汎用科学サンプリングプロセッサ
ドライバソフトウェア

取 扱 説 明 書

日 本 通 信 機 株 式 会 社

1　はじめに

本文書は、USBデバイス「時刻同期式データサンプラー」を、Ubuntu 14.04 LTSにて動作させるために作成したドライバについて記述したものです。

この文書では、ドライバソフトウェアおよびライブラリを総称して**U3TDS** パッケージと呼びます。また、時刻同期式データサンプラードライバソフトウェアのことを単に**U3TDS** ドライバと呼びます。

現在のU3TDSパッケージの最新バージョンは1.00で、この版では、Ubuntu 14.04 LTS 3.13.0-24-genericに対応しています。本書では、カーネルバージョン表記上の煩雑さを避けるため、以下の記法を用います。適宜本文を読み替えてください。

表記	置換文字列
<KERNELRELEASE>	3.13.0-24-generic

2　動作確認環境

Ubuntu 14.04 LTS 3.13.0-24-generic 64bitで動作を確認。[1]

サンプリングデータをバイナリ形式にてローカルハードディスクに保存するテストにおいて、サンプリング設定1024Mbps まで動作を確認。

[1] Ubuntu 14.04 LTS 以外の環境での動作は未確認です。

3　ディレクトリ/ファイル構成

tar＋gzip によって作成されたアーカイブを解凍したディレクトリ/ファイルの構成は、以下のようになります。

```
vlbi-usb3-linux/
  |
  +Makefile
  +VERSION U3TDS パッケージバージョン
  +doc/
  |  |
  |  +manual_9277_DRV.doc 取扱説明書（ MS-WORD 形式 ）
  |  |
  |  +manual_9277_DRV.pdf 取扱説明書（ PDF 形式 ）
  |
  +driver/
  |  |
  |  +Makefile U3TDS ドライバビルド用のMakefile
  |  |
  |  +u3tds.c U3TDS ドライバソースファイル
  |
  +examples/
  |  |
  |  +Makefile サンプルプログラムビルド用Makefile
  |  |
  |  +*.c サンプルプログラム（ C ソース ）
  |
  +include/
  |  |
  |  +u3tdsio.h U3TDS ドライバI/O コントロールコマンドファイル
  |  |
  |  +u3tdssdh.h 時刻同期式データサンプラーのサンプリングデータの
  |     ヘッダフォーマット定義ファイル
  |
  +libtds/
  |
  +Makefile ライブラリビルド用Makefile
  |
  +libtds.c ライブラリソースファイル
```

4　ローダブルモジュール

4.1　ローダブルモジュールの作成

(a) vlbi-usb3-linux.tar.gz を展開して vlbi-usb3-linux に移動する。

```
# tar xvfz vlbi-usb3-linux.tar.gz
# cd vlbi-usb3-linux/
```

(b) モジュールを構築、インストールする。この時ライブラリも同時にビルド、インストールされます。

```
# make
# make install
```

(c) デバイスノードを作成する。

```
cd /dev
mknod –m 666 u3tds0 c 180 222
```

デバイスファイルは、/dev/u3tds0 となります。もし Linux 上で udevd が動作している場合はデバイスノードを作成する必要はありません。

(d) モジュールをロードする。

```
#　 insmod /lib/modules/<KERNELRELEASE>/kernel/drivers/usb/misc/u3tds.ko
```

5　インストールするファイル

make install を実行する事によって次のファイルをそれぞれ別にインストールします。

tds.ko /lib/molules/<KERNELRELEASE>/kernel/drivers/usb/misc/u3tds.ko
tdsio.h /usr/include/sys/tdsio.h
tdssdh.h /usr/include/sys/tdssdh.h
libtds.so /usr/local/lib/libtds.so

6　デバイスノード

6.1 デバイスノードの作成

mknod コマンドを使用し、デバイスノードを作成します。
デバイスには割り当てられるメジャー番号は180 です。マイナー番号は通常は222が割り当てられます。もしLinux 上でudevd が動作している場合はデバイスノードを作成する必要はありません。

```
# cd /dev
# mknod -m 666 u3tds0 c 180 222
```

デバイス接続時のログに以下いずれかのメッセージが表示された場合は、想定外のマイナー番号が割り当てられています。その場合は、次節で説明する対応が必要になります。
1. The minor number is dynamically allocated. It is recommend to use udev.
2. Could not allocated the expected sequence number of minor. Please re-create device node yourself.

