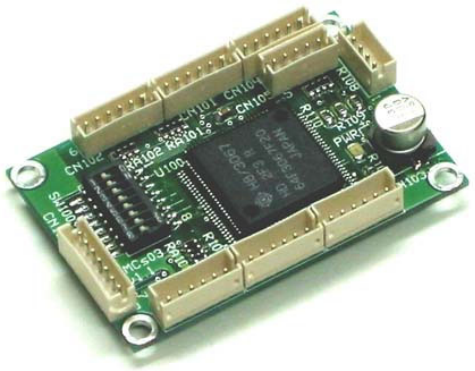
iXs Research Corp.
Cool Robotics

iMCs03 ソフトウェアマニュアル

Ver. 1.1



株式会社イクシスリサーチ

目　次

１．はじめに

本マニュアルは USB 接続 I/O，アナログ入出力ボード iMCs03 のソフトウェアに関する説明，および諸設定に関して記述してあります．なお，本マニュアルは Linux の基礎知識があり，ビット演算が出来るものとして説明がされております．各 Linux コマンド等に関しては，Linux の入門書等を参照してください．

本マニュアルは不定期に改訂されます．最新版は Web よりダウンロードしてください．

２．CH の機能

２．１　ピン配置図（CN100 ～ CN108)

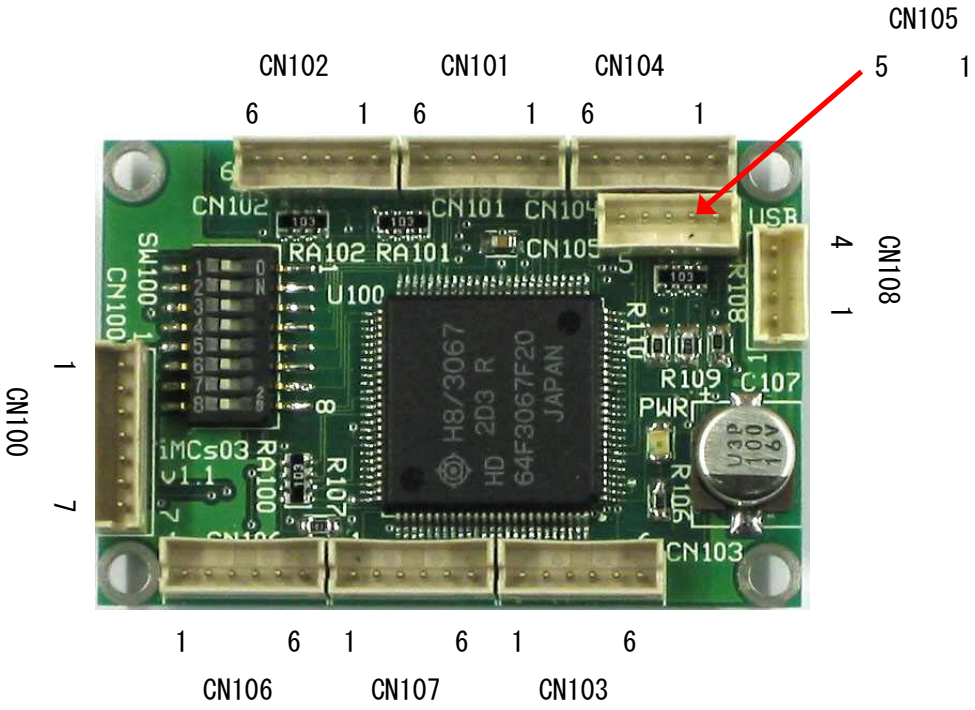


Fig.1 ピン配置

２．２　モータ出力ピン（CN101 ～ CN105）

	CN101	CN102	CN103	CN104	CN105
Pin1	GND	GND	GND	GND	GND
Pin2	PB0	PB4	PA0	PA4	P90
Pin3	PB1	PB5	PA1	PA5	P92
Pin4	PB2	PB6	PA2	PA6	P94
Pin5	PB3	PB7	PA3	PA7	VCC
Pin6	VCC	VCC	VCC	VCC	-

