

## 要点

- USB規約
  - ・ USB DFUクラス基準
  - ・ 自動通信速度(8/16MHzクリスタル)
- 実装書き込み
  - ・ チップ上メモリのフラッシュメモリとEEPROMの読み書き
  - ・ デバイスIDの読み込み
  - ・ 完全なチップ消去
  - ・ 応用開始命令
- 応用内プログラミング
  - ・ チップ上フラッシュドライバ用ソフトウェア入口

## 1. 叙述

USBインターフェースを持つ8ビットmegaデバイスは工場では制御器のチップ上フラッシュブート領域に配置されたUSBブートローダと共に形態付けされます。

USBブートローダは系からデバイスを取り外すことなく、または予め書かれた応用なしに、そして外部のどんなプログラミングインターフェースもなしに、USBホスト制御器からの実装プログラミングの実行を許します。

この資料はチップ上のフラッシュメモリ(フラッシュメモリとEEPROM)での効率的な操作実行のための直列規約だけでなくUSBブートローダの機能も記述します。

## 2. ブートローダ環境

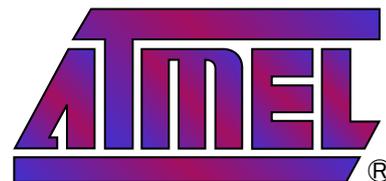
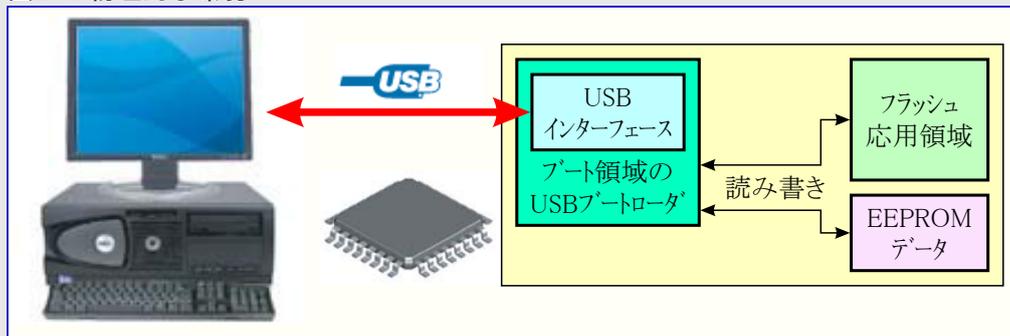
ブートローダはチップ上のフラッシュメモリのブート領域に配置され、USB通信規約を管理してチップ上のメモリ(フラッシュメモリ/EEPROM)への読み書き操作を実行します。

USBブートローダはチップ上のフラッシュメモリの“ブートローダフラッシュ領域”に配置されます。ブートローダフラッシュ領域の大きさはブートローダ容量よりも大きくなければなりません。USB製品は既定のチップ上USBブートローダとブート領域形態設定が工場では形態設定されています。

表2-1. USBブートローダパラメータ

製品	フラッシュメモリのブート領域容量形態設定	VID/PID	ブートローダ開始アドレス(語アドレス)	
AT90USB1287	4K語	\$03EB/\$2FFB	\$F000	
AT90USB1286				
AT90USB647	2K語	\$03EB/\$2FF9	\$7800	
AT90USB646				
AT90USB162			\$03EB/\$2FFA	\$1800
AT90USB82			\$03EB/\$2FF7	\$0800
ATmega32U4			\$03EB/\$2FF4	\$3800
ATmega16U4			\$03EB/\$2FF3	\$1800

図2-1. 物理的な環境



## USB DFU ブートローダ データシート

AT90USB128x  
AT90USB64x  
AT90USB162  
AT90USB82  
ATmega32U4  
ATmega16U4

本書は一般の方々の便宜のため有志により作成されたもので、ATMEL社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

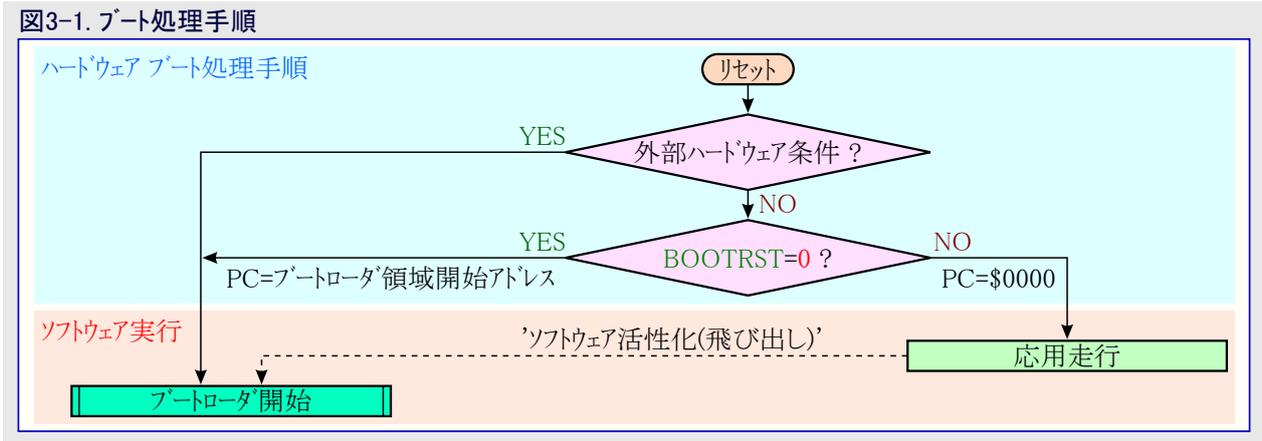
Rev. 7618C-07/08, 7618CJ1-03/14

### 3. ブートローダの活性化

AT90USBデータシートで詳述されるように、ブートローダは以下の条件の1つによって活性化することができます。

- **通常の応用実行**：使用者応用プログラムからの飛び出しまたは呼び出し。これはUSB、USARTまたはSPI経由で受信された命令と応用による復号のような起動によって始められます。
- **ブートリセットヒューズ**：ブートリセット(BOOTRST)ヒューズはリセット後にリセットベクタがブートフラッシュ領域開始アドレスを指示するようにプログラム(0)することができます。一旦使用者コードが格納されたなら、ブートローダ命令("応用開始")は応用コードの実行を開始することができます。ヒューズがMCU自身によって変更できないことに注意してください。これは一旦ブートリセットヒューズがプログラム(0)されると、リセットベクタが常にブートローダリセットを指示し、そしてヒューズが直列または並列のプログラミングインターフェースを通してだけ変更することができることを意味します。BOOTRSTヒューズは既定工場設定に於いて活性ではありません。
- **外部ハードウェア条件**：ハードウェアブート許可(HWBE)ヒューズはリセット下で特別なハードウェア条件になり、リセット後にブートローダが強制されるようにプログラム(0)することができます。