6.2 例外的事象への対応

実用上、多くのLinux ディストリビューション、カーネルコンフィグレーションでは、6.1デバイスノードの作成で想定するマイナー番号の割り当てが行われます。ただし、以下のいずれかの環境下では、不一致が発生するためユーザ側で対応が必要です。

(a) カーネルコンフィグレーションUSB DYNAMIC MINORS がenable の場合
カーネルコンフィグレーションUSB DYNAMIC MINORSがenable の場合は、U3TDSドライバが要求する静的なマイナー番号の割り当て222は無視され、動的な割り当てが行われます。そのため、デバイスノードを事前に作成しておくことができません。

_ カーネルコンフィグレーションの確認方法
カーネルコンフィグレーションは、ターゲットシステム上の
/boot/config-<KERNELRELEASE> というファイルで確認できます。
コンフィグレーションが有効の場合、以下のように記述されています。

```
CONFIG_USB_DYNAMIC_MINORS=y
```

コンフィグレーションが無効の場合、以下のように記述されているか、もしくはファイル中に記述が存在しません。

```
# CONFIG_USB_DYNAMIC_MINORS is not set
```

_ 対応方法
udevd の使用を必須とします。udevd を使用することでデバイスノードの自動生成を行います。

(b) 3rd パーティ製のUSB デバイスドライバと併用された場合

同一PC 上で、3rd パーティ製のUSB デバイスドライバと併用された場合で、かつ、そのUSBデバイスドライバが222のマイナー番号を要求した場合も、マイナー番号が衝突します。

_ 対応方法

デバイスノードの再作成を行います。デバイスに割り当てられたマイナー番号はデバイス接続時のログで確認することが出来ます。

7　システムコールインターフェース

7.1 サポートしているシステムコール

1 open()

2. close()

3. read()

4. ioctl()

7.2 システムコール使用例

以下に、各システムコールの使用例を示す。特に記述していない点に関しては、Ubuntu 14.04 LTS 標準のシステムコールの仕様に準じている。

1. open()
   デバイスをオープンし、ファイルディスクリプタを返す。引数にはデバイス名、オープンモードを指定する当デバイスは読み出し専用であるので、オープンモードは O_RDONLY を指定する。

   （使用例）

```
int fd;

fd = open("/dev/u3tds0", O_RDONLY);
```

2. close()
   デバイスをクローズする。引数には 1 の open()で返されたファイルディスクリプタを指定する。

   （使用例）

```
close(fd);
```

3. read()

オープンしたデバイスから、データを読み込む。引数には 1 の open()で返されたファイルディスクリプタ、データ読み込み用バッファのアドレス、そのバッファサイズ（バイト数）を指定する。実際に読み込まれたデータサイズ（バイト数）が返される。

（使用例）

```
int nread;
unsigned int buffer[1024];
int nbyte = sizoef(buffer);

nread = read(fd, buffer, nbyte);
```

4. ioctl()

デバイスの各種操作を行う。第 1 引数には 1 の open()で返されたファイルディスクリプタ、第 2 引数には行うべき操作、必要に応じて第 3 引数以降にその操作ごとに必要なパラメータを指定する。

- 初期化

  デバイスドライバ内パラメータを初期化する。第 2 引数には TDSIO_INIT を指定する。
  この操作を実行すると、以下の状態になる。
  - チャンネル数設定がなされていない状態（4 参照）
  - ビット数設定がなされていない状態（4 参照）
  - フィルタ周波数設定がなされていない状態（4 参照）
  - サンプリング周波数設定がなされていない状態（4 参照）
  - 1PPS 同期がなされていない状態（4 参照）
  - 時刻設定がなされていない状態（4 参照）
  - エラー情報をリセットした状態（4 参照）

