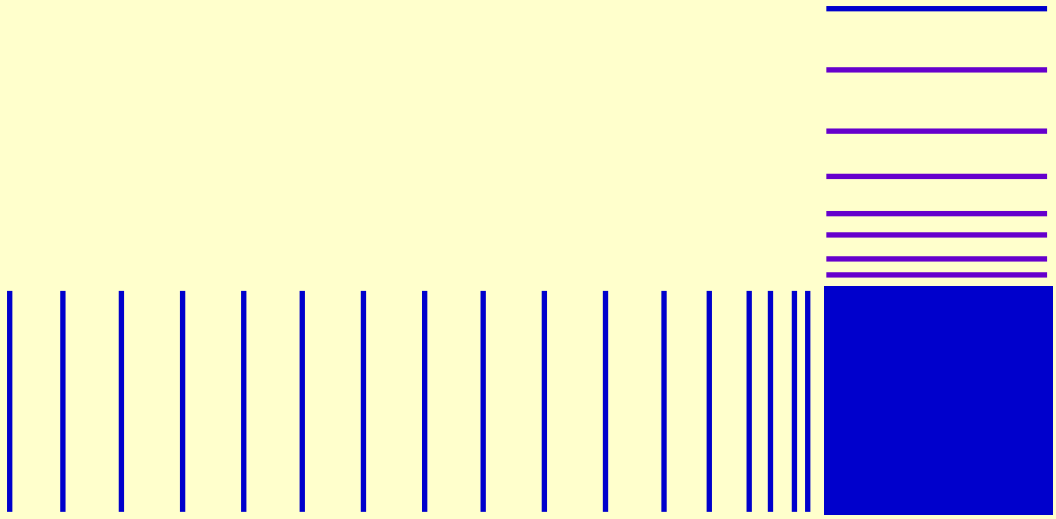
リアルタイムデータ共有メモリ機能

RT-DSM

ユーザーズガイド



HIOT-DSM-03

この製品を輸出される場合には、外国為替及び外国貿易法の規制
並びに米国輸出管理規則など外国の輸出関連法規をご確認のうえ、
必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

２０２１年 ５月（第１版）ＨＩＯＴ－ＤＳＭ－０１（廃版）
２０２１年 ９月（第２版）ＨＩＯＴ－ＤＳＭ－０２（廃版）
２０２２年 ７月（第３版）ＨＩＯＴ－ＤＳＭ－０３

ご注意

- このソフトウェアをご使用になる前に、このマニュアルの記載内容をよく読み、書かれている指示や注意を十分理解してください。

- このマニュアルの記載内容について理解できない内容、疑問点または不明点がございましたら、最寄りの当社営業までお知らせください。

- 当社提供ソフトウェアを改変して使用した場合には、発生した事故や損害につきましては、当社は責任を負いかねますのでご了承ください。

- 当社提供以外のソフトウェアを使用した場合の信頼性については、当社は責任を負いかねますのでご了承ください。

- このソフトウェアが万一故障したり、誤動作やプログラムに欠陥があった場合でも、ご使用されるシステムの安全が十分に確保されるよう、保護・安全回路は外部に設け、人身事故・重大な災害に対する安全対策などが十分確保できるようなシステム設計としてください。

はじめに

本マニュアルは、「RT-DSM（S-763A-96P)」の使い方などについて記述したものです。本ソフトウェアをご使用になる前に、このマニュアルをよくお読みください。

＜マニュアル構成＞

このマニュアルは、次のような構成となっています。

第1章 RT-DSMとは
第2章 ソフトウェア構成
第3章 RT-DSMのインストール
第4章 ライブラリ使用手順
第5章 Windows®側API仕様
第6章 CODESYS®側API仕様
第7章 制限事項
第8章 開発ツールとデータ定義ファイル
第9章 コマンドリファレンス
第10章 サンプルプログラム
第11章 組込みCDMSとのデータ連携

＜商標について＞

・Microsoft®、Windows®は、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
・CODESYS®は、ドイツCODESYS GmbHの登録商標です。
・EtherCAT®は、ドイツBeckhoff Automation GmbHによりライセンスされた特許取得済み技術であり、登録商標です。
・上記以外にこのマニュアルに記載されている他社製品名（ソフトウェア、ハードウェア）は、各社の登録商標、商標、または商品です。

＜関連ドキュメント＞

本機能に対応した CODESYS®を搭載する IoT 対応産業用コントローラ HF-W/IoT のソフトウェア PLC の使い方などについては、下記ドキュメントをご参照ください。

No.	ドキュメント名	備考
1	HF-W100E/IoT スタートアップガイド	(*1)
2	HF-W2000/IoT モデル 58/55/50 HF-W400E/IoT スタートアップガイド	(*1)

(*1) 以下の URL からご参照ください。
https://www.hitachi-ip.co.jp/products/hfw/products/iot_ctr/download.html

目次

第1章 RT-DSM とは

リアルタイムデータ共有メモリ機能（RT-DSM）は、Windows®および CODESYS®上で動作するプログラム間で、データの受け渡しを漏れなく行うためのソフトウェアです。リングバッファ構造のデータ共有メモリ（データ共有リングバッファ）を用いて、機能を実現しています。特長を以下に示します。

(1) API を介して、Windows®（情報系）および CODESYS®（制御系）上のプログラム間でデータ共有（受け渡し）を行います。

(2) 物理メモリ上で直接読み書きを行うため、高速にデータの受け渡しが可能です。

(3) 共有するデータ構成の定義は、定義ファイル（テキスト形式）で設定できます。

(4) データ共有リングバッファに書き込むデータは、約 30 秒間分蓄積可能（1ms 毎にデータを書き込む場合）なため、CODESYS®の高速周期データを漏れなく Windows®に渡すことができます。

このデータ共有リングバッファに、CODESYS®が制御する装置の各種データや情報系からの指令や解析フィードバックデータを格納することで、設備の監視や生産効率改善などに活用できます。また、API を介したデータ共有のため、本機能を搭載した環境であれば、ユーザープログラムの移行が行えます。

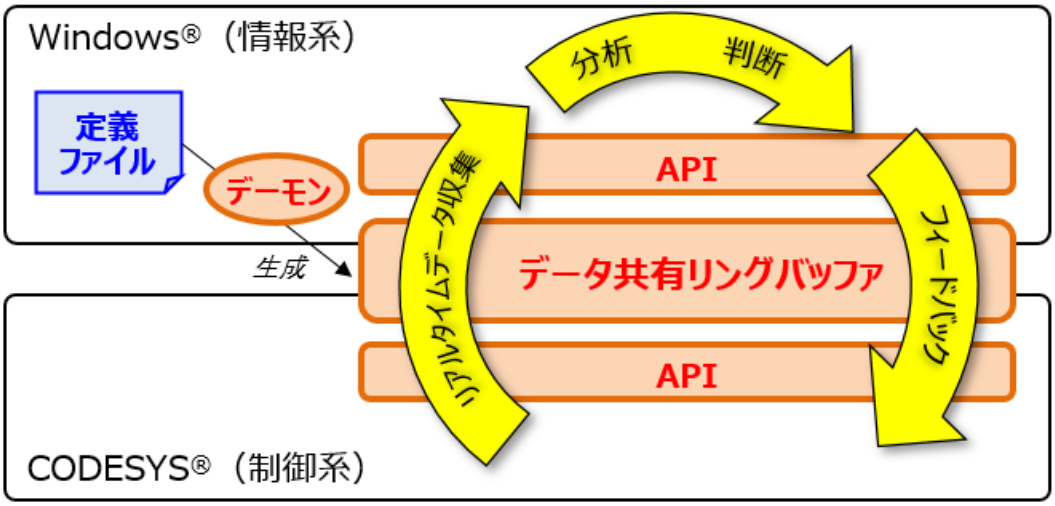


図 1　RT-DSM の利用イメージ

第2章 ソフトウェア構成

データ共有リングバッファは、リングバッファ定義ファイルに基づき PC 起動時に Windows®上のデーモンにより作成されます。リングバッファ定義ファイルは、ユーザーが作成するデータ定義ファイルを開発ツール（コマンド）に読み込ませて自動生成します。開発ツールとデータ定義ファイルの詳細については、「第 8 章 開発ツールとデータ定義ファイル」をご参照ください。

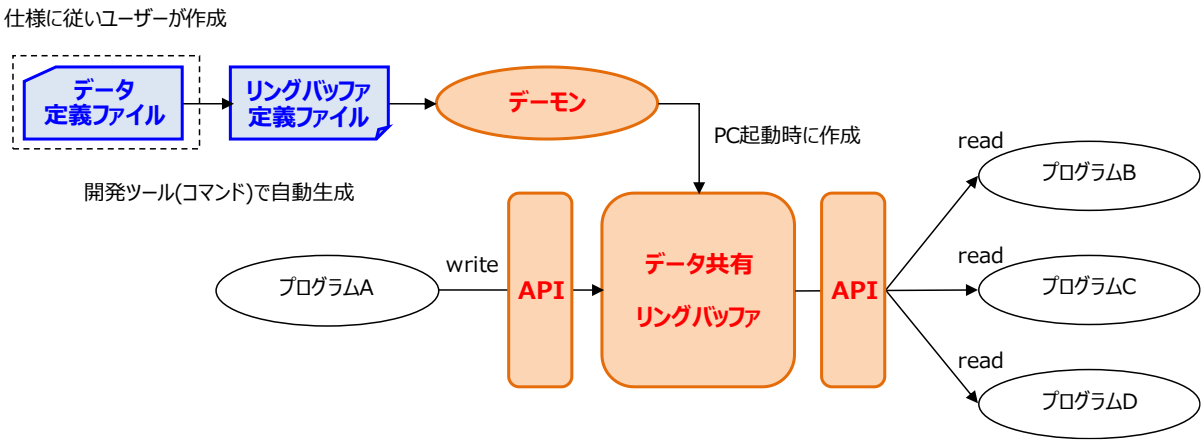


図 2　RT-DSM ソフトウェア構成

データ共有リングバッファは、図 3 に示す構造をしており複数作成できます。1 つのリングバッファに対し、書き込みが行えるのは 1 つのプログラムのみで、読み込みは複数のプログラムから可能です。リングバッファへの書き込み および 読み込みは、複数のデータを一まとめにしたデータ群で行います。このデータ群の単位を「ブロック」と呼びます。リングバッファはブロック N 個分のデータを蓄積し、N 個をオーバーして書き込むと古いブロックから順に上書きされます。

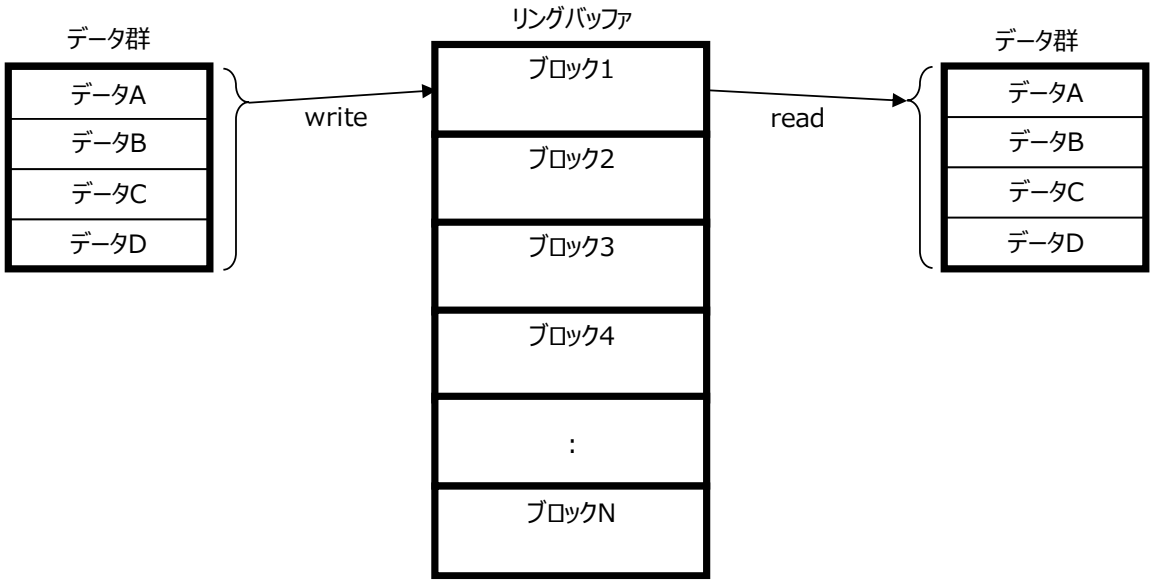


図 3　データ共有リングバッファの構造

第3章 RT-DSM のインストール

RT-DSM のインストール方法を以下に記載します。

3.1. インストール作業の前に

RT-DSM のインストールに際して、インストール作業で必要な項目 および 動作環境について説明します。

インストール作業で必要な項目を以下に示します。作業を開始する前に予め確認しておいてください。

＜インストール作業で必要になる項目＞

項目	内容
Name	任意の名前を入力ください。
Serial number	本ソフトウェアのインストールで必要な 12 桁のシリアルナンバーです。 本製品購入時に、弊社より通知されたものを入力ください。

ソフトウェア開発・実行環境を以下に示します。

＜ソフトウェア開発環境＞

項目	内容
Windows®アプリケーション開発環境	C 言語での開発が可能な任意の環境
CODESYS®開発環境	CODESYS® Development System（V3.5 SP16 Patch 4）

＜ソフトウェア実行環境＞

項目	内容
OS（Windows®）	Microsoft® Windows® 10 Iot Enterprise（64bit）
CODESYS®リアルタイム実行環境	CODESYS® Control RTE（V3.5 SP16 Patch 4）
.NET Framework	.NET Framework 3.5 または .NET Framework 4

留意事項

装置に DVD ドライブが搭載されていない場合は、USB 接続の外付け光ディスクドライブ（DVD メディアを読み込めるドライブ）を用意して装置に接続してください。

3.2. RT-DSM のインストール

RT-DSM のインストール手順について説明します。なお、インストールはコンピューターの管理者アカウントでログオンして行ってください。

① Administrator 権限を持つアカウントでログオンします。

② 「ファイル名を指定して実行」ウィンドウを開きます。［スタート］ボタンを右クリックし、表示されたメニューより「ファイル名を指定して実行」をクリックします。

③ セットアッププログラムを起動します。名前のボックスに以下を入力して[Enter]キーを押します。
"D:¥DSM.msi"

※ DVD 媒体でのインストール手順として、DVD ドライブを D ドライブと仮定したパスを指定しています。下線部のパスはセットアッププログラムが格納されている場所に応じて変更してください。

④ 「RT-DSM」用のセットアップウィザード画面が表示されます。「Next」ボタンをクリックします。

⑤ 画面に従いインストールします。Name には任意の名前を、Serial number には本ライブラリ購入時に弊社から提供されたキーを入力します。
「ユーザーアカウント制御」画面が表示される場合は、「はい」ボタンをクリックします。

