OS9 DEVICE DRIVER IN C

K.Tamezane,H.Sakaki,M.Kodera,H.Yoshikawa,
S.Suzuki,K.Yanagida,A.Mizuno,T.Hori and H.Yokomizo.
SPring-8 Project Team, Japan Atomic Energy Research Institute.
2-4 Tokai-mura, Ibaraki-ken, 319-11

ABSTRACT

The injector accelerators of SPring-8, 1GeV linac and 8GeV synchrotron, have distributed control systems using network and many VME computers. These are having many points must be modified after the facility is completed, and the system should be adapted to many requirements of users in a short time and low cost. For this purpose, the concept of object oriented programming(OOP) is inevitable to build up the software. We rewrite device drivers in C to get flexibility, transparency and ease of maintenance. Flexibility and transparency of device access contribute to good achievement of OOP control system. In this paper, the know-how the making device drivers in C and how to combine the C module with the assembler module are discribed.

C言語によるＯＳ９・デバイス・ドライバの製作

１．はじめに

現在建設が進められている大型放射光施設（ＳＰｒｉｎｇ－８）の入射系制御の構築は、線型加速器の電子入射部をテストベンチとして進められている。ネットワークを利用した分散制御系を構築するために、各デバイスの特殊性を隠ぺいしてアプリケーションからの透過性を高めなければならない。そのためにはデバイス・アクセスを行うファイルマネージャやデバイスドライバに必要な機能を、オーバヘッドを大きくしないで実現する必要がある。しかし、ファイルマネージャやデバイスドライバは一般にアセンブラで記述しなければならず、それが可読性や機能付加のしやすさを損ない、維持管理の困難さの原因になっている。

この問題点を解決するために我々はデバイスドライバをＣ言語で開発した。カーネルとのやりとりに必要な枠組みのみをアセンブラで記述し、デバイスドライバとして実現される機能はＣ言語で記述された関数を呼び出す方式を用いた。ここではアセンブラで記述したオブジェクトファイルとＣ言語で記述したオブジェクトファイルの違いや、Ｃ言語ドライバをつくるときのノウハウとなるポイントを示す。

２．ＳＰｒｉｎｇ－８線型加速器の制御系

大型放射光施設の入射系１ＧｅＶ線型加速器は、複数のパルス幅モード、電子／陽電子モードとそれに対応する小電流／大電流モード、入射モードと個別利用運転モード等多くの運転モードを持つ。従って、初期の調整運転終了後も単なる入射運転だけでなく、様々な運転モードに対応できるように、先進的な制御系を構築する必要がある。そして性能向上のためにマシンと制御系の改良が継続的に行われることが予想される。したがって制御系は、単にビームを出すだけのリジッドな機能を満足するだけではなく、変更要請に対して最小限の作業とコストで対応できなければならない。これらに対応するためには、オブジェクト指向プログラミング（ＯＯＰ）による制御系ソフトウェアの構築が不可欠である。ハードウェアの構成もＯＯＰを適用し易いように配慮されている[1]。ＯＯＰの概念そのものは新しいものではないが、入出力を扱い実時間性を要求される制御システムへの応用例は日本ではまだ見られない。これは一般のプラント系の制御システムというものが、実績（すなわち信頼性）を重要視し、新規性を排除する傾向にあること、変更の容易性を必要とする事があまりない（そのまま償却するまで運転できれば良い）ことなどにより、構造化プログラミングの域を脱しておらず、計算機利用技術の進歩に遅れをとっているといえる。またＯＯＰの特質として、既存のシステムに対する部分的な置き換えや共存が困難であることも原因している。これに対し、加速器の制御というものは規模が大きくなっても基本的な最低限必要な機能はセット＆トライの単純なものである。これがコライダのマイクロバンチや第３世代放射光のマイクロビームを実現しようとしたとき、極端に高速で高精度の制御が要求されるようになる。現実にはその高速高精度の制御を目指して改造改良を継続的に行っていくことになり、変更の容易さはたいへん重要なポイントになる。ここが一般のプラント制御と大きく異なるところで、ＯＯＰのメリットが非常に有効に生きてくるところになる。

