組込みシステムにおける PDIC 機能ガイドライン
―DIC アーキテクチャの導入―

２００４年４月版

目　次

まえがき

本ドキュメントは、ITRON 仕様検討グループのデバイスドライバ検討会における検討結果を報告するものである。デバイスドライバ検討会は、デバイスを制御する基本プログラムの流通を目的に、2002 年 10 月より検討作業を行ってきた。
デバイスドライバ検討会では、1999 年 11 月に公開された『デバイスドライバ設計ガイドライン WG　中間報告ーDIC アーキテクチャの提案ー』で提案されている PDIC 機能を実際のデバイスに当てはめ、具体的な機能を検討した。
本ガイドラインが策定する機能や API を持つ基本プログラムがオープンなソフトウェアとして流通することにより、組込みシステムの開発効率の向上を狙っている。
本ガイドラインで策定する機能は、システムが OS の利用有無によらず広い範囲で利用できるように考慮した。
本ガイドラインが提案する要件に従いさまざまなデバイスを制御する基本プログラムが開発されていくことを願う。
組込みシステムではさまざまなデバイスが利用されている。本ガイドラインで選定したデバイスはほんの一部にすぎないが、デバイスを制御する基本プログラムの流通を目指し、本ガイドラインを公開する。

デバイスドライバ検討会の参加メンバ（あいうえお順）

大西 誠	沖電気工業（株）
片山 吉章	三菱電機（株）
木下 稔章	（株）デンソークリエイト
金田一 勉	（株）エルミックシステム
工藤 健治	富士通デバイス（株）
小玉 将義	松下電器産業（株）
小林 康浩	富士通（株）
酒井 淳	NEC 東芝スペースシステム（株）
坂部 健一	NEC 東芝スペースシステム（株）
佐野 一弘	（株）エイコット
澤田 勉	イーソル（株）
宿口 雅弘	三菱電機マイコン機器ソフトウェア（株）
鈴木 克義	茨城日立情報サービス（株）
高田 広章	名古屋大学
高野 秀隆	東芝情報システム（株）
高橋 忠輝	NEC 東芝スペースシステム（株）
竹内 良輔	（株）リコー
角田 知明	（株）日立超 LSI システムズ
中本 幸一	日本電気（株）
橋本 真一	（株）ACCESS
早川 大	（株）リコー
細谷 雅寿	沖電気工業（株）
松村 進平	（社）トロン協会
宮下 光明	（株）グレープシステム
三好 健児	（株）ルネサステクノロジ
村木 宏行	（株）ルネサスソリューションズ
山田 真二郎	（株）ルネサステクノロジ
脇坂 新路	（株）ルネサステクノロジ

１．概要

９９年に公開されたデバイスドライバ設計ガイドライン WG による中間報告である「DIC アーキテクチャの提案」を受け継ぎ、代表的なデバイスに関して具体的なガイドラインを示す。

DIC（Device Interface Component）は、以下のような構造としている。

- GDIC（General DIC）
- PDIC（Primitive DIC）
- SIL（System Interface Layer）

GDIC は、アプリケーションやソフトウェア部品からの要求の受付け及び同期処理を行い、OS に依存する処理を行う。
PDIC は、デバイス固有の処理を行い、OS に依存しない処理を行う。PDIC は、割込みサービスルーチン（ISR）と PDSR（Primitive Device Service Routine）に分かれる。
SIL は、システムの動作環境（CPU のクロックなど）に依存する処理を行う。

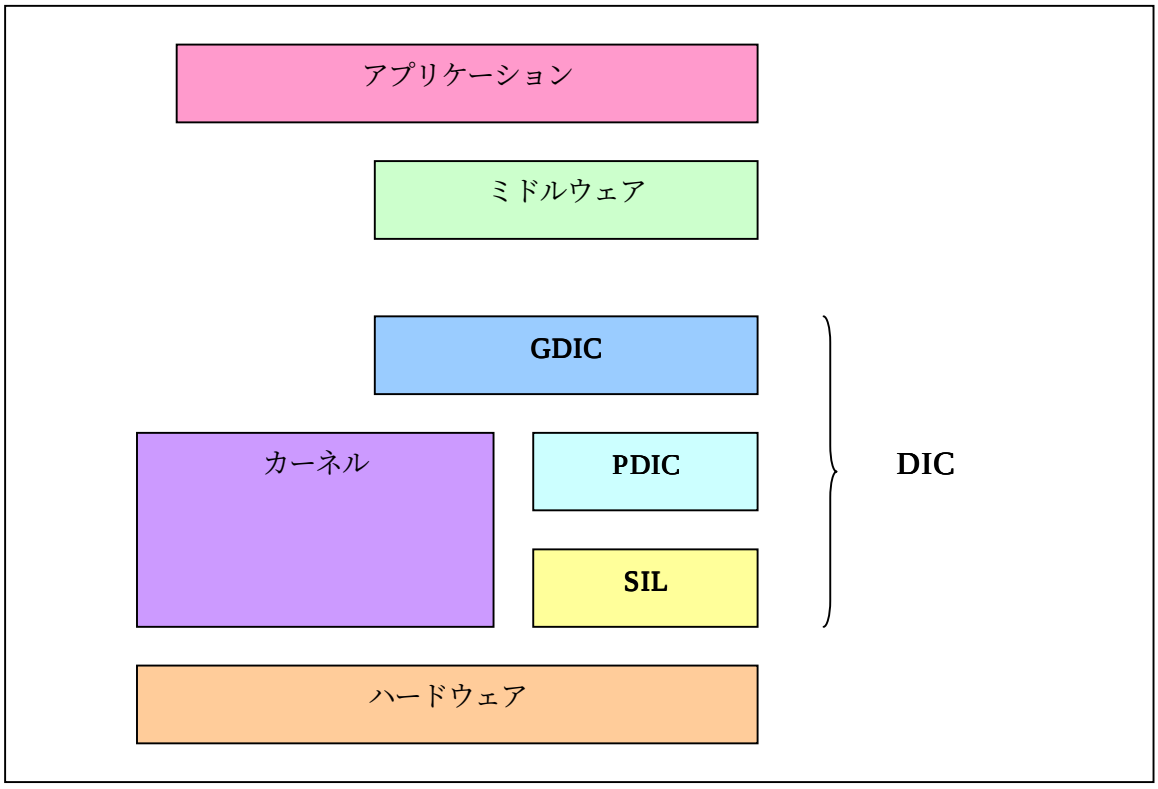

図 1　DIC の位置付けと概略構造

DIC 構造のうち、PDIC と SIL に関し、主要なデバイスを選定し、その機能のガイドラインを策定する。
策定の目的を以下に示す。

- 上位ソフトウェアとハードウェア制御プログラムとを分離することで、ソフトウェアの流用性を向上させる
- PDIC や SIL の具体的な機能例を策定することで、デバイスメーカでサンプルプログラムとして公開して貰える状況にする

ガイドラインの策定にあたり、下記のような基本的な方針とする。

- デバイスについての基本的な機能について策定を行い、デバイス固有の性能を出すための機能追加については拒まないものとする
- PDIC 及び SIL を OS に非依存とすることで、デバイスメーカがデバイスのサンプルプログラムとしての開発を行いやすいようにする

PDIC のガイドラインの策定に関する基本思想を以下のようにする。

- デバイス種別毎に基本機能を策定する
- 種別としては、シリアル、Ethernet、ストレージの３種を代表デバイスして考えていく
- PDIC では待ち状態を持たない（数μsec 程度のループ待ちは許容する）
- 割込み処理を含むこととするが、割込み処理はユーザで改変できるようにする
- 事象発生通知はコールバックルーチンの呼出しとする
- 初期化機能とオープン機能を分ける
  （初期化機能はデバイスのハードウェア的な初期化を主目的とし、オープン機能は PDIC に対する利用開始の宣言になり、この機能でデバイスを利用する際の条件を設定する）

２．PDIC の基本思想

PDIC の機能については、一般的な用途について本ガイドラインでは規定している。ただし、それ以外の用途やデバイスの持つ機構により求められる機能が異なる場合がある。
本ガイドラインで規定する機能や動作が異なる PDIC を提供する場合は、その違いをドキュメントに明記するようにする。

組込みシステムでは、多種のデバイスが利用されており、そのデバイスが動作するハードウェアも多岐に渡っている。
それら個々のデバイスや動作環境について全てを網羅することは困難なため、代表的なデバイスについてその機能を規定する。同類のデバイスについては、できるだけ合わせるようにし、異なるデバイスについては、PDIC の規定で行おうとする趣旨を把握してもらい、同様な機能を実装者で定義してもらうものとする。

２．１　基本思想

PDIC は、対象となるデバイスの基本的な機能をサポートするプログラムである。
同様の機能を果たす各種のデバイスの違いを PDIC で吸収する。すなわち、PDIC によりデバイスを抽象化することにより、PDIC を利用するプログラムに対しデバイスによる違いをあまり意識しないようにする。
PDIC を利用するプログラムがカーネルを利用する／しないにかかわらず、PDIC 自体ではカーネルの機能を利用しない。PDIC は、対象となるデバイスを制御するプログラムであり、パソコンの BIOS のような位置付けになる。ただし、PDIC は、要求される機能が処理できない場合は、ビジー返却を行い、PDIC 内部では待ち状態にはならない。
デバイスに対して複数のコマンドを出力するような場合、コマンドを連続出力する間に一定の時間を保証する必要がある。そのような時間を確保する際に、その時間が十数μsec 以内（CPU により異なるが、コンテキストの切替えによるオーバヘッドの数倍程度の時間）である場合、PDIC の内部でビジーループ（空ループ）を行ってもよい。

PDIC は割込みを利用する／しない場合の両方で利用できることを想定する。
たとえば、システムのブートプログラムでは、割込みを利用しないで PDIC を利用し、カーネルを利用したシステムでは割込みを利用して利用するようなことが考えられる。

本ガイドラインでは､デバイスに設定する情報やステータスなどその種別及びデータの値などを規定していない。
たとえば、シリアルコントローラのボーレートの値やステータス状態のビット情報などがある。
これらの情報はデバイス毎に異なる場合が多く、それらを規定することで変換のためのオーバヘッドがかかることが考えられ、規定しないこととした。
これら情報は、PDIC の実装者がドキュメントで説明をするようにする。

２．２　基本構成

PDIC は、PDSR（Primitive Device Service Routine）と ISR（Interrupt Service Routine）で構成される。
PDSR は、PDIC がサポートする機能を処理するプログラムである。
ISR は、割込み処理を行い、事象によりコールバックルーチンを呼出すプログラムである。
PDSR 及び ISR は、SIL が提供する機能を介しデバイスに対しアクセスする。

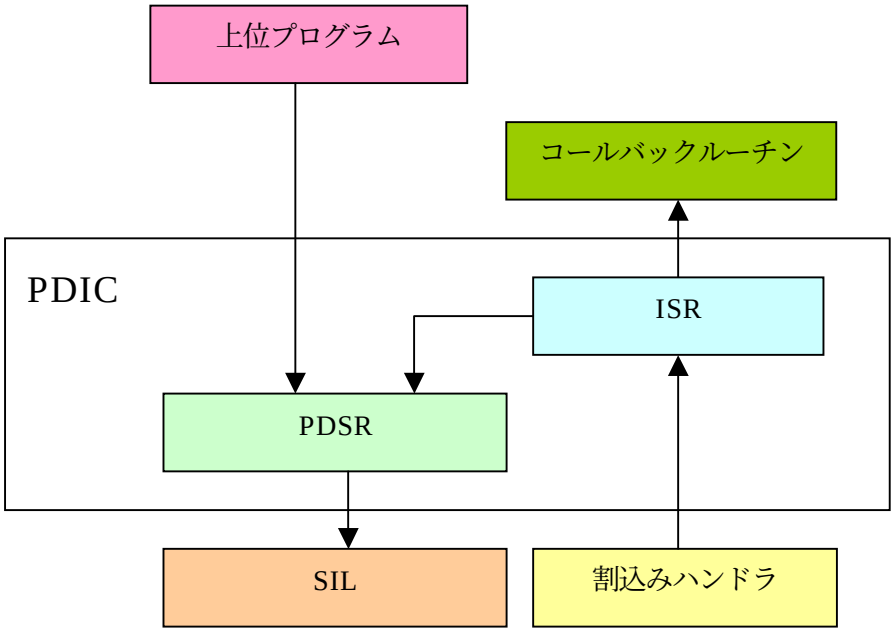

図 2　PDIC の基本構成

２．３　想定デバイス

PDIC で想定するデバイスは、プログラムで制御するコントローラを対象にしている。
このコントローラは、割込みを利用するものは多いと思われるが、割込みを利用しないものも対象とすることができる。
本ガイドラインでは、コントローラの単位をユニットとすることにした。ユニットは、ユニット番号で区別する。１つのデバイスで同様の機能を持つコントローラを複数持つものは、それぞれのユニットに対し要求できるものとする。１つのデバイスで、１つのコントローラの機能を持つ場合は、１つのユニットを持つことになる。
このユニットという名称は、デバイスによりチャネルやポートというデバイスの特性に合わせて名前が変わる場合がある。

複数のユニットを持つデバイスにおいて、それらのユニットを同時に制御できないものもある。たとえば、FDC であるμPD７６５は、４つのドライブを制御することができるが、複数のドライブに対し同時にアクセスすることができない。このような場合、複数にユニットにアクセスを要求する際に、PDIC を利用するプログラムにおいて排他制御を行う必要がある旨ドキュメントに記載する。
上記の記載がドキュメントにない場合は、複数のユニットに対し排他制御を行うことなく要求ができることとする。

（1）１つのコントローラで１つの機能を果たす場合

インテル社のシリアルコントローラである 8251 を対象とする PDIC は、１つのコントローラに１つのユニットを持つことになる。
ターゲットハードウェアに複数の 8251 のようなコントローラが実装されている場合は、PDIC は複数のユニットを持つ。
ただし、同じ機能を果たすコントローラ（たとえばシリアル）であっても、デバイスが異なる場合は、違う PDIC で制御を行う。

（2）１つのコントローラで複数の機能を果たす場合

日本電気社のシリアルコントローラであるμPD72001 のように１つのコントローラで２つのシリアルチャネルをサポートしている場合は、１つの PDIC で２つのユニットを持つ。

（3）１つのコントローラで複数の機能を果たすが、同時アクセスができない場合

日本電気社の FDC であるμPD765 のように１つのコントローラで４つのドライブをサポートしているが同時にアクセスができないものがある。
この場合、PDIC では４つのユニットを持つが、各ユニットを制御する処理の排他制御は行わない。
各ユニットへの要求が同時に発生しないように、PDIC の利用者に排他制御してもらう。このような PDIC のドキュメントには、各ユニットへの要求時には排他制御が必要なことを記載する。

２．４　基本機能

本ガイドラインで規定する基本的な機能を示す。

- デバイスの初期化機能
- ユニットのオープン機能
- ユニットのクローズ機能
- ユニットに対する出力機能
- ユニットに対する入力機能
- コールバックルーチンの許可／不許可機能
- ユニットのステータスの取得機能

その他の機能については、デバイスの特徴に合わせて機能を持たせるものとする。
PDIC を利用する前に、デバイスの初期化処理を行うため、初期化する要求を行う必要がある。
ユニットは、オープン状態とクローズ状態を持ち、オープン状態にて各種要求を受付けるようにする。オープン要求時にユニットに関する動作条件を設定する。クローズ状態の場合は、要求を受付けない状態とする。

（1）デバイスの初期化機能

指定される初期化情報に従い、デバイスを初期化する。
初期化する際に、リセット時のデフォルトの値を期待しないようにすることを推奨する。これは、システム稼動時に、再初期化を行うために、本初期化要求を行うことが考えられるからである。
初期化処理時に、対象となるデバイスが実装されていなかったり、ハードウェア不良を検出したような場合は、エラーを返却する。
本要求が行われた場合は、PDIC がサポートするユニットの状態をクローズ状態とする。

（2）ユニットのオープン機能

ユニットをオープン状態とし、各種の要求を受付けることができる状態にする。オープン要求時に指定される情報に従い、ユニットを初期化する。
デバイスからの割込みが受付けられる状態かどうかは、PDIC の実装者が定義し、その旨をドキュメントに記載する。
ドキュメントに記載がない場合は、割込みが受付けられない状態とする。

（3）ユニットのクローズ機能

ユニットをクローズ状態とし、各種の要求を受付けない状態とする。
本要求により、デバイスからの割込みを受付けない状態とする。

（4）ユニットに対する出力機能

ユニットに対し、データの出力を行う。
データは１バイト単位か、ブロック単位（複数バイト）であるかは、PDIC の実装者が定義する。
コントローラの状態が、データの出力ができない状態である場合、ビジー返却し、データの出力処理は行わない。

（5）ユニットに対する入力機能

ユニットからデータの入力を行う。
データは１バイト単位か、ブロック単位（複数バイト）であるかは、PDIC の実装者が定義する。
コントローラの状態が、データの入力ができない状態である場合、ビジー返却し、データの入力処理は行わない。

（6）コールバックルーチンの許可／不許可機能

コールバックルーチンの許可とは、実際はデバイスからの割込みを受付けられる状態にする（割込みのマスクを解除）ことである。
コールバックルーチンの不許可とは、実際はデバイスからの割込みを受付けない状態にする（割込みのマスクを設定）ことである。
コールバックルーチンとは、事象発生の割込み時に ISR から呼出されるルーチンである。
どのような割込みを許可／不許可にするかは、要求時にパラメータで指定する。

