FUJITSU Software
SIMPLIA TF-MDPORT

ユーザーズガイド
(開発資産流用支援ツール)

Solaris

2021年9月

CONTENTS

GLOSSARY

改行コード

拡張子

カタカナ

可変長

行順ファイル

警告メッセージ

コードセット3

コード変換機能

シフトコード

シフトJIS

他社コード

データ定義文

データ定義文ファイル

データファイル

データファイル変換

テキストファイル

テキストモード変換

特殊指示

日本電気JIPSコード

バリデーション機能

バリデーションポリシーファイル

日立KEISコード

標準エラー出力

標準出力

標準入力

ファイル形式

利用者定義変換テーブル

利用者定義文字

レイアウトファイル

レコード変換機能

COBOL固定長ソース

COBOLファイル

cobolEUC

EUC

IBM日本語コード

JEF

JIS8

mdportfコマンド

mdportgコマンド

RDBローダ型ファイル

RDW

-amend1

-amend2

-cvcobgen

-cvcobuse

-cvtable

-delimita

-emu

-f

-formatf

-icobfl

-icode

-iconv

-icopy

-if

-ikana

-informix

-irecdlm

-irecfm

-ireclen

-iso

-itype

-lfs

-msg

-msgcont

-msgmax

-noblc

-noconv

-nocs3

-nosignature

-ocobfl

-ocode

-ocopy

-of

-okana

-oracle

-orecdlm

-orecfm

-oreclen

-otype

-policyf

-rdb2

-rep

-shift

-validation

まえがき

[本書の目的]

本書はSolaris版のSIMPLIA TF-MDPORT、Linux版のSIMPLIA/TF-MDPORT、Linux for Itanium版のSIMPLIA/TF-MDPORT、x64-Linux版のSIMPLIA TF-MDPORTの機能およびその使用方法について記述しています。MDPORTは、開発保守支援システムの一つとして位置づけられ、汎用機・オフコン・Solaris・Linux・Windowsとの間で、データやソースの流通を支援するツールです。本書では、読者が次の二つについて理解できることを目的としています。

- MDPORTの使用目的および特徴
- MDPORTの機能

[本書の読者]

本書は、MDPORTを使用してデータやソースの流用を行う方に読んでいただくように書かれています。本書を読むためには以下の知識が必要です。

- 文字コードに関する基本的な知識(汎用機 EBCDICコード、Solaris/Linux機で使われるEUCコード、シフトJISコード、Unicodeなど)
- ご使用になるOSに関する基本的な知識

[本書の構成]

本書の構成と内容は次のとおりです。

第1章 概要
MDPORTの機能と動作環境について説明しています。

第2章 相手側交換手順
MDPORTの入出力である、非Solaris/Linux機側の交換手順について、説明しています。

第3章 使用手引
MDPORTのコマンドと、利用者定義の入力ファイルの記述形式について、説明しています。

付録A 使用例
MDPORTの使用例を、様々なパターンごとに図示して説明しています。

付録B 変換仕様
MDPORTにおけるコード変換・ファイル形式変換について、変換仕様を説明しています。

付録C 出力メッセージ
MDPORTで出力されるメッセージについて説明しています。

付録D コード変換/レコード変換機能(ライブラリ)
コード変換/レコード変換機能(ライブラリ)について説明しています。

[本書の読み方]

本書は、目的別に次のようにお読みください。

概要を知りたい方	第1章
非Solaris/Linux機側での手順を知りたい方	第2章
使い方を知りたい方	第3章
使用例を知りたい方	付録A
変換仕様について知りたい方	付録B
出力メッセージについて知りたい方	付録C
ユーザアプリケーションを作成したい方	付録D

[本書の位置づけ]

本書を読むにあたって、必要に応じて次のマニュアルをお読みください。

使用目的	参照マニュアル
COBOLの文法規約、データ属性	COBOL文法書
文字コード体系の詳細	富士通文字コード解説書
COBOLファイル仕様	NetCOBOL 使用手引書

[本書の表記上の規約]

本書はSolaris版のSIMPLIA TF-MDPORT、Linux版のSIMPLIA/TF-MDPORT、Linux for Itanium版のSIMPLIA/TF-MDPORT、x64-Linux版のSIMPLIA TF-MDPORTの4製品に関するユーザーズガイドです。

本書中に**[Solaris]**という記載がある場合、その内容はSolaris版 MDPORT固有の説明をさします。

本書中に**[Linux]**という記載がある場合、その内容はLinux版 MDPORT固有の説明をさします。

本書中に**[Linux for Itanium]**という記載がある場合、その内容はLinux for Itanium版 MDPORT固有の説明をさします。

本書中に**[x64-Linux]**という記載がある場合、その内容はx64-Linux版 MDPORT固有の説明をさします。

本書中におけるロケールの初期値は、EUC環境で利用する場合を前提としています。

[輸出管理規制について]

本ソフトウェアを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

[登録商標について]

- Linuxは米国およびその他の国におけるLinus Torvaldsの登録商標です。
- UNIXは、米国およびその他の国におけるオープン・グループの登録商標です。
- Red Hat、Red Hat Enterprise Linuxは米国およびその他の国において登録されたRed Hat, Inc.の商標です。
- Intel、Itaniumは、アメリカ合衆国およびその他の国における Intel Corporation またはその子会社の商標です。
- OracleとJavaは、Oracle Corporationおよびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- Microsoft、Windows、Visual Studio、Visual Basic、Internet Explorer、Windows Server、その他のマイクロソフト製品の名称および製品名は、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。
- そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

なお、本文中では、TM、®マークは略してあります。

平成 8年 9月 初版
平成11年 6月 第2版
平成12年 6月 第3版
平成13年 4月 第4版
平成14年 6月 第5版
平成15年12月 第6版
平成16年 4月 第7版
平成16年11月 第8版

平成17年 7月 第9版
平成17年10月 第10版
平成18年 3月 第11版
平成18年12月 第12版
平成20年 5月 第13版
平成21年 6月 第14版
平成22年 1月 第15版
平成24年 1月 第16版
令和 3年 9月 第17版

第1章 概要

本章では、MDPORTの構成・機能概要等を説明します。

1.1 システム構成

MDPORTを利用する際に必要となるシステム構成を図1.1に示します。

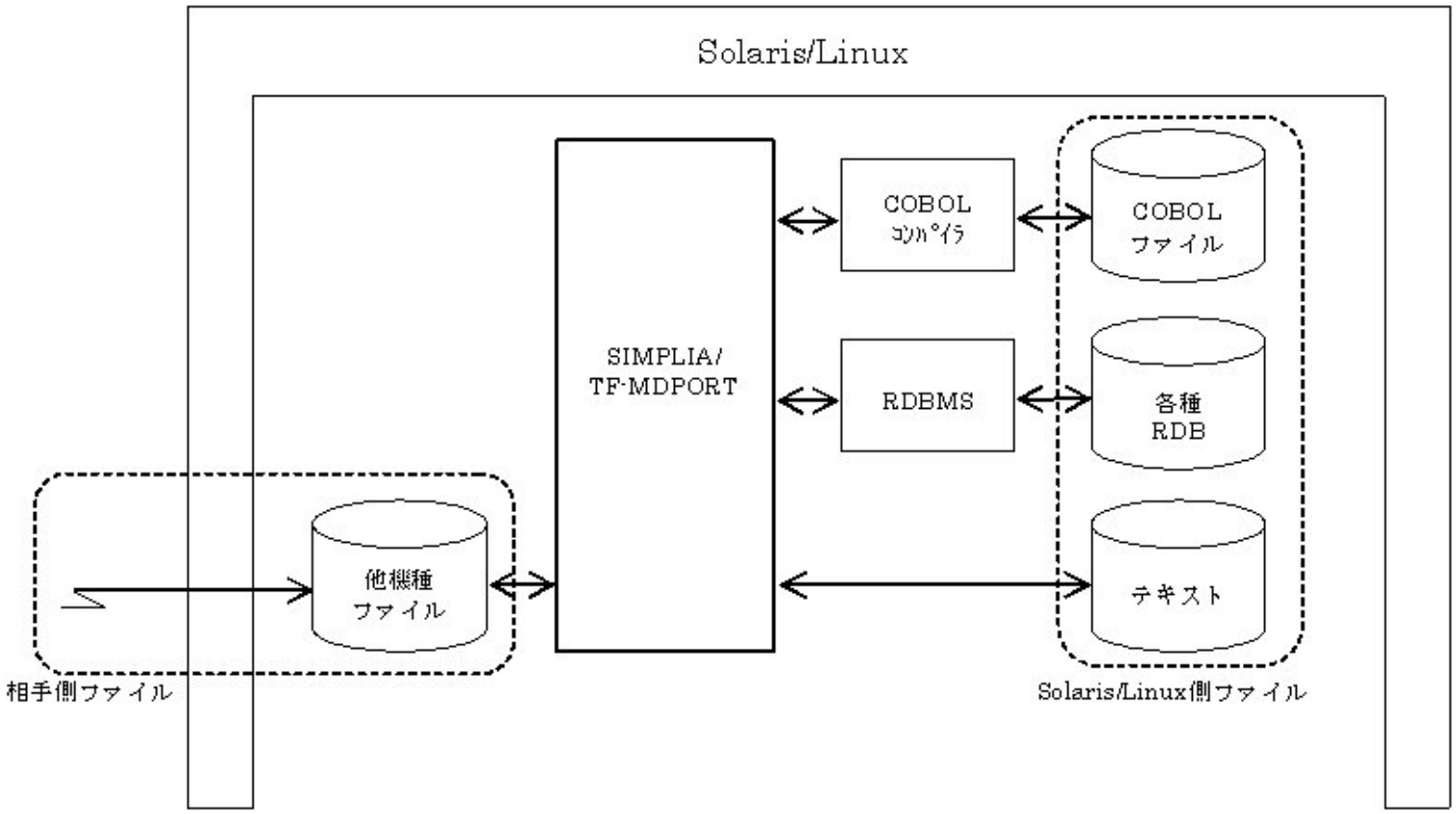


図1.1 MDPORTのシステム構成

1.2 機能概要

MDPORTは、Solaris/Linux機と汎用機・オフコン・PC間で、ソースやデータファイル等の資産を流用させることを目的としており、以下の二つの機能に分類されます。
-ファイル交換機能
-コード変換/レコード変換機能

本節では、各機能の概要について説明します。

1.2.1 ファイル交換機能の概要

ファイル交換機能は、**mdportf**コマンドによって起動されます。ファイル交換機能には、汎用機(オフコン・PC含む)のデータ形式と、Solaris/Linux上の各種ファイル形式間との変換を行う機能があります。図1.2にファイル交換機能の概要を示します。

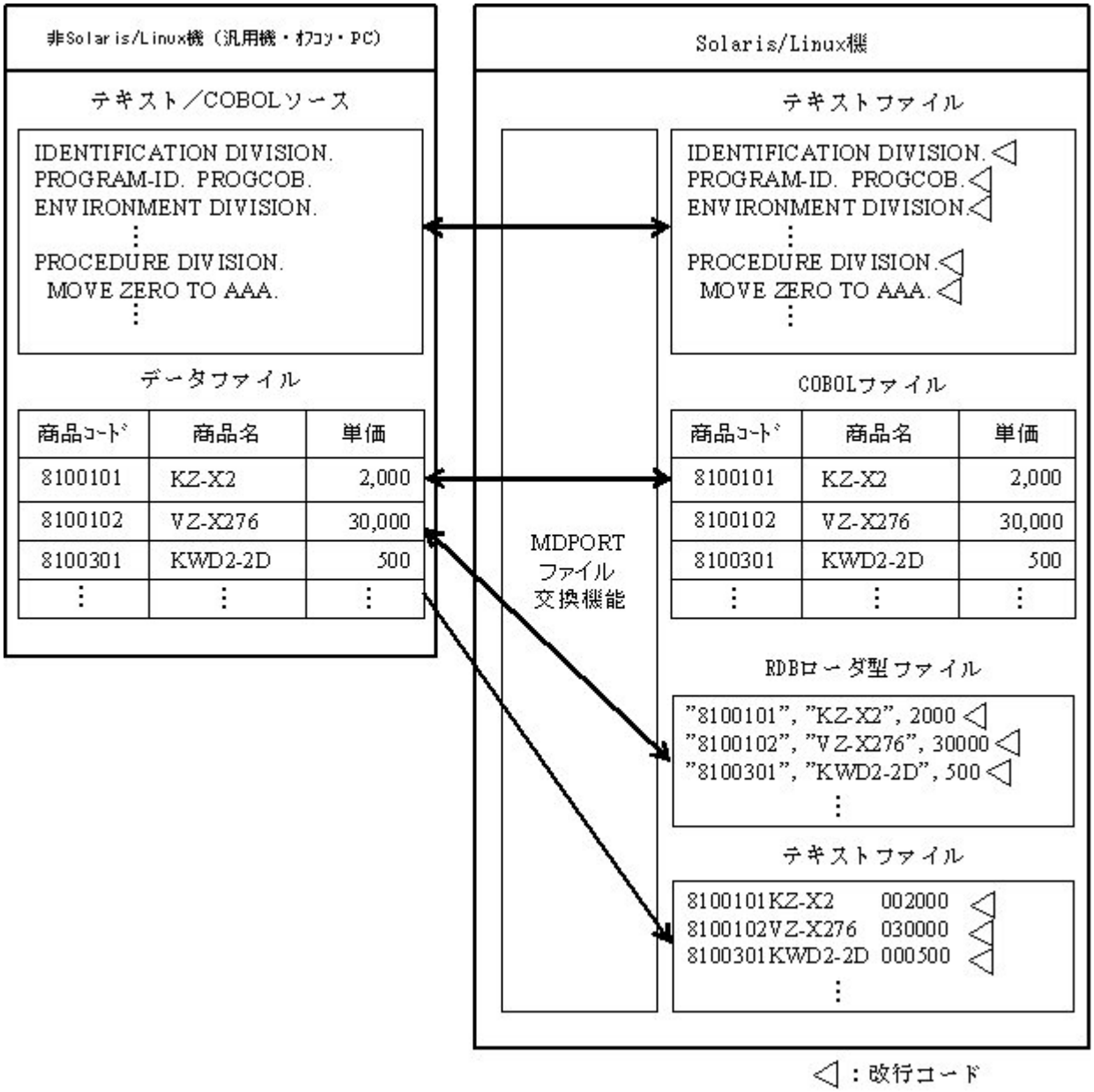


図1.2 ファイル交換機能の概要

1.2.1.1 テキストモードの変換

汎用機での固定長および可変長レコードのテキスト型ファイルや、各種テキスト型ファイル間での相互変換ができます。
また、汎用機での固定長COBOLソースの、73カラム以降の注釈欄を削除し、可変長COBOLソースの形式へ移入することができます。
表1.1にテキストモード変換での変換パスを示します。

表1.1 テキストモード変換での変換パス

変換先 変換元	Solaris/Linuxテキスト	汎用機ソース	Windowsテキスト
Solaris/Linuxテキスト	○	○	○
汎用機ソース	○	-	-
汎用機COBOLソース	○	-	-
Windowsテキスト	○	-	-

○：指定可 -：指定不可

1.2.1.2 データファイルの変換

以下に示すファイル形式間にて、データファイルを変換することができます。この変換では、レコード内の項目属性等のレイアウトを、データ定義文と呼ばれるCOBOLのレコード記述で指示します。
表1.2にデータファイル変換での変換パスを示します。

データファイル

汎用機上の順ファイルを、ファイル転送や外部媒体経由で移入したバイナリファイルが、この形式に該当します。データ属性(内部表現形式)として、COBOLのデータ型(COBOL特有の符号付外部10進や内部10進等)を扱うことができます。
また、以下のファイル編成についても、順ファイルを経由することによりデータファイルで扱うことができます。

Mシリーズ	VSAMファイル
Kシリーズ	索引ファイル、RDB物理ファイル
FM Gシリーズ	RDM物理ファイル

COBOLファイルシステム

Solaris/LinuxのCOBOLファイルシステムをアクセスすることができます。MDPORTでは、以下のCOBOLファイルシステムに対応しています。

[Solaris]
- NetCOBOL for Solaris

[Linux] [Linux for Itanium] [x64-Linux]
- NetCOBOL for Linux

COBOLのファイル編成として、以下の形式を扱うことができます。

- レコード順ファイル
- 行順ファイル
- 相対ファイル
- 索引ファイル

RDBローダ型ファイル

RDBローダ型ファイルとは、データ項目間に区切り文字を挿入したCSV形式のテキストファイルを示します。このRDBローダ型ファイル形式により、各種RDB付属のローダ機能を使用してのデータ移出入が可能です。MDPORTでは、以下のSolaris/Linux上のRDBローダ入力形式に対応しています。

- Symfoware
- ORACLE
- INFORMIX

また、RDBローダ型ファイルは、各種表計算ソフト等へデータを流通させることができます。

テキストファイル

データファイル変換では、全てキャラクタ属性に変換したテキスト形式のファイルへ出力することができます。全てがキャラクタで構成されているので、標準出力での表示や、Cプログラム等で入出力を行うのに適しています。

表1.2 データファイル変換での変換パス

変換先 / 変換元	データファイル形式	COBOLファイルシステム	RDBローダ型	Solaris/Linuxテキスト
データファイル形式	○	○	○	○
COBOLファイルシステム	○	○	○	○
RDBローダ型ファイル	○	○	△	-

○ : 指定可 - : 指定不可 △ : テキストモード変換で可能

1.2.2 文字コードの変換

前項のファイル形式の変換機能に伴い、文字コードの変換を行うことができます。表1.3にコード変換パスを示します。

表1.3 データファイル変換での変換パス

変換元/変換先	EUC	シフトJIS	Unicode	JIS	EBCDIC(カナ)+JEF	EBCDIC(カナ)+他社漢字コード
EUC	○	◎	▲	▲	◎	◎
シフトJIS	◎	○	▲	▲	◎	◎
Unicode	▲	▲	▲※	×	▲	▲
JIS	▲	▲	×	×	×	×
EBCDIC(カナ)+JEF	◎	◎	▲	×	×	×
EBCDIC(カナ)+他社漢字コード	◎	◎	▲	×	×	×

○：変換可能

◎：標準コード変換(iconv)による変換も可能

▲：標準コード変換(iconv)が必要

×：未サポート

※：同一コード変換の場合は標準コード変換(iconv)を使用しない変換のみ可能

各日本語コードは、JIS非漢字・JIS第1水準漢字・JIS第2水準漢字に対応しています。各コード体系の利用者定義文字は、利用者定義変換テーブルで指定することにより変換できます。
JISコードとして、以下のコード体系に対応しています。

- ASCIIおよびJISコード
- ASCIIおよび78JISコード
- EBCDICおよびJISコード
- EBCDICおよび78JISコード

Unicodeは、UTF8、UCS2(ビッグ/リトルエンディアン)形式に対応しています。UTF8形式については4バイト以上の対応は行っていません。

富士通以外の他社コードとして、以下の日本語コード体系に対応しています。

- IBM日本語コード
- 日立KEISコード
- 日本電気JIPS(E)コード
- 日本電気JIPS(J)コード

JEF拡張非漢字・JEF拡張漢字の変換では、EUCコードのコードセット3(G3領域)へ変換することができます。(U90コードのJEF拡張領域へ変換します。)

1.2.2.1 標準コード変換(iconv)

MDPORTは標準コード変換(iconv)を使用してiconv変換を行います。この変換では、厳密な文字の対応付けを実現し、iconvによる外字連携を行うことができます。iconvを使用した変換方法については、3.5の実行環境を参照してください。

1.2.2.2 ファイル形式のサポート文字コード

データ変換における各種ファイル形式とサポート文字コードを以下に示します。

形式/文字コード	EUC	シフトJIS	JIS	JEF	UCS2	UTF8	COBOL Unicode系	他社コード
データファイル	○	○	○	○	○	○	○	○
COBOLファイル	○	○	×	×	×	×	○	×
RDBローダ型ファイル	○	○	×	×	○	○	×	×

[COBOL Unicode系]
COBOL Unicode系は日本語項目をUCS2形式、英数字項目をUTF8形式で格納するUnicode形式のCOBOLファイル形式です。
MDPORTでは、データファイルとしては順編成をサポートし、COBOLファイルとしてレコード順ファイル、行順ファイル、相対ファイル、索引ファイルに対応しています。

○：変換可能

×：未サポート

1.2.2.3 バリデーション機能(Charset Validator)を適用した変換

MDPORTは、文字コード変換を行う際に、特定の文字種ポリシーに従い変換を行うことができます。その機能をバリデーション機能と呼んでいます。
特定の文字種ポリシーとは、"バリデーションポリシーファイル"に定義された文字の運用ポリシー（方針）であり、現行システムで使用できない文字をブロックします。

本機能の詳細な説明は、"3.7 バリデーション機能(Charset Validator)を適用した変換" を参照してください。

1.2.3 コード変換/レコード変換機能の概要

コード変換/レコード変換機能では、ユーザアプリケーションに組み込むことによりSolaris/Linux機と汎用機・オフコン・PC間で、ソースやデータファイル等の資源を流用することができます。図1.3にコード変換/レコード変換機能の概要を示します。

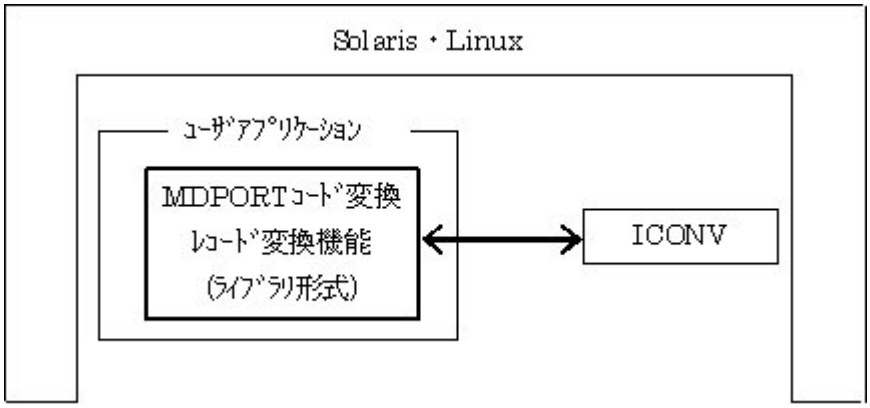


図1.3 コード変換/レコード変換機能の概要

コード変換/レコード変換機能は以下のような機能を提供する関数群です。

- コード変換機能　　各種コード変換を行う関数群です。
- レコード変換機能　　レコード単位にコード変換、レコード形式変換を行う関数群です。

MDPORTのシステムフローを図1.4に示します。

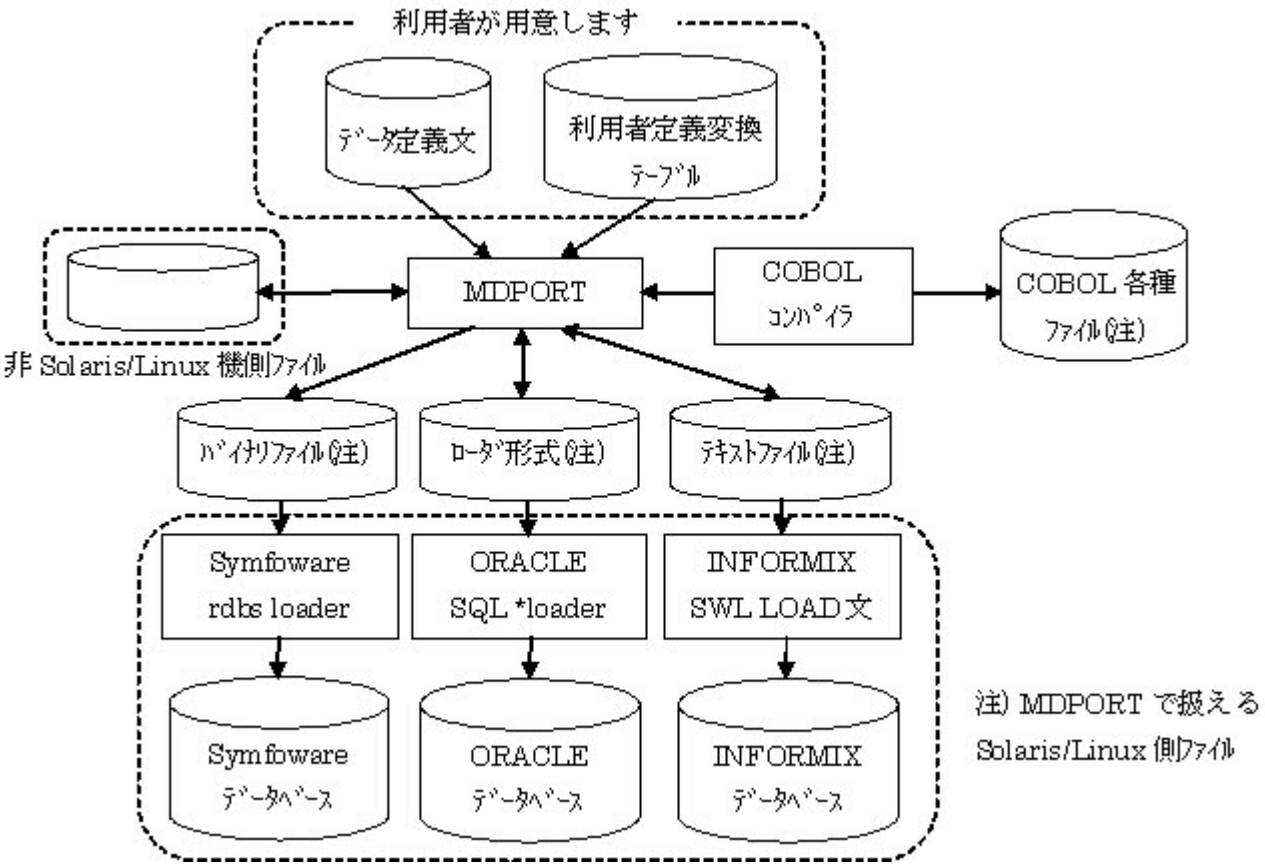


図1.4 MDPORTのシステムフロー

1.4 注意事項

使用にあたって、以下の注意事項があります。

- 入力ファイルおよび出力ファイルで、扱える最大レコード長は、32767バイトまでです。変換後に32767バイトを超える場合はエラーとなります。(ただし、mdportfコマンドにおいては、バイナリファイルの最大レコード長が32767バイトに収まっていれば、RDBローダ型ファイルの入出力で32767バイトを超える最大レコード長を保証します。)
- 文字データ項目の最大領域長は、32767バイトまでです。
- COBOLファイルで、副キーを持つ索引ファイル、または、1つのキーが連続した領域でない索引ファイルについては、扱うことはできません。
- UnicodeのCOBOLファイルで、レコードキーに日本語項目を指定した索引ファイルについては、扱うことができません。レコードキーに日本語項目以外を設定してください。
- コマンドで指定する各ファイル名の長さは、絶対パス名表現で127バイト以内でなければなりません。それを超える場合は、シンボリックリンクを指定して絶対パス名の長さを短くする、等の対応が必要です。
- データ定義文は、ANSI'85のCOBOL文法規約およびマニュアルで記載されている構文規則で記述されている必要があります。MDPORTでは、厳密な構文検査を行っていないため、例外的な記述がされた場合の動作保証はできません。
- データ定義文上に記述出来る項目数は、最大3000項目までです。
  マルチフォーマット定義の場合は、全てのフォーマットで定義された項目数を合わせて、最大3000項目までです。
  [Solaris] レコード変換機能では、最大1000項目までです。
  なお、反復項目、レコードや集団項目名は1として数えます。
  特殊指示V,L及びマルチフォーマットのフォーマット指示制御文は項目として数えません。
- Interstage Charset Managerの文字コード変換表のカスタマイズで全角アンダーバー（＿）を指定しないでください。指定する場合は、MDPORTの利用者定義変換テーブルを使用してください。
- 同一コード無変換にて、入力データに変換エラーとなるデータがある場合、コード変換を行わないため変換エラーは出力されません。出力結果の使用の際は注意が必要です。
- データファイル／COBOLファイルからデータファイル／COBOLファイルへの変換以外のデータ変換、テキスト変換での同一コード無変換では、数値項目の無変換を行いません。そのため、変換エラーが発生する可能性があります。
- 同一コード無変換を指定した場合や利用者定義変換テーブルの変換先に空白を定義した場合でも後続空白削除処理を行います。そのため、入力ファイルと出力ファイルで後続空白部分に差異が生じる場合があります。これを避けるためには、-noblcオペランドを指定して実行します。
- コード変換/レコード変換機能は再頒布できません。
- 数値項目データの厳密チェック指定と利用者定義変換テーブルのコード範囲制限解除指定は、コード変換機能では指定できません。
- コード変換/レコード変換機能を使用したアプリケーションを、旧形式から移行する場合は"3.6.3 旧形式からの移行のポイント"を参照してください。
- 入力形式がデータ形式で項目属性がBINARY、COMP、COMP-5などのバイナリ型の属性で出力形式がRDBローダ型、テキスト形式の場合、18桁を超える入力データは、出力される値が0となります。
- Solaris版V6/Linux for Itanium版V5L4/x64-Linux版V5L5以降のMDPORTでは他社コードを扱う場合、標準コード変換の使用がデフォルトになります。V5L3以前のMDPORTと同様の他社コード変換を行う場合は、mdportfコマンドで-iconv no指定が必要です。
- Unicode、JISコードを扱う場合iconvが必須です。
- バリデーション機能(Charset Validator)を適用した変換の注意点は、"3.7 バリデーション機能(Charset Validator)を適用した変換の注意"を参照してください。
- データ定義文にてPICTURE句を"N"で定義した日本語項目では、内容が全て日本語文字としてみなされ、2バイト系コード変換が行います。"B.5.3 日本語項目における変換仕様"を参照してください。
- コード/レコード変換機能を利用する場合、以下の注意が必要です。
  [Solaris] [Linux] 32ビット以外のアプリケーションとリンクして使用することはできません。
  [Linux for Itanium] [x64-Linux] 64ビット以外のアプリケーションとリンクして使用することはできません。
- レコード変換APIのMDP_initの引数で、入力文字コードまたは出力文字コードを指定しなかった場合、ロケールに従った文字コードで動作します。ただし、Unicodeロケールの場合は、eucコードと判断するため、MDP_initの引数の入力文字コードと出力文字コードを必ず指定してください。

[Linux] Linux版MDPORTの注意事項

- 起動する環境の文字コード体系について、シフトJISロケールは未サポートです。

[Linux for Itanium] [x64-Linux] Linux for Itanium/x64-Linux版MDPORTの注意事項

- 起動する環境の文字コード体系のサポートはUnicodeロケールのみです。

第2章 相手側交換手順

本章では、MDPORTの移入元/移入先である相手側機種上での操作手順について説明します。

2.1 相手側機種からSolaris/Linux機への移入手順

本節では、相手側機種上での移出からSolaris/Linux機上への移入までの手順について説明します。
図2.1に相手側機種からSolaris/Linux機への移入手順フローを示します。表2.1に、相手側機種で使用するコマンドおよびユーティリティを示します。

手順1

順編成以外のファイルは、全て順編成ファイルに変換する必要があります。
区分編成ファイルやライブラリ上のソース等を扱うには、1メンバ単位に順編成ファイルに変換することになります。

手順2

Solaris/Linux機へ移出するために、外部媒体への出力またはファイル転送を行います。
ファイル転送の場合、テキストモードでの転送は行わず、バイナリモード(透過モード)での転送を行うようにしてください。

手順3

Solaris/Linux機側で外部媒体からの入力、またはファイル転送の受け取りを行います。
外部媒体からの入力方法として"FDA制御オプション"等を使用します。

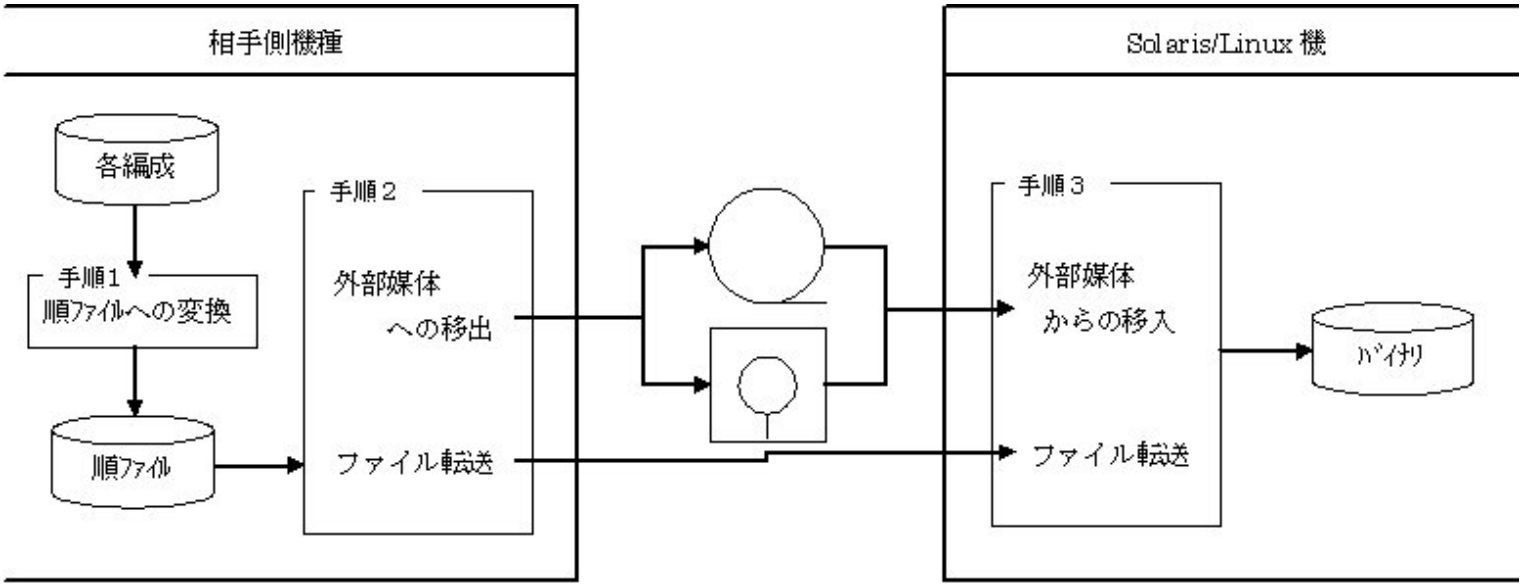


図2.1 相手側機種からの移入手順フロー

表2.1 相手側機種からの移入で使用するコマンド・ユーティリティ

相手側			順ファイルへの変換方法	外部媒体への出力方法	ファイル転送手段
機 種	OS	ファイル編成			
GS/Mシリーズ	MSP	GEMライブラリ	JQHGEM3/PUT	JSDGENER, JSDFLPDK/CNVDF, 等	F6680エミュレータ, HICS, ftp, 等
		VSAM(KSDS)	KQCAMS/REPRO		
	XSP	GEMライブラリ	JQHGEM3/PUT	FCPY, FDCV/CNVFD, 等	
		VSAM(KSDS)	KQCAMS/REPRO		
PRIMERGY 6000 Kシリーズ	ASP	ライブラリソース	不要	EXPSRC	F6680エミュレータ, HICS, ftp, 等
		SF,PF	不要	CNVFILE	
FM G	SX/G	原始テキスト	不要	オブジェクト一覧の"移出"	ftp, 等
		順,物理	不要	expfile	
PC	DOS/Windows	DOSファイル	不要	COPYコマンド, 等	ftp, 等

2.2 Solaris/Linux機から相手側機種への移出手順

本節では、Solaris/Linux機上の移出から相手側機種上への移入までの手順について説明します。
図2.2に相手側機種からSolaris/Linux機への移入手順フローを示します。表2.2に、相手側機種で使用するコマンドおよびユーティリティを示します。

手順1

Solaris/Linux機側で外部媒体への出力、またはファイル転送を行います。
ファイル転送の場合、テキストモードでの転送は行わず、バイナリモード(透過モード)での転送を行うようにしてください。
外部媒体への出力方法として"FDA制御オプション"等を使用します。

手順2

相手側機種上での移入では、外部媒体またはファイル転送の移入先として直接扱えないファイル編成もあるため、一旦順編成ファイルとして移入します。区分編成ファイルやライブラリ上のソースへ移入するには、1メンバ単位に指定します。

手順3

必要に応じて、順編成ファイルから対象のファイル編成へ変換します。

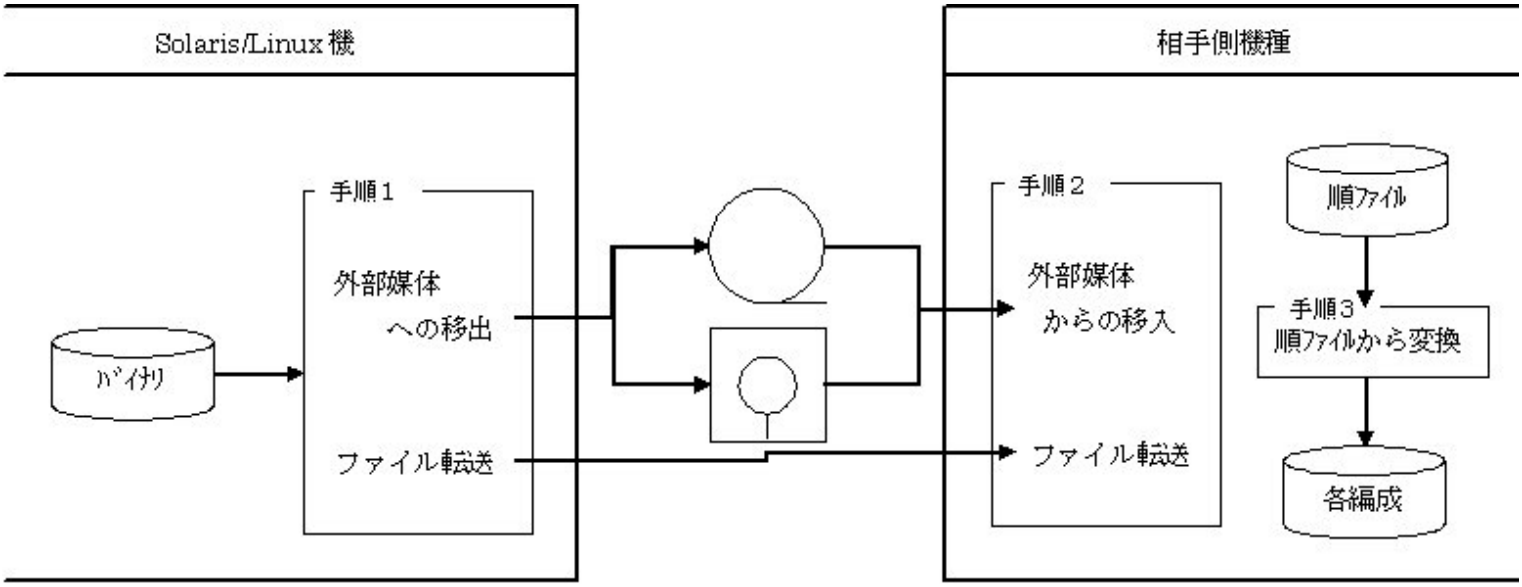


図2.2 相手側機種への移出手順フロー

表2.2 相手側機種への移出で使用するコマンド・ユーティリティ

相手側			順ファイルからの変換方法	外部媒体からの入力方法	ファイル転送手段
機 種	OS	ファイル編成			
GS/Mシリーズ	MSP	GEMライブラリ	JQHGEM3/ADDPS	JSDGENER, JSDFLPDK/CNVFD,等	F6680エミュレータ, HICS, ftp,等
		VSAM(KSDS)	KQCAMS/REPRO		
	XSP	GEMライブラリ	JQHGEM3/ADDPS	FCPY, FDCV/CNVB,等	
		VSAM(KSDS)	KQCAMS/REPRO		
PRIMERGY 6000 Kシリーズ	ASP	ライブラリソース	不要	IMPSRC	F6680エミュレータ, HICS, ftp,等
		SF	不要	CNVEXTF	
		PF	SETPF		
FM G	SX/G	原始テキスト	不要	オブジェクト一覧の"移入"	ftp,等
		順,物理	不要	impfile	
PC	DOS/Windows	DOSファイル	不要	COPYコマンド,等	ftp,等

第3章 使用手引

本章では、MDPORTの使用方法について説明します。入力または出力となる非Solaris/Linux側機種の手順については、"第2章 相手側交換手順"を参照してください。
MDPORTは、Solaris/Linux上のコマンドとして起動されます。利用者は、コマンドのオペランド上に、入出力ファイル名や変換方法等を指定します。オペランドの指定以外に利用者が定義する、定義情報ファイルを以下に示します。

- データ定義文ファイル
MDPORTのファイル変換機能で、入力または出力のファイル形式がデータファイル変換の場合に使用します。データファイルのレコードフォーマットを、COBOL文法のデータレコード記述を使用して記述したテキストファイルで定義します。COBOL文法は、MDPORTでサポートしているANSI'85規定範囲内で記述します。詳細は、"3.2 データ定義文の入力形式"を参照してください。

- 利用者定義変換テーブル
MDPORTのコード変換機能では変換できない、JIS水準外の拡張文字や利用者定義文字のコードを、本ファイルにより利用者が定義した変換テーブルに従った規則で変換します。詳細は、"3.4 利用者定義変換テーブルの入力形式"を参照してください。

3.1 コマンド

本節では、MDPORTのコマンド入力形式とその機能について説明します。MDPORTで用意されているコマンドは、以下のとおりです。

- mdportfコマンド

  ファイル形式の変換を行います。汎用機型のファイル形式や、テキストファイル、COBOLファイル、RDBローダ型ファイルでの形式変換を行います。ファイル形式変換にともない、データ属性に合わせたコード変換も行います。

  各コマンドは起動する環境の文字コード体系が、EUCロケール/シフトJISロケールにより、出力メッセージおよびオペランドのデフォルトが異なるので注意してください。本文中にはEUCロケール/シフトJISロケールで表現しています。

  [Linux] 起動する環境の文字コード体系について、シフトJISロケールは未サポートです。

  [Linux for Itanium] [x64-Linux] 起動する環境の文字コード体系はUnicodeロケールのみです。

  各コマンドは次の形式で説明します。

[名 前]

コマンドの名称と機能の概要を以下の形式で示します。

コマンド名-機能の概要

[形 式]

コマンドの記述形式を示します。記述形式は以下の規約で記述しています。

- 通常の文字で記述されている語は、そのとおりに入力することを示しています。
- 形式中の日本語の語は、置き換えて入力することを示しています。
- { }で囲まれている部分は、その括弧中の一つを明に指定する必要があることを示しています。
- [ ]で囲まれている部分は、省略可能であることを示しています。また、省略された場合は、括弧中の下線のある語が選択されることを示しています。
- 前述の括弧中に"α|β"と語句がわかれている部分は、αおよびβが選択対象であることを示しています。

[機能説明]

コマンドの機能やオペランドについて説明します。

[注 意]