これらの各種条件は図3-1.で要約されます。



### 4. 規約

#### 4.1. デバイスファームウェア更新紹介

デバイスファームウェア更新(DFU)はデバイスファームウェア変更を実行するために実装された機構です。どのUSBデバイスもこの資料で指定される必要条件を支援することによってこの能力を利用することができます。

DFU動作とその通常走行時活動の両方を同時に実行することがデバイスに対して非実用的なので、それらの通常の活動はDFU動作中の間、中止されなければなりません。それを行うことはその動作形態を変更しなければならないことを意味し、換言すると、印刷機はそれがファームウェア更新実行下であると時に**印刷機ではなく**、それはEPROM書き込み器です。けれども、DFUを支援するデバイスはそれ自身でその動作形態を変更する能力がありません。外部(人またはホストオペレーティングシステム)の介在が必要とされます。

#### 4.2. DFU特定要求

USB標準要求に加えて、更新操作を達成するために7つのDFUクラス特定要求が使用されます。

表4-1. DFUクラス特定要求

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 01 00001	DFU_DETACH (0)	wTimeout	インターフェース (4)	0	なし
0 01 00001	DFU_DNLOAD (1)	wBlock	インターフェース (4)	データ長	ファームウェア
1 01 00001	DFU_UPLOAD (2)	wBlock	インターフェース (4)	データ長	ファームウェア
1 01 00001	DFU_GETSTATUS (3)	0	インターフェース (4)	6	状況
0 01 00001	DFU_CLRSTATUS (4)	0	インターフェース (4)	0	なし
1 01 00001	DFU_GETSTATE (5)	0	インターフェース (4)	1	状態
0 01 00001	DFU_ABORT (6)	0	インターフェース (4)	0	なし

#### 4.3. DFU記述子1式

装置は以下を含むDFU記述子1式を出します。

- DFU装置記述子
- 1つの形態設定記述子
- (存在するなら、代替設定用の記述子を含む)1つのインターフェース記述子

### 4.3.1. DFU装置記述子

この記述子はDFU形態記述子1式にだけ存在します。DFUクラス符号がこの記述子の**bDeviceClass**領域で報告されます。

表4-2. DFU形態装置記述子

変位(オフセット)	領域	バイト数	値	説明
0	bLength	1	\$12	バイトでのこの記述子の量
1	bDescriptorType	1	\$01	DFU機能記述子形式
2	bcdUSB	2	\$0100	BCDでのUSB仕様公布版番号
4	bDeviceClass	1	\$FE	応用固有クラス符号
5	bDeviceSubClass	1	\$01	デバイスファームウェア更新符号
6	bDeviceProtocol	1	\$00	装置はこのインターフェースでクラス固有規約を使用しません。
7	bMaxPacketSize0	1	32	(ホスト側ドライバのために32に制限される)エンドポイント0の最大パケット容量
8	idVendor	2	\$03EB	供給者ID
10	idProduct	2	\$2FFB	製品ID
12	bcdDevice	2	\$0000	BCDでの装置公開版番号
14	iManufacturer	1	0	(製造業者)文字列記述子の指標
15	iProduct	1	0	(製品)文字列記述子の指標
16	iSerialNumber	1	0	(通番)文字列記述子の指標
17	bNumConfigurations	1	\$01	DFUに対して1つ形態設定のみ

### 4.3.2. DFU形態設定記述子

この記述子は**bNumInterfaces**領域が値\$01を含まなければならない例外を除き、USB DFU仕様1.0版で記述される標準形態設定記述子と同じです。

#### 4.3.2.1. DFUインターフェース記述子

これはDFU形態での動作時に利用可能なインターフェースだけのための記述子です。従って、**bInterfaceNumber**領域の値は常に0です。

表4-3. DFU形態インターフェース記述子

変位(オフセット)	領域	バイト数	値	説明
0	bLength	1	\$09	バイトでのこの記述子の量
1	bDescriptorType	1	\$04	インターフェース記述子形式
2	bInterfaceNumber	1	\$00	このインターフェースの番号
3	bAlternateSetting	1	\$00	代替設定(注)
4	bNumEndpoints	1	\$00	制御パイプだけが使用されます。
5	bInterfaceClass	1	\$FE	応用固有クラス符号
6	bInterfaceSubClass	1	\$01	デバイスファームウェア更新符号
7	bInterfaceProtocol	1	\$00	装置はこのインターフェースでクラス固有規約を使用しません。
8	iInterface	1	\$00	このインターフェース用の文字列記述子の指標

**注:** 代替設定は追加のメモリ区分をアクセスするために応用によって使用することができます。この場合、各代替設定は目的対象メモリ区分を示すように文字列記述子を使用(例えば"EEPROM")することが示唆されます。代替設定の他に可能な使用に関する詳細はこの資料の範囲外です。けれども、著者は実装が代替設定に関して追加の創造的な使用を考案するのを予想するため、それらの使用は意図的に制限されていません。

## 4.4. 命令説明

AT90USBブートローダに実装された規約は以下を許します。

- 通信の開始
- フラッシュメモリまたはEEPROMのデータ書き込み
- フラッシュメモリまたはEEPROMのデータ読み込み
- 形態設定情報の書き込み
- 形態設定と製造者の情報の読み込み
- フラッシュメモリの消去
- 応用の開始