（7）ユニットのステータスの取得機能

ユニットの状態を取得する。
状態の情報については、PDIC の実装者が定義する。

２．５　コールバックルーチン

コールバックルーチンは、デバイスからの割込みが発生した際に ISR から呼出される。
呼出しの事象内容については、デバイスに対応した PDIC で定義する。
コールバックルーチンは、ユニークな固定シンボルを持つ関数とする。

コールバックルーチンの引き数は、ユニットをオープンする際に指定する拡張情報を通知する。
コールバックルーチンは、引き数を元に処理を行う。具体的には、事象に対する処理と、事象の発生を PDIC を利用するプログラムに通知する処理を行う。
コールバックルーチンからのリターンパラメータはなしとする。

２．６　割込みとコールバックルーチン

デバイスが動作するハードウェア環境、もしくはデバイスの機構により、複数の割込みの要因毎に割込みが分かれている場合と共有になっている場合がある。
送信と受信の割込みが共有になっている場合は、PDIC 中の ISR にて割込み要因ステータスを取得し（この場合、PDIC に割込み要因を取得する機能を追加する必要がある）、それに従いコールバックルーチンを呼出す。
エラー発生時の割込みについては、コールバックルーチンを通常処理のコールバックルーチンと分けるかどうかは実装で定義する。

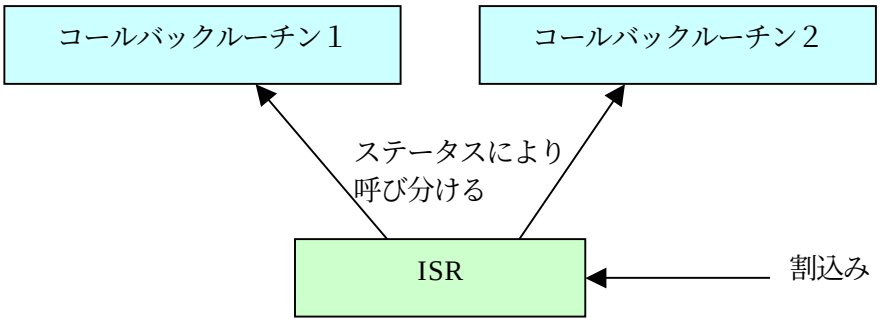

図 3　複数の要因が１つの割込みの場合

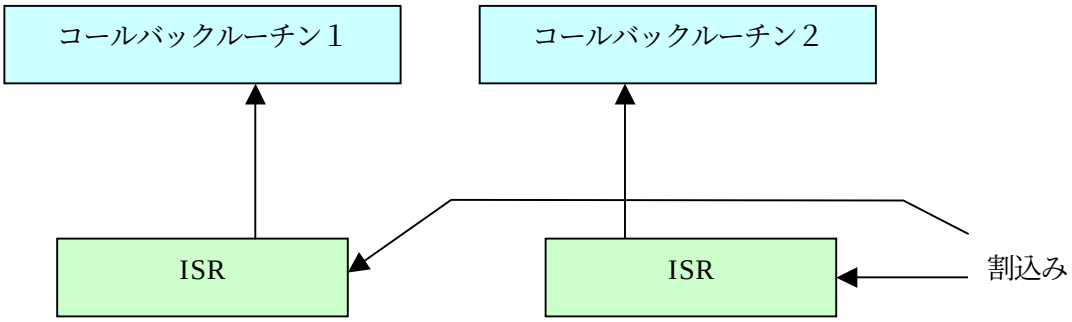

図 4　複数の要因が別々の割込みの場合

２．７　シンボル

本ガイドラインでは、各種の名称（シンボル）を規定する。

（1）API
PDIC が持つ機能の API 名称についてのシンボル規則を以下に示す。

xxx_yyy_zzz

xxx は、PDIC の対象とするデバイスの代表名を略名で示すものである。PDIC を利用するプログラムは、本ガイドラインで規定するシンボルを利用することを推奨する。PDIC を実装する場合は、xxx の部分を対象とするデバイス固有のユニークな名称とすることを推奨する。
PDIC を利用する場合は、C 言語の#define などによりシンボルの置き換えを行うようにする。

yyy_zzz は、ITRON の基本的な名称規則に準じ、yyy を操作の略称とし、zzz は対象を示す略称とする。

（2）各種シンボル

デバイスの各種設定情報やステータスなどの情報に対するシンボルについては、デバイス固有の情報として、特に規定しない。
具体的には伝送条件やハードウェアステータスなどの情報が考えられるが、デバイス毎に異なるため、デバイスの性能を引き出すために規定しないこととした。

（3）コールバックルーチン

コールバックルーチンは、PDIC の実装者が関数の枠組みを作成しておくこととする。
デバイスからの割込みが発生した場合、その事象に従い対応したコールバックルーチンを呼出す。コールバックルーチンの名称は、PDIC の実装者が定義することとする。

（4）シンボル利用例

PDIC の実装者は、API シンボルの xxx_yyy_zzz の”xxx_”をデバイス固有のユニークな名称とする。
PDIC の利用者は、本ガイドラインで示す API シンボル（”xxx_”がデバイスを代表する名称）でプログラムを記述しておき、C 言語の#define マクロによりシンボルを置換することを推奨する。
ターゲットハードウェアに同種のデバイスが複数あり、複数の PDIC を利用する場合の例を示す。

- デバイスを利用するプログラムのソースファイルを分け、それぞれのソースファイルでシンボルの置換を行う。そのソースファイルのヘッダ部にどのようなデバイスを利用するのか、どの PDIC を利用するのかを明記しておく。
- 利用する PDIC のシンボルを直接利用する。

２．８　移植性

本ガイドラインでは、PDIC の機能を規定しているが、デバイスの特性に依存する部分は、PDIC の実装者が定義することとしている。
デバイスに依存する部分は、PDIC の利用者がある程度の対応した処理を行うものとする。
以下のような箇所については、利用する PDIC により異なる場合がある。

- デバイスの初期化情報
- ユニットのオープン設定情報
- ステータス情報

３．シリアルコントローラ

シリアルコントローラといっても、各デバイスでそれぞれ動作が異なる場合が多い。

- ステータスが保持されるもの／１度ステータスを読み出すとクリアするもの
- FIFO バッファを有するもの／ないもの
- 送信時に送信レディで割込み事象が発生するもの／送信完了で割込みが発生するもの

動作環境によっても異なる。

- 送信と受信で割込みが１つの事象として通知されるもの／別事象として通知されるもの
- １つのハードウェアに複数の異なるシリアルコントローラが実装されている
- 独立したデバイスとして実装されている／CPU に内蔵されている

シリアルコントローラを利用する場合でも幾つか考えられる。

- １バイト毎に送受信を行う／DMA などを利用し連続で送受信を行う
- 同期型で利用する／非同期型で利用する

上記の全ての利用ケースを満足するガイドラインの策定は困難なため、シリアルコントローラは低速の１バイト毎に送受信を行う利用法を前提に考えていくこととする。

３．１　シリアル PDIC の機能

シリアルコントローラの PDIC(PDSR)には、以下の機能を規定する。

- 初期化要求　：　デバイスの初期化
- オープン要求　：　デバイスの利用開始
- クローズ要求　：　デバイスの利用終了
- 受信要求　：　データ受信
- 送信要求　：　データ送信
- コールバックルーチンの呼出し許可／不許可
  ：　事象発生（割込み）時にコールバックルーチンの呼出しの許可／不許可
- ステータス取得　：　デバイスステータス取得

シリアルコントローラを利用する場合は、システム立ち上がり時に「初期化要求」でデバイスの初期化を行う。
シリアルコントローラを利用しはじめる場合に、「オープン要求」を行い、コールバックルーチンの呼出し許可を行う。
シリアルの伝送条件は、オープン要求で設定する。
PDIC 内の割込みサービスルーチン（ISR）の登録は、PDIC を利用する側が行う。

３．２　利用想定

シリアルコントローラの PDIC をプログラムで利用する際に、割込みを利用して使う場合と、割込みを利用しないで使う場合とが考えられる。
本ガイドラインにおいて想定する利用方法を示す。

（1）データ送信

シリアルコントローラに対し PDIC を利用してデータを送信する例を示す。

例：割込みを利用しないでデータ送信する場合

PDIC を利用するプログラムで割込みを利用しないでデータを送信する場合は、CPU の割込みの受付けを禁止し（送信コールバックルーチンの呼出しを不許可）、PDIC に対しデータの送信要求（①）を行う。
データの送信要求が正常に行われたら、PDIC から正常返却される。連続してデータを送信する場合は、再度データの送信要求を行う。
PDIC がデータを送信できない場合は、ビジー返却されるので、PDIC の利用者は再度データの送信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、データの送信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

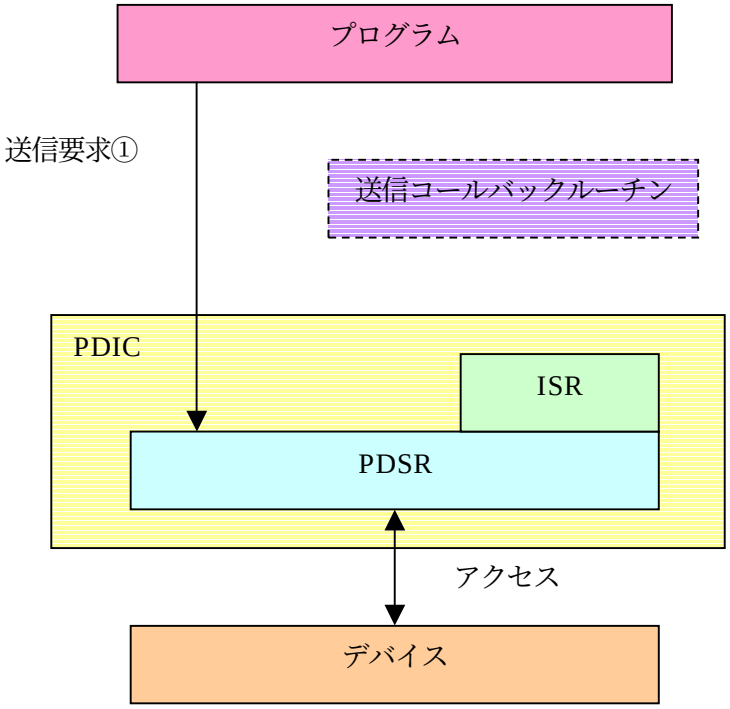

図 5　割込みを利用しないデータ送信

例：割込みを利用してデータ送信を行う場合

PDIC を利用するプログラムで割込みを利用してデータを送信する場合は、最初に PDIC に対しデータの送信要求（①）を行う。
データの送信要求が正常に行われたら、PDIC から正常返却される。連続してデータを送信する場合は、再度データの送信要求を行う。
PDIC がデータを送信できない場合は、ビジー返却されるので、PDIC の利用者は送信コールバックルーチンの呼出しを許可し、コールバックルーチンからの終了の事象を待つ。
送信割込みが発生すると、送信コールバックルーチン（②）が呼び出されるので、送信コールバックルーチンで PDIC に対しデータの送信要求（③）を行う。連続してデータを送信する場合は、再度送信コールバックルーチンが呼び出されるのを待ち、継続して PDIC に対しデータの送信要求を行う。
データの送信が終了したら、送信コールバックルーチンの呼出しを不許可にし、PDIC を利用するプログラムに対し、終了した旨の事象を通知（④）する。
送信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

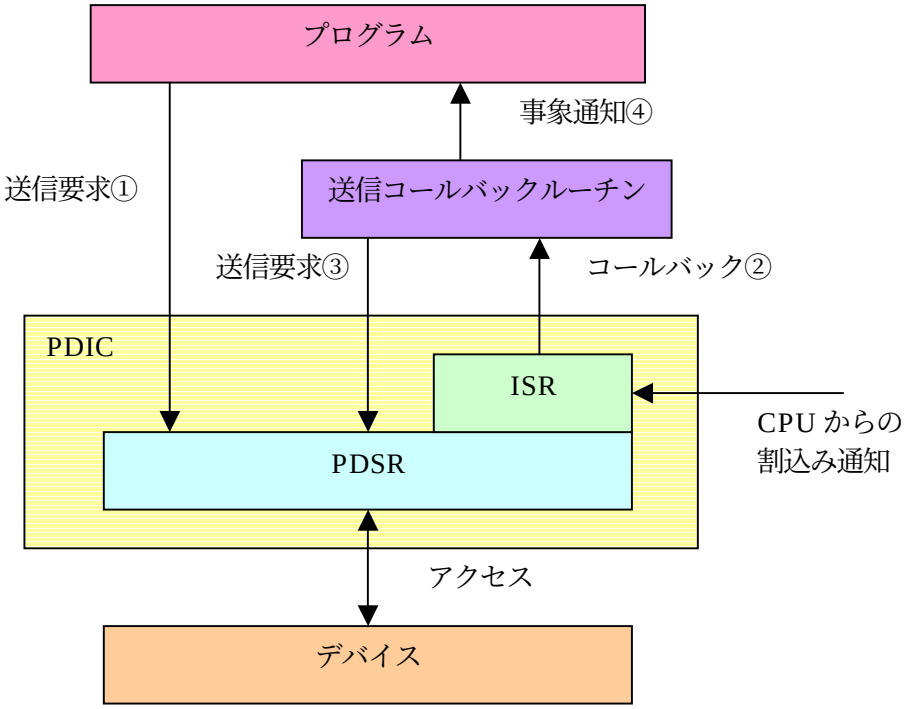

図 6　割込みを利用するデータ送信

補足説明

送信処理にて利用する割込みは、デバイスからの送信完了の割込みか、コントローラにデータがない（エンプティ）の割込みになる。どのような割込みかは、デバイスもしくは PDIC の実装により異なる。どのような事象により、送信コールバックルーチンが呼出されるかをドキュメントに記載する。
送信コールバックルーチンから PDIC を利用するプログラムに１バイトの送信毎に事象を通知すると、その処理にかかるオーバヘッドが大きい。
本ガイドラインでは、連続してデータを送信する場合、送信コールバックルーチンで PDIC に対し継続の出力要求を行うこととした。

（2）データ受信

PDIC では、受信要求が行われていない状態で受信したデータを保持するためのリングバッファなどを持たない。プログラムからの受信要求がなくても受信したデータを保持したい場合は、リングバッファなどを管理する処理は、PDIC を利用する側で用意するものとする。
または、プログラムからのデータの受信の要求が無い場合は破棄するとか、必要な時だけ受信を許可するとかは、PDIC を利用する側で決定する。

例：割込みを利用しないでデータ受信を行う場合

PDIC を利用するプログラムで割込みを利用しないでデータを受信する場合は、CPU の割込みの受付けを禁止し（受信コールバックルーチンの呼出し不許可）、PDIC に対しデータの受信要求（①）を行う。
データの受信要求が正常に行われたら、PDIC から受信したデータが返却される。連続してデータを受信する場合は、再度データの受信要求を行う。
PDIC がデータを受信できない場合は、ビジー返却されるので、PDIC の利用者は再度データの受信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、データの受信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

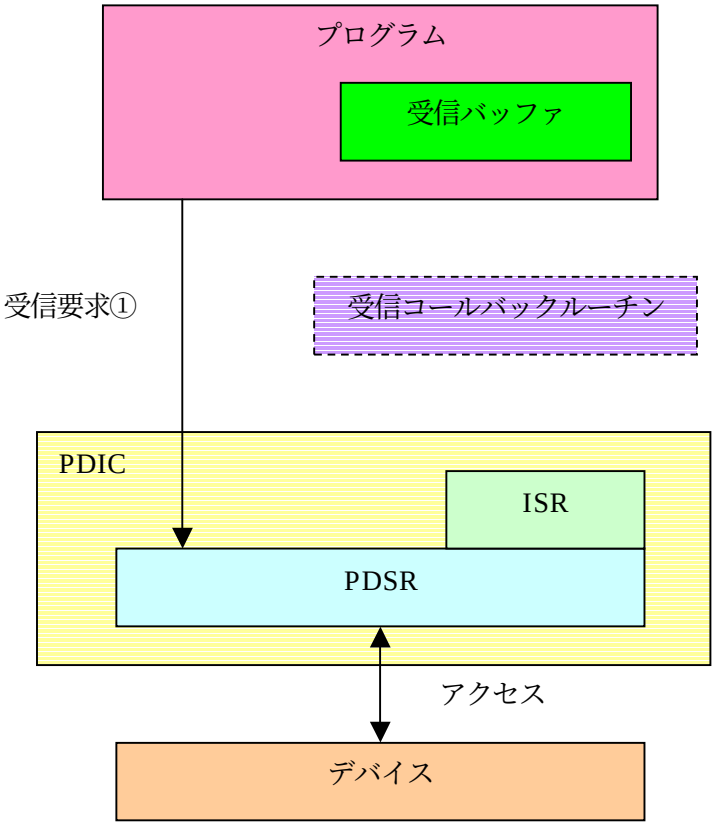

図 7　割込みを利用しないデータ受信

例：割込みを利用してデータ受信を行う場合

ここでは、プログラムで受信処理を開始する時点からデータの受信処理を行うことを前提とし説明する（先行されるデータはないという前提）。

PDIC を利用するプログラムは、PDIC にデータの受信要求（①）を行い、データが受信できた場合は、バッファに格納する。連続してデータを受信する場合は、再度 PDIC にデータの受信要求を行う。