コマンドを使用するうえで注意すべき事項を記述しています。

3.1.1 mdportfコマンド

[名 前]

mdportf - ファイル形式の変換

[形 式]

mdportf { -f (ファイル名) |
[-if (入力ファイル名)] [-of (出力ファイル名)]
[-msg (メッセージ出力先ファイル名)]
[-formatf (データ定義文ファイル名)]
} [-rep]
-itype { text | cobolsrc | data | cblfile | rdb }
[-otype { text | data | cblfile | rdb }]
[-msgmax { 50 | no | (メッセージ出力件数) }] [-msgcont]
[-icode { euc | sjis | utf8 | ucs2b | ucs2l | cobub | cobul | jis | jise | jis78 | jis78e | jef | (他社コード) }]
[-ocode { euc | sjis | utf8 | ucs2b | ucs2l | cobub | cobul | jis | jise | jis78 | jis78e | jef | (他社コード) }]
[-iconv { yes | no }] [-emu | -iso]
[-cvtable (利用者定義変換テーブルファイル名)]
[-amend1 (1バイト系代替コード)] [-amend2 (2バイト系代替コード)]
[-delimita { "," | no | (文字列) }] [-oracle | -rdb2 | -informix]
[-irecdlm { lf | crlf }] [-orecdlm { lf | crlf }]
[-ikana { euc | jis8 }] [-okana { euc | jis8 | eucG1 }]
[-shift { left | blank }] [-nocs3] [-noblc] [-noconv]
[-icobfl { seq | lsq | rel | idx [, { 1 | キー位置 } , { 1 | キー長 } [, dup]] }]
[-ocobfl { seq | lsq | rel | idx [, { 1 | キー位置 } , { 1 | キー長 } [, dup] [, { s | r }]] }]
[-cvcobgen | -cvcobuse] [-lfs]
[-irecfm { f | v }] [-orecfm { f | v }]
[-ireclen (入力レコード長)] [-oreclen (出力レコード長)]
[-icopy (項目入換え元登録集) -ocopy (項目入換え先登録集)]
[-nosignature]
[-validation { yes | no }] [-policyf (バリデーションポリシーファイル名)]

[機能説明]

入力ファイルを指定された出力ファイル形式に変換して出力します。
形式変換は、入力ファイル形式の指定により以下の二つの種類に分類されます。

- テキストモード変換
COBOLソースや各種制御文ソース等の、テキスト形式間でのファイル変換を行います。

- データファイル変換
データ定義文ファイル上の定義情報をもとに、レコードフォーマットの1項目単位に識別・コード変換し、データファイルや各種ファイル形式との変換を行います。

mdportfコマンドの復帰値を、表3.1に示します。正常に処理が終了した場合、mdportfコマンドでは入出力のレコード件数を標準エラー出力へ出力します。データファイル変換を行う場合、データ定義文の解析後、定義した入力/出力のレコード長が標準エラー出力へ出力されてから変換処理が行われます。

表3.1 mdportfコマンドの復帰値

復帰値	意味
0	正常終了
1	正常終了(警告エラーあり)
(上記以外の値)	異常終了

各オペランドには、入力・出力ファイル形式(-itype, -otype)および入力・出力コード体系(-icode, -ocode)の組み合わせにより、指定をしても意味を持たない場合があります。表3.2に各オペランドが有効となる組み合わせを示します。

表3.2 ファイル形式/コード体系に関連するオペランドの組み合わせ

		有効となる組み合わせ条件	

オペランド	設定値	-itype	-icode	-otype	-ocode	備考
-emu -iso	-	-	euc sjis utf8, ucs2b, ucs2l cobub, cobul	-	jef 他社	EBCDIC系とASCII系とのコード変換にて有効。
		-	jef 他社	-	euc sjis utf8, ucs2b, ucs2l cobub, cobul	
-delimita	文字列 "," no	rdb	-	-	-	デフォルト：","
	文字列 "," no	-	-	rdb	-	
		data	-	text	-	デフォルト：no
-rdb2 -oracle -informix	-	rdb	-	-	-	デフォルト：-oracle
		-	-	rdb	-	
-irecdlm	lf crlf	data	euc sjis utf8, ucs2b, ucs2l cobub, cobul	-	-	デフォルト：指定なし
		text rdb	euc sjis utf8, ucs2b, ucs2l	-	-	デフォルト：lf
-orecdlm	lf crlf	-	-	data	euc sjis utf8, ucs2b, ucs2l cobub, cobul	デフォルト：指定なし
		-	-	text rdb	euc sjis utf8, ucs2b, ucs2l	デフォルト：lf
-ikana	euc jis8	data cblfile	euc	- -	- -	デフォルト：euc
-okana	euc jis8	- -	- -	data cblfile	euc	
	eucG1	-	-	data cblfile rdb text	euc	
-shift	left blank	data	jef 他社	data cblfile text	euc sjis utf8, ucs2b, ucs2l cobub, cobul	デフォルト：left
-nocs3	-	-	euc	-	-	デフォルト：指定無し
	-	-	-	-	euc	
-noblc	-	data rdb text cblfile	-	-	-	デフォルト：設定無し
-icobfl	seq lsq rel idx	cblfile	euc sjis cobub, cobul	-	-	デフォルト：seq
-ocobfl	seq lsq rel idx	-	-	cblfile	euc sjis cobub, cobul	デフォルト：seq
-cvcobgen	-	-	-	cblfile	euc sjis cobub, cobul	-cvcobuseおよび-fの指定はできない。
-cvcobuse	-	-	-	cblfile	euc sjis cobub, cobul	-cvcobgenおよび-fの指定はできない。

-irecfm	f v	data cblfile	-	-	-	デフォルト：f
		text	jef 他社	-	-	
-orecfm	f v	-	-	data cblfile	-	デフォルト：f
		-	-	text	jef 他社	
-ireclen	1～32767	text	jef 他社	-	-	デフォルト：80
-oreclen	1～32767	-	-	text	jef 他社	デフォルト：80
-icopy -ocopy	-	-	-	cblfile	-	-icopy、-ocopyを同時に指定しなければならない。
-nosignature	-	-	-	rdb text	utf8, ucs2b, ucs2l	デフォルト：指定無し

備考. 設定できる値の組み合わせを示す。この表にない組み合わせは指定しても無視される。

- ：設定値に依存しない、または設定値がないことを示す。

3.1.2 オペランド

指定できるオペランドについて説明します。なお、各オペランドは任意の順序で指定することができます。全てのオペランドを省略した場合は、使用方法が標準出力へ出力されます。

-f (ファイル名)

変換元の入力ファイル名を指定します。-fオペランドを指定した場合の仕様は以下となります。

- 入力ファイル名に従い、出力ファイル名/データ定義文ファイル名、メッセージ出力先ファイル名を決定します。詳細は表3.3を参照してください。
- 入力ファイル名にはワイルドカードを指定することができます。データファイル変換でワイルドカードを使用した場合、ファイルの拡張子が".format"のファイル名は、入力ファイルの対象とならずデータ定義文ファイルとして扱います。
- 出力ファイルが既に存在し、かつ-repオペランドが指定されていない場合は、その入力ファイルは処理されません。
- メッセージ出力ファイルが既に存在している場合は、上書きされます。
- -f オペランドを指定して実行すると、入力ファイルごとに開始メッセージが標準エラー出力へ出力されます。

表3.3 -f オペランド指定で付加される拡張子

使用されるファイル	付加される拡張子	例(-f ABCD)
入力ファイル名	(なし)	ABCD
出力ファイル名	.out	ABCD.out
データ定義文ファイル名(注1)	.format	ABCD.format
メッセージ出力先ファイル名(注2)	.msg	ABCD.msg

注1) データファイル変換を行う場合に使用されます。
注2) 警告メッセージが発生した場合に出力されます。
注3)ワイルドカードを指定した場合、拡張子「.out」のファイルは入力対象外です。

[オペランド有効条件]
特になし。

[オペランド省略時]
-if, -of, -formatf, -msgを使用して任意のファイル名を指定します。その場合は、ワイルドカードは指定できません。

-if (入力ファイル名)

任意の変換元の入力ファイル名をパス名で指定します。

[オペランド有効条件]
特になし。

[オペランド省略時]
-f オペランドを指定しない場合に省略すると標準入力となります。
-itype cblfile指定の場合、本オペランドは省略できません。

-of (出力ファイル名)

任意の変換後の出力ファイル名をパス名で指定します。
-repオペランドが指定されていないと、既に存在するファイルは指定できません。

[オペランド有効条件]
特になし。

[オペランド省略時]
-f オペランドを指定しない場合に省略すると標準出力となります。
-otype cblfile指定で-cvcobgenを指定していない場合、本オペランドは省略できません。

-msg (メッセージ出力先ファイル名)

任意のメッセージ出力先ファイル名をパス名で指定します。出力先ファイルが既に存在している場合は内容が上書きされるため、使用には注意してください。メッセージ出力先ファイルが他外部ファイル指定パラメタと重複している場合には、実行を行いません。
メッセージ出力先ファイルはmdportfコマンド実行後に出力するファイルです。出力される警告メッセージの種類については、"C.2 警告メッセージ"を参照してください。

[オペランド有効条件]
特になし。

[オペランド省略時]
-f オペランドを指定しない場合に省略すると警告メッセージは標準エラー出力に出力されます。

-formatf (データ定義文ファイル名)

任意のデータ定義文のファイル名を指定します。
データ定義文の記述方法については、"3.2 データ定義文の入力形式"を参照してください。

[オペランド有効条件]
データファイル変換を行う場合。

[オペランド省略時]
-f オペランドを指定しない場合で、かつ、データファイル変換を行う場合に省略することはできません。

-rep

出力ファイルが既に存在する場合に、上書きして出力します。
本オペランドを指定すると、コマンドが異常終了した場合も元の出力ファイルの内容は保証されません。

[オペランド有効条件]
特になし。

[オペランド省略時]
既に、出力ファイルが存在している場合、処理を中止します。

-itype { text | cobolsrc | data | cbfile | rdb }

入力ファイル形式を指定します。指定可能なファイル形式と意味については下記を参照してください。
textまたはcobolsrcが指定された場合はテキストモード変換として動作します。テキストモード変換での出力ファイル形式の指定は、テキストファイル(-otype text)のみとなります。data、cblfileまたはrdbが指定された場合はデータファイル変換として動作します。

オペランド	ファイル形式	ファイル形式の説明
text	テキストファイル	文字キャラクタのみで構成されたファイルを示します。コード体系によりレコードの単位が異なり、ASCII系コードでは改行コードをレコードの区切りとします。
cobolsrc	COBOL固定長ソース	EBCDIC系コードで記述された、80バイト固定長レコードのCOBOLソースファイルを示します。入力ファイル形式としてのみ指定でき、出力ファイル形式としては指定できません。
data	データファイル	バイナリのデータファイルを示します。汎用機上の順ファイルを、ファイル転送または外部媒体経由で移入したファイルが、この形式に該当します。レコードフォーマットは、データ定義文により識別されます。
cblfile	COBOLファイル	Solaris/Linux上のCOBOLファイルシステムを示します。
rdb	RDBローダ型ファイル	データ項目間に区切り文字を挿入した、CSV形式のテキストファイルを示します。EUC、シフトJISコードまたはUnicodeの文字キャラクタのみで構成され、改行コードをレコードの区切りとします。

入力ファイル形式の指定により、指定できるコード体系、出力ファイル形式が異なります。表3.4にテキストモード変換で指定できる変換パスを、表3.5にデータファイル変換で指定できる変換パスを示します。

[オペランド有効条件]
特になし。

[オペランド省略時]
省略することはできません。

表3.4 テキストモード変換で指定できる変換パス

-otype -ocode / -itype -icode		text				
		euc	sjis	utf8,ucs2b,ucs2l	jef	他社
text	euc	○	○	○	○	○
	sjis	○	○	○	○	○
	utf8,ucs2b,ucs2l	○	○	○	○	○

	jef	○	○	○	-	-
	他社	○	○	○	-	-
cobolsrc	jef	○	○	-	-	-
	他社	○	○	-	-	-

○ : 指定可 - : 指定不可
備考. この表に載っていない指定はできません。

表3.5 データファイル変換で指定できる変換パス

【x64-Linux版 以外の場合】

-otype -ocode / -itype -icode		data							cblfile			rdb			text		
		euc	sjis	jis	utf8 ucs2b ucs2l	cobub cobul	jef	他社	euc	sjis	cobub cobul	euc	sjis	utf8 ucs2b ucs2l	euc	sjis	utf8 ucs2b ucs2l
data	euc	○	○	○	○	○	○	○	○	○	△	○	○	○	○	○	○
	sjis	○	○	○	○	○	○	○	○	○	△	○	○	○	○	○	○
	jis	○	○	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	utf8 ucs2b ucs2l	○	○	-	○	○	○	○	○	○	△	○	○	○	○	○	○
	cobub cobul	○	○	-	○	○	○	○	○	○	△	○	○	○	○	○	○
	jef	○	○	-	○	○	-	-	○	○	△	○	○	○	○	○	○
	他社	○	○	-	○	○	-	-	○	○	△	○	○	○	○	○	○
cblfile	euc	○	○	-	○	○	○	○	○	○	-	○	○	○	○	○	○
	sjis	○	○	-	○	○	○	○	○	○	-	○	○	○	○	○	○
	cobub cobul	△	△	-	△	△	△	△	-	-	-	△	△	△	-	-	-
rdb	euc	○	○	-	○	○	○	○	○	○	△	-	-	-	-	-	-
	sjis	○	○	-	○	○	○	○	○	○	△	-	-	-	-	-	-
	utf8 ucs2b ucs2l	○	○	-	○	○	○	○	○	○	△	-	-	-	-	-	-

○　：指定可

-　：指定不可

△　：
[Solaris] cobubのみ指定可
[Linux] cobulのみ指定可
[Linux for Itanium] cobub, cobul共に指定可

この表に載っていない指定はできません。
備考：UnicodeのCOBOLファイルを扱う場合には、動作環境のコードがUnicodeロケールである必要があります。

【x64-Linux版 の場合】

-otype -ocode / -itype -icode		data							cblfile				rdb			text		
		euc	sjis	jis	utf8 ucs2b ucs2l	cobub cobul	jef	他社	euc	sjis	cobub	cobul	euc	sjis	utf8 ucs2b ucs2l	euc	sjis	utf8 ucs2b ucs2l
data	euc	○	○	○	○	○	○	○	-	○	△		○	○	○	○	○	○
	sjis	○	○	○	○	○	○	○	-	○	△		○	○	○	○	○	○
	jis	○	○	-	-	-	-	-	-	-	-		-	-	-	-	-	-
	utf8 ucs2b ucs2l	○	○	-	○	○	○	○	-	○	△		○	○	○	○	○	○
	cobub cobul	○	○	-	○	○	○	○	-	○	△		○	○	○	○	○	○
	jef	○	○	-	○	○	-	-	-	○	△		○	○	○	○	○	○
		○	○	-	○	○	-	-	-	○			○	○	○	○	○	○

	他社										△							
cblfile	euc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	sjis	○	○	-	○	○	○	○	-	○	-	○	○	○	○	○	○	○
	cobub	△	△	-	△	△	△	△	-	-	△	-	△	△	△	△	△	△
	cobul										-	△						
rdb	euc	○	○	-	○	○	○	○	-	○	△		-	-	-	-	-	-
	sjis	○	○	-	○	○	○	○	-	○	△		-	-	-	-	-	-
	utf8 ucs2b ucs2l	○	○	-	○	○	○	○	-	○	△		-	-	-	-	-	-

○　：指定可

-　：指定不可

△　：cobub, cobul共に指定可

備考：この表に載っていない指定はできません。
UnicodeのCOBOLファイルを扱う場合には、動作環境のコードがUnicodeロケールである必要があります。
[x64-Linux] ロケールがja_JP.UTF-8以外はサポート外となります。

-otype { text | data | cblfile | rdb }

出力ファイル形式を指定します。指定可能なファイル形式と意味については-itypeオペランドと同じです。
ただし、表3.4または表3.5のように、-itypeオペランドの指定により、本オペランドで指定できるファイル形式が異なります。

[オペランド有効条件]
特になし。

[オペランド省略時]
データファイル変換では省略できません。
テキストモード変換では-otype textとして扱います。

-icode { euc | sjis | utf8 | ucs2b | ucs2l | cobub | cobul | jis | jise | jis78 | jis78e | jef | (他社コード) }

入力ファイル上のコード体系を指定します。指定可能なコード体系と意味を示します。
ファイル上に日本語が存在しないEBCDICコードのみの場合は、jefを指定してください。
表3.4または表3.5のように、-itypeオペランドの指定により、本オペランドで指定できるコード体系が異なります。

オペランド	コード体系
euc	ASCIIおよびEUCコード
sjis	JIS8およびシフトJISコード
utf8	UTF8形式
ucs2b	UCS2(ビッグエンディアン)形式
ucs2l	UCS2(リトルエンディアン)形式
cobub	Unicode形式のCOBOLファイル仕様に準拠。 X, M項目：UTF8形式 N項目：UCS2(ビッグエンディアン形式)
cobul	Unicode形式のCOBOLファイル仕様に準拠。 X, M項目：UTF8形式 N項目：UCS2(リトルエンディアン形式)
jis	ASCIIおよびJISコード
jise	EBCDICおよびJISコード
jis78	ASCIIおよび78JISコード
jis78e	EBCDICおよび78JISコード
jef	EBCDICおよびJEFコード
(他社コード)	EBCDICおよび他社日本語コード

他社コードには、以下の指定を行うことができます。

オペランド	コード体系
ibm	EBCDICおよびIBM漢字コード

keis	EBCDICおよび日立KEISコード
jipse	EBCDICおよび日本電気JIPS(E)コード
jipsj	EBCDICおよび日本電気JIPS(J)コード

[オペランド有効条件]
特になし。

[オペランド省略時]
EUCロケール配下では-icode euc、シフトJISロケール配下では-icode sjis、Unicodeロケール配下では-icode utf8とします。

-ocode { euc | sjis | utf8 | ucs2b | ucs2l | cobub | cobul | jis | jise | jis78 | jis78e | jef | (他社コード) }

出力ファイルのコード体系を指定します。指定可能なコード体系と意味は、-icodeオペランドと同じです。
表3.4または表3.5のように、-otypeオペランドの指定により、本オペランドで指定できるコード体系が異なります。

[オペランド有効条件]
特になし。

[オペランド省略時]
EUCロケール配下では-ocode euc、シフトJISロケール配下では-ocode sjis、Unicodeロケール配下では-ocode utf8とします。

-iconv { yes | no }

標準コード変換(iconv)を利用したコード変換の扱いについて指定します。
標準コード変換を利用した詳細キーワードの指定は環境変数で指定します。詳細は"3.5実行環境"を参照してください。

オペランド	意味
yes	標準コード変換を使用します。 ※-icode,-ocodeにjis, utf8, ucs2b, ucs2l, cobub, cobulを指定している場合、標準コード変換は必須です。
no	標準コード変換を使用しません。

[オペランド有効条件]
特になし。

[オペランド省略時]
yes指定として扱います。

-msgmax { 50 | no | (メッセージ出力件数) }

mdportfコマンド実行後に出力される警告メッセージの出力件数の上限値を指定します。
-msgcontオペランドが指定されていない場合に、警告メッセージの出力件数が上限値を超えるとコマンドは異常終了します。

オペランド	意味
no	警告メッセージを無制限に出力します。
0～9999	上限値範囲で出力します。

[オペランド有効条件]
特になし。

[オペランド省略時]
メッセージ出力件数の上限値を50とします。

-msgcont

本オペランドを指定すると、警告メッセージが上限値を超えて発生しても処理を続行します。
上限値を超えた分の警告メッセージは出力されません。

[オペランド有効条件]
特になし。

[オペランド省略時]
-msgmaxオペランドの指定に従います。

-emu | -iso

EBCDICコードとASCII(またはJIS8)コード間の変換方法を指定します。
-isoを指定した場合は、富士通のISOコード変換仕様に基づいた変換を行います。

-emu指定と-iso指定の相違点を表3.6に示します。

表3.6 -emuと-isoオペランドの相違点

変換パス	変換前	変換後	
		-emu指定	-iso指定
EBCDIC→ASCII	!	!	]
	|	]	!
ASCII→EBCDIC	!	!	|
	]	|	!
	(英小文字)	(英大文字)	(変換エラー)

[オペランド有効条件]
-icodeまたは-ocodeオペランドに、EBCDIC系コード(jef,他社コード)を指定している場合。

[オペランド省略時]
emu指定として扱います。両方指定することはできません。

-cvtable (利用者定義変換テーブルファイル名)

利用者定義変換テーブルを使用する場合に、定義ファイル名を指定します。利用者定義変換テーブルの詳細は、"3.4 利用者定義変換テーブルの入力形式"を参照してください。

[オペランド有効条件]
特になし。

[オペランド省略時]
iconvの変換仕様、または、mdportの変換仕様に従います。

-amend1 (1バイト系代替コード)

1バイト系コード(半角文字)の変換において、変換不可能な文字が発生した場合の代替文字を指定します。
変換不能が発生した場合は、警告メッセージが出力されます。
代替コードは出力先のコード体系に応じて、16進数2桁、または、4桁で指定します。指定方法を以下に示します。

出力先コード体系	指定桁数	補足事項
ucs2b, ucs2l	4桁	UCS2形式の場合、1バイト文字(半角文字)は2バイトで表現するため、16進4桁で指定する必要があります。
上記以外	2桁	なし。

[オペランド有効条件]
特になし。

[オペランド省略時]
変換不可能文字は"_"(アンダーバー)で出力します。

-amend2 (2バイト系代替コード)

2バイト系コード(マルチバイト文字)の変換において、変換不可能な文字が発生した場合の代替文字を指定します。
変換不能が発生した場合は、警告メッセージが出力されます。
代替コードは出力先のコード体系に応じて、16進数4桁、または、6桁で指定します。本オペランドを省略した場合、変換不可能文字は"■"(塗りつぶした四角文字)で出力されます。

出力先コード体系	指定桁数	補足事項
utf8	4桁 または 6桁	UTF8形式の場合、2バイト文字(マルチバイト文字)は2バイト、または、3バイトで表現するため、16進4桁、または、6桁で指定する必要があります。
cobub, cobul	4桁	cobub,cobulの -amend2はUCS2の2バイト(4桁)を指定します。 M項目に2バイト系代替コードを出す場合は指定された16進をUTF8に変換して出力しています。
上記以外	4桁	なし。

[オペランド有効条件]
特になし。

[オペランド省略時]
変換不可能文字は"■"(塗りつぶした四角文字)で出力します。

-delimita { "," | no | (文字列) }

RDBローダ型ファイルの項目間の区切り文字を指定します。

オペランド	意味・使用方法
no	区切り文字を使用しない(-itype rdbでは指定不可)。
(文字列)	指定文字列を区切り文字として使用します。 ・文字列は1バイトの1文字で指定します。 ・記号文字を指定する場合は、引用符(")で括って指定します。

[補足事項]
INFORMIX向けローダ型ファイルを出力する場合、データ上に区切り文字と同じ文字があると、ローダにて正常に処理できませんので注意が必要です。
区切り文字に'-'(ハイフン)を指定する事はできません。

[オペランド有効条件]
-itypeまたは-otypeオペランドがrdb、または、-itype dataかつ-otype textの場合。
※上記以外は無視されます。

[オペランド省略時]
-itypeまたは-otypeオペランドがrdbの場合、","(カンマ)として扱います。
-itype dataかつ-otype textの場合、noとして扱います。

-oracle | -rdb2 | -informix

RDBローダ型ファイルを扱う場合に、対象RDB向けローダ形式を指定します。
RDBローダ型ファイル変換の詳細は、"B.7 RDBローダ型ファイルの変換"を参照してください。
ローダ形式は1つだけ指定できます。指定可能なローダ形式と意味を示します。

オペランド	意味
-oracle	[Oracleのローダ形式] 文字型データは引用符(")でくくられます。データ中の引用符1文字(")を表す場合、引用符を2つ並べます。ヌル文字列も引用符で括ります。
-rdb2	[Symfowareのローダ形式] 文字型データは引用符(")でくくられます。データ中の引用符1文字(")を表す場合、引用符を2つ並べます。ヌル文字列は引用符で括りません。
-informix	[INFORMIXのローダ形式] 文字型データは引用符(")で括られません。

[オペランド有効条件]
-itypeまたは-otypeオペランドにrdbを指定した場合。
※上記以外は無視されます。

[オペランド省略時]
-oracleとして扱います。

-irecdlm { lf | crlf }

入力レコードの改行コードの有無または種類を指定します。
指定可能な改行形式と意味を示します。

オペランド	意味
設定なし	入力レコードには改行コードは付加されていないことを示します。
lf	入力レコードには1バイトのLF改行コード(0x0A)が付加されていることを示します。
crlf	入力レコードには2バイトのCR+LF改行コード(0x0D0A)が付加されていることを示します。

[オペランド有効条件]
-itypeがtext、dataまたはrdbで、かつ-icodeがeuc、sjis、または Unicode(ucs2b,ucs2l,utf8,cobub,cobul)の場合。

[オペランド省略時]
-itype dataであれば設定なし、それ以外ではlfとして扱います。

-orecdlm { lf | crlf }

出力レコードの改行コードの有無または種類を指定します。
指定可能な改行形式と意味を示します。

オペランド	意味
設定なし	出力レコードには改行コードは付加されていないことを示します。
lf	出力レコードには1バイトのLF改行コード(0x0A)を付加して出力することを示します。
crlf	出力レコードには2バイトのCR+LF改行コード(0x0D0A)を付加して出力することを示します。

[オペランド有効条件]
-otypeがtext、dataまたはrdbで、かつ-ocodeがeuc、sjis、または Unicode(ucs2b,ucs2l,utf8,cobub,cobul)の場合。

[オペランド省略時]
-otype dataであれば設定なし、それ以外ではlfとして扱います。

-ikana { euc | jis8 }

入力コード体系がEUCのデータファイルを扱う場合に、データ中の英数字項目1バイト系JISカタカナ(半角カナ)の扱いを指定します。
指定可能なコード体系と意味を示します。

オペランド	意味
euc	入力データ中の半角カナは、EUCコード体系のコードセット2で構成される(2バイト/1文字)
jis8	入力データ中の半角カナは、JIS8コードで構成される(1バイト/1文字)

[オペランド有効条件]
-icode euc、かつ、-itypeオペランドがdataまたはcblfileの場合。
英数字日本語混在項目では、常にEUCコード体系のコードセット2で扱います。

[オペランド省略時]
eucとして扱います。

-okana { euc | jis8 | eucG1 }

出力コード体系がEUCのデータファイルを扱う場合に、 出力データ中の英数字項目1バイト系JISカタカナ(半角カナ)の出力方法の扱いを指定します。

オペランド	意味	出力ファイル形式
euc	半角カナは、EUCコード体系のコードセット2で出力する(2バイト/1文字)	data、 cblfile
jis8	半角カナは、JIS8コードで出力する(1バイト/1文字)	
eucG1	半角カナは、EUCコード体系のコードセット1(全角カナ)で出力する(2バイト/1文字)	data、 cblfile、 rdb、 text

[オペランド有効条件]
英数字日本語混在項目では、jis8を指定した場合にEUCコード体系のコードセット2で出力されます。

[オペランド省略時]
eucとして扱います。

-shift { left | blank }

EBCDIC系コードからのデータファイル変換において、入力ファイル上の制御コード(日本語コードへのシフトコード、印刷用制御コード)の扱いを指定します。

オペランド	意味
left	制御コードを取り除き、フィールド内で左詰めします。
blank	制御コードを全て1バイト系の空白(0x20)へ置換します。

	-ocode ucs2b(ucs2l)の場合、0x0020(0x2000)へ置換します。

[オペランド有効条件]
-icodeオペランドがjef,他社コードの場合。
テキストモード変換および出力がRDBローダ型の場合、常に制御コードは取り除かれて後続文字を詰めます。(leftと同一)

[オペランド省略時]
leftとして扱います。

-nocs3

変換後または変換元のEUCコードとして、コードセット3(3バイト/1文字表現)を使用しない場合に指定します。
入力EUCデータにコードセット3の文字が存在すれば、変換エラーとなり2バイト系代替文字コードで出力します。
出力EUCデータにコードセット3の文字が存在すれば、2バイト系代替文字コードを出力します。
利用者定義変換テーブルにEUCコードセット3を定義した場合は、定義内容が有効となり代替文字コードで出力されません。

[補足事項]
JEFコードとの変換時に、運用として拡張文字を使用しない場合等に指定します。

[オペランド有効条件]
-icode -ocodeがeucの場合。

[オペランド省略時]
特になし。

-noblc

COBOLファイル/データファイル/RDBローダ型ファイル/テキストファイル中の文字項目を変換する際に、出力項目のデータの後続空白を削除しない場合に指定します。

[オペランド有効条件]
-itypeオペランドが cblfile、data、rdb、textの場合。

[オペランド省略時]
特になし。

-noconv

入力コードと出力コードが同一の場合で、コード変換を行わない場合に指定します。
入力データを出力データにそのままコピーしますが、後続空白を削除してコピーします。
入力ファイル形式と出力ファイル形式がデータファイルまたはCOBOLファイルの場合、すべての項目でコード変換を行いません。
それ以外のファイル形式では、文字項目のみコード変換を行いません。
EUCコード、シフトJISコード、Unicodeで指定可能です。
表3.13にテキストモード変換で指定できる変換パスを、表3.14にデータファイル変換で指定できる変換パスを示します。

[補足事項]
特殊指示Nを同時に指定した場合は、特殊指示Nは無効となります。

[オペランド有効条件]
-icodeオペランドと-ocodeオペランドが同一の場合。

[オペランド省略時]
特になし。

表3.13 テキストモード変換で指定できる同一コード無変換の変換パス

-otype -ocode / -itype -icode		text						
		euc	sjis	utf8	ucs2b	ucs2l	jef	他社
text	euc	○	-	-	-	-	-	-
	sjis	-	○	-	-	-	-	-
	utf8	-	-	○	-	-	-	-
	ucs2b	-	-	-	○	-	-	-
	ucs2l	-	-	-	-	○	-	-
	jef	-	-	-	-	-	×	-
	他社	-	-	-	-	-	-	×

cobolsrc	jef	-	-	-	-	-	×	-
	他社	-	-	-	-	-	-	×

○ : 指定可 × : 指定不可 - : 同一コードでない
備考. この表に載っていない指定はできません。

表3.14 データファイル変換で指定できる同一コード無変換の変換パス

【x64-Linux版 以外の場合】

-otype -ocode / -itype -icode		data										cblfile				rdb					text				
		euc	sjis	jis	utf8	ucs2b	ucs2l	cobub	cobul	jef	他社	euc	sjis	cobub	cobul	euc	sjis	utf8	ucs2b	ucs2l	euc	sjis	utf8	ucs2b	ucs2l
data	euc	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-
	jis	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	utf8	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○	-	-
	ucs2b	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○	-
	ucs2l	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○
	cobub	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-
	cobul	-	-	-	-	-	-	-	○	-	-	-	-	-	△	-	-	-	-	-	-	-	-	-	-
	jef	-	-	-	-	-	-	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	他社	-	-	-	-	-	-	-	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-
cblfile	euc	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-
	cobub	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-
	cobul	-	-	-	-	-	-	-	△	-	-	-	-	-	△	-	-	-	-	-	-	-	-	-	-
rdb	euc	○	-	-	-	-	-	-	-	-	-	○	-	-	-	×	-	-	-	-	×	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	×	-	-	-	-	×	-	-	-
	utf8	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×	-	-
	ucs2b	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×	-
	ucs2l	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×

○ ：指定可

△ ：[Solaris] 指定不可
[Linux][Linux for Itanium] 指定可

× ：指定不可

- ：同一コードでない

備考：この表に載っていない指定はできません。
UnicodeのCOBOLファイルを扱う場合には、動作環境のコードがUnicodeロケールである必要があります。
-icodeオペランド、-ocodeオペランドで指定できない文字コードは指定できません。

【x64-Linux版 の場合】

-otype -ocode / -itype -icode		data										cblfile				rdb					text				
		euc	sjis	jis	utf8	ucs2b	ucs2l	cobub	cobul	jef	他社	euc	sjis	cobub	cobul	euc	sjis	utf8	ucs2b	ucs2l	euc	sjis	utf8	ucs2b	ucs2l
data	euc	○	-	-	-	-	-	-	-	-	-	×	-	-	-	○	-	-	-	-	○	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-
	jis	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	utf8	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○	-	-
	ucs2b	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○	-
	ucs2l	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	○	-	-	-	-	○
	cobub	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-
	cobul	-	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-

	jef	-	-	-	-	-	-	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	他社	-	-	-	-	-	-	-	-	-	×	-	-	-	-	-	-	-	-	-	-	-	-	-	-
cblfile	euc	×	-	-	-	-	-	-	-	-	-	×	-	-	-	×	-	-	-	-	×	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	○	-	-	-	-	○	-	-	-
	cobub	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-
	cobul	-	-	-	-	-	-	-	○	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-
rdb	euc	○	-	-	-	-	-	-	-	-	-	×	-	-	-	×	-	-	-	-	×	-	-	-	-
	sjis	-	○	-	-	-	-	-	-	-	-	-	○	-	-	-	×	-	-	-	-	×	-	-	-
	utf8	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×	-	-
	ucs2b	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×	-
	ucs2l	-	-	-	-	-	○	-	-	-	-	-	-	-	-	-	-	-	-	×	-	-	-	-	×

○　：指定可

×　：指定不可

-　：同一コードでない

この表に載っていない指定はできません。

備考：UnicodeのCOBOLファイルを扱う場合には、動作環境のコードがUnicodeロケールである必要があります。

-icodeオペランド、-ocodeオペランドで指定できない文字コードは指定できません。

[x64-Linux] ロケールがja_JP.UTF-8以外はサポート外となります。

-icobfl { seq | lsq | rel | idx [, { 1 | キー位置 } , { 1 | キー長 } [, dup]] }

入力ファイルがSolaris/Linux上のCOBOLファイルの場合に、ファイル編成を指定します。

例)mdportf -if data1.cbl -of data1.out -formatf data1.format -itype cblfile -otype data -icode jef -ocode euc -icobfl lsq

オペランド	ファイル形式	説明
seq	レコード順ファイル	レコード順ファイルに変換する場合に指定します。 デフォルトです。
lsq	行順ファイル	行順ファイルに変換する場合に指定します。
rel	相対ファイル	相対ファイルに変換する場合に指定します。
idx, キー位置, キー長, [dup]	索引ファイル	idxオペランドに続けてキー位置、キー長をカンマで区切って指定します。 キー位置：レコードの先頭を1とした相対バイト位置 キー長：キー位置からのバイト数 [dup]：重複キーを持つ索引ファイルの場合に指定

[補足事項]

キーが連続した状態でない場合、または副キーのある索引ファイルは扱えません。

[オペランド有効条件]

-itypeオペランドが cblfileの場合。

※それ以外は無視されます。

[オペランド省略時]

seqとして扱います。

idx指定後のキー位置とキー長を省略した場合、キー位置を1、キー長を1、重複指定無しとして扱います。

-ocobfl { seq | lsq | rel | idx [, { 1 | キー位置 } , { 1 | キー長 } [, dup] [, { s | r }]] }

出力ファイルがSolaris/Linux上のCOBOLファイルの場合に、ファイル編成を指定します。

以下にオペランドの説明を示します。

オペランド	ファイル形式	説明
seq	レコード順ファイル	レコード順ファイルに変換する場合に指定します。 デフォルトです。
lsq	行順ファイル	行順ファイルに変換する場合に指定します。
rel	相対ファイル	相対ファイルに変換する場合に指定します。
		idxオペランドに続けてキー位置、キー長をカンマで区切って指定します。 キー位置：レコードの先頭を1とした相対バイト位置 キー長：キー位置からのバイト数

<table>
<tr><td>idx,
キー位置,
キー長,
[dup],
[s|r]</td><td>索引ファイル</td><td>[dup]</td><td>重複キーを持つ索引ファイルの場合に指定</td></tr>
<tr><td></td><td></td><td>[s | r]</td><td>ファイルのアクセス方法を指定
<table><tr><td>s</td><td>[SEQUENTIAL](デフォルト)
順呼び出し：指定入力ファイルのレコードがキー順にソートされている場合に指定</td></tr><tr><td>r</td><td>[RANDOM]
乱呼び出し：指定入力ファイルのレコードがキー順にソートされていない場合に指定</td></tr></table></td></tr>
</table>

[補足事項]
キーが連続した領域でない場合、または副キーのある索引ファイルは扱えません。

[オペランド有効条件]
-otypeオペランドが cblfileの場合。
※それ以外は無視されます。

[オペランド省略時]
seqとして扱います。
idx指定後のキー位置とキー長を省略した場合、キー位置を1、キー長を1、重複指定無し、順呼び出し(SEQUENTIAL)として扱います。

-cvcobgen

Solaris/Linux上のCOBOLファイルへの出力を行うためのCOBOLプログラムの生成を行います。詳細説明や活用方法については、"B.6 COBOLファイルの変換"を参照してください。
本オペランドを指定すると、カレントディレクトリ上に"MDPORTCV.cobol"という名前でCOBOLソースファイルが保存されて、変換処理をせずにコマンドは終了します。保存したプログラムを利用して変換処理を行う場合は、-cvcobuseオペランドを指定します。

[オペランド有効条件]
-otypeオペランドが cblfileの場合。
※それ以外は無視されます。
※-cvcobuseオペランドと同時に指定することはできません。
※-fオペランドによるファイル名指定はできません。データ定義文ファイルは、-formatfオペランドで指定してください。
※当指定は、COBOLプログラムを生成するためのものであるため変換処理は行いません。従って、-ifや-ofオペランドの入出力ファイルは意味を持ちません。

[オペランド省略時]
特になし。
※生成COBOLソースファイルは変換処理終了後削除されます。

-cvcobuse

-cvcobgenオペランドにより保存した変換プログラムを利用して、変換処理を行う場合に本オペランドを指定します。詳細説明や活用方法については、"B.6 COBOLファイルの変換"を参照してください。
本オペランドを指定すると、変換プログラムの生成処理は迂回され、カレントディレクトリ上の"MDPORTCV.cobol"の名前のCOBOLソースファイルを翻訳して変換処理が行われます。

[オペランド有効条件]
-otypeオペランドが cblfileの場合。
※それ以外は無視されます。
※-cvcobgenオペランドと同時に指定することはできません。
※-fオペランドによるファイル名指定はできません。

[オペランド省略時]
特になし。
※COBOLソースの生成処理から行われます。

-lfs [Solaris], [Linux]

入出力ファイルがSolaris/Linux上の1GBを超えるCOBOLラージファイルの場合に指定します。

[オペランド有効条件]
特になし。

[オペランド省略時]
1GB以下のCOBOLファイルとして扱います。

-irecfm { f | v }

入力ファイルのレコード属性を指定します。指定可能なレコード属性と意味を示します。

オペランド	意味
f	固定長レコード
v	可変長レコード

[補足事項]
可変長(vのとき)を指定すると、入力レコードの先頭4バイトにRDW(Record Descriptor Word)が必要となります。RDWは、先頭2バイトに2進数値でレコード長が表されているものとして扱います。

[オペランド有効条件]
-itype data、-itype cblfileまたはEBCDIC系コードのテキストファイルの場合。
※ASCII系コードのテキストファイル、またはRDB型ローダファイルでは、改行コードをレコードの区切りとするため、本オペランドは無効です。

[オペランド省略時]
fとして扱います。

-orecfm { f | v }

出力ファイルのレコード属性を指定します。指定方法は-irecfmと同じです。

[補足事項]
可変長(vのとき)を指定すると、出力レコードの先頭4バイトにRDW(Record Descriptor Word)が付加されます。RDWには、先頭2バイトに2進数値でレコード長を表しています。

[オペランド有効条件]
-otype data、-otype cblfileまたは出力がEBCDIC系コードのテキストファイルの場合。
※ASCII系コードのテキストファイル、またはRDB型ローダファイルの出力では、改行コードをレコードの区切りとするため、本オペランドは無効です。

[オペランド省略時]
fとして扱います。

-ireclen (入力レコード長)

入力ファイルがEBCDIC系テキストファイルの場合のレコード長を指定します。
各ファイル形式の入力レコード長の判断を以下に示します。

入力データ形式	レコード長の判断
EBCDIC系テキストファイル(固定長)	-ireclenオペランドで指定
EBCDIC系テキストファイル(可変長)	RDWよりレコード長を取得
COBOLソースファイル	80バイト固定
データファイル(固定長)	データ定義文より取得
データファイル(可変長)	RDWよりレコード長を取得
COBOLファイル(固定長)	データ定義文より取得
COBOLファイル(可変長)	COBOLファイルアクセス時に取得
RDBローダ型ファイル	改行コードをレコードの区切りとする

[オペランド有効条件]
-itype textかつ、コード体系がEBCDIC系の場合。

[オペランド省略時]
80バイト固定のレコード長として扱います。

-oreclen (出力レコード長)

出力ファイルがEBCDIC系コードの固定長テキストファイルの場合のレコード長を指定します。

[オペランド有効条件]
-otype textかつ、コード体系がEBCDIC系の場合。

[オペランド省略時]
80バイト固定のレコード長として扱います。

-icopy (項目入換え元登録集) -ocopy (項目入換え先登録集)

項目入換え機能の、入換え元となる登録集と入換え先となる登録集を指定します。項目入換え機能の詳細については"3.3 項目入換え機能"を参照してください。登録集は、拡張子に.cob、.cbl、.cobolのいずれかを用い、本オペランドには拡張子を除いた名前で指定します。また、レベル01番号の項目名と登録集名(拡張子を除いたもの)は同じである必要があります。

[補足事項]
本機能はCOBOLのMOVE CORRESPONDING文により行われ、詳細仕様はCOBOLに準じます。
指定登録集がカレントディレクトリ上にない場合は、以下の方法で相対位置を指定することで運用可能です。
　環境変数 COBCOPYを指定し、パスを設定する。
なお、当環境変数はCOBOLの環境変数です。詳細は、COBOLの仕様手引書を参照してください。

[オペランド有効条件]
出力がCOBOLファイル(-otype cblfile)の場合。
※-ocode utf8、ucs2b、ucs2l、jef、jisの場合は動作しません。
-icopy、-ocopyオペランドを同時に指定します。

[オペランド省略時]
特になし。

-nosignature

出力ファイルがUnicode系のテキストファイルまたはRDBローダ形式の場合、シグネチャ出力を省略する場合に指定します。

[オペランド有効条件]
-otype text、または、rdbを指定した場合でかつ、-ocode utf8、ucs2b、ucs2lの場合。

[オペランド省略時]
出力ファイルがUnicode系のテキストファイルまたはRDBローダ形式の場合、-ocodeオペランドに対応したシグネチャを付加して出力します。

-validation { yes | no }

バリデーション機能を適用した変換を行うかどうかを -validation オペランドで指定します。

実際の使用の際には、-policyfオペランド、環境変数の指定により実行します。

バリデーション機能(Charset Validator)を適用した変換について詳細な説明は、"3.7 バリデーション機能(Charset Validator)を適用した変換"を参照してください。

-validation オペランドに関係した環境変数については、"バリデーション機能(Charset Validator)を適用した変換を行う際の環境変数"を参照してください。

-validation オペランドに続けて指定するパラメタとその意味を下表に説明します。

パラメタ	パラメタの意味	注意点
yes	バリデーション機能を適用した変換を行います。 入力または出力コード(-icode又は、-ocode) に指定した文字コード(Unicode)に対して、-policyfオペランド、もしくは、環境変数で指定したバリデーションポリシーファイルを適用した変換を行います。	バリデーション機能を適用した変換はUnicodeに対してのみ指定が可能です。入力コード(-icode)、出力コード(-ocode)ともUnicodeの場合は、入力コード(-icode) に対してバリデーション機能を適用した変換を行います。
no	バリデーション機能を適用した変換を行いません。	オペランド省略時のデフォルト指定となります。 オペランドを一時的に無効としたいときなどに指定します。

[オペランド有効条件]
「-noconv」オペランドが指定されていない場合。

[オペランド省略時]
no 指定として扱います。

-policyf (バリデーションポリシーファイル名)

バリデーション機能を適用した変換を行う際、バリデーションポリシーファイルを「-policyf」オペランドで指定します。

実際の使用の際には、-validation オペランド、環境変数の指定により実行します。

バリデーション機能(Charset Validator)を適用した変換の詳細な説明は、"3.7 バリデーション機能(Charset Validator)を適用した変換"を参照してください。

-policyf オペランドに関係した環境変数については、"バリデーション機能(Charset Validator)を適用した変換を行う際の環境変数"を参照してください。

-policyf オペランドに続けて指定するパラメタとその意味を下表に説明します。

パラメタ	パラメタの意味	注意点
ファイル名	バリデーション機能を適用して変換する際に使用するバリデーションポリシーファイル名です。	ファイル名、もしくは相対パスを指定した場合は、環境変数 MDPORT_VALIDATION_PATH に値を設定します。この環境変数の設定がない場合には、カレントからのファイル名、もしくは相対パスとして扱います。 また、 -policyf オペランドにフルパスで指定した場合には、環境変数 MDPORT_VALIDATION_PATH に値を設定してはいけません。 また、パラメタの省略は、不可です。

[オペランド有効条件]
-validation yes が指定された場合、もしくは環境変数 MDPORT_VALIDATION で yes が指定されている場合。

[オペランド省略時]
環境変数(MDPORT_VALIDATION_PATH, MDPORT_VALIDATE_UNICODE) に指定された値を参照します。

[注意]

- 非EUCコードからEUCコードへのデータファイル変換にて、入力データの英数字項目内にJISカタカナが存在するケースでは、領域長変動(カナ1バイト/文字→EUCカナ2バイト/文字)により項目内に収まらず、切り捨てられることが多く発生します。この場合、データ定義文特殊指示により出力項目長を変更するか、JIS8コードによる運用(-okana jis8指定)等の対応が考えられます。
- データファイル変換にて、RDBローダ型またはテキストファイルへ出力した場合、出力データ中に文字として認識できないコードであっても検査されません。RDBローダ起動にてローダ型ファイル中に文字以外が存在すると、正常に処理できないので注意が必要です。

3.2 データ定義文の入力形式

本節では、mdportfコマンドでデータファイル変換を行う際必要となる、レコードフォーマットを定義するデータ定義文の入力形式について説明します。
データ定義文で指定するレコードフォーマットは、入力または出力となるデータファイル(およびCOBOLファイル)に対して定義します。例えば、入力がRDBローダ型ファイルで、出力がデータファイルの場合は、出力するデータファイルのレイアウトに対して定義します。RDBローダ型ファイルの場合、項目ごとの領域長は可変のため、データ定義文で定義した項目長は意味を持ちません。
データ定義文はCOBOL言語のデータ記述項のイメージで記述します。

[テキスト構成]

データ定義文は、可変長のテキストファイルに記述し、1行の長さは251バイト以下です。図3.1にデータ定義文の正書法を示します。
データ定義文中のタブは、1つの空白としてみなされます。そのため、見かけ上のカラム位置と解析上のカラム位置とは異なります。
各々の領域の意味は以下のとおりです。

一連番号領域(1～6カラム)

行番号を記述します。ただし、MDPORTでは意味を持たないため、空白でも構いません。
この領域に"######"の記述があると、その行はフォーマット指示制御文とみなします。フォーマット指示制御文については、"3.2.1 フォーマット指示制御文によるマルチフォーマット定義"を参照してください。

特殊指示(7カラム)

特殊指示のある文のデータフィールド定義について、特殊なフォーマット編集指示を行うことができます。表3.7に指定可能な特殊指示を示します。

データフィールド定義(8カラム以降)

データ定義文のフィールド定義を、COBOL文法のデータ記述項に従って記述します。後述のCOBOL文法規約に反する記述がされている場合の動作は、保障できません。

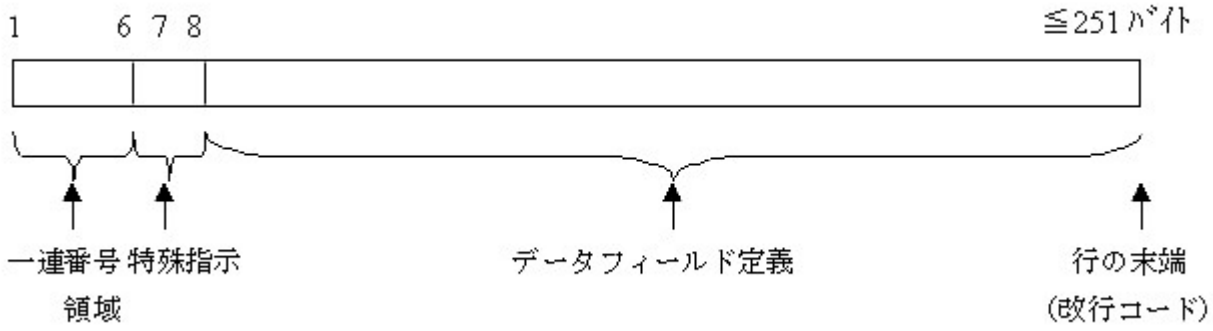


図3.1 データ定義文の正書法

表3.7 指定可能な特殊指示

指定文字	意 味
ブランク	特殊な意味はもたない。
*	注記行であることを示す。
I	指定行のデータフィールド文は、出力側レコードに空白文字の項目を追加することを示す。
A V	指定行のデータフィールド文は、出力側レコードに後続の特殊指示"V"で指定された値の項目を追加することを示す。 ※mdportgコマンドでは使用できません。(解析できません。)
D	指定行のデータフィールド文は、出力側レコードに出力しないことを示す。
U L	"U"指定行のデータフィールド文の領域長を、後続の特殊指示"L"で指定された領域長に変更して出力することを示す。
N	指定行のデータフィールド文は、一切のコード変換を行わないことを示す。
S	出力側がRDB型ファイルの場合、入力フィールドのデータが全て空白なら、1文字分の半角空白を出力することを示す。

[記述形式]

レベル番号 {FILLER | データ名-1} [{PICTURE | PIC} 文字列]
[[USAGE IS] {BINARY | COMPUTATIONAL | COMP | COMPUTATIONAL-3 | COMP-3 | DISPLAY | PACKED-DECIMAL |

COMP-5 | COMPUTATIONAL-5 }]
[[SIGN IS] {TRAILING | LEADING} [SEPARATE CHARACTER]]
[OCCURS 整数-1 TIMES].