⑥ RT-DSM のインストールが完了したことを示す画面が表示されます。「Close」ボタンをクリックしてセットアッププログラムを終了します。

⑦ Windows®を再起動します。

3.3. RT-DSM のアンインストール

RT-DSM のアンインストール手順について説明します。なお、アンインストールはコンピューターの管理者アカウントでログオンして行ってください。

① 「スタート」メニューから「コントロールパネル」を開きます。［スタート］ボタンをクリックし、［Windows システムツール］-［コントロールパネル］をクリックします。

② 「プログラムのアンインストール」を選択します。

③ インストールされているプログラムのリストから、「RT-DSM」を選択し、アンインストールを実行します。

④ 「RT-DSM」の削除を確認するメッセージが表示されます。「はい」ボタンをクリックします。
「ユーザーアカウント制御」画面が表示される場合は、「はい」ボタンをクリックします。

⑤ Windows®を再起動します。

第4章 ライブラリ使用手順

RT-DSM のライブラリの使用手順を以下に記載します。

（1） Windows®側ライブラリ使用手順

【ライブラリ格納場所】

ライブラリ関連ファイルは、以下に格納しています。

「C:¥Program Files¥HX-DSM」

No.	フォルダ	ファイル	内容
1	bin	DsmWlib.dll	データ共有リングバッファ API の DLL
2		defconv.exe	定義ファイル作成用開発ツール
3		defconvDB.exe	定義ファイル作成用開発ツール（組込み CDMS 対応版 (*1)）
4		dsmconf.exe	データ共有リングバッファの定義情報表示コマンド
5		dsmconfchk.exe	データ共有リングバッファの定義ファイル文法チェックコマンド
6		dsmresetwop.exe	データ共有リングバッファの読み書きオープン状態リセットコマンド
7	conf	buffer.ini	お使いになるリングバッファ定義ファイルを格納してください。ini ファイルは複数格納しないでください。（初期状態では No.8 を格納しています）
8	conf/sample	buffer.ini	リングバッファ定義ファイル（サンプル）
9	inc	dsm_rb.h	データ共有リングバッファ API のヘッダファイル
10	lib	DsmWlib.lib	データ共有リングバッファ API のインポートライブラリファイル

(*1) 組込み CDMS の詳細は「第 11 章 組込み CDMS とのデータ連携」を参照。

【ライブラリ使用手順】

ライブラリの使用手順を以下に記載します。

＜ヘッダファイルの設定＞

コンパイルに必要な API のヘッダファイルは、dsm_rb.h をご使用ください。

① コンパイラのインクルードディレクトリに dsm_rb.h の格納フォルダを追加

② ソースコードでインクルード（#include "dsm_rb.h"の指定）

＜インポートライブラリファイルの設定＞

リンクに必要な API のインポートライブラリファイルは、DsmWlib.lib をご使用ください。

① リンカーの依存ライブラリファイルに DsmWlib.lib を追加

（２） CODESYS®側ライブラリ使用手順

【ライブラリ格納場所】

ライブラリファイルは、以下に格納しています。

「C:¥Program Files¥HX-DSM¥Library¥Dsm_Cdsys.package」

【ライブラリ使用手順】

CODESYS®の開発環境に、ライブラリを追加する手順を説明します。以下の手順を行う必要があります。

[1] ライブラリリポジトリに、ライブラリ（パッケージ形式）をインストール

[2] ライブラリマネージャーに、ライブラリを追加

＜ライブラリインストール方法（パッケージ形式）＞

ライブラリのインストールを一度でも実施している場合、以下の手順は不要です。「＜ライブラリ追加方法(CODESYS®プロジェクト毎に実施)＞」の手順から実施してください。

① CODESYS®開発環境上部の「ツール」-「パッケージマネージャー」をクリックします。

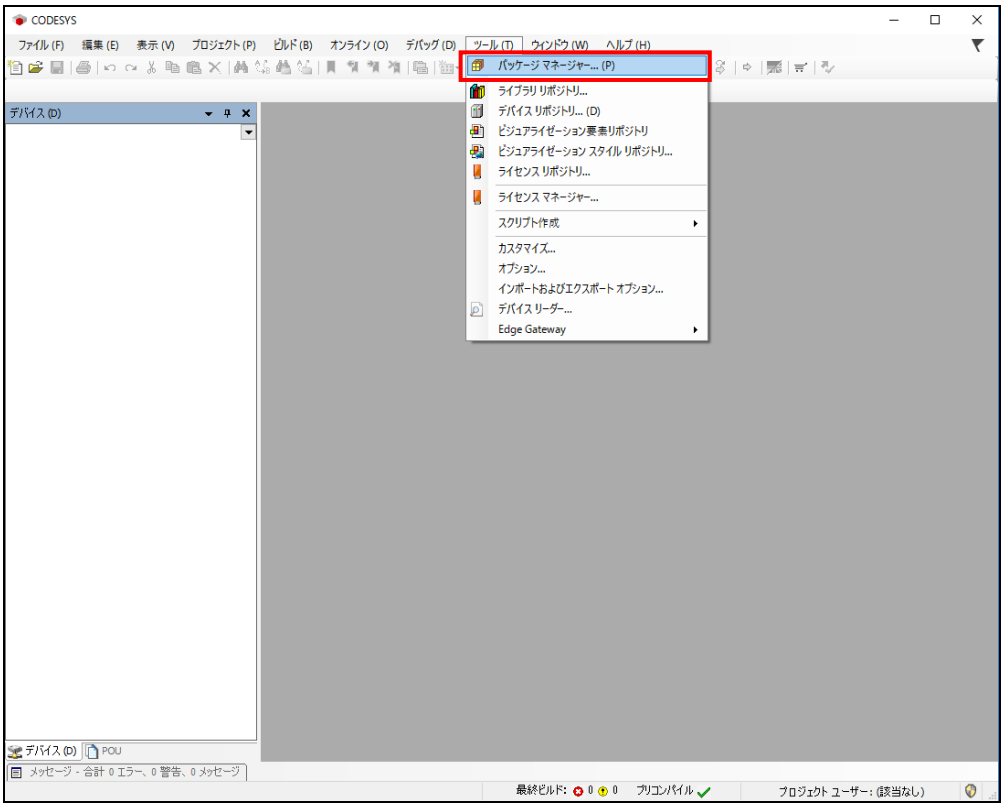


図 4 パッケージマネージャー表示

② 表示されるパッケージマネージャーダイアログで「インストール」ボタンを押下します。

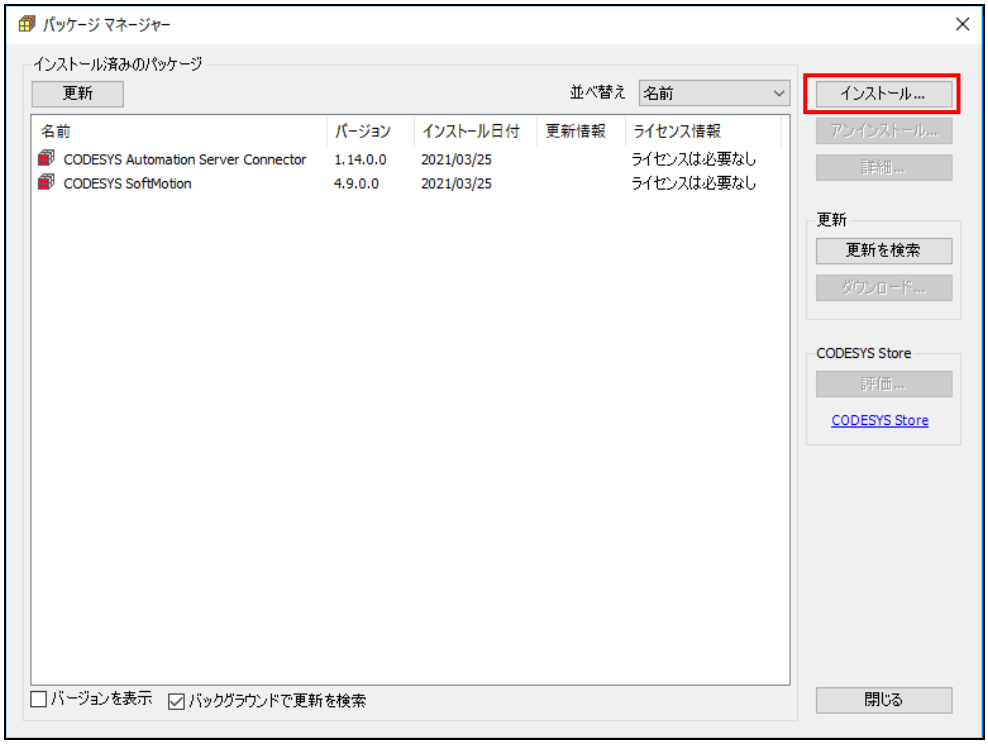


図 5 パッケージマネージャーダイアログ

③ 「C:¥Program Files¥HX-DSM¥Library」フォルダ内の「Dsm_Cdsys.package」ファイルを指定し、「開く(O)」ボタンを押下します。

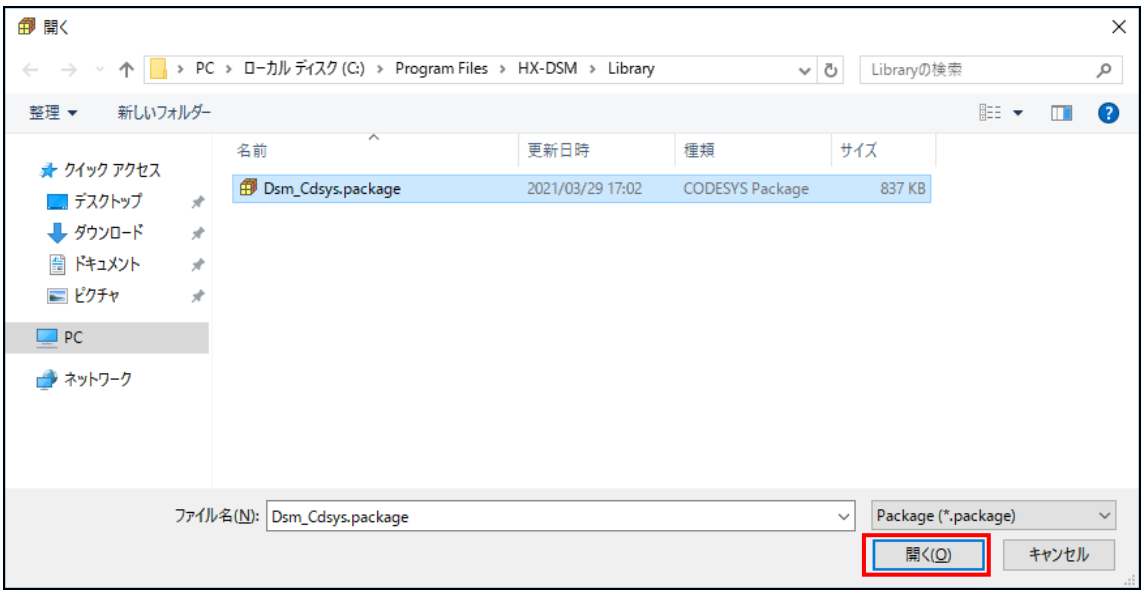


図 6 ライブラリパッケージインストール

④　パッケージのインストール画面が表示されます。画面に従いインストールを進めます。License Agreement の画面が表示されますので、内容を確認し「私は、上記の使用許諾契約を読んで理解したので同意します。」にチェックを入れ「Next」ボタンを押下します。

図 7 インストール画面(License Agreement)

⑤　セットアップの種類を選択します。デフォルトの「代表的なセットアップ」を選択したまま「Next」ボタンを押下します。「ユーザーアカウント制御」画面が表示される場合は、「はい」ボタンをクリックします。

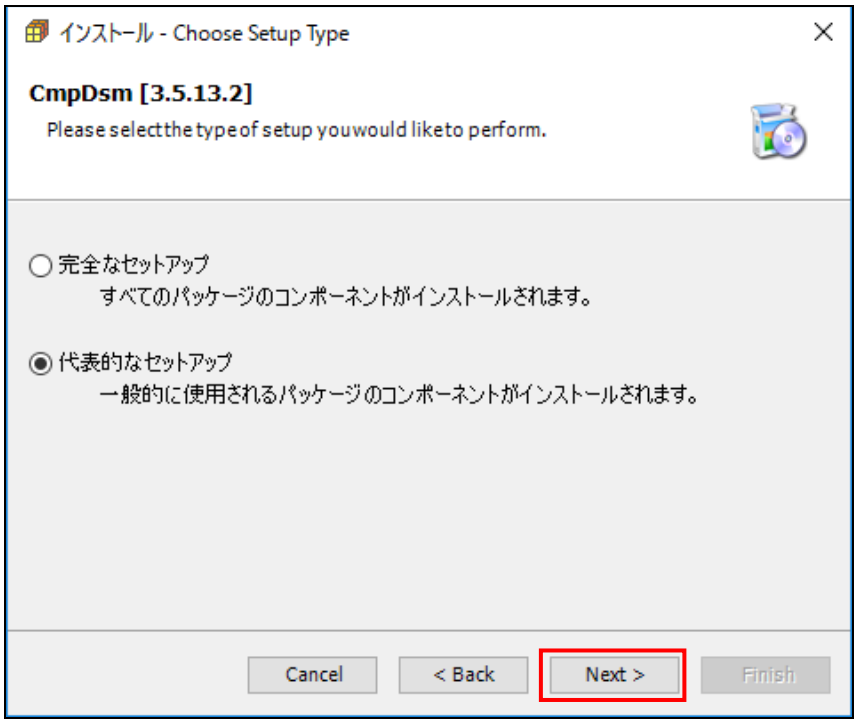


図 8 インストール画面(Choose Setup Type)

⑥　パッケージのインストールを開始します。1～2 分後、下記画面が表示されインストールが正常終了したことを確認し「Next」ボタンを押下します。

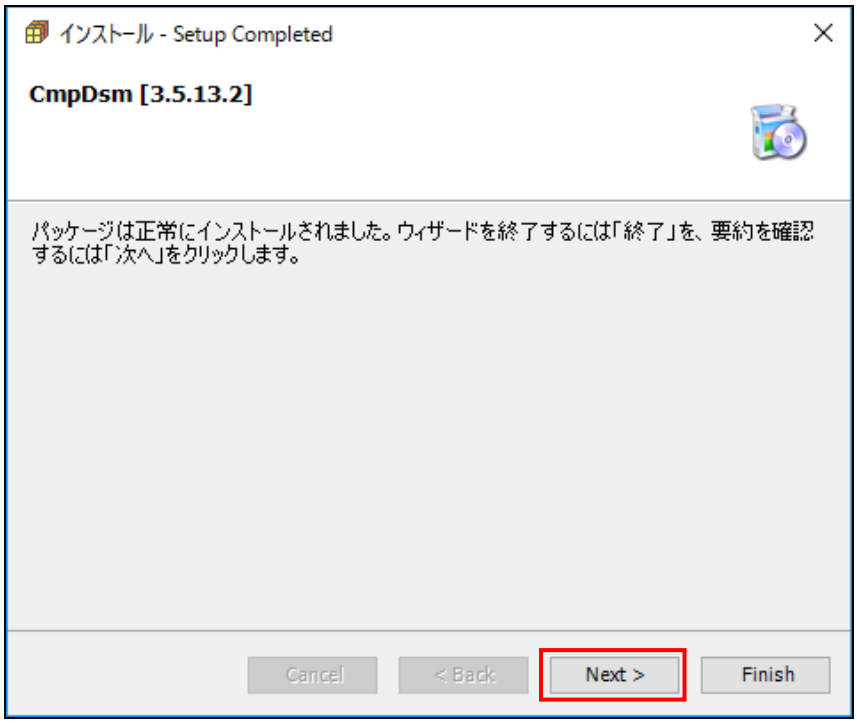


図 9 インストール画面(Setup Completed)

