K5 技術資料
2007/03/03 改
T.KONDO

K5/VSSP32 ユニットチェック手順書

１．ドライバーの更新

ドライバーの更新手順は以下の通りである
（スーパーユーザにて作業を行うこと）

vlbi のホームディレクトリにて

tar xvzf vlbi-usb-linux-YYYYMMDD.tar.gz　　(YYYYMMDD はアーカイブの年月日)

vlbi-usb-linux というディレクトリに上書きで展開される

```
# cd vlbi-usb-linux
# make
# make install

#cd /dev
# mknod -m 666 utds0 c 180 222

# insmod /lib/modules/2.6.8-2-386/kernel/drivers/usb/misc/utds.ko
```

ここでエラーが出るときは、 # rmmod utds.ko を行ってから insmod を再び行う

２．ソフトウェアの再コンパイル

src ディレクトリにて

```
k56a> make clean
k56a> make -f makefile.vssp32
```

３．ＶＳＳＰ３２ユニットのテスト

テスト用のシェルスクリプト vssp32test.sh、vssp32test2.sh または vssp32test3.sh を使用してテストを行う。それぞれのシェルスクリプトは引数無しで実行すると簡単な使用法が表示される。

それぞれのテストのフローチャートを図１および２に示す。なお vssp32test3.sh は vssp32test.sh のサンプリング周波数の下限をオペレータがセットできるようにしたものである。

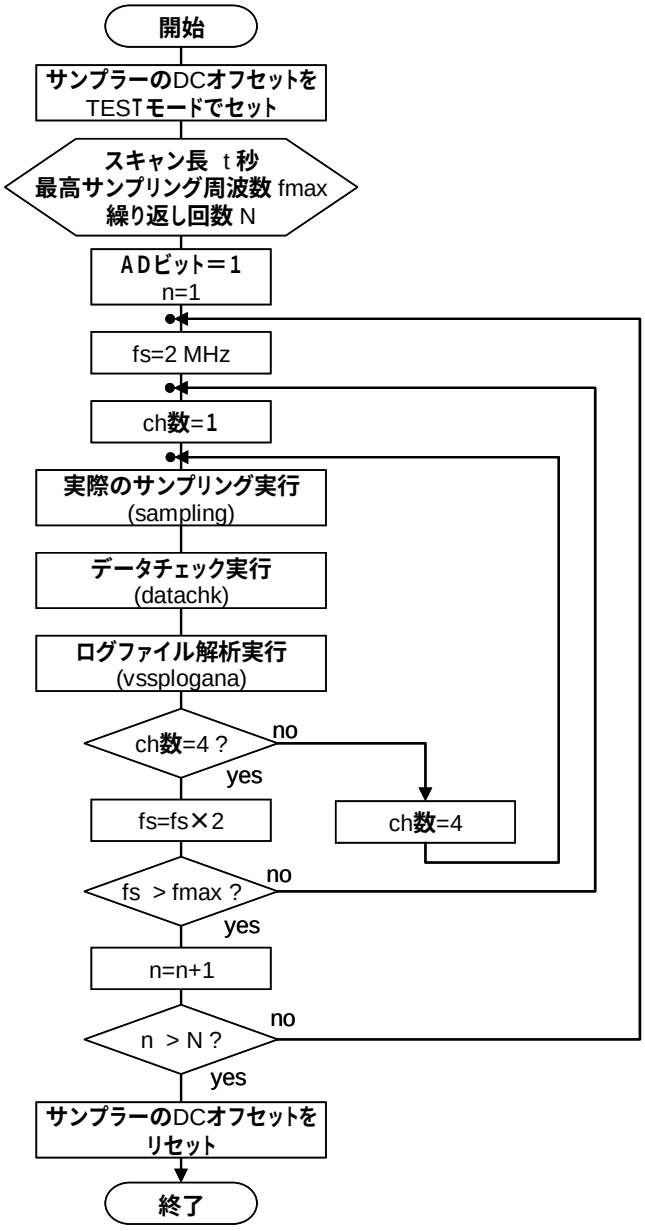

図 1 vssp32test.sh のフローチャート

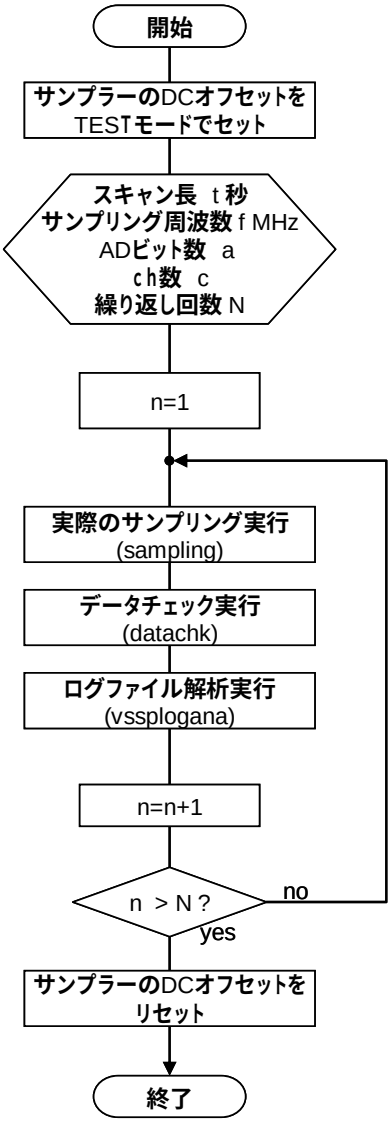

図２ vssp32test2.sh のフローチャート

３．１　サンプリング周波数を変えての自動テスト(vssp32test.sh)

