

aZealNET

AZ9360SDK Customize Guide

1. 本資料は、お客様が AZ9360SDK をご使用いただくための資料であり、本資料中に記載の技術情報について株式会社ワイ・デー・ケー（以下、「YDK」と称す）が所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載のデータ、図、表、プログラム、その他の使用に起因する損害、第三者所有の権利に対する侵害に関し、YDKは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、その他全ての情報は本資料発行時点のものであり、YDKは、予告なしに、本資料に記載した内容を変更することがあります。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したのですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、YDKはその責任を負いません。
5. 本資料に記載のデータ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。YDKは、適用可否に対する責任を負いません。
6. AZ9360SDK は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。AZ9360SDK を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際は、YDK、または、販売代理店へご照会ください。
7. 本資料の全部、または、一部を転載、複製する場合、YDKの事前の承諾が必要です。

AZ9360SDKは、「非営利特別法人TOPPERSプロジェクト」の開発成果物である、TOPPERS/JSPカーネルを使用しています。

ご使用にあたっては、以下に示す、TOPPERSライセンス、および、株式会社ワイ・デー・ケーが設定する使用許諾件に従ったご使用をお願いします。

尚、株式会社ワイ・デー・ケーが設定する使用許諾件では、複製、再配布を禁止しておりますので、ご使用にあたっては、TOPPERSライセンスの(3)(a)、または(3)(b)を適用してください。

TOPPERSライセンス

TOPPERS/JSP Kernel

Toyohashi Open Platform for Embedded Real-Time Systems/
Just Standard Profile Kernel

Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
Toyohashi Univ. of Technology, JAPAN

上記著作権者は、以下の(1)~(4)の条件が、Free Software Foundationによって公表されているGNU General Public LicenseのVersion 2に記載されている条件を満たす場合に限り、本ソフトウェア(本ソフトウェアを改変したものを含む、以下同じ)を使用・複製・改変・再配布(以下、利用と呼ぶ)することを無償で許諾する。

- (1) 本ソフトウェアをソースコードの形で利用する場合には、上記の著作権表示、この利用条件および下記の無保証規定が、そのままの形でソースコード中に含まれていること。
- (2) 本ソフトウェアを、ライブラリ形式など、他のソフトウェア開発に使用できる形で再配布する場合には、再配布に伴うドキュメント(利用者マニュアルなど)に、上記の著作権表示、この利用条件および下記の無保証規定を掲載すること。
- (3) 本ソフトウェアを、機器に組み込むなど、他のソフトウェア開発に使用できない形で再配布する場合には、次のいずれかの条件を満たすこと。
 - (a) 再配布に伴うドキュメント(利用者マニュアルなど)に、上記の著作権表示、この利用条件および下記の無保証規定を掲載すること。
 - (b) 再配布の形態を、別に定める方法によって、TOPPERSプロジェクトに報告すること。
- (4) 本ソフトウェアの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者およびTOPPERSプロジェクトを免責すること。

本ソフトウェアは、無保証で提供されているものである。上記著作権者およびTOPPERSプロジェクトは、本ソフトウェアに関して、その適用可能性も含めて、いかなる保証も行わない。また、本ソフトウェアの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わない。

1 . はじめに.....	5
2 . ガイド.....	6
2.1 割り込みハンドラ追加 / 削除.....	6
2.1.1 割り込みハンドラの追加.....	6
2.2 メモリマップ変更.....	6
2.3 G P I O変更.....	6
2.4 FlashROM変更.....	7
2.5 PHY変更.....	7
2.6 U A R T通信条件.....	7
2.7 T C P / I Pスタック.....	8
2.7.1 上位レイヤの実装.....	8
2.7.2 N I C枚数とI Pアドレス保持数.....	9
2.7.3 F T Pログインパスワードの変更.....	10
2.7.4 F T Pパッシブモードの禁止 / 許可切り替え.....	10
2.7.5 H T T P D関連のパラメータ.....	10
2.8 F i l e S y s t e m関連.....	13
2.8.1 ファイルの説明.....	13
2.8.2 E F _ U S E R . cの説明.....	13
2.8.3 注意事項.....	14

1 . はじめに

本書は、Net Silicon NS9360 Net+Worksの環境でITRONの実現を行うSDKのカスタマイズ方法、手順について記します。

1.1 本書では、以下の項目についてガイドします。

- 割り込みハンドラ追加 / 削除
- メモリマップ変更
- G P I O 変更
- FlashROM 変更
- PHY 変更
- U A R T 通信条件
- T C P / I P スタック
- F i l e S y s t e m 関連