PDIC からビジー返却された場合、受信コールバックルーチンの許可を行い、コールバックルーチンからの終了の事象を待つ。

受信割込みが発生すると、受信コールバックルーチンが呼び出される（②）ので、受信コールバックルーチンで PDIC に対しデータの受信要求（③）を行う。連続してデータを受信する場合は、再度送信コールバックルーチンが呼び出されるのを待ち、継続して PDIC に対しデータの受信要求を行う。

データの受信処理が終了したら、受信コールバックルーチンの呼出しを禁止し、PDIC を利用するプログラムに対し、終了した旨の事象を通知（④）する。

受信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

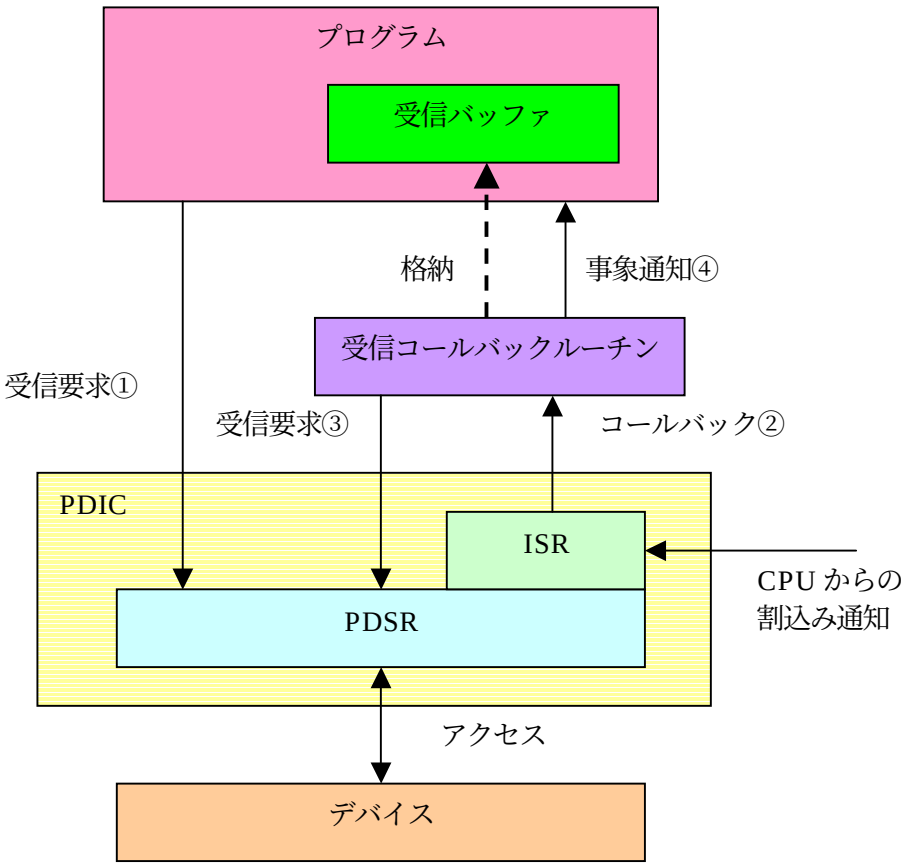

図 8　割込みを利用するデータ受信

補足説明

受信時のエラーを別のコールバックルーチンとするか､受信処理にてステータスから判断するかはデバイス及び実装で定義する。
どのような事象により､受信コールバックルーチンが呼出されるかをドキュメントに記載する。
受信コールバックルーチンから PDIC を利用するプログラムに１バイトの受信毎に事象を通知すると、その処理にかかるオーバヘッドが大きい。
本ガイドラインでは、連続してデータを受信する場合、受信コールバックルーチンで PDIC に対し継続の入力要求を行うこととした。

３．３　コールバックルーチン

シリアルコントローラの PDIC は、送信コールバックルーチンと受信コールバックルーチンの２つを定義する。
送信コールバックルーチンは、送信フレームが空いた（送信が可能になった）場合に呼出される。具体的には、送信完了の割込み発生時に呼出される。
受信コールバックルーチンは、シリアルコントローラからの受信割込みにより呼出される。受信コールバックルーチンでは、PDIC に対し受信要求を行い、受信したデータを保存する。

PDIC の実装者は、PDIC に対し送信要求または受信要求を行うコールバックルーチンを標準で提供する。ただし、送信または受信の完了の判定及び完了処理は、利用者で修正できるようにする。
PDIC の利用者は、コールバックルーチンを実装時に動作する環境に合わせて修正を行う。

ポートのオープン時にはコールバックルーチンの呼出しは不許可状態になっている。割込みを利用する PDIC の利用者は、ポートをオープンした後、受信コールバックルーチンの許可を要求する。

コールバックルーチンの引き数は、割込みが発生したポート番号と、ポートの拡張情報とする。
コールバックルーチンは、この２つの引き数を元に処理を行う。

３．４ PDIC 機能

シリアルコントローラの PDIC の機能として規定する機能を以下に示す。

sio_ini_dev	：	初期化要求 デバイスの初期化
sio_opn_por	：	オープン要求 ポートのオープン
sio_cls_por	：	クローズ要求 ポートのクローズ
sio_snd_chr	：	送信要求 データの送信
sio_rcv_chr	：	受信要求 データの受信
sio_ena_cbr	：	コールバック許可要求 コールバックルーチンの呼出し許可
sio_dis_cbr	：	コールバック不許可要求 コールバックルーチンの呼出し不許可
sio_get_sts	：	ステータス取得要求 デバイスのステータスの取得

シリアルコントローラの PDIC の機能として規定する API 名称において、sio_xxx_yyy の “ sio_ ” の部分は、対象とするコントローラを示すユニークな名称とする。

以下に示す機能のパラメータにおいて、網掛けしたパラメータ構造は PDIC 実装者がコントローラに合わせて実装定義する。
PDIC 利用者は、実装定義のパラメータの内容を変更する。

シリアルコントローラを利用する場合は、システム立ち上がり時に「初期化要求」でデバイスの初期化を行う。
シリアルコントローラを利用しはじめる場合に、「オープン要求」を行い、コールバックルーチンの呼出し許可を行う。
各ユニットの伝送条件は、オープン要求で設定する。
PDIC 内の割込みサービスルーチン（ISR）の登録は、PDIC を利用する側が初期化要求後に行う。

（1）初期化要求（SIO　Initialize　Device）

【API】
ER　　sio_ini_dev（T_IINFO　*info）

【パラメータ】
　　T_IINFO　　*info　　：　初期化パラメータ
　　デバイスを初期化するために必要な情報
　　T_IINFO　　実装依存

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　シリアルコントローラの初期化パラメータに従い初期化を要求する。

【注意事項】
　　初期化処理中は、PDIC 内で CPU の割込みを禁止して行うことを原則とし、要求時の CPU の割込み禁止／許可状態は保存されるものとする。
　　PDIC は、システムの起動時やデバイスのリセット時に呼び出されてもいいように、電源 ON 時の初期値に依存しないようにする。

（2）オープン要求（SIO　Open　Port）

【API】

ER　　sio_opn_por（INT　portno, T_OINFO　*info, VP_INT　exinf）

【パラメータ】

INT　　portno　：　ポート番号（0～）
複数のポート（チャネル）をサポートするデバイスの場合、
それぞれを区別するための番号

T_OINFO　　*info　：　通信パラメータ
伝送条件を設定する情報（実装定義）
T_OINFO　　ボーレート
ビット数
ストップビット
パリティ
フロー制御

VP_INT　　exinf　：　拡張情報
コールバックルーチンへの引き数
内容は PDIC の利用者が利用できる

【リターンパラメータ】

E_OK　：　正常
負数　：　エラーコード

【機能】

指定されるポート番号に相当するポートを指定伝送条件に従いデバイスの設定を行う。
デバイスに FIFO バッファなどが搭載されている場合はバッファをクリアし、対象ポートのエラーリセットを行う。FIFO バッファなどが搭載されている場合の設定は基本的に PDIC で行う。
正常に処理できた場合、対象ポートをオープン状態とし、通信に関する要求を受付ける状態とする。
PDIC からコールバックルーチンを呼び出す場合に、指定される拡張情報を引き数として渡す。

【注意事項】

オープン処理中は、PDIC 内で CPU の割込みを禁止して行うことを原則とし、要求時の CPU の割込み禁止／許可状態は保存されるものとする。
伝送条件は、システム稼動中に変更される可能性があるため、PDIC は、電源 ON 時の初期値に依存しないようにする。
多重のオープン要求はエラーとする。

（3）クローズ要求（SIO　Close　Port）

【API】
ER　　sio_cls_por（INT　portno）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートの通信を停止する。
　　対象ポートの通信処理を停止し、送信／受信のコールバック呼出しを不許可にする。
　　対象ポートをクローズ状態とし、通信に関する要求を受付けない状態とする。

【注意事項】
　　クローズ処理中は、PDIC 内で CPU の割込みを禁止して行うことを原則とし、要求時の CPU の割込み禁止／許可状態は保存されるものとする。
　　多重のクローズ要求はエラーとする。

（4）送信要求（SIO　Send　Character）

【API】
ER　　sio_snd_chr（INT　portno, B　data）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　B　　data　　：　送信データ

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに対しデータを出力する。
　　対象ポートが出力できない状態の場合、ビジー返却する。

【注意事項】
　　PDIC では、送信要求されたデータを編集せずにそのまま出力する（編集とは、TAB コードの変換や不可視コードの可視化などをいう）。

（5）受信要求（SIO　Receive　Character）

【API】
ER_UINT　　sio_rcv_chr（INT　portno）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）

【リターンパラメータ】
　　正数　　：　受信データ
　　負数　　：　エラーコード
　　　　（ハードウェアエラー発生時はステータスを反映する）

【機能】
　　指定されるポート番号に相当するポートからデータを入力する。
　　対象ポートからデータの入力がない状態の場合、ビジー返却する。

【注意事項】

（6）コールバックルーチン許可要求（SIO　Enable　Call-Back　Routine）

【API】
ER　　sio_ena_cbr（INT　portno, UINT　flag）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　UINT　flag　　：　許可する送信／受信コールバックルーチンを示すフラグ
　　　　送信コールバック許可　　：　ｘｘｘ
　　　　受信コールバック許可　　：　ｙｙｙ
　　　　これらのフラグは、OR で指定することができる

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに関する指定処理のコールバックの呼出しを許可する。
　　具体的には、送信割込み、受信割込み、エラー割込みを受付けられるようにする（割込みを許可する）。

【注意事項】

（7）コールバックルーチン不許可要求（SIO　Disable　Call-Back　Routine）

【API】
ER　　sio_dis_cbr（INT　portno, UINT　flag）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　UINT　flag　　：　不許可する送信／受信コールバックルーチンを示すフラグ
　　　　送信コールバック不許可　：　ｘｘｘ
　　　　受信コールバック不許可　：　ｙｙｙ
　　　　これらのフラグは、OR で指定することができる

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに関する指定処理のコールバックルーチンの呼出しを不許可にする。
　　具体的には、送信割込み、受信割込み、エラー割込みが発生しないようにする。

【注意事項】

（8）ステータス取得（SIO　Get　Status）

【API】
ER_UINT　　sio_get_sts（INT　portno）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）

【リターンパラメータ】
　　正数　　：　対象デバイスのステータス（実装定義）
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートの情報を返却する。

【注意事項】

４．Ethernet コントローラ

Ethernet を用いた通信は、その性質上高速な応答性が求められることから、事前に複数の受信バッファを確保しておくことが必要となる。又、Ethernet コントローラは、送受信バッファとのデータ授受に DMA 転送を用いるのが一般的であるため、PDIC の構造はコントローラに対するデータブロックの授受が主体となる。（Ethernet コントローラ用の PDIC の場合、キャラクタ単位の入出力インタフェースは現実的でない）

Ethernet コントローラによっては、送受信バッファをチップ内に内蔵するものと、外部メモリ（コントローラからみた）やバス空間（例えば PCI バス空間に対して DMA 転送する等）を利用する場合がある。本ガイドライン（Ethernet コントローラ）では、前者を「コントローラ内にバッファが用意されている」，後者を「外部メモリのバッファを利用する」と表記する。

上記の何れの場合でも、送受信バッファを管理するための情報が必要となる。この管理情報を以降「バッファディスクリプタ」又は単に「ディスクリプタ」と記述する。

バッファディスクリプタの構造及び配置は、各 Ethernet コントローラの仕様により異なるが、基本的なアーキテクチャは共通であると考えられる。
一般的なバッファディスクリプタの構成要素は、「送受信バッファの容量」，「送受信バッファの先頭アドレス」，「バッファの消費順番」，「バッファのアクセス権（CPU 及び Ethernet コントローラの何れがバッファに対するアクセスが可能か）」及び「データ転送結果」等を管理するテーブルで、Ethernet コントローラ内（レジスタ又はメモリ）又は、外部メモリやバス空間に配置される。

Ethernet コントローラ用の PDIC では、イーサネットフレームの送受信を行うための基本動作は「バッファディスクリプタを効率的に操作する」ことにより実現するものとした。
PDIC 内では送受信データのコピーを一切行わないことを基本とし、バッファへの送受信データの書込み及び読出しは、PDIC を利用する上位プログラムが実施するものとした。

【補足説明】
Ethernet コントローラの他にもバッファディスクリプタを用いたコントローラ（例えば、高速シリアル通信（HDLC などの）コントローラ）が存在する。本ガイドラインでは代表的なデバイスを対象にしているため特に言及していないが、他のデバイスでもバッファディスクリプタを構成するコントローラであれば、本章で示す PDIC 実装を適用することが可能である。

４．１　データバッファについて

前述のとおり、Ethernet コントローラには、送受信バッファの配置方法として「コントローラ内にバッファが用意されている」と「外部メモリのバッファを利用する」がある。
前者の場合、PDIC を利用する上位プログラムにて、コントローラ内のバッファに対する送信データのコピーが必須となる。これに対して後者の場合には、PDIC を利用する上位プログラムが用意したバッファ（送信データが格納されている）から直接送信することが可能となる。後者の方式を用いることにより「ゼロコピー」を実現することができる。
本ガイドラインでは、上記の２つの方式について PDIC の機能を規定する。

（1）「コントローラ内にバッファが用意されている」場合

Ethernet コントローラがデータの送受信を行うのに、コントローラ内のバッファを利用する必要がある場合、コントローラ内に送信バッファと受信バッファを構成する。

PDIC のオープン要求時に、送信バッファ及び受信バッファのディスクリプタの情報を設定する。
それぞれのバッファ数などの設定情報については、PDIC の実装者が定義する。

PDIC の利用者は、データを送信する場合は、PDIC が用意する「送信フレーム取得要求」で送信バッファ（本ガイドラインではこれをフレームという）を取得し、そのバッファに送信データを格納する。送信データを格納したら、PDIC に対し「フレーム送信要求」を行う。「送信フレーム取得要求」により取得されたバッファの順番に「フレーム送信要求」されることを原則とするが、PDIC の実装により任意の順番に「フレーム送信要求」ができる実装にしてもよい。
利用者は、コールバックルーチンの呼出しもしくはステータスを参照することにより、送信の完了を知ることができる。

データを受信する場合は、PDIC に対し「フレーム受信要求」で受信データがあるバッファのアドレスを取得する。受信データを参照し処理が完了した時点で PDIC に対し「受信フレームバッファ解放要求」を要求する。

「受信フレームバッファ解放要求」により、該当バッファのアクセス権がコントローラ側に移り、コントローラは該当バッファを受信データの格納に利用することが可能になる。
解放要求を行わない限り、バッファ内のデータが次の受信データにて上書きされることは無い。
（一旦、解放要求を行った後に、該当バッファを参照した場合には内容が保証されない。）

【補足説明】
PDIC を利用する上位プログラムが、コントローラ内のバッファにアクセスする場合には、コントローラ固有のアクセス手順が必要になる場合がある。例えば、ISA バスのインタフェースを内蔵したコントローラの場合、所定のレジスタにアドレスを書込んだ後にデータアクセスする等の手順が必要となる。本ガイドラインでは、これら個々のアクセス手順については言及しない。

（2）「外部メモリのバッファを利用する」場合

Ethernet コントローラがデータの送受信を行うのに、外部メモリのバッファを指定できる場合、データを受信するための受信バッファ領域を確保し PDIC に通知する。PDIC は通知された情報により受信ディスクリプタを構成する。コントローラはフレーム受信時に受信ディスクリプタの情報にもとづき受信バッファ領域へ受信データの転送を行う。
フレーム送信時は、送信要求時に指定される送信バッファ（PDIC を利用する上位プログラムが確保したバッファ）の情報を送信ディスクリプタに構成する。コントローラは送信ディスクリプタにもとづき送信バッファよりフレームの送出を行う。

PDIC のオープン要求時に受信バッファの情報にもとづきディスクリプタを構成する。バッファの数やアドレスなどの設定の情報については、PDIC の実装者が定義する。

PDIC の利用者は、データを送信する場合は、送信データが格納済みのバッファを指定し、PDIC に対し「フレーム送信要求」を行う。PDIC は「フレーム送信要求」にもとづき、送信ディスクリプタを（既にいくつかの送信要求が行われている場合は最後尾へ）追加する。コントローラは、送信ディスクリプタを参照し順次フレームの送出を行う。

利用者は、コールバックルーチンの呼出しもしくはステータスを参照することにより、送信の完了を知ることができる。
データの受信は、PDIC に対する「フレーム受信要求」を要求し、受信データがあるバッファのアドレスを取得する。受信データを参照し処理が完了した時点で PDIC に対し「受信フレームバッファ解放要求」を要求する。

