

# aZealNET

## AZ9360SDK

### Make Guide

### ～KIT 編～

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、慎重に制作したものです。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には、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。

1.	はじめに.....	4
2.	SDKの概要.....	4
2.1	SDKの位置づけ.....	4
3.	インストール.....	4
4.	開発環境.....	5
4.1	フォルダ構成.....	5
4.2	ビルド.....	7
4.2.1	サンプルプログラムの内容.....	7
4.2.2	ビルドの実行.....	8
4.2.3	ブートローダーのビルド.....	10
4.3	デバッガ起動.....	11
4.3.1	RAVEN.....	11
5.	タスクの追加.....	14

## 1. はじめに

本書は、AZ9360MB上で動作するITRON環境の実現を行うAZ9360SDKの開発環境および開発手順について記します。

## 2. SDKの概要

### 2.1 SDKの位置づけ

本SDKは、AZ9360MB、NS9360評価ボードまたは、Connectcore9C評価ボード上で動作するITRONによる開発環境を提供します。

## 3. インストール

インストールファイルを任意のフォルダへ解凍してください。

Net+OSがインストールされている場合は、以下にインストールしてください。

- /netos63\_gnu/src/ (NET+OSがVer6.3の場合)

本書では、「c:¥」にインストールしたとして進めます。

## 4. 開発環境

開発環境（コンパイラ、デバッガ）は、KITについている開発ツールを使用します。（GNU GCC）。

### 4.1 フォルダ構成

```
c:\¥-- itron --- apps
                |
                +- jsp
                |
                +- obj
                |
                +- <CONFIG>
                |
                +- TCP
                |
                +- EFFSLIB
                |
                +- boot
                |
                +- defapl
```

- apps  
ユーザータスク用のフォルダです。  
タスクのフォルダ、ファイルをここに作成します。
- jsp  
TOPPERS/JSP 関連のファイルがここに入っています。  
※詳しくは、「TOPPERS/JSP Kernel USER'S MANUAL」を参照してください。
- <CONFIG>  
サンプルタスクのソースファイル、コンフィグレーションファイル、Makeファイル、binヘッダファイルが入っています。  
サンプルはスタートアップタスクとして利用できます。

**本フォルダ名は実際には装置名やプロジェクト名を付けます。（その名称をコンフィグレーション名といいます）**

**SDKでは、az9360、9360EVA、CC9Cになっています。**

また、フォルダ名を変更した場合は、フォルダ内のファイル（.cfg, .c, .h）のファイル名もフォルダと同じ名前にしてください。

- obj  
makeの結果、xxxx.srec、xxx.elf、xxx.mapがここに作成されます。  
ターゲットで書き込むimage.bin(defapl.bin)もここに作成されます。
- TCP  
ysockのライブラリファイル、ysockのヘッダファイル、ysockの一部のソースファイルが入っています。
- EFFSLIB  
ファイルシステム関連のファイルが入っています。

- boot  
ブートローダープログラムが入っています。  
make も本フォルダで実行します。  
make 後は、boot.bin が出来上がります。
- defapl  
user プログラムが壊れた際に使用するデフォルトアプリケーションの環境。  
SDKでは、サンプルをおいてあります。  
user アプリケーションとデフォルトアプリケーションの切り替えは、AZ9360MBのDip  
SW5を切り替えます。同SWがON場合に、デフォルトアプリケーションが立ち上がります。

## 4.2 ビルド

サンプルのタスクを使ってビルド手順について以下に説明します。

### 4.2.1 サンプルプログラムの内容

サンプルプログラムは以下の機能を実現します。

- **CSP (コンソールサービスプログラム)**  
UART 経由で接続した PC からコマンドを入力し、コマンドの実行を行う。  
ネットワークパラメータの設定やメモリの R/W などが可能です。  
評価ボード (NS9360, connectcore9c) のシリアルポート A と PC を RS 232 C ケーブルで接続してください。