2. ガイド

2.1 割り込みハンドラ追加 / 削除

2.1.1 割り込みハンドラの追加

割り込みハンドラを追加する場合は、以下の手順で行います。

割り込みハンドラルーチンを作成

Toppers/JSP のコンフィグレーションファイルに、DEF_INH 宣言を追加する。
例えば、Int_Isr() を登録する場合は、xxxx.cfg に

```
DEF_INH( INHNO_ISR, { TA_HLNG, Int_Isr } );
```

と記述します。

・・・/jsp/ config/armv4/ NS9360EVA/にある sys_config.c で定義している、int_id_table[] に割り込みハンドラ番号を登録します。

int_id_table[] は、IRQ No の順に並んでいるので、該当する IRQ の位置に割り込みハンドラ番号を登録します。

この割り込みハンドラ番号(Interrupt ID)は、ns9360.h に定義してあります。

IRQ No の定義

ns9360.h に IRQ No のシンボルを定義します。
これは静的 API の DEF_INH 定義に使用します。
上記 の例にある " INHNO_ISR " がそれです。

2.2 メモリマップ変更

SDK では、Flash ROM を CS1 に、SDRAM を CS4 にマッピングしています。

SDRAM の追加、FPGA などの I/O 空間を新たに追加する場合は、CSn を定義する必要があります。

CSn の設定は、・・・/jsp/ config/armv4/にある cpu_config.c で定義している、cpuSetupCSx() を変更してください。

SDK で作成するプログラムは、Net + OS での image.bin に相当するので、CS1 と CS4 は、Bootloader で設定する。したがって SDK 側の CPU 初期化処理には不要です。

CS1 と CS4 の変更が必要な場合は、Net + OS の BSP を変更し、ROM.bin を UP する必要があります。

現状は、cpuSetupCS1() と cpuSetupCS4() 以外の関数は、中身が空になっています。

関数の中は、各 CS に対応したレジスタの設定になります。

詳しくは、ns9360 の Hardware Reference を参照願います。

2.3 GPIO 変更

GPIO は、H/W 構成によって変更が必要になります。

GPIO の設定を変更する場合は以下の手順で行います。

Control Registers の設定

出力ポートとして使用する場合の初期値を設定します。

(Control Registers#1~3)

・・・/jsp/ config/armv4/NS9360EVA/にある sys_user.c にて、BbusGpio_ContSts1~3 が定義してありますので、この変数に、該当する BBUS_GPIO_CONT_STS_n_GPIO_{xx} を OR 設定してください。

BBUS_GPIO_CONT_STS_n_GPIO_{xx} は、NS9360.h に定義してあります。

Configuration Registers の設定

・・・/jsp/ config/armv4/NS9360EVA/にある NS9360.h にて、各 GPIO のファンクションを

設定します。

G P I Oのファンクションは、BBUS_GPIO_CONFIGn_GPIOxx にそれぞれ該当 4 bit を定義します。
設定する 4 bit の詳細は、ns9360 の Hardware Reference を参照願います。

2.4 FlashROM 変更

F l a s h R O Mを変更した場合は、F l a s h R O Mドライバ処理の変更が必要です。
F l a s h R O Mドライバ処理は、・・・/TCP/の下にある CPU_I0.c に定義してあります。

- FROM_EraseSector()
- FROM_WriteData()

また、関連シンボルが、CPU_I0.h に定義してありますので、そちらの変更も必要です。

```
#define FROM_TOP          0x50000000          /* FLASH ROM Top address */
#define FROM_BTM          0x50400000          /* FLASH ROM end address +1 */

#define FROM_CHIPSIZE     0x00800000          /* 8MB @ chip( 4MB para ) */
#define FROM_SECTOR      0x00020000          /* 32KB/block(16KB para) */

#define FROM_WRITETRY     ((CPU_CLK / 1000000) * 550) /* Aprox. 550us */
#define FROM_WRITETRY_X   (((CPU_CLK / 1000000) * 550) * 10000) /* Aprox. 550us * 100 */
#define FROM_ERASETRY     (FROM_WRITETRY * 27273) /* Aprox. 15 sec 27273 : (15s/550us) */

#define FROM_BOOT_TOP     (FROM_TOP + 0)      /* BOOT Top */
#define FROM_APL_TOP      (FROM_TOP + 0x40000) /* APL Top */
```

