M108V05

Debug Solutions Power Debugger

Debug System for Boundary Scan Board

コマンドリファレンス　付録－Ｄ

Ｄｅｖｉｃｅファイル解説

（フラッシュメモリ編）

Debug Solutions Power Debugger

ご注意

１．このソフトウェアの著作権は、Ｄｅｂｕｇ　Ｓｏｌｕｔｉｏｎｓ社にあります。
２．このソフトウェアおよびマニュアルの一部または全てを無断で使用、複製することはできません。
３．ソフトウェアは、コンピュータ１台につき１セット購入が原則となっております。
４．このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用可能です。
５．このソフトウェアおよびマニュアルを運用した結果の影響については、いっさい責任をおいかねますのでご了承下さい。

目次

１．フラシュメモリＤｅｖｉｃｅファイル概要

フラッシュメモリDeviceファイルはオンボード書き込みを行うフラッシュメモリデバイスの特性を表現したテキストファイルで、Power　Debugger固有のフォーマットです。例えばフラッシュメモリのデータ書き込みを行うためには、データを書き込む前に、特定アドレスに対して複数回アクセスすることで、書き込み可能となります。このようなデバイスに特有なシーケンスが記述された、ファイルを読み取ることで、Power　Debuggerはオンボード書き込みを実現しています。

２．Ｄｅｖｉｃｅファイルで使用される単語

２．１　キャラクタセット

Deviceファイルでの単語表記は大文字と小文字の認識はおこないません。又使用できる文字は以下のとおりです。

1）大文字、小文字　　:A, B, C･･･Z ，　a, b, c･･･z
2）数字　　:０－９
3）特殊文字　　:“　＆　‘（　）　＊　，　－　．　；　＜　＝　＞　＿　その他の特殊文字はコメントとしては使用可能です。
4）区切り文字　　:空白文字（スペース）　Tab文字

２．２　単語

BSDLで使用される単語は、１文字または複数のアルファベット文字、数字或いはアンダースコア（__）から構成されます。又、単語の文字数の上限はありません。以下の文字は有効な単語とみなされます。

```
DCF                          // 有効な単語
DCF__FILE__No1               // 有効な単語
```

２．３　ラベル

シーケンス定義内のコロン（:）ではじまる単語は、ラベルとして認識されます。ラベルはシーケンスを表現するための、ジャンプ先の位置を示す目的で使用されます。ラベルはシーケンス定義内のみで有効です。

２．４　コメント

２つのスラッシュ（//）　から行の最後までがコメントとしてみなされます。改行のみ、もしくは空白行は，//で定義しなくても無視されます。

３．文法定義

３．１　Ｄｅｖｉｃｅファイルの構造について

Deviceファイルの構造は以下のような構造をとらなければなりません。

＜DevVer　コマンド＞
[＜DevMaker　コマンド＞]
[＜DevName　コマンド＞]
{＜DevDef　コマンド＞}
＜ProgramStart　コマンド＞
　　＜シーケンス定義コマンド＞
　　　　：
　　＜シーケンス定義コマンド＞
＜ProgramEnd　コマンド＞
＜ChipEraseStart　コマンド＞
　　＜シーケンス定義コマンド＞
　　　　：
　　＜シーケンス定義コマンド＞
＜ChipEraseEnd　コマンド＞
＜SectorEraseStart　コマンド＞
　　＜シーケンス定義コマンド＞
　　　　：
　　＜シーケンス定義コマンド＞
＜SectorEraseEnd　コマンド＞
{＜Sector　コマンド＞}

※Deviceファイル表記においては、その要素順序は上記の構造であらわされる順序をまもらなければなりません。

３．２　コマンド構造について

Deviceファイルのコマンド構造は以下のような構成をとります。

フラッシュメモリDeviceファイルのコマンドは、以下の書式で構成されます。

コマンド　*パラメータ１　パラメータ２　・・・・　パラメータｎ*

コマンド ：　DevVer ， DevMaker ， DevName ， DevDef ， Sector ，
ProgramStart ， ProgramEnd ，
ChipEraseStart ， ChipEraseEnd ，
SectorEraseStart ，SectorEraseEnd

パラメータ ：　コマンドの種類により、必要なパラメータの数は異なります。

コマンド及びそのパラメータは1行で完結し、空白もしくはTABで区切られます。パラメータの数はコマンドによって0もしくは複数のパラメータを持ちます。又パラメータで定義される数値は、10進数もしくは16進数で表現され、16進数で表現する場合は先頭に0xを付与する必要があります。

例:

```
DevDef  Sector  40                              // 10進数
Sector  0x0  0x00000000  0x0000ffff             // 16進数
```