【補足説明】
PDIC を利用する上位プログラムが、バス空間のバッファにアクセスする場合には、利用するバスのアクセス手順が必要になる場合がある。例えば、PCI バスのインタフェースを内蔵したコントローラの場合、PCI バスコントローラを介してアクセスする必要が生じる。このようなケースで、上位プログラムが利用するデータ領域からバス空間へ送信データのコピーを必要とする実装の場合には、前述（1）のインタフェースを適用すればよい。
本ガイドラインでは、特定のバスのアクセス手順については言及しない。

（3）ポータビリティを重要視する場合

上記の（1）と（2）とでは、送信バッファの扱いが大きく異なる。そのため、それぞれで PDIC を利用する上位プログラムを変更する必要がある。
そのため PDIC を利用するプログラムのポータビリティを考慮した場合、（2）のコントローラであっても（1）と同様の機能を持たせることによりインタフェースを共通化することができる。
どのような機能を提供するかは、PDIC の実装者が定義する。

４．２　利用想定

Ethernet コントローラの PDIC を上位プログラムから利用する際に、割込みを利用して使う場合と、割込みを利用しないで使う場合とが考えられる。
本ガイドラインにおいて想定する利用方法を示す。

４．２．１　「コントローラ内にバッファが用意されている」場合

コントローラ内にバッファが用意されている場合、オープン時に送信バッファディスクリプタと受信バッファディスクリプタとを設定する。
フレームを送信する場合は、送信フレームバッファを取得し、そのバッファに送信データを格納し、送信要求を行う。
フレームを受信する場合は、受信要求で受信フレームバッファを取得し、データの参照が終了したら、受信フレームバッファを返却する。

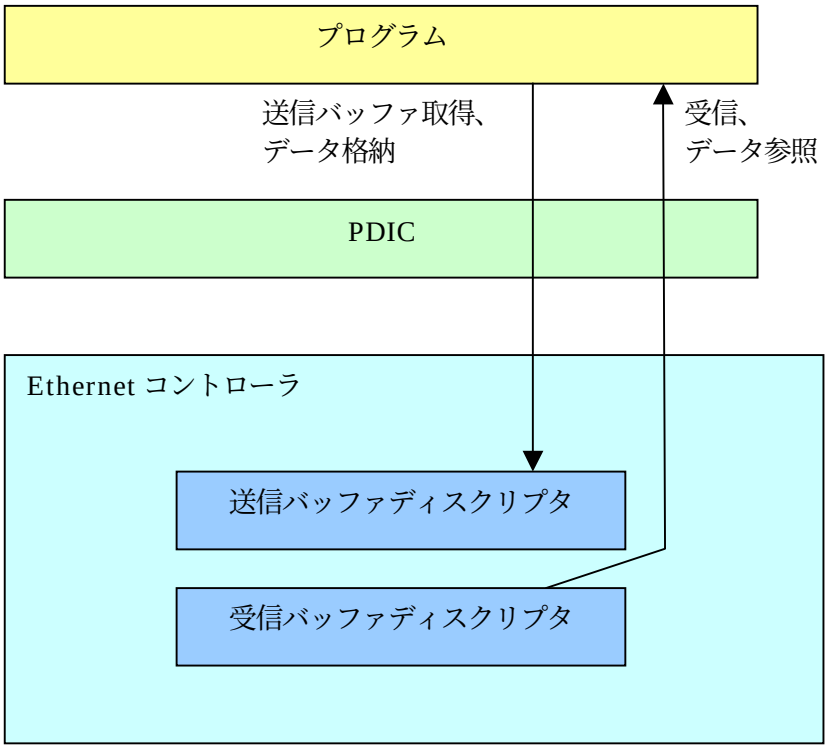

図 9　コントローラ内にバッファが用意されている場合

（1）データ送信

Ethernet コントローラに対し PDIC を利用してデータを送信する例を示す。

例：割込みを利用しないでデータ送信する場合

PDIC を利用するプログラムで割込みを利用しないでデータを送信する場合は、送信フレームバッファを取得（①）し、そのバッファにデータを格納する。
CPU の割込みを禁止し（送信コールバックルーチンの呼出しを不許可）、PDIC に対しフレーム送信要求（②）を行う。
データの送信要求が正常に行われたら、PDIC から正常返却される（ただし、送信処理そのものは完全に完了しているとは限らない）。データが完全に送信できたかどうかは、ステータスを取得し、ステータスをチェックする。
連続してデータを送信する場合は、再度データのフレーム送信要求を行う。
PDIC がデータを送信できない場合は、ビジー返却されるので、PDIC の利用者は再度フレーム送信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC の利用者は、フレーム送信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

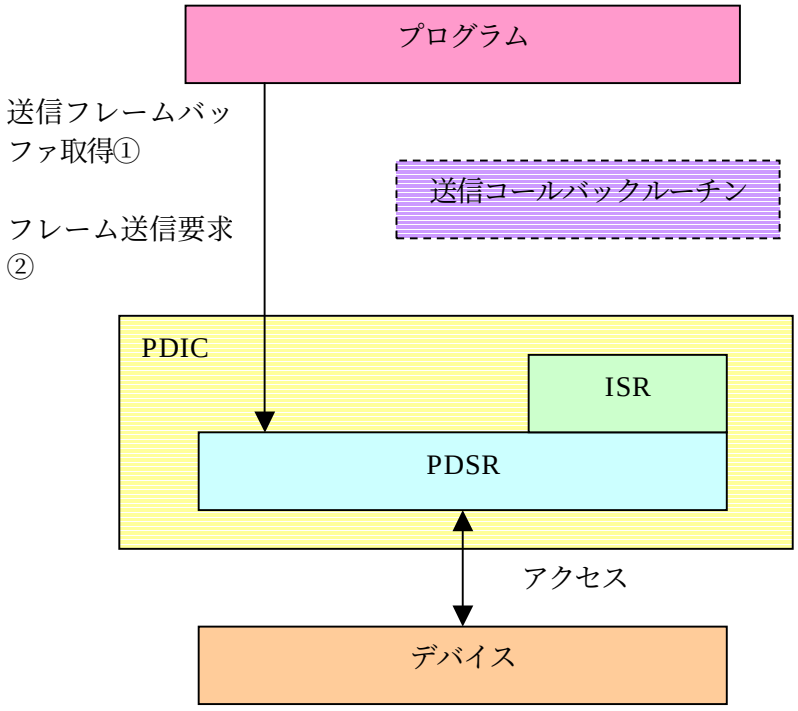

図 10　割込みを利用しないデータ送信

例：割込みを利用してデータ送信を行う場合

PDIC を利用するプログラムで割込みを利用してデータを送信する場合は、送信フレームバッファを取得（①）し、そのバッファにデータをコピーする。
PDIC に対し、送信コールバックルーチンの呼出しを許可し、フレーム送信要求（②）を行う。
フレーム送信要求が正常に行われたら、PDIC から正常返却される。送信処理が完全に終了したことを知るには、送信コールバックルーチンからの通知を待つ。
連続してデータを送信する場合は、再度フレーム送信要求を行う。
送信割込みが発生すると、送信コールバックルーチンが呼び出される（③）。
PDIC がデータを送信できない場合は、ビジー返却されるので、送信コールバックルーチンからの事象通知（④）を待つ。
送信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

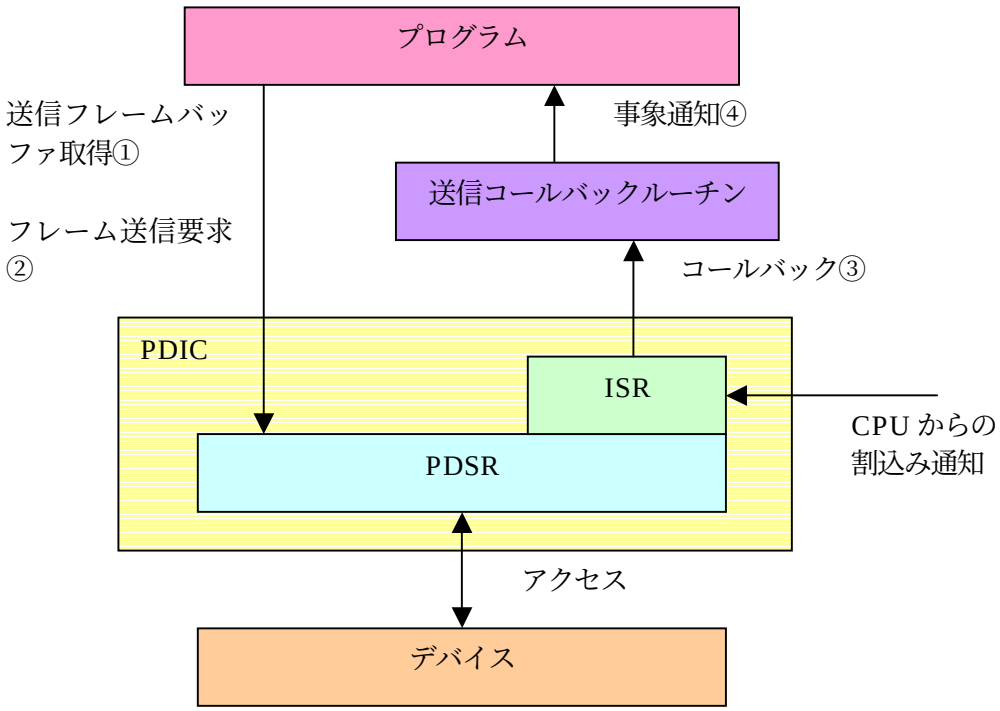

図 11　割込みを利用するデータ送信

（2）データ受信

PDIC に対する受信要求では、Ethernet コントローラ内にある受信データが格納されているフレームバッファのアドレスを通知する。

例：割込みを利用しないでデータ受信を行う場合

PDIC を利用するプログラムで割込みを利用しないでデータを受信する場合は、CPU の割込みの受付けを禁止し、PDIC に対しフレーム受信要求（①）を行う。
データの受信要求が正常に行われたら、PDIC から受信データがあるフレームバッファのアドレスが返却される。連続してデータを受信する場合は、再度フレーム受信要求を行う。
PDIC がデータを受信できない場合は、ビジー返却されるので、PDIC の利用者は再度フレーム受信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、受信データの参照が終了したら、受信フレーム解放通知（②）を要求する。
受信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

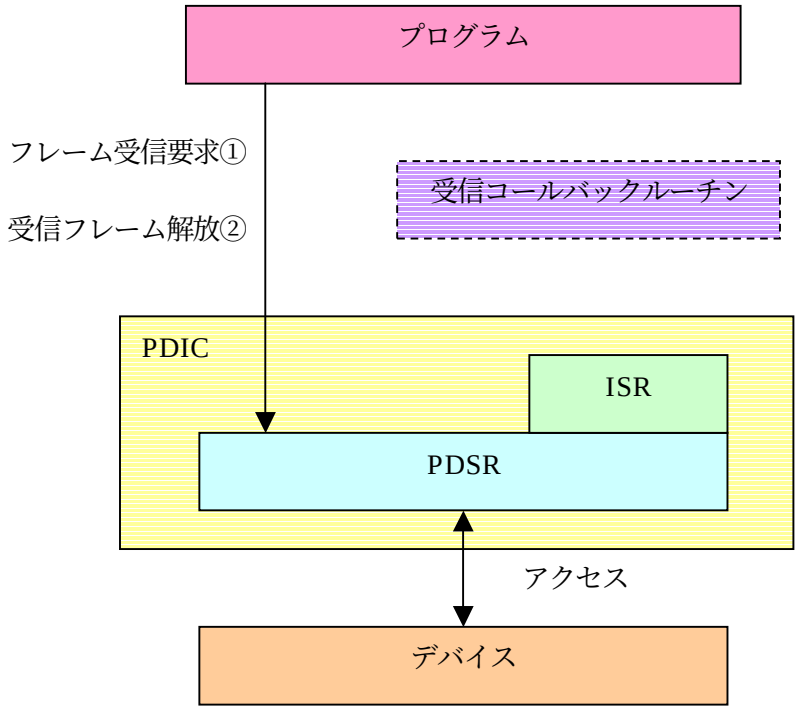

図 12　割込みを利用しないデータ受信

例：割込みを利用してデータ受信を行う場合

PDIC を利用するプログラムは、ポートに対しオープン要求を行った後、受信コールバックルーチンの許可を要求しておく。

受信割込みが発生すると、受信コールバックルーチンが呼び出され（①)、受信コールバックルーチンから事象通知（②）が行われる。

PDIC を利用するプログラムは、PDIC にフレーム受信要求（③）を行い、受信できた場合は、受信フレームバッファのアドレスを取得し、受信データを参照する。

PDIC を利用するプログラムは、受信データの参照が終了したら、受信フレーム解放通知（④）を要求する。

受信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

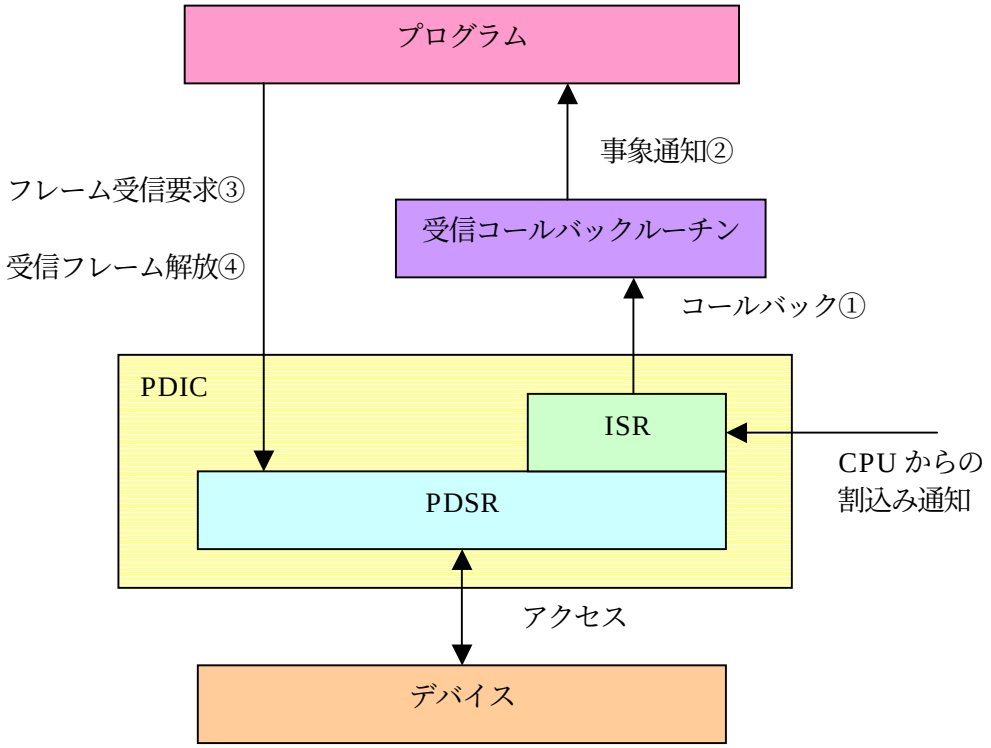

図 13　割込みを利用するデータ受信

４．２．２　「外部メモリのバッファを利用する」場合

プログラムが編集したバッファを送受信フレームのバッファとして、そのまま利用できる場合、オープン時に利用者が確保した領域に受信用のディスクリプタを設定する。

フレームを送信する場合は、送信データがあるフレームバッファを指定し、フレーム送信要求を行う。
フレームを受信する場合は、受信要求で受信フレームバッファを取得し、データの参照が終了したら、受信フレームバッファを返却する。

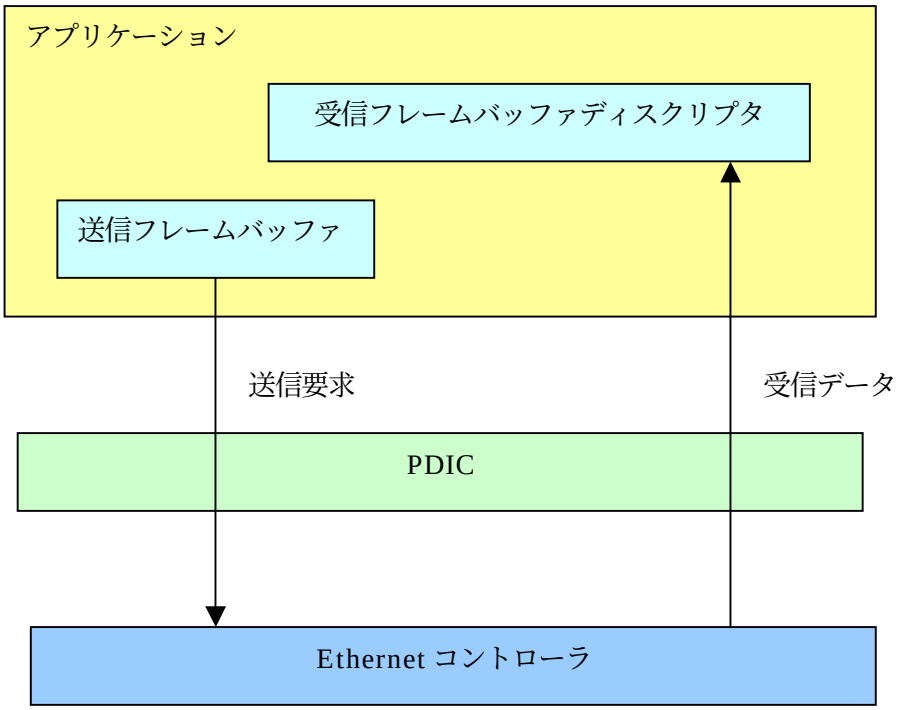

図 14　外部メモリのバッファを利用する場合

（1）データ送信

Ethernet コントローラに対し PDIC を利用してデータを送信する例を示す。

例：割込みを利用しないでデータ送信する場合

PDIC を利用するプログラムで割込みを利用しないでデータを送信する場合は、CPU の割込みの受付けを禁止し、送信フレームバッファを指定して、PDIC に対しフレーム送信要求（①）を行う。
データの送信要求が正常に行われたら、PDIC から正常返却される（ただし、送信処理そのものは完全に完了しているとは限らない）。データが完全に送信できたかどうかは、ステータスを取得し、ステータスをチェックする。
連続してデータを送信する場合は、再度フレーム送信要求を行う。
PDIC でデータを送信できない場合は、ビジー返却されるので、PDIC の利用者は再度フレーム送信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、フレームの送信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