  （使用例）

  ```
  ioctl(fd, TDSIO_INIT);
  ```

- サンプラーFIFO クリア

  サンプラー上の FIFO をクリアする。第 2 引数には TDSIO_BUFFER_CLEAR を指定する。
  （使用例）

  ```
  ioctl(fd, TDSIO_BUFFER_CLEAR);
  ```

- 1PPS 同期
  外部 1PPS 入力信号への同期を行う。第 2 引数には TDSIO_SYNC_1PPS を指定する。
  外部 1PPS 入力信号が検出できない場合には、エラーが発生する。
  (使用例)

```
ioctl(fd, TDSIO_SYNC_1PPS);
```

- 時刻設定
  サンプラーに時刻を設定する。第 2 引数には TDSIO_SET_TIME を、第 3 引数には設定すべき時刻データを保持する変数(32bit)のアドレスを指定する。
  時刻の形式は、下位 17 ビット(0bit-16bit)に 00:00:00 からの通算秒で、18bit-27bit (9bit 幅)に 1 月 1 日からの通算日数で指定する。[2]
  簡易マクロとして、以下を用意している。

マクロ	機能
TDS_MAKE_TIME(day, sec)	day(1 月 1 日からの通算日数)と sec(00:00:00 からの通算秒)から、適切な値を生成する。

  (使用例)

```
/* 2000/09/22 14:39:28 を指定する場合 */
unsigned int time;
unsigned int day;
unsigned int sec;

day = 31 + 29 + 31 + 30 + 31 + 30 + 31 + 31 + 22;
sec = ((14 * 60) + 39) * 60 +28;
time = TDS_MAKE_TIME(day, sec);

ioctl(fd, TDSIO_SET_TIME, &time);
```

[2] デバイスレジスタへの書き込み時に指定する時刻と同じ形式

- 時刻取得

  サンプラーが保持する時刻を取得する。第 2 引数には TDSIO_GET_TIME を、第 3 引数には取得した時刻データを格納する変数(32bit)のアドレスを指定する。取得される時刻データの形式は、4 時刻設定における形式と同様。簡易マクロとして、以下を用意している。

マクロ	機能
TDS_GET_SEC(arg)	引数 arg から 00:00:00 からの通算秒数部分を取り出す。
TDS_GET_SEC_CARRY(arg)	引数 arg から通算秒数のキャリービットを取り出す。結果は 0 または 1 になる。
TDS_GET_DAYS(arg)	引数 arg から 1 月 1 日からの通算日数部分を取り出す。
TDS_GET_DAYS_CARRY(arg)	引数 arg から通算日数のキャリービットを取り出す。結果は 0 または 1 になる。

  （使用例）

```
unsigned int time;
unsigned int day;
unsigned int sec;

ioctl(fd, TDSIO_GET_TIME, &time);

day = TDS_GET_DAYS(time);
sec = TDS_GET_SEC(time);
```

- 年情報設定

  サンプラーに年情報を設定する。第 2 引数には TDSIO_GET_YEAR を、第 3 引数には、設定する年データを格納する変数(32bit)のアドレスを指定する。

  （使用例）

```
unsigned int year;

ioctl(fd, TDSIO_SET_YEAR, &year);
```

- 年情報取得
  サンプラーが保持している年情報を取得する。第 2 引数には TDSIO_GET_YEAR を、第 3 引数には、取得した年データを格納する変数(32bit)のアドレスを指定する。
  （使用例）

```
unsigned int year;

ioctl(fd, TDSIO_GET_YEAR, &year);
```

- チャンネル数設定
  サンプリングを行うチャンネル数を設定する。第 2 引数には TDSIO_SET_CHMODE を、第 3 引数にはチャンネル数設定情報を保持する変数(32bit)のアドレスを指定する。

  ー チャンネル数指定

チャンネル数	指定マクロ
1ch	TDSIO_SAMPLING_1CH
2ch	TDSIO_SAMPLING_2CH
4ch	TDSIO_SAMPLING_4CH

