NuSDaS
（数値予報標準データセットシステム）
バージョン 1.4

2018-08-30

気象庁

改訂履歴

日付	変更点
2007-03-30	初版 (豊田　英司, 原　旅人)
2007-04-16	第 2 版 GS, RG の使い分けと「検証」用の種別追加 (横井　信太郎)
2007-05-02	第 3 版 STDIO を標準にするオプション、入力バッファを使わないオプションの実装と記述の追加 (原　旅人)
2007-06-22	第 4 版 GS と RG の定義ファイルについて、補足説明追加と。Sun sparc でのコンパイルの仕方を記述 (横井　信太郎)
2007-06-28	第 5 版 基本時刻のリスト問い合わせ関数の戻値の説明を追加 (横井　信太郎)
2008-04-03	第 6 版 SUBC DPRD のフォーマットの説明を追加。(横井　信太郎)
2008-06-13	第 7 版 NuSDaS 要素名の追加と要素名の先頭の「 _ 」の扱いを明記。(横井　信太郎)
2008-06-18	第 8 版 NuSDaS 要素名の追加と要素名の先頭の「 . 」の扱いを明記。(横井　信太郎)
2008-12-11	第 9 版 要素名の先頭の「 A_ 」、「 I_ 」の扱いを明記。(横井　信太郎)
2009-01-23	第 9.1 版 誤字修正 (田浦　俊太郎)
2009-01-30	第 10 版 要素名 PSI(流線関数) を追加。(横井　信太郎)
2009-02-19	第 10.1 版 要素の単位 SnWe と SnowD の単位を追加 (横井　信太郎)
2011-11-09	第 10.2 版 「間引きガウス格子」→「適合ガウス格子」(長谷川 昌樹)
2012-09-19	第 10.3 版 ランベルトの表式の誤りを訂正; パッキング 2UPJ の説明 (豊田 英司)
2014-11-21	r4351–r4353 パッキング 1PAC, 2PAC, 2UPJ, 4PAC の説明の訂正; r4354 用語: パラメータ名→要素名 (豊田 英司)
2014-11-25	r4359 (設定初期値の誤り); r4360 (時間範囲に関するドキュメント改善) (豊田 英司)
2014-12-10	r4369 aio, mmap, ibmshmat 廃止 (豊田 英司)
2014-12-12	r4376 実行時オプション GSVB 新設 (豊田 英司)
2015-01-08	r4380 ES 対応, r4386 OpenMP 対応, r4388 実行時・コンパイル時オプションの説明改善 (豊田 英司)
2015-03-31	r4407 リトルエンディアン機でのバグ対応, r4413 SUBC を上書きすると END レコードが壊れる問題対処, r4420 複合差分圧縮, r4432 NAPS インストールツール再整備 (豊田 英司)
2017-06-09	「ビットマスク」と「マスクビット」を「マスクビット」に統一。定義ファイルでバージョン省略した際の挙動を最新の仕様に変更。R4(R8) で欠損値を利用する際の注意点を追記 (雁津 克彦)
2017-07-27	2UPP 展開関数を実装 (雁津 克彦)
2017-10-20	2UPP の圧縮手順を追記 (雁津 克彦)
2017-12-18	PATH 文の説明修正 (雁津 克彦)
2018-06-07	NAPS10 テープライブラリに関する説明追加 (雁津 克彦)
2018-06-15	レコード形式と適合ガウス格子の誤植修正 (雁津 克彦)
2018-08-30	fsync に関する記述を追加 (雁津 克彦)

Contents

1 はじめに

1.1 NuSDaS の概要

1.1.1 NuSDaS とは

NuSDaS は NWP Standard Dataset System の略で、数値予報格子点データ (GPV; grid point value) を格納するために作られたデータ形式である。C および Fortran で NuSDaS データを読み書きするためのサブルーチン集を NuSDaS インターフェイス (または NuSDaS ライブラリ) という。

気象庁の数値予報ルーチンにおいては NuSDaS 形式および NuSDaS インターフェイスの利用が必須とされている(注 1)。これはディレクトリも含めたデータ形式や入出力手段の標準化によって、次のような目標を達成するためである。

- データの読み書きの方法について調整する手間を簡便化する
- データ作成者によるメタデータ (データを利用するために必要となる付随的情報) の提供し忘れを防ぐ
- デコード・可視化・エンコードなどのアプリケーションの共通化を図る

1.1.2 NuSDaS の歴史

数値解析予報システムは第 6 世代 (NAPS6; 1996 年 3 月–2001 年 2 月) GPV の格納には GVD1 (直接編成ファイル) および GVS1 (順番編成ファイル) という形式が用いられていた。GVD1 や GVS1 という言葉がファイル形式およびその読み書きライブラリを指す点を含め、NuSDaS の先駆者といえるが、この頃のオペレーティングシステム (OS) は UNIX とはまったく異なった VOS3 というもので、たとえば階層的ファイルシステムが存在しないとか、OS のレベルで直接編成ファイルと順番編成ファイルが区別されるなど、現在とはまったく異なる世界であった。

2001 年 3 月から運用された NAPS7 では UNIX (HI-UX/MPP) が用いられることとなり、可変データの取り回しなどの数値予報ルーチン運用上の諸規則 (数値予報ルーチンルール) はすべて再構築しなければならなくなった。データ形式も例外ではなく、ディレクトリを用いたデータセット管理のため NuSDaS が開発され、NAPS7 行用期間中の数値予報ルーチンで使われた。このときの NuSDaS が NuSDaS 1.0 と呼ばれる(注 2)。

NAPS7 はほとんどビッグエンディアン機で構成されていたため、i386-linux などのリトルエンディアン環境での NuSDaS の利用が課題となった。また JRA-25 (電力中央研究所の富士通機) や共生プロジェクト (地球シミュレータ) など気象庁モデルを用いた共同研究が盛んになったこともあり、プログラムの移植性が問われるようになった。このため NuSDaS 1.0 をベースとしてバイトオーダー対応、configure システムなどを備えた portable NuSDaS (pnusdas) が開発され、開発環境で利用された。後に pnusdas にはパンドラ手順によってネットワークを通じて遠隔ホスト上のデータセットにアクセスする機能が追加され、レーダー情報作成装置などで活躍することとなった。

NAPS7 における開発・運用を通じて NuSDaS の設計にはさまざまな問題点が指摘されるようになった。

- データファイルの形式が Fortran 順番探査形式に似ているが違うので直接 Fortran で開くことができない(注 3)
- データファイルの最大長が 2GB (NuSDaS 1.1 以降は 4GB) に制限されている
- 複数の基準時刻のデータを 1 ファイルに納めることができないので小さなデータセットのファイル数が過大となる(注 4)

データモデルの再設計を含んだライブラリの全面的刷新は 2003 年度後半 (NuSDaS 2.0) と NAPS8 移行期 (NuSDaS 3.0) との 2 回試みられたが、いずれも最終的には断念されることとなり、NAPS8 更新当初は pnusdas をベースとしてレコード形式を Fortran の順番編成形式に改め、またデータ出力の効率化を図った NuSDaS 1.1(注 5)が用いられることとなった。

(注 1) ジョブグループ内で閉じたデータのやりとりを除く

(注 2) 開発管理サーバのリポジトリ trunk/pnusdas/honke 以下が NAPS7 ルーチン版に対応する。

(注 3) 国土地理院向け配信 GPV は特別に Fortran 形式に変換されていた。一方研究所向け配信ではこのような措置が行われず、NuSDaS の普及を妨げる一因となった

(注 4) NAPS7 では季節予報モデルがリスタートするためにルーチンルール上は基準時刻毎にファイルを別ける必要があり、ファイル数が過大となるという問題であった

(注 5) 数値予報課 CVS サーバの pandora/pnusdas (特に honke ではなく src サブディレクトリ) のタグ NAPS8_0603 を付した版が対応していたが、該当する版は現在運用されているの開発管理サーバのリポジトリに存在しない

しかしながらこの措置は一時的なものでしかなく、2007 年 5 月からメソ予報が 33 時間に延長され、気圧面予報値ファイルが 4.8 GB に達するためついに NuSDaS ファイル形式の変更とこれに伴うライブラリの大規模改修が不可避となった。

この機にライブラリの構造を全面刷新し見通しのよいプログラムにするとともに、NuSDaS 2.0/3.0 の反省をふまえてデータモデルや API レベルでの互換性を極力重視しつつ、データサイズ制限の撤廃、新しいメタデータのための補助管理部 (2007 年秋に更新予定の T_L959 全球モデルの対応を含む)、仕様の明確化、さらなる入出力の効率化など (詳細は F.4 節 (p. 183) 参照) などを行ったのが NuSDaS 1.3 [(注 6)]である。NuSDaS 1.3 はメソ 33 時間予報から導入されることになる。

なお新しい補助管理部関連 API の実験として先行的に NuSDaS 1.1 に当該機能を追加したもの[(注 7)]が NuSDaS 1.2 と呼ばれている。

NAPS8 においては NuSDaS 1.1 と NuSDaS 1.3 が用いられた。移行当初に NuSDaS 1.1 とリンクして導入されたロードモジュールは特に支障がないかぎりそのまま用いられ、ルーチン変更にあたっても従来どおり NuSDaS 1.1 (libnusdas.pbf, libnwp.pbf) をリンクする申請を出すことができた。

2012 年 6 月から運用されている NAPS9 においては NuSDaS 1.1 が廃され、NuSDaS 1.3 だけが用いられている。

さらに NAPS9 での機能拡張とバグフィックス (詳細は F.5 節 (p. 184) 参照) を導入したものが NuSDaS 1.4 [(注 8)]である。2018 年 6 月に運用開始された NAPS10 では NuSDaS 1.4 のみ利用可能であり、今後の NuSDaS ライブラリのメンテナンスは 1.4 版についてだけ行われる。

(注 6) 開発管理サーバのリポジトリ trunk/nusdas1.3 以下
(注 7) 開発管理サーバのリポジトリ trunk/pnusdas 以下
(注 8) 開発管理サーバのリポジトリ trunk/nusdas1.4 以下

2 利用の手引

2.1 動作環境

- 現在のところ、トランク更新のための動作確認は x86_64 Linux + gcc で行われている
- 些細でない変更は x86_64 Linux + Intel C および XTC OS + 富士通 C でテストされる
- HPPA2 HP-UX や SPARC Solaris は 2007 年頃から動作確認されていない (実機がない)。レコード長が 4 の倍数でないかマスクビットを使うと SIGBUS を起こすことがある
- gcc4.0 以降では、コンパイラの最適化に問題があり float 型が正常に扱えない場合がある
  gcc4.0 以降を利用する場合は、`CFLAGS="-g -O1"`として最適化のレベルを下げることによりこの問題を回避できる。ただし、本件は少なくとも gcc4.8.5 では解決しているようである。

2.2 利用手順

2.2.1 利用手順概説

1. 使っている計算機に NuSDaS ライブラリがインストールされていなければインストールする
   - NuSDaS ライブラリのソースコードを取得する
   - NuSDaS ライブラリをコンパイル・インストールする (詳細は INSTALL ファイルをあわせて参照されたい。)
2. アプリケーションを書く
   - Fortran ならば 5 章 (p. 46) のサブルーチンを呼ぶサブルーチンの先頭で `use "nusdas_fort.h"` する
   - C ならば 6 章 (p. 88) の関数を呼ぶソースコードの先頭で`#include <nusdas.h>` と、通算分の計算で必要ならば`#include <nwpl_capi.h>` を使う
   - その他のヘッダ (数値予報標準ライブラリと降水短時間ライブラリを除く) を使っている C プログラムは NuSDaS のバージョンに依存するため NuSDaS 1.3 以降では使えない
3. アプリケーションをコンパイル・リンクする
   - 数値予報ルーチンでは PBF を書くのでコンパイラに渡すオプションの詳細に付いて心配する必要はない
4. 定義ファイル・データセットを作成する
   - 定義ファイルの構文に付いては 4 章 (p. 34) を参照されたい。
   - データ設計については 3.1 節 (p. 27) を参考にされたい。
5. アプリケーションを実行する

2.2.2 典型的なコマンドライン

とりあえず最低限動かすためにシンプルな例を挙げる。

インストール

```
$ tar xvfz nusdas-1.4.tar.gz
$ cd nusdas1.4
$ F90=f90 sh configure --prefix=/opt/nusdas1.4 --without-zlib
$ make
$ sudo make install
```

ここで `/opt/nusdas1.4` は任意でよいが、書込ができなければならない。一般ユーザ権限しかないならば `/opt/nusdas1.4` とあるところを`${HOME}` と読みかえるとよいだろう。

C 言語のコンパイル

```
$ cc -I/opt/nusdas1.4/include -c app.c
```

C 言語のリンク

```
$ cc -o app app.o -L/opt/nusdas1.4/lib -lnusdas -lnwp -lm
```

Fortran のコンパイル

```
$ f90 -I/opt/nusdas1.4/include -c app.f90
```

Fortran のリンク

```
$ f90 -o app app.o -L/opt/nusdas1.4/lib -lnusdas -lnwp -lm
```

データセットの作成

```
$ mkdir fcst_p.nus
$ mkdir fcst_p.nus/nusdas_def
$ vi fcst_p.nus/nusdas_def/fcst_p.def
$ ln -s fcst_p.nus NUSDAS10
```

このようにして始めてプログラムが実行できる。

2.2.3 ライブラリ間の依存関係

気象庁のライブラリ間の依存性 降水短時間ライブラリ (libsrf) は NuSDaS に依存する。そして NuSDaS は数値予報標準ライブラリ (nwplib) に依存する。したがって、アプリケーションのリンク時には`-lsrf -lnusdas -lnwp`の順で記述すべきである。

ZLib 依存性 ネットワーク NuSDaS で gzip 圧縮を可能に設定していると ZLib が必要になる。

アプリケーションのリンク時に deflate, Inflate, crc32 などのシンボルが未定義だといってエラーになるようならば、リンクのコマンドラインに `-lz` オプションを追加(注 1)されたい。ZLib が OS 標準ではないディレクトリにインストールされている場合は `libz.a` を探して `-L` オプションを `-lz` の前に挿入する。たとえば `/usr/local/lib/libz.a` があるならば、`-L/usr/local/lib -lz` を cc なり f90 のコマンドラインの末尾に追加するとよい。

数学ライブラリ依存性 NuSDaS 1.1 では使っている関数によっては数学ライブラリをまったく使わないでアプリケーションをビルド出来る場合があったが、NuSDaS 1.3 からはほとんど全てのアプリケーションで数学ライブラリが必要となった。

アプリケーションのリンク時に lrint, sin, cos などのシンボルが未定義だといってエラーになるならば、コマンドラインの最後に `-lm` を追加せよ。

(注 1) コマンドラインの最後がよい。

2.2.4 設定情報提供スクリプト nusdas-conf

上述のように NuSDaS ライブラリがどのような設定になっているかによって必要なライブラリが変わって来るため、NuSDaS 1.3 からは必要なコンパイルオプションに関する情報を提供するスクリプト nusdas-conf を提供している。

このスクリプトは「必要なコンパイルオプション」「必要なライブラリを示すリンカオプション」などの文字列を標準出力に印字するので、たとえば次のように使うことができる。

コンパイル

```
$ `/opt/nusdas1.4/bin/nusdas-config --cc --cflags -cppflags` \
  -c app.c
```

リンク

```
$ `/opt/nusdas1.4/bin/nusdas-config --cc --cflags` \
  -o app app.o `/opt/nusdas1.4/bin/nusdas-config --ldflags --libs`
```

2.3 オプション

NuSDaS 1.4 の動作を調整するためにライブラリのコンパイル時およびアプリケーションの実行時にさまざまなオプションを指定できます。

2.3.1 クイックガイド

なんか動きがおかしい時

みなさんのアプリケーションをそのまま、環境変数を追加して実行時オプション GDBG を指定して実行すると詳細ログが標準エラー出力に出てきます。お問い合わせの前にこれを採取していただくと話が早いです。

```
$ NUSDAS_OPTS=GDBG ./a.out args ...
D   pds_open.c:1830 $PANDORA_SERVER_LIST unset
D glb_dsscan.c:78   allds_push_callback => 0
D       dset.c:102  allds_push => 0
D  api_write.c:69   --- NuSDaS_write _GSMLLPP.FCSV.STD1/2006-07-24t1200/none/20
06-07-24t1200/SURF/PSEA
D   io_posix.c:368  open(NUSDAS10/PPSTD1/200607241200, 0102, 0755) => 3
...
D   io_posix.c:46   lseek skipped pio.fd.tell=384
D   io_posix.c:250  write(3, 164) => 164
D   io_posix.c:125  close(3) => 0
D   dds_open.c:506  df_close => 0
D glb_dsscan.c:94   listp_each => 0
```

何らかの事情により環境変数が使えない場合は、カレントディレクトリの設定ファイルを使えます。

```
$ echo GDBG > nusdas.ini
$ ./a.out args ...
```

なお、この機能は configure スクリプトに `--disable-debug` オプションを与えている場合は使えません。NAPS10 にインストールされているライブラリは`--disable-debug` が指定されています。

入出力バッファ長を増やしたい

ディスクの性能によっては、バッファ長を大きくすると性能が改善することがあります。

まず、次表によってデータファイル入出力の方式を判定します。

	実行時オプション		
コンパイル環境	なし	ISTD	IPSX
(数値予報ルーチン環境)	stdio	stdio	POSIX API
`configure --with-sio`	stdio	stdio	POSIX API
`configure`	POSIX API	stdio	POSIX API

よく分からない場合は、前項の GDBG オプションをつけて実行すると、色々のメッセージでソースコードのファイル名が表示されます。`io_stdio.c` が使われる場合は stdio (fopen などのライブラリ関数) で、`io_posix.c` が使われる場合は POSIX API (open などのシステムコール) です。

まず stdio の場合、既定のバッファ長は 512 KB ですが、KB 単位の数値を GSVB オプションで指定することによってさらに拡張することができます。次の例は 16 メガバイトのバッファを与える例です。

```
$ NUSDAS_OPTS=GSVB:16384 ./a.out args ...
```

POSIX API の場合、実行時オプション FWBF が書き込みバッファ長 (KB 単位) を指定し、 FRBF が読み込みバッファ長 (2 の指数マイナス 1) を指定します。既定値は出力にバッファなし、入力に 256 KB のバッファをとります。次の例は読み書きともに 16 メガバイトのバッファを与える例です。

```
$ NUSDAS_OPTS=FWBF:16384,FRBF:24 ./a.out args ...
```

後述日立 CSES ライブラリを使う場合も同じ FWBF/FRBF オプションが使えます。

OpenMP を使いたい

デフォルトで OpenMP 指示行が入るようになっていますので、コンパイラのオプションで OpenMP を有効にしてください。たとえば gcc であれば次のように CFLAGS 環境変数を与えて configure を起動すれば Makefile にコンパイルオプションが書き込まれるでしょう。

```
$ CFLAGS="-fopenmp -O2" CC=gcc sh configure (その他のオプション)
$ make
```

OpenMP 3.1 以降が使える場合は、configure に `--enable-omp=31` を与えると適切な指示行が使われるようになります。

NuSDaS の OpenMP 対応は、ビットパックなどの配列演算を並列化しようとするものです。複数の並列スレッドから API を呼び出して使えるようにはできていません。

MPI で高速化したい

無理です。

NuSDaS には複数プロセス間で競合を解決するような機構がありません。したがって、MPI の複数プロセスから同じデータファイルに書き出しを行うと、まともには動きません。気象庁では、MPI のランク 0 のプロセスに出力を担当させ、他のプロセスで演算を行うようにしています。

上記の方法では出力が律速になってしまい、なおかつ出力を行う MPI プロセス数を増やせば総帯域が向上するような場合 (比較的計算量が少ないまたはメモリバンド幅が大きい場合、あるいは所謂 I/O ノードを多数占有できる計算機の場合)、2 個から数個のプロセスで別のデータセットに出力するようにすれば、時間短縮が見込めるはずです。

日立 CSES ファイルシステムを使いたい

CSES ファイルシステムは一種のメモリファイルシステムです。今日的には所要時間短縮というよりは、ディスク出力負荷集中軽減を狙って使われることになるだろうと思います。

コンパイル前の configure に `--enable-cses` を与えると CSES ライブラリが見つかった場合には使うようにコンパイルされます (CSES ライブラリが見つからない場合には POSIX ファイルシステムでエミュレーションされます)。

さらに定義ファイルには `PATH NWP_ESF` 及び `OPTION IESF` を書いてください。また、実行時には環境変数 `CSES_DEVICE_NAME` を与えて CSES デバイス名を指定してください (エミュレーション時はデータファイルを置くディレクトリ名)。これで、データファイルは NRD の下ではなく CSES デバイスに置かれるようになります。

環境変数 `CSES_DEVICE_NAME` を与えない場合の既定値は `es_dev` ですが、エミュレーション時には `/dev/shm` となります (Linux tmpfs によるデバグを想定)。

既存のアプリケーションを改修せずに CSES 出力時の速度性能を評価したい場合は、環境変数で実行時オプションを `NUSDAS_OPTS=IESF,GPTH:NWP_ESF` のように指定してやれば、すべてのデータセットについて、あたかも `PATH NWP_ESF` 及び `OPTION IESF` が書かれているかのように動作します。この他に実行時オプション FWBF, FRBF, GESP, GESS などがチューニングに使われます。

ライブラリ内部の実行時間測定

NuSDaS には簡易なプロファイリング（実行時間測定）機能があります。これを有効にするには configure に `--enable-profile` を与えてコンパイルする必要があります。そのうえで実行時オプション GRPF を指定すると、プログラムの終了時にカレントディレクトリの `nusprof.txt` に次のような書式で実行時間 (マイクロ秒単位) を出力します (ファイルが既にある場合は追記します)。

```
GRPF 00:          0.000048
GRPF 01:          0.000423
GRPF 02:          0.000108
GRPF 03:          0.000018
GRPF 04:          0.000000
GRPF 05:          0.000000
```

項目番号はそれぞれ
00: ユーザロジック (最初の API 呼び出し以前はカウントされません)、
01: NuSDaS インターフェイスの下記以外 (主にデータセット探索)、
02: データレコード書き出し、
03: データレコードのエンコード、
04: 未使用 (ゼロになる)、
05: データレコード書き出し時の `write(2)` または `es_write(2)` システムコールです。
ちょっと分類がわかりにくいと思われるので今後改訂するかもしません。

2.3.2 configure オプション

NuSDaS ライブラリをコンパイルする際に実行する configure スクリプトは、環境変数とコマンドラインオプションによっていろいろの設定ができます。

次の例は Fortran コンパイラ gfortran と C コンパイラ gcc を用いて、JasPer ライブラリと OpenJPEG ライブラリを使わないように設定を行います。

```
$ F90=gfortran CC=gcc sh configure --disable-jasper --disable-openjpeg
```

configure に与える環境変数

`ARFLAGS` オブジェクトファイル `*.o` からライブラリ `lib*.a` を作るのには ar コマンドが用いられます (環境変数 `AR` で変更可能)。通常は `ar rv libXXX.a *.o` のように起動するのですが、AIX で 64 ビットオブジェクトを扱う場合は `ar -X32_64 rv libXXX.a *.o` のように起動する必要があります。そこで `AR="-X32_64 rv"` のように環境変数を与えることになります。環境変数の値に空白が入るので、引用符をつけなければなりません。

`CC` C コンパイラの名前を指定します。ほうっておくと gcc があれば優先されますので、 cc を使いたいなら `CC=cc` とします。

`CFLAGS` C コンパイラに与えるオプションを指定します。たとえば最適化オプション -O3 を渡すならば `CFLAGS=-O3` とします。なお、Makefile に同じ名前の CFLAGS マクロが書き込まれますが、configure が別のオプション (-I など) を付加しますので、configure の後で `make CFLAGS=-O3` のように指定しても同じようには動きません。

`F90` Fortran コンパイラの名前を指定します。configure のオプション `--with-f90=`*compiler* でも指定できますので、そちらも参照してください。

`FFLAGS` Fortran コンパイラに与えるオプションを指定します。

configure に与えるコマンドラインオプション

`--prefix=`*path* NuSDaS ライブラリのインストール先を指定します。既定値 `--prefix=/usr/local` を指定した場合はライブラリが `/usr/local/lib` に、ヘッダファイル等が `/usr/local/include` に、nusdas-config コマンドが `/usr/local/bin` にインストールされます。システムディレクトリの書き込み権限をもっていない場合はホームディレクトリなどを指定するとよいでしょう。

`--without-net` ネットワーク NuSDaS 機能を無効にする。NAPS10 ルーチン環境では_net と名前の付いたライブラリ以外こうなっている。

`--disable-debug` デバッグ機能を無効にする。NAPS10 ルーチン環境からは無効にしていないので環境変数 (2.3.3) の指定によりデバッグ出力が可能となっている。

`--enable-profile` 組込みプロファイラを有効にする。

`--enable-le` ビッグエンディアンと誤判定されてしまうリトルエンディアン機で強制的にリトルエンディアンにする。

`--without-srf` 降水短時間ライブラリをビルドしない。

`--with-f90=`*command* Fortran 90 コンパイラを指示する。単に `--with-f90` とすると、環境変数 F90, ifort, gfortran, f95, f90 の順に試す。

`--enable-dfver` デフォルトのファイル出力形式を 14 から 11 に変更する。NAPS10 ルーチン環境ではこうなっている。

`--enable-dfver=13` デフォルトのファイル出力形式を 14 から 13 に変更する。

`--with-sio` STDIO による入出力をデフォルトとする。NAPS10 ルーチン環境ではこうなっている。

`--without-zlib` ネットワーク NuSDaS 内部で使われる圧縮機能を使わない。

`--disable-nusmalloc` メモリが足りなくなったら整理する機能を使わない。速いがメモリが足りなくなったときに落ちやすくなる。

`--enable-cses` 日立 CSES 対応機能を有効にする。

`--disable-omp` トランク r4386 (2015-01-05) 以降では、デフォルトでは OpenMP 並列化のための指示行をソースコードに入れるようになった (コンパイラが対応していない、あるいは適切な最適化オプションが指定されていない場合は並列実行はされない)。本オプション `--disable-omp` はそれを無効にして、日立最適化 C のための要素並列化オプションを有効にする (これもコンパイラの適切なオプションが指定されなければ無視される)。

`--enable-omp=31` OpenMP 3.1 以降で使えるようになった `#pragma omp for reduction` 指示行を挿入するようになる。首都圏装置の_omp と名前の付いたライブラリに設定されている。

`--disable-jasper` JasPer ライブラリを利用しなくなる。2UPJ 形式を扱う場合は JasPer ライブラリ又は OpenJPEG ライブラリのどちらかが必要。JasPer ライブラリと OpenJPEG ライブラリの両方が有効な場合は OpenJPEG ライブラリの利用が優先される。

`--disable-openjpeg` OpenJPEG ライブラリを利用しなくなる。2UPJ 形式を扱う場合は JasPer ライブラリ又は OpenJPEG ライブラリのどちらかが必要。JasPer ライブラリと OpenJPEG ライブラリの両方が有効な場合は OpenJPEG ライブラリの利用が優先される。

`--enable-gpfs-archive-chk` GPFS ライブラリを利用して IBM Spectrum Archive のテープライブラリ上のデータかを判別し、テープ上のデータであれば読込を中止する。NAPS10 ルーチン環境では無効にしているが NAPS10 Pandora 用の NuSDaS では有効にしているため、Pandora 経由でのテープアクセスはできない。

`--enable-fsync` ファイル close の前に fsync を行う。NAPS10 ルーチン環境では有効にしている。

`--host=`*host* 通常不要であるが、クロスコンパイラ環境では以下のようなメッセージが表示され、configure に失敗する場合がある。

```
# 例 1
configure: error: C compiler cannot create executables
See ‘config.log’ for more details
```

```
# 例 2
configure: error: cannot run C compiled programs.
If you meant to cross compile, use ‘--host’.
See ‘config.log’ for more details
```

こうした場合 `--host=x86_64-unknown-linux-gnu` や`--host=sparc64-unknown-linux-gnu` といったホストの指定により解決できる場合がある。なお `--host` を指定した場合 configure から環境変数 INTMODEL を指定するよう指示される場合があるので、表示された選択肢の中から適切なものを設定して再度 configure を行う。

2.3.3 実行時オプション

NuSDaS サブルーチン[注 2]が最初に呼ばれたときに次の場所から実行時オプションが読み取られる。

- カレントディレクトリのファイル “`nusdas.ini`” (全部小文字)
- ついで環境変数 `NUSDAS_OPTS`

読み取られたテキストは次の文法にしたがって設定項目に分解される[注 3]。

- 入力はヌル文字、空白、コンマ (`,`)、またはセミコロン (`;`) によって設定項目に区切られる。
- 設定項目は英数字または下線 (`_`) が 4 文字であり、それに加えて等号 (`=`) またはコロン (`:`) に続けて任意の引数文字列を続けることができる。
- 引数文字列が英数下線以外の文字を含む場合は二重引用符 (`"`) で囲む。
- 引数文字列内で二重引用符を記述するにはバックスラッシュ(\) を前置する。
- 今のところコメントは用意されていない[注 4]。

分解された設定項目は一つ一つ評価され、設定を変更する。認識される設定項目は次の通りである。認識されない設定項目は黙って無視される。引数の記述がない設定項目に引数を与えても無視される。

名前の先頭が ‘G’ の設定項目は、NuSDaS ライブラリ全体に同じ効力をもつ。定義ファイルの OPTIONS 文 (4.15 節 (p. 39)) では上書きできない。

名前の先頭が ‘F’ または ‘I’ の設定項目は、各データセットについて別の設定が管理される。定義ファイルの OPTIONS 文で上書きした設定は他のデータセットに波及しない。

GFCL API 関数が終了するたびにデータファイルを閉じない。(開いたままにする)

`nusdas_iocntl(N_IO_W_FCLOSE, 0)` と `nusdas_iocntl(N_IO_R_FCLOSE, 0)` を併用した場合に同じ。データファイルの操作が終了した後でファイルを閉じる関数 `nusdas_allfile_close` または `nusdas_onefile_close` を呼ぶ必要がある。

GDBG デバッグおよび警告のメッセージを有効化する。`nusdas_iocntl(N_IO_WARNING_OUT, 2)` (これは NuSDaS 1.3 からの新機能だが従来のライブラリでもエラーにはならない) とした場合に同じ。コンパイル時オプションで無効化されることがある。既定値では警告メッセージだけ有効になっている。

GWRN 警告のメッセージを有効化する。`nusdas_iocntl(N_IO_WARNING_OUT, 1)` とした場合に同じ。既定値で有効になっているので今のところ意味はない。

FVER 引数文字列は整数値として評価され、定義ファイルに定めがなかった場合のデータファイルのバージョン番号となる。既定値は `configure` 時の `--enable-dfver` の設定に依存する。`--enable-dfver` なら 11, `--enable-dfver=13` なら 13, これらが無いか `--disable-dfver` なら 14 が規定値となる。判読できない文字列またはライブラリが対応していない版数を与えると最新版である 14 とみなされる。

FWBF 引数文字列は整数値として評価され、nusdas_write (p. 47, 89) 等のデータ出力時のバッファ長 (1024 バイト単位) として用いられる。既定値はゼロでこのとき出力バッファリングは行われない (GSVB も参照)。

FRBF 引数文字列は整数値 f として評価され、nusdas_read (p. 47, 89) 等のデータ入力時のバッファ長が 2^{f+1} バイトになる。0 の場合は入力バッファリングは行われない (GSVB も参照)。負値または 30 以上の値を設定した場合の動作は保証されない。既定値は 17 で、バッファ長 256KB に相当する。ただし、configure で `--with-sio` を指定した場合既定値は 0 となる (数値予報ルーチンはそうなっている)。

GSVB 引数文字列は非負整数値として評価される。既定値は 512 である。非零に設定された場合、標準入出力ライブラリを用いてデータファイルを開く際に (`configure --with-sio` または ISTD 実行時オプション参照)、setvbuf(3) 関数を用いてバッファを拡張する際の大きさとして使われる (1024 バイト単位)。0 になっている場合は setvbuf() を呼ばず、OS のデフォルトの長さのバッファが用いられる。

[注 2] サービスサブルーチン (5.7, 6.7) を除く

[注 3] 関数 nusdas_opts()

[注 4] ファイルにセーブできるのにこれはダメだな。要改修

GALG このオプションはたとえば HP PA-RISC のような、int * に 4 の倍数でないアドレスを設定すると SIGBUS シグナルによりプログラムが異常終了するような環境のためにある。値 4 を設定すると、データ入力時のレコードの先頭アドレスが 4 の倍数でない場合、レコードが別途 malloc() で確保されたバッファに転写されることにより問題を回避する。値 0 を設定するとこのような処置は行われない。既定値は通常の環境では 0 だが、PA-RISC 2.0 環境で configure すると既定値が 4 となる。

GKCF 引数文字列は整数値 n として評価され、`nusdas_parameter_change(N_PC_KEEP_CFILE, n)` と同様、n 個のデータファイルを閉じた後でメモリの掃除を行う。メモリが足りないばあい小さな値を設定すると有効なことがある。デフォルトは -1 で、メモリ掃除をしない。

GBAD データファイル作成時に行う投影法パラメタのチェックで問題がみつかっても処理を続行する: `nusdas_iocntl(N_IO_BADGRID, 1)` と同様。

GRCK GBAD の逆のデフォルト設定で、`nusdas_iocntl(N_IO_BADGRID, 0)` と同じくデータファイル作成時に行う投影法パラメタのチェックで問題がみつかったら処理を続行しない。設定ファイル指示を環境変数で上書きするなどの目的のために用意されている。

GPTH 定義ファイルの PATH 文 4.18 節 (p. 40) の指定を引数文字列で上書きする。例 1: 実行時に強制的に CSES ファイルシステムを使わせるには `GPTH:NWP_ESF,IESF` と指定する。例 2: 相対パス指定の上書きは `GPTH:./dir` のようにする。

IPSX データファイルを開くときに POSIX 標準システムコールつまり open(2), read(2), write(2), close(2) などを用いる。32 ビット環境でも 2GB/4GB を超えるファイルが読み書きできれば対応して open64(2) などが用いられる。デフォルト設定 (ただし configure で変更される)。

ISTD データファイルを開くときに標準入出力ライブラリつまり fopen(3), fread(3), fwrite(3), fclose(3) などを用いる。UNIX ではない環境でも動作するが、32 ビット環境では大抵の場合 2GB (4GB のこともある) を超える大きさのファイルを読み書きすることができない (先頭部分だけとかいうのもダメ)。

IESF 日立 CSES という一種のメモリファイルシステムにデータファイルを置くためのオプション。コンパイル時設定 `configure --enable-cses` によって有効となる。(CSES ライブラリがインストールされていない場合は、open(2) などの標準システムコールでエミュレーションを行うので Linux tmpfs でデータファイル名などの確認ができる)。CSES ES ファイルシステムにはディレクトリ階層がないため、定義ファイルで決まるパス 4.18 節 (p. 40) のうちディレクトリは無視される。通常はジョブスクリプトから固定名称のデータファイルを確保できるようにするために定義ファイルにおいて `PATH NWP_ESF` を指定する。ファイルが置かれる CSES デバイスは環境変数 `$CSES_DEVICE_NAME` で指定される (未定義時は `es_dev`, エミュレーション時は `/dev/shm` が既定値)。なお、コンパイル時設定 `configure --enable-cses` を設定していない場合は `IPSX` とまったく同じ動作になる。

GESP 引数文字列は非負整数と解釈される。既定値は 256 である。上述 IESF オプションによって es_open(2) でファイルが作成される場合、メガバイト単位で初期確保量を指定する。

GESS 引数文字列は非負整数と解釈される。既定値は 256 である。上述 IESF オプションによって es_open(2) でファイルが作成される場合、メガバイト単位で増分確保量を指定する。

IASY 廃止されたオプションで、 `IPSX` と同じ動作になる。ファイルの読み書きに非同期入出力システムコール aio_read(2), aio_write(2) を用いるものであった。

IMMP 廃止されたオプションで、 `IPSX` と同じ動作になる。ファイルの読み書きにメモリマップ入出力 mmap(2) を用いるものであった。コンパイル時設定で無効化された場合 `IPSX` と同じ動作となる。

IBMS 廃止されたオプションで、 `IPSX` と同じ動作になる。mmap(2) のかわりに IBM AIX 固有のシステムコール shmat(SHM_MAP) を用いるものであった。

GRPF 組込みプロファイラを有効にする。コンパイル前に configure に `--enable-profile` を指定しないとこのオプションは有効になりません。

2.4 データ構造

2.4.1 データモデル

NuSDaS で扱うデータは整数または浮動小数点型の 2 次元配列の集まりである。この 2 次元配列はデータ記録と呼ばれる。ほとんどの場合、データ記録の 2 つの次元は水平方向にとられる。データ記録は次の 6 項目の情報で特定される。

種別 *type* データセットを識別する 16 文字の名前。先頭 8 文字の**種別 1** (type1), 続く 4 文字の**種別 2**, 続く 4 文字の**種別 3** からなる。たとえば `_GSMLLPP FCSV STD1` は全球気圧面予報値を表わす、といったようにデータの種類によって大部分その名前が決まる。詳細は B.2 節 (p. 152) 参照。

基準時刻 *basetime* データセットを作成するジョブが基準として参照する時刻。予報・解析データについては初期時刻が用いられ、観測データについては観測時刻の近傍の「きりのよい」(日単位、時単位など) 時刻が用いられる。計算機上では 4 バイト (32 ビット) 符号付き整数でグレゴリオ暦 1801 年 1 月 1 日 0 時 GMT (寛政 12 年 11 月 17 日巳刻) を 0 として分単位で表現される。この表現を**通算分**という。

メンバ名 *member* アンサンブルモデルのメンバー、またはレーダーサイトをあらわす 4 文字の名前。メンバが 1 つしか存在しないデータセットにあってはスペース 4 つ “`ΔΔΔΔ`” が用いられる。

対象時刻 *validtime* データ記録の表現する時刻。基準時刻と同様、通算分で表現される。対象時刻から基準時刻を引いた差を**予報時間** *forecast time* といって対象時刻と区別する(注 5)。

面名 *plane* ほとんどの場合高度を表わす 6 文字の名前。きちんというと、データ記録の 2 つの次元と直交する空間方向の位置。気圧面データの “`500ΔΔΔ`” は 500 hPa 面、地表面は “`SURFΔΔ`” など。表 B.6 参照。

要素名 *element* 物理量を識別するための 6 文字の名前。表 B.5 参照。

上記の 6 項目は **次元** *dimensions* とも呼ばれる。関係データベースに親しい読者は、これらをキーと考えてもよいだろう。名前すなわち種別名、メンバ名、面名、要素名には英字 (大文字と小文字は区別される)、下線、数字が用いられる。

なお、対象時刻および面は範囲指定もできるようになっていたが実際には使われていない。詳細に付いては G 節 (p. 187) を参照。

2.4.2 データの物理構造

NuSDaS データは大から小へ次のような階層構造をもつ。

NuSDaS ルートディレクトリ NuSDaS インターフェイスが取り扱うデータはすべて環境変数 `NUSDAS`*nn* の指すディレクトリまたは `./NUSDAS`*nn* に存在しなければならない。これらのディレクトリを NuSDaS ルートディレクトリ (NRD) という。ここで *nn* は 01 から 99 までの整数で、ファイルの書き込み許可属性にかかわらず 01 から 49 までの NRD は書き込み可、50 から 99 までの NRD は読み込みのみ許可される。数値予報ルーチンにおいては NRD を `fcst_p.nus` などといった (サフィックス `.nus` を持つ) 名称で作成し、`NUSDAS`*nn* はそれへのシンボリックリンクとして作成する(注 6)。

データセット NRD の下には `nusdas_def` というディレクトリがあり、その中の `.def` または `.DEF` で終わる名前のファイル [**定義ファイル**, 4 章 (p. 34)] が NRD 内のデータファイル (後述) の配置およびデータの構造を記述している。ひとつの定義ファイルが記述するデータの総体をデータセットという。ひとつの NRD には複数のデータセットがあってよいが、たいていの NRD はひとつだけのデータセットを含む。データセットに対する操作を行う NuSDaS インターフェイス関数では種別を使ってデータセットを指示する。

データファイル データファイルは Fortran 順番探査ファイルで、その構造は A 節 (p. 138) で詳述する。種別や基準時刻の異なるデータは必ず別のデータファイルに格納される。また、面や要素だけが違う 2 つのデータは必ず同じデータファイルに格納される。メンバや対象時刻がデータファイルを別にするかどうかは定義ファイルの設定による。データセットに対する操作を行う NuSDaS インターフェイス関数では種別、基準時刻、メンバ名、対象時刻を使ってデータセットを指示する。

(注 5) にもかかわらずプログラムで用いる各種の名前および旧ドキュメントには多少の混乱がみられるので注意されたい。
(注 6) これらの処理は new_nusdas.sh で行われる

データ記録 データファイル中に格納されている 2 次元配列。データ記録に対する操作を行う NuSDaS インターフェイス関数では種別から要素名までのすべての次元を使ってデータ記録を指示する。

2.4.3 定義ファイルとデータファイルの関係

定義ファイルに書かれている情報は 3 種類に大別される。

- 種別 [TYPE1 文 (4.25 節, p. 42), TYPE2 文, TYPE3 文]
- ディレクトリ構造 [PATH 文 (4.18 節, p. 40), FILENAME 文 (4.8 節, p. 37)]
- メタデータ (その他)

メタデータはすべてデータファイル (主に CNTL レコード) に格納される。そのため両者の関係に注意が必要である。

データセットには多数の基準時刻のデータファイルを置くことができる。たとえば数年間分のデータファイルが蓄積されて行くような場合、運用の都合上で要素数・要素名・格子系などが微妙に変更されることがある。このような時は nusdas_inq_def (p. 62, 104) と nusdas_inq_cntl (p. 59, 101) の結果が違うことになる。

3 ガイドライン

3.1 データ作成時のチェックリスト

本節では新しいデータを作成する際に考慮すべき事項を一覧する。

3.1.1 種別名の決定

- Table B.1 (p. 152) によってモデル名 (種別 1 の先頭 4 文字) を決める。適切なものがなければ数値予報課プログラム班に連絡して割当を受ける。
- Table B.2 (p. 153) によって 2 次元座標名 (種別 1 の続く 2 文字) を決める。
  - データが子午面断面 (鉛直は気圧座標) の場合 2 次元座標名は `YP` となる。
  - データが東西断面 の場合 2 次元座標名は `XP` となる。
  - データが極座標上の格子の場合 2 次元座標名は `RT` となる。
  - データが河川に沿った格子の場合 2 次元座標名は `FG` となる。
  - データが不規則に分布した地点データの場合 2 次元座標名は `ST` となる。
  - データが細分番号の場合 2 次元座標名は `SB` となる。
  - データが水平方向に 2 次元で等間隔に並んだ格子でできている場合
    - 2 次元座標名は `LL`, `GS`, `RG`, `MR`, `PS`, `LM`, `OL`, `RD` のいずれかである。よくわからなければ C 章 (p. 164) を見て判定されたい。
    - 上記のいずれかが動的に選択され、定義ファイル作成時にはあらかじめ決めがたい場合 [台風モデルなど。C.16 節 (p. 172) を参照] は 2 次元座標名を `XX` とする。
  - それ以外の場合、数値予報課プログラム班に連絡されたい。おそらくこのマニュアルを拡充する必要がある。
- Table B.3 (p. 153) によって 3 次元座標名 (種別 1 の最後の 2 文字) を決める。適切なものがなければ数値予報課プログラム班に連絡して割当を受ける。
- Table B.4 (p. 153) によって属性名 (種別 2 の先頭 2 文字) を決める。適切なものがなければ数値予報課プログラム班に連絡して割当を受ける。
- Table B.5 (p. 154) によって時間種類名 (種別 2 の末尾 2 文字) を決める。適切なものがなければ数値予報課プログラム班に連絡して割当を受ける。
- 種別 3 を決める。英大文字と数字を 4 文字にすることを推奨する。いいかえると、小文字、下線、末尾のスペースは規則上禁止されていないが非公認ツール等で問題を起こすかもしれず、やめておいたほうがいい。また、開発用に作る臨時的データセットに無闇に `STD1` を用いるのはデータセットの混同を引き起こすため避けるべきである。

3.1.2 空間表現に関する考慮事項

- 2 次元座標が `LL`, `LM`, `PS`, `GS`, `RG`, `MR`, `OL`, `RD`, `RT`, `XX` であれば、C 章 (p. 164) に従って投影法パラメタ [SIZE 文 (4.22 節, p. 42), BASEPOINT 文 (4.2 節, p. 34), STANDARD 文 (4.23 節, p. 42), OTHERS 文 (4.16 節, p. 39)] を決める。
- 2 次元座標が `RG` ならば SUBC RGAU レコードを作る。定義ファイルには SUBCNTL 文 (4.24 節, p. 42) を書いておく。データ作成プログラムは nusdas_subc_rgau_preset1 (p. 71, 113) を呼ぶことが推奨される。
- 3 次元座標が `ET` ならば、SUBC ETA レコードを作る。定義ファイルには SUBCNTL 文 (4.24 節, p. 42) を書いておく。データ作成プログラムは nusdas_subc_preset1 (p. 69, 111) を呼ぶことが推奨される。
- 3 次元座標が `SG` ならば、SUBC SIGM レコードを作る。定義ファイルには SUBCNTL 文 (4.24 節, p. 42) を書いておく。データ作成プログラムは nusdas_subc_preset1 (p. 69, 111) を呼ぶことが推奨される。
- 3 次元座標が `ZS` ならば、SUBC ZHYB レコードを作る。定義ファイルには SUBCNTL 文 (4.24 節, p. 42) を書いておく。データ作成プログラムは nusdas_subc_zhyb_preset1 (p. 74, 116) を呼ぶことが推奨される。

鉛直座標に関する SUBC レコードを作成するときの鉛直層数については D 章 (p. 174) を参照されたい。

3.1.3 時間軸に関する考慮事項

- 積算値 (降水量など) や平均値は一定の時間範囲に対して定義されるものですから、対象時刻の通算分ひとつで表現するのには向いていません。そのため、対象時刻を 2 つの通算分の対で指定する API [G 章 (p. 187)、廃止予定] が設計されましたが、実際にはこれらの関数は使われません。積算・平均に使った時間範囲の始点または終点を分単位に丸めたものが対象時刻 1 として使われて、時間範囲は SUBC TDIF レコード [Table A.8 (p. 143)] に格納されます。データ作成プログラムは、すべての対象時刻に

$$\begin{aligned} \mathit{diff\_time} &= [(\text{時間範囲起点}) - (\text{対象時刻 1})]\,/\text{s} \\ \mathit{total\_sec} &= [(\text{時間範囲終点}) - (\text{時間範囲起点})]\,/\text{s} \end{aligned}$$

  を引数にして nusdas_subc_tdif (p. 73, 115) を呼びます。

- 上記 SUBC TDIF レコードを用いて期間内の平均、積算、最大、最小値を格納する場合、要素名の先頭に、期間平均値の場合「 _ 」、期間積算値の場合「 . 」、期間最大値の場合「 A_ 」、期間最小値の場合「 I_ 」を付加することで、瞬間値と区別します。

- データを読むアプリケーションが時間積分のタイムステップを知る必要がある場合は、SUBC DELT レコード [Table A.9 (p. 144)] を作成します。そのためには各データファイルについて 1 回 (自信がなければメンバー・対象時刻が変わるたびに呼べば、重複して書かれたとしても同じところを上書きするだけです) nusdas_subc_delt (p. 67, 109) を呼びます。

3.1.4 その他のメタデータに関する考慮事項

- レーダーデータの場合 SUBC RADR/RADS/ISPC/DPRD レコードが作成されることがある。

3.2 データを読み出す Fortran プログラムの例

以下は、namelist から type1, type2, type3, メンバー名 (member), 面名 (level), 要素 (elem), 初期時刻 (idate), 予報時間 (ft: 単位 hour) を指定して、NuSDaS からデータを読み出して、格子点の値を順にディスプレーに書き出すプログラムである。

```
program nusdas_sample
  implicit none

  character(8) :: type1
  character(4) :: type2, type3

  integer(4) :: ibase, ivalid
  character(4) :: member

  character(6) :: level, elem
  integer(4) :: idate(5)
  integer(4) :: ft

  integer(4) :: gsize(2), nx, ny
  real(4) :: ginfo(14)
  character(4) :: proj, mean

  integer(4) :: dnum, irt
  real(4), allocatable :: data(:, :)
  integer(4) :: ix, jy

  include 'nusdas_fort.h'

  namelist/nampar/ type1, type2, type3, member, level, elem, idate, ft

  ! namelist を読み込む前にデフォルトの値を設定する
  type1 = '_NHMLMLY'
  type2 = 'FCSV'
  type3 = 'STD1'

  member = '    '
  level = 'SURF  '
  elem = 'T     '

  idate(1) = 2004
  idate(2) = 9
  idate(3) = 7
  idate(4) = 12
  idate(5) = 0

  ft = 3

  ! namelist を標準入力から読み込む
  read(5, nampar)

  ! idate から 1801/1/1 0:00 からの通算分に変換する。
  ! 通算分は ibase に格納される。
  call nwp_ymdhm2seq(idate(1), idate(2), idate(3), idate(4), idate(5), ibase)

  ivalid = ibase + ft * 60
```

```
  ! 格子の情報を読み出す
  call nusdas_grid(type1, type2, type3, ibase, member, ivalid, &
    proj, gsize, ginfo, mean, N_IO_GET, irt)
  ! エラーチェック
  if(irt /= 0) then
    write(6, *) 'nusdas_grid error: irt = ', irt
    stop 1
  end if
  nx = gsize(1)
  ny = gsize(2)
  dnum = nx * ny

  allocate(data(nx, ny))

  ! NuSDaS データを読み出す
  call nusdas_read(type1, type2, type3, ibase, member, ivalid, level, elem, &
    & data, N_R4, dnum, irt)
  ! エラーチェック
  if(irt /= dnum) then
    write(6, *) 'nusdas_read error: irt=', irt
  end if

  ! データの書き出し
  do jy = 1, ny
    write(6, *) (data(ix, jy), ix = 1, nx)
  end do

  deallocate(data)

end program nusdas_sample
```

このプログラムを `sample1` という名前でコンパイルし、以下の shell script で実行する。読み出したい NuSDaS Root Directory(NRD) を `NUSDAS??`(??は数字) でシンボリックリンクを張ることがポイントである(注 1)。

```
#!/bin/sh

# 読み出したいデータがある NuSDaS Root Directory (NRD)
DATADIR=/home/taro/fcst_sfc.nus

# 初期時刻の指定
YYYY=2004
MO=09
DD=07
HH=12
MI=00

# TYPE123, member, level, elem の設定
TYPE1="_NHMLMLY"
TYPE2="FCSV"
TYPE3="STD1"
MEMBER="    "

LEVEL="SURF  "
ELEM="T     "
```