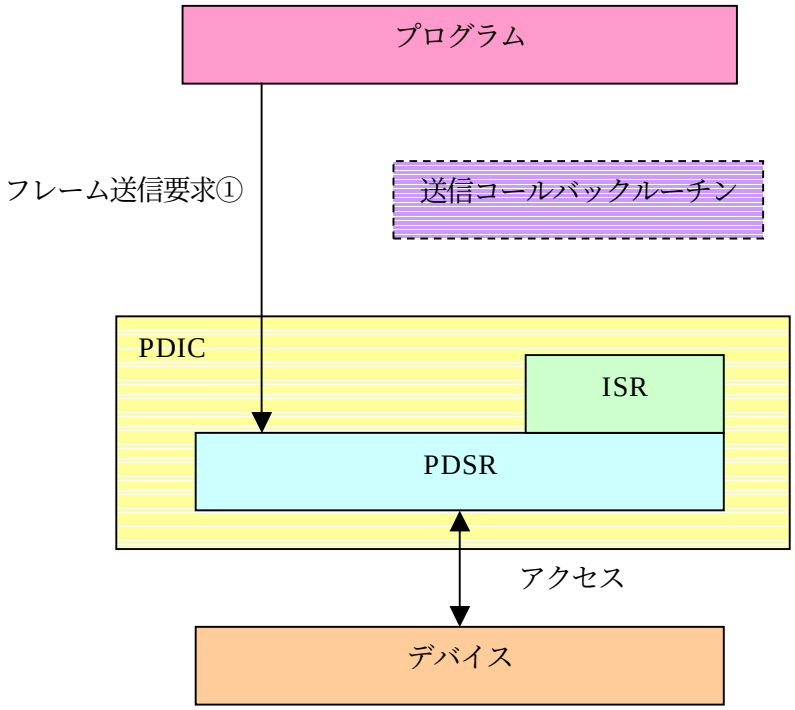

図 15　割込みを利用しないデータ送信

例：割込みを利用してデータ送信を行う場合

PDIC を利用するプログラムで割込みを利用してデータを送信する場合は、送信フレームバッファを指定して、送信コールバックルーチンの呼出しを許可し、PDIC に対し、フレーム送信要求（①）を行う。データの送信要求が正常に行われたら、PDIC から正常返却される。送信処理が完全に終了したことを知るには、送信コールバックルーチンからの通知を待つ。

連続してフレームを送信する場合は、再度フレーム送信要求を行う。

送信割込みが発生すると、送信コールバックルーチンが呼び出される（②）。

PDIC がフレームを送信できない場合は、ビジー返却されるので、PDIC の利用者は、送信コールバックルーチンからの前のフレーム送信の終了通知（③）を待つ。

送信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

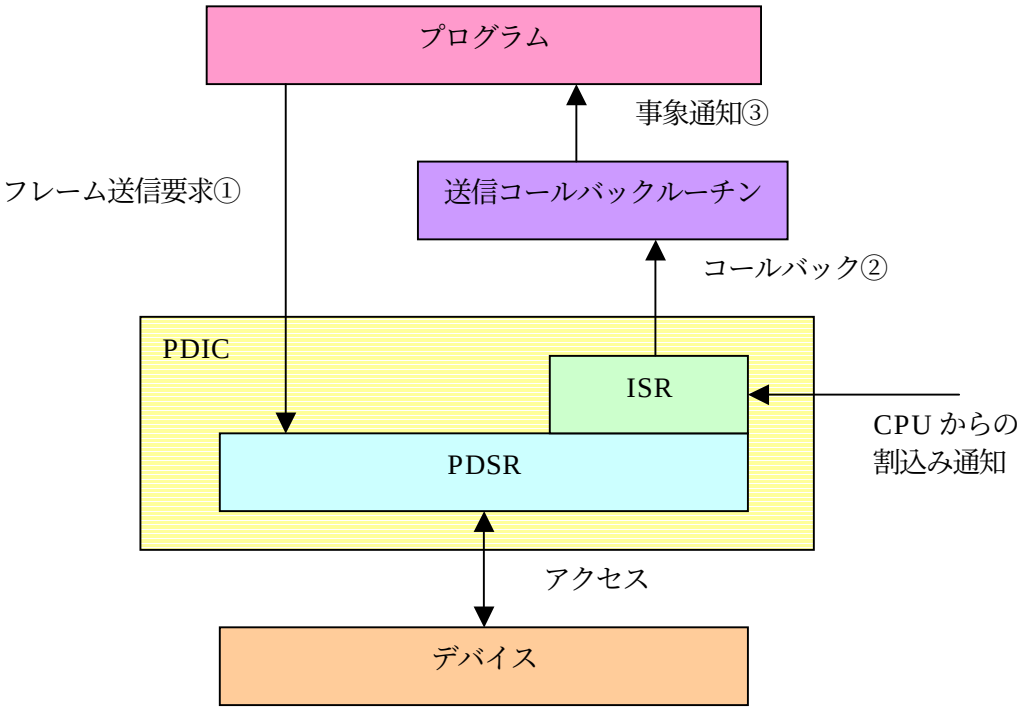

図 16　割込みを利用するデータ送信

（2）データ受信

PDIC に対する受信要求では、オープン時に設定するディスクリプタ内にある受信データが格納されているフレームバッファのアドレスを通知する。

例：割込みを利用しないでデータ受信を行う場合

PDIC を利用するプログラムで割込みを利用しないでデータを受信する場合は、CPU の割込みの受付けを禁止し、PDIC に対しフレーム受信要求（①）を行う。
データの受信要求が正常に行われたら、PDIC から受信データがあるフレームバッファのアドレスが返却される。連続してデータを受信する場合は、再度フレーム受信要求を行う。
PDIC がデータを受信できない場合は、ビジー返却されるので、PDIC の利用者は再度フレーム受信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、受信データの参照が終了したら、受信フレーム解放（②）を要求する。
受信処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

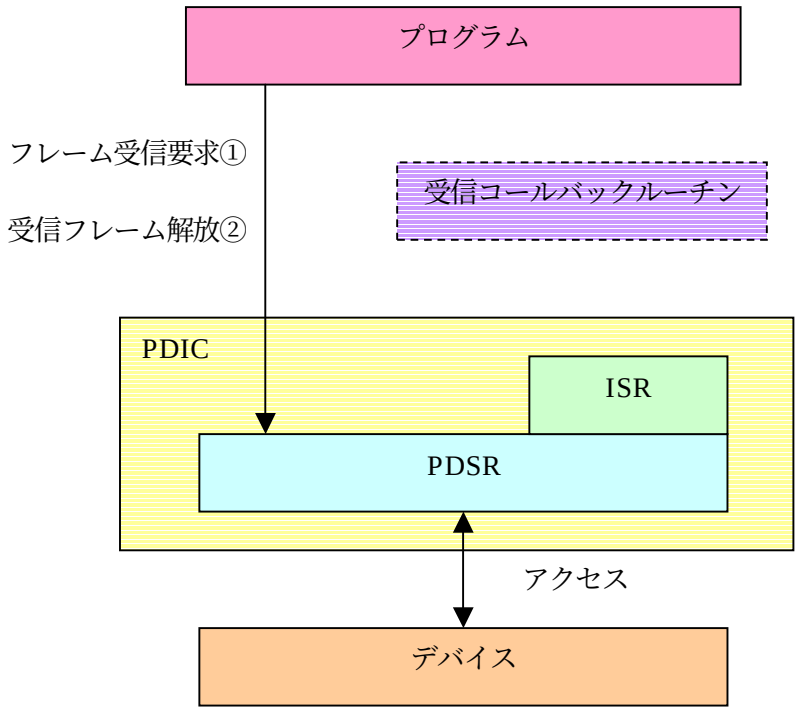

図 17　割込みを利用しないデータ受信

例：割込みを利用してデータ受信を行う場合

PDIC を利用するプログラムは、ポートに対しオープン要求を行った後、受信コールバックルーチンの許可を要求しておく。

受信割込みが発生すると、受信コールバックルーチンが呼び出され（①）、受信コールバックルーチンから事象通知（②）が行われる。

PDIC を利用するプログラムは、PDIC にフレーム受信要求（③）を行い、受信できた場合は、受信フレームバッファのアドレスを取得し、受信データを参照する。

PDIC を利用するプログラムは、受信データの参照が終了したら、受信フレーム解放通知（④）を要求する。

受信フレームバッファの取得ができたら、受信コールバックルーチンの呼出しを不許可にし、PDIC を利用するプログラムに対し、受信した旨を通知する。

受信コールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

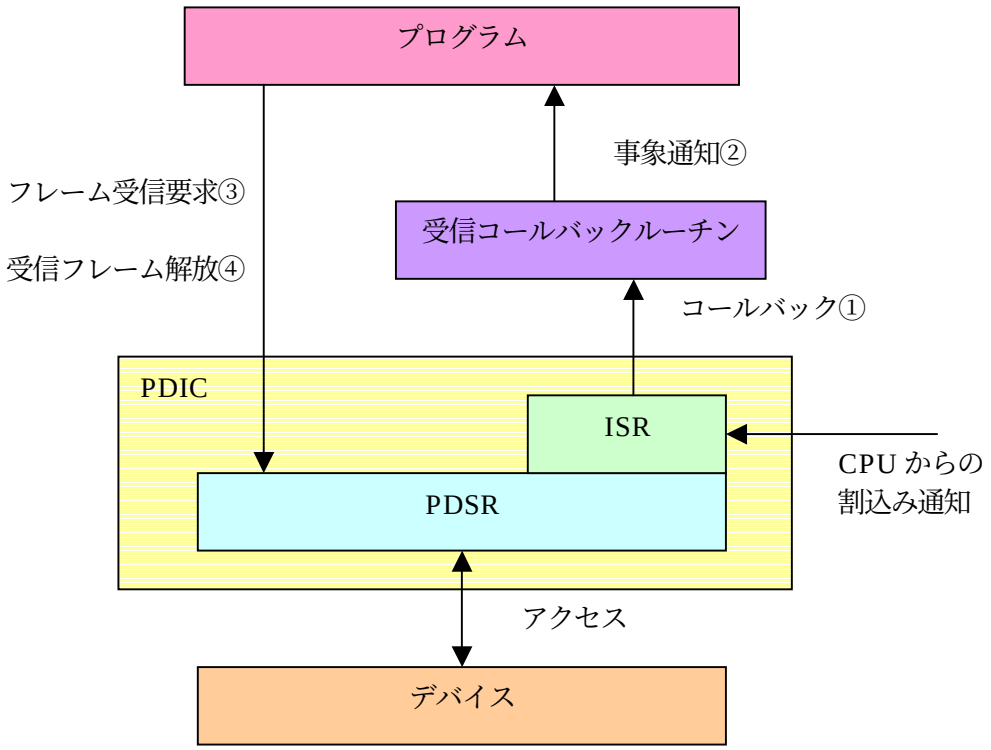

図 18　割込みを利用するデータ受信

４．３　コールバックルーチン

Ethernet コントローラの PDIC は､受信コールバックルーチンと送信コールバックルーチンの２つを定義する。受信コールバックルーチンは、Ethernet コントローラからの１つ以上の受信割込みにより呼出される。受信コールバックルーチンでは、受信の事象を PDIC を利用するプログラムに通知する。
送信コールバックルーチンは、送信フレームが空いた（送信が可能になった）場合に呼出される。具体的には、送信完了の割込み発生時に呼出される。

PDIC の実装者は、PDIC を利用するプログラムに対しフレーム送信処理または受信処理の終了事象を通知する枠を設けたコールバックルーチンを標準で提供する。ただし、送信または受信の完了の判定は、利用者で修正できるようにする。
PDIC の利用者は、コールバックルーチンを実装時に動作する環境に合わせて修正を行う。

ポートのオープン時にはコールバックルーチンの呼出しは不許可状態になっている。割込みを利用する PDIC の利用者は、ポートをオープンした後、受信コールバックルーチンの許可を要求する。

コールバックルーチンの引き数は、割込みが発生したポートの拡張情報と、事象の対象バッファのアドレスになる。
コールバックルーチンは、この２つの引き数を元に処理を行う。

４．４ PDIC 機能

Ethernet コントローラの PDIC(PDSR)には、以下の機能を規定する。

eth_ini_dev	：	初期化要求 デバイスの初期化
eth_opn_por	：	オープン要求 ポートの利用開始
eth_cls_por	：	クローズ要求 ポートの利用終了
eth_snd_frm	：	フレーム送信要求 フレーム送信
eth_get_sbf	：	送信フレーム取得要求 送信フレームバッファの取得
eth_rcv_frm	：	フレーム受信要求 フレーム受信
eth_rel_rdb	：	受信フレーム解放通知 受信フレームバッファの解放
eth_ena_cbr	：	コールバックルーチン呼出し許可 事象発生（割込み）時にコールバックルーチンの呼出し許可
eth_dis_cbr	：	コールバックルーチン呼出し不許可 事象発生（割込み）時にコールバックルーチンの呼出し不許可
eth_get_sts	：	ステータス取得要求 デバイススのテータス取得
eth_red_mac	：	MAC アドレスの読出し要求 MAC アドレスを読出す
eth_wrt_mac	：	MAC アドレスの書込み要求 MAC アドレスを書込む

Ethernet コントローラの PDIC の機能として規定する API 名称において、eth_xxx_yyy の“ eth_ ”の部分は、対象とするコントローラを示すユニークな名称とする。

以下に示す機能のパラメータにおいて、網掛けしたパラメータ構造は PDIC 実装者がコントローラに合わせて実装定義する。
PDIC 利用者は、実装定義のパラメータの内容を変更する。

Ethernet コントローラを利用する場合は、システム立ち上がり時に「初期化要求」でデバイスの初期化を行う。
Ethernet コントローラを利用しはじめる場合に、「オープン要求」を行い、コールバックルーチンの呼出し許可を行う。
Ethernet の伝送条件は、オープン要求で設定する。
PDIC 内の割込みサービスルーチン（ISR）の登録は、PDIC を利用する側が初期化要求後に行う。

（1）初期化要求（Ethernet　Initialize　Device）

【API】
ER　　eth_ini_dev（T_IINFO *info）

【パラメータ】
　　T_IINFO　　*info　　：　初期化パラメータ
　　デバイスを初期化するために必要な情報
　　T_IINFO　　実装定義
　　（MAC アドレスなど）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　Ethernet コントローラの初期化パラメータに従い初期化を行う。

【注意事項】
　　初期化処理中は、PDIC 内で CPU の割込みを禁止して行うことを原則とする。
　　要求時の CPU の割込み禁止／許可状態は保存する。
　　システムの起動時やデバイスのリセット時に呼び出すため、電源 ON 時の初期値に依存しないようにする。

（2）オープン要求（Ethernet　Open　Port）

【API】

ER　　eth_opn_por（INT　portno, T_OINFO　*info, VP_INT　exinf）

【パラメータ】

INT	portno	：	ポート番号（0～） 複数のポート（チャネル）をサポートするデバイスの場合、 それぞれを区別するための番号
T_OINFO	*info	：	オープンパラメータ T_OINFO　実装定義 （伝送条件、受信バッファディスクリプタの アドレス、ディスクリプタ数など）
VP_INT	exinf	：	拡張情報 コールバックルーチンへの引き数 内容は PDIC の利用者が利用できる

【リターンパラメータ】

E_OK　：　正常
負数　　：　エラーコード

【機能】

指定されるポート番号に相当するポートを指定伝送条件に従いデバイスの設定を行う。
オープン処理後、PDIC は指定される受信バッファディスクリプタを使い、受信処理を開始する。
送信／受信コールバックルーチンは不許可の状態とする。
正常に処理できた場合、対象ポートをオープン状態とし、通信に関する要求を受付ける状態とする。
PDIC からコールバックルーチンを呼び出す場合に、指定される拡張情報を引き数として渡す。

【注意事項】

初期化処理中は、PDIC 内で CPU の割込みを禁止して行うことを原則とする。
要求時の CPU の割込み禁止／許可状態は保存する。
システムの起動時やデバイスのリセット時に呼び出すため、電源 ON 時の初期値に依存しないようにする。
多重のオープン要求はエラーとする。

（3）クローズ要求（Ethernet　Close　Port）

【API】
ER　　eth_cls_por（INT　portno）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートの通信を停止する。
　　対象ポートの通信処理を停止し、デバイスの割込みを不許可にする。
　　対象ポートをクローズ状態とし、通信に関する要求を受付けない状態とする。
　　通信中のデータがある場合は、破棄する。