- **HTTP サーバ**  
ysock の HTTP サーバが動作します。

- **FTP サーバ**  
ysock の FTP サーバが動作します。  
HTTP サーバで使用する、HTML ファイルをアップロードできます。

ユーザー名 : "user"  
パスワード : "pass"

- **Filesystem**  
yfile の Filesystem が動作します。

- **TELNET**  
ysock の TELNET が動作します。  
TELNET 経由で CSP の実行が出来ます。

#### 4.2.2 ビルドの実行

- ① デスクトップにある、“Macraigor Cygwin Shell” ショートカットをダブルクリックし、C Y G W I N を起動します。



← クリック！

```
USER@USER~  
$
```

- ② カレントディレクトリを “. /itron/<CONFIG>” へ移動します。

```
USER@USER~  
$ cd /cygdrive/c/itron/az9360  
  
USER@USER/cygdrive/c/itron/az9360  
$
```

- ③ “make. sh”を実行します。

```
USER@USER/cygdrive/c/itron/az9360
$ bash make. sh
/cygdrive/c/itron
configure: Generating Makefile from /cygdrive/c/itron/az9360/Makefile.
configure: Makefile exists. Save as Makefile.bak.
configure: Generating az9360.c from /cygdrive/c/itron/az9360/az9360.c.
configure: Generating az9360.h from /cygdrive/c/itron/az9360/az9360.h.
configure: Generating az9360.cfg from /cygdrive/c/itron/az9360/az9360.cfg.
make: Entering directory `/cygdrive/c/itron/jsp/obj'
rm -f Makefile.depend
arm-elf-gcc -S -mcpu=arm9tdmi -Wa -mbig-endian -O2 -DAZ9360MB -DRAVEN -I. -
-I../include -I../config/armv4/AZ9360MB -I../config/armv4 -I../TCP -I../ap
ps -I../EFFSLIB -I../az9360 -I../kernel ../config/armv4/makeoffset.c
../utils/genoffset makeoffset.s > tmpfile3
mv tmpfile3 offset.h
arm-elf-gcc -E -I. -I../include -I../config/armv4/AZ9360MB -I../config/armv4 -I
../TCP -I../apps -I../EFFSLIB -I../az9360 -DAZ9360MB -DRAVEN -x c-h
eader az9360.cfg > tmpfile1
../cfg/cfg -s tmpfile1 -c -obj -cpu armv4 -system AZ9360MB
rm -f tmpfile1
Generating Makefile.depend.
:
:
:
-I../EFFSLIB -I../az9360 -nostdlib -Wl,-Map,jsp.map -mcpu=arm9tdmi -N -
mbig-endian -Wl,-Ttext,0x0000003c -Wl,-Tdata,0x000B0000 -T ../config/armv4/AZ93
60MB/ns9360.ld -o jsp.elf ¥
        start.o az9360.o CPU_IO.o PS_TELNET.o HTTPD_CGI.o CSP.o PS_N
IC.o mib_if.o mib_com.o mib_Priv.o mib_Sys.o mib_table.o PS_USER.o halt.o
        PS_SNMP_USER.o EF_USER.o PS_MAIL.o PS_CONFIG.o MNT.o timer.o serial.o logt
ask.o log_output.o vasyslog.o t_perror.o strerror.o kernel_cfg.o ../TCP/ysoc
k.a ../EFFSLIB/EFFSFAT.a libkernel.a -lgcc
arm-elf-nm jsp.elf > jsp.syms
arm-elf-objcopy -O srec -S jsp.elf jsp.srec
arm-elf-objcopy -O binary -S jsp.elf jsp.bin
make: Leaving directory `/cygdrive/c/itron/jsp/obj'
az9360.elf ..OK
az9360.srec
az9360.map
    may be created ..

USER@USER/cygdrive/c/itron/az9360
```

ビルドの作業は以上です。  
出来上がるファイルは. . .