(注 1) シンボリックリンクではなく、実体の名前が `NUSDAS??` でもかまわないが、おすすめはしない。

```
FT=3

# NuSDaS Root Directory を NUSDAS?? (??は数字) に
# シンボリックリンクを張る。
# 読み出しの場合は ?? は 50 〜 99, 書き出しの場合は 01 〜 49 とする。
# プログラム内で nusdas_parameter_change を使って、NRD の番号を指定して
# いる場合にはその番号にする。そうでなければ、01〜99 の範囲で任意。
ln -s ${DATADIR} NUSDAS60

# namelist を作成の上、sample1 を実行
cat<<EOF | ./sample1
&NAMPAR
TYPE1='${TYPE1}', TYPE2='${TYPE2}', TYPE3='${TYPE3}', MEMBER='${MEMBER}',
LEVEL='${LEVEL}', ELEM='${ELEM}',
IDATE=${YYYY}, ${MO}, ${DD}, ${HH}, ${MI},
FT=${FT}
&END
EOF

rm -f NUSDAS*
```

3.3 データ読み込み時のよくある質問

3.3.1 データの走査方向

X 方向が (通常地図を描く方向で) 左から右、Y 方向が上から下に走査する。経緯度格子のデータでは、北からデータを格納する。走査順は X 方向に走査した後、Y 方向に走査を一つ進める。

(注) 格子間隔 (DISTANCE) に負値を設定することで格納する方向を逆に指定することは可能だが、現在用例は無い。

3.3.2 データ読み込みの際の NuSDaS の内部動作

NuSDaS は新規にデータファイルを作成する際に、定義ファイルの内容を CNTL レコードに格納する。データ読み込みの際には、データセットを特定する TYPE123, データファイルのパス構造 (path) と対象時刻のリスト (validtime) だけを定義ファイルから参照し、これらの情報から開くべきファイルを決める。ファイルが開ければ、メタ情報は定義ファイルではなくてファイルの CNTL レコードを参照する。従って、データファイルがいったんできてしまえば、定義ファイルのメタ情報を変更しても効果がないことになる。たとえば、データファイルが存在している状態で定義ファイルに新しい物理量を付け加えても、物理量のリストの情報は定義ファイルではなく、既存ファイルの CNTL レコードから取得されるので、効果がない (いったん、データファイルを消す必要がある)。

3.3.3 nusdas_inq_cntl, nusdas_grid の内部動作

各データファイルには一つの CNTL レコードを持っている。これらの API は、この CNTL レコードの内容を問い合わせるものである。

これらの API では、対象時刻 (validtime) までを引数で指定するが、これは開くファイルを特定するためのものであり、その対象時刻のメタデータを指定するものでは**ない**ことに注意が必要である。(ファイルと対象時刻は 1 対 1 に対応するとは限らない)

3.3.4 基準時刻を調べずに読む

観測データなど、対象時刻ひとつに 1 つしかデータ記録がない場合がある。基準時刻はデータ作成の都合で便宜的につけられるが、付け方がいろいろなのでプログラムするのが難しい。

型	規定の欠損値	変更・取得のためのキーワード
1 バイト整数	N_MV_UI1	N_PC_MISSING_UI1
2 バイト整数	N_MV_SI2	N_PC_MISSING_SI2
4 バイト整数	N_MV_SI4	N_PC_MISSING_SI4
4 バイト実数	N_MV_R4	N_PC_MISSING_R4
8 バイト実数	N_MV_R8	N_PC_MISSING_R8

Table 3.1: データ読み取り時に設定される欠損値

こういうときは基準時刻に特別な値 −1 を設定すると、データの読み込みの関数 (例えば nusdas_read) は自動的に指定された対象時刻を持つデータファイルを探索して読み込み動作を行う。

3.3.5 欠損値はどのように得られるか

nusdas_read (p. 47, 89) で得られた配列の欠損値のところに書かれるべき値は、利用者側配列の型に応じて表 3.1 のようになる。この値を変更するにはあらかじめ nusdas_parameter_change (p. 56, 98) を呼び、現在設定されている値を取得するには nusdas_inq_parameter (p. 57, 99) を呼ぶ。

なお、nusdas_parameter_change (p. 56, 98) を呼ぶ際にはセットする値の型に注意が必要である。具体例として

```
call nusdas_parameter_change(N_PC_MISSING_R4, 1.d+10, irt)
```

としてしまうと、API nusdas_parameter_change (p. 56, 98) の第 2 引数は N_MV_R4 と同じ型、すなわち R4 型を期待するため、欠損値の値として 1.d+10 の上位 4byte(すなわち 0.0) をセットしてしまう。この場合の正しい使い方は

```
call nusdas_parameter_change(N_PC_MISSING_R4, 1.e+10, irt)
```

である。

注意 PACKING が I1, I2, I4, R4, R8 の際に欠損値を指定して書き込みを行う場合、nusdas_parameter_change (p. 56, 98) でどのような値を設定しようとも、以下の値は必ず欠損値として扱われるので注意が必要。

- I1, I2, I4 では全 bit が 1 となる -128, -32768, -2147483648
- R4, R8 では INF を除いた最大値に相当する 3.40282347e+38, 1.7976931348623157d+308

なお、 PACKING が 2UPC などの上記一覧に該当しないものは、これに該当しない。

4 定義ファイルリファレンス

4.1 はじめに

4.1.1 定義ファイルの文法

NuSDaS 定義ファイルは概ねフリーフォーマットのテキストである。

- 定義ファイルは改行文字で区切られる行からなる。
- 各行は空白 (スペース、タブ、鉛直タブ、フォームフィードのいずれか) で区切られる語に分解される。
- 行頭の語が '`#`' で始まるとき、その行は無視される。(NuSDaS 1.1 では公式にはコメント機能はなかったが、たまたま定義ファイルパーザの作りがよかったので動作していた)
- 行頭の語が次のキーワード (大文字・小文字は区別されない) のひとつであるとき(注 1)、その名をもつ文の解析が始まる。

  NUSDAS PATH FILENAME CREATOR TYPE1 TYPE2 TYPE3 MEMBER MEMBERLIST BASETIME VALIDTIME VALIDTIME1 VALIDTIME2 PLANE PLANE1 PLANE2 ELEMENT ELEMENTMAP SIZE BASEPOINT DISTANCE STANDARD OTHERS VALUE PACKING MISSING INFORMATION SUBCNTL FORCEDRLEN OPTIONS

- 複数の語からなる文は次の行にまたがって書いてもよい。ただし、行頭の語が上のキーワードになってはならない。
- 複数の文をひとつの行に書くべきではない。NuSDaS 1.1 で読めなくなるからである。
- 一部の文は決まった順序で書かなければならない。
- ELEMENTMAP 文と INFORMATION 文をのぞき、同種の文が複数現れることはない。

4.1.2 凡例

- タイプライタ体 `typewriter font` の文字は文字どおり用いられることを意味する。
- 斜体 *italic* の文字は適宜置き換えるべき語を意味する。
- 「書式」の項の各語は定義ファイルの 1 語に対応する。
- 「書式」の項で "..." とあるのは語数が不定であることを示す。不定といっても先行する語や文などで数は定まるのであり、本文を参照されたい。
- 経緯度を設定するとき λ_*`E`, φ_*`N` のように書いてあるところにそれぞれ λ_*`W`, φ_*`S` のように書けば、それぞれ西経・南緯を設定することができる。
- 経緯度を設定する文の経緯度 (書式の項に λ_*`E` φ_*`N` と書いてある) は逆順に書いてもよい。

4.2 BASEPOINT: 参照点の設定

書式

`BASEPOINT` x_0 y_0 λ_0`E` φ_0`N`

機能 BASEPOINT 文はデータセットの格子番号 (x_0, y_0) が地球上の経緯度 (λ_0, φ_0) を持つ地点であることを設定する。

省略時 あたかも `BASEPOINT 0 0 0 0` が書かれたかのように扱われる。この設定値が適切か否かは 2 次元座標系しだいであり、一概に誤りと断定することはできないが、格子番号は 1 から数え始めることを考えるとほとんどの場合有用ではない。

(注 1) NuSDaS 1.3 以降でもルーチンに見られるスペル誤り OTHER, MISSSING, SUBCTNL と OPTION を許容する。

4.3 BASETIME: 基準時刻の設定

書式

BASETIME *yyyymmddhhnn*

機能 BASETIME 文は定数データセットが持つ唯一つの基準時刻を設定する。ここに設定した以外の基準時刻によるデータの読み書きは失敗するようにコーディングされるべきである、が、NuSDaS 1.1 のコードを見ても引数 *yyyymmddhhnn* を使っているようには見えない。

省略時 データセットには任意の基準時刻のデータファイルを作成できるようになる。

未実装注意 この機能は使われていないので NuSDaS 1.4 でもまだ実装されていない。

4.4 CREATOR: データ作成者の設定

書式

CREATOR *creator*

機能 CREATOR 文はデータファイルの NUSD レコード作成者フィールド (Table A.2 (p. 140) 項番 5) に書き込む文字列を設定する。

省略時 コンパイル時に設定した値が用いられる。デフォルトは

Japan Meteorological Agency, http://www.jma.go.jp/

である。

バグ 空白を含んだ任意文字列を設定する方法がない。

4.5 DISTANCE: 格子間隔の設定

書式 以下のいずれか:

DISTANCE D_i D_j
DISTANCE D_r D_θ
DISTANCE D_X D_Y

機能 DISTANCE 文は格子間隔を設定する。

- 2 次元座標系が LL, GS, RG のとき、DISTANCE D_i D_j は経緯度の度単位での格子間隔 D_i, D_j を設定する。
- 2 次元座標系が RT のとき、DISTANCE D_r D_θ はメートル単位での格子間隔 D_X と度単位での方位角間隔 D_θ を設定する。
- 2 次元座標系が地図投影 (MR, PS, LM, OL) のとき、DISTANCE D_X D_Y はメートル単位での格子間隔 D_X, D_Y を設定する。

省略時 あたかも DISTANCE 0 0 が書かれたかのように動作する。2 次元座標系が ST など、格子間隔の概念をもたず、当然グリッド情報チェックでも値ゼロが許容される場合に使える。

4.6 ELEMENT: 要素数の設定

書式

`ELEMENT` N_E

順序制約 ELEMENTMAP 文 (4.7.1 節, p. 36) よりも先に記述しなければならない。

機能 ELEMENT 文は要素の数を設定する。ELEMENTMAP 文は N_E 個設定する必要がある。

必須性 ELEMENT 文を省略してはならない。

4.7 ELEMENTMAP: 要素名と書込禁止制約

4.7.1 概要

書式 以下のいずれか:

`ELEMENTMAP` *elemname* `0`
`ELEMENTMAP` *elemname* `1` *bitmap ...*
`ELEMENTMAP` *elemname* `2` [n_V *bitmap ...*] *...*

順序制約 MEMBER 文 (4.11 節, p. 38) より前に記述してはならない。VALIDTIME 文 (4.28 節, p. 43) よりも後に記述しなければならない。PLANE 文 (4.19 節, p. 41) よりも後に記述しなければならない。ELEMENT 文 (4.6 節, p. 36) よりも後に記述しなければならない。ELEMENTMAP 文の N_E 個記述しなければならないし、それを越えてもいけない。

機能 定義ファイル中に現れた i_E 番目の ELEMENTMAP 文は要素名 *elemname* とともにその要素に関する書込禁止制約のビットマップ (elementmap と呼ばれる) をデータセットの i_E 番目の要素として登録する。

必須性 ELEMENTMAP 文を省略してはならない。

4.7.2 第 0 種 ELEMENTMAP 文

第 3 語が `0` の ELEMENTMAP 文は、要素 *elemname* に書込禁止制約を設定しないことを意味する。たとえば

```
ELEMENTMAP T 0
```

ならば、`T` という要素はすべてのメンバー、対象時刻、面の組み合わせについて書込可能である。

4.7.3 第 1 種 ELEMENTMAP 文

第 3 語が `1` の ELEMENTMAP 文は、すべてのメンバー・対象時刻に共通のビットマップで要素 *elemname* に書込禁止制約を設定する。第 4 語以下 N_Z [PLANE 文 (4.19 節, p. 41)] 個のビットが並び、各ビットは `0` が書込禁止、`1` が書込許可を意味する。

たとえば

```
PLANE         5
PLANE1                SURF 1000 700 500 300
ELEMENTMAP U       1  0    1    1   1   1
```

ならば、`U` という要素は `SURF` 以外の面について書込可能である。

なお、ビット数が多過ぎるときは行末まで読み飛ばされ、少な過ぎるときは不足分に `0` が補われるが、将来の版でエラーに変更されるかもしれない。

4.7.4 第 2 種 ELEMENTMAP 文

第 3 語が 2 の ELEMENTMAP 文は、すべてのメンバーについて共通だけれど対象時刻によって異なるビットマップで要素 *elemname* に書込禁止制約を設定する。第 4 語以下はビットマップのくり返しである。各ビットマップはくり返し数 n_V を前置した N_Z 個のビットで、くり返し数の合計が N_V [VALIDTIME 文 (4.28 節, p. 43)] となったところで文が終わる。

たとえば

```
VALIDTIME     12 HOUR in
VALIDTIME1 ARITHMETIC 0 1
PLANE         5
PLANE1                    SURF 1000 700 500 300
ELEMENTMAP TKE     2   1 0    0    0   0   0
                      10 0    1    1   0   0
                       1 0    0    0   0   0
```

ならば、要素 TKE は予報時間 1 から 10 の間だけ、面 1000 と 700 についてだけ書込できる。

なお、くり返し数が過大のときは行末まで読み飛ばされ、過小であるときは 0 が補われる、つまり該当する対象時刻は書込禁止となるが、将来の版でエラーに変更されるかもしれない。

4.7.5 第 3 種 ELEMENTMAP 文 (参考)

第 3 語が 3 の ELEMENTMAP 文はメンバーによっても対象時刻によっても異なる ELEMENTMAP を設定できる。第 4 語以下は「メンバーくり返し数を前置した第 1 種または第 2 種 ELEMENTMAP 文の第 3 語目以降」のくり返しであり、メンバーくり返し数の合計が N_M [MEMBER 文 (4.11 節, p. 38)] となったところで文が終わる。

バグ 第 3 種 ELEMENTMAP 文の実装はいいかげんであり、よくテストされていない。

4.8 FILENAME: データファイル名の設定

書式

FILENAME *filename*

機能 FILENAME 文は PATH 文 (4.18 節, p. 40) と組み合わせてデータファイルの名前を設定する。詳細は PATH 文の項を参照せよ。

注意 いわゆる NWP 系のパス指定を用いると FILENAME 文の設定は無視される。

4.9 FORCEDRLEN: 強制レコード長の設定

書式

FORCEDRLEN *nbytes*

機能 FORCEDRLEN 文はデータファイルのレコード長を *nbytes* バイトに設定する。ここでレコード長とはレコード全体、つまり Table A.1 (p. 139) の項番 1–7 の長さである。

省略時 データファイルのレコード長は内容に応じて可変となる。ただし、ファイル形式 13 以降では、レコード長は 8 の倍数となるように調整される。

4.10 INFORMATION: INFO レコードの定義

書式

`INFORMATION` *group nbytes filename*

位置制約 INFORMATION 文は好きな数だけ記述してよい。

機能 INFORMATION 文は群名 *group* の INFO レコードを定義する。データファイル作成時には長さ *nbytes* のファイル *filename* を読み込んでレコード内容が作られる。したがって *nbytes* はレコードのペイロード長すなわち Table A.1 (p. 139) の項番 5 の長さである。

省略時 SUBCNTL 文 (4.24 節, p. 42) と同様に INFO レコードを作ることができなくなる、と言いたいところであるが、残念ながら INFO レコードは後から書き足すことができる。

4.11 MEMBER: メンバー数の設定

書式 つぎのいずれか:

`MEMBER` N_M `IN`
`MEMBER` N_M `OUT`

位置制約 MEMBERLIST 文 (4.12 節, p. 38), ELEMENTMAP 文 (4.7.1 節, p. 36) より先に記述しなければならない。

機能 MEMBER 文はメンバー数を設定する。第 3 語の `IN` は異なるメンバーのデータがひとつのデータファイルに書かれることを意味し、第 3 語の `OUT` は異なるメンバーのデータは別のデータファイルに書かれることを意味する。

省略時 メンバーはただひとつスペース 4 文字 `␣␣␣␣` のものが作られる。

バグ 第 3 語の IN/OUT と PATH 文 (4.18 節, p. 40) や FILENAME 文 (4.8 節, p. 37) との矛盾が検査されない。

4.12 MEMBERLIST: メンバー名の設定

書式

`MEMBERLIST` *name ...*

位置制約 MEMBER 文 (4.11 節, p. 38) より後に記述しなければならない。MEMBER 文を書いたならば省略してはならない。

機能 MEMBERLIST 文は N_M 個のメンバー名 *name* を設定する。メンバー名は 4 文字以下の英数字または下線である。

4.13 MISSING: 欠損値表現法の設定

書式 次のいずれか:

```
MISSING NONE
MISSING UDFV
MISSING MASK
```

機能　MISSING 文は欠損値の表現方法を設定する。それぞれの機能は Table B.9 (p. 155) を参照。

省略時　`MISSING NONE` が仮定される。

注意　UDFV で欠損値を設定して書き込みを行う場合、PACKING が I1, I2, I4, R4, R8 の際には以下の値は必ず欠損値として扱われる。これは nusdas_parameter_change (p. 56, 98) で欠損値として扱う値を変更した場合も変わらない。

- I1, I2, I4 では全 bit が 1 となる -128, -32768, -2147483648
- R4, R8 では INF を除いた最大値に相当する 3.40282347e+38, 1.7976931348623157d+308

なお、 PACKING が 2UPC など I1, I2, I4, R4, R8 以外の場合は本件に該当しない。

4.14　NUSDAS: データファイルの版番号

書式

`NUSDAS` *version*

機能　この定義ファイルを持つデータセットで新規に作成されるデータファイルの形式を *version* に設定する。有効な *version* の値は次の通り:

10 ファイル形式 10 となる。

11 ファイル形式 11 となる。

13 ファイル形式 13 となる。

14 ファイル形式 14 となる。

省略時　configure オプションによる。特に設定せずに configure した場合は `NUSDAS 14` が仮定される。`--enable-dfver` を設定した場合は `NUSDAS 11` が、`--enable-dfver=13` を設定した場合は `NUSDAS 13` が仮定される。NAPS9, NAPS10 およびこれらの関連機器では `NUSDAS 11` が仮定されるようになっている。アプリケーションプログラマは将来デフォルトが `NUSDAS 14` などに変更される可能性を考慮すべきである。

4.15　OPTIONS: データセットの実行時オプションの設定

書式

`options` *optstring*

機能　OPTIONS 文は実行時オプションを設定する。設定ファイル “`nusdas.ini`” または環境変数 “`NUSDAS_OPTS`” と異なり、設定の効力はデータセットに限定される。一方、システム全体の設定 (項目名が ‘G’ ではじまるもの) は設定できない。項目の詳細については 2.3.3 節 (p. 22) 参照。

4.16　OTHERS: 斜軸ランベルトのパラメタ設定

書式

`others` $\hat{\lambda}_P$`E` $\hat{\varphi}_P$`N` λ_E`E` φ_E`N`

機能　OTHERS 文は地図投影パラメタが 5 個以上あるときの設定に用いる。実際には斜軸ランベルト図法でしか使わないので、C.8 節 (p. 169) を参照されたい。

省略時　`OTHERS 0E 0N 0E 0N` が仮定される。斜軸ランベルト以外では OTHERS 文を省略する。

4.17 PACKING: パッキング方式設定

書式

PACKING *packing*

機能 PACKING 文はデータレコードのパック方式を *packing* に設定する。許される値については Table B.8 (p. 155) を、またパック方式については E 章 (p. 178) 参照。

省略時 PACKING 2UPC が仮定される。

4.18 PATH: ディレクトリ構造の設定

書式 次のいずれか:

```
PATH NWP_PATH_S
PATH NWP_PATH_BS
PATH NWP_PATH_M
PATH NWP_PATH_VM
PATH RELATIVE_PATH relpath
PATH NWP_ESF
```

機能 PATH 文は FILENAME 文 (4.8 節, p. 37) とともにデータファイルの位置を設定する。PATH 文の第一引数に対応して、次のようにデータファイルが決められる。

RELATIVE_PATH	引数 *relpath* はスラッシュを含みうる文字列である。データファイルの名前は NUSDAS*nn*/*relpath*/*filename* あるいは FILENAME 文を省略すると NUSDAS*nn*/*relpath* となる。ここで *nn* は NRD 番号であり、*relpath* をスラッシュで区切ったものと *filename* のそれぞれについてパス名置換 (後述) が行われる。MEMBER 文、VALIDTIME 文の設定はパス名に影響しない。
PATH 文省略時	前項に準じて、定義ファイルにあたかも PATH RELATIVE_PATH _model/_attribute/_space/_time/_name/_basetime と書かれているかのようにふるまう。ただし、MEMBER 文に OUT が設定されている場合はこのあとに /_member が追加され、VALIDTIME 文に OUT が設定されている場合はさらにそのあとに /_validtime が追加される。FILENAME 文が存在すれば、さらにこの後に /*filename* が追加される。
NWP_PATH_S	データファイルの名前は NUSDAS*nn*/*3dname*/*validtime* となる。ここで、*3dname* はパス名置換の _3d と _name を連結したもの、*validtime* はパス名置換の _validtime と同じである。FILENAME 文、MEMBER 文、VALIDTIME 文の設定はパス名に影響しない。
NWP_PATH_BS	データファイルの名前は NUSDAS*nn*/*3dname*/*basetime* となる。ここで *basetime* はパス名置換の _basetime と同じである。FILENAME 文、MEMBER 文、VALIDTIME 文の設定はパス名に影響しない。
NWP_PATH_M	データファイルの名前は NUSDAS*nn*/*3dname*/*member*/*validtime* となる。ここで *member* はパス名置換の _member と同じである。FILENAME 文、MEMBER 文、VALIDTIME 文の設定はパス名に影響しない。
NWP_PATH_VM	データファイルの名前は NUSDAS*nn*/*3dname*/*member* となる。FILENAME 文、MEMBER 文、VALIDTIME 文の設定はパス名に影響しない。
NWP_ESF	データファイルの名前は NUSD*type1type2type3* となる。通常は日立 CSES のために実行時オプション IESF 2.3.3 節 (p. 22) とともに使う。
上記に該当しない時	PATH 文省略時と同じ挙動になる。

パス名置換 それぞれのパス要素が次のようなものであるとき、次元値に応じた値に置換される。

`_basetime` 基準時刻を 12 文字の数字列で表わしたもの。たとえば 200703312330 は 2007 年 3 月 31 日 12 時 30 分である。
`_validtime` 対象時刻を 12 文字の数字列で表わしたもの。
`_member` メンバ名。
`_model` モデル名 (種別 1 の前半 4 文字)。
`_space` 空間種別名 (種別 1 の後半 4 文字)。
`_2d` 2 次元座標名 (空間種別名の前半 2 文字)。
`_3d` 3 次元座標名 (空間種別名の後半 2 文字)。
`_attribute` 属性名 (種別 2 の前半 2 文字)。
`_time` 時間種別名 (種別 2 の後半 2 文字)。
`_name` 種別 3。

空白を含む名前が用いられたときは、ファイル名を生成するまえに空白を下線 ('_') に置換する。

互換性 `PATH NWP_ESF` は NAPS8, NAPS9 の数値予報ルーチンでは使えません [NuSDaS 1.1 と NuSDaS 1.3 の r4374 (2014-12-11) 以降及び NuSDaS 1.4 でだけサポートされています。]

4.19 PLANE: 面数の設定

書式

`PLANE` N_Z

位置制約 ELEMENTMAP 文 (4.7.1 節, p. 36), PLANE1 文 (4.20 節, p. 41) や PLANE2 文 (4.21 節, p. 41) より先に書かねばならない。

機能 PLANE 文は面の数を設定する。

必須性 PLANE 文を省略してはならない。

4.20 PLANE1: 面 1 の名前を設定

書式

`PLANE1` *name* ...

位置制約 PLANE 文 (4.19 節, p. 41) より後に書かねばならない。

機能 PLANE1 文は N_Z 個の面 1 の名前 *name* のリストを設定する。名前は 6 文字以下の英数字または下線で、表 B.6 に従う。ライブラリは名前を検査しないが、その他の任意の名前は開発用途の一時的使用に限定される。

必須性 PLANE1 文を省略してはならない。

4.21 PLANE2: 面 2 の名前を設定

書式

`PLANE2` *name* ...

機能 PLANE2 文は N_Z 個の面 2 の名前 *name* のリストを設定する。その他の事項は PLANE1 文と共通である。面を範囲で設定するデータセットでだけ用いられるため、結局のところ実用例は皆無である。

省略時 面 2 のリストには面 1 と同じ名前が登録される。

4.22 SIZE: 格子数を設定

書式

SIZE N_X N_Y

機能 SIZE 文はデータレコード 1 つに含まれる格子の数 N_X, N_Y を設定する。

必須性 SIZE 文を省略してはならない。

4.23 STANDARD: 地図投影法パラメタ設定

書式

STANDARD λ_1E φ_1N λ_1E φ_2N

機能 STANDARD 文は地図投影法パラメタを設定する。概ね φ_1 と φ_2 が標準緯度、概ね λ_1 が標準経度と呼ばれるが、その具体的な意味については付録 C を参照されたい。

省略時 STANDARD 0E ON 0E ON が仮定される。それが適切か否かは 2 次元座標系しだいである。

4.24 SUBCNTL: SUBC レコードの登録

書式

SUBCNTL N_s [*name nbytes*] ...

機能 SUBCNTL 文はデータファイルが持つべき SUBC レコードについて設定する。INFORMATION 文 (4.10 節, p. 38) と異なり、1 つの定義ファイルには SUBCNTL 文は 1 つだけ記述し、そこに N_s 個の SUBC レコードすべてを記述する。

名前 *name* は 4 文字以下の名前で、使える名前は ETAΔ, SIGM, ZHYB, RGAU, RADR, RADS, DPRD, ISPC, DELT, TDIF, LOCA のいずれかである。

長さ *nbytes* は SUBC のペイロードつまり Table A.6 (p. 143) の項番 6 の長さであるが、RADR, RADS, DPRD, ISPC, TDIF, LOCA (いいかえると preset できないもの) についてはメンバ数 N_M (もしあれば) と対象時刻数 N_V だけ繰り返されるので本当のレコード長は次式で与えられる:

$$(\text{項番 6 のバイト数}) = N_M \times N_V \times nbytes.$$

省略時 登録されていない SUBC レコードは書き出せないので、データファイルに SUBC レコードを作ることはできなくなる。

4.25 TYPE1: 種別 1 の設定

書式

TYPE1 *model 2d 3d*

機能 TYPE1 文は種別 1 を *model*, *2d*, *3d* を連結した文字列に設定する。引数 *model*, *2d*, *3d* はそれぞれモデル名 [Table B.1 (p. 152)], 2 次元座標名 [Table B.2 (p. 153)], 3 次元座標名 [Table B.3 (p. 153)], である。

必須性 TYPE1 文を省略してはならない。そのような定義ファイルが存在しても参照する方法がない。

4.26 TYPE2: 種別 2 の設定

書式

`TYPE2` *attribute time*

機能 TYPE2 文は種別 2 を *attribute* と *time* を連結した文字列に設定する。引数 *attribute* と *time* はそれぞれ属性名 [Table B.4 (p. 153)] と時間種別名 [Table B.5 (p. 154)] である。

必須性 TYPE2 文を省略してはならない。そのような定義ファイルが存在しても参照する方法がない。

4.27 TYPE3: 種別 3 の設定

書式

`TYPE3` *name*

機能 TYPE3 文は種別 3 を *name* に設定する。種別 3 は 4 文字の英数字または下線である。末尾には空白があってもよい、つまり 3 文字の種別 3 を設定することができるが、混乱の元なので推奨しない。

必須性 TYPE3 文を省略してはならない。そのような定義ファイルが存在しても参照する方法がない。

4.28 VALIDTIME: 対象時刻の数を設定

書式 次のいずれか:

`VALIDTIME` N_V *tunits* `IN`
`VALIDTIME` N_V *tunits* `OUT`
`VALIDTIME` N_V `IN` *tunits*
`VALIDTIME` N_V `OUT` *tunits*

位置制約 ELEMENTMAP 文 (4.7.1 節, p. 36), VALIDTIME1 文 (4.29 節, p. 43), VALIDTIME2 文 (4.30 節, p. 44) (あれば) より先に記述しなければならない。

機能 VALIDTIME 文は対象時刻の数を設定する。引数 `IN` は異なる対象時刻のデータがひとつのデータファイルに書かれることを意味し、引数 `OUT` は異なる対象時刻のデータは別のデータファイルに書かれることを意味する。時間の単位 *tunits* は予報時間の単位で、次のいずれかである; `MIN`: 分, `HOUR`: 時, `DAY`: 日, `WEEK`: 週, `PENT`: 暦日半旬, `JUN`: 旬, `MONT`: 月.

必須性 VALIDTIME 文を省略してはならない。

4.29 VALIDTIME1: 予報時間リストを設定

書式 次のいずれか:

`VALIDTIME1 ARITHMETIC` *first step*
`VALIDTIME1 ALL_LIST` *ft* ...

位置制約 VALIDTIME 文 (4.28 節, p. 43) より後に記述しなければならない。

機能　VALIDTIME1 文は予報時間のリストを設定する。キーワード `ARITHMETIC` が用いられた場合は初項 *first*, 交差 *step* の等差数列となる。キーワード `ALL_LIST` が用いられた場合は N_V 個の予報時間のリストを記述する。予報時間は整数でなければならないが、負値でもよい。

必須性　VALIDTIME1 文を省略してはならない。

4.30　VALIDTIME2: 範囲の予報時間を設定

書式　次のいずれか

`VALIDTIME2` -*ft*
`VALIDTIME2` *ft* ...

機能　VALIDTIME2 文は対象時刻を範囲で設定するデータセットのための機能で、実例が皆無である。最初の数値が負値　$-$ *ft* であれば、N_V 個の予報時間 2 はすべて *ft* (符号反転値) である。そうでなければあたかも VALIDTIME1 文の ALL_LIST 設定時のように N_V 個の予報時間 2 の一覧を設定する。

省略時　`VALIDTIME2 -1` が設定されたかのようにすべて 1 が仮定される。

4.31　VALUE: 格子の空間代表性を設定

書式　次のいずれか:

```
VALUE PVAL
VALUE MEAN
VALUE REPR
```

機能　VALUE 文はデータが格子点近傍の場をどのように代表しているかを設定する。許される値については Table B.10 (p. 155) を参照。

省略時　`VALUE PVAL` が仮定される。

5 Fortran リファレンスマニュアル

5.1 凡例

- 引数 *fmt* または *utype* (配列の型) に与えるべき定数名は、Table B.7 (p. 154) 参照。

5.2 最低限知るべきサブルーチン

5.2.1 NUSDAS_READ: データ記録の読取

書式

CALL **NUSDAS_READ**(*utype1*, *utype2*, *utype3*, *basetime*, *member*, *validtime*, *plane*, *element*, *data*, *fmt*, *size*, *result*)

引数名	引数の型	配列長	I/O	役割
utype1	CHARACTER(8)		IN	種別 1
utype2	CHARACTER(4)		IN	種別 2
utype3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面の名前
element	CHARACTER(6)		IN	要素名
data	任意	可変	OUT	結果格納配列
fmt	CHARACTER(2)		IN	結果格納配列の型
size	INTEGER(4)		IN	結果格納配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　引数で指定した TYPE, 基準時刻、メンバー、対象時刻、面、要素のデータを読み出す。

終了コード

正 読み出して格納した格子数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

-5 格納配列の型とレコードの記録形式が不整合

注意　nusdas_read では、返却値 0 はエラーであることに注意が必要。nusdas_read のエラーチェックは返却値が求めている格子数と一致していることを確認するのが望ましい。

互換性　NuSDaS1.1 では「ランレングス圧縮で、データが指定最大値を超えている」(返却値-6) が定義されていたが、はデータの最初だけを見ているだけで意味がないと思われるので、 NuSDaS 1.3 からはこのエラーは返さない。また、「ユーザーオープンファイルの管理部又はアドレス部が不正である」(返却値-7) は、共通部分の-54〜-57 に対応するので、このエラーは返さない

5.2.2 NUSDAS_WRITE: データ記録の書出

書式

CALL **NUSDAS_WRITE**(*utype1*, *utype2*, *utype3*, *basetime*, *member*, *validtime*, *plane*, *element*, *data*, *fmt*, *nelems*, *result*)

引数名	引数の型	配列長	I/O	役割
utype1	CHARACTER(8)		IN	種別 1
utype2	CHARACTER(4)		IN	種別 2
utype3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面の名前
element	CHARACTER(6)		IN	要素名
data	任意	可変	IN	データ配列
fmt	CHARACTER(2)		IN	データ配列の型
nelems	INTEGER(4)		IN	データ配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　データレコードを指定された場所に書き出す。

終了コード

正 実際に書き出された要素数

-2 メンバー名、面名、要素名が間違っている

-2 このレコードは ELEMENTMAP によって書き出しが禁止されている

-3 与えられたデータ要素数 *nelems* が必要より小さい

-4 指定データセットにはデータ配列の型 *fmt* は書き出せない

-5 データレコード長が固定レコード長を超える

-6 データセットの欠損値指定方式と RLEN 圧縮は併用できない

-7 マスクビットの設定がされていない

-8 エンコード過程でのエラー (数値が過大または RLEN 圧縮エラー)

注意

- データセットの指定と異なる大きさのレコードを書き出すにはあらかじめ nusdas_parameter_change(p. 56) を使って設定を変えておく。
- 格子数 (データセットの指定または nusdas_parameter_change(p. 56) 設定) より大きい要素数 *nelems* を指定するとエラーにはならず、余った要素が書き出されない結果となるので注意されたい。

履歴　この関数は NuSDaS 1.0 から存在した。

5.2.3 NUSDAS_IOCNTL: 入出力フラグ設定

書式

CALL **NUSDAS_IOCNTL**(*param*, *value*, *result*)

引数名	引数の型	配列長	I/O	役割
param	INTEGER(4)		IN	設定項目コード
value	INTEGER(4)		IN	設定値
result	INTEGER(4)		OUT	終了コード

説明　入出力にかかわるフラグを設定する.

N_IO_MARK_END 既定値 1. 零にすると nusdas_write(p. 47) などの出力関数を呼び出すたびにデータファイルへの出力を完結させ END 記録を書き出すのをやめる.

N_IO_W_FCLOSE 既定値 1. 零にすると nusdas_write(p. 47) などの出力関数を呼び出すたびに書き込み用に開いたファイルを閉じるのをやめる. 速度上有利だが、データファイルの操作が終了した後でファイルを閉じる関数 nusdas_allfile_close(p. 49) または nusdas_onefile_close(p. 55) を適切に呼んでファイルを閉じないと出力ファイルが不完全となり、後で読むことができない. なお、このフラグを変更すると N_IO_MARK_END も連動する.

N_IO_R_FCLOSE 既定値 1. 零にすると nusdas_read(p. 47) などの入力関数を呼び出すたびに読み込み用に開いたファイルを閉じるのをやめる. 速度上有利だが、多数のファイルから入力するプログラムではファイルハンドルが枯渇する懸念があるのでファイルを明示的に閉じることが推奨される.

N_IO_WARNING_OUT 既定値 1. 0 にするとエラーメッセージだけが出力される. 1 にすると、それに加えて警告メッセージも出力されるようになる. 2 にすると、それに加えてデバッグメッセージも出力されるようになる.

N_IO_BADGRID 既定値 0. 1 にすると投影法パラメタの検査で不適切な値が検出されてもデータファイルが作成できるようになる。

終了コード

0 正常終了

-1 サポートされていないパラメタである

履歴 この関数は NuSDaS 1.0 から存在した. **N_IO_WARNING_OUT** の値 2 は NuSDaS 1.3 からの拡張である. **N_IO_BADGRID** も NuSDaS 1.3 からの拡張である.

5.2.4 NUSDAS_ALLFILE_CLOSE: 全てのデータファイルを閉じる

書式

CALL **NUSDAS_ALLFILE_CLOSE**(*param*, *result*)

引数名	引数の型	配列長	I/O	役割
param	INTEGER(4)		IN	閉じるファイルの種類
result	INTEGER(4)		OUT	終了コード

説明 今までに NuSDaS インターフェイスで開いた全てのファイルを閉じる. 引数 param は次のいずれかを用いる:

N_FOPEN_READ 読み込み用に開いたファイルだけを閉じる

N_FOPEN_WRITE 書き込み可で開いたファイルだけを閉じる

N_FOPEN_ALL すべてのファイルを閉じる

終了コード

正 正常に閉じられたファイルの数

0 閉じるべきファイルがなかった

負 閉じる際にエラーが起こったファイルの数

履歴 この関数は NuSDaS 1.0 から存在した.

5.3 データ読書サブルーチン

5.3.1 NUSDAS_CUT: 領域限定のデータ読取

書式

CALL **NUSDAS_CUT**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *udata*, *utype*, *usize*, *ixstart*, *ixfinal*, *iystart*, *iyfinal*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面
element	CHARACTER(6)		IN	要素名
udata	任意	可変	OUT	データ格納先配列
utype	CHARACTER(2)		IN	データ格納先配列の型
usize	INTEGER(4)		IN	データ格納先配列の要素数
ixstart	INTEGER(4)		IN	x 方向格子番号下限
ixfinal	INTEGER(4)		IN	x 方向格子番号上限
iystart	INTEGER(4)		IN	y 方向格子番号下限
iyfinal	INTEGER(4)		IN	y 方向格子番号上限
result	INTEGER(4)		OUT	終了コード

説明 nusdas_read(p. 47) $*$ と同様だが、データレコードのうち格子点 (*ixstart* , *iystart*)–(*ixfinal* , *iyfinal*) だけが *udata* に格納される。

格子番号は 1 から始まるものとするため、*ixstart* や *iystart* は正でなければならず、また *ixfinal* や *iyfinal* はそれぞれ *ixstart* や *iystart* 以上でなければならない。この規則に反する指定を行った場合は、返却値-8 のエラーとなる。なお、*iyfinal*, *jyfinal* の上限が格子数を超えていることのチェックはしていないので注意が必要。

終了コード

正 読み出して格納した格子数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

-5 格納配列の型とレコードの記録形式が不整合

-8 領域指定パラメータが不正

履歴 本関数は NuSDaS 1.1 で導入され、 NuSDaS 1.3 で初めてドキュメントされた。

互換性 NuSDaS 1.1 では、ローカルのデータファイルに対しては、$ixstart \leq 0$ の場合は $ixstart = 1$ に (*jystart* も同様), *ixfinal* が X 方向の格子数を超える場合には、*ixfinal* は X 方向の格子数に (*jyfinal* も同様) に読み替えられていたが、 NuSDaS 1.3 からは返却値-8 のエラーとする。また、pandora データについては、*ixstart*, *ixfinal*, *jystart*, *jyfinal* が非負であることだけがチェックされていた。NuSDaS 1.3 からはデータファイル、pandora とも上述の通りとなる。

5.3.2 NUSDAS_CUT_RAW: 領域限定の DATA 記録直接読取

書式

CALL **NUSDAS_CUT_RAW**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *udata*, *usize*, *ixstart*, *ixfinal*, *iystart*, *iyfinal*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面
element	CHARACTER(6)		IN	要素名
udata	任意	可変	OUT	データ格納先配列
usize	INTEGER(4)		IN	データ格納先配列のバイト数
ixstart	INTEGER(4)		IN	x 方向格子番号下限
ixfinal	INTEGER(4)		IN	x 方向格子番号上限
iystart	INTEGER(4)		IN	y 方向格子番号下限
iyfinal	INTEGER(4)		IN	y 方向格子番号上限
result	INTEGER(4)		OUT	終了コード

説明　nusdas_read2_raw(p. 51) と類似だが、データレコードのうち格子点 (*ixstart* , *iystart*)–(*ixfinal* , *iyfinal*) に対応する部分だけが *udata* に格納される。ただしパッキングが 2UPP, 2UPJ, RLEN の場合、または欠損値が MASK の場合は全体が格納される。

終了コード

正 読み出して格納したバイト数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

履歴　この関数は NuSDaS 1.1 で導入された。エラーコード -4 は NuSDaS 1.3 で新設されたもので、それ以前はエラーチェックがなされていなかった。

5.3.3 NUSDAS_READ2_RAW: DATA 記録内容の直接読取

書式

CALL **NUSDAS_READ2_RAW**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *buf*, *buf_nbytes*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1
validtime2	INTEGER(4)		IN	対象時刻 2
plane1	CHARACTER(6)		IN	面 1
plane2	CHARACTER(6)		IN	面 2
element	CHARACTER(6)		IN	要素名
buf	任意	可変	OUT	データ格納配列
buf_nbytes	INTEGER(4)		IN	データ格納配列のバイト数
result	INTEGER(4)		OUT	終了コード

説明　引数で指定した TYPE, 基準時刻、メンバー、対象時刻、面、要素のデータをファイルに格納されたままの形式で読み出す。データは、DATA レコードのフォーマット表の項番 10〜14 までのデータが格納される。

終了コード

正 読み出して格納したバイト数。

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

履歴　この関数は NuSDaS1.1 で導入された。

5.3.4 NUSDAS_READ_3D: 高次元読み込み

書式

CALL **NUSDAS_READ_3D**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *nrecs*, *udata*, *utype*, *usize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻
member	CHARACTER(4)	可変	IN	メンバ名の配列
validtime	INTEGER(4)	可変	IN	対象時刻の配列
plane	CHARACTER(6)	可変	IN	面名の配列
element	CHARACTER(6)	可変	IN	要素名の配列
nrecs	INTEGER(4)		IN	レコード数
udata	任意	可変	OUT	結果格納配列
utype	CHARACTER(2)		IN	結果格納配列の型
usize	INTEGER(4)		IN	レコードあたり要素数
result	INTEGER(4)		OUT	終了コード

説明　*member*, *validtime*, *plane*, *element* には *nrecs* 個の大きさを持つ配列を指定する。これらの配列の各要素を指定して nusdas_read(p. 47) を順次呼びだし *udata* に格納する。*udata* の要素数は *nrecs* * *usize* 個以上でなければならない。nusdas_read の返却値（終了コード）が *usize* と一致しなかった場合は読み込みを終了する。

終了コード

正 全てのデータを読むことができた場合は *nrecs* * *usize* を返す。

負 最後に呼び出した nusdas_read(p. 47) の終了コードを返す。

注意 読み込みを途中で終了した場合、どこまで処理したかを知る方法は無い。

5.3.5 NUSDAS_WRITE_3D: 高次元書き出し

書式

CALL **NUSDAS_WRITE_3D**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *nrecs*, *udata*, *utype*, *usize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻
member	CHARACTER(4)	可変	IN	メンバ名の配列
validtime	INTEGER(4)	可変	IN	対象時刻の配列
plane	CHARACTER(6)	可変	IN	面名の配列
element	CHARACTER(6)	可変	IN	要素名の配列
nrecs	INTEGER(4)		IN	レコード数
udata	任意	可変	IN	データ配列
utype	CHARACTER(2)		IN	データ配列の型
usize	INTEGER(4)		IN	レコードあたり要素数
result	INTEGER(4)		OUT	終了コード

説明 *member*, *validtime*, *plane*, *element* には *nrecs* 個の大きさを持つ配列を指定する。これらの配列の各要素を指定して nusdas_write(p. 47) を順次呼びだす。データ配列の要素数は *nrecs* * *usize* 個でなければならない。nusdas_write の返却値（終了コード）が *usize* と一致しなかった場合は書き出しを終了する。

終了コード

正 全てのデータを書き出すことができた場合は *nrecs* * *usize* を返す。

負 最後に呼び出した nusdas_write(p. 47) の終了コードを返す。

注意 読み込みを途中で終了した場合、どこまで処理したかを知る方法は無い。

5.4 動作制御用サブルーチン

5.4.1 NUSDAS_ESF_FLUSH: NAPS7 型 ES ファイルの出力完了

書式

CALL **NUSDAS_ESF_FLUSH**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻
result	INTEGER(4)		OUT	終了コード

説明

履歴 nusdas_esf_flush(p. 54) は NuSDaS 1.0 から存在する。

バグ NuSDaS 1.3 からは ES をサポートしていないため、この関数はダミーである。

5.4.2 NUSDAS_MAKE_MASK: マスクビット配列の作成

書式

CALL **NUSDAS_MAKE_MASK**(*udata*, *utype*, *usize*, *output*, *mb_bytes*, *result*)

引数名	引数の型	配列長	I/O	役割
udata	任意	可変	IN	格子データ
utype	CHARACTER(2)		IN	格子データの型
usize	INTEGER(4)		IN	格子データの要素数
output	任意	可変	OUT	マスクビット配列
mb_bytes	INTEGER(4)		IN	マスクビット配列のバイト数
result	INTEGER(4)		OUT	終了コード

説明 配列 *udata* の内容をチェックしてマスクビット列を作成し *output* に書き込む。引数 *utype* と欠損値は配列の型に応じて次のように指定する。

1 バイト整数型 引数 *utype* に N_I1 を指定する。配列中の欠損扱いしたい要素に N_MV_UI1 を設定しておく。

2 バイト整数型 引数 *utype* に N_I2 を指定する。配列中の欠損扱いしたい要素に N_MV_SI2 を設定しておく。

4 バイト整数型 引数 *utype* に N_I4 を指定する。配列中の欠損扱いしたい要素に N_MV_SI4 を設定しておく。

4 バイト実数型 引数 *utype* に N_R4 を指定する。配列中の欠損扱いしたい要素に N_MV_R4 を設定しておく。

8 バイト実数型 引数 *utype* に N_R8 を指定する。配列中の欠損扱いしたい要素に N_MV_R8 を設定しておく。

終了コード

0 正常終了

-1 配列長 *mb_bytes* が不足している

-5 未知の型名 *utype* が与えられた

サイズ要件　*mb_bytes* は少なくとも (*usize* + 7) / 8 バイト以上必要である。

履歴　nusdas_make_mask(p. 54) は NuSDaS 1.0 から存在する。

5.4.3 NUSDAS_SET_MASK: 改善型マスクビット設定関数

書式

CALL **NUSDAS_SET_MASK**(*type1*, *type2*, *type3*, *udata*, *utype*, *usize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
udata	任意	可変	IN	データ配列
utype	CHARACTER(2)		IN	データ配列の型
usize	INTEGER(4)		IN	配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　配列 *udata* の内容に従って nusdas_make_mask(p. 54) と同様にマスクビット列を作成し指定した種別のデータセットに対して設定する。

終了コード

0 正常終了

-5 未知の型名 *utype* が与えられた

注意　本関数によるマスクビットの設定は nusdas_parameter_change(p. 56) に優先するが、他のデータセットには効果をもたない。

履歴　本関数は NuSDaS 1.3 で新設された。

5.4.4 NUSDAS_ONEFILE_CLOSE: 指定データファイルを閉じる

書式

CALL **NUSDAS_ONEFILE_CLOSE**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻
result	INTEGER(4)		OUT	終了コード

終了コード

0 正常終了

1 ファイルクローズ前の書き込み時に定義ファイルを読み込めなかった

-1 ファイルクローズ前の書き込み時に IO エラーが発生

説明

履歴　この関数は NuSDaS 1.0 から存在した.