規約の概要は11頁の「[追補-A](#)」で詳述されます。

## 4.5. 装置状況

### 4.5.1. 状況取得

装置との同期を容易にするためにホストはDFU\_GETSTATUS要求を使用します。この状況は直前の要求の実行での情報を進行中/OK/失敗などで与えます。

bmRequestType	bRequest	wValue	wIndex	wLength	データ
1 01 00001	DFU_GETSTATUS (3)	0	インターフェース (4)	6	状況
0 01 00001	DFU_CLRSTATUS (4)	0	インターフェース (4)	0	なし

装置は以下のデータを含むデータ本体パケットでDFU\_GETSTATUS要求に応答します。

表4-4. DFU\_GETSTATUS応答

変位(オフセット)	領域	バイト数	値	説明
0	bStatus	1	数値	最後の要求の実行からの結果状況の表示
1	bwPollTimeOut	3	数値	DFU_GETSTATUSに続く送信前にホストが待つべきmsでの最小時間。この領域の目的はDFU_GETSTATUS経由での装置状態の継続的な請求と次のDFU_DNLOADの状態段階との間で待つことを装置がホストに期待する時間の量を動的に調整することを装置に許します。
4	bState	1	数値	この応用の送信直後に装置が入る状態の表示
5	iString	1	指標	文字列表内の状態記述子の指標

表4-5. bStatus値

状況	値	説明
OK	\$00	異常なし
errTARGET	\$01	ファイルがこのデバイスによって使用されるための目的対象ではありません。
errFILE	\$02	ファイルはこのデバイス用ですが、いくつかの供給者固有照合検査を失敗しました。
errWRITE	\$03	メモリ書き込みできないデバイスID
errERASE	\$04	メモリ消去機能失敗
errCHECK_ERASED	\$05	メモリ消去検査失敗
errPROG	\$06	メモリ書き込み機能失敗
errVERIFY	\$07	メモリ書き込み検査失敗
errADDRESS	\$08	受信したアドレスが範囲外のためメモリ書き込み不能
errNOTDONE	\$09	wLength=0でのDFU_DNLOAD受信、装置は全てのデータを未だ持っていると思いません。
errFIRMWARE	\$0A	装置のファームウェアが不正、走行時操作へ戻れません。
errVENDOR	\$0B	iStringが供給者固有の異常を指示
errUSBR	\$0C	装置が予期せぬUSBリセット信号を検出
errPOR	\$0D	装置が予期せぬ電源ONリセットを検出
errUNKNOWN	\$0E	何か不正なものが来たけれど、装置はそれが何か判りません。
errSTALLEDPK	\$0F	装置が予期せぬ要求で立ち往生

表4-6. bState値

状況	値	説明
appIDLE	0	装置は通常応用で走行中
appDETACH	1	装置は通常応用で走行中、受信したDFU_DETACH要求を持ち、USBリセット待機中
dfuIDLE	2	装置はDFU形態で動作中、そして要求待ち
dfuDNLOAD-SYNC	3	装置は受信した塊を持ち、GET_STATUS経由で状況を求めるホスト待ち
dfuDNBUSY	4	装置は不揮発性メモリに塊書き込み中
dfuDNLOAD-IDLE	5	装置はダウンロード操作処理中、DFU_DNLOAD要求を期待
dfuMANIFEST-SYNC	6	装置はホストから受信したファームウェアの最後の塊を持ち、具現段階を始めるためのDFU_GET_STATUS受信待ち、または、装置は具現段階を完了してDFU_GETSTATUSの受信待ち
dfuMANIFEST	7	装置は具現化段階
dfuMANIFEST-WAIT-RESET	8	装置はメモリ書き込みが行われ、USBリセットまたは電源ONリセット待ち
dfuUPLOAD-IDLE	9	装置はアップロード操作処理中、DFU_UPLOAD要求を期待
dfuERROR	10	異常発生、DFU_CLRSTATUS要求待機

#### 4.5.2. 状況解除

DFU\_GETSTATUS要求への応答でホストへ異常表示状況を検出して報告する度毎に、dfuERROR状態に入ります。何れかの異常状況報告後、装置はDFU\_CLRSTATUS要求が受信されるまでdfuERROR状態を抜け出すことができません。DFU\_CLRSTATUSの受信で、装置は状況をOKに設定し、dfuIDLE状況へ移動します。一旦装置がdfuIDLE状況になると、その後に他の状況へ移動できます。

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 01 00001	DFU_CLRSTATUS (4)	0	インターフェース (4)	0	なし

#### 4.5.3. 装置状態

報告した状態は応答の送信までの装置の現在の状態です。bState領域で指定した値はDFU\_GETSTATUSで報告されるそれらと同じです。

bmRequestType	bRequest	wValue	wIndex	wLength	データ
1 01 00001	DFU_GETSTATE (5)	0	インターフェース (4)	1	状態

#### 4.5.4. DFU\_ABORT要求

DFU\_ABORT要求は他のそんな状態からも抜け出してdfuIDLE状況へ戻ることを装置に強制します。装置はこの要求の受信でOK状況を設定します。より多くの情報については対応する状況遷移要約をご覧ください。

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 01 00001	DFU_ABORT (6)	0	インターフェース (4)	0	なし

#### 4.6. フラッシュメモリまたはEEPROMのデータ書き込み

ファームウェアイメージはDFU\_DNLOADクラス固有要求によって始められる制御書き込み転送経由でダウンロードされます。ホストは制御書き込み転送に於いて装置へbMaxPacketSize0とwTransferSizeの間のバイト数を送ります。各ダウンロードする塊に続いて、ホストはDFU\_GETSTATUS要求で装置の状況を求めます。

USB DFU仕様で記述されるように、「特定装置に対するファームウェアイメージは定義によって供給者指定です。従ってそれは目的対象アドレス、記録容量、そしてファームウェアイメージに内包される更新の支援に関連する他の全ての情報が必要とされます。それらの内包されたデータをそれらの装置が処理できることを保証するのは装置製造業者とファームウェア開発業者の責任です。DFUファイル接尾子の例外有りてファームウェアイメージファイルの内容はホストと無関係です。」