⑦　インストールの概要を確認し、「Finish」ボタンを押下して終了します。

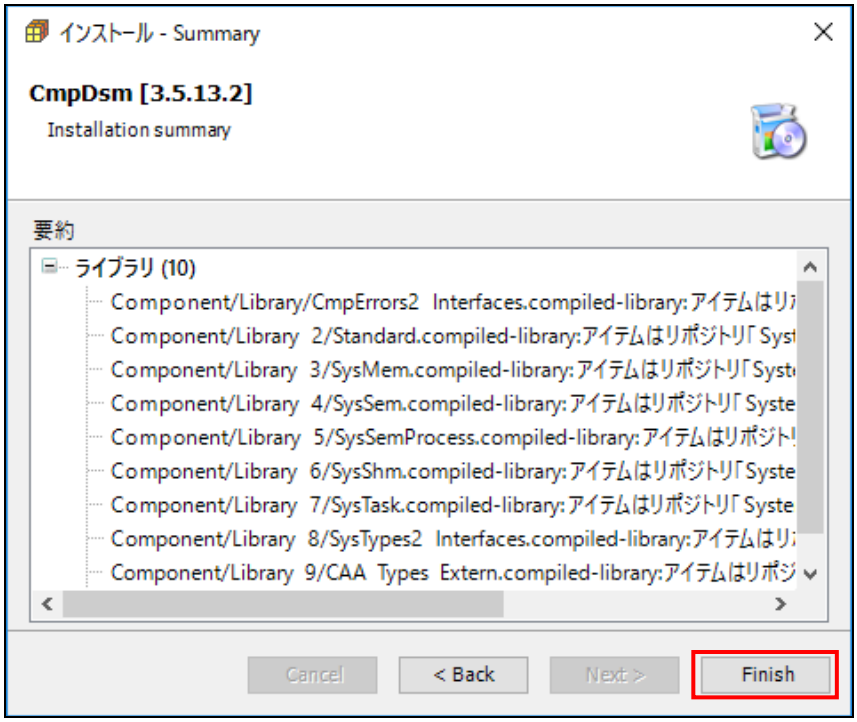


図 10 インストール画面(Summary)

⑧　パッケージマネージャーダイアログで「CmpDsm」が表示されることを確認します。

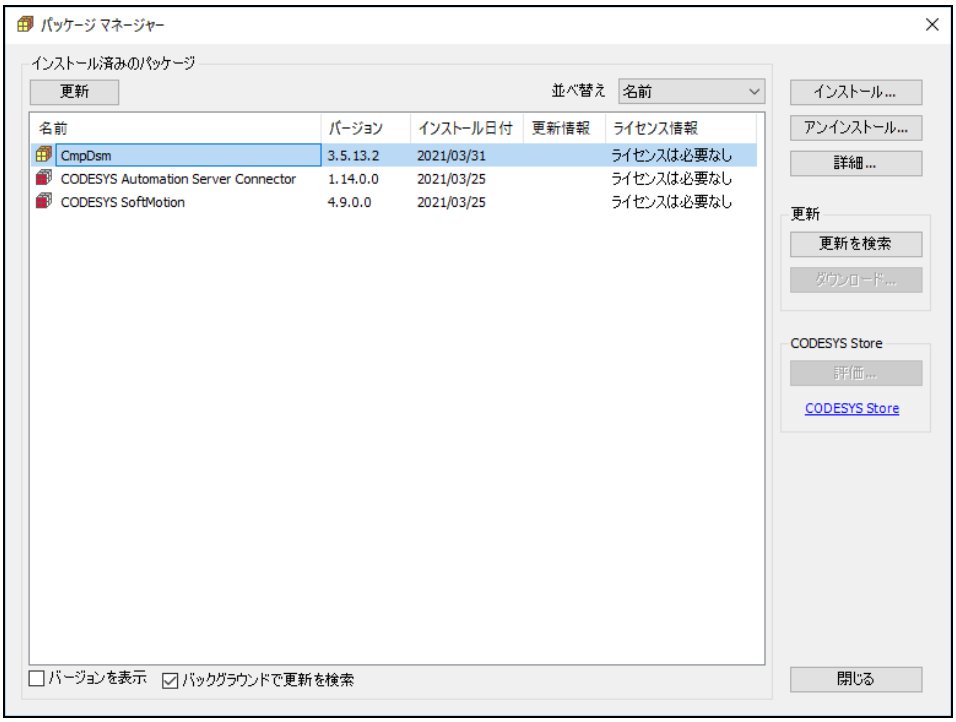


図 11　パッケージマネージャーダイアログ(インストール後)

