TDC レポート 2003.1.9　(1.18 改)
T.KONDO

Mark5 データから K5 データへのフォーマット変換について

１．Mark5 データ

（注：ここで扱う Mark5 データは FTP 用のファイルに落とされたデータ）

Mark５データファイルは３２ビット（４バイト）がデータ単位となっている。１４ｃｈデータの場合には３２ビットの内、１４ビットが実際のｃｈに割り当てられる。その割り当ては、２００２年１０月に実施した e-VLBI 実験に関しては表１のようであった。

表１．Mark5 データのＣＨ番号とビット位置の対応表
（ＡＵＸ情報は Mark5 データヘッダー中の AUX データ）

ＣＨ＃ (1-14)	ＡＵＸ情報	ビット位置 (0-31)	ＣＨ＃ (1-14)	ＡＵＸ情報	ビット位置 (0-31)
１	１８００	１６	８	１００７	８
２	０４０１	２	９	２６０８	２４
３	２００２	１８	１０	１２０９	１０
４	０６０３	４	１１	２８０ａ	２６
５	２２０４	２０	１２	１４０ｂ	１２
６	０８０５	６	１３	３００ｃ	２８
７	２４０６	２２	１４	１６０ｄ	１４

表１の対応に従って、３２ビット単位毎にあるビット位置のデータを抜き出して並べるとそのビット位置に対応したＣＨデータの時系列が得られる（図１）

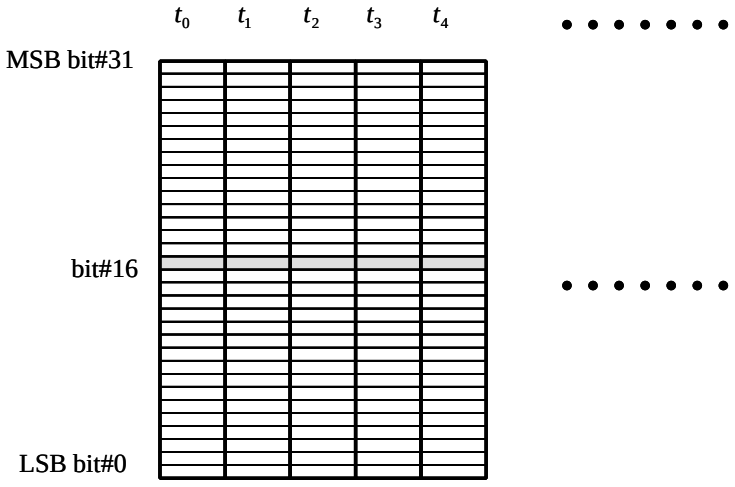

図　１　Mark5 データファイルの構造。例えばＣＨ＃１のデータ時系列を抜き出すには bit#16 の１ビット（斜線を施した部分を取り出せばよい。

こうして抜き出したデータは今回の実験の場合は Mark3 形式のフォーマットとなっていた。

２．Mark3 データフォーマット

Mark3 データの構成を図２に示す。１バイトはパリティ１ビットを伴った９ビットデータから構成され、４バイトで１ワードを構成し、５ワードで１ブロックを構成している。１２５ブロックが１フレームを構成するが、フレームの先頭はシンクパターンなどを含むヘッダーブロックとなっている。

