

iMCs04 ソフトウェアマニュアル

**iXs Research Corp.**

Cool Robotics

# iMCs04 ソフトウェアマニュアル Ver1.1

株式会社イクスリサーチ

## 目 次

|  |    |
|--|----|
| 1. 概要 .....                                  | 3  |
| 2. ch の機能 .....                              | 3  |
| 2. 1 モータ出力ピン(CN101~CN104) .....              | 3  |
| 2. 2 センサ入力ピン(CN105~CN108).....               | 3  |
| 2. 3 USB ピン(CN109).....                      | 3  |
| 2. 4 シリアルピン(CN100) .....                     | 3  |
| 2. 5 ディップスイッチ .....                          | 3  |
| 3. エンドポイント .....                             | 4  |
| 3. 1 エンドポイント 2 .....                         | 4  |
| 3. 2 エンドポイント 3 .....                         | 4  |
| 4 USB ドライバについて.....                          | 5  |
| 4. 1 デバイスのオープン, クローズ.....                    | 5  |
| 4. 2 iMCs04 のデータを取り込む.....                   | 5  |
| 4. 3 iMCs04 に制御データを書き込む.....                 | 6  |
| 4. 4 LITTLE_ENDIAN, BIG_ENDIAN について.....     | 6  |
| 5. ディップスイッチの設定.....                          | 8  |
| 6. プログラムの実行.....                             | 9  |
| 6. 1 USB デバイスの登録.....                        | 9  |
| 6. 2 USB ドライバのロード.....                       | 9  |
| 6. 3 ボードの接続.....                             | 9  |
| 6. 4 終了処理 .....                              | 10 |
| 6. 5 プログラムの実行.....                           | 11 |
| 6. 5. 1 センサ値の取得.....                         | 11 |
| 6. 5. 2 RC サーボモータの制御.....                    | 11 |
| 6. 5. 3 RC サーボモータの制御 (iMCs04 の複数台同時接続) ..... | 11 |

## 1. 概要

## 2. ch の機能

### 2. 1 モータ出力ピン(CN101~CN104)

| No |     |                           |
|----|-----|---------------------------|
| 1  | GND | 信号用電源 (GND) *RC サーボ電源供給不可 |
| 2  | PWM | PWM 出力                    |
| 3  | NC  |                           |
| 4  | NC  |                           |
| 5  | VCC | 信号用電源 (+5V) *RC サーボ電源供給不可 |

### 2. 2 センサ入力ピン(CN105~CN108)

| No |      |                             |
|----|------|-----------------------------|
| 1  | AVSS | アナログ用電源 (GND) *RC サーボ電源供給不可 |
| 2  | NC   |                             |
| 3  | AN   | アナログ入力 (10bit)              |
| 4  | NC   |                             |
| 5  | AVCC | アナログ用電源 (+5V) *RC サーボ電源供給不可 |

### 2. 3 USB ピン(CN109)

| No |     |                           |
|----|-----|---------------------------|
| 1  | VCC | 信号用電源 (+5V) *RC サーボ電源供給不可 |
| 2  | D-  | USB 信号                    |
| 3  | D+  | USB 信号                    |
| 4  | GND | 信号用電源 (GND) *RC サーボ電源供給不可 |

### 2. 4 シリアルピン(CN100)

| No |      |                                   |
|----|------|-----------------------------------|
| 1  | VCC  | 信号用電源 (+5V) *RC サーボ電源供給不可         |
| 2  | RXD1 | RXD1                              |
| 3  | TXD1 | TXD1                              |
| 4  | FWE  | Flash write enable (内部 Pull Down) |
| 5  | MD2  | MD2                               |
| 6  | RST  | リセット信号                            |
| 7  | GND  | 信号用電源 (GND) *RC サーボ電源供給不可         |

### 2. 5 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   |

### 3. エンドポイント

#### 3. 1 エンドポイント2

エンドポイント2はバルク転送で1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.1の構造体で表されます。Shortのバイトオーダー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できます。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。

```
struct ccmd {  
    unsigned short duty[4]; // Duty 比設定用  
    char dummy[56];  
};
```