ＳＰｒｉｎｇ－８入射系制御では、多岐に渡る利用形態に軽快に対応できるように、ＯＯＰを取り入れデジタル化を進めた加速器制御構築を行う。

３-1．入出力インターフェーシング

アプリケーションレベルのソフトウェア構築に適用できるＯＯＰのＣＡＳＥツールやＧＵＩ構築ツールは数多くのものが製品化されているが、信号入出力を伴うシステムの構築ツールはまだ種類も少なく高価である。ハードウェアと密接な関連を持ち、詳細な情報を必要とするこの部分のソフトウェアは、できれば充分にチェックされたものをメーカから供給を受けたいが、アプリケーションの柔軟な構築を許容できる透過性の高いドライバをハードメーカが供給している例がな

い』従ってこの部分も内作せざるを得ないが、それによって全体のパフォーマンスの向上と、アプリとドライバとの適切な負荷配分を行うことができる。環境は６８０３０のＶＭＥとＯＳ９ｖｅｒ２．４である。

3-2. ファイルマネージャ・デバイスドライバ・デバイスディスクリプタの役割

ＯＳ－９では、プログラムが、直接ボードのポートを参照してボードを操作することは禁止されている。システムコール等を使用し、カーネルの管理のもとにボードにアクセスしなければならない。その際に、必要なのがファイルマネージャ、デバイスドライバ、及びデバイスディスクリプタである。

・ファイルマネージャは同じ種類のデバイスを同一に扱えるように論理的操作を行う。例えば、ハードディスクとフロッピーディスクの様なデバイスは同じファイルマネージャで扱われる。

・デバイスドライバは、実際のデバイスを操作し、ファイルマネージャが要求する論理構造にデータを変換することを行う。

・デバイスディスクリプタは、個々のボードの情報を含んだデータテーブルで、例えば、同じＡＤボードであっても、ディップスイッチの設定でゲインが変更できる場合、その情報をデバイスディスクリプタに登録することができる。また、ポートアドレスなどは、システム内でユニークなものであり、このような情報もデバイスディスクリプタに登録しておく必要がある。

これらの、詳しい内容はＯＳ－９テクニカルマニュアルの３章に記述されている。

3-3. ファイルマネージャの構造

ファイルマネージャは、１３個のサブルーチンの集合体として構成される。カーネルが必要に応じてそれらの内の必要なサブルーチンをコールする。

１３のサブルーチンの名前は

Creat, Open, Makdir, Chgdir, Delete, Seek, Read,

Write, Readln, Writeln, GetStat, SetStat, Close である。

これらサブルーチンの意味は、ＯＳ－９テクニカルマニュアルの３．３．２に記述されている。

これらの、サブルーチンはカーネルからコールされるときテーブルジャンプによるサブルーチンコールで参照されるようになっている。そのため、ファイルマネージャは必ずＬｉｓｔ１の様な構造になっていなければならない。ファイルマネージャがコールされるときのレジスタは以下の値がセットされる。

(a1)：パスディスクリプタへのポインタ。

(a2)：現在のプロセスディスクリプタへのポインタ。

(a5)：ユーザのレジスタ格納域へのポインタ。システムコールをコールするときに設定されていたレジスタが保存されている。ここを参照することによりシステムコールへのパラメータの授受が行える。

(a6)：システムのグローバルデータ領域へのポインタ。

これ以外のレジスタに関しては情報がない。またファイルマネージャからカーネルにリターンするときのレジスタの状態についても記述がない（ただし、一般的にエラーが生じた場合は、CCRのキャリービットをセットして、エラーコードをd1.wで返す）。そのため、使用する全てのレジスタは、安全のためにスタックにＰＵＳＨしてからの使用し、カーネルにリターンするときはスタックからＰＯＰする。

3-4. デバイスドライバの構造

デバイスドライバは、ファイルマネージャと同じ様に７つのサブルーチンの集合体である。カーネルまたはファイルマネージャが必要に応じてこれらのサブルーチンコールする。７つのサブルーチンは、

