bx]         ;Call ASPI
                add  sp,4                   ;Restore the stack
                    .
                    .
ASPI_Exit:      mov  ax,4C00h               ;Exit to DOS
                int  21h
                ret

NoASPIManager:      .                       ;No ASPI Manager
found!!
                jmp  ASPI_Exit              ;  Handle it.
                END
```

保守／廃止

C.4　SCSI REQUSET BLOCK（SRB）

SRBはアプリケーション・プログラムがASPIへアクセスするときに実行要求内容をセット，および実行結果を参照するためのテーブルです。表C－1にSRBを示します。なお，表C－1のSRBのあとに続く，各コマンド・コード用のSRBがあります。

表C－1　SRB

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION＝0	—

備考1.　W：SCSIデバイス・ドライバがライトします。

2.　R：ASPIマネージャがライトします。

3.　—：リザーブ

●COMMAND CODE

ASPIが実行するコマンドを表C－2に示します。

表C－2　ASPIが実行するコマンド

COMMAND CODE	DESCRIPTION
00H	HOST ADAPTER INQUIRY
01H	GET DEVICE TYPE
02H	EXECUTE SCSI I/O COMMAND
03H	ABORT SCSI I/O COMMAND
04H	RESET SCSI DEVICE
05H	SET HOST ADAPTER PARAMETERS
06H	GET DISK DRIVE INFORMATION
07H-7FH	RESERVED FOR FUTURE EXPANSION
80H-FFH	RESERVED FOR VENDOR UNIQUE

C

●STATUS

ASPIの実行結果を示すステータスを表C－3に示します。

表C－3　ASPIの実行結果を示すステータス

STATUS BYTE	DESCRIPTION
00H	SCSI REQUEST IN PROGRESS
01H	SCSI REQUEST COMPLETED WITHOUT ERROR
02H	SCSI REQUEST ABORTED BY HOST
04H	SCSI REQUEST COMPLETED WITH ERROR
80H	INVALID SCSI REQUEST
81H	INVALID HOST ADAPTER NUMBER
82H	SCSI DEVICE NOT INSTALLED

●HOST ADAPTER NUMBER

HOST ADAPTER NUMBERは１枚目のホスト・アダプタを“0”，２枚目のホスト・アダプタを“1”という順番で番号を降ります。

●SCSI REQUEST FLAGS

各コマンド・コードごとに内容が異なります。詳細は各コマンド・コードの説明を参照してください。

保守／廃止

C.5 HOST ADAPTER INQUIRY (COMMAND CODE=0)

このコマンドはホスト・アダプタの数を含むホスト・アダプタの情報を得るために使用されます。このコマンドはホスト・アダプタの数を得るためホスト・アダプタ・ナンバに00Hを指定して一度だけ発行されます。このコマンドのSRBを表C－4に示します。

表C－4　SRB (HOST ADAPTER INQUIRY)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=0	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	01H (01)	NUMBER OF HOST ADAPTERS	R
09H (09)	01H (01)	ID OF HOST ADAPTER	R
0AH (10)	10H (16)	SCSI MANAGER ID	R
1AH (26)	10H (16)	HOST ADAPTER ID	R
2AH (42)	10H (16)	HOST ADAPTER UNIQUE PARAMETERS	R

ステータスに (01H) がセットされていれば有効なデータがあることを示し，(81H) がセットされていれば指定した番号のホスト・アダプタがインストールされていないことを示します。

●SCSI REQUEST FLAGS

このコマンドでは使用しないので00Hを指定します。

●SCSI MANAGER ID

16バイトのASCIIコードによってSCSIマネージャを記述します。

●HOST ADAPTER ID

16バイトのASCIIコードによってホスト・アダプタを記述します。

●HOST ADAPTER UNIQUE PARAMETERS

ホスト・アダプタの特別な注意事項を記述します。

C

C.6 GET DEVICE TYPE (COMMAND CODE=1)

このコマンドはターゲットの種類を確認するために使用されます。たとえば，CD-ROMドライバはインストールされているホスト・アダプタに接続されるターゲット／LUNをスキャンすることで探しているデバイス・タイプと一致するCD-ROMドライブを探します。このコマンドのSRBを表C-5に示します。

表C-5 SRB (GET DEVICE TYPE)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=1	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	01H (01)	TARGET ID	W
09H (09)	01H (01)	LUN	W
0AH (10)	01H (01)	PERIPHERAL DEVICE TYPE OF TARGET/LUN	R

備考1. 00H：direct access device (e.g., hdd, mo-disk)

2. 01H：sequential access device (e.g., tape drive)

3. 02H：printer device

4. 03H：processor device