ファームウェアイメージ:

- 32バイト: 命令
- xバイト: xはファームウェアの先頭バイトの前に追加される(\$00値の)バイト数です。x数はファームウェアの先頭をフラッシュのページに整列するように計算されます。x=開始アドレス[32](**訳補**: ÷32での剰余)。例えば、開始アドレスが\$00AF(175)の場合、x=175[32]=15。
- ファームウェア
- 16バイトのDFU接尾子

表4-7. DFUファイル接尾子

変位(オフセット)	領域	バイト数	値	説明
-0	dwCRC	4	数値	dwCRCを除く、ファイル全体のCRC
-4	bLength	1	16	dwCRCを含む、このDFU接尾子の長さ
-5	ucDfuSignature	3	5:\$44 6:\$46 7:\$55	固有DFU識票領域
-8	bcdDFU	2	BCD(\$0100)	DFU仕様番号
-10	idVendor	2	ID	このファイルと連携する供給者ID。\$FFFFか、または装置の供給者IDと一致しなければなりません。
-12	idProduct	2	ID	このファイルと連携する製品ID。\$FFFFか、または装置の製品IDと一致しなければなりません。
-14	bcdDevice	2	BCD	このファイルと連携する装置の公開番号。\$FFFFか、またはBCDでのファームウェア公開または版の番号。

#### 4.6.1. ホストからの要求

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 01 00001	DFU_DNLOAD (1)	wBlock	インターフェース (4)	データ長	ファームウェア

##### 4.6.1.1. 書き込み命令

命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_prog_start (\$01)	\$00	開始アドレス		終了アドレス		フラッシュメモリ書き込み開始
	\$01					EEPROM書き込み開始

書き込み命令は6バイト長です。USB仕様の制御形式転送に合わせるため、この書き込み命令は26(=32-6)の重要でないバイトと共に完了されます。そしてこの命令長は32バイトで、これは既定制御エンドポイントの長さです。

##### 4.6.1.2. ファームウェア

ファームウェアは今や装置へダウンロード(書き込む)ことができます。フラッシュメモリのページ容量に従うため、書き込む先頭バイトの前にx個の重要でないバイトが追加されます。x数はファームウェアの始めがフラッシュメモリのページで整列するように計算されます。x=開始アドレス[32](**訳補**: ÷32での剰余)。例えば、開始アドレスが\$00AF(175)の場合、x=175[32]=15。

##### 4.6.1.3. DFU接尾子

書き込む最終バイト直後に16バイトのDFU接尾子が追加されます。この接尾子は将来の使用のために予約されています。

図4-1. ファームウェアダウンロード(書き込み)0長DFU\_DNLOAD要求の例



ホストはファームウェアイメージファイルの転送が完了したことを示すために0長パケット(ZLP)でDFU\_DNLOAD要求を送ります。これはダウンロード(書き込み)操作の最後の本体パケットです。

##### 4.6.1.4. プートローダからの返答

各書き込み要求後、ホストはDFU\_GETSTATUSメッセージを送ることによって装置の状態と状況を要求することができます。装置状況が異常を示す場合、ホストは装置にDFU\_CLRSTATUSを送らなければなりません。

#### 4.7. フラッシュメモリまたはEEPROMのデータ読み込み

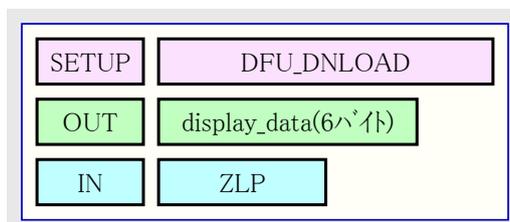
下で記述される流れはフラッシュメモリまたはEEPROMデータメモリ内のデータ読み込みを使用者に許します。フラッシュメモリでの空き検査命令はこの流れで可能です。

この操作は次の2段階で実行されます。

- ・読み込み命令のDFU\_DNLOAD要求 (6バイト)
- ・直前の命令に対応するDFU\_UPLOAD要求

##### 4.7.1. ホストからの最初の要求

ホストはデータ領域内に表示(display\_data)命令を持つDFUダウンロード(書き込み)命令を送ります。



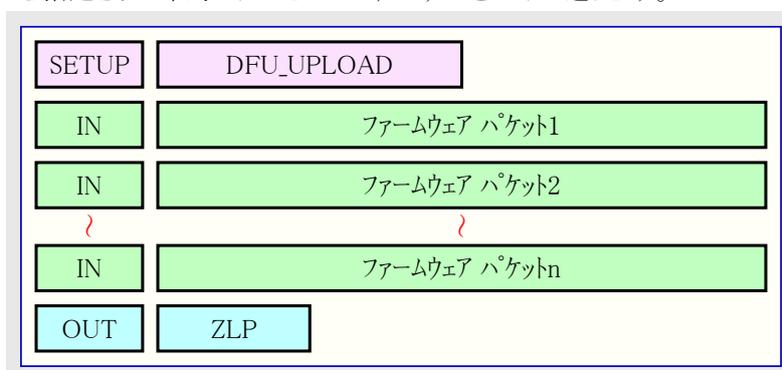
命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_display_data (\$03)	\$00	開始アドレス			終了アドレス	フラッシュメモリデータ表示(読み)
	\$01					フラッシュメモリ空き検査
	\$02					EEPROMデータ表示(読み)