- /itron/obj  
"az9360.elf",  
"az9360.map",  
"az9360.srec",  
"image.bin"
- /itron/jsp/obj  
"jsp.bin",  
"jsp.elf",  
"jsp.map",  
"jsp.srec"  
各オブジェクトファイル

#### 4.2.3 ブートローダーのビルド

ブートローダー (/itron/boot) のビルドもサンプルと同様で、"boot" フォルダにて make.sh を実行します。

出来上がったバイナリは同フォルダに格納されます。

#### 4.3 デバッガ起動

サンプルのタスクを使ってデバッガ起動手順について以下に説明します。

##### 4.3.1 RAVEN

RAVENを使用した場合の手順を以下に記します。

- ① カレントディレクトリを “/itron/jsp/obj”へ移動します。

```
USER@USER/cygdrive/c/itron/az9360
$ cd ../jsp/obj

USER@USER/cygdrive/c/itron/jsp/obj
$
```

- ② ターゲット、ICEの電源を入れます。
- ③ `Ocd Remote`を起動します。  
`Ocd Remote`を起動するために、もう一つ、`Cygwin Shell`を起動します。



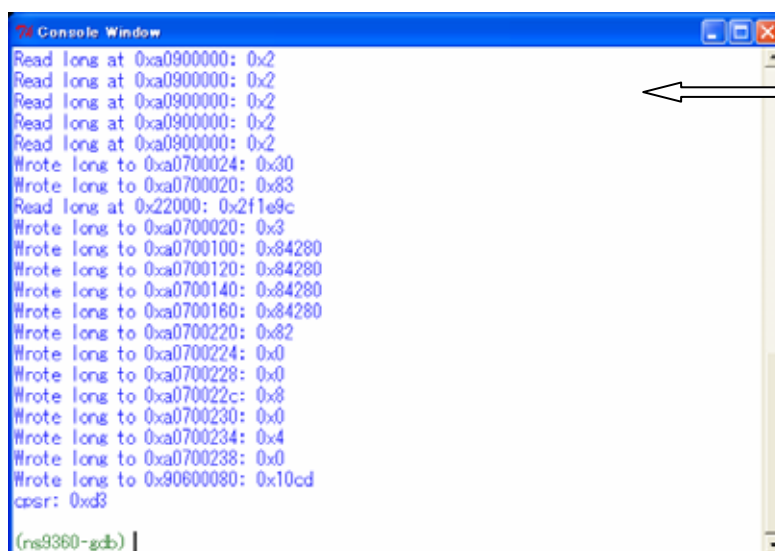
← クリック！

Cygwin Shellにて、以下のコマンドを実行します。

```
USER@USER~
$ ocdremote --cpu NS9360
ocdremote 2.14: RAVEN via LPT 1 at speed : 1
JTAG SDO <-| CPU(1) NS9360 : listening on port 8888 |<- JTAG SDI
CPU[1] Accepted gdb connection on port 8888.
```

- ④ デバッガを起動します。

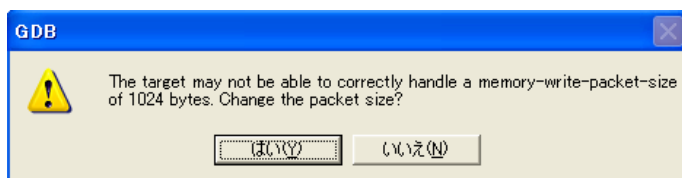
```
USER@USER/cygdrive/c/itron/jsp/obj  
$ arm-elf-insight -se=jsp.elf
```