2.5 PHY 変更

E t h e r n e t L A Nの P H Yを変更した場合は、N I Cドライバ処理の変更が必要です。
N I Cドライバ処理は、・・・/TCP/の下にある PS_NIC.c に定義してあります。
また、関連シンボルが、PS_NIC.h に定義してありますので、そちらの変更も必要です。
使用する P H Yに合わせて、変更してください。

2.6 U A R T通信条件

U A R Tの通信条件を変更する場合は、・・・/jsp/ config/armv4/ NS9360EVA/にある ns9360.c で定義している、
• uart_init()
• uart_opn_por()
を変更してください。

2.7 TCP/IPスタック

TCP/IPスタックはysock6を使用しています。
TCP/IP関連のカスタマイズについて以下に記します。

2.7.1 上位レイヤの実装

本SDKに実装しているysock6では、以下の上位レイヤプロトコルをサポートしています。

- DHCPクライアント
- DHCPサーバー
- DNS
- FTPサーバー
- HTTPサーバー
- NTP
- TFTP
- SNMP
- SMTP
- POP3
- TELNET

また、下位レイヤの以下の機能をサポートしています。

- IPV6
- VLAN
- IPSEC

(1) プロトコル別 Enable フラグ

各プロトコルの実装は、スイッチで切り替えることが可能です。

スイッチの定義は、・・・/TCP/の下にあるPS_USER.cで定義されている「プロトコル別 Enable フラグ」を変更してください。

スイッチは、0で未実装、1で実装となります。

```
/*
 * プロトコル別 Enable フラグ
 */
const int    IPV6_ENABLE      = 0;    /* IPv6   : なし      */
const int    VLAN_ENABLE     = 0;    /* VLAN   : なし      */
const int    IPSEC_ENABLE     = 0;    /* IPSEC  : なし      */

const int    DHCP_C_ENABLE    = 0;    /* DHCP-c : なし      */
const int    DHCP_S_ENABLE    = 0;    /* DHCP-s : なし      */
const int    DNS_ENABLE       = 0;    /* DNS    : なし      */
const int    FTPD_ENABLE      = 1;    /* FTP-d  : あり      */
const int    HTTPD_ENABLE     = 1;    /* HTTP-d : あり      */
const int    NTP_ENABLE       = 0;    /* NTP    : なし      */
const int    TFTP_ENABLE      = 0;    /* TFTP   : なし      */
const int    SNMP_ENABLE      = 0;    /* SNMP   : なし      */
const int    SMTP_ENABLE      = 0;    /* SMTP   : なし      */
const int    POP3_ENABLE      = 0;    /* POP3   : なし      */

const int    TELNET_ENABLE    = 1;    /* TELNET : あり      */
```

(2) 静的APIの追加

各プロトコルは、タスクとしてOSに登録する必要があります。

Toppers / JSPでは、タスクの登録は、静的APIの "CRE_TSK" を使用します。

CRE_TSKの宣言は、コンフィグレーション・ファイルに

```
CRE_TSK( ID tskid, {ATR tskatr, VP_INT exinf, FP task, PRI itskpri,
                SIZE stksz, VP stk} );
```

と記述します。

サンプルの場合は、・・・/itron/nsl_itron/の下にある nsl_itron.cfg に定義します。

```
CRE_TSK( TASKID_HTTPD, { TA_HLNG           , 0, HTTPD_Main,  PRI_HTTPD ,STACK_HTTPD, NULL });
CRE_TSK( TASKID_HTTPD00,{ TA_HLNG           , 0, HTTPD_Child, PRI_HTTPD ,STACK_HTTPD, NULL });
CRE_TSK( TASKID_HTTPD01,{ TA_HLNG           , 0, HTTPD_Child, PRI_HTTPD ,STACK_HTTPD, NULL });
CRE_TSK( TASKID_HTTPD02,{ TA_HLNG           , 0, HTTPD_Child, PRI_HTTPD ,STACK_HTTPD, NULL });
CRE_TSK( TASKID_HTTPD03,{ TA_HLNG           , 0, HTTPD_Child, PRI_HTTPD ,STACK_HTTPD, NULL });

CRE_TSK( TASKID_TELNET, { TA_HLNG           , 0, TELNET_Main, PRI_TELNET ,STACK_TELNET, NULL });
CRE_TSK( TASKID_FTPD,   { TA_HLNG           , 0, FTPD_Main,   PRI_FTPD   ,STACK_FTPD,  NULL });
CRE_TSK( TASKID_DNS,    { TA_HLNG           , 0, DNS_Main,    PRI_DNS    ,STACK_DNS,   NULL });
CRE_TSK( TASKID_DHCPD,  { TA_HLNG           , 0, DHCPD_Main,  PRI_DHCPD  ,STACK_DHCPD, NULL });
CRE_TSK( TASKID_TFTP,   { TA_HLNG           , 0, TFTP_Main,   PRI_TFTP   ,STACK_TFTP,  NULL });
CRE_TSK( TASKID_DHCPD4, { TA_HLNG           , 0, DHCPD4_Main, PRI_DHCPD4 ,STACK_DHCPD4, NULL });
CRE_TSK( TASKID_NTP,    { TA_HLNG           , 0, NTP_Main,    PRI_NTP    ,STACK_NTP,   NULL });
CRE_TSK( TASKID_SNMP,   { TA_HLNG           , 0, SNMP_Main,   PRI_SNMP   ,STACK_SNMP,  NULL });
```