##### 4.7.2. ホストからの2つ目の要求

ホストはDFUアップロード(読み込み)要求を送ります。

##### 4.7.3. 装置からの返答

装置は指定された開始アドレスから指定された終了アドレスまでのファームウェアをホストへ送ります。



##### 4.7.4. 空き検査命令に対する装置からの返答

ホスト制御器は装置にGET\_STATUS要求を送ります。一旦内部的な空き検査が完了すると、装置はその状況を送ります。

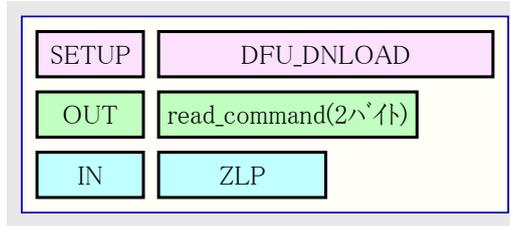
- ・装置状況が"OK"なら、装置のメモリは空きで、装置は次のホスト要求を待ちます。
- ・装置状況が"errCHECK\_ERASED"なら、装置のメモリは空きではありません。装置は\$FFでないバイトの先頭アドレスを送るためのDFU\_UPLOAD要求を待ちます。

## 4.8. 形態設定情報または製造業者情報の読み込み

これより後で記述される流れは形態設定または製造業者の情報読み込みを使用者に許します。

### 4.8.1. ホストからの要求

プログラミング操作を開始するために、ホストはデータ領域(2バイト)内に読み込み命令を持つDFU\_DNLOAD要求を送ります。

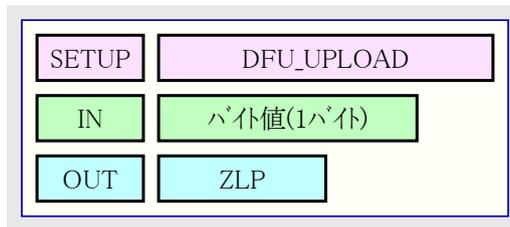


命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_read_command (\$05)	\$00	\$00				ブートローダ版番号読み込み
		\$01				装置ブートID1読み込み
		\$02				装置ブートID2読み込み
	\$01	\$30				製造業者符号読み込み
		\$31				系列符号読み込み
		\$60				製品名読み込み
		\$61				製品改訂番号読み込み

### 4.8.2. ブートローダからの返答

装置はDFU\_GETSTATUS要求に対して2つの可能な返答を持ちます。

- ・チップがプログラミング アクセスから保護されている場合、ホストへ"err\_VENDOR"状況が返されます。
- ・さもなければ、装置状況は"OK"です。ホストは要求した領域の値を得るために装置へDFU\_UPLOAD要求を送ることができます。



## 4.9. フラッシュメモリの消去

下で記述される流れはフラッシュメモリの消去を使用者に許します。

### 4.9.1. ホストからの要求

消去操作を開始するために、ホストはデータ領域(2バイト)内に書き込み命令を持つDFU\_DNLOAD要求を送ります。

命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_write_command (\$04)	\$00	\$FF				チップ全消去(\$FFでのビット)

### 4.9.2. ブートローダからの返答

装置はDFU\_GETSTATUS要求に対して2つの可能な返答を持ちます。

- ・チップがプログラミング アクセスから保護されている場合、ホストへ"err\_WRITE"状況が返されます。
- ・さもなければ、装置状況は"OK"です。

## 4.10. 応用の開始

下で記述される流れは特定命令受信でブートローダからの直接的な応用の開始を許します。

以下のように2つの任意選択が可能です。

- ・ウォッチドッグを使用して内部的なハードウェアリセットで応用を開始します。装置がこの命令を受信すると、ウォッチドッグが許可されてブートローダはウォッチドッグがデバイスのリセットするまで待機繰り返しへ移行します。
- ・リセットなしで応用を開始します。リセットなしで応用を開始するのにアドレス\$0000へ飛ぶことが用いられます。

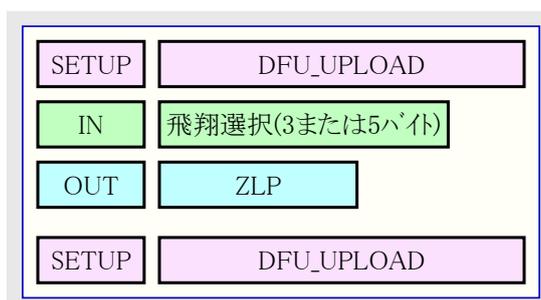
応用を開始するために、ホストはデータ領域(3または5バイト)内に特定の応用開始形式でDFU\_UPLOAD要求を送ります。

この要求は2つの任意選択の1つで応用を開始するためのデータ領域なしの2つ目のDFU\_UPLOAD要求が直ちに後続します。

### 重要注意:

ブートローダは応用領域の実行を許す“ハードウェアリセット”を生成するためにウォッチドッグリセットを実行します。ウォッチドッグリセット発生後、AVRのウォッチドッグはまだ走行中で、従って応用はプログラム始動でウォッチドッグを禁止するように取り計らうべきです(さもないとハードウェアウォッチドッグを管理しない応用は無限のリセット繰り返しで走行するでしょう)。

### 4.10.1. ホストからの要求



命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_write_data (\$04)	\$03	\$00				ハードウェアリセット
		\$01	アドレス			

### 4.10.2. ブートローダからの返答

装置によって返答は全く返されません。

## 5. 安全性

USBブートローダ接続開始時、ブートローダは(製品の施錠ビット設定に拘らず)自動的にソフトウェア読み書き安全動作形態へ移行します。これはUSBインターフェース上の読み書きアクセスからチップ上のフラッシュメモリ内容の保護を許します。従ってUSBブートローダ接続後に許されるDFU命令は“チップ全消去”だけです。

