

装置通信制御エンジン (SECS/HSMS)

ソフトウェア・パッケージ

DSHEng3、DSHEng、DSHGemLib 共通

## 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

2012年4月

株式会社データマップ

**[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.	2005. 11 月	初版	
2.	2006. 04. 17	文書番号変更	
3.	2012. 4	DSHEng 3用を共通化	• Compile プログラム名を DshCompile. exe に変えた。 • 出力ファイルに .Net C#, VB を加えた。
4.			

## 目次

1. 概要.....	1
2. 操作.....	2
3. オブジェクトファイルの形式.....	3
付録-A システム管理情報関連ファイルのサンプル.....	6



# 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 DSEng3 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。

DSEng3 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。)

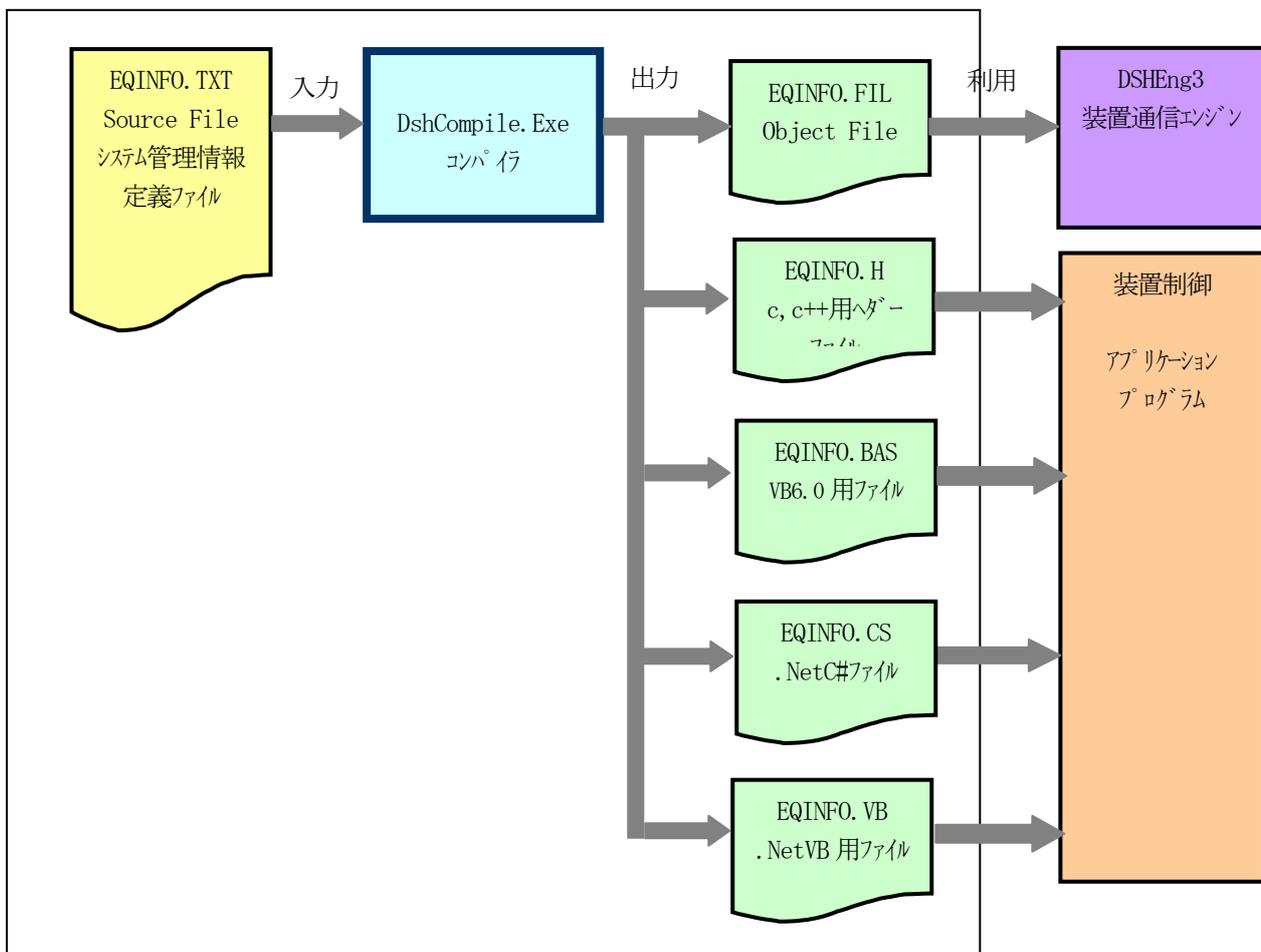
それから、本コンパイラは、ユーザがアプリケーションプログラム作成時に利用することができる管理情報の ID 値の名前をマクロ定義したヘッダファイルも生成してくれます。(c 言語 .H, VB6.0-.BAS, .Net C#, VB ファイル)

