DSHEng4　装置通信エンジン（GEM+GEM300）

ソフトウェア・パッケージ

システム管理情報定義ファイル

（変数、収集イベント、アラームその他）

コンパイラ説明書

（DshCompile.exe）

２００９年7月

株式会社データマップ

文書番号　DSHEng4-09-30303-00

[取り扱い注意]

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、（株）データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2009.7 月	初版	DSHGEM-LIB の説明書をベースにしています。
2.			
3.			
4.			

目次

１．概要

本説明書は、テキストファイルに定義された情報を DSHEng4 装置通信エンジンが読み取ることができる形式にコンパイルするためのプログラムツールの操作と生成されるファイルなどについて説明します。

DSHGEM-LIB エンジンは、変数を始めとするシステム情報の管理を行います。
ユーザは、システム管理情報を人に判りやすいテキストファイル上にホストとの通信メッセージ仕様書の規定に基づいて管理情報を定義します。（管理情報定義の詳細については、「システム管理情報定義仕様書」を参照してください。）

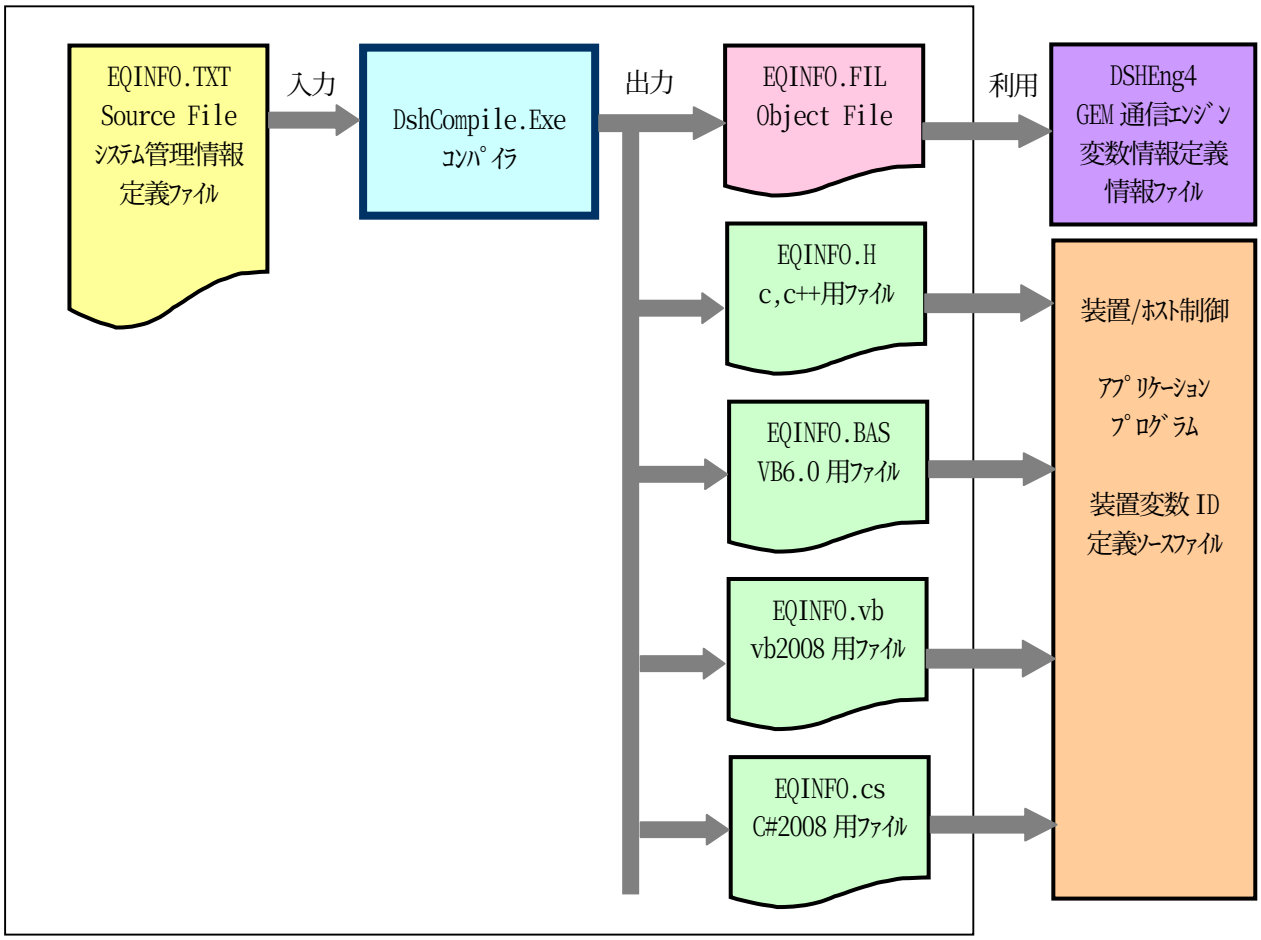
それから、本コンパイラは、ユーザがアプリケーションプログラム作成時に利用することができる管理情報の ID 値の名前を定義したソースファイルも生成してくれます。(c 言語-.H, VB6.0-.BAS、VB2008-.vb, C#2008-.cs ﾌｧｲﾙ)
これらのソースファイルはそのままでもそれぞれの言語によるアプリケーションプログラムに含めて使用できます。