src　ディレクトリにて

テストプロシジャー　vssp32test.sh を動かす
（vssp32test.sh に実行権限を与えておくこと！）

使用法
./vssp32test.sh span [nkai [sfmax [folder [logfile1 [logfile2 [logfile3]]]]]

ここで　span　--- スキャン長(sec)
nkai　--- 一連のテストの繰り返し回数　（デフォルトは１）
sfmax　--- 最大のサンプリング周波数(2,4,8,16,32,64)
（デフォルトは 64)
folder --- データを書き込むディレクトリ
（デフォルトは現在のディレクトリ）
logfile1 -- datachk の結果を出力するファイル名
（デフォルトは v32dtchkYYMMDDHHMMSS.HOST.log）
logfile2 -- sampling プログラムのログ出力ファイル名
（デフォルトは v32smplgYYMMDDHHMMSS.HOST.log）
logfile3 -- datachk のエラーログ出力ファイル名。エラーのあった
データファイル名情報が出力される。
（デフォルトは v32errorYYMMDDHHMMSS.HOST.log）
ここで、YYMMDDHHMMSS はチェック開始時の時刻
HOST はＰＣのホスト名

＊ 一連のテストとはサンプリング条件を

サンプリング周波数　２，４，８，１６，３２，６４ＭＨｚ
（オペレータが sfmax を与えたときはその値が最大値となる）
ＣＨ数　１，４
ＡＤビット数　１ビット　固定

の１２通りに変えながら span で与えられた時間ずつ sampling によるデータ収集と datachk によるデータチェックおよび vssplogana によるログファイルの解析を繰り返す。

span を３００秒とするとデフォルトパラメータ時の一連のテストを行う時間は約３００秒×１２＝１時間となる。

注１：テスト開始時に sampling のログファイル中の時刻（ＰＣ時刻を使用）とサンプリング中のデータの時刻の同期をとるため、サンプラーボードの時刻はＰＣの時間を使ってセットされる(timesetpc を使用)。

例１　k56a> vssp32test.sh 300 24
上記は各スキャン長３００秒で一連のテストを２４回繰り返す。
所要時間の概算はデフォルトパラメータの時は
所要時間＝ span×12×nkai　（秒）

で求めることができる。例１の場合、約２４時間となる
ホストＰＣが k56a だった場合、以下のような２つのログファイルが作成される
v32dtchk070209132512.k56a.log <= datachk のサマリーログ
v32smplg070209132512.k56a.log <= sampling のサマリーログ
v32error070209132512.k56a.log <= datachk のエラーファイルログ
ただし、テスト開始の時間が 2007 年 2 月 9 日 13:25:12 だったとする

エラーログファイル以外のログファイルは自動的に１スキャンごとに解析されるが、「４．ログファイルの解析」の章で述べる方法で独立に行うことも可能である。

３．２　サンプリングパラメータを固定しての耐久テスト(vssp32test2.sh)

src　ディレクトリにて

テストプロシジャー　vssp32test2.sh を動かす
（vssp32test2.sh に実行権限を与えておくこと！）

使用法
./vssp32test2.sh span [sfreq [adbit [ch [nkai [folder [logfile1 [logfile2 [logfile3]]]]]]]

ここで　span　--- スキャン長(sec)
sfreq --- サンプリング周波数(MHz) （デフォルトは 32）
adbit --- ＡＤビット数　1,2,4,8 （デフォルトは１）
ch --- ｃｈ数　1,4 （デフォルトは１）
nkai --- スキャン数 （デフォルトは１）
folder --- データを書き込むディレクトリ
（デフォルトは現在のディレクトリ）
logfile1 -- datachk の結果を出力するファイル名
（デフォルトは v32dtchkYYMMDDHHMMSS.HOST.log）
logfile2 -- sampling プログラムのログ出力ファイル名
（デフォルトは v32smplgYYMMDDHHMMSS.HOST.log）
logfile3 -- datachk のエラーログ出力ファイル名。エラーのあったデータファイル名情報が出力される。
（デフォルトは v32errorYYMMDDHHMMSS.HOST.log）
ここで、YYMMDDHHMMSS はチェック開始時の時刻
HOST はＰＣのホスト名

オペレータで与えられたサンプリングパラメータで span で与えられた時間ずつ sampling によるデータ収集と datachk によるデータチェックおよび vssplogana によるログファイルの解析を繰り返す。

注１：テスト開始時に sampling のログファイル中の時刻（ＰＣ時刻を使用）とサンプリング中のデータの時刻の同期をとるため、サンプラーボードの時刻はＰＣの時間を使ってセットされる(timesetpc を使用)。