備考
下線のある語は、必要語を示します。その機能を用いるときは必ず記述が必要です。下線のない語は補助語であり省略しても構いません。日本語の語は、利用者の与える語、定数等を示します。
{ }で囲まれている部分は、その括弧中の一つを明に指定しなければなりません。
[ ]で囲まれている部分は、省略することができます。

表3.8 PICTURE句で指定可能な文字列

文字列	意 味	備 考
X	英数字項目	1バイト系コードのみの項目(半角カナ含む) 日本語文字は認識されない
N	日本語項目	日本語文字のみの項目(2バイト/1文字)
M	英数字日本語混在項目	1または2バイト系コードが混在している項目
9	数字項目	数字編集項目は指定できない
S	符号	数字項目にて符号の有無を指定
V	小数点	数字項目にて小数点の位置を指定
BINARY,COMP COMPUTATIONAL または、 COMP-5 COMPUTATIONAL-5	2進形式	COBOL独自のバイナリ形式 BINARY, COMP : ビッグエンディアン形式で処理します。 COMP-5 : **[Solaris]** ビッグエンディアン形式で処理します。 **[Linux]** , **[Linux for Itanium]** , **[x64-Linux]** リトルエンディアン形式で処理します。
COMP-3 COMPUTATIONAL-3 または PACKED-DECIMAL	内部10進形式	COBOL独自のパック形式
DISPLAY または (USAGE指定なし)	外部10進形式	COBOL独自のゾーン形式
TRAILING	外部10進形式	COBOL独自のゾーン形式 符号部は最終バイトの上位4ビット（DISPLAY または (USAGE指定なし) と同じ）。 [例](ASCIIの場合) データ定義文 : S9(4) TRAILING データ（10進） : +1234 内部表現（16進） : 31 32 33 44
LEADING	外部10進形式	COBOL独自のゾーン形式 符号部は先頭バイトの上位4ビット。 [例](ASCIIの場合) データ定義文 : S9(4) LEADING データ（10進） : +1234 内部表現（16進） : 41 32 33 34
TRAILING SEPARATE	外部10進形式	COBOL独自のゾーン形式 符号部は最終1バイト。 [例](ASCIIの場合) データ定義文 : S9(4) TRAILING SEPARATE データ（10進） : +1234 内部表現（16進） : 31 32 33 34 2B
LEADING SEPARATE	外部10進形式	COBOL独自のゾーン形式 符号部は先頭1バイト。 [例](ASCIIの場合) データ定義文 : S9(4) LEADING SEPARATE データ（10進） : +1234 内部表現（16進） : 2B 31 32 33 34

[構文規則・一般規則]

- MDPORTでは、厳密な構文規則の検査を行っていないため、記述形式に反する記述をされた場合の動作は保証できない場合があります。
- 記述形式にないCOBOL言語のキーワードがあると、MDPORTでは構文検査により異常終了します。

- 可変長レコードの場合のRDWは、このレコード定義上に含めてはいけません。
- 可変長レコードを定義する場合は、最大レコード長分のデータフィールド定義を行う必要があります。
- 先頭のデータフィールド定義文のレベル番号は、01で集団項目属性でなければなりません。その後に続く文は、それに従属するデータフィールド定義でなくてはなりません。01レベルのデータフィールド定義は先頭の一文でしか記述できません。
- データフィールドに対する、データ項目名またはFILLER句は省略できません。
- USAGE句で指定される数値項目の表現形式は、領域長の大きさとコード変換仕様に影響されます。
- OCCURS句の整数-1に指定できる値は1～32767の範囲で、OCCURS句の入れ子は7階層まで記述できます。
- 各種データフィールド文の指定における内部表現仕様等は、"B.5 データファイル変換での変換仕様"を参照してください。
- PICTURE句で指定できる文字列の種類を表3.8に示します。英字、英数字編集、数字編集、日本語編集、ブールのデータ種類は指定できません。なお、文字列"N","M"については規格外の拡張仕様です。
- PICTURE句で英数字項目を指定した場合、PICTURE文字列"X"で表現されるけた数はバイト数と一致します。EUCコードでのコードセット2の1バイト系JISカタカナは、コマンドオペランドの指定(-ikanaまたは-okana)により、内部表現をJIS8コード(1バイト/文字)またはEUCコード(2バイト/文字)として扱うことができます。
- PICTURE句で日本語項目を指定した場合、内部表現は1文字あたり2バイトとなります。JEF, IBM, KEISおよびJISコードでの日本語切替制御コード(シフトイン/シフトアウト)は、項目内に存在してはいけません。
- PICTURE句で英数字日本語混在項目を指定した場合、PICTURE文字列"M"で表現されるけた数はバイト数と一致します。EBCDIC系コード(JEF、他社)およびJISコードでは、混在項目内に日本語切替制御コードが存在していないと、日本語文字として識別されません。また、EUCコードではコマンドオペランドの指定と関係なく、半角カナは2バイト/文字となります。
- EUCコードでのコードセット3の文字(JEF拡張、利用者定義文字)は、日本語項目ではcobolEUC(注)表現(2バイト/文字)となり、英数字日本語混在項目では3バイト/文字となります。
- データ名-1に記述できる項目名は90バイトまでです。

(注) cobolEUCは、COBOL独自の日本語内部表現形式16ビットワイドキャラクタを示します。

[UnicodeのCOBOLファイルについて]
UnicodeのCOBOLファイルでは、日本語項目をUCS2形式、英数字項目をUTF8形式で格納するため、行順ファイルを扱う場合、データ定義文は英数字項目"X"、符号なし外部10進項目"9"か、日本語項目"N"のみから構成される必要があります。

[特殊指示における構文規則・一般規則]

7カラム目に指定できる特殊指示において、以下の規則に従っていなければなりません。また、データ定義文の記述例を図3.2に示します。

●項目追加指示(I)
- 本指定がされている項目は、入力レコード上にはなく、出力レコードへの編集の際に新たに領域を追加することを示します。追加領域には空白文字が埋められます。
- 集団項目に対して、本指定はできません。
- PICTURE文字列は"X"で定義されていなければなりません。
- 出力がRDBローダ型ファイルの場合は、領域長を指定しても後続ブランクが削除されるため、ヌル文字列として出力されます。

●データ指定項目追加指示(A, V)
- 本指定がされている項目は、入力レコード上にはなく、出力レコードへの編集の際に新たに領域を追加することを示します。追加領域には直後の"V"指定の行に16進数で記述されたデータが埋められます。
- 集団項目に対して、本指定はできません。
- PICTURE文字列は"X", "M", "N"で定義されていなければなりません。
- 指定されたデータの長さが項目長より短い場合、項目長分まで指定データが繰り返されます。
- 指定されたデータの長さが項目長より長い場合、項目長分で指定データが切り捨てられます。
- 追加されたデータはコード変換されずに出力されます。
- 追加指定された項目の種別と、指定されたデータの種別が同一でない場合の動作は保証されません。
- Vの後に追加するデータは、16進で指定しなければなりません。
- PICTURE句に日本語項目を設定した場合、次行のV指定では偶数バイトを指定しなければなりません。

●項目削除指示(D)
- 本指定がされている項目は、入力レコード上から出力レコードへの編集の際に、出力しないことを示します。
- 集団項目に対して、本指定はできません。
- 入力がRDBローダ型ファイルの場合は、入力項目が固定の領域長を持っていないため、データ定義文のけた数は意味を持ちません。項目削除指示された項目名全体が削除対象となります。

●領域長変更指示(U,L)
- "U"指定の行には変更されるデータフィールド定義を行い、直後の"L"指定の行にはデータフィールド定義欄に変更後の領域長のみを指定します。
- "U"指定の行には、集団項目は定義できません。

- "U"指定の行のPICTURE文字列は、"X", "M"または"N"で定義されていなければなりません。
- "U"指定行の直後に、"L"指定行がなくてはなりません。
- "L"指定の行には、変更後領域バイト長を0～32767の範囲で、外部10進数で記述します。
- 入力がRDBローダ型ファイルの場合は、"U"指定行の領域長は意味がなく、"L"指定行の領域長で出力されます。出力がRDBローダ型ファイルの場合は、"L"指定行の領域長は意味がなく、"U"指定行の領域の変換結果がRDBローダ型に編集されて出力されます。

●無変換指示(N)

- 本指定がされている領域は、一切の変換処理を行わずに、入力項目の領域をそのまま出力することを示します。ただし、出力がRDBローダ型又はテキストファイルの場合は空白で出力されます。
- 集団項目に対して、本指定はできません。
- -noconvオペランドを同時に指定した場合は、本指定は無効になります。
- 入力がRDBローダ型ファイルの場合は、一切の変換処理を行わずに、入力項目の内容をそのまま出力しますが、出力長に満たない場合にはバイナリゼロ(0x00)が埋められます。ただし、入力項目が空の場合、出力項目が文字項目の場合は空白が埋められ、数値項目の場合は0を示す値が埋められます。

●ヌル抑制指示(S)

- 本指定は、入力ファイル形式がデータファイル(-itype data)かつ出力ファイル形式がRDBローダ型ファイル(-otype rdb)の場合に有効となります。それ以外では本指定があっても意味を持ちません。また、-informix指定の場合も意味を持ちません。
- 本指定がされている入力領域が全て空白文字の場合、出力データは1文字分の半角空白を引用符(")でくくって出力します。この機能は、対象項目の出力をヌルとしたくない場合に有効です。
- 集団項目および数値項目に対して、本指定はできません。

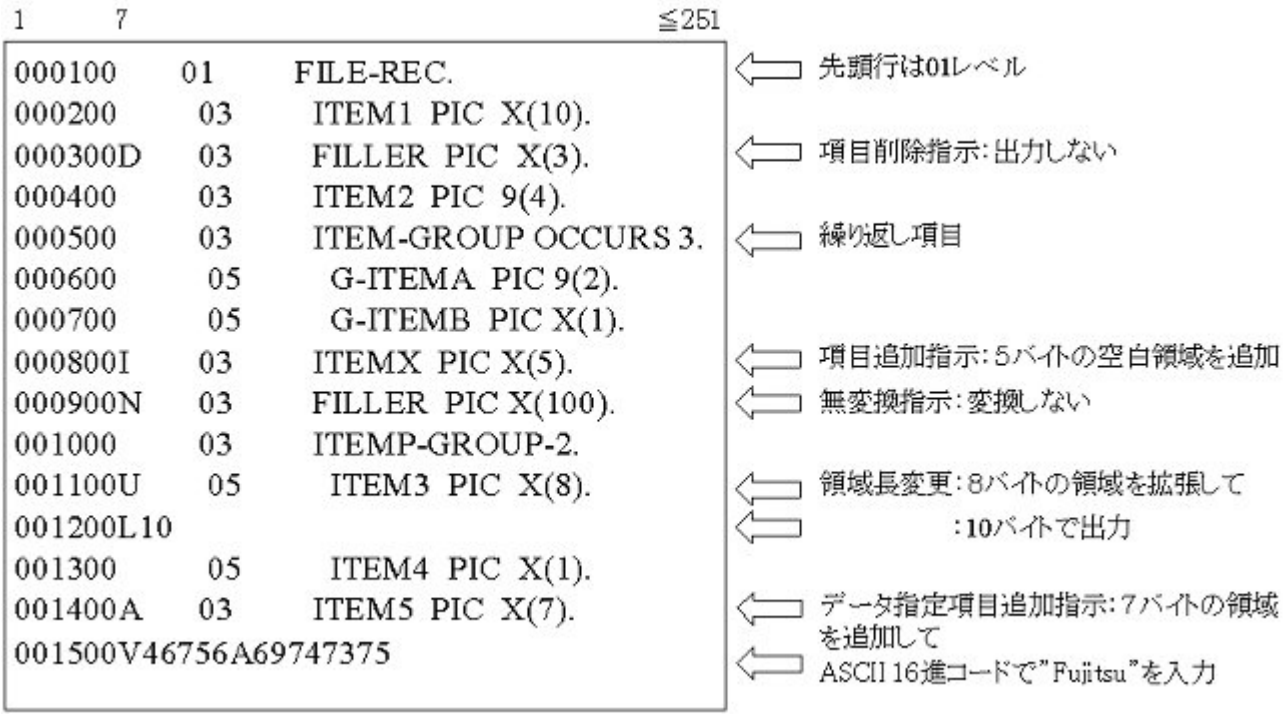


図3.2 データ定義文の使用例

3.2.1 フォーマット指示制御文によるマルチフォーマット定義

データファイル変換において、1ファイル上のレコードフォーマットが、レコードによって変わる(複数のレコードフォーマットが存在する)ものを、本書ではマルチフォーマットと呼びます。ここでは、フォーマット指示制御文を使用したマルチフォーマットの対応方法について説明します。

フォーマット指示制御文では、定義したレコードフォーマットが、どのような条件のレコードについて適用するかを指示します。なお、この機能を応用することにより、入力ファイルに対するレコード抽出も行うことができます。

フォーマット指示制御文は、データ定義文上に記述して定義します。フォーマット指示制御文は、1つのデータ定義文ファイル上のフォーマットの区切りとなります。

フォーマット指示制御文によるマルチフォーマット定義の例を、図3.4に示します。

[テキスト構成]

制御文識別領域
　データ定義文のデータフィールド定義行と区別するため、この領域には"######"を指定し、フォーマット指示制御文であることを定義します。

フォーマット指示文
　フォーマット指示を記述します。

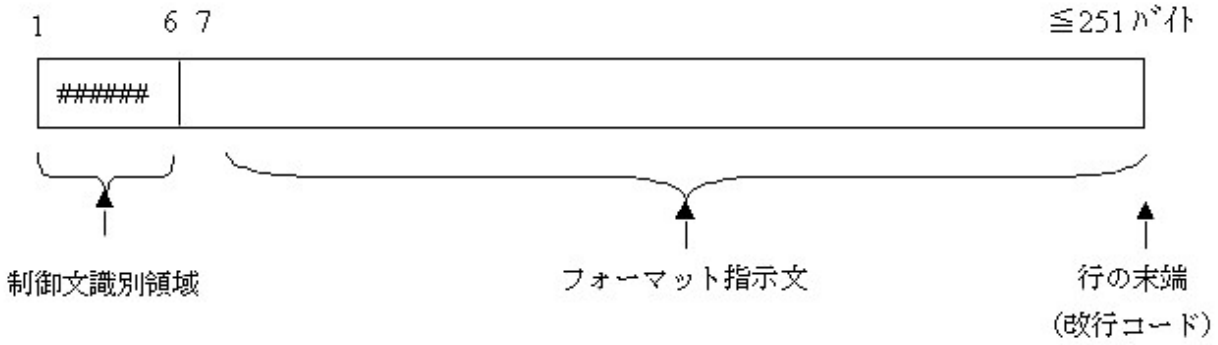


図3.3 フォーマット指示制御文の書式

[記述形式]

{ SELECT フィールド位置, フィールド長, { EQ | NOT }, 最小比較値[, 最大比較値] }
[OTHER]

[構文規則・一般規則]

- フォーマット指示制御文で定義するフォーマットは、次のフォーマット指示制御文の出現、またはデータ定義文の末尾まで有効となります。
- フォーマット指示制御文を使用する場合、データ定義文の1行目は、フォーマット指示制御文である必要があります。また、フォーマット指示制御文の次の行は、01レベルのデータフィールド定義でなければなりません。
- SELECT文では、フォーマットを適用させるレコードの条件を定義します。OTHER文では、SELECT文に該当しないレコードに対してのフォーマットを定義します。
- 1つのデータ定義文に対して、SELECT文を複数定義することにより、フォーマットは最大99まで定義できます。OTHER文のフォーマット定義は複数定義できません。
- 複数のSELECT文がある場合、SELECT文の記述順でフォーマット判定が行われます。最初に条件に合致したSELECT文でフォーマットで処理され、それ以降のSELECT文条件判定は行われません。
- どのSELECT文にも合致しない場合、OTHER文のフォーマットで処理されます。OTHER文が定義されていないと、そのレコードは変換処理の対象とならず、出力もされません。(この機能により、レコード抽出が可能です)
- レコード長が固定長の場合(-irecfmまたは-orecfmオペランドが、fの場合)、レコードフォーマットで定義されている全てのレコード長は、同じでなければなりません。
- フォーマット指示制御文では、レコード相対番号による条件指定はできません。(1件目と2件目以降でフォーマットが異なる場合、等)
- SELECT文では、レコード中の固定の領域について内容判定を行います。SELECT文で記述する内容は以下のとおりです。

フィールド位置	条件判定を行う領域の開始位置を、先頭を1としたレコード内相対バイト数で指定します。
フィールド長	条件判定を行う領域の長さを、バイト数で指定します。
EQまたはNOT	EQでは、領域の内容が次の比較値に合致すれば真とします。NOTでは、比較値に合致しなければ真とします。
最小比較値	領域の内容に対する最小比較値を指定します。指定方法は、文字列を二重引用符(")で囲む方法と、16進数による直定数指定があります。指定した内容は、フィールド長と合致しなければなりません。
最大比較値	領域の内容に対する最大比較値を指定します。指定方法は、最小比較値と同じで

	す。最大比較値を省略した場合、最小比較値と同じ値で条件判定されます。

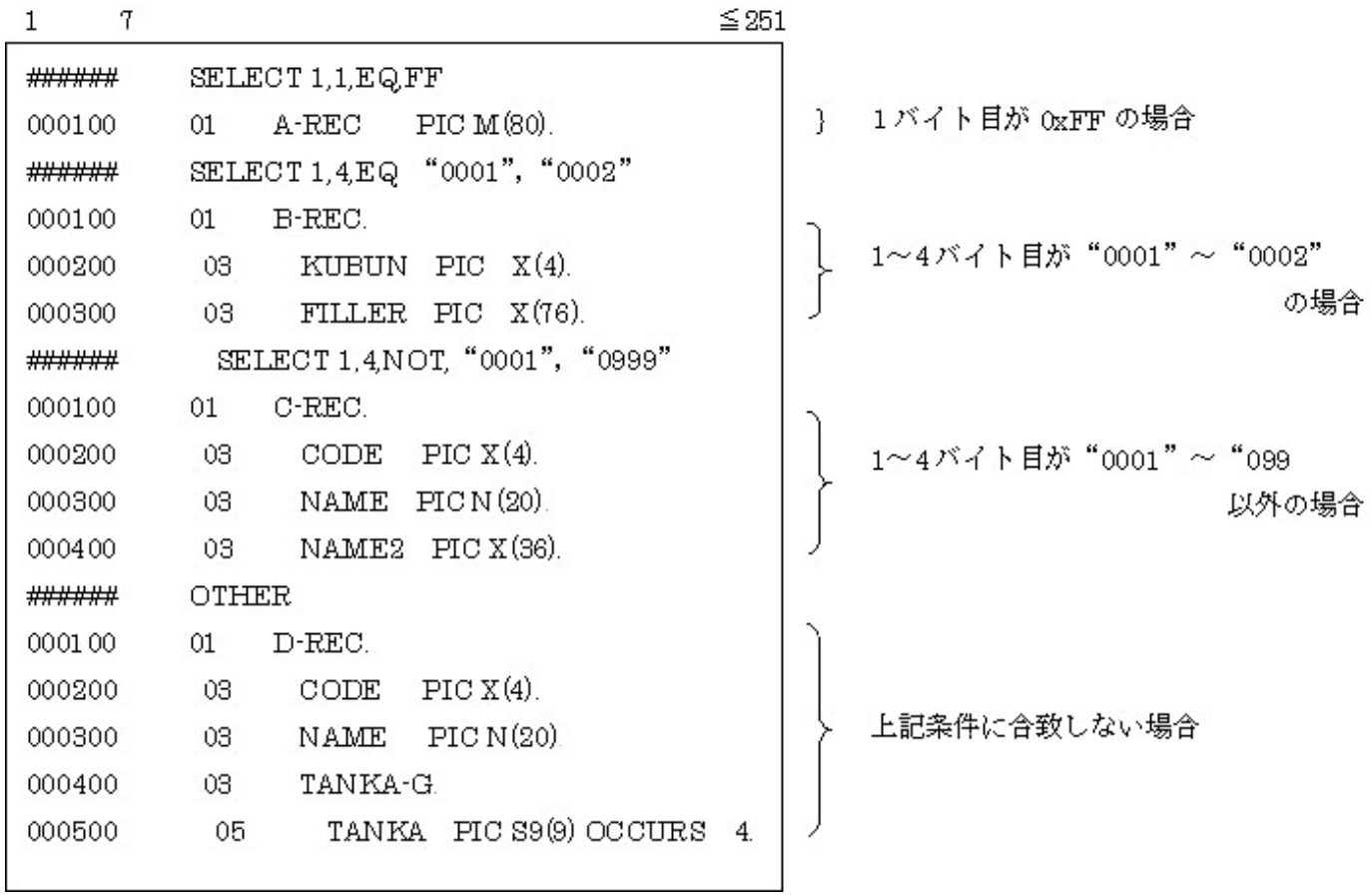


図3.4 フォーマット指示制御文によるマルチフォーマット定義の例

3.3 項目入換え機能

本節では、mdportfコマンドでCOBOLファイルに出力する際、入力レコードフォーマットと異なる項目順で出力する機能について説明します。
データ定義文で指定するレコードフォーマットは入力となるデータファイル(およびCOBOLファイル)に対して定義しますが、それに加え、コード変換後のレコードフォーマットとなる項目 入換え元登録集、項目を入換えた後のレコードフォーマットとなる項目入換え先登録集を定義します。

項目入換え元登録集と項目入換え先登録集はCOBOL言語のデータ記述項に準じている必要があります。

[テキスト構成]

項目入換え元登録集および項目入換え先登録集を記述するソースは、可変長のテキストファイルに定義し、COBOL言語のデータ記述項に準じている必要があります。

[構文規則・一般規則]

- この機能は、COBOLのMOVE CORRESPONDING文によって行われます。
- 指定された登録集は、COBOLコンパイラに渡されるため、記述形式に反する記述をされた場合の動作は保証できません。
- 登録集名に使用する拡張子は.cob、.cbl、.cobolのいずれかである必要があります。
- 登録集名のうち拡張子を除いた部分と、集団項目名は同じである必要があります。
- データ定義文と項目入換え元登録集が、同じレイアウトでない場合の動作は保証できません。
- UnicodeのCOBOLファイルを扱う場合、指定する登録集はUTF-8形式で記述する必要があります。

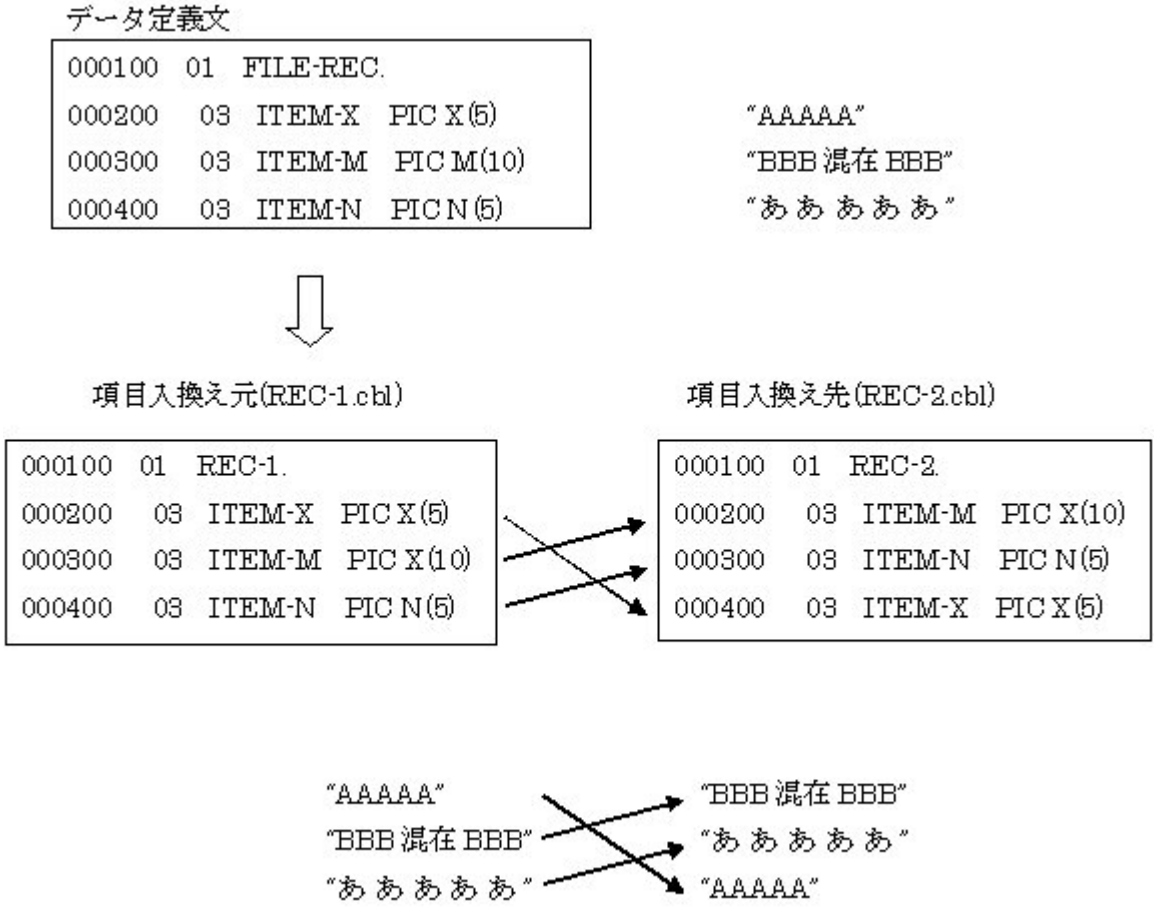


図3.5 項目入換えの例

図3.5の場合、コマンドは以下の様に指定する。
% mdportf … -icopy REC-1 -ocopy REC-2 …

3.4 利用者定義変換テーブルの入力形式

本節では、MDPORTのコード変換機能において、利用者が任意に変換方法を指示する場合に指定する、利用者定義変換テーブルの入力形式を説明します。
利用者定義変換テーブルにより、利用者定義文字(外字)や拡張文字の変換仕様を任意に指定することができます。利用者定義文字(外字)や拡張漢字・非漢字を任意の文字コードに変換するには、以下の方法があります。

- iconvの文字コード変換表に対応させるコードを定義する。

  mdportfコマンドで-iconv yes指定が必要です。

- MDPORTの利用者定義変換テーブルを使用する。

MDPORTのコード変換仕様における利用者定義変換テーブルの位置づけを図3.6に示します。MDPORTのコード変換機能は、利用者定義変換テーブルを参照し、指定のコードが定義されていれば、定義にしたがってコード変換を行い、利用者定義変換テーブル上に指定のコードが定義されていなければ、MDPORTのコード変換基準にしたがってコード変換を行います。MDPORTにおけるコード変換基準は"付録B 変換仕様"を参照してください。

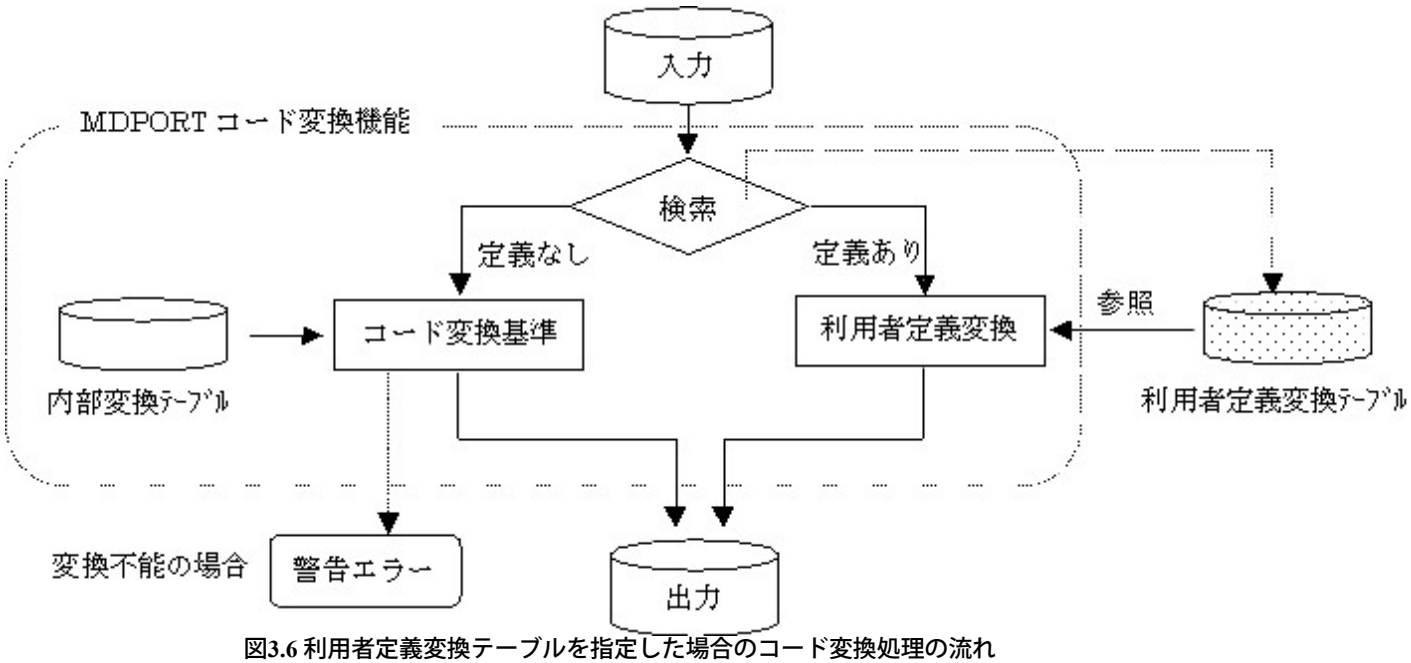


図3.6 利用者定義変換テーブルを指定した場合のコード変換処理の流れ

[テキスト構成]

利用者定義変換テーブルは、1行が255バイト以内の可変長テキストファイルとして定義します。
利用者定義変換テーブルは、コード系定義文とコード対応定義データで構成されており、それぞれは1カラム目の内容("#"または空白)で識別されます。1カラム目が"*"の場合は、注釈行とみなされます。

[記述形式]

コード系定義文
　　# コード系名1 コード系名2
コード対応定義データ
　　文字コード1[:文字コード2]

[構文規則・一般規則]

- 1行に対し1文のみの記述とし、行をまたいで文を記述することはできません。
- 記述形式の[ ]で囲まれた部分は、省略することができます。
- コード系定義文の1カラム目は"#"でなくてはなりません。
- コード対応定義データの1カラム目は空白でなくてはなりません。
- コード対応定義データのコロン(:)の前後に空白はあってはなりません。
- コード系定義文のコード系名1およびコード系名2には、mdportfコマンドの-icode、-ocodeオペランドで指定するキーワードを指定します。キーワード以外のコード系名、または、"cobub"、"cobul"を指定した場合、そのコード対応定義は意味を持ちません。
- コード対応定義データはコード系定義文の直後に指定され、コード系定義文で指定されたコード間の変換を定義します。文字コード1にはコード系名1、文字コード2にはコード系名2に対応したコードを16進で指定します。
- コード系定義文は複数指定できます。同じコード系名同士の対応が複数あった場合は、最初に出現するコード系定義文のコード対応定義データのみが有効となります。

- コード系名1とコード系名2に同じキーワードを指定することもできます。この場合のコード対応定義データは、文字コード1は変換元、文字コード2は変換先コードを意味します。
- 文字コード2を省略した場合、そのコード対応定義データは意味を持ちません。
- 文字コード1または文字コード2に"NO"を指定した場合、対応するコードがないことを意味し、コード変換の際コード変換エラー扱いとなります。
- 同一の変換元コードに対して変換規則が複数存在する場合は、最後に記述されている指定が有効となります。
- コード対応定義データに指定する文字コードは、コード系名にUnicodeを含む/含まないによって指定規則が異なります。

[コード系定義文にUnicode(utf8、ucs2b、ucs2l)を含まない場合、かつ標準コード変換(iconv)による変換の他社コードを含まない場合]

- 文字コード1および文字コード2には16進コードで、2桁(1バイト文字)、4桁(2バイト文字)、6桁(3バイト文字)のいずれかが指定できます。6桁指定はEUCコードのみです。
- 2桁で指定する場合は、対応するコードも2桁で指定しなければなりません。
- 6桁を指定する場合、"8F"で始まるコードでなくてはなりません。
- EUCカナ(コードセット2)を指定する場合は、"8E"を除いた2桁で指定します。
- JISコードで半角の英数カタカナを指定する場合は8ビットコードで指定します。
- 以下のコード範囲を指定することはできません。

	1バイト目	2バイト目
SJIS	0x00～0x80 0xA0～0xDF 0xFD～0xFF	0x00～0x3F 0x7F 0xFD～0xFF
EUC(コードセット2)	0x00～0xA0 0xFF	
※その他のコード、及び、1バイト系文字についての注意事項はありません。		

[コード系定義文にUnicode(utf8、ucs2b、ucs2l)を含む場合、または標準コード変換(iconv)による変換の他社コードを含む場合]

- 文字コード1および文字コード2には16進コードで、2桁(1バイト文字)、4桁(2バイト文字)、6桁(3バイト文字)のいずれかが指定できます。6桁指定はEUC、UTF8コードのみです。
- 半角文字については、通常、2桁で指定します。ただし、以下のケースでは対応するコードを4桁、または、6桁で指定します。

特殊な半角文字	指定方法
EUCカナ(コードセット2)	"8E"を付加した4桁で指定します。8Eに続く2バイト目の範囲は A1-DFです。これ以外の範囲は定義エラーになります。
UCS2(半角文字)	[ビッグエンディアン(ucs2b)の場合] 以下の範囲を半角文字とみなします。 ・(1バイト目 == 0x00 かつ 2バイト目 ≦ 0x7F) または ・(1バイト目 == 0xFF かつ 0x61 ≦ 2バイト目 ≦ 0x9F)
UTF8(半角カナ)	以下の範囲を半角文字とみなします。 ・0xEFBDA1～0xEFBDBF または ・0xEFBE80～0xEFBE9F の場合

- 4桁または6桁(全角文字)を指定する場合、文字コードとして認識できない範囲を指定することはできません。(一部、指定可能な範囲あり)
- EUC(コードセット3)を指定する場合は、"8F"で始まるコードでなくてはなりません。
- -icode、-ocodeに"cobub"、"cobul"を指定して運用する場合、利用者定義テーブルは、コード系名として、"utf8"、"ucs2b(またはucs2l)"の2通りの指定を行う必要があります。(X/M項目はUTF8、N項目はUCS2コードで処理を行うため)
- 以下のコード範囲は利用者定義変換テーブルのコード指定範囲解除に関わらず指定することはできません。

コード体系	1バイト	2バイト	3バイト
SJIS	81-9F E0-FC	先頭１バイトが 00-7F,A1-DF	-
EUC	8E,8F A1-FE	1バイト目が00-7F 1バイト目が8Fで 2バイト目が A1-FE (G3の1,2バイト目)	1バイト目が 00-7F,8E

JEF	28,29,38,30	左記コードで始まる2バイト	-
JIPSE	3F	左記コードで始まる2バイト	-
JIPSJ	1A	左記コードで始まる2バイト	-
KEIS	0A,30	左記コードで始まる2バイト	-
IBM	0E,0F	左記コードで始まる2バイト	-
UCS2	-	2バイト(4桁)定義以外	-
UTF8	C0-FD (先頭1文字が他のバイト数文字の先頭以外)	１バイト目 00-7F 4,5,6バイト系で始まる先頭1バイト F0-F7, F8-FB, FC-FD 3バイト系で始まる先頭1バイト E0-EF	１バイト目 00-7F 4,5,6バイト系で始まる先頭1バイト F0-F7, F8-FB, FC-FD 1,2バイト目が2バイト文字 C080-DFBFで始まる3バイト

例:

```
*JEF-EUC間での変換
# jef euc
  47C9:C4CD
  00:5F
  80A1:8FDDB1
  80A2:8FDDB2
*SJIS-EUC間での変換
# sjis euc
  8140:A2A3
  00:00
  8140:A2A4
```

[利用者定義変換テーブルのコード指定範囲解除]

利用者定義変換テーブルは、利用者定義文字や変換不能文字について任意のコードへ変換する機能として提供しています。ただし、変換可能なコードの範囲は、文字コードとして定義されている領域に限られています。特殊な領域へコード変換を行いたい場合は、コード指定範囲制限解除の指定を行います。指定方法は、「3.5 実行環境」の環境変数MDPORT_TABLE_CHECKを参照してください。

[注意事項]
コード系名にUnicode(utf8、ucs2b、ucs2l)を含んでいる場合、または標準コード変換(iconv)による変換の他社コードを含んでいる場合は、コード指定範囲制限解除の指定を行っても全ての領域は指定できません。

例

```
*JEF-SJIS間での変換
*MDPORT_TABLE_CHECKの値がnoの場合のみ指定可能
# jef sjis
  47C9:0000
  0000:0000
  FFFF:FFFF
```

3.4.1 利用者定義変換テーブルの入力形式(旧形式)

ここでは、旧版のMDPORTで使用されていた利用者定義変換テーブルの入力形式を説明します。本形式は互換性のために用意されており、新たに定義を行う場合は"3.4 利用者定義変換テーブルの入力形式"に記載されている新形式を使用することを薦めます。

[テキスト構成]

利用者定義変換テーブルは、1行が255バイト以内の可変長テキストファイルとして定義します。変換規則はどこから記述されていてもかまいません。

[記述形式]