３．３　Ｄｅｖｉｃｅファイルコマンドリスト

FLMファイルで使用されるコマンドを以下に示します。

DevVer	：	デバイス定義ファイルバージョンを指定
DevMaker	：	デバイスメーカー定義
DevName	：	デバイス型各
DevDef	：	デバイス信号情報定義
Sector	：	セクタ情報定義
ProgramStart	：	プログラムシーケンス定義の始まりを指定
ProgramEnd	：	プログラムシーケンス定義の終了を指定
ChipEraseStart	：	チップ消去シーケンス定義の始まりを指定
ChipEraseEnd	：	チップ消去シーケンス定義の終了を指定
SectorEraseStart	：	セクタ消去シーケンス定義の始まりを指定
SectorEraseEnd	：	セクタ消去シーケンス定義の終了を指定

３．４　シーケンス定義コマンドリスト

シーケンス定義で使用されるコマンドを以下に示します。

WriteData	：	デバイスに対して書き込みアクセスを行います。
ReadData	：	デバイスに対して読み取りアクセスを行います。
CompData	：	内部に取り込んだデータとの比較を行います。
End	：	シーケンス定義を終了します。
Error	：	シーケンス定義をエラーを渡して終了します。

４．コマンド詳細

４．１　ＤｅｖＶｅｒ　コマンド

DevVer コマンドは、フラッシュメモリ定義ファイルのバージョンを定義します。

構文　　:DevVer *Majバージョン Minバージョン*

Majバージョン　：　major　バージョン[文字列型]
Minバージョン　：　miner　バージョン[文字列型]

例　　:DevVer　１　２

※現在のフラッシュメモリ定義ファイルのバージョンは　１．２　です。

４．２ ＤｅｖＭａｋｅｒ コマンド

DevMaker コマンドは、フラッシュメモリデバイスのメーカに関する属性を定義します。

構文 :DevMaker *Maker名*

Maker名 : フラッシュメモリデバイスメーカー名[文字列型]

例 :DevMaker Fujitsu

※DevMaker コマンドは定義されていなくても、エラーにはなりません。

４．３　ＤｅｖＮａｍｅ　コマンド

DevName コマンドは、フラッシュメモリデバイス名に関する属性を定義します。

構文　　:DevName *デバイス名*

デバイス名 ： フラッシュメモリデバイス名[文字列型]

例　　:DevName MBM29DLxxxxx

※DevName コマンドは定義されていなくても、エラーにはなりません。

４．４　ＤｅｖＤｅｆ　コマンド

DevDef コマンドは、フラッシュメモリの信号に関する属性を定義します。

DevDef *Parameter1 [Parameter2] [Parameter3]*

Parameter1 ： 属性定義定数

Adr	：	アドレス信号本数の定義
Data	：	データ信号本数の定義
CS1	：	CS1信号の属性定義
CS2	：	CS2信号の属性定義
OE1	：	OE1信号の属性定義
OE2	：	OE2信号の属性定義
WE1	：	WE1信号の属性定義
WE2	：	WE2信号の属性定義
BYTE	：	BYTE信号の属性定義
RESET	：	RESET信号の属性定義
WP	：	WP信号の属性定義
Sector	：	Sector数の定義

４．４．１　Ａｄｄｒｅｓｓ　定義

フラッシュメモリデバイスのアドレス信号本数を示します。

構文　　:DevDef Adr *アドレス信号数 アドレス取り込みタイミング（POS／NEG）*

アドレス信号数	:	フラッシュメモリデバイスのアドレス信号本数[DWORD型]
アドレス取り込みタイミング	:	アドレス信号の取り込みタイミング（POS／NEG）

例　　:DevDef Adr 21 POS

※アドレス取り込みタイミングが定義は、デバイスファイルバージョン1．2以降の対応となります。

４．４．２　Ｄａｔａ　定義

フラッシュメモリデバイスのデータ信号本数を示します。

構文　　:DevDef Data *データ信号数*

データ信号数 ： フラッシュメモリデバイスのデータ信号本数[DWORD型]

例　　:DevDef Data 16

４．４．３　ＣＳ１　定義

Chip Select 信号の属性を定義します。

構文　:DevDef CS1 *有効／無効 極性 定常状態*

有効／無効　：CS1信号の有効／無効を設定します。[文字列定義型]
- ON　：CS1信号が有効
- OFF　：CS1信号が無効

極性　：有効／無効設定がONのとき、CS1信号の極性を定義します。[文字列定義型]
- NEG　：負論理ーアクティブL
- POS　：正論理ーアクティブH

定常状態　：有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L　：定常状態　＝　L
- H　：定常状態　＝　H