このソースファイルが自動作成されることによって、ユーザは、情報 ID を、マクロ定義された ID の名前を使ってプログラミングすることができ、プログラム作成と保守に掛かる工数を削減することができます。

プログラムツールの名前は **DshCompile.exe** です。

定義ファイル名を EQINFO.TXT を例にコンパイルの流れと生成されるファイルとその利用形態を下図に表します。

コマンド　DshCompile EQINFO.TXT　で実行できます。



２．操作

次のように操作します。

```
DshCompile <定義ﾌｧｲﾙ名>
```

定義ファイルは、システム管理情報定義ファイルです。DshCompile.exe と違う場合は、フルパスで指定してください。

（1）正常にコンパイルできた場合

オブジェクトファイルは、指定された定義ファイル名と同じで、ファイル拡張子が .FIL のファイルが生成されます。また、アプリケーションプログラムで使用できる情報 ID が定義されたソースファイルも生成されます。

```
End of DshCompile Compilation.
  number of error = 0
<DshCompile result >
   source file (in)  = eqinfo.txt
   object file (out) = eqinfo.fil
   .h      file (out) = eqinfo.h
   .bas    file (out) = eqinfo.bas
   .vb     file (out) = eqinfo.vb
   .cs     file (out) = eqinfo.cs
```

情報定義ﾌｧｲﾙ名
ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名
c, C++言語用ﾍﾀﾞｰﾌｧｲﾙ名
VB6 用ﾌｧｲﾙ
VB2008 用ﾌｧｲﾙ
C#2008 用ﾌｧｲﾙ

このオブジェクトファイル名はエンジン起動ファイルの INFO_FILE コマンドを使ってエンジンに与えられます。

（2）コンパイル上にエラーが検出された場合

次のように検出した行番号とエラー行の内容が表示されます。

```
 ! error line-35 "mominal:    "1""
 End of DshCompile Compilation.
   number of error = 1
==> Comile error end
```