＜ライブラリ追加方法(CODESYS®プロジェクト毎に実施)＞

① デバイス欄の「ライブラリマネージャー」をダブルクリックします。

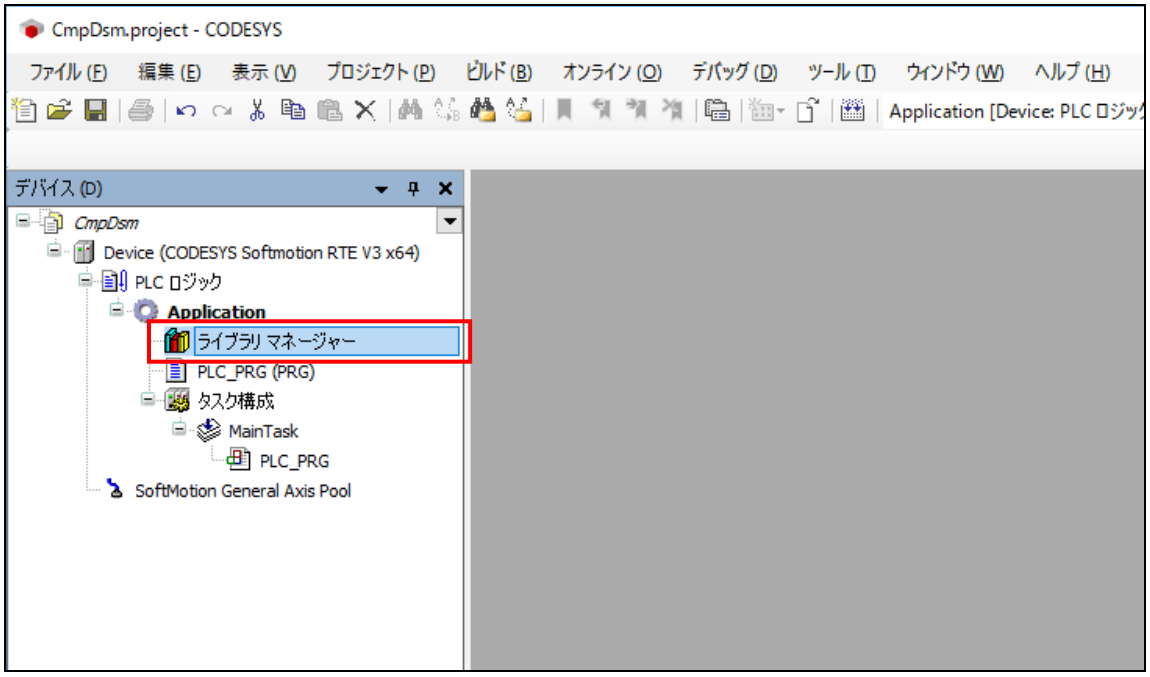


図 12 ライブラリマネージャー表示

② ライブラリマネージャー内の「ライブラリの追加」を押下します。

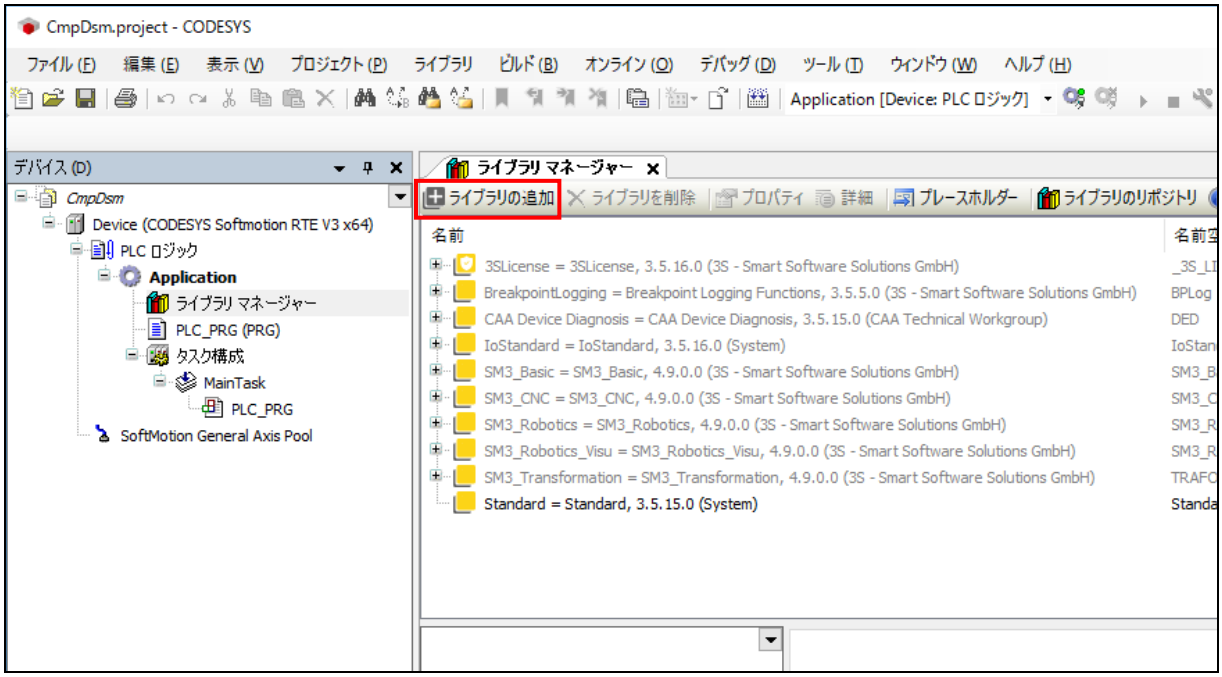


図 13 ライブラリ追加ダイアログ表示

③　ダイアログが表示されたら、左下部の「上級者向け...」ボタンを押下します。

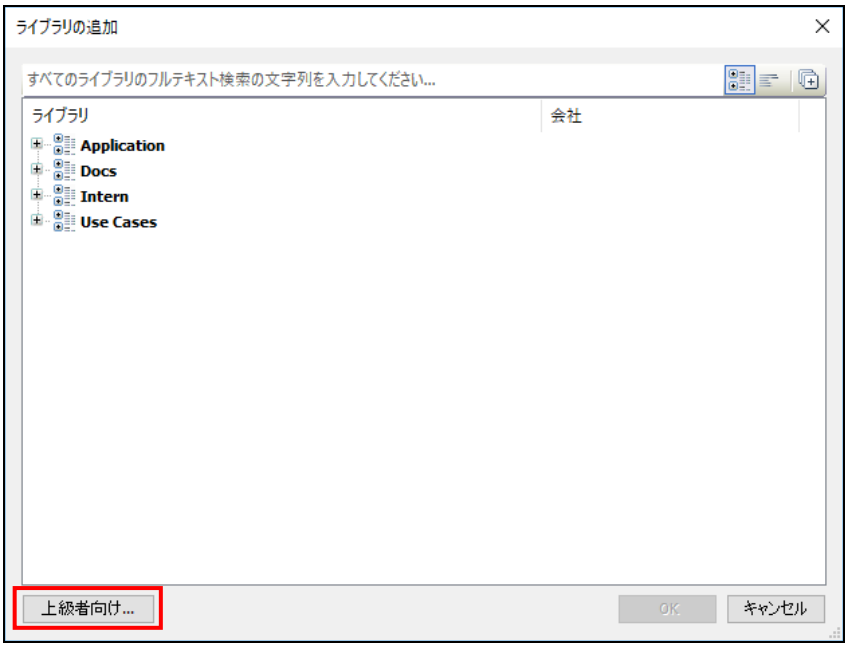


図 14 ライブラリ追加ダイアログ

④　ダイアログが切り替わったら、「System ⇒ SysLibs」内の「CmpDsm」を選択し、OK ボタンを押下します。

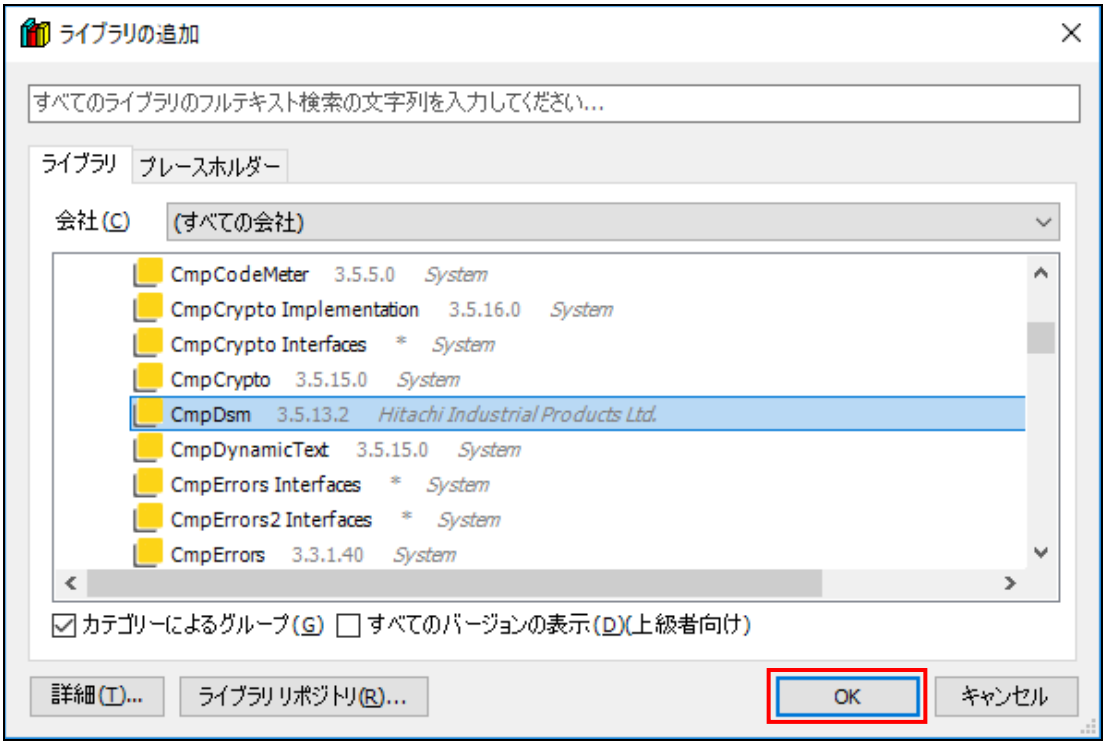


図 15 ライブラリ追加

⑤　なお、検索バーを利用し、ライブラリの検索をすることも可能です。

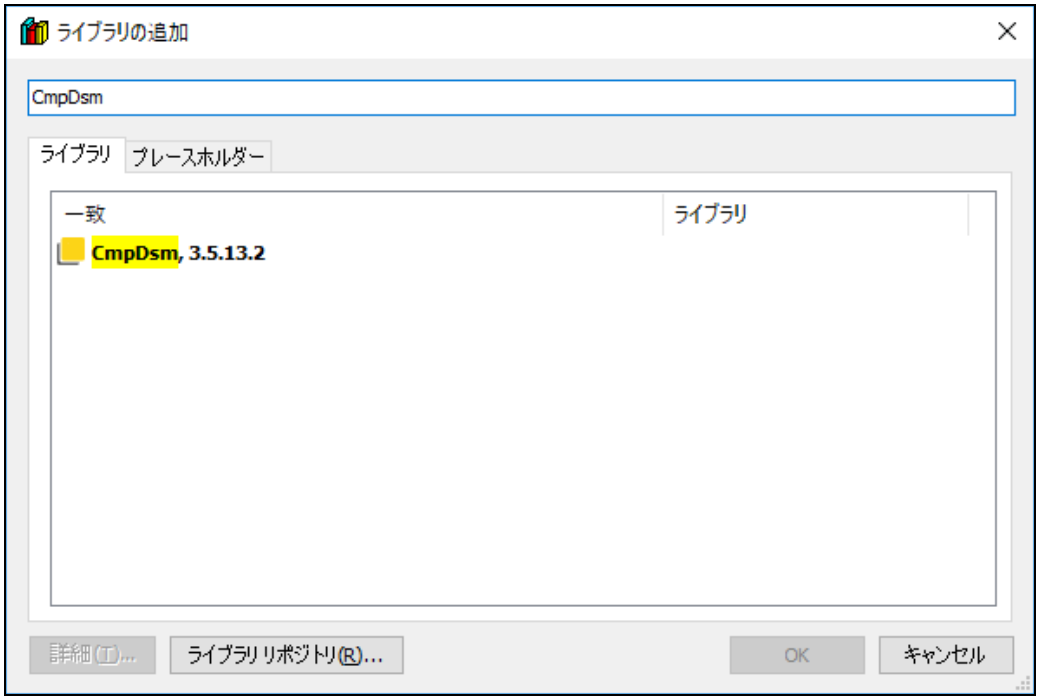


図 16　ライブラリ追加(検索バー使用時)

⑥　ライブラリマネージャーに以下のライブラリが追加されたら完了となります。

・CmpDsm, 3.5.13.2 (Hitachi Industrial Products Ltd.)

※　会社情報には「, (カンマ)」が使用できないため省略しています。

第5章 Windows®側 API 仕様

RT-DSM の Windows®側の API 仕様を以下に記載します。

【Windows®側 API 一覧】

データ共有リングバッファへアクセスするための Windows®側 API の一覧を表 1 に示します。

表 1　Windows®側 API 一覧

No.	関数名	説明	参照項
1	Dsm_WinOpenRb	データ共有リングバッファのオープン	5.1(1)
2	Dsm_WinWriteRb	データ共有リングバッファへのデータ書き込み	5.1(2)
3	Dsm_WinReadRb	データ共有リングバッファからの最新データ読み込み	5.1(3)
4	Dsm_WinMReadRb	データ共有リングバッファからの未読データ読み込み	5.1(4)
5	Dsm_WinCloseRb	データ共有リングバッファのクローズ	5.1(5)
6	Dsm_WinResetWOpenRb	データ共有リングバッファの読み書きオープン状態をリセット	5.1(6)

【Windows®側 API 仕様】

Windows®側 API の仕様一覧を表 2 に示します。

表 2　Windows®側 API の仕様

No.	項目	内容
1	API の提供形態	C 言語用の関数として 64 ビット DLL で提供
2	エンディアン	リトルエンディアン ただし、リングバッファに格納するデータ群のエンディアンは書き込みプログラムに依存します。
3	構造体のアライメント	8 バイトアライメント リングバッファ定義ファイルは、予めアライメントを意識した開発ツールを用いて作成してください。

5.1. Windows®側 API リファレンス

（1）Dsm_WinOpenRb

＜名　　前＞　Dsm_WinOpenRb － データ共有リングバッファのオープン

＜形　　式＞

```
int             Dsm_WinOpenRb(buffname, type, &hBuff, &blksz, &blkcase, hashcode)
char            *buffname;
unsigned int    type;
DSM_RB_HANDLE   *hBuff;
unsigned int    *blksz;
unsigned int    *blkcase;
unsigned int    hashcode;
```

＜機能説明＞

Dsm_WinOpenRb は、データ共有リングバッファをオープンする関数です。

*buffname　：データ共有リングバッファに割り当てた名称を指定してください。書き込み側と読み込み側で、同じリングバッファ名称を指定することでデータを共有できます。リングバッファ名称の長さは最大 31 文字です。31 文字以上を指定した場合、31 文字までの名称を使用します。名称には英数字、"_"(アンダーバー)、および"-"(ハイフン)を使用できます。指定された名称がリングバッファ定義ファイルの定義内容と不一致の場合は、異常終了します。

type　　：読み込み用、または読み書き用のアクセス種別を指定してください。
　0：読み込み用
　1：読み書き用

*hBuff　　：リングバッファのハンドルの格納先アドレスを指定してください。指定したリングバッファのハンドル情報を格納します。リングバッファの読み込み、書き込み、クローズ時にそれぞれの関数に渡してください。

*blksz　　：リングバッファの 1 ブロックのサイズが返ります。8 バイトアラインのサイズになります。

*blkcase　：リングバッファのブロック数が返ります。2 のべき乗で切り上げた数になります。

hashcode　：オープンするリングバッファのハッシュコードを指定してください。ハッシュコードは、開発ツールが構造体の情報を基にインクルードファイルに出力します。また、構造体の使用先リングバッファとの関連付け情報として、ハッシュコードはリングバッファ定義ファイルにも出力します。インクルードファイルにおけるハッシュコードの出力フォーマットは下記の通りです。

”struct xxx { ... };”　　(xxx はリングバッファ名称)

”#define　xxx_HASH　yyy”　(xxx はリングバッファ名称、yyy はハッシュコード)

リングバッファ定義ファイルにおけるハッシュコードの出力フォーマットは下記のとおりです。

”BuffName=xxx”　(xxx はリングバッファ名称)

”Hash=yyy”　　(yyy はハッシュコード)

指定したハッシュコードとリングバッファ定義ファイルの定義内容が異なる場合は異常終了します。

＜診　　断＞

Dsm_WinOpenRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_INIT
3	DSM_E_WINAPI
4	DSM_E_EXTWRTOPN
5	DSM_W_STOP
6	DSM_E_RBMALLOC
7	DSM_E_MGROPEN
8	DSM_E_RBOPEN
9	DSM_E_RBNAME
10	DSM_E_READONLY
11	DSM_E_UNMATCHHASH

＜注意事項＞

- 読み書き用リングバッファのオープンは、１つのリングバッファにつき１つのスレッドのみが実行可能です。既に読み書き用でリングバッファがオープンされている場合、その後の読み書き用のリングバッファオープンは失敗します。
- リングバッファの書込可能プログラム種別が「Windows」以外の場合、アクセス種別に読み書き用を指定したリングバッファオープンは失敗します。
- 既に読み込み用でオープンしているリングバッファに再度読み込み用オープンを発行した場合、異常終了せずにリングバッファのハンドルを新しく確保します。
- Dsm_WinCloseRb を発行せずにオープンを繰り返すとリングバッファのハンドルに関連するWindows®のリソースが枯渇します。
- 同一リングバッファハンドルを複数のスレッドから同時に使用できません。スレッド毎にオープンを実施して、リングバッファハンドルを用意してください。同一リングバッファハンドルに対して複数のスレッドから同時に使用した場合、プログラムエラーや API のエラーが発生します。

（2）Dsm_WinWriteRb

＜名　　前＞　Dsm_WinWriteRb　－　データ共有リングバッファへのデータ書き込み

＜形　　式＞

```
int                 Dsm_WinWriteRb(hBuff, &pData, size)
DSM_RB_HANDLE       hBuff;
unsigned char       *pData;
unsigned int        size;
```

＜機能説明＞

Dsm_WinWriteRb は、データ共有リングバッファにデータを書き込む関数です。

hBuff　　: Dsm_WinOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

*pData　　: 書き込むデータの先頭アドレスを指定してください。書き込むデータのエリアとして、書き込みサイズ以上を用意してください。

size　　: 書き込むデータのサイズ(バイト)を指定してください。1 ブロックのサイズより大きい値を指定した場合は、異常終了します。

＜診　　断＞

Dsm_WinWriteRb は、処理が正常終了すると書き込んだデータの個数である 1 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_STOP
3	DSM_E_OVERSIZE
4	DSM_E_NOWRITE
5	DSM_E_RESETWOP

＜注意事項＞

- 読み書き用にオープンしたリングバッファのみ、書き込みできます。
- 性能の低下を避けるため、リングバッファへの書き込みの排他制御は行いません。
- size は 8 バイトの倍数以外を指定すると性能低下の要因になります。

（3）Dsm_WinReadRb

＜名　　前＞Dsm_WinReadRb － データ共有リングバッファからの最新データ読み込み

＜形　　式＞

```
int                 Dsm_WinReadRb(hBuff, &pData)
DSM_RB_HANDLE       hBuff;
unsigned char       *pData;
```

＜機能説明＞

Dsm_WinReadRb は、データ共有リングバッファの最新データを読み込む関数です。Dsm_WinReadRb では、未読や既読の管理は行いません。常に最新のデータを読み込み対象とします。データの読み込みサイズは、リングバッファのブロックサイズ固定です。

hBuff　　：Dsm_WinOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

*pData　　：読み込みデータの格納先の先頭アドレスを指定してください。読み込みデータの格納先エリアとして、リングバッファのブロックサイズ以上を用意してください。

＜診　　断＞

Dsm_WinReadRb は、処理が正常終了すると読み込んだデータの個数である 1 を返し、異常終了すると負の値を返します。書き込み関数による書き込みが一度も無い場合は、0 が返ります。最新データの読み込み中に書き込み関数で読み込みデータが上書きされた場合、異常終了します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_STOP
3	DSM_E_OVERWRITE

＜注意事項＞

- 性能の低下を避けるため、リングバッファからの読み込みの排他制御は行いません。
- Dsm_WinOpenRb の引数 blksz にブロックサイズが返りますので、データの格納先のエリアサイズとして使用してください。
- エリアサイズが不足している場合、プログラムエラーやデータ破壊が発生します。

（4）Dsm_WinMReadRb

＜名　　前＞Dsm_WinMReadRb － データ共有リングバッファからの未読データ読み込み

＜形　　式＞

int	Dsm_WinMReadRb(hBuff, num ,&pData, &onSkip)
DSM_RB_HANDLE	hBuff;
unsigned int	num;
unsigned char	*pData;
unsigned int	*onSkip;

＜機能説明＞

Dsm_WinMReadRb は、データ共有リングバッファの時系列データ群を、最新から指定した個数分の中で未読データのみを全て読み込みます。読み込んだデータは既読データとなり、Dsm_WinMReadRb の読み込み対象から外れます。未読と既読は、リングバッファのハンドル毎に管理します。データの読み込み最大サイズは「リングバッファのブロックサイズ × num で指定した個数」です。

hBuff　：Dsm_WinOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

num　：読み込み対象とするデータの個数(1～ブロック数)を指定してください。

*pData　：読み込みデータの格納先の先頭アドレスを指定してください。読み込みデータの格納先エリアとして、リングバッファのブロックサイズ以上を用意してください。読み込んだデータは、最古から最新の順番に格納します。

*onSkip　：データ連続確認フラグの格納先の先頭アドレスを指定してください。データ連続確認フラグの格納先として「unsigned int」のサイズ以上を用意してください。未読データの読み込みが正常終了した時､前回の Dsm_WinMReadRb で読み込んだデータとの連続性の判定結果を返します。同一リングバッファのハンドルに対してのみ有効な判定です。

0：前回の読み込みデータに対して、今回の読み込みデータは連続しています。

1：前回の読み込みデータに対して、今回の読み込みデータは連続していません。

＜診　　断＞

Dsm_WinMReadRb は、処理が正常終了すると読み込んだデータの個数を返し、異常終了すると負の値を返します。書き込み関数による書き込みが一度も無い場合、または未読データが無い場合は 0 が返ります。最新データの読み込み中に書き込み関数でいずれかの読み込みデータが上書きされた場合、異常終了します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_STOP
3	DSM_E_OVERBLOCK
4	DSM_E_OVERWRITE

＜注意事項＞

- 性能の低下を避けるため、リングバッファからの読み込みの排他制御は行いません。
- Dsm_WinOpenRb の引数 blksz にブロックサイズが返りますので、データの格納先のエリアサイズとして使用してください。
- エリアサイズが不足している場合、プログラムエラーやデータ破壊が発生します。
- Dsm_WinMReadRb 発行後に Dsm_WinReadRb を発行しても、データの未読状態と既読状態は変化しません。

＜ご参考：num、ブロック数のチューニング＞

num に指定する個数は、リングバッファのブロック数より小さい値を指定してください。例えば、ブロック数が 16 の時、num も 16 にすると、最新から 16 個目のブロックは常に書き込み対象になるため、読み込み対象のデータが上書きされ Dsm_WinMReadRb が異常終了となり読み込みできません。16 個目を読む場合は、書き込み側プログラムを停止してください。num および ブロック数は、プログラムの処理、実行環境を考慮した上で調整してください。

（５）Dsm_WinCloseRb

＜名　　前＞　Dsm_WinCloseRb　－　データ共有リングバッファのクローズ

＜形　　式＞

```
int                 Dsm_WinCloseRb(hBuff)
DSM_RB_HANDLE       hBuff;
```

＜機能説明＞

Dsm_WinCloseRb は、オープン中のデータ共有リングバッファをクローズする関数です。

hBuff　　：Dsm_WinOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

＜診　　断＞

Dsm_WinCloseRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG

＜注意事項＞

・ リングバッファを読み書き用でオープンしているプログラムが異常終了、もしくは強制終了させた場合、対象のリングバッファを読み書き用でオープンできなくなる場合があります。その場合は、[Dsm_WinResetWOpenRb]API、または[dsmresetwop]コマンドで読み書き用オープン状態をリセットしてください。

・ プログラムが正常に終了する場合、Dsm_WinCloseRb を発行しなくても Dsm_WinOpenRb で確保したリソースはプログラムの終了時に自動的に解放します。

（6）Dsm_WinResetWOpenRb

＜名　　前＞　Dsm_WinResetWOpenRb － データ共有リングバッファの読み書きオープン状態をリセット

＜形　　式＞

```
int             Dsm_WinResetWOpenRb(buffname)
char            *buffname;
```