この“チップ全消去”が受信されて正しく実行された後で、DFU命令が許され、従ってチップ上のフラッシュメモリは書き換えと照合を行うことができます。

## 6. 低位フラッシュドライブのアクセス

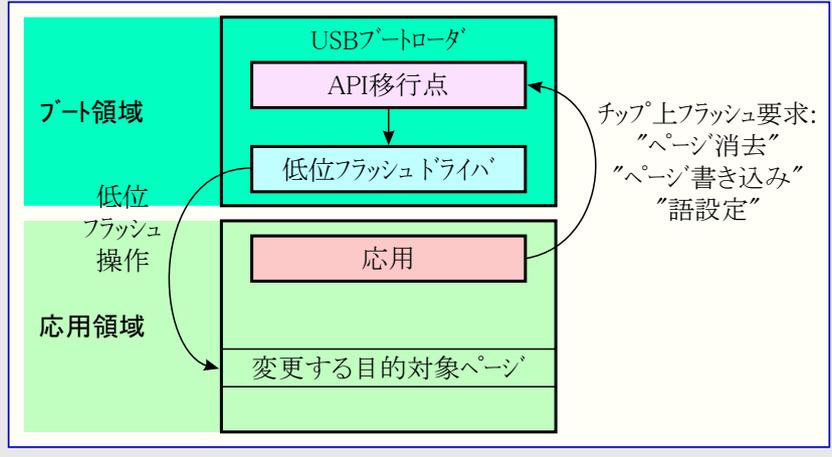
AT90USB USBブートローダはチップ上のフラッシュメモリのブート領域に配置され、同時にこのブートローダ領域はチップ上のフラッシュメモリ書き込み操作の実行が許された固有のメモリ位置です(SPM命令はこの領域でだけ復号されます)。

従ってチップ上のフラッシュメモリ書き込みを必要とする応用はUSBブートローダ内に配置された特定の移行点呼び出しを実行することができます。

USBブートローダはブート領域内に配置された低位フラッシュドライブのアクセスを応用に許す多数の応用プログラミングインターフェース(API)を提供します。これらのAPIは以下のような操作を許します。

- ・ページ消去
- ・ページ書き込み
- ・ページ一時緩衝部への話設定

図6-1. USBブートローダAPI呼び出し



APIはUSBブートローダファームウェア内の絶対アドレスに配置され、パラメータとして特定レジスタを受け入れます。これらのパラメータはCコンパイラの呼び出し規則と互換で、従って下の例のように宣言された関数ポインタで直接呼び出すことができます。

### C言語プログラム例

```
#if (FLASH_END==0x1FFFF) // 128Kバイトのデバイス
    #define LAST_BOOT_ENTRY 0xFFFE
#elif (FLASH_END==0xFFFE) // 64Kバイトのデバイス
    #define LAST_BOOT_ENTRY 0x7FFE
#else
    #error バイトでFLASH_ENDを定義しなければなりません。
#endif

// これらの関数ポインタはブートローダ内の関数移行点呼び出しに使用されます。

void (*boot_flash_page_erase_and_write)(unsigned long adr)=(void (*)(unsigned long))(LAST_BOOT_ENTRY-12);
U8 (*boot_flash_read_sig)(unsigned long adr)=(U8 (*)(unsigned long))(LAST_BOOT_ENTRY-10);
U8 (*boot_flash_read_fuse)(unsigned long adr)=(U8 (*)(unsigned long))(LAST_BOOT_ENTRY-8);
void (*boot_flash_fill_temp_buffer)(unsigned int data,unsigned int adr)=(void (*)(unsigned int, unsigned int))(LAST_BOOT_ENTRY-6);
void (*boot_flash_prg_page)(unsigned long adr)=(void (*)(unsigned long))(LAST_BOOT_ENTRY-4);
void (*boot_flash_page_erase)(unsigned long adr)=(void (*)(unsigned long))(LAST_BOOT_ENTRY-2);
void (*boot_lock_wr_bits)(unsigned char val)=(void (*)(unsigned char))(LAST_BOOT_ENTRY);

// この関数はUSBブートローダに配置されたフラッシュドライバを呼び出してフラッシュメモリ上の$1200に$55AAを書きます。
void basic_flash_access(void)
{
    unsigned long address;
    unsigned int temp16;
    temp16=0x55AA;
    address=0x12000;
    (*boot_flash_fill_temp_buffer)(temp16, address);
    (*boot_flash_page_erase)(address);
    (*boot_flash_prg_page)(address);
}
```

フラッシュAPIドライバ用の完全なアセンブリ言語コードは12ページの「[追補-B](#)」で与えられます。

## 7. 実装書き込みに対するUSBブートローダの使用

FlipソフトウェアはUSBブートローダと通信するために使用されるPC側の応用です(FlipはATMELのウェブサイトで無料で入手可能です)。

Flipとブートローダの使用についての詳細な指示に関しては「[AVR282:AT90USBに対するUSBファームウェア更新](#)」(doc7769)を参照してください。

## 8. ブートローダ履歴

以下の表は各種ブートローダ改訂と関連する変更を支援します。

表8-1. USBブートローダ履歴

製品	ブートローダ改訂	変更
AT90USB646 AT90USB647 AT90USB1286 AT90USB1287	1.0.1	初版
AT90USB82 AT90USB162	1.0.0	初版
	1.0.1	3.3V電源での16MHzクリスタルとCKDIV8ヒューズの使用を許す。
	1.0.5	USB自動速度処理を改善
ATmega16U4 ATmega32U4	1.0.0	初版

## 9. 追補-A

表9-1. ホストからのフレームの要約

命令識別子	データ(0)	データ(1)	データ(2)	データ(3)	データ(4)	説明
id_prog_start (\$01)	\$00	開始アドレス			終了アドレス	フラッシュメモリ書き込み開始
	\$01					EEPROM書き込み開始
id_display_data (\$03)	\$00	開始アドレス			終了アドレス	フラッシュメモリデータ表示
	\$01					フラッシュメモリ空き検査
	\$02					EEPROMデータ表示
id_write_command (\$04)	\$00	\$FF				チップ全消去(\$FFでのビット)
	\$03	\$00				ハードウェアリセット
		\$01	アドレス			アドレスへの無条件分岐
id_read_command (\$05)	\$00	\$00				ブートローダ版番号読み込み
		\$01				装置ブートID1読み込み
		\$02				装置ブートID2読み込み
	\$01	\$30				製造業者符号読み込み
		\$31				系列符号読み込み
		\$60				製品名読み込み
		\$61				製品改訂番号読み込み
id_change_base_address (\$06)	\$03	\$00	"PP"			"PP"64Kバイトフラッシュページ番号選択