（上の例は、 nominal とあるべきところが、 mominal になっていたのでエラーになった。）

エラー検出した場合、オブジェクトファイルは生成されません。
エラー箇所を直して再コンパイルする必要があります。

３．オブジェクトファイルの形式

コンパイルして得られたオブジェクトファイルはテキストファイルで、その形式は以下のようになります。

一般的な形式は、頭の XX:　が情報の種類になります。

（１）装置変数 EC，SV，DVVAL 情報の形式

```
XX:                          // XX は EC，SV or DV です。
NAME<ﾊﾞｲﾄ数><変数名>          // 変数名のﾊﾞｲﾄ数と変数名(文字列)です。
ID  <ID 値>                  // ID 値は 8 桁固定の 16 進表現です。
FORM<ItemCode>               // ﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>              // ﾃﾞｰﾀのｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>         // ﾃﾞｰﾀの最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
UNIT<ﾊﾞｲﾄ数><単位名>         // 単位名のﾊﾞｲﾄ数と単位名(文字列)です。
MIN <ﾊﾞｲﾄ数><最小値>         // 最小値は文字列で表現します。ﾊﾞｲﾄ長と値文字列です。
MAX <ﾊﾞｲﾄ数><最小値>         // 最大値は文字列で表現します。ﾊﾞｲﾄ長と値文字列です。
NOMI<ﾊﾞｲﾄ数><最小値>         // 初期値は文字列で表現します。ﾊﾞｲﾄ長と値文字列です。
LMID<ﾊﾞｲﾄ数><LIMITID 値>     // LimitID 値は文字列で表現します。ﾊﾞｲﾄ長と ID 値文字列です。
LOWE<ﾊﾞｲﾄ数><LowerLimit 値> // LowerLimit 値は文字列で表現します。ﾊﾞｲﾄ長と ID 値文字列です。
UPPE<ﾊﾞｲﾄ数><UpperLimit 値> // LimitID 値は文字列で表現します。ﾊﾞｲﾄ長と ID 値文字列です。
EVNT<ﾊﾞｲﾄ数><CE 名>          // 収集ｲﾍﾞﾝﾄ名のﾊﾞｲﾄ数と収集ｲﾍﾞﾝﾄ名文字列です。
```

（LMID，LOWE，UPPE、EVNT は複数個定義可能。）

（２）収集イベント CE 情報の形式

```
CE:                          // CE(Collection Event)
NAME<ﾊﾞｲﾄ数><変数名>          // CE 名のﾊﾞｲﾄ数と CE 名(文字列)です。
ID  <ID 値>                  // ID 値は 8 桁固定の 16 進表現です。
ENAB<ﾊﾞｲﾄ数><状態値>         // ENABLE 状態のﾊﾞｲﾄ数と状態値文字列
RPNA<ﾊﾞｲﾄ数><RPID 名>        // LINK ﾚﾎﾟｰﾄ ID の名前ﾊﾞｲﾄ長と名前（0 個以上）
```

（３）レポート、 REPORT 情報の形式

```
RP:                          // RP(Report)
NAME<ﾊﾞｲﾄ数><変数名>          // Report 名のﾊﾞｲﾄ数と Report 名(文字列)です。
ID  <ID 値>                  // ID 値は 8 桁固定の 16 進表現です。
VNAM<ﾊﾞｲﾄ数><変数名>         // LINK 変数名の名前ﾊﾞｲﾄ長と名前(0 個以上)
```

（４）アラーム　ALARM 情報の形式

```
AL:                          // AL(Alarm)
NAME<ﾊﾞｲﾄ数><変数名>          // Alarm 名のﾊﾞｲﾄ長と Alarm 名(文字列)です。
ID  <ID 値>                  // ID 値は 8 桁固定の 16 進表現です。
ALCD<ﾊﾞｲﾄ数><ALCD 値>        // ALCD のﾊﾞｲﾄ長と値文字列
ALTX<ﾊﾞｲﾄ数><ALTX 値>        // ALTX のﾊﾞｲﾄ長と値文字列
CEON<ﾊﾞｲﾄ長><CE 名>          // ALARM 発生に LINK された CE の名前のﾊﾞｲﾄ長と CE 名
CEOF<ﾊﾞｲﾄ長><CE 名>          // ALARM 復旧に LINK された CE の名前のﾊﾞｲﾄ長と CE 名
```

（5）プロセスプログラム　PP 情報の形式

PP:	// PP(Process Program)
NAME<ﾊﾞｲﾄ数><変数名>	// PP 名のﾊﾞｲﾄ長と PP 名(文字列)です。
ID　<ｻｲｽﾞ><ID 値>	// PPID 値のﾊﾞｲﾄ長と PPID 値(文字列)です。
FORM<ItemCode>	// PPID のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// PPID のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>	// PPID の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
PPBO<ｻｲｽﾞ><PPBODY 値>	// PPBODY 値のﾊﾞｲﾄ長と PPBODY 値(文字列)です。
FORM<ItemCode>	// PPBODY のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// PPBODY のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>	// PPBODY の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。