5. 04H：Write-once device (e.g., some optical disks)

6. 05H：CD-ROM device

7. 06H：scanner device

8. 07H：optical memory device (e.g., some optical disks)

ステータスに(01H)がセットされていればPERIPHERAL DEVICE TYPEフィールドに有効なデータがあることを示します。(82H)がセットされていればPERIPHERAL DEVICE TYPEフィールドに有効なデータがないことを示します。

●PERIPHERAL DEVICE TYPE

SCSI INQUIRYコマンドを送信した結果です。

●SCSI REQUEST FLAGS

このコマンドでは使用しないので00Hを指定します。

保守／廃止

C.7 EXECUTE SCSI I/O REQUEST(COMMAND CODE=2)

このコマンドはASPIにSCSIコマンドの実行を要求するときに使用されます。このコマンドのSRBを表C-6に示します。

表C-6 SRB (EXECUTE SCSI I/O REQUEST)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=2	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	01H (01)	TARGET ID	W
09H (09)	01H (01)	LUN	W
0AH (10)	04H (04)	DATA ALLOCATION LENGTH	W
0EH (14)	01H (01)	SENSE ALLOCATION LENGTH (N)	W
0FH (15)	02H (02)	DATA BUFFER POINTER (OFFSET)	W
11H (17)	02H (02)	DATA BUFFER POINTER (SEGMENT)	W
13H (19)	02H (02)	SRB LINK POINTER (OFFSET)	W
15H (21)	02H (02)	SRB LINK POINTER (SEGMENT)	W
17H (23)	01H (01)	SCSI CDB LENGHT (M)	W
18H (24)	01H (01)	HOST ADAPTER STATUS	R
19H (25)	01H (01)	TARGET STATUS	R
1AH (26)	02H (02)	POST ROUTINE ADDRESS (OFFSET)	W
1CH (28)	02H (02)	POST ROUTINE ADDRESS (SEGMENT)	W
1EH (30)	22H (34)	RESERVED FOR ASPI WORKSPACE	—
40H (64)	M	SCSI COMMAND DESCRIPTOR BLOCK	W
40H+M	N	SENSE ALLOCATION AREA	R

このコマンドはコマンド要求のキューイングが成功されたことを示すためステータス（00H）を返します。アプリケーション・プログラムがコマンド完了を確認するにはステータスが(00H)以外になるまでポーリングするかPOSTルーチンを使用します。ポーリングを使用するときは必ず割り込みをイネーブルにしなければなりません。

●SCSI REQUEST FLAGS

構成を以下に示します。

7	6	5	4	3	2	1	0
Rsvd	Rsvd	Rsvd	Direction		Rsvd	Link	Post

○リザーブ・ビット

0をセットします。

○Postビット

Postをイネーブルするときは1をセットし，ディスエーブルするときは0をセットします。

○Linkビット

Linkをイネーブルするときは1をセットし，ディスエーブルするときは0をセットします。

○Directionビット

データ・フェーズにおけるデータ転送方向を指定します。

00H　－　Direction determined by SCSI command. Length not checked.
(SCSIコマンドによって決定)

01H　－　Transfer from SCSI target to host. Length checked.
(ホスト→ターゲットの転送)

10H　－　Transfer from host to SCSI target. Length checked.
(ターゲット→ホストの転送)

11H　－　No data transfer.
(データなし)

●TARGET ID

目的のSCSIIDを指定します。

●LUN

目的のLUNを指定します。

●DATA ALLOCATION LENGTH

転送データ数をセットします。もしデータ転送のないSCSIコマンドならばこのフィールドには“0”をセットします。

●SENSE ALLOCATION LENGTH

SRBの最後に割り当てるSENSE ALLOCATION AREAのバイト数を指定します。ASPIはSCSIコマンド実行後チェック・コンディションを受信するとREQUEST SENSEコマンドを発行し，センス・データを受信します。ASPIはSENSE ALLOCATION AREAにセンス・データをセットします。

●DATA BUFFER POINTER

データ・バッファのアドレスを指定します。ASPIはこのポインタを物理アドレスに変換します。

●SRB LINK POINTER

続けて実行するSRBのポインタを指定します。

●SCSI CDB LENGTH

CDBのバイト数を指定します。

●HOST ADAPTER STATUS

内容を以下に示します。

00H　－　Host adapter did not detect any error.
（ホスト・アダプタがエラーを検出しなかった）

11H　－　Selection timeout.

12H　－　Data over-run/under -run

13H　－　Unexpected Bus Free
（バス・フリーしないところでバス・フリーになった）

14H　－　Target bus phase sequence failure
（ターゲットが間違ったバス・フェーズ・シーケンスを行った）

●TARGET STATUS

内容を以下に示します。

00H　－　No target status.

02H　－　Check status (Sense data is in Sense Allocation Area).

08H　－　Specified Target/LUN is busy.
（指定したターゲット／LUNがビジィだった）

18H　－　Reservation conflict.