＜機能説明＞

Dsm_WinResetWOpenRb は、指定したデータ共有リングバッファの読み書きオープン状態をリセットする関数です。読み書きオープンしていた処理が異常終了し、読み書きオープンができなくなった場合に使用します。

*buffname　：読み書きオープン状態をリセットするリングバッファの名称を指定してください。

＜診　　断＞

Dsm_WinResetWOpenRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_INIT
3	DSM_E_MGROPEN
4	DSM_W_STOP
5	DSM_E_RBNAME

＜注意事項＞

・ API の誤動作の原因になるため、正常に動作している状態では使用しないでください。

5.2. Windows®側 API エラーコード

API の関数が返すエラーコードの一覧を表 3 に示します。

表 3　Windows®側 API エラーコード一覧

No.	エラーコード	値	内容	対処方法
1	DSM_W_INIT	-1	デーモンによる初期化が終わっていません。	プログラムの起動順序を見直し、リトライしてください。
2	DSM_E_ARG	-2	引数の設定が異常です。	引数を確認し、プログラムを見直してください。プログラムの修正を実施の上、再立ち上げしてください。
3	DSM_E_WINAPI	-3	WINAPI のエラーによる内部異常です。	logsave コマンドで RAS 情報を収集して、製品責任元に連絡してください。
4	DSM_E_EXTWRTOPN	-4	既に読み書き用にリングバッファがオープンされています。	他のリングバッファを使用するか、読み書き用にリングバッファをオープンした処理のクローズ漏れがないか確認してください。
5	DSM_E_NOWRITE	-5	リングバッファに書き込む権限を持っていません。	読み込み用でオープンしたリングバッファに書き込みをしていないか確認してください。
6	DSM_W_STOP	-6	デーモンが停止しています。(停止処理中も含む)	処理を停止し、リングバッファをクローズしてください。
7	DSM_E_RBMALLOC	-7	メモリの割り当て失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。
8	DSM_E_MGROPEN	-8	バッファ管理テーブルのオープン失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。解決しない場合は、logsave コマンドで RAS 情報を収集して、製品責任元に連絡してください。
9	DSM_E_RBOPEN	-9	リングバッファのオープン失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。解決しない場合は、logsave コマンドで RAS 情報を収集して、製品責任元に連絡してください。

10	DSM_E_RBNAME	-10	リングバッファ名称の取得失敗による異常です。	取得対象とするリングバッファ名称が異なっていないか、引数を確認してください。
11	DSM_E_OVERSIZE	-11	書き込みサイズが、0 または 1 ブロックサイズより大きいサイズで指定されています。	引数に設定しているサイズを確認してください。
12	DSM_E_OVERBLOCK	-12	指定ブロック数が 0 または最大ブロック数を超えて指定されています。	引数に設定しているブロック数を確認してください。
13	DSM_E_OVERWRITE	-13	データの読み込み中に上書きされています。	リングバッファの最大ブロック数やブロックサイズ または データの読み書きタイミングを見直してください。
14	DSM_E_READONLY	-14	読み込み専用のリングバッファを読み書き用にオープンしようとしています。	リングバッファの書込可能プログラム種別が「Windows」以外になっていないか、定義を見直してください。
15	DSM_E_UNMATCHHASH	-15	ハッシュコードが不一致です。	開発ツールで構造体を変更した後に、構造体を使用するプログラムが再コンパイルされているか確認してください。また、変更した後にデーモンを再起動しているか確認してください。
16	DSM_E_RESETWOP	-16	読み書きオープン状態がリセットされています。	dsmresetwop コマンド、または DsmWinResetWOpenRb API で読み書きオープン状態がリセットされています。 再度読み書き用でオープンしてから書き込みしてください。

第6章 CODESYS®側 API 仕様

RT-DSM の CODESYS®側の API 仕様を以下に記載します。

【CODESYS®側 API 一覧】

データ共有リングバッファへアクセスするための CODESYS®側 API 一覧を表 4 に示します。

表 4　CODESYS®側 API 一覧

No.	関数名	説明	参照項
1	Dsm_CdsysOpenRb	データ共有リングバッファのオープン	6.1(1)
2	Dsm_CdsysWriteRb	データ共有リングバッファへのデータ書き込み	6.1(2)
3	Dsm_CdsysReadRb	データ共有リングバッファからの最新データ読み込み	6.1(3)
4	Dsm_CdsysCloseRb	データ共有リングバッファのクローズ	6.1(4)
5	Dsm_CdsysResetWOpenRb	データ共有リングバッファの読み書きオープン状態をリセット	6.1(5)

【CODESYS®側 API 仕様】

CODESYS®側 API の仕様一覧を表 5 に示します。

表 5　CODESYS®側 API の仕様

No.	項目	内容
1	API の提供形態	CODESYS®アプリケーション開発用ライブラリとして提供
2	エンディアン	リトルエンディアン ただし、リングバッファに格納するデータ群のエンディアンは書き込みプログラムに依存します。また、外部接続機器(例：FL-net 接続機器)等が作成するデータについては、外部接続機器の仕様等に依存します。
3	構造体のアライメント	8 バイトアライメント リングバッファ定義ファイルは、予めアライメントを意識した開発ツールを用いて作成してください。

6.1. CODESYS®側 API リファレンス

（1）Dsm_CdsysOpenRb

＜名　　前＞　Dsm_CdsysOpenRb　－ データ共有リングバッファのオープン

＜形　　式＞

```
DINT          Dsm_CdsysOpenRb(buffname, accesstype, phBuff, pblksz, pblkcase, hashcode)
VAR_INPUT
    buffname:     STRING(31);
    accesstype:   UDINT;
    phBuff:       POINTER TO DSM_CD_RB_HANDLE;
    pblksz:       POINTER TO UDINT;
    pblkcase:     POINTER TO UDINT;
    hashcode:     UDINT
END_VAR
```

＜機能説明＞

Dsm_CdsysOpenRb は、データ共有リングバッファをオープンする関数です。

buffname　：データ共有リングバッファに割り当てた名称を指定してください。書き込み側と読み込み側で、同じリングバッファ名称を指定することでデータを共有できます。リングバッファ名称の長さは最大 31 文字です。31 文字以上を指定した場合、31 文字までの名称を使用します。名称には英数字、"_"(アンダーバー)、および"-"(ハイフン)を使用できます。指定された名称がリングバッファ定義ファイルの定義内容と不一致の場合は、異常終了します。

accesstype　：読み込み用、または読み書き用のアクセス種別を指定してください。
　0：読み込み用
　1：読み書き用

phBuff　：リングバッファのハンドルアドレスを指定してください。指定したリングバッファのハンドルを格納します。リングバッファの読み込み、書き込み、クローズ時にそれぞれの関数に渡してください。

pblksz　：リングバッファのブロックサイズを返す変数のアドレスを指定してください。pblksz が示す変数にブロックサイズが返ります。8 バイトアラインのサイズになります。

pblkcase　：リングバッファのブロック数を返す変数のアドレスを指定してください。pblkcase が示す変数にブロック数が返ります。2 のべき乗で切り上げた数になります。

hashcode　:オープンするリングバッファのハッシュコードを指定してください。ハッシュコードは、開発ツールが構造体の情報を基に ST 言語ソースファイルに出力します。出力フォーマットは下記のとおりです。

”TYPE xxx : STRUCT ・・・” (xxx はリングバッファ名称)
”VAR CONSTANT ・・・ xxx_HASH : UDINT := 16#yyy;　END_VAR”
(xxx はリングバッファ名称、yyy はハッシュコード)

指定したハッシュコードとリングバッファ定義ファイルの定義内容が異なる場合は異常終了します。

＜診　　断＞

Dsm_CdsysOpenRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_INIT
3	DSM_W_STOP
4	DSM_E_RBMALLOC
5	DSM_E_MGROPEN
6	DSM_E_RBOPEN
7	DSM_E_RBNAME
8	DSM_E_READONLY
9	DSM_E_UNMATCHHASH
10	DSM_E_WRONGVAL
11	DSM_E_SEMERROR

＜注意事項＞

- 読み書き用リングバッファのオープンは、1 つのリングバッファにつき 1 つのタスクのみが実行可能です。既に読み書き用でリングバッファがオープンされている場合、その後の読み書き用のリングバッファオープンは失敗します。
- あるタスクが読み書き属性でオープンした際、取得したハンドルを他タスクで使用しリングバッファの読み書き等を行った場合、その動作は保証されません。
- リングバッファの書込可能プログラム種別が「CODESYS」以外の場合、アクセス種別に読み書き用を指定したリングバッファのオープンは失敗します。
- 既に読み込み用でオープンしているリングバッファに再度読み込み用オープンを発行した場合、異常終了せずにリングバッファのハンドルを新しく確保します。
- リングバッファのオープンは、読み書きの場合約 10ms、読み込みの場合約 0.5ms の時間がかかります（複数タスクから同時に使用すると＋α の時間になります）。そのため、同じタスク内で EtherCAT の制御を行っている場合、同期ズレの原因となる可能性があります。その場合、EtherCAT タスクより実行優先度の低い別のタスクで、リングバッファのオープンを行うといった対応を推奨します。

（2）Dsm_CdsysWriteRb

＜名　　前＞　Dsm_CdsysWriteRb － データ共有リングバッファへのデータ書き込み

＜形　　式＞

```
DINT          Dsm_CdsysWriteRb(hBuff, pData, size)
VAR_INPUT
    hBuff:        DSM_CD_RB_HANDLE;
    pData:        POINTER TO BYTE;
    size:         __UXINT;
END_VAR
```

＜機能説明＞

Dsm_CdsysWriteRb は、データ共有リングバッファにデータを書き込む関数です。

hBuff　　: Dsm_CdsysOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

pData　　: 書き込むデータの先頭アドレスを指定してください。書き込むデータのエリアとして、書き込みサイズ以上を用意してください。

size　　: 書き込むデータのサイズ(バイト)を指定してください。1 ブロックのサイズより大きい値を指定した場合は、異常終了します。

＜診　　断＞

Dsm_CdsysWriteRb は、処理が正常終了すると書き込んだデータの個数である 1 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_INIT
3	DSM_W_STOP
4	DSM_E_OVERSIZE
5	DSM_E_NOWRITE
6	DSM_E_WRONGVAL

＜注意事項＞

- 読み書き用にオープンしたリングバッファのみ、書き込みできます。
- 性能の低下を避けるため、リングバッファへの書き込みの排他制御は行いません。
- size は 8 バイトの倍数以外を指定すると性能低下の要因になります。
- 書き込むデータのサイズは、使用するリングバッファのブロックサイズ以下を指定してください。書き込むデータのサイズが大きい場合、プログラムエラーやデータ破壊が発生します。

（3）Dsm_CdsysReadRb

＜名　　前＞　Dsm_CdsysReadRb － データ共有リングバッファからの最新データ読み込み

＜形　　式＞

```
DINT          Dsm_CdsysReadRb(hBuff, pData)
VAR_INPUT
    hBuff:        DSM_CD_RB_HANDLE;
    pData:        POINTER TO BYTE;
END_VAR
```

＜機能説明＞

Dsm_CdsysReadRb は、データ共有リングバッファの最新データを読み込む関数です。Dsm_CdsysReadRb では、未読や既読の管理は行いません。常に最新のデータを読み込み対象とします。データの読み込みサイズは、リングバッファのブロックサイズ固定です。

hBuff　　: Dsm_CdsysOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

pData　　: 読み込みデータの格納先の先頭アドレスを指定してください。読み込みデータの格納先エリアとして、リングバッファのブロックサイズ以上を用意してください。

＜診　　断＞

Dsm_CdsysReadRb は、処理が正常終了すると読み込んだデータの個数である 1 を返し、異常終了すると負の値を返します。Dsm_CdsysWriteRb による書き込みが一度も無い場合は 0 が返ります。最新データの読み込み中に書き込み関数で読み込みデータが上書きされた場合、異常終了します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_W_INIT
3	DSM_W_STOP
4	DSM_E_OVERWRITE
5	DSM_E_WRONGVAL

＜注意事項＞

- 性能の低下を避けるため、リングバッファからの読み込みの排他制御は行いません。
- データの格納先のエリアサイズは、使用するリングバッファのブロックサイズ以上を用意してください。エリアサイズが不足している場合、プログラムエラーやデータ破壊が発生します。Dsm_CdsysOpenRb の引数 pblksz にブロックサイズが返りますので、データの格納先のエリアサイズとして使用してください。

（4）Dsm_CdsysCloseRb

＜名　　前＞　Dsm_CdsysCloseRb　－　データ共有リングバッファのクローズ

＜形　　式＞

```
DINT            Dsm_CdsysCloseRb(hBuff)
VAR_INPUT
    hBuff:          DSM_CD_RB_HANDLE;
END_VAR
```

＜機能説明＞

Dsm_CdsysCloseRb は、オープン中のデータ共有リングバッファをクローズする関数です。

hBuff：Dsm_CdsysOpenRb 関数でオープンした、データ共有リングバッファのハンドルを指定してください。

＜診　　断＞

Dsm_WinCloseRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_E_ARG
2	DSM_E_SYSERROR

＜注意事項＞

- リングバッファを読み書き用でオープンしているプログラムが異常終了、もしくは強制終了させた場合、対象のリングバッファを読み書き用でオープンできなくなる場合があります。その場合は、[Dsm_CdsysResetWOpenRb]API、または[dsmresetwop]コマンドで読み書き用オープン状態をリセットしてください。
- CODESYS®アプリケーションを終了する際は、Dsm_CdsysCloseRb を発行し、Dsm_CdsysOpenRb で獲得したリソースを解放してください。

（５）Dsm_CdsysResetWOpenRb

＜名　　前＞　Dsm_CdsysResetWOpenRb　－　データ共有リングバッファの読み書きオープン状態をリセット

＜形　　式＞

```
DINT            Dsm_CdsysResetWOpenRb(Buffname)
VAR_INPUT
    Buffname:           STRING(31);
END_VAR
```

＜機能説明＞

Dsm_CdsysResetWOpenRb は、指定したデータ共有リングバッファの読み書きオープン状態をリセットする関数です。読み書きオープンしていた処理が異常終了し、読み書きオープンができなくなった場合に使用します。