例１　k56a> vssp32test2.sh 300 32 1 4 24
上記は各スキャン長３００秒で 32MHz×1bit×4ch モードのサンプリング２４回繰り返す。
所要時間の概算は　　所要時間＝ span×nkai　（秒）
で求めることができる。例１の場合、約 120 分となる
ホストＰＣが k56a だった場合、以下のような２つのログファイルが作成される
v32dtchk070209132512.k56a.log　<=　datachk のサマリーログ
v32smplg070209132512.k56a.log　<=　sampling のサマリーログ
v32error070209132512.k56a.log　<=　datachk のエラーファイルログ
ただし、テスト開始の時間が 2007 年 2 月 9 日 13:25:12 だったとする

エラーログファイル以外のログファイルは自動的に１スキャンごとに解析されるが、「４．ログファイルの解析」の章で述べる方法で独立に行うことも可能である。

３．３　サンプリング周波数を変えての自動テスト(vssp32test3.sh)。周波数の範囲を設定。vssp32test.sh とほぼ同じだが、最低のサンプリング周波数を引数に追加

src　ディレクトリにて

テストプロシジャー　vssp32test3.sh を動かす
（vssp32test3.sh に実行権限を与えておくこと！）

使用法
./vssp32test3.sh span [nkai [sfmin [sfmax [folder [logfile1 [logfile2]]]]]

ここで　span　--- スキャン長(sec)
nkai　--- 一連のテストの繰り返し回数　（デフォルトは１）
sfmin　--- 最低のサンプリング周波数(2,4,8,16,32,64)
（デフォルトは 2)
sfmax　--- 最大のサンプリング周波数(2,4,8,16,32,64)
（デフォルトは 64)
folder --- データを書き込むディレクトリ
（デフォルトは現在のディレクトリ）
logfile1 -- datachk の結果を出力するファイル名
（デフォルトは v32dtchkYYMMDDHHMMSS.HOST.log)
logfile2 -- sampling プログラムのログ出力ファイル名
（デフォルトは v32smplgYYMMDDHHMMSS.HOST.log)
ここで、YYMMDDHHMMSS はチェック開始時の時刻
HOST はＰＣのホスト名

例１　k56a> vssp32test.sh 10 24 16
上記は各スキャン長１０秒でサンプリング周波数を 16,32,64MHz と変えながら一連のテストを２４回繰り返す。

４．ログファイルの解析

ログファイルはプログラム vssplogana を実行して解析する。

vssplogana の実行方法

```
vssplogana logfile1 [logfile2 [ … ]]
    ここで  logfile1, logfile2 ・・・は
            datachk のサマリー出力ファイルまたは sampling の
            ログファイルを指定する。
```

vssplogana は datachk のサマリー出力ファイルおよび sampling（autoobs も可）のログファイルの解析を行う。ログファイルの形式は自動的に判定され、datachk の出力に対しては、スキャン中のエラーのあったフレーム数の統計結果を表示し、sampling（または autoobs）のログファイルに対してはサンプリング中のエラーの発生件数に関する統計結果を表示する。

WINDOWS の PV-WAVE にてもログファイルの解析は可能である。その際は、机上 VAIO の D:¥IPVLBI¥pvwave 以下にて行う。使用する PV-WAVE プログラムとログファイルの対応は以下の通り

vssp32check.pro　　-- datachk が出力するログファイルの解析
vssp32log_anal.pro -- sampling が出力するログファイルの解析
D:¥IPVLBI¥utility¥k5vssp32¥check　--- ログファイル用フォルダ

４．１　統計結果の例

４．１．１　sampling ログファイルの場合

[解析例]

```
******** SUMMARY OF K5/VSSP32 SAMPLING LOG FILE ANALYSIS **************
  FILE NAME          : v32smplg070207231200.k56b.log
  SCAN LENGTH (sec) : MIN =   300       MAX =   300
  TOTAL # of SCANS  :      305
  START TIME         : 2007/02/07 (038) 23:12:50
  END   TIME         : 2007/02/09 (040) 00:59:06
  PERIOD (hour)      :       25.7711
================================================================================
   SAMPLER MODE   # of SCANs              # of ERRORS
 fs(MHz) #AD #CH   ALL   Bad       ALL   Err16   Err19   Err20  Others
--------------------------------------------------------------------------------
    2     1   1    26     0         0       0       0       0       0
    2     1   4    26     0         0       0       0       0       0
    4     1   1    26     1         1       0       0       1       0
    4     1   4    26     0         0       0       0       0       0
    8     1   1    26     0         0       0       0       0       0
    8     1   4    25     0         0       0       0       0       0
   16     1   1    25     0         0       0       0       0       0
   16     1   4    25     0         0       0       0       0       0
   32     1   1    25     0         0       0       0       0       0
```

```
  32     1   4    25     0          0       0       0       0       0
  64     1   1    25     1          1       0       0       1       0
  64     1   4    25     0          0       0       0       0       0
=================================================================
 Note: Err16  -- DMA process error
       Err19  -- Header is not found at expected position in data
       Err20  -- Header-like data is found in first garbage data
       Others -- Other error code
```