Init, Read, Write, GetStat. SetStat, TrmNat, Error である。

これらのサブルーチンの意味はＯＳ－９テクニカルマニュアルの３．４．１に記述されている。

この内Errorのエントリは現在のバージョンでは使用していないので、０としておくことが望まれる。また、

Init, TrmNatはカーネルからのみコールされ、ファイルマネージャからはコールされない。一方、

Read, Write, GetStat, SetStatはファイルマネージャからコールされ、カーネルからはコールされることない。

これのサブルーチンは、コールされるときテーブルジャンプによるサブルーチンコールで参照されるので、デバイスドライバーはＬｉｓｔ２の様なソースになっている。デバイスドライバーがコールされるときのレジスタは以下の値がセットされる。

【Init, TrmNatがコールされるとき】

(a1)：デバイスディスクリプタモジュールへのポインタ

(a2)：ドライバのスタティックストレージへのポインタ

(a4)：入出力機能を要求するプロセスデスクリプタへのアドレス

(a6)：システムグローバル領域へのポインタ

【Read, Write, GetStat, SetStatがコールされるとき】

これらのサブルーチンはファイルマネージャからコールされるのでレジスタの値はファイルマネージャに依存するので、レジスタの内容は固定的に決めることはできないが、テクニカルマニュアルには以下のようにすることが勧められている。

(a1)：パスデスクリプタへのアドレス。

(a2)：ドライバのスタティックストレージへのアドレス。

(a4)：入出力機能を要求するプロセスデスクリプタへのアドレス。

(a5)：呼び出しているプロセスのレジスタ格納域へのポインタ。

(a6)：システムグローバル領域へのポインタ。

つまり、逆に言うとファイルマネージャを設計するときに、デバイスドライバが上記のような設定でコールされるよう配慮するべきである。

4．デバイスドライバをＣで作成する前に

ＯＳ９上で複数のソースファイルから実行可能プログラムをつくる場合、何らかの方法により、プログラムの出発点を示すメインライン・セグメントを含めなければならない。

ソースコードがアセンブリ言語で記述されている場合、まずファイルごとにアセンブルを行うことで生成される複数のリロケータブル・オブジェクト・ファイルがリンカによってそのまま結合される。従ってプログラマがいずれか一つのファイルの中に必ずメインライン・セグメントを記述しておかなくてはならない（このことは、プログラムだけでなくファイルマネージャやデバイスドライバにもあてはまる）。ソースコードがＣ言語で記述されていて、ｃｃでコンパイル

をする場合には、まずｃｐｐがコメント除去やマクロ展開等を行い、その次にｃ６８がそれぞれのファイルをアセンブリ言語に翻訳し、さらにそれがアセンブルされて、リロケータブル・オブジェクト・ファイルが生成される。ところがそれらのリロケータブル・オブジェクト・ファイルのいずれにもメインライン・セグメントは含まれておらず、ｃｃがリンカを起動するときに、メインライン・セグメントを含みｍａｉｎ関数のコールが記述されているｃｓｔａｒｔ．ｒというリロケータブル・オブジェクト・ファイルを自動的に結合させる。今回のデバイスドライバは、オプションでｃｓｔａｒｔ．ｒの代わりに別のファイルを指定してリンクする。そのファイルがメインラン・セグメントを含むデバイスドライバの骨組みとなる部分で、この部分だけはアセンブラで記述しなければならない。

5-1. Device Driver in C

デバイスドライバをＣでつくるときは、デバイスドライバ用のｃｓｔａｒｔ．ｒに相当するものが必要であるが、現在のＣコンパイラには、デバイスドライバ用のｃｓｔａｒｔ．ｒは用意されていない。そこで今回は、デバイスドライバ用のｃｓｔａｒｔ．ｒも作成した。当然のことながらこの部分はアセンブラで記述しなければならない。（今回のソースでは、din3ca.aというファイルをアセンブルしたものがデバイスドライバ用のｃｓｔａｒｔ．ｒに相当する。）