Buffname　：読み書きオープン状態をリセットするリングバッファの名称を指定してください。

＜診　　断＞

Dsm_CdsysResetWOpenRb は、処理が正常終了すると 0 を返し、異常終了すると負の値を返します。

No.	エラーコード
1	DSM_W_INIT
2	DSM_W_STOP
3	DSM_E_MGROPEN
4	DSM_E_RBNAME
5	DSM_E_WRONGVAL
6	DSM_E_SYSERROR

＜注意事項＞

・ API の誤動作の原因になるため、正常に動作している状態では使用しないでください。

6.2. CODESYS®側 API エラーコード

API の関数が返すエラーコードの一覧を表 6 に示します。

表 6　CODESYS®側 API エラーコード一覧

No.	エラーコード	値	内容	対処方法
1	DSM_W_INIT	-1	デーモンによる初期化が終わっていません。	プログラムの起動順序を見直し、リトライしてください。
2	DSM_E_ARG	-2	引数の設定が異常です。	引数を確認し、プログラムを見直してください。プログラムの修正を実施の上、再立ち上げしてください。
3	DSM_E_NOWRITE	-5	リングバッファに書き込む権限を持っていません。	読み込み用でオープンしたリングバッファに書き込みをしていないか確認してください。
4	DSM_W_STOP	-6	デーモンに停止要求が出ているか、デーモンが停止中です。	入れ替えなどのためにデーモンを停止する要求が出ているか、デーモンが停止中です。処理を停止し、リングバッファをクローズしてください。
5	DSM_E_RBMALLOC	-7	メモリの割り当て失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。
6	DSM_E_MGROPEN	-8	バッファ管理テーブルのオープン失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。解決しない場合は、logsave コマンドで RAS 情報を収集して、製品責任元に連絡してください。
7	DSM_E_RBOPEN	-9	リングバッファのオープン失敗による異常です。	空きメモリが確保されているか確認し、空きメモリがない場合は確保してください。解決しない場合は、logsave コマンドで RAS 情報を収集して、製品責任元に連絡してください。
8	DSM_E_RBNAME	-10	リングバッファ名称の取得失敗による異常です。	取得対象とするリングバッファ名称が異なっていないか、引数を確認してください。

9	DSM_E_OVERSIZE	-11	書き込みサイズが、0 または1ブロックサイズより大きいサイズで指定されています。	引数に設定しているサイズを確認してください。
10	DSM_E_OVERWRITE	-13	データの読み込み中に上書きされています。	リングバッファの最大ブロック数やブロックサイズ または データの読み書きタイミングを見直してください。
11	DSM_E_READONLY	-14	読み込み専用のリングバッファを読み書き用にオープンしようとしています。	リングバッファの書込可能プログラム種別が「CODESYS」以外になっていないか、定義を見直してください。
12	DSM_E_UNMATCHHASH	-15	ハッシュコードが不一致です。	開発ツールで構造体を変更した後に、構造体を使用するプログラムが再コンパイルされているか確認してください。また、変更した後にデーモンを再起動しているか確認してください。
13	DSM_E_WRONGVAL	-16	管理テーブル、リングバッファ管理情報の値が不正です。	開発ツールで生成した ini ファイルの値に、仕様外の値がないか確認してください。
14	DSM_E_SYSERROR	-17	システムエラーが発生しました。	タスク管理等の OS の基本機能のエラーですので、OS が正常に動作しているか確認してください。
15	DSM_E_SEMERROR	-18	CODESYS®のセマフォの獲得解放にかかわるエラーが発生しました。	複数タスクが、読み書き属性でリングバッファをオープンしていないか確認してください。

第7章 制限事項

RT-DSM では、処理性能を優先しているためいくつかの制限事項があります。制限事項の一覧を表 7 に示します。

表 7　制限事項一覧

No.	項目	内容
1	リングバッファアクセス可能対象	書き込みは、一つのリングバッファに対して一つのスレッド(CODESYS®の場合はタスク)のみアクセスが可能です。読み込みは、一つのリングバッファに対して複数のスレッドのアクセスが可能です。 同一リングバッファハンドルを複数のスレッドから同時に使用できません。スレッド毎にオープンを実施してリングバッファハンドルを用意してください。同一リングバッファハンドルに対して複数のスレッドから同時に使用した場合、プログラムエラーや API のエラーが発生します。
2	読み込み処理停止、または遅延によるデータ上書き	書き込み処理が周期実行中に、読み込み処理が停止、または遅延すると、リングバッファ内の時系列の古いデータから順に新しいデータで上書きされる場合があります。
3	書き込み処理停止、または遅延による同一データ読み込み	書き込み処理が停止、または遅延すると、最新データ読み込み処理が同一データを読み込む場合があります。 シーケンス番号などを使用する側でデータに埋め込み、同一データかどうかを判断する必要があります。
4	ブロック数	ブロック数は、メモリアクセスの効率化から、2 のべき乗に切り上げで調整されます。例えば、「10」を設定した場合、切り上げで「16」に調整されます。
5	1 ブロックのサイズ	1 ブロックのサイズは、メモリアクセスの効率化から、8 バイト境界に合わせて切り上げで調整されます。例えば、「20byte」を設定した場合、切り上げで「24byte」に調整されます。
6	エンディアン変換	リングバッファに対して書き込むデータ群、および読み込むデータ群に対して、エンディアン変換は行いません。
7	パディングの追加・削除	リングバッファに対して書き込むデータ群、および読込むデータ群に対してパディングの追加・削除によるアライン補正は行いません。
8	読み込みデータの構成	読み込みデータは、読み込み処理の性能確保のため、書き込みデータのサイズに依らず、1 ブロックのサイズ分を読み込みます。書き込みデータの指定サイズ以降の領域は、書込対象外であるため不定値のままとなります。 複数のブロックを読み込む場合は、1 ブロックのサイズ×読み込み個数分のサイズを読み込みます。

9	リングバッファのアクセスタイミング	リングバッファは Windows®のサービス起動時に作成します。このため、Windows®の起動中にアクセスした場合は失敗することがあります。その場合は、アクセスに成功するまでリトライしてください。
10	予約語について	関数名、変数名、およびリソース名のプレフィックスに以下の名称を使用しないでください。 ・「dsm_」 ・「DSM_」 ・「Dsm_」 API 内部で使用している名称と重複して、コンパイルエラーになる可能性があります。
11	書き込みデータサイズ	書き込むデータのサイズは、使用するリングバッファのブロックサイズ以下を指定してください。書き込むデータのサイズが大きい場合、プログラムエラーやデータ破壊が発生します。
12	読み込みデータエリアサイズ	データの格納先のエリアサイズは、「使用するリングバッファのブロックサイズ×読み込むデータの個数」以上を用意してください。エリアサイズが不足している場合、プログラムエラーやデータ破壊が発生します。
13	未読データ読み込み	未読データ読み込み処理の未読管理は、書き込み毎にブロックにシーケンス番号を割り当て、読み込みは、前回読み込んだシーケンス番号を覚えて、そのシーケンス番号以降のブロックを読み込む仕組みとしています。 シーケンス番号は有限(1～2,147,483,648)のため、最大値を超える場合は 1 に戻ります。 このため、一度読み込み後に、しばらく書き込みのみ行い、未読のブロック数がシーケンス番号の最大数を超えると、同一のシーケンス番号が出現してしまい、再度読み込もうとした際に正しく読み込めなくなる場合があります。 この場合、読み込み処理でリングバッファを一旦クローズし、再度オープンすることで正しく読み込みができるようになります。

第8章 開発ツールとデータ定義ファイル

RT-DSM を使い始める際、最初に必要になるのは「データ定義ファイル」の作成です。データ定義ファイルには、ユーザーが使用するリングバッファのブロック構造などを記載します。そのデータ定義ファイル作成した後、「開発ツール」を用いて以下の情報に変換します。

・リングバッファ定義ファイル
・Windows®側の C 言語プログラム用ヘッダファイル
・CODESYS®側の IEC プログラム用変数定義ファイル

変換の際、構造体にパディング情報も追加して、構造体の実サイズ(=ブロックサイズ)を自動計算すると共に、Windows®アプリケーションと CODESYS®アプリケーションで構造体内のデータのオフセットが合うようにします。また、データ構造更新時にアプリへの定義反映漏れを防ぐため、上記ファイルにデータ定義を基にしたハッシュ値を出力し、Dsm_WinOpenRb 関数、Dsm_CdsysOpneRb 関数で整合性チェックを行います。

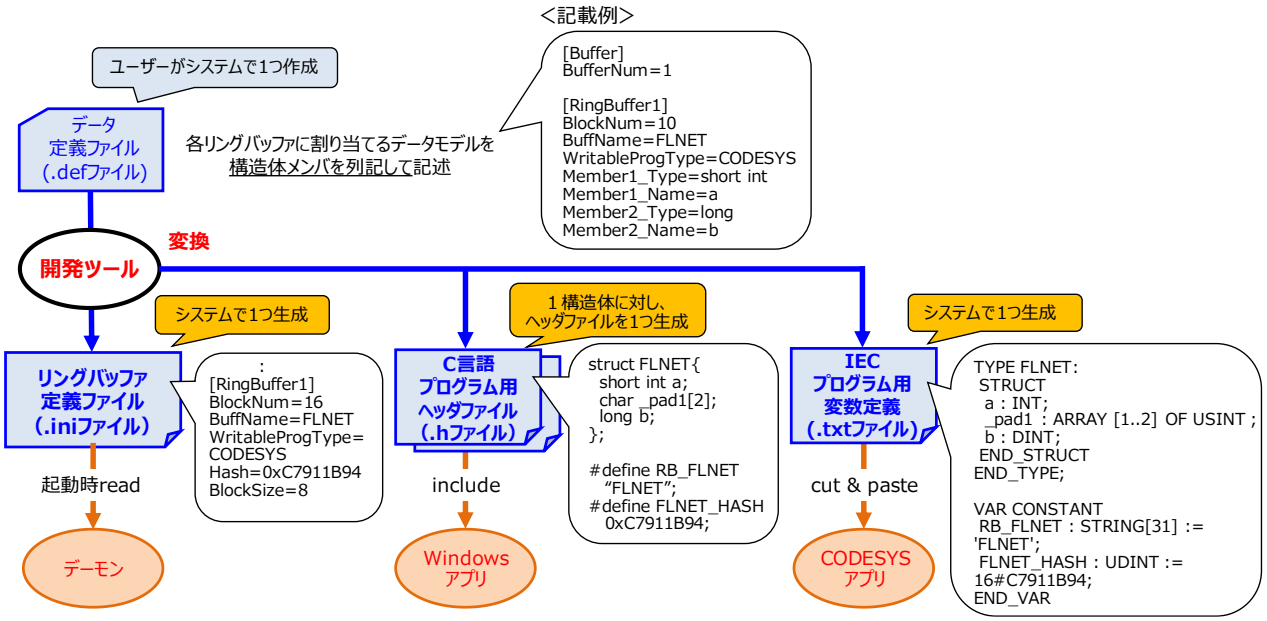


図 17　開発ツールの概要

8.1. リファレンス

開発ツールの使い方を以下に示します。

defconv [入力ファイル名] [出力ファイルのパス]

[入力ファイル名]　　：データ定義ファイル(.def)の格納パスとファイル名を指定します。

[出力ファイルのパス]　：defconv が出力するファイルの格納先を指定します。

指定されたパスには、生成した次のファイルを格納します。

- リングバッファ定義ファイル(.ini)
- C 言語プログラム用ヘッダファイル(.h)
- IEC プログラム用変数定義ファイル(.txt)

※1 パラメータが省略された場合、プログラムのバージョンとヘルプを表示します。

※2 出力パスを省略した場合、データ定義ファイルが存在するパスが出力先になります。

開発ツールが表示するエラーコードは次の通りです。

表 8　defconv コマンドエラーコード一覧

No.	エラーコード	値	内容	対処方法
1	システムエラー	-1	OS の基本機能が動作していません。	Windows®の動作環境に問題がないか確認してください。
2	入力ファイルエラー	-2	データ定義ファイルが開けません。	指定した[入力ファイル名]に、データ定義ファイルが格納されているか確認してください。
3	出力ファイルエラー	-3	生成したファイルが指定先に書き込めません。	指定した[出力ファイルのパス]のファイルが書き込める状態か、あるいは、容量に空きがあるか確認してください。
4	定義誤り	-4	データ定義ファイルの定義内容が誤っているか、不足しています。	データ定義ファイルの定義内容に誤りがないか確認してください。
5	設定値エラー	-5	データ定義ファイルに設定した値が許されない値です。	データ定義ファイルの定義値が仕様範囲内か確認してください。
6	データ型エラー	-6	指定されたデータ型がサポートされていません。	データ定義ファイルの構造体メンバで指定したデータ型が仕様で規定された型か確認してください。
7	パラメータエラー	-7	パラメータが不正です。	コマンドラインパラメータの指定内容を確認してください。

8.2. データ定義ファイル定義項目

以下にデータ定義ファイルの設定項目一覧を示します。

表 9　データ定義ファイル設定項目一覧

No.	セクション名	項目名(KEY)	説明	初期値	範囲	要否
1	Buffer	BufferNum	リングバッファの数	1	1～256	必須
2	RingBuffer*N* *(N=1-256)* (*1) (*2)	BlockNum	リングバッファのブロック数	128	2～32768	必須
3		BuffName	バッファ名称	1	英数字、"_"、"-" 31 文字まで	必須
4		WritableProg Type	書込可能プログラム種別	Windows	Windows、 CODESYS	必須
5		Member*M*_Type *(M=1-2048)*	構造体の M 番目メンバの型	NULL	表 10 規定のキーワード	必須
6		Member*M*_Name *(M=1-2048)*	構造体の M 番目のメンバの名称	NULL	英数字、"_"、"[]" 配列は 2 次元まで。 配列数は数字、または DefineX_Name で定義した文字列。	必須
7		Define*X*_Name *(X=1-4096)*	Menber*M*_Name の配列数として指定可能な define 定義文字列	NULL	英数字、"_" ただし、数字のみは不可	任意
8		Define*X*_Value *(X=1-4096)*	Define*X*_Name に対応する数値	NULL	正の整数	任意

(*1) リングバッファの数だけ定義します。

(*2) 構造体のサイズは、100KB 以内にする必要があります。

8.3. 構造体メンバとして記述できるデータ型

データ定義ファイルで使用可能なデータ型と、それに対し、開発ツールが出力する C 言語プログラム用ヘッダファイルと、IEC プログラム用変数定義ファイルで使用されるデータ型の対応関係を表 10 に示します。

表 10　使用可能なデータ型一覧

No.	データ定義ファイル	C 言語プログラム用 ヘッダファイル	IEC プログラム用 変数定義ファイル	備考
1	unsigned char	unsigned char	USINT	符号なし 1byte 整数型
2	char	char	SINT	符号つき 1byte 整数型
3	unsigned short int	unsigned short int	UINT	符号なし 2byte 整数型
4	short int	short int	INT	符号つき 2byte 整数型
5	unsigned int	unsigned long	UDINT	符号なし 4byte 整数型
6	int	long	DINT	符号つき 4byte 整数型
7	unsigned long	unsigned long	UDINT	符号なし 4byte 整数型
8	long	long	DINT	符号つき 4byte 整数型
9	unsigned long long	unsigned long long	ULINT	符号なし 8byte 整数型
10	long long	long long	LINT	符号つき 8byte 整数型
11	float	float	REAL	単精度浮動小数点型(32bit)
12	double	double	LREAL	倍精度浮動小数点型(64bit)
13	string	char	STRING(*1)	文字列型(配列のみ)

(*1) 文字型配列に関しては、IEC プログラムでは文字列のサイトに EOT が 1 バイト追加されるため、1 文字少ない配列の文字型に変換します。

8.4. C 言語プログラム用ヘッダファイル出力仕様

C 言語プログラム用ヘッダファイルは、データ定義ファイルを開発ツールに読み込ませることで自動生成されます。データ定義ファイルに定義された、複数の RingBuffer それぞれに 1 つのヘッダファイルを生成します。