5.4.5 NUSDAS_PARAMETER_CHANGE: オプション設定

書式

CALL **NUSDAS_PARAMETER_CHANGE**(*param*, *value*, *result*)

引数名	引数の型	配列長	I/O	役割
param	INTEGER(4)		IN	設定項目コード
value	任意	可変	IN	設定値
result	INTEGER(4)		OUT	終了コード

説明 *param* で指定されるパラメータに値 *value* を設定する。整数値の項目については、互換性のため値ゼロのかわりに名前 N_OFF を用いることができる。

N_PC_MISSING_UI1 1 バイト整数の欠損値 (既定値: N_MV_UI1)

N_PC_MISSING_SI2 2 バイト整数の欠損値 (既定値: N_MV_SI2)

N_PC_MISSING_SI4 4 バイト整数の欠損値 (既定値: N_MV_SI4)

N_PC_MISSING_R4 4 バイト実数の欠損値 (既定値: N_MV_R4)

N_PC_MISSING_R8 8 バイト実数の欠損値 (既定値: N_MV_R8)

N_PC_MASK_BIT マスクビット配列へのポインタ (既定値は NULL ポインタだが Fortran では直接設定できないので nusdas_parameter_reset(p. 57) を用いられたい)

N_PC_SIZEX 非零値を設定すると強制的にデータレコードの x 方向格子数を設定する (0)

N_PC_SIZEY 強制格子サイズ: 既定値 (0) 以外を設定するとデータレコードの y 方向格子数を設定する

N_PC_PACKING 4 文字のパッキング名を設定すると、定義ファイルの指定にかかわらず nusdas_write(p. 47) 等データ記録書き込みの際に用いられるパッキング方式が変更される。既定値に戻す (定義ファイルどおりに書かせる) には 4 バイト整数値 0 を設定する。

N_PC_ID_SET NRD 番号制約: 既定値 (-1) 以外を設定すると、その番号の NRD だけを入出力に用いるようになる

N_PC_WBUFFER 書き込みバッファサイズ (既定値: 0) 実行時オプション FWBF に同じ。

N_PC_RBUFFER 読み取りバッファサイズ (既定値: 17) 実行時オプション FRBF に同じ。

N_PC_KEEP_CFILE ファイルを閉じたあと CNTL/INDX などのヘッダ情報をキャッシュしておく数。負にするとキャッシュを開放しなくなる（既定値: -1)。実行時オプション GKCF に同じ。

N_PC_OPTIONS 設定のみでリセットはできない。ヌル終端した文字列を与えると実行時オプションとして設定する。Fortran インターフェイスでもヌル終端しなければならないことに注意。

終了コード

0 正常終了

-1 サポートされていないパラメタである

履歴 NuSDaS 1.0 から存在する。

NuSDaS 1.1 ではデータセット探索のキャッシュ論理に問題があり、N_PC_ID_SET で NRD 番号制約をかけて入出力を行った後で NRD 番号制約を解除して同じ種別にアクセスしても探索が行われない (あらかじめ NRD 制約をかけずに入出力操作をしていれば探索される)。この問題は NuSDaS 1.3 以降では解決されている。

Fortran 版の仕様変更 NuSDaS 1.2 以前では C API と同様、引数 *value* に nusdas_fort.h で定義される変数 NULL を与えるとパラメタを既定値に戻すことができたが、この機能は可搬性の問題から廃止された。上述の既定値を明示的に設定するか、nusdas_parameter_reset() を利用されたい。

5.4.6 NUSDAS_INQ_PARAMETER: オプション取得

書式

CALL **NUSDAS_INQ_PARAMETER**(*param*, *value*, *result*)

引数名	引数の型	配列長	I/O	役割
param	INTEGER(4)		IN	設定項目コード
value	任意	可変	OUT	設定値
result	INTEGER(4)		OUT	終了コード

説明 nusdas_parameter_change(p. 56) の項目 *param* で設定されるパラメータの値を *value* の指す領域 (型は以下を参照) に書き込む。

N_PC_MISSING_UI1 1 バイト整数の欠損値

N_PC_MISSING_SI2 2 バイト整数の欠損値

N_PC_MISSING_SI4 4 バイト整数の欠損値

N_PC_MISSING_R4 4 バイト実数の欠損値

N_PC_MISSING_R8 8 バイト実数の欠損値

N_PC_SIZEX 4 バイト整数に x 方向強制格子サイズを与える

N_PC_SIZEY 4 バイト整数に y 方向強制格子サイズを与える

N_PC_MASK_BIT マスクビット配列を返す。この問合せは設定値が nusdas_make_mask(p. 54) で作られた場合にしか機能しない。

N_PC_PACKING 4 バイトの文字列に強制パック方式名を与える。設定されていない場合は 4 文字のスペースが書き込まれる。

N_PC_ID_SET NRD 番号制約がかかっている場合その値、かかっていない場合 -1 を与える。

N_PC_WBUFFER 4 バイト整数に書き込みバッファサイズ (実行時オプション FWBF) を与える。

N_PC_RBUFFER 4 バイト整数に読み取りバッファサイズ (実行時オプション FRBF) を与える。

終了コード

0 正常終了

-1 サポートされていないパラメタである

-2 マスクビット配列は設定されていない

-3 マスクビット配列は設定されているが長さがわからない

履歴 NuSDaS 1.3 で導入された。

5.4.7 NUSDAS_PARAMETER_RESET: オプションを既定値に戻す

書式

CALL **NUSDAS_PARAMETER_RESET**(*param*, *result*)

引数名	引数の型	配列長	I/O	役割
param	INTEGER(4)		IN	設定項目コード
result	INTEGER(4)		OUT	終了コード

説明 nusdas_parameter_change(p. 56) で設定されたパラメタを既定値に戻します。

履歴 この関数 は NuSDaS 1.3 で導入されました。それ以前のバージョンでは nusdas_parameter_change(p. 56) に既定値または定数 NULL を与える方法が使われていました。

5.5 問合せサブルーチン

5.5.1 NUSDAS_GRID: 格子情報へのアクセス

書式

CALL **NUSDAS_GRID**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *proj*, *gridsize*, *gridinfo*, *value*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
proj	CHARACTER(4)		I/O	投影法 3 字略号
gridsize	INTEGER(4)	2	I/O	格子数
gridinfo	REAL(4)	14	I/O	投影法緒元
value	CHARACTER(4)		I/O	格子点値が周囲の場を代表する方法
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　この API は、CNTL レコードに格納された格子情報 (つまり定義ファイルに書かれた格子情報) を返す。nusdas_parameter_change を使って、定義ファイルに書いた格子数から変更した場合には正しい情報が得られない。このような場合は nusdas_inq_data を使う。

gridinfo には 4 バイト単精度浮動小数点型の配列で 14 要素存在するものを指定する。

これは CNTL レコードの項番 15 〜 21 に対応する。順に基準点 X 座標、基準点 Y 座標、基準点緯度、基準点経度、X 方向格子間隔、Y 方向格子間隔、標準緯度、標準経度、第 2 標準緯度、第 2 標準経度、緯度 1、経度 1、緯度 2、経度 2 となる。

value の値については Table B.10 (p. 155) を参照。

終了コード

0 正常

-5 入出力指示が不正

履歴　この関数は NuSDaS 1.0 から実装されていた。

5.5.2 NUSDAS_INFO: INFO 記録へのアクセス

書式

CALL **NUSDAS_INFO**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *group*, *info*, *bytesize*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
group	CHARACTER(4)		IN	群名
info	CHARACTER	可変	I/O	INFO 記録内容
bytesize	INTEGER(4)		IN	INFO 記録のバイト数
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明

終了コード

非負 書き出した INFO のバイト数

-3 バッファが不足している

-5 入出力指示が不正

注意 NuSDaS1.1 では、バッファが不足している場合でもバッファの大きさの分だけを書き込み、そのサイズを返していたが、 NuSDaS 1.3 からはこのような場合は-3 が返る。また、INFO のサイズは NuSDaS 1.3 で新設された nusdas_inq_subcinfo で問い合わせ項目を N_INFO_NUM にすれば得ることができる。

5.5.3 NUSDAS_INQ_CNTL: データファイルの諸元問合せ

書式

CALL **NUSDAS_INQ_CNTL**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *param*, *data*, *datasize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
param	INTEGER(4)		IN	問合せ項目コード
data	任意	可変	OUT	問合せ結果配列
datasize	INTEGER(4)		IN	問合せ結果配列の要素数
result	INTEGER(4)		OUT	終了コード

説明 引数 *type1* から *validtime* で指定されるデータファイルに書かれた CNTL 記録について、引数 *param* で指定される問合せを行う。

N_MEMBER_NUM メンバーの個数が 4 バイト整数型変数 *data* に書かれる。

N_MEMBER_LIST データファイルに定義されたメンバー名が配列 *data* に書かれる。配列 *data* は長さ 4 文字の文字型で *N_MEMBER_NUM* 要素存在しなければならない。

N_VALIDTIME_NUM validtime の個数が 4 バイト整数型変数 *data* に書かれる。

N_VALIDTIME_LIST データファイルに定義された validtime が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_LIST2 データファイルに定義された validtime2 が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_PLANE_NUM 面の個数が 4 バイト整数型変数 *data* に書かれる。

N_PLANE_LIST データファイルに定義された面の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_PLANE_LIST2 N_PLANE_LIST と全く同じ動作である。

N_ELEMENT_NUM 要素の個数が 4 バイト整数型変数 *data* に書かれる。

N_ELEMENT_LIST データファイルに定義された要素の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_ELEMENT_NUM* 要素存在しなければならない。

N_NUSD_NBYTES NUSD レコードのサイズ (単位バイト) が 4 バイト整数型変数 *data* に書かれる。(先頭・末尾に付加されるレコード長の大きさ (4*2 バイト) を含む)

N_NUSD_CONTENT NUSD レコードの内容を配列 *data* に格納する。配列 *data* は N_NUSD_NBYTES バイト存在しなくてはならない。(先頭・末尾に付加されるレコード長を含む)

N_CNTL_NBYTES CNTL レコードのサイズ (単位バイト) が 4 バイト整数型変数 *data* に書かれる。(先頭・末尾に付加されるレコード長の大きさ (4*2 バイト) を含む)

N_CNTL_CONTENT CNTL レコードの内容を配列 *data* に格納する。配列 *data* は N_CNTL_NBYTES バイト存在しなくてはならない。(先頭・末尾に付加されるレコード長を含む)

N_PROJECTION 地図投影法の情報を 4 文字の文字型 *data* に格納する (記号の意味は巻末の表参照)。

N_GRID_SIZE X 方向、Y 方向の格子数がこの順序で 4 バイト整数型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_GRID_BASEPOINT 基準点の x 座標、y 座標、緯度、経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_GRID_DISTANCE X 方向、Y 方向の格子間隔がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_STAND_LATLON 標準緯度、標準経度、第 2 標準緯度、第 2 標準経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_SPARE_LATLON 緯度 1、経度 1、緯度 2、経度 2 がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_INDX_SIZE INDX の個数が 4 バイト整数型の変数 *data* に書かれる。(この問い合わせは NuSDaS 1.3 で追加)

N_ELEMENT_MAP データの格納が許容されているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。(この問い合わせは NuSDaS 1.3 で追加)

N_DATA_MAP データが書き込まれているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。(この問い合わせは NuSDaS 1.3 で追加)

終了コード

正 格納要素数

-1 データの配列数が不足している。

-2 データの配列が確保されていない。

-3 問い合わせ項目が不正

注意　NuSDaS1.1 以前では、同じ構造のデータセットでも N_VALIDTIME_NUM, N_VALIDTIME_LIST の問い合わせ結果が 1 つの basetime に複数の validtime を格納するか否かによって異なっていた。これは、validtime でファイルを分ける (異なる validtime のファイルが異なる) 設定ならばデータファイルには 1 つの validtime だけが書かれていたからである。しかし NuSDaS 1.3 からは定義ファイルに指定されたすべての validtime が各データファイルの validtime に格納されているので、問い合わせ結果は格納形態を問わず一定である。

5.5.4 NUSDAS_INQ_DATA: データ記録の諸元問合せ

書式

CALL **NUSDAS_INQ_DATA**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *param*, *data*, *nelems*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面
element	CHARACTER(6)		IN	要素名
param	INTEGER(4)		IN	問合せ項目コード
data	任意	可変	OUT	結果格納配列
nelems	INTEGER(4)		IN	結果格納配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　引数 *type1* から *element* までで指定されるデータ記録について引数 *query* で指定される問合せを行う。

N_DATA_QUADRUPLET 16 バイトのメモリ領域を引数に取り、 N_GRID_SIZE から N_MISSING_VALUE までの情報が返される。

N_GRID_SIZE 引数 *data* に 4 バイト整数の長さ 2 の配列を取り、そこに X, Y 方向の格子数が書かれる。

N_PC_PACKING 引数 *data* に 4 バイトの文字列を取り、そこにパック方式名称が書かれる。文字列はヌル終端されないことに注意。

N_MISSING_MODE 引数 *data* に 4 バイトの文字列を取り、そこに欠損値表現方式名が書かれる。文字列はヌル終端されないことに注意。

N_MISSING_VALUE 引数には上述 N_PC_PACKING 項目によって決まる型の変数を取り、そこにデータ記録上の欠損値が書かれる。この値は nusdas_read(p. 47) で得られる配列で用いられる欠損値とは異なることに注意。

N_DATA_EXIST 引数 *data* に 4 バイト整数型変数をとり、そこにデータの存在有無を示す値が書かれる。0 はデータの不在、1 は存在を示す。

N_DATA_NBYTES 引数 *data* に 4 バイト整数型変数をとり、そこにデータ記録のバイト数が書かれる。

N_DATA_CONTENT 引数 *data* が指すバイト列にデータ記録がそのまま書かれる。

N_RECORD_TIME 引数 *data* に 4 バイト整数型変数をとり、そこにデータ記録の作成時刻が書かれる。この問合せはデータ記録の更新確認用に用意されており、結果は大小比較だけに用いるべきもので日時等を算出すべきではない。この値は time システムコールの返す値の下位 32 ビットであり、2038 年問題の対策のためいずれ機種依存の意味を持つようになるものと思われる。

終了コード

正 格納要素数

-1 データの配列数が不足している

-2 データの配列が確保されていない

-3 問い合わせ項目が不正

履歴　この関数は pnusdas では実装はされていたが、ドキュメント化されていなかった。

5.5.5 NUSDAS_INQ_DEF: データセットの諸元問合せ

書式

CALL **NUSDAS_INQ_DEF**(*type1*, *type2*, *type3*, *param*, *data*, *datasize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
param	INTEGER(4)		IN	問合せ項目コード
data	任意	可変	OUT	結果格納配列
datasize	INTEGER(4)		IN	結果格納配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　引数 *type1* から *type3* で指定されるデータセットの定義ファイルに書かれた内容について、引数 *param* で指定される問合せを行う。

N_MEMBER_NUM 定義ファイルに書かれたメンバーの個数が 4 バイト整数型変数 *data* に書かれる。

N_MEMBER_LIST 定義ファイルに書かれたメンバー名が配列 *data* に書かれる。配列 *data* は長さ 4 文字の文字型で *N_MEMBER_NUM* 要素存在しなければならない。

N_VALIDTIME_NUM 定義ファイルに書かれた validtime の個数が 4 バイト整数型変数 *data* に書かれる。

N_VALIDTIME_LIST 定義ファイルに書かれた validtime が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_LIST2 定義ファイルに書かれた validtime2 が配列 *data* に書かれる。
配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_UNIT 定義ファイルに書かれた validtime の単位が 4 文字の文字型変数 *data* に書かれる。

N_PLANE_NUM 定義ファイルに書かれた面の個数が 4 バイト整数型変数 *data* に書かれる。

N_PLANE_LIST 定義ファイルに書かれた面の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_PLANE_LIST2 定義ファイルに書かれた面 2 の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_ELEMENT_NUM 定義ファイルに書かれた要素の個数が 4 バイト整数型変数 *data* に書かれる。

N_ELEMENT_LIST 定義ファイルに書かれた要素の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_ELEMENT_NUM* 要素存在しなければならない。

N_PROJECTION　定義ファイルに書かれた地図投影法の情報を 4 文字の文字型 *data* に格納する (記号の意味は巻末の表参照)。

N_GRID_SIZE　定義ファイルに書かれた X 方向、Y 方向の格子数がこの順序で 4 バイト整数型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。

N_GRID_BASEPOINT　定義ファイルに書かれた基準点の x 座標、y 座標、緯度、経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_GRID_DISTANCE　定義ファイルに書かれた X 方向、Y 方向の格子間隔がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。

N_STAND_LATLON　定義ファイルに書かれた標準緯度、標準経度、第 2 標準緯度、第 2 標準経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_SPARE_LATLON 定義ファイルに書かれた緯度 1、経度 1、緯度 2、経度 2 がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_INDX_SIZE 定義ファイルから算出される INDX の個数が 4 バイト整数型の変数 *data* に書かれる。(この問い合わせは NuSDaS1.3 で追加)

N_ELEMENT_MAP 定義ファイルでデータの格納が許容されているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。

N_SUBC_NUM 定義ファイルに書かれた SUBC 記録の個数が 4 バイト整数型変数 *buf* に書かれる。

N_SUBC_LIST 定義ファイルに書かれた SUBC 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_SUBC_NUM* 要素存在しなければならない。

N_INFO_NUM 定義ファイルに書かれた INFO 記録の個数が 4 バイト整数型変数 *buf* に書かれる。

N_INFO_LIST 定義ファイルに書かれた INFO 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_INFO_NUM* 要素存在しなければならない。

終了コード

正 格納要素数

-1 格納配列が不足

-2 格納配列が確保されていない

-3 問い合わせが不正

履歴 この関数は NuSDaS1.0 より実装されていたが、NuSDaS1.3 で N_INDX_SIZE の問い合わせ機能が追加されている。

5.5.6 NUSDAS_INQ_NRDBTIME: データセットの基準時刻リスト取得

書式

CALL **NUSDAS_INQ_NRDBTIME**(*type1*, *type2*, *type3*, *btlist*, *btlistsize*, *pflag*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
btlist	INTEGER(4)	可変	OUT	基準時刻が格納される配列
btlistsize	INTEGER(4)		IN	配列の要素数
pflag	INTEGER(4)		IN	動作過程印字フラグ
result	INTEGER(4)		OUT	終了コード

説明 種別 1〜種別 3 で指示されるデータセットに存在する基準時刻を配列 *btlist* に書き込む。引数 *pflag* に非零値を設定すると動作過程の情報を警告メッセージとして印字するようになる。

終了コード

非負 基準時刻の個数

-1 ファイル I/O エラー

-2 ファイルに管理部が存在しない

-3 ファイルのレコード長が不正

-4 ファイルあるいはディレクトリのオープンに失敗

履歴 本関数は NuSDaS 1.0 から存在した。

注意

- 配列長 *btlistsize* より多くの基準時刻が存在する場合は、配列長を越えて書き込むことはない。リターンコードと配列長を比較して、リターンコードが大きかったらその数だけ配列を確保し直して本関数を呼び直すことにより、すべてのリストを得ることができる。
- NuSDaS 1.1 までは見付かったデータセットがネットワークでなければ、それについてだけ探索が行われた。NuSDaS 1.3 からは、指定した種別にマッチするすべてのデータセットについて探索が行われる。
- 種別に対応するデータセットが見つからない場合 (たとえば種別名を間違えた場合)、返却値はゼロとなる。データセットが存在して空の場合と異なり、このとき “Can not find NUSDAS root directory for selected type1-3” “type1<...> type2<...> type3<...> NRD=...” というメッセージが標準エラー出力に表示される。NRD= の後の数値が -1 でなければ、NRD 番号を指定したために存在しているデータセットが見つからなくなっている可能性がある。

5.5.7 NUSDAS_INQ_NRDVTIME: データセットの対象時刻リスト取得

書式

CALL **NUSDAS_INQ_NRDVTIME**(*type1*, *type2*, *type3*, *vtlist*, *vtlistsize*, *basetime*, *pflag*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
vtlist	INTEGER(4)	可変	OUT	対象時刻が書かれる配列
vtlistsize	INTEGER(4)		IN	配列の要素数
basetime	INTEGER(4)		IN	基準時刻 (通算分)
pflag	INTEGER(4)		IN	動作詳細印字フラグ
result	INTEGER(4)		OUT	終了コード

説明　種別 1〜種別 3 で指示されるデータセットに基準時刻 *basetime* のもとで存在する対象時刻を配列 *vtlist* に書き込む。引数 *pflag* に非零値を設定すると動作過程の情報を警告メッセージとして印字するようになる。

終了コード

非負 対象時刻の個数

履歴　本関数は NuSDaS 1.0 から存在したがドキュメントされていなかった。

注意

- 配列長 *vtlistsize* より多くの対象時刻が存在する場合は、配列長を越えて書き込むことはない。リターンコードと配列長を比較して、リターンコードが大きかったらその数だけ配列を確保し直して本関数を呼び直すことにより、すべてのリストを得ることができる。
- 対象時刻の探索はファイルの有無または CNTL レコードによる。リスト中の対象時刻についてデータレコードが書かれていない場合もありうる。
- 基準時刻 *basetime* に -1 を指定すると、基準時刻を問わない検索になる。
- 検索にあたってメンバー名は問わない。
- NuSDaS 1.1 までは見付かったデータセットがネットワークでなければ、それについてだけ探索が行われた。NuSDaS 1.3 からは、指定した種別にマッチするすべてのデータセットについて探索が行われる。
- 種別に対応するデータセットが見つからない場合 (たとえば種別名を間違えた場合)、返却値はゼロとなる。データセットが存在して空の場合と異なり、このとき “Can not find NUSDAS root directory for selected type1-3” “type1<...> type2<...> type3<...> NRD=...” というメッセージが標準エラー出力に表示される。NRD= の後の数値が -1 でなければ、NRD 番号を指定したために存在しているデータセットが見つからなくなっている可能性がある。

5.5.8 NUSDAS_INQ_SUBCINFO: SUBC/INFO の問合せ

書式

CALL **NUSDAS_INQ_SUBCINFO**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *query*, *group*, *buf*, *bufnelems*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻
member	CHARACTER(4)		IN	メンバー
validtime	INTEGER(4)		IN	対象時刻
query	INTEGER(4)		IN	問合せ項目
group	CHARACTER(4)		IN	群名
buf	任意	可変	OUT	結果格納配列
bufnelems	INTEGER(4)		IN	結果格納配列の要素数
result	INTEGER(4)		OUT	終了コード

説明 引数 *type1* から *validtime* で指定されるデータファイルに書かれた SUBC または INFO 記録について、引数 *query* で指定される問合せを行う。

N_SUBC_NUM SUBC 記録の個数が 4 バイト整数型変数 *buf* に書かれる。引数 *group* は無視される。

N_SUBC_LIST データファイルに定義された SUBC 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_SUBC_NUM* 要素存在しなければならない。引数 *group* は無視される。

N_SUBC_NBYTES 群名 *group* の SUBC 記録のバイト数が 4 バイト整数型変数 *buf* に書かれる。

N_SUBC_CONTENT 群名 *group* の SUBC 記録が配列 *buf* に書かれる。上述のバイト数だけの長さを確保しておかねばならない。

N_INFO_NUM INFO 記録の個数が 4 バイト整数型変数 *buf* に書かれる。引数 *group* は無視される。

N_INFO_LIST データファイルに定義された INFO 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_INFO_NUM* 要素存在しなければならない。引数 *group* は無視される。

N_INFO_NBYTES 群名 *group* の INFO 記録のバイト数が 4 バイト整数型変数 *buf* に書かれる。

終了コード

正 格納要素数

履歴 この関数は NuSDaS 1.3 で新設された。

注意 「レコード内容」として取得されるのは表 A.6 項番 6 と同じであり、その長さはレコード有効長から 16 を引いたものに等しい。

5.5.9 NUSDAS_SCAN_DS: データセットの一覧

書式

CALL **NUSDAS_SCAN_DS**(*type1*, *type2*, *type3*, *nrd*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		OUT	種別 1
type2	CHARACTER(4)		OUT	種別 2
type3	CHARACTER(4)		OUT	種別 3
nrd	INTEGER(4)		OUT	NRD 番号
result	INTEGER(4)		OUT	終了コード

説明　返却値が負になるまで呼出しを繰り返すと、ライブラリが認識しているデータセットの一覧が得られる。

終了コード

0 引数の配列にデータセットの情報が格納された。

-1 もうこれ以上データセットは認識されていない。

履歴　この関数は NuSDaS 1.3 で追加された。pnusdas には非公開の nusdas_list_type という関数があり類似の機能を持つ。

5.6 メタデータ用サブルーチン

5.6.1 NUSDAS_SUBC_DELT: SUBC DELT へのアクセス

書式

CALL **NUSDAS_SUBC_DELT**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *delt*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
delt	REAL(4)		I/O	DELT 数値へのポインタ
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　モデルの時間積分間隔を補助管理情報に記録しておくものである。

終了コード

0 正常終了

-2 レコードが存在しない、または書き込まれていない。

-3 レコードサイズが不正

-5 入出力指示が不正

履歴　この関数は NuSDaS1.2 で導入された。

5.6.2 NUSDAS_SUBC_DELT_PRESET1: SUBC DELT のデフォルト設定

書式

CALL **NUSDAS_SUBC_DELT_PRESET1**(*type1*, *type2*, *type3*, *delt*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
delt	REAL(4)		IN	DELT 数値へのポインタ
result	INTEGER(4)		OUT	終了コード

説明　ファイルが新たに生成される際に DELT レコードに書き込む値を設定する。DELT レコードや引数については nusdas_subc_delt を参照。

終了コード

0 正常終了

-1 定義ファイルに "DELT" が登録されていない

-2 メモリの確保に失敗した

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

5.6.3 NUSDAS_SUBC_ETA: SUBC ETA へのアクセス

書式

CALL **NUSDAS_SUBC_ETA**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *n_levels*, *a*, *b*, *c*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
n_levels	INTEGER(4)		I/O	鉛直層数
a	REAL(4)	可変	I/O	係数 a
b	REAL(4)	可変	I/O	係数 b
c	REAL(4)		I/O	係数 c
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　鉛直座標に ETA 座標系を用いるときに、鉛直座標を定めるパラメータへのアクセスを提供する。パラメータは 4 バイト単精度浮動小数点型の配列 *a*, *b*, *c* で構成され、*a*, *b*, は鉛直層数 *n_levels* に対して、*n_levels*+1 要素の配列、*c* は 1 要素の配列 (変数) を確保する必要がある。*n_levels* は nusdas_subc_inq_nz で問い合わせることができる。入出力指示が *GET* の場合においても、*n_levels* は書込み対象変数として扱われる。特に parameter 宣言された変数を *n_levels* に指定してはならない。

終了コード

0 正常終了

-2 レコードが存在しない、またはレコードの書き込みがされていない。

-3 レコードサイズが不正

-4 ユーザーの鉛直層数がファイルの中の鉛直層数より小さい

-5 入出力指示が不正。

履歴　この関数は NuSDaS1.0 から存在した。NuSDaS1.1 までは、レコードが書き込まれたかの情報を持ち合わせていなかったために無記録のレコードをファイルから読んで正常終了していた。 NuSDaS 1.3 からはファイルの初期化時にレコードを初期化し、未記録を判定できるようにした。その場合のエラーは-2 としている。

注意　SUBC ETA に使われている鉛直層数 *n_levels* は実際のモデルの鉛直層数と異なっている場合があるので、配列確保の際には nusdas_subc_inq_nz で問い合わせた結果を用いること。

5.6.4 NUSDAS_SUBC_ETA_INQ_NZ: SUBC 記録の鉛直層数問合せ

書式

CALL **NUSDAS_SUBC_ETA_INQ_NZ**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *group*, *n_levels*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
group	CHARACTER(4)		IN	群名
n_levels	INTEGER(4)		OUT	鉛直層数
result	INTEGER(4)		OUT	終了コード

説明　SUBC レコードの ETA, SIGM, ZHYB に記録された鉛直層数を問い合わせる。群名には ”ETA ”, ”SIGM”, ”ZHYB” のいずれかを指定する。

終了コード

正 正常終了

履歴　この関数は NuSDaS1.2 で導入された。

5.6.5 NUSDAS_SUBC_PRESET1: SUBC ETA/SIGM のデフォルト値設定

書式

CALL **NUSDAS_SUBC_PRESET1**(*type1*, *type2*, *type3*, *group*, *n_levels*, *a*, *b*, *c*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
group	CHARACTER(4)		IN	群名
n_levels	INTEGER(4)		IN	鉛直層数
a	REAL(4)	可変	I/O	係数 a
b	REAL(4)	可変	I/O	係数 b
c	REAL(4)		I/O	係数 c
result	INTEGER(4)		OUT	終了コード

説明　ファイルが新たに生成される際に ETA, SIGM に書き込む値を設定する。SIGM や引数については nusdas_subc_eta を参照。引数の「群名」には、”ETA ” または ”SIGM” を指定する。

終了コード

0 正常終了

-1 定義ファイルに指定した群名が登録されていない

-2 メモリの確保に失敗した

-3 レコードのサイズが不正

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

5.6.6 NUSDAS_SUBC_RGAU: SUBC RGAU へのアクセス

書式

CALL **NUSDAS_SUBC_RGAU**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *j*, *j_start*, *j_n*, *i*, *i_start*, *i_n*, *lat*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
j	INTEGER(4)		I/O	全球の南北分割数
j_start	INTEGER(4)		I/O	データの最北格子の番号 (1 始まり)
j_n	INTEGER(4)		I/O	データの南北格子数
i	INTEGER(4)	可変	I/O	全球の東西格子数
i_start	INTEGER(4)	可変	I/O	データの最西格子の番号 (1 始まり)
i_n	INTEGER(4)	可変	I/O	データの東西格子数
lat	REAL(4)	可変	I/O	緯度
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　Reduced Gauss 格子を使う場合の補助管理情報へのアクセスを提供する。入出力指示が *GET* の場合においても、j_n の値はセットする。特に parameter 宣言された変数を j_n に指定してはならない。この j_n の値は nusdas_subc_rgau_inq_jn を使って問い合わせできる。i, i_start, i_n, lat は j_n 要素をもった配列を用意する。

終了コード

0 正常終了
-2 レコードが存在しない、または書き込まれていない。
-3 サイズの情報が引数と定義ファイルで不一致
-4 指定した入力値 (j_n, j_start, j_n, i, i_start, i_n) が不正 (PUT のときのみ)
-5 入出力指示が不正
-6 指定した入力値 (j_n) が不正 (GET のときのみ)

注意　Reduced Gauss 格子を使う場合は 1 次元でデータを格納するので、定義ファイルの size(格子数) には (実際の格子数) 1 と指定する。また、SUBC のサイズは $16 * \text{j\_n} + 12$ を計算した値を定義ファイルに書く。

履歴　この関数は NuSDaS1.2 で実装された

5.6.7 NUSDAS_SUBC_RGAU_INQ_JN: SUBC RGAU 記録の大きさを問合せ

書式

CALL **NUSDAS_SUBC_RGAU_INQ_JN**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *j_n*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
j_n	INTEGER(4)		OUT	南北格子数
result	INTEGER(4)		OUT	終了コード

説明　RGAU に記録されている j_n (南北格子数) を問い合わせる。

終了コード

正 正常終了

-2 要求されたレコードが存在しない、または書き込まれていない。

-3 レコードのサイズが不正

履歴　この関数は NuSDaS1.2 で導入された。

5.6.8 NUSDAS_SUBC_RGAU_PRESET1: SUBC RGAU のデフォルト値を設定

書式

CALL **NUSDAS_SUBC_RGAU_PRESET1**(*type1*, *type2*, *type3*, *j*, *j_start*, *j_n*, *i*, *i_start*, *i_n*, *lat*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
j	INTEGER(4)		IN	全球の南北分割数
j_start	INTEGER(4)		IN	データの最北格子の番号 (1 始まり)
j_n	INTEGER(4)		IN	データの南北格子数
i	INTEGER(4)	可変	IN	全球の東西格子数
i_start	INTEGER(4)	可変	IN	データの最西格子の番号 (1 始まり)
i_n	INTEGER(4)	可変	IN	データの東西格子数
lat	REAL(4)	可変	IN	緯度
result	INTEGER(4)		OUT	終了コード

説明　ファイルが新たに生成される際に RGAU レコードに書き込む値を設定する。RGAU レコードや引数については nusdas_subc_rgau を参照。

終了コード

0 正常終了

-1 定義ファイルに ”RGAU” が登録されていない

-2 メモリの確保に失敗した

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

5.6.9 NUSDAS_SUBC_SIGM: SUBC SIGM へのアクセス

書式

CALL **NUSDAS_SUBC_SIGM**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *n_levels*, *a*, *b*, *c*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
n_levels	INTEGER(4)		I/O	鉛直層数
a	REAL(4)	可変	I/O	係数 a
b	REAL(4)	可変	I/O	係数 b
c	REAL(4)		I/O	係数 c
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　鉛直座標に ETA 座標系を用いるときに、鉛直座標を定めるパラメータへのアクセスを提供する。関数の仕様は、nusdas_subc_eta と同じである。

5.6.10 NUSDAS_SUBC_SRF: 降短系 SUBC へのアクセス

書式

CALL **NUSDAS_SUBC_SRF**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *plane*, *element*, *group*, *data*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
plane	CHARACTER(6)		IN	面
element	CHARACTER(6)		IN	要素名
group	CHARACTER(4)		IN	群名
data	INTEGER(4)		I/O	データ配列
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　降水短時間予報系のデータの補助管理部へのアクセスを提供する。群名には次のもののいずれかを指定する。

ISPC レーダーや雨量計の運用情報、レベル値変換テーブルが格納される。data には 128 要素の 4 バイト整数型配列を用意する。内部のフォーマットは 4 バイト整数型であることは関係ないが、バイトオーダーの変換はされるので注意が必要。

THUN 詳細未詳。data には 4 バイト整数型変数を用意する。

RADR レーダー観測に関する情報。data には 4 バイト整数型変数を用意する。

RADS レーダー観測に関する情報。data には 6 要素の 4 バイト整数型配列を用意する。

DPRD ドップラーレーダー観測に関する情報。data には 8 要素の 4 バイト整数型配列を用意する。

終了コード

0 正常終了

-2 要求されたレコードが存在しない、または書かれていない。

-3 レコードサイズが不正

-4 群名が不正

-5 入出力指示が不正

履歴　この関数は NuSDaS1.0 から存在した。

5.6.11 NUSDAS_SUBC_TDIF: SUBC TDIF へのアクセス

書式

CALL **NUSDAS_SUBC_TDIF**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *diff_time*, *total_sec*, *getput*, *result*)

引数名	引数の型　配列長	I/O	役割
type1	CHARACTER(8)	IN	種別 1
type2	CHARACTER(4)	IN	種別 2
type3	CHARACTER(4)	IN	種別 3
basetime	INTEGER(4)	IN	基準時刻 (通算分)
member	CHARACTER(4)	IN	メンバー名
validtime	INTEGER(4)	IN	対象時刻 (通算分)
diff_time	INTEGER(4)	I/O	対象時刻からのずれ (秒)
total_sec	INTEGER(4)	I/O	総予報時間 (秒)
getput	CHARACTER(3)	IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)	OUT	終了コード

説明　格納した値の時刻の対象時間とのずれ、積算時間を格納する補助管理部 TDIF へのアクセスを提供する。

終了コード

0 正常終了

-2 要求されたレコードが存在しない、または書き込まれていない。

-3 レコードサイズが不正

-5 入出力指示が不正

補足

- diff_time = 時間範囲始点 - 対象時刻 [秒単位]
- total_sec = 時間範囲終点 - 時間範囲始点 [秒単位]

履歴　この関数は NuSDaS1.0 から存在した。

5.6.12 NUSDAS_SUBC_ZHYB: SUBC ZHYB へのアクセス

書式

CALL **NUSDAS_SUBC_ZHYB**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime*, *nz*, *ptrf*, *presrf*, *zrp*, *zrw*, *vctrans_p*, *vctrans_w*, *dvtrans_p*, *dvtrans_w*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime	INTEGER(4)		IN	対象時刻 (通算分)
nz	INTEGER(4)		I/O	鉛直層数
ptrf	REAL(4)		I/O	温位の参照値
presrf	REAL(4)		I/O	気圧の参照値
zrp	REAL(4)	可変	I/O	モデル面高度 (フルレベル)
zrw	REAL(4)	可変	I/O	モデル面高度 (ハーフレベル)
vctrans_p	REAL(4)	可変	I/O	座標変換関数 (フルレベル)
vctrans_w	REAL(4)	可変	I/O	座標変換関数 (ハーフレベル)
dvtrans_p	REAL(4)	可変	I/O	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	REAL(4)	可変	I/O	座標変換関数の鉛直微分 (ハーフレベル)
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

説明　鉛直座標に鉛直ハイブリッド座標をを使う場合の補助管理情報 ZHYB へのアクセスを提供する。入出力指示が *GET* の場合においても、nz の値をセットする。特に parameter 宣言された変数を nz に指定してはならない。この nz の値は nusdas_subc_eta_inq_nz を使って問い合わせできる。zrp, zrw, vctrans_p, vctrans_w, dvtrans_p, dvtrans_w は nz 要素をもった配列を用意する。

終了コード

0 正常終了

-2 レコードが存在しない、または書き込まれていない。

-3 サイズの情報が引数と定義ファイルで不一致

-4 指定した入力値 (ptrf, presrf) が不正 (PUT のときのみ)

-5 入出力指示が不正

-6 指定した入力値 (nz) が不正 (GET のときのみ)

注意　SUBC のサイズは $24 * \mathrm{nz} + 12$ を計算した値を定義ファイルに書く。

履歴　この関数は NuSDaS1.2 で実装された

5.6.13 NUSDAS_SUBC_ZHYB_PRESET1: SUBC ZHYB のデフォルト値を設定

書式

CALL **NUSDAS_SUBC_ZHYB_PRESET1**(*type1*, *type2*, *type3*, *nz*, *ptrf*, *presrf*, *zrp*, *zrw*, *vctrans_p*, *vctrans_w*, *dvtrans_p*, *dvtrans_w*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
nz	INTEGER(4)		IN	鉛直層数
ptrf	REAL(4)		IN	温位の参照値
presrf	REAL(4)		IN	気圧の参照値
zrp	REAL(4)	可変	IN	モデル面高度 (フルレベル)
zrw	REAL(4)	可変	IN	モデル面高度 (ハーフレベル)
vctrans_p	REAL(4)	可変	IN	座標変換関数 (フルレベル)
vctrans_w	REAL(4)	可変	IN	座標変換関数 (ハーフレベル)
dvtrans_p	REAL(4)	可変	IN	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	REAL(4)	可変	IN	座標変換関数の鉛直微分 (ハーフレベル)
result	INTEGER(4)		OUT	終了コード

説明　ファイルが新たに生成される際に ZHYB レコードに書き込む値を設定する。ZHYB レコードや引数については nusdas_subc_zhyb を参照。

終了コード

0 正常終了

-1 定義ファイルに ”ZHYB” が登録されていない

-2 メモリの確保に失敗した

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

5.7 サービスサブルーチン

5.7.1 ENDIAN_SWAB2: 2 バイト整数のバイトオーダー変換

書式

CALL **ENDIAN_SWAB2**(*ary*, *count*)

引数名	引数の型	配列長	I/O	役割
ary	任意	可変	I/O	配列
count	INTEGER(4)		IN	配列の要素数

説明　リトルエンディアン機では、2 バイト整数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

5.7.2 ENDIAN_SWAB4: 4 バイト整数のバイトオーダー変換

書式

CALL **ENDIAN_SWAB4**(*ary*, *count*)

引数名	引数の型	配列長	I/O	役割
ary	任意	可変	I/O	配列
count	INTEGER(4)		IN	配列の要素数

説明　リトルエンディアン機では、4 バイト整数または実数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

5.7.3 ENDIAN_SWAB8: 8 バイト整数のバイトオーダー変換

書式

CALL **ENDIAN_SWAB8**(*ary*, *count*)

引数名	引数の型	配列長	I/O	役割
ary	任意	可変	I/O	配列
count	INTEGER(4)		IN	配列の要素数

説明　リトルエンディアン機では、8 バイト整数または実数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

5.7.4 ENDIAN_SWAB_FMT: 任意構造のバイトオーダー変換

書式

CALL **ENDIAN_SWAB_FMT**(*ptr*, *fmt*)

引数名	引数の型	配列長	I/O	役割
ptr	任意	可変	I/O	変換対象
fmt	CHARACTER(∗)	可変	IN	書式

説明　リトルエンディアン機では、さまざまな長さのデータが混在するメモリ領域 *ptr* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

メモリのレイアウトは文字列 *fmt* で指定される。文字列は以下に示す型を表わす文字の羅列である。

D, d, L, l 8 バイト

F, f, I, i 4 バイト

H, h 2 バイト

B, b, N, n 1 バイト (なにもしない)

文字の前に数字をつけると繰り返し数をあらわす。たとえば "`4c8i`" は最初の 4 バイトが無変換、次に 4 バイト単位で 8 個変換を行うことを示す。

注意

- 数字は strtoul(3) で解釈しているので十進だけではなく八進や十六進も使える。たとえば "`0xFFi`" は 4 バイト単位で 255 個変換することを示し、"`0100h`" は 2 バイト単位で 64 個変換することを示す。

履歴　本関数は pnusdas から存在し、NuSDaS 1.3 で Fortran ラッパーを伴うサービスサブルーチンとしてドキュメントされた。

5.7.5　NUSDAS_GUNZIP: gzip 圧縮データを展開

書式

CALL **NUSDAS_GUNZIP**(*in_data*, *in_nbytes*, *out_buf*, *out_nbytes*, *result*)

引数名	引数の型	配列長	I/O	役割
in_data	任意	可変	IN	圧縮データ
in_nbytes	INTEGER(4)		IN	圧縮データのバイト数
out_buf	任意	可変	OUT	展開結果を格納する領域
out_nbytes	INTEGER(4)		IN	結果領域のバイト数
result	INTEGER(4)		OUT	終了コード

説明　入力データ *in_data* を gzip 展開して *out_buf* に格納する。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。

-99 入力は gzip 圧縮形式ではない。

-5 展開結果の長さが圧縮データと不整合。

-4 結果領域の長さ *out_nbytes* が不足している。

-3 展開結果の CRC32 が圧縮データと不整合。

-2 ZLib の inflateInit 関数がエラーを起こした。

-1 ZLib の inflate 関数がエラーを起こした。

他 展開データのバイト数

履歴　本関数は NuSDaS 1.3 で新設された。

5.7.6 NUSDAS_GUNZIP_NBYTES: gzip 圧縮データの展開後の長さを得る

書式

CALL **NUSDAS_GUNZIP_NBYTES**(*in_data*, *in_nbytes*, *result*)

引数名	引数の型	配列長	I/O	役割
in_data	任意	可変	IN	圧縮データ
in_nbytes	INTEGER(4)		IN	圧縮データのバイト数
result	INTEGER(4)		OUT	終了コード

説明 入力データ *in_data* を gzip 展開するときに必要となる結果格納領域のバイト数を返す。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。
正 展開後の長さ

履歴 本関数は NuSDaS 1.3 で新設された。

5.7.7 NUSDAS_GZIP: gzip 圧縮

書式

CALL **NUSDAS_GZIP**(*in_data*, *in_nbytes*, *out_buf*, *out_nbytes*, *result*)

引数名	引数の型	配列長	I/O	役割
in_data	任意	可変	IN	入力データ
in_nbytes	INTEGER(4)		IN	入力データのバイト数
out_buf	任意	可変	OUT	圧縮結果を格納する領域
out_nbytes	INTEGER(4)		IN	結果領域のバイト数
result	INTEGER(4)		OUT	終了コード

説明 入力データ *in_data* を gzip 圧縮して *out_buf* に格納する。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。
-9 ZLib の deflateEnd 関数がエラーを起こした。
-4 結果領域の長さ *out_nbytes* が不足している。
-2 ZLib の deflateInit2 関数がエラーを起こした。
-1 ZLib の deflate 関数がエラーを起こした。
他 圧縮データの長さ

履歴 本関数は NuSDaS 1.3 で新設された。

5.7.8 NUSDAS_UNPACK: 生 DATA レコードの解読

書式

CALL **NUSDAS_UNPACK**(*pdata*, *udata*, *utype*, *usize*, *result*)

引数名	引数の型	配列長	I/O	役割
pdata	任意	可変	IN	パックされたバイト列
udata	任意	可変	I/O	展開先配列
utype	CHARACTER(2)		IN	展開する型
usize	INTEGER(4)		IN	展開先配列の要素数
result	INTEGER(4)		OUT	終了コード

説明　nusdas_inq_data(p. 61) の問い合わせ N_DATA_CONTENT で得られるバイト列を解読して数値配列を得る。パッキング型 2UPJ では利用できない。

終了コード

正 正常終了、値は要素数

-4 展開先の大きさ *usize* がデータレコードの要素数より少ない

-5 パッキング型・欠損値型・展開型の組合せが不適

-6 利用できないパッキング型が与えられた

履歴　本関数は NuSDaS 1.3 で追加された。エラーコード -6 は NuSDaS 1.4 で新設されたもので、それ以前はエラーチェックがなされていなかった。

5.7.9　NUSDAS_UNCPSD: 2UPP を 2UPC に展開する

書式

CALL **NUSDAS_UNCPSD**(*pdata*, *cdata*, *ctype*, *result*)

引数名	引数の型	配列長	I/O	役割
pdata	任意	可変	IN	入力する 2UPP データ
cdata	任意	可変	I/O	展開先配列
csize	INTEGER(4)		IN	展開先配列のバイト数
result	INTEGER(4)		OUT	終了コード

説明　nusdas_inq_data(p. 61) の問い合わせ N_DATA_CONTENT で得られる 2UPP のバイト列を展開して 2UPC のバイト列として得ます。結果として返るバイト列は 2UPC 形式のデータに対して nusdas_inq_data(p. 61) の問い合わせ N_DATA_CONTENT で得られるデータと同じ形式です。

終了コード

正 正常終了、値は展開後のバイト数

-4 展開先配列の大きさ *csize* が必要バイト数より少ない

-5 入力データが 2UPP ではない

-6 2UPP から 2UPC への展開時にエラーが発生

履歴　本関数は NuSDaS 1.4 で追加された。

5.7.10　NUSDAS_UNCPSD_NBYTES: 2UPC 展開後の長さを得る

書式

CALL **NUSDAS_UNCPSD_NBYTES**(*pdata*, *udata*, *utype*, *usize*, *result*)

引数名	引数の型	配列長	I/O	役割
pdata	任意	可変	IN	入力する 2UPP データ
result	INTEGER(4)		OUT	終了コード

説明　入力データ *pdata* を nusdas_uncpsd(p. 79) で展開した後の展開後バイト数を得る。

終了コード

正 正常終了、値は展開後のバイト数

-5 入力データが 2UPP ではない

履歴　本関数は NuSDaS 1.4 で追加された。

5.7.11　N_DECODE_RLEN_NBIT_I1: RLE データを展開する

書式

CALL **N_DECODE_RLEN_NBIT_I1**(*udata*, *compressed_data*, *compressed_nbytes*, *udata_nelems*, *maxvalue*, *nbit*, *result*)

引数名	引数の型	配列長	I/O	役割
udata	CHARACTER	可変	OUT	結果格納配列
compressed_data	CHARACTER	可変	IN	圧縮データ
compressed_nbytes	INTEGER(4)		IN	圧縮データのバイト数
udata_nelems	INTEGER(4)		IN	圧縮データの要素数
maxvalue	INTEGER(4)		IN	データの最大値
nbit	INTEGER(4)		IN	圧縮データのビット数
result	INTEGER(4)		OUT	終了コード

説明

履歴　この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

5.7.12　N_ENCODE_RLEN_8BIT: 4 バイト整数を RLE 圧縮する

書式

CALL **N_ENCODE_RLEN_8BIT**(*udata*, *compressed_data*, *udata_nelems*, *max_compress_nbytes*, *maxvalue*, *result*)

引数名	引数の型	配列長	I/O	役割
udata	INTEGER(4)	可変	IN	元データ配列
compressed_data	CHARACTER	可変	OUT	結果格納配列
udata_nelems	INTEGER(4)		IN	元データの要素数
max_compress_nbytes	INTEGER(4)		IN	結果配列のバイト数
maxvalue	INTEGER(4)		OUT	元データの最大値
result	INTEGER(4)		OUT	終了コード

説明

履歴　この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

5.7.13　N_ENCODE_RLEN_8BIT_I1: 1 バイト整数を RLE 圧縮する

書式

CALL **N_ENCODE_RLEN_8BIT_I1**(*udata*, *compressed_data*, *udata_nelems*, *max_compress_nbytes*, *maxvalue*, *result*)

引数名	引数の型	配列長	I/O	役割
udata	CHARACTER	可変	IN	元データ配列
compressed_data	CHARACTER	可変	OUT	結果格納配列
udata_nelems	INTEGER(4)		IN	元データの要素数
max_compress_nbytes	INTEGER(4)		IN	結果配列のバイト数
maxvalue	INTEGER(4)		OUT	元データの最大値
result	INTEGER(4)		OUT	終了コード

説明

履歴　この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

5.8 降水短時間ライブラリ

5.8.1 概要

降水短時間ルーチンに関連するアメダスデータ・レーダーデータに特有な処理のためのサブルーチンをまとめたものが降水短時間ライブラリ libsrf.a として提供される。ライブラリ全体の関数プロトタイプを与えるヘッダは存在しない(注 1)が、SRF_AMD_RDIC(p. 83) 及び SRF_SEARCH_AMDSTN(p. 86) を呼ぶ時は SRF_AMD_SINFO 型や定数の定義を参照するため`"srf_amedas_fort.h"` をインクルードする必要がある。

5.8.2 RDR_LV_TRANS: レベル値から代表値への変換

書式

CALL **RDR_LV_TRANS**(*idat*, *fdat*, *dnum*, *param*, *result*)

引数名	引数の型	配列長	I/O	役割
idat	INTEGER(4)	可変	I/O	入力データ
fdat	REAL(4)	可変	OUT	結果格納配列
dnum	INTEGER(4)		IN	データ要素数
param	CHARACTER(∗)	可変	IN	テーブル名
result	INTEGER(4)		OUT	終了コード

説明 配列 *idat* のレベル値を代表値 *fdat* に変換する。変換テーブルとしてファイル ./SRF_LV_TABLE/param.ltb を読む。ここで *param* は変換テーブル名 (最長 4 字) である。

終了コード

-1 変換テーブルを開くことができない

-2 変換テーブルに 256 以上のレベルが指定されている

非負 変換に成功。返却値は不明値以外となったデータの要素数

注

- 不明値は -1 となる。
- NAPS8 では変換テーブルとして/grpK/nwp/Open/Const/Vsrf/Comm/lvtbl.txd 以下に her ie2 ier kor pft pi10 pm2 pmf pr2 prr rr60 sr1 sr2 sr3 srf srj srr yar yrr が置かれている。ルーチンジョブではこのディレクトリへシンボリックリンク SRF_LV_TABLE を張って利用する。
- 上記変換テーブルのうち、pi10 と rr60 は 1 行にレベル値と代表値の 2 列が書かれており、その他はレベル値、最小値、代表値の 3 列が書かれているが、本サブルーチンはどちらにも対応している。

履歴 この関数は NAPS7 時代には存在しなかったようである。レーダー情報作成装置に関連して開発されたと考えられているが、NuSDaS 1.3 以前にはきちんとメンテナンスされていなかった。

5.8.3 SRF_AMD_AQC: AQC のパックを展開

書式

CALL **SRF_AMD_AQC**(*aqc_in*, *num*, *aqc_out*, *param*)

引数名	引数の型	配列長	I/O	役割
aqc_in	INTEGER(2)	可変	IN	AQC 配列
num	INTEGER(4)		IN	配列要素数
aqc_out	INTEGER(2)	可変	I/O	結果配列
param	CHARACTER(∗)	可変	IN	要素名

(注 1) このため IBM 系環境では Fortran ラッパーが提供できない。

説明　アメダス デコード データセットに含まれる AQC から要素名 *param* で指定される各ビットフィールドを取り出す。

UNYOU △ 入電・休止・運用情報 (-1:休止, 0:入電無し, 正:入電回数)

RRfr0 △ 降水量の情報 (0:入電無し, 1:ハードエラー・欠測・休止, 2:AQC 該当値, 3:正常値; 以下同じ)

SSfr0 △ 日照時間の情報

T △△△△△ 気温の情報

WindD △ 風向の情報

WindS △ 風速の情報

SnowD △ 積雪深の情報

注意　要素名が不正な場合は警告後なにもせず終了する。(NuSDaS 1.3 より前は不定動作)

履歴　この関数は NAPS7 時代から存在した。

5.8.4 SRF_AMD_RDIC: アメダス地点辞書の読み込み

書式

CALL **SRF_AMD_RDIC**(*amd*, *amdnum*, *btime*, *amd_type*, *result*)

引数名	引数の型	配列長	I/O	役割
amd	type(SRF_AMD_SINFO)	可変	I/O	地点辞書格納配列
amdnum	INTEGER(4)		IN	地点辞書配列長
btime	INTEGER(4)		IN	探索日時 (通算分)
amd_type	INTEGER(4)		IN	地点種別
result	INTEGER(4)		OUT	終了コード

説明　環境変数 NWP_AMDDCD_STDICT が指す地点辞書ファイル (環境変数未定義時は amddic.txt となる) から SRF_AMD_SINFO 構造型の配列 *amd* にアメダス地点情報を読出す。読出される地点は時刻 *btime* に存在するものが選ばれ、さらに引数 *amd_type* によって次のように限定される。配列長 *amdnum* を越えて書き出すことはない。

SRF_KANS 官署

SRF_ELM4 4 要素を観測している地点

SRF_AMEL ロボット雨量計

SRF_AIRP 航空官署

SRF_YUKI 積雪観測地点

SRF_ALL 全地点

終了コード

非負 地点数

-1 地点種別が不正

-2 結果格納配列の長さ不足

-3 地点辞書ファイルが開けない

参考　NAPS8 においては地点辞書は/grpK/nwp/Open/Const/Pre/Dcd/amddic.txt に置かれている。

履歴　この関数は NAPS7 時代から存在した。

5.8.5 SRF_AMD_SLCT: アメダスデータを指定の地点番号順に並べる

書式

CALL **SRF_AMD_SLCT**(*st_r*, *n_st_r*, *d_r*, *t_r*, *st_n*, *n_st_n*, *d_n*, *t_n*, *sort_f*, *result*)