（6）書式付プロセスプログラム FPP 情報の形式

FP:	// FP(Formatted Process Program)
NAME<ﾊﾞｲﾄ数><変数名>	// FPP 名のﾊﾞｲﾄ長と FPP 名(文字列)です。
ID　<ｻｲｽﾞ><ID 値>	// FPPID 値のﾊﾞｲﾄ長と FPPID 値(文字列)です。
FORM<ItemCode>	// FPPID のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// FPPID のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>	// FPPID の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MDLN<ﾊﾞｲﾄ数><MDLN 値>	// MDLN のﾊﾞｲﾄ長と値文字列
SOFT<ﾊﾞｲﾄ数><SOFTREV 値>	// SOFTREV のﾊﾞｲﾄ長と値文字列
COUN<ｶｳﾝﾄ値>	// ﾌﾟﾛｾｽｺﾏﾝﾄﾞの数,8 桁固定の 16 進表現です。(>=0)
CCOD<ｻｲｽﾞ><CCODE 値>	// CCODE(Command Code)値のﾊﾞｲﾄ長と CCODE 値(文字列)です。
FORM<ItemCode>	// CCODE 値のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// CCODE 値のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>	// CCODE 値の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
PCOU<ｶｳﾝﾄ値>	// CCODE に対するﾊﾟﾗﾒｰﾀの数,8 桁固定の 16 進表現です。 (>=0)
PPAR<ｻｲｽﾞ><PARAMETER 値>	// PARA(parameter)値のﾊﾞｲﾄ長と PARA 値(文字列)です。
FORM<ItemCode>	// PARA 値のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// PARA 値のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>	// PARA 値の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。

（COUN 分の CCODE が続き、RCOU 分のﾊﾟﾗﾒｰﾀが続きます。）

（7）レシピ RCP 情報の形式

RC:	// RC(RECIPE)
NAME<ﾊﾞｲﾄ数><変数名>	// RCP 名のﾊﾞｲﾄ長と RCP 名(文字列)です。
ID　<ｻｲｽﾞ><ID 値>	// RCPID 値のﾊﾞｲﾄ長と RCPID 値(文字列)です。
PPBO<ﾊﾞｲﾄ数><RCPBODY 値>	// RCPBODY のﾊﾞｲﾄ長と値文字列
COUN<ｶｳﾝﾄ値>	// PARAMETER の数,8 桁固定の 16 進表現です。(>= 0)
PNAM<ｻｲｽﾞ><Parameter 名>	// PARAMETER 名のﾊﾞｲﾄ長と PARAMETER 名(文字列)です。
PVAL<ｻｲｽﾞ><Parameter 値>	// PVAL(ﾊﾟﾗﾒｰﾀ値)ﾊﾞｲﾄ長と PVAL 値(文字列)です。
FORM<ItemCode>	// PVAL 値のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>	// PVAL 値のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。

（COUN 分の PNAM,PVAL,FORM,SIZE が続きます。）

（8）スプール　SPOOL 情報の形式

SP:　　// SP(SPOOL)
NAME<ﾊﾞｲﾄ数><変数名>　　// SPOOL 名のﾊﾞｲﾄ長と SPOOL 名(文字列)です。
STRE<Stream 値>　　// Stream の値,8 桁固定の 16 進表現です。
COUN<ｶｳﾝﾄ値>　　// Function の数,8 桁固定の 16 進表現です。(> 0)
FUNC<Function 値>　　// Function の値,8 桁固定の 16 進表現です。
（COUN 分の FUNC が続きます。）