２．３　アナログ入力ピン（CN106 ～ CN107）

	CN106	CN107
Pin1	AGND	AGND
Pin2	AN0	AN4
Pin3	AN1	AN5
Pin4	AN2	AN6
Pin5	AN3	AN7
Pin6	AVCC	AVCC

２．４　ディップスイッチ

Pin	機能	ON	OFF
1	ID　(0x00 ～ 0x1f)	1	0
2		1	0
3		1	0
4		1	0
5		1	0
6	-	1	0
7	-	1	0
8	-	1	0

３．パケット

３．１　パケット１

パケット１の大きさは 64 バイト，方向は IN（H8→PC）です．パケットの中身は Fig.2 の構造体で表されます．Short のバイトオーダはリトルエンディアン（インテルなど）です．従ってインテル系の CPU では，そのまま扱うことが出来ます．H8 内部はビッグエンディアンですが，転送前にリトルエンディアンに変換しています．A/D コンバータの値は左詰めです．例えば A/D の値（10bit）を 0～1023 の値で扱いたい場合には，6bit 右シフトしてください．内部カウンタの値は 1[ms]毎に１上がり，65535 の次は 0 になります．

```
struct udata{
        unsigned short time;     /* 内部カウンタ (1ms 周期)の値 */
        unsigned short magicno; /* EEPROM 内の数字 */
        unsigned short ad[8];    /* A/D コンバータの値(10bit) */
        unsigned char P9DR;      /* P9 データレジスタの値 */
        unsigned char PADR;      /* PA データレジスタの値 */
        unsigned char PBDR;      /* PB データレジスタの値 */
        char dmy[41];
```

```
};
```

Fig.2 パケット１の udata 構造体

P9DR, PADR, PBDR の値は，各ポート P9，PA，PB の対応したポートの状態を表示します．P9DDR, PADDR, PBDDR で入力に設定したポートは，接続したセンサ類の状態(Hi/Low)が，出力に設定したポートは，出力値(Hi/Low)が表示されます．

Table 1　ポート番号とビット対応表

ポート番号	7	6	5	4	3	2	1	0
P9DR P9DDR PADR PADDR PBDR PBDDR	ビット７	ビット６	ビット５	ビット４	ビット３	ビット２	ビット１	ビット０

３．２　パケット２

パケット 1 の大きさは 64 バイト，方向は OUT（PC→H8）です．パケットの中身は次の構造体で表されます．

```
struct udr{
        unsigned char retval;
        unsigned char P9DR;     /* P9 データレジスタ値 */
        unsigned char PADR;     /* PA データレジスタ値 */
        unsigned char PBDR;     /* PB データレジスタ値 */
        char dmy[60];
};
```

Fig.3 パケット２の udr 構造体

P9DR, PADR, PBDR の値は，各ポート P9，PA，PB の対応したポートへ 1（Hi），0（Low）を出力します．設定は，各ポートに対応したビットを 1（Hi），0（Low）にすることで，指定できます．ただし，P9DDR, PADDR, PBDDR で出力に設定したビット以外へ値を書き込まないよう注意してください．

各ポートとデータレジスタの対応は Table 1 のように表されます．

例えば PA0 に Hi の信号を送る場合は

```
/************************************************************
```

```
cmd1.PADR |= 0x01;        /* - - - - - - - - 1 */
*************************************************************/
```

ここで cmd1 はユーザー定義のパケット 2 の udr 型構造体です.

また, PA5 に Low の信号を送る場合は,

```
/*************************************************************
cmd1.PADR &= ~0x20;       /* - - 0 - - - - - */
*************************************************************/
```

とします.

さらに, PA0, PA5 に Hi を, PA3, PA5 に Low の信号を送る場合は,

```
/*************************************************************
cmd1.PADR |= 0x21;        /* - 0 1 - 0 - - 1 */
cmd1.PADR &= ~0x48;
*************************************************************/
```

とします.
(この辺は, ビット演算になっているため, 少々慣れが必要です.)