引数名	引数の型	配列長	I/O	役割
st_r	INTEGER(4)		IN	結果の順を指示する地点番号表
n_st_r	INTEGER(4)		IN	結果配列長
d_r	任意	可変	OUT	結果配列
t_r	CHARACTER(∗)	可変	IN	結果配列の型
st_n	INTEGER(4)		I/O	元データ地点番号配列
n_st_n	INTEGER(4)		IN	元データ配列長
d_n	任意	可変	I/O	元データ配列
t_n	CHARACTER(∗)	可変	IN	元データ配列の型
sort_f	INTEGER(4)		IN	未ソートフラグ
result	INTEGER(4)		OUT	終了コード

説明　長さ *n_st_n* の地点番号配列 *st_n* と対応する順に並んだ *t_n* 型の配列 *d_n* から、別の地点番号配列 *st_r* (要素数 *n_st_r* 個) の順に並んだ *t_r* 型の配列 *d_r* (要素数 *n_st_r* 個) を作る。

配列 *st_n* と *d_n* があらかじめソートされている場合 *sort_f* に N_OFF (nusdas.h で定義される) を指定する。そうでない場合 *sort_f* に N_ON を指定するとソートされる。

終了コード

0 配列 *st_r* の全地点が見付かった

正 みつからなかった地点数

- 型は nusdas.h で定義される N_R4, N_I4, N_I2 のいずれかで指定する。
- 配列 *st_r* に含まれる地点番号が *st_n* で見付からない場合は nusdas.h で定義される欠損値 N_MV_R4, N_MV_SI4, N_MV_SI2 が入る。

履歴　この関数は NAPS7 時代から存在した。

5.8.6 SRF_LV_SET: 実数からレベル値への変換

書式

CALL **SRF_LV_SET**(*idat*, *fdat*, *dnum*, *ispec*, *param*, *result*)

引数名	引数の型	配列長	I/O	役割
idat	INTEGER(4)	可変	OUT	レベル値格納配列
fdat	REAL(4)	可変	IN	変換元データ配列
dnum	INTEGER(4)		IN	データ配列要素数
ispec	INTEGER(4)	可変	I/O	新 ISPC
param	CHARACTER(∗)	可変	IN	変換テーブル名
result	INTEGER(4)		OUT	終了コード

説明　配列 *fdat* の実数データをレベル値 *idat* に変換し、ISPC 配列 *ispec* にレベル代表値をセットする。変換テーブルとしてファイル ./SRF_LV_TABLE/param.ltb を読む。ここで *param* は変換テーブル名 (最長 4 字) である。

終了コード

-1 変換テーブルを開くことができない

-2 変換テーブルに 256 以上のレベルが指定されている

正 変換に成功した。返却値はレベル数

注

- NAPS7 では変換テーブルとして her ier prr pmf srr srf pr2 を用意していた。NAPS8 では /grpK/nwp/Open/Const/Vsrf/Comm/lvtbl.txd 以下に her ie2 ier kor pft pm2 pmf pr2 prr sr1 sr2 sr3 srf srj srr yar yrr が置かれている。ルーチンジョブではこのディレクトリへシンボリックリンク SRF_LV_TABLE を張って利用する。
- 変換テーブル名が ie2, kor, pft のとき, ISPEC には変換テーブルに書かれた代表値の 1/10 が書かれる。それ以外の場合は変換テーブルの代表値がそのまま書かれる。
- 変換テーブル名が sr2 または srj のときは実数データが -900.0 より小さいものが欠損値とみなされる。そうでなければ、負値が欠損値とみなされる (NAPS7 のマニュアルでは欠損値は -1 を指定することとされている)。
- 変換テーブル名が srj のときは、実数データが変換テーブルの下限値に正確に一致しないと最も上の階級 (具体的には 42 で 21.0 を意味する) に割り当てられる。この挙動はバグかもしれない。
- 変換テーブルに 191 行以上書かれているとき、最初の 190 行だけが用いられ、レベル値は 0..190 となるが、返却値には実際のレベル数 (変換テーブルの行数 + 1) が返される。これは ispec の配列をオーバーフローしないためである。

履歴　この関数は NAPS7 時代から存在した。Fortran ラッパーが文字列の長さを伝えないバグは NuSDaS 1.3 で解決した。

5.8.7 SRF_LV_TRANS: レベル値を実数データ (代表値) に変換

書式

CALL **SRF_LV_TRANS**(*idat*, *fdat*, *dnum*, *ispec*, *result*)

引数名	引数の型	配列長	I/O	役割
idat	INTEGER(4)	可変	IN	入力データ
fdat	REAL(4)	可変	OUT	結果格納配列
dnum	INTEGER(4)		IN	データ要素数
ispec	INTEGER(4)	可変	IN	ISPEC 配列
result	INTEGER(4)		OUT	終了コード

説明　新 ISPEC 配列 *ispec* にしたがって配列 *idat* のレベル値を代表値 *fdat* に変換する。

返却値　不明値以外となったデータの要素数

注

- 不明値は -1 となる。ただし、ISPEC のデータ種別 (先頭 4 バイト) が SRR2, SRF2, SRRR, SRFR の場合に限り -9999.0 となる。
- ISPEC のレベル表は通常 0.1mm 単位と解釈される。ただし、ISPEC の先頭 3 バイトが 'IER' であるか、あるいは ISPEC の先頭から 4 バイト目が '1' のときは 0.01mm 単位と解釈される。

履歴　この関数は NAPS7 時代から存在した。

5.8.8 SRF_RD_RDIC: レーダーサイト情報の問い合わせ

書式

CALL **SRF_RD_RDIC**(*stnum*, *iseq*, *lat*, *lon*, *hh*, *offx*, *offy*, *type1*, *type2*, *result*)

引数名	引数の型	配列長	I/O	役割
stnum	INTEGER(4)		IN	地点番号
iseq	INTEGER(4)		IN	日時 (通算時)
lat	REAL(4)		OUT	緯度
lon	REAL(4)		OUT	経度
hh	REAL(4)		OUT	高度
offx	INTEGER(4)		OUT	中心のオフセット
offy	INTEGER(4)		OUT	中心のオフセット
type1	INTEGER(4)		OUT	デジタル化タイプ
type2	INTEGER(4)		OUT	デジタル化タイプ
result	INTEGER(4)		OUT	終了コード

説明　ファイル名 RADAR_DIC のレーダー地点辞書から日時 *iseq* (通算時であって通算分でないことに注意) における地点番号 *stnum* のレーダーサイトの情報を読出す。

終了コード

1 正常終了
0 指定されたレーダーサイトがみつからなかった
-1 レーダー地点辞書が開けなかった

注

- レーダー地点辞書は NAPS8 では/grpK/nwp/Open/Const/Vsrf/Dcd/rdrdic.txt に置かれている。
- NAPS8 初期版 libsrf.a には経度のかわりに誤って緯度を返すバグがある。

履歴　この関数は NAPS7 時代からルーチン環境には存在したが、pnusdas から NuSDaS 1.1 に至る CVS 版ソースには含まれていなかった。NuSDaS 1.3 で両者が統合された。

5.8.9 SRF_SEARCH_AMDSTN: 地点番号の辞書内通番を探す

書式

CALL **SRF_SEARCH_AMDSTN**(*amd*, *ac*, *stn*, *amd_type*, *result*)

引数名	引数の型	配列長	I/O	役割
amd	type(SRF_AMD_SINFO)	可変	IN	地点辞書配列
ac	INTEGER(4)		IN	地点辞書配列の長さ
stn	INTEGER(4)		IN	地点番号
amd_type	INTEGER(4)		IN	地点種別
result	INTEGER(4)		OUT	終了コード

説明　SRF_AMD_SINFO 構造型の配列 *amd* (地点数 *ac* 個) から地点番号 *stn* の地点情報を収めた添字 (1 始まり) を返す。

終了コード

正 地点の辞書内格納順位 (1 始まり)
-1 地点がみつからない

注意

- 地点種別 *amd_type* は無視される。
- 配列が地点番号順にソートされていることを前提に二分探索を使っている。

履歴　この関数は NAPS7 時代から存在したようであるがドキュメントされていなかった。NuSDaS 1.3 リリースに際してドキュメントされるようになった。

6 C リファレンスマニュアル

6.1 凡例

- 引数 *fmt* または *utype* (配列の型) に与えるべき定数名は、Table B.7 (p. 154) 参照。

6.2 最低限知るべき関数

6.2.1 nusdas_read: データ記録の読取

書式

N_SI4 **nusdas_read**(const char *utype1*[8], const char *utype2*[4], const char *utype3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], void ∗*data*, const char *fmt*[2], const N_SI4 ∗*size*);

引数名	引数の型	役割
utype1	const char [8]	種別 1
utype2	const char [4]	種別 2
utype3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー
validtime	const N_SI4 ∗	対象時刻 (通算分)
plane	const char [6]	面の名前
element	const char [6]	要素名
data	void ∗	結果格納配列
fmt	const char [2]	結果格納配列の型
size	const N_SI4 ∗	結果格納配列の要素数

説明　引数で指定した TYPE, 基準時刻、メンバー、対象時刻、面、要素のデータを読み出す。

終了コード

正 読み出して格納した格子数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

-5 格納配列の型とレコードの記録形式が不整合

注意　nusdas_read では、返却値 0 はエラーであることに注意が必要。nusdas_read のエラーチェックは返却値が求めている格子数と一致していることを確認するのが望ましい。

互換性　NuSDaS1.1 では「ランレングス圧縮で、データが指定最大値を超えている」(返却値-6) が定義されていたが、はデータの最初だけを見ているだけで意味がないと思われるので、 NuSDaS 1.3 からはこのエラーは返さない。また、「ユーザーオープンファイルの管理部又はアドレス部が不正である」(返却値-7) は、共通部分の-54〜-57 に対応するので、このエラーは返さない

6.2.2 nusdas_write: データ記録の書出

書式

N_SI4 **nusdas_write**(const char *utype1*[8], const char *utype2*[4], const char *utype3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], const void ∗*data*, const char *fmt*[2], const N_SI4 ∗*nelems*);

引数名	引数の型	役割
utype1	const char [8]	種別 1
utype2	const char [4]	種別 2
utype3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
plane	const char [6]	面の名前
element	const char [6]	要素名
data	const void *	データ配列
fmt	const char [2]	データ配列の型
nelems	const N_SI4 *	データ配列の要素数

説明　データレコードを指定された場所に書き出す。

終了コード

正 実際に書き出された要素数

-2 メンバー名、面名、要素名が間違っている

-2 このレコードは ELEMENTMAP によって書き出しが禁止されている

-3 与えられたデータ要素数 *nelems* が必要より小さい

-4 指定データセットにはデータ配列の型 *fmt* は書き出せない

-5 データレコード長が固定レコード長を超える

-6 データセットの欠損値指定方式と RLEN 圧縮は併用できない

-7 マスクビットの設定がされていない

-8 エンコード過程でのエラー (数値が過大または RLEN 圧縮エラー)

注意

- データセットの指定と異なる大きさのレコードを書き出すにはあらかじめ nusdas_parameter_change (p. 98) を使って設定を変えておく。
- 格子数 (データセットの指定または nusdas_parameter_change (p. 98) 設定) より大きい要素数 *nelems* を指定するとエラーにはならず、余った要素が書き出されない結果となるので注意されたい。

履歴　この関数は NuSDaS 1.0 から存在した。

6.2.3 nusdas_iocntl: 入出力フラグ設定

書式

N_SI4 **nusdas_iocntl**(N_SI4 *param*, N_SI4 *value*);

引数名	引数の型	役割
param	N_SI4	設定項目コード
value	N_SI4	設定値

説明　入出力にかかわるフラグを設定する.

N_IO_MARK_END 既定値 1. 零にすると nusdas_write (p. 89) などの出力関数を呼び出すたびにデータファイルへの出力を完結させ END 記録を書き出すのをやめる.

N_IO_W_FCLOSE 既定値 1. 零にすると nusdas_write (p. 89) などの出力関数を呼び出すたびに書き込み用に開いたファイルを閉じるのをやめる. 速度上有利だが、データファイルの操作が終了した後でファイルを閉じる関数 nusdas_allfile_close (p. 91) または nusdas_onefile_close (p. 97) を適切に呼んでファイルを閉じないと出力ファイルが不完全となり、後で読むことができない. なお、このフラグを変更すると N_IO_MARK_END も連動する.

N_IO_R_FCLOSE 既定値 1. 零にすると nusdas_read (p. 89) などの入力関数を呼び出すたびに読み込み用に開いたファイルを閉じるのをやめる. 速度上有利だが、多数のファイルから入力するプログラムではファイルハンドルが枯渇する懸念があるのでファイルを明示的に閉じることが推奨される.

N_IO_WARNING_OUT 既定値 1. 0 にするとエラーメッセージだけが出力される. 1 にすると、それに加えて警告メッセージも出力されるようになる. 2 にすると、それに加えてデバッグメッセージも出力されるようになる.

N_IO_BADGRID 既定値 0. 1 にすると投影法パラメタの検査で不適切な値が検出されてもデータファイルが作成できるようになる。

終了コード

0 正常終了

-1 サポートされていないパラメタである

履歴 この関数は NuSDaS 1.0 から存在した. **N_IO_WARNING_OUT** の値 2 は NuSDaS 1.3 からの拡張である. **N_IO_BADGRID** も NuSDaS 1.3 からの拡張である.

6.2.4 nusdas_allfile_close: 全てのデータファイルを閉じる

書式

N_SI4 **nusdas_allfile_close**(N_SI4 *param*);

引数名	引数の型	役割
param	N_SI4	閉じるファイルの種類

説明 今までに NuSDaS インターフェイスで開いた全てのファイルを閉じる. 引数 param は次のいずれかを用いる:

N_FOPEN_READ 読み込み用に開いたファイルだけを閉じる

N_FOPEN_WRITE 書き込み可で開いたファイルだけを閉じる

N_FOPEN_ALL すべてのファイルを閉じる

終了コード

正 正常に閉じられたファイルの数

0 閉じるべきファイルがなかった

負 閉じる際にエラーが起こったファイルの数

履歴 この関数は NuSDaS 1.0 から存在した.

6.3 データ読書関数

6.3.1 nusdas_cut: 領域限定のデータ読取

書式

N_SI4 **nusdas_cut**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], void ∗*udata*, const char *utype*[2], const N_SI4 ∗*usize*, const N_SI4 ∗*ixstart*, const N_SI4 ∗*ixfinal*, const N_SI4 ∗*iystart*, const N_SI4 ∗*iyfinal*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
plane	const char [6]	面
element	const char [6]	要素名
udata	void ∗	データ格納先配列
utype	const char [2]	データ格納先配列の型
usize	const N_SI4 ∗	データ格納先配列の要素数
ixstart	const N_SI4 ∗	x 方向格子番号下限
ixfinal	const N_SI4 ∗	x 方向格子番号上限
iystart	const N_SI4 ∗	y 方向格子番号下限
iyfinal	const N_SI4 ∗	y 方向格子番号上限

説明 nusdas_read (p. 89) ∗ と同様だが、データレコードのうち格子点 (*ixstart* , *iystart*)–(*ixfinal* , *iyfinal*) だけが *udata* に格納される。

格子番号は 1 から始まるものとするため、*ixstart* や *iystart* は正でなければならず、また *ixfinal* や *iyfinal* はそれぞれ *ixstart* や *iystart* 以上でなければならない。この規則に反する指定を行った場合は、返却値-8 のエラーとなる。なお、*iyfinal*, *jyfinal* の上限が格子数を超えていることのチェックはしていないので注意が必要。

終了コード

正 読み出して格納した格子数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

-5 格納配列の型とレコードの記録形式が不整合

-8 領域指定パラメータが不正

履歴 本関数は NuSDaS 1.1 で導入され、 NuSDaS 1.3 で初めてドキュメントされた。

互換性 NuSDaS 1.1 では、ローカルのデータファイルに対しては、$ixstart \leq 0$ の場合は $ixstart = 1$ に (*jystart* も同様), *ixfinal* が X 方向の格子数を超える場合には、*ixfinal* は X 方向の格子数に (*jyfinal* も同様) に読み替えられていたが、 NuSDaS 1.3 からは返却値-8 のエラーとする。また、pandora データについては、*ixstart*, *ixfinal*, *jystart*, *jyfinal* が非負であることだけがチェックされていた。NuSDaS 1.3 からはデータファイル、pandora とも上述の通りとなる。

6.3.2 nusdas_cut_raw: 領域限定の DATA 記録直接読取

書式

N_SI4 **nusdas_cut_raw**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], void ∗*udata*, const N_SI4 ∗*usize*, const N_SI4 ∗*ixstart*, const N_SI4 ∗*ixfinal*, const N_SI4 ∗*iystart*, const N_SI4 ∗*iyfinal*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
plane	const char [6]	面
element	const char [6]	要素名
udata	void ∗	データ格納先配列
usize	const N_SI4 ∗	データ格納先配列のバイト数
ixstart	const N_SI4 ∗	x 方向格子番号下限
ixfinal	const N_SI4 ∗	x 方向格子番号上限
iystart	const N_SI4 ∗	y 方向格子番号下限
iyfinal	const N_SI4 ∗	y 方向格子番号上限

説明　nusdas_read2_raw (p. 93) と類似だが、データレコードのうち格子点 (*ixstart* , *iystart*)–(*ixfinal* , *iyfinal*) に対応する部分だけが *udata* に格納される。ただしパッキングが 2UPP, 2UPJ, RLEN の場合、または欠損値が MASK の場合は全体が格納される。

終了コード

正 読み出して格納したバイト数

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

履歴　この関数は NuSDaS 1.1 で導入された。エラーコード -4 は NuSDaS 1.3 で新設されたもので、それ以前はエラーチェックがなされていなかった。

6.3.3 nusdas_read2_raw: DATA 記録内容の直接読取

書式

N_SI4 **nusdas_read2_raw**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], void ∗*buf*, const N_SI4 ∗*buf_nbytes*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1
validtime2	const N_SI4 *	対象時刻 2
plane1	const char [6]	面 1
plane2	const char [6]	面 2
element	const char [6]	要素名
buf	void *	データ格納配列
buf_nbytes	const N_SI4 *	データ格納配列のバイト数

説明　引数で指定した TYPE, 基準時刻、メンバー、対象時刻、面、要素のデータをファイルに格納されたままの形式で読み出す。データは、DATA レコードのフォーマット表の項番 10〜14 までのデータが格納される。

終了コード

正 読み出して格納したバイト数。

0 指定したデータは未記録 (定義ファイルの elementmap によって書き込まれることは許容されているが、まだデータが書き込まれていない)

-2 指定したデータは記録することが許容されていない (elementmap によって禁止されている場合と指定した面名、要素名が登録されていない場合の両方を含む)。

-4 格納配列が不足

履歴　この関数は NuSDaS1.1 で導入された。

6.3.4 nusdas_read_3d: 高次元読み込み

書式

N_SI4 **nusdas_read_3d**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 **basetime*, const char *member[]*[4], const N_SI4 *validtime*[ ], const char *plane[]*[6], const char *element[]*[6], const N_SI4 **nrecs*, void **udata*, const char *utype*[2], const N_SI4 **usize*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻
member	const char [][4]	メンバ名の配列
validtime	const N_SI4 []	対象時刻の配列
plane	const char [][6]	面名の配列
element	const char [][6]	要素名の配列
nrecs	const N_SI4 *	レコード数
udata	void *	結果格納配列
utype	const char [2]	結果格納配列の型
usize	const N_SI4 *	レコードあたり要素数

説明　*member*, *validtime*, *plane*, *element* には *nrecs* 個の大きさを持つ配列を指定する。これらの配列の各要素を指定して nusdas_read (p. 89) を順次呼びだし *udata* に格納する。*udata* の要素数は *nrecs* * *usize* 個以上でなければならない。nusdas_read の返却値（終了コード）が *usize* と一致しなかった場合は読み込みを終了する。

終了コード

正 全てのデータを読むことができた場合は *nrecs* * *usize* を返す。

負 最後に呼び出した nusdas_read (p. 89) の終了コードを返す。

注意 読み込みを途中で終了した場合、どこまで処理したかを知る方法は無い。

6.3.5 nusdas_write_3d: 高次元書き出し

書式

N_SI4 **nusdas_write_3d**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member[]*[4], const N_SI4 *validtime*[ ], const char *plane[]*[6], const char *element[]*[6], const N_SI4 ∗*nrecs*, const void ∗*udata*, const char *utype*[2], const N_SI4 ∗*usize*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻
member	const char [ ][4]	メンバ名の配列
validtime	const N_SI4 [ ]	対象時刻の配列
plane	const char [ ][6]	面名の配列
element	const char [ ][6]	要素名の配列
nrecs	const N_SI4 ∗	レコード数
udata	const void ∗	データ配列
utype	const char [2]	データ配列の型
usize	const N_SI4 ∗	レコードあたり要素数

説明 *member*, *validtime*, *plane*, *element* には *nrecs* 個の大きさを持つ配列を指定する。これらの配列の各要素を指定して nusdas_write (p. 89) を順次呼びだす。データ配列の要素数は *nrecs* * *usize* 個でなければならない。nusdas_write の返却値（終了コード）が *usize* と一致しなかった場合は書き出しを終了する。

終了コード

正 全てのデータを書き出すことができた場合は *nrecs* * *usize* を返す。

負 最後に呼び出した nusdas_write (p. 89) の終了コードを返す。

注意 書き出しを途中で終了した場合、どこまで処理したかを知る方法は無い。

6.4 動作制御用関数

6.4.1 nusdas_esf_flush: NAPS7 型 ES ファイルの出力完了

書式

N_SI4 **nusdas_esf_flush**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻

説明

履歴 nusdas_esf_flush (p. 96) は NuSDaS 1.0 から存在する。

バグ NuSDaS 1.3 からは ES をサポートしていないため、この関数はダミーである。

6.4.2 nusdas_make_mask: マスクビット配列の作成

書式

N_SI4 **nusdas_make_mask**(const void ∗*udata*, const char *utype*[2], const N_SI4 ∗*usize*, void ∗*output*, const N_SI4 ∗*mb_bytes*);

引数名	引数の型	役割
udata	const void ∗	格子データ
utype	const char [2]	格子データの型
usize	const N_SI4 ∗	格子データの要素数
output	void ∗	マスクビット配列
mb_bytes	const N_SI4 ∗	マスクビット配列のバイト数

説明 配列 *udata* の内容をチェックしてマスクビット列を作成し *output* に書き込む。引数 *utype* と欠損値は配列の型に応じて次のように指定する。

1 バイト整数型 引数 *utype* に N_I1 を指定する。配列中の欠損扱いしたい要素に N_MV_UI1 を設定しておく。

2 バイト整数型 引数 *utype* に N_I2 を指定する。配列中の欠損扱いしたい要素に N_MV_SI2 を設定しておく。

4 バイト整数型 引数 *utype* に N_I4 を指定する。配列中の欠損扱いしたい要素に N_MV_SI4 を設定しておく。

4 バイト実数型 引数 *utype* に N_R4 を指定する。配列中の欠損扱いしたい要素に N_MV_R4 を設定しておく。

8 バイト実数型 引数 *utype* に N_R8 を指定する。配列中の欠損扱いしたい要素に N_MV_R8 を設定しておく。

終了コード

0 正常終了

-1 配列長 *mb_bytes* が不足している

-5 未知の型名 *utype* が与えられた

サイズ要件　*mb_bytes* は少なくとも (*usize* + 7) / 8 バイト以上必要である。

履歴　nusdas_make_mask (p. 96) は NuSDaS 1.0 から存在する。

6.4.3　nusdas_set_mask: 改善型マスクビット設定関数

書式

N_SI4 **nusdas_set_mask**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const void *∗udata*, const char *utype*[2], N_SI4 *usize*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
udata	const void ∗	データ配列
utype	const char [2]	データ配列の型
usize	N_SI4	配列の要素数

説明　配列 *udata* の内容に従って nusdas_make_mask (p. 96) と同様にマスクビット列を作成し指定した種別のデータセットに対して設定する。

終了コード

0 正常終了

-5 未知の型名 *utype* が与えられた

注意　本関数によるマスクビットの設定は nusdas_parameter_change (p. 98) に優先するが、他のデータセットには効果をもたない。

履歴　本関数は NuSDaS 1.3 で新設された。

6.4.4　nusdas_onefile_close: 指定データファイルを閉じる

書式

N_SI4 **nusdas_onefile_close**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *∗basetime*, const char *member*[4], const N_SI4 *∗validtime*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻

終了コード

0 正常終了

1 ファイルクローズ前の書き込み時に定義ファイルを読み込めなかった

-1 ファイルクローズ前の書き込み時に IO エラーが発生

説明

履歴　この関数は NuSDaS 1.0 から存在した.

6.4.5 nusdas_parameter_change: オプション設定

書式

N_SI4 **nusdas_parameter_change**(N_SI4 *param*, const void ∗*value*);

引数名	引数の型	役割
param	N_SI4	設定項目コード
value	const void ∗	設定値

説明 *param* で指定されるパラメータに値 *value* を設定する。整数値の項目については、互換性のため値ゼロのかわりに名前 N_OFF を用いることができる。

N_PC_MISSING_UI1 1 バイト整数の欠損値 (既定値: N_MV_UI1)

N_PC_MISSING_SI2 2 バイト整数の欠損値 (既定値: N_MV_SI2)

N_PC_MISSING_SI4 4 バイト整数の欠損値 (既定値: N_MV_SI4)

N_PC_MISSING_R4 4 バイト実数の欠損値 (既定値: N_MV_R4)

N_PC_MISSING_R8 8 バイト実数の欠損値 (既定値: N_MV_R8)

N_PC_MASK_BIT マスクビット配列へのポインタ (既定値は NULL ポインタだが Fortran では直接設定できないので nusdas_parameter_reset (p. 98) を用いられたい)

N_PC_SIZEX 非零値を設定すると強制的にデータレコードの x 方向格子数を設定する (0)

N_PC_SIZEY 強制格子サイズ: 既定値 (0) 以外を設定するとデータレコードの y 方向格子数を設定する

N_PC_PACKING 4 文字のパッキング名を設定すると、定義ファイルの指定にかかわらず nusdas_write (p. 89) 等データ記録書き込みの際に用いられるパッキング方式が変更される。既定値に戻す (定義ファイルどおりに書かせる) には 4 バイト整数値 0 を設定する。

N_PC_ID_SET NRD 番号制約: 既定値 (-1) 以外を設定すると、その番号の NRD だけを入出力に用いるようになる

N_PC_WBUFFER 書き込みバッファサイズ (既定値: 0) 実行時オプション FWBF に同じ。

N_PC_RBUFFER 読み取りバッファサイズ (既定値: 17) 実行時オプション FRBF に同じ。

N_PC_KEEP_CFILE ファイルを閉じたあと CNTL/INDX などのヘッダ情報をキャッシュしておく数。負にするとキャッシュを開放しなくなる（既定値: -1)。実行時オプション GKCF に同じ。

N_PC_OPTIONS 設定のみでリセットはできない。ヌル終端した文字列を与えると実行時オプションとして設定する。Fortran インターフェイスでもヌル終端しなければならないことに注意。

終了コード

0 正常終了

-1 サポートされていないパラメタである

履歴 NuSDaS 1.0 から存在する。

NuSDaS 1.1 ではデータセット探索のキャッシュ論理に問題があり、N_PC_ID_SET で NRD 番号制約をかけて入出力を行った後で NRD 番号制約を解除して同じ種別にアクセスしても探索が行われない (あらかじめ NRD 制約をかけずに入出力操作をしていれば探索される)。この問題は NuSDaS 1.3 以降では解決されている。

6.4.6 nusdas_parameter_reset: オプションを既定値に戻す

書式

N_SI4 **nusdas_parameter_reset**(N_SI4 *param*);

引数名	引数の型	役割
param	N_SI4	設定項目コード

説明　nusdas_parameter_change (p. 98) で設定されたパラメタを既定値に戻します。

履歴　この関数 は NuSDaS 1.3 で導入されました。それ以前のバージョンでは nusdas_parameter_change (p. 98) に既定値または定数 NULL を与える方法が使われていました。

6.4.7 nusdas_inq_parameter: オプション取得

書式

N_SI4 **nusdas_inq_parameter**(N_SI4 *param*, void ∗*value*);

引数名	引数の型	役割
param	N_SI4	設定項目コード
value	void ∗	設定値

説明　nusdas_parameter_change (p. 98) の項目 *param* で設定されるパラメータの値を *value* の指す領域 (型は以下を参照) に書き込む。

N_PC_MISSING_UI1 1 バイト整数の欠損値

N_PC_MISSING_SI2 2 バイト整数の欠損値

N_PC_MISSING_SI4 4 バイト整数の欠損値

N_PC_MISSING_R4 4 バイト実数の欠損値

N_PC_MISSING_R8 8 バイト実数の欠損値

N_PC_SIZEX 4 バイト整数に x 方向強制格子サイズを与える

N_PC_SIZEY 4 バイト整数に y 方向強制格子サイズを与える

N_PC_MASK_BIT マスクビット配列を返す。この問合せは設定値が nusdas_make_mask (p. 96) で作られた場合にしか機能しない。

N_PC_PACKING 4 バイトの文字列に強制パック方式名を与える。設定されていない場合は 4 文字のスペースが書き込まれる。

N_PC_ID_SET NRD 番号制約がかかっている場合その値、かかっていない場合 -1 を与える。

N_PC_WBUFFER 4 バイト整数に書き込みバッファサイズ (実行時オプション FWBF) を与える。

N_PC_RBUFFER 4 バイト整数に読み取りバッファサイズ (実行時オプション FRBF) を与える。

終了コード

0 正常終了

-1 サポートされていないパラメタである

-2 マスクビット配列は設定されていない

-3 マスクビット配列は設定されているが長さがわからない

履歴　NuSDaS 1.3 で導入された。

6.5 問合せ関数

6.5.1 nusdas_grid: 格子情報へのアクセス

書式

N_SI4 **nusdas_grid**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, char *proj*[4], N_SI4 *gridsize*[2], float *gridinfo*[14], char *value*[4], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
proj	char [4]	投影法 3 字略号
gridsize	N_SI4 [2]	格子数
gridinfo	float [14]	投影法緒元
value	char [4]	格子点値が周囲の場を代表する方法
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　この API は、CNTL レコードに格納された格子情報 (つまり定義ファイルに書かれた格子情報) を返す。nusdas_parameter_change を使って、定義ファイルに書いた格子数から変更した場合には正しい情報が得られない。このような場合は nusdas_inq_data を使う。

gridinfo には 4 バイト単精度浮動小数点型の配列で 14 要素存在するものを指定する。

これは CNTL レコードの項番 15 〜 21 に対応する。順に基準点 X 座標、基準点 Y 座標、基準点緯度、基準点経度、X 方向格子間隔、Y 方向格子間隔、標準緯度、標準経度、第 2 標準緯度、第 2 標準経度、緯度 1、経度 1、緯度 2、経度 2 となる。

value の値については Table B.10 (p. 155) を参照。

終了コード

0 正常

-5 入出力指示が不正

履歴　この関数は NuSDaS 1.0 から実装されていた。

6.5.2 nusdas_info: INFO 記録へのアクセス

書式

N_SI4 **nusdas_info**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *group*[4], char *info*[ ], const N_SI4 ∗*bytesize*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
group	const char [4]	群名
info	char []	INFO 記録内容
bytesize	const N_SI4 *	INFO 記録のバイト数
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明

終了コード

非負 書き出した INFO のバイト数

-3 バッファが不足している

-5 入出力指示が不正

注意 NuSDaS1.1 では、バッファが不足している場合でもバッファの大きさの分だけを書き込み、そのサイズを返していたが、 NuSDaS 1.3 からはこのような場合は-3 が返る。また、INFO のサイズは NuSDaS 1.3 で新設された nusdas_inq_subcinfo で問い合わせ項目を N_INFO_NUM にすれば得ることができる。

6.5.3 nusdas_inq_cntl: データファイルの諸元問合せ

書式

N_SI4 **nusdas_inq_cntl**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 **basetime*, const char *member*[4], const N_SI4 **validtime*, N_SI4 *param*, void **data*, const N_SI4 **datasize*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
param	N_SI4	問合せ項目コード
data	void *	問合せ結果配列
datasize	const N_SI4 *	問合せ結果配列の要素数

説明 引数 *type1* から *validtime* で指定されるデータファイルに書かれた CNTL 記録について、引数 *param* で指定される問合せを行う。

N_MEMBER_NUM メンバーの個数が 4 バイト整数型変数 *data* に書かれる。

N_MEMBER_LIST データファイルに定義されたメンバー名が配列 *data* に書かれる。配列 *data* は長さ 4 文字の文字型で *N_MEMBER_NUM* 要素存在しなければならない。

N_VALIDTIME_NUM validtime の個数が 4 バイト整数型変数 *data* に書かれる。

N_VALIDTIME_LIST データファイルに定義された validtime が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_LIST2 データファイルに定義された validtime2 が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_PLANE_NUM 面の個数が 4 バイト整数型変数 *data* に書かれる。

N_PLANE_LIST データファイルに定義された面の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_PLANE_LIST2 N_PLANE_LIST と全く同じ動作である。

N_ELEMENT_NUM 要素の個数が 4 バイト整数型変数 *data* に書かれる。

N_ELEMENT_LIST データファイルに定義された要素の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_ELEMENT_NUM* 要素存在しなければならない。

N_NUSD_NBYTES NUSD レコードのサイズ (単位バイト) が 4 バイト整数型変数 *data* に書かれる。(先頭・末尾に付加されるレコード長の大きさ (4∗2 バイト) を含む)

N_NUSD_CONTENT NUSD レコードの内容を配列 *data* に格納する。配列 *data* は N_NUSD_NBYTES バイト存在しなくてはならない。(先頭・末尾に付加されるレコード長を含む)

N_CNTL_NBYTES CNTL レコードのサイズ (単位バイト) が 4 バイト整数型変数 *data* に書かれる。(先頭・末尾に付加されるレコード長の大きさ (4∗2 バイト) を含む)

N_CNTL_CONTENT CNTL レコードの内容を配列 *data* に格納する。配列 *data* は N_CNTL_NBYTES バイト存在しなくてはならない。(先頭・末尾に付加されるレコード長を含む)

N_PROJECTION 地図投影法の情報を 4 文字の文字型 *data* に格納する (記号の意味は巻末の表参照)。

N_GRID_SIZE X 方向、Y 方向の格子数がこの順序で 4 バイト整数型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_GRID_BASEPOINT 基準点の x 座標、y 座標、緯度、経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_GRID_DISTANCE X 方向、Y 方向の格子間隔がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_STAND_LATLON 標準緯度、標準経度、第 2 標準緯度、第 2 標準経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_SPARE_LATLON 緯度 1、経度 1、緯度 2、経度 2 がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。(この問い合わせは NuSDaS 1.3 で追加)

N_INDX_SIZE INDX の個数が 4 バイト整数型の変数 *data* に書かれる。(この問い合わせは NuSDaS 1.3 で追加)

N_ELEMENT_MAP データの格納が許容されているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。(この問い合わせは NuSDaS 1.3 で追加)

N_DATA_MAP データが書き込まれているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。(この問い合わせは NuSDaS 1.3 で追加)

終了コード

正 格納要素数

-1 データの配列数が不足している。

-2 データの配列が確保されていない。

-3 問い合わせ項目が不正

注意　NuSDaS1.1 以前では、同じ構造のデータセットでも N_VALIDTIME_NUM, N_VALIDTIME_LIST の問い合わせ結果が 1 つの basetime に複数の validtime を格納するか否かによって異なっていた。これは、validtime でファイルを分ける (異なる validtime のファイルが異なる) 設定ならばデータファイルには 1 つの validtime だけが書かれていたからである。しかし NuSDaS 1.3 からは定義ファイルに指定されたすべての validtime が各データファイルの validtime に格納されているので、問い合わせ結果は格納形態を問わず一定である。

6.5.4 nusdas_inq_data: データ記録の諸元問合せ

書式

N_SI4 **nusdas_inq_data**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], N_SI4 *param*, void ∗*data*, const N_SI4 ∗*nelems*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
plane	const char [6]	面
element	const char [6]	要素名
param	N_SI4	問合せ項目コード
data	void ∗	結果格納配列
nelems	const N_SI4 ∗	結果格納配列の要素数

説明　引数 *type1* から *element* までで指定されるデータ記録について引数 *query* で指定される問合せを行う。

N_DATA_QUADRUPLET 16 バイトのメモリ領域を引数に取り、 N_GRID_SIZE から N_MISSING_VALUE までの情報が返される。

N_GRID_SIZE 引数 *data* に 4 バイト整数の長さ 2 の配列を取り、そこに X, Y 方向の格子数が書かれる。

N_PC_PACKING 引数 *data* に 4 バイトの文字列を取り、そこにパック方式名称が書かれる。文字列はヌル終端されないことに注意。

N_MISSING_MODE 引数 *data* に 4 バイトの文字列を取り、そこに欠損値表現方式名が書かれる。文字列はヌル終端されないことに注意。

N_MISSING_VALUE 引数には上述 N_PC_PACKING 項目によって決まる型の変数を取り、そこにデータ記録上の欠損値が書かれる。この値は nusdas_read (p. 89) で得られる配列で用いられる欠損値とは異なることに注意。

N_DATA_EXIST 引数 *data* に 4 バイト整数型変数をとり、そこにデータの存在有無を示す値が書かれる。0 はデータの不在、1 は存在を示す。

N_DATA_NBYTES 引数 *data* に 4 バイト整数型変数をとり、そこにデータ記録のバイト数が書かれる。

N_DATA_CONTENT 引数 *data* が指すバイト列にデータ記録がそのまま書かれる。

N_RECORD_TIME 引数 *data* に 4 バイト整数型変数をとり、そこにデータ記録の作成時刻が書かれる。この問合せはデータ記録の更新確認用に用意されており、結果は大小比較だけに用いるべきもので日時等を算出すべきではない。この値は time システムコールの返す値の下位 32 ビットであり、2038 年問題の対策のためいずれ機種依存の意味を持つようになるものと思われる。

終了コード

正 格納要素数

-1 データの配列数が不足している

-2 データの配列が確保されていない

-3 問い合わせ項目が不正

履歴　この関数は pnusdas では実装はされていたが、ドキュメント化されていなかった。

6.5.5 nusdas_inq_def: データセットの諸元問合せ

書式

N_SI4 **nusdas_inq_def**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *param*, void ∗*data*, const N_SI4 ∗*datasize*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
param	const N_SI4	問合せ項目コード
data	void ∗	結果格納配列
datasize	const N_SI4 ∗	結果格納配列の要素数

説明　引数 *type1* から *type3* で指定されるデータセットの定義ファイルに書かれた内容について、引数 *param* で指定される問合せを行う。

N_MEMBER_NUM 定義ファイルに書かれたメンバーの個数が 4 バイト整数型変数 *data* に書かれる。

N_MEMBER_LIST 定義ファイルに書かれたメンバー名が配列 *data* に書かれる。配列 *data* は長さ 4 文字の文字型で *N_MEMBER_NUM* 要素存在しなければならない。

N_VALIDTIME_NUM 定義ファイルに書かれた validtime の個数が 4 バイト整数型変数 *data* に書かれる。

N_VALIDTIME_LIST 定義ファイルに書かれた validtime が配列 *data* に書かれる。配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_LIST2 定義ファイルに書かれた validtime2 が配列 *data* に書かれる。
配列 *data* は長さ 4 byte 整数型で *N_VALIDTIME_NUM* 要素存在しなければならない。

N_VALIDTIME_UNIT 定義ファイルに書かれた validtime の単位が 4 文字の文字型変数 *data* に書かれる。

N_PLANE_NUM 定義ファイルに書かれた面の個数が 4 バイト整数型変数 *data* に書かれる。

N_PLANE_LIST 定義ファイルに書かれた面の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_PLANE_LIST2 定義ファイルに書かれた面 2 の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_PLANE_NUM* 要素存在しなければならない。

N_ELEMENT_NUM 定義ファイルに書かれた要素の個数が 4 バイト整数型変数 *data* に書かれる。

N_ELEMENT_LIST 定義ファイルに書かれた要素の名前が配列 *data* に書かれる。配列 *data* は長さ 6 文字の文字型で *N_ELEMENT_NUM* 要素存在しなければならない。

N_PROJECTION　定義ファイルに書かれた地図投影法の情報を 4 文字の文字型 *data* に格納する (記号の意味は巻末の表参照)。

N_GRID_SIZE　定義ファイルに書かれた X 方向、Y 方向の格子数がこの順序で 4 バイト整数型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。

N_GRID_BASEPOINT　定義ファイルに書かれた基準点の x 座標、y 座標、緯度、経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_GRID_DISTANCE　定義ファイルに書かれた X 方向、Y 方向の格子間隔がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 2 要素存在しなくてはならない。

N_STAND_LATLON　定義ファイルに書かれた標準緯度、標準経度、第 2 標準緯度、第 2 標準経度がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_SPARE_LATLON 定義ファイルに書かれた緯度 1、経度 1、緯度 2、経度 2 がこの順序で 4 バイト単精度浮動小数点型の配列 *data* に書かれる。配列 *data* は 4 要素存在しなくてはならない。

N_INDX_SIZE 定義ファイルから算出される INDX の個数が 4 バイト整数型の変数 *data* に書かれる。(この問い合わせは NuSDaS1.3 で追加)

N_ELEMENT_MAP 定義ファイルでデータの格納が許容されているか否かが 1 or 0 によって、1 バイト整数型の配列 *data* に書かれる。配列 *data* は *N_INDX_SIZE* 要素存在しなくてはならない。*data* はメンバー、*validtime*, 面、要素をインデックスにした配列で、それぞれの順序は *N_MEMBER_LIST*, *N_VALIDTIME_LIST*, *N_PLANE_LIST*, *N_ELEMENT_LIST* の問い合わせ結果と一致する。

N_SUBC_NUM 定義ファイルに書かれた SUBC 記録の個数が 4 バイト整数型変数 *buf* に書かれる。

N_SUBC_LIST 定義ファイルに書かれた SUBC 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_SUBC_NUM* 要素存在しなければならない。

N_INFO_NUM 定義ファイルに書かれた INFO 記録の個数が 4 バイト整数型変数 *buf* に書かれる。

N_INFO_LIST 定義ファイルに書かれた INFO 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_INFO_NUM* 要素存在しなければならない。

終了コード

正 格納要素数

-1 格納配列が不足

-2 格納配列が確保されていない

-3 問い合わせが不正

履歴 この関数は NuSDaS1.0 より実装されていたが、NuSDaS1.3 で N_INDX_SIZE の問い合わせ機能が追加されている。

6.5.6 nusdas_inq_nrdbtime: データセットの基準時刻リスト取得

書式

N_SI4 **nusdas_inq_nrdbtime**(const char *type1*[8], const char *type2*[4], const char *type3*[4], N_SI4 ∗*btlist*, const N_SI4 ∗*btlistsize*, N_SI4 *pflag*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
btlist	N_SI4 ∗	基準時刻が格納される配列
btlistsize	const N_SI4 ∗	配列の要素数
pflag	N_SI4	動作過程印字フラグ

説明 種別 1〜種別 3 で指示されるデータセットに存在する基準時刻を配列 *btlist* に書き込む。引数 *pflag* に非零値を設定すると動作過程の情報を警告メッセージとして印字するようになる。

終了コード

非負 基準時刻の個数

-1 ファイル I/O エラー

-2 ファイルに管理部が存在しない

-3 ファイルのレコード長が不正

-4 ファイルあるいはディレクトリのオープンに失敗

履歴 本関数は NuSDaS 1.0 から存在した。

注意

- 配列長 *btlistsize* より多くの基準時刻が存在する場合は、配列長を越えて書き込むことはない。リターンコードと配列長を比較して、リターンコードが大きかったらその数だけ配列を確保し直して本関数を呼び直すことにより、すべてのリストを得ることができる。
- NuSDaS 1.1 までは見付かったデータセットがネットワークでなければ、それについてだけ探索が行われた。NuSDaS 1.3 からは、指定した種別にマッチするすべてのデータセットについて探索が行われる。
- 種別に対応するデータセットが見つからない場合 (たとえば種別名を間違えた場合)、返却値はゼロとなる。データセットが存在して空の場合と異なり、このとき “Can not find NUSDAS root directory for selected type1-3” “type1<...> type2<...> type3<...> NRD=...” というメッセージが標準エラー出力に表示される。NRD= の後の数値が -1 でなければ、NRD 番号を指定したために存在しているデータセットが見つからなくなっている可能性がある。

6.5.7 nusdas_inq_nrdvtime: データセットの対象時刻リスト取得

書式

N_SI4 **nusdas_inq_nrdvtime**(const char *type1*[8], const char *type2*[4], const char *type3*[4], N_SI4 ∗*vtlist*, const N_SI4 ∗*vtlistsize*, const N_SI4 ∗*basetime*, N_SI4 *pflag*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
vtlist	N_SI4 ∗	対象時刻が書かれる配列
vtlistsize	const N_SI4 ∗	配列の要素数
basetime	const N_SI4 ∗	基準時刻 (通算分)
pflag	N_SI4	動作詳細印字フラグ

説明 種別 1〜種別 3 で指示されるデータセットに基準時刻 *basetime* のもとで存在する対象時刻を配列 *vtlist* に書き込む。引数 *pflag* に非零値を設定すると動作過程の情報を警告メッセージとして印字するようになる。

終了コード

非負 対象時刻の個数

履歴 本関数は NuSDaS 1.0 から存在したがドキュメントされていなかった。

注意

- 配列長 *vtlistsize* より多くの対象時刻が存在する場合は、配列長を越えて書き込むことはない。リターンコードと配列長を比較して、リターンコードが大きかったらその数だけ配列を確保し直して本関数を呼び直すことにより、すべてのリストを得ることができる。
- 対象時刻の探索はファイルの有無または CNTL レコードによる。リスト中の対象時刻についてデータレコードが書かれていない場合もありうる。
- 基準時刻 *basetime* に -1 を指定すると、基準時刻を問わない検索になる。
- 検索にあたってメンバー名は問わない。
- NuSDaS 1.1 までは見付かったデータセットがネットワークでなければ、それについてだけ探索が行われた。NuSDaS 1.3 からは、指定した種別にマッチするすべてのデータセットについて探索が行われる。
- 種別に対応するデータセットが見つからない場合 (たとえば種別名を間違えた場合)、返却値はゼロとなる。データセットが存在して空の場合と異なり、このとき “Can not find NUSDAS root directory for selected type1-3” “type1<...> type2<...> type3<...> NRD=...” というメッセージが標準エラー出力に表示される。NRD= の後の数値が -1 でなければ、NRD 番号を指定したために存在しているデータセットが見つからなくなっている可能性がある。

6.5.8 nusdas_inq_subcinfo: SUBC/INFO の問合せ

書式

N_SI4 **nusdas_inq_subcinfo**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 *query*, const char *group*[4], void ∗*buf*, const N_SI4 *bufnelems*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻
member	const char [4]	メンバー
validtime	const N_SI4 ∗	対象時刻
query	N_SI4	問合せ項目
group	const char [4]	群名
buf	void ∗	結果格納配列
bufnelems	const N_SI4	結果格納配列の要素数

説明 引数 *type1* から *validtime* で指定されるデータファイルに書かれた SUBC または INFO 記録について、引数 *query* で指定される問合せを行う。

N_SUBC_NUM SUBC 記録の個数が 4 バイト整数型変数 *buf* に書かれる。引数 *group* は無視される。

N_SUBC_LIST データファイルに定義された SUBC 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_SUBC_NUM* 要素存在しなければならない。引数 *group* は無視される。

N_SUBC_NBYTES 群名 *group* の SUBC 記録のバイト数が 4 バイト整数型変数 *buf* に書かれる。

N_SUBC_CONTENT 群名 *group* の SUBC 記録が配列 *buf* に書かれる。上述のバイト数だけの長さを確保しておかねばならない。

N_INFO_NUM INFO 記録の個数が 4 バイト整数型変数 *buf* に書かれる。引数 *group* は無視される。

N_INFO_LIST データファイルに定義された INFO 記録の群名が配列 *buf* に書かれる。配列 *buf* は長さ 4 文字の文字型で *N_INFO_NUM* 要素存在しなければならない。引数 *group* は無視される。

N_INFO_NBYTES 群名 *group* の INFO 記録のバイト数が 4 バイト整数型変数 *buf* に書かれる。

終了コード

正 格納要素数

履歴 この関数は NuSDaS 1.3 で新設された。

注意 「レコード内容」として取得されるのは表 A.6 項番 6 と同じであり、その長さはレコード有効長から 16 を引いたものに等しい。

6.5.9 nusdas_scan_ds: データセットの一覧

書式

N_SI4 **nusdas_scan_ds**(char *type1*[8], char *type2*[4], char *type3*[4], N_SI4 ∗*nrd*);

引数名	引数の型	役割
type1	char [8]	種別 1
type2	char [4]	種別 2
type3	char [4]	種別 3
nrd	N_SI4 ∗	NRD 番号

説明　返却値が負になるまで呼出しを繰り返すと、ライブラリが認識しているデータセットの一覧が得られる。

終了コード

0 引数の配列にデータセットの情報が格納された。

-1 もうこれ以上データセットは認識されていない。

履歴　この関数は NuSDaS 1.3 で追加された。pnusdas には非公開の nusdas_list_type という関数があり類似の機能を持つ。

6.6 メタデータ用関数

6.6.1 nusdas_subc_delt: SUBC DELT へのアクセス

書式

N_SI4 **nusdas_subc_delt**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, float ∗*delt*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
delt	float ∗	DELT 数値へのポインタ
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明 モデルの時間積分間隔を補助管理情報に記録しておくものである。

終了コード

0 正常終了

-2 レコードが存在しない、または書き込まれていない。

-3 レコードサイズが不正

-5 入出力指示が不正

履歴 この関数は NuSDaS1.2 で導入された。

6.6.2 nusdas_subc_delt_preset1: SUBC DELT のデフォルト設定

書式

N_SI4 **nusdas_subc_delt_preset1**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const float ∗*delt*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
delt	const float ∗	DELT 数値へのポインタ