[EUCコード, 相手側コード]; [注釈]

[構文規則・一般規則]

- 一行に対し一文のみの記述とし、行をまたいで文を記述することはできません。
- 記述形式の[ ]で囲まれた部分は、省略することができます。
- 変換規則として、EUCコードと対応する相手側コードを、","で区切り、文末に";"を記述します。
- ";"で始まる行は、注釈のみの行となります。
- 相手側コードには、EUCコードに対応するコード値を16進数2桁または4桁で記述します。
- EUCコードには、相手側コードに対応するEUCコードを16進数2桁、4桁、または6桁(コードセット3)で記述します。また、EUCコードの記述には、以下の制約があります。
  - 16進数4桁を指定した場合、先頭の2桁に"8F"の指定はできません。
  - 16進数6桁を指定した場合、先頭の2桁は"8F"でなければなりません。
- 相手側コードまたはEUCコードに16進数2桁を記述した場合、対応するコードも16進数2桁でなければなりません。
- 同一の変換元コードに対して、変換規則を複数記述した場合は、最後に記述されている指定が有効となります。
- 変換先コードの記述で先頭2桁が"00"で始まるコード値を指定した場合、MDPORTでは変換不能文字として扱われるため、注意が必要です。
- コマンドのオペランド指定で、入出力ともにEUCコードの場合(-icode eucかつ-ocode euc)は、利用者定義変換テーブルには変換元、変換先コードの順で記述します。この場合、変換元コードと変換先コードの記述けた数は同じでなければなりません。また、記述するコードはEUCコードの割り当て範囲内でなければなりません。

例:

```
;利用者定義文字テーブル
AAAA, 80C1      ;利用者定義文字
01   , F1       ;特殊コード
8FDDB1, 80A1    ;単位記号
8FDDB2, 80A2    ;一般記号
      :
```

3.5 実行環境

本節では、MDPORTを実行するにあたっての環境について、説明します。

[実行環境のコード体系]
MDPORTで出力されるメッセージは、OSの文字コード環境によって、EUCコードまたはシフトJISコードで出力されます。OSの文字コード環境は環境変数によって、決定されます。

[Linux] シフトJISロケールは未サポートです。
[Linux for Itanium] [x64-Linux] Unicodeロケールのみサポートしています。

[参照される環境変数]
MDPORTでは表3.9に示す環境変数により、動作や変換仕様等を制御することができます。

表3.9 MDPORTで参照される環境変数

環境変数	省略時の値	意味
TMPDIR	/var/tmp **[Linux][Linux for Itanium] [x64-Linux]** /tmp	作業用ディレクトリを設定します。
MDPORT_DIR	/opt/FJSVmdprt **[x64-Linux]** /opt/FJSVmdprt64	MDPORTの格納ディレクトリを設定します。
MDPORT_MFSZ(注1)	1000	メッセージ出力先ファイルの最大サイズをキロバイト単位で指定します。 (1～2000までの範囲で指定します。)
MDPORT_LOCALE	eucコードまたは utf-8コードと判断	MDPORTを実行するロケール情報を指定します。下記に該当するロケールを指定してください。 EUCの場合 : ja, japanese, ja_JP.eucJP の何れか SJISの場合 : ja_JP.SJIS, ja_JP.PCK の何れか UTF-8の場合 : ja_JP.UTF-8, en_US.UTF-8 の何れか 上記以外のロケール情報を設定した場合は、EUCロケールとして判断します。
MDPORT_EXACT_CHECK	no	入力データに対し、数値項目の厳密チェック処理を行う場合、環境変数を「yes」に指定します。(no指定時は従来チェック仕様です。)本指定は、変換エラーの出力レベルを制御するものであり、変換仕様が変わることはありません。詳細については、「B.9 数値項目チェックにおける変換仕様」を参照してください。
MDPORT_INITIAL_VALUE	初期化なし	数値項目の変換処理中にエラーが発生した場合に出力する初期値を指定します。 MDPORT_EXACT_CHECKを設定している場合にのみ有効です。 設定方法は表3.9.1を参照してください。
MDPORT_TABLE_CHECK	yes	利用者定義変換テーブル内で定義するコード指定範囲制限を解除する場合に、「no」を指定します。(yes指定時は従来仕様です)詳細については、「3.4 利用者定義変換テーブルの入力形式」を参照してください。

注1)コード変換/レコード変換機能(ライブラリ)[旧形式]を使用する場合に指定します。

表3.9.1 MDPORT_INITIAL_VALUEの指定方法

指定方法	指定例	意味
10進数	0, -1, 100, 9999	10進数の値を指定します(符号、小数点指定可能)。変換エラーが発生した場合、指定した10進データを出力の属性に合わせて出力します。 ※指定10進データと出力項目の属性が異なる場合の出力仕様 - 指定桁数と出力項目の桁数が異なる場合 例) 9(3) 環境変数 1234 出力結果 234(桁落ちします) - 指定符号(小数点)と出力項目の属性が異なる場合 例) 9(3) 環境変数 -123.4 出力結果 123(符号部[小数点]は省略します)
16進数	0x00, 0x31	0xに続けて、1バイトの16進データを指定します。変換エラーが発生した場合、出力項目の領域長に合わせて指定した16進データをそのまま出力します。) 9(3) 0x00 000000

		例　　環境変数　　出力結果
SPACE(空白)	space, SPACE	変換エラーが発生した場合、出力項目の領域長に合わせて指定された出力コードの空白データを出力します。 例)SJIS出力の場合 9(3) 環境変数 space 出力結果 202020 例)JEF出力の場合 9(3) 環境変数 space 出力結果 404040
指定なし	未設定	変換エラーが発生した場合、初期化を行いません。
指定エラー		「Illegal environment variable」のメッセージを表示して処理を中断します。 (MDPORT_EXACT_CHECKに「no」が指定されていてもチェックします。)

[COBOLファイルアクセスのために必要な環境変数]

COBOLファイルとの入出力を行うには、以下の環境変数の設定が必要です。

- 環境変数PATH
  COBOLコンパイラを起動するためのcobolコマンドが、環境変数PATHで設定されているディレクトリ配下に存在する必要があります。
- 環境変数LD_LIBRARY_PATH
  COBOLプログラム実行にあたっての、COBOLランタイムシステムの格納ディレクトリが設定されている必要があります。
- 環境変数LANG
  UnicodeのCOBOLファイルを扱う場合、動作環境のコードがUnicodeである必要があります。

[変換仕様に係わる環境変数]

MDPORTのコード変換で、-iconv noオペランドが指定されてない場合、内部的にiconv関数を使用しています。表3.10に示す環境変数を設定することにより、iconv_openを呼び出すキーワードを制御することができます。

表3.10 iconvに関連する環境変数

コード系	環境変数	設定できる値 (下線は省略時)
JEF	MDPORT_ICONV_JEF	Jefkana Jefcorekana Jefaugkana
EUC	MDPORT_ICONV_EUC	U90 S90
シフトJIS	MDPORT_ICONV_SJIS	sjis sjisdos sjisms
JIS	MDPORT_ICONV_JIS	jiskana jiskana7 jiskana8
日本電気JIPS(E)コード	MDPORT_ICONV_JIPSE	N_JIPSEAUG N_JIPSECORE
日本電気JIPS(J)コード	MDPORT_ICONV_JIPSJ	N_JIPSJEAUG N_JIPSJECORE

-Jefkana、sjisが指定された場合、変換仕様はiconv側の変換規則に従います。

-MDPORT_ICONV_JIPSE、MDPORT_ICONV_JIPSJは、相手コードがsjisms、S90のときのみ指定可能です。

-JISコードの場合の、JISローマ字とJISカタカナの切り替えの方法が、環境変数MDPORT_ICONV_JISの値により、以下のように変わります。

-EUCコードの場合に、iconv側をS90で変換するように指定している場合は、環境変数MDPORT_ICONV_EUCの値もS90を指定する必要があります。

jiskana	エスケープシーケンスで切り替え
jiskana8	1バイトコード(半角英数カナ)を8ビットコードで行う。
jiskana7	シフトコードで切り替え

[iconv不使用に関する環境変数]

環境変数	省略時の値	意味
		"no"を指定した場合、MDPORTの内部テーブルを利用した変換を行います。(iconvは利用しません。) [コマンド機能]

MDPORT_ICONV	各機能の使用方法に従う	-iconv yes、noを指定した場合は、オペランド指定が優先されます。 ※-iconvコマンド省略時に内部テーブルを利用したい場合に、使用してください。 [コード/レコード変換機能] MDCOINIT、mdp_init関数でiconvの使用を指定した場合は、環境変数指定が優先されます。

[バリデーション機能(Charset Validator)を適用した変換を行う際の環境変数]

・ -validation オペランドに関連した環境変数

環境変数	値	意味	注意
MDPORT_VALIDATION	yes	バリデーション機能を適用した変換を行います。 入力コード(-icode又は、-ocode)に指定した文字モードに対して、 -policyf オペランド、もしくは、環境変数で指定したバリデーションポリシーファイルを適用した変換を行います。	バリデーション機能を適用した変換は入力側(-icode)、出力側(-ocode)同時に行うことができないため、その際は、入力側(-icode)の文字コードでバリデーション機能を適用した変換を行います。 パラメタの指定では、大文字、小文字の区別は行いません。
	no	バリデーション機能を適用した変換を行いません。	オペランド省略時のデフォルト指定となります。 パラメタの指定では、大文字、小文字の区別は行いません。
	上記以外	バリデーション機能を適用した変換を行いません。	エラーにはせず、 no と同じ動作をします。

※オペランドと環境変数が同時に指定されている場合は、オペランドを優先します。

・-policyf オペランドに関連した環境変数

以下に、バリデーションポリシーファイルのパス情報を指定する環境変数について説明します。

環境変数	値
MDPORT_VALIDATION_PATH	バリデーションポリシーファイルの格納パス情報を指定します。 その指定方法には、以下の２通りがあります。 (1)バリデーションポリシーファイルのパス情報（ファイル名も含む）を指定。 ※この場合、-policyf オペランド、バリデーションポリシーファイルを指定する環境変数にバリデーションポリシーファイルを指定しないようにしてください。 (2)バリデーションポリシーファイルのパス情報（パスのみ）を指定。 ※この場合、-policyf オペランド、バリデーションポリシーファイルを指定する環境変数のどれか１つには、バリデーションポリシーファイル名を指定してください。

次に、バリデーションポリシーファイル名を指定する環境変数について説明します。

文字コード	環境変数	意味
utf8 ucs2b ucs2l cobub cobul	MDPORT_VALIDATE_UNICODE	Unicode で共通のバリデーションポリシーファイル名を指定する環境変数。 この環境変数にフルパスでバリデーションポリシーファイル名を指定した場合には環境変数 MDPORT_VALIDATION_PATH に値を指定してはいけません。 ファイル名、もしくは相対パスで指定した場合は、環境変数 MDPORT_VALIDATION_PATH に、指定したバリデーションポリシーファイルが格納されたディレクトリのパスを指定しなければいけません。

[環境変数の指定方法に関する注意事項]

- MDPORT_VALIDATE_UNICODE の環境変数には、ファイル名、フルパス、相対パスのいずれかによりバリデーションポリシーファイルを指定します。

その際、環境変数 MDPORT_VALIDATION_PATH を指定した場合は、フルパスで指定しないでください。

3.6 コード変換/レコード変換機能(ライブラリ)

本節では、コード変換/レコード変換機能について説明します。コード変換/レコード変換機能は、Solaris/Linux機と汎用機・オフコン・PC間で、ソースやデータファイル等の資源を流用させることを目的とした関数群で、以下の機能に分類されます。

- コード変換機能
  各種コードへのコード変換を行います。
- レコード変換機能
  レコード単位にコード変換、レコード形式変換を行います。

3.6.1 コード変換機能

コード変換機能は、共用オブジェクト・ライブラリとして提供され、ユーザアプリケーションと動的にリンクすることにより、各種コードへのコード変換を行います。

[機能説明]

コード変換機能は以下の関数をユーザアプリケーションに組込むことにより、各種コード変換を行います。コード変換関数の詳細については"付録D コード変換/レコード変換機能(ライブラリ)"を参照して下さい。

№	関数名	機能
1	MDCOINIT()	使用するコード変換関数の宣言を行います。
2	MDCOFREE()	コード変換終了処理を行います。
3	MDCOSTAT()	詳細エラー情報を取得します。
4	MDCOEBEU()	EBCDICからEUC(半角英数カナ)に変換します。
5	MDCOEBJ7()	EBCDICからJIS7に変換します。
6	MDCOEBJ8()	EBCDICからJIS8に変換します。
7	MDCOEUEB()	EUC(半角英数カナ)からEBCDICに変換します。
8	MDCOEUJ7()	EUC(半角英数カナ)からJIS7に変換します。
9	MDCOEUJ8()	EUC(半角英数カナ)からJIS8に変換します。
10	MDCOEUJE()	EUCからJEFに変換します。
11	MDCOEUJI()	EUCからJISに変換します。
12	MDCOEUSJ()	EUCからSJISに変換します。
13	MDCOJ7EB()	JIS7からEBCDICに変換します。
14	MDCOJ7EU()	JIS7からEUC(半角英数カナ)に変換します。
15	MDCOJ7J8()	JIS7からJIS8に変換します。
16	MDCOJ8EB()	JIS8からEBCDICに変換します。
17	MDCOJ8EU()	JIS8からEUC(半角英数カナ)に変換します。
18	MDCOJ8J7()	JIS8からJIS7に変換します。
19	MDCOJEEU()	JEFからEUCに変換します。
20	MDCOJEJI()	JEFからJISに変換します。
21	MDCOJESJ()	JEFからSJISに変換します。
22	MDCOJIEU()	JISからEUCに変換します。
23	MDCOJIJE()	JISからJEFに変換します。
24	MDCOJISJ()	JISからSJISに変換します。
25	MDCOSJEU()	SJISからEUCに変換します。
26	MDCOSJJE()	SJISからJEFに変換します。
27	MDCOSJJI()	SJISからJISに変換します。

[動作環境]

コード変換機能のシステムフローを図3.7に示します。

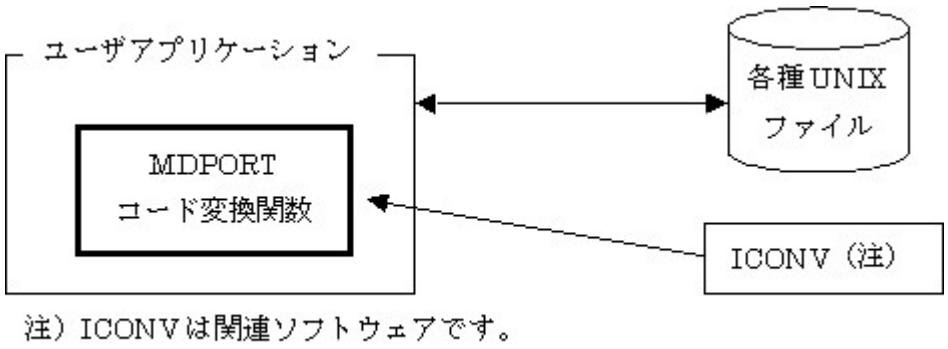


図3.7 コード変換機能のシステムフロー

[使用方法]

コード変換機能によるコード変換手順を図3.8に示します。

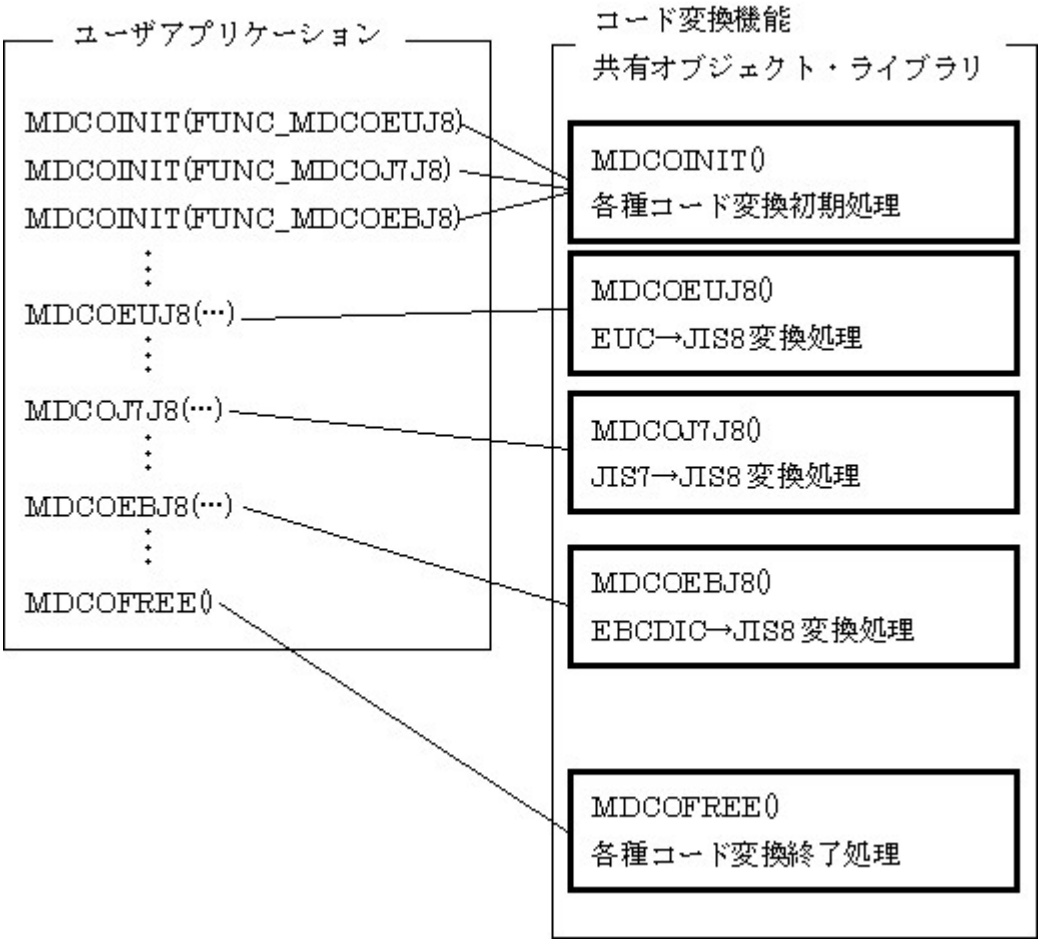


図3.8 コード変換機能によるコード変換手順

[実行環境]

コード変換機能では、関数の変換仕様に関わる環境変数を指定することができます。表3.11にコード変換機能で指定可能な環境変数を示します。その他の環境変数は、"3.5 実行環境"を参照して下さい。

表3.11 コード変換機能に関連する環境変数

環境変数	設定できる値 (下線は省略時)
MDPORT_JIS	jiskana jiskana8 jiskana7
MDPORT_CS3	ON OFF
MDPORT_ISO	ON OFF

MDPORT_JIS

変換元または変換後のコードがJISの時に有効で、半角・英数カナのモードを指定します。

jiskana	JISローマ字とJISカタカナの切り替えをエスケープシーケンスで行います。
jiskana8	1バイトコード(半角・英数カナ)を、8ビットコードで扱います。
jiskana7	JISローマ字とJISカタカナの切り替えをSI/SOで行います。

MDPORT_CS3

変換元または変換後のコードがEUCの時に、コードセット3(3バイト/1文字表現)を使用するかしないかを指定します。OFFにすると、コードセット3の文字が扱えなくなります。

詳細はmdportfコマンドの-nocd3オペランドの説明を参照してください。

MDPORT_ISO

EBCDICコードとASCIIコード間の変換における、変換仕様を指定します。OFFにすると、emuモードで変換されます。

詳細はmdportfコマンドの-emu | -isoオペランドの説明を参照してください。

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼び出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。

Cプログラムからの呼出

コード変換機能をCプログラムから呼び出す場合の使用方法を以下に示します。

[ヘッダファイル]

コード変換機能で使用する各関数やコード変換パラメタ構造体などは、ヘッダファイルとして提供されています。アプリケーションから使用する場合には、このヘッダファイルをインクルードしてください。

${MDPORT_DIR}/include/MDPcomm.h

[コンパイル方法]

コード変換機能は、共用オブジェクト(動的リンクライブラリ)として提供されます。コンパイル時には、MDP_MTプリプロセッサを指定し、リンク時にはlibmdpco_mt.soをライブラリ名[mdpco_mt]としてリンクして下さい。コンパイル/リンクする例を以下に示します。

```
$ cc -o prog1 -DMDP_MT -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -lricv -lmdpco_mt -I${MDPORT_DIR}/include
prog1.c
```

[使用例]

コード変換機能を行う一連の流れを以下に示します。詳細なソースコードについてはサンプルプログラムを参照してください。

${MDPORT_DIR}/sample/c/CODEsample_mt.c

ソースコード

```
#include "MDPcomm.h"                    /*ヘッダファイルのインクルード*/
:
struct MDCOPARA MDCOpara;               /*MDCOPARA構造体宣言*/
struct mdco_stat MDCOStat;              /*mdco_stat構造体宣言*/
char   indata[1024];
char   outdata[1024];
char OutFileName[1024];
int rtncd;
int    inlen;
 int    outlen;
:
/*************************************
* コード変換の初期処理(テキスト(JEF)→テキスト(EUC))
*************************************/
:
/* コード変換初期化関数呼出し */
rtncd = MDPINIT(
FUNC_MDCOEUJE,FJICV_ON,&
(MDCOpara.codtbl));
:
 :
 /*************************************
 * テキスト(JEF)→テキスト(EUC)の変換処理
*************************************/
 rtncd = MDCOEUJE( &MDCOpara, InData,
OutData );
 :
/*************************************
* エラー・ワーニング処理
*************************************/
MDCOSTAT(&MDCOStat, &MDCOpara);         /*詳細エラー情報の取得*/
if ( MDCOStat.st_etyp == MDCO_WAR)      /*条件付正常終了の場合*/
{
  :
```

```
}
else if ( MDCOStat.st_etyp == MDCO_ERR)          /*異常終了の場合*/
{
  if (MDCOStat.st_eno == MDCO_ENV) {             /*実行環境のエラー*/
       break;
  }
  else if (MDCOStat.st_eno == MDCO_PAR) {        /*パラメタのエラー*/
       break;
  }
  else {                                         /*その他のエラー*/
       break;
  }
}
 :
************************************/
  * レコード変換後処理
************************************/
/* 終了処理 */
MDCOFREE(&(MDCOpara.codtbl));
 :
 return(0);
```

[注 意]

- プログラムをコンパイルするときには、MDP_MTプリプロセッサを必ず指定してください。
- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。

COBOLプログラムからの呼出

COBOLプログラムからコード変換機能の関数を使用する場合には、プログラム間連絡機能を使用します。なお、詳細についてはCOBOL文法書、COBOL使用手引書等を参照してください。

[COBOL登録集]

コード変換機能で使用するMDPORT変換指示構造体などは、COBOL登録集として提供されています。アプリケーションから使用する場合には、このCOBOL登録集をプログラム中で展開してください。

```
${MDPORT_DIR}/include/MDPcomm_mt.cob
```

[コンパイル方法]

コード変換機能は、共用オブジェクト(動的リンクライブラリ)として提供されます。リンク時にはライブラリ名をmdpco_mtとしてlibmdpco_mt.soをリンクして下さい。コンパイル/リンクする例を以下に示します。

```
$ cobol -Tm -M -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -lthread -lricv -lmdpco_mt -
I${MDPORT_DIR}/include prog1.cob
```

[各機能の呼び出し方法]

以下にコード変換の各関数を呼び出す方法を説明します。

■MDPORT共通COBOL登録集の展開

変換情報などを定義しているCOBOL登録集のCOBOL登録集展開を行います。

```
000010    DATA DIVISION.
000020     WORKING-STORAGE SECTION.
             :
000050     COPY "MDPcomm_mt.cob".
             :
```

■MDPORT初期化処理の呼び出し

MDPORTコード変換情報を定義し、MDPORT初期化処理を行います。

```
000090    DATA DIVISION.
000100    WORKING-STORAGE SECTION.
             :
000160    01 ICONV                          PIC X(01).
             :
000230    PROCEDURE DIVISION.
             :
000260  *  MDPORTコード変換情報の初期化
000270     INITIALIZE MDCOPARA.
000280  *  ICONVを使用する設定
000290     MOVE 1 TO ICONV.
             :
000580  *  EUC→JEF変換を指定
000590     SET FUNC-MDCOEUJE                 TO TRUE.
000600  *  MDPORT初期化関数呼び出し
000610     CALL "MDCOINIT" USING             BY VALUE DEF-FUNC
000620                                       BY VALUE ICONV
000630                                       BY REFERENCE CODTBL.
```

■コード変換処理の呼び出し

コード変換処理を実行します。コード変換に渡すパラメータはあらかじめ初期化しておく必要があります。

```
000090    DATA DIVISION.
000100    WORKING-STORAGE SECTION.
             :
```

```
000260    01 入力データ                          PIC X(256).
000270    01 出力データ                          PIC X(256).
              :
000320    PROCEDURE DIVISION.
              :
000360  *  入力・出力情報長の設定
000370     MOVE 10                               TO INLEN.
000380     MOVE 30                               TO OUTLEN.
000390  *  コード変換処理を実行
000400     CALL "MDCOEUJE" USING                 MDCOPARA
000410                                           入力データ
000420                                           出力データ.
000430  *  コード変換エラー対処
000440     IF PROGRAM-STATUS NOT = 0 THEN
000450  *        エラー詳細情報の取得
000460           CALL "MDCOSTAT" USING           MDCO-STAT
000470                                           BY REFERENCE CODTBL
              :
000610     END-IF.
```

■MDPORT変換情報の解放

MDCOINITで初期化したMDPORT変換情報は、使用後に解放する必要があります。

```
000090    DATA DIVISION.

000200    WORKING-STORAGE SECTION.
              :
000320    PROCEDURE DIVISION.
              :
000500  *  コード変換情報の解放
000510     CALL "MDCOFREE" USING                 BY REFERENCE CODTBL.
```

■詳細情報の取得

コード変換等で発生したエラーの詳細情報を取得します。

```
000090    DATA DIVISION.
            :
000200    WORKING-STORAGE SECTION.
              :
000320    PROCEDURE DIVISION.
              :
000510  *  詳細情報の取得
000520     CALL "MDCOSTAT" USING                 MDCO-STAT
000530                                           BY REFERENCE CODTBL.
000510  *  環境設定エラー
000540     IF ST-ENO = DEF-ST-ENO THEN
              :
000550  * その他のエラー
000560     ELSE
              :
000570     END-IF.
```

[サンプルプログラム]

サンプルプログラムは以下のディレクトリにあります。

```
${MDPORT_DIR}/sample/COBOL/CODEsample_mt.cob
```

コンパイルを実行する場合は、以下のシェルスクリプトをご利用下さい。

${MDPORT_DIR}/sample/COBOL/CODEsample_mt.cob.sh

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。
- 本ライブラリは、内部的にthreadライブラリを使用しています。コンパイル時には、必ずthreadライブラリをmdpco_mtの前に定義してください。

Javaプログラムからの呼出

ここでは、サンプルを例にJNI(Java Native Interface)を使用してJavaプログラムからコード変換機能を使用する方法を説明します。

[プログラム構成]

Javaからコード変換機能を呼び出すプログラムの構成を図3.11に示します。

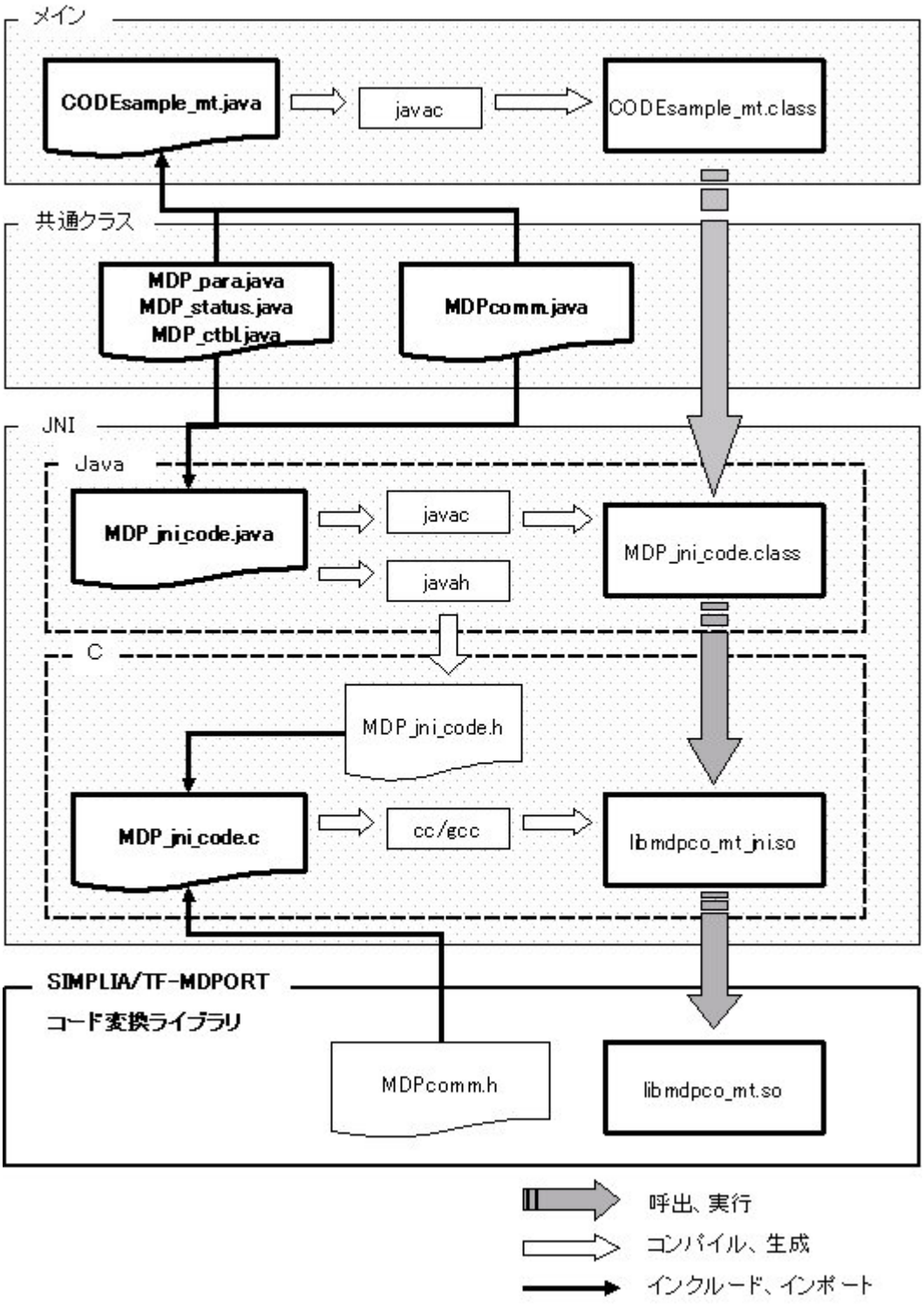


図3.11 JNIからコード変換機能を呼び出すプログラムの構成

[説明]

各プログラムの説明を以下に示します。

CODEsample_mt.java	このプログラムのメインプログラムです。MDP_jni_codeクラスのネイティブメソッドを呼び出し、変換するレコードを渡します。
MDP_para.java MDP_status.java MDP_ctbl.java	コード変換時に使用するMDCOPARA構造体情報やMDCOSTATのStat情報を定義した変換指示構造体を提供します。

MDPcomm.java	コード変換機能で使用する定数を定義したインターフェースクラス。MDPcomm.h、MDPcomm_mt.cobで定義されている定数を定義しています。
MDP_jni_code.java	コード変換機能の各メソッドを定義したクラス。メインプログラムでは、このクラスのネイティブメソッドを呼び出します。
MDP_jni_code.h	MDP_jni_codeクラスから抽出される、ライブラリ用のヘッダファイル。javahコマンドでMDP_jni_codeクラスを指定することで生成されます。
MDP_jni_code.c	Javaのネイティブメソッドから呼び出される、コード変換機能を実際に呼び出すプログラム。コンパイル時にコード変換機能をリンクします。

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。

3.6.1.1 コード変換機能(旧形式)[Solaris]

コード変換機能は、共用オブジェクト・ライブラリとして提供され、ユーザアプリケーションと動的にリンクすることにより、各種コードへのコード変換を行います。

[機能説明]

コード変換機能は以下の関数をユーザアプリケーションに組込むことにより、各種コード変換を行います。コード変換関数の詳細については“付録D コード変換/レコード変換機能(ライブラリ)"を参照して下さい。

№	関数名	機能
1	MDCOINIT()	使用するコード変換関数の宣言を行います。
2	MDCOFREE()	コード変換終了処理を行います。
3	MDCOSTAT()	詳細エラー情報を取得します。
4	MDCOEBEU()	EBCDICからEUC(半角英数カナ)に変換します。
5	MDCOEBJ7()	EBCDICからJIS7に変換します。
6	MDCOEBJ8()	EBCDICからJIS8に変換します。
7	MDCOEUEB()	EUC(半角英数カナ)からEBCDICに変換します。
8	MDCOEUJ7()	EUC(半角英数カナ)からJIS7に変換します。
9	MDCOEUJ8()	EUC(半角英数カナ)からJIS8に変換します。
10	MDCOEUJE()	EUCからJEFに変換します。
11	MDCOEUJI()	EUCからJISに変換します。
12	MDCOEUSJ()	EUCからSJISに変換します。
13	MDCOJ7EB()	JIS7からEBCDICに変換します。
14	MDCOJ7EU()	JIS7からEUC(半角英数カナ)に変換します。
15	MDCOJ7J8()	JIS7からJIS8に変換します。
16	MDCOJ8EB()	JIS8からEBCDICに変換します。
17	MDCOJ8EU()	JIS8からEUC(半角英数カナ)に変換します。
18	MDCOJ8J7()	JIS8からJIS7に変換します。
19	MDCOJEEU()	JEFからEUCに変換します。
20	MDCOJEJI()	JEFからJISに変換します。
21	MDCOJESJ()	JEFからSJISに変換します。
22	MDCOJIEU()	JISからEUCに変換します。
23	MDCOJIJE()	JISからJEFに変換します。
24	MDCOJISJ()	JISからSJISに変換します。
25	MDCOSJEU()	SJISからEUCに変換します。
26	MDCOSJJE()	SJISからJEFに変換します。
27	MDCOSJJI()	SJISからJISに変換します。

[動作環境]

コード変換機能のシステムフローを図3.7に示します。

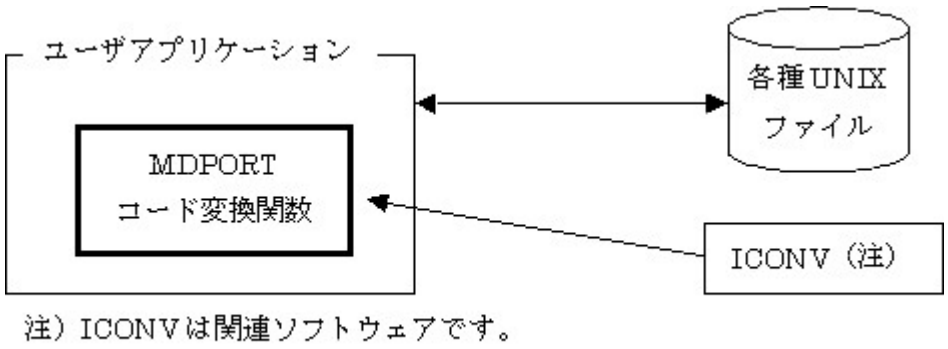


図3.7 コード変換機能のシステムフロー

[使用方法]

コード変換機能によるコード変換手順を図3.8に示します。

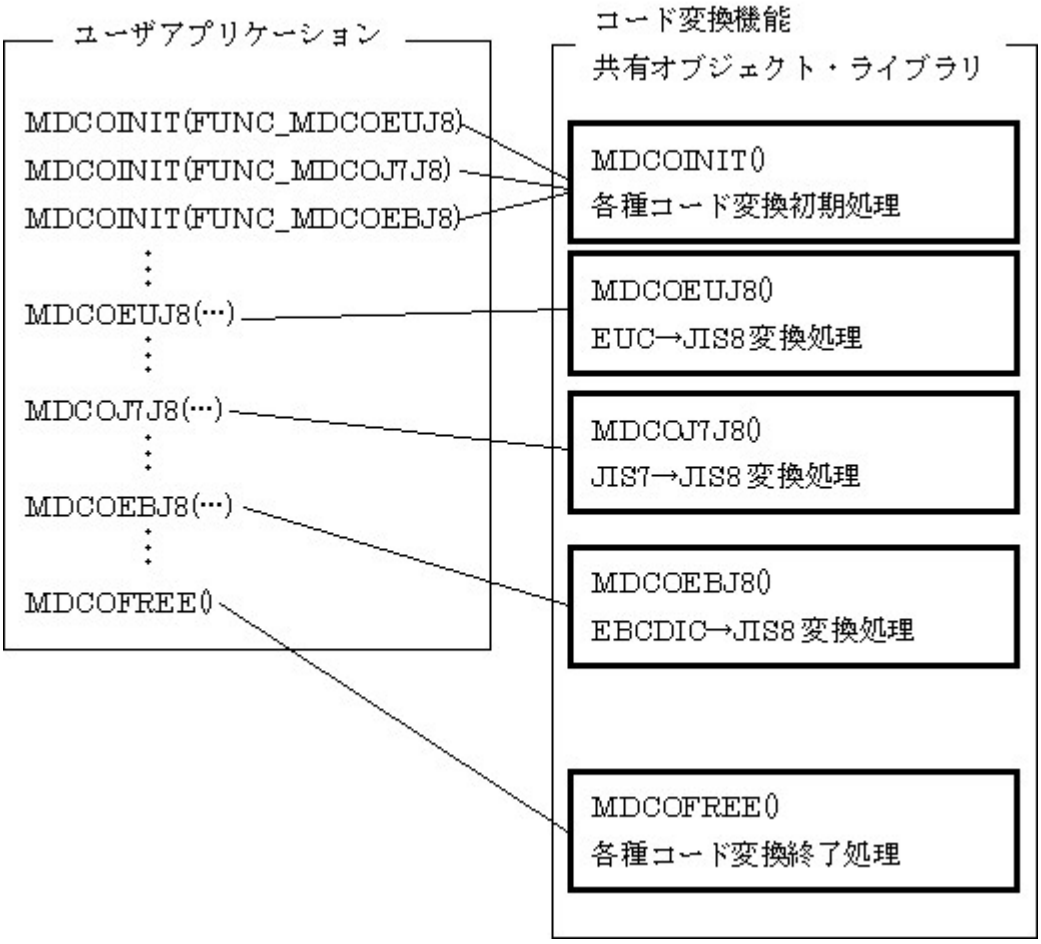


図3.8 コード変換機能によるコード変換手順

[実行環境]

コード変換機能では、関数の変換仕様に関わる環境変数を指定することができます。表3.11にコード変換機能で指定可能な環境変数を示します。その他の環境変数は、"3.5 実行環境"を参照して下さい。

表3.11 コード変換機能に関連する環境変数

環境変数	設定できる値 (下線は省略時)
MDPORT_JIS	jiskana jiskana8 jiskana7
MDPORT_CS3	ON OFF
MDPORT_ISO	ON OFF

MDPORT_JIS

変換元または変換後のコードがJISの時に有効で、半角・英数カナのモードを指定します。

jiskana	JISローマ字とJISカタカナの切り替えをエスケープシーケンスで行います。
jiskana8	1バイトコード(半角・英数カナ)を、8ビットコードで扱います。
jiskana7	JISローマ字とJISカタカナの切り替えをSI/SOで行います。

MDPORT_CS3

変換元または変換後のコードがEUCの時に、コードセット3(3バイト/1文字表現)を使用するかしないかを指定します。OFFにすると、コードセット3の文字が扱えなくなります。

詳細はmdportfコマンドの-nocd3オペランドの説明を参照してください。

MDPORT_ISO

EBCDICコードとASCIIコード間の変換における、変換仕様を指定します。OFFにすると、emuモードで変換されます。

詳細はmdportfコマンドの-emu | -isoオペランドの説明を参照してください。

[使用例]

C言語プログラムからコード変換機能の関数を使用する場合の記述例を示します。

```
#include "MDPcomm.h"
:
        struct CODPARA codpara;                 /* コード変換パラメタ   */
        char   indata[256];                     /* 入力データ           */
        char   outdata[256};                    /* 出力データ           */
:
        codpara.inlen = indata_len;             /* 入力データ長設定     */
        codpara.outlen = outdata_len;           /* 出力データ長設定     */
:
        /** コード変換初期処理変換パスの宣言 */
        /** EUC<-->JIS8,  JIS7<-->JIS8,  EBCDIC<-->JIS8 */
        rtn = MDCOINIT( FUNC_MDCOEUJ8 );        /* MDCOEUJ8()の宣言     */
        rtn = MDCOINIT( FUNC_MDCOJ7J8 );        /* MDCOJ7J8()の宣言     */
        rtn = MDCOINIT( FUNC_MDCOEBJ8 );        /* MDCOEBJ8()の宣言     */
:
        /** コード変換ライブラリ呼出し */
        rtn = MDCOEUJ8( &codpara, indata, outdata );    /* EUC -> JIS8      */
        rtn = MDCOJ7J8( &codpara, indata, outdata );    /* JIS7 -> JIS8     */
        rtn = MDCOEBJ8( &codpara, indata, outdata );    /* EBCDIC -> JIS8   */
:
        /** コード変換終了処理 */
        rtn = MDCOFREE();
        return(0);
```

COBOLプログラムからコード変換機能の関数を使用する場合の記述例を示します。

```
 WORKING-STORAGE SECTION.
*MDPORT共通COBOL 登録集の展開
 COPY  "MDPcomm.cob".
:
01 入力データ1                          PIC X(256).
01 出力データ1                          PIC X(256).
01 入力データ2                          PIC X(256).
01 出力データ2                          PIC X(256).
:
:
 PROCEDURE DIVISION.
:
*コード変換関数初期化
     SET FUNC_MDCOEUJ8 TO TRUE.
      CALL "MDCOINIT" USING BY VALUE DEF-FUNC.
      SET FUNC_MDCOJ7J8 TO TRUE.
      CALL "MDCOINIT" USING BY VALUE DEF-FUNC.
      SET FUN_MDCOEBJ8 TO TRUE.
      CALL "MDCOINIT" BY VALUE DEF-FUNC.
:
*コード変換パラメタの設定
      INITIALIZE MDCOPARA.
      MOVE 入力データ長WK TO INLEN.
* 代替文字設定
      MOVE  X' 3F'              TO ALT1.
      MOVE  X' A1A9'            TO ALT2.
:
*MDCOEUJ8呼出し
```

```
    CALL "MDCOEUJ8"  USING MDCOPARA
入力データ1.
出力データ1.
    IF PROGRAM-STATUS NOT = 0
:
:
*コード変換パラメタの設定
    INITIALIZE MDCOPARA.
    MOVE 入力データ長WK TO INLEN.
    MOVE 出力データ長WK TO OUTLEN.
:
*MDCOJ7J8呼出し
    CALL "MDCOJ7J8"  USING MDCOPARA
入力データ2.
出力データ2.
:
```

[コンパイル方法]

コード変換機能は、共用オブジェクト・ライブラリ(動的リンクライブラリ)として提供されます。コンパイル時にlibmdpco.soをリンクして下さい。コンパイル/リンクする例を以下に示します。