つまり、結果としてデバイスドライバの骨組み部分をアセンブラで記述し、そこからＣの関数をコールすることで、デバイスドライバの大部分をＣで記述するとういう手法を用いた。したがってアセンブラの骨組み部分とそこからコールされるＣの関数のインターフェースが必要になる。また、ファイルマネージャが異なれば、デバイスドライバの主要部分はかなり変わってくるので、アセンブラ部分をかなり書き換えなくてはならず、アセンブラの難解さを完全に排除することはできない。しかし、この方法でデバイスドライバの大部分をＣで記述できると言うことは、ソースの可読性を高め、アセンブラでは作成するのに非常に困難となる複雑な処理も容易に行えるようになり、開発も容易になる。そして一度製作したデバイスドライバに、後から機能を追加する場合は、アセンブラの記述を追加するする必要はほとんどない。

Ｃで記述することで処理速度が損なわれることが予想される。ＥＳＲＦでは、ＳＣＦとの組み合わせで使用した場合１０％の速度劣化が観測されている。ＳＰｒｉｎｇ－８では自作のファイルマネージャ（ａｄｉｏｍ）との組み合わせで調査中であるが、数％の劣化に留まっていると思われる。近年のＣＰＵの高速化はそれを補ってなお余りある。

5-2. ソースについて

例として以下に示すソースのターゲット・ボードはデジタル社製のDIN3である。機能はパス名で指定したビット列を読み込むというもので、デバイスドライバー名はDIN3C、デバイスディスクリプタ名はDC0、ファイルマネージャ名はADIOMである。DIN3Cのソースは、DIN3ca.a、DIN3c.c、DIN3c.h、DC0のソースはDC0.cである。DIN3ca.a、DIN3c.c、DIN3c.h、DC0.c及びこれらのmakefileをＬｉｓｔ１～３に示す。

5-3. デバイスドライバをＣで記述する際の要点

Ｃコンパイラは、デフォルトではリンクの段階でｃｓｔａｒｔ．ｒをリンクするが、ここではコンパイラの"cs"オプションを用い、デバイスドライバ用のｃｓｔａｒｔ．ｒ、すなわちｄｉｎ３ｃａ．ｒをリンクする。ｄｉｎ３ｃａ．ｒのソースはアセンブラで記述されたｄｉｎ３ｃａ．ａである。その結果、アセンブラからＣの関数をコールことになる。そこで、デバイスドライバの各エントリに必要な（有用な）レジスタを、Ｃの関数の引き数にして渡す。アセンブラからＣの関数をコールする方法については、Ｃコンパイラ・ユーザーズ・マニュアルの" 3.1.5 アセンブリ言語とのインターフェース" に記述されている。アセンブラからＣの関数をコールする際に特に注意すべき点は、デバイススタティックストレージの参照方法である。デバイススタティックストレージは、アセンブラでは、vsectで宣言し、a2がデバイススタテッィクストレージへのポインタとなっており、普通a2を用いたアドレッシングモードを使用して参照する。ところが、Ｃのソースをアセンブラにコンパイルすると、Ｃのスタティック変数がvsectで宣言される変数となり、その参照はa6を用いたアドレッシングモードを使用する。つまり、Ｃでデバイススタティックストレージを宣言して参照を行うためには、それをスタティック変数で宣言し、Ｃの関数をコールする前に"movea.l a2,a6"を行って、アドレッシングモードの違いを吸収する必要がある。この事を忘れるとシステムがクラッシュする可能性がある。なぜなら、デバイスドライバがコールされるときのa6は、システムグローバル変数へのポインタとなっているため、システムグローバル変数を無意味に変更する可能性があるからである。

6．まとめ

制御機器の中心となる計算機はすさまじい勢いで進歩している。信号の入出力インターフェース部分となるＡＤ変換の精度と速度は、ＣＰＵのそれほど急激な進展はないが、それでも１００ＭＨｚ～１ＧＨｚサンプルで８チャンネル１スロット幅の製品が現れたりする。当然これらの新しいデバイスがモニタの性能向上とビーム安定化の新しいアルゴリズムの実現を可能にする可能性もある。ソフトウェア管理と予備ボードの確保などから、ある程度使用するハードを限定することは必要であるが、必要な新規デバイスをすぐ取り入れられる様にすることも重要である。Ｃ言語によるデバイスドライバは、そのような要求に対応することにも貢献する。さらに、改造によって、カウンタモードやアベレージングモードなどをデバイスドライバが持つことによってアプリケーションプロセスの負荷を軽減し、ボードによってはＣＰＵ負荷を軽減することもできる。そしてＣで記述することによる可読性と柔軟性が、ＯＯＰのメリットを発揮できるデバイスアクセスを提供する。