  （使用例）

```
/* 4ch を設定する場合 */

unsigned int chmode;

chmode = TDSIO_SAMPLING_4CH;

ioctl(fd, TDSIO_SET_CHMODE, &chmode);
```

- **ビット数設定**

  サンプリングを行うビット数を設定する。第 2 引数には TDSIO_SET_RESOLUTIONBIT を、第 3 引数にはビット数設定情報を保持する変数(32bit)のアドレスを指定する。

  ー **データ幅指定**

ビット数	指定マクロ
1bit	TDSIO_SAMPLING_1BIT
2bit	TDSIO_SAMPLING_2BIT
4bit	TDSIO_SAMPLING_4BIT
8bit	TDSIO_SAMPLING_8BIT

  （使用例）

```
/* 1bit を設定する場合 */

unsigned int resolutionbit;

resolutionbit = TDSIO_SAMPLING_1BIT;

ioctl(fd, TDSIO_SET_RESOLUTIONBIT, &resolutionbit);
```

- サンプリング設定
  サンプリングを行うサンプリング周波数を設定する。
  第 2 引数には TDSIO_SET_FSAMPLE を、第 3 引数にはサンプリング周波数設定情報を保持する変数(32bit)のアドレスを指定する。

  ー 周波数指定

サンプリング周波数	指定マクロ
2MHz	TDSIO_SAMPLING_2MHZ
4MHz	TDSIO_SAMPLING_4MHZ
8MHz	TDSIO_SAMPLING_8MHZ
16MHz	TDSIO_SAMPLING_16MHZ
32MHz	TDSIO_SAMPLING_32MHZ
64MHz	TDSIO_SAMPLING_64MHZ
128MHz	TDSIO_SAMPLING_128MHZ

  （使用例）

```
/* 2MHz を設定する場合 */
unsigned int fsample;

fsample = TDSIO_SAMPLING_2MHZ;

ioctl(fd, TDSIO_SET_FSAMPLE, &fsample);
```

- チャンネル数設定取得
  現在のチャンネル数設定を取得する。第 2 引数には TDSIO_GET_CHMODE を、第 3 引数には取得した設定情報を格納する変数(32bit)のアドレスを指定する。得られるチャンネル数設定情報は、4 のチャンネル数設定時に指定するものと同じ形式になっている。
  （使用例）

```
unsigned int chmode;

ioctl(fd, TDSIO_GET_CHMODE, &chmode);

if (chmode == TDSIO_SAMPLING_4CH) {
/* 4ch */
}
```

- ビット数設定取得
  現在のビット数設定を取得する。第 2 引数には TDSIO_GET_RESOLUTIONBIT を、第 3 引数には取得した設定情報を格納する変数(32bit)のアドレスを指定する。得られるチャンネル数設定情報は、4 のビット数設定時に指定するものと同じ形式になっている。

  （使用例）

```
unsigned int resolutionbit;

ioctl(fd, TDSIO_GET_RESOLUTIONBIT, &resolutionbit);

if (resolutionbit == TDSIO_SAMPLING_4BIT) {
/* 4Bit */
}
```

- サンプリング周波数設定取得
  現在のサンプリング周波数設定を取得する。第 2 引数には TDSIO_GET_FSAMPLE を、第 3 引数には取得した設定情報を格納する変数(32bit)のアドレスを指定する。得られるサンプリング周波数設定情報は、4 のサンプリング周波数設定時に指定するものと同じ形式になっている。

  （使用例）

```
unsigned int fsample;

ioctl(fd, TDSIO_GET_FSAMPLE, &fsample);

if (fsample == TDSIO_SAMPLING_4MHZ) {
/* 4MHz */
}
```

- ステータスの取得
  サンプラーのステータス情報を取り出す。第 2 引数には TDSIO_GET_STATUS を、第 3 引数にはステータス情報を格納すべき変数(32bit)のアドレスを指定する。[3]