- C言語プログラムのコンパイル例
  ```
  $ cc -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -licv -lmdpco -I${MDPORT_DIR}/include prog1.c
  ```
- COBOLプログラムのコンパイル例
  ```
  $ cobol -M -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -licv -lmdpco -I${MDPORT_DIR}/include
  prog1.cob
  ```

[注 意]

- 本機能は、マルチスレッドアプリケーションには対応していません。
- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼び出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。

3.6.2 レコード変換機能

レコード変換機能は、以下の二つの機能に分類されます。

- mdportgコマンド
  mdportgコマンドは、レコード変換機能が、変換処理を行う際に使用するレイアウトファイルを生成します。
  レイアウトファイルとは、データ定義文を解析して作成される中間ファイルです。
  レコード変換機能を使用してデータ変換を行う場合は、あらかじめmdportgコマンドによりレイアウトファイルを生成しておく必要があります。
  V5L3以前のバージョンで作成したレイアウトファイルは、Solaris版V6/Linux for Itanium版V5L4以降のバージョンのレコード変換機能で使用できます。
  Solaris版V6/Linux for Itanium版V5L4以降のバージョンで作成したレイアウトファイルは、V5L3以前のバージョンのレコード変換機能で使用できません。
- レコード変換関数
  レコード変換関数は、共用オブジェクト・ライブラリとして提供され、ユーザアプリケーションと動的にリンクすることにより、各種コードへのコード変換を行います。
  mdportgコマンドとレコード変換関数の詳細については“付録D コード変換/レコード変換機能(ライブラリ)"を参照して下さい。

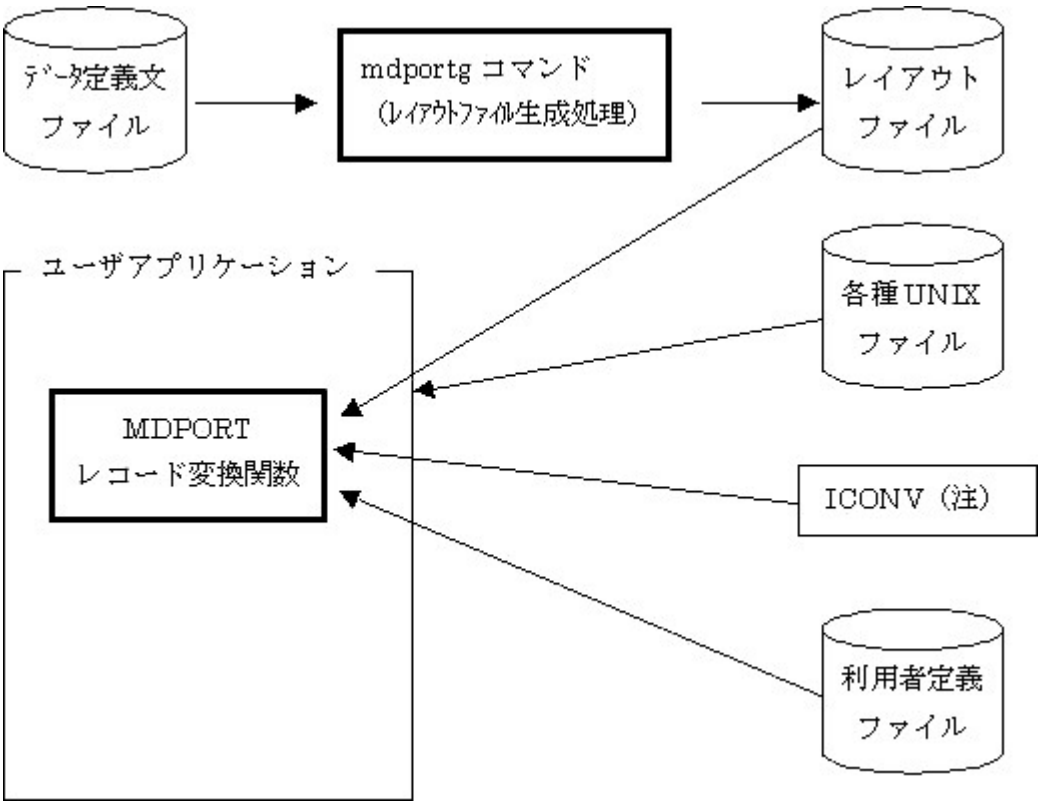
[機能説明]

レコード変換機能は、以下の関数をユーザアプリケーションに組込み、レコード単位にコード変換、レコード形式変換を行います。
レコード変換機能で提供される関数は以下のとおりです。

№	関数名	機能
1	MDP_init()	変換パスの宣言、レイアウトファイルの指定、変換に使用する領域の動的確保等、コード変換の初期処理を行います。
2	MDP_initmsg()	コード変換で発生したメッセージを格納するメッセージ格納領域を確保し、初期化処理を行います。
3	MDP_conv()	MDP_init()指示された内容に従ってコード変換、またはレコード形式変換を行います。
4	MDP_getmsg()	コード変換で発生したメッセージをメッセージ格納領域から取得します。
5	MDP_nextmsg()	メッセージ格納領域内の次メッセージへの移動を行います。
6	MDP_prevmsg()	メッセージ格納領域内の前メッセージへの移動を行います。
7	MDP_outputmsg()	コード変換時に発生したメッセージをファイルへ出力します。
8	MDP_freemsg()	MDP_initmsg()で確保したメッセージ格納領域の解放処理を行います。
9	MDP_fin()	MDP_init()で確保した領域の解放等、コード変換の終了処理を行います。
10	MDP_stat()	MDP_conv(), MDP_fin()で発生したエラーの詳細情報を取得します。

[動作環境]

レコード変換部品のシステムフローを図3.9に示します。



注)ICONVは関連ソフトウェアです。

図3.9 レコード変換部品のシステムフロー

[使用方法]

レコード変換機能によるコード変換手順を図3.10に示します。

手順1： mdportgコマンド(レイアウトファイル生成処理)を使用して、データ定義文ファイルと変換形式 指示をもとにレイアウトファイルを生成します。ただし、テキスト変換の場合、手順1は必要ありません。

手順2： ユーザアプリケーションで使用する変換パス毎に、MDPORT初期化処理(MDP_init)により必要情報(変換パス、レイアウトファイル名等)を宣言します。

手順3： メッセージ格納領域の確保処理(MDP_initmsg)を行います。

手順4： MDPORT変換処理(MDP_conv)によりコード変換、データ形式変換処理を行います。

手順5： メッセージ取得処理(MDP_getmsg)、次メッセージ移動処理(MDP_nextmsg)またはメッセージ出力処理(MDP_outputmsg)でコード変換で発生したメッセージを取得します。

手順6： メッセージ格納領域の解放処理(MDP_freemsg)を行います。

手順7： MDPORT終了処理(MDP_fin)で、終了処理を行います。

例：1つのアプリケーションで複数の変換パスを使用(データ定義文が異なる等)する場合

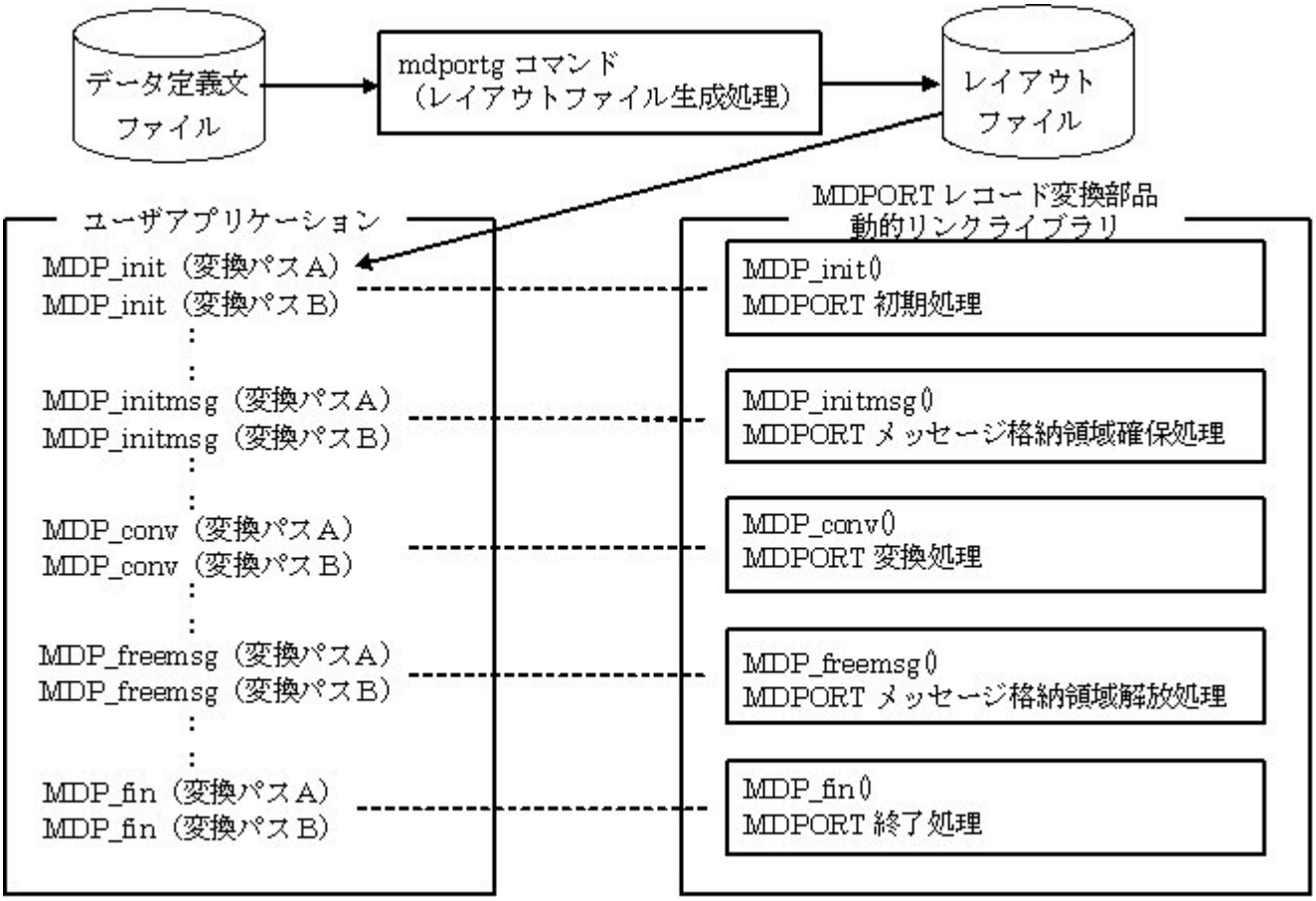


図3.10 レコード変換機能によるコード変換手順

[実行環境]

レコード変換機能では、関数の変換仕様に関わる環境変数を指定することができます。表3.12にレコード変換機能で指定可能な環境変数を示します。その他の環境変数は、"3.5 実行環境"を参照して下さい。

表3.12 レコード変換で使用する環境変数

環境変数	設定できる値 （下線は省略時）
MDPORT_JIS	jiskana jiskana8 jiskana7
MDPORT_CS3	ON OFF
MDPORT_ISO	ON OFF
MDPORT_ICONV	YES NO

MDPORT_JIS

変換元または変換後のコードがJISの時に有効で、半角・英数カナのモードを指定します。

jiskana : JISローマ字とJISカタカナの切り替えをエスケープシーケンスで行います。
jiskana8 : 1バイトコード(半角・英数カナ)を、8ビットコードで扱います。
jiskana7 : JISローマ字とJISカタカナの切り替えをSI/SOで行います。

MDPORT_CS3

変換元または変換後のコードがEUCの時に、コードセット3(3バイト/1文字表現)を使用するかしないかを指定します。OFFにすると、コードセット3の文字が扱えなくなります。
詳細はmdportfコマンドの-nocs3オペランドの説明を参照してください。

MDPORT_ISO

EBCDICコードとASCIIコード間の変換における、変換仕様を指定します。OFFにすると、emuモードで変換されます。
詳細はmdportfコマンドの-emu | -isoオペランドの説明を参照してください。

MDPORT_ICONV

コード変換にICONVを使用するかしないかを指定します。NOにするとICONVを使用せずにMDPORTの変換テーブルを使用します。
YESの場合に使用するICONVライブラリはコンパイル時に指定したリンケージオプションに依存します。

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。

Cプログラムからの呼出

レコード変換機能をCプログラムから呼び出す場合の使用方法を以下に示します。

[ヘッダファイル]

レコード変換機能で使用する各関数やMDPORT変換指示構造体などは、ヘッダファイルとして提供されています。アプリケーションから使用する場合には、このヘッダファイルをインクルードしてください。

${MDPORT_DIR}/include/MDPcomm.h

[コンパイル方法]

レコード変換機能は、共用オブジェクト(動的リンクライブラリ)として提供されます。コンパイル時には、MDP_MTプリプロセッサを指定し、リンク時にはlibmdp_mt.soをライブラリ名[mdp_mt]としてリンクして下さい。コンパイル/リンクする例を以下に示します。

```
$ cc -o prog1 -DMDP_MT -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -lricv -lmdp_mt -I${MDPORT_DIR}/include
prog1.c
```

[使用例]

レコード変換機能を行う一連の流れを以下に示します。詳細なソースコードについてはサンプルプログラムを参照してください。

${MDPORT_DIR}/sample/c/RECsample_mt.c

ソースコード

```
#include "MDPcomm.h"
:
struct MDPcvinf cvinf;
char   *mdpinf;
char   *msginf;
char   *msg;
char   indata[256];
char   outdata[256];
int    inlen;
 int    outlen;
:
/*************************************
* テキスト(JEF)→テキスト(EUC)の初期処
理
*************************************/
strcpy(cvinf.icode,  "jef");          /* 入力コード : JEF */
strcpy(cvinf.ocode,  "euc");          /* 出力コード : EUC */
cvinf.itype = TYPE_T;                 /* 入力形式 : テキスト */
cvinf.otype = TYPE_T;                 /* 出力形式 : テキスト */
cvinf.irecdlm = DLM_NO;               /* 入力区分文字なし */
cvinf.orecdlm = DLM_LF;               /* 出力区分文字 (LF) */
cvinf.iso = ISO;                      /* ISOモード変換 */
cvinf.okana = KANA_E;                 /* EUC CS2 */
cvinf.irecfm = RECFM_F;               /* 入力固定長レコード */
cvinf.amend1 = 0x2f;                  /* 1BYTE 代替文字"?" */
cvinf.amend2 = 0xa1a9;                /* 2BYTE 代替文字"？" */
cvinf.msgmax = 100;                   /* WARNING 最大件数 */
cvinf.msgcont = MSGCONT_ON;           /* 処理続行 */
cvinf.usrfname = "USRTBL";            /* 利用者定義テーブル */
:
/* レコード変換初期化関数呼出
し */
mdpinf = MDP_init( &cvinf );
:
/* メッセージ格納領域確保 */
msginf = MDP_initmsg( mdpinf );
```

```
:
 :
 /************************************
 * テキスト(JEF)→テキスト(EUC)の変換処理
************************************/
 rtncod = MDP_conv( mdpinf, indata, inlen, outdata, &outlen, msginf );
 :
/************************************
* メッセージ処理
************************************/
while (1) {
    msg = MDP_getmsg(msginf);
    if (msg == NULL) {
        break;
    }
    :
    if (MDP_nextmsg(msginf) == MDPORT_NG)
    {
        break;
    }
}
 :
/************************************
  * レコード変換後処理
************************************/
/* メッセージ格納領域解放 */
MDP_freemsg( msginf );
/* 終了処理 */
MDP_fin( mdpinf );
 :
 return(0);
```

[注 意]

- プログラムをコンパイルするときには、MDP_MTプリプロセッサを必ず指定してください。
- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。

COBOLプログラムからの呼出

COBOLプログラムからレコード変換機能の関数を使用する場合には、プログラム間連絡機能を使用します。なお、詳細についてはCOBOL文法書、COBOL使用手引書等を参照してください。

[COBOL登録集]

レコード変換機能で使用するMDPORT変換指示構造体などは、COBOL登録集として提供されています。アプリケーションから使用する場合には、このCOBOL登録集をプログラム中で展開してください。

```
${MDPORT_DIR}/include/MDPcomm_mt.cob
```

[コンパイル方法]

レコード変換機能は、共用オブジェクト(動的リンクライブラリ)として提供されます。リンク時にはライブラリ名をmdp_mtとしてlibmdp_mt.soをリンクして下さい。コンパイル/リンクする例を以下に示します。

```
$ cobol -Tm -M -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -lthread -lricv -lmdp_mt -
I${MDPORT_DIR}/include prog1.cob
```

[各機能の呼び出し方法]

以下にレコード変換の各関数を呼び出す方法を説明します。

■MDPORT共通COBOL登録集の展開

変換情報などを定義しているCOBOL登録集のCOBOL登録集展開を行います。

```
000010    DATA DIVISION.
000020      WORKING-STORAGE SECTION.
              :
000050      COPY "MDPcomm_mt.cob".
              :
```

■MDPORT初期化処理の呼び出し

MDPORTコード変換情報を定義し、MDPORT初期化処理を行います。

```
000090    DATA DIVISION.
000100    WORKING-STORAGE SECTION.
              :
000190    01 レイアウトＦ名                PIC X(64).
              :
000210    01 コード変換情報Ｐ              PIC S9(09) COMP-5.
              :
000250    PROCEDURE DIVISION.
              :
000260  *   MDPORTコード変換情報の初期化
000270      INITIALIZE MDPCVINF.
              :
000300  *   コード変換初期化関数のためのパラメタ設定
000310  *   入出力文字コード
000320      STRING "euc" X'00' DELIMITED BY SIZE
000330                                      INTO ICODE.
000340      STRING "sjis" X'00' DELIMITED BY SIZE
000350                                      INTO OCODE.

000380  *   入出力データ形式
000390      SET TYPE-D                      TO TRUE.
000400      MOVE DEF-TYPE                   TO ITYPE.
000410      SET TYPE-O                      TO TRUE.
000420      MOVE DEF-TYPE                   TO OTYPE.
              :
000500      STRING "sample.cob.mdg" X'00' DELIMITED BY SIZE
000510                                      INTO レイアウトＦ名.
```

```
              :
000550     MOVE FUNCTION ADDR(レイアウトＦ名) TO LAYFNAME.
              :
000580  *  MDPORT初期化関数呼び出し
000590     CALL "MDP_init"              USING MDPCVINF
                                        RETURNING コード変換情報Ｐ.
```

■メッセージ格納領域の初期化

メッセージ格納領域の初期化を行います。外部プログラムに渡すパラメータはMDP_initで初期化処理を行っておく必要があります。

```
000090     DATA DIVISION.
000100     WORKING-STORAGE SECTION.
              :
000210     01 コード変換情報Ｐ          PIC S9(09) COMP-5.
000220     01 メッセージ情報Ｐ          PIC S9(09) COMP-5.
              :
000250     PROCEDURE DIVISION.
              :
000300  *  メッセージ情報の初期化
000310     CALL "MDP_initmsg"           USING BY VALUE コード変換情報Ｐ
                                        RETURNING メッセージ情報Ｐ.
```

■コード変換処理の呼び出し

コード変換処理を実行します。コード変換に渡すパラメータはあらかじめ初期化しておく必要があります。

```
000090     DATA DIVISION.
000100     WORKING-STORAGE SECTION.
              :
000210     01 コード変換情報Ｐ          PIC S9(09) COMP-5.
000220     01 メッセージ情報Ｐ          PIC S9(09) COMP-5.
              :
000260     01 入力データ               PIC X(256).
000270     01 出力データ               PIC X(256).
000280     01 入力データ長             PIC S9(09) COMP-5.
000290     01 出力データ長             PIC S9(09) COMP-5.
000300     01 復帰値                   PIC S9(09) COMP-5.
              :
000320     PROCEDURE DIVISION.
              :
000390  *  コード変換実行
000400     CALL "MDP_conv"USING         BY VALUE コード変換情報Ｐ
000410                                  BY REFERENCE 入力データ
000420                                  BY VALUE 入力データ長
000430                                  BY REFERENCE 出力データ
000440                                  BY REFERENCE 出力データ長.
```

■メッセージの取得(その1)

コード変換で発生したメッセージをMDP_getmsg、MDP_nextmsg関数を使用して取得します。MDP_getmsgでNULLとなるか、MDP_nextmsgで失敗するまで繰り返しながら取得します。

```
000090     DATA DIVISION.
000100     BASED-STORAGE SECTION.
000110     01 MSG-AREA                  BASED ON MSG-POINT.
000120       03 MSG                     PIC X(80).
              :
000200     WORKING-STORAGE SECTION.
```

```
           :
000250   01 メッセージ情報Ｐ              PIC S9(09) COMP-5.
000260   01 メッセージ                    PIC X(80).
000270   01 MSG-POINT                     POINTER.
000280   01 MSG-FLG                       PIC X(1).
           :
000320   PROCEDURE DIVISION.
           :
000390     SET MDPORT-NG                  TO TRUE.
000400     MOVE SPACE                     TO MSG-FLG.
           :
000450 *   メッセージ取得
000460     PERFORM UNTIL MSG-FLG = "E"
000470       CALL "MDP_getmsg" USING      BY VALUE メッセージ情報Ｐ
000480                                    RETURNING MSG-POINT
000490       IF MSG-POINT NOT = NULL THEN
000500           MOVE MSG                 TO メッセージ
           :
000550           CALL "MDP_nextmsg" USING BY VLAUE メッセージ情報Ｐ
000560           IF PROGRAM-STATUS = DEF-MDPORT THEN
000570               MOVE "E"             TO MSG-FLG
000580           END-IF
000590       ELSE
000600           MOVE "E"                 TO MSG-FLG
000610       END-IF
000620     END-PERFORM.
```

■メッセージの取得(その2)

コード変換で発生したメッセージを取得します。MDP_outputmsgを使用し、ファイルに出力します。

```
000090   DATA DIVISION.

000200   WORKING-STORAGE SECTION.
           :
000250   01 メッセージ情報Ｐ              PIC S9(09) COMP-5.
000260   01 メッセージＦ名                PIC X(80).
000270   01 モード                        PIC S9(09) COMP-5.
           :
000320   PROCEDURE DIVISION.
           :
000380     STRING "sample.msg" X'00'      DELIMITED BY SIZE
000390                                    INTO メッセージＦ名.
000400     MOVE 2                         TO モード.
           :
000450     SET MDPORT-NG                  TO TRUE.
           :
000500 *   メッセージ取得
000510     CALL "MDP_outputmsg" USING     BY VALUE メッセージ情報Ｐ
000520                                    BY REFERENCE メッセージＦ名
000530                                    BY VALUE モード.
000540     IF PROGRAM-STATUS = DEF-MDPORT THEN
```

■メッセージ情報の解放

MDP_initmsgで初期化したメッセージ情報は、使用後に解放する必要があります。

```
000090   DATA DIVISION.

000200   WORKING-STORAGE SECTION.
           :
```

```
000250    01 メッセージ情報Ｐ                PIC S9(09) COMP-5.
            :
000320    PROCEDURE DIVISION.
            :
000500  *  メッセージ情報の解放
000510      CALL "MDP_freemsg" USING       BY VALUE メッセージ情報Ｐ.
```

■MDPORT変換情報の解放

MDP_initで初期化したMDPORT変換情報は、使用後に解放する必要があります。

```
000090    DATA DIVISION.

000200    WORKING-STORAGE SECTION.
            :
000250    01 コード変換情報Ｐ                PIC S9(09) COMP-5.
            :
000320    PROCEDURE DIVISION.
            :
000500  *  コード変換情報の解放
000510      CALL "MDP_fin" USING           BY VALUE コード変換情報Ｐ.
```

■詳細情報の取得

コード変換等で発生したエラーの詳細情報を取得します。

```
000090    DATA DIVISION.

000200    WORKING-STORAGE SECTION.
            :
000250    01 コード変換情報Ｐ                PIC S9(09) COMP-5.
000250    01 メッセージ情報Ｐ                PIC S9(09) COMP-5.
            :
000320    PROCEDURE DIVISION.
            :
000500      SET MDP-ERROR                  TO TRUE.
000510  *  詳細情報の取得
000520      CALL "MDP_stat" USING          BY VALUE DEF-MDP
000530                                     BY VALUE コード変換情報Ｐ
000540                                     BY VALUE メッセージ情報Ｐ.
```

[サンプルプログラム]

サンプルプログラムは以下のディレクトリにあります。

```
${MDPORT_DIR}/sample/COBOL/RECsample_mt.cob
```

コンパイルを実行する場合は、以下のシェルスクリプトをご利用下さい。

```
${MDPORT_DIR}/sample/COBOL/RECsample_mt.cob.sh
```

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。
- 本ライブラリは、内部的にthreadライブラリを使用しています。コンパイル時には、必ずthreadライブラリをmdp_mtの前に定義してください。

Javaプログラムからの呼出

ここでは、サンプルを例にJNI(Java Native Interface)を使用してJavaプログラムからレコード変換機能を使用する方法を説明します。

[プログラム構成]

Javaからレコード変換機能を呼び出すプログラムの構成を図3.11に示します。

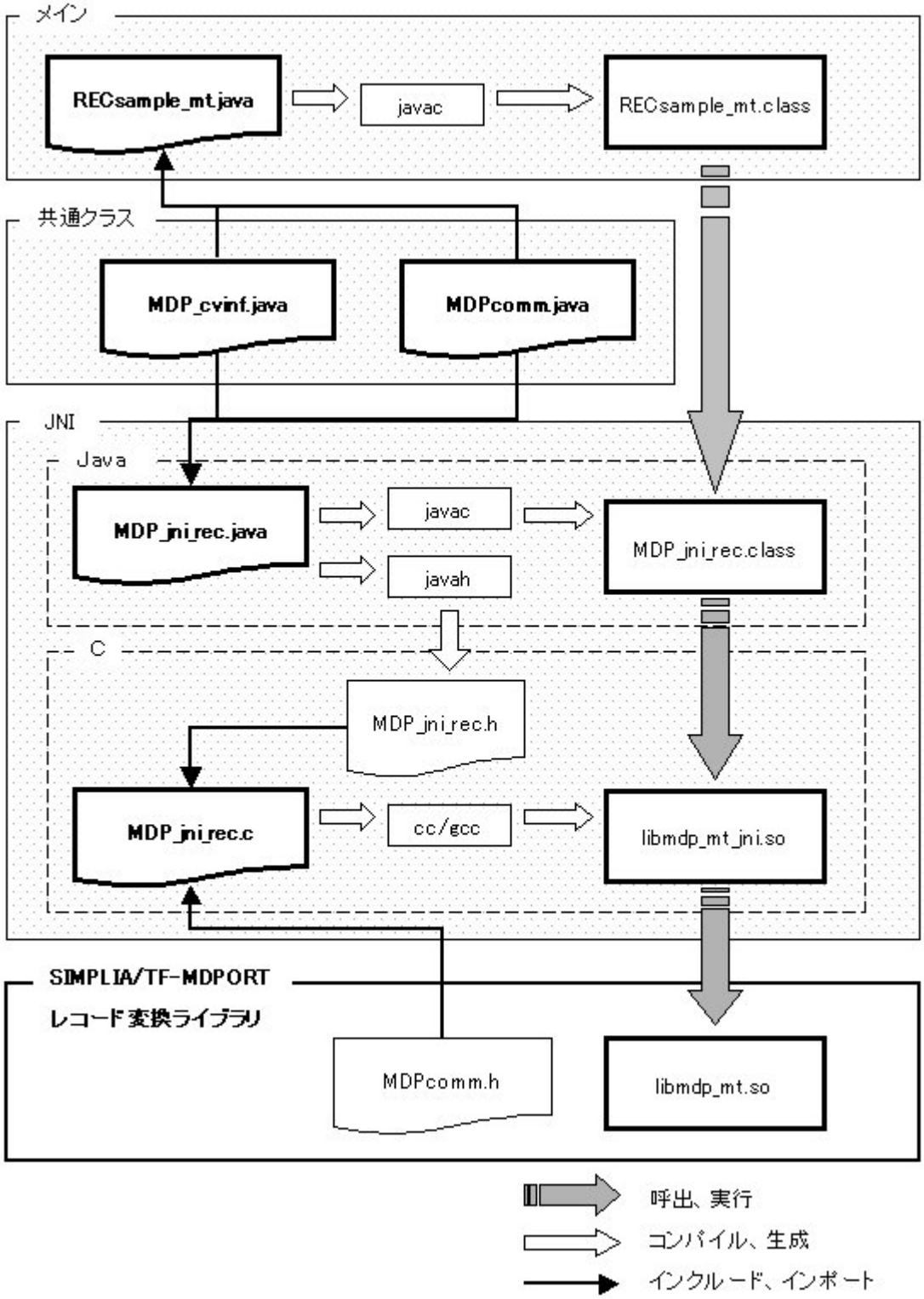


図3.12 JNIからレコード変換機能を呼び出すプログラムの構成

[説明]

各プログラムの説明を以下に示します。

RECsample_mt.java	このプログラムのメインプログラムです。MDP_jni_recクラスのネイティブメソッドを呼び出し、変換するレコードを渡します。
MDP_cvinf.java	レコード変換時に使用する変換指示情報を定義した変換指示構造体を提供します。
MDPcomm.java	レコード変換機能で使用する定数を定義したインターフェースクラ

	ス。MDPcomm.h、MDPcomm_mt.cobで定義されている定数を定義しています。
MDP_jni_rec.java	レコード変換機能の各メソッドを定義したクラス。メインプログラムでは、このクラスのネイティブメソッドを呼び出します。
MDP_jni_rec.h	MDP_jni_recクラスから抽出される、ライブラリ用のヘッダファイル。javahコマンドでMDP_jni_recクラスを指定することで生成されます。
MDP_jni_rec.c	Javaのネイティブメソッドから呼び出される、レコード変換機能を実際に呼び出すプログラム。コンパイル時にレコード変換機能をリンクします。

[注 意]

- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
  iconv関数を使用しない変換を行いたい場合は、 -lricvを省略してコンパイルします。

3.6.2.1 レコード変換機能(旧形式)[Solaris]

レコード変換関数には、共用オブジェクト・ライブラリとして提供され、ユーザアプリケーションと動的にリンクすることにより、各種コードへのコード変換を行います。レコード変換関数の詳細については“付録D コード変換/レコード変換機能(ライブラリ)"を参照して下さい。

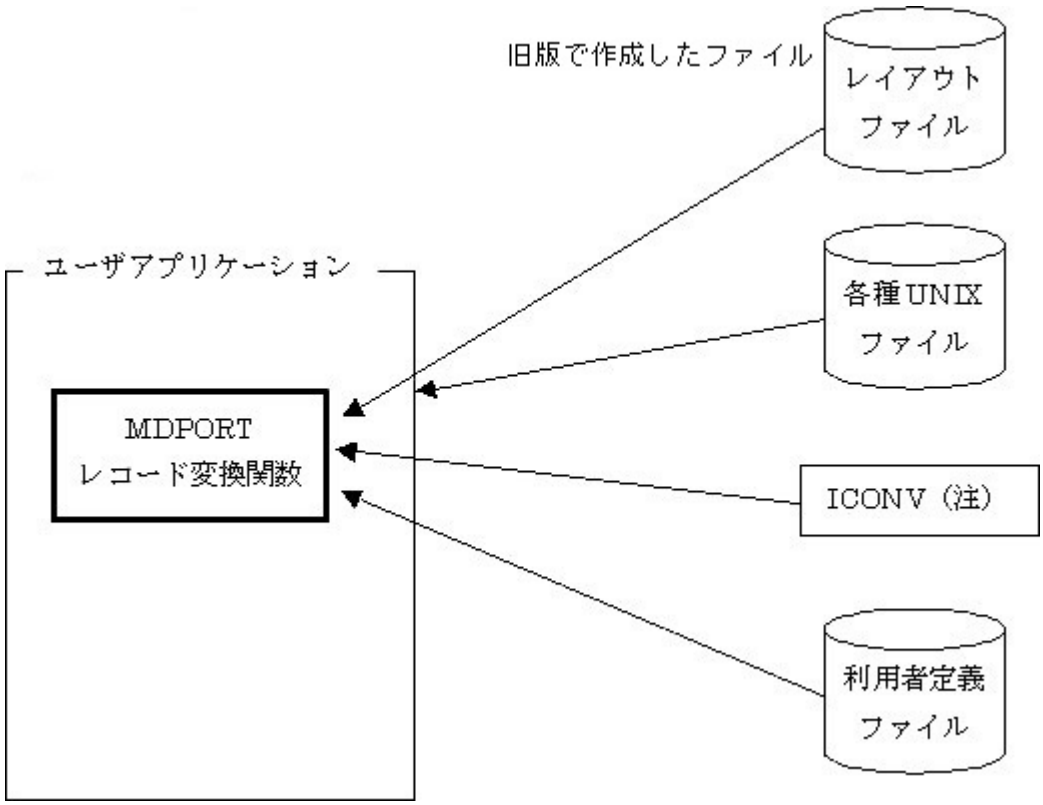
[機能説明]

レコード変換機能は、以下の関数をユーザアプリケーションに組込み、レコード単位にコード変換、レコード形式変換を行います。
レコード変換機能で提供される関数は以下のとおりです。

№	関数名	機能
1	MDP_init()	変換パスの宣言、レイアウトファイルの指定、変換に使用する領域の動的確保等、コード変換の初期処理を行います。
2	MDP_conv()	MDP_init()指示された内容に従ってコード変換、またはレコード形式変換を行います。
3	MDP_fin()	MDP_init()で確保した領域の解放等、コード変換の終了処理を行います。
4	MDP_stat()	MDP_conv(), MDP_fin()で発生したエラーの詳細情報を取得します。

[動作環境]

レコード変換部品のシステムフローを図3.13に示します。



注)ICONVは関連ソフトウェアです。

図3.13 レコード変換部品のシステムフロー

[使用方法]

レコード変換機能によるコード変換手順を図3.14に示します。
手順1： 旧版で作成したレイアウトファイルを用意します。
手順2： ユーザアプリケーションで使用する変換パス毎に、MDPORT初期化処理(MDP_init)により必要情報(変換パス、レイアウトファイル名等)を宣言します。
手順3： MDPORT変換処理(MDP_conv)によりコード変換、データ形式変換処理を行います。
手順4： MDPORT終了処理(MDP_fin)で、終了処理を行います。

例：1つのアプリケーションで複数の変換パスを使用(データ定義文が異なる等)する場合

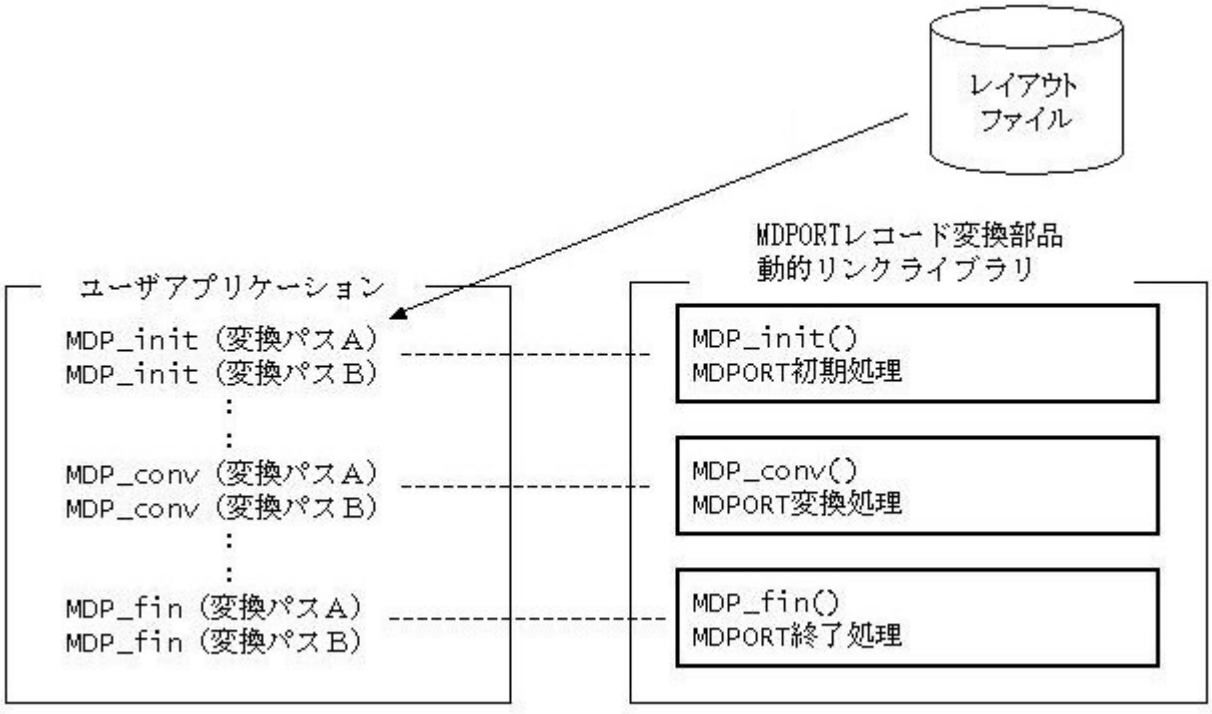


図3.14 レコード変換機能によるコード変換手順

[実行環境]

レコード変換機能では、関数の変換仕様に関わる環境変数を指定することができます。表3.12にレコード変換機能で指定可能な環境変数を示します。その他の環境変数は、"3.5 実行環境"を参照して下さい。

表3.12 レコード変換機能で使用する環境変数

環境変数	設定できる値 （下線は省略時）
MDPORT_JIS	jiskana jiskana8 jiskana7
MDPORT_CS3	ON OFF
MDPORT_ISO	ON OFF

MDPORT_JIS

変換元または変換後のコードがJISの時に有効で、半角・英数カナのモードを指定します。

jiskana : JISローマ字とJISカタカナの切り替えをエスケープシーケンスで行います。

jiskana8 : 1バイトコード(半角・英数カナ)を、8ビットコードで扱います。

jiskana7 : JISローマ字とJISカタカナの切り替えをSI/SOで行います。

MDPORT_CS3

変換元または変換後のコードがEUCの時に、コードセット3(3バイト/1文字表現)を使用するかしないかを指定します。OFFにすると、コードセット3の文字が扱えなくなります。

詳細はmdportfコマンドの-nocs3オペランドの説明を参照してください。

MDPORT_ISO

EBCDICコードとASCIIコード間の変換における、変換仕様を指定します。OFFにすると、emuモードで変換されます。

詳細はmdportfコマンドの-emu | -isoオペランドの説明を参照してください。

[使用例]

C言語プログラムからレコード変換機能の関数を使用する場合の記述例を示します。

```
#include "MDPcomm.h"
:
struct MDPcvinf cvinf[2];
char   *mdpinf[2];
char   indata[256];
```

```
char   outdata[256];
int    inlen;
 int    outlen;
:
/************************************
* テキスト(JEF)→テキスト(EUC)の初期処
理
************************************/
strcpy(cvinf[0].icode,  "jef");                 /* 入力コード : JEF          */
strcpy(cvinf[0].ocode,  "euc");                 /* 出力コード : EUC          */
cvinf[0].itype = TYPE_T;                        /* 入力形式 : テキスト       */
cvinf[0].otype = TYPE_T;                        /* 出力形式 : テキスト       */
cvinf[0].irecdlm = DLM_NO;                      /* 入力区分文字なし          */
cvinf[0].orecdlm = DLM_LF;                      /* 出力区分文字 (LF)         */
cvinf[0].iso = ISO;                             /* ISOモード変換             */
cvinf[0].okana = KANA_E;                        /* EUC CS2                   */
cvinf[0].irecfm = RECFM_F;                      /* 入力固定長レコード        */
cvinf[0].amend1 = 0x2f;                         /* 1BYTE 代替文字"?"         */
cvinf[0].amend2 = 0xa1a9;                       /* 2BYTE 代替文字"？"        */
cvinf[0].msgmax = 100;                          /* WARNING 最大件数          */
cvinf[0].msgcont = MSGCONT_ON;                  /* 処理続行                  */
cvinf[0].msgfname = "/var/tmp/MDP01.msg";       /* MSG 出力ファイル          */
cvinf[0].usrfname = "USRTBL";                   /* 利用者定義テーブル        */
/* レコード変換初期化関数呼出し */
mdpinf[0] = MDP_init( &cvinf[0] );
:
/************************************
* データ(SJIS)→CSVローダ型(EUC)の初期
処理
************************************/
strcpy(cvinf[1].icode,  "sjis");                /* 入力コード : SJIS         */
strcpy(cvinf[1].ocode, "euc");                  /* 出力コード : EUC          */
cvinf[1].itype = TYPE_D;                        /* 入力形式 :データ          */
cvinf[1].otype = TYPE_R;                        /* 出力形式 :RDBローダ       */
cvinf[1].orecdlm = DLM_LF;                      /* 出力区切文字(LF)          */
cvinf[1].okana = KANA_E;                        /* EUC CS2                   */
cvinf[1].irecfm = RECFM_F;                      /* 入力固定長レコード        */
cvinf[1].amend1 = 0x2f;                         /* 1BYTE 代替文字"?"         */
cvinf[1].amend2 = 0xa1a9;                       /* 2BYTE 代替文字"？"        */
cvinf[1].delimita = ',';                        /* 区切り文字","             */
cvinf[1].msgmax = 100;                          /* WARNING 最大件数          */
cvinf[1].msgcont = MSGCONT_ON;                  /* 処理続行                  */
cvinf[1].msgfname = "/var/tmp/MDP02.msg";       /* MSG 出力ファイル          */
/* レコード変換初期化関数呼出し */
mdpinf[1] = MDP_init( &cvinf[1] );
 :
 /************************************
 * テキスト(JEF)→テキスト(EUC)の変換処
理
 ************************************/
 rtncod = MDP_conv( mdpinf[0], indata, inlen, outdata, &outlen );
 :
 /************************************
 * データ(SJIS)→CSVローダ型(EUC)の変換
処理
```

```
************************************/
rtncod =MDP_conv( mdpinf[1], indata, inlen, outdata, &outlen );
 :
 /************************************
  * レコード変換後処理
************************************/
MDP_fin( mdpinf[0] );
MDP_fin( mdpinf[1] );
 :
 return(0);
```

COBOLプログラムからレコード変換機能の関数を使用する場合の記述例を示します。

```
 WORKING-STORAGE SECTION.
*MDPORT共通COBOL登録集の展開
COPY "MDPcomm.cob".
:
01 レイアウトF名                    PIC X(64).
01 利用者定義F名                    PIC X(64).
01 メッセージF名                    PIC X(64).
;
01 コード変換情報:                  PIC S9(09) COMP-5.
01 入力データ                       PIC X(256).
01 出力データ                       PIC X(256).
01 入力データ長                     PIC S9(09) COMP-5.
01 入力データ長                     PIC S9(09) COMP-5.
:
:
PROCEDURE DIVISION.
:
INITIALIZE レコード変換情報.
*コード変換初期化関数のためのパラメタ設定(文字列にはNULLを付加)
* 入出力文字コード
STRING "euc"   X'00'                DELIMITED BY SIZE
                                    INTO ICODE.
STRING "jef"    X'00'               DELIMITED BY SIZE
                                    INTO OCODE.
* 入力形式=データ形式.
SET    TYPE-D                       TO    TRUE.
MOVE DEF-TYPE                       TO    ITYPE.
*    出力形式=RDBローダ型(ORACLE)
SET    TYPE-O                       TO    TRUE.
MOVE DEF-TYPE                       TO    OTYPE
*    出力改行コード(LF)
SET   DLM-LF                        TO    TRUE.
MOVE DEF-TYPE                       TO    ORECLDM
* レイアウトF名、利用者定義F名、メッセージF名
STRING "/home/usr01/C01.cob.mdg"  X'00'  DELIMITED BY SIZE
                                    INTO レイアウトF名.
STRING "/home/usr01/USRTBL" X'00'   DELIMITED BY SIZE
                                    INTO 利用者定義F名.
STRING "/var/tmp/MDPORT.msg"  X'00'  DELIMITED BY SIZE
                                    INTO メッセージF名.
*    各ファイル名アドレス設定
MOVE FUNCTION ADDR(メッセージF名) TO MSGFILE.
MOVE FUNCTION ADDR(利用者定義F名) TO USRFNAME.
```

```
MOVE FUNCTION ADDR(レイアウトF名) TO LAYFNAME.
*レコード変換初期化関数呼出し
CALL "MDP_init" USING MDPCVINF.
*    MDP_init()関数の戻り値判定
IF PROGRAM-STATUS = 0 THEN
:
*初期化関数の戻り値(アドレス)を設定
MOVE PROGRAM-STATUS TO コード変換情報P.
 *レコード変換関数呼出し
CALL "MDP_conv" USING                BY VALUE コード変換情報P
                                     BY REFERENCE 入力データ
                                     BY VALUE 入力データ長
                                     BY REFERENCE 出力データ
                                     BY VALUE 出力データ長.
*    MDP_conv()関数の戻り値判定
 IF PROGRAM-STATUS NOT = 0 THEN
:
*レコード変換終了
CALL "MDP_fin" USING BY VALUE コード変換情報P.
:
```