謝辞

このデバイスドライバ製作にはＥＳＲＦのコンピュータサービスグループの助言とＰｅｔｒｉやＥｍａｎｎｕｅｌのドライバを参考にさせて頂いたので、ここに感謝の意を述べる。

参考文献

[1]H.Yoshikawa, et.al."Design of a Control System of the Linac for SPring-8", Proc. of ICALEPCS'91.

List1
--- filename : din3ca.a ---

```
            nam    DIN3C
            ttl    Driver module for DIN3
            use    "defsfile"
            use        "../defs/ss_.def"

Edition             equ        1                    current edtion number
Typ_Lang set        (Drivr<<8)+Objct DeviceDriver/Assembler
Attr_Rev  set       (ReEnt+SupStat<<8)+0
Stack               equ        500

                    psect      DIN3C,Typ_Lang,Attr_Rev,Edition,Stack,Entry

Entry  dc.w   Init
       dc.w   Read
       dc.w   Write
       dc.w   GetStat
       dc.w   PutStat
       dc.w   Term
       dc.w   0

Init:
            move       sr,-(a7)              save sr
            ori        #$0700,sr             mask all IRQ
            movem.l    d0/d2-d7/a0-a7,-(a7)
            movea.l    a2,a6

            move.l     a2,d1
            move.l     a1,d0
            bsr        CInit
            tst.l      d0
            bne.s      InitError
            movem.l    (a7)+,d0/d2-d7/a0-a7
            clr.l      d1
       move    (a7)+,sr              restore sr
            rts

InitError
            move.l     d0,d1
            movem.l    (a7)+,d0/d2-d7/a0-a7
       move    (a7)+,sr              restore sr
       ori     #Carry,ccr            set carry
       rts

Read:
            movem.l    d2-d7/a0-a7,-(a7)
            movea.l    a2,a6
            move.l     a5,-(a7)
            move.l     a4,-(a7)
            move.l     a2,d1
            move.l     a1,d0
            bsr        Cread
            movea.l    (a7)+,a4
            movea.l    (a7)+,a5
            movem.l    (a7)+,d2-d7/a0-a7

       clr.l   d1                    clear error code/flag
       rts

Write:
            movem.l    d2-d7/a0-a7,-(a7)
            movea.l    a2,a6
            move.l     a5,-(a7)
            move.l     a4,-(a7)
            move.l     a2,d1
            move.l     a1,d0
            bsr        Cwrite
            movea.l    (a7)+,a4
            movea.l    (a7)+,a5
            movem.l    (a7)+,d2-d7/a0-a7

       move.w  #E$BMode,d1           write not allowed
       ori     #Carry,ccr            set carry
       rts

GetStat:
            movem.l    d2-d7/a0-a7,-(a7)
            movea.l    a2,a6
            move.l     a5,-(a7)
            move.l     a4,-(a7)
            move.l     a2,-(a7)
            move.l     a1,d1
            bsr        Cgetstat
            movea.l    (a7)+,a2
            movea.l    (a7)+,a4
            movea.l    (a7)+,a5
            tst.l      d0
            bne.s      GetStatError
            clr.l      d1
            movem.l    (a7)+,d2-d7/a0-a7