（9）トレース　TRACE 情報の形式

TR:　　// TR(TRACE)
NAME<ﾊﾞｲﾄ数><変数名>　　// TRACE 名のﾊﾞｲﾄ長と TRACE 名(文字列)です。
ID　<ｻｲｽﾞ><ID 値>　　// TRID(Trace ID)値のﾊﾞｲﾄ長と TRID 値(文字列)です。
FORM<ItemCode>　　// TRID 値のﾃﾞｰﾀﾌｫｰﾏｯﾄ 2 桁固定の 16 進表現です。
SIZE<値のｻｲｽﾞ>　　// TRID 値のｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
MSIZE<値の最大ｻｲｽﾞ>　　// TRID 値の最大ｻｲｽﾞ (=配列ｻｲｽﾞ)8 桁固定の 16 進表現です。
DSPE<ｻｲｽﾞ><DSPER 値>　　// DSPER のﾊﾞｲﾄ長と DSPER 値文字列
TOTS<合計ｻﾝﾌﾟﾙ数>　　// TotSmp の値の 8 桁固定の 16 進表現です。
REPG<ﾚﾎﾟｰﾄｸﾞﾙｰﾌﾟ数>　　// RepGSz の値の 8 桁固定の 16 進表現です。
COUN<ｶｳﾝﾄ値>　　// Trace 対象変数の数,8 桁固定の 16 進表現です。(> 0)
SVNA<ﾊﾞｲﾄ数><変数名>　　// Trace 対象の変数名のﾊﾞｲﾄ長と変数名（文字列）です。
（COUN 分の SVNA が続きます。）

付録-A　システム管理情報関連ファイルのサンプル

簡単な例、定義ファイルの中から CE_ReadyToLoad イベントについて抜粋してサンプルを紹介します。

（1）情報定義ファイル（ソース）

```
def_sv  SV_Clock{
        svid:       0x10000
        format:     A[16]
        }
def_sv  SV_ReadyToLoad{
        svid:       0x10040
        format:     A[0..80]
        nominal:    "Equipment is ready to load a carrier."
        }
def_report  RP_ReadyToLoad{
        rptid:  0x10040
        vname:  SV_Clock
        vname:  SV_ReadyToLoad
        }
def_ce  CE_ReadyToLoad{
        ceid:       0x10040
        enabled:    1
        rptname:    RP_ReadyToLoad
```

（2）オブジェクトファイル

```
SV: NAME00000008SV_ClockID  00010000FORM00000010SIZE00000010MSIZ00000010
SV: NAME0000000eSV_ReadyToLoadID  00010040FORM00000010SIZE00000000MSIZ00000050NOMI00000025
                                                    Equipment is ready to load a carrier.
CE: NAME0000000eCE_ReadyToLoadID  00010040ENAB0000000011RPNA0000000eRP_ReadyToLoad
RP: NAME0000000eRP_ReadyToLoadID  00010040VNAM00000008SV_ClockVNAM0000000eSV_ReadyToLoad
```

（3）プログラムソースファイル

.H ファイル(c, C++用)

```
#define  SV_Clock                65536          // 0x00010000
#define  SV_ReadyToLoad          65600          // 0x00010040
#define  CE_ReadyToLoad          65600          // 0x00010040
#define  RP_LoadPort             65792          // 0x00010100
```

.BAS ファイル(VB6.0用)

```
Public Const SV_Clock =             65536          ' 0x00010000
Public Const SV_ReadyToLoad =       65600          ' 0x00010040
Public Const CE_ReadyToLoad =       65600          ' 0x00010040
Public Const RP_ReadyToLoad =       65600          ' 0x00010040
```

.vb ファイル(VB2008 用)

```
Public Const SV_Clock As Integer        65536       ' 0x00010000
Public Const SV_ReadyToLoad As Integer  65600       ' 0x00010040
Public Const CE_ReadyToLoad As Integer  65600       ' 0x00010040
Public Const RP_LoadPort As Integer     65792       ' 0x00010100
```

.cs ファイル(C#2008 用)

```
public const int  SV_Clock =          65536     // 0x00010000
public const int  SV_ReadyToLoad =    65600     // 0x00010040
public const int  CE_ReadyToLoad =    65600     // 0x00010040
public const int  RP_ReadyToLoad =    65792     // 0x00010100
```