  （使用例）

```
unsigned int stat;

ioctl(fd, TDSIO_GET_STATUS, &stat);
```

- サンプリング開始
  サンプリング開始を指示する。第 2 引数には TDSIO_SAMPLING_START を指定する。

  （使用例）

```
ioctl(fd, TDSIO_SAMPLING_START);
```

[3] 得られるステータス情報はサンプラーの持つレジスタから得られる値そのものになる。

- サンプリング停止
  サンプリング停止を指示する。第 2 引数には TDSIO_SAMPLING_STOP を指定する。
  なお、ホスト PC 上においてデバイスを open()しているプロセスが全て終了した場合には、サンプリングは自動的に停止される。

  （使用例）

```
ioctl(fd, TDSIO_SAMPLING_STOP);
```

- DC オフセット設定
  現在の DC オフセット値を設定する。第 2 引数には TDSIO_SET_DCOFFSET を、第 3 引数には設定する情報を格納する構造体(struct tdsio)のアドレスを指定する。DC オフセット(1CH)の値を取得したい場合は、struct tdsio のメンバ key に 0 を、2CH の値を取得したい場合は 1 を設定する。

  （使用例）

```
struct tdsio tdsio;

tdsio.key = 0; /* DCoffset 1 */
tdsio.value = 0x10;
ioctl(fd, TDSIO_SET_DCOFFSET, &tdsio);
```

- DC オフセット取得
  現在の DC オフセット値を設定する。第 2 引数には TDSIO_GET_DCOFFSET を、第 3 引数には設定する情報を格納する構造体(struct tdsio)のアドレスを指定する。DC オフセット(1CH)の値を取得したい場合は、struct tdsio のメンバ key に 0 を、2CH の値を取得したい場合は 1 を設定する。

  （使用例）

```
struct tdsio tdsio;

tdsio.key = 0; /* DCoffset 1 */
ioctl(fd, TDSIO_GET_DCOFFSET, &tdsio);

tdsio.key = 1; /* DCoffset 2 */
ioctl(fd, TDSIO_GET_DCOFFSET, &tdsio);
```

- サンプラーFIFO リセット
  サンプラー上の FIFO をリセットする。
  第 2 引数には TDSIO_BUFFER_RESET を指定する。

  （使用例）

```
ioctl(fd, TDSIO_BUFFER_RESET);
```

- 1PPS 入力状態チェック
  サンプラーへの 1PPS 入力状態をチェックする。
  第 2 引数には TDSIO_GET_1PPSINPUT を指定する。
  （使用例）

```
unsigned int pps;

ioctl(fd, TDSIO_GET_1PPSINPUT, &pps);

if (pps == 0) {
/* no 1PPS Input */
}
```

- 基準信号入力状態チェック
  サンプラーへの基準信号入力状態をチェックする。
  第 2 引数には TDSIO_GET_REFINPUT を指定する。
  （使用例）

```
unsigned int ref;

ioctl(fd, TDSIO_GET_REFINPUT, &ref);

if (ref == 0) {
/* no Ref Input */
}
```

- AUX データ設定
  AUX データを設定する。第 2 引数には TDSIO_SET_AUX を、第 3 引数には AUX データを格納する構造体(struct tdsauxio)のアドレスを指定する。メンバ auxsize に AUX データのサイズを、メンバ auxfield に AUX データを設定する。auxfield に設定できるデータは最大 256 バイトである。
  (使用例)

```
struct tdsauxio tdsauxio;

tdsauxio.auxsize = 10;
memcpy(tdsio.auxfield, "TEST AUX!" tdsauxio.auxsize);

ioctl(fd, TDSIO_SET_AUX, &tdsauxio);
```

- AUX データ取得
  AUX データを取得する。第 2 引数には TDSIO_GET_AUX を、第 3 引数には AUX データを格納する構造体(struct tdsauxio)のアドレスを指定する。メンバ auxsize に AUX データのサイズを、メンバ auxfield に AUX データが取得できる。
  (使用例)

```
struct tdsauxio tdsauxio;
char buf[256];

ioctl(fd, TDSIO_GET_AUX, &tdsauxio);
memcpy(buf, tdsio.auxfield, tdsauxio.auxsize);
```