（指定したターゲット／LUNがほかのホストによってリザーブされていた）

● **POST ROUTINE ADDRESS**

C.7.2　ASPI COMMAND POSTINGを参照してください。

● **SCSI COMMAND DESCRIPTOR BLOCK**

ここにはSCSIコマンドを指定します。

● **SENSE ALLOCATION AREA**

センス・データがセットされます。このフィールドの最大長はSENSE ALLOCATION LENGTHフィールドで指定されます。ただしターゲットはセンス・バイト要求より少ないバイト数をセットすることがあります。

C.7.1　SCSI COMMAND LINKING WITH ASPI

ASPIはいくつかのコマンドを続けて実行するSCSIリンク機能をサポートしています。ただしターゲットがSCSIリンク機能をサポートしていないと使用できません。SCSIリンク機能を使用するためには，SRBのLINK POINTERを使ってあとに続くSRBを連結します。連結された最後のSRBを除いてすべてのSRBのSCSI REQUEST FLAGSのLINKビットをセットします。ターゲットがリンクSCSIコマンドを完了したステータスを返すとASPIはただちに次のSRBを処理し，次のCDBを送信します。SCSIリンク機能を使用する場合SCSI REQUEST FLAGSのリンク・ビットと同様SCSI CDBのリンク・ビットをセットしなければなりません。この矛盾は予想できない動作の原因となります。

備考　SCSIリンク機能を使用していないホスト・アダプタ同様に，リンク機能をサポートしていないターゲットが多く存在するので互換性を考慮する場合はリンク機能を使用しないことを推奨します。

C.7.2　ASPI COMMAND POSTING

POSTINGを使用する場合はSCSI REQUEST FLAGSのpostビットをセットします。POSTINGとはASPIがSRBに示されるPOSTルーチンへFARコールすることです。POSTルーチンはSRBが完了したことを示すためASPIからコールされます。SRBの完了内容はスタック上の４バイトのSRBポインタによって示されます。このためPOSTルーチンはすべてのレジスタをセーブする必要があります。

ASPIは最初にSRBポインタのセグメント値をスタックにプッシュし，次にオフセット値をプッシュします。

POSTルーチンはできるだけ早く抜け出なくてはなりません。

POSTINGは割り込み処理が必要なデバイス・ドライバや常駐プログラムで使用されます。アプリケーション・プログラムはマルチタスクDOS環境の互換性を確保するためにPOSTINGよりむしろポーリングを使用すべきです。

図C－５にASPI POSTルーチンのサンプルを示します。

図C－5　POSTルーチン

```
ASPI_Post       proc far
                push bp
                mov  bp,sp

                pusha                       ;Save all registers
                push ds
                push es

                mov  bx,[bp+6]        ;BX = SRB's offset
                mov  es,[bp+8]        ;ES = SRB's segment
                    .                       ;ES:BX points to SRB
                    .
                    .
                pop  es
                pop  ds
                popa
                pop  bp               ;Restore all registers
                retf                  ;  and return to ASPI
ASPI_Post       endp
```

ＰＯＳＴルーチンが最初にアクセスされるとスタックは以下のようになっている。

```
    Top Of Stack  [SP+0] ->  Return Address (Offset)

                  [SP+2] -> Return Address (Segment)
                  [SP+4] -> SRB Pointer (Offset)
                  [SP+6] -> SRB Pointer (Segment)
                          ...
                          ...
                          ...
```

C

C.8 ABORT SCSI I/O REQUEST (COMMAND CODE=3)

このコマンドはSRBの実行をアポートするために使用されます。このコマンドのSRBを表C-7に示します。

表C-7 SRB (ABORT SCSI I/O REQUEST)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=3	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	02H (02)	SRB POINTER TO ABORT (OFFSET)	W
0AH (10)	02H (02)	SRB POINTER TO ABORT (SEGMENT)	W

このコマンドはステータスにSCSI REQUEST COMPLETED WITHOUT ERRORを必ず返します。

●SCSI REQUEST FLAGS

使用しないので (00H) を指定します。

●SRB POINTER TO ABORT

アボート実行後のジャンプ先を示すポインタです。

C.9 RESET SCSI DEVICE (COMMAND CODE=4)

このコマンドは指定したSCSIターゲットをリセットするために使用されます。このコマンドのSRBを表C-8に示します。

表C－8　SRB (RESET SCSI REQUEST)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=4	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	01H (01)	TARGET ID	W
09H (09)	01H (01)	LUN	W
0AH (10)	0EH (14)	RESERVED	—
18H (24)	01H (01)	HOST ADAPTER STATUS	R
19H (25)	01H (01)	TARGET STATUS	R
1AH (26)	02H (02)	POST ROUTINE ADDRESS (OFFSET)	W
1CH (28)	02H (02)	POST ROUTINE ADDRESS (SEGMENT)	W