[解釈]

4MHz×1bit×1ch モードおよび 64MHz×1bit×1ch モードでエラーコード２０が１回発生している。それ以外のエラーはない。

注：2007.3.2 以降のドライバーではエラー１９，２０の発生を抑制しているのでこのエラーがでることはない。

４．１．２　datachk サマリー出力の場合

[解析例]

```
*************** SUMMARY OF DATACHK LOG FILE ANALYSIS ********************
  FILE NAME          : v32dtchk070207231200.k56b.log
  SCAN LENGTH (sec)  : MIN =   300       MAX =   301
  TOTAL # of SCANS   :      304
  START TIME         : 2007/02/07 (038) 23:12:52
  END   TIME         : 2007/02/09 (040) 00:59:02
  PERIOD (hour)      :       25.7694
=============================================================================
   SAMPLER MODE   # of SCANs   TOTAL              # of BAD FRAMES
 fs(MHz) #AD #CH   ALL  Herr   FRAME   SLIP  T_DIS T_W_BS AUXMIS   EFLG  B_REV
-----------------------------------------------------------------------------
    2     1   1     0     0       0      0      0      0      0      0      0
    2     1   1    26     0    7800      0      0      0      0      0      0
    2     1   4    26     0    7800      0      0      0      0      0      0
    4     1   1    26     0    7801      1      0      0      0      0      0
    4     1   4    26     0    7800      0      0      0      0      0      0
    8     1   1    25     0    7500      0      0      0      0      0      0
    8     1   4    25     0    7500      0      0      0      0      0      0
   16     1   1    25     0    7500      0      0      0      0      0      0
   16     1   4    25     0    7500      0      0      0      0      0      0
   32     1   1    25     0    7500      1      0      0    300      0      0
   32     1   4    25     0    7500      0      0      0      0      0      0
   64     1   1    25     0    7501      1      0      0      0      0      0
   64     1   4    25     0    7500      0      0      0      0      0      0
=============================================================================
 Note: Herr   -- # of scans where 1st 64bit is not a header
       SLIP   -- # of frames where bit lack or make occurred
       T_DIS  -- # of frames where time discontinuity occurred
       T_W_BS -- # of frames where both time discontinuity and
                 bit slip or make occurred
       AUXMIS -- # of frames where AUX field misalignment occurred (data is OK)
```

```
EFLG   -- # of frames where EFLG (burst error) detected
B_REV  -- # of frames where one bit reversal in a sync field detected
```

[解釈]

4MHz×1bit×1ch モードおよび 64MHz×1bit×1ch モードでビットスリップまたはビットメークのあったフレームが１つある。それは sampling のエラー解析からこれらのモードでエラーコード２０が１回発生しているので、エラーコード２０発生に伴う再サンプリング開始により、フレーム内のデータが不連続となったためと思われる。

32MHz×1bit×1ch モードで AUXMIS(AUX フィールドの並び異常)のあったフレームが３００ある。そのモードでビットスリップまたはビットメークのあったフレームが１つであり、また１スキャンは３００秒（３００フレーム）なので、恐らくスキャンの最初のフレームでビットスリップまたはビットメークが発生し、その後スキャンの終わりまで AUXMIS が発生していると思われる。

Ａ．付録　単体プログラムの使い方

Ａ．１　sampling

機能

手動によるデータ収集

実行方法

```
sampling span sfreq adbit numch [filename [logfile]]
     ここで  span ---- 観測時間 (sec)
             sfreq ---- サンプリング周波数
                       40,100,200,500 (for kHz)
                       1,2,4,8,16 (for MHz)
             adbit ---- ＡＤ分解能（ビット）
                       1,2,4,8
             numch ---- チャネル数
                       1, 4
             filename --  データ出力ファイル名
                        デフォルトは tds.data
             logfile -- ログファイル名
                  デフォルトは sampling.log
                  デフォルトのログファイルは上書き
                  ユーザーが logfile パラメータで指定したログファイルには
                  ログはアペンドされる
```

例１　　サンプリング周波数４ＭＨｚ、２ビットＡＤ、４ｃｈ使用して１０秒間データ収集

```
sampling 10 4 2 4
```

例２　　サンプリング周波数８ＭＨｚ、４ビットＡＤ、４ｃｈ使用して２０秒間データ収集し、カレントディレクトリに　tds2.data というファイルを作成し、出力する

```
sampling 20 8 4 4 tds2.data
```

上記　例１，例２ではカレントディレクトリにログファイル sampling.log が作成されるが、その都度ファイルは上書きされる。上書きされないようにするには logfile パラメータでログファイル名を指定してやる。

例３　　サンプリング周波数４ＭＨｚ、２ビットＡＤ、４ｃｈ使用して１０秒間データ収集。出力ファイルを tds.dat、ログファイルを rei3.log とする。

```
sampling 10 4 2 4 tds.dat rei3.log
```

rei3.log というログファイルが作成されますが、２回目以降同じログファイルを指定して sampling を走らせると、ログはアペンドされていく。（この時、出力データファイル名は省略不可）