Fig.1 エンドポイント2のccmd構造体

#### 3. 2 エンドポイント3

エンドポイント3はバルク転送で1パケットの大きさは64バイト、方向はIN (H8→PC)です。パケットの中身は次の構造体で表されます。scmd構造体はFig.2のようになっています。

```
struct uin {  
    unsigned short ad[4]; // A/D コンバータの値(10bit)  
    unsigned char magicno;  
    char dummy[55];  
};
```

Fig.2 エンドポイント3のuin構造体

## 4 USB ドライバについて

### 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/urbtc0" , 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/urbtc0";
if (argc>1) dev = argv[1];
if ((fd = open(dev, O_RDWR)) == -1) {
    fprintf(stderr, "%s: Open error¥n", dev);
    exit(1);
}
*****/

```

### 4. 2 iMCs04 のデータを取り込む

iMCs04 からのデータを連続して取り込むには, まず

```
ioctl(fd, URCC_GET_DATA);
```

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 はユーザー定義のエンドポイント 3 の uin 型構造体です.

プログラムは以下のようになります.

```

/*****
struct uin buf;
if (ioctl(fd, URCC_GET_DATA) < 0) {
    fprintf(stderr, "ioctl: URCC_GET_DATA error¥n");
    exit(1);
}
if ((i = read(fd, &buf, sizeof(buf))) != 64) {
    fprintf(stderr, "Warning: read size mismatch (%d!=%d).¥n", i, sizeof(buf));
    continue;
}
*****/

```

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

#### 4. 3 iMCs04 に制御データを書き込む

iMCs04 に制御データ（エンドポイント 2 の ccmd）を書き込むには、まず

```
ioctl(fd, URCC_DUTY_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, &obuf, sizeof(obuf));
```

で値を書き込みます。ここで obuf はユーザー定義のエンドポイント 3 の ccmd 型構造体です。

プログラムは以下のようになります。

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

```
struct ccmd obuf;
if (ioctl(fd, URCC_DUTY_SET) < 0) {
    fprintf(stderr, "ioctl: URCC_DUTY_SET error\n");
    exit(1);
}
if (write(fd, &obuf, sizeof(obuf)) < 0) {
    fprintf(stderr, "write error\n");
    exit(1);
}

```

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

#### 4. 4 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。

iMCs04 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、初期化データ (ccmd 型構造体) の offset に 0x7fff を代入する場合、以下のような記述になります。

```
/*  
    unsigned char duty_num = 0x1388;  
    #if __BYTE_ORDER == __LITTLE_ENDIAN  
        obuf.duty[0] = duty_num;  
    #else  
        obuf.duty[0] = (0xff & duty_num)<<8 | (0xff00 & duty_num)>>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。

## 5. ディップスイッチの設定

iMCs04 上のディップスイッチの Pin1~5 の操作により iMCs04 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 |

## 6. プログラムの実行

本章では、Linux (Kernel 2.4 以上) で iMCs04 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。

```
> su -l
```

### 6. 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/urc0 c 180 140
```

```
> chmod 666 /dev/urc0
```

### 6. 2 USB ドライバのロード

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 iMCs04 を USB ポートに接続しないでください。

```
> cd /home/user1/iMCs04/driver/
```

```
> insmod urbtc.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
```

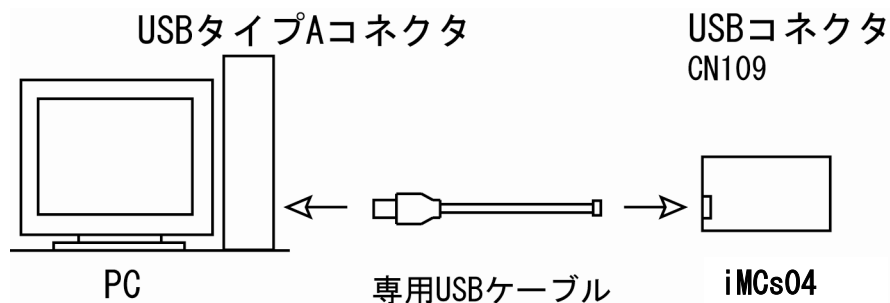
と入力し、

| Module | Size | Used by    |
|--------|------|------------|
| urbtc  | 7360 | 0 (unused) |

と表示されることを確認してください。

### 6. 3 ボードの接続

ボードを接続します。



接続後、