1EH (30)	22H (34)	RESERVED FOR ASPI WORKSPACE	—

このコマンドのSRB構造は使用されないフィールドを除いてEXECUTE SCSI I/O SRBとほとんど同じです。このコマンドはコマンド要求のキューイングが成功されたことを示すためステータス(00H)を返します。アプリケーション・プログラムがコマンド完了を確認するには，ステータスが (00H) 以外になるまでポーリングするか，POSTルーチンを使用します。ポーリングを使用するときは必ず割り込みをイネーブルにしなければなりません。

C.10　SET HOST ADAPTER PARAMETERS (COMMAND CODE=5)

このコマンドはホスト・アダプタ固有のパラメータを要求するときに使用されます。このコマンドのSRBを表C－9に示します。

表C－9　SRB (SET HOST ADAPTER PARAMETERS)

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE=5	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION=0	—
08H (08)	10H (16)	HOST ADAPTER UNIQUE PARAMETERS	W

ステータスに（01H）がセットされていれば有効なデータがあることを示し，（81H）がセットされていれば指定した番号のホスト・アダプタがインストールされていないことを示します。

●SCSI REQUEST FLAGS

このコマンドでは使用しないので00Hを指定します。

●HOST ADAPTER UNIQUE PARAMETERS

ホスト・アダプタ固有のパラメータを記述します。

C.11　GET DISK DRIVE INFORMATION（COMMAND CODE＝6）

このコマンドはディスク・ドライブがBIOS/DOSによってすでに使用されている状態なのかそれともディスク・ドライブがディスク・ドライバによって使用できる状態なのか調べるためにデバイス・ドライバによって使用されます。このコマンドは，ドライブがBIOS/DOS下で制御できないものであるにもかかわらずINT13Hを経由してアクセスしようとしているか調べるためにも使用されます。このコマンドのSRBを表C－10に示します。

表C－10　SRB（GET DISK DRIVE INFORMATION）

OFFSET	#BYTES	DESCRIPTION	R/W
00H (00)	01H (01)	COMMAND CODE＝6	W
01H (01)	01H (01)	STATUS	R
02H (02)	01H (01)	HOST ADAPTER NUMBER	W
03H (03)	01H (01)	SCSI REQUEST FLAGS	W
04H (04)	04H (04)	RESERVED FOR EXPANSION＝0	—
08H (08)	01H (01)	TARGET ID	W
09H (09)	01H (01)	LUN	W
0AH (10)	01H (01)	DRIVE FLAGS	R
0BH (11)	01H (01)	INT 13H DRIVE	R
0CH (12)	01H (01)	PREFERRED HEAD TRANSLATION	R
0DH (13)	01H (01)	PREFERRED SECTOR TRANSLATION	R
0EH (14)	0AH (10)	RESERVED FOR EXPANSION＝0	—

このコマンドをサポートするASPIはCOMMAND COMPLETE WITHOUT ERROR（01H）ステータスを返します。このコマンドをサポートしないASPIはINVALID SCSI REQUEST（80H）を返します。

●SCSI REQUEST FLAGS

使用しないので00Hをセットします。

●DRIVE FLAGS

構成を以下に示します。

7	6	5	4	3	2	1	0
Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	Rsvd	INT 13H	Info

○リザーブ・ビット

0をセットします。

○INT13H, Info

内容を以下に示します。

00H　－　ホスト・アダプタ番号，ターゲット番号，LUNで指定されたドライブはINT13Hを経由してアクセスできません。このドライブをアクセスするときは，ASPIマネージャを経由する必要があります。INT13Hフィールドは無効です。

01H　－　ホスト・アダプタ番号，ターゲット番号，LUNで指定されたドライブはINT13Hを経由してアクセスできます。INT13Hフィールドはドライブの INT13Hドライブ番号がセットされます。このドライブはDOSの制御下にあります。

10H　－　ホスト・アダプタ番号，ターゲット番号，LUNで指定されたドライブはINT13Hを経由してアクセスできます。INT13Hフィールドはドライブの INT13Hドライブ番号がセットされます。このドライブはDOSの制御下にありません。このドライブはSCSIデバイス・ドライバなどで使用できます。

11H　－　Invalid.

●INT13H DRIVEフィールド

INT13H DRIVE番号が返されます。INT13H DRIVE番号の幅は00H-FFHです。

●PREFERRED HEAD TRANSLATION

ディスク・ドライブのヘッド数を示します。

●PREFERRED SECTOR TRANSLATION

ディスク・ドライブのトラック当たりのセクタ数を示します。

保守／廃止

保守／廃止

付録D　SCSIハード・ディスク仕様

このアプリケーション・ノートで使用したSCSIハード・ディスクの仕様を示します。

D.1　SCSIメッセージ

このアプリケーション・ノートにおいてSCSIハード・ディスクを制御するために使用したSCSIメッセージを表D－1，拡張SCSIメッセージを表D－2に示します。

また，SYNCHRONOUS DATA TRANSFER REQUESTメッセージの構成を表D－3に示します。

表D－1　SCSIメッセージ

メッセージ・コード	方　向	メッセージ名
00H	IN	COMMAND COMPLETE注
01H	—	拡張メッセージ
02H	IN	SAVE DATA POINTER
03H	IN	RESTORE POINTER
04H	IN	DISCONNECT
05H	OUT	INITIATOR DETECTED ERROR
06H	OUT	ABORT注
07H	IN/OUT	MESSAGE REJECT注
08H	OUT	NO OPERATION注
09H	OUT	MESSAGE PARITY ERROR
0CH	OUT	BUS DEVICE RESET注
80H-FFH	IN/OUT	IDENTIFY

注　CCSサポート必須メッセージ

表D－2　拡張SCSIメッセージ

拡張メッセージコード	方　向	メッセージ名
01H	IN/OUT	SYNCHRONOUS DATA TRANSFER REQUEST

表D－3　SYNCHRONOUS DATA TRANSFER REQUESTメッセージ

1バイト目	01H	
2バイト目	03H	