例:
```
DevDef CS1 ON NEG H
DevDef CS1 OFF
```

４．４．４　ＣＳ２　定義

デバイスによっては、Chip Select 信号が2本ある場合があります。CS2 信号は2本目の Chip Select 信号の属性を定義します。

構文　:DevDef CS2 *有効／無効 極性 定常状態*

有効／無効　：CS2信号の有効／無効を設定します。[文字列定義型]
- ON　：CS2信号が有効
- OFF　：CS2信号が無効

極性　：有効／無効設定がONのとき、CS2信号の極性を定義します。[文字列定義型]
- NEG　：負論理－アクティブL
- POS　：正論理－アクティブH

定常状態　：有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L　：定常状態 ＝ L
- H　：定常状態 ＝ H

例:
```
DevDef CS2 ON NEG H
DevDef CS2 OFF
```

４．４．５　ＯＥ１　定義

Output Enable 信号の属性を定義します。

構文　:DevDef OE1 *有効／無効 極性 定常状態*

有効／無効　：OE1信号の有効／無効を設定します。[文字列定義型]
- ON　：OE1信号が有効
- OFF　：OE1信号が無効

極性　：有効／無効設定がONのとき、OE1信号の極性を定義します。[文字列定義型]
- NEG　：負論理ーアクティブL
- POS　：正論理ーアクティブH

定常状態　：有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L　：定常状態　＝　L
- H　：定常状態　＝　H

例:
```
DevDef OE1 ON NEG H
DevDef OE1 OFF
```

４．４．６　ＯＥ２　定義

デバイスによっては、Output Enable 信号が2本ある場合があります。OE2 信号は2本目の Output Enable 信号の属性を定義します。

構文　:DevDef OE2 *有効／無効 極性 定常状態*

有効／無効　：OE2信号の有効／無効を設定します。[文字列定義型]
- ON　：OE2信号が有効
- OFF　：OE2信号が無効

極性　：有効／無効設定がONのとき、OE2信号の極性を定義します。[文字列定義型]
- NEG　：負論理－アクティブL
- POS　：正論理－アクティブH

定常状態　：有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L　：定常状態　＝　L
- H　：定常状態　＝　H

例:
```
DevDef OE2 ON NEG H
DevDef OE2 OFF
```

４．４．７　ＷＥ１　定義

Write Enable 信号の属性を定義します。

構文　:DevDef WE1 *有効／無効 極性 定常状態*

有効／無効　: WE1信号の有効／無効を設定します。[文字列定義型]
- ON : WE1信号が有効
- OFF : WE1信号が無効

極性　: 有効／無効設定がONのとき、WE1信号の極性を定義します。[文字列定義型]
- NEG : 負論理ーアクティブL
- POS : 正論理ーアクティブH

定常状態　: 有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L : 定常状態 ＝ L
- H : 定常状態 ＝ H

例:
```
DevDef WE1 ON NEG H
DevDef WE1 OFF
```

４．４．８ ＷＥ２ 定義

デバイスによっては、Write Enable 信号が2本ある場合があります。WE2 信号は2本目の Write Enable 信号の属性を定義します。

構文 :DevDef WE2 *有効／無効 極性 定常状態*

有効／無効 : WE2信号の有効／無効を設定します。[文字列定義型]
ON : WE2信号が有効
OFF : WE2信号が無効

極性 : 有効／無効設定がONのとき、WE2信号の極性を定義します。[文字列定義型]
NEG : 負論理－アクティブL
POS : 正論理－アクティブH

定常状態 : 有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
L : 定常状態 ＝ L
H : 定常状態 ＝ H

例: DevDef WE2 ON NEG H
DevDef WE2 OFF

４．４．９　ＢＹＴＥ　定義

データビット幅が16ビット以上のデバイスによっては、BYTEモードで使用するか、Wordモードで使用するかの、切替信号をもつものがあります。BYTE は BYTE／WORDモードの属性を定義します。

構文　:DevDef BYTE *有効／無効 極性 定常状態*

有効／無効　: BYTE信号の有効／無効を設定します。[文字列定義型]
ON : BYTE信号を持つ。
OFF : BYTE信号を持たない。

極性　: 有効／無効設定がONのとき、BYTE信号の極性を定義します。[文字列定義型]
NEG : 負論理－アクティブL（L入力でBYTEモード）
POS : 正論理－アクティブH（H入力でBYTEモード）

定常状態　: 有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
L : 定常状態 ＝ L
H : 定常状態 ＝ H