[コンパイル方法]

レコード変換機能は、共用オブジェクト(動的リンクライブラリ)として提供されます。コンパイル時にlibmdp.soをリンクして下さい。コンパイル/リンクする例を以下に示します。

- C言語プログラムのコンパイル例
  $ cc -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -licv -lmdp -I${MDPORT_DIR}/include prog1.c
- COBOLプログラムのコンパイル例
  $ cobol -M -o prog1 -L${MDPORT_DIR}/lib -L/opt/FSUNiconv/lib -licv -lmdp -I${MDPORT_DIR}/include prog1.cob

[注 意]

- 本機能は、マルチスレッドアプリケーションには対応していません。
- 本機能は、iconv関数が使用できる場合、内部的にiconvを呼出しています。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。

3.6.3 旧形式からの移行のポイント

旧形式のコード変換/レコード変換機能を使用している既存のアプリケーションを移行する場合には、リンクするライブラリのほかに、修正する点があります。

- コード変換/レコード変換機能の各関数について
- ICONV(コード変換ライブラリ)について
- メッセージについて
- コンパイルオプション、COBOL登録集について
- レイアウトファイルの再作成について

[各関数について]

新規に追加された関数、関数のパラメータが変更になった関数があります。それぞれについては"3.6.1 コード変換機能", "3.6.2 レコード変換機能"を参照してください。

[ICONV(コード変換ライブラリ)について]

- Javaアプリケーションなどでは、JavaVMの起動時にOS標準のICONVをロードするため、プログラムのコンパイル時に標準コード変換ライブラリをリンクしても、起動することができません。このため、これらのアプリケーションから呼び出すICONV(標準コード変換)を指定する場合には、変換指示構造体のfjiconvにFJICV_ONを指定する必要があります。
- C及びCOBOLアプリケーションでは、ビルド時の-licv指定以外に、変換指示構造体のfjconvとの組み合わせによりコード変換方法が異なります。

fjiconv -licv	FJICNV_ON	FJICNV_OFF
有	iconv(標準コード変換)を動的ロードして、変換を行います。	ビルド時にリンクした標準コード変換ライブラリを使用して、変換を行います。
無	ライブラリのロードに失敗した場合、異常終了します。	MDPORT内部の標準コード変換テーブルを使用します。

[メッセージについて]

- 旧形式では、MDP_conv()関数で発生したメッセージは、変換指示構造体のmsgfnameに指定したメッセージファイルにコード変換が行われるたびに出力していましたが、この変換指示構造体のmsgfnameは使用されなくなりました。
- レコード変換で発生したメッセージは、MDP_conv()関数が呼ばれるごとに、MDP_initmsg()関数で指定したメッセージ格納領域を初期化して出力します。このメッセージ格納領域からメッセージを取得するには、呼び出すプログラム側でMDP_getmsg()関数、MDP_nextmsg()関数、MDP_prevmsg()関数もしくはMDP_outputmsg()関数を使用して取り出す必要があります。
- MDP_conv()関数は、呼び出されるたびにメッセージ領域を初期化します。このため、以前行ったコード変換時のメッセージ情報は取得することはできません。
- 変換指示構造体のmsgmaxでは1～9999の正の整数を指定してください。
- MDP_outputmsg()関数で出力されるメッセージの形式は、旧形式のレコード変換機能でmsgfnameに指定したメッセージファイルとは形式が異なります。メッセージファイルの出力形式等は"付録C 出力メッセージ"を参照してください。

[コンパイルオプション、COBOL登録集等について]

使用している言語によって、ヘッダーファイルやCOBOL登録集が異なります。それぞれについては"Cプログラムからの呼出", "COBOLプログラムからの呼出"を参照してください。

[レイアウトファイルの再作成について]

レコード変換機能で使用するレイアウトファイルは、mdportgコマンドで再作成する必要があります。

3.7 バリデーション機能(Charset Validator)を適用した変換

MDPORTは、文字コード変換を行う際に、特定の文字種ポリシーに従い変換を行うことができます。その機能をバリデーション機能と呼んでいます。
特定の文字種ポリシーとは、"バリデーションポリシーファイル"に定義された文字の運用ポリシー（方針）であり、現行システムで使用できない文字をブロックします。

[機能説明]

Interstage Charset Manager が提供するCharset Validator 関数を利用し、文字コード変換と同時にポリシーチェックも行います。
指定した文字種ポリシーに違反した文字は、ポリシーエラーとして出力され、代替文字が出力されます。
文字種ポリシーチェックの対象は、文字項目です。定義しているすべての文字項目に指定した文字種ポリシーが適用されます。
Interstage Charset Manager が提供するCharset Validator関数の詳細な説明については、「Interstage Charset Managerのバリデーション編」を参考にしてください。

[動作環境]

バリデーション機能(Charset Validator)を適用した変換のシステムフローを図3.15に示します。

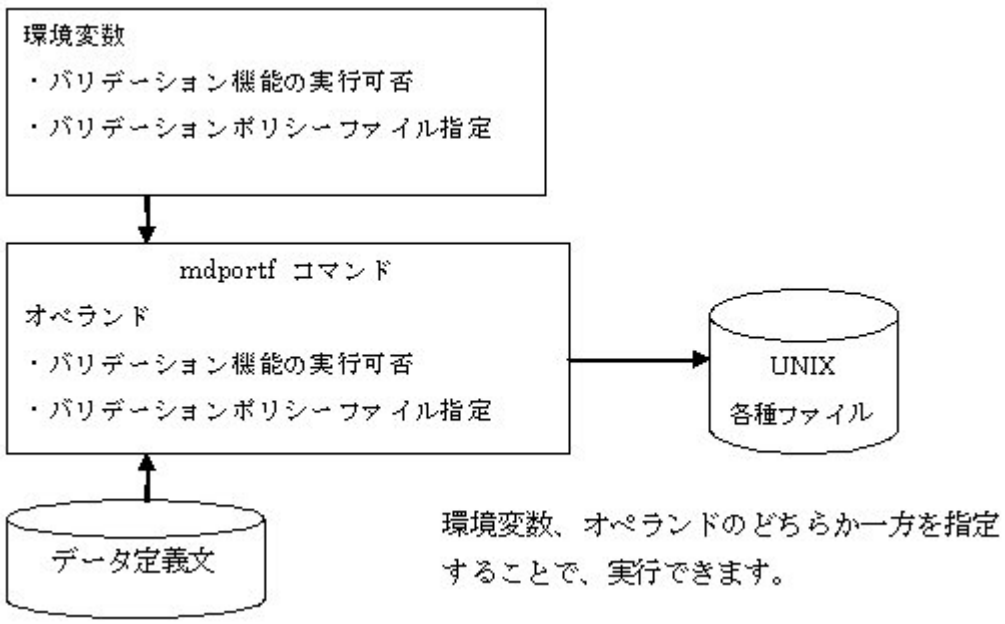


図3.15 バリデーション機能(Charset Validator)を適用した変換のシステムフロー

[使用方法]

バリデーション機能(Charset Validator)を適用した変換を行うには、mdportfコマンドのオペランドにより、「バリデーション機能を適用した変換を行う／行わない」の指定(-validation)と、「バリデーションポリシーファイル名」の指定(-policyf)を行う必要があります。
レコード変換機能では、MDPORT変換指示構造体で「バリデーション機能を適用した変換を行う／行わない」の指定(validation)と、「バリデーションポリシーファイル名」の指定(policyfname)を行います。

バリデーションポリシーファイルとは、Charset Validator 関数が使用するファイルで、システムで利用する文字の集合と、その中の各文字について入力可／不可が定義されているファイルです。
（詳細な説明は、「Interstage Charset Managerのバリデーション編」を参考にしてください。）

また、-validation、-policyfオペランド、それぞれに対応した環境変数も用意していますので、オペランドで指定せず、環境変数で指定することもできます。

mdportfコマンドのオペランドについては、"バリデーション機能を適用した変換を行う／行わない」の指定(-validation)"
"バリデーションポリシーファイル名の指定(-policyf)"

環境変数については、"バリデーション機能(Charset Validator)を適用した変換を行う際の環境変数"
を参照してください。

[オペランド、環境変数の指定による有効性]

バリデーション機能(Charset Validator)を適用した変換を文字項目に対して行うときは、オペランドまたは環境変数を定義して実行します。
以下に、オペランド、環境変数の指定による有効性を説明します。

表3.16 実行の有効性

[-validation]オペランド もしくは、 環境変数	[-policyf]オペランド もしくは、 環境変数	文字種ポリシーチェックを使用した変換
オペランド(yes) もしくは 環境変数(yes)	指定なし	エラー
	指定あり	全項目実行
オペランド(no) もしくは 環境変数(no,指定なし)	指定なし	しない
	指定あり	

※オペランドと環境変数が同時に指定されている場合は、オペランドを優先します。

[注 意]

- バリデーションポリシーファイルはC言語用で作成しなければなりません。
- バリデーション機能を適用した変換を入力側、出力側同時に実施することはできません。環境変数により入力側、出力側コード系に正しくバリデーション機能を適用した変換が定義されている場合は、入力側の文字コードで変換を実施します。
- 入力側、出力側ともUnicodeの場合、入力側でバリデーション機能を適用した変換を実施します。
- バリデーション機能を適用した変換を行う際、利用者定義変換テーブルの扱いの注意点を説明します。
  変換元あるいは変換先で文字種ポリシーチェックを行い、ある変換「xxxx」→「yyyy」がポリシーエラーになっているとします。このとき利用者定義変換テーブルにおいて変換元コードに「xxxx」または変換先コードに「yyyy」を定義した場合は、利用者定義変換テーブルの定義を優先し変換を行います。設定したポリシーと矛盾のない変換を行いたい場合にはポリシーエラーとしているコードの定義には注意が必要です。指定ポリシーに矛盾した利用者定義変換テーブルの指定は利用者の責任となります。

  例：
  jef → ucs2bでの変換で 「①（0x77C9)」 → 「①(0x2460)」 の変換をポリシーエラーにしているとします。
  このとき 変換元に0x77C9を定義した場合は指定通りの変換を行います。
  #jef ucs2b
  77C9 : xxxx xxxxは任意のコード

  変換先に0x2460を定義した場合もポリシーエラーとしません。
  #jef ucs2b
  yyyy : 2460 yyyyは任意のコード 指定通りの変換を行う。

  またポリシーエラーの変換をそのまま記述してもポリシーエラーとしません。
  #jef ucs2b
  77C9 : 2460 ポリシーエラーにはしない。この指定通りの変換を行う。

- MDPORTではX,M,Nの属性項目により変換できる文字が異なります。対象の属性項目でサポートしていないコードなどをポリシーエラーに設定した場合は、通常の変換エラー扱いになります。

表3.17 属性によりサポートする文字

属性	サポートする文字
X項目	半角文字(TAB、アルファベット、数字、カナ) ※TAB以外の制御コードは変換対象としません。
N項目	全角文字（UCS2間の変換では半角文字も可能）
M項目	半角文字と全角文字

例1：
ucs2bで0x0000や0x0001等の制御文字をポリシーエラーとなるように指定してもポリシーエラーにはなりません。通常の変換エラーになります。

例2：
ucs2bで0x0041等のアルファベットをポリシーエラーとなるように指定してもN項目では相手先コードがucs2以外の場合ポリ

シーエラーにはなりません。通常の変換エラーとなります。

例3：
ucs2bで0x0000～0x001F(TABを除く),0x007Fの制御コードの範囲はMDPORTの仕様によりバリデーション機能で有効にしても変換可能にはなりません。

付録A 使用例

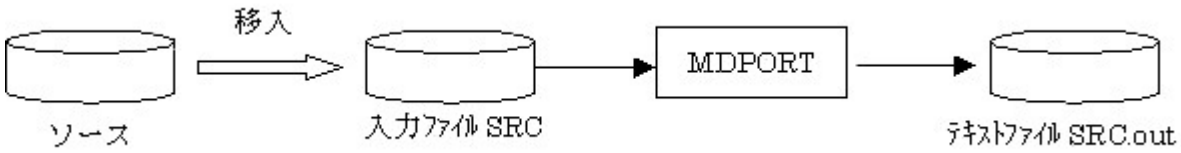
ここでは、適用パターンをあげてMDPORTの使用例を示しています。相手側機種上での各種ファイル編成と、外部媒体またはファイル転送を行うまでの手順については、"第2章 相手側交換手順"を参照してください。

A.1 汎用機上の一般ソースからのテキスト移入

[要 件]

EBCDICおよびJEFコードで記述されている固定長80バイトの一般ソースを、Solaris/Linux上のテキストファイルへ移入する。

[フロー図]



[コマンド呼出形式]

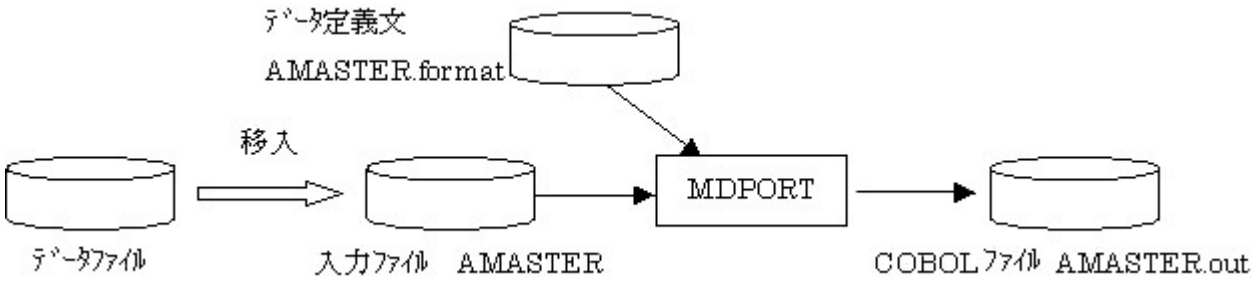
mdportf -f SRC -itype text -icode jef

A.2 データファイルからSolaris/Linux上のCOBOLファイルへの変換

[要 件]

EBCDICおよびJEFコードで構成される順編成ファイルを、そのままSolaris/Linux上COBOLレコード順ファイルへ移入する。

[フロー図]



[コマンド呼出形式]

mdportf -f AMASTER -itype data -otype cblfile -icode jef

[データ定義文例]

AMASTER.formatの内容

```
000100      01  AMASTER-REC.
000200        03  FILLER          PIC  X(20).
000300        03  FILLER          PIC  9(05).
000400        03  FILLER          PIC S9(07) PACKED-DECIMAL.
000500        03  AAA-G           OCCURS  7.
000600          05  FILLER        PIC  X(08).
000700          05  FILLER        PIC S9(04) BINARY.
000800        03  FILLER          PIC  X(50).
```

[備 考]

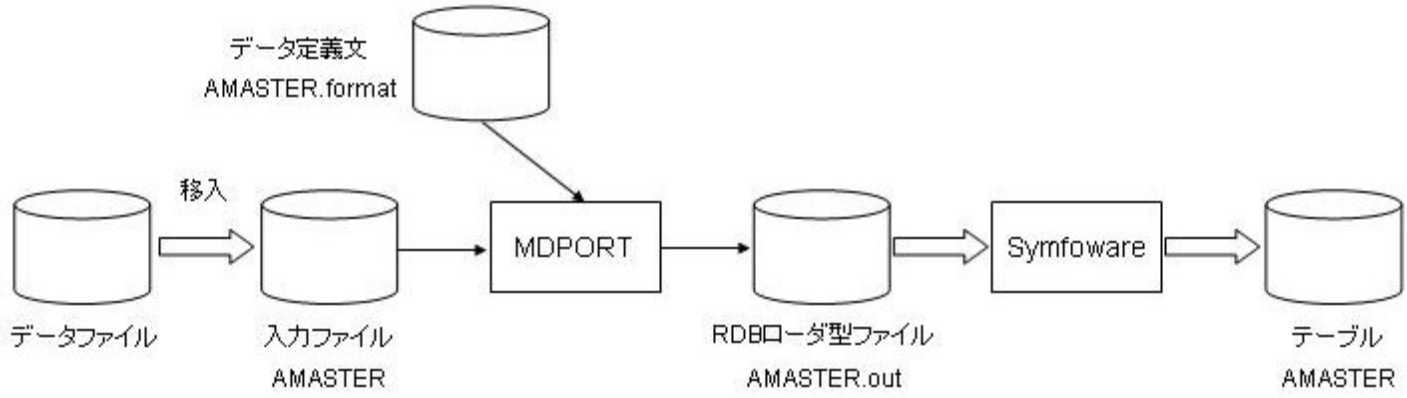
順ファイル以外のCOBOLファイルへ出力する場合は、-ocobflオペランドにより指定する。
相手側機種が富士通機以外の汎用機では、-icodeに入力データのコード体系を指定する。

A.3 データファイルからSymfowareローダ型ファイルへの変換

[要 件]

EBCDICおよびJEFコードで構成される固定長の順編成ファイルを、Symfoware向けのローダ型ファイルへ変換する。

[フロー図]



[コマンド呼出形式]

mdportf -f AMASTER -itype data -otype rdb -rdb2 -icode jef

[データ定義文例]

AMASTER.formatの内容

```
000100     01  AMASTER-REC.
000200       03  ITEM1          PIC  X(20).
000300       03  ITEM2          PIC  9(05).
000400       03  ITEM3          PIC S9(07) PACKED-DECIMAL.
000500       03  AAA-G.
000600         05  ITEM4        PIC  X(08).
000700         05  ITEM5        PIC S9(04) BINARY.
000800D      03  ITEM6          PIC  X(50).
```

⇐ 指定の領域を出力しない

[ローダの使用例]

```
rdbsloader -mi -i DDD.AMASTERDSI -t
AMASTER.out
```

[RDBテーブルの定義]

```
CREATE TABLE
S1.AMASTER
  (ITEM1 CHAR(30),
   ITEM2 INTEGER,
   ITEM3 INTEGER,
   ITEM4 CHAR(8),
   ITEM5 INTEGER)
```

A.4 汎用機上の一般ソースへのテキスト移出

[要 件]

Solaris/Linux上のテキストファイルから、EBCDICおよびJEFコード記述の固定長80バイトの一般ソースへ移出する。

[フロー図]

[コマンド呼出形式]

mdportf -f SRC -itype text -otype text -ocode jef -oreclen 80

[備 考]

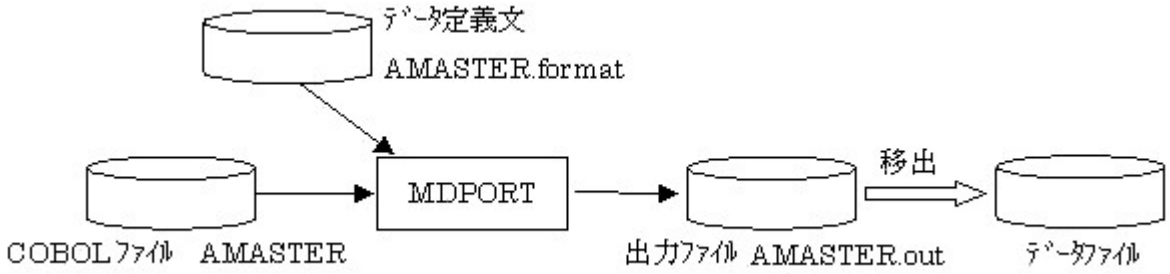
-oreclenオペランドの省略値が80バイトのため、-oreclenオペランドは省略可能。

A.5 Solaris/Linux上のCOBOLファイルからデータファイルへの変換

[要 件]

Solaris/Linux上のCOBOLレコード順ファイルから、EBCDICおよびJEFコードで構成される固定長の順編成ファイルへ移出する。

[フロー図]



[コマンド呼出形式]

mdportf -f AMASTER -itype cblfile -otype data -ocode jef

[データ定義文例]

AMASTER.formatの内容

```
000100     01  AMASTER-REC.
000200       03  FILLER          PIC  X(20).
000300       03  FILLER          PIC  9(05).
000400       03  FILLER          PIC S9(07) PACKED-DECIMAL.
000500       03  AAA-G.          OCCURS 7.
000600         05  FILLER        PIC  X(08).
000700         05  FILLER        PIC S9(04) BINARY.
000800       03  FILLER          PIC  X(50).
```

[備 考]

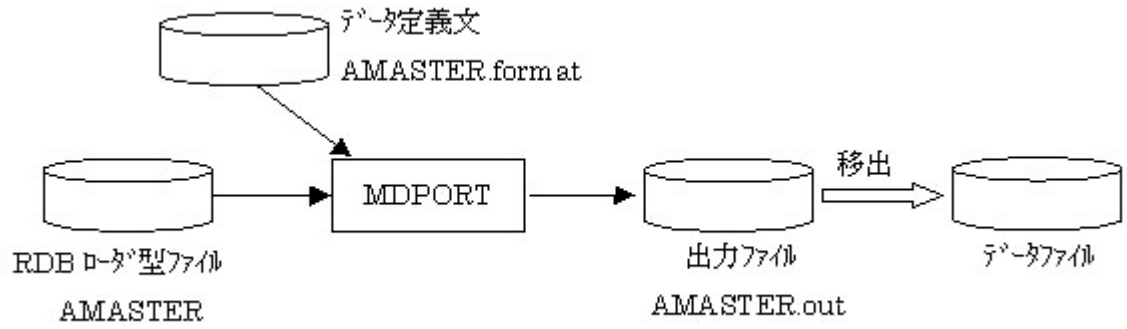
順ファイル以外のCOBOLファイルから入力する場合は、-icobflオペランドにより指定する。
相手側機種が富士通機以外の汎用機では、-ocodeに入力データのコード体系を指定する。

A.6 RDBローダ型ファイルからデータファイルへの変換

[要 件]

RDBローダ型ファイルから、EBCDICおよびJEFコードで構成される固定長のデータファイルへ移出する。

[フロー図]



[コマンド呼出形式]

mdportf -f AMASTER -itype rdb -otype data -ocode jef

[データ定義文例]

AMASTER.formatの内容

```
000100     01  AMASTER-REC.
000200       03  FILLER          PIC  X(20).
000300       03  FILLER          PIC  9(05).
000400       03  FILLER          PIC S9(07) PACKED-DECIMAL.
000500       03  FILLER          PIC  X(50).
```

[備 考]

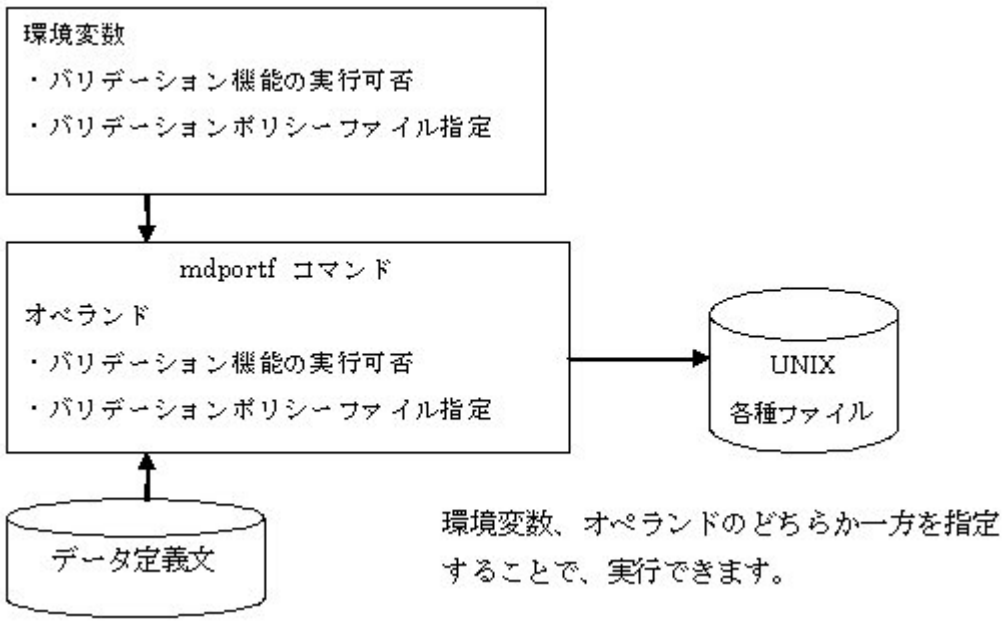
入力のRDBローダ型ファイルは、データ項目間が区切り文字","(-delimita省略時)で区切られていなければならない。
相手側機種が富士通機以外の汎用機では、-ocodeに入力データのコード体系を指定する。

A.7 バリデーション機能(Charset Validator)を適用した変換

[要 件]

文字コード変換を行う際、バリデーション機能(Charset Validator)を適用した変換を行う。

[フロー図]



[使用例]

バリデーション機能(Charset Validator)を適用した環境として、policy.bin がバリデーションポリシーファイル、格納場所は /home/mdport、データ定義文は data.format とした場合に、その環境においてバリデーション機能を使用するには、オペランド、環境変数の組み合わせ方がいろいろあります。用途に合わせて組み合わせてください。
使用例を以下に示します。

例1: mdportf -validation yes -icode ucs2b -ocode euc

-policyf オペランド及び環境変数(MDPORT_VALIDATION_PATH, MDPORT_VALIDATE_UNICODE)の指定が無い場合、バリデーション機能(Charset Validator)を適用した変換を行ってもエラーとなります。

例2:
```
setenv MDPORT_VALIDATE_UNICODE /home/mdport/policy.bin
mdportf -validation yes -icode ucs2b -ocode euc -formatf data.format
```

-policyf オペランドの指定がなくても、バリデーションポリシーファイルが環境変数に指定されているので、その内容に従い入力コードucs2bに対して、バリデーション機能(Charset Validator)を適用した変換を行います。

例3: mdportf -validation yes -policyf /home/bin/policy.bin -icode ucs2b -ocode euc

バリデーションポリシーファイル(policy.bin)に従い、入力コードucs2bに対して、全項目にバリデーション機能(Charset Validator)を適用した変換を行います。

例4: mdportf -validation no -policyf policy.bin -icode ucs2b -ocode euc -formatf data.format

バリデーションポリシーファイルを指定しても、-validation no 指定の場合はバリデーション機能を適用した変換は行いません。

例5:
```
setenv MDPORT_VALIDATION yes
setenv MDPORT_VALIDATION_PATH /home/mdport/
setenv MDPORT_VALIDATE_UNICODE policy.bin
mdportf -icode euc -ocode ucs2b
```

環境変数で指定された /home/mdport 配下にあるバリデーションポリシーファイル(policy.bin)を適用した変換を行います。

[備 考]

バリデーション機能(Charset Validator)の詳細な説明は、"3.7 バリデーション機能(Charset Validator)を適用した変換"を参照してください。

付録B 変換仕様

ここでは、MDPORTのファイル交換機能における、コード・属性・形式変換の詳細を説明します。

B.1 1バイト系コードの変換仕様

EBCDIC(カナ)コードとUnicode/EUC/シフトJISコード間での変換では、一部の記号文字において代替文字への変換が発生します。付表B.1に1バイト系コード変換による代替文字を示します。また、EUC/シフトJIS/UnicodeからEBCDICの変換では、英小文字は英大文字に代替変換します(-emuオペランド指定時)。

シフトJISコード/EUCコード/Unicode間での変換では、規則的な変換を行うため、代替文字は発生しません。ただし、コード定義の違いにより"¥"は"＼"(半角のバックスラッシュ)となります。

付表B.1 EBCDIC(カナ)-EUC/シフトJIS/Unicode変換による代替文字

EBCDIC(カナ)			£ (4A)	! (5A)	¬(5F)
EUC	-emu	] (5D)	[(5B)	! (21)	^ (5E)
シフトJIS Unicode	-iso	! (21)	[(5B)	] (5D)	^ (5E)

備考. EUC/シフトJIS/Unicodeの-emu, -isoは、mdportfコマンドのオペランドを示す。

EUCコードのカナ文字は、mdportfコマンドの-ikana, -okanaオペランドによりコードセット2またはJIS8コードとして扱われます。

文字として定義されていない1バイト系コードが変換時に発見された場合は、変換不能文字とし-amend1オペランドに指定されたコードで出力されます。

B.2 2バイト系コードの変換仕様

MDPORTでの各種2バイト系コード(日本語コード)の変換仕様を以下に示します。変換仕様で、規則的な変換とされているものは、JISコードに準拠されたコードの並びで対応させた変換のことを示します。
MDPORTでの変換は、デフォルトでiconvを使用しているので、詳細の変換仕様は該当マニュアルを参照してください。

[EUCコード]

U90またはS90コードを前提としています。JEFコードおよびシフトJISコードとの変換では、iconvを内部処理として使用しており、iconvの変換仕様に係わる環境変数の指定が有効となります。
なお、U90コード独自の拡張文字やJEF拡張漢字の表示・印刷には、U90提供のフォントを使用する必要があります。そのため、一般的なPC上のtelnet等では、U90フォントは表示することができません。

[JEFコード]

MDPORTでのJEFコードは、84JEFを対象としています。
iconv機能にもとづく字体重視変換/領域重視変換の切替えは、環境変数の指定により行うことができます。字体重視変換(環境変数省略時)では、83JISコード体系と不規則な対応の文字("塚", "靴", "槙"等の旧字体)はコードセット3(3バイト/文字)へ変換されます。
JEF拡張漢字/JEF拡張非漢字の変換では、対応するEUCコードは1文字につき2バイトあるいは3バイト(コードセット3)になります。ただし、-nocs3オペランドを指定した場合は、コードセット3の文字は変換不能になり、2バイト系代替文字コード(-amend2オペランドに指定されたコード)で出力されます。
一部のJEF拡張非漢字(ローマ数字, "(株)", "(有)", "Tel"等のOASYS拡張文字)の変換では、U90コードのコードセット1(2バイト/文字)の94区に対応づけされます。

[シフトJISコード]

83年JIS準拠のシフトJISを対象にしています。

[Unicode]

Unicode形式として、UTF8/UCS2をサポートしています。(UCS2はビッグエンディアンとリトルエンディアン形式をサポート)

[JISコード]

jisおよびjiseでは90年版のJISコードを対象にしています。
jis78およびjis78eでは78年版のJISコードを対象にしています。

[IBM日本語コード]

IBM日本語コードは、83JIS対応版を対象としています。変換対象文字はJIS規格の範囲です。
ただし、以下のJIS範囲外の文字については、U90コードセット1の拡張文字との変換を行います。

｜ ’ ” (株) No. Tel [ローマ数字] ｉ ～ ｘ　Ⅰ～Ⅹ

[日立KEISコード]

日立KEISコードでは、83JISに準拠した規則的な変換を行います。83JISコード体系の範囲外については変換できません。

[日本電気JIPSコード]

JIPS(E), JIPS(J)コードでは、83JISに準拠した規則的な変換を行います。83JISコード体系の範囲外については変換できません。

各コード体系における日本語コードの検出は、以下のように行われます。

[シフトコードによる検出(入力コードがJEF, 他社コードの場合)]

テキストモード変換およびデータファイル変換の混在項目では、日本語コードへの切替制御コード(シフトコード)が出現した場合に日本語コードであることを検出します。データファイル変換の日本語項目では、無条件に日本語コードとして扱われます。

[コードの値による検出(入力コードがEUC, シフトJISコードの場合)]

文字コードの先頭ビットにより1バイト系または2バイト系コードかの判断をし、日本語コードであることを検出します。

シフトJISコードの場合は、0x81〜0x9F, 0xE0〜0xFCで始まるコードを日本語コードとし、EUCコードの場合は0xA1〜0xFEで始まるコードを日本語コードとします。また、EUCコードの場合、0x8Fで始まるコードはコードセット3の日本語コードとみなします。

[コードの値による検出(入力コードがUTF8、UCS2の場合)]

UTF8の場合、先頭の1バイトコードにより1〜6バイトコードかの判断をし、日本語コードを検出します。ただし、4バイト以上のコードは全て変換エラーとなります。UCS2の場合、常に2バイトコードの判断をし、日本語コードを検出します。(1バイト目 == 0x00 かつ 2バイト目 ≦ 0x7F) または(1バイト目 == 0xFF かつ 0x61 ≦ 2バイト目 ≦ 0x9F)の場合を除き日本語コードを判断します。(ビッグエンディアン形式の場合)

B.3 テキストモードの変換

テキストモードでの変換では、入出力のコード体系によって変換仕様が異なります。

EBCDIC系からASCII系への変換

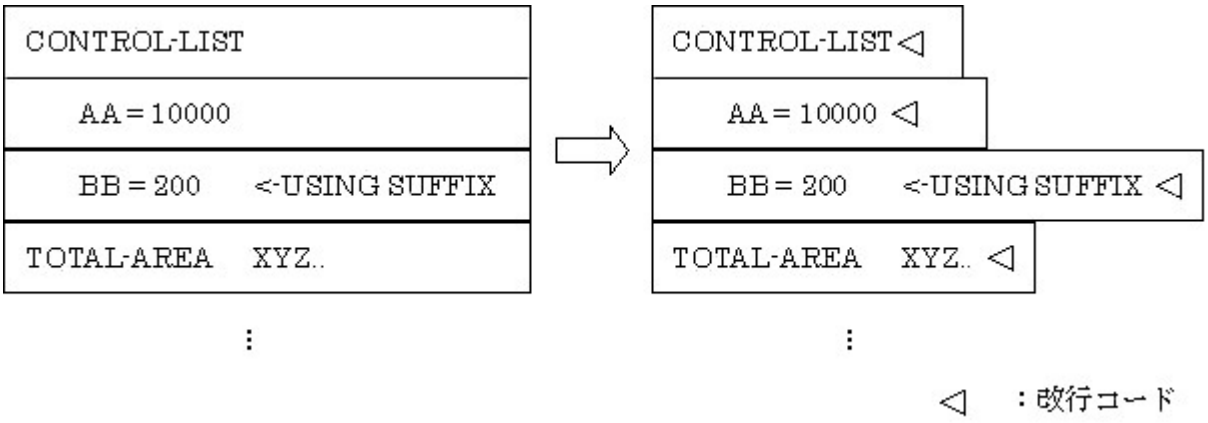
入力レコードが固定長の場合は、指定されたレコード長(-ireclenオペランド)でファイル全体の領域を分割し、コード変換が行われます。
入力レコードが可変長の場合は、各レコードの先頭のRDW(Record Descriptor Word)よりレコード長を認識して、レコード分割を行います。
コード変換後、各レコードの末尾ブランクは削られ、行端に改行コードが挿入されて出力されます。
付図B.1にEBCDIC系コードからのテキストモード変換を示します。

ASCII系からASCII系への変換

入力レコードは、改行コードで区切られているものとして処理されます。また、入力レコードの末尾ブランクは削られて出力されます。

ASCII系からEBCDIC系への変換

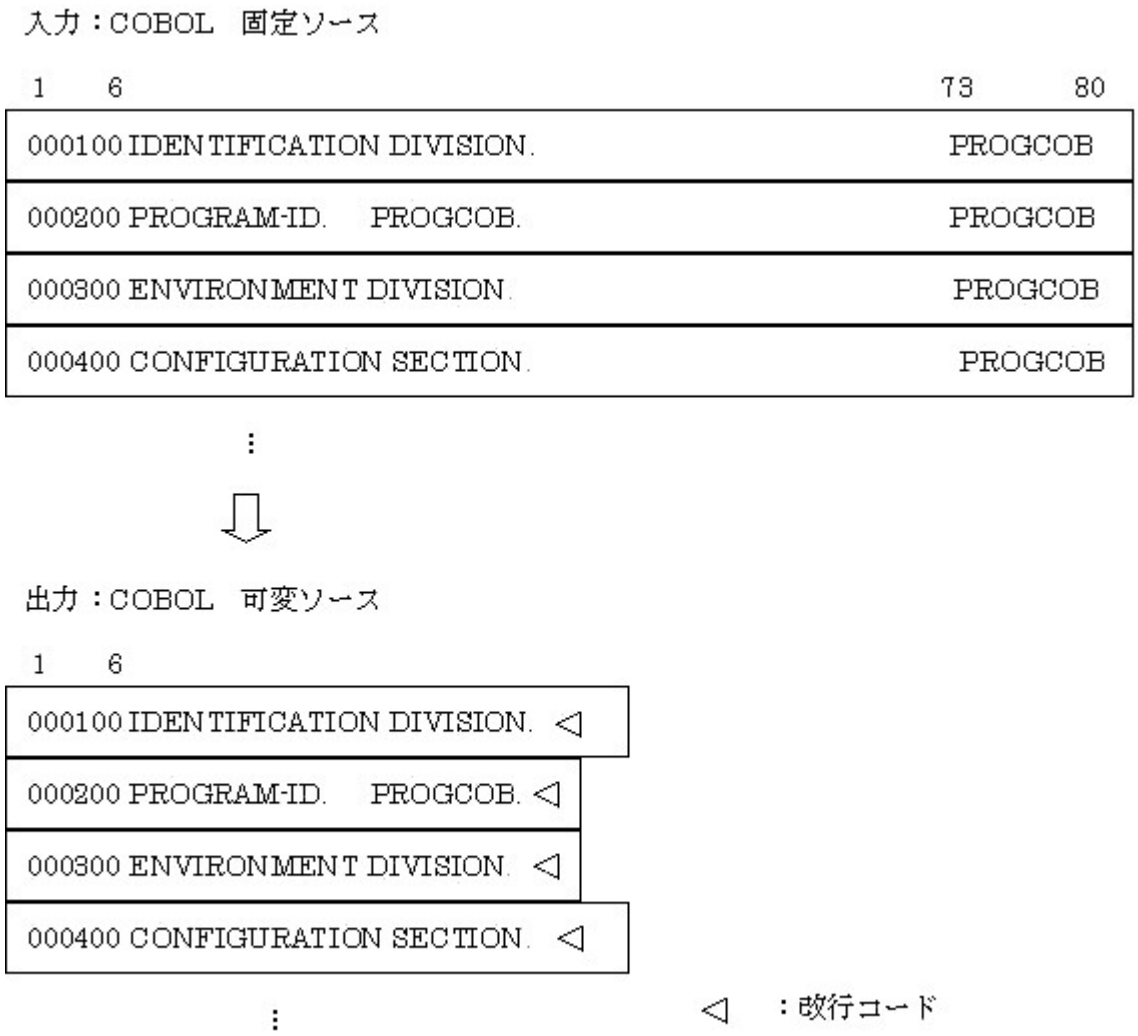
入力レコードは、改行コードで区切られているものとして処理されます。出力レコードは、-orecfmオペランドで指定されたレコード属性で出力されます。固定長出力の場合は、-oreclenオペランドで指定されたレコード長で出力されます。レコード長を超える部分は削られ、満たない部分には空白が埋められます。可変長出力の場合は、各レコードの後続ブランクは削られ、レコードの先頭には4バイトのRDW(Record Descriptor Word)が付加されます。



付図B.1 EBCDIC系コードからのテキストモード変換

B.4 COBOLソース型の変換

COBOLソース型の変換は、テキストモード変換の固定長レコードEBCDIC系→ASCII系変換の変換仕様とほぼ同じです。相違点としては、COBOL仕様において固定ソース形式の73～80カラムがプログラム識別領域のため、EUCコードの可変ソース形式へ出力する際にプログラム識別領域を取り除きます。付図B.2にCOBOLソース型の変換を示します。



付図B.2 COBOLソース型の変換

B.5 データファイル変換での変換仕様

データファイル変換を行う場合、データ定義文を元にコード変換を行います。本節では、各データ項目属性ごとの変換仕様について説明します。

B.5.1 数値項目における変換仕様

EBCDICコード系とASCIIコード系との数値項目では、外部10進(USAGE DISPLAY指定)の数値データ属性において内部表現が異なります。付表B.2 外部10進におけるEBCDIC/ASCIIコード系の相違を示します。
内部10進(PACKED DECIMAL)については相違はありません。

付表B.2 外部10進におけるEBCDIC/ASCIIコード系の相違

外部10進での相違点	内部表現(16進)	
	EBCDICコード	ASCIIコード
ゾーンビット (4ビット)	F	3
正の符号ビット (4ビット)	C	4
負の符号ビット (4ビット)	D	5

内部10進	内部表現(16進)	
	EBCDICコード	ASCIIコード
絶対値の符号ビット(ビット)	F	F
正の符号ビット (4ビット)	C	C
負の符号ビット (4ビット)	D	D

B.5.2 英数字項目における変換仕様

データ定義文にてPICTURE句を"X"で定義した英数字項目では、内容が全て1バイト系文字(半角カナ含む)としてみなされ、1バイト系コード変換が行われます。

以下のケースの場合、変換後の1文字あたりのバイト長が異なるため、変換後データ長が変動して出力領域が溢れる場合があります。

オペランド	領域溢れ条件	対処方法
-ocode euc -okana euc	半角カナを含む場合 ※半角カナは2バイトで出力する。	・-okanaオペランドでjis8を指定する。 ・データ定義文特殊指示"U", "L"により出力領域長を変更する。
-ocode utf8 | cobub | cobul	半角カナを含む場合 ※半角カナは3バイトで出力する。	・データ定義文特殊指示"U", "L"により出力領域長を変更する。
-ocode ucs2b | ucs2l	1バイト系全データ ※UCS2形式は全てのコードを2バイトで出力する。	・データ定義文特殊指示"U", "L"により出力領域長を変更する。
-ocode jis | jis78	半角カナを含む場合 ※半角カナ切替制御コードを出力する。	・環境変数MDPORT_ICONV_JISに"jiskana8"を指定する。 ・データ定義文特殊指示"U", "L"により出力領域長を変更する。

※上表の領域溢れ条件は、入力コードがEBCDIC系またはSJISコードを想定しています。

B.5.3 日本語項目における変換仕様

データ定義文にてPICTURE句を"N"で定義した日本語項目では、内容が全て日本語文字としてみなされ、2バイト系コード変換が行われます。
JISとJEFおよび他社コードでは、本領域中に日本語切替制御コードが存在してはいけません。EUCコードのコードセット3の文字は、cobolEUC(2バイト/文字)で表現されます。UTF8では、3バイト系コードを変換する場合があります。

B.5.4 英数字日本語混在項目における変換仕様

データ定義文にてPICTURE句を"M"で定義した英数字日本語混在項目では、不定に英数字/日本語文字が混在しているとみなされ、各コード系の変換が行われます。
JIS/ASCII(jis,jis78)とJEFおよび他社コードでは、日本語切替制御コードが存在しないと、日本語文字の開始/終了の認識ができません。
JIS/EBCDIC(jise,jis78e)の場合、混在項目は扱えません。英数字項目とみなし処理します。
EUCコードでは、コードセット3の文字は、3バイト/文字で表現され、1バイト系JISカタカナは-ikana, -okanaオペランドの指定に関係なく、2バイト/文字で表現されます。