Ａ．２　datachk

機能

ヘッダー部を頼りにヘッダー間のデータビット数をカウントすることにより、ビットスリップまたはビットメイクがあったかどうかでデータをチェックする。またアナログ信号が＋のサインであった割合を%で表示することも可能である。更にエラーが起こったデータファイルの記録機能も有する。この機能は、サンプラーの連続試験時にエラーが起こったデータファイルだけ保存したい場合に便利な機能である。

データの中身については　例えば　ＯＳ付属の　od 等で確認する

実行方法

datachk file_name [mode [logfile [errlog [keepmode]]]]

ここで　file_name --- データファイル名　（デフォルトは tds.data）

mode　-------サンプリング統計（ゼロバランス）表示モード

0 ：サンプリング統計は表示しない（デフォルト）。最初と最後のフレーム情報およびエラーの生じたフレーム情報を表示

1 ：サンプリング統計を表示する

2 ： モードＯと同じ。ただし全フレーム表示する

logfile ---- モードＯの場合にチェック結果のサマリーを出力するファイル名。
このファイル名がの先頭が“-”の場合サマリー出力は既存のファイルに追加されていく。
デフォルトはサマリー出力なし。

errlog　---- モードＯの場合にエラーが発生したデータファイルの情報を出力するファイル名。このファイル名がの先頭が“-”の場合、出力は既存のファイルに追加されていく。
チェックしたデータファイルにエラーが無い場合は、このファイルは作成（出力）されない。
デフォルトはエラーログ出力なし。

keepmode --- モードＯの場合にエラーの生じたデータファイルを保存するモード

0 ： 何もしない（デフォルト）

1 ： データファイルの名前を元の名前＋”.NNNN.err”に変更する。

2 ： データファイルを　　元の名前＋”.NNNN.err”にコピーする。

ここで NNNN は 0001 から 9999 で繰り返す
この通し番号は datachk を実行するディレクトリ下の“counter_file_datachk.tmp”という名のテキストファイルで管理する。

ゼロバランス表示はアナログ信号が＋のサインであった割合を%で表示

サマリーファイル

サマリーファイルの例を以下に示す

File Name:

```
D:¥IPVLBI¥data¥test02.dat
# FMT  A/D CH  f(kHz)  LPF(MHz):
  VS32  1   1   32000       16
# Start and Last Time:
  2006/318 23:20:28  84028
  2006/318 23:25:27  84327
# Duration:
300
# Byte offset of 1st header:
0
# STATISTICS total bad discon discon_with_bitslip aux_err EFLG:
300 1 0 0 147 0
# BIT SLIP:
1 26432
```

エラーログファイル

エラーログファイルの例を以下に示す

keepmode=0 の場合

```
# Errored Data File Name:
test02.dat
# FMT  A/D CH  f(kHz)  LPF(MHz):
  VS32  1   1   32000       16
```

keepmode=1 の場合（リネームモード）

```
# Errored Data File Name:
test02.dat
# FMT  A/D CH  f(kHz)  LPF(MHz):
  VS32  1   1   32000       16
# Renamed to:
test02.dat.0006.err
```

keepmode=2 の場合（コピーモード）

```
# Errored Data File Name:
test02.dat
# FMT  A/D CH  f(kHz)  LPF(MHz):
  VS32  1   1   32000       16
# Copied to:
test02.dat.0007.err
```

Ａ．３　timesetpc

機能

ＰＣの時刻を使って K5/VSSP（K5/VSSP32）ボードの時刻をセットする。ボードチェック用であり、ＶＬＢＩ観測時には timesetpc は決して使用しないこと！）

実行方法

timesetpc 1

引数の１をつけずに timesetpc のみとすると使用方法が出る

環境変数のモニター　timesetpc env

環境変数

K5LOGDIR　　---　ログディレクトリの指定
プログラムデフォルトはカレントディレクトリ

ログ出力

ディレクトリ　--　環境変数 K5LOGDIR でセットしたディレクトリ
K5LOGDIR がセットされていないときはカレントディレクトリ
ファイル名　　--　YYYY+ホスト名+".log"　　例：2003k51d.log
出力形式　　--　timesettk の項を参照のこと
出力例

```
07038231259:p timesetpc(Ver. 2007-02-07) 2007 2 7 23 12 59
07038231259/  timesetpc (board time set using PC time is OK)
```

Ａ．４　vssplogana

機能

sampling（autoobs も可）のログファイルおよび datachk のサマリー出力ファイルを解析し、発生エラー状況の統計結果を表示する。ログファイルの形式は自動的に判定され、datachk の出力に対しては、スキャン中のエラーのあったフレーム数の統計結果を表示し、sampling（または autoobs）のログファイルに対してはサンプリング中のエラーの発生件数に関する統計結果を表示する。

実行方法

vssplogana logfile1 [logfile2 […]]
ここで　logfile1, logfile2 ・・・は
datachk のサマリー出力ファイルまたは sampling(autoobs)ログファイル名