例:
```
DevDef BYTE ON NEG H
DevDef BYTE OFF
```

４．４．１０　ＲＥＳＥＴ　定義

デバイスによっては、フラッシュメモリの初期化を行なう（メモリのクリアではない）為にRESET信号をもつものがあります。RESET は RESET信号の属性を定義します。

構文　:DevDef RESET *有効／無効 極性 定常状態*

有効／無効　: RESET信号の有効／無効を設定します。[文字列定義型]
- ON : RESET信号を持つ。
- OFF : RESET信号を持たない。

極性　: 有効／無効設定がONのとき、RESET信号の極性を定義します。[文字列定義型]
- NEG : 負論理－アクティブL（L入力でRESET有効）
- POS : 正論理－アクティブH（H入力でRESET有効）

定常状態　: 有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L : 定常状態 ＝ L
- H : 定常状態 ＝ H

例:
```
DevDef RESET ON NEG H
DevDef RESET OFF
```

４．４．１１　ＷＰ　定義

デバイスによっては、書き込みデータの保護を行なう Write Protect 信号を持つものがあります。WP は Write Protect 信号の属性を定義します。

構文　:DevDef WP *有効／無効 極性 定常状態*

有効／無効　：WP信号の有効／無効を設定します。[文字列定義型]
- ON　：WP信号を持つ。
- OFF　：WP信号を持たない。

極性　：有効／無効設定がONのとき、WP信号の極性を定義します。[文字列定義型]
- NEG　：負論理－アクティブL（L入力でWP有効）
- POS　：正論理－アクティブH（H入力でWP有効）

定常状態　：有効／無効設定がONのとき、初期状態、及び定常状態での、信号レベルを定義します。[文字列定義型]
- L　：定常状態　＝　L
- H　：定常状態　＝　H

例:
```
DevDef WP ON NEG H
DevDef WP OFF
```

４．４．１２　Ｓｅｃｔｏｒ　定義

フラッシュメモリデバイスのセクタ数を示します。

構文　　:DevDef Sector *Sector数*

セクタ数 : フラッシュメモリデバイスのセクタ数[DWORD型]

例　　:DevDef Sector 32

４．５　Ｓｅｃｔｏｒ　コマンド

Sector コマンドは、フラッシュメモリのセクタ情報を定義します。

構文　　:Sector *Sector番号 開始アドレス 終了アドレス 消去アドレス*

Sector番号 ： セクタ番号（DevDef Sector で定義された値以下でなければなりません）
［DWORD型］

開始アドレス ： Sector番号で定義されたセクタの開始アドレス［DWORD型］

終了アドレス ： Sector番号で定義されたセクタの終了アドレス［DWORD型］

消去アドレス ： Sector番号で定義されたセクタを消去する際にアクセスするアドレス
［DWORD型］

例　　:Sector 8 0x00008000 0x00008fff 0x00008000

４．６ ＰｒｏｇｒａｍＳｔａｒｔ 定義

フラッシュメモリのプログラムシーケンス定義の始まりを示します。

構文 :ProgramStart

例 :

```
DevDef xxxxx
DevDef xxxxx
DevDef xxxxx
DevDef xxxxx
DevDef xxxxx

ProgramStart

～～～～～～～～～～～～～～～～～～～～～
  ここにプログラムシーケンスを記述
～～～～～～～～～～～～～～～～～～～～～

ProgramEnd
```

４．７ ＰｒｏｇｒａｍＥｎｄ 定義

フラッシュメモリのプログラムシーケンス定義の終わりを示します。

構文 :ProgramEnd

４．８　ＣｈｉｐＥｒａｓｅＳｔａｒｔ　定義

フラッシュメモリのチップ消去シーケンス定義の始まりを示します。

構文　　:ChipEraseStart

例　　:

```
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx

ChipEraseStart

～～～～～～～～～～～～～～～～～～～～
  ここにプログラムシーケンスを記述
～～～～～～～～～～～～～～～～～～～～

ChipEraseEnd
```

４．９　ＣｈｉｐＥｒａｓｅＥｎｄ　定義

フラッシュメモリのチップ消去シーケンス定義の終わりを示します。

構文　　:ChipEraseEnd

４．１０　ＳｅｃｔｏｒＥｒａｓｅＳｔａｒｔ　定義

フラッシュメモリのセクタ消去シーケンス定義の始まりを示します。

構文　　:SectorEraseStart

例　　:

```
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx
DevDef  xxxxx

SectorEraseStart

～～～～～～～～～～～～～～～～～～～～
  ここにプログラムシーケンスを記述
～～～～～～～～～～～～～～～～～～～～

SectorEraseEnd
```