B.5.5 文字項目での領域長変動による文字の切捨て・埋め込み

データファイル変換で英数字項目、日本語項目、混在項目のコード変換を行うと、1項目の入力データ長と出力データ長が異なる場合があります。領域長が変動される要因となるコード体系の考え方を示します。

	3バイトコード	2バイトコード	1バイトコード	その他
英数字項目	UTF8の半角カナ	EUCカナ(コードセット2) UCS2	ASCII/EBCDIC系	JIS半角カナ切替制御コード
日本語項目	UTF8日本語コード	UTF8日本語コード以外	なし	なし
英数字日本語混在項目	UTF8日本語コード EUCコードセット3	UTF8日本語コード以外	英数字項目と同一	日本語切替制御コード

[領域長の変動例]

- EBCDIC/JIS8コードのカナ(1バイト/文字)とEUCコードのカナ(2バイト/文字)間での変換
- JIS/JEF/他社コードとEUC/シフトJISコードとの変換では、日本語切替制御コードの分、相違がある
- JEFコードの拡張漢字/拡張非漢字(2バイト/文字)とEUCコードのコードセット3(3バイト/文字)間での変換
- 利用者定義変換テーブルにて、相手側コード2バイトに対し、EUCコードのコードセット3(3バイト)を変換を指定
- データ定義文特殊指示の領域長変更(7カラム目に"U", "L"指示)
- EBCDIC/ASCIIコードからUCS2間の変換
- シフトJIS/EUC/JEFの日本語コードからUTF8間の変換

また、入力ファイル形式がRDBローダ型の場合は、区切り文字で区切られた範囲の文字列を入力長とするため、入力項目の長さは可変となります。出力ファイル形式がRDBローダ型の場合は、出力領域長が可変となります。

これらの要素が発生し、変換後データ長と出力領域長が一致しない場合、領域長の調整を行います。変換後データフィールド長が出力フィールド領域に満たない場合は空白(日本語項目では2バイト系の空白文字、それ以外では1バイト系の空白文字)が埋め込まれ、超える場合は文字単位に切り捨てられて、警告エラーが出力されます。

付図B.3に領域長変動の例を示します。

[補足事項]

- データ定義文においてX項目の情報長に奇数を指定した場合、UCS2ビッグエンディアンまたはUCS2リトルエンディアンに変換すると1バイト分満たない、または溢れます。
  その1バイト分の領域には、それぞれ半角空白の1バイト目(ビッグエンディアン:0x0020、リトルエンディアン:0x2000)を埋めます。
- -noconvオペランドを指定し、入力フィールド領域が出力フィールド領域を超える場合は、出力長で切り捨てられ、警告エラーが出力されます。文字単位での切り捨ては行われません。

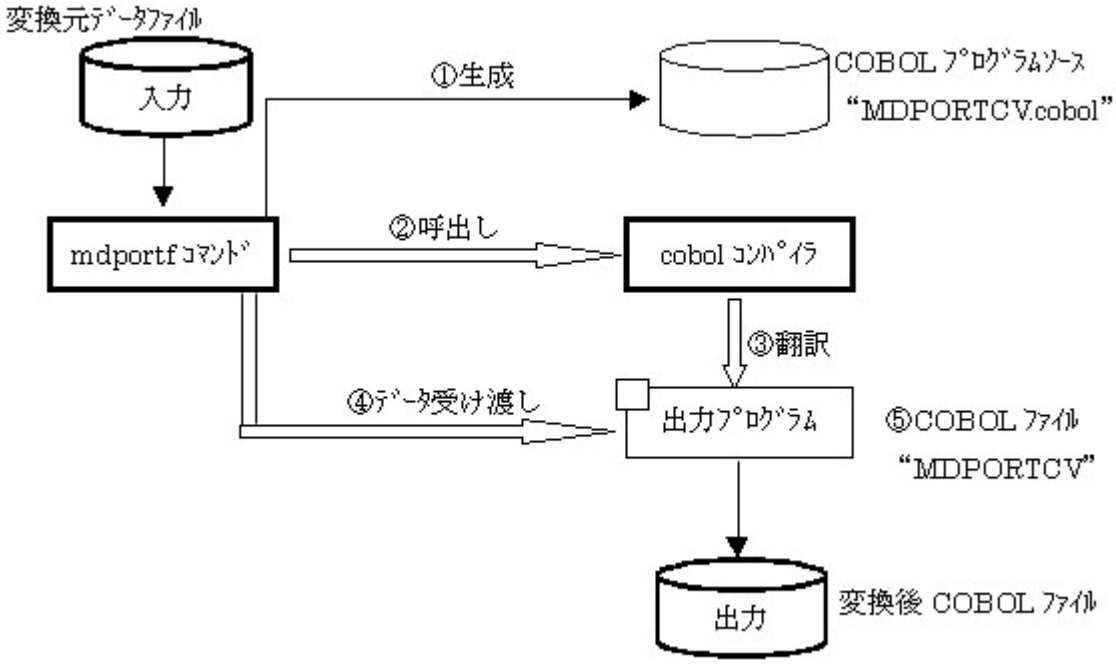
付図B.3 領域長変動例

EBCDICカナ(1バイト/文字) "アイウエオ" : 5バイト	⇒	EUCコードセット2カナ(2バイト/文字) "アイウエオ" : 10バイト
EBCDIC + JEF混在(シフトコードあり) "AB#あいうえお#DE" : 16バイト (#は日本語切替制御コードを意味する)	⇒	EUC 1バイト/日本語混在 "ABあいうえおDE" : 14バイト
JEF 拡張漢字(2バイト/文字) "■■" : 4バイト	⇒	EUC コードセット3 JEF拡張(3バイト/文字) "■■" : 6バイト

B.6 COBOLファイルの変換

MDPORTではSolaris/Linux上のCOBOLファイルの変換を行う際、付図B.4に示すように、COBOLファイルへの入出力を行うCOBOLプログラムの生成/翻訳/実行を自動的に行います。

通常、生成されるCOBOLプログラムはファイル変換終了後に削除されますが、-cvcobgenオペランドを指定することにより、変換は行わずにプログラムの生成(付図B.4の1)のみ行うことが可能です。また、-cvcobuseオペランドにより、-cvcobgen指定で保存したCOBOLプログラムを使用してファイル変換処理(付図B.4の2～5)を行うことができます。



1. MDPORTが出力用COBOLプログラムソースを生成する。
2. MDPORTがCOBOLコンパイラを呼び出す。
3. 生成されたCOBOLソースを翻訳し、オブジェクトを生成及びリンクする。
4. MDPORTが生成オブジェクトにコード変換結果データを受け渡す。
5. 生成されたプログラムが、渡されたデータをCOBOLファイルへ出力する後処理として、生成されたCOBOLソース、オブジェクトを削除する。

備考. COBOLファイルからの入力処理も、同様の内部処理が行われます。
-otypeにcblfileを指定した場合のみ有効です。
UnicodeのCOBOLファイルを扱う場合、動作環境のコードがUnicodeが指定されている必要があります。

付図**B.4 COBOL**ファイル出力のための内部処理

B.7 RDBローダ型ファイルの変換

RDBローダ型ファイルでは、全てのデータ属性をキャラクタで表現します。本節では、RDBローダ型ファイルとデータファイルとの変換仕様について説明します。

B.7.1 データファイルからRDBローダ型ファイルへの変換

データ定義文の各データ項目属性にしたがい、項目ごとに以下の変換が行われます。

[文字属性]
データ定義文の英数字・日本語・混在項目が対象となります。
コード変換後、項目単位で後続のブランクが削除され、-rdb2または-oracleオペランド指定の場合、項目の前後に引用符(")が付加されます。項目の内容が全て空白の場合はヌルデータとなり、-rdb2では引用符が付加されず、-oracleでは引用符が二つ並ぶことになります。
出力としてヌルを抑制したい場合には、データ定義文特殊指示のヌル抑制(7カラム目に"S"指示)を行うことにより、入力内容が全て空白の場合、1文字の空白を出力することができます。

例: -rdb2,-oracleオペランド指定

入力項目の内容		編集後
AB あいうえおDE F	⇒	“AB あいうえおDE F”

[数値属性]
数値属性からキャラクタへの変換後、整数部十の位以上の先行ゼロおよび小数部の後続ゼロは削除されます。小数部のあるデータは小数点が付加されます。符号つきの項目は、"+"または"-"の符号文字を、データの先頭に付加します。
例:

入力項目の内容		編集後
00123450 (符号つき小数部2けた)	⇒	+1234.5

編集された個々のデータ項目間には、コマンドの-delimitaオペランドで指定された区切り文字が挿入され、レコードの末尾には改行コード(0x0A)が付加されて出力されます。付図B.5に例を示します。
なお、RDBローダ型ファイルへ変換した結果、RDBで対応していないコードセット3の文字や、ローダ制御記号が、出力データ上に出現する場合がありますので、RDBへのロード時には注意が必要です。

入力データ	9(4)	N(4)	X(6)	X(4)	S9(4)V99
レコード1	0100	あいう	ＡＢＣ		000000
レコード2	0200	亜胃宇絵	XXXXXX	#$#$	010000

出力データ(-rdb2 オペランド指定)

レコード1 100,"あいう","ＡＢＣ",,+0.0◁
レコード2 200,"亜胃宇絵","XXXXXX","#$#$",+100.0◁

出力データ(-oracle オペランド指定)

レコード1 100,"あいう","ＡＢＣ","",+0.0◁
レコード2 200,"亜胃宇絵","XXXXXX","#$#$",+100.0◁

出力データ(-informix オペランド指定)

レコード1 100,あいう,ＡＢＣ,,+0.0◁
レコード2 200,亜胃宇絵,XXXXXX,#$#$,+100.0◁

◁:改行コード

付図B.5 RDBローダ型ファイルへの変換例

B.7.2 RDBローダ型ファイルからデータファイルへの変換

1行を1レコードとみなし、コマンドの-delimitaオペランドで指定された区切り文字が項目間の区切りとして分離します。データ定義文の各データ項目属性にしたがい、項目ごとに以下の変換が行われます。

[文字属性]

RDBローダ型からデータファイル変換の場合は、コード変換後データ長が出力領域長に満たないと項目属性にあわせた空白文字が充填され、超える場合は文字単位で切り捨てられて警告エラーが出力されます。
-rdb2または-oracle指定で、項目の先頭に引用符(")がある場合、文字列のくくり記号とみなされ、次の引用符の出現までを1つの文字項目として扱います。

[数値属性]

RDBローダ型からデータファイル変換の場合は、キャラクタから数値属性への変換を行いますが、入力項目がヌル文字列の場合はゼロとして扱われ、符号つきの項目で入力文字列の前後に"+"または"-"の符号文字が付加されていない場合は正数として扱います。また、変換処理は小数点を基準に行われますが、整数部または小数部のけた溢れが発生する場合は、溢れ分を削除し警告エラーを出力します。

データ定義文の項目数に対し、入力データの項目数が多い場合、はみ出し分は削除され警告エラーが出力されます。入力データの項目数がデータ定義文の項目数に満たない場合、不足分の項目は各データ属性にしたがい、空白文字または値ゼロが充填されます。

B.8 データファイルからテキスト型への変換

データファイルからテキスト型への変換仕様は、基本的にはデータファイルからRDBローダ型への変換仕様と同じです。以下の点で異なります。

- 文字属性はキャラクタへの変換時、後続ブランクを削除し、出力領域のバイト数まで半角空白を埋めます。
- 数値属性はキャラクタへの変換時、先行ゼロ削除の処理は行われません。
- 出力側テキストファイルの項目開始位置は、各項目で固定です。よって、フォーマットが固定になり、C言語等のアプリケーションで扱いやすくなります。
- -delimitaオペランドを省略すると、項目間の区切り文字無しとして出力されます。

各項目の出力領域長は、データ定義文のPICTURE句で指定したけた数で、以下のように決定され、原則として各フィールドの開始カラムは全レコード共通となります。

[英数字項目・混在項目]

PICTURE句"X"または"M"のけた数が、そのまま出力領域のバイト長となります。

[日本語項目]

PICTURE句"N"のけた数を2倍したものが、出力領域のバイト数となります。

[数値項目]

USAGE句の属性に関係なく、PICTURE句"9"のけた数が、出力領域のバイト長となります。小数点("V")または符号("S")の指定があれば、出力領域はさらにそれぞれ1バイト加算されます。

[特殊指示"U", "L"指定の場合]

項目属性に関係なく、データ定義文で特殊指示"U", "L"指定の項目では、"L"指定の出力領域長となります。

B.9 数値項目チェックにおける仕様

MDPORTの入力データ(数値項目)の厳密チェック処理の仕様について示します。
厳密チェック処理の指定方法については"3.5 実行環境"のMDPORT_EXACT_CHECKを参照してください。
本指定は、変換エラーの出力レベルを制御するものであり、変換仕様が変わることはありません。
ただし、初期化指定を有効にしている場合には、数値項目のエラー発生時に初期化出力を行います。
初期化指定の詳細については"3.5 実行環境"のMDPORT_INITIAL_VALUEを参照してください。

1.ゾーンビットエラー

- ゾーンビット(外部10進)のデータに異常データが存在する場合の変換
  ゾーンビットの仕様については、"B.5.1 数値項目における変換仕様"を参照してください。
  [例] EBCDICの場合

データ定義文	入力データ(16進)	エラー条件	MDPORT_EXACT_CHECK = no
S9(4)	F1 F2 X3 C4	X → F以外	ゾーン部チェックを行わずに無視

2.数値部エラー

- 内部10進の数値部データに数値以外のデータが存在する場合の変換(DATA→DATA変換時)
  [例] EBCDICの場合

データ定義文	入力データ(16進)	エラー条件	MDPORT_EXACT_CHECK = no
S9(4) COMP-3	01 2X 4C	X → 0-9以外	数値部チェックを行わずに無視

3.符号部エラー

- 外部10進/内部10進/RDB数値型の符号部にデータ定義文の型とは異なる符号データが存在する場合の変換
  外部10進/内部10進の仕様については、"B.5.1 数値項目における変換仕様"を参照してください。
  [例]

データ定義文	入力データ(16進)	エラー条件	MDPORT_EXACT_CHECK = no
9(4)	F1 F2 F3 X4	X → C, Dの場合	絶対値符号とみなして変換
9(4) COMP-3	01 23 4X	X → C, Dの場合	絶対値符号とみなして変換
S9(4)	F1 F2 F3 X4	X → Fの場合	正の符号値として変換
9(4) COMP-3※	X1234(10進数)	X → +, -の場合	絶対値とみなして変換

※入力データ形式がRDBの場合

4.桁溢れエラー

- 内部10進/バイナリ(COMP)の数値部に桁溢れが発生する場合の変換
  [例]

データ定義文	入力データ(16進)	エラー条件	MDPORT_EXACT_CHECK = no
S9(4) COMP-3	X1 23 4C	X → 0以外の数値	桁溢れエラーチェックを行わずに無視
9(4) BINARY	XX XX	X → 4桁を超えるデータ 例:FFFF	桁溢れエラーチェックを行わずに無視

付録C 出力メッセージ

ここでは、MDPORTが出力するメッセージについて説明します。
MDPORTが出力するメッセージには、標準エラー出力へ出力される一般のメッセージと、-msgオペランドで指定した出力先へ出力する警告メッセージがあります。以降では、それぞれの出力メッセージの形式について説明します。

C.1 メッセージ形式

本節では、一般のメッセージの出力形式を説明します。以下にメッセージ形式を示します。

MDPORT　メッセージレベル : メッセージ本文

[メッセージレベル]

以下に、出力されるメッセージレベルを示します。

(なし)	実行に関して、利用者に通知する確認メッセージです。
error	エラーが発生しだい処理を中断し、コマンドは異常終了します。
warning	軽度のエラーが存在しますが、処理を続行します。
reply	エラーが発生しだい処理を一時中断しますが、オペレータ応答により復旧できます。

[メッセージ本文]

メッセージの内容を示します。メッセージによっては2行にわたって出力され、2行目には、メッセージの示す箇所や詳細情報が示されます。

例 :

```
MDPORT error :      データ定義文に文法上の誤りがあります
line:5 detail:      不当な文字列が存在
```

このメッセージ例では、データ定義文の5行目に、指定可能キーワード以外の文字列を検出したことを意味しています。
エラー等が発生した場合は、メッセージ本文中の発生理由をもとに、正しい指定を行う等して再度コマンドを実行してください。以下に、メッセージ本文の2行目に出力される詳細情報の意味を示します。

line	データ定義文等の入力となるファイルの行番号を示します
obj	エラーが発生したMDPORTのプログラム名を示します
file	エラーが発生した時にアクセスを試みたファイル名を示します
sys(*1)	エラーが発生した関数名・システムコール名を示します
errno	システムコールより通知されたエラーコードを示します
detail	エラーの詳細情報を示します

(*1)
sys に Charset Manager validator 関数名が出力される場合の4桁のerrnoはバリデーション関数の詳細エラー番号です。
以下に表示される詳細エラー番号の意味を示します。

2001 : メモリ不足。
2021 : 指定されたエンコーディング形式とバリデーションポリシーファイルの組み合わせはサポートしていない。
2041 : 指定されたファイルはバリデーションポリシーファイルではない。

C.2 警告メッセージ

本節では、警告メッセージの出力形式や意味を説明します。
警告メッセージは、処理続行可能な軽度のエラーが発生した場合に出力されます。警告メッセージの出力先は、以下のように決定されます。

- mdportfコマンドで-fオペランド指定をした場合は、-fオペランドのファイル名に".msg"を付加したファイル名に出力されます。
- -msgオペランドを指定した場合は、-msgで指定したメッセージ出力先ファイル名に出力されます。
- 上述以外の場合は、標準エラー出力へ出力されます。

警告メッセージは以下の形式で出力されます。

nnnn-mmm : 警告メッセージ内容

nnnn: 入力ファイルに対する1から始まるレコード番号
mmm: 入力レコードに対する1から始まる発生カラム位置

付表C.1に、出力される警告メッセージ内容とその意味を示します。

付表C.1 警告メッセージ内容と意味

警告メッセージ内容	意 味
char:XX	変換不能文字の発生(XXは変換不能なコード)
char:XX!	文字種ポリシーチェックを使用した変換でエラー
num error	数値項目における数値例外または符号例外
RDW error	可変長レコードRDW情報に誤り
multi-c error	2バイト系コード値の異常
length over	変換結果が出力長を超えた
item over	入力データの項目数がデータ定義文より多い

警告メッセージは出力時に、1行80バイト以内におさまるように編集されます。以下に、出力の例を示します。なお、レコード変換機能のMDP_outputmsg()関数では、1メッセージごとに改行され、レコード番号は常に1を出力します。レコード変換機能では、メッセージの件数がmsgmaxで指定した数よりも多い場合には最後のメッセージの後ろに『*』が出力されます。

例

mdportfコマンド、レコード変換機能の旧形式での出力

```
1-10:char:01    1-25:char:09    1-27:char:09!    1-305:num error    1-400:multi-c error
2-404:item over    5-10:char:06    28-404:item over    100-1:char:FF
```

レコード変換機能の出力形式

```
1-10:char:01
1-25:char:09
1-27:char:09!
1-305:num error
1-400:multi-c error
2-404:item over
5-10:char:06
28-404:item over
100-1:char:FF
```

C.3 実行環境に関するメッセージの対処

本節では、実行環境の問題によって発生しうるエラーメッセージについてのみ抜粋し、その原因・対処方法について説明します。

指定されたコードの変換はできません。入力コード:(コード名1)出力コード:(コード名2)

[原因]

mdportfコマンドで指定された-icode、-ocodeの組み合わせが不当な場合に発生します。
内部で呼び出されているiconv_openに失敗した場合にも発生します。iconvに関連する環境変数の設定が不当な場合に発生します。

[対処方法]

コード系の指定を正しく行ってください。

生成COBOLプログラムのコンパイルに失敗しました

[原因]

COBOLコンパイラが導入されていない。または、COBOLコンパイラに関する環境が取得できない場合等に発生します。

[対処方法]

COBOLコンパイラが導入されていることを確認してください。
COBOLコンパイルに関する環境変数(PATH等)を設定してください。

生成COBOLプログラムのリンクに失敗しました

[原因]

COBOLプログラムをリンクするための環境に不備がある場合に発生します。

[対処方法]

COBOLプログラムのリンクに関する環境変数(LD_LIBRARY_PATH等)を設定してください。

カレントディレクトリに生成COBOLソースを作成できません

[原因]

COBOLプログラムを生成しようとしましたが、カレントディレクトリ上にCOBOLソースファイルが作成できなかったことを示します。カレントディレクトリ上の書き込み権限がない、またはデイスク容量がない場合等に発生します。

[対処方法]

カレントディレクトリの環境を見直してください。

MDPORT error: システムエラーが発生しました。
obj:Dtablemake file: libqgfv.so sys:dlopen errno:2

[原因]

Interstage Charset Managerのバリデーション機能が導入されていない。またはバリデーション機能に関する環境が取得できない場合等に発生します。

[対処方法]

Interstage Charset Managerのバリデーション機能が導入されていることを確認してください。
バリデーションライブラリの格納先に関する環境変数(LD_LIBRARY_PATH等)を設定してください。

MDPORT error:システムエラーが発生しました。
obj:Dtablemake file: sys:FJiconv_open errno:22

[原因]

メモリ不足のためInterstage Charset Managerが正常動作できない等の原因が考えられます。
この場合本メッセージの前に 「ld.so.1: mdportf: 重大なエラー: 27icv.so: open に失敗しました: ファイルもディレクトリもありません。」
等のメッセージが表示されます。

[対処方法]

メモリ、ディスク容量（スワップ領域）を増やしてください。

MDPORT error:システムエラーが発生しました。
obj:Dtablemake file: sys:malloc() errno:11

[原因]

メモリ不足の場合に発生します。

[対処方法]

メモリ、ディスク容量（スワップ領域）を増やしてください。

※errnoの数値は[Solaris]のものです。他のＯＳでは異なる場合があります。

付録D コード変換/レコード変換機能(ライブラリ)

ここでは、コード変換/レコード変換機能(ライブラリ)について説明します。

- コード変換機能
- レコード変換機能

以下の節で、関数を下記の形式で説明します。

※C言語での記述形式のため、COBOL言語の場合にはパラメータ等の名称が一部異なります。COBOLプログラムから使用する場合には提供するCOBOL登録集を参照してください。

[名 前]

関数名とその機能の概要を以下の形式で示します。
関数名 - 機能の概要

[形 式]

C言語での記述形式を示します。通常の字体は、示されているとおりに記述することを示しています。斜体の字体の引数は、置き換えて記述することを示しています。

[機能説明]

関数の機能などについて説明します。

[復帰値]

関数の復帰値について説明します。

[変換仕様]

コード変換時の仕様について説明します。

[注 意]

関数を使用するにあたっての注意事項を記述します。

D.1 コード変換機能

コード変換機能は、各種コード変換を行う関数群を提供します。
本節では、マルチスレッド対応版のコード変換機能の各関数仕様について説明します。

MDCOINIT	使用するコード変換関数を宣言し、コード変換の初期処理を行います。
MDCOFREE	コード変換の終了処理を行います。
MDCOSTAT	詳細エラー情報を取得します。
MDCOEBEU	EBCDICコードからEUC(半角・英数カナ)コードに変換します。
MDCOEBJ7	EBCDICコードからJIS7コードに変換します。
MDCOEBJ8	EBCDICコードからJIS8コードに変換します。
MDCOEUEB	EUC(半角・英数カナ)からEBCDICコードに変換します。
MDCOEUJ7	EUC(半角・英数カナ)コードからJIS7コードに変換します。
MDCOEUJ8	EUC(半角・英数カナ)コードからJIS8コードに変換します。
MDCOEUJE	EUCコードからJEFコードに変換します。
MDCOEUJI	EUCコードからJISコードに変換します。
MDCOEUSJ	EUCコードからシフトJISコードに変換します。
MDCOJ7EB	JIS7コードからEBCDICコードに変換します。
MDCOJ7EU	JIS7コードからEUC(半角・英数カナ)コードに変換します。
MDCOJ7J8	JIS7コードからJIS8コードに変換します。
MDCOJ8EB	JIS8コードからEBCDICコードに変換します。
MDCOJ8EU	JIS8コードからEUCコードに変換します。
MDCOJ8J7	JIS8コードからJIS7コードに変換します。
MDCOJEEU	JEFコードからEUCコードに変換します。
MDCOJEJI	JEFコードからJISコードに変換します。
MDCOJESJ	JEFコードからシフトJISコードに変換します。
MDCOJIEU	JISコードからEUCコードに変換します。
MDCOJIJE	JISコードからJEFコードに変換します。
MDCOJISJ	JISコードからシフトJISコードに変換します。
MDCOSJEU	シフトJISコードからEUCコードに変換します。
MDCOSJJE	シフトJISコードからJEFコードに変換します。
MDCOSJJI	シフトJISコードからJISコードに変換します。

[名 前]

MDCOINIT - 使用するコード変換関数を宣言し、コード変換の初期処理を行います。

[形 式]

#include "MDPcomm.h"
int MDCOINIT(int funcnm, char fjiconv, char** codtbl);

[機能説明]

■funcnm

funcnmには、アプリケーション内で使用するコード変換関数を以下の中から指定し、メモリの確保、コード変換テーブルの展開等の初期処理を行います。

FUNC_MDCOEBEU	関数MDCOEBEU()を表します。
FUNC_MDCOEBJ7	関数MDCOEBJ7()を表します。
FUNC_MDCOEBJ8	関数MDCOEBJ8()を表します。
FUNC_MDCOEUEB	関数MDCOEUEB()を表します。
FUNC_MDCOEUJ7	関数MDCOEUJ7()を表します。
FUNC_MDCOEUJ8	関数MDCOEUJ8()を表します。
FUNC_MDCOEUJE	関数MDCOEUJE()を表します。
FUNC_MDCOEUJI	関数MDCOEUJI()を表します。
FUNC_MDCOEUSJ	関数MDCOEUSJ()を表します。
FUNC_MDCOJ7EB	関数MDCOJ7EB()を表します。
FUNC_MDCOJ7EU	関数MDCOJ7EU()を表します。
FUNC_MDCOJ7J8	関数MDCOJ7J8()を表します。
FUNC_MDCOJ8EB	関数MDCOJ8EB()を表します。
FUNC_MDCOJ8EU	関数MDCOJ8EU()を表します。
FUNC_MDCOJ8J7	関数MDCOJ8J7()を表します。
FUNC_MDCOJEEU	関数MDCOJEEU()を表します。
FUNC_MDCOJEJI	関数MDCOJEJI()を表します。
FUNC_MDCOJESJ	関数MDCOJESJ()を表します。
FUNC_MDCOJIEU	関数MDCOJIEU()を表します。
FUNC_MDCOJIJE	関数MDCOJIJE()を表します。
FUNC_MDCOJISJ	関数MDCOJISJ()を表します。
FUNC_MDCOSJEU	間数MDCOSJEU()を表します。

FUNC_MDCOSJJE	関数MDCOSJJE()を表します。
FUNC_MDCOSJJI	関数MDCOSJJI()を表します。

本関数1回の呼び出しで、複数の関数名を同時に指定することはできません。1つのアプリケーションで異なる複数のコード変換関数を使用する場合は、使用するコード変換関数の分だけ本関数を事前に呼び出す必要があります。

■fjiconv

コード変換に使用するiconv関数の指定を行います。

FJICV_OFF	iconv関数が使用できる場合、内部的にiconvを呼出します。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
FJICV_ON	iconv関数のロード処理を行い、iconv関数が使用できる場合、内部的にiconvを呼出します。iconv関数が使用できない場合には、エラーを返します。Javaから呼び出す場合には、FJICV_ONにする必要があります。

■codtbl

codtblには各変換関数で指定するMDCOPARA構造体のcodtblパラメタへ指定する値が返却されます。アプリケーション内で複数のコード変換関数を使用する場合、codtblに設定される内容は各関数毎に用意する(関数ごとに変数を変えておく)必要があります。

[復帰値]

復帰値	意味
0	正常終了(条件つき正常終了含む)

0以外が返ってきた場合のエラーは以下のとおりです。

復帰値	意味
MDCO_DIR	MDPORT_DIR is wrong variable.
MDCO_PAS	PASSWORD error.
MDCO_FNM	function number is mismatch.
MDCO_MEM	memory allocate error.
MDCO_OCD	ocode parameter error.
MDCO_ICD	icode parameter error.
MDCO_TBL	convert table make error.
-1	上記以外のエラー

[名 前]

MDCOFREE - コード変換の終了処理を行います。

[形 式]

```
#include "MDPcomm.h"
int MDCOFREE( char**codtbl);
```

[機能説明]

本関数は、MDCOINIT()関数で展開された文字コードテーブル変換テーブル等の動的に確保した領域の解放等のコード変換の終了処理を行います。引数としてMDCOINITで定義したcodtblを指定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[注 意]

本関数を呼び出すと、MDCOINIT()により確保された資源は全て解放されます。本関数の呼び出し後は、再度MDCOINIT()を呼び出し初期化処理を行わないと、コード変換処理を行えません。

[名 前]

MDCOSTAT - 詳細エラー情報を取得します。

[形 式]

```
#include "MDPcomm.h"
void MDCOSTAT(struct mdco_stat *stat, struct MDCOPARA* codpara);
```

[機能説明]

本関数は、各コード変換関数で発生したエラーの詳細情報を取得します。mdco_stat構造体宣言とその内容は以下の通りです。

■stat

```
struct mdco_stat {
        short       st_etyp;    /* エラー種別          */
        short       st_eno;     /* エラー番号          */
        short       st_wnum;    /* ワーニング発生数    */
        short       *st_wpos;   /* ワーニング発生位置  */
        short       *st_wtyp;   /* ワーニング発生種別  */
} ;
```

st_etypには、各コードへ変換関数で発生したエラーの種別が返されます。エラーの種別は以下のとおりです。

0	エラーはありません。
MDCO_WAR	文字の変換エラーが発生しました。
MDCO_ERR	上記以外のエラー(システムエラー等)が発生しました。

st_enoは、st_etypの値が、MDCO_ERRの場合に以下の値が設定されます。MDCO_ERR以外の場合は無効となります。

MDCO_ENV	実行環境でエラーが発生しました。
MDCO_PAR	パラメタに誤りがあります。
その他	上記以外の場合、システムのerrnoが設定されます。

st_wnumは、st_etypの値がMDCO_WARの場合に、ワーニングの発生件数を返します。MDCO_WAR以外の場合は無効となります。なお、設定される最大件数は9999件までです。
st_wposは、st_etypの値がMDCO_WARの場合に、発生した変換エラーの先頭からの位置(バイト単位)がst_wnumで示される数の配列として格納され、そのポインタが返されます。
st_wtypは、st_etypの値がMDCO_WARの場合に、発生した変換エラー種別がst_wnumで示される数の配列として格納され、そのポインタが返されます。
エラー種別は以下のとおりです。

MDCO_WCH1	未定義文字(1バイト系)です。
MDCO_WCH2	未定義文字(2バイト系)です。
MDCO_WNUM	数値エラーです。
MDCO_WOVR	領域長オーバー(変換後文字列長>出力文字列長)です。

■codepara

各コード変換関数で宣言したMDCOPARA構造体を指定します。

[復帰値]

復帰値はありません。

[注 意]

本関数によって返されるエラー詳細情報は、直前に呼び出されたコード変換関数についての情報のみです。対象とするコード変換関数から復帰後、本関数を呼び出す前に、ほかのシステムコールや関数を呼び出すとエラー詳細情報は無効になります。

[名 前]

MDCOEBEU - EBCDICコードからEUC(半角・英数カナ)コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                          */
        int             rlen;           /* 変換結果長                      */
        char            filler1[32];    /* 空き領域                        */
        short           int inlen;      /* 入力データ長                    */
        short           int outlen;     /* 出力データ長                    */
        char            alt1[1];        /* 1バイト系代替文字               */
        char            alt2[2];        /* 2バイト系代替文字               */
        char            filler2[1];     /* 空き領域                        */
        int             opt;            /* 変換オプション                  */
        int             MdcoStErrTyp    /*  エラー種別                     */
        int             MdcoStErrNo     /*  エラー番号                     */
        short           MdcoStWarPos[9999]  /*  ワーニング発生位置          */
        short           MdcoStWarTyp[9999]  /*  ワーニング発生種別          */
        char*           codtbl          /* コード変換テーブル管理用構造体   */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはEUCコード(ASCIIコード+JISカタカナ(CS2))として変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEBJ7 - EBCDICコードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBJ7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
	JIS7コードへ変換し、上位1ビットを奇数パリティビット

MDCOOPT_PO	とします。
その他	MDCOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはJIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOEBJ8 - EBCDICコードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBJ8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

[

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
};
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはJIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUEB - EUC(半角・英数カナ)からEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUEB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナ)コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999]  /*  ワーニング発生位置       */
        short           MdcoStWarTyp[9999]  /*  ワーニング発生種別       */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))、出力データの文字コード

は、EBCDIC(カナ)コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJ7 - EUC(半角・英数カナ)コードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJ7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナ)コードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
	JIS7コードへ変換し、上位1ビットを奇数パリティビットとしま

MDCOOPT_PO	す。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))コード、出力データの文字コードは、JIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOEUJ8 - EUC(半角・英数カナ)コードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJ8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナ)コードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))コード、出力データの文字コード

は、JIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJE - EUCコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を

	行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJI - EUCコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を

	行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUSJ - EUCコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUSJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を

	行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ7EB - JIS7コードからEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7EB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードである

	ものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS7コード、出力データの文字コードは、EBCDIC(カナ)コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ7EU - JIS7コードからEUC(半角・英数カナ)コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7EU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからEUC(半角・英数カナ)コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードである

	ものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS7コード、出力データの文字コードは、EUC(ASCIIコード+JISカタカナ(CS2))コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ7J8 - JIS7コードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7J8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードである

	ものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS7コード、出力データの文字コードは、JIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ8EB - JIS8コードからEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8EB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS8コード、出力データの文字コードは、EBCDIC(カナ)コードとして変換

を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ8EU - JIS8コードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8EU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS8コード、出力データの文字コードは、EUC(ASCIIコード+JISカタカ

ナ(CS2))コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ8J7 - JIS8コードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8J7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長7nbsp;または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
	JIS7コードへ変換し、上位1ビットを奇数パリティビットと

MDCOOPT_PO	します。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS8コード、出力データの文字コードは、JIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOJEEU - JEFコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJEEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                          */
        int             rlen;           /* 変換結果長                      */
        char            filler1[32];    /* 空き領域                        */
        short           int inlen;      /* 入力データ長                    */
        short           int outlen;     /* 出力データ長                    */
        char            alt1[1];        /* 1バイト系代替文字               */
        char            alt2[2];        /* 2バイト系代替文字               */
        char            filler2[1];     /* 空き領域                        */
        int             opt;            /* 変換オプション                  */
        int             MdcoStErrTyp    /*  エラー種別                     */
        int             MdcoStErrNo     /*  エラー番号                     */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置           */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別           */
        char*           codtbl          /* コード変換テーブル管理用構造体   */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、EUCコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJEJI - JEFコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJEJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen:           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999]  /*  ワーニング発生位置       */
        short           MdcoStWarTyp[9999]  /*  ワーニング発生種別       */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJESJ - JEFコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJESJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJIEU - JISコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJIEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJISコード、出力データの文字コードは、EUCコードとして変換を行います。
それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJIJE - JISコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJIJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                          */
        int             rlen;           /* 変換結果長                      */
        char            filler1[32];    /* 空き領域                        */
        short           int inlen;      /* 入力データ長                    */
        short           int outlen;     /* 出力データ長                    */
        char            alt1[1];        /* 1バイト系代替文字               */
        char            alt2[2];        /* 2バイト系代替文字               */
        char            filler2[1];     /* 空き領域                        */
        int             opt;            /* 変換オプション                  */
        int             MdcoStErrTyp;   /*  エラー種別                     */
        int             MdcoStErrNo;    /*  エラー番号                     */
        short           MdcoStWarPos[9999]; /*  ワーニング発生位置         */
        short           MdcoStWarTyp[9999]; /*  ワーニング発生種別         */
        char*           codtbl;         /* コード変換テーブル管理用構造体  */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJISコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJISJ - JISコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJISJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999]  /*  ワーニング発生位置       */
        short           MdcoStWarTyp[9999]  /*  ワーニング発生種別       */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJISコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJEU - シフトJISコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、EUCコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJJE - シフトJISコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJJI - シフトJISコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
        int             rtncod;         /* 復帰値                       */
        int             rlen;           /* 変換結果長                   */
        char            filler1[32];    /* 空き領域                     */
        short           int inlen;      /* 入力データ長                 */
        short           int outlen;     /* 出力データ長                 */
        char            alt1[1];        /* 1バイト系代替文字            */
        char            alt2[2];        /* 2バイト系代替文字            */
        char            filler2[1];     /* 空き領域                     */
        int             opt;            /* 変換オプション               */
        int             MdcoStErrTyp    /*  エラー種別                  */
        int             MdcoStErrNo     /*  エラー番号                  */
        short           MdcoStWarPos[9999] /*  ワーニング発生位置        */
        short           MdcoStWarTyp[9999] /*  ワーニング発生種別        */
        char*           codtbl          /* コード変換テーブル管理用構造体 */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。
MdcoStErrTyp、MdcoStErrNo、MdcoStWarPos、MdcoStWarTypは、本関数では使用しません。
codtblには、MDCOINIT関数で引数で渡された変数にCODTBL構造体の領域を設定します。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
	入力および出力データはCOBOLの英数字日本語混在項目と

MDCOOPT_M	して変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

D.1.1 コード変換機能(旧形式)[Solaris]

コード変換機能は、各種コード変換を行う関数群を提供します。
本節では、コード変換機能の各関数仕様について説明します。

MDCOINIT	使用するコード変換関数を宣言し、コード変換の初期処理を行います。
MDCOFREE	コード変換の終了処理を行います。
MDCOSTAT	詳細エラー情報を取得します。
MDCOEBEU	EBCDICコードからEUC(半角・英数カナ)コードに変換します。
MDCOEBJ7	EBCDICコードからJIS7コードに変換します。
MDCOEBJ8	EBCDICコードからJIS8コードに変換します。
MDCOEUEB	EUC(半角・英数カナ)からEBCDICコードに変換します。
MDCOEUJ7	EUC(半角・英数カナ)コードからJIS7コードに変換します。
MDCOEUJ8	EUC(半角・英数カナ)コードからJIS8コードに変換します。
MDCOEUJE	EUCコードからJEFコードに変換します。
MDCOEUJI	EUCコードからJISコードに変換します。
MDCOEUSJ	EUCコードからシフトJISコードに変換します。
MDCOJ7EB	JIS7コードからEBCDICコードに変換します。
MDCOJ7EU	JIS7コードからEUC(半角・英数カナ)コードに変換します。
MDCOJ7J8	JIS7コードからJIS8コードに変換します。
MDCOJ8EB	JIS8コードからEBCDICコードに変換します。
MDCOJ8EU	JIS8コードからEUCコードに変換します。
MDCOJ8J7	JIS8コードからJIS7コードに変換します。
MDCOJEEU	JEFコードからEUCコードに変換します。
MDCOJEJI	JEFコードからJISコードに変換します。
MDCOJESJ	JEFコードからシフトJISコードに変換します。
MDCOJIEU	JISコードからEUCコードに変換します。
MDCOJIJE	JISコードからJEFコードに変換します。
MDCOJISJ	JISコードからシフトJISコードに変換します。
MDCOSJEU	シフトJISコードからEUCコードに変換します。
MDCOSJJE	シフトJISコードからJEFコードに変換します。
MDCOSJJI	シフトJISコードからJISコードに変換します。

[名 前]

MDCOINIT - 使用するコード変換関数を宣言し、コード変換の初期処理を行います。

[形 式]

```
#include "MDPcomm.h"
int MDCOINIT(int funcnm);
```

[機能説明]

funcnmには、アプリケーション内で使用するコード変換関数を以下の中から指定し、メモリの確保、コード変換テーブルの展開等の初期処理を行います。

FUNC_MDCOEBEU	関数MDCOEBEU()を表します。
FUNC_MDCOEBJ7	関数MDCOEBJ7()を表します。
FUNC_MDCOEBJ8	関数MDCOEBJ8()を表します。
FUNC_MDCOEUEB	関数MDCOEUEB()を表します。
FUNC_MDCOEUJ7	関数MDCOEUJ7()を表します。
FUNC_MDCOEUJ8	関数MDCOEUJ8()を表します。
FUNC_MDCOEUJE	関数MDCOEUJE()を表します。
FUNC_MDCOEUJI	関数MDCOEUJI()を表します。
FUNC_MDCOEUSJ	関数MDCOEUSJ()を表します。
FUNC_MDCOJ7EB	関数MDCOJ7EB()を表します。
FUNC_MDCOJ7EU	関数MDCOJ7EU()を表します。
FUNC_MDCOJ7J8	関数MDCOJ7J8()を表します。
FUNC_MDCOJ8EB	関数MDCOJ8EB()を表します。
FUNC_MDCOJ8EU	関数MDCOJ8EU()を表します。
FUNC_MDCOJ8J7	関数MDCOJ8J7()を表します。
FUNC_MDCOJEEU	関数MDCOJEEU()を表します。
FUNC_MDCOJEJI	関数MDCOJEJI()を表します。
FUNC_MDCOJESJ	関数MDCOJESJ()を表します。
FUNC_MDCOJIEU	関数MDCOJIEU()を表します。
FUNC_MDCOJIJE	関数MDCOJIJE()を表します。
FUNC_MDCOJISJ	関数MDCOJISJ()を表します。
FUNC_MDCOSJEU	間数MDCOSJEU()を表

	します。
FUNC_MDCOSJJE	関数MDCOSJJE()を表します。
FUNC_MDCOSJJI	関数MDCOSJJI()を表します。

本関数1回の呼び出しで、複数の関数名を同時に指定することはできません。1つのアプリケーションで異なる複数のコード変換関数を使用する場合は、使用するコード変換関数の分だけ本関数を事前に呼び出す必要があります。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[名 前]

MDCOFREE - コード変換の終了処理を行います。

[形 式]

```
#include "MDPcomm.h"
int MDCOFREE();
```

[機能説明]

本関数は、MDCOINIT()関数で展開された文字コードテーブル変換テーブル等の動的に確保した領域の解放等のコード変換の終了処理を行います。引数はありません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[注 意]

本関数を呼び出すと、MDCOINIT()により確保された資源は全て解放されます。本関数の呼び出し後は、再度MDCOINIT()を呼び出し初期化処理を行わないと、コード変換処理を行えません。

[名 前]

MDCOSTAT - 詳細エラー情報を取得します。

[形 式]

```
#include "MDPcomm.h"
void MDCOSTAT(struct mdco_stat *stat);
```

[機能説明]