【注意事項】
　　処理中は、CPU の割込みを禁止して行うことを原則とする。
　　多重のクローズ要求はエラーとする。

（4）フレーム送信要求（Ethernet　Send　Frame）

【API】
ER　　eth_snd_frm（INT　portno, VP　frame, INT　size）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　VP　　frame　：　送信フレームバッファのアドレス
　　INT　　size　　：　送信サイズ（バイト数）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに対しフレーム内のデータを出力する。
　　対象ポートが出力できない状態の場合、ビジー返却する。

【注意事項】
　　送信フレームバッファをディスクリプタ形式とした場合、送信バッファは送信バッファの取得要求が行われた順のバッファが FIFO 順で指定されることを原則とする。この場合、バッファのアドレスは、チェックだけ行うことになる。
　　FIFO 順でない場合は、指定されたバッファのアドレスにある送信データの内容を送信処理する。

（5）送信フレーム取得要求（Ethernet　Get　Send　Frame　Buffer）

【API】

ER　　eth_get_sbf（INT　portno, VP　*p_frame）

【パラメータ】

INT　　portno　：　ポート番号（0～）
VP　　*p_frame：　送信フレームバッファのアドレスを格納するポインタ

【リターンパラメータ】

E_OK　：　正常
負数　　：　エラーコード

【機能】

送信フレームとして利用するバッファを取得する。
本要求は、Ethernet コントローラに送信バッファが用意されている場合、用意するものである。

【注意事項】

プログラム領域のバッファが指定できる Ethernet コントローラ用の PDIC でも、本要求を機能として実装してもよい。

（6）フレーム受信要求（Ethernet　Receive　Frame）

【API】
ER_UINT　　eth_rcv_frm（INT　portno, VP　*p_frame）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　VP　　*p_frame：　受信フレームバッファのアドレスを格納するポインタ

【リターンパラメータ】
　　正数　　：　受信データサイズ（バイト単位）
　　負数　　：　エラーコード
　　　　（ハードウェアエラー発生時はステータスを反映する）

【機能】
　　指定されるポート番号に相当するポートから受信フレームを取得する。
　　対象ポートに受信フレームができない状態の場合、ビジー返却する。

【注意事項】

（7）受信フレーム解放要求（Ethernet　Release　Receive　Frame　Buffer）

【API】
ER_UINT　　eth_rel_rbf（INT　portno, VP　frame）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　VP　　frame　：　解放する受信フレームのバッファのアドレス

【リターンパラメータ】

【機能】
　　指定されるポート番号に相当するポートに対し受信フレームを解放する。

【注意事項】

（8）コールバックルーチン許可要求（Ethernet　Enable　Call-Back　Routine）

【API】
ER　　eth_ena_cbr（INT　portno,　UINT　flag）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　UINT　flag　　：　許可する送信／受信コールバックルーチンを示すフラグ
　　　　送信コールバック許可　　：　ｘｘｘ
　　　　受信コールバック許可　　：　ｙｙｙ
　　　　これらのフラグは、OR で指定することができる

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに関する割込み処理を許可する。
　　具体的には、送信割込み、受信割込み、エラー割込みを受付けられるようにする（割込みを許可する）。

【注意事項】

（9）コールバックルーチン不許可要求（Ethernet　Disable　Call-Back　Routine）

【API】
ER　　eth_dis_cbr（INT　portno, UINT　flag）

【パラメータ】
　　INT　　portno　：　ポート番号（0～）
　　UINT　flag　　：　不許可にする送信／受信コールバックルーチンを示すフラグ
　　　　送信コールバック不許可　：　ｘｘｘ
　　　　受信コールバック不許可　：　ｙｙｙ
　　　　これらのフラグは、OR で指定することができる

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートに関する割込み処理を不許可にする。
　　具体的には、送信割込み、受信割込み、エラー割込みが発生しないようにする。

【注意事項】

（１０）ステータス取得（Ethernet　Get　Status）

【API】
ER_UINT　　eth_get_sts（INT　portno）

【パラメータ】
　　INT　　portno　：　ポート番号（０～）

【リターンパラメータ】
　　正数　　：　対象デバイスのステータス（実装定義）
　　負数　　：　エラーコード

【機能】
　　指定されるポート番号に相当するポートの情報を返却する。

【注意事項】

（１１）MAC アドレスの読出し要求（Ethernet　Read　MAC　Address）

【API】
ER　　eth_red_mac（INT　portno, VP　*p_mac）

【パラメータ】
　　VP　　*p_mac　：　MAC アドレスを格納するポインタ
　　INT　　portno　：　ポート番号（０～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　MAC アドレスを取得する。

【注意事項】
　　動作ハードウェア環境で MAC アドレスの格納場所が異なるため、関数の枠を用意しておいて、利用する側で定義してもらうようにする。
　　本機能は、オープン／クローズ状態に関わらず利用できる。

（１２）MAC アドレスの書込み要求（Ethernet　Write　MAC　Address）

【API】
ER　　eth_wrt_mac（INT　, VP　mac）

【パラメータ】
　　VP　　*mac　　：　MAC アドレスがあるアドレス
　　INT　　portno　：　ポート番号（０～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　MAC アドレスを書込む。

【注意事項】
　　動作ハードウェア環境で MAC アドレスの格納場所が異なるため、関数の枠を用意しておいて、利用する側で定義してもらうようにする。
　　本機能は、オープン／クローズ状態に関わらず利用できる。

５．ディスク関連コントローラ

ディスク関連のメディアは、数多くあり、またインタフェース規格も様々である。
それら個々に機能を規定するのは困難である。
ここでは、一般的な機能とパラメータについて規定し、それぞれの定義については実装で定義するものとする。

ディスク関連の PDIC は物理的なデバイスを制御し、ファイルシステムについては PDIC を利用する側で考慮するものとしている。
データが格納されているブロック（セクタ）を指定する番号は、ディスク関連コントローラによっては、ヘッドやセクタ、シリンダなどの物理情報を指定するものがある。また、別のコントローラでは、先頭のセクタを０番とする論理的なセクタ情報を扱うものもある。
本ガイドラインで規定する機能は、セクタを指定する番号を論理的に扱うものとし、物理的な情報に変換するのは、それぞれの PDIC で行うものとした。

ディスク関連のデバイスとインタフェースの例を以下に示す。

デバイス	インタフェース
FD	ATA
HD	ATAPI
RAM　Disk	USB
Flash	SCSI
CD	UFI
DVD	

５．１　ディスク関連 PDIC について

（1）　セクタとアドレス

ディスクには、一般にセクタと呼ばれるブロックがあり、そのアドレスを指定することで読出し／書込みを行う。
ブロックは連続アクセスが可能であることを前提としている。
アドレスは、先頭ブロックをアドレス０とする論理アドレスでの指定を基本とする。
ファイルシステムでは、ベースとするブロックとの相対アドレスを指定することがあるが、PDIC を利用する側では、それを論理アドレスに変換し要求を行う。
つまり、PDIC では、パーティションのような１つの物理ユニットを複数の論理ユニットのように扱うことはせず、物理的な制御を行うものとする。

（2）　デバイスとユニット

ディスク関連のデバイスには、複数のユニットをサポートできるものがある。
しかし、そのようなデバイスにおいては、各ユニットへのアクセスは排他的にアクセスする必要がある場合が多い。
そのようなデバイスの PDIC では、PDIC の利用者側で排他制御を行うようにする必要がある。PDIC の提供者は、その旨をドキュメントに記載するようにする。
PDIC が、１つのユニットに対しアクセスを行なっている最中に他のユニットに対しアクセスが行なわれた場合、ビジー返却を行う。

（3）　デバイスとインタフェース

ディスクに対するアクセスを考慮する際、直接ディスク・コントローラを制御する場合と ATAPI などのインタフェースを経由する場合とが考えられる。

PDIC で直接コントローラを制御する場合は、シリアル・デバイスの PDIC と同等の構成になる。
ATAPI などのインタフェースを経由する場合は、インタフェース・コントローラを制御する PDIC を介すような構成になる。
本ガイドラインでは、直接コントローラを制御する場合を想定し、PDIC の機能を規定する。

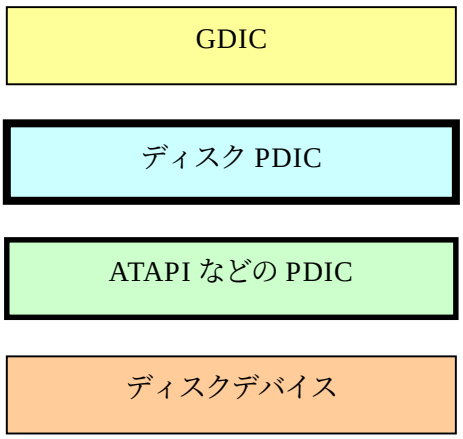

図 19　インタフェースを介す場合の PDIC 構成

（4）シーク

ディスクが回転物である場合、ヘッドを対象とするシリンダに移動するシークを行う必要がある。
シークは、読出し／書込み要求処理時に行うのではなく、別機能として用意する。
回転物を対象としないディスクの PDIC の場合は、何も処理せず正常を返却する。

（5）メディアの抜き差し

FD のように抜き差しが可能なメディアがある。
PDIC では物理的なコントローラ制御を行うものとし、メディアの抜き差しは PDIC の利用者が考慮することを前提とする。それにより、セクタのデータのブロッキング管理も PDIC の利用者側で行うものとする。
ただし、コントローラにメディアの抜き差しを検出できる機能を持つ場合、コールバックルーチンなどにより、利用者側にその旨を通知する機能を設けてもよい。

５．２　利用想定

ディスク・コントローラの PDIC をプログラムで利用する際に、割込みを利用して使う場合と、割込みを利用しないで使う場合とが考えられる。
本ガイドラインにおいて想定する利用方法を示す。
利用する想定としては､回転物を持つメディアを対象とし､ヘッドのシークを必要とするメディアを対象とした。シークを必要としないメディアの場合は、PDIC のシーク要求に対する処理は何もせず、正常を返却することする。

（1）データの読出し

ディスク・コントローラに対し PDIC を利用してデータを読出す例を示す。

例：割込みを利用しないでデータを読出す場合

PDIC を利用するプログラムで割込みを利用しないでデータを読出す場合は、CPU の割込みの受付けを禁止し、PDIC に対しシーク要求（①）を行う。その後、シーク動作の終了をシーク完了チェック（②）で確認する。正常にシーク動作が完了した時点で、データの読出し要求（③）を行う。データの読出しが正常に行われるのを読出し完了チェック（④）で確認する。
PDIC がデータを読出すことができない場合は、ビジー返却されるので、PDIC の利用者は再度データの読出し信要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。
PDIC を利用するプログラムは、データの読出し処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

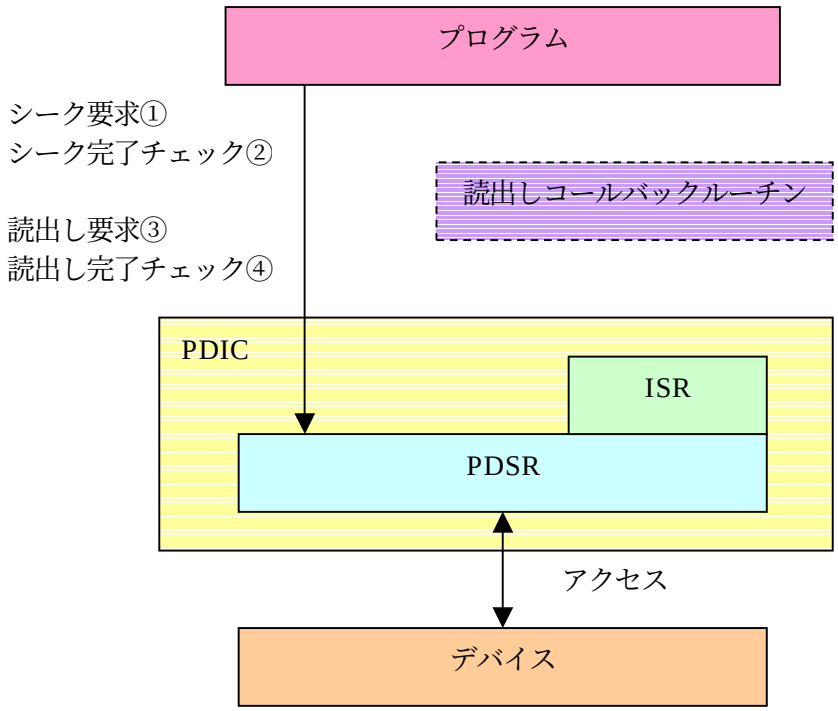

図 20　割込みを利用しないデータ読出し

例：割込みを利用してデータを読出す場合

PDIC を利用するプログラムで割込みを利用してデータを読出す場合は、コールバックルーチンの呼出しを許可し、最初にシーク要求（①）を行い、PDIC からの事象通知が待つ。シーク動作が完了するとコールバックルーチンが呼出され（②）、シーク動作の完了が通知（③）されるので、PDIC に対しデータの読出し要求（④）を行う。

その後、コールバックルーチンからの事象通知が持つ。データの読出しが正常に行われた場合、コールバックルーチンからデータの読出し完了の事象通知（⑤）を行う。

データの読出しが終了したら、読出しコールバックルーチンの呼出しを禁止する。

読出しコールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

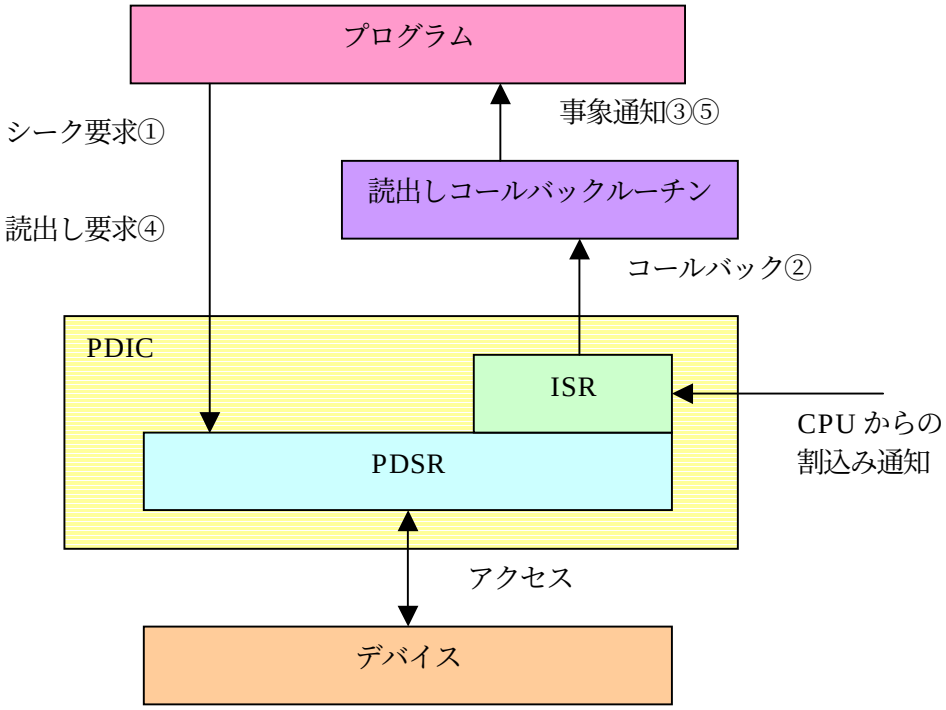

図 21　割込みを利用するデータ読出し１

また、PDIC の提供者が、PDIC の利用想定として割込みを利用することを前提とし、PDIC の利用者側のオーバヘッドをできるだけ削減しようとした場合、シーク動作終了時に、コールバックルーチンから PDIC に対し、読出し要求を発行する使い方もある。

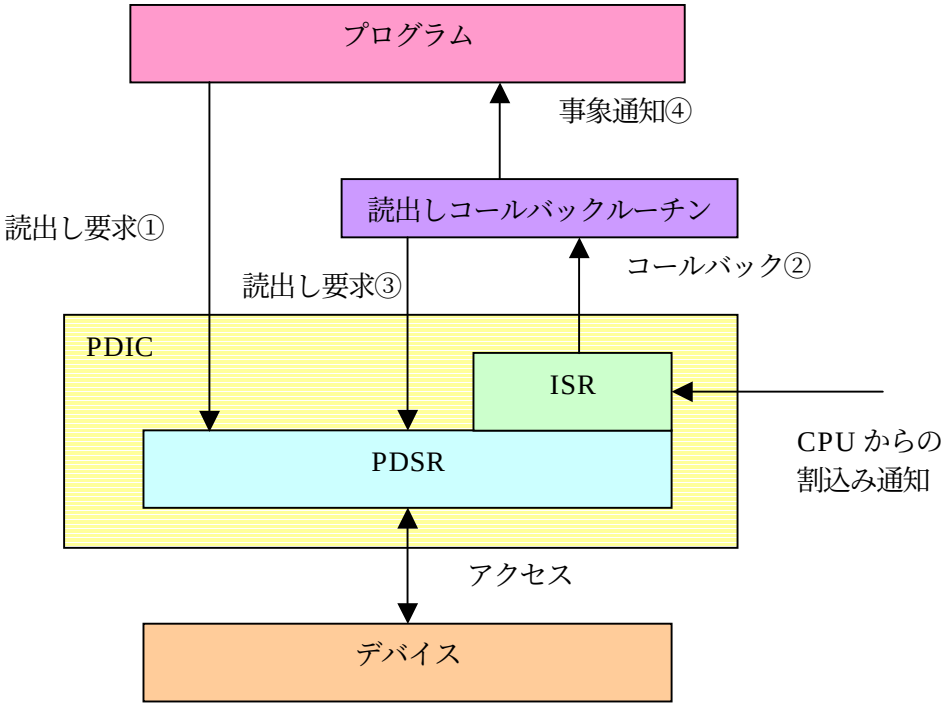

図 22　割込みを利用するデータ読出し 2

この場合、PDIC の利用者は、PDIC に対し読出し要求（①）を行い、事象通知を待つ。
PDIC ではシーク処理を行い、シーク処理完了時にコールバックルーチンが呼出され（②）、PDIC に対し読出し要求（③）を行う。読出し処理が完了した時点で、PDIC の利用者に事象通知（④）を行う。