3バイト目	01H	
4バイト目	転送時間（m）	← mH×4 ns＝最大100 ns
5バイト目	オフセット値	← 最大8

D.2　SCSIステータス

このアプリケーション・ノートにおいてSCSIハード・ディスクを制御するために使用したSCSIステータスを表D－4に示します。

表D－4　SCSIステータス

ビット	7	6	5	4	3	2	1	0	
コード	R	R						R	
R=0			0	0	0	0	0		GOOD
			0	0	0	0	1		CHECK CONDITION
			0	0	1	0	0		BUSY
			0	1	1	0	0		RESERVATION CONFLICT

保守／廃止

D.3　SCSIコマンド

このアプリケーション・ノートにおいてSCSIハード・ディスクを制御するために使用したSCSIコマンドを表D－5に示します。以降にコマンドの機能を説明します。

なお，このアプリケーション・ノートではリンク・ビットおよびフラグ・ビットはサポートしません。

表D－5　SCSIコマンド

コード	コマンド名	コード	コマンド名
00H	Test Unit Ready	1BH	Start/Stop Unit
01H	Rezero Unit	1DH	Send Diagnostic
03H	Request Sense	25H	Read Capacity
04H	Format Unit	28H	Read Extended
12H	Inquiry	2AH	Write Extended
15H	Mode Select	2BH	Seek Extended
16H	Reserve Unit	2EH	Write & Verify
17H	Release Unit	2FH	Verify
1AH	Mode Sense		

D.3.1　Test Unit Ready

このコマンドはターゲットが動作可能かどうかチェックします。

表D－6　Test Unit Ready

バイト＼ビット	7	6	5	4	3	2	1	0
0	00H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

備考1．F：フラグ・ビット

2．L：リンク・ビット

D

D.3.2　Rezero Unit

このコマンドはシリンダ０へヘッドを移動させ，内部バッファを初期化します。

表D－7　Rezero Unit

バイト＼ビット	7	6	5	4	3	2	1	0
0	01H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

D.3.3　Request Sense

このコマンドはチェック・コンディション・ステータスをイニシエータが受信したあとに詳細な情報（センス・データ）をターゲットから受信するものです。

なお，Extended Sense Dataの13バイトをサポートしました。

表D－8　Request Sense

バイト＼ビット	7	6	5	4	3	2	1	0
0	03H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	0DH							
5	00H						F	L

表D－9　Extended Sense Data

バイト＼ビット	7	6	5	4	3	2	1	0
0	V	1	1	1	0	0	0	0
1	セグメント番号（00H）							
2	0	0	I	0	センス・キー			
3	論理ブロック・アドレス　（MSB）							
4	論理ブロック・アドレス							
5	論理ブロック・アドレス							
6	論理ブロック・アドレス　（LSB）							
7	05H							
8	00H							
9	00H							
10	00H							
11	00H							
12	センス・コード							

備考1.　V＝0：論理ブロック・アドレスに有効な値がセットされていないことを示します。

V＝1：論理ブロック・アドレスに有効な値がセットされていることを示します。

2.　I＝1：要求されたブロック長がターゲットの論理ブロック長と一致しなかったことを示します。

3.　論理ブロック・アドレス：センス・キーに関係する論理ブロック・アドレスを示します。

4.　アディショナル・センス・データ：バイト８以降に続くセンス・データ数を示します。

表D－10　センス・キー

センス・キー	センス・キー・ネーム	内　　容
0H	ノー・センス	該当するセンス・キーがないことを示します。
1H	リカバード・エラー	コマンドの実行がコントローラの回復処理によって正常に終了したことを示します。
2H	ノット・レディ	ロジカル・ユニットへのアクセスができないことを示します。
3H	メディア・エラー	メディアの欠陥または記憶データのエラーを検出したことを示します。
4H	ハードウエア・エラー	ロジカル・ユニットのハードウエアに障害が発生し，コマンドの実行が正常に終了できなかったことを示します。
5H	イリーガル・エラー	CDBまたは追加パラメータに無効なパラメータが検出され，コマンドが実行されなかったことを示します。
6H	ユニット・アテンション	ターゲットがリセット信号またはBUS DEVICE RESETメッセージによってリセットされたことを示します。
7H	データ・プロテクト	プロテクトされている領域をアクセス（リード，ライト）したがコマンドは実行されなかったことを示します。
8H	ブランク・チェック	SCSIハード・ディスク仕様はダイレクト・アクセス・デバイスを対象にしているので，このセンス・キーはサポートしません。
AH	コピー・アボート	SCSIハード・ディスク仕様ではCOPY，COMPARE，COPY AND VERIFYコマンドをサポートしないので，このセンス・キーはサポートしません。
BH	アボート・コマンド	ターゲットがイニシエータからのコマンドを異常終了させたことを示します。イニシエータはコマンドをリトライします。
CH	イコール	SCSIハード・ディスク仕様ではSEARCH DATAコマンドをサポートしないので，このセンス・キーはサポートしません。
EH	ミスコンペア	SCSIハード・ディスク仕様ではCOPY，COMPARE，COPY AND VERIFYコマンドをサポートしないので，このセンス・キーはサポートしません。

表D－11　センス・コード（1/4）

センス・コード	センス・キー	内　　容
00H	0H	NO ADDITIONAL SENSE CODE 該当するセンス・コードがないことを示します。
01H	4H	NO INDEX/SECTOR リード／ライト動作に必要なインデクス，セクター信号が検出できなかったことを示します。
02H	4H	NO SEEK COMPLETE SEEK動作を伴うコマンドにおいてSEEKが正常に動作しなかったことを示します。
03H	4H	WRITE FAULT ライト動作がハードウエア上の問題から失敗したことを示します。
04H	2H	DRIVE NOT READY ロジカル・ユニットへのアクセスができないことを示します。
05H	2H	DRIVE NOT SELECT 指定されたロジカル・ユニットがなかったことを示します。
06H	4H	NO TRACK ZERO FOUND リゼロ動作においてトラック・ゼロが見つからなかったことを示します。