Ａ．５　setdcoffset

機能

K5/VSSP32 サンプラーのＤＣオフセットをセットする。任意の値にセットする機能、初期値（０）に戻す機能、ユニット試験に適した値にセットする機能、観測に適した値にセットする機能を有する。ログ出力も行う。K5/VSSP に対してもどのモードも STAT（信号の統計出力）モードと見なされる。

実行方法

```
setdcoffset dc1 dc2 dc3 dc4 [logfile]
```
または
```
setdcoffset TEST|AUTO|OBS|STAT|RESET [logfile]
```

ここで dc1 ---- CH1 にセットする DC オフセット (-128 から 128)
dc2 ---- CH2 にセットする DC オフセット (-128 から 128)
dc3 ---- CH3 にセットする DC オフセット (-128 から 128)
dc4 ---- CH4 にセットする DC オフセット (-128 から 128)
TEST --- サンプラーボードの信頼性試験を行う際に適したオフセット
を自動的にセットする。
AUTO --- 観測に適したオフセットを自動的にセットする。
OBS --- AUTO と同じ
STAT --- 信号の統計結果（平均、標準偏差）のみを出力する
RESET - DC オフセットを０にセットする
logfile -- ログファイル名
デフォルトは dcoffset.log
デフォルトのログファイルは上書き。
ユーザーが logfile パラメータで指定したログファイルには
ログはアペンドされる

ログ出力

例１．チャンネル毎に DC オフセット値を指定して走らせたとき

```
07047011017:p setdcoffset: set 5  10  -5  -10
07047011017:p setdcoffset: VSSP32 OFFSET CH1 = 5  CH2 = 10  CH3 = -5  CH4 = -10
07047011017:p setdcoffset:                        Signal Statistics
07047011017:p setdcoffset:              CH1        CH2        CH3        CH4
07047011017:p setdcoffset: average     5.8/256   10.7/256   -4.8/256   -9.4/256
07047011017:p setdcoffset: std dev     0.5/256    0.4/256    0.5/256    0.3/256
```

最後の２行は信号の平均値(average)と標準偏差(std dev)を表示している。信号のレベルは-127.5～+127.5 でさる。この例は、入力信号を入れていないときであり、標準偏差が小さくなっている。

例２．TEST モードで走らせたとき

```
07047011429:p setdcoffset: set TEST mode
07047011429:p setdcoffset: VSSP32 OFFSET CH1 = -6  CH2 = -6  CH3 = -6  CH4 = -6
07047011429:p setdcoffset:                        Signal Statistics
07047011429:p setdcoffset:              CH1        CH2        CH3        CH4
07047011429:p setdcoffset: average    -5.2/256   -5.3/256   -5.9/256   -5.4/256
07047011429:p setdcoffset: std dev     0.5/256    0.4/256    0.5/256    0.3/256
```

自動的に DC オフセットが-6 にセットされているのがわかる。これは、入力がない場合に、１ビットサンプリング後の信号をすべて０として、ヘッダー部の誤動作を起こりにくくするためである。まお、通常の信号が入っている場合は、単に平均値が０となるように、DC オフセットが自動的に調整される。

例３．AUTO モードで走らせたとき

```
07047011806:p setdcoffset: set AUTO mode
07047011806:p setdcoffset: VSSP32 OFFSET CH1 = 0  CH2 = 0  CH3 = 0  CH4 = 0
07047011806:p setdcoffset:                        Signal Statistics
07047011806:p setdcoffset:              CH1        CH2        CH3        CH4
07047011806:p setdcoffset: average     0.8/256    0.7/256    0.0/256    0.6/256
```