３．３　パケット３

パケットの大きさは 64 バイト, 方向は OUT (PC→H8) です. パケットの中身は Fig.3 の構造体で表されます.

```
struct uddr{
        unsigned char P9DDR;     /* PB データディレクションレジスタ値 */
        unsigned char PADDR;     /* PB データディレクションレジスタ値 */
        unsigned char PBDDR;     /* PB データディレクションレジスタ値 */
        char dummy[61];
};
```

Fig.4 パケット 3 の uddr 構造体

P9DDR, PADDR, PBDDR の値は, 各ポート P9, PA, PB の対応したポートの入出力の状態を設定します. 設定は, 各ポートに対応したビットを 1 (Hi) にすることで出力ポートに, 0 (Low) にすることで入力ポートに指定できます.

各ポートとデータディレクションレジスタの対応は Table 1 のように表されます.

３．４　USB ドライバについて

３．４．１　デバイスのオープン，クローズ

USB デバイスをオープンするためには，プログラム内で

```
int fd;
fd = open(“/dev/usc0”,O_RDWR);
```

と記述します．デバイスのオープンに失敗したときは，戻り値-1 が返ります．クローズするためには

```
close(fd)
```

とします．

プログラムは以下のようになります．

```
/*************************************************************
  char *dev = "/dev/usc0";
  if (argc>1) dev = argv[1];

  if ((fd = open(dev, O_RDWR)) == -1) {
    fprintf(stderr, "%s: Open error¥n", dev);
    exit(1);
  }
*************************************************************/
```

３．４．２　iMCs03 に初期化データ書き込む

iMCs03 に初期化データ（パケット 3 の cmd）を書き込むには，まず

```
ioctl(fd,USC_DDR_SET);
```

を実行しておく必要があります．（一度実行すれば，変更があるまで有効）その後，

```
write(fd, &cmd, sizeof(cmd));
```

で値を書き込みます．ここで cmd はユーザー定義のパケット 3 の uddr 型構造体です．

プログラムは以下のようになります．

```
/*************************************************************
  struct uddr cmd;

  if (ioctl(fd, USC_DDR_SET) < 0){
    fprintf(stderr, "ioctl: USC_DDR_SET error¥n");
    exit(1);
  }
  if (write(fd, &cmd, sizeof(cmd)) < 0) {
    fprintf(stderr, "write error¥n");
```

```
    exit(1);
  }
**************************************************************/
```

３．４．３　iMCs03 からデータを取り込む

iMCs03 からのデータを連続して取り込むには，まず

```
ioctl(fd,USC_BUFREAD);
```

を実行しておく必要があります．（一度実行すれば，変更があるまで有効）その後，

```
read(fd, &buf, sizeof(buf));
```

で値を取得します．ここで buf はユーザー定義のパケット 1 の udata 型構造体です．
プログラムは以下のようになります．

```
/**************************************************************
  struct udata buf;

  if (ioctl(fd, USC_BUFREAD) < 0){
    fprintf(stderr, "ioctl: USC_BUFREAD error¥n");
    exit(1);
  }
  if (read(fd, &buf, sizeof(buf)) != sizeof(buf)) {
    fprintf(stderr, "Warning: read size mismatch");
    continue;
  }
**************************************************************/
```

３．４．４　iMCs03 に制御データを書き込む

iMCs03 に制御データ（パケット 2 の cmd1）を書き込むには，まず

```
ioctl(fd,USC_DR_SET);
```

を実行しておく必要があります．（一度実行すれば，変更があるまで有効）その後，

```
write(fd, &cmd1, sizeof(cmd1));
```

で値を書き込みます．ここで cmd1 はユーザー定義のエンドポイント 2 の udr 型構造体です．
プログラムは以下のようになります．

```
/**************************************************************
  struct udr cmd1;
  if (ioctl(fd, USC_DR_SET) < 0){
    fprintf(stderr, "ioctl: USC_DR_SET error¥n");
    exit(1);
  }
  if (write(fd, &cmd1, sizeof(cmd1)) < 0) {
```

```
    printf("write err¥n");
    break;
  }
**************************************************************/
```