生成されるヘッダファイルのファイル名は、”RingBuffer_”+BuffName+”.h”とします。

(例) BuffName=“FLNET”であれば、ファイル名は“RingBuffer_FLNET.h“

ヘッダファイルには、次のデータが定義されます。

(1) RingBuffer に割り当てて定義される C 言語形式の構造体

- 構造体の名称は、BuffName になります。
- 構造体には M の値が若い順に、メンバ名が MemberM_Name、その型が MemberM_Type の構造体メンバを生成します。
- MemberM_Name において、メンバ名に[(配列数)] または [(配列数 1)][(配列数 2)]で配列を記述すると、構造体メンバとして MemberM_Type で指定された型の配列を生成します。

(2) データ定義ファイルで定義されたリングバッファ名

・リングバッファ名は、“RB_”＋“(リングバッファ名)”の定数として定義されます。

(3) データ定義ファイルで定義されたリングバッファのハッシュ値

・リングバッファのハッシュ値は、“(リングバッファ名)”＋“_HASH”の定数として定義されます。

8.5. IEC プログラム用変数定義ファイルの出力仕様

IEC プログラム用変数定義ファイルは、データ定義ファイルを開発ツールに読み込ませることで自動生成されます。データ定義ファイルに定義された、複数の RingBuffer に対応する構造体を記述した変数定義ファイルを生成します。

生成される IEC プログラム用変数定義ファイルのファイル名は、入力ファイル名を継承します。

(例) データ定義ファイルが sys1.def であれば、IEC プログラム用変数定義ファイル名は sys1.txt

IEC プログラム用変数定義ファイルには、次のデータが定義されます。

(1) IEC 61131-3 規格形式の構造体定義

・構造体の名称は、BuffName になります。

・M の値が若い順に、メンバ名が MemberM_Name、その型が MemberM_Type の構造体メンバを生成します。

・MemberM_Name において、メンバ名に[(配列数)] または [(配列数 1)][(配列数 2)]で配列を記述すると、構造体メンバとして MemberM_Type で指定された型の配列を生成します。

(2) データ定義ファイルで定義されたリングバッファ名

・リングバッファ名は、“RB_”＋“(リングバッファ名)”の定数として定義されます。

(3) データ定義ファイルで定義されたリングバッファのハッシュ値

・リングバッファのハッシュ値は、“(リングバッファ名)”＋“_HASH”の定数として定義されます。

8.6. データ定義ファイルサンプル

＜記載例＞

```
;Caution!! Don't delete this message.
[Buffer]
BufferNum=2

[RingBuffer1]
BlockNum=10 （←任意の整数値を記述可能）
BuffName=FLNET
WritableProgType=CODESYS
Member1_Type=int
Member1_Name=a
Member2_Type=long
Member2_Name=b
（←リングバッファに割り当てる構造体メンバを列記）

[RingBuffer2]
BlockNum=32
BuffName=LOGDATA
WritableProgType=Windows
Define1_Name=MESS_LEN
Define1_Value=32
（←リングバッファの配列数用の定数を定義可能）
Member1_Type=int
Member1_Name=c
Member2_Type=long long
Member2_Name=d[4] （←構造体メンバに配列も記述可能）
Member3_Type=char
Member3_Name=e[100][MESS_LEN]（←2 次元配列も記述可能、配列数の定数定義も利用可能）
```

第9章 コマンドリファレンス

RT-DSM が提供する各種コマンドの一覧を示します。

表 11　コマンドリファレンス一覧

No.	コマンド名	説明	参照項	備考(*1)
1	dsmconf	データ共有リングバッファの定義情報を表示	9.(1)	○
2	dsmconfchk	データ共有リングバッファの定義ファイルの文法チェック	9.(2)	●
3	dsmresetwop	データ共有リングバッファの読み書きオープン状態をリセット	9.(3)	○

(*1) コマンド実行時、RT-DSM の動作状態により結果が以下となります。

○：RT-DSM が停止していた場合、エラーメッセージを出力します。

●：RT-DSM が停止していても、エラーになりません。

＜各コマンドの実行権限について＞

上記コマンドを実行する場合、Administrator 権限を持つアカウントでログオンしてください。

Administrator 権限以外でコマンドを実行した場合、下記のエラーメッセージが表示されます。

表 12　コマンド実行権限エラーメッセージ

No.	エラーメッセージ	説明
1	This command needs to be executed as Administrator.	Administrator で実行してください。

（1）dsmconf コマンド

＜名　　前＞　dsmconf － データ共有リングバッファの定義情報を表示

＜形　　式＞　dsmconf

＜機能説明＞

dsmconf コマンドは、データ共有リングバッファの定義情報を表示します。

＜診断＞

操作が正常に終了した場合、終了コードは 0 となります。操作が正常に完了しなかった場合、終了コードは 0 以外の値となります。

＜表示フォーマット＞

```
# dsmconf <Enter>
[Buffer]
Number of buffers                   2
Prefix of shared memory name    Global¥Hitachi_Windows_CODESYS_shared_buffer

[Ring buffer]
[Buffer name]           [Number of blocks]  [Block size]  [Writable program type]
 FLNET                      16                  8            CODESYS
 LOGDATA                    32                  3240         Windows
```

表示項目について、以下に詳細を示します。

表 13　dsmconf 表示項目

No.	表示項目		説明
1	Buffer	Number of buffer	データ共有リングバッファの数
2		Prefix of shared memory name	データ共有リングバッファの 共有メモリ名プレフィックス
3	Ring Buffer	Buffer name	バッファ名称
4		Number of blocks	ブロック数
5		Block size	1 ブロックのサイズ
6		Writable program type	書込可能プログラム種別

表 14　dsmconf コマンドエラー一覧

No.	エラーメッセージ	説明	対処方法
1	HX-DSM Manager is not running.	データ共有リングバッファ管理サービスが起動されていません。	データ共有リングバッファ管理サービスを起動してからコマンドを実行してください。
2	DSM_DIR is not setting.	環境変数[DSM_DIR]が設定されていません。	環境変数が変更されています。 再インストールしてください。
3	[ファイルパス名] is not found.	[ファイルパス名]のファイルが見つかりません。	動作に必要なファイルが見つかりません。 再インストールしてください。
4	Internal error. (詳細情報)	内部処理で異常が発生しました。	① 起動中の空きメモリが充分に確保されているか確認してください。空きメモリがない場合は確保してください。 ② ①で解決しない場合は、障害情報を収集の上、弊社のサポートへ問合せください。

（2）dsmconfchk コマンド

＜名　　前＞　dsmconfchk － リングバッファ定義ファイルの文法チェック

＜形　　式＞　dsmconfchk

＜機能説明＞

dsmconfchk コマンドは、リングバッファ定義ファイルの文法チェックを行います。

＜表示フォーマット＞

```
# dsmconfchk <Enter>
HX-DSM Configuration checker start.....
（異常があった場合、診断に示す内容を表示します。）
HX-DSM Configuration checker end.....
```

＜診　　断＞

定義チェックが正常に終了した場合、終了コードは 0 となります。定義チェックが正常に完了しなかった場合、終了コードは 0 以外の値となります。

＜注意事項＞

定義の文法に誤りがなければ、次回データ共有リングバッファ管理サービス立ち上げ時に新しい定義内容を反映します。

表 15　dsmconfchk コマンドエラー一覧

No.	メッセージ	内容	対処方法
1	The configuration file is not found. ([path])	リングバッファ定義ファイルが見つかりません。 [path]:リングバッファ定義ファイルのファイルパス	リングバッファ定義ファイル作成後、再実行してください。
2	A mandatory-key is not defined. (section=[section],key=[key])	必須定義項目が定義されていません。 [section]:セクション名 [key]:項目名	必須定義項目を定義して、再実行してください。
3	A configured syntax was is wrong.(section=[section],key=[key],val=[val])	値の定義に誤りがあります。 [section]:セクション名 [key]:項目名 [val]:値	数値の範囲外、文字数の最大長超過、禁則文字の使用がないか、値の定義を見直して修正後に、再実行してください。
4	DSM_DIR is not setting.	環境変数[DSM_DIR]が設定されていません。	環境変数が変更されています。 再インストールしてください。
5	DSM_DIR directory is not found. ([path])	環境変数[DSM_DIR]に設定されているパスが見つかりません。 [path]:環境変数[DSM_DIR]に設定されているパス名	環境変数が変更されています。 再インストールしてください。
6	Not enough number of 'RingBuffer[n]' definition.(BufferNum=[num], Number of 'RingBuffer[n]' definition=[val])	リングバッファ数の定義に対して'RingBuffer[n]'セクションの定義が不足しています。 [num]:リングバッファ数 [val]:'RingBuffer[n]'セクションの定義数	リングバッファ数、または'RingBuffer[n]'セクションの定義数を見直して修正後に、再実行してください。
7	[ファイルパス名] is not found.	[ファイルパス名]のファイルが見つかりません。	動作に必要なファイルが見つかりません。 再インストールしてください。
8	BuffName is multi-defined. (BuffName=[buffname], section=[section])	バッファ名称が重複しています。 [buffname]:バッファ名称 [section]:セクション名	バッファ名称が重複しないように修正してください。

（3）dsmresetwop コマンド

＜名　　前＞　　dsmresetwop － データ共有リングバッファの読み書きオープン状態をリセット

＜形　　式＞　　dsmresetwop -all | -n buffname

＜機能説明＞

dsmresetwop コマンドは、指定したデータ共有リングバッファの読み書きオープン状態をリセットします。読み書きオープンしていた処理が異常終了し、読み書きオープンができなくなった場合に使用します。指定可能なオプションを以下に示します。

-all	：全てのデータ共有リングバッファの読み書きオープン状態をリセットします。
-n buffname	：buffname に指定したバッファ名称のデータ共有リングバッファの読み書きオープン状態をリセットします。

＜表示フォーマット＞

```
# dsmresetwop -all <Enter>
Reset OK.
```

＜診　　断＞

操作が正常に終了した場合、終了コードは 0 となります。操作が正常に完了しなかった場合、終了コードは 0 以外の値となります。

＜注意事項＞

API の誤動作の原因になるため、正常に動作している状態では使用しないでください。

表 16　dsmresetwop コマンドエラー一覧

No.	エラーメッセージ	説明	対処方法
1	The parameter is incorrect. Usage: dsmresetwop -all | -n buffname	コマンドに無効なオプションを指定しています。	オプションを見直してください。
2	HX-DSM Manager is not running.	データ共有リングバッファ管理サービスが起動されていません。	データ共有リングバッファ管理サービスを起動してからコマンドを実行してください。
3	Buffer name is not found.([バッファ名称])	指定されたバッファ名称は見つかりません。	バッファ名称を見直してください。
4	DSM_DIR is not setting.	環境変数[DSM_DIR]が設定されていません。	環境変数が変更されています。 再インストールしてください。
5	[ファイルパス名] is not found.	[ファイルパス名]のファイルが見つかりません。	動作に必要なファイルが見つかりません。 再インストールしてください。
6	Internal error. (詳細情報)	内部処理で異常が発生しました。	① 起動中の空きメモリが充分に確保されているか確認してください。空きメモリがない場合は確保してください。 ② ①で解決しない場合は、障害情報を収集の上、弊社のサポートへ問合せください。

第10章 サンプルプログラム

RT-DSM を使用したサンプルプログラムを以下に示します。

＜概要＞

CODESYS®アプリケーションから、一定周期でカウントアップした値（ULINT 型）をデータ共有リングバッファに書き込みます。その値を Windows®上のアプリケーション（コマンド）から Dsm_WinMReadRb 関数を用いて未読データ（最大 1024 個）を読み込み、内容表示するサンプルです。

※ Dsm_WinMReadRb 関数によるデータの読み込みと内容表示は、2 回実施します。初回は最大 1024 個の未読データを読み込んで表示し、1 秒後に 2 回目を実施するため、その間に書き込まれた未読データを表示することになります。

＜構成＞

サンプルプログラムの構成を以下に示します。

No.	項目	説明
1	buffer.def	サンプルプログラム用データ定義ファイル
2	buffer.ini	dsmconf コマンドに buffer.def を読み込ませて自動生成されるリングバッファ定義ファイル
3	buffer.txt	No.2 と併せて生成される IEC プログラム用変数定義ファイル
4	RingBuffer_sample_data.h	No.2 と併せて生成される C 言語プログラム用ヘッダファイル
5	CODESYS_sample	CODESYS®側からデータ共有リングバッファにデータを書き込むサンプルプログラム(ST 言語)
6	Windows_sample	CODESYS®側から共有データ共有リングバッファに書き込まれたデータを読み込んで表示する C 言語サンプルプログラム

＜詳細内容＞

サンプルプログラムの内容を示します。

【buffer.def】

```
;Caution!! Don't delete this message.
[Buffer]
BufferNum=1

[RingBuffer1]
BlockNum=8192
BuffName=sample_data
WritableProgType=CODESYS
Member1_Type=unsigned long long
Member1_Name=count
```

【buffer.ini（buffer.def を開発ツールに読み込ませて自動生成）】

```
;Caution!! Don't delete this message.

[Buffer]
BufferNum=1
NamePrefix=Global¥Hitachi_Windows_CODESYS_shared_buffer
[RingBuffer1]
BlockNum=8192
BuffName=sample_data
WritableProgType=CODESYS
Hash=0xD409FE2B
BlockSize=8
```

【buffer.txt（buffer.def を開発ツールに読み込ませて自動生成）】

```
// Structure definition for ring buffer : sample_data

TYPE sample_data:
STRUCT
        count : ULINT;
END_STRUCT
END_TYPE;


VAR CONSTANT

// Name definition for ring buffer :sample_data
        RB_sample_data : STRING[31] := 'sample_data' ;

// Hash value definition for ring buffer :sample_data
        sample_data_HASH : UDINT := 16#D409FE2B ;

END_VAR
```

CODESYS®開発環境で
“DUT”として追加。

CODESYS®開発環境で
“PLC_PRG(PRG)”の
変数宣言部の最後に追加。

【RingBuffer_sample_data.h（buffer.def を開発ツールに読み込ませて自動生成）】

```
// Structure definition for ring buffer : sample_data

struct sample_data{
        unsigned long long   count;
};


// Name definition for ring buffer : sample_data

#define    RB_sample_data      "sample_data"


// Hash value definition for ring buffer : sample_data

#define    sample_data_HASH 0xD409FE2B
```

【CODESYS_sample】

【変数宣言部】

```
PROGRAM PLC_PRG
VAR
        hBuff : DSM_CD_RB_HANDLE := 0;
        blksz: UDINT;
        blkcase: UDINT;

        input_data: sample_data;
        input_size: UDINT := SIZEOF(INPUT_DATA);

        result : DINT ;
        RB_OPCL_Flag : BOOL := TRUE;
END_VAR

VAR CONSTANT

// Name definition for ring buffer :sample_data
        RB_sample_data : STRING[31] := 'sample_data' ;

// Hash value definition for ring buffer :sample_data
        sample_data_HASH : UDINT := 16#D409FE2B ;

END_VAR
```