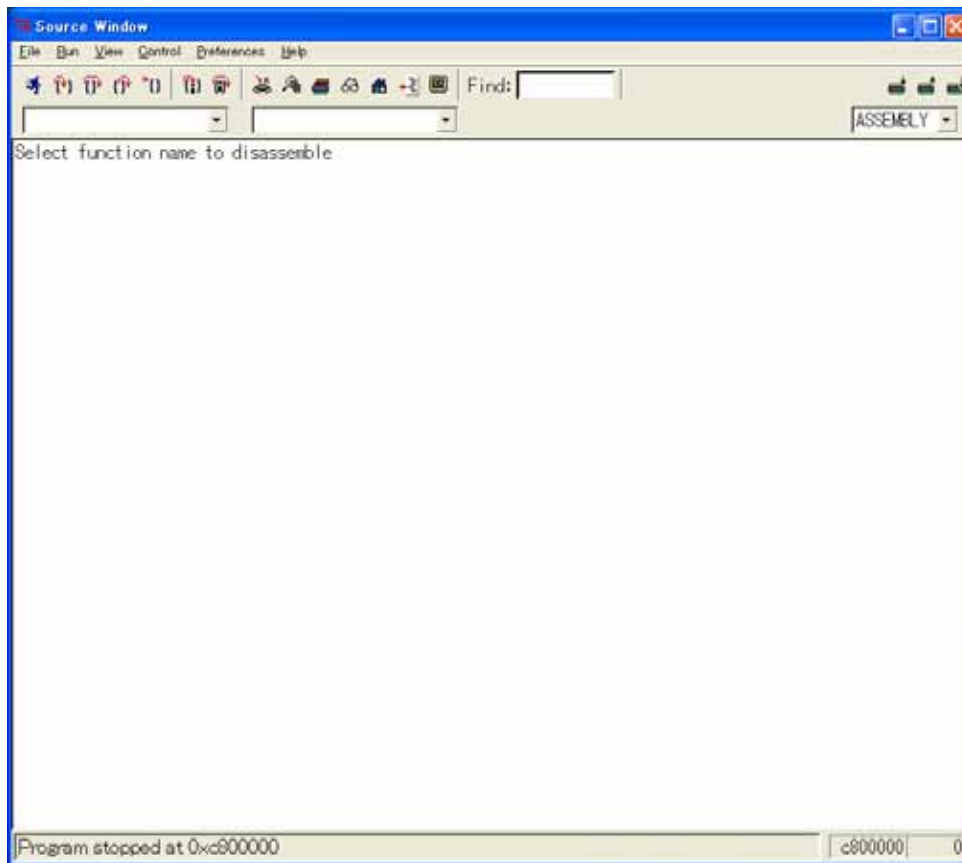
```
Console Window  
Read long at 0xa0900000: 0x2  
Read long at 0xa0900000: 0x2  
Read long at 0xa0900000: 0x2  
Read long at 0xa0900000: 0x2  
Read long at 0xa0900000: 0x2  
Wrote long to 0xa0700024: 0x30  
Wrote long to 0xa0700020: 0x83  
Read long at 0x22000: 0x2f1e9c  
Wrote long to 0xa0700020: 0x3  
Wrote long to 0xa0700100: 0x84280  
Wrote long to 0xa0700120: 0x84280  
Wrote long to 0xa0700140: 0x84280  
Wrote long to 0xa0700160: 0x84280  
Wrote long to 0xa0700220: 0x82  
Wrote long to 0xa0700224: 0x0  
Wrote long to 0xa0700228: 0x0  
Wrote long to 0xa070022c: 0x8  
Wrote long to 0xa0700230: 0x0  
Wrote long to 0xa0700234: 0x4  
Wrote long to 0xa0700238: 0x0  
Wrote long to 0x90600080: 0x10cd  
cpsr: 0xd3  
  
(ns9360-gdb) |
```