            rts

GetStatError
            movem.l    (a7)+,d2-d7/a0-a7
            move.l     d0,d1
            ori        #Carry,ccr
            rts

PutStat:
            move   sr,-(a7)           save sr
            ori    #$0700,sr          mask all IRQ
            movem.l    d2-d7/a0-a7,-(a7)
            movea.l    a2,a6
            move.l     a5,-(a7)
            move.l     a4,-(a7)
            move.l     a2,-(a7)
            move.l     a1,d1
            bsr        Cputstat
            movea.l    (a7)+,a2
            movea.l    (a7)+,a4
            movea.l    (a7)+,a5

RetCput:
            tst.l      d0
            bne.s      PutStatError
            clr.l      d1
            movem.l    (a7)+,d2-d7/a0-a7
       move    (a7)+,sr              restore sr
            rts

PutStatError:
            move.l     d0,d1
            movem.l    (a7)+,d2-d7/a0-a7
       move    (a7)+,sr              restore sr
            ori    #Carry,ccr
            rts

Term:
            move   sr,-(a7)               save sr
            ori    #$0700,sr              mask all IRQ
            movem.l    d2-d7/a0-a7,-(a7)
            movea.l    a2,a6

            move.l     a4,-(a7)
            move.l     a2,d1
            move.l     a1,d0
            bsr        Cterm
            movea.l    (a7)+,a4
            clr.l      d1
            movem.l    (a7)+,d2-d7/a0-a7
            move   (a7)+,sr           restore sr
            rts

Set_IRQ:
            movem.l    d1-d7/a0-a7,-(a7)
            movea.l    68(a7),a3
            movea.l    64(a7),a2
            lea.l      IRQHandle(pc),a0
            bra        CallIRQ

Reset_IRQ:
            movem.l    d1-d7/a0-a7,-(a7)
            movea.l    68(a7),a3
            movea.l    64(a7),a2
            movea.l    #0,a0
            bra        CallIRQ

CallIRQ
            os9        F$IRQ
            bcs        Sys_Call_F$IRQErr
            clr.l      d0
            movem.l    (a7)+,d1-d7/a0-a7
            rts

Sys_Call_F$IRQErr
            move.l     d1,d0
            movem.l    (a7)+,d1-d7/a0-a7
            rts

IRQHandle:
            move   sr,-(a7)               save sr
            ori    #$0700,sr              mask all IRQ
            movem.l    d1-d7/a0-a7,-(a7)
            move.l     a2,d0
            move.l     a3,d1
            move.l     a6,-(a7)
            bsr        CIRQHandle
            movea.l    (a7)+,a6
            bne.s      IRQHandleError
            tst.l      d0
            bne.s      IRQHandleError
            movem.l    (a7)+,d1-d7/a0-a7
            move   (a7)+,sr
            clr.l      d1
            rts

IRQHandleError
            movem.l    (a7)+,d1-d7/a0-a7
            move   (a7)+,sr
            ori    #Carry,ccr
            rts

            ends
```