表9-2. DFUクラス特定要求

bmRequestType	bRequest	wValue	wIndex	wLength	データ
0 01 00001	DFU_DETACH (0)	wTimeout	インターフェース (4)	0	なし
0 01 00001	DFU_DNLOAD (1)	wBlock	インターフェース (4)	データ長	ファームウェア
1 01 00001	DFU_UPLOAD (2)	wBlock	インターフェース (4)	データ長	ファームウェア
1 01 00001	DFU_GETSTATUS (3)	0	インターフェース (4)	6	状況
0 01 00001	DFU_CLRSTATUS (4)	0	インターフェース (4)	0	なし
1 01 00001	DFU_GETSTATE (5)	0	インターフェース (4)	1	状態
0 01 00001	DFU_ABORT (6)	0	インターフェース (4)	0	なし

## 10. 追補-B

```

;*****
; $RCSfile: flash_boot_drv. s90, v $
;
-----
; Copyright (c) Atmel.
-----
; 公開:      $Name:  $
; 改訂:      $Revision: 1.7 $
; FILE_CVSID: $Id: flash_boot_drv. s90, v 1.7 2005/10/03 15:50:12 $
-----
; 目的:
; このファイルはフラッシュ メモリ アクセス用の低位ドライバを含みます。
;*****
NAMEflash_drv(16)

;_____ インクルード _____
#define ASM_INCLUDE
#include "config.h"

;*****
; これは低位フラッシュ メモリドライバ用の絶対移行点表です。この表は以下のようなチップ上のフラッシュ メモリ操作を実行するために、
; 応用領域から呼び出すことができる移行点を定義します。
;
; entry_flash_page_erase_and_write:
;     R18:17:R16: ページのバイト アドレス
;
; entry_flash_fill_temp_buffer:
;     data16 : R16/R17: 一時緩衝部へ設定する語
;     address: R18/R19: 一時緩衝部内の語のアドレス
;
; entry_flash_prg_page:
;     R18:17:R16: ページのバイト アドレス
;
; entry_flash_page_erase:
;     R18:17:R16: ページのバイト アドレス
;
;*****
ASEG FLASH_END-0x0001B
entry_flash_page_erase_and_write:
    JMP flash_page_erase_and_write
entry_flash_read_sig:
    JMP flash_read_sig
entry_flash_read_fuse:
    JMP flash_read_fuse
entry_flash_fill_temp_buffer:
    JMP flash_fill_temp_buffer
entry_flash_prg_page:
    JMP flash_prg_page
entry_flash_page_erase:
    JMP flash_page_erase_public
entry_lock_wr_bits:
    JMP lock_wr_bits

```

## RSEGBOOT

```

;F*****
; 名前: flash_page_erase_and_write
;-----
; 引数: R18:17:R16: ページのバイトアドレス
;-----
; 目的: この関数は使用者応用から呼ばれます。この関数は選択した目的対象ページの
; 消去操作を実行して同じ目的対象ページのプログラミング手順を開始します。この関数は応用
; 領域で256バイトのソフトウェア一時緩衝部の保存を許します。
;*****
flash_page_erase_and_write:
    PUSH    R18
    RCALL   flash_page_erase
    POP     R18
    RCALL   flash_prg_page
    RET

;F*****
; 名前: flash_prg_page
;-----
; 引数: R18:17:R16: ページのバイトアドレス
;-----
; 目的: 目的対象ページのプログラミング手順開始
;*****
flash_prg_page:
    RCALL   WAIT_SPMEN           ;SPMENフラグ解除(0)待機
    MOV     R31, R17             ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30, R16
    OUT     RAMPZ, R18
    LDI     R20, $05             ;(1<<PGWRT) + (1<<SPMEN))
    OUT     SPMCSR, R20          ;ページ書き込み指定
    SPM     ;ページ書き込み開始
    RCALL   WAIT_SPMEN          ;ページ書き込み完了待機
    RCALL   flash_rww_enable
    RET

;F*****
; 名前: flash_page_erase
;-----
; 引数: R18:17:R16: ページのバイトアドレス
;-----
; 目的: 目的対象ページの消去手順開始
;-----
; 注意: この関数は消去後にRWWSEビットを設定しません。従ってハードウェア一時緩衝部を消去しません。この関数はブートローダが
; 使用するものです。
;-----
; 必要条件:
;*****
flash_page_erase:
    RCALL   WAIT_SPMEN           ;SPMENフラグ解除(0)待機
    MOV     R31, R17             ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30, R16
    OUT     RAMPZ, R18
    LDI     R20, $03             ;(1<<PGRS) + (1<<SPMEN))
    OUT     SPMCSR, R20          ;ページ消去指定
    SPM     ;ページ消去開始
    RCALL   WAIT_SPMEN          ;ページ消去完了待機
; RCALL   flash_rww_enable      ;警告:ここを活かさないでください。生かすと、ページ緩衝部の全内容を失います。!!!
    RET

```

```

;*****
; 名前: flash_page_erase_public
;-----
; 引数: R18:17:R16: ページのバイトアドレス
;-----
; 目的: 目的対象ページの消去手順開始
;-----
; 注意: !!!!この関数は消去後にRWWSEビットを設定します。従ってハードウェア一時緩衝部を消去します。
;*****
flash_page_erase_public:
    RCALL    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
    MOV     R31,R17            ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30,R16
    OUT     RAMPZ,R18
    LDI     R20,$03           ;(1<<PGERS) + (1<<SPMEN))
    OUT     SPMCSR,R20        ;ページ消去指定
    SPM                                ;ページ消去開始
    RCALL    WAIT_SPMEN        ;ページ消去完了待機
    RCALL    flash_rww_enable
    RET

;*****
; 名前: flash_rww_enable
;-----
; 引数: なし
;-----
; 目的: RWWSEビット設定。フラッシュメモリのプログラミング(消去または書き込み)後の応用領域でのコード実行を許します。
;*****
flash_rww_enable:
    RCALL    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
    LDI     R20,$11           ;(1<<WWSRE) + (1<<SPMEN))
    OUT     SPMCSR,R20        ;RWW(応用)領域許可指定
    SPM                                ;RWW(応用)領域許可実行
    RJMP    WAIT_SPMEN        ;RWW(応用)領域許可完了待機

;*****
; 名前: flash_read_sig
;-----
; 引数: 識票列のバイトアドレス
; 戻り値: R16: 識票値
;-----
; 目的: ハードウェア識票バイト読み込み。バイトは引数として渡されるアドレスを通して選択されます(製品のデータシートをご覧ください)。
;*****
flash_read_sig:
    RCALL    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
    MOV     R31,R17            ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30,R16
    OUT     RAMPZ,R18
    LDI     R20,$21           ;(1<<SPMEN) | (1<<SIGRD))
    OUT     SPMCSR,R20        ;識票読み込み指定
    LPM                                ;識票読み込み実行
    MOV     R16,R0            ;戻り値格納(1バイト⇒R16レジスタ)
    RJMP    WAIT_SPMEN        ;SPMENフラグ解除(0)待機