（2）データの書込み

ディスク・コントローラに対し PDIC を利用してデータを書込む例を示す。

例：割込みを利用しないでデータを書込む場合

PDIC を利用するプログラムで割込みを利用しないでデータを書込む場合は、CPU の割込みの受付けを禁止し、PDIC に対しシーク要求（①）を行う。その後、シーク動作の終了をシーク完了チェック（②）で確認する。正常にシーク動作が完了した時点で、データの書込み要求（③）を行う。データの書込みが正常に行われるのを書込み完了チェック（④）で確認する。

PDIC がデータを書込めない場合は、ビジー返却されるので、PDIC の利用者は再度データの書込み要求を行う（リトライ）。PDIC からビジー返却された場合、どのくらいリトライするかは PDIC の利用者が決定する。

PDIC を利用するプログラムは、データの書込み処理が終了したら、CPU の割込みの禁止状態を処理開始時の状態に戻すものとする。

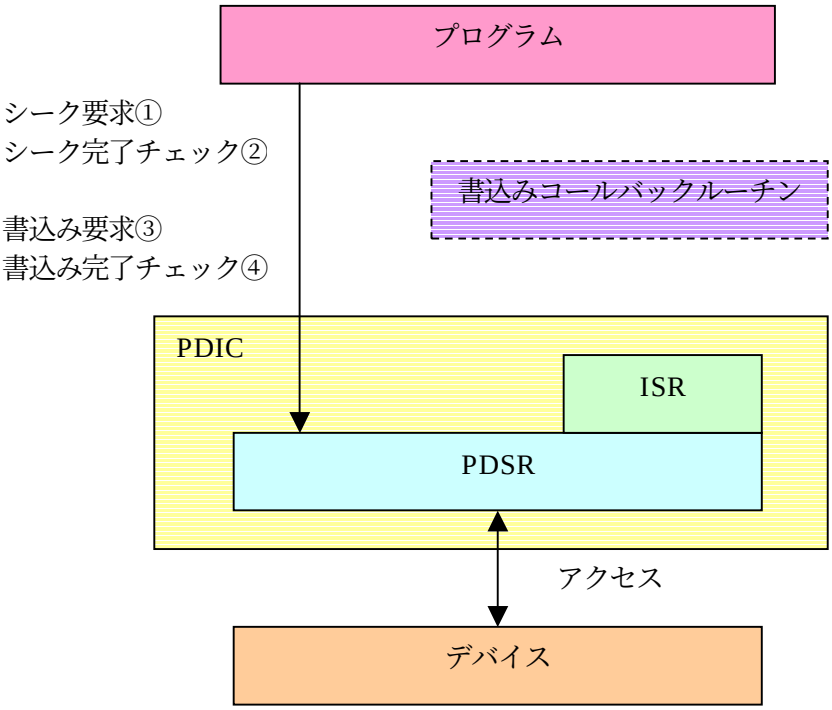

図 23　割込みを利用しないデータ書込み

例：割込みを利用してデータを書込む場合

PDIC を利用するプログラムで割込みを利用してデータを書込む場合は、コールバックルーチンの呼出しを許可し、最初にシーク要求（①）を行い、PDIC からの事象通知が待つ。シーク動作が完了するとコールバックルーチンが呼出され（②）、シーク動作の完了が通知される（③）ので、PDIC に対しデータの書込み要求（④）を行う。

その後、コールバックルーチンからの事象通知が持つ。データの読出しが正常に行われた場合、コールバックルーチンからデータの書込み完了の事象通知（⑤）を行う。

データの書込みが終了したら、書込みコールバックルーチンの呼出しを禁止する。

書込みコールバックルーチンから PDIC を利用する側へのプログラムの事象通知方法は、プログラムが動作する環境に合わせる。たとえば、OS を利用している場合は、OS が用意する機能を利用する。

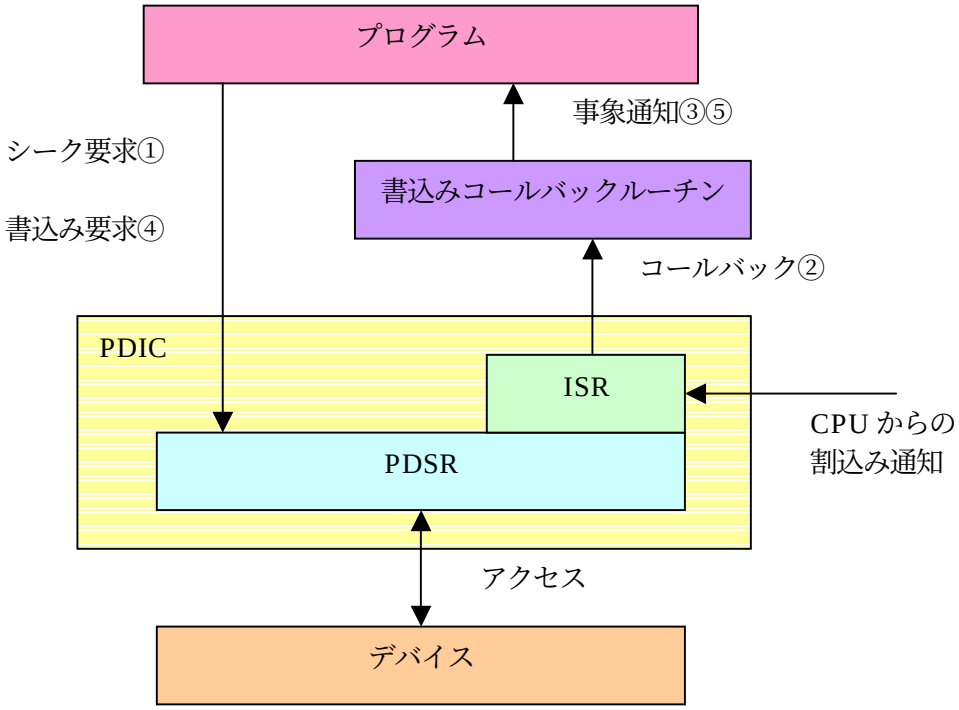

図 24　割込みを利用するデータ書込み１

また、PDIC の提供者が、PDIC の利用想定として割込みを利用することを前提とし、PDIC の利用者側のオーバヘッドをできるだけ削減しようとした場合、シーク動作終了時に、コールバックルーチンから PDIC に対し、書込み要求を発行する使い方もある。

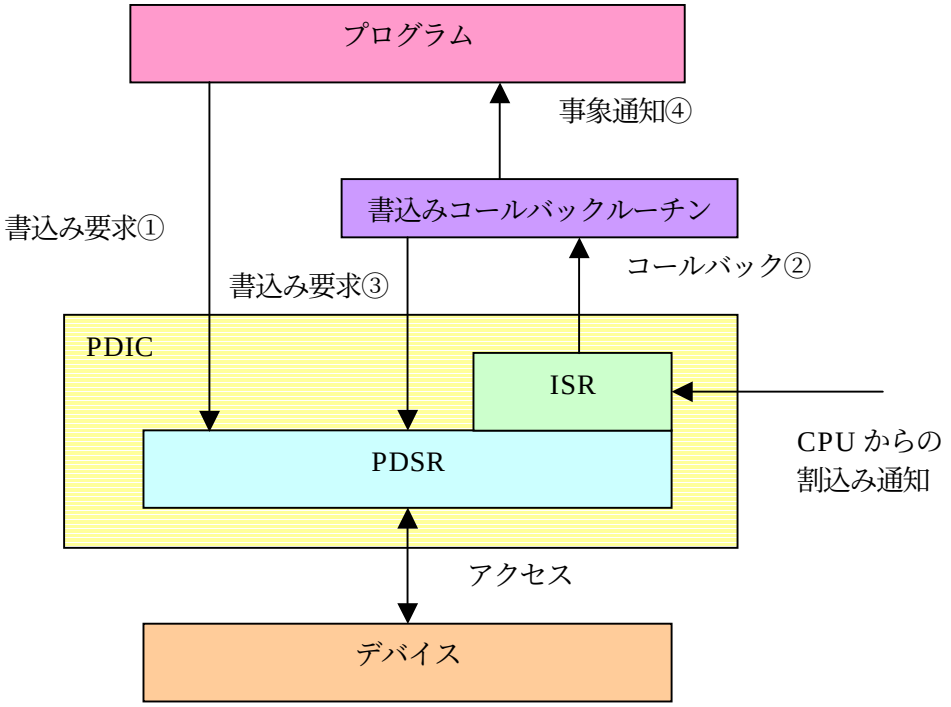

図 25　割込みを利用するデータ書込み 2

この場合、PDIC の利用者は、PDIC に対し書込み要求（①）を行い、事象通知を待つ。
PDIC ではシーク処理を行い、シーク処理完了後にコールバックルーチンが呼出され（②）、PDIC に対し書込み要求（③）を行う。書込み処理が完了した時点で、PDIC の利用者に事象通知（④）を行う。

５．３　コールバックルーチン

本ガイドラインにおいて、ディスク関連 PDIC は、読出しコールバックルーチンと書込みコールバックルーチンの２つを定義するものとしている。
読出しコールバックルーチンは、ディスクコントローラからの読出し割込みにより呼出される。読出しコールバックルーチンでは、読出し完了の事象を PDIC を利用するプログラムに通知する。
書込みコールバックルーチンは、ディスクコントローラからの書込み割込みにより呼出される。書込みコールバックルーチンでは、書込み完了の事象を PDIC を利用するプログラムに通知する。

PDIC の実装者は、PDIC に対し書込みまたは読出しの完了通知を行う枠を設けたコールバックルーチンを標準で提供する。。
PDIC の利用者は、コールバックルーチンを実装時に動作する環境に合わせて修正を行う。

ただし、ディスク関連 PDIC として、RAM や Flash ROM などディスクコントローラという形態を持たないものを対象とする場合がある。そのような形態では、割込み発生機構を持たないことがある。この場合でも、コールバックルーチンの関数枠だけを提供するようにする。
また、割込み機構によりコントローラからの割込み要因を特定することが困難であったり、処理にかかるオーバヘッドのために、コールバックルーチンを１つにして提供することは認められる。

ユニットのオープン時にはコールバックルーチンの呼出しは不許可状態になっている。
コールバックルーチンの引き数は、割込みが発生したユニット番号とユニットの拡張情報とする。
コールバックルーチンは、この２つの引き数を元に処理を行う。

５．４ PDIC 機能

ディスク関連のコントローラの PDIC(PDSR)には、以下の機能を用意する。

dsk_ini_dev	：	初期化要求 デバイスの初期化
dsk_opn_unt	：	オープン要求 ユニットの利用開始
dsk_cls_unt	：	クローズ要求 ユニットの利用終了
dsk_sek_unt	：	シーク要求 ヘッドのシーク処理
dsk_chk_sek	：	シーク完了チェック要求 シーク動作の完了チェック
dsk_wrt_blk	：	ブロック書込み要求 ブロック書込み
dsk_chk_wrt	：	書込みチェック要求 書込み動作の完了チェック
dsk_red_blk	：	ブロック読出し要求 ブロック読出し
dsk_chk_red	：	読出し完了チェック要求 読出し動作の完了チェック
dsk_ena_cbr	：	コールバックルーチンの呼出し許可要求 事象発生（割込み）時にコールバックルーチンの呼出し許可
dsk_dis_cbr	：	コールバックルーチンの呼出し不許可要求 事象発生（割込み）時にコールバックルーチンの呼出し不許可
dsk_get_sts	：	ステータス取得要求 デバイスステータス取得
dsk_ref_dsk	：	ディスク情報取得要求 ディスクの情報取得

ディスク関連コントローラの PDIC の機能として規定する API 名称において、dsk_xxx_yyy の“ dsk_ ”の部分は、対象とするコントローラを示すユニークな名称とする。

以下に示す機能のパラメータにおいて、網掛けしたパラメータ構造は PDIC 実装者がコントローラに合わせて実装定義する。
PDIC 利用者は、実装定義のパラメータの内容を変更する。

ディスク関連コントローラを利用する場合は、システム立ち上がり時に「初期化要求」でデバイスの初期化を行う。
ディスク関連コントローラを利用しはじめる場合に、「オープン要求」を行い、コールバックルーチンの呼出し許可を行う。
PDIC 内の割込みルーチン（ISR）の登録は、PDIC を利用する側が初期化要求後に行う。

（1）初期化要求（Disk　Initialize　Device）

【API】

ER　　dsk_ini_dev（T_IINFO *info）

【パラメータ】

T_IINFO　　*info　　：　初期化パラメータ

デバイスを初期化するために必要な情報

T_IINFO　　実装依存

【リターンパラメータ】

E_OK　：　正常

負数　　：　エラーコード

【機能】

ディスク関連コントローラの初期化パラメータに従い初期化を行う。

【注意事項】

初期化処理中は、割込みを禁止して行うことを原則とする。

システムの起動時やデバイスのリセット時に呼び出すため、電源 ON 時の初期値に依存しないようにする。

（2）オープン要求（Disk　Open　Unit）

【API】

ER　　dsk_opn_unt（INT　unitno, T_OINFO　*info, VP_INT　exinf）

【パラメータ】

INT	unitno	：	ユニット番号（0～） 複数のユニットをサポートするデバイスの場合、それぞれを区別するための番号
T_OINFO	*info	：	ディスクパラメータ T_OINFO　　実装依存
VP_INT	exinf	：	拡張情報 コールバックルーチンへの引き数 内容は実装依存

【リターンパラメータ】

E_OK　：　正常
負数　　：　エラーコード

【機能】

指定されるユニット番号に相当するユニットを指定ディスクパラメータに従いデバイスの設定を行う。
設定が正常に完了したら、対象ディスクの存在を確認する。
正常に処理できた場合、対象ユニットをオープン状態とし、アクセスに関する要求を受付ける状態とする。
PDIC からコールバックルーチンを呼び出す場合に、指定される拡張情報を引き数として渡す。

【注意事項】

処理中は、割込みを禁止して行うことを原則とする。
多重のオープン要求はエラーとする。

（3）クローズ要求（Disk　Close　Unit）

【API】
ER　　dsk_cls_unt（INT　unitno）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるユニット番号に相当するユニットへのアクセスを停止する。
　　対象ユニットに関する割込みを不許可にする。
　　対象ユニットをクローズ状態とし、要求を受付けない状態とする。

【注意事項】
　　処理中は、割込みを禁止して行うことを原則とする。
　　多重のクローズ要求はエラーとする。

（4）シーク要求（Seek）

【API】
ER_UNIT　　dsk_sek_unt（INT　unitno, UW　laddr）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）
　　UW　　laddr　：　シークするディスク論理アドレス

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　ディスクのヘッドを指定ディスク論理アドレスにシークする。

【注意事項】
　　本機能を持たないディスクをサポートする PDIC では、何も処理せず正常を返却する。

（5）シーク完了チェック（Seek　Finish　Check）

【API】
ER_UNIT　　dsk_chk_sek（INT　unitno）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　ディスクのヘッドのシーク処理が完了したことを確認する。
　　完了している場合は正常を返却し、完了していない場合はビジーを返却する。

【注意事項】
　　本機能を持たないディスクをサポートする PDIC では、何も処理せず正常を返却する。

（6）ブロック書込み要求（Disk　Write　Blocks）

【API】
ER_UNIT　　dsk_wrt_blk（INT　unitno, UW　laddr, VP　blk, UINT　num）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）
　　UW　　laddr　：　ブロックを書込むディスク論理アドレス
　　VP　　blk　　：　書込むブロックのベースアドレス
　　UINT　num　　：　書込むブロック数

【リターンパラメータ】
　　正数　　：　正常に書込めたブロック数
　　負数　　：　エラーコード
　　　　（ハードウェアエラー発生時はステータスを反映する）

【機能】
　　指定されるユニット番号に相当するユニットに対しブロック内のデータを書込む。
　　対象ユニットが書込めない状態の場合、ビジー返却する。

【注意事項】

（7）書込みチェック要求（Write　Finish　Check）

【API】
ER_UNIT　　dsk_chk_wrt（INT　unitno）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるユニット番号に対するデータを書込み処理が完了したことを確認する。
　　完了している場合は正常を返却し、完了していない場合はビジーを返却する。

【注意事項】

（8）ブロック読出し要求（Disk　Read　Blocks）

【API】
ER_UINT　　dsk_red_blk（INT　unitno, UW　laddr, VP　blk, UW　num）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）
　　UW　　laddr　：　読出すブロックのディスク論理アドレス
　　VP　　blk　　：　ブロックを読出すバッファのアドレス
　　UW　　num　　：　読出しブロック数