2.7.2 NIC枚数とIPアドレス保持数

ysock6では、複数のNICと複数のIPアドレスを使用できるようになっています。
(それぞれ最大10件まで指定可能です。)

SDKでは、NIC = 1枚、IPアドレス保持数 = 1の設定になっています。

これらの設定を変更する場合は、・・・/TCP/の下にあるPS_USER.cで定義されている「NIC&IP
アドレス数」を変更してください。

```
const int    MAX_NIC = 1;           /* NIC枚数 (設定範囲: 1-10) */
const int    MAX_IPADRS = 1;       /* IPアドレス保持数 (設定範囲: 1-10) */
```

尚、MAX_NIC および MAX_IPADRS の設定は、最大10件までとします。

2.7.3 FTPログインパスワードの変更

FTP - dを実装した際にFTPサーバーへログインするためのユーザー名とパスワード名は変更が可能です。

ユーザー名とパスワード名の定義は、・・・/TCP/の下にあるPS_USER.cで定義されている「FTPユーザ/パスワード登録テーブル」を変更してください。（追加も可能です）

```
/*
   FTP ユーザ/パスワード登録テーブル
*/
const FTPD_USER_TBLE   FtpdUserTable[] = {
/*--  USER 名,      Password,   ドライブ名,カレントディレクトリ  --*/
  { "anonymous", "",           "A:",      "/" }, /* anonymous                               */
  { "user",      "pass",       "A:",      "/" }, /* user/password/drive/login-dir          */
  { "ySOCK",    "0423788511",  "A:",      "/" } /* user/password/drive/login-dir          */
};
```

また、「FTPユーザ/パスワード登録テーブル」に設定している、ドライブ名、カレントディレクトリの設定を変更することにより、ユーザー/パスワードによる、ディレクトリのアクセスレベルを変更することも可能です。

2.7.4 FTPパッシブモードの禁止/許可切り替え

FTP - dを実装した際にFTPパッシブモードの禁止/許可を変更できます。

パッシブモードの禁止/許可の切り替えは、・・・/TCP/の下にあるPS_USER.cで定義されている「PASVコマンドの使用」を変更してください。

設定は、1で許可、0で禁止となります。

```
const   FTPD_PasvCmd_Enable = 0;           /* PASV コマンドの使用 1=許可, 0=禁止 */
```

2.7.5 HTTPD関連のパラメータ

(1) HTTPDで稼働させる擬似CGI用関数とそのCGIファイル名

HTTPDから実行させるCGIの関数名を登録できます。

CGIの関数名の登録は、・・・/TCP/の下にあるPS_USER.cで定義されている「HTTP-d関連パラメータ」を変更してください。

```
const   CGITBL   HTTP_cgi00[] = {           /* lpIndex=0          */
  { (CGIFNC)&CGI_aaa,      (U8*)"cgi/aaa.cgi" }, /* cgi-function & CGI request name */
  { (CGIFNC)&CGI_bbb,      (U8*)"cgi/bbb.cgi" },
  { (CGIFNC)&CGI_testbin,  (U8*)"test.bin"   },
  { 0, 0 }                 /* stopper            */
};
```