【処理部】

```
// Ring Buffer Open(Access Type : Read/Write)
IF hBuff = 0 AND RB_OPCL_Flag = TRUE THEN
        result := Dsm_CdsysOpenRb(RB_sample_data, 1, ADR(hBuff), ADR(blksz), ADR(blkcase) ,
                sample_data_HASH);
END_IF;

// Write Data to Ring Buffer
IF hBuff <> 0 AND RB_OPCL_Flag = TRUE THEN

        // Input data count up
        input_data.count := input_data.count + 1;

        result := Dsm_CdsysWriteRb (hBuff , ADR(input_data), input_size);
END_IF;

// Ring Buffer Close
IF hBuff <> 0 AND RB_OPCL_Flag = FALSE THEN
        result := Dsm_CdsysCloseRb(hBuff);

        IF result = 0 THEN
                hBuff := 0;
        END_IF;
END_IF;
```

【Windows_sample】

```
#include <stdio.h>

#include "dsm_rb.h"
#include "RingBuffer_sample_data.h"

int main()
{
	DSM_RB_HANDLE		hBuff;
	unsigned int		blksz, blkcase;

	int iReadBlockCnt = 1024;

	unsigned int hash = sample_data_HASH;
	struct sample_data output[10240];

	unsigned int uiOnSkip = 0;
	unsigned char* ucStr;

	int result = 0, iDataCnt = 0, i = 0;

	printf("¥n¥n<*** Dsm_WinMReadRb sample Start ***>¥n¥n");

	// Ring Buffer Open(Access Type : Read)
	result = Dsm_WinOpenRb(RB_sample_data, 0, &hBuff, &blksz, &blkcase, hash);

	if (result < 0) {
		printf("Dsm_WinOpenRb Error(%d).¥n", result);
		return -1;
	}
	else {
		printf(" block size :%d¥n", blksz);
		printf(" block case :%d¥n", blkcase);
	}

	ucStr = (unsigned char*)malloc(blkcase * blksz);

	memset(ucStr, 0x00, blkcase * blksz);

	// MRead Data from Ring Buffer
	for (i = 0; i < 2; i++) {

		Sleep(1000);
		printf("¥n Number of executions - %d¥n", i + 1);

		result = Dsm_WinMReadRb(hBuff, iReadBlockCnt, &ucStr[0], &uiOnSkip);

		memcpy(&output, &ucStr[0], blksz * iReadBlockCnt);

		if (result >= 0) {
			for (iDataCnt = 0; iDataCnt < result; iDataCnt++) {
				printf(" [%4d] Count Data : %llu¥n", iDataCnt, output[iDataCnt].count);
			}
		}
	}

	free(ucStr);

	if (result >= 0) {
		result = Dsm_WinCloseRb(hBuff);
	}

	printf("¥n<*** Dsm_WinMReadRb sample end	***>¥n");

	return 0;
}
```

＜実行手順＞

サンプルプログラムの実行手順を以下に示します。なお、CODESYS®開発環境、CODESYS®リアルタイム実行環境、C 言語ビルド環境は別 PC であるものとして記載します。

① CODESYS®リアルタイム実行環境にて、リングバッファ定義ファイル（buffer.ini）を「C:¥Program Files¥HX-DSM¥conf」に格納します。また、「Start PLC」を実施し、PLC プログラムをダウンロードできるようにしておきます。

※ その他の ini ファイルを格納しないようご注意ください。

② CODESYS®開発環境にて、新規プロジェクトを作成します（テンプレートは、“標準プロジェクト”）。デバイスの種類はご使用の環境に合わせたものを、開発言語は ST 言語を選択します。

③ 作成した CODESYS®プロジェクトにて、デバイスツリーの“ライブラリマネージャー”に RT-DSM のライブラリ（“CmpDsm”）を追加します。

④ デバイスツリーの“Application”を右クリックし、“オブジェクトの追加”から“DUT”を構造体で追加します。構造体の名称は、「buffer.txt」記載上部の“sample_data”をコピーして使用します。

⑤ 追加した“sample_data”構造体の内容を、「buffer.txt」の”TYPE”から”END_TYPE;”までをコピー・ペーストして置き換えます。

⑥ デバイスツリーの“PLC_PRG(PRG)”を開き、変数宣言部の最後に「buffer.txt」の”VAR CONSTANT”から”END_VAR”までをコピー・ペーストします。

⑦ 「CODESYS_sample」の変数宣言部と処理部の内容を、“PLC_PRG(PRG)”にコピー・ペーストします。

⑧ ビルドした後、CODESYS®リアルタイム実行環境に PLC プログラムをダウンロード（ログイン操作）します。その後、ログアウトします。

⑨ C 言語ビルド環境にて、プロジェクトを作成し、ヘッダファイルのインクルードパスに「RingBuffer_sample_data.h」と「dsm_rb.h」を格納するフォルダのパスを設定します。また、「DsmWlib.lib」をインポートライブラリとして設定します。

⑩ 「Windows_sample」の内容をコピーして、C 言語のコマンドとしてビルドします。

⑪ CODESYS®リアルタイム実行環境にて Windows®を再起動し、リングバッファ定義ファイルの変更を反映します。再起動後、「Start PLC」することでデータ共有リングバッファにカウントアップ値の書き込みを開始します。

⑫ 続けて、コマンドプロンプトを管理者権限で起動し、⑩で作成したコマンドを実行します。CODESYS®側からデータ共有リングバッファに書き込まれたカウントアップ値が取得&表示できているのを確認します。

第11章 組込み CDMS とのデータ連携

RT-DSM は、組込み装置向けデータ管理基盤(NX Context-based Data Management System for Embedded device、以降「組込み CDMS」と略す)とのデータ連携が可能です。第 8 章で説明した開発ツールの機能を拡張した、組込み CDMS 対応版開発ツールで出力した定義ファイルを使用します。

なお、組込み CDMS の使い方や定義ファイルの仕様に関しては、組込み CDMS のマニュアル「NX Context-based Data Management System for Embedded device ユーザーズガイド」をご参照ください。

組込み CDMS 対応版開発ツールでは、第 8 章のリングバッファ定義ファイル、C 言語プログラム用ヘッダファイル、IEC プログラム用変数定義ファイルの 3 種類のファイルに加え、組込み CDMS で使用する次の 5 種類のファイルを出力します。

(1) データ項目紐付け定義ファイル（cdms_emb_mapped.conf）
(2) データ名称定義ファイル（cdms_emb_named.conf）
(3) イベントトリガー定義ファイル(cdms_emb_trigger.conf)
(4) データ用リングバッファ定義ファイル（cdms_emb_dataring.conf）
(5) トリガー用リングバッファ定義ファイル（cdms_emb_triggerring.conf）

データ定義ファイルに新たな KEY 定義を追加し、組込み CDMS 対応版開発ツールに入力すると、上記の(1)～(5)の定義ファイルを出力することができます。

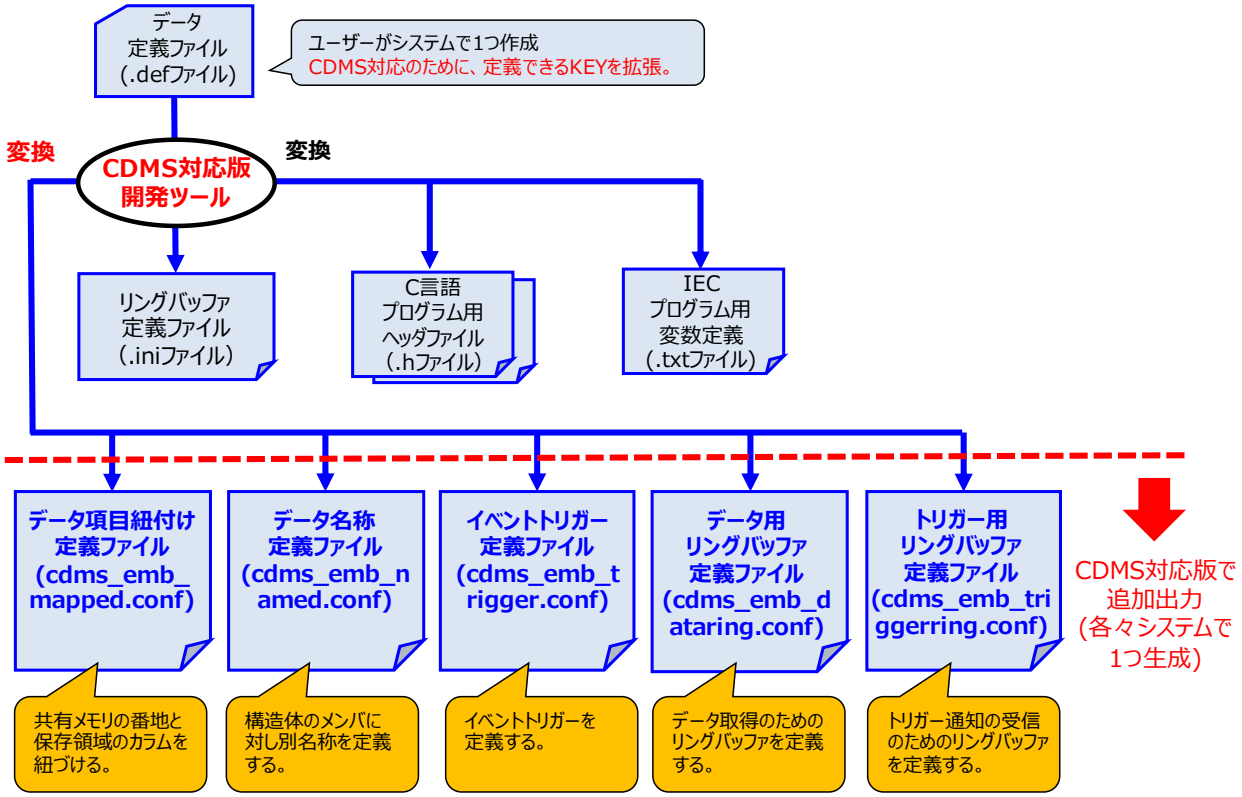


図 18　組込み CDMS 対応版開発ツールの概要

なお、新たに追加した KEY 定義をデータ定義ファイルに記述しない場合、あるいは 組込み CDMS 対応の KEY 定義の不足や不整合がある場合は、第 8 章の開発ツール同じ 3 種類のファイルのみを出力します。

11.1. リファレンス

defconvDB [入力ファイル名] [出力ファイルのパス]

[入力ファイル名]　　：データ定義ファイル(.def)の格納パスとファイル名を指定します。

[出力ファイルのパス]　：defconvDB が出力するファイルの格納先を指定します。

指定されたパスには、生成した次のファイルを格納します。

・リングバッファ定義ファイル(.ini)

・C 言語プログラム用ヘッダファイル(.h)

・IEC プログラム用変数定義ファイル(.txt)

・データ項目紐付け定義ファイル(cdms_emb_mapped.conf)

・データ名称定義ファイル(cdms_emb_named.conf)

・イベントトリガー定義ファイル(cdms_emb_trigger.conf)

・データ用リングバッファ定義ファイル(cdms_emb_dataring.conf)

・トリガー用リングバッファ定義ファイル(cdms_emb_triggerring.conf)

※1 パラメータが省略された場合、プログラムのバージョンとヘルプを表示します。

※2 出力パスを省略した場合、データ定義ファイルが存在するパスが出力先になります。

組込み CDMS 対応版開発ツールがエラー出力で表示するエラーコードは次の通りです。

表 17　defconvDB コマンドエラーコード一覧

No.	エラーコード	値	内容	対処方法
1	システムエラー	-1	OS の基本機能が動作していません。	Windows®の動作環境に問題がないか確認してください。
2	入力ファイルエラー	-2	データ定義ファイルが開けません。	指定した[入力ファイル名]に、データ定義ファイルが格納されているか確認してください。
3	出力ファイルエラー	-3	生成したファイルが指定先に書き込めません。	指定した[出力ファイルのパス]のファイルが書き込める状態か、あるいは、容量に空きがあるか確認してください。
4	定義誤り	-4	データ定義ファイルの定義内容が誤っているか、不足しています。	データ定義ファイルの定義内容に誤りがないか確認してください。
5	設定値エラー	-5	データ定義ファイルに設定した値が許されない値です。	データ定義ファイルの定義値が仕様範囲内か確認してください。
6	データ型エラー	-6	指定されたデータ型がサポートされていません。	データ定義ファイルの構造体メンバで指定したデータ型が仕様で規定された型か確認してください。
7	パラメータエラー	-7	パラメータが不正です。	コマンドラインパラメータの指定内容を確認してください。

11.2. データ定義ファイル定義項目

以下に組込み CDMS 対応版のデータ定義ファイルの設定項目一覧を示します。新たに追加した KEY 定義を**太字**で記載します。

表 18　データ定義ファイル設定項目一覧

No.	セクション名	項目名(KEY)	説明	初期値	範囲	要否
1	Buffer	BufferNum	リングバッファの数	1	1～256	必須
2	RingBuffer*N* *(N=1-256)* (*1)	**DbAttrib**	**組込み CDMS での用途を示す種別(組込み CDMS で使用するもののみに指定)**	**NULL**	**DATA,TRIGGER_STAT,ERR_TRIGGER,EVT_TRIGGER のいずれか**	**任意**
3		BlockNum	リングバッファのブロック数	128	2～32768	必須
4		BuffName	バッファ名称	1	英数字 31 文字まで	必須
5		WritableProg Type	書込可能プログラム種別	Windows	Windows、CODESYS	必須
6		Member*M*_Type *(M=1-2048)*	構造体の M 番目メンバの型	NULL	表 10 規定のキーワード	必須
7		Member*M*_Name *(M=1-2048)*	構造体の M 番目のメンバの名称	NULL	英数字、"_"、"[]" 配列は 2 次元まで。配列数は数字、または Define*X*_Name で定義した文字列。	必須
8		**MemberM_Alias *(システム全体で Max1000 個)***	**構造体メンバのエイリアス定義*(DbAttrib= DATA の時のみ有効)***	**NULL**	**英数字、"_"、"-“ 31 文字まで “,” 区切りで複数記述可能**	**任意**
9		**TargetData**	**イベントトリガーで収集したいデータを、定義済みの Alias で指定 *(DbAttrib=EVT_TRIGGER の時のみ有効)***	**NULL**	**英数字、"_"、"-“ 31 文字まで “,”区切りで最大 16 個記述可能**	**任意**
10		**SampCyc**	**サンプリング周期 *(DbAttrib=EVT_TRIGGER の時のみ有効)***	**NULL**	**msec 単位数値 (1-2147483647)**	**任意**
11		**BeforeTime**	**Before 時間 *(DbAttrib=EVT_TRIGGER の時のみ有効)***	**NULL**	**msec 単位数値 (0-4000)**	**任意**
12		**AfterTime**	**After 時間 *(DbAttrib=EVT_TRIGGER の時のみ有効)***	**NULL**	**msec 単位数値 (0-2147483647)**	**任意**
13		Define*X*_Name *(X=1-4096)*	Menber*M*_Name の配列数として指定可能な define 定義文字列	NULL	英数字、"_" ただし、数字のみは不可	任意
14		Define*X*_Value *(X=1-4096)*	Define*X*_Name に対応する数値	NULL	正の整数	任意

(*1) リングバッファの数だけ定義します。