説明 ファイルが新たに生成される際に DELT レコードに書き込む値を設定する。DELT レコードや引数については nusdas_subc_delt を参照。

終了コード

0 正常終了

-1 定義ファイルに "DELT" が登録されていない

-2 メモリの確保に失敗した

互換性 NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

6.6.3 nusdas_subc_eta: SUBC ETA へのアクセス

書式

N_SI4 **nusdas_subc_eta**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 ∗*n_levels*, float *a*[ ], float *b*[ ], float ∗*c*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
n_levels	N_SI4 ∗	鉛直層数
a	float [ ]	係数 a
b	float [ ]	係数 b
c	float ∗	係数 c
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　鉛直座標に ETA 座標系を用いるときに、鉛直座標を定めるパラメータへのアクセスを提供する。パラメータは 4 バイト単精度浮動小数点型の配列 *a*, *b*, *c* で構成され、*a*, *b*, は鉛直層数 *n_levels* に対して、*n_levels*+1 要素の配列、*c* は 1 要素の配列 (変数) を確保する必要がある。*n_levels* は nusdas_subc_inq_nz で問い合わせることができる。入出力指示が *GET* の場合においても、*n_levels* は書込み対象変数として扱われる。特に const で宣言された変数を *n_levels* に指定してはならない。

終了コード

0 正常終了
-2 レコードが存在しない、またはレコードの書き込みがされていない。
-3 レコードサイズが不正
-4 ユーザーの鉛直層数がファイルの中の鉛直層数より小さい
-5 入出力指示が不正。

履歴　この関数は NuSDaS1.0 から存在した。NuSDaS1.1 までは、レコードが書き込まれたかの情報を持ち合わせていなかったために無記録のレコードをファイルから読んで正常終了していた。 NuSDaS 1.3 からはファイルの初期化時にレコードを初期化し、未記録を判定できるようにした。その場合のエラーは-2 としている。

注意　SUBC ETA に使われている鉛直層数 *n_levels* は実際のモデルの鉛直層数と異なっている場合があるので、配列確保の際には nusdas_subc_inq_nz で問い合わせた結果を用いること。

6.6.4 nusdas_subc_eta_inq_nz: SUBC 記録の鉛直層数問合せ

書式

N_SI4 **nusdas_subc_eta_inq_nz**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *group*[4], N_SI4 ∗*n_levels*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
group	const char [4]	群名
n_levels	N_SI4 ∗	鉛直層数

説明　SUBC レコードの ETA, SIGM, ZHYB に記録された鉛直層数を問い合わせる。群名には ”ETA ”, ”SIGM”, ”ZHYB” のいずれかを指定する。

終了コード

正 正常終了

履歴　この関数は NuSDaS1.2 で導入された。

6.6.5 nusdas_subc_preset1: SUBC ETA/SIGM のデフォルト値設定

書式

N_SI4 **nusdas_subc_preset1**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const char *group*[4], const N_SI4 ∗*n_levels*, float *a*[ ], float *b*[ ], float ∗*c*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
group	const char [4]	群名
n_levels	const N_SI4 ∗	鉛直層数
a	float [ ]	係数 a
b	float [ ]	係数 b
c	float ∗	係数 c

説明　ファイルが新たに生成される際に ETA, SIGM に書き込む値を設定する。SIGM や引数については nusdas_subc_eta を参照。引数の「群名」には、”ETA ” または ”SIGM” を指定する。

終了コード

0 正常終了

-1 定義ファイルに指定した群名が登録されていない

-2 メモリの確保に失敗した

-3 レコードのサイズが不正

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

6.6.6 nusdas_subc_rgau: SUBC RGAU へのアクセス

書式

N_SI4 **nusdas_subc_rgau**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 ∗*j*, N_SI4 ∗*j_start*, N_SI4 ∗*j_n*, N_SI4 *i*[ ], N_SI4 *i_start*[ ], N_SI4 *i_n*[ ], float *lat*[ ], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
j	N_SI4 *	全球の南北分割数
j_start	N_SI4 *	データの最北格子の番号 (1 始まり)
j_n	N_SI4 *	データの南北格子数
i	N_SI4 []	全球の東西格子数
i_start	N_SI4 []	データの最西格子の番号 (1 始まり)
i_n	N_SI4 []	データの東西格子数
lat	float []	緯度
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　Reduced Gauss 格子を使う場合の補助管理情報へのアクセスを提供する。入出力指示が *GET* の場合においても、j_n の値はセットする。特に const で宣言された変数を j_n に指定してはならない。この j_n の値は nusdas_subc_rgau_inq_jn を使って問い合わせできる。i, i_start, i_n, lat は j_n 要素をもった配列を用意する。

終了コード

0 正常終了
-2 レコードが存在しない、または書き込まれていない。
-3 サイズの情報が引数と定義ファイルで不一致
-4 指定した入力値 (j_n, j_start, j_n, i, i_start, i_n) が不正 (PUT のときのみ)
-5 入出力指示が不正
-6 指定した入力値 (j_n) が不正 (GET のときのみ)

注意　Reduced Gauss 格子を使う場合は 1 次元でデータを格納するので、定義ファイルの size(格子数) には (実際の格子数) 1 と指定する。また、SUBC のサイズは 16 * j_n + 12 を計算した値を定義ファイルに書く。

履歴　この関数は NuSDaS1.2 で実装された

6.6.7　nusdas_subc_rgau_inq_jn: SUBC RGAU 記録の大きさを問合せ

書式

N_SI4 **nusdas_subc_rgau_inq_jn**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 **basetime*, const char *member*[4], const N_SI4 **validtime*, N_SI4 **j_n*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
j_n	N_SI4 *	南北格子数

説明　RGAU に記録されている j_n (南北格子数) を問い合わせる。

終了コード

正 正常終了
-2 要求されたレコードが存在しない、または書き込まれていない。
-3 レコードのサイズが不正

履歴　この関数は NuSDaS1.2 で導入された。

6.6.8 nusdas_subc_rgau_preset1: SUBC RGAU のデフォルト値を設定

書式

N_SI4 **nusdas_subc_rgau_preset1**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*j*, const N_SI4 ∗*j_start*, const N_SI4 ∗*j_n*, const N_SI4 *i*[ ], const N_SI4 *i_start*[ ], const N_SI4 *i_n*[ ], const float *lat*[ ]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
j	const N_SI4 ∗	全球の南北分割数
j_start	const N_SI4 ∗	データの最北格子の番号 (1 始まり)
j_n	const N_SI4 ∗	データの南北格子数
i	const N_SI4 [ ]	全球の東西格子数
i_start	const N_SI4 [ ]	データの最西格子の番号 (1 始まり)
i_n	const N_SI4 [ ]	データの東西格子数
lat	const float [ ]	緯度

説明　ファイルが新たに生成される際に RGAU レコードに書き込む値を設定する。RGAU レコードや引数については nusdas_subc_rgau を参照。

終了コード

0 正常終了

-1 定義ファイルに ”RGAU” が登録されていない

-2 メモリの確保に失敗した

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

6.6.9 nusdas_subc_sigm: SUBC SIGM へのアクセス

書式

N_SI4 **nusdas_subc_sigm**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 ∗*n_levels*, float *a*[ ], float *b*[ ], float ∗*c*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
n_levels	N_SI4 ∗	鉛直層数
a	float [ ]	係数 a
b	float [ ]	係数 b
c	float ∗	係数 c
getput	const char [3]	入出力指示 (”*GET*” または ”*PUT*”)

説明　鉛直座標に ETA 座標系を用いるときに、鉛直座標を定めるパラメータへのアクセスを提供する。関数の仕様は、nusdas_subc_eta と同じである。

6.6.10　nusdas_subc_srf: 降短系 SUBC へのアクセス

書式

N_SI4 **nusdas_subc_srf**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, const char *plane*[6], const char *element*[6], const char *group*[4], N_SI4 ∗*data*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
plane	const char [6]	面
element	const char [6]	要素名
group	const char [4]	群名
data	N_SI4 ∗	データ配列
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　降水短時間予報系のデータの補助管理部へのアクセスを提供する。群名には次のもののいずれかを指定する。

ISPC レーダーや雨量計の運用情報、レベル値変換テーブルが格納される。data には 128 要素の 4 バイト整数型配列を用意する。内部のフォーマットは 4 バイト整数型であることは関係ないが、バイトオーダーの変換はされるので注意が必要。

THUN 詳細未詳。data には 4 バイト整数型変数を用意する。

RADR レーダー観測に関する情報。data には 4 バイト整数型変数を用意する。

RADS レーダー観測に関する情報。data には 6 要素の 4 バイト整数型配列を用意する。

DPRD ドップラーレーダー観測に関する情報。data には 8 要素の 4 バイト整数型配列を用意する。

終了コード

0 正常終了

-2 要求されたレコードが存在しない、または書かれていない。

-3 レコードサイズが不正

-4 群名が不正

-5 入出力指示が不正

履歴　この関数は NuSDaS1.0 から存在した。

6.6.11　nusdas_subc_srf_ship: SUBC LOCA へのアクセス

書式

N_SI4 **nusdas_subc_srf_ship**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 ∗*lat*, N_SI4 ∗*lon*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
lat	N_SI4 *	緯度
lon	N_SI4 *	経度
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　船レーダーの観測データに関する補助管理情報 (緯度、経度) へのアクセスを提供する。

終了コード

0 正常終了

-2 要求されたレコードが存在しない、または書かれていない。

-3 レコードサイズが不正

-5 入出力指示が不正

履歴　この関数は NuSDaS1.0 から存在したが、ドキュメントされていなかった。

6.6.12　nusdas_subc_tdif: SUBC TDIF へのアクセス

書式

N_SI4 **nusdas_subc_tdif**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime, N_SI4 *diff_time, N_SI4 *total_sec, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 *	対象時刻 (通算分)
diff_time	N_SI4 *	対象時刻からのずれ (秒)
total_sec	N_SI4 *	総予報時間 (秒)
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　格納した値の時刻の対象時間とのずれ、積算時間を格納する補助管理部 TDIF へのアクセスを提供する。

終了コード

0 正常終了

-2 要求されたレコードが存在しない、または書き込まれていない。

-3 レコードサイズが不正

-5 入出力指示が不正

補足

- diff_time = 時間範囲始点 - 対象時刻 [秒単位]
- total_sec = 時間範囲終点 - 時間範囲始点 [秒単位]

履歴　この関数は NuSDaS1.0 から存在した。

6.6.13　nusdas_subc_zhyb: SUBC ZHYB へのアクセス

書式

N_SI4 **nusdas_subc_zhyb**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime*, N_SI4 ∗*nz*, float ∗*ptrf*, float ∗*presrf*, float *zrp*[ ], float *zrw*[ ], float *vctrans_p*[ ], float *vctrans_w*[ ], float *dvtrans_p*[ ], float *dvtrans_w*[ ], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime	const N_SI4 ∗	対象時刻 (通算分)
nz	N_SI4 ∗	鉛直層数
ptrf	float ∗	温位の参照値
presrf	float ∗	気圧の参照値
zrp	float [ ]	モデル面高度 (フルレベル)
zrw	float [ ]	モデル面高度 (ハーフレベル)
vctrans_p	float [ ]	座標変換関数 (フルレベル)
vctrans_w	float [ ]	座標変換関数 (ハーフレベル)
dvtrans_p	float [ ]	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	float [ ]	座標変換関数の鉛直微分 (ハーフレベル)
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

説明　鉛直座標に鉛直ハイブリッド座標をを使う場合の補助管理情報 ZHYB へのアクセスを提供する。入出力指示が *GET* の場合においても、nz の値をセットする。特に const で宣言された変数を nz に指定してはならない。この nz の値は nusdas_subc_eta_inq_nz を使って問い合わせできる。zrp, zrw, vctrans_p, vctrans_w, dvtrans_p, dvtrans_w は nz 要素をもった配列を用意する。

終了コード

0 正常終了

-2 レコードが存在しない、または書き込まれていない。

-3 サイズの情報が引数と定義ファイルで不一致

-4 指定した入力値 (ptrf, presrf) が不正 (PUT のときのみ)

-5 入出力指示が不正

-6 指定した入力値 (nz) が不正 (GET のときのみ)

注意　SUBC のサイズは 24 ∗ nz + 12 を計算した値を定義ファイルに書く。

履歴　この関数は NuSDaS1.2 で実装された

6.6.14　nusdas_subc_zhyb_preset1: SUBC ZHYB のデフォルト値を設定

書式

N_SI4 **nusdas_subc_zhyb_preset1**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*nz*, const float ∗*ptrf*, const float ∗*presrf*, const float *zrp*[ ], const float *zrw*[ ], const float *vctrans_p*[ ], const float *vctrans_w*[ ], const float *dvtrans_p*[ ], const float *dvtrans_w*[ ]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
nz	const N_SI4 *	鉛直層数
ptrf	const float *	温位の参照値
presrf	const float *	気圧の参照値
zrp	const float []	モデル面高度 (フルレベル)
zrw	const float []	モデル面高度 (ハーフレベル)
vctrans_p	const float []	座標変換関数 (フルレベル)
vctrans_w	const float []	座標変換関数 (ハーフレベル)
dvtrans_p	const float []	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	const float []	座標変換関数の鉛直微分 (ハーフレベル)

説明　ファイルが新たに生成される際に ZHYB レコードに書き込む値を設定する。ZHYB レコードや引数については nusdas_subc_zhyb を参照。

終了コード

0 正常終了

-1 定義ファイルに "ZHYB" が登録されていない

-2 メモリの確保に失敗した

互換性　NuSDaS1.1 では、一つの NuSDaS データセットに設定できる補助管理部の数は最大 10 に制限されており、それを超えると-2 が返された。一方、 NuSDaS 1.3 からはメモリが確保できる限り数に制限はなく、-2 をメモリ確保失敗のエラーコードに読み替えている。

6.7 サービスサブルーチン関数

6.7.1 bfopen: ファイルを開く

書式

N_BIGFILE **bfopen(const char *pathname, const char *mode)*;

引数名	引数の型	役割
pathname	const char *	ファイル名
mode	const char *	モード指定

説明 ファイルを開く。OS がサポートしていれば 32 ビット環境でも 2GB または 4GB を超えるラージファイルを開くことができる。

終了コード

NULL 失敗
他 成功。このポインタを今後のファイル操作に用いる

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.2 bfclose: ファイルを閉じる

書式

int **bfclose**(N_BIGFILE **bf*);

引数名	引数の型	役割
bf	N_BIGFILE *	ファイル

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* を閉じる。これ以後ポインタ *bf* を参照してはならない。

終了コード

0 正常終了
-1 エラー

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.3 bfread: ファイル入力

書式

unsigned long **bfread**(void **ptr*, unsigned long *nbytes*, N_BIGFILE **bf*);

引数名	引数の型	役割
ptr	void *	読みだし先バッファ
nbytes	unsigned long	バイト数
bf	N_BIGFILE *	ファイル

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* から *ptr* が指すバッファに *nbytes* バイト読み出す。

終了コード

正 読み出されたバイト数 (ファイル末尾などでは *nbytes* より少ない)

0 ちょうどファイル末尾から読み出そうとしたか、エラー

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.4 bfwrite: ファイル出力

書式

unsigned long **bfwrite**(void ∗*ptr*, unsigned long *nbytes*, N_BIGFILE ∗*bf*);

引数名	引数の型	役割
ptr	void ∗	書き込み元バッファ
nbytes	unsigned long	バイト数
bf	N_BIGFILE ∗	ファイル

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* に対して *ptr* が指すバッファから *nbytes* バイト書き出す。

終了コード

正 書き込まれたバイト数 (エラー時に *nbytes* より少ないことがある)

0 ちょうど書き込み開始時にエラーが起こった

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.5 bfread_native: バイトオーダー変換付きファイル入力

書式

unsigned long **bfread_native**(void ∗*ptr*, unsigned long *size*, unsigned long *nmemb*, N_BIGFILE ∗*bf*);

引数名	引数の型	役割
ptr	void ∗	読出し先バッファ
size	unsigned long	オブジェクトの幅
nmemb	unsigned long	オブジェクトの個数
bf	N_BIGFILE ∗	ファイル

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* から幅 @size バイトのオブジェクトを *nmemb* 個読出す。ファイルにはビッグエンディアンで書かれていることが仮定され、結果は機械に自然なバイトオーダーで *ptr* に書き出される。

終了コード

正 読み込まれたオブジェクト数 (エラー時に nmemb より少ないことがある)

0 ちょうどファイル末尾から読出そうとしたか、エラー

参考

- 引数 *size* にふさわしい値は sizeof 演算子によって得られる。

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.6 bfwrite_native: バイトオーダー変換付きファイル出力

書式

unsigned long **bfwrite_native**(void **ptr*, unsigned long *size*, unsigned long *nmemb*, N_BIGFILE **bf*);

引数名	引数の型	役割
ptr	void *	データ
size	unsigned long	オブジェクト長
nmemb	unsigned long	オブジェクト数
bf	N_BIGFILE *	ファイル

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* に幅 @size バイトのオブジェクトを *nmemb* 個書き込む。書き込むデータは機械に自然なバイトオーダーで *ptr* から読み込まれ、ファイルにはビッグエンディアンで書かれる。

終了コード

正 書き出されたオブジェクト数 (エラー時に nmemb より少ないことがある)

0 エラー

参考

- 引数 *size* にふさわしい値は sizeof 演算子によって得られる。

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.7 bfgetpos: ファイル位置取得

書式

int **bfgetpos**(N_BIGFILE **bf*, N_SI8 **pos*);

引数名	引数の型	役割
bf	N_BIGFILE *	ファイル
pos	N_SI8 *	位置

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* の現在位置を *pos* に書き出す。

終了コード

0 正常終了

-1 エラー

注意

- 64 ビット整数がないコンパイラ用に configure した場合 N_SI8 は構造体であり算術演算に用いることはできない。

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.8 bfsetpos: ファイル位置設定

書式

int **bfsetpos**(N_BIGFILE ∗*bf*, N_SI8 *pos*);

引数名	引数の型	役割
bf	N_BIGFILE ∗	ファイル
pos	N_SI8	位置

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* の現在位置をあらかじめ bfgetpos (p. 120) で得られた位置 *pos* に設定する。

終了コード

0 正常終了

-1 エラー

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.9 bfseek: ファイル位置設定

書式

int **bfseek**(N_BIGFILE ∗*bf*, long *offset*, int *whence*);

引数名	引数の型	役割
bf	N_BIGFILE ∗	ファイル
offset	long	相対位置
whence	int	起点

説明 あらかじめ bfopen (p. 118) で開かれたファイル *bf* の現在位置を設定する。*offset* は本来 off_t 型が望ましいのだろうが、long 型で実装されているので注意。

ファイル位置の起算原点は *whence* によって異なる。

SEEK_SET ファイル先頭から *offset* バイト (非負) 進んだ位置

SEEK_CUR 現在位置から *offset* バイト (負でもよい) 進んだ位置

SEEK_END ファイル末尾から *offset* バイト進んだ位置

終了コード

0 正常終了

-1 エラー

注意

- long が 32 ビット幅の場合、2 ギガバイトを超えるファイルでは指定できない場所がある。
- *whence* に SEEK_END を指定して正の *offset* を指定した場合の挙動については OS の lseek(2) 等のマニュアルを参照されたい。

履歴 この関数は NuSDaS 1.3 で追加された。

6.7.10 endian_swab2: 2 バイト整数のバイトオーダー変換

書式

void **endian_swab2**(void ∗*ary*, const N_UI4 *count*);

引数名	引数の型	役割
ary	void ∗	配列
count	const N_UI4	配列の要素数

説明 リトルエンディアン機では、2 バイト整数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

6.7.11 endian_swab4: 4 バイト整数のバイトオーダー変換

書式

void **endian_swab4**(void ∗*ary*, const N_UI4 *count*);

引数名	引数の型	役割
ary	void ∗	配列
count	const N_UI4	配列の要素数

説明 リトルエンディアン機では、4 バイト整数または実数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

6.7.12 endian_swab8: 8 バイト整数のバイトオーダー変換

書式

void **endian_swab8**(void ∗*ary*, const N_UI4 *count*);

引数名	引数の型	役割
ary	void ∗	配列
count	const N_UI4	配列の要素数

説明 リトルエンディアン機では、8 バイト整数または実数の配列 *ary* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

6.7.13 endian_swab_fmt: 任意構造のバイトオーダー変換

書式

void **endian_swab_fmt**(void ∗*ptr*, const char ∗*fmt*);

引数名	引数の型	役割
ptr	void ∗	変換対象
fmt	const char ∗	書式

説明　リトルエンディアン機では、さまざまな長さのデータが混在するメモリ領域 *ptr* のバイトオーダーを逆順にする。ビッグエンディアンのデータを読んだ後整数として解釈する前、または整数として値を格納した後ビッグエンディアンで書き出す前に呼ぶ。

ビッグエンディアン機ではなにもしない。

メモリのレイアウトは文字列 *fmt* で指定される。文字列は以下に示す型を表わす文字の羅列である。

D, d, L, l 8 バイト

F, f, I, i 4 バイト

H, h 2 バイト

B, b, N, n 1 バイト (なにもしない)

文字の前に数字をつけると繰り返し数をあらわす。たとえば "`4c8i`" は最初の 4 バイトが無変換、次に 4 バイト単位で 8 個変換を行うことを示す。

注意

- 数字は strtoul(3) で解釈しているので十進だけではなく八進や十六進も使える。たとえば "`0xFFi`" は 4 バイト単位で 255 個変換することを示し、"`0100h`" は 2 バイト単位で 64 個変換することを示す。

履歴　本関数は pnusdas から存在し、NuSDaS 1.3 で Fortran ラッパーを伴うサービスサブルーチンとしてドキュメントされた。

6.7.14　nusdas_gunzip: gzip 圧縮データを展開

書式

N_SI4 **nusdas_gunzip**(const void ∗*in_data*, N_UI4 *in_nbytes*, void ∗*out_buf*, N_UI4 *out_nbytes*);

引数名	引数の型	役割
in_data	const void ∗	圧縮データ
in_nbytes	N_UI4	圧縮データのバイト数
out_buf	void ∗	展開結果を格納する領域
out_nbytes	N_UI4	結果領域のバイト数

説明　入力データ *in_data* を gzip 展開して *out_buf* に格納する。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。

-99 入力は gzip 圧縮形式ではない。

-5 展開結果の長さが圧縮データと不整合。

-4 結果領域の長さ *out_nbytes* が不足している。

-3 展開結果の CRC32 が圧縮データと不整合。

-2 ZLib の inflateInit 関数がエラーを起こした。

-1 ZLib の inflate 関数がエラーを起こした。

他 展開データのバイト数

履歴　本関数は NuSDaS 1.3 で新設された。

6.7.15　nusdas_gunzip_nbytes: gzip 圧縮データの展開後の長さを得る

書式

N_SI4 **nusdas_gunzip_nbytes**(const void ∗*in_data*, N_UI4 *in_nbytes*);

引数名	引数の型	役割
in_data	const void ∗	圧縮データ
in_nbytes	N_UI4	圧縮データのバイト数

説明　入力データ *in_data* を gzip 展開するときに必要となる結果格納領域のバイト数を返す。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。
正 展開後の長さ

履歴　本関数は NuSDaS 1.3 で新設された。

6.7.16　nusdas_gzip: gzip 圧縮

書式

N_SI4 **nusdas_gzip**(const void ∗*in_data*, N_UI4 *in_nbytes*, void ∗*out_buf*, N_UI4 *out_nbytes*);

引数名	引数の型	役割
in_data	const void ∗	入力データ
in_nbytes	N_UI4	入力データのバイト数
out_buf	void ∗	圧縮結果を格納する領域
out_nbytes	N_UI4	結果領域のバイト数

説明　入力データ *in_data* を gzip 圧縮して *out_buf* に格納する。

終了コード

-98 NuSDaS が ZLib を使うように設定されていない。
-9 ZLib の deflateEnd 関数がエラーを起こした。
-4 結果領域の長さ *out_nbytes* が不足している。
-2 ZLib の deflateInit2 関数がエラーを起こした。
-1 ZLib の deflate 関数がエラーを起こした。
他 圧縮データの長さ

履歴　本関数は NuSDaS 1.3 で新設された。

6.7.17　nusdas_snprintf: 固定バイト数対応 snprintf()

書式

int **nusdas_snprintf**(char ∗*buf*, unsigned *bufsize*, const char ∗*fmt*, ...);

引数名	引数の型	役割
buf	char ∗	結果格納配列
bufsize	unsigned	結果配列サイズ
fmt	const char ∗	書式
...	任意	印字すべき値

説明　書式 *fmt* に従いそれ以降の引数を文字列化し、長さ *bufsize* の配列 *buf* に格納する。printf(3) と同様、変換指定は '%' 文字、フラグ (オプション)、数字列 (1–9) による印字幅 (オプション)、ピリオド ('.') を前置した数字列による精度 (オプション)、型指定 (オプション)、変換指定文字からなる。変換指定以外の *fmt* 内の文字はそのまま転写される。

終了コード

正 実際に書き込まれた文字数 (ヌル終端を含まない)
-1 エラー

フラグ

0 (ゼロ) 数値変換 (‘a’, ‘d’, ‘b’, ‘x’, ‘X’, ‘u’, ‘o’) の結果が印字幅に満たない場合、左側にゼロを埋める。

− 変換後の文字列を印字幅内で左寄せする。

+ 変換指定 ‘d’ または ‘a’ について、数値が正のとき ‘+’ 記号を省略しない。

代替的書式を使用する。

印字幅

- 印字幅は、変換の結果埋め込まれる文字が占める最小の字数を指定する。
- 変換結果が印字幅に満たない場合、左側にスペースが補われる (前項フラグによって挙動を変えられる)。
- 変換結果が印字幅を超える場合、そのまま用いられる。つまり、充分な印字幅を与えないと書式が崩れることがある。
- 型指定 ‘y’, ‘m’ または変換指定 ‘%’, ‘c’ では印字幅は効果をもたない。

精度

- 浮動小数点数変換 (‘a’) については、精度が小数部の占める桁数となる。
- 浮動小数点数変換 (‘g’) については、精度が仮数部の占める桁数となる。
- 文字列変換 (‘s’) については、変換で埋め込まれる文字は最大で (精度) 個である。Fortran から渡されたデータのようにヌル終端されていない固定長文字列の印字に用いられるが、ヌル文字があればそこで止まってしまう (例外あり後述)。
- その他の変換指定においては精度は効果をもたない。

型指定　適用可能な変換指定に付いては個々の変換指定を見よ。

l (小文字のエル) 引数は long 型または unsigned long 型である。

z 引数は size_t 型である (C99 準拠)。

P 引数は N_SI4 型または N_UI4 型である。

Q 引数は N_SI8 型または N_UI8 型である。これらの型が構造体として実装されている環境でも、アプリケーションプログラムは構造体へのポインタではなく構造体そのものを引数に渡すこと。

y 引数は struct nustype_t (ライブラリ内部の非公開構造型) へのポインタである。‘s’ 変換指定にだけ使用できて、結局のところ `%ys` はあたかも `%Qs%Ps%Ps` であるかのように種別 1, 種別 2, 種別 3 を連結して印字する。また `%#ys` はあたかも `%Qs.%Ps.%Ps` であるかのようにピリオド区切りで印字する (パンドラ準拠)。このとき印字幅と精度は効果をもたない。

m 引数は struct nusdims_t (ライブラリ内部の非公開構造型) へのポインタである。‘s’ 変換指定にだけ使用できて、結局のところ `%ms` はあたかも`%12PT/%4.4s/%12PT/%6.6s/%6.6s` であるかのようにスラッシュ区切りで基準時刻、メンバ名、対象時刻 1, 面 1, 要素名を印字する。メンバ名がすべてスペースのときは “`none`” に置換される。また `%#ms` はあたかも`%#15PT/%4.4s/%#15PT/%Ps/%Ps` であるかのように時間に区切り文字を入れるとともに空白を詰める (パンドラ準拠)。このとき印字幅と精度は効果をもたない。

変換指定

% パーセント記号 ‘%’ そのものを印字する。引数は使わない。

d 符号付き整数型引数を十進表記する。型指定は ‘Q’, ‘P’, ‘l’, ‘z’ を認識する。さもなくば引数は int とみなされる。

b 符号無し整数型引数を二進表記する。型指定は ‘Q’, ‘P’, ‘l’, ‘z’ を認識する。さもなくば引数は unsigned とみなされる。

x 符号無し整数型引数を十六進表記する (英字は小文字を用いる)。型指定は ‘Q’, ‘P’, ‘l’, ‘z’ を認識する。さもなくば引数は unsigned とみなされる。

X 符号無し整数型引数を十六進表記する (英字は大文字を用いる)。型指定は 'Q', 'P', 'l', 'z' を認識する。さもなくば引数は unsigned とみなされる。

u 符号無し整数型引数を十進表記する。型指定は 'Q', 'P', 'l', 'z' を認識する。さもなくば引数は unsigned とみなされる。

o 符号無し整数型引数を八進表記する。型指定は 'Q', 'P', 'l', 'z' を認識する。さもなくば引数は unsigned とみなされる。

p ポインタを整数にキャストしたものを十六進表記する。AIX の 64 ビットモードではシステムの printf(3) が不当にも下位 32 ビットのみを表示するのに対して本関数は正しく 64 ビットを表示できる。

T 符号付き整数型引数を数値予報通算分とみなして印字する。'#' フラグを指定すると年月日をハイフン ('-') で区切り、日と時の間に 't' 文字を挿入する (パンドラ準拠)。型指定は 'Q', 'P', 'l', 'z' を認識する。'Q' を指定するとあたかも `%PT/%PT` であるかのように上位・下位 32 ビットの数値を 2 つの時刻として印字する。

a 浮動小数点数を十六進指数表記する (C99 準拠)。計算機の浮動小数点形式にかかわらず IEEE 形式による。

g 浮動小数点数を十進指数表記する。指数部が -4 から (精度 - 1) のときは固定小数点表記となる。省略時は精度 6 である。いずれにしても仮数部末尾のゼロは印字されず、小数部がゼロなら小数点も印字されない。

c 引数を unsigned 整数値とみなしてその文字コードをもつ文字を印字する。印字幅は効果をもたない。

s 引数を文字列として評価して印字する。型指定がない場合は引数はヌル終端の char * とみなされる。型指定が 'Q' または 'P' のときは整数値を表わす 8 バイトまたは 4 バイトのバイト列を文字列としてそのまま (バイトオーダー変換等せずに) 解釈したものである。このときヌル文字は終端ではなく、末尾に任意個のスペースがある場合はあたかも文字列終端であるかのように扱われる。型指定 'y' と 'm' については前項参照。

注意

- 配列長 *bufsize* が不足する場合エラーとなる。これによってバッファオーバーフローを安全に避けることができる。
- 標準関数 sprintf(3) の書式の一部はサポートされていない。
- 文字列 (s) 変換指定で ASCII 図形文字以外のバイナリを印字しようとすると`\0`, `\t`, `\n`, `\r` あるいは`\377` のような八進表記で印字される。ために `%.8s` のように精度を指定すると 8 バイトすべて印字されないことがある。
- バグ: ゼロフラグは数値の変換結果に符号が付く場合に対応していない (NuSDaS 内部では使っていない)。

履歴 この関数は NuSDaS 1.3 で導入された。

6.7.18 nusdas_unpack: 生 DATA レコードの解読

書式

N_SI4 **nusdas_unpack**(const void **pdata*, void **udata*, const char *utype*[2], N_SI4 *usize*);

引数名	引数の型	役割
pdata	const void *	パックされたバイト列
udata	void *	展開先配列
utype	const char [2]	展開する型
usize	N_SI4	展開先配列の要素数

説明 nusdas_inq_data (p. 103) の問い合わせ N_DATA_CONTENT で得られるバイト列を解読して数値配列を得る。パッキング型 2UPJ では利用できない。

終了コード

正 正常終了、値は要素数

-4 展開先の大きさ *usize* がデータレコードの要素数より少ない

-5 パッキング型・欠損値型・展開型の組合せが不適

-6 利用できないパッキング型が与えられた

履歴 本関数は NuSDaS 1.3 で追加された。エラーコード -6 は NuSDaS 1.4 で新設されたもので、それ以前はエラーチェックがなされていなかった。

6.7.19 nusdas_uncpsd: 2UPP を 2UPC に展開する

書式

N_SI4 **nusdas_uncpsd**(const void ∗*pdata*, void ∗*cdata*, N_SI4 *csize*);

引数名	引数の型	役割
pdata	const void ∗	入力する 2UPP データ
udata	void ∗	展開先配列
usize	N_SI4	展開先配列のバイト数

説明 nusdas_inq_data (p. 103) の問い合わせ N_DATA_CONTENT で得られる 2UPP のバイト列を展開して 2UPC のバイト列として得ます。結果として返るバイト列は 2UPC 形式のデータに対して nusdas_inq_data (p. 103) の問い合わせ N_DATA_CONTENT で得られるデータと同じ形式です。

終了コード

正 正常終了、値は展開後のバイト数

-4 展開先配列の大きさ *csize* が必要バイト数より少ない

-5 入力データが 2UPP ではない

-6 2UPP から 2UPC への展開時にエラーが発生

履歴 本関数は NuSDaS 1.4 で追加された。

6.7.20 nusdas_uncpsd_nbytes: 2UPC 展開後の長さを得る

書式

N_SI4 **nusdas_uncpsd_nbytes**(const void ∗*pdata*);

引数名	引数の型	役割
pdata	const void ∗	入力する 2UPP データ

説明 入力データ *pdata* を nusdas_uncpsd (p. 127) で展開した後の展開後バイト数を得る。

終了コード

正 正常終了、値は展開後のバイト数

-5 入力データが 2UPP ではない

履歴 本関数は NuSDaS 1.4 で追加された。

6.7.21 n_decode_rlen_nbit_I1: RLE データを展開する

書式

N_SI4 **n_decode_rlen_nbit_I1**(unsigned char *udata*[ ], const unsigned char *compressed_data*[ ], N_SI4 *compressed_nbytes*, N_SI4 *udata_nelems*, N_SI4 *maxvalue*, N_SI4 *nbit*);

引数名	引数の型	役割
udata	unsigned char [ ]	結果格納配列
compressed_data	const unsigned char [ ]	圧縮データ
compressed_nbytes	N_SI4	圧縮データのバイト数
udata_nelems	N_SI4	圧縮データの要素数
maxvalue	N_SI4	データの最大値
nbit	N_SI4	圧縮データのビット数

説明

履歴 この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

6.7.22 n_encode_rlen_8bit: 4 バイト整数を RLE 圧縮する

書式

N_SI4 **n_encode_rlen_8bit**(const N_SI4 *udata*[ ], unsigned char *compressed_data*[ ], N_SI4 *udata_nelems*, N_SI4 *max_compress_nbytes*, N_SI4 ∗*maxvalue*);

引数名	引数の型	役割
udata	const N_SI4 [ ]	元データ配列
compressed_data	unsigned char [ ]	結果格納配列
udata_nelems	N_SI4	元データの要素数
max_compress_nbytes	N_SI4	結果配列のバイト数
maxvalue	N_SI4 ∗	元データの最大値

説明

履歴 この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

6.7.23 n_encode_rlen_8bit_I1: 1 バイト整数を RLE 圧縮する

書式

N_SI4 **n_encode_rlen_8bit_I1**(const unsigned char *udata*[ ], unsigned char *compressed_data*[ ], N_SI4 *udata_nelems*, N_SI4 *max_compress_nbytes*, N_SI4 ∗*maxvalue*);

引数名	引数の型	役割
udata	const unsigned char [ ]	元データ配列
compressed_data	unsigned char [ ]	結果格納配列
udata_nelems	N_SI4	元データの要素数
max_compress_nbytes	N_SI4	結果配列のバイト数
maxvalue	N_SI4 ∗	元データの最大値

説明

履歴 この関数は NuSDaS 1.0 から存在するが、ドキュメントされていなかった。NuSDaS 1.3 から Fortran API を伴うサービスサブルーチンとして採録された。

6.8 降水短時間ライブラリ

6.8.1 概要

降水短時間ルーチンに関連するアメダスデータ・レーダーデータに特有な処理のための関数をまとめたものが降水短時間ライブラリ libsrf.a として提供される。ライブラリ全体の関数プロトタイプを与えるヘッダは存在しない(注 1)が、srf_amd_rdic (p. 130) 及び srf_search_amdstn (p. 133) を呼ぶ時は SRF_AMD_SINFO 型やマクロの定義を参照するため<srf_amedas.h> をインクルードする必要がある。

6.8.2 rdr_lv_trans: レベル値から代表値への変換

書式

int **rdr_lv_trans**(N_SI4 *idat*[ ], float *fdat*[ ], int *dnum*, const char ∗*param*);

引数名	引数の型	役割
idat	N_SI4 [ ]	入力データ
fdat	float [ ]	結果格納配列
dnum	int	データ要素数
param	const char ∗	テーブル名

説明 配列 *idat* のレベル値を代表値 *fdat* に変換する。変換テーブルとしてファイル ./SRF_LV_TABLE/param.ltb を読む。ここで *param* は変換テーブル名 (最長 4 字) である。

終了コード

-1 変換テーブルを開くことができない

-2 変換テーブルに 256 以上のレベルが指定されている

非負 変換に成功。返却値は不明値以外となったデータの要素数

注

- 不明値は -1 となる。
- NAPS8 では変換テーブルとして/grpK/nwp/Open/Const/Vsrf/Comm/lvtbl.txd 以下に her ie2 ier kor pft pi10 pm2 pmf pr2 prr rr60 sr1 sr2 sr3 srf srj srr yar yrr が置かれている。ルーチンジョブではこのディレクトリへシンボリックリンク SRF_LV_TABLE を張って利用する。
- 上記変換テーブルのうち、pi10 と rr60 は 1 行にレベル値と代表値の 2 列が書かれており、その他はレベル値、最小値、代表値の 3 列が書かれているが、本サブルーチンはどちらにも対応している。

履歴 この関数は NAPS7 時代には存在しなかったようである。レーダー情報作成装置に関連して開発されたと考えられているが、NuSDaS 1.3 以前にはきちんとメンテナンスされていなかった。

6.8.3 srf_amd_aqc: AQC のパックを展開

書式

void **srf_amd_aqc**(const N_UI2 *aqc_in*[ ], int *num*, N_SI2 *aqc_out*[ ], const char ∗*param*);

引数名	引数の型	役割
aqc_in	const N_UI2 [ ]	AQC 配列
num	int	配列要素数
aqc_out	N_SI2 [ ]	結果配列
param	const char ∗	要素名

(注 1) このため IBM 系環境では Fortran ラッパーが提供できない。

説明　アメダス デコード データセットに含まれる AQC から要素名 *param* で指定される各ビットフィールドを取り出す。

UNYOU △ 入電・休止・運用情報 (-1:休止, 0:入電無し, 正:入電回数)

RRfr0 △ 降水量の情報 (0:入電無し, 1:ハードエラー・欠測・休止, 2:AQC 該当値, 3:正常値; 以下同じ)

SSfr0 △ 日照時間の情報

T △△△△△ 気温の情報

WindD △ 風向の情報

WindS △ 風速の情報

SnowD △ 積雪深の情報

注意　要素名が不正な場合は警告後なにもせず終了する。(NuSDaS 1.3 より前は不定動作)

履歴　この関数は NAPS7 時代から存在した。

6.8.4 srf_amd_rdic: アメダス地点辞書の読み込み

書式

int **srf_amd_rdic**(SRF_AMD_SINFO ∗*amd*, int *amdnum*, int *btime*, int *amd_type*);

引数名	引数の型	役割
amd	SRF_AMD_SINFO ∗	地点辞書格納配列
amdnum	int	地点辞書配列長
btime	int	探索日時 (通算分)
amd_type	int	地点種別

説明　環境変数 NWP_AMDDCD_STDICT が指す地点辞書ファイル (環境変数未定義時は amddic.txt となる) から SRF_AMD_SINFO 構造型の配列 *amd* にアメダス地点情報を読出す。読出される地点は時刻 *btime* に存在するものが選ばれ、さらに引数 *amd_type* によって次のように限定される。配列長 *amdnum* を越えて書き出すことはない。

SRF_KANS 官署

SRF_ELM4 4 要素を観測している地点

SRF_AMEL ロボット雨量計

SRF_AIRP 航空官署

SRF_YUKI 積雪観測地点

SRF_ALL 全地点

終了コード

非負 地点数

-1 地点種別が不正

-2 結果格納配列の長さ不足

-3 地点辞書ファイルが開けない

参考　NAPS8 においては地点辞書は/grpK/nwp/Open/Const/Pre/Dcd/amddic.txt に置かれている。

履歴　この関数は NAPS7 時代から存在した。

6.8.5 srf_amd_slct: アメダスデータを指定の地点番号順に並べる

書式

int **srf_amd_slct**(const N_SI4 ∗*st_r*, int *n_st_r*, void ∗*d_r*, const char ∗*t_r*, N_SI4 ∗*st_n*, int *n_st_n*, void ∗*d_n*, const char ∗*t_n*, int *sort_f*);

引数名	引数の型	役割
st_r	const N_SI4 ∗	結果の順を指示する地点番号表
n_st_r	int	結果配列長
d_r	void ∗	結果配列
t_r	const char ∗	結果配列の型
st_n	N_SI4 ∗	元データ地点番号配列
n_st_n	int	元データ配列長
d_n	void ∗	元データ配列
t_n	const char ∗	元データ配列の型
sort_f	int	未ソートフラグ

説明　長さ *n_st_n* の地点番号配列 *st_n* と対応する順に並んだ *t_n* 型の配列 *d_n* から、別の地点番号配列 *st_r* (要素数 *n_st_r* 個) の順に並んだ *t_r* 型の配列 *d_r* (要素数 *n_st_r* 個) を作る。

配列 *st_n* と *d_n* があらかじめソートされている場合 *sort_f* に N_OFF (nusdas.h で定義される) を指定する。そうでない場合 *sort_f* に N_ON を指定するとソートされる。

終了コード

0 配列 *st_r* の全地点が見付かった

正 みつからなかった地点数

- 型は nusdas.h で定義される N_R4, N_I4, N_I2 のいずれかで指定する。
- 配列 *st_r* に含まれる地点番号が *st_n* で見付からない場合は nusdas.h で定義される欠損値 N_MV_R4, N_MV_SI4, N_MV_SI2 が入る。

履歴　この関数は NAPS7 時代から存在した。

6.8.6 srf_lv_set: 実数からレベル値への変換

書式

int **srf_lv_set**(N_SI4 *idat*[ ], const float *fdat*[ ], int *dnum*, N_SI4 *ispec*[ ], const char ∗*param*);

引数名	引数の型	役割
idat	N_SI4 [ ]	レベル値格納配列
fdat	const float [ ]	変換元データ配列
dnum	int	データ配列要素数
ispec	N_SI4 [ ]	新 ISPC
param	const char ∗	変換テーブル名

説明　配列 *fdat* の実数データをレベル値 *idat* に変換し、ISPC 配列 *ispec* にレベル代表値をセットする。変換テーブルとしてファイル ./SRF_LV_TABLE/param.ltb を読む。ここで *param* は変換テーブル名 (最長 4 字) である。

終了コード

-1 変換テーブルを開くことができない

-2 変換テーブルに 256 以上のレベルが指定されている

正 変換に成功した。返却値はレベル数

注

- NAPS7 では変換テーブルとして her ier prr pmf srr srf pr2 を用意していた。NAPS8 では /grpK/nwp/Open/Const/Vsrf/Comm/lvtbl.txd 以下に her ie2 ier kor pft pm2 pmf pr2 prr sr1 sr2 sr3 srf srj srr yar yrr が置かれている。ルーチンジョブではこのディレクトリへシンボリックリンク SRF_LV_TABLE を張って利用する。
- 変換テーブル名が ie2, kor, pft のとき, ISPEC には変換テーブルに書かれた代表値の 1/10 が書かれる。それ以外の場合は変換テーブルの代表値がそのまま書かれる。
- 変換テーブル名が sr2 または srj のときは実数データが -900.0 より小さいものが欠損値とみなされる。そうでなければ、負値が欠損値とみなされる (NAPS7 のマニュアルでは欠損値は -1 を指定することとされている)。
- 変換テーブル名が srj のときは、実数データが変換テーブルの下限値に正確に一致しないと最も上の階級 (具体的には 42 で 21.0 を意味する) に割り当てられる。この挙動はバグかもしれない。
- 変換テーブルに 191 行以上書かれているとき、最初の 190 行だけが用いられ、レベル値は 0..190 となるが、返却値には実際のレベル数 (変換テーブルの行数 + 1) が返される。これは ispec の配列をオーバーフローしないためである。

履歴　この関数は NAPS7 時代から存在した。Fortran ラッパーが文字列の長さを伝えないバグは NuSDaS 1.3 で解決した。

6.8.7　srf_lv_trans: レベル値を実数データ (代表値) に変換

書式

int **srf_lv_trans**(const N_SI4 *idat*[ ], float *fdat*[ ], N_SI4 *dnum*, const N_SI4 *ispec*[ ]);

引数名	引数の型	役割
idat	const N_SI4 [ ]	入力データ
fdat	float [ ]	結果格納配列
dnum	N_SI4	データ要素数
ispec	const N_SI4 [ ]	ISPEC 配列

説明　新 ISPEC 配列 *ispec* にしたがって配列 *idat* のレベル値を代表値 *fdat* に変換する。

返却値　不明値以外となったデータの要素数

注

- 不明値は -1 となる。ただし、ISPEC のデータ種別 (先頭 4 バイト) が SRR2, SRF2, SRRR, SRFR の場合に限り -9999.0 となる。
- ISPEC のレベル表は通常 0.1mm 単位と解釈される。ただし、ISPEC の先頭 3 バイトが 'IER' であるか、あるいは ISPEC の先頭から 4 バイト目が '1' のときは 0.01mm 単位と解釈される。

履歴　この関数は NAPS7 時代から存在した。

6.8.8　srf_rd_rdic: レーダーサイト情報の問い合わせ

書式

int **srf_rd_rdic**(int *stnum*, int *iseq*, float ∗*lat*, float ∗*lon*, float ∗*hh*, N_SI4 ∗*offx*, N_SI4 ∗*offy*, N_SI4 ∗*type1*, N_SI4 ∗*type2*);

引数名	引数の型	役割
stnum	int	地点番号
iseq	int	日時 (通算時)
lat	float *	緯度
lon	float *	経度
hh	float *	高度
offx	N_SI4 *	中心のオフセット
offy	N_SI4 *	中心のオフセット
type1	N_SI4 *	デジタル化タイプ
type2	N_SI4 *	デジタル化タイプ

説明　ファイル名 RADAR_DIC のレーダー地点辞書から日時 *iseq* (通算時であって通算分でないことに注意) における地点番号 *stnum* のレーダーサイトの情報を読出す。

終了コード

1 正常終了

0 指定されたレーダーサイトがみつからなかった

-1 レーダー地点辞書が開けなかった

注

- レーダー地点辞書は NAPS8 では/grpK/nwp/Open/Const/Vsrf/Dcd/rdrdic.txt に置かれている。
- NAPS8 初期版 libsrf.a には経度のかわりに誤って緯度を返すバグがある。

履歴　この関数は NAPS7 時代からルーチン環境には存在したが、pnusdas から NuSDaS 1.1 に至る CVS 版ソースには含まれていなかった。NuSDaS 1.3 で両者が統合された。

6.8.9 srf_search_amdstn: 地点番号の辞書内通番を探す

書式

int **srf_search_amdstn**(const SRF_AMD_SINFO **amd*, int *ac*, int *stn*, int *amd_type*);

引数名	引数の型	役割
amd	const SRF_AMD_SINFO *	地点辞書配列
ac	int	地点辞書配列の長さ
stn	int	地点番号
amd_type	int	地点種別

説明　SRF_AMD_SINFO 構造型の配列 *amd* (地点数 *ac* 個) から地点番号 *stn* の地点情報を収めた添字 (1 始まり) を返す。

終了コード

正 地点の辞書内格納順位 (1 始まり)

-1 地点がみつからない

注意

- 地点種別 *amd_type* は無視される。
- 配列が地点番号順にソートされていることを前提に二分探索を使っている。

履歴　この関数は NAPS7 時代から存在したようであるがドキュメントされていなかった。NuSDaS 1.3 リリースに際してドキュメントされるようになった。

7 ネットワーク NuSDaS

7.1 はじめに

ネットワーク NuSDaS は、pandora サーバーによって提供されている NuSDaS データを、ローカルにあるファイルに対する API と同じもので取得しようとするためのものである。

防災情報影響センター向けの気象庁・河川局統合レーダープロダクトの開発段階で生まれたものであり、当初は C 言語での使用だけを前提にし、レーダーデータを取り扱うのに必要な API だけが対応していた。

その後、Fortran インターフェースへの対応、ほぼすべての API への対応を経て、NAPS8 では各課業務サーバーからデータバンクのデータの取得には、ネットワーク NuSDaS の利用が前提になっている。

NuSDaS1.3 では多くの API が追加されたが、データ取得に関するものはほとんど対応している。

7.2 ネットワーク NuSDaS の仕組み

pandora は、要求するデータを URL で指定して、HTTP プロトコルによってデータの要求および取得を行う。ネットワーク NuSDaS は API で指定された要求データを URL に翻訳し、それを用いて HTTP プロトコルで pandora サーバーと通信をしている。

HTTP プロトコルによる通信は、ネットワーク NuSDaS と同時に開発された `pandora_lib`(H.4 参照) を用いている。

7.2.1 データとサーバーの対応テーブル: PANDORA_SERVER_LIST

NuSDaS は、TYPE123 によってデータセットを特定することができる。同様にネットワーク NuSDaS においても TYPE123 とサーバーおよびパスを対応させるテーブルを以下のフォーマットでテキストファイルで作成して、そのファイル名を環境変数 `PANDORA_SERVER_LIST` に指定する。

フォーマット

```
_MSMLMLY.FCSV.STD1 192.168.0.1:8080 /NAPS8_NUSDAS/data/Mf/Fcst/fcst_sfc.nus
_MSMLMPP.FCSV.STD1 192.168.0.1:8080 /NAPS8_NUSDAS/data/Mf/Fcst/fcst_p.nus
_MSMLMPP.FCSV.STD1 192.168.0.2:8080 /NAPS8_NUSDAS/data/Mf/Fcst/fcst_p.nus
_MSMLMLY.FCSV.2D_1 192.168.0.1:8080 /NAPS8_NUSDAS/data/Mf/Fcst/fcst_phy2m.nus 70
_MSMLMLY.FCSV.2D_1 192.168.0.2:8080 /NAPS8_NUSDAS/data/Mf/Fcst/fcst_phy2m.nus 80
```

第 1 カラム 取得しようとするデータ種別を type1.type2.type3 という形で指定する。

第 2 カラム 第 1 カラムで指定したデータを提供しているサーバーを server:port で指定する。

第 3 カラム 振り分け先のパスを指定する。必ず / で始める。なお、NAPS8 のデータバンク管理サーバーでは、振り分け名は `NAPS8_NUSDAS` で、その後が NuSDaS Root Directory になっている。

第 4 カラム NRD 番号を指定する。省略可 (省略すると NRD 番号=99 として扱う)。

注意事項

- 上の例のように一つの種別の資源に対して複数の異なるサーバーを指定することができる。複数のサーバーを指定した場合には、上にあるサーバーから順にデータ取得が試みられる。データ取得に成功したサーバーの指定は、次の接続では優先的にデータ取得が試みられる。
- 接続 (TCP の connect) に失敗したエントリーには「接続不能」のフラグが付加され、同一プロセスではそのサーバーへの接続は試みない。

7.2.2 “データセット” の概念

NuSDaS1.1 における実装は、ファイルに対する API と pandora データに対する API を wrap したものになっており、ファイルに対する API を呼んで失敗した場合には pandora データに対する API を呼ぶようになっていた。

NuSDaS1.3 においては、ファイルも pandora データも対等の “データセット” という位置づけになっており、データセットの下の層での挙動をデータの種類に応じて変えている。先に述べた「データ取得に成功したサーバーの指定は、次の接続では優先的にデータ取得が試みられる」という動作は、データセットのリストの順序を入れ替えることによって実現されている。

7.2.3 URL の規則

各 API は、それぞれの機能に応じた URL を作成し、それを用いて pandora server に対して HTTP プロトコルによってデータの要求、および取得を行っている。URI 規則については H.2.1 を参照のこと。

`nusdas_inq_nrdbtime`, `nusdas_inq_nrdvtime` については、それぞれ basetime 一覧、validtime 一覧の index 資源を要求するので、TYPE123 の指定まで、member の指定の指定までであるが、そのほかの API においては、指定をしない場合でも最後の element まで指定する。ただし、指定されない要素については “none” を埋めるようにしている。また、`INFO` の場合は、グループ名を便宜的に element のところに格納して URL を構成している。

例） `_MSMLMZS.FCSV.STD1/2004-01-01t0000/none/2004-01-01t0000/none/GRID`

7.3 制限事項

- ネットワーク NuSDaS でサポートしているのはデータの取得のみで、書き込みには対応していない。

A データファイル形式

NuSDaS データファイル形式はライブラリ改修とともに若干変遷をとげている。これらはファイル形式番号という整数値で区別される。今まで存在したのは 1 (10 と同一形式), 10, 11, 13, 14 である。ここでは最新ファイル形式 14 を基本としつつ、旧形式で異なる点はその都度別に説明する。

A.1 レコード形式の一般形

NuSDaS データファイル形式は、ほとんどの Fortran コンパイラが順番探査ファイルとして採用している形式を持つ (ファイル形式 10 は微妙に違っていた)。データファイルはレコード (Table A.1 (p. 139)) の集まりであり、レコード内容は最大約 4 ギガバイト $(4 \times 1024^3 - 24) = 4\,294\,967\,272$byte の長さを持つことができる。

レコードの機能は先頭部に書かれている 4 文字のレコード名で決まる。レコード名には `NUSD`, `CNTL`, `INDY`, `INDX`, `SUBC`, `DATA`, `INFO`, および `END␣` [(注 1)]があり、配列には以下の規則がある。

- NUSD レコードはファイルの先頭に 1 つだけ存在する。
- CNTL レコードは NUSD レコードの次に 1 つだけ存在する。
- INDY レコードが CNTL レコードの次に 1 つだけ存在する。ファイル形式 11 以前ではこの場所に INDX レコードがある[(注 2)]。
- SUBC レコードは INDY レコードの後に、定義ファイルで指定した数だけ存在する。
  - 定義ファイルで指定しない SUBC レコードを作成することはできない。
  - 定義ファイルで指定した SUBC レコードを実際に書き込まなくてもファイルを閉じる際にエラーになる (初期値設定が可能な場合) か不定内容のレコードが作成される。
- END レコードはファイルの末尾に 1 つだけ存在する。
- DATA レコードと INFO レコードは SUBC[(注 3)]と END の間に任意個存在する。
  - これらの順序はデータ作成プログラムが nusdas_write (p. 47, 89) および nusdas_info (p. 58, 100) を呼び出した順序に依存するため、INFO レコードが DATA レコードの後に来ることは保証されない。
  - 本来 INFO レコードは定義ファイルに記述すべきものであり、正しく記述すれば DATA レコードの前に配置される。

[(注 1)]以下 `␣` はスペース文字である。
[(注 2)]NAPS7 の ES ファイルにはどちらも存在しない。しかし、今となってはそれを言っても仕方がないようにも思う
[(注 3)]NuSDaS 1.1 系では SUBC より INFO が後と保証できるか?