(2) P U T / D E L E T E の実行を許可するユーザ名 / パスワード

P U T / D E L E T E の実行を許可するユーザ名 / パスワードを登録することが可能です。

ユーザ名 / パスワード登録は、・・・ / T C P / の下にある P S _ U S E R . c で定義されている「 P U T / D E L E T E の実行を許可するユーザ名 : パスワード」を変更してください。

```
const  U8*      HTTP_AcceptUser00[] = {      /* PUT/DELETE accept user:password */
        (U8*)"ydk-tec:1234",
        (U8*)"soft:999",
        0                                     /* stopper */
    };
```

(3) P U T / D E L E T E の実行の禁止

P U T / D E L E T E の実行を許可 / 禁止を設定することが可能です。

許可 / 禁止の設定は、・・・ / T C P / の下にある P S _ U S E R . c で定義されている「 P U T / D E L E T E の実行を禁止するファイル、ディレクトリ」を変更してください。

```
const  U8*      HTTP_RejectFile00[] = {      /* PUT/DELETE reject file name */
        (U8*)"index.html",                  /* index.html は PUT/DELETE 禁止 */
        (U8*)"secret/*",
        0                                     /* stopper */
    };
```

(4) H T T P D の動作パラメータ

H T T P D の動作パラメータ (デフォルトのファイル名や作業用ファイルのディレクトリ等) が変更可能です。

H T T P D の動作パラメータの変更は、・・・ / T C P / の下にある P S _ U S E R . c で定義されている「 P U T / D E L E T E の実行を禁止するファイル、ディレクトリ」を変更してください。

```
const  HTTP_CTRL HTTP_CtITbl[] = {
    { 0,                                     /* 稼動 ipindex 値 MyIpTbl[0]の自 IP */
      PORT_WWW,                             /* Listen するポート番号 */
      (U16*)HTTP_id_Tbl00,                  /* 動作させる HTTP 子タスク id テーブル */
      (CGITBL*)HTTP_cgi00,                 /* 擬似 CGI エントリーテーブル */
      (U8*)"A:/",                          /* 動作させるディスクのディレクトリ */
      (U8*)"temp",                         /* 作業用ファイルのディレクトリ */
      (U8**)HTTP_AcceptUser00,            /* PUT/DELETE 許可ユーザ名リスト */
      (U8**)HTTP_RejectFile00,           /* PUT/DELETE 禁止ファイルリスト */
      (U8*)"index.html"                   /* ファイル指定が無いときの読出 file 名 */
    },
    { 0xff,                                 /* table stopper */
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0
    }
};
```

(5) 同時接続クライアントの変更

デフォルトでは、2クライアントとの同時接続を可能としています。
同時接続クライアント数を増やす場合は、HTTP 子タスクの数を追加することで対応可能です。
HTTP 子タスクを追加するには、以下の修正が必要です。

CRE_TSK の追加

追加する HTTP 子タスク用に、コンフィグレーションファイルに CRE_TSK を追加する必要があります。このとき、タスク ID 以外は同一の情報を設定してください。

```
CRE_TSK( TASKID_HTTPDxx, { TA_HLNG           , 0, HTTPD_Child, PRI_HTTPD   ,STACK_HTTPD, NULL } );  
xx : 05 から順に ID 名称を決めてください。
```

id 登録テーブルの追加

HTTP 子タスクのタスク ID を設定するテーブルがあります。このテーブルに追加したタスク ID を追加設定してください。

テーブルの実体は、・・・/TCP/の下にある PS_USER.c で定義されている「HTTP 子タスク ID 登録テーブル」です。

```
const    U16      HTTP_id_Tbl00[] = {  
        TASKID_HTTPD00,  
        TASKID_HTTPD01,  
        TASKID_HTTPD02,  
        TASKID_HTTPD03,  
        0 } ;
```

2.8 File System関連

File Systemのカスタマイズのためのファイルは、フォルダ EFFSLIB に格納されています。

2.8.1 ファイルの説明

EFFSFAT.a : FAT 互換ファイルシステムのライブラリ
VFILESYS.h : ファイルシステムを使用するためのヘッダファイル
EF_USER.c : ファイルシステムカスタマイズ用ファイル

上記のファイルのうち、カスタマイズが必要なファイルは、EF_USER.c です。

2.8.2 EF_USER.c の説明

(1) F_MAXVOLUME

ファイルシステムにて使用可能なドライブ数
この値は変更しないでください。

(2) Efs_SemUse, Efs_SemId

ファイルシステム内で使用するセマフォの管理用データ
このデータは変更しないでください。

(3) RAM ディスク関連

RAM ドライブの容量宣言(RAM_DRIVE_0_SIZE, RAM_DRIVE_1_SIZE)