本関数は、各コード変換関数で発生したエラーの詳細情報を取得します。mdco_stat構造体宣言とその内容は以下の通りです。

```
struct mdco_stat {
    short       st_etyp;    /* エラー種別              */
    short       st_eno;     /* エラー番号              */
    short       st_wnum;    /* ワーニング発生数        */
    short       *st_wpos;   /* ワーニング発生位置      */
    short       *st_wtyp;   /* ワーニング発生種別      */
} ;
```

st_etypには、各コードへ変換関数で発生したエラーの種別が返されます。エラーの種別は以下のとおりです。

0	エラーはありません。
MDCO_WAR	文字の変換エラーが発生しました。
MDCO_ERR	上記以外のエラー(システムエラー等)が発生しました。

st_enoは、st_etypの値が、MDCO_ERRの場合に以下の値が設定されます。MDCO_ERR以外の場合は無効となります。

MDCO_ENV	実行環境でエラーが発生しました。
MDCO_PAR	パラメタに誤りがあります。
その他	上記以外の場合、システムのerrnoが設定されます。

st_wnumは、st_etypの値がMDCO_WARの場合に、ワーニングの発生件数を返します。MDCO_WAR以外の場合は無効となります。なお、設定される最大件数は9999件までです。
st_wposは、st_etypの値がMDCO_WARの場合に、発生した変換エラーの先頭からの位置(バイト単位)がst_wnumで示される数の配列として格納され、そのポインタが返されます。
st_wtypは、st_etypの値がMDCO_WARの場合に、発生した変換エラー種別がst_wnumで示される数の配列として格納され、そのポインタが返されます。
エラー種別は以下のとおりです。

MDCO_WCH1	未定義文字(1バイト系)です。
MDCO_WCH2	未定義文字(2バイト系)です。
MDCO_WNUM	数値エラーです。
MDCO_WOVR	領域長オーバー(変換後文字列長>出力文字列長)です。

[復帰値]

復帰値はありません。

[注 意]

本関数によって返されるエラー詳細情報は、直前に呼び出されたコード変換関数についての情報のみです。対象とするコード変換関数から復帰後、本関数を呼び出す前に、ほかのシステムコールや関数を呼び出すとエラー詳細情報は無効になります。

[名 前]

MDCOEBEU - EBCDICコードからEUC(半角・英数カナ)コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはEUCコード(ASCIIコード+JISカタカナ(CS2))として変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEBJ7 - EBCDICコードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBJ7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常:システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
MDCOOPT_PO	JIS7コードへ変換し、上位1ビットを奇数パリティビットとします。
その他	MDCOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはJIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOEBJ8 - EBCDICコードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEBJ8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEBCDICコードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常:システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEBCDIC(カナ)コード、出力データの文字コードはJIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUEB - EUC(半角・英数カナ)からEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUEB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナ)コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値                */
    int         rlen;       /* 変換結果長            */
    char        filler1[32]; /* 空き領域             */
    short       int inlen;  /* 入力データ長          */
    short       int outlen; /* 出力データ長          */
    char        alt1[1];    /* 1バイト系代替文字     */
    char        alt2[2];    /* 2バイト系代替文字     */
    char        filler2[1]; /* 空き領域              */
    int         opt;        /* 変換オプション        */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))、出力データの文字コードは、EBCDIC(カナ)コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJ7 - EUC(半角・英数カナ)コードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJ7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナ)コードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
MDCOOPT_PO	JIS7コードへ変換し、上位1ビットを奇数パリティビットとします。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))コード、出力データの文字コードは、JIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOEUJ8 - EUC(半角・英数カナ)コードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJ8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUC(半角・英数カナコードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUC(ASCIIコード+JISカタカナ(CS2))コード、出力データの文字コードは、JIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJE - EUCコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUJI - EUCコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOEUSJ - EUCコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOEUSJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをEUCコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはEUCコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ7EB - JIS7コードからEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7EB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS7コード、出力データの文字コードは、EBCDIC(カナ)コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ7EU - JIS7コードからEUC(半角・英数カナ)コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7EU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからEUC(半角・英数カナ)コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値                */
    int         rlen;       /* 変換結果長            */
    char        filler1[32]; /* 空き領域             */
    short       int inlen;  /* 入力データ長          */
    short       int outlen; /* 出力データ長          */
    char        alt1[1];    /* 1バイト系代替文字     */
    char        alt2[2];    /* 2バイト系代替文字     */
    char        filler2[1]; /* 空き領域              */
    int         opt;        /* 変換オプション        */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS7コード、出力データの文字コードは、EUC(ASCIIコード+JISカタカナ(CS2))コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ7J8 - JIS7コードからJIS8コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ7J8(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS7コードからJIS8コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、入力データのパリティビットのチェックを行うことができます。

MDCOOPT_PN	入力データは、通常のJIS7コードで、パリティビットは付加されていないものとして変換を行います。入力データパリティビットが付加されている場合でも、上位1バイトは無視して変換を行います。
MDCOOPT_PE	入力データは、偶数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
MDCOOPT_PO	入力データは、奇数パリティビット(上位1バイト)付のJIS7コードであるものとして変換を行います。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS7コード、出力データの文字コードは、JIS8コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、入力データであるJIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用してJISローマ字、JISカタカナの切り替えを行い、下位7ビットのみで表されているものとします。また、パリティビットのチェックでエラーとなった場合は、変換対象外の文字として扱います。

[名 前]

MDCOJ8EB - JIS8コードからEBCDICコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8EB(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからEBCDICコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS8コード、出力データの文字コードは、EBCDIC(カナ)コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ8EU - JIS8コードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8EU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、本関数では使用しません。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJIS8コード、出力データの文字コードは、EUC(ASCIIコード+JISカタカナ(CS2))コードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJ8J7 - JIS8コードからJIS7コードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJ8J7(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJIS8コードからJIS7コードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、本関数では使用しません。
optは、以下の値を設定することにより、出力データにパリティビットを付加することができます。

MDCOOPT_PN	JIS7コードへ通常の変換を行います。パリティビットは付加しません。
MDCOOPT_PE	JIS7コードへ変換し、上位1ビットを偶数パリティビットとします。
MDCOOPT_PO	JIS7コードへ変換し、上位1ビットを奇数パリティビットとします。
その他	MDCOOPT_PNと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJIS8コード、出力データの文字コードは、JIS7コードとして変換を行います。それ以外の文字コードは変換エラーとします。本関数では、JIS7コードはASCII SO(シフトアウト)とSI(シフトイン)を使用し、下位7ビットのみでJISローマ字、JISカタカナを表します。なお、パリティビットを付加する場合は、JIS7コードで使用されない上位1ビットを使用します。

[名 前]

MDCOJEEU - JEFコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJEEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、EUCコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJEJI - JEFコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJEJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen:       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJESJ - JEFコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJESJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJEFコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJEFコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJIEU - JISコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJIEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードJISコード、出力データの文字コードは、EUCコードとして変換を行います。
それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJIJE - JISコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJIJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJISコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOJISJ - JISコードからシフトJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOJISJ(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをJISコードからシフトJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはJISコード、出力データの文字コードは、シフトJISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJEU - シフトJISコードからEUCコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJEU(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからEUCコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、EUCコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJJE - シフトJISコードからJEFコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJJE(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからJEFコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、JEFコードとして変換を行います。それ以外の文字コードは変換エラーとします。

[名 前]

MDCOSJJI - シフトJISコードからJISコードに変換します。

[形 式]

```
#include "MDPcomm.h"
int MDCOSJJI(struct MDCOPARA *codpara, char *indata, char *outdata);
```

[機能説明]

本関数は、indataで指定された入力データをシフトJISコードからJISコードに変換し、outdataに設定します。MDCOPARA構造体宣言とその設定内容は以下の通りです。

```
struct MDCOPARA {
    int         rtncod;     /* 復帰値              */
    int         rlen;       /* 変換結果長          */
    char        filler1[32]; /* 空き領域           */
    short       int inlen;  /* 入力データ長        */
    short       int outlen; /* 出力データ長        */
    char        alt1[1];    /* 1バイト系代替文字   */
    char        alt2[2];    /* 2バイト系代替文字   */
    char        filler2[1]; /* 空き領域            */
    int         opt;        /* 変換オプション      */
} ;
```

rtncodには本関数の復帰詳細コードが返されます。設定される内容は以下のとおりです。

0x00	正常(コード変換成功)
0x08	条件付正常(出力データ長 > 変換後データ長)
0x09	条件付正常(入力データに変換対象外の値があります)
0x10	異常(入力データ長 < 1)
0x11	異常(出力データ長 < 変換後データ長)
0x12	異常(入力データ長 > 最大入力データ長 または 出力データ長 > 最大出力データ長)
0x18	異常(その他の異常 : システムエラー等)

rlenには、変換後のデータ長が返されます。
inlenは、入力データ長を設定します。
outlenは、出力データ長を設定します。
alt1は、入力データに変換不能な文字が含まれる場合に、代替する1バイト系の文字を変換先のコード体系で指定します。
alt2は、入力データに変換不能な文字が含まれる場合に、代替する2バイト系の文字を変換先のコード体系で指定します。
optは、以下の値を設定することにより、変換仕様の切り替えを行います。

MDCOOPT_N	入力および出力データはCOBOLの日本語項目に指定されるデータ(2バイト/1文字)として変換を行い、シフトコードは含まないものとします。
MDCOOPT_M	入力および出力データはCOBOLの英数字日本語混在項目として変換を行い、必要であればシフトコードが含まれるものとします。
その他	MDCOOPT_Mと同意です。

[復帰値]

正常終了時(条件付き正常終了含む)は0を、異常終了時は-1を返します。

[変換仕様]

入力データの文字コードはシフトJISコード、出力データの文字コードは、JISコードとして変換を行います。それ以外の文字コードは変換エラーとします。

D.2 レコード変換機能

レコード変換機能は、レコード単位にコード変換、レコード変換を行う関数群を提供します。
本節では、マルチスレッド対応版のレコード変換機能の各関数仕様およびレイアウトファイル生成コマンドについて説明します。

mdportg	データ定義文解析/レイアウトファイル生成コマンド
MDP_init	レコード変換関数の初期化を行います。
MDP_initmsg	メッセージ格納領域の確保を行います。
MDP_conv	文字コードの変換/レコード形式の変換を行います。
MDP_getmsg	メッセージの取得を行います。
MDP_nextmsg	メッセージ格納領域内の次メッセージへの移動を行います。
MDP_prevmsg	メッセージ格納領域内の前メッセージへの移動を行います。
MDP_outputmsg	メッセージ格納領域内の全てのメッセージをファイルへ出力します。
MDP_freemsg	メッセージ格納領域の解放を行います。
MDP_fin	レコード変換関数の終了処理を行います。
MDP_stat	詳細情報を取得します。

[名 前]

mdportg - データ定義文解析/レイアウトファイル生成コマンド

[形 式]

mdportg -if (入力ファイル名)[-of (出力ファイル名)]
-itype { data| rdb } -otype { data | rdb | text } [-oracle | -informix | -rdb2]

[機能説明]

mdportgは、指定されたデータ定義文と変換対象のファイル/レコードの形式を解析し、レコード変換機能(関数群)で使用可能なレイアウトファイルを生成します。
以下にmdportgのオプションと意味を説明します。

-if (入力ファイル名)	解析の対象となるデータ定義文ファイル名を指定します。データ定義文の入力形式については"3.2 データ定義文の入力形式"を参照して下さい。
-of (出力ファイル名)	レイアウト情報ファイル名を指定します。レイアウト情報ファイルは-ifで指定されたデータ定義文ファイルを解析して作成されます。このオプションを省略した場合は、"infile.mdg"に出力されます。既にファイルが存在している場合は上書きされます。
-itype { data | rdb }	入力対象となるレコードまたはファイルの形式を指定します。形式については"3.1.1 mdportfコマンド"を参照して下さい。
-otype { data | rdb | text }	出力対象となるレコードまたはファイルの形式を指定します。形式については"3.1.1 mdportfコマンド"を参照して下さい。
-oracle | -informix | -rdb2	RDBローダ型ファイル/レコードを扱う場合に、対象のRDB向けローダ形式を指定します。-itypeまたは-otypeにrdbを指定したときのみ有効です。RDBローダ形式については、"3.1.1 mdportfコマンド"を参照して下さい。

[注 意]

Solaris版V6/Linux for Itanium版V5L4以降のバージョンのmdportgで作成したレイアウトファイルは、V5L3以前のバージョンのレコード変換機能で使用できません。

[名 前]

MDP_init - レコード変換関数の初期化を行います。

[形 式]

```
#include "MDPcomm.h"
char *MDP_init(struct MDPcvinf *inf);
```

[機能説明]

本関数は、infが指すコード変換指示に従って、コード変換テーブルの展開/レイアウト情報の展開を行い、コード変換情報として、その先頭アドレスを復帰値として返します。MDPcvinf構造体宣言とその設定内容は以下の通りです。

```
struct MDPcvinf {       /* MDPORT 変換指示構造体 */
    char  icode[8];     /* 入力文字コード */
    char  ocode[8];     /* 出力文字コード */
    int   itype;        /* 入力ファイル形式 */
    int   otype;        /* 出力ファイル形式 */
    int   irecdlm;      /* 入力レコード改行コード */
    int   orecdlm;      /* 出力レコード改行コード */
    int   isomode;      /* ISO/エミュレータモード */
    int   ikana;        /* 入力半角カナモード */
    int   okana;        /* 出力半角カナモード */
    int   irecfm;       /* 入力レコード属性 */
    int   orecfm;       /* 出力レコード属性 */
    int   shift;        /* シフトコード除去モード */
    int   cs3mode;      /* コードセット3モード */
    int   amend1;       /* 1バイト系代替文字 */
    int   amend2;       /* 2バイト系代替文字 */
    char  delimita;     /* CSV区切り文字 */
    char  noblc;        /* 後続空白削除抑止フラグ */
    char  fjiconv;      /* コード変換ライブラリフラグ */
    char  noconv;       /* 同一コード無変換指示フラグ */
    int   msgmax;       /* メッセージ出力MAX 件数 */
    int   msgcont;      /* MSG 上限到達処理続行 */
    char  *policyfname; /*バリデーションポリシーファイル名*/
    char  *usrfname;    /* 利用者定義ファイル名 */
    char  *layfname;    /* レイアウト情報ファイル名 */
    short errcode;      /* MDP_initエラーコード */
    short derrcode;     /* MDP_init詳細エラーコード */
    char  validation;   /* 文字種ポリシーチェック指示フラグ */
    char  filler3[3];   /* 未使用 */
} ;
```

■icode, ocode
入力文字コード、出力文字コードのキーワードを指定します。以下の値が指定可能です。

euc	ASCIIおよびEUCコード
sjis	ASCIIおよびシフトJISコード
jef	EBCDICおよびJEFコード
他社コード	EBCDICおよび他社日本語コード
utf8	UTF8形式

ucs2b	UCS2(ビッグエンディアン)形式
ucs2l	UCS2(リトルエンディアン)形式
cobub	Unicode形式のCOBOLファイル仕様に準拠。 X, M項目 : UTF8形式 N項目 : UCS2(ビッグエンディアン形式)
cobul	Unicode形式のCOBOLファイル仕様に準拠。 X, M項目 : UTF8形式 N項目 : UCS2(リトルエンディアン形式)

ページ先頭へ

■itype, otype
入力レコード形式、出力レコード形式を以下の値の中から指定します。

TYPE_T	テキスト
TYPE_D	データ
TYPE_S	COBOL固定長ソース
TYPE_O	RDBローダ型(oracle)
TYPE_R	RDBローダ 型(RDB2,Symfoware)
TYPE_I	RDBローダ型(INFORMIX)

ページ先頭へ

■irecdlm, orecdlm
入力レコード区切文字、出力レコード区切文字を以下の値の中から指定します。

DLM_CRLF	復帰改行(CRLF)
DLM_LF	改行(LF)
DLM_NO	コードによる区切りなし

ページ先頭へ

■isomode
isomodeはEBCDIC系のコード変換において、以下の値の中から指定します。なお、EBCDIC系以外のコード変換では無視されます。(本オプションの指定内容については、"3.1.1 mdportfコマンド"の-emu | -isoオペランドを参照して下さい。)

ISO	ISOモード変換
EMU	エミュレータモード変換

ページ先頭へ

■ikana
入力の半角カナモードを以下の値の中から指定します。
(入力がEUCコードの時のみ指定が有効になります。)

KANA_E	EUCコード体系のコードセット2(2バイト/1文字)
KANA_J	JIS8コード(1バイト/1文字)

ページ先頭へ

■okana
出力の半角カナモードを以下の値の中から指定します。
(出力がEUCコードの時のみ指定が有効になります。)

KANA_E	EUCコード体系のコードセット2で出力(2バイト/1文字)
KANA_J	JIS8コードで出力(1バイ

	ト/1文字)
KANA_G	EUCコード体系のコードセット1(全角カナ)で出力

ページ先頭へ

■irecfm
入力レコードの属性を指定します。itypeがデータ、またはEBCDIC系のテキストの場合に有効です。省略した場合は固定長レコードとして扱います。(入力レコード属性については、"3.1.1 mdportfコマンド"の-irecfmオペランドを参照して下さい。)

RECFM_F	固定長レコード

ページ先頭へ

■orecfm
出力レコードの属性を指定します。otypeがデータ、またはEBCDIC系のテキストの場合に有効です。省略した場合は固定長レコードとして扱います。(出力レコード属性については、"3.1.1 mdportfコマンド"の-orecfmオペランドを参照して下さい。)

RECFM_F	固定長レコード

ページ先頭へ

■shift
EBCDIC系コードからのデータ変換において、入力レコード上の制御コードの扱いを以下の値の中から指定します。テキストモード変換および出力がRDBローダ型の場合、常に制御コードは取り除かれて後続文字を詰めます。(SHIFT_Lと同一)

SHIFT_L	シフトコード除去後 左詰め
SHIFT_B	シフトコードを半角空白へ置換

ページ先頭へ

■cs3mode
変換後または変換元のEUCコードとして、コードセット3(3バイト/1文字表現)の使用する/しないを指定します。

NOCS3_BLK	コードセット3を使用しません。
NOCS3_EUC	コードセット3を使用します。

ページ先頭へ

■amend1
1バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを指定します。省略した場合、出力コードで指定した文字コードの"_"(アンダーバー)を代替文字とします。

ページ先頭へ

■amend2
2バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを指定します。省略した場合、出力コードで指定した文字コードの"■"(塗りつぶした四角)を代替文字とします。

ページ先頭へ

■delimita
RDBローダ型の項目間の区切り文字を指定します。以下の値の中から指定して下さい。なお、指定する文字は1バイトの1文字でなければなりません。

省	","(カンマ)を区切り文字とし

略(NULL)	て使用します。
文字	指定文字を区切り文字として使用します。

ページ先頭へ

■noblc
変換後の文字項目データについて、後続空白を削除する/しないを指定します。
出力側のファイル形式がデータファイル/RDBローダ型ファイルに対して有効です。

BLC_ON	後続空白を削除する。
BLC_OFF	後続空白は削除しない。

ページ先頭へ

■fjiconv
コード変換に使用するiconv関数の指定を行います。

FJICV_OFF	iconv関数が使用できる場合、内部的にiconvを呼出します。また、iconv関数が使用できない場合には、内部の変換テーブルを使用しています。
FJICV_ON	iconv関数のロード処理を行い、iconv関数が使用できる場合、内部的にiconvを呼出します。iconv関数が使用できない場合には、エラーを返します。Javaから呼び出す場合には、FJICV_ONにする必要があります。

ページ先頭へ

■noconv
入力コードと出力コードが同じ場合に無変換する／しないを指定します。

NOCONV_ON	同一コード無変換を行います。
NOCONV_OFF	同一コード無変換を行いません。

ページ先頭へ

■msgmax
コード変換時に発生した警告メッセージの出力件数の上限値を以下の値の中から指定します。

0	警告メッセージを出力しません。
1～9999	指定された値を警告メッセージの出力の上限値とします。

ページ先頭へ

■msgcont
コード変換時に発生した警告メッセージの件数がmsgmaxで指定した値に達した時、その後の変換処理を続行するか否かを指定します。省略した場合、MSGCONT_ONとして扱います。

MSGCONT_ON	処理を続行します。
MSGCONT_OFF	処理を中断します。

ページ先頭へ

■policyfname
バリデーションポリシーファイル名を指定します。バリデーションポリシーファイルを使用しない場合はNULLを指定してください。指定されたファイル名が存在しない場合は異常終了します。

ページ先頭へ

■usrfname
利用者側で定義した定義変換テーブルファイル名を指定します。利用者定義ファイルを使用しない場合はNULLを指定してください。指定されたファイル名が存在しない場合は異常終了します。

ページ先頭へ

■layfname
mdportgコマンドにより出力されたレイアウト情報ファイル名を指定します。データファイル変換以外の場合は無視されます。指定されたファイル名が存在しない場合は異常終了します。

ページ先頭へ

■errcode
MDP_init関数でエラーが発生した場合、エラーコードを設定します。付表D.1に設定されるエラーコードとその意味を示します。

ページ先頭へ

■derrcode
MDP_init関数でエラーが発生した場合、エラー詳細コードを設定します。付表D.1に設定されるエラー詳細コードとその意味を示します。

付表D.1 エラーコードおよびエラー詳細一覧

エラーコード errcode	エラーの種類	詳細エラーコード derrcode	エラーの詳細
1	MDPORT_DIRのエラー	0	MDPORT_DIRに指定するディレクトリの指定に誤りがあります。
2	利用期限のエラー	0	仮提供版の期限切れなど利用期限に関する異常が発生しました。
3	設定したパラメタの誤り	1	入力レコード形式（itype)の指定に誤りがあります。
		2	出力レコード形式（otype)の指定に誤りがあります。
		3	ISO/エミュレータモード(isomode）の指定に誤りがあります。
		4	入力半角カナモード(ikana)の指定に誤りがあります。
		5	出力半角カナモード(okana)の指定に誤りがあります。
		6	同一コード無変換フラグ(noconv)の指定に誤りがあります。
		7	メッセージ出力情報（msgmax)の指定に誤りがあります。
		8	入力レコード属性(irecfm)の指定に誤りがあります。
		9	出力レコード属性(orecfm)の指定に誤りがあります。
		10	シフトコード除去モード（shift)の指定に誤りがあります。
		11	コードセット３モード（cs3mode)の指定に誤りがあります。
		12	入力レコード区切り文字(irecdlm)の指定に誤りがあります。
		13	出力レコード区切り文字(orecdlm)の指定に誤りがあります。
		14	入力文字コード(icode)の指定に誤りがあります。
		15	出力文字コード(ocode)の指定に誤りがあります。
		16	レイアウト定義ファイル（データ定義文）が必須の変換でデータ定義文の指定がありません。

		17	1バイト系代替文字（amend1) の指定に誤りがあります。
		18	2バイト系代替文字（amend2) の指定に誤りがあります。
		19	厳密チェックの環境変数の指定に誤りがあります。
		20	文字種ポリシーチェック指示フラグ(validation)の指定に誤りがあります。
		21	文字種ポリシーチェック指示フラグ(validation)と同一コード無変換フラグ(noconv)の組み合わせに誤りがあります。
		22	文字種ポリシーチェック指示フラグ(validation)と変換パスの組み合わせに誤りがあります。
		23	指定されたバリデーションポリシーファイル名が1024バイトを超えています。
		24	指定されたバリデーションポリシーファイルが存在しません。
		25	指定されたバリデーションポリシーファイルはディレクトリです。
		26	バリデーションポリシーファイル名が指定されていません。
4	メモリ確保エラー	0	メモリ不足が発生しました。
5	iconvキーワードの誤りまたは利用者定義変換テーブルの記述誤り	0	環境変数に設定するiconvキーワードが長すぎます
			iconvのロードに失敗しました。iconvが正しくインストールされていないかLD_LIBRARY_PATHの指定が正しくない可能性があります。
			iconvのオープンに失敗 しました。
			iconv関数がエラーを返しました。
			iconvのクローズに失敗しました。
			iconv_openの関数ポインタの取得に失敗しました。
			iconvの関数ポインタの取得に失敗しました。
			iconv_closeの関数ポインタの取得に失敗しました。
			Fjiconvのオープンに失敗しました。Fjiconvのキーワードが間違っている可能性があります。
			Fjiconv関数でエラーが発生しました。
			Fjiconvのクローズに失敗しました。
			利用者定義変換テーブルファイル操作でエラーが発生しました。
			必要なメモリ領域を確保できません。
			変換できないパスが指定されました。iconvが必要なパスでiconvを使用しない指定がされているかもしれません。
			etc 配下の変換テーブルのファイル操作

			に失敗しました。 MDPORTのインストールにフォルダに問題が発生しました。
			利用者定義変換ファイルのアクセスエラーが発生しました。
			利用者定義変換ファイルに記述規則違反があります。
			利用者定義変換ファイルの変換元が指定できない範囲を指定しています。
			利用者定義変換ファイルの変換先が指定できない範囲を指定しています。
6	レイアウト定義ファイル（データ定義文）に関するエラー	1	レイアウト定義ファイルが存在しません。
		2	レイアウト定義ファイルのファイル操作でエラーが発生しました。
		3	MDPORTのバージョンとレイアウト定義ファイルのバージョンに不整合があります。

ページ先頭へ

■validation
バリデーション機能を適用した変換を行う／行わないを指定します。

VALIDATE_NO	バリデーション機能を適用した変換を行いません。
VALIDATE_YES	バリデーション機能を適用した変換を行います。

[復帰値]

正常終了の場合確保されたコード変換情報のアドレスが返されます。異常終了の場合はNULLが返されます。NULLが返却された場合、以下のエラーが考えられます。

-	MDPcvinf構造体の各項目に設定した値が間違っている
-	環境変数が設定されていない、環境変数が不正である
-	製品が正しくインストールされていない。
-	システムのメモリ領域の不足
-	システムエラー

[注 意]

MDPcvinf構造体の各メンバの設定値の意味、または省略値等については特に記述の無い限り、mdportfコマンドの該当オペランドと同意です。詳細については、"3.1.1 mdportfコマンド"を参照してください。

MDP_init()関数で初期化を行わずに、他のレコード変換関数を呼び出さないで下さい。動作は保証されません。

MDP_init()関数での初期化は1度だけ行ってください。重複して初期化を行う必要はありません。

mdportgコマンドとMDP_init()関数で指定する入出力ファイル形式は同じにして下さい。動作は保証されません。

[名 前]

MDP_initmsg - メッセージ格納領域の確保を行います。

[形 式]

```
#include "MDPcomm.h"
char *MDP_initmsg(char *inf);
```

[機能説明]

本関数は、MDP_init()で取得したコード変換情報を受け取り、このコード変換情報に対応するメッセージを格納するメッセージ格納領域を確保し、メッセージ格納領域の先頭アドレスを復帰値として返します。確保されるメッセージの数はMDP_init()で指定するMDPORT変換指示構造体のmsgmaxで設定します。

[復帰値]

メッセージ格納領域の先頭アドレスが返されます。異常終了の場合はNULL値が返されます。

[名 前]

MDP_conv - 文字コードの変換/レコード形式の変換を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_conv(char *inf, char *irec, int ireclen, char *orec, int *oreclen, char *msginfo);
```

[機能説明]

本関数は、MDP_init()、MDP_initmsg()の復帰値として返されたコード変換情報inf、メッセージ格納情報msginfoに従い、入力レコードirecをireclenバイトだけ読み込み、変換結果、変換後バイト長をそれぞれorec、oreclenに返します。変換時に発生したメッセージはMDP_initmsg()で確保したメッセージ格納領域へ格納されます。メッセージの取得にはMDP_getmsg()、MDP_nextmsg()あるいはMDP_outputmsg()を使用します。なお、orecの領域はireclen、文字コード、ファイルタイプなどを考慮したうえで呼び出し元で十分な領域を確保し、そのバイト長をoreclenに設定して下さい。

[復帰値]

正常終了時はMDPORT_OKを、異常終了時はMDPORT_NGを返します。

[注 意]

- 入出力レコードが行区切り文字ありの場合、それぞれのレコード長は、行区切文字長を含むものとします。
- 入出力レコード長に指定できる最大レコード長は、32767バイトまでです。
- 入出力レコードが可変長レコードの場合、RDW(Record Descriptor Word)は含まないものとします。

[名 前]

MDP_getmsg - メッセージ情報の取得を行います。

[形 式]

```
#include "MDPcomm.h"
char *MDP_getmsg(char *msginfo);
```

[機能説明]

本関数は、指定されたMDP_initmsg()で取得したメッセージ格納情報msginfoに対応するコード変換情報に対して行われたコード変換処理MDP_conv()で発生したメッセージの一つを返します。1回のコード変換処理で複数のメッセージが発生した場合には、MDP_nextmsg()、MDP_prevmsg()関数を使用してメッセージ格納情報の格納位置を変更することにより、2つ目以降のメッセージを取得することができます。

[復帰値]

コード変換で発生したメッセージ文字列の1つの先頭アドレスを返します。メッセージが無い場合はNULLを返します。

[名 前]

MDP_nextmsg - メッセージ格納領域内の次のメッセージへの移動を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_nextmsg(char *msginfo);
```

[機能説明]

本関数は、指定されたMDP_initmsg()で確保したメッセージ格納情報msginfoに対応するコード変換情報に対して行われたコード変換処理のMDP_conv()において発生したメッセージ格納領域の、現在のメッセージ位置の次の位置へ移動を行います。現在のメッセージよりも前に移動する場合はMDP_prevmsg()を使用します。

[復帰値]

正常に次メッセージへ移動した場合はMDPORT_OKを、メッセージの位置が移動できない(ポイントが最後)場合はMDPORT_NGを返します。

[名 前]

MDP_prevmsg - メッセージ情報格納領域の前のメッセージへの移動を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_prevmsg(char *msginfo);
```

[機能説明]

本関数は、指定されたMDP_initmsg()で確保したメッセージ情報格納領域へのアドレスに対応するコード変換情報に対して行われたコード変換処理のMDP_conv()において発生したメッセージ情報領域の、現在のメッセージ位置の前の位置へ移動を行います。現在のメッセージよりも次に移動する場合はMDP_nextmsg()を使用します。

[復帰値]

正常に前メッセージへ移動した場合はMDPORT_OKを、メッセージの位置が移動できない(ポイントが先頭)場合はMDPORT_NGを返します。

[名 前]

MDP_outputmsg - 一回のコード変換処理で発生したメッセージ格納領域のすべてのメッセージをファイルへ出力します。

[形 式]

```
#include "MDPcomm.h"
int MDP_outputmsg(char *msginfo, char *filename, int mode);
```

[機能説明]

本関数は、指定したMDP_initmsg()で取得したメッセージ格納領域msginfoに格納されている全てのメッセージをfilenameで指定したファイルへmodeで指定したモードで出力します。メッセージ格納領域msginfoは、コード変換処理MDP_conv()が呼び出されるたびに初期化されます。このため、一度呼び出したMDP_conv()のメッセージ情報を2回目以降のMDP_conv()で取得することはできません。modeで指定する値は以下の値です。なお、出力されるメッセージの形式については"付録C.2 警告メッセージ"を参照してください。本関数は、マルチスレッド環境での動作は保証しておりません。

0	filenameでファイルを作成し、同一名のファイルが存在する場合は上書きします。
1	filenameでファイルを作成し、同一名のファイルが存在する場合は上書きせず、終了します。
2	filenameでファイルを作成し、同一名のファイルが存在する場合はファイルの終了位置に追加書き込みを行います。

[復帰値]

出力できればMDPORT_OKを、できなければMDPORT_NGまたはエラーコードを返し、エラーメッセージを標準出力へ表示します。

[名 前]

MDP_freemsg - メッセージ格納領域の解放を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_freemsg(char *msginfo);
```

[機能説明]

本関数は、指定されたMDP_initmsg()で確保したメッセージ格納情報msginfoの解放処理を行います。

[復帰値]

正常に解放できればMDPORT_OKを、できなければMDPORT_NGを返します。

[名 前]

MDP_fin - レコード変換関数の終了処理を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_fin(char *inf);
```

[機能説明]

本関数は、MDP_init()の復帰値として返されたコード変換情報infの領域を解放し、レコード変換の後処理を行います。

[復帰値]

正常終了時はMDPORT_OKを、異常終了時はMDPORT_NGを返します。

[名 前]

MDP_stat - 詳細情報を取得します。

[形 式]

```
#include "MDPcomm.h"
int MDP_stat(int para,  char *inf, char *msginfo);
```

[機能説明]

本関数は、MDP_init()の復帰値として返されたコード変換情報infと、MDP_initmsg()の復帰値として返されたメッセージ格納情報msginfoからparaに従った情報を取得します。paraに以下の値を指定することができます。

MDP_WARNUM	ワーニング発生件数を取得します。
MDP_ERROR	本関数を呼び出す直前のMDP_conv()で発生したエラー詳細コードを取得します。

[復帰値]

paraに指定された値により以下の値が返されます。
MDP_WARNUM

0	: ワーニングは発生していません。
正の整数	: ワーニング発生件数

MDP_ERROR

0	: 正常終了しました。
MDPERR_PARA	: 引数に誤りがあります。
MDPERR_CONV	: コード変換エラー(ワーニング)が発生しました。
MDPERR_ENV	: 環境に誤りがあります。
その他の数値	: システムのerrnoです。

[注 意]

- 本関数では、MDP_init()、MDP_initmsg()の詳細情報は取得できません。MDP_init()の詳細情報はメッセージの出力先ファイルまたは、標準エラー出力を参照して下さい。
- 本関数によって返されるエラー詳細情報は、直前に呼び出されたレコード変換関数についてのみの情報です。対象とするレコード変換関数から復帰後、本関数を呼び出す前に、他のシステムコールや関数を呼び出すとエラー詳細情報は無効になります。

D.2.1 レコード変換機能(旧形式)[Solaris]

レコード変換機能は、レコード単位にコード変換、レコード変換を行う関数群を提供します。
本節では、レコード変換機能の各関数仕様について説明します。

MDP_init	レコード変換関数の初期化を行います。
MDP_conv	文字コードの変換/レコード形式の変換を行います。
MDP_fin	レコード変換関数の終了処理を行います。
MDP_stat	詳細情報を取得します。

[名 前]

MDP_init - レコード変換関数の初期化を行います。

[形 式]

```
#include "MDPcomm.h"
char *MDP_init(struct MDPcvinf *inf);
```

[機能説明]

本関数は、infが指すコード変換指示に従って、コード変換テーブルの展開/レイアウト情報の展開を行い、コード変換情報として、その先頭アドレスを復帰値として返します。MDPcvinf構造体宣言とその設定内容は以下の通りです。

```
struct MDPcvinf { /* MDPORT 変換指示構造体   */
    char        icode[8];   /* 入力文字コード          */
    char        ocode[8];   /* 出力文字コード          */
    int         itype;      /* 入力ファイル形式        */
    int         otype;      /* 出力ファイル形式        */
    int         irecdlm;    /* 入力レコード改行コード  */
    int         orecdlm;    /* 出力レコード改行コード  */
    int         isomode;    /* ISO/エミュレータモード  */
    int         ikana;      /* 入力半角カナモード      */
    int         okana;      /* 出力半角カナモード      */
    int         irecfm;     /* 入力レコード属性        */
    int         orecfm;     /* 出力レコード属性        */
    int         shift;      /* シフトコード除去モード  */
    int         cs3mode;    /* コードセット3モード     */
    int         amend1;     /* 1バイト系代替文字       */
    int         amend2;     /* 2バイト系代替文字       */
    char        delimita;   /* CSV区切り文字           */
    char        noblc;      /* 後続空白削除抑止フラグ  */
    char        filler[2];  /* 未使用                  */
    int         msgmax;     /* メッセージ出力MAX 件数  */
    int         msgcont;    /* MSG 上限到達処理続行    */
    char        *msgfname;  /* メッセージ出力先ファイル名 */
    char        *usrfname;  /* 利用者定義ファイル名    */
    char        *layfname;  /* レイアウト情報ファイル名 */
} ;
```

icode, ocodeはそれぞれ入力文字コード、出力文字コードのキーワードを指定します。

euc	ASCIIおよびEUCコード
sjis	ASCIIおよびシフトJISコード
jef	EBCDICおよびJEFコード

他社コード	EBCDICおよび他社日本語コード

itype, otypeはそれぞれ入力レコード形式、出力レコード形式を以下の値の中から指定します。

TYPE_T	テキスト
TYPE_D	データ
TYPE_S	COBOL固定長ソース
TYPE_O	RDBローダ型(oracle)
TYPE_R	RDBローダ型(RDB2,Symfoware)
TYPE_I	RDBローダ型(INFORMIX)

irecdlm, orecdlmはそれぞれ入力レコード区切文字、出力レコード区切文字を以下の値の中から指定します。

DLM_CRLF	復帰改行(CRLF)
DLM_LF	改行(LF)
DLM_NO	コードによる区切りなし

isomodはEBCDIC系のコード変換において、以下の値の中から指定します。なお、EBCDIC系以外のコード変換では無視されます。(本オプションの指定内容については、"3.1.1 mdportfコマンド"の-emu | -isoオペランドを参照して下さい。)

ISO	ISOモード変換
EMU	エミュレータモード変換

ikanaは入力の半角カナモードを以下の値の中から指定します。
(入力がEUCコードの時のみ指定が有効になります。)

KANA_E	EUCコード体系のコードセット2(2バイト/1文字)
KANA_J	JIS8コード(1バイト/1文字)

okanaは出力の半角カナモードを以下の値の中から指定します。
(出力がEUCコードの時のみ指定が有効になります。)

KANA_E	EUCコード体系のコードセット2で出力(2バイト/1文字)
KANA_J	JIS8コードで出力(1バイト/1文字)
KANA_G	EUCコード体系のコードセット1(全角カナ)で出力

irecfmは入力レコードの属性を指定します。itypeがデータ、またはEBCDIC系のテキストの場合に有効です。省略した場合は固定長レコードとして扱います。(入力レコード属性については、"3.1.1 mdportfコマンド"の-irecfmオペランドを参照して下さい。)

RECFM_F	固定長レコード

orecfmは出力レコードの属性を指定します。otypeがデータ、またはEBCDIC系のテキストの場合に有効です。省略した場合は固定長レコードとして扱います。(出力レコード属性については、"3.1.1 mdportfコマンド"の-orecfmオペランドを参照して下さい。)

RECFM_F	固定長レコード

shiftはEBCDIC系コードからのデータ変換において、入力レコード上の制御コードの扱いを以下の値の中から指定します。

SHIFT_L	シフトコード除去後 左詰め

SHIFT_B	シフトコードを空白(0x20)へ置換

cs3modeは変換後または変換元のEUCコードとして、コードセット3(3バイト/1文字表現)の使用する/しないを指定します。

NOCS3_BLK	コードセット3を使用しません。
NOCS3_EUC	コードセット3を使用します。

amend1は1バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを指定します。省略した場合、"_"(アンダーバー)を代替文字とします。

amend2は2バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを指定します。省略した場合、"■"(塗りつぶした四角)を代替文字とします。

delimitaはRDBローダ形式の項目間の区切り文字を指定します。以下の値の中から指定して下さい。なお、指定する文字は1バイトの1文字でなければなりません。

省略(NULL)	","(カンマ)を区切り文字として使用します。
文字	指定文字を区切り文字として使用します。

noblcは変換後の文字項目データについて、後続空白を削除する/しないを指定します。
出力側のファイル形式がデータファイル/RDBローダ型ファイルに対して有効です。

BLC_ON	後続空白を削除する。
BLC_OFF	後続空白は削除しない。

msgmaxはコード変換時に発生した警告メッセージの出力件数の上限値を以下の値の中から指定します。

MSG_NO	警告メッセージを出力しません。
0	警告メッセージを無制限に出力します。
1〜9999	指定された値を警告メッセージの出力の上限値とします。

msgcontはコード変換時に発生した警告メッセージの件数がmsgmaxで指定した値に達した時、その後の変換処理を続行するか否かを指定します。省略した場合、MSGCONT_ONとして扱います。

MSGCONT_ON	処理を続行します。
MSGCONT_OFF	処理を中断します。

msgfnameはコード変換時に発生した警告メッセージの出力先ファイル名を指定します。NULLが指定された場合、警告メッセージは出力されません。指定されたファイル名が存在しない場合は新規に作成し、既に存在する場合は上書きします。

usrfnameは利用者定義変換テーブルファイル名を指定してください。利用者定義ファイルを使用しない場合はNULLを指定してください。指定されたファイル名が存在しない場合は異常終了します。

layfnameはmdportgコマンドにより出力されたレイアウト情報ファイル名を指定してください。データファイル変換以外の場合は無視されます。指定されたファイル名が存在しない場合は異常終了します。

[復帰値]

正常終了の場合確保されたコード変換情報のアドレスが返されます。異常終了の場合はNULLが返されます。

[注 意]

MDPcvinf構造体の各メンバの設定値の意味、または省略値等については特に記述の無い限り、mdportfコマンドの該当オペランドと同意です。詳細については、"3.1.1 mdportfコマンド"を参照してください。

MDP_init()関数で初期化を行わずに、他のレコード変換関数を呼び出さないで下さい。動作は保証されません。

MDP_init()関数での初期化は1度だけ行ってください。重複して初期化を行う必要はありません。

mdportgコマンドとMDP_init()関数で指定する入出力ファイル形式は同じにして下さい。動作は保証されません。

[名 前]

MDP_conv - 文字コードの変換/レコード形式の変換を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_conv(char *inf,char *irec,int ireclen,char *orec,int *oreclen);
```

[機能説明]

本関数は、MDP_init()の復帰値として返されたコード変換情報infに従い、入力レコードirecをireclenバイトだけ読み込み、変換結果、変換後バイト長をそれぞれorec、oreclenに返します。なお、orecの領域はireclen、文字コード、ファイルタイプなどを考慮したうえで呼び出し元で十分な領域を確保し、そのバイト長をoreclenに設定して下さい。

[復帰値]

正常終了時はMDPORT_OKを、異常終了時はMDPORT_NGを返します。

[注 意]

- 入出力レコードが行区切り文字ありの場合、それぞれのレコード長は、行区切文字長を含むものとします。
- 入出力レコード長に指定できる最大レコード長は、32767バイトまでです。
- 入出力レコードが可変長レコードの場合、RDW(Record Descriptor Word)は含まないものとします。

[名 前]

MDP_fin - レコード変換関数の終了処理を行います。

[形 式]

```
#include "MDPcomm.h"
int MDP_fin(char *inf);
```

[機能説明]

本関数は、MDP_init()の復帰値として返されたコード変換情報infの領域を解放し、レコード変換の後処理を行います。

[復帰値]

正常終了時はMDPORT_OKを、異常終了時はMDPORT_NGを返します。

[名 前]

MDP_stat - 詳細情報を取得します。

[形 式]

```
#include "MDPcomm.h"
int MDP_stat(int para,  char *inf);
```

[機能説明]

本関数は、MDP_init()の復帰値として返されたコード変換情報infからparaに従った情報を取得します。paraに以下の値を指定することができます。

MDP_WARNUM	ワーニング発生件数を取得します。
MDP_ERROR	本関数を呼び出す直前のMDP_conv()または、MDP_fin()で発生したエラー詳細コードを取得します。

[復帰値]

paraに指定された値により以下の値が返されます。
MDP_WARNUM

0	: ワーニングは発生していません。
正の整数	: ワーニング発生件数

MDP_ERROR

0	: 正常終了しました。
MDPERR_PARA	: 引数に誤りがあります。
MDPERR_CONV	: コード変換エラー(ワーニング)が発生しました。
MDPERR_ENV	: 環境に誤りがあります。
その他の数値	: システムのerrnoです。

[注 意]

- 本関数では、MDP_init()の詳細情報は取得できません。MDP_init()の詳細情報はメッセージの出力先ファイルまたは、標準エラー出力を参照して下さい。
- 本関数によって返されるエラー詳細情報は、直前に呼び出されたレコード変換関数についてのみの情報です。対象とするレコード変換関数から復帰後、本関数を呼び出す前に、他のシステムコールや関数を呼び出すとエラー詳細情報は無効になります。