このヘッダ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 がマクロ定義されたヘッダファイルも生成されます。

```
End of Eng3conf Compilation.
number of error = 0
<Eng3Conf result >
source file (in) = eqinfo.txt
object file (out) = eqinfo.fil
.h file (out) = eqinfo.h
.bas file (out) = eqinfo.bas
```

情報定義ファイル名  
オブジェクトファイル名  
c, C++言語用ヘッダファイル名  
VB6 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 "nominal: "1""
End of Eng3conf Compilation.
number of error = 1
=> Comile error end
```

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

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<バ 1 数><変数名> // 変数名のバ 1 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> // データフォーマット 2 桁固定の 16 進表現です。
SIZE<値のサイズ> // データのサイズ (=配列サイズ)8 桁固定の 16 進表現です。
MSIZE<値の最大サイズ> // データの最大サイズ (=配列サイズ)8 桁固定の 16 進表現です。
UNIT<バ 1 数><単位名> // 単位名のバ 1 数と単位名(文字列)です。
MIN <バ 1 数><最小値> // 最小値は文字列で表現します。バ 1 長と値文字列です。
MAX <バ 1 数><最大値> // 最大値は文字列で表現します。バ 1 長と値文字列です。
NOMI<バ 1 数><初期値> // 初期値は文字列で表現します。バ 1 長と値文字列です。
LMID<バ 1 数><LIMITID 値> // LimitID 値は文字列で表現します。バ 1 長と ID 値文字列です。
LOWE<バ 1 数><LowerLimit 値> // LowerLimit 値は文字列で表現します。バ 1 長と ID 値文字列です。
UPPE<バ 1 数><UpperLimit 値> // UpperLimit 値は文字列で表現します。バ 1 長と ID 値文字列です。
EVNT<バ 1 数><CE 名> // 収集イベント名のバ 1 数と収集イベント名文字列です。

```

(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<バ 1 数><変数名> // CE 名のバ 1 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<バ 1 数><状態値> // ENABLE 状態のバ 1 数と状態値文字列
RPNA<バ 1 数><RPID 名> // LINK ポート ID の名前バ 1 長と名前 (0 個以上)

```

#### (3) レポート、REPORT 情報の形式

```

RP: // RP(Report)
NAME<バ 1 数><変数名> // Report 名のバ 1 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<バ 1 数><変数名> // LINK 変数名の名前バ 1 長と名前(0 個以上)

```

#### (4) アラーム ALARM 情報の形式