```
> dmesg
```

と入力し,

```
usb.c: registered new driver urbtc
```

```
urbtc.c: H8 based USB motor controller driver v0.1
```

```
hub.c: USB new device connect on bus1/2, assigned device number 2
```

```
urbtc.c: USB robot controller now attached to urbtc0
```

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 iMCs04 をまだ接続しないでください),

```
> dmesg
```

と入力し,

```
usb.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 iMCs04 が正しく認識されているかを確認するために,

```
> ./h8test
```

と入力し,

```
Vendor xxxxxxxx
```

```
Product xxxxxxxx
```

と表示されることを確認してください。

#### 6. 4 終了処理

最後に、iMCs04 を USB ポートから取り外す際、dmesg で、

```
usb.c: USB disconnect on device 2
```

```
urbtc.c: urbtc0 now disconnected
```

と表示されることを確認してください。その後、

```
> rmmod urbtc
```

と入力し、USB ドライバを解放します。

## 6. 5 プログラムの実行

### 6. 5. 1 センサ値の取得

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み込みプログラム `uread` を実行する場合は、

```
> ./uread
```

と入力します。

```
CH0:0xFFC0    CH1:0xFFC0    CH2:0xFFC0    CH3:0xFFC0
CH0:0xFFC0    CH1:0xFFC0    CH2:0xFFC0    CH3:0xFFC0
CH0:0xFFC0    CH1:0xFFC0    CH2:0xFFC0    CH3:0xFFC0
```

のような値が表示されます。

左から、`buf.ad[0]`、`buf.ad[1]`、`buf.ad[2]`、`buf.ad[3]`を表しています。

### 6. 5. 2 RC サーボモータの制御

プログラムを実行します。サンプルで添付されている RC サーボ出力プログラム `sample` を実行する場合は、

```
> make sample
```

```
> ./sample
```

と入力します。

### 6. 5. 3 RC サーボモータの制御 (iMCs04 の複数台同時接続)

2 台目以降のコントローラを接続する場合は、以下のように接続する数だけ USB ドライバを追加します。本操作は、各 PC において最初の 1 回だけ行います。

```
> mknod /dev/urc1 c 180 141
```

```
> chmod 666 /dev/urc1
```

ここで、180 は USB ドライバのメジャーNo, 141 はマイナーNo です。

3 台同時接続の場合は以下ようになります。

```
> mknod /dev/urc1 c 180 141
```

```
> chmod 666 /dev/urc1
```

```
> mknod /dev/urc2 c 180 142
```

```
> chmod 666 /dev/urc2
```

続いてプログラム (2 台接続時) を実行します。サンプルで添付されている 2 台同時接続 RC サーボモータ制御プログラム `samplem` を実行する場合は、

```
> make samplem
```

```
> ./samplem
```

と入力します。

改訂履歴

2002 年 10 月

2003 年 1 月

初版

Ver1.1

- ・ピンファンクション訂正
- ・ドライバ名訂正
- ・ドライバのマイナ番号訂正

お問い合わせ(お問い合わせはメールにてお願いいたします)

**株式会社イクスリサーチ**

E-mail : info@ixs.co.jp

本社所在地

〒212-0055

神奈川県川崎市幸区南加瀬 4-17-14

横浜工場

〒230-0071

神奈川県横浜市鶴見区駒岡 5-14-10

本書の内容の一部または全部を無断転載・無断複写することは禁止されています。  
本書の内容については将来予告なしに変更することがあります。

この取扱説明書は、再生紙を使用しています。