３．４．５　LITTLE_ENDIAN, BIG_ENDIAN について

コンピュータは 2 バイト以上のデータを扱う際に 1 バイトごとに分割して処理しますが，これを最下位のバイトから順番に記録/送信する方式をリトルエンディアン(LITTLE ENDIAN)と呼び，最上位のバイトから順番に記録/送信する方式をビッグエンディアン(BIG ENDIAN)と呼びます．Intel 系のプロセッサはリトルエンディアン，Motorola 系のプロセッサはビッグエンディアンのため，PC から送信する際に，データの上位バイトと下位バイトの入れ替えの必要が生じる場合があります．

iMCs03 は LITTLE ENDIAN 形式でデータを扱っているため，Motorola 系のプロセッサを持つコンピュータと接続する場合，データの入れ替えが必要です．例えば，A/D コンバータの値は，以下のような記述になります．

```
/**************************************************************
#if __BYTE_ORDER == __BIG_ENDIAN
      buf.ad[i] = (0xff & buf.ad[i])<<8 | (0xff00 & buf.ad[i])>>8;
#endif
**************************************************************/
```

ただし，使用する PC が，どちらか一方に決まっている場合は，どちらか一方を記述するだけで正しく処理されます．

３．５　iMCs03 側のプログラムについて（ディップスイッチの設定）

iMCs03 上のディップスイッチの Pin1～5 の操作により iMCs03 に固有の ID 番号を振ることが出来ます．全てのピンを 0 とすることで ID は 0x00（0）になり，全てのピンを 1 にすることで ID は 0x1f（31）になります．

ID	Pin				
	1	2	3	4	5
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	1	0	0	0
4	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮
30	0	1	1	1	1

31	1	1	1	1	1

４．プログラムの実行

本章では，LINUX（Kernel 2.4 以上）で iMCs03 を操作する方法を説明します．以下の操作は全て root 権限で行ってください．なお，以下の説明では，コマンドラインからの入力は黒背景にしてあり，ユーザーモードでの入力は「$」で，ルートモードでの入力は「#」で記述されています．

```
$ su -l
```

４．１　USB デバイスの登録

USB デバイスを登録します．本操作は，各 PC において最初の 1 回だけ行います．
コマンドライン上で以下のように入力してください．

```
# mknod /dev/usc0 c 180 200
# chmod 666 /dev/usc0
```

４．２　USB ドライバのロード

iMCs03 は，USB マウス等，他の USB 機器との併用は出来ません．既に他の USB 機器を接続されている場合は，それらの USB 機器を外してください．また，hid のドライバが入っている場合は，以下のようにしてドライバを削除してください．

```
# rmmod hid
```

また，USB ドライバ usc.o をロードするに当たり，UHCI(Universal Host Controller Interface)がロードされている必要があります．ロードされていない場合は，以下のようにしてドライバをロードしてください．

```
# insmod uhci
```

usc のファイルがあるディレクトリに移動し，USB のモジュールをロードします． まだ iMCs03 を USB ポートに接続しないでください．

```
# cd /home/user1/usc/driver/
# make
# insmod usc.o
```