```

```

;*****
; 名前: flash_read_fuse
;-----
; 戻り値: R16: ヒューズ値
;-----
; 目的: ヒューズバイト読み込み。ヒューズバイトは引数として渡されるアドレスを通して選択されます(アドレス値については製品のデータシートをご覧ください)。
;*****
flash_read_fuse:
    RCALL    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
    MOV     R31, R17           ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30, R16
    OUT     RAMPZ, R18
    LDI     R20, $09           ;(1<<SPMEN) | (1<<BLBSET))
    OUT     SPMCSR, R20        ;フート施錠ビット読み込み指定
    LPM     ;フート施錠ビット読み込み実行
    MOV     R16, R0           ;戻り値格納(1バイト⇒R16レジスタ)
    RJMP    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
;*****
; 名前: flash_fill_temp_buffer
;-----
; 引数:
; data16 : R16/R17: 一時緩衝部に設定されるべき語
; address: R18/R19: 語のアドレス
; 戻り値: なし
;-----
; 目的: この関数はフラッシュメモリ一時緩衝部への語設定を許します。
;-----
; 例:* fill_temp_buffer(data16, address);
;-----
; 注意: 最初の引数はR17:R16を使用します。2つ目の引数はR19:R18を使用します。
;*****
flash_fill_temp_buffer:
    MOV     R31, R19           ;Z位置指示子にアドレス複写(R31=ZH,R30=ZL)
    MOV     R30, R18
    MOV     R0, R17           ;R1:R0に語複写
    MOV     R1, R16
    LDI     R20, (1<<SPMEN)
    OUT     SPMCSR, R20        ;一時緩衝部語設定指定
    SPM     ;一時緩衝部語設定実行
    RJMP    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
;*****
; 名前: lock_wr_bits
;-----
; 引数: R16: 書き込み値
;-----
; 目的:
;*****
lock_wr_bits:
    RCALL    WAIT_SPMEN        ;SPMENフラグ解除(0)待機
    MOV     R0, R16
    LDI     R18, ((1<<BLBSET) | (1<<SPMEN))
    OUT     SPMCSR, R18        ;フート施錠ビット書き込み指定
    SPM     ;フート施錠ビット書き込み開始
    RJMP    WAIT_SPMEN        ;フート施錠ビット書き込み完了待機

```

```

;*****
; 名前: wait_spmen
;-----
; 引数: なし
;-----
; 目的: 活動完了(SPMEN=0)待機
;*****
WAIT_SPMEN:
    MOV     R0, R18
    IN      R18, SPMCSR      ;SPMCSR値取得
    SBRC   R18, SPMEN
    RJMP   WAIT_SPMEN      ;SPMENフラグ解除(0)まで待機継続
;
    MOV     R18, R0
    RET
END

```

## 11. 資料改訂履歴

- 11.1. 7618B-03/08    1. 資料を全体を通してDFU機能記述子への参照を削除
- 11.2. 7618C-07/08    1. AT90USB82/162,AT90USB64x,ATmega16U4/32U4用に更新  
                       2. フォトリソダ改訂履歴を更新



## 本社

### *Atmel Corporation*

2325 Orchard Parkway  
San Jose, CA 95131  
USA  
TEL 1(408) 441-0311  
FAX 1(408) 487-2600

## 国外営業拠点

### *Atmel Asia*

Unit 1-5 & 16, 19/F  
BEA Tower, Millennium City 5  
418 Kwun Tong Road  
Kwun Tong, Kowloon  
Hong Kong  
TEL (852) 2245-6100  
FAX (852) 2722-1369

### *Atmel Europe*

Le Krebs  
8, Rue Jean-Pierre Timbaud  
BP 309  
78054 Saint-Quentin-en-  
Yvelines Cedex  
France  
TEL (33) 1-30-60-70-00  
FAX (33) 1-30-60-71-11

### *Atmel Japan*

104-0033 東京都中央区  
新川1-24-8  
東熱新川ビル 9F  
アトメル ジャパン株式会社  
TEL (81) 03-3523-3551  
FAX (81) 03-3523-7581

## 製品窓口

### ウェブサイト

[www.atmel.com](http://www.atmel.com)

### 技術支援

[avr@atmel.com](mailto:avr@atmel.com)

### 販売窓口

[www.atmel.com/contacts](http://www.atmel.com/contacts)

### 文献請求

[www.atmel.com/literature](http://www.atmel.com/literature)

お断り: 本資料内の情報はATMEL製品と関連して提供されています。本資料またはATMEL製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。ATMELのウェブサイトに位置する販売の条件とATMELの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、ATMELはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえATMELがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益の損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してATMELに責任がないでしょう。ATMELは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。ATMELはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、ATMEL製品は車載応用に対して適当ではなく、使用されるべきではありません。ATMEL製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

© Atmel Corporation 2008. 全権利予約済 ATMEL®、ロゴとそれらの組み合わせ、AVR®、STK®とその他はATMEL Corporationの登録商標または商標またはその付属物です。他の用語と製品名は一般的に他の商標です。

### © HERO 2014.

本応用記述はATMELのDFUデータシート(doc7618.pdf Rev.7618C-07/08)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。