07047011806:p setdcoffset: std dev　　0.5/256　　0.4/256　　0.5/256　　0.3/256

平均値が０になるように自動的にDCオフセットが調整される。

Ｂ．付録　各シェルスクリプトの中身

vssp32test.sh

```
#!/bin/sh
#

if [ -z $1 ]; then   #  display how to use
echo '

# K5/VSSP32 check procedure by T.Kondo 2007-02-16
#
#  How to execute
#
#   ./vssp32test.sh span [nkai [sfmax [data_folder
#         [datachk_log_file [sampling_log_file
#         [error_log_file ]]]]]
#       where
#          span  --------  sampling period for a scan (sec)
#          nkai     ----- # of repeat   (default is 1)
#          sfmax    ----- maximum sampling frequency 2,4,8,16,32,64
#                           (default is 64)
#          data_folder -- folder where sampled data file is created
#                          (default is current directory)
#          datachk_log_file -- file name for 'datachk' summary output
#                               (default is "v32dtchkYYMMDDHHMMSS.HOST.log")
#          sampling_log_file ---- log file name for 'sampling' program
#                               (default is  "v32smplgYYMMDDHHMMSS.HOST.log")
#          error_log_file  ---- error log file name for 'datachk' program
#                               (default is  "v32errorYYMMDDHHMMSS.HOST.log")
#              where YYMMDDHHMMSS is start time of this shell execution
#                    HOST is a host PC name
#
#   Check items
#         sampling frequency :  2, 4, 8, 16, 32, 64 MHz
#         AD bits            :  1
#         Channels           :  1, 4
#         Scan length        :  given by operator
#         # of repeat        :  1 or given by operator
#
#   Checking above all items (frequency max is sfmax it is given by
#   an operator) will be repeated 'nkai' times
#
#   Typical execution example
#      ./vssp32test.sh 300 24
#         ==>  this will give 24 hour check of 300 sec scans.

'

else
```

```
# actual execution
pdir="./"        # program directory
spl="sampling"
chk="datachk"
ana="vssplogana"
dco="setdcoffset"
ad=1
count=0
tscan=0
span=$1     # sampling span (sec)
nkai=$2     # repeat number
sfmax=$3    # sampling frequency max
d_dir=$4    # data out folder
d_log=$5    # log file name for 'datachk'
s_log=$6    # log file name for 'sampling' and 'setdcoffset'
e_log=$7    # errpr log file name for 'datachk'

# get system time as YYMMDDHHMMSS
ymd=`date +%y%m%d%H%M%S`
# get a host PC name
hn=`uname -n`

if [ $1 = 0 ]; then  # $1 is 0
   span=300
fi
if [ -z $2 ]; then  # $2 is null
   nkai=1
fi
if [ -z $3 ]; then  # $3 is null
   sfmax=64
fi
if [ -z $4 ]; then  # $4 is null
   d_dir="./"
fi
if [ -z $5 ]; then  # $5 is null
   d_log="v32dtchk"$ymd".$hn.log"
fi
if [ -z $6 ]; then  # $6 is null
   s_log="v32smplg"$ymd".$hn.log"
fi
if [ -z $7 ]; then  # $7 is null
   e_log="v32error"$ymd".$hn.log"
fi
echo 'span  = ' $span
echo 'nkai  = ' $nkai
echo 'sfmax = ' $sfmax
echo 'data directory    is ' $d_dir
echo 'datachk  log file is ' $d_log
```

```
echo 'sampling and setdcoffset log file is ' $s_log
echo 'datachk error log file is ' $e_log

# delete sampling log file just in case
# echo 'delete existed sampling log file (' $s_log ')'
# rm $s_log

# set board time using PC time to synchronize time in log file
./timesetpc 1

# set DC affset adequate for testing
echo $pdir$dco TEST $s_log
$pdir$dco TEST $s_log

# loop counter starts from 1
count=`expr $count + 1`

# repeating kaisu
while [ $count -le $nkai ];
do

   # Loop in sampling frequency
   sf=2
   # for sf in 2 4 8 16 32 64
   while [ $sf -le $sfmax ];
   do
       #  Loop in Channel
       for ch in 1 4
       do
         echo '******** Loop# ' $count 'out of ' $nkai '********'
         echo $sf'MHz '$ad'bit '$ch'ch '$span'sec testing...'
         echo $pdir$spl -$span $sf $ad $ch $d_dir/test$ch.dat $s_log
         $pdir$spl -$span $sf $ad $ch $d_dir/test$ch.dat $s_log
         if [ $tscan -eq 0 ]; then
            # datacheck log file newly created
            echo     $pdir$chk $d_dir/test$ch.dat 0 $d_log $e_log 1
            $pdir$chk $d_dir/test$ch.dat 0 $d_log $e_log 1
         else
            # datacheck log file open as append mode
            echo     $pdir$chk $d_dir/test$ch.dat 0 -$d_log -$e_log 1
            $pdir$chk $d_dir/test$ch.dat 0 -$d_log -$e_log 1
         fi
         # log file analysis
         echo $pdir$ana $s_log $d_log
         $pdir$ana $s_log $d_log
         tscan=`expr $tscan + 1`
       done
       sf=`expr $sf ¥* 2`    #  sf=sf*2
   done
```

```
   count=`expr $count + 1`
done

# reset DC affset adequate for testing
echo $pdir$dco RESET $s_log
$pdir$dco RESET $s_log

# end of actual execution
echo ' '
echo 'VSSP32 TEST completed.'
echo '   sampling log file        : '$s_log
echo '   datachk summary log file : '$d_log
echo '   datachk error   log file : '$e_log
echo '  You can execute '$ana' again to analyze above log files as follows.'
echo '   '$pdir$ana $s_log $d_log
echo '  See '$e_log' for kept errored data files.'
echo ' '
fi
# end of all shell script
```