← コンソールウィンドウが表示されます。

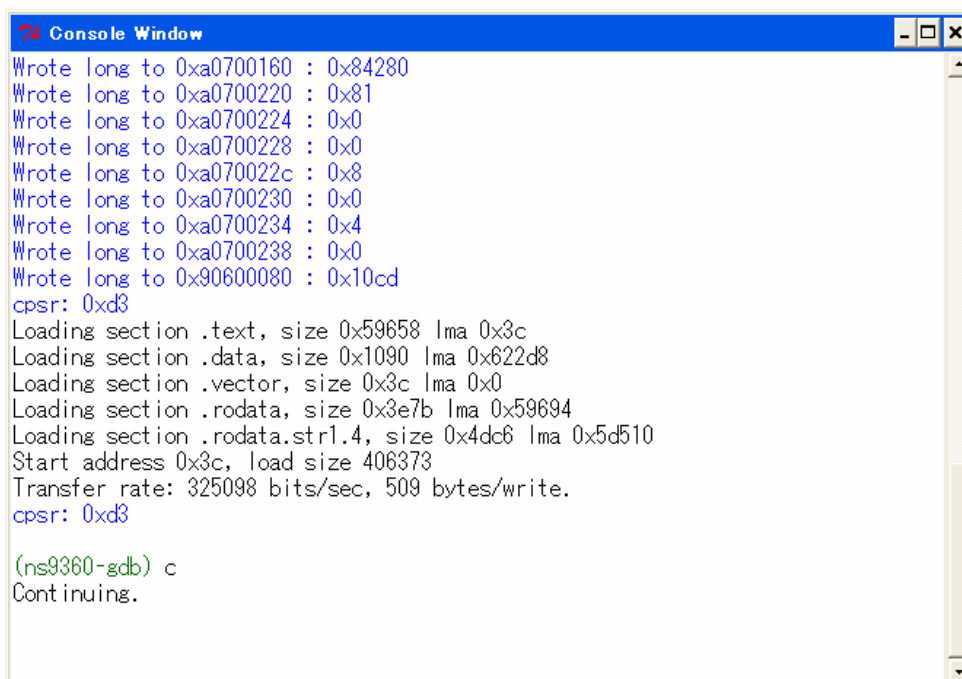
- ⑤ 以下の警告が出た場合は、「はい」をクリックしてください。



- ⑥ プログラム (jsp.elf) のロードが終了すると、デバッガのメインウィンドウが表示されます。



- ⑦ コンソールウィンドウで、” c ” を入力すると、プログラムが実行されます。



- ⑧ その他操作方法は、HELPを参照ください。

## 5. タスクの追加

タスクを追加する場合は、以下の作業が必要となります。  
コンフィグ名が” az9360 ” の場合の例を以下に示します。

- (1) az9360.cfg(itron/az9360)の変更  
作成したタスクや使用するリソースの登録を行います  
Toppers/JSPでは、タスクの登録は、静的APIの”CRE\_TASK”を使用します。

CRE\_TASKの宣言は、コンフィグレーション・ファイルに

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

と記述します。

```
CRE_TASK(APL1, { TA_HLNG, (VP_INT) 1, apl1, MID_PRIORITY, STACK_SIZE, NULL });  
CRE_TASK(APL2, { TA_HLNG, (VP_INT) 2, apl2, MID_PRIORITY, STACK_SIZE, NULL });  
:  
CRE_TASK(TASK3, { TA_HLNG, (VP_INT) 3, task, MID_PRIORITY, STACK_SIZE, NULL });  
CRE_TASK(MAIN_TASK, { TA_HLNG|TA_ACT, 0, main_task, MAIN_PRIORITY, STACK_SIZE, NULL });
```

- (2) make.sh(itron/az9360)の変更  
追加したファイルを make.sh 内の”APP\_OBJ”変数に追加します。

```
APP_OBJ="mnt.o   ¥  
        apl1    ¥  
        apl2    "
```

- (3) makefile(itron/az9360)の変更  
追加したファイルを makefile 内の”UTASK\_COBJS”変数に追加します。

```
UTASK_COBJS = $(UNAME).o CPU_IO.o PS_TELNET.o HTTPD_CGI.o CSP.o PS_NIC.o ¥  
              mib_if.o mib_com.o mib_Priv.o mib_Sys.o mib_table.o ¥  
              PS_USER.o halt.o PS_SNMP_USER.o EF_USER.o apl1 apl2
```

- (4) ファイルの追加  
追加するファイルを”itron/apps”の下に追加します。