07H	4H	MULTIPLE DRIVE SELECTED 指定されたロジカル・ユニットが複数存在していることを示します（同じロジカル・ユニット・ナンバーがあります）。
08H	4H	LOGICAL UNIT COMMUNICATION FAILURE ハードウエアのエラーによりターゲット内のロジカル・ユニットへアクセスできなかったことを示します。
09H	4H	TRACK FOLLOWING ERROR ライト中に規定値以上のオフ・トラックが発生したことを示します。
0AH-0FH		リザーブ
10H	3H 4H	ID CRC OR ECC ERROR 目的のブロックのID部にCRCまたはECCエラーが発生し，リトライによる回復ができなかったことを示します。
11H	3H	UNRECOVERED READ ERROR OF DATA BLOCKS 目的のブロックのデータ部にエラーが発生し，リトライによる回復ができなかったことを示します。
12H	3H	NO ADDRESS MARK FOUND IN ID FIELD ID部のアドレス・マークが検出できなかったためID部が検出できなかったことを示します。
13H	3H	NO ADDRESS MARK FOUND IN DATA FIELD データ部のアドレス・マークが検出できなかったためデータ部が検出できなかったことを示します。

表D－11　センス・コード（2/4）

センス・コード	センス・キー	内　　容
14H	3H	NO RECORD FOUND 目的のブロックを検出できなかったことを示します。
15H	3H 4H	SEEK POSITIONING ERROR シーク動作中に異常が発生したことを示します。
16H	1H 3H	DATA SYNCHRONIZATION MARK ERROR Synchronization Markによってデータをリードするための同期クロックを生成できなかったことを示します。
17H	1H	RECOVERED READ DATA WITH TARGET'S READ RETRIES ECC機能によるものではなくリード・リトライ動作によってデータが読めたことを示します。
18H	1H	RECOVERED READ DATA WITH TARGET'S ECC CORRECTION ECC機能によってリード・データのエラーを回復できたことを示します。
19H	1H 3H	DEFECT LIST ERROR Defect Listのリード時にエラーが発生し，正常動作ができないことを示します。
1AH	5H	PARAMETER OVERRUN パラメータ転送時にオーバランが発生したことを示します。
1BH	4H	SYNCHRONOUS TRANSFER ERROR 同期転送中にエラーが発生したことを示します。
1CH	3H 5H	PRIMARY DEFECT LIST NOT FOUND P-LISTが検出できないため正常動作ができないことを示します。
1EH	1H 3H	RECOVERED ID WITH TARGT'S ECC CORRECTION ECC機能によってIDのリード・エラーを回復できたことを示します。
1FH		リザーブ
20H	5H	INVALID COMMAND OPERATION CODE CDBのコマンド・コードが有効でないことを示します。
21H	5H	ILLEGAL BLOCK ADDRESS ロジカル・ユニットの総ブロック数より大きい値のブロックが指定されていることを示します。
22H	5H	ILLEGAL FUNCTION FOR DEVICE TYPE ロジカル・ユニットのデバイス・タイプが違うことを示しています。
23H		リザーブ
24H	5H	ILLEGAL FIELD IN CDB CDBの２バイト以降に不正なバイトがあったことを示します。
25H	5H	INVALID LUN ロジカル・ユニット・ナンバーが不適当であったことを示します。
26H	5H	INVALID FIELD IN PARAMETER LIST パラメータ・リストに不適当な内容があったことを示します。

表D－11　センス・コード（3/4）

センス・コード	センス・キー	内　　容
27H	7H	WRITE PROTECTED ライト・プロテクトされている領域をアクセス（ライト）するコマンドを受け取ったことを示します。
28H	6H	MEDIUM CHANGED ロジカル・ユニットのメディアが交換されたことを示します。
29H	6H	POWER ON/RESET/BUS DEVICE RESET OCCURRED パワー・オン，リセット，バス・デバイス・リセット・メッセージによってロジカル・ユニットが初期化されたため，モード・セレクト・コマンドによる初期化が必要です。
2AH	6H	MODE SELECT PARAMETERS CHANGED ほかのイニシエータによりモード・セレクタ・パラメータが変更されたことを示します。
2BH-2FH		リザーブ
30H	3H	INCOMPATIBLE CARTRIDGE カートリッジが矛盾することを示します。
31H	3H	MEDIUM FORMAT CORRUPTED フォーマットが正常終了しなかったことを示します。
32H	3H	NO DEFECT SPARE LOCATION AVAILABLE スペア・アロケーションがなくなったことを示します。
33H-3FH		リザーブ
40H	4H	RAM FAILURE 診断した結果，データ・バッファに異常があったことを示します。
41H	4H	DATA PATH DIAGNOSTIC FAILURE 診断した結果，ECC回路に異常があったことを示します。
42H	4H	POWER ON DIAGNOSTIC FAILURE パワー・オン時の診断においてターゲットに異常があったことを示します。
43H	4H BH	MESSAGE REJECT ERROR イニシエータからメッセージ・リジェクト・メッセージを受信したことを示します。
44H	4H BH	INTERNAL CONTROLLER ERROR コントローラ内部でエラーが発生したことを示します。
45H	4H BH	SELECT/RESELECT FAILED リセレクトが失敗したことを示します。
46H	4H BH	UNSUCCESSFUL SOFT RESET バス・デバイス・リセット・メッセージの実行が成功しなかったことを示します。
47H	4H BH	SCSI INTERFACE PARITY ERROR イニシエータからの情報でパリティ・エラーが発生したことを示します。

表D－11　センス・コード（4/4）

センス・コード	センス・キー	内　　容
48H	4H BH	INITIATOR DETECTED ERROR イニシエータからイニシエータ・ディテクティド・エラー・メッセージを受信したことを示します。
49H	4H BH	INAPPROPRIATE/ILLEGAL MESSAGE ターゲットがサポートしていないメッセージを受信したことを示します。

D.3.4　Format Unit

このコマンドはメディアをフォーマットします。

表D－12　Format Unit

ビット／バイト	7	6	5	4	3	2	1	0
0	04H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