vssp32test2.sh

```
#!/bin/sh
#

if [ -z $1 ]; then   #  display how to use
        echo '

        # K5/VSSP32 check procedure 2 by T.Kondo 2007-02-16
        #
        #  How to execute
        #
        #   ./vssp32test2.sh span [sfreq [adbit [ch [nkai [data_folder
        #               [datachk_log_file [sampling_log_file
        #               [error_log_file ]]]]]
        #       where
        #          span  --------  sampling period for a scan (sec)
        #          sfreq --------  sampling frequency 2, 4, 8, 16, 32, 64 (MHz)
        #                            (default is 32)
        #          adbit --------  AD bits 1, 2, 4, 8  (default is 1)
        #          ch    --------  # of channels  1, 4  (default is 1)
        #          nkai     ----- # of repeat   (default is 1)
        #          data_folder -- folder where sampled data file is created
        #                           (default is current directory)
        #          datachk_log_file -- file name for 'datachk' summary output
        #                          (default is "v32dtchkYYMMDDHHMMSS.HOST.log")
```

```
        #          sampling_log_file ---- log file name for 'sampling' program
        #                          (default is  "v32smplgYYMMDDHHMMSS.HOST.log")
        #          error_log_file  ---- error log file name for 'datachk' program
        #                                                          (default  is
"v32errorYYMMDDHHMMSS.HOST.log")
        #              where YYMMDDHHMMSS is start time of this shell execution
        #                    HOST is a host PC name
        #
        #   Check items
        #         sampling frequency :  given by operator (default is 32 MHz)
        #         AD bits            :  given by operator (default is 1)
        #         Channels           :  given by operator (default is 1)
        #         Scan length        :  given by operator
        #         # of repeat        :  given by operator (default is 1)
        #
        #   Repeat 'nkai' times sampling with above fixed parameters
        #
        #   Execution example
        #      ./vssp32test2.sh 300 32 1 4 12
        #         ==>  this will last for about 1 hour.

        '

else

        # actual execution
        pdir="./"        # program directory
        spl="sampling"
        chk="datachk"
        ana="vssplogana"
        dco="setdcoffset"
        count=0
        tscan=0
        span=$1     # sampling span (sec)
        sf=$2       # sampling frequency
        ad=$3       # AD bits
        ch=$4       # channels
        nkai=$5     # repeat number
        d_dir=$6    # sampling data out directory
        d_log=$7    # log file name for 'datachk'
        s_log=$8    # log file name for 'sampling' and 'setdcoffset'
        e_log=$9    # errpr log file name for 'datachk'

        # get system time as YYMMDDHHMMSS
        ymd=`date +%y%m%d%H%M%S`
        # get a host PC name
        hn=`uname -n`

        if [ $1 = 0 ]; then  # $1 is 0
           span=300
```

```
fi
if [ -z $2 ]; then  # $2 is null
   sf=32
fi
if [ -z $3 ]; then  # $3 is null
   ad=1
fi
if [ -z $4 ]; then  # $4 is null
   ch=1
fi
if [ -z $5 ]; then  # $5 is null
   nkai=1
fi
if [ -z $6 ]; then  # $6 is null
   d_dir="./"
fi
if [ -z $7 ]; then  # $7 is null
   d_log="v32dtchk"$ymd".$hn.log"
fi
if [ -z $8 ]; then  # $8 is null
   s_log="v32smplg"$ymd".$hn.log"
fi
if [ -z $9 ]; then  # $9 is null
   e_log="v32error"$ymd".$hn.log"
fi
echo 'span  = ' $span
echo 'sfreq = ' $sf
echo 'adbit = ' $ad
echo 'ch    = ' $ch
echo 'nkai  = ' $nkai
echo 'data directory    is ' $d_dir
echo 'datachk  log file is ' $d_log
echo 'sampling and setdcoffset log file is ' $s_log
echo 'datachk error log file is ' $e_log

# delete sampling log file just in case
# echo 'delete existed sampling log file (' $s_log ')'
# rm $s_log

# set board time using PC time to synchronize time in log file
./timesetpc 1

# set DC affset adequate for testing
echo $pdir$dco TEST $s_log
$pdir$dco TEST $s_log

# loop counter starts from 1

count=`expr $count + 1`
```

```
        tscan=`expr $tscan + 1`

        # repeating kaisu
        while [ $count -le $nkai ];
        do
       # data file name generation
       if [ `expr $count % 2` -eq 1 ]; then
           datf=$d_dir/test1.dat
       else
           datf=$d_dir/test4.dat
       fi
           echo '******** Loop# ' $count 'out of ' $nkai '********'
           echo $sf'MHz '$ad'bit '$ch'ch '$span'sec testing...'
           echo $pdir$spl -$span $sf $ad $ch $datf $s_log
           $pdir$spl -$span $sf $ad $ch $datf $s_log
           if [ $tscan -eq 1 ]; then
              # datacheck log file newly created
              echo     $pdir$chk $datf 0 $d_log $e_log 1
              $pdir$chk $datf 0 $d_log $e_log 1
           else
              # datacheck log file open as append mode
              echo     $pdir$chk $datf 0 -$d_log -$e_log 1
              $pdir$chk $datf 0 -$d_log -$e_log 1
           fi
           # log file analysis
           echo $pdir$ana $s_log $d_log
           $pdir$ana $s_log $d_log
           tscan=`expr $tscan + 1`

           count=`expr $count + 1`
        done

        # reset DC affset adequate for testing
        echo $pdir$dco RESET $s_log
        $pdir$dco RESET $s_log

        # end of actual execution
        echo ' '
        echo 'VSSP32 TEST completed.'
        echo '   sampling log file        : '$s_log
        echo '   datachk summary log file : '$d_log
        echo '   datachk error   log file : '$e_log
        echo '  You can execute '$ana' again to analyze above log files as follows.'
        echo '   '$pdir$ana $s_log $d_log
        echo '  See '$e_log' for kept errored data files.'
    echo ' '
fi
# end of all shell script
```