List3
--- filename : makefile ---

```
CFLAGS = -jsgixt=/dd -cs=./rels/din3ca.r -k=2lclf -tp=020
RFLAGS = -m=3
RDIR   = RELS
RC     = r68020

din3c.t: din3c DC0 DC2
         touch din3c.t

din3c:din3c.c din3ca.r

din3ca.r:din3ca.a

DC0:     DC0.r /h0/lib/sys.l
  l68 RELS/DC0.r -O=../OBJ/DC0 -l=/h0/lib/sys.l

DC0.r:   DC0.a
  r68020 DC0.a -o=RELS/DC0.r
```

List2
—— filename : din3c.c ——

```
#include  <module.h>
#include  <sysio.h>
#include  <MACHINE/reg.h>
#include  <path.h>
#include  <procid.h>
#include  <modes.h>
#include  <errno.h>
#include  <sg_codes.h>
#include  "din3c.h"
sysioStatic             sysio;
static    union         nint        Ssharebit,
                                    Gsharebit,
                                    readwork,
                                    interrupt;
int                                 errno;
unsigned int Cinit(dev_desc,dev_static)
mod_dev                 *dev_desc; /*        ptr to device descriptor        */
                        /*          /dd/defs/module.hの中で宣言されている */
sysioStatic             *dev_static;/*       prt to device static storage    */
                        /*          /dd/defs/sysio.hの中で宣言されている   */
{         /*            デバイスドライバの初期化処理を行う                  */
          struct        din3port  *board;
          /*            device static storage  を初期化する      */
          Ssharebit.d.l1 = 0;
          ..................................
          /*            ボードを初期化する                      */
          board = (struct       din3port *)dev_desc->_mport;
          board->IE = 0;
          ..................................
          /*            割り込み登録                                */
          return(Set_IRQ(dev_desc->_mvector,dev_desc-
>_mpriority,dev_static,dev_desc->_mport));
}
unsigned int Cread(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
                        /*          /dd/defs/path.hの中で宣言されている    */
sysioStatic             *dev_static;/*       ptr to device static storage    */
                        /*          /dd/defs/sysio.hの中で宣言されている   */
procid                  *procs_desc;/*       ptr to process descriptor       */
                        /*          /dd/defs/procid.hの中で宣言されている  */
struct    usrstack      *usr_stack;
                        /*          ptr to user register stack      */
                        /*          din3c.hの中で宣言されている    */
{         /*            読み込み処理を行う                  */
          ..................................
          return( readwork.d.l0);
          /*            データを返す                                */
}
unsigned int Cwrite(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            書き込み処理を行う                          */
          /*            デジタルインのボードのため何もしません      */
          return(0); /*          d0に 0 を入れて帰る                 */
}
unsigned int Cget_option(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            パスディスクリプタのオプション領域を
                        プロセスのバッファにコピーする
*/
}
unsigned int Cget_path_size(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{
          /*            オープンしているパスの情報をプロセスに返す  */
}
unsigned int Cgetstat(func_code,path_desc,dev_static,procs_desc,usr_stack)
unsigned short          func_code;/*         function code
*/
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            function code で必要な関数をコールする */
          switch        (func_code)
          {
                        case      SS_Opt
:return(Cget_option(path_desc,dev_static,procs_desc,usr_stack));
                        case
SS_Size:return(Cget_path_size(path_desc,dev_static,procs_desc,usr_stack));
                        default                 :return(E_UNKSVC);
          }
                        /*          エラーの場合エラーコードを返す。    */
                        /*          正常の場合、0 を返す        */
}
unsigned int Cput_open(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            パスをオープンする                  */
          ..................................
}
unsigned int Cput_close(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            パスをクローズする                  */
          ..................................
}
unsigned int Cput_set_EvSig(path_desc,dev_static,procs_desc,usr_stack)
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            割り込み発生時に発行するイベントシグナルを
                        登録する                                    */
          ..................................
          return(0);
}
unsigned int Cputstat(func_code,path_desc,dev_static,procs_desc,usr_stack)
unsigned short          func_code;/*         function code
*/
union pathdesc          *path_desc;/*        ptr to path descriptor          */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
struct    usrstack      *usr_stack;
{         /*            function code で必要な関数をコールする */
          switch        (func_code)
          {
                        case      SS_Open:
return(Cput_open(path_desc,dev_static,procs_desc,usr_stack));
                        case      SS_Close:
return(Cput_close(path_desc,dev_static,procs_desc,usr_stack));
                        case      SS_SSig:
return(Cput_set_EvSig(path_desc,dev_static,procs_desc,usr_stack));
                        default                         :return(E_UNKSVC);
          }
          /*            エラーの場合エラーコードを返す。    */
          /*            正常の場合、0 を返す        */
}
unsigned int Cterm(dev_desc,path_desc,dev_static,procs_desc)
mod_dev                 *dev_desc;/*         ptr to device descriptor        */
sysioStatic             *dev_static;/*       ptr to device static storage    */
procid                  *procs_desc;/*       ptr to process descriptor       */
{         /*            終了処理を行う                              */
          struct        din3port  *board;
          board = (struct       din3port *)dev_desc->_mport;
          board->IE = 0;
          /*            割り込み登録を解除する              */
          return(Reset_IRQ(dev_desc->_mvector,dev_desc-
>_mpriority,dev_static,dev_desc->_mport));
}
CIRQHandle(dev_static,board,sys_global)
sysioStatic             *dev_static;/*       ptr to device static storage    */
struct    din3port      *board;/*  ptr to port                 */
                        /*          din3c.hの中で宣言されている */
void      *sys_global;
{         /*            割り込み処理を行う */
          ..................................
}
```