```

AL: // AL(Alarm)
NAME<バ 1 数><変数名> // Alarm 名のバ 1 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<バ 1 数><ALCD 値> // ALCD のバ 1 長と値文字列
ALTX<バ 1 数><ALTX 値> // ALTX のバ 1 長と値文字列
CEON<バ 1 長><CE 名> // ALARM 発生に LINK された CE の名前のバ 1 長と CE 名
CEOF<バ 1 長><CE 名> // ALARM 復旧に LINK された CE の名前のバ 1 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<バイト数><変数名> // PP 名のバイト長と PP 名(文字列)です。  
 ID <サイズ><ID 値> // PPID 値のバイト長と PPID 値(文字列)です。  
 FORM<ItemCode> // PPID のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // PPID のサイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MSIZE<値の最大サイズ> // PPID の最大サイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 PPBO<バイト数><PPBODY 値> // PPOBODY 値のバイト長と PPOBODY 値(文字列)です。  
 FORM<ItemCode> // PPOBODY のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // PPOBODY のサイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MSIZE<値の最大サイズ> // PPOBODY の最大サイズ (=配列サイズ)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<バイト数><変数名> // FPP 名のバイト長と FPP 名(文字列)です。  
 ID <サイズ><ID 値> // FPPID 値のバイト長と FPPID 値(文字列)です。  
 FORM<ItemCode> // FPPID のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // FPPID のサイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MSIZE<値の最大サイズ> // FPPID の最大サイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MDLN<バイト数><MDLN 値> // MDLN のバイト長と値文字列  
 SOFT<バイト数><SOFTREV 値> // SOFTREV のバイト長と値文字列  
 COUN<カウント値> // プログラムの数, 8 桁固定の 16 進表現です。(>=0)  
 CCOD<サイズ><CCODE 値> // CCODE(Command Code) 値のバイト長と CCODE 値(文字列)です。  
 FORM<ItemCode> // CCODE 値のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // CCODE 値のサイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MSIZE<値の最大サイズ> // CCODE 値の最大サイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 PCOU<カウント値> // CCODE に対するパラメータの数, 8 桁固定の 16 進表現です。(>=0)  
 PPAR<サイズ><PARAMETER 値> // PARA(parameter) 値のバイト長と PARA 値(文字列)です。  
 FORM<ItemCode> // PARA 値のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // PARA 値のサイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 MSIZE<値の最大サイズ> // PARA 値の最大サイズ (=配列サイズ)8 桁固定の 16 進表現です。  
 (COUN 分の CCODE が続き、RCOU 分のパラメータが続きます。)

(7) レシピ RCP 情報の形式

RC: // RC(RECIPE)  
 NAME<バイト数><変数名> // RCP 名のバイト長と RCP 名(文字列)です。  
 ID <サイズ><ID 値> // RCPID 値のバイト長と RCPID 値(文字列)です。  
 PPBO<バイト数><RCPBODY 値> // RCPBODY のバイト長と値文字列  
 COUN<カウント値> // PARAMETER の数, 8 桁固定の 16 進表現です。(>=0)  
 PNAM<サイズ><Parameter 名> // PARAMETER 名のバイト長と PARAMETER 名(文字列)です。  
 PVAL<サイズ><Parameter 値> // PVAL(パラメータ値)のバイト長と PVAL 値(文字列)です。  
 FORM<ItemCode> // PVAL 値のデータフォーマット 2 桁固定の 16 進表現です。  
 SIZE<値のサイズ> // PVAL 値のサイズ (=配列サイズ)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<バイト数><変数名>                 // SPOOL 名のバイト長と 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<バイト数><変数名>                 // TRACE 名のバイト長と TRACE 名(文字列)です。
ID <サイズ><ID 値>                     // TRID(Trace ID) 値のバイト長と TRID 値(文字列)です。
FORM<ItemCode>                        // TRID 値のデータフォーマット 2 桁固定の 16 進表現です。
SIZE<値のサイズ>                      // TRID 値のサイズ (=配列サイズ) 8 桁固定の 16 進表現です。
MSIZE<値の最大サイズ>                // TRID 値の最大サイズ (=配列サイズ) 8 桁固定の 16 進表現です。
DSPE<サイズ><DSPER 値>                // DSPE のバイト長と DSPE 値文字列
TOTS<合計サンプル数>                 // TotSmp の値の 8 桁固定の 16 進表現です。
REPG<レポートグループ数>            // 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<バイト数><変数名>               // Trace 対象の変数名のバイト長と変数名(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 00010040ENAB000000011RPNA0000000eRP_ReadyToLoad
RP: NAME0000000eRP_ReadyToLoadID 00010040VNAM00000008SV_ClockVNAM0000000eSV_ReadyToLoad
```

### (3) ヘダーファイル

.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
```

.C# ファイル(.Net C#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(V.Net VB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
```