【リターンパラメータ】
　　正数　　：　正常に読出せたブロック数
　　負数　　：　エラーコード
　　　　（ハードウェアエラー発生時はステータスを反映する）

【機能】
　　指定されるユニット番号に相当するユニットからブロックを読出す。
　　対象ユニットからブロックを読出すことができない状態の場合、ビジー返却する。

【注意事項】

（9）読出しチェック要求（Read　Finish　Check）

【API】
ER_UNIT　　dsk_chk_red（INT　unitno）

【パラメータ】
　　INT　　unitno　：　ユニット番号（0～）

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるユニット番号に対するデータを読出し処理が完了したことを確認する。
　　完了している場合は正常を返却し、完了していない場合はビジーを返却する。

【注意事項】

（１０）コールバックルーチン許可要求（Disk　Enable　Call-Back　Routine）

【API】

ER　　dsk_ena_cbr（INT　unitno, UINT　flag）

【パラメータ】

INT　　unitno　：　ユニット番号（０～）
UINT　flag　　：　許可する書込み／読出しコールバックルーチンを示すフラグ
書込みコールバック許可　　：　ｘｘｘ
読出しコールバック許可　　：　ｙｙｙ
これらのフラグは、OR で指定することができる

【リターンパラメータ】

E_OK　：　正常
負数　　：　エラーコード

【機能】

指定されるユニット番号に相当するユニットに関する指定処理のコールバックの呼出しを許可する。

【注意事項】

割込み要因マスクが個別にできない場合は、どちらかが指定されていれば両方の割込みのマスクを解除する。
１つのユニットに対する割込み許可要求が、デバイスがサポートするユニット全体の割込み許可になってもよい。
割込みを使用しない PDIC でも、機能としては用意しておくことを推奨する。

（１１）コールバックルーチン不許可要求（Disk　Disable　Call-Back　Routine）

【API】

ER　　dsk_dis_cbr（INT　unitno, UINT　flag）

【パラメータ】

INT　　unitno　：　ユニット番号（０～）
UINT　flag　　：　不許可にする書込み／読出しコールバックルーチンを示すフラグ
書込みコールバック不許可　：　ｘｘｘ
読出しコールバック不許可　：　ｙｙｙ
これらのフラグは、OR で指定することができる

【リターンパラメータ】

E_OK　：　正常
負数　　：　エラーコード

【機能】

指定されるユニット番号に相当するユニットに関する割込み処理を許可する。

【注意事項】

割込み要因マスクが個別にできない場合は、どちらかが指定されていれば両方の割込みのマスクをする。
１つのユニットに対する割込み許可要求が、デバイスがサポートするユニット全体の割込み不許可になってもよい。
割込みを使用しない PDIC でも、機能としては用意しておくことを推奨する。

（１２）ステータス取得（Disk　Get　Status）

【API】
ER_UINT　　dsk_get_sts（INT　unitno）

【パラメータ】
　　INT　　unitno　：　ユニット番号（０～）

【リターンパラメータ】
　　正数　　：　対象デバイスのステータス（実装定義）
　　負数　　：　エラーコード

【機能】
　　指定されるユニット番号に相当するユニットの情報を返却する。

【注意事項】

（１３）ディスク情報取得（Disk　Refer　Disk　Information）

【API】
ER_UINT　　dsk_ref_dsk（INT　unitno, T_DINFO　*info）

【パラメータ】
　　INT　　unitno　：　ユニット番号（０～）
　　T_DINFO　　*info　：　ディスク情報を格納するパケットのアドレス
　　パケットの内容については実装定義
　　（ブロックサイズ、シリンダ数、セクタ数、ヘッド数）
　　（種別（EJECT あり／なし、RW／RO など））

【リターンパラメータ】
　　E_OK　：　正常
　　負数　　：　エラーコード

【機能】
　　指定されるユニット番号に相当するユニットの物理情報を返却する。

【注意事項】
　　メディアのライトプロテクトなどについては、本情報により取得する。

6．リターンパラメータ

本章では、PDIC のリターンパラメータの一覧を示す。
エラーコードのシンボルは、μITRON4.0 仕様に準拠した。
以下に示すリターンパラメータ以外については、実装で定義する。

E_OK	：	正常
E_PAR	：	パラメータエラー
E_OBJ	：	状態エラー（ビジーエラー）
E_ILUSE	：	不正使用（二重オープン他）

7．SIL

本章では、システムインタフェースレイヤ（SIL）について説明する。
SIL についての詳細の説明は、『デバイスドライバ設計ガイドライン』を参照していただきたい。

7．1　はじめに

SILは、PDICの移植性を上げるためにプロセッサの機能を抽象化する。
SILが抽象化するものは、具体的には以下の項目である。

- プロセッサのエンディアン
- プロセッサの速度
- 割込み禁止の方法
- プロセッサがデバイスとデータ入出力する方法

SILは、PDIC及びシステムが動作するハードウェアに依存する。SILの提供者は、操作するハードウェアに合わせてSILを実装する。
PDICは、SILが提供する機能を利用し、できる限り動作するハードウェアの環境に依存しないように実装することが望ましい。

7．2　SIL の機能

SILは次のような機能を備える。

(a)　デバイスとのデータ入出力
(b)　微小時間の遅延
(c)　割込みロック状態の制御
(d)　割込みサービスルーチンの管理

これらの機能は、プロセッサのエンディアン、割込みの禁止や許可、実行速度などのプロセッサの能力、割込みコントローラの機能といったものを抽象化している。

７．２．１　デバイスとのデータ入出力

（１）　I/O 空間とメモリ空間を区別する
ソフトウェアから見ると、プロセッサが備えているデータ入出力の方法は大きく次の2つに分類できる。

(a)　I/O空間での入出力(ただし、I/O空間を備えないプロセッサも存在する)
(b)　メモリ空間での入出力

プロセッサがデバイスと入出力するとき、同じデータサイズであっても、I/O空間とメモリ空間では異なる入出力命令を用いる。
このような空間の違いに対応できるように、SILはI/O空間・メモリ空間それぞれに接続したデバイスと入出力する次のようなサービスコールを定義している。

8ビットのデータを入力
　I/O空間上のI/Oポートから　VB data = sil_reb_iop(VP iop);
　メモリ空間上のメモリから　VB data = sil_reb_mem(VP mem);
8ビットのデータを出力
　I/O空間上のI/Oポートへ　void sil_wrb_iop(VP iop, VB data);
　メモリ空間上のメモリへ　void sil_wrb_mem(VP mem, VB data);

（２）　データサイズを区別する
通常のメモリと異なり、デバイスは、特定のデータサイズで入出力することをプロセッサに要求する場合がある。例えば、あるデバイスレジスタでは、8ビットサイズで読み取りを、16ビットサイズで出力をおこなう。
このようなデータサイズの違いに対応できるように、SILは特定のデータサイズでデバイスと入出力する次のようなサービスコールを定義している。

メモリ空間上の指定したアドレスからデータを入力
　8ビットサイズ　VB data = sil_reb_mem(VP mem);
　16ビットサイズ　VH data = sil_reh_mem(VP mem);
　32ビットサイズ　VW data = sil_rew_mem(VP mem);

（３）　エンディアンを区別する
SILは、プロセッサのエンディアンにかかわらず、デバイスの指定するエンディアンで入出力する次のようなインタフェースを定義している。

メモリ空間上の指定したアドレスから16ビットのデータを入力
　自然なエンディアン　VH data = sil_reh_mem(VP mem);
　リトルエンディアン　VH data = sil_reh_lem(VP mem);
　ビッグエンディアン　VH data = sil_reh_bem(VP mem);
メモリ空間上の指定したアドレスへ16ビットのデータを出力
　自然なエンディアン　void sil_wrh_mem(VP mem, VH data);
　リトルエンディアン　void sil_wrh_lem(VP mem, VH data);
　ビッグエンディアン　void sil_wrh_bem(VP mem, VH data);

自然なエンディアンでは、エンディアンの変換を考慮しない。つまり、SILが動作するエンディアン（通常システムが動作するエンディアンと同じ）で入出力を行う。

７．２．２　微小時間の遅延

デバイスは入出力と入出力との間に一定時間以上の間隔を要求することがある。
SILは微小時間を待つために、タイマカウンタをポーリングしながら、あるいは単にプロセッサのサイクルを空費するようなビジーウェイトとして実装される、次のサービスコールを定義している。

微小時間の遅延	void sil_dly_nse(UINT dlytim);

sil_dly_nseのパラメータdlytimには1nsから100μs程度の微小時間が指定されるものと考え、dlytimの単位をnsとしている。時間の精度は、SILの実装によることになる。SILの実装者は、サービスコールの呼出し／復帰の時間を考慮して実装する必要がある。

７．２．３　割込みロック状態の制御

PDICでデバイスを制御する場合、割込みロック状態（割込み禁止状態）で制御しなければならない場合がある。そのためSILは、割込みをロック状態にする機能と、ロック解除状態にする機能を提供する。
割込みロック状態はNMI以外のすべての割込みを禁止する。割込みロック状態では、割込みハンドラは起動されずタスクディスパッチも起らない。
SILは、割込みロック状態を制御する次のサービスコールを定義する。

割込みロック状態への移行	
タスクコンテキストから	void SIL_LOC_INT(void);
非タスクコンテキストから	void ISIL_LOC_INT(void);
割込みロック解除状態への移行	
タスクコンテキストから	void SIL_UNL_INT(void);
非タスクコンテキストから	void ISIL_UNL_INT(void);

割込みロック解除状態へ移行するサービスコールは、必ずしも割込みロック解除状態になるものではない。詳細は、APIの要求を参照する。
割込みロック状態の制御に関するサービスコールの名称が大文字であるのは、マクロで実装されることがあることを示している。
タスクコンテキストと非タスクコンテキストからの要求を分けて記述するようにした。タスクコンテキスト用のサービスコールを非タスクコンテキストで要求した場合（またはその逆）の動作は未定義とする。
SILは、動作するシステムがOSを利用するかどうかには依存しないことを基本とするが、SILの実装者がOSが提供する機能を利用する場合を考慮し、PDICで動作するコンテキストにより呼び分けるようにする。
SILをOSに依存しないように実装する場合は、タスクコンテキストからの要求のサービスコール名称をデフォルトの名称とする。

７．３ SIL のＣ言語 API

７．３．１ I/0 ポートからのデータ入力

【API】

```
        VB data         sil_reb_iop(VP iop);
        VH data         sil_reh_iop(VP iop);
        VW data         sil_rew_iop(VP iop);
(リトルエンディアン)
        VH data         sil_reh_lep(VP iop);
        VW data         sil_rew_lep(VP iop);
(ビッグエンディアン)
        VH data         sil_reh_bep(VP iop);
        VW data         sil_rew_bep(VP iop);
```

【パラメータ】

VP	iop	データを入力するアドレス

【リターンパラメータ】

VB/VH/VW	data	入力したデータ

【機能】

I/0空間上の指定したアドレスからデータを入力する。I/0ポートから入力するデータのサイズによりサービスコールが異なる。

また、I/0ポートに置かれているデータの並び順(リトルエンディアン・ビッグエンディアン)ごとにサービスコールが異なる。エンディアンを指定しなければプロセッサにとって自然な並び順と見なす。

【補足説明】

これらのサービスコールはデバイスとプロセッサとの間のバス幅ごとに設けてあるのではない。入力するデータのサイズごとに異なっている。

I/0空間をサポートしないプロセッサでは、これらI/0ポートからデータを入力するサービスコールを、メモリからデータを入力するように実装してよい。そのため、I/0空間をサポートしないプロセッサでは、I/0ポートからのデータ入力サービスコールとメモリからのデータ入力サービスコールの実装に違いがないことがある。

７．３．２　I/0 ポートへのデータ出力

【API】

```
        void    sil_wrb_iop(VP iop, VB data);
        void    sil_wrh_iop(VP iop, VH data);
        void    sil_wrw_iop(VP iop, VW data);
(リトルエンディアン)
        void    sil_wrh_lep(VP iop, VH data);
        void    sil_wrw_lep(VP iop, VW data);
(ビッグエンディアン)
        void    sil_wrh_bep(VP iop, VH data);
        void    sil_wrw_bep(VP iop, VW data);
```

【パラメータ】

VP	iop	データを出力するアドレス
VB/VH/VW	data	出力するデータ

【機能】

I/0空間上の指定したアドレスへデータを出力する。I/0ポートへ出力するデータのサイズによりサービスコールが異なる。

また、データをI/0ポートに置くときの並び順(リトルエンディアン・ビッグエンディアン)ごとにサービスコールが異なる。エンディアンを指定しなければプロセッサにとって自然な並び順となる。

【補足説明】

これらのサービスコールはデバイスとプロセッサとの間のバス幅ごとに設けてあるのではない。出力するデータのサイズごとに異なっている。

I/0空間をサポートしないプロセッサでは、これらI/0ポートへデータ出力するサービスコールを、メモリへデータ出力するように実装してよい。そのため、I/0空間をサポートしないプロセッサでは、I/0ポートへのデータ出力サービスコールとメモリへのデータ出力サービスコールの実装に違いがないことがある。

７．３．３　メモリからのデータ入力

【API】

```
        VB data         sil_reb_mem(VP mem);
        VH data         sil_reh_mem(VP mem);
        VW data         sil_rew_mem(VP mem);
(リトルエンディアン)
        VH data         sil_reh_lem(VP mem);
        VW data         sil_rew_lem(VP mem);
(ビッグエンディアン)
        VH data         sil_reh_bem(VP mem);
        VW data         sil_rew_bem(VP mem);
```

【パラメータ】

VP	mem	データを入力するアドレス

【リターンパラメータ】

VB/VH/VW	data	入力したデータ

【機能】

メモリ空間上の指定したアドレスからデータを入力する。I/O空間の場合と同様に、メモリから入力したデータのサイズによりサービスコールが異なる。
また、メモリに置かれているデータの並び順(リトルエンディアン・ビッグエンディアン)ごとにサービスコールが異なる。エンディアンを指定しなければプロセッサにとって自然な並び順とみなす。

【補足説明】

これらのサービスコールはデバイスとプロセッサとの間のバス幅ごとに設けてあるのではない。データのサイズごとに、プロセッサのロード命令に対応するサービスコールを設けてある。
32ビットのロード命令がないプロセッサでは、32ビットサイズのデータを読み取るサービスコールを、16ビットのロード命令を複数回繰り返すように実装してよい。

７．３．４　メモリへのデータ出力

【API】

```
        void    sil_wrb_mem(VP mem, VB data);
        void    sil_wrh_mem(VP mem, VH data);
        void    sil_wrw_mem(VP mem, VW data);
(リトルエンディアン)
        void    sil_wrh_lem(VP mem, VH data);
        void    sil_wrw_lem(VP mem, VW data);
(ビッグエンディアン)
        void    sil_wrh_bem(VP mem, VH data);
        void    sil_wrw_bem(VP mem, VW data);
```

【パラメータ】

VP	mem	データを出力するアドレス
VB/VH/VW	data	出力するデータ

【機能】

メモリ空間上の指定したアドレスへデータを出力する。I/0空間の場合と同様に、メモリへ出力するデータのサイズによりサービスコールが異なる。
また、データをメモリに置くときの並び順(リトルエンディアン・ビッグエンディアン)ごとにサービスコールが異なる。エンディアンを指定しなければプロセッサにとって自然な並び順となる。

【補足説明】

これらのサービスコールはデバイスとプロセッサとの間のバス幅ごとに設けてあるのではない。データのサイズごとに、プロセッサのストア命令に対応するサービスコールを設けてある。
32ビットのストア命令がないプロセッサでは、32ビットサイズのデータを出力するサービスコールを、16ビットのストア命令を複数回繰り返すように実装してよい。

７．３．５　微小時間の遅延

【API】

```
void    sil_dly_nse(UINT dlytim);
```

【パラメータ】

UINT　　dlytim　遅延時間(1nsを単位とする相対時間)

【機能】

デバイスの仕様により一定時間デバイスを操作できないときに、dlytimで指定される時間(ns)の間、ビジーウェイトしてプロセッサの処理の進行を遅延させる。この遅延は解除できない。sil_dly_nseから復帰すると、少なくともdlytimで指定された以上の時間が経過した後であることが保証される。

【補足説明】

sil_dly_nseは割込みロック状態を変更しないので、割込みロック解除状態で呼び出すと、プリエンプションなどのために指定されたdlytim時間より長く遅延する場合がある。一定時間以上の時間経過を防ぐには、sil_dly_nseを呼ぶ前に割込みロック状態にしなければならない。
長時間の遅延のためにsil_dly_nseを用いてはならない。
ここで指定する時間の精度は、プロセッサの性能やSILの実装により異なる場合がある。

７．３．６　割込みロック状態への移行

【API】

```
void    SIL_LOC_INT(void);
void    ISIL_LOC_INT(void);
```

【機能】

割込みロック状態へ移行する。すでに割込みロック状態であった場合、状態の変化はない。
非タスクコンテキストからは、SIL_LOC_INTにかえてISIL_LOC_INTを呼ぶ。

【補足説明】

SIL_UNL_INTとISIL_UNL_INTを呼び間違えたときの動作は未定義になる。

７．３．７割込みロック解除状態への移行

【API】

```
void    SIL_UNL_INT(void);
void    ISIL_UNL_INT(void);
```

【機能】

SIL_UNL_INT(ISIL_UNL_INT)は、割込ロック解除状態に必ず移行するサービスコールではない。
対応するSIL_LOC_INT(ISIL_LOC_INT)が実行される前の状態(割込みロック状態または割込ロック解除状態)に復帰する。
非タスクコンテキストからは、SIL_UNL_INTにかえてISIL_UNL_INTを呼ぶ。

[補足説明]

SIL_UNL_INTとISIL_UNL_INTを呼び間違えたときの動作は未定義になる。