４．１１　ＳｅｃｔｏｒＥｒａｓｅＥｎｄ　定義

フラッシュメモリのセクタ消去シーケンス定義の終わりを示します。

構文　　:SectorEraseEnd

４．１２　プログラムシーケンス定義文

４．１２．１　プログラムシーケンス定義文構文

それぞれ ProgramStart から ProgramEnd ， ChipEraseStart から ChipEraseEnd ， SectorEraseStart から SectorEraseEnd ではさまれた中でシーケンス定義を行います。

シーケンス定義文には、シーケンス定義コマンドとラベルで構成されます。シーケンスの実行に際しては、Startコマンド（ProgramStart ， ChipEraseStart ， SectorEraseStart） の次のシーケンス定義コマンドがら順に実行され、シーケンス定義文は、End コマンド又はErrorコマンドの実行で終了します。

☆ラベル名
ラベル名は コロン（:）で始まる文字列で、条件分岐文によるジャンプ先となります。

☆シーケンス定義コマンド

WriteData	:	アドレス／データを指定して Write Enable を有効にします。
ReadData	:	アドレス／データを指定して OutPut Enable を有効にします。
CompData	:	内部変数に保持した値とパラメータ値と比較し、条件分岐を行います。
End	:	シーケンスを終了します。
Error	:	エラーを通知して、シーケンスを終了します。

４．１２．２ ＷｒｉｔｅＤａｔａ 定義

フラッシュメモリにアドレス／データを指定して Write Enable を有効にします。

構文 :WriteData *Address値* *Data値*

Address値 : フラッシュメモリのアドレス信号に入力する値を指定します。[DWORD型]
定数の他に対象となるデバイス書き込みアドレス／消去アドレスを指定する場合は ＃Address を使用します。[文字列定義型]

Data値 : フラッシュメモリのデータ信号に入力する値を指定します。[DWORD型]
定数の他に対象となるデバイス書き込みアドレス／消去アドレスを指定する場合は ＃Data を使用します。[文字列定義型]

例 : WriteData 0x555 0xaa
WriteData ＃Address ＃Data

４．１２．３　ＲｅａｄＤａｔａ　定義

フラッシュメモリにアドレス／データを指定して OutPut Enable を有効にし、データ信号上の値を内部変数に保持します。保持した値は、CompData で使用します。

構文　　:ReadData *Address値*

Address値 ： フラッシュメモリのアドレス信号に入力する値を指定します。[DWORD型]
定数の他に対象となるデバイス書き込みアドレス／消去アドレスを指定する場合は ＃Address を使用します。[文字列定義型]

例 ： WriteData 0x555 0xaa
WriteData ＃Address ＃Data

４．１２．４　ＣｏｍｐＤａｔａ　定義

ReadData で内部変数に保持した値とパラメータ値と比較し、条件分岐を行います。

構文　　:CompData *Mask値 比較Data値 条件式1 ラベル名1 ［ 条件式2 ラベル名2 ］*

Mask値　　：比較の対象となるビットを1としたビットマスクを指定します。[DWORD型]

比較Data値　：内部変数に保持した値と比較を行うデータ値を指定します。[DWORD型]
定数の他に対象となるデバイス書き込みアドレス／消去アドレスを指定する場合は ＃Data を使用します。[文字列定義型]

条件式1　　：Then 又は Else を指定します。Then の時は、比較結果が一致した場合にElse の時は比較結果が不一致の場合に、ラベル名で指定した、ジャンプ先にジャンプします。[文字列定義型]

ラベル名1　：ジャンプ先のラベル名を指定します。[文字列型]

条件式2　　：Else を指定します。比較結果が不一致の場合に、ラベル名で指定した、ジャンプ先にジャンプします。[文字列定義型]

ラベル名2　：ジャンプ先のラベル名を指定します。[文字列型]

内部変数とMask値、比較データ値の関係は以下のとおりとなります。

if （（内部変数 ＆ Mask値）＝＝（比較データ値 ＆ Mask値））then ラベル名1 else ラベル名2

例 ： CompData 0x80 ＃Data then :OK
CompData 0x20 0x20 else :Polling
CompData 0x80 ＃Data then :OK else :NG

４．１２．５　Ｅｎｄ　定義

シーケンスを終了します。

構文　　:End

例 :

```
:OK
  End

:NG
  Error
```

４．１２．６　Ｅｒｒｏｒ　定義

エラー通知を行ってシーケンスを終了します。

構文　　:Error

例 :

```
:OK
  End

:NG
  Error
```

Web http://www.debsol.com
E-Mail mail@debsol.com

デバッグソリューションズ
Debug Solutions