- エラー情報取得

  デバイス操作において発生したエラー情報を取得する。

  第 2 引数には TDSIO_GET_ERROR を、第 3 引数にはエラー情報を格納すべき変数(32bit)のアドレスを指定する。ホスト PC がブートした直後、あるいは、4 初期化が実行された場合に、エラー情報がリセットされる。それ以降、エラーが発生した場合に、その種別に関する情報が保持され、さらに次のエラーが発生した場合には、エラー情報が上書きされる。従って、エラー情報取得を行った場合に得られる値は、最も最近に発生したエラーの種別を示すことになる。

  エラー種別は下記に定義している。[4]

エラー値	エラー種別
0	エラーなし。(正常)
1	以下の分類に当てはまらないエラー。
2	サポートしていない操作あるいはパラメータを指定した。
3	サンプリング開始状態で行うべき操作を、サンプリングが開始されていない状態で指定した。
4	サンプリング停止状態で行うべき操作を、サンプリングが開始されている状態で指定した。
5	サンプリング設定(周波数/データ幅/チャンネル数)がなされた状態で行うべき操作をサンプリング設定がされていない状態で指定した。
6	ボード上の FIFO がオーバーフローした。
7	外部 1PPS 入力信号が検出できなかった。
8	外部 10MHz 入力信号が検出できなかった。
9	外部時刻入力信号が検出できなかった。
10	1PPS 同期がなされた後で行うべき操作を、1PPS 同期されていない状態で指定した。
11	時刻同期(あるいは時刻設定)がなされた後で行うべき操作を、時刻同期(あるいは時刻設定)されていない状態で指定した。
12	他のプロセスなどが read()を実行している状態で、さらに read()を行おうとした。
13	PCI ボード用の拡張ボードが接続されていないエラー。
14	DMA バッファが無くなった。
15	ユーザランドとの I/O 処理中でエラーになった。
16	DMA 処理がエラーになった。
17	データサンプラーから一定時間内に応答が無かった。
18	ワード単位でアラインされていないデータを検出した。
19	サンプリングデータ内で想定した位置にヘッダが見つからなかった。
20	サンプリング開始時のゴミデータの中にもヘッダと認識可能なデータが存在した。

[4] vlbi-usb3-linux/include/u3tdsio.h 参照。

（使用例）

```
int err;
ioctl(fd, TDSIO_GET_ERROR, &err);
```

- レジスタ読み出し
  任意のレジスタの値を読み出す。第 2 引数には TDSIO_REGIO_READ を、第 3 引数にはレジスタの値を格納する構造体(struct tdsio_regio)のアドレスを指定する。読み出し後、メンバ address にレジスタアドレスが、メンバ value に現在の値が格納される。
  （使用例）

```
struct tdsio_regio region;

ioctl(fd, TDSIO_REGIO_READ, &region);
printf("address=0x%02x, value=0x%02x¥n", region.address, region.value);
```

- バージョン情報読み出し
  ドライバのバージョン情報値を読み出す。第 2 引数には TDSIO_GET_VERSION を、第 3 引数にはバージョン情報値を格納する変数のアドレスを指定する。
  バージョン情報は、メジャー、マイナー、リビジョン番号の三つの値からなる。それぞれの番号を取り出すには次のマクロを使用する。

TDS_VERSION_MAJOR()	メジャー番号を取り出すマクロ
TDS_VERSION_MINOR()	マイナー番号を取り出すマクロ
TDS_VERSION_REVISION()	リビジョン番号を取り出すマクロ

（使用例）

```
uint32_t version;
ioctl(fd, TDSIO_GET_VERSION, &version);
printf("version=%02d.%02d.%02d¥n", TDS_VERSION_MAJOR(version),
    TDS_VERSION_MINOR(version), TDS_VERSION_REVISION(version));
```

8 サンプルプログラム

サンプルプログラムとして vlbi-usb3-linux/example 以下にある sample.c を参照してください。