項番	フィールド	型	長さ オクテット	内容
1	レコード長	非負整数	4	項番 2〜6 の占めるオクテット数 (注 1)
2	レコード名	文字	4	`NUSD`, `CNTL`, ...
3	レコード有効長	非負整数	4	項番 2〜5 の占めるオクテット数
4	更新時刻	整数	4	データ作成機上での time_t のシステム時間
5	レコード内容		可変	後続の表を見よ
6	すき間		可変	レコード長調整のため (注 2)
7	レコード長	非負整数	4	項番 1 と同じ

Table A.1: NuSDaS データファイルのレコード形式の一般型。注 1: ファイルバージョン 10 においては項番 1〜7 の占めるオクテット数だった。注 2: 定義ファイルの FIXEDRLEN 機能を用いた場合は、項番 1〜7 の長さが指定のレコード長に一致するように幅が決まる。そうでない場合、ファイル形式 13 以降においてはレコード長が 8 の倍数となるように 0〜7 オクテットのすき間が挿入される。

A.2 NUSD レコード

NUSD レコードの形式を Table A.2 (p. 140) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`NUSD`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	作成元	文字	72	定義ファイルの CREATOR 文 (注 1)
6	88	ファイル長	整数	8	ファイルのオクテット数
7	96	ファイル版番号	整数	4	14
8	100	ファイル長	非負整数	4	(注 2)
9	104	レコード数	非負整数	4	NUSD から END まで
10	108	INFO レコード数	非負整数	4	N_i
11	112	SUBC レコード数	非負整数	4	N_s
12	116	すき間		可変	
13		レコード長	非負整数	4	

Table A.2: NUSD レコードの形式。注 1: ファイルバージョン 11 以前では 80 オクテット長で項番 6 は存在しない。注 2: ファイル長が 0xFFFF FFFF オクテットを超えるときは 0xFFFF FFFF.

A.3 CNTL レコード

CNTL レコードの形式を Table A.3 (p. 141) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`CNTL`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	種別	文字	16	定義ファイルの TYPE1〜TYPE3
6	32	基準時刻	文字	12	`yyyymmddhhnn` 形式
7	44	基準時刻	整数	4	通算分
8	48	予報時間単位	文字	4	定義ファイルの VALIDTIME 文
9	52	メンバー数	非負整数	4	N_M
10	56	予報時間数	非負整数	4	N_V
11	60	面数	非負整数	4	N_Z
12	64	要素数	非負整数	4	N_E
13	68	投影法	文字	4	定義ファイルの PROJECTION 文
14	72	x, y 格子数	非負整数	4×2	定義ファイルの SIZE 文
15	80	基準点座標	単精度	4×2	定義ファイルの BASEPOINT 文
16	88	基準点緯経度	単精度	4×2	同上
17	96	格子間隔	単精度	4×2	定義ファイルの DISTANCE 文
18	104	標準緯経度	単精度	4×2	定義ファイルの STANDARD 文
19	112	第 2 標準緯経度	単精度	4×2	同上
20	120	緯経度 1	単精度	4×2	定義ファイルの OTHERS 文
21	128	緯経度 2	単精度	4×2	同上
22	136	格子点の空間代表性	文字	4	定義ファイルの VALUE 文
23	140	予約 (投影)	単精度	4×2	
24	148	予約	未定義	4×6	
25	172	メンバー名	文字	$4 \times N_M$	
26	$+4N_M$	対象時刻 1	整数	$4 \times N_V$	
27	$+4N_V$	対象時刻 2	整数	$4 \times N_V$	
28	$+4N_V$	面名 1	文字	$6 \times N_Z$	
29	$+6N_Z$	面名 2	文字	$6 \times N_Z$	
30	$+6N_Z$	要素名	文字	$6 \times N_E$	
31	$+6N_E$	すき間		可変	
32		レコード長	非負整数	4	

Table A.3: CNTL レコードの形式。繰り返し数 N_M, N_V, N_Z および N_E はデータファイル作成時の定義ファイルのリスト長である。ただし、NuSDaS 1.2 以前は定義ファイルの設定により N_V がこれより短くなる場合があった。

A.4 INDX/INDY レコード

INDY レコードの形式を Table A.4 (p. 142) に、同じ役割でバージョン 1.1 まで用いられた INDX レコードの形式を Table A.5 (p. 142) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`INDY`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	DATA レコード位置	整数	$8 \times N_M \times N_V \times N_Z \times N_E$	(注 1,2)
6		DATA レコード長	非負整数	$4 \times N_M \times N_V \times N_Z \times N_E$	(注 1,3)
7		DATA レコード要素数	非負整数	$4 \times N_M \times N_V \times N_Z \times N_E$	(注 1)
8		すき間		可変	
9		レコード長	非負整数	4	

Table A.4: INDY レコードの形式。注 1: 次元順を間違えないようにくどく書くと、この構造は「『「『整数値を N_E 個並べた 1 次元配列』が N_Z 個並んだ 2 次元配列」が N_V 個並んだ 3 次元配列』が N_M 個並んだ 4 次元配列」である。各次元の中での順序は定義ファイルの記述順に同じ。ただし、定義ファイルで `MEMBER` N_M `OUT` または `VALIDTIME` N_V `OUT` を指定した場合は、NuSDaS 1.1 および NAPS9 以降ではそれぞれ N_M または N_V に代えて次元長が 1 となる（当該データファイルで使われているメンバーまたは対象時刻だけが記述される）。注 2: 定義ファイルの ELEMENTMAP により出力禁止されているレコードは −1, 出力許可だが未だ書かれていないレコードは 0 が書かれる。注 3: ここで言うレコード長とは TableA.1 で示している項番 2〜6 の占めるオクテット数ではなく、レコード全体の長さである項番 1〜7 の占めるオクテット数である。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`INDX`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	DATA レコード位置	非負整数	$4 \times N_M \times N_V \times N_Z \times N_E$	(注 1,2)
6		すき間		可変	
7		レコード長	非負整数	4	

Table A.5: ファイルバージョン 11 までの INDX レコードの形式。注 1: 次元順は INDY レコードに同じ。注 2: 定義ファイルの ELEMENTMAP により出力禁止されているレコードは 0xFFFF FFFF, 出力許可だが未だ書かれていないレコードは 0 が書かれる。

A.5 SUBC レコード

SUBC レコードの一般形式を Table A.6 (p. 143) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`SUBC`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	群名	文字	4	`ETAΔ`, `ZHYB`, `TDIF`, ...
6	20	レコード内容	本節下表参照	可変	群によって異なる
7		すき間		可変	
8		レコード長	非負整数	4	

Table A.6: SUBC レコードの形式 (一般型)。

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	面の数	非負整数	4	N
6-2	24	係数 A	単精度	$4 \times (N+1)$	
6-3	$24 + 4 \times (N+1)$	係数 B	単精度	$4 \times (N+1)$	
6-4	$24 + 8 \times (N+1)$	係数 C	単精度	4	

Table A.7: SUBC ETA の TableA.6 の項番 6 の形式

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	対象時刻からのずれ	整数	$4 \times N_M \times N_V$	
6-2	$24 + 4 \times N_M \times N_V$	積算秒	整数	$4 \times N_M \times N_V$	

Table A.8: SUBC TDIF の TableA.6 の項番 6 の形式。内容については 3.1.3 節 (p. 28) 参照。

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	時間積分間隔	整数	4	

Table A.9: SUBC DELT の TableA.6 の項番 6 の形式。内容については 3.1.3 節 (p. 28) 参照。

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	鉛直層数	整数	4	`nz`
6-2	24	温位の基準値	単精度	4	`ptrf`
6-3	28	気圧の基準値	単精度	4	`presrf`
6-4	32	モデル面高度 (フルレベル)	単精度	$4 \times$ `nz`	`zrp(nz)`
6-5	$32 + 4 \times$ `nz`	モデル面高度 (ハーフレベル)	単精度	$4 \times$ `nz`	`zrw(nz)`
6-6	$32 + 8 \times$ `nz`	座標変換関数 (フルレベル)	単精度	$4 \times$ `nz`	`vctrans_p(nz)`
6-7	$32 + 12 \times$ `nz`	座標変換関数 (ハーフレベル)	単精度	$4 \times$ `nz`	`vctrans_w(nz)`
6-8	$32 + 16 \times$ `nz`	座標変換関数の微分 (フルレベル)	単精度	$4 \times$ `nz`	`dvtrans_p(nz)`
6-9	$32 + 20 \times$ `nz`	座標変換関数の微分 (ハーフレベル)	単精度	$4 \times$ `nz`	`dvtrans_w(nz)`

Table A.10: SUBC ZHYB の TableA.6 の項番 6 の形式

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	全球の南北分割数	整数	4	`j`
6-2	24	格納されている最北の緯度の格子番号	整数	4	`j_start`
6-3	28	データの南北格子数	整数	4	`j_n`
6-4	32	全球の東西分割数	整数	$4 \times$ `j_n`	`i(j_n)`
6-5	$32 + 4 \times$ `j_n`	データの最西格子の番号	整数	$4 \times$ `j_n`	`i_start(j_n)`
6-6	$32 + 8 \times$ `j_n`	データの東西格子数	整数	$4 \times$ `j_n`	`i_n(j_n)`
6-7	$32 + 12 \times$ `j_n`	緯度	単精度	$4 \times$ `j_n`	`lat(j_n)`

Table A.11: SUBC RGAU の TableA.6 の項番 6 の形式

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	レーダー運用情報	整数	$4 \times N_M \times N_V \times N_Z \times N_E$	0:No Data, 1: Echo, 2: No Echo, 3: No Ope

Table A.12: SUBC RADR の TableA.6 の項番 6 の形式

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	合成情報	整数	$4 \times 128 \times \times N_M \times N_V \times N_Z \times N_E$	表?? 参照

Table A.13: SUBC ISPC の TableA.6 の項番 6 の形式

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	観測モード	整数	$4 \times N_M \times N_V \times N_Z \times N_E$	1: MODE1, 2: MODE2, 4: MODE3, 7: off
6-2	24	エコーフラグ	整数	$4 \times N_M \times N_V \times N_Z \times N_E$	0:Echo あり, 1: No-Echo, 2: No Ope
6-3	28	N_0 値の 10 倍	整数	$4 \times N_M \times N_V \times N_Z \times N_E$	
6-4	32	パラメータ B		$4 \times N_M \times N_V \times N_Z \times N_E$	
6-5	36	パラメータ β		$4 \times N_M \times N_V \times N_Z \times N_E$	

Table A.14: SUBC RADS の TableA.6 の項番 6 の形式。雨量強度 R[mm/hr] は、レーダーごとに異なる定数 N_0、パラメータ B, β、距離補正済みの受信電力 N を用いると、$R = \left(\frac{200}{B}\right)^{1/\beta} \times 10^{\frac{80}{256} \times \frac{N - N_0}{10\beta}}$ で算出される。

項番	位置	フィールド	型	長さ オクテット	内容
6-1	20	観測時刻	整数	4	
6-2	24	周波数	整数	4	
6-3	28	レンジ解像度	整数	4	
6-4	32	方位角解像度	整数	4	
6-5	36	データ解像度	整数	4	
6-6	40	MTI フィルタ	整数	4	
6-7	44	有効データ数	整数	4	
6-8	48	仰角	整数	4	

Table A.15: SUBC DPRD の TableA.6 の項番 6 の形式。

A.6 INFO レコード

INFO レコードの形式を Table A.16 (p. 145) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`INFO`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	群名	文字	4	利用者定義
6	20	レコード内容	文字	可変	利用者定義
7		すき間		可変	
8		レコード長	非負整数	4	

Table A.16: INFO レコードの形式。レコード内容はバイナリであってもよいが、NuSDaS インターフェイスはバイトオーダーの相違について関知しない。

A.7 DATA レコード

DATA レコードの一般形式を Table A.17 (p. 146) に示す。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`DATA`
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	メンバー名	文字	4	
6	20	対象時刻	整数	4×2	
7	28	面名	文字	6×2	
8	40	要素名	文字	6	
9	46	予約	未定義	2	
10	48	x, y 格子数	非負整数	4×2	
11	56	パッキング方式	文字	4	定義ファイルの PACKING 文
12	60	欠損値表現方式	文字	4	定義ファイルの MISSING 文
13	64	パックデータ	本節下表参照	可変	
14		すき間		可変	
15		レコード長	非負整数	4	

Table A.17: DATA レコードの形式。

項番	位置	フィールド	型	長さ オクテット	内容
13-DATA	64	データ		可変	

Table A.18: 欠損値表現方式が “NONE” の場合の Table A.17 の項番 13 の形式。

項番	位置	フィールド	型	長さ オクテット	内容
13-1	64	欠損値の値	PACK に依存	n_ud = 1∼8	
13-DATA	64 + n_ud	データ		可変	

Table A.19: 欠損値表現方式が “UDFV” の場合の Table A.17 の項番 13 の形式。

項番	位置	フィールド	型	長さ オクテット	内容
13-1	64	マスクビット	ビット列	`n_ms` = (x 格子数 × y 格子数 - 1) / 8 + 1	
13-DATA	64 + n_ms	データ		可変	

Table A.20: 欠損値表現方式が “MASK” の場合の Table A.17 の項番 13 の形式。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	base(b)	単精度	4	
2	4	amp(a)	単精度	4	
3	8	パックされたデータ	非負整数 (1PAC)	1 × 格子数	
			整数 (2PAC)	2 × 格子数	
			非負整数 (2UPC)	2 × 格子数	

Table A.21: パッキング方式が “1PAC”, “2PAC”, “2UPC” の場合の Table A.18, Table A.19, Table A.20, の項番 13-DATA の形式。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	base(b)	単精度	4	
2	4	amp(a)	単精度	4	
3	8	N	非負整数	4	資料数
4	12	R_i	非負整数	2 × 群数 N_G	各群の参照値 ($i = 0 \cdots N_G - 1$)
5	$12 + 2N_G$	W_i	4bit	N_H	各群の幅 ($i = 0 \cdots N_G - 1$)
6	$12 + 2N_G + N_H$	圧縮データ	$(W_i + 1)$ bit	各群 4 $(W_i + 1)$	32 $(W_i + 1)$bit = 4 $(W_i + 1)$ byte

Table A.22: パッキング方式が “2UPP” の場合の Table A.18, Table A.19, Table A.20, の項番 13-DATA の形式。$N_G = ((N - 1)/32) + 1$. $N_H = ((N_G - 1)/2) + 1$. 圧縮データは各群について $(W_i + 1)$ ビット幅の整数 32 個 (末端に欠有り得) の 2 階差分値 (オフセットあり)。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	base(b)	倍精度	8	
2	8	amp(a)	倍精度	8	
3	16	パックされたデータ	整数 (4PAC)	4 × 格子数	

Table A.23: パッキング方式が “4PAC” の場合の Table A.18, Table A.19, Table A.20, の項番 13-DATA の形式。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	nbit	非負整数	4	ランレングスビット数
2	4	maxv	非負整数	4	データ最大値
3	8	num	非負整数	4	データ列の数
3	12	データ列		(nbit × num - 1)/ 8 + 1	

Table A.24: パッキング方式が “RLEN” の場合の Table A.18, Table A.19, Table A.20, の項番 13-DATA の形式。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	データ	非負整数 (I1)	1 × 格子数	
			整数 (I2)	2 × 格子数	
			整数 (N1I2)	2 × 格子数	
			整数 (I4)	4 × 格子数	
			単精度 (R4)	4 × 格子数	
			倍精度 (R8)	8 × 格子数	

Table A.25: パッキング方式が “I1 ”, “I2 ”, “N1I2”, “I4 ”, “R4 ”, “R8 ” の場合の Table A.18, Table A.19, Table A.20, の項番 13-DATA の形式。

A.8 END レコード

END レコードの形式を Table A.26 (p. 148) に示す。NuSDaS 1.3 にはバグがあって、既に補助レコードが存在するデータファイルを開いて書き込むと、補助レコード表 (項番 7) に不正値が入っていた。このバグは NuSDaS 1.4 で修正されている。

項番	位置	フィールド	型	長さ オクテット	内容
1	0	レコード長	非負整数	4	
2	4	レコード名	文字	4	`END`Δ
3	8	レコード有効長	非負整数	4	
4	12	更新時刻	整数	4	
5	16	ファイル長	非負整数	4	NUSD 項番 8 に同じ
6	20	レコード数	非負整数	4	NUSD 項番 9 に同じ
7	24	補助レコード表		$24 \times (N_s + N_i)$	7.1〜7.5 を繰り返し
7.1	+0	レコード名	文字	4	`SUBC` または `INFO`
7.2	+4	群名	文字	4	
7.3	+8	レコード位置	整数	8	
7.4	+16	レコード長	非負整数	4	(注 1)
7.5	+20	予約	未定義	4	0
8		すき間		可変	
9		レコード長	非負整数	4	

Table A.26: END レコードの形式。ファイル形式 11 以前は項番 7 を欠く。注 1: ここで言うレコード長とは TableA.1 で示している項番 2〜6 の占めるオクテット数ではなく、レコード全体の長さである項番 1〜7 の占めるオクテット数である。

B 数値・名前の表

B.1 共通エラーコードの表

NuSDaS インターフェイスが返すエラーコードのうち、−10 以下の値は関数によらず次のような共通の意味をもつ。

-10 メモリが足りない。

-13 種別 1 の後半 4 文字をスペースにした場合の補間機能が働かなかった。

-18 データファイルに書かれたバージョン番号指定が不正。

-21 指定された種別に対応する NuSDaS Root Directory が見つからなかった。

-33 定義ファイルの行の順序が不正である。

-34 定義ファイルの elementmap の指定が不正である。

-35 定義ファイルの解読中に予期せぬ EOF があった。

-36 定義ファイルの解読中にエラーが発生した (その他の定義ファイル関連のエラーに属しないもの)。

-40 定義ファイルで TYPE1 が未定義。

-41 定義ファイルで TYPE2 が未定義。

-42 定義ファイルで TYPE3 が未定義。

-43 定義ファイルで対象時刻の数が未定義。

-44 定義ファイルで対象時刻のリストが未定義。

-45 定義ファイルで層の数が未定義。

-46 定義ファイルで層のリストが未定義。

-47 定義ファイルで物理量の数が未定義。

-48 定義ファイルで物理量のリストが未定義。

-49 定義ファイルで格子の数が未定義。

-50 要求された対象時刻が、定義ファイルで指定した validtime に含まれていない。または定義ファイルに memberlist が未定義。

-51 type123, 基準時刻、メンバー、対象時刻から決まる NuSDaS データファイルが存在しない (open に失敗した)。

-53 データファイルの書き込み時、新規作成に失敗。

-54 既存データファイルの識別部 (NUSD) が不正または読み込みに失敗。

-55 既存データファイルの管理部 (CNTL) が不正または読み込みに失敗。

-56 既存データファイルのアドレス部 (INDX) が不正または読み込みに失敗。

-57 既存データファイルの終了部 (END) が不正または読み込みに失敗。

-58 基準時刻に -1 を指定したときに、指定された対象時刻を持つデータが見つからなかった。

-59 既存データファイルのデータ部 (DATA) が不正または読み込みに失敗。

-63 レコード長がレコード強制長 (定義ファイルにおいて forced rlen で指定した値) を超えた。

-64 定義ファイルに指定した INFO レコードの内容を記述したファイルの読み込みに失敗した。

-65 ファイルの close 処理において、識別部 (NUSD) の書き込みに失敗した。

-66 ファイルの close 処理において、管理部 (CNTL) またはアドレス部 (INDX) の書き込みに失敗した。

-67 ファイルの close 処理において、終了部 (END) の書き込みに失敗した。

-68 読み込み限定で open しているファイルに書き込みをしようとした。

-69 ID が 50 以上の NuSDaS Root Directory に、書き込みを行おうとした。

-80 4GiB 以上の書き込みをサポートしていない NuSDaS1.1 形式データに 4GiB を超える書き込みをしようとした。

-81 不正な投影法パラメタを持つデータファイルを作成しようとした、または、nusdas_grid で不正な投影法パラメタを設定しようとした。

-98 gz 圧縮に関する操作を行おうとしたが、Zlib がリンクされていないため、できない。

-99 ファイル I/O エラーが発生。pandora サーバーとの通信におけるエラーも含む。

互換性

-18 NuSDaS1.3 で新設。

-20 NuSDaS1.1 では「定義ファイルの同時オープン数が NuSDaS の最大値を超えた」としているが、NuSDaS1.3 ではメモリが許す限りこの上限がないので廃止する。ちなみに、NuSDaS1.1 におけるこの最大値は 99。

-31 NuSDaS1.1 のドキュメントには「定義ファイルのある項目の数の指定が NuSDaS 最大値を超えた」となっているが、NuSDaS1.3 ではメモリが許す限り上限がないので廃止する。ちなみに、NuSDaS 1.1 における上限値はメンバー数、層数、要素数がそれぞれ 256、対象時刻の数が 36500 であった。

-32 NuSDaS1.1 では定義ファイルを読み込むときのメモリ不足に用いていたが、NuSDaS1.3 では −10 に統合して廃止する。

-35, -36 NuSDaS1.3 で新設。

-52 NuSDaS1.1 では「データファイルの同時オープン数が NuSDaS 最大値を超えた」としているが、NuSDaS1.3 ではメモリおよびファイルハンドルの数が許される限りは上限がないので廃止する。ちなみに、NuSDaS1.1 におけるこの最大値は 99。

-58 NuSDaS1.1 のドキュメントでは説明されていなかったが、同意味。

-59 NuSDaS1.3 で新設。

-60, -61, -62 NuSDaS1.1 ではファイルの初期化 (新規作成) におけるエラーとしてこれらのエラーが定義されていたが、NuSDaS1.1 と NuSDaS1.3 ではファイルの初期作成の方法が異なるため、NuSDaS1.3 ではこのエラーに対応するエラーは発生しない (発生する可能性があるのは −53 である)。よって NuSDaS1.3 では廃止する。

-63 NuSDaS1.1 では初期化時に限定したエラーであったが、NuSDaS1.3 では初期化時に限らず、レコード長がレコード強制長を超えた場合に用いる。

-66 NuSDaS1.1 ではアドレス部 (INDX) の書き込みに限定しているが、NuSDaS1.1 ではファイルの close 処理においては、管理部 (CNTL) は書き込みをしてないためである。NuSDaS1.3 においては書き込み処理をしているため、ここに管理部の書き込みエラーを含めることにする。

-70〜-79 NuSDaS1.3 では ES 利用をサポートしないので、廃止。

-80 ドキュメントされていなかったが、NuSDaS1.1 で設定された。

-81, -98 NuSDaS1.3 で新設された。

B.2 種別名

現在はオンライン登録制となっている。最新登録状況は `http://nusdas.npd.naps.kishou.go.jp/` 参照のこと (庁内限定)。

名前	意味
_AVN	米国全球モデル
_CTM	オゾン
_CWM	波浪モデル
_DCD	観測デコード
_FDV	4 次元変分同化用定数
_GSM	全球モデル
_LLM	低解像度モデル
_LRM	低解像度モデル
_MSG	エエロゾル (MASINGER)
_MSM	メソモデル
__QA	毎時解析
_QMA	毎時解析
_RSM	領域モデル
_SF1	1ヶ月予報
_SF4	4ヶ月予報
_SGM	高潮
_SRF	降水短時間予報
_SST	SST 解析
_SWM	浅海波浪モデル
_TEM	台風アンサンブルモデル
_TYM	台風モデル
_VRF	検証
_WFM	週間予報
_XXX	不明

Table B.1: 種別に用いるモデル名

種別 1 の先頭 4 文字はモデル名と呼ばれ、表 B.1 に従ってデータを作成した処理の名称をつける。種別 1 の続く 2 文字は 2 次元座標名と呼ばれ、表 B.2 に従ってデータ記録の 2 次元配列がとられた座標系の名称をつける。種別 1 の続く 2 文字は 3 次元座標名と呼ばれ、表 B.3 に従って面名がつけられた座標系の名称をつける。

種別 2 の最初の 2 文字は属性名と呼ばれ、表 B.4 に従って予報値、解析値、観測値などの区別を表わす。種別 2 の末尾 2 文字は時間種類名と呼ばれ、表 B.5 に従って瞬間値、時間平均値などの区別を表わす。ただし実運用においては時間平均値や積算値を瞬間値とまとめて `SV` のデータセットに格納することがある。

種別 3 は任意に付けられるデータセット名である。名前 `STD1` はデータ作成処理にとって最も標準的な出力 (標準ファイル) に付けられる名前である。

2 字略号	3 字略号	意味
FG	FGΔΔ	自由格子
GS	GSΔΔ	矩形ガウス格子 (SUBC RGAU を設定した場合を除く)
LL	LLΔΔ	経緯度座標
LM	LMNΔ	ランベルト正角円錐図法
LM	LMSΔ	同 (南半球)
MR	MERΔ	メルカトル図法
OL	OLΔΔ	斜軸ランベルト
PS	PSNΔ	ポーラーステレオ図法
PS	PSSΔ	同 (南半球)
RD	RDΔΔ	レーダー単サイト
RG	RGΔΔ	適合ガウス格子 (SUBC RGAU を設定した矩形ガウス格子を含む)
RT	RTΔΔ	極座標レーダー
SB	SBΔΔ	細分
ST	STΔΔ	地点観測
XX	XXΔΔ	不明
YP	YPΔΔ	子午面断面

Table B.2: 2 次元座標の名称。種別 1 には 2 字略号を用い、その他の場合、つまり CNTL レコードや NUSDAS_GRID で受け渡す値には 3 字略号を用いる。

名前	意味
ET	モデル面 (η 座標)
FL	フライトレベル (F010 ... F450)
LO	経度 (用例は ZONAL のみ)
LY	高度面、エコートップ、大気全層など
PP	気圧座標
SF	地表面
SG	モデル面 (σ 座標)
TO	閾値
ZS	モデル面 (z^* 座標)
XX	不明

Table B.3: 種別に用いる 3 次元座標名

名前	意味
AA	同化解析
AF	解析 (変分同化等) の中の予報
EF*	アンサンブル予報
CC	定数
EA	速報解析
FC	予報値
GS	解析の推定値 (guess)
IN	初期値
OB	観測値
PT	アンサンブルの摂動
RA*	再解析
VR	検証値
XX	不明

Table B.4: 種別に用いる属性名.

名前	意味
AN	積算値の気候値
AV	積算値
DN	標準偏差の気候値
DV	標準偏差
MA	期間平均の気候値からの偏差
MN	期間平均の気候値
MV	(気候に比べ短期間の) 平均値
PV	確率値
SN	瞬間値の気候値
SV	瞬間値
XX	不明

Table B.5: 種別に用いる時間種類名

B.3 面名の表

名前	意味
ATMTOP	大気上端
CBTOP	積乱雲の雲頂
COLUMN	大気全層
ECTOP	レーダーのエコー頂
MXWIND	風速最大の高度
SURF	地表面 (ただし転用多し)
TOTAL	土壌全層
TROPO	対流圏界面
ZONAL	大気全体 (YP 座標系での東西平均)

Table B.6: 面名

面名はこの表に示したものの他に、種別 1 の 3 次元座標名（表 B.3）に応じて気圧座標 (PP) であれば 850, 500 といった気圧の値、フライトレベル (FL) であれば F450 といった値を利用可能。

B.4 その他の表

名前	意味
I1	1 バイト符号なし整数型
I2	2 バイト符号付き整数型
I4	4 バイト符号付き整数型
R4	4 バイト単精度浮動小数点型
R8	8 バイト倍精度浮動小数点型
NC	パッキングをしたままのバイト列 (パッキングが 2UPC,2UPP または 2UPJ のときのみ)

Table B.7: ユーザ配列の型の名称

名前	意味
1PAC	符号なし 1 バイト整数によるバイトパック
2PAC	符号付き 2 バイト整数によるバイトパック
2UPC	符号なし 2 バイト整数によるバイトパック
2UPJ	符号なし 2 バイト整数によるバイトパック、 JPEG2000 圧縮
2UPP	符号なし 2 バイト整数によるバイトパック、 複合差分圧縮
4PAC	符号つき 4 バイト整数によるバイトパック
N1I2	10 倍値の 2 バイト符号付き整数による格納
RLEN	8 ビットランレングス圧縮
I1	1 バイト符号なし整数による格納
I2	2 バイト符号つき整数による格納
I4	4 バイト符号つき整数による格納
R4	単精度浮動小数点型による格納
R8	倍精度浮動小数点型による格納

Table B.8: パッキング方式名称。それぞれの形式については E 章 (p. 178) 参照

名前	意味
NONE	欠損値なし。あらゆる数値が意味を持つデータである (デフォルト)
UDFV	データレコード毎に定めるある値が欠損値である。 欠損値は nusdas_parameter_change (p. 56, 98) で設定する
MASK	マスクビットによってデータの存在格子を指定する。 nusdas_make_mask (p. 54, 96) でマスクビットを生成し、 nusdas_parameter_change (p. 56, 98) または nusdas_set_mask (p. 55, 97) で欠損格子を設定する。

Table B.9: 欠損値表現方式

名前	意味
PVAL	データは格子点における値である (デフォルト)
MEAN	データは格子点近傍の場の平均値である
REPR	データは格子点近傍の場を何らかの意味で代表する値である

Table B.10: 格子点の空間代表性

B.5 要素名の表

SUBC TDIF レコードとともに期間内の平均、積算、最大、最小値を格納する場合、要素名の先頭に、期間平均値の場合「 _ 」、期間積算値の場合「 . 」、期間最大値の場合「 A_ 」、期間最小値の場合「 I_ 」を付加する。

現在はオンライン登録制となっている。最新登録状況は `http://nusdas.npd.naps.kishou.go.jp/` 参照のこと（庁内限定）。

要素名	GRIB1 番号	旧形式	単位	物理量
Pres	1		Pa	気圧
P	1	GVD	hPa	気圧
PAI	1	GVD	hPa	気圧
Pmsl	2		Pa	海面更正気圧
PSEA	2	GVD	hPa	海面更正気圧
SLP	2	GVD	hPa	海面更正気圧
Ptend	3		Pa/s	気圧変化の傾向
pVOR	4		$K \cdot m^2/(kg \cdot s)$	ポテンシャル渦度
sarH	5		m	ＩＣＡＯ標準大気参照高度
G	6		m^2/s^2	ジオポテンシャル
PHI	6	GVD	m^2/s^2	ジオポテンシャル
gpH	7		gpm	ジオポテンシャル高度
Z	7	GVD	m	ジオポテンシャル高度
gmH	8		m	幾何学的高度
sdH	9		m	高度の標準偏差
tOZON	10		Dobson	オゾン全量
T	11	GVD	K	気温
vT	12		K	仮温度
pT	13		K	温位
papT	14		K	偽断熱温位
maxT	15		K	最高気温
minT	16		K	最低気温
dT	17		K	露点温度
TTD	18	GVD	K	湿数
TRate	19		K/m	気温減率
VIS	20		m	視程
Radr1	21			レーダースペクトル（ａ）
Radr2	22			レーダースペクトル（ｂ）
Radr3	23			レーダースペクトル（ｃ）
PLI50	24		K	500hPa 面への気塊持ち上げ指数
Tano	25		K	気温偏差
Pano	26		hPa	気圧偏差
gpHan	27		gpm	ジオポテンシャル高度偏差
Zan	27		m	ジオポテンシャル高度偏差
Wave1	28			波のスペクトル（ａ）
Wave2	29			波のスペクトル（ｂ）
Wave3	30			波のスペクトル（ｃ）
WindD	31		degree true	風向
WindS	32		m/s	風速
U	33	GVD	m/s	風のｘ軸成分
WindX	33		m/s	風のｘ軸成分
UU	33	GVD	m/s	風の東西成分
V	34	GVD	m/s	風のｙ軸成分
WindY	34		m/s	風のｙ軸成分
VV	34	GVD	m/s	風の南北成分
PSI	35		m^2/s	流線関数
CHI	36	GVD	m^2/s	速度ポテンシャル
mPSI	37		m^2/s^2	モンゴメリーの流線関数

sVV	38		1/s	シグマ座標における鉛直速度
VVPa	39		Pa/s	鉛直速度（気圧座標）
OMG	39	GVD	hPa/hour	鉛直速度（気圧座標）
VVm	40		m/s	鉛直速度（m単位）
W	40		m/s	鉛直速度
aVOR	41		1/s	絶対渦度
aDIV	42		1/s	絶対発散
rVOR	43		1/s	相対渦度
VOR	43	GVD	10^{-6}/s	相対渦度
rDIV	44		1/s	相対発散
DIV	44	GVD	10^{-6}/s	相対発散
vUS	45		1/s	鉛直シヤーのｘ成分
vVS	46		1/s	鉛直シヤーのｙ成分
VWS	未定義		kt/1000ft	鉛直速度シアー
CrntD	47		degree true	流れの方向
CrntS	48		m/s	流れの速さ
CrntU	49		m/s	流れのｘ成分
CrntV	50		m/s	流れのｙ成分
Q	51		kg/kg	比湿
RH	52		Percent	相対湿度
HMR	53		kg/kg	混合比
TPW	54		kg/m^2	可降水量
VP	55		Pa	蒸気圧
VPVPD	56		Pa	飽差
Evap	57		kg/m^2	蒸発量
CIC	58		kg/m^2	雲氷
RRate	59		$kg/(m^2 \cdot s)$	降水率
ThndP	60		Percent	雷電の発生確率
RAIN	61	GVD	kg/m^2	総降水量
RR10	61		mm	前 10 分間降水量
RR60	61		mm	前 60 分間降水量
RR1H	61		mm	前 1 時間降水量
RR3H	61		mm	前 3 時間降水量
RR6H	61		mm	前 6 時間降水量
RR1D	61		mm	前 1 日間降水量
RR1M	61		mm	前 1 月間降水量
RRfr0	61		mm	正時からの降水量
RRL	62	GVD	kg/m^2	層状性降水量
RRLpD	62		mm	層状性降水量 (1 日当たり)
RRC	63	GVD	kg/m^2	対流性降水量
RRCpD	63		mm	対流性降水量 (1 日当たり)
SnRWe	64		$kg/(m^2 \cdot s)$	降雪率の水当量
SnWe	65		m	積雪の水当量
SnowD	66		m	積雪の深さ
MLD	67		m	混合層の厚さ
tTcD	68		m	非定常水温躍層の深さ
mTcD	69		m	主水温躍層の深さ
mTcan	70		m	主水温躍層の偏差
CLA	71	GVD		全雲量
CLC	72			対流雲の雲量
CLL	73	GVD		下層雲量
CLM	74	GVD		中層雲量
CLH	75	GVD		上層雲量
CWC			kg/kg	雲水量 (氷相を含む)
TCWC	76		kg/m^2	雲水量
BLI50	77		K	500hPa 面への最適持ち上げ指数
SnC	78		kg/m^2	対流性の降雪量

SnL	79		kg/m^2	ラージスケールの降雪量
WatrT	80		K	水温
SST	80	GVD	K	水温
Land	81		Proportion	陸域
Sldev	82		m	海面水位の平均値からの偏差
Z0	83	GVD	m	地表面粗度
Albed	84			アルベド
SoilT	85		K	土壌温度
SoilW	86			積算土壌水分量
Veget	87			植生被覆率
Sali	88		kg/kg	塩分
Dens	89		kg/m^3	密度
RunOf	90		kg/m^2	流出量
ROF	90	GVD	mm/day	流出量
ROFS	90	GVD	mm/day	表面排水による流出量
ROFD	90	GVD	mm/day	重力排水による流出量
IceC	91		Proportion	氷域
ICE	91	GVD	Proportion	氷域
IceD	92		m	氷の厚さ
IceMD	93		degree true	氷の移動方向
IceMS	94		m/s	氷の移動速度
IceMU	95		m/s	氷の移動速度のｘ成分
IceMV	96		m/s	氷の移動速度のｙ成分
IceGR	97		m/s	氷の成長率
IceDV	98		1/s	氷の発散
SNMlt	99		kg/m^2	融雪量
CWSSH	100		m	風浪とうねりの合成有義波高
WWvD	101		degree true	風浪の向き
WWvSH	102		m	風浪の有義波高
WWvMP	103		s	風浪の平均周期
SWvD	104		degree true	うねりの向き
SWvSH	105		m	うねりの有義波高
SWvMP	106		s	うねりの平均周期
PWvD	107		degree true	第１波の方向
PWvMP	108		s	第１波の平均周期
2WvD	109		degree true	第２波の方向
2WvMP	110		s	第２波の平均周期
RSNB	111	GVD	W/m^2	正味短波放射フラックス（地表面）
RLNB	112	GVD	W/m^2	正味長波放射フラックス（地表面）
RSNT	113	GVD	W/m^2	正味短波放射フラックス（大気上端）
RLNT	114	GVD	W/m^2	正味長波放射フラックス（大気上端）
RL	115		W/m^2	長波放射フラックス
RLUB	115	GVD	W/m^2	長波放射フラックス (上向き、地表面）
RLUBc	115		W/m^2	長波放射フラックス (上向き、地表面、晴天）
RLDB	115	GVD	W/m^2	長波放射フラックス (下向き、地表面）
RLUT	115	GVD	W/m^2	長波放射フラックス (上向き、大気上端）
RLDBc	115	GVD	W/m^2	長波放射フラックス (下向き、地表面、晴天）
RLUTc	115	GVD	W/m^2	長波放射フラックス (上向き、大気上端、晴天）
RS	116		W/m^2	短波放射フラックス
RSUB	116	GVD	W/m^2	短波放射フラックス（上向き、地表面）
RSDB	116	GVD	W/m^2	短波放射フラックス（下向き、地表面）
RSUT	116	GVD	W/m^2	短波放射フラックス（上向き、大気上端）
RSDT	116	GVD	W/m^2	短波放射フラックス（下向き、大気上端）

RSDBc	116	GVD	W/m^2	短波放射フラックス（下向き、地表面、晴天）
RSUBc	116	GVD	W/m^2	短波放射フラックス（上向き、地表面、晴天）
RSUTc	116	GVD	W/m^2	短波放射フラックス（上向き、大気上端、晴天）
RSDSn	116	GVD	W/m^2	短波放射フラックス（下向き、地表面、積雪内）
GlRad	117		W/m^2	全天日射フラックス
BrT	118		K	輝度温度
WNRad	119		$W/(m \cdot sr)$	放射（波数に関して）
WLRad	120		$W/(m^3 \cdot sr)$	放射（波長に関して）
FLLH	121	GVD	W/m^2	潜熱フラックス
FLSH	122	GVD	W/m^2	顕熱フラックス
BLDsp	123		W/m^2	境界層における散逸
FLMU	124	GVD	N/m^2	運動量フラックス（地表面風応力）ｘ成分
FLMV	125	GVD	N/m^2	運動量フラックス（地表面風応力）ｙ成分
WMixE	126		J	風の混合エネルギー
Image	127			画像資料
WatrT	128		K	水温
CLC2	129		Percent	雲量
AvTBB	130		K	ＴＢＢの平均値
MnTBB	131		K	ＴＢＢの最小値
SDTBB	132		K	ＴＢＢの標準偏差
SNCov	133		Percent	雪氷域
Tsun	134		J/m^2	全天日射量
HZanP	140			ジオポテンシャル高度の高偏差確率
PSprd	141			気圧のスプレッド
ZSprd	142			ジオポテンシャル高度のスプレッド
TSprd	143			気温のスプレッド
EAvSLP	200		hPa	海面更正気圧（アンサンブルメンバーの平均）
EAvZ	201		gpm	ジオポテンシャル高度（アンサンブルメンバーの平均）
EAvT	202		K	気温（アンサンブルメンバーの平均）
EAvU	203		m/s	風のｘ軸方向成分（アンサンブルメンバーの平均）
EAvV	204		m/s	風のｙ軸方向成分（アンサンブルメンバーの平均）
ESDSLP	210		hPa	海面更正気圧（アンサンブルメンバーの標準偏差）
ESDZ	211		gpm	ジオポテンシャル高度（アンサンブルメンバーの標準偏差）
ESDT	212		K	気温（アンサンブルメンバーの標準偏差）
ESDU	213		m/s	風のｘ軸方向成分（アンサンブルメンバーの標準偏差）
ESDV	214		m/s	風のｙ軸方向成分（アンサンブルメンバーの標準偏差）
RAM0	未定義			雨量換算係数
RAM1	未定義			雨量換算係数
RAM2	未定義			雨量換算係数
RAM3	未定義			雨量換算係数
TANKLV	未定義			土壌雨量タンクレベル値
TKRANK	未定義			土壌雨量履歴順位レベル値
cUVI	未定義			紫外線指数（晴天時）
wUVI	未定義			紫外線指数
TDSCS	未定義		g/m^3	ダスト下層濃度

TDSCI	未定義		kg/m^2	ダスト気柱積算量
CWMR	未定義		kg/kg	雲水混合比
CIMR	未定義		kg/kg	雲氷混合比
SkinT	未定義		K	地面からの長波放射量（温度換算値）
WindDM	未定義			レーダー VVP 風ベクトル　最大風速差
VVPSD	未定義			レーダー VVP 風ベクトル　標準偏差
VVPSN	未定義			レーダー VVP 風ベクトル　標本数
Altit	未定義		m	モデル地面の高度
FGSU	未定義	GVD		重力波抵抗短波運動量フラックスｘ成分
FGSV	未定義	GVD		重力波抵抗短波運動量フラックスｙ成分
FGLU	未定義	GVD		重力波抵抗長波運動量フラックスｘ成分
FGLV	未定義	GVD		重力波抵抗長波運動量フラックスｙ成分
LTRS	未定義	GVD		蒸散
LINT	未定義	GVD		キャノピー面にたまった水からの潜熱フラックス
MSC	未定義	GVD		キャノピーの水分量
MSG	未定義	GVD		地面・下草の水分量
TSC	未定義	GVD		キャノピーの温度
TSG	未定義			地面・下草の温度
ISC	未定義			キャノピーの氷
ISG	未定義			下草の氷
SoilI	未定義			積算土壌氷分量
SoilQ	未定義			土壌伝導熱
TSN	未定義			積雪表面温度
SnTmp	未定義			積雪温度
SnQ	未定義			積雪伝導熱
SnW	未定義			積雪の含水量
SnDen	未定義			積雪密度
SnFr	未定義		Proportion	積雪被覆率 (部分積雪の面積比率)
KIND	未定義	GVD		地表面状態
U1	未定義			モデル最下層の風のｘ成分
V1	未定義			モデル最下層の風のｙ成分
T1	未定義			モデル最下層の気温
Q1	未定義			モデル最下層の比湿
WET	未定義	GVD		土壌水分飽和度
UWV	未定義	GVD		水蒸気フラックスｘ成分
VWV	未定義	GVD		水蒸気フラックスｙ成分
RCST	未定義	GVD		放射強制力（短波・天頂）
RCSB	未定義	GVD		放射強制力（短波・地表面）
RCLT	未定義	GVD		放射強制力（長波・天頂）
RCLB	未定義	GVD		放射強制力（長波・地表面）
PBLH	未定義			境界層の高さ
HRRS	未定義		K/day	短波放射による気温の変化率（加熱率）
HRRL	未定義		K/day	長波放射による気温の変化率（加熱率）
HRCV	未定義		K/day	積雲対流による気温の変化率（加熱率）
HRLC	未定義		K/day	層状性降水による気温の変化率（加熱率）
HRVD	未定義		K/day	鉛直拡散による気温の変化率（加熱率）
HRAD	未定義		K/day	断熱過程による気温の変化率（加熱率）
HR	未定義		K/day	気温の変化率（加熱率）
QRCV	未定義		kg/(kg · day)	積雲対流による比湿の変化率
QRLC	未定義		kg/(kg · day)	層状性降水による比湿の変化率
QRVD	未定義		kg/(kg · day)	鉛直拡散による比湿の変化率
QRAD	未定義		kg/(kg · day)	断熱過程による比湿の変化率
URGW	未定義		m/(s · day)	重力波抵抗による u の変化率
URCV	未定義		m/(s · day)	積雲対流による u の変化率
URVD	未定義		m/(s · day)	鉛直拡散による u の変化率
URAD	未定義		m/(s · day)	断熱過程による u の変化率

VRGW	未定義	m/(s · day)	重力波抵抗による v の変化率
VRCV	未定義	m/(s · day)	積雲対流による v の変化率
VRVD	未定義	m/(s · day)	鉛直拡散による v の変化率
VRAD	未定義	m/(s · day)	断熱過程による v の変化率
CVR	未定義		雲量
UMF	未定義		上向きマスフラックス
UMB	未定義		雲底での上向きマスフラックス
CWF	未定義		雲仕事関数
MXWIN	未定義		最大風速面
TROP1	未定義		第 1 圏界面
TROP2	未定義		第 2 圏界面
CBTOP	未定義		積乱雲頂
NUM	未定義		地点番号
MLAT	未定義	deg	緯度
MLON	未定義	deg	経度
LAT	未定義	deg	緯度
LON	未定義	deg	経度
HIGH	未定義	m	高度
AQC	未定義		アメダス AQC 情報
Sunsh	未定義		日照時間のある割合
SSfr0	未定義	min	日照時間
SEC	未定義	s	時間
CSEC	未定義	s	時間
TDDKK	未定義		雷多重度＋放電種別
TEC	未定義	kA	雷推定電流
SM	未定義		マップファクター
PI10LV	未定義		レーダー降水強度 10 分レベル値
RR60LV	未定義		レーダー積算降水量 60 分レベル値
RR10LV	未定義		レーダー積算降水量 10 分レベル値
HIGHLV	未定義		レーダーエコー頂高度レベル値
SB_NUM	未定義		二次細分区番号 (注警報に使用しているもの)
GRDFLG	未定義		二次細分区における検証格子存在チェック
FO	未定義		検証四分割表における予報あり、観測あり
NFO	未定義		検証四分割表における予報なし、観測あり
FNO	未定義		検証四分割表における予報あり、観測なし
NFNO	未定義		検証四分割表における予報なし、観測なし
FCST	未定義		検証四分割表における予報ありの数
OBS	未定義		検証四分割表における観測ありの数
PS	1		地表面気圧
P1	1	hPa	モデル最下層気圧面
US	33	m/s	高度 10m における x 軸成分
VS	34	m/s	高度 10m における y 軸成分
TS	11	K	高度 2m における気温
QS	51	kg/kg	高度 2m における比湿
maxWS	未定義	m/s	前出力からの高度 10m における最大風速
CLAR	71		放射スキームの全雲量
CW	76	kg/m^2	鉛直積算雲液水量
TCWCR	76	kg/m^2	放射スキームの鉛直積算雲水量
XL0	未定義	m	混合距離
Sunsc	未定義	Proportion	晴天時の日照時間のある割合
Sn	未定義	mm/day	降雪強度 (雨換算)
CTOP	未定義	hPa	雲頂
CBASE	未定義	hPa	雲底
CVTOP	未定義	hPa	対流スキームにおける雲頂
FrCV	未定義		深い積雲対流の発生率
FrCVs	未定義		浅い積雲対流の発生率
UMBdf	未定義	$kg/(m^2 \cdot s)$	雲底での対流性下降流によるマスフラックス