D.3.5　Inquiry

このコマンドはターゲットのインクワイアリ・データをリードします。

表D－13　Inquiry

ビット／バイト	7	6	5	4	3	2	1	0
0	12H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	05H							
5	00H						F	L

表D－14　インクワイアリ・データ

ビット／バイト	7	6	5	4	3	2	1	0
0	00H							
1	00H							
2	バージョン							
3	01H							
4	アロケーション・レングス							

D.3.6　Mode Select

このコマンドはメディアのモード（ブロック数，1ブロック当たりのバイト数など）を指定します。

表D－15　Mode Select

バイト＼ビット	7	6	5	4	3	2	1	0
0	15H							
1	LUN			0	0	0	0	SP
2	00H							
3	00H							
4	2CH							
5	00H						F	L

備考　SP＝1：パラメータをセーブします。

表D－16　Header

バイト＼ビット	7	6	5	4	3	2	1	0
0	00H							
1	00H							
2	バージョン（00H）							
3	08H							

表D－17　Block Descriptors

バイト＼ビット	7	6	5	4	3	2	1	0
0	00H							
1	00H							
2	00H							
3	00H							
4	00H							
5	00H							
6	02H							
7	00H							

備考　1ブロック当たり512バイト

表D－18　Error Recovery Parameters

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	PAGE CODE＝1H					
1	06H							
2	0	0	1	0	0	1	0	0
3	10H							
4	08H							
5	00H							
6	00H							
7	00H							

備考　リトライを行ったあとECC機能により修正を行い，修正可能ならばデータ転送後，CHECK CONDITION STATUS，RECOVERED SENSE KEYをセットします。リトライ数は16回です。

表D－19　Direct Access Device Parameters

ビット / バイト	7	6	5	4	3	2	1	0
0	0	0	PAGE CODE=3H					
1	16H							
2	00H							
3	00H							
4	00H							
5	00H							
6	00H							
7	00H							
8	00H							
9	00H							
10	00H							
11	00H							
12	02H							
13	00H							
14	00H							
15	00H							
16	00H							
17	00H							
18	00H							
19	00H							
20	00H							
21	00H							
22	00H							
23	00H							

D.3.7　Reserve Unit

このコマンドはターゲットを予約し，ほかのイニシエータからのアクセスを禁止します。

表D－20　Reserve Unit

バイト＼ビット	7	6	5	4	3	2	1	0
0	16H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

D.3.8　Release Unit

このコマンドは予約されたターゲットを解除し，ほかのイニシエータからのアクセスを許可します。

表D－21　Release Unit

バイト＼ビット	7	6	5	4	3	2	1	0
0	17H							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

D.3.9　Mode Sense

このコマンドはメディアのモード（ブロック数，1ブロック当たりのバイト数など）を調べます。

表D－22　Mode Sense

ビット / バイト	7	6	5	4	3	2	1	0
0	1AH							
1	LUN			0	0	0	0	0
2	PCF	01H						
3	00H							
4	14H							
5	00H						F	L

備考　PCF＝1，1 REPORT SAVED VALUE

表D－23　Header

ビット / バイト	7	6	5	4	3	2	1	0
0	3FH							
1	00H							
2	00H							
3	08H							

保守／廃止

表D－24　Block Descriptors

バイト＼ビット	7	6	5	4	3	2	1	0
0	00H							
1	**H							
2	**H							
3	**H							
4	00H							
5	00H							
6	02H							
7	00H							

備考1.　＊：任意

2.　1ブロック当たり512バイト

表D－25　Error Recovery Parameters

バイト＼ビット	7	6	5	4	3	2	1	0
0	1	0	PAGE CODE=1H					
1	06H							
2	0	0	1	0	0	1	0	0
3	10H							
4	08H							
5	00H							
6	00H							
7	FFH							

備考　リトライを行ったあとECC機能により修正を行い，修正可能ならばデータ転送後，CHECK CONDITION STATUS，RECOVERED SENSE KEYをセットします。リトライ数は16回です。

保守／廃止

D.3.10　Start/Stop Unit

このコマンドはターゲットに対して装置の起動／停止を指示します。

表D－26　Start/Stop Unit

バイト＼ビット	7	6	5	4	3	2	1	0
0	1BH							
1	LUN			0	0	0	0	0
2	00H							
3	00H							
4	0	0	0	0	0	0	0	S
5	00H						F	L

備考1．S＝0：ストップ・ユニット（停止）
S＝1：スタート・ユニット（起動）

2．コマンド実行後，ステータスが送信します。

D.3.11　Send Diagnostic

このコマンドはターゲットに診断テストを指示します。

表D－27　Send Diagnostic

バイト＼ビット	7	6	5	4	3	2	1	0
0	1DH							
1	LUN			0	0	1	0	0
2	00H							
3	00H							
4	00H							
5	00H						F	L

D.3.12　Read Capacity

このコマンドはロジカル・ユニットの記憶容量を調べます。

表D－28　Read Capacity

バイト＼ビット	7	6	5	4	3	2	1	0