ここで，正常にモジュールがロードされているかを確認します．

```
# lsmod
```

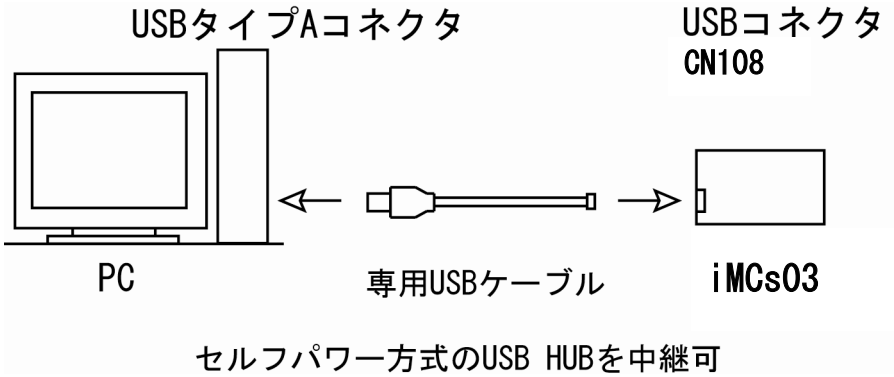
と入力し，

```
Module                  Size  Used by
usc                     7360   0  (unused)
```

と表示されることを確認してください．

４．３　ボードの接続

ボードを接続します.



セルフパワー方式のUSB HUBを中継可

接続後,

```
# dmesg
```

と入力し,

```
usb.c: registered new driver usc
usc.c: H8 based USB sensor controller driver
hub.c: USB new device connect on bus1/2, assigned device number 2
usc.c: USB robot controller now attached to usc0
```

と表示されることを確認してください. また, HUB を中継する場合は, HUB の接続後（HUB には iMCs03 をまだ接続しないでください),

```
# dmesg
```

と入力し,

```
ub.c: USB new device connect on bus1/2, assigned device number 3
hub.c: USB hub found
hub.c: 4 ports detected
```

と表示されることを確認してください.

次に iMCs03 が正しく認識されているかを確認するために,

```
# make
# ./h8test
```

と入力し,

```
Vendor xxxxxxxx
Product xxxxxxxx
read status 0
read status 1
write status 3
write status 3
```

と表示されることを確認してください.

４．４　終了処理

最後に，iMCs03 を USB ポートから取り外す際，dmesg で，

```
usb.c: USB disconnect on device 2
usc.c: usc0 now disconnected
```

と表示されることを確認してください．その後，

```
# rmmod usc
```

と入力し，USB ドライバを解放します．

４．５　プログラムの実行

各サンプルプログラムを実行します．添付の CD-ROM の内容を適当な場所にコピーして使用してください．全てのサンプルプログラムは iMCs03 と弊社モータドライバ iMDs03 とを接続した場合について書かれています．なお，iMCs03 に付属するドライバ，サンプルソースの Makefile の INCLUDE は

```
/*****************************************************************
INCLUDE= /usr/src/linux/include
*****************************************************************/
```

となっています．RedHat7.3，RedHat8.0 等を使用して，コンパイルエラーが出る場合は，

```
/*****************************************************************
INCLUDE= /usr/src/linux-2.4/include
*****************************************************************/
```

のように，Path を変更してみてください．

４．５．１　センサ値の取得

サンプルプログラムを実行します．サンプルで添付されているセンサ読込みプログラム uread を実行する場合は，

```
# make uread
# ./uread
```

と入力します．

```
20324 0 1023 1023 1023 1023 1023 1023 1023 1023 fd ff ff
20326 0 1023 1023 1023 1023 1023 1023 1023 1023 fd ff ff
20328 0 1023 1023 1023 1023 1023 1023 1023 1023 fd ff ff
```

のような値が表示されます．

左から，time，magicno，ad[0]，ad[1]，ad[2]，ad[3]，ad[4]，ad[5]，ad[6]，ad[7]，P9DR,PADR,PBDR を表しています．

４．５．２　ポートからの出力

サンプルプログラムを実行します．サンプルで添付されている書き込みプログラム uwrite を実行する場合は，

```
# make uwrite
# ./uwrite
```

と入力します．

改訂履歴

2003 年 4 月	初版 Ver1.0
2009 年 1 月	ピンレイアウト修正

お問合せ(お問い合わせはメールにてお願いいたします)
株式会社イクシスリサーチ
E-mail : info@ixs.co.jp