UMBmc	未定義	kg/(m^2 · s)	雲底での中層対流によるマスフラックス
DCAPE	未定義		力学過程による CAPE 生成率
FrSc	未定義	Proportion	層積雲スキームの働く割合
CWCI	未定義	kg/kg	雲氷量
CWCW	未定義	kg/kg	雲液水量
ROFB	90	mm/day	底面排水による流出量
SMC	未定義		土壌水分量
VEG1	未定義	Proportion	キャノピー被覆率
VEG2	未定義	Proportion	下草被覆率
SMCFC	未定義	kg/m^3	圃場容水量 (Field Capacity)
WETFC	未定義		圃場容水量 (Field Capacity) における土壌含水量
SMCWP	未定義	kg/m^3	しおれ点における土壌水分量
WETWP	未定義		しおれ点における単位体積あたりの土壌水分量
SMC02	未定義		土壌 0 ～ 20cm の土壌水分
LSBL	未定義	W/m^2	昇華潜熱
FLG0	未定義	W/m^2	土壌への下向き熱フラックス
FLS0	未定義	W/m^2	雪への下向き熱フラックス
ALBSV	未定義		可視に対する雪アルベド
ALBSI	未定義		近赤外に対する雪アルベド
ST02	未定義	K	土壌 0 ～ 20cm の土壌温度
TSCn	未定義	K	キャノピーの温度 (雪なし域)
TSCs	未定義	K	キャノピーの温度 (雪あり域)
TSGn	未定義	K	地面・下草の温度 (雪なし域)
TSGs	未定義	K	地面・下草の温度 (雪あり域)
SoilTn	未定義	K	土壌温度 (雪なし域)
SoilTs	未定義	K	土壌温度 (雪あり域)
SoilWn	未定義		土壌液水率 (雪なし域)
SoilWs	未定義		土壌液水率 (雪あり域)
SoilIn	未定義		土壌氷率 (雪なし域)
SoilIs	未定義		土壌氷率 (雪あり域)
SnI	未定義	kg/m^2	積雪の含氷量
R1Dy	未定義	mm/day	日平均降水量
R1Dano	未定義	mm/day	日平均降水量偏差
Uano	未定義	m/s	風の x 軸成分偏差
Vano	未定義	m/s	風の y 軸成分偏差
RHano	未定義	Percent	相対湿度偏差
Qano	未定義	kg/kg	比湿偏差
SSTano	未定義	K	海面水温偏差
SMQR	未定義	kg/m^2	予報開始からの積算降水量（雨）
SMQS	未定義	kg/m^2	予報開始からの積算降水量（雪）
SMQG	未定義	kg/m^2	予報開始からの積算降水量（あられ）
SMQH	未定義	kg/m^2	予報開始からの積算降水量（ひょう）
RU	未定義		単位体積あたりの x 方向の運動量と座標変換係数との積
RV	未定義		単位体積あたりの y 方向の運動量と座標変換係数の積
PAIRF	未定義		無次元気圧
AsTide	未定義	m	天文潮位
SeaLev	未定義	m	潮位（＝天文潮位＋潮位偏差）
EOWvH	未定義	m	換算沖波波高

C 2次元座標系と地図投影法パラメタ

C.1 概要

種別 1 の 5 文字目と 6 文字目は NuSDaS のデータ記録が持つ 2 次元格子と空間位置との対応関係を表わす。水平 2 次元格子についてこの関係が数式で書けるようなものの場合、地図投影法 map projection ともよばれる。

以下の説明ではデータ記録の格子点が $(1,1)$, $(1,2)$, ... $(1,N_X)$, $(2,1)$, $(2,2)$, ... (N_X,N_Y) という格子番号 (i,j) をもつものとする。水平格子では (i,j) から地球上の経緯度 (λ,φ) を与える写像がわかれば、与えられた NuSDaS データの各格子が表現する位置を知ることができることになる。

地図投影では説明の便宜上、経緯度から必ずしも整数でない格子位置 x, y を求める式を紹介するが、各格子点から経緯度を求める際にはこの逆写像[注 1] $\lambda(x,y)$, $\varphi(x,y)$ に $x=i$, $y=j$ を与えれば経緯度を得ることになる。

気象用途で用いられる地図投影法は地図学で考案されたものが何でも用いられるわけでなく、数値予報モデルを構成するときの式変形の便宜、また天気図などの図形表現が歪まないようにするためなどの目的から正角 (または等角) 図法 conformal projection すなわち経線・緯線がどこでも直交し (次の C.1)、かつ方向によって局所的縮尺が異ならない (C.2) ものだけが用いられる。

$$\frac{\partial\lambda}{\partial x}\frac{\partial\varphi}{\partial x}+\frac{\partial\lambda}{\partial y}\frac{\partial\varphi}{\partial y} = 0 \tag{C.1}$$

$$\left(a\cos\varphi\frac{\partial\lambda}{\partial x}\right)^2+\left(\frac{\partial\varphi}{\partial x}\right)^2 = \left(a\cos\varphi\frac{\partial\lambda}{\partial y}\right)^2+\left(\frac{\partial\varphi}{\partial y}\right)^2 \tag{C.2}$$

ここで a は地球半径である。斜軸ランベルト以外では地球を球体としており、原則として日本測地系 2000 で用いる GRS80 楕円体の平均半径 6371 000 m が用いられる。

C.2 経緯度座標 (LL)

表式　経緯度座標は格子番号と経緯度に次のような線形関係があるものである[注 2]

$$\lambda = \lambda_0+(i-i_0)D_i \tag{C.3}$$

$$\varphi = \varphi_0-(j-j_0)D_j \tag{C.4}$$

ここで $D_j>0$ は北から南の順で格納することを意味することに注意されたい[注 3]。

定義ファイル　各定数は格子間隔 $(D_i,D_j$; 度単位), 参照格子点 (i_0,j_0), 参照点の経緯度 $(\lambda_0,\varphi_0$; 度単位) と呼ばれ、次のように定義ファイルに書かれる:

```
distance    Di   Dj
basepoint   i0   j0    λ0E   φ0N
```

参照点 (i_0,j_0) は通常 (1, 1) が用いられる。たとえば全球 1.25 度格子ならば次のようである:

```
size       288  145
basepoint  1    1    0E   90N
distance   1.25 1.25
```

パラメタの許容範囲　写像が出来るための数学的制約は $D_i\neq 0$, $D_j\neq 0$ だけであるが、常識的に $-180°<D_i<180°$, $-180°<D_j<180°$, 書法上 $-180°<\lambda_0\leq 180°$, $-90°\leq\varphi_0\leq 90°$, が要請される。

C.3 矩形ガウス格子 (GS)

表式　矩形ガウス格子 Gaussian grid[注 4]は経緯度座標に似ているが、y 方向の格子点がルジャンドル多項式 $P_n(\sin\varphi)$ の零点にとられる点が異なる。

[注 1] 必ずしも解析的に求められるとは限らない

[注 2] 地図学上は正距円筒図法 equidistant cylindrical projection あるいは正方形図法 equirectangular projection 仏 plate careé と呼ばれる。なお、自明とは思うがこの図法は直交だが正角ではない。

[注 3] 南から北の順で格納するためには $D_j<0$ とするが、現在用例はない。

[注 4] おそらく気象界でしか通用しない用語

ルジャンドル多項式は帯球関数とも呼ばれるもので

$$P_n(\mu) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(\mu^2-1)^n \tag{C.5}$$

$$= \sum_{s=0}^{s\le\frac{n}{2}}(-1)^s\frac{(2n-2s)!}{2^n s!(n-s)!(n-2s)!}\mu^{n-2s} \tag{C.6}$$

などで与えられる $\mu=\sin\varphi$ について n 次の多項式で $-1\le\mu\le 1$ の間に n 個の零点を持つ [3, 第 10 章]。

定義ファイル　この格子系は球面調和関数による直交関数展開を用いる全球スペクトルモデルで用いられ、このとき n は偶数である。経緯度格子の東西軸の情報に加えてルジャンドル多項式の次数 (モデルの切断波数) n がわかれば格子位置がわかるため定義ファイルでは次のように記述する:

`distance`	D_i	D_j		
`basepoint`	i_0	j_0	λ_0`E`	`0.0N`
`standard`	n`E`	`0N`	`0E`	`0N`

ここで j_0 は赤道に対応する格子番号 j (2 格子の中点なので半奇数) であり、全球モデルから得られた全格子ならば $D_i=360°/(2n)$, $j_0=(n+1)/2$ である。なお、D_j には D_i と同じ値を書くが、これは平均的な南北方向の格子間隔をおおざっぱに示したものである。次の例は T_L319 のガウス格子 (全球) を表わす。

```
size        640  320
basepoint   1.0  160.5  0.0E  0.0N
distance    0.5625  0.5625
standard    320E 0N 0E 0N
```

パラメタの許容範囲　NuSDaS インターフェイスでは経緯度座標と同様 $0<|D_i|<180°$, $0<|D_j|<180°$ を要請している。また、STANDARD 文から n が読み取れない場合 $n=180/D_j$ として推測する。これは、過去のデータの救済措置であり、size, basepoint, distance, standard の省略はしないこと。

注意　矩形ガウスであっても、SUBC RGAU とともに用いる場合は、適合ガウス格子の特殊な場合として扱い、種別　1 の 2 次元座標の名称を「RG」と設定すること。

C.4　適合ガウス格子 (RG)

表式　適合ガウス格子 reduced Gaussian grid は矩形ガウス格子の東西方向の格子数を緯度によって可変にすることで高緯度地方の格子密度を低緯度並みに減らしたものである。なお、東西方向の格子数を緯度によらず一定とした場合、矩形ガウス格子となる。

東西格子は全円周を等分するので

$$\lambda=\lambda_0+(i-i_0)\frac{2\pi}{N_{X,j}} \tag{C.7}$$

のようであり、東西格子数 $N_{X,j}$ はそれぞれの緯度 (または南北格子番号 j) によって異なる。NuSDaS の適合ガウス格子は経緯度で矩形の領域を切り出して保存することを想定しているため、東西格子番号 i の範囲も緯度 (または j) によって異なる。これらの配列は SUBC RGAU レコードに格納される。

定義ファイル　NuSDaS の 2 次元配列は矩形のものだけを前提にしているので、定義ファイル上は適合ガウス格子は 1 次元の配列として扱う。standard, distance, basepoint は省略可とするが、書く場合は、GS のルールに準拠すること。

```
size        1573 1
standard    320E 0N 0E 0N   (省略可)
distance    0.5625  0.5625 (省略可)
```

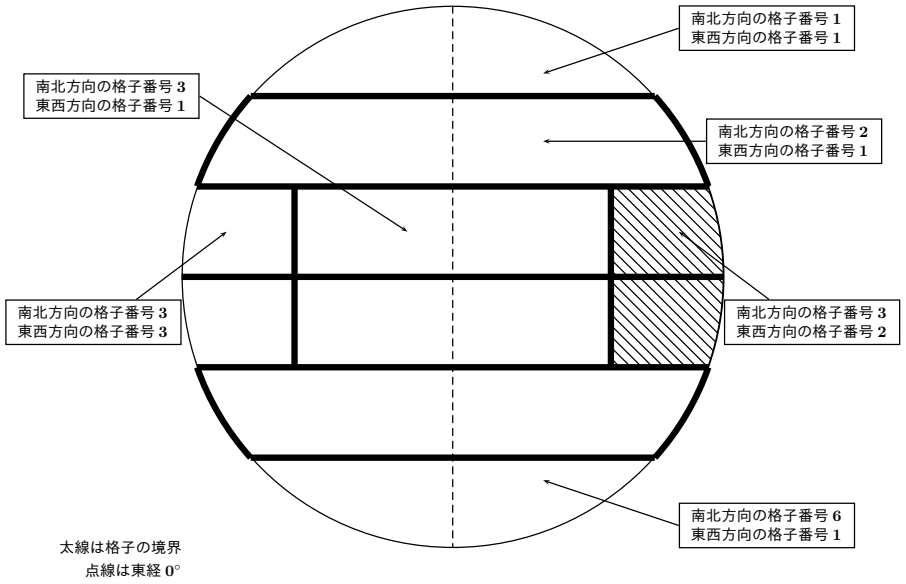

Figure C.1: 適合ガウス格子の例

パラメタの許容範囲　特に、チェックは行わない。よって、格子情報に関する定義は、SUBC RGAU の情報を使用すること。

注意　SUBC RGAU レコードをもつ矩形ガウス格子も適合ガウス格子の特殊な場合として扱う。

例　Figure C.1 のように全球を分割する場合を考える (ガウス格子では緯度の間隔は一定ではないが、ここでは簡単にするため等間隔の場合を考える)。全球のすべての格子を格納する場合は、SUBC RGAU に設定する値は
`j = 6`, `j_start = 1`, `j_n = 6`,
`i(1) = 1`, `i(2) = 2`, `i(3) = 3`, `i(4) = 3`, `i(5) = 2`, `i(6) = 1`,
`i_start(1) = 1`, `i_start(2) = 1`, `i_start(3) = 1`,
`i_start(4) = 1`, `i_start(5) = 1`, `i_start(6) = 1`,
`i_n(1) = 1`, `i_n(2) = 2`, `i_n(3) = 3`, `i_n(4) = 3`, `i_n(5) = 2`, `i_n(6) = 1`,
`lat(1) = 75.0`, `lat(2) = 45.0`, `lat(3) = 15.0`, `lat(4) = -15.0`, `lat(5) = -45.0`, `lat(6) = -75.0`
とする。このとき、データは 1 次元配列に

(1,1)	(1,2)	(2,2)	(1,3)	(2,3)	(3,3)	(1,4)	(2,4)	(3,4)	(1,5)	(2,5)	(1,6)

のように格納する。

Figure C.1 の斜線部分のみを格納する場合は
`j = 6`, `j_start=3`, `j_n = 2`,
`i(1) = 1`, `i(2) = 1`,
`i_start(1) = 2`, `i_start(2) = 2`,
`i_n(1) = 1`, `i_n(2) = 1`,
`lat(1) = 15.0`, `lat(2) = -15.0`,
とする。このとき、データは 1 次元配列に

(2,3)	(2,4)

のように格納する。

C.5 メルカトル図法 (MR)

表式 メルカトル図法 Mercator projection は世界地図などによく用いられる円筒図法で正角図法でもある。気象庁では一部の FAX 図や低緯度の台風モデル(注 5)などで用いられている。

投影の表式は次のようである(注 6):

$$(x - x_0)D_X = R(\lambda - \lambda_0) \tag{C.8}$$

$$(y - y_0)D_Y = R\ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right] - R\ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi_0}{2}\right)\right] \tag{C.9}$$

$$= R\tanh^{-1}(\sin\varphi) - R\tanh^{-1}(\sin\varphi_0) \tag{C.10}$$

$$R = a\cos\varphi_1 \tag{C.11}$$

ただしここで $\tanh^{-1}$ は tanh の逆関数であり逆数ではない。

定義ファイル この投影を決定するために必要なパラメタは参照格子点 x_0, y_0, 参照経緯度 λ_0, φ_0, 格子間隔 D_X, D_Y (メートル単位, D_i などと混同しないように) および標準緯度(注 7)φ_1 であり、定義ファイルに次のように書かれる。

`distance`	D_X	D_Y		
`basepoint`	x_0	y_0	λ_0`E`	φ_0`N`
`standard`	`0E`	φ_1`0N`	`0E`	`0N`

パラメタの許容範囲 投影法は $-90° < \varphi_1 < 90°$, $D_X \neq 0$, $D_Y \neq 0$ を要請する。また NuSDaS インターフェイスによって $0 \leq |\lambda_0| < 180°$, が要請される。理論上何ら悪いことはないのだが、$|D_X| < 180$m または $|D_Y| < 180$m という場合は経緯度格子と間違えている可能性が高いので、エラーではないが警告が表示される。

C.6 ポーラーステレオ図法 (PS)

表式 ポーラーステレオ図法 polar stereographic projection は高緯度の天気図や航空向け GPV プロダクトなどに用いられる方位図法で正角図法でもある。表式は次のようである。

$$(x - x_0)D_X = R\sin(\lambda - \lambda_1) - R_0\sin(\lambda_0 - \lambda_1) \tag{C.12}$$

$$(y - y_0)D_Y = R\cos(\lambda - \lambda_1) - R_0\cos(\lambda_0 - \lambda_1) \tag{C.13}$$

$$R = a(1 + \sin\varphi_1)\tan\frac{\frac{\pi}{2} - \varphi}{2} \tag{C.14}$$

$$R_0 = a(1 + \sin\varphi_1)\tan\frac{\frac{\pi}{2} - \varphi_0}{2} \tag{C.15}$$

参照点への平行移動や標準緯度 (φ_1) による縮尺調整の関係で式が長いが、要は極からの半径が $a(1+\sin\varphi_1)\tan[\frac{1}{2}(\frac{\pi}{2}-\varphi)]$ といっているに過ぎない。南極に光源を置いて北極に接するように置いた平面に映る影がステレオ図法だと説明されることが多いが、ポーラーステレオについては間違っていない(注 8)。ここで標準緯度とは、縮尺を変えない円周を与える緯度であり、つまり投影面を置く緯度に他ならない。

定義ファイル 投影を同定するのに必要なパラメタは (x_0, y_0), (λ_0, φ_0) および標準緯度 φ_1 と標準経度 λ_1 であり定義ファイルに次のように書かれる。

`distance`	D_X	D_Y		
`basepoint`	x_0	y_0	λ_0`E`	φ_0`N`
`standard`	λ_1`E`	φ_1`N`	`0E`	`0N`

(注 5) 台風モデルは中緯度ではランベルト正角円錐図法となり、事前に投影法が定めがたいため、データセット種別名では `_TYMXXET` などのように `XX` を用いる。

(注 6) 巷間よく行われる説明では地球中心に光源を置き地球に巻きつけた円筒に写る像だといっているがあれは子供だましで正角図法にはならない。正しくは経線等間隔を仮定して式 C.2 (p. 164) を南北に積分する。

(注 7) D_X は標準緯度における東西方向の長さである。

(注 8) ヒントは円周角の定理

次の例は標準緯度 60°N で 140°E が y 軸になるように投影した面の中で 40km 間隔 83×71 格子の (65, 53) が (30°N, 140°E) に来るように配置したものを意味する。

```
size        83   71
basepoint   65.0   53.0   140.0E  30.0N
distance    40000.0  40000.0
standard    140.0E 60.0N 0.0E 0.0N
```

パラメタの許容範囲　投影法は $0 < |\varphi_1| < 90°$, $D_X \neq 0$, $D_Y \neq 0$ を要請する。NuSDaS インターフェイスはメルカトル図法のチェックに加えて $|\lambda_1| \leq 180°$ を要請する。

C.7　ランベルト正角円錐図法 (LM)

表式　ランベルト正角円錐図法 Lambert conformal conic projection (正角割円錐図法 などいろいろに呼ばれる) はポーラーステレオに比べれば中緯度での縮尺の変動が小さい円錐図法で、正角図法であることから中緯度の天気図や領域モデルによく用いられる。

$$
\begin{aligned}
(x - x_0)D_X &= R\sin[n(\lambda - \lambda_1)] - R_0 \sin[n(\lambda_0 - \lambda_1)] && \text{(C.16)}\\
(y - y_0)D_Y &= R\cos[n(\lambda - \lambda_1)] - R_0 \cos[n(\lambda_0 - \lambda_1)] && \text{(C.17)}\\
R &= F\tan^n\left(\frac{\frac{\pi}{2} - \varphi}{2}\right) && \text{(C.18)}\\
R_0 &= F\tan^n\left(\frac{\frac{\pi}{2} - \varphi_0}{2}\right) && \text{(C.19)}\\
F &= \frac{a}{n}\cos\varphi_1 \tan^{-n}\left(\frac{\frac{\pi}{2} - \varphi_1}{2}\right)\\
&= \frac{a}{n}\cos\varphi_2 \tan^{-n}\left(\frac{\frac{\pi}{2} - \varphi_2}{2}\right) && \text{(C.20)}\\
n &= \frac{\ln\cos\varphi_1 - \ln\cos\varphi_2}{\ln\tan\left(\frac{\frac{\pi}{2}-\varphi_1}{2}\right) - \ln\tan\left(\frac{\frac{\pi}{2}-\varphi_2}{2}\right)} && \text{(C.21)}
\end{aligned}
$$

なお、$\varphi_1 = \varphi_2$ の特殊な場合は 式 C.21 (p. 168) は零割る零になってしまうので計算できず、かわりに極限である $n = \sin\varphi_1$ とする。

定義ファイル　この投影法を同定するために必要なパラメタは x_0, y_0, λ_0, φ_0, および 標準経度 λ_1 と 2 つの標準緯度 φ_1, φ_2, であり、定義ファイルには次のように書かれる (λ_1 は重出)。

```
distance    D_X    D_Y
basepoint   x_0    y_0     λ_0E    φ_0N
standard    λ_1E   φ_1N    λ_1E    φ_2N
```

次の例は RSM で用いられているもので、標準緯度 (30°N, 60°N) として 140°E が y 軸となるように投影した面の中で 20km 間隔 325×257 格子の (200, 185) が (30°N, 140°E) に来るように配置したものを意味する。

```
size        325 257
basepoint   200.  185.  140.0E  30.0N
distance    20000. 20000.
standard    140.0E  30.0N 140.0E  60.0N
```

パラメタの許容範囲　投影法が $0 < |\varphi_1| \leq |\varphi_2| < 90°$, $\varphi_1\varphi_2 > 0$, $D_X \neq 0$, $D_Y \neq 0$ を要請する。NuSDaS インターフェイスは**ポーラーステレオの条件に加えて**上のチェックを行う。

C.8 斜軸ランベルト図法 (OL)

表式 斜軸ランベルト図法は基本的に上述のランベルト図法の投影中心を北極から中緯度の適当な地点に移したもので、マップファクターの等しい線が緯線ではなく経線・緯線と斜めに交わる任意の小円に設定できて、日本列島のように描画対象が弧状に存在している場合に図全体での縮尺の差を小さくすることができることに意義がある。ただし地球の楕円体性の考慮の方式にいろいろあり、単に斜軸ランベルトというだけでは具体的地図投影写像はひとつには決まらない。

経緯度から地図座標への変換は大きく分けて 3 段階からなる。

- 回転楕円体等角写像
- 斜軸回転
- ランベルト正角円錐図法

まず回転楕円体等角写像は回転楕円体上の点(注 9) (λ, φ) を球面上の点 $(\hat{\lambda}, \hat{\varphi})$ に対応付けるもので、次のようなものである:

$$\hat{\lambda} = c(\lambda - \lambda_E) + \lambda_E \tag{C.22}$$

$$\hat{\varphi} = 2\tan^{-1}\left[\tan\frac{\frac{\pi}{2}+\varphi}{2}\left(\frac{1-e\sin\varphi}{1+e\sin\varphi}\right)^{\frac{e}{2}}\right]^c - \frac{\pi}{2} \tag{C.23}$$

$$c = \sqrt{\frac{1+e^2\cos^4(\varphi_E)}{1-e^2}} \tag{C.24}$$

ここでパラメタ e は回転楕円体の離心率と呼ばれ GRS80 系の 0.081819218 が用いられる。なお、Snyder [1] によれば

$$\hat{\lambda} = \lambda \tag{C.25}$$

$$\hat{\varphi} = 2\tan^{-1}\left[\tan\frac{\frac{\pi}{2}+\varphi}{2}\left(\frac{1-e\sin\varphi}{1+e\sin\varphi}\right)^{\frac{e}{2}}\right] - \frac{\pi}{2} \tag{C.26}$$

のようにしても等角写像を得ることができて、もちろんこのほうが 2 つのパラメタが不要で簡単だし全球が写像できるメリットもあるのだが座標値が 10km 単位でずれるので数値予報標準ライブラリ以外の方法で地図投影している GIS ソフトウエアの利用にあたっては注意を要する。

次に行われる斜軸回転とは、正軸球座標における $(\hat{\lambda}_P, \hat{\varphi}_P)$ を新座標における北極 $(\varphi' = \pi/2)$ に対応付け、正軸球座標における北極を新座標における経度ゼロとするような新しい球座標 (λ', φ') であり、次のような式が用いられている。

$$\varphi' = \sin^{-1}\left[\sin\hat{\varphi}_P\sin\hat{\varphi} + \cos\hat{\varphi}_P\cos\hat{\varphi}\cos(\hat{\lambda}-\hat{\lambda}_P)\right] \tag{C.27}$$

$$\lambda' = \tan^{-1}\frac{\cos\hat{\varphi}\sin(\hat{\lambda}-\hat{\lambda}_P)}{\cos\hat{\varphi}_P\sin\hat{\varphi} - \sin\hat{\varphi}_P\cos\hat{\varphi}\cos(\hat{\lambda}-\hat{\lambda}_P)} \tag{C.28}$$

この変換は地軸に関する経度方向の $\hat{\lambda}_P$ 回転と東経 90 度軸に関する $\hat{\theta}_P = \frac{\pi}{2} - \hat{\varphi}_P$ だけの回転の合成であるはずだから

$$\begin{pmatrix}\cos\varphi'\cos\lambda'\\ \cos\varphi'\sin\lambda'\\ \sin\varphi'\end{pmatrix} = \begin{pmatrix}\cos\hat{\theta}_P & 0 & -\sin\hat{\theta}_P\\ 0 & 1 & 0\\ \sin\hat{\theta}_P & 0 & \cos\hat{\theta}_P\end{pmatrix}\begin{pmatrix}\cos\hat{\lambda}_P & \sin\hat{\lambda}_P & 0\\ -\sin\hat{\lambda}_P & \cos\hat{\lambda}_P & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\hat{\varphi}\cos\hat{\lambda}\\ \cos\hat{\varphi}\sin\hat{\lambda}\\ \sin\hat{\varphi}\end{pmatrix}$$

$$= \begin{pmatrix}\sin\hat{\varphi}_P\cos\hat{\varphi}\cos(\hat{\lambda}-\hat{\lambda}_P) - \cos\hat{\varphi}_P\sin\hat{\varphi}\\ \cos\hat{\varphi}\sin(\hat{\lambda}-\hat{\lambda}_P)\\ \cos\hat{\varphi}_P\cos\hat{\varphi}\cos(\hat{\lambda}-\hat{\lambda}_P) + \sin\hat{\varphi}_P\sin\hat{\varphi}\end{pmatrix}$$

となるが、式 C.28 (p. 169) の λ' と符号が逆であるから λ' は西経が正となっていることを意味する。

最後のランベルト正角円錐図法による地図投影は、式 C.16–C.21 の経緯度 λ', φ' に斜軸球面経緯度 λ', φ' をあてはめればよいが、次のようなことに注意されたい。

(注 9) 一般に用いられる経緯度は回転楕円体上の経緯度である。

- パラメタ、特に λ_1 は斜軸座標の値である
- 気象庁の斜軸ランベルトでは極から参照点に向かう軸 (南東向き) を x, それに右向き直交する軸 (南西向き) を y としているので式 C.16–C.21 の x, y を入れ換えたものにあたる

定義ファイル　この投影法を同定するために必要なパラメタは正軸ランベルトのパラメタに加えて斜軸回転のパラメタ $\hat{\lambda}_P, \hat{\varphi}_P$, と楕円体等角写像のパラメタ λ_E, φ_E, であり、定義ファイルには次のように書かれる。

`distance`	D_X	D_Y		
`basepoint`	x_0	y_0	λ_0E	φ_0N
`standard`	λ_0E	φ_1N	λ_0E	φ_2N
`others`	$\hat{\lambda}_P$E	$\hat{\varphi}_P$N	λ_EE	φ_EN

STANDARD 文にも λ_0 が書かれておりランベルト投影の標準経度 λ_1 が独立に示されていないが、気象庁では参照点 λ_0, φ_0 を楕円体等角写像・斜軸回転して得る斜軸経度を用いるためである。次の例は 1km 格子である。

```
size        1600 3600
distance    1000 1000
basepoint   1053 1403 35.3572N 138.7306E
standard    45.8183N 138.7306E 50.9429N 138.7306E
others      56.1920N 82.7382E 37.0N 137.0E
```

投影法パラメタの由来　気象庁で用いられている斜軸ランベルト図法は格子間隔と参照点格子番号を除いて上の定義ファイル例のパラメタを用いているが、これは国土地理院が 1972 年に発行した 300 万分の 1 「日本とその周辺」図の投影法に由来する。$\lambda_E = 137°, \varphi_E = 37°$ は別にすると $\hat{\lambda}_P = 82°44'17''.4$, $\hat{\varphi}_P = 56°11'31''.3$, $\varphi_1 = 45°49'06''.0$, $\varphi_2 = 50°56'34''.4$ は大変切りの悪い数値であるが、導線 (縮尺が最小となる線で斜軸座標の緯線) が占守島 (50°46′N, 156°03′E)[注 10] 名古屋 (35°10′N, 136°58′E), 台北 (25°02′N, 121°31′E), を通り、その縮尺が 0.999 となるように決められているという [2]。参照点 $\lambda_0 = 138°43'50''$, $\varphi_0 = 35°21'26''$ はレーダーエコー合成図に由来し、富士山の経緯度である [4]。

当時のことであるから (特に富士山の) 経緯度は旧日本測地系によるものであるが、現在でも前節の投影法パラメタをそのままに地球の形だけ GRS80 系に変更して用いている。理論上この斜軸座標は 2002 年 4 月 1 日の改正測量法施行時あるいは地球の形として GRS80 系が採用された NAPS8 の移行時に 400–500m ほどシフトされたことになるが、それが問題になるプロダクトは今のところ存在しない。

パラメタの許容範囲　現在のところ正軸ランベルトと同じチェックが行われる。

C.9　子午面断面図 (YP)

定義ファイル　鉛直 22 層、南北 73 格子の例を次に示す。

```
plane 1
plane1 ZONAL
size 73 22
basepoint 1 1 0E 90N
distance 2.5 2.5
```

パラメタの許容範囲　$0 < |D_J| < 180°$, $|\lambda_0| \leq 180°$, $|\varphi_0| \leq 90°$ が要請される。

[注 10] 占守島は幌筵島 (32215 SEVERO-KRIL'SK 50:41N, 145:08E) より東だからこの経度は明らかに間違っている。ちなみに名古屋と台北の経緯度はちょうど 47636 と 46692 に一致する。

C.10 東西断面図 (XP)

定義ファイル 鉛直 1 層、東西 360 格子の例を次に示す。

```
size 360 1
basepoint 1 1 0E 0N
distance 1.0 1.0
```

パラメタの許容範囲 $0 < |D_I| < 180°$, $|\lambda_0| \leq 180°$, $|\varphi_0| \leq 90°$ が要請される。

C.11 レーダー画像 (RD)

定義 国内二進電文の資料から察してレーダーサイトを中心とした正距方位図法の投影面上の等間隔格子であると解されるが投影法諸元を掲げることができるような厳密な定義は調査中である。

定義ファイル 格子間隔が書かれる。メンバー毎に違うレーダーサイトを表現する多メンバーのデータセットにすることが多いので参照点は書かれないのが普通である。また、レーダーデータについては通常 `value REPR` が書かれる。

```
size        200 200
distance    2500 2500
value       REPR
```

パラメタの許容範囲 投影法は $D_X \neq 0$, $D_Y \neq 0$ を要請する。理論上何ら悪いことはないのだが、$|D_X| < 180\text{m}$ または $|D_Y| < 180\text{m}$ という場合は経緯度格子と間違えている可能性が高いので、エラーではないが警告が表示される。また、BASEPOINT 文 (4.2 節, p. 34) を書いた場合は経緯度が過大だとエラーになる。

注意 なお、次項で述べる極座標に `RD` を用いている例があるため注意を要する。このようなデータは D_X に比べて D_Y だけが桁違いに小さい。

C.12 レーダー極座標格子 (RT)

定義 X (r) 軸が距離、Y (θ) 軸が方位角である。したがって通常 $D_\theta = 360°/N_Y$ である。方位角の原点は北であると思うがデータ作成元に確認されたい (将来の版ではこの点を明確化する必要がある)。

定義ファイル 次のような形式で記述する。

`size`	N_X	N_Y
`distance`	D_r	D_θ

具体例を挙げると次のようになる。

```
size        200  256
distance    1500  1.40625
value       REPR
```

パラメタの許容範囲 投影法は $D_r \neq 0$, $0 < |D_\theta| < 180°$ を要請する。理論上何ら悪いことはないのだが、$|D_r| < 180\text{m}$ という場合は経緯度格子と間違えている可能性が高いので、エラーではないが警告が表示される。

C.13 地点 (ST)

表式 アメダスデコードについて利用されている。この格子系のデータについては、データ要素として経緯度 LAT, LON, または MLAT, MLON, が格納されていなければ、各格子点の位置についてはデータ作成元に問い合わせるほかない。

定義ファイル 投影法関連の文 [DISTANCE 文 (4.5 節, p. 35) 等] は通常省略される。

パラメタの許容範囲 特段のチェックは行われない。

C.14 細分 (SB)

表式 検証デコードについて利用されている。この格子系のデータについては、データ要素として、細分番号 SB_NUM, が格納されていなければ、各格子点の位置についてはデータ作成元に問い合わせるほかない。

定義ファイル 投影法関連の文 [DISTANCE 文 (4.5 節, p. 35) 等] は通常省略される。

パラメタの許容範囲 特段のチェックは行われない。

C.15 自由格子 (FG)

表式 河川に沿った格子について利用されている。この格子系のデータについては、各格子点の位置についてはデータ作成元に問い合わせるほかない。

定義ファイル 投影法関連の文 [DISTANCE 文 (4.5 節, p. 35) 等] は通常省略される。

パラメタの許容範囲 特段のチェックは行われない。

C.16 その他の格子 (XX)

この「座標系」は台風モデルによって用いられている。台風モデルは台風の中心位置が低緯度のときメルカトル図法、中緯度のときランベルト正角円錐図法となるので、事前に種別名を定めておきたい (あるいは、台風モデルの出力をひとつのデータセットに蓄積したい) という目的から便宜上 XX を用いる。台風モデルは実行時に nusdas_grid (p. 58, 100) を用いて投影法パラメタを設定している。

このような運用に配慮して、2 次元座標系が XX のときはデータファイルを作成しようとするときに行われる投影法パラメタチェックが抑止される。そのかわり、ファイルを閉じようとするさいに依然として投影法が XX であれば、nusdas_grid を呼び忘れたものとみなしてエラーになる。

D 3次元座標系と鉛直座標パラメタ

D.1 気圧座標 (PP)

気圧 p を独立変数にした座標系である。座標を構成するにあたって必要なパラメータはない。

面名は hPa 単位で数値を 6 文字の文字列 (左詰めで余りには空白を埋める) で表す。

D.2 エータ座標 (ET)

$\sigma - p$ 座標とも呼ばれ、下層では地形に沿った σ 座標系、上層では水平な p 座標系になっている (Figure D.1)。

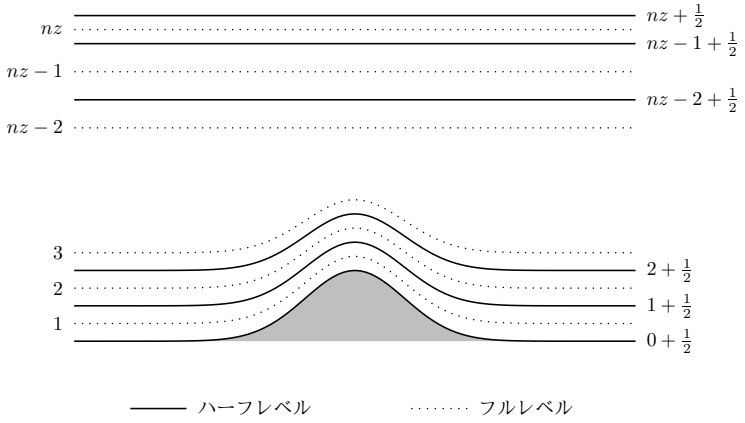

Figure D.1: エータ座標系の模式図 (GSM, RSM など)。地表面はハーフレベルの最下層に一致する。この地表面まで含めたときにハーフレベルの層数は鉛直層数 (フルレベルの層数) より 1 だけ多いことに注意。そのため、ハーフレベルに対応する値を格納する SUBC ETA の A, B の要素数が (鉛直層数+1) 必要になる。

各点の気圧は、SUBC ETA レコードに格納されている係数 A, B, C を用いて

$$p(x, y, z) = B(z)[p_G(x, y, z) - C] + A(z)$$

と表現される (実際には $C = 0$ である)。

A, B は、ハーフレベルに対応した値が格納されており(注 1)、ハーフレベルの最下層が地表面に一致するようになっている。また、ハーフレベルはフルレベルより 1 層多いので、A, B は鉛直層数 (フルレベルの層数) に 1 を加えた数の配列を確保する必要がある。

データは通常、フルレベルの値が格納され、層名は下からつけた番号を 6 文字の文字列 (左詰めで余りには空白を埋める) で表す。

D.3 シグマ座標 (SG)

ある一定の気圧を p_T、地表面の気圧を p_G、各点の気圧を p としたとき、

$$\sigma(x, y, z) = \frac{p(x, y, z) - p_T}{p_G(x, y) - p_T}$$

で定義された変数を独立変数にする座標系である。

地形に沿った座標系で、地球表面に山があっても、地表面は $\sigma = 1$ で表される。

各点の気圧は、ETA 座標と同様に、SUBC SIGM レコードに格納されている係数 A, B, C を用いて

$$p(x, y, z) = B(z)[p_G(x, y, z) - C] + A(z)$$

(注 1) p はハーフレベルに配置されている。

と表現される (A, B, C については、ETA と同じ要素数を持つ配列または変数)。A が p_T に、B が σ に、C が p_T に対応する(注 2)。

データは通常、フルレベルの値が格納され、層名は下からつけた番号を 6 文字の文字列 (左詰めで余りには空白を埋める) で表す。

D.4 ハイブリッド気圧座標 (HB)

用例はない。名前だけ予約されていると解されたい。鉛直座標パラメタを格納するために SUBC ETA レコードで対応できるか否かも未定である。

D.5 バイブリッド鉛直座標 (Z*座標の拡張)(ZS)

一般に、絶対高度座標 z と変換した座標系の座標 ζ が

$$z = \zeta + z_s f(\zeta)$$

の関係式に表される座標系である。Z*座標系 (ζ を z^* と標記する)

$$z^* = \frac{z_T(z - z_s)}{H - z_s}$$

は、

$$f(\zeta) = 1 - \frac{\zeta}{z_T}$$

としたものである。ここで、z_s は地形、z_T はモデルトップである。

運動方程式に中に現れる座標変換に伴うテンソルは

$$\begin{aligned} G^{\frac{1}{2}} &= 1 + z_s f'(\zeta) \\ G^{\frac{1}{2}} G^{13} &= -f(\zeta)\frac{\partial z_s}{\partial x} \\ G^{\frac{1}{2}} G^{23} &= -f(\zeta)\frac{\partial z_s}{\partial y} \end{aligned}$$

と表現できて、高度 z やテンソルの算出にはモデル面の高度 ζ、$f(\zeta)$ および $f'(\zeta)$ があればよい。

NuSDaS では SUBC ZHYB に、モデル面の高度 ζ を `zrp`(フルレベル)、`zrw`(ハーフレベル)、$f(\zeta)$ を `vctrans_p`(フルレベル)、`vctrans_w`(ハーフレベル)、$f'(\zeta)$ を `dvtrans_p`(フルレベル)、`dvtrans_w`(ハーフレベル) として格納されている。

D.6 高度による鉛直座標 (ZZ)

(地形を考慮しない) 高度を鉛直座標にとった座標系である。

D.7 温位座標 (TH)

温位を鉛直座標にとった座標系である。用例は確立されていないが、温位がケルビン単位の整数であれば単にその値を `printf("%g")` したものを面名にするのが適当である。

D.8 経度 (LO)

二次元座標系が `YP` の時に用いられる。実際には経度の値が用いられることはなく、東西平均 `ZONAL` だけが用いられる。

そうでないときはおとなしく `LL PP` を使ってもらいたい。

(注 2) p_T はスカラーであるが、ETA とデータ構造を同じにするため、配列である A に格納している。格納の際には、すべての要素に同じ値を格納するようにする。

D.9 緯度 (LA)

二次元座標系が `XP` の時に用いられるため予約された名前である。用例はない。

D.10 閾値 (TO)

閾値を面とみなして扱うもので、実際の鉛直座標系ではない。主に、検証で使用する。

D.11 その他の鉛直座標 (XX)

GRIB 等のデコーダで未知の座標を表現するために予約された名前であり、積極的に使うべきでない。

E パッキング

E.1 一般論

パッキング (packing) とは、数値データを圧縮保存する技法のひとつで、線形変換によってデータをビット数の少ない別の型 (たいていは整数) に近似的に変換することによって使用ビット数を減らすものです。以下の `1PAC`, `2PAC`, `2UPC`, `4PAC` がその (本来の) パッキングにあたります (数字がバイト数を表します)。この圧縮は不可逆で、データを `nusdas_write` してから `nusdas_read` しても元のデータとは一致しません。

NuSDaS ではパッキングという言葉が少々拡大解釈されていて、定義ファイルの PACKING 文 4.17 節 (p. 40) などによってパッキングだけでなく、ランレングス圧縮 `RLEN` などの圧縮 (compression) や数値型をそのまま保存する方式が選択できるようになっています。

E.2 1PAC

8 ビット (1 バイト) 符号なし整数で保存するパッキングです。

エンコード (データ列 y_i からファイルに書くべき整数列 n_i を求めるとき) には次のようにします:

$$b := \min_i(y_i) \tag{E.1}$$

$$a := \frac{\max_i(y_i) - \min_i(y_i)}{252} \tag{E.2}$$

$$n_i := \mathrm{floor}\left[\frac{y_i - b}{a} + 0.5\right] \tag{E.3}$$

ここで、floor$[x]$ は小数部切り捨てです。式から明らかですが $0 \le n_i \le 252$ となりますので 8 ビット符号なし整数で表現できるわけです。データ列がすべて同じ値の場合は、$a = 1, n_i = 0$ とされます。ちなみに a の名前は amplitude (振幅), b の名前は base (ここでは最小値のつもりの和製英語) からきています。

デコード (ファイルから読んだ a, b, n_i からデータを復元するとき) は

$$y_i' := an_i + b \tag{E.4}$$

として、もとのデータに近い浮動小数点数 y_i' を復元します。

`1PAC` の読み書きは、ユーザー配列の型が `I2`, `I4`, `R4`, `R8` でなければなりません。

E.3 2PAC

16 ビット (2 バイト) **符号付き**整数で保存するパッキングです。**これを使ってはいけません。**

エンコードは次のようになっています:

$$b := \frac{\max_i(y_i) + \min_i(y_i)}{2} \tag{E.5}$$

$$a := \frac{\max\left[\max_i(y_i) - b, b - \min_i(y_i)\right]}{32765} \tag{E.6}$$

$$n_i := \mathrm{floor}\left[\frac{y_i - b}{a} + 0.5\right] \tag{E.7}$$

デコードの方法は 1PAC と同じです。かつて a の決め方が悪く $n_i > 32767$ となってとんでもない値がデコードされるバグがあったのは修正されています。しかし最小値が再現しないため、積算降水量の差をとると負の降水量が生じる、といった問題は本質的に解消不能なため、NAPS8 以降の数値予報ルーチンでは 2PAC の使用は禁止されています。

`2PAC` の読み書きは、ユーザー配列の型が `I4`, `R4`, `R8` でなければなりません。

E.4 2UPC

16 ビット (2 バイト) 符号なし整数で保存するパッキングです。

エンコードは次のようです:

$$b := \min_i(y_i) \tag{E.8}$$

$$a := \frac{\max_i(y_i) - \min_i(y_i)}{65532} \tag{E.9}$$

$$n_i := \mathrm{floor}\left[\frac{y_i - b}{a} + 0.5\right] \tag{E.10}$$

2UPC の読み書きは、ユーザー配列の型が I4, R4, R8 でなければなりません。

E.5 4PAC

32 ビット (4 バイト) 符号つき整数で保存するパッキングです。あまり使われません。
エンコードは次のようです:

$$b := \frac{\max_i(y_i) + \min_i(y_i)}{2} \tag{E.11}$$

$$a := \frac{\max\left[\max_i(y_i) - b, b - \min_i(y_i)\right]}{2147483645} \tag{E.12}$$

$$n_i := \mathrm{floor}\left[\frac{y_i - b}{a} + 0.5\right] \tag{E.13}$$

4PAC の読み書きは、ユーザー配列の型が R4 または R8 でなければなりません。

E.6 N1I2

10 倍した値を符号付き 2 バイト整数に変換してファイルに格納する。
ユーザー配列の型は R4, I2, I4 に限られる。
この PACK 型の場合は、ユーザー配列の型によって以下のように挙動が異なるので注意が必要である。

E.6.1 ユーザー配列型が R4 の場合

エンコードの際には、ユーザー配列の値を 10 倍した値を符号付き 2 バイト整数にキャストした値をファイルに格納する。デコードの際には、符号付き 2 バイト整数 を 10 で除した値を浮動小数点数にキャストしてユーザー配列に格納する。
すなわち、ユーザー配列の値は実際の値でよい。

E.6.2 ユーザー配列型が I2, I4 の場合

エンコードの際には、ユーザー配列そのままの値を符号付き 2 バイト整数にキャストした値をファイルに格納する。デコードの際には、符号付き 2 バイト整数を符号付き 2 バイト整数、または符号付き 4 バイト整数にキャストしてユーザー配列に格納する。
ユーザー側で、エンコードの際には 10 倍を、デコードの際には 0.1 倍をする必要があるので注意。

E.7 RLEN

非負のデータを圧縮するためのランレングス符号化法である。1 次元に連続したデータがある場合、その値と同じ値のデータの継続する長さ (ランレングス) を 1 つのセットとし、セットをつなげることで 1 次元に連続したデータを表現する手法である。
圧縮データの中で一つの格子点値が占めるビット数を変えることで、様々な範囲の (非負の) 整数データを圧縮できるが、NuSDaS では 8 ビットのみをサポートしている。

ユーザー配列の型は I1, I2, I4, R4, R8 が使えるが、エンコードできるデータは符号なし 1 バイト整数で表現できる範囲である(注 1)。

ランレングスのデータは、ビット数 NBIT、データの最大値 MAXV、データ列から構成される。

データ列を符号なしで NBIT ずつ区切って読んだとき、MAXV 以下の値は格子点の値とする。MAXV を超える値は、ランレングスを表すものとする。1 セットは、まず値を配置し、もしその値が連続するようであれば後ろにランレングスを付加することによって作られる。MAXV より大きなデータが続く場合はすべてそのセットのランレングスの情報であり、MAXV 以下のデータが現れた時点でそのセットは終了し、この MAXV 以下のデータは次のセットの値になる。また、同じ値が連続しない場合はランレングスは付加されず、次のセットに移る。

ランレングスは $L = 2^{\mathtt{NBIT}} - 1 - \mathtt{MAXV}$ としたとき、L 進数によって表現している。MAXV を超える値が N 個続いた場合、その値を $a_n (n = 1, 2, \cdots, N)$(インデックスは出現順) とするとランレングス R は

$$R = \sum_{n=1}^{N} \left[L^{(n-1)} \{ a_n - (\mathtt{MAXV} + 1) \} \right] + 1 \tag{E.14}$$

と求めることができる。

E.8 I1

ユーザー配列を符号なし 1 バイト整数型にキャストして格納する。

E.9 I2

ユーザー配列を符号つき 2 バイト整数型にキャストして格納する。

E.10 I4

ユーザー配列を符号つき 4 バイト整数型にキャストして格納する。

E.11 R4

ユーザー配列を単精度浮動小数点型にキャストして格納する。

E.12 R8

ユーザー配列を倍精度浮動小数点型にキャストして格納する。

E.13 2UPJ

2UPC と同様にデータを 16 ビット符号なし整数で表現した後、整数列を JPEG2000 圧縮したものをファイルに格納する。レポジトリでは 2009-11-04 に追加された機能。

2UPJ の読み書きは、ユーザー配列の型が I4, R4, R8 に限られる (N_NC によるレコード内容直接読み取りは 2010-04-21 に追加された)。

(注 1) 厳密には、符号なし 1 バイト整数の最大値である 255 を含むデータは圧縮できない。データの最大値 MAXV を超えたデータは、値 – MAXV をその直前の値のランレングスとするが、MAXV が 255 ではランレングスを表現できないからである。254 ではランレングスとして 1 だけを表現できるが、ランレングスが 1 の場合は圧縮の意味がない。

E.14 2UPP

2UPC と同様にデータを 16 ビット符号なし整数で表現した後、整数列を複合差分圧縮 (GRIB2 DRT5.3 に類似) したものをファイルに格納する。NuSDaS の複合差分圧縮では 32 格子単位を一つの群として扱う。格子数 N のときの群の数は $N_G = (N-1)/32 + 1$[注 2]で得られる。ただし最後の群の格子数は 32 未満になる可能性がある。

2UPP は 2UPC で得られたバイト列 n_k を、群 i の j 番目のデータとして $U_{ij} = n_{32i+j}$ と割り当てた後、以下の処理を行うことで求められる

$$P_{ij} = \begin{cases} \mathrm{mod}\,(U_{ij} + 32768,\, 65536) & (j = 0, 1)\,, \\ \mathrm{mod}\,(U_{ij} - 2U_{i,j-1} + U_{i,j-2} + 32768,\, 65536) & (j \geq 2)\,, \end{cases} \tag{E.15}$$

$$R_i = \min_j (P_{ij})\,, \tag{E.16}$$

$$D_{ij} = P_{ij} - R_i\,, \tag{E.17}$$

$$W_i = \mathrm{bits}\left(\max_j (D_{ij})\right) - 1\,. \tag{E.18}$$

$\mathrm{bits}(m)$ は整数 m の bit 幅を求める関数で例えば $\mathrm{bits}(5) = 3$ となる。ただし $m = 0$ のときは $\mathrm{bits}(0) \equiv 1$ として扱う。R_i, D_{ij} の取りうる値は 0 から 65535、W_i は 0 から 15 の値をとりうる。このようにして求めた R_i, W_i をそれぞれ 2 バイト幅、4bit 幅で A.22 に格納する。複合差分圧縮されたデータ本体 D_{ij} は A.22 の圧縮データの位置に $(W_i + 1)$bit 幅で格納する

2UPP の読み書きは、ユーザー配列の型が I4, R4, R8 に限られる。

[注 2] $N = 0$ のとき $N_G = 1$ となるので注意

F 仕様の変更点

F.1 pnusdas

リトルエンディアン機で動作するようになりました。

F.2 NuSDaS 1.1

F.2.1 ファイルにおける「レコード長」の変更

ファイルの各レコードの先頭、末尾各 4 バイトに書き込まれているレコード長をそのレコード長そのものの 8 バイトを除いた長さにするようにしました (NuSDaS10 では、その 8 バイトを含めた長さになっていました)。これによって、ファイルが Fortran 順編成ファイルになりました。