0	25H							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	00H							
8	0	0	0	0	0	0	0	PM
9	00H						F	L

備考　PM＝0：リード・キャパシティ・データの論理ブロック・アドレスにはロジカル・ユニットの総ブロック数がセットされ，ブロック長には１ブロック当たりのバイト数がセットされます。
なお，リード・キャパシティ・コマンドの論理ブロック・アドレスには00000000Hをセットしてください。

PM＝1：リード・キャパシティ・データの論理ブロック・アドレスにはコマンドで指定された論理ブロック・アドレスがあるシリンダの最終論理ブロック・アドレスがセットされます。ブロック長には１ブロック当たりのバイト数がセットされます。

表D－29　Read Capacity Data

バイト＼ビット	7	6	5	4	3	2	1	0
0	(MSB)							
1	論理ブロック・アドレス							
2								
3								(LSB)
4	(MSB)							
5	ブロック長							
6	（１ブロック当たりのバイト数）							
7								(LSB)

D.3.13　Read Extended

このコマンドはデータをリードします。

表D－30　Read Extended

バイト＼ビット	7	6	5	4	3	2	1	0
0	28H							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	(MSB)							
8	転送長							(LSB)
9	00H						F	L

D.3.14　Write Extended

このコマンドはデータをライトします。

表D－31　Write Extended

バイト＼ビット	7	6	5	4	3	2	1	0
0	2AH							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	(MSB)							
8	転送長							(LSB)
9	00H						F	L

D.3.15　Seek Extended

このコマンドは指定したブロックへヘッドをシークします。

表D－32　Seek Extended

バイト＼ビット	7	6	5	4	3	2	1	0
0	2BH							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	00H							
8	00H							
9	00H						F	L

D.3.16　Write and Verify

このコマンドはデータをライトしたあと，ベリファイ（書き込みデータのECCチェック）します。

表D－33　Write and Verify

バイト＼ビット	7	6	5	4	3	2	1	0
0	2EH							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	(MSB)							
8	転送長							(LSB)
9	00H						F	L

D.3.17　Verify

このコマンドはデータをベリファイ（ECCチェック）します。

表D－34　Verify

バイト＼ビット	7	6	5	4	3	2	1	0
0	2FH							
1	LUN			0	0	0	0	0
2	(MSB)							
3	論理ブロック・アドレス							
4								
5								(LSB)
6	00H							
7	(MSB)							
8	ベリファイ・データ長							(LSB)
9	00H						F	L

保守／廃止

〔メ　モ〕

保守／廃止

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン（インフォメーションセンター）
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044-548-8899
FAX　：044-548-7900
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部		〒108-8001	東京都港区芝５－７－１　（日本電気本社ビル）	(03)3454-1111
中部支社	半導体第一販売部 半導体第二販売部	〒460-8525	愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222-2170 (052)222-2190
関西支社	半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540-8551	大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06)6945-3178 (06)6945-3200 (06)6945-3208

北海道支社	札　幌	(011)251-5599	宇都宮支店	宇都宮	(028)621-2281	北陸支社	金　沢	(076)232-7303
東北支社	仙　台	(022)267-8740	小山支店	小　山	(0285)24-5011	京都支社	京　都	(075)344-7824
岩手支店	盛　岡	(019)651-4344	甲府支店	甲　府	(055)224-4141	神戸支社	神　戸	(078)333-3854
郡山支店	郡　山	(024)923-5511	長野支社	松　本	(0263)35-1662	中国支社	広　島	(082)242-5504
いわき支店	いわき	(0246)21-5511	静岡支社	静　岡	(054)254-4794	鳥取支店	鳥　取	(0857)27-5311
長岡支店	長　岡	(0258)36-2155	立川支社	立　川	(042)526-5981,6167	岡山支店	岡　山	(086)225-4455
水戸支店	水　戸	(029)226-1717	埼玉支社	大　宮	(048)649-1415	松山支店	松　山	(089)945-4149
土浦支店	土　浦	(0298)23-6161	千葉支社	千　葉	(043)238-8116	九州支社	福　岡	(092)261-2806
群馬支店	高　崎	(027)326-1255	神奈川支社	横　浜	(045)682-4524			
太田支店	太　田	(0276)46-4011	三重支店	津	(059)225-7341			

保守／廃止

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］ μPD72611 アプリケーション・ノート PC/AT互換（EISAバス）デバイス・ドライバ編
（S14325JJ1V0AN00（第1版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）（　　）
ご住所（　　）
お電話番号（　　）
お仕事の内容（　　）
お名前（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC販売員，特約店販売員，NEC半導体ソリューション技術本部員，
その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