サンプルでは、2つのRAMドライブを使用できるようになっています。
各ドライブの容量は以下のようになっています。

RAM ドライブ0 : 2MByte

RAM ドライブ1 : 128KByte

2ドライブの場合はこの値を変更することで各ドライブの容量を変更することができます。

RAM ドライブ用エリア定義(ramdrv0, ramdrv1)

サンプルでは、2つのRAMドライブを使用できるようになっています。

RAM ディスク制御テーブル(RamDrv)

RAM ドライブの数を変更するときは、このテーブルを追加・削除してください。
このテーブルは以下の構造体になっています。

```
typedef struct {  
    char *ramdrv;  
    unsigned long maxsector;  
    int use;  
} t_RamDrv;
```

ramdrv : RAM ドライブ用エリアへのポインタ

maxsector : RAM ドライブのセクタ数

RAM ドライブのサイズ/SECTOR_SIZE で指定

use : 使用中フラグ

ファイルシステム内部で使用、宣言時は0

RAM ディスクドライブ数(RAMDRV_CNT)

ファイルシステム内でドライブの数を認識するためのデータ。
変更する必要はありません。

RAM ドライブ容量(RAMDRVO_SIZE)

az_sys.hにてRAM_SYNC_USEが1に設定されている場合、CPU_IO.cにてFlashROMとRAM
ドライブ0の間でデータのコピーを行います。

このときに、RAMディスクの容量を取得するために使用しています。

RAMドライブの容量宣言で宣言したRAM_DRIVE_0_SIZEをそのまま使用します。

(4) MMC 関連

MMC ディスク制御テーブル(MmcDrv)

複数の MMC ポートを使用する場合、このテーブルを追加します。
このテーブルは以下の構造体になっています。

```
typedef void (*PORT_INI)( );
typedef void (*CS_LOW)( );
typedef void (*CS_HIGH)( );
typedef int (*CD_READ)( );
typedef int (*WP_READ)( );
typedef struct {
    unsigned long ser_base;      /* SPI serial base */
    int use;                    /* actual entry in use? */
    int active;                 /* actual entry is active? */
    int initok;                 /* successfully initialized */
    int sd;                     /* actual card is sd or mmc */
    CS_LOW cs_low;              /* Function of CS Low */
    CS_HIGH cs_high;            /* Function of CS High */
    CD_READ cd_read;            /* Function of CD read */
    WP_READ wp_read;            /* Function of WP read */
    PORT_INI port_initial;      /* Function of I/O port initialize */
} t_MmcDrv;

ser_base      : SPI コントローラのベースアドレス
use           : ファイルシステムにて使用 (0 を設定)
active        : ファイルシステムにて使用 (0 を設定)
initok        : ファイルシステムにて使用 (0 を設定)
sd            : ファイルシステムにて使用 (0 を設定)
cs_low;       : CS を Low にする関数
cs_high;      : CS を High にする関数
cd_read;      : CD を入力する関数 (CD が Low の場合 0 を返す)
wp_read;      : WP を入力する関数 (WP が Low の場合 0 を返す)
port_initial; : GPIO 初期化関数
```

cs_low, cs_high, cd_read, wp_read, port_initial の各関数は、ターゲットボードに合わせて、作成する必要があります。

バスクロックの設定(EFS_BbusClk)

SPI の bit レートを設定するために使用する情報。

NS9360/9750 では、シリアルコントローラに入力されるクロックは、CPU クロックの 1/4 であるため、CPU クロック/4 を指定。

内蔵シリアルコントローラ以外は未サポートですので、その場合はファイルシステムのソースコードが必要になります。

2.8.3 注意事項

(1) RAM ドライブ 0 の初期化

az_sys.h にて RAM_DISK_USE が 1 に設定されている場合、スタートアップ時に RAM ドライブ 0 のボリュームの初期化が実施されます。

また、az_sys.h にて RAM_SYNC_USE が 1 に設定されている場、スタートアップ時に CPU_I0.c にて FlashROM のデータを RAM ドライブ 0 の RAM エリアに書き込み、その後 RAM ドライブ 0 の初期化を行います。

このため、ユーザープログラムにて RAM ドライブ 0 の初期化(VFS_InitRamDrive)を実行するとエラーが返されます。