F.2.2 ファイル長の上限が 2GB から 4GB へ

従来は、`INDX` レコードを符号付き 4byte 整数としていたため、ファイルの大きさが符号付き 4byte 整数の表現範囲の上限である約 2GB に制限されていましたが、これを符号なし 4byte 整数として解釈するように変更して、約 4GB までに制限が緩和されました。

F.2.3 同一 TYPE の NRD が複数ある時の NRD 探索

同一の TYPE1,2,3 の NuSDaS Root Directory(NRD) が複数ある場合、従来は最初に見つかった NRD からデータ取得ができなければ、失敗としてエラーを返していましたが、データ取得が成功するまで NRD の探索を継続するようにしました。

F.2.4 ランレングス圧縮において 1byte 整数のユーザーデータをサポート

従来はランレングス圧縮のユーザーデータの型は 4byte 整数だけをサポートしていましたが、1byte 整数もサポートするようにしました。

F.2.5 高速化

ファイル書き込みの際にバッファリングを行うことにより、ファイル書き込みの高速化を図りました。

F.3 NuSDaS 1.2

SUBC RGAU, SUBC ZHYB, SUBC DELT 記録のアクセス関数が追加されました。

F.4 NuSDaS 1.3

F.4.1 Fortran での定数 NULL の廃止

あまりに実装が難しいため、Fortran インターフェイスでの定数 **NULL** が廃止されました。
nusdas_parameter_change() で設定したパラメタを既定値に戻すには、明示的に既定値を設定するか、新設の nusdas_parameter_reset() を用いてください。

F.4.2 ファイル名生成

定義ファイルの filename 文にスラッシュと_basetime などの置換指定を使うと互換性がなくなります。ディレクトリは path 文に記述するようにしてください。

F.4.3 CNTL 記録の大きさ

NuSDaS 1.3 で作られる CNTL 記録に書かれる対象時刻のリストは、定義ファイルの予報時間リスト (VALIDTIME1 文) と基準時刻から導かれる対象時刻すべてとなります。

従来は定義ファイルの設定により、対象時刻のリストが 1 対象時刻だけとなる場合がありました。

F.4.4 コーデック

nusdas_read や nusdas_write などで与えるユーザ配列の型とパッキング方式の組合せはなんでもよいわけではないのですが、許される組合せが増えました。

特に、RLEN パッキングで不適切なユーザ配列型 (**N_I2**, **N_R4**, **N_R8**) を与えた場合に不定動作となるバグに対処しました。

一方、**N_NC** 機能は 2012 年 9 月 19 日時点では 2UPC パッキングおよび 2UPJ パッキングだけについて実装されています。(NuSDaS 1.4 からは 2UPP も追加)

F.4.5 SUBC 記録

SUBC 記録を書き込んでいない状況で nusdas_subc_tdif(), nusdas_subc_srf(), nusdas_subc_delt() (NuSDaS 1.2 のみ) で読出しを行った場合、従来は異常が通知されずすべてゼロの値が返されますが、エラーコード −2 で異常終了するようにしました。

データファイルが存在しない状況で SUBC 記録を書き出すと従来はエラーコード −51 で異常終了していましたが、正常に書き出せるようになりました。

性能上の理由から NUSD–SUBC, INFO, END 記録の書き出しはファイルを閉じるときまで遅延されます。そこで SUBC 記録を出力したプログラムがファイルを閉じることなく異常終了すると当該 SUBC 記録の内容が読出せなくなります (従来版では読出せることがある)。

F.4.6 INFO 記録

リトルエンディアン機で定義ファイルによって INFO 記録を作成すると群名が反転 (例: VSRF → FRSV) するバグが対処されています。

INFO 記録を読み出すときバッファ長が不足していると従来はバッファ長だけ読出して正常終了していましたがエラーコード −3 で異常終了するようにしました。

データファイルが存在しない状況で INFO 記録を書き出すと従来はエラーコード −51 で異常終了していましたが、正常に書き出せるようになりました。

F.5 NuSDaS 1.4

NAPS9 運用開始後 2017 年 11 月までの主な変更は次のとおりです。チケット番号 (#717) 、リビジョン番号 (r4369) などは開発管理サーバ Redmine の番号です。

#717, r4369 入出力モジュール aio, mmap, ibmshmat が廃止されました。

r4376 実行時オプション GSVB が新設されました。

#720, r4380 ES 対応が入りました。

#745, r4386 OpenMP に対応しました。

#771, r4420　複合差分圧縮 packing = 2UPP を導入しました。

#848　複合差分圧縮で OpenMP に対応しました。

#1037　2UPP 展開サブルーチンを追加しました

G 範囲指定型 API (廃止予定)

G.1 概要

NuSDaS 内部では、対象時刻および面は点ではなく範囲として表現できるようになっています (GRIB から変換ができるように配慮したのでしょう)。対象時刻は整数 2 つで期間の始点と終点を表わし、面は 6 文字の名前 2 つで上端と下端の高度を表わすことができます。これらはそれぞれ「対象時刻 1」「対象時刻 2」「面 1」「面 2」と呼ばれます。

対象時刻 2 や面 2 が指定されない「範囲でないデータ」は、対象時刻 2 が値 1[(注 1)]で面 2 と面 1 が同じものとして表現されています。

対象時刻または面名に範囲を用いるデータセットを読み書きするために、本章で説明する API が用意されています。これらは関数名の末尾に '2' がつくので区別されます。しかし、実際には積算・平均の時間範囲は範囲指定ではなく、SUBC TDIF レコード [Table A.8 (p. 143)] によって表現されます。

対象時刻 2 や面 2 を用いた範囲指定は Pandora では利用できなくなりました。本章で説明される関数・サブルーチンも、将来のライブラリ再設計時には削除される予定ですので、開発環境のアプリケーションに用例があった場合は速やかに '2 なし' API に移行するように努めてください。

G.2 Fortran API

G.2.1 NUSDAS_CUT2: 領域限定のデータ読取

書式

CALL **NUSDAS_CUT2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *udata*, *utype*, *usize*, *ixstart*, *ixfinal*, *iystart*, *iyfinal*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1
validtime2	INTEGER(4)		IN	対象時刻 2
plane1	CHARACTER(6)		IN	面 1
plane2	CHARACTER(6)		IN	面 2
element	CHARACTER(6)		IN	要素名
udata	任意	可変	OUT	データ格納配列
utype	CHARACTER(2)		IN	データ格納配列の型
usize	INTEGER(4)		IN	データ格納配列の要素数
ixstart	INTEGER(4)		IN	x 方向格子番号下限
ixfinal	INTEGER(4)		IN	x 方向格子番号上限
iystart	INTEGER(4)		IN	y 方向格子番号下限
iyfinal	INTEGER(4)		IN	y 方向格子番号上限
result	INTEGER(4)		OUT	終了コード

G.2.2 NUSDAS_CUT2_RAW: 領域限定の DATA 記録直接読取

書式

CALL **NUSDAS_CUT2_RAW**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *udata*, *usize*, *ixstart*, *ixfinal*, *iystart*, *iyfinal*, *result*)

[(注 1)]通算分 1 は欠損をあらわします。1801 年 1 月 1 日 00:01Z と区別つきませんが、おそらく問題にはならないでしょう

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
plane1	CHARACTER(6)		IN	面 1
plane2	CHARACTER(6)		IN	面 2
element	CHARACTER(6)		IN	要素名
udata	任意	可変	OUT	データ格納先配列
usize	INTEGER(4)		IN	データ格納先配列のバイト数
ixstart	INTEGER(4)		IN	x 方向格子番号下限
ixfinal	INTEGER(4)		IN	x 方向格子番号上限
iystart	INTEGER(4)		IN	y 方向格子番号下限
iyfinal	INTEGER(4)		IN	y 方向格子番号上限
result	INTEGER(4)		OUT	終了コード

説明　nusdas_cut_raw の範囲指定版。nusdas_cut を参照。

G.2.3　NUSDAS_GRID2: 格子情報へのアクセス

書式

CALL **NUSDAS_GRID2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *proj*, *gridsize*, *gridinfo*, *value*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
proj	CHARACTER(4)		I/O	投影法 3 字略号
gridsize	INTEGER(4)	2	I/O	格子数
gridinfo	REAL(4)	14	I/O	投影法緒元
value	CHARACTER(4)		I/O	格子点値が周囲の場を代表する方法
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.4　NUSDAS_INFO2: INFO 記録へのアクセス

書式

CALL **NUSDAS_INFO2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *group*, *info*, *bytesize*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
group	CHARACTER(4)		IN	群名
info	CHARACTER	可変	I/O	INFO 記録内容
bytesize	INTEGER(4)		IN	INFO 記録のバイト数
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.5 NUSDAS_INQ_CNTL2: データファイルの諸元問合せ

書式

CALL **NUSDAS_INQ_CNTL2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *param*, *data*, *datasize*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
param	INTEGER(4)		IN	問合せ項目コード
data	任意	可変	OUT	問合せ結果配列
datasize	INTEGER(4)		IN	問合せ結果配列の要素数
result	INTEGER(4)		OUT	終了コード

G.2.6 NUSDAS_INQ_DATA2: データ記録の諸元問合せ

書式

CALL **NUSDAS_INQ_DATA2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *item*, *data*, *nelems*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
plane1	CHARACTER(6)		IN	面 1
plane2	CHARACTER(6)		IN	面 2
element	CHARACTER(6)		IN	要素名
item	INTEGER(4)		IN	問合せ項目コード
data	任意	可変	OUT	結果格納配列
nelems	INTEGER(4)		IN	結果格納配列要素数
result	INTEGER(4)		OUT	終了コード

G.2.7 NUSDAS_ONEFILE_CLOSE2: ひとつのファイルを閉じる

書式

CALL **NUSDAS_ONEFILE_CLOSE2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
result	INTEGER(4)		OUT	終了コード

G.2.8 NUSDAS_READ2: データ記録の読取

書式

CALL **NUSDAS_READ2**(*utype1*, *utype2*, *utype3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *data*, *fmt*, *size*, *result*)

引数名	引数の型	配列長	I/O	役割
utype1	CHARACTER(8)		IN	種別 1
utype2	CHARACTER(4)		IN	種別 2
utype3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
plane1	CHARACTER(6)		IN	面の名前 1
plane2	CHARACTER(6)		IN	面の名前 2
element	CHARACTER(6)		IN	要素名
data	任意	可変	OUT	結果格納配列
fmt	CHARACTER(2)		IN	結果格納配列の型
size	INTEGER(4)		IN	結果格納配列の要素数
result	INTEGER(4)		OUT	終了コード

G.2.9 NUSDAS_SUBC_DELT2: SUBC DELT へのアクセス

書式

CALL **NUSDAS_SUBC_DELT2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *delt*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
delt	REAL(4)		I/O	DELT 数値へのポインタ
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.10 NUSDAS_SUBC_ETA2: SUBC ETA へのアクセス

書式

CALL **NUSDAS_SUBC_ETA2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *n_levels*, *a*, *b*, *c*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
n_levels	INTEGER(4)		I/O	鉛直層数
a	REAL(4)	可変	I/O	係数 a
b	REAL(4)	可変	I/O	係数 b
c	REAL(4)		I/O	係数 c
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.11 NUSDAS_SUBC_ETA_INQ_NZ2: SUBC 記録の鉛直層数問合せ

書式

CALL **NUSDAS_SUBC_ETA_INQ_NZ2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *group*, *n_levels*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
group	CHARACTER(4)		IN	群名
n_levels	INTEGER(4)		OUT	鉛直層数
result	INTEGER(4)		OUT	終了コード

G.2.12 NUSDAS_SUBC_RGAU2: SUBC RGAU へのアクセス

書式

CALL **NUSDAS_SUBC_RGAU2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *j*, *j_start*, *j_n*, *i*, *i_start*, *i_n*, *lat*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
j	INTEGER(4)		I/O	全球の南北分割数
j_start	INTEGER(4)		I/O	データの最北格子の番号 (1 始まり)
j_n	INTEGER(4)		I/O	データの南北格子数
i	INTEGER(4)	可変	I/O	全球の東西格子数
i_start	INTEGER(4)	可変	I/O	データの最西格子の番号 (1 始まり)
i_n	INTEGER(4)	可変	I/O	データの東西格子数
lat	REAL(4)	可変	I/O	緯度
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.13 NUSDAS_SUBC_RGAU_INQ_JN2: SUBC RGAU 記録の大きさを問合せ

書式

CALL **NUSDAS_SUBC_RGAU_INQ_JN2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *j_n*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
j_n	INTEGER(4)		OUT	南北格子数
result	INTEGER(4)		OUT	終了コード

G.2.14 NUSDAS_SUBC_SIGM2: SUBC SIGM へのアクセス

書式

CALL **NUSDAS_SUBC_SIGM2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *n_levels*, *a*, *b*, *c*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
n_levels	INTEGER(4)		I/O	鉛直層数
a	REAL(4)	可変	I/O	係数 a
b	REAL(4)	可変	I/O	係数 b
c	REAL(4)		I/O	係数 c
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.15 NUSDAS_SUBC_SRF2: 降短系 SUBC へのアクセス

書式

CALL **NUSDAS_SUBC_SRF2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *group*, *data*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
plane1	CHARACTER(6)		IN	面 1
plane2	CHARACTER(6)		IN	面 2
element	CHARACTER(6)		IN	要素名
group	CHARACTER(4)		IN	群名
data	INTEGER(4)		I/O	データ配列
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.16 NUSDAS_SUBC_TDIF2: SUBC TDIF へのアクセス

書式

CALL **NUSDAS_SUBC_TDIF2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *diff_time*, *total_sec*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
diff_time	INTEGER(4)		I/O	対象時刻からのずれ (秒)
total_sec	INTEGER(4)		I/O	総予報時間 (秒)
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.17 NUSDAS_SUBC_ZHYB2: SUBC ZHYB へのアクセス

書式

CALL **NUSDAS_SUBC_ZHYB2**(*type1*, *type2*, *type3*, *basetime*, *member*, *validtime1*, *validtime2*, *nz*, *ptrf*, *presrf*, *zrp*, *zrw*, *vctrans_p*, *vctrans_w*, *dvtrans_p*, *dvtrans_w*, *getput*, *result*)

引数名	引数の型	配列長	I/O	役割
type1	CHARACTER(8)		IN	種別 1
type2	CHARACTER(4)		IN	種別 2
type3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
nz	INTEGER(4)		I/O	鉛直層数
ptrf	REAL(4)		I/O	温位の参照値
presrf	REAL(4)		I/O	気圧の参照値
zrp	REAL(4)	可変	I/O	モデル面高度 (フルレベル)
zrw	REAL(4)	可変	I/O	モデル面高度 (ハーフレベル)
vctrans_p	REAL(4)	可変	I/O	座標変換関数 (フルレベル)
vctrans_w	REAL(4)	可変	I/O	座標変換関数 (ハーフレベル)
dvtrans_p	REAL(4)	可変	I/O	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	REAL(4)	可変	I/O	座標変換関数の鉛直微分 (ハーフレベル)
getput	CHARACTER(3)		IN	入出力指示 (*"GET"* または *"PUT"*)
result	INTEGER(4)		OUT	終了コード

G.2.18 NUSDAS_WRITE2: データ記録の書出

書式

CALL **NUSDAS_WRITE2**(*utype1*, *utype2*, *utype3*, *basetime*, *member*, *validtime1*, *validtime2*, *plane1*, *plane2*, *element*, *data*, *fmt*, *nelems*, *result*)

引数名	引数の型	配列長	I/O	役割
utype1	CHARACTER(8)		IN	種別 1
utype2	CHARACTER(4)		IN	種別 2
utype3	CHARACTER(4)		IN	種別 3
basetime	INTEGER(4)		IN	基準時刻 (通算分)
member	CHARACTER(4)		IN	メンバー名
validtime1	INTEGER(4)		IN	対象時刻 1(通算分)
validtime2	INTEGER(4)		IN	対象時刻 2(通算分)
plane1	CHARACTER(6)		IN	面の名前 1
plane2	CHARACTER(6)		IN	面の名前 2
element	CHARACTER(6)		IN	要素名
data	任意	可変	IN	データを与える配列
fmt	CHARACTER(2)		IN	data の型
nelems	INTEGER(4)		IN	data の要素数
result	INTEGER(4)		OUT	終了コード

G.3 C API

G.3.1 nusdas_cut2: 領域限定のデータ読取

書式

N_SI4 **nusdas_cut2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], void ∗*udata*, const char *utype*[2], const N_SI4 ∗*usize*, const N_SI4 ∗*ixstart*, const N_SI4 ∗*ixfinal*, const N_SI4 ∗*iystart*, const N_SI4 ∗*iyfinal*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1
validtime2	const N_SI4 *	対象時刻 2
plane1	const char [6]	面 1
plane2	const char [6]	面 2
element	const char [6]	要素名
udata	void *	データ格納配列
utype	const char [2]	データ格納配列の型
usize	const N_SI4 *	データ格納配列の要素数
ixstart	const N_SI4 *	x 方向格子番号下限
ixfinal	const N_SI4 *	x 方向格子番号上限
iystart	const N_SI4 *	y 方向格子番号下限
iyfinal	const N_SI4 *	y 方向格子番号上限

G.3.2 nusdas_cut2_raw: 領域限定の DATA 記録直接読取

書式

N_SI4 **nusdas_cut2_raw**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], void ∗*udata*, const N_SI4 ∗*usize*, const N_SI4 ∗*ixstart*, const N_SI4 ∗*ixfinal*, const N_SI4 ∗*iystart*, const N_SI4 ∗*iyfinal*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
plane1	const char [6]	面 1
plane2	const char [6]	面 2
element	const char [6]	要素名
udata	void *	データ格納先配列
usize	const N_SI4 *	データ格納先配列のバイト数
ixstart	const N_SI4 *	x 方向格子番号下限
ixfinal	const N_SI4 *	x 方向格子番号上限
iystart	const N_SI4 *	y 方向格子番号下限
iyfinal	const N_SI4 *	y 方向格子番号上限

説明　nusdas_cut_raw の範囲指定版。nusdas_cut を参照。

G.3.3 nusdas_grid2: 格子情報へのアクセス

書式

N_SI4 **nusdas_grid2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, char *proj*[4], N_SI4 *gridsize*[2], float *gridinfo*[14], char *value*[4], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
proj	char [4]	投影法 3 字略号
gridsize	N_SI4 [2]	格子数
gridinfo	float [14]	投影法緒元
value	char [4]	格子点値が周囲の場を代表する方法
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.4 nusdas_info2: INFO 記録へのアクセス

書式

N_SI4 **nusdas_info2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, const char *group*[4], char *info*[ ], const N_SI4 *bytesize, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
group	const char [4]	群名
info	char [ ]	INFO 記録内容
bytesize	const N_SI4 *	INFO 記録のバイト数
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.5 nusdas_inq_cntl2: データファイルの諸元問合せ

書式

N_SI4 **nusdas_inq_cntl2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, N_SI4 *param*, void *data, const N_SI4 *datasize);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
param	N_SI4	問合せ項目コード
data	void *	問合せ結果配列
datasize	const N_SI4 *	問合せ結果配列の要素数

G.3.6 nusdas_inq_data2: データ記録の諸元問合せ

書式

N_SI4 **nusdas_inq_data2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], N_SI4 *item*, void ∗*data*, const N_SI4 ∗*nelems*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)
plane1	const char [6]	面 1
plane2	const char [6]	面 2
element	const char [6]	要素名
item	N_SI4	問合せ項目コード
data	void ∗	結果格納配列
nelems	const N_SI4 ∗	結果格納配列要素数

G.3.7 nusdas_onefile_close2: ひとつのファイルを閉じる

書式

N_SI4 **nusdas_onefile_close2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)

G.3.8 nusdas_read2: データ記録の読取

書式

N_SI4 **nusdas_read2**(const char *utype1*[8], const char *utype2*[4], const char *utype3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], void ∗*data*, const char *fmt*[2], const N_SI4 ∗*size*);

引数名	引数の型	役割
utype1	const char [8]	種別 1
utype2	const char [4]	種別 2
utype3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
plane1	const char [6]	面の名前 1
plane2	const char [6]	面の名前 2
element	const char [6]	要素名
data	void *	結果格納配列
fmt	const char [2]	結果格納配列の型
size	const N_SI4 *	結果格納配列の要素数

G.3.9 nusdas_subc_delt2: SUBC DELT へのアクセス

書式

N_SI4 **nusdas_subc_delt2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, float *delt, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
delt	float *	DELT 数値へのポインタ
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.10 nusdas_subc_eta2: SUBC ETA へのアクセス

書式

N_SI4 **nusdas_subc_eta2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, N_SI4 *n_levels, float *a*[ ], float *b*[ ], float *c, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
n_levels	N_SI4 *	鉛直層数
a	float [ ]	係数 a
b	float [ ]	係数 b
c	float *	係数 c
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.11 nusdas_subc_eta_inq_nz2: SUBC 記録の鉛直層数問合せ

書式

N_SI4 **nusdas_subc_eta_inq_nz2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *group*[4], N_SI4 ∗*n_levels*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)
group	const char [4]	群名
n_levels	N_SI4 ∗	鉛直層数

G.3.12 nusdas_subc_rgau2: SUBC RGAU へのアクセス

書式

N_SI4 **nusdas_subc_rgau2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, N_SI4 ∗*j*, N_SI4 ∗*j_start*, N_SI4 ∗*j_n*, N_SI4 *i*[ ], N_SI4 *i_start*[ ], N_SI4 *i_n*[ ], float *lat*[ ], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)
j	N_SI4 ∗	全球の南北分割数
j_start	N_SI4 ∗	データの最北格子の番号 (1 始まり)
j_n	N_SI4 ∗	データの南北格子数
i	N_SI4 [ ]	全球の東西格子数
i_start	N_SI4 [ ]	データの最西格子の番号 (1 始まり)
i_n	N_SI4 [ ]	データの東西格子数
lat	float [ ]	緯度
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.13 nusdas_subc_rgau_inq_jn2: SUBC RGAU 記録の大きさを問合せ

書式

N_SI4 **nusdas_subc_rgau_inq_jn2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, N_SI4 ∗*j_n*);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
j_n	N_SI4 *	南北格子数

G.3.14 nusdas_subc_sigm2: SUBC SIGM へのアクセス

書式

N_SI4 **nusdas_subc_sigm2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, N_SI4 *n_levels, float *a*[ ], float *b*[ ], float *c, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
n_levels	N_SI4 *	鉛直層数
a	float [ ]	係数 a
b	float [ ]	係数 b
c	float *	係数 c
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.15 nusdas_subc_srf2: 降短系 SUBC へのアクセス

書式

N_SI4 **nusdas_subc_srf2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 *basetime, const char *member*[4], const N_SI4 *validtime1, const N_SI4 *validtime2, const char *plane1*[6], const char *plane2*[6], const char *element*[6], const char *group*[4], N_SI4 *data, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
plane1	const char [6]	面 1
plane2	const char [6]	面 2
element	const char [6]	要素名
group	const char [4]	群名
data	N_SI4 *	データ配列
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.16 nusdas_subc_srf_ship2: SUBC LOCA へのアクセス

書式

N_SI4 **nusdas_subc_srf_ship2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, N_SI4 ∗*lat*, N_SI4 ∗*lon*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)
lat	N_SI4 ∗	緯度
lon	N_SI4 ∗	経度
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.17 nusdas_subc_tdif2: SUBC TDIF へのアクセス

書式

N_SI4 **nusdas_subc_tdif2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, N_SI4 ∗*diff_time*, N_SI4 ∗*total_sec*, const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 ∗	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 ∗	対象時刻 1(通算分)
validtime2	const N_SI4 ∗	対象時刻 2(通算分)
diff_time	N_SI4 ∗	対象時刻からのずれ (秒)
total_sec	N_SI4 ∗	総予報時間 (秒)
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.18 nusdas_subc_zhyb2: SUBC ZHYB へのアクセス

書式

N_SI4 **nusdas_subc_zhyb2**(const char *type1*[8], const char *type2*[4], const char *type3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, N_SI4 ∗*nz*, float ∗*ptrf*, float ∗*presrf*, float *zrp*[ ], float *zrw*[ ], float *vctrans_p*[ ], float *vctrans_w*[ ], float *dvtrans_p*[ ], float *dvtrans_w*[ ], const char *getput*[3]);

引数名	引数の型	役割
type1	const char [8]	種別 1
type2	const char [4]	種別 2
type3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
nz	N_SI4 *	鉛直層数
ptrf	float *	温位の参照値
presrf	float *	気圧の参照値
zrp	float []	モデル面高度 (フルレベル)
zrw	float []	モデル面高度 (ハーフレベル)
vctrans_p	float []	座標変換関数 (フルレベル)
vctrans_w	float []	座標変換関数 (ハーフレベル)
dvtrans_p	float []	座標変換関数の鉛直微分 (フルレベル)
dvtrans_w	float []	座標変換関数の鉛直微分 (ハーフレベル)
getput	const char [3]	入出力指示 (*"GET"* または *"PUT"*)

G.3.19 nusdas_write2: データ記録の書出

書式

N_SI4 **nusdas_write2**(const char *utype1*[8], const char *utype2*[4], const char *utype3*[4], const N_SI4 ∗*basetime*, const char *member*[4], const N_SI4 ∗*validtime1*, const N_SI4 ∗*validtime2*, const char *plane1*[6], const char *plane2*[6], const char *element*[6], const void ∗*data*, const char *fmt*[2], const N_SI4 ∗*nelems*);

引数名	引数の型	役割
utype1	const char [8]	種別 1
utype2	const char [4]	種別 2
utype3	const char [4]	種別 3
basetime	const N_SI4 *	基準時刻 (通算分)
member	const char [4]	メンバー名
validtime1	const N_SI4 *	対象時刻 1(通算分)
validtime2	const N_SI4 *	対象時刻 2(通算分)
plane1	const char [6]	面の名前 1
plane2	const char [6]	面の名前 2
element	const char [6]	要素名
data	const void *	データを与える配列
fmt	const char [2]	data の型
nelems	const N_SI4 *	data の要素数

H pandora

H.1 pandora の概要

Pandora とは気象庁内部で数値予報システムを中心に使われている、RESTful な HTTP データ転送規約です。

H.2 pandora における NuSDaS データの取り扱い

H.2.1 URL 規則

NuSDaS データに対しては以下のようにして指定するものとする。

```
dataname ≡ '/NUSDAS/' nrd '/' type '/' basetime '/' member '/' validtime '/' validtime2 '/'
  plane '/' plane2 '/' element [['/' ys, ye ]'/' xs, xe]
type ≡ type1 '.' type2 '.' type3
basetime, validtime, validtime2 ≡ time
time ≡ yyyy '-' mm '-' dd 't' hhMM
type1 ≡ n[n[n[n[n[n[n]]]]]]
type2, type3, member, plane, plane2, element ≡ n[n[n[n]]]
y, m, d, h, M  ≡ digit
xs, xe, ys, ye ≡ digit または *
digit ≡ <数字>
n ≡ <名前文字すなわち英数字、下線 (_)、単価記号 (@)、ハイフンマイナス (-)、加算記号 (+)>
```

なお、振り分け名を NUSDAS の代わりに、NAPS8_NUSDAS にすることで、validtime2、plane2 を省略できる。すなわち、

```
dataname ≡ '/NAPS8_NUSDAS/' nrd '/' type '/' basetime '/' member '/'
validtime '/' plane '/' element [['/' ys, ye ]'/' xs, xe]
```

- nrd は pandora オリジンサーバーにおける NuSDaS Root Directory
- xs, xe, ys, ye はそれぞれ領域指定における x 方向の始点、x 方向の終点、y 方向の始点、y 方向の終点を表す。なお、始点の格子点番号は meta データの first index x, first index y によって決められる (NAPS の場合は 1)。これらの領域指定を省略した場合や “*” は全領域を表す。

  例)

  - 1,10/11,20 : X 方向は 11〜20, Y 方向は 1〜10 を切り出す
  - 1,10: X 方向は全部、Y 方向は 1〜10
  - */11,20 : X 方向は 11〜20, Y 方向は全部
  - */* : X も Y も全部 (指定しないのと同じ)
- 要素名に 'LAT' または 'LON' を指定すると、各格子点の緯度、および経度を投影情報を元に pandora サーバーで計算して返す。

H.2.2 driver の仕様

pandora の規約が要求するように index, data, meta, schema 資源に要求に対するドライバーを整備している。また、meta については、さまざまなメタ情報に対応した資源がある。

index 資源

data 資源

nusdas_read によって取得した GPV 値。nusview/nusdump によって出力されている。

meta 資源

H.2.3 nusview ツールの仕様

共通事項

各ツールの出力には、HTTP のヘッダが付加される。最低限付加されるのは

- Content-Type
- Content-Length

であり、出力をそのまま HTTP のレスポンスに利用することができる。その他、ツール独自のヘッダが付加されることがある。

改行が 2 つ連続で続くところが、ヘッダとボディの境界になる。

nusdump

`nusdas_read` によって取得した GPV 値をオプションによって指定された形式で出力する。http のためのヘッダがつく。

- オプション一覧

`-tf`	32 ビット単精度浮動小数点形式で出力
`-tp`	ASCII テキストで出力
`-tu`	8bit 符号なし整数形式で出力
`-ti`	32bit 符号付き整数形式で出力
`-tr`	8bit ランレングス圧縮形式で出力
`-td`	64 ビット倍精度浮動小数点形式で出力
`-pG` or `-PG`	Portable Graymap(テキスト) 形式で出力
`-pg` or `-Pg`	Portable Graymap(テキスト) 形式で出力
`-pp` or `-Pp`	Portable Pixmap 形式で出力
`-R`*ixst/ixen/jyst/jyen*	切り出し領域指定

`-t` と `-p` で始まるオプションは排他。複数指定された場合は最後に指定されたものが有効。

なにもオプションをつけない場合は `-tf` がついていて、領域は全領域であるとする。また、バイナリ形式のバイトオーダーはすべて big endian である。

- ヘッダ一覧

`X-missing-value`	欠損値 (NuSDaS の欠損値の取り扱いが MISS か MASK のときのみ)
`X-notice`	注意喚起情報 (変数型の変更)
`X-Nusdas-Return-Code`	NuSDaS API の戻り値
`X-Data-Num`	出力したデータ数 (格子点数)
`X-Data-Range`	出力したデータの領域 (X 方向始点、同終点、Y 方向始点、同終点の順にカンマ区切りで出力)
`X-value-max`, `X-value-min`	データの最大・最小値 (Portable Graymap/Pixmap の時のみ)
`X-gradation-step`	Portable Graymap/Pixmap のときの一つの色の幅の値

nusdump_rawgz

GPV 値を NuSDaS の DATA レコードに記録されている形式のまま出力する。`nusdas_read_raw` の出力に対応。DATA レコードの項番 10 の「格子配列の大きさ」以降のデータが出力される。また、zlib がインストールされている環境では、デフォルトで gzip 圧縮されたデータが出力される。

- オプション一覧
  `-r` gzip 圧縮をせずに出力する
- ヘッダ一覧
  `nusdump` と同じ。

nusmeta

`nusdas_grid` の出力に対応する情報を出力する。切り出し領域指定の情報が反映された値が返される (後述の nuscntl では切り出し領域指定は考慮されないので注意)。

- オプション一覧

`-t`	html で出力 (デフォルト)
`-tr`	rd(ruby document) 形式で出力
`-tt`	タブ区切りテキスト
`-b`	バイナリ形式で出力
`-l`	テキストの場合にタイトルを出力する
`-R`*ixst*/*ixen*/*jyst*/*jyen*	切り出し領域指定

html, rd, tsv(tab separated values) などのテキストファイルの場合には、各項目の値を示すのに次表の文字列をキーにしている。

キー名	値
`projection type`	投影法
`number of x grids`	x 方向の格子数
`number of y grids`	y 方向の格子数
`base point x`	基準点の X 座標
`base point y`	基準点の Y 座標
`base point lat`	基準点の緯度
`base point lon`	基準点の経度
`grid interval x`	X 方向の格子間隔
`grid interval y`	Y 方向の格子間隔
`standard lat 1`	標準緯度
`standard lon 1`	標準経度
`standard lat 2`	第 2 標準緯度
`standard lon 2`	第 2 標準経度
`latitude 1`	緯度 1
`longitude 1`	経度 1
`latitude 2`	緯度 2
`longitude 2`	経度 2
`representation`	格子点の空間代表性
`first index x`	X 方向の最初のインデックスの値
`first index y`	Y 方向の最初のインデックスの値

また、`-b` によるバイナリ出力のフォーマットは次表の通り。バイトオーダーは big endian である。

項目	変数型	データ長	配列数
投影法	char	4	1
格子配列の大きさ	int	4	2
基準点の座標	float	4	2
基準点の緯度経度	float	4	2
格子間隔	float	4	2
標準緯度経度	float	4	2
第 2 標準緯度経度	float	4	2
緯度経度 1	float	4	2
緯度経度 2	float	4	2
格子点の意味	char	4	1
最初のインデックス値	float	4	2

nuscntl

`nusdas_inq_cntl` の出力に対応する情報を出力する。領域切り出し指定に関係なく、データファイルの CNTL レコードの内容を返す。ELEMENTMAP, DATAMAP は 1 バイトバイナリ、その他は rd 形式の一覧のみ対応している。

- オプション一覧

`-tr` rd 形式で出力する (デフォルト)
`-m` ELEMENTMAP, DATAMAP を 1 バイトバイナリで出力する
`-l` タイトルを出力する

`member_list`, `validtime_list`, `plane_list`, `element_list` の場合、リストの各要素との間は空白で区切る。また、`validtime_list` 以外の文字列の場合は、シングルクォーテーション (') で各要素を囲む。

RD 出力の場合、各項目の値を示すのに次表の文字列をキーにしている。

キー名	値
`number of member`	メンバー数
`member list`	メンバーのリスト
`number of validtime`	validtime の数
`validtime list`	validtime のリスト
`number of plane`	面の数
`plane list`	面のリスト
`number of element`	要素の数
`element list`	要素のリスト
`projection type`	投影法
`number of x grids`	x 方向の格子数
`number of y grids`	y 方向の格子数
`base point x`	基準点の X 座標
`base point y`	基準点の Y 座標
`base point lat`	基準点の緯度
`base point lon`	基準点の経度
`grid interval x`	X 方向の格子間隔
`grid interval y`	Y 方向の格子間隔
`standard lat 1`	標準緯度
`standard lon 1`	標準経度
`standard lat 2`	第 2 標準緯度
`standard lon 2`	第 2 標準経度
`latitude 1`	緯度 1
`longitude 1`	経度 1
`latitude 2`	緯度 2
`longitude 2`	経度 2
`representation`	格子点の空間代表性

nussigm

`nusdas_subc_sigm` または `nusdas_subc_eta` の出力に対応する。バイナリ出力 (big endian) のみに対応している。

- オプション一覧
  `-s` `SIGM` レコードを読み出す (デフォルト)
  `-e` `ETA` レコードを読み出す
  `-b` バイナリ出力をする (デフォルト)

  バイナリ出力のフォーマットは次表の通り (`n_lv` は面の数)。

項目	データ型	データ長	配列数
$A(k)$	float	4	`n_lv+1`
$B(k)$	float	4	`n_lv+1`
$C(k)$	float	4	1

- ヘッダ一覧
  `X-PLANE-NUM` 面の数 (`n_lv`)

nusinqdef

`nusdas_inq_def` に対応する情報を出力する。ELEMENTMAP は 1 バイトバイナリ、その他は rd 形式の一覧のみ対応している。

- オプション一覧
  `-tr` rd 形式で出力する (デフォルト)
  `-m` ELEMENTMAP, DATAMAP を 1 バイトバイナリで出力する
  `-l` タイトルを出力する

nussubc_srf

nusinfo

nusinqnz

nusrgau

nusrgaujn

nuszhyb

H.3 pandora driver 共通ライブラリ: pandora_driver.rb

pandora driver を作成する場合には、必要なヘッダーを付けたりなにかと気にすべきことが多い。

`pandora_driver.rb` は pandora driver が必要とすることをライブラリ化し、簡単にドライバーを作成できるようにしたものである。

たとえば、`nusdas_read` の結果を単精度浮動小数点型で渡すドライバは以下のように書ける。

```
#!/usr/bin/env ruby

load("libs/pandora_driver.rb")

drv = NusdasDataDriver.new
drv.do_nusview("#{drv.private}/nusdump")
drv.send_response()
```

ここで、`nusdump` は H.2.3 で説明されているものであり、HTTP のレスポンスである条件を満たすように、ヘッダに Content-Type を出力する必要がある。この例では、`nusdump` は

```
Content-Type: application/x-float32-stream

(単精度浮動小数点数のバイナリ列)
```

という出力をするようになっている。

H.4 pandora client のための TCP/IP 通信ライブラリ: pandora_lib

pandora サーバーとの TCP/IP 通信を支援することを念頭に構成されたライブラリである。pandora サーバーとの通信は HTTP プロトコルで行われるが、このライブラリは対 pandora サーバーと意識しているものの、汎用の HTTP プロトコルによる通信のためのライブラリとしても利用できる。

C 言語で記述されており、C 言語での利用が前提になっている。

H.4.1 ソースファイルの構成

pandora_lib.c およびヘッダファイル pandora_lib.h から構成されている。

pandora_lib の関数を使用するソースファイルには、pandora_lib.h を include する。

H.4.2 関数リファレンス

`pandora_data* pdr_new()`

あたらしい pandora_data オブジェクトを作成して初期化する.

成功すれば，そのポインタを，失敗すれば NULL を返す.

`int pdr_delete(pandora_data *pdr)`

pandora_data オブジェクトを削除する.
引数:
* pdr: 処理対象の pandora_data オブジェクトのポインタ

返却値:
* 0: 正常
* -11 : pdr が allocate されていない

`int pdr_data_free(pandora_data *pdr)`

データ (http status，header，body) の領域を解放する.
pandora_data オブジェクト自体は削除されず，set_***関数で設定した属性
についてはそのまま残る.
引数:
* pdr: 処理対象の pandora_data オブジェクトのポインタ

返却値
* 0: 正常
* -11: pdr が allocate されていない

`int pdr_set_server(pandora_data *pdr, char *server)`

pdr オブジェクトに対して，リクエストを出すサーバー名 (port 番号も含む) を
設定する．指定の仕方は

```
server:port
```

であり，server で示される文字列は\0 で終了していなければならない.

ポート番号が指定されていない場合は，8080 に設定される.

この関数を呼び出した結果，サーバーが変更されるようであれば，接続中の
コネクションは切断される．(この関数を呼び出しても変更がなければ切断
されない)

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* server : 設定するサーバー名の文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_path(pandora_data *pdr. char *path)

資源を指定する path を指定する. path で示される文字列は\0 で
終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* path : 設定する資源の path の文字列の先頭ポインタ (\0 終端)

返却値:
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗

int pdr_set_root(pandora_data *pdr, char *root)

振り分け先 (資源を示すパスの前につく) を指定する.
root で示される文字列は\0 で終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* root : 設定する振り分け先の文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_resource_type(pandora_data *pdr, char *resource_type)

取得するデータの種別を指定する. たとえば, data.f32, meta.html など.
resource_type で示される文字列は\0 で終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* resource_type : 設定するデータ種別の文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_req_hdr_init(pandora_data *pdr)

レスポンスのヘッダを格納するテーブルを初期化する

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

返却値
* 0: 正常

int pdr_set_host_header(pandora_data *pdr, char *host_header)

Host ヘッダーの文字列を指定する.
host_header で示される文字列は\0 で終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* host_header : 設定する Host ヘッダーの文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_accept_header(pandora_data *pdr, char *accept_header)

Accept ヘッダーの文字列を指定する.
accept_header で示される文字列は\0 で終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* accept_header : 設定する Accept ヘッダーの文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_accept_encoding_header(pandora_data *pdr, char *header)

Accept-Encoding ヘッダーの文字列を指定する.
header で示される文字列は\0 で終了していなければならない.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* header : 設定する Accept-Encoding ヘッダーの文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常
* 1: 正常 (設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_proxy(pandora_data *pdr, char *proxy)

proxy サーバを使用する際に指定する.
指定の方法は pdr_set_server と同じである.

proxy サーバーの指定を解除するためには,

```
pdr_set_proxy(pdr, "");
または
pdr_set_proxt(pdr, NULL);
```

とする.

pdr_set_server と同様，この関数の呼び出しの結果，proxy サーバーが変更
された場合は接続中のコネクションは切断される.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* proxy : 設定する proxy サーバーの文字列の先頭ポインタ (\0 終端)

返却値
* 0: 正常 (proxy_server がセットされた)
* 1: 正常 (proxy_server の設定が解除された)
* -10: メモリーの確保に失敗
* -11: 対象の pdr が allocate されていない

int pdr_set_timeout(pandora_data *pdr, int timeout)

タイムアウトを指定する．このタイムアウトは read
に適用される.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* timeout : 設定する timeout の時間 (単位は秒)

返却値
* 0: 正常
* -11: 対象の pdr が allocate されていない

int pdr_set_connect_timeout(pandora_data *pdr, int timeout)

タイムアウトを指定する．このタイムアウトは connection
に適用される.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* timeout : 設定する timeout の時間 (単位は秒)

返却値
* 0: 正常
* -11: 対象の pdr が allocate されていない

int pdr_process(pandora_data *pdr)

Pandora Server にリクエストを出して，データの取得が行なわれる.

連続してこの関数を呼び出すと，その前に取得したデータを解放して
(pdr_data_free が呼び出される) からデータ取得をすることに注意.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

返却値
 * ステータスコードが，200 の場合には，取得したデータのサイズを返す.
 * それ以外の場合には，ステータスコードに-1 をかけた値を返す.
 * その他のエラーは-99 以上 0 未満の値を返す.

void* pdr_get_data(pandora_data *pdr)

取得したデータの先頭のポインターを返す.
データがない場合は NULL を返す.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

int pdr_get_data_len(pandora_data *pdr)

取得したデータのサイズを返す.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

int pdr_get_status_code(pandora_data *pdr)

HTTP ステータスコードを返す

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

int pdr_sock_close(pandora_data *pdr)

コネクションを強制的に切断する.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

返却値
* 0 : 正常終了
* <0 : エラー

char* pdr_header_find(pandora_data *pdr, char *header)

header で示されるヘッダーを検索して，その値の文字列 (\0 終端) の先頭ポインタを返す.
検索の際には大文字/小文字の区別はしない.

該当するものがない場合は NULL を返す.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* header : 探索する header の文字列の先頭ポインタ (\0 終端)

int pdr_print_all_headers(pandora_data *pdr, FILE *fp)

受けとったヘッダーをすべて，fp に出力する.

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ
* fp : 出力先 FILE ポインタ

返却値
* 0: 正常
* -99: I/O エラー

char* pdr_get_request_path(pandora_data *pdr)

リクエストとして出すパス文字列 (\0 終端) の先頭ポインタを返す

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

char* pdr_get_host(pandora_data *pdr)

pdr にセットされた host 名文字列 (\0 終端) の先頭ポインタを返す

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

```
int pdr_get_port(pandora_data *pdr)
```

pdr にセットされた port 番号を返す

引数
* pdr : 処理対象 pandora_data オブジェクトのポインタ

H.4.3 使用例

pandora server から NuSDaS データを取得するクライアントの例

pandora server(localhost:8080) から引数で指定した日時の全球速報解析
(NuSDaS TYPE 名: _GSMLLPP.EASV.STD1) の地表面データを取得するサンプルである。
使用例:

```
$ ./panlib_samlple1 2017/08/01/00:00
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pandora_lib.h"

#define TYPE1 "_GSMLLPP"
#define TYPE2 "EASV"
#define TYPE3 "STD1"
#define MEMBER "none"
#define LAYER "SURF"
#define SERVER "localhost:8080"
#define ROOT "/NAPS/db2/eternal/Ea/Anl/anl_p.nus"

int main(int argc, char* argv[])
{
    pandora_data *pdr;
    char elem[3][7]={"T","U","V"};
    int i;
    int code, len;
    FILE *fp;
    char filename[256];
    char nus_path[256];
    int year, month, day, hour, min;
    int data_num;
    char *data_num_str;

    if(argc <2){
        fprintf(stderr,"usage: %s yyyy/mm/dd/hh:mm\n",argv[0]);
        return -1;
    }
    if(sscanf(argv[1],"%d/%d/%d/%d:%d",&year, &month, &day, &hour, &min)!=5){
        fprintf(stderr,"Time Format Error\n");
        return -2;
    }

    pdr = pdr_new();
```

```
    /* オブジェクトを作成 */

    pdr_set_server(pdr,SERVER);
    /* アクセスするサーバー名をセット*/

    pdr_set_root(pdr, ROOT);
    /* 振り分け先をセット */

    pdr_set_resource_type(pdr,"data.txt");
    /* 資源の種類と形式をセット */

    for(i=0;i<3;i++){
        sprintf(nus_path,
            "%s.%s.%s/%04d-%02d-%02dt%02d%02d/%s/%04d-%02d-%02dt%02d%02d//%s//%s",
            TYPE1, TYPE2, TYPE3,
            year, month, day, hour, min, MEMBER,
            year, month, day, hour, min, LAYER, elem[i]
        );
        /* アクセスパスを作成する validtime2, plane2 の値は不要だがスラッシュは必要 */
        pdr_set_path(pdr, nus_path);
        if(pdr_process(pdr) < 0){
            /* リクエストの送信, レスポンスの受信*/
            code = pdr_get_status_code(pdr); /*エラーコードの取得*/
            fprintf(stderr, "%d: Request failed. code=%d\n",i, code);
        }
        len = pdr_get_data_len(pdr); /* レスポンスの長さの取得 */
        if((data_num_str = pdr_header_find(pdr,"X-Data-Num"))!=NULL){
            data_num = atoi(data_num_str);
        }
        else{
            data_num = 0;
        }
        fprintf(stderr,"%d: Data received: %d byte\n",i, len);
        fprintf(stderr,"Data-Num: %d\n", data_num);
        sprintf(filename, "data-%04d%02d%02d%02d%02d-%d.txt",
                year, month, day, hour, min, i);
        fp = fopen(filename,"w");
        fwrite(pdr_get_data(pdr),len,1,fp);
        /* pdr_get_data(pdr) から始まるメモリー上のデータを長さ len
           (つまり全部) 出力 */

        fclose(fp);

    }
    pdr_delete(pdr);

    return 0;
}
```

汎用 HTTP クライアントの例

引数に与えた URL の内容を HTTP によって取得する。

```
panlib_sample2 http://<server>:<port>/<path>
```

```
/* ------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>

#include "pandora_lib.h"
#define DEFAULT_TIMEOUT 60

/*------------------------------------------------------------------------*/
void usage(char *argv[]){
    fprintf(stderr,"usage: %s [option] http://<server>:<port>/<path>\n",argv[0]);
    fprintf(stderr, "        option: -h   print all headers to stderr\n");
}
/*------------------------------------------------------------------------*/
int main(int argc, char *argv[])
{
    char *server_path, *server, *path, *host_header;
    pandora_data *pdr;
    char *p;
    int code;
    int i, len;
    int print_header_flag;
    int timeout;
    int rt;

    if(argc < 2){
        usage(argv);
        return -1;
    }

    print_header_flag = 0;
    server_path = NULL;
    timeout = DEFAULT_TIMEOUT;
    for(i=1;argv[i]!=NULL;i++){
        if(argv[i][0]=='-'){
            switch(argv[i][1]){
            case 'h':
                print_header_flag = 1;
                break;
            case 't':
                timeout = atoi(argv[++i]);
                if(timeout == 0){
                    fprintf(stderr,"Invalid Timeout Parameter:%s\n",
                            argv[i]);
                }
                break;
            case 'j':
                host_header = argv[++i];
                break;
            default:
                break;
            }
            continue;
        }

        if(server_path == NULL){
            server_path = argv[i];
        }
    }
```

```
if(server_path == NULL){
    usage(argv);
    return -2;
}

/* server と path の分解 */
if((p=strstr(server_path, "http://"))==NULL){
    usage(argv);
    return -3;
}
server = p + strlen("http://");
for(p = server; *p!='\0'; p++){
    if(*p=='/'){
        *p = '\0';
        path = p+1;
        break;
    }
}

/*
  pandora_lib では，root,, path, resource_type を設定するのが原則で
  あるが，今回のようにパスが完全にわかっている場合には，以下のよう
  に pdr_set_resource_type に完全なパスをセットすることも可能である．
  (root と path は設定しなくてよい)

  なお，このように 3 つにわかれているのでは，
  1) path に nusdims_to_path の出力をそのままセットできる．
     (関数 pdr_set_path_nus は NuSDaS の要素を pandora path に変換して，
     pdr_set_path を実行する)
  2) path だけが違う (たとえば時刻がちがうなど) とか，同じ資源の data と meta
     をとりたい場合など，変更のある部分だけを変更できる

  という利点があると考えている．
 */

pdr = pdr_new();
if(pdr == NULL){
    fprintf(stderr, "malloc error:%s,%d\n",__FILE__,__LINE__);
    return -10;
}

if((rt = pdr_set_timeout(pdr, 60))<0){
    fprintf(stderr, "pdr_set_timeout error:%d\n", rt);
    return rt;
}
else if((rt = pdr_set_server(pdr,server))<0){
    fprintf(stderr, "pdr_set_server error:%d\n", rt);
    return rt;
}
else if((rt = pdr_set_resource_type(pdr, path))<0){
    fprintf(stderr, "pdr_set_resource error:%d\n", rt);
    return rt;
}

if(host_header != NULL){
    if((rt = pdr_set_host_header(pdr, host_header)) < 0){
        fprintf(stderr, "pdr_set_host_header error:%d\n", rt);
```

```
            return rt;
        }
    }

    if(pdr_process(pdr)<0){
        code = pdr_get_status_code(pdr);
        fprintf(stderr, "Error! status code=%d\n", code);

    }
    fprintf(stderr,"Data Length: %d\n", pdr_get_data_len(pdr));
    if(print_header_flag == 1){
        pdr_print_all_headers(pdr,stderr);
    }
    fwrite(pdr_get_data(pdr), pdr_get_data_len(pdr),1,stdout);

    pdr_delete(pdr);
    return 0;
}
```

Bibliography

[1] John Parr Snyder. *Map Projections — a working manual.* U.S. G.P.O., Washington D.C., 1987. U.S. Geological Survey professional paper 1395.

[2] 国土地理院地理調査部地図編集課 (金沢敬). 300 万分の 1 「日本とその周辺」2. 国土地理院時報, Vol. 44, pp. 27–38, 1972.

[3] 寺澤寛一. 物理応用のための数学概論 (増訂版). 岩波書店, 東京, 1954.

[4] 牧原康隆. レーダーエコー合成図における合成手法について. レーダー観測技術資料, Vol. 39, pp. 1–10, 1990.