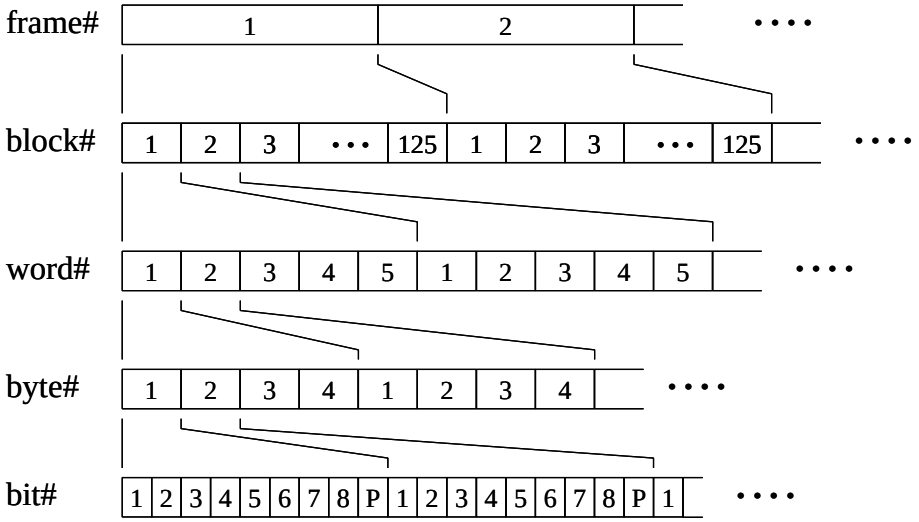

図２　Mark3 データフォーマット（テープフォーマット）

ヘッダーブロックにはシンクパターンの他、時刻情報および AUX 情報が含まれる（図３）。

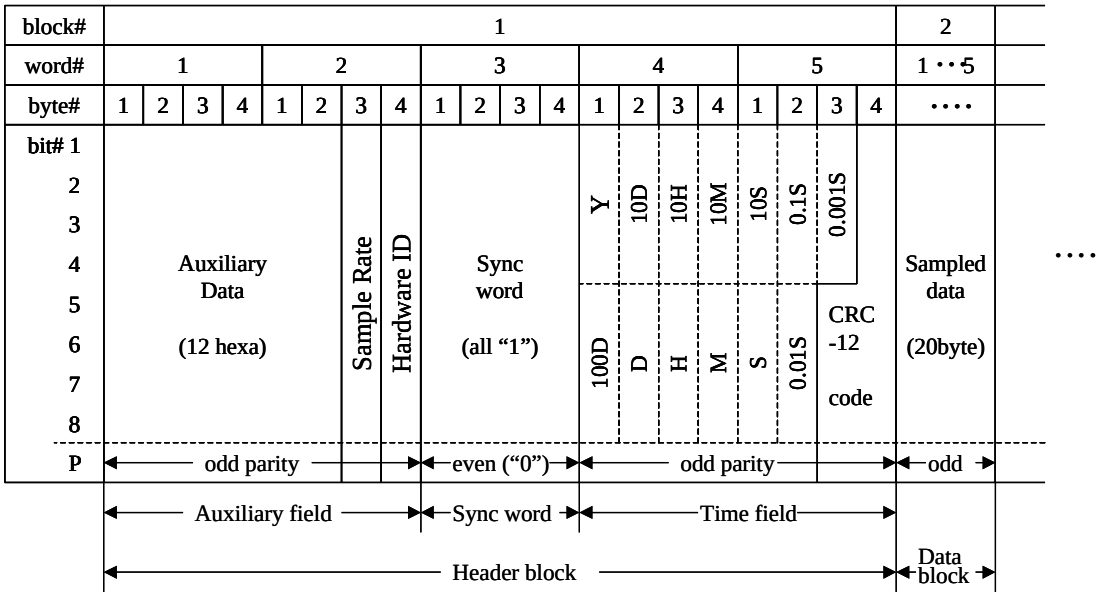

図３　Mark3 データ・ヘッダーブロックフォーマット

サンプリング周波数が４ＭＨｚの１ビットサンプリングデータの場合、１フレームは２２５００ビット（パリティビットを除くと２００００ビット＝5 msec）から構成され、先頭の１８０ビット（パリティビットを除くと１６０ビット）がヘッダーブロックとなる。へ

ッダーブロックはサンプリングデータが置き換えられる方式である。つまりその部分のサンプリングデータは欠損する。ＶＬＢＡフォーマットでは１フレームは２２６８０ビットと規定されているが、これはヘッダー１６０ビットが２００００ビットに付け加わったデータにパリティビットを付加したもの(20160×(9/8)=22680)に相当するため、ヘッダーブロックは挿入形式のようである。なお、ＩＰ－ＶＬＢＩ（Ｋ５）では、１秒ごとにヘッダー部分が挿入される形式であるため、サンプリングデータの欠損はない。

３．ＮＲＺ－ＬとＮＲＺ－Ｍ

Mark3 データフォーマットの構成は前章で示したが、デジタルデータのエンコード方式は NRZ-M 形式と規定されている。通常の１ビットサンプリングデータは NRZ-L（Non Return to Zero-Level）形式と呼ばれる信号で、レベルの１(high)と０(low)がデジタルデータの１と０に１対１に対応するエンコード方式である。これに対して NRZ-M （ Non Return to Zero-Mark）と呼ばれる形式はデータの変化に応じて１と０を対応づける方式で、現在の NRZ-M の状態に対してデータが変化した場合を１，変化は無い場合を０とするエンコード方式である。ちなみに、変化が無い場合を１，変化がある場合を０とするエンコード方式も存在し、NRZ-S（Non Return to Zero-Space）形式と呼ばれる。

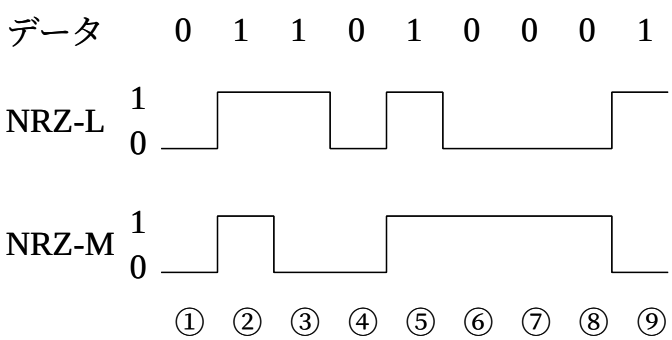

図 ４　NRZ-L と NRZ-M

図４にデータとそれに対応した NRZ-L､NRZ-M 信号を示す。シンクパターンを検出したり、相関処理に使うデータを復元するためには、ＮＲＺ－Ｍでエンコードされている信号をＮＲＺ－Ｌに変換しなくてはならないが、その変換の理解を助けるために、NRZ-L から NRZ-M への変換を詳細に説明しておこう。図４において、①→②では、NRZ-M が０の状態であったところに、データ１が入ってくるので、変化があるということで、NRZ-M は１となる。②→③では、NRZ-M が１の状態のところに、入力データも１であるので、状態に変化はないため、NRZ-M の定義（変化がない場合は０）に従い、０となる。③→④では、NRZ-M が０の状態のところに、入力データも０であるので、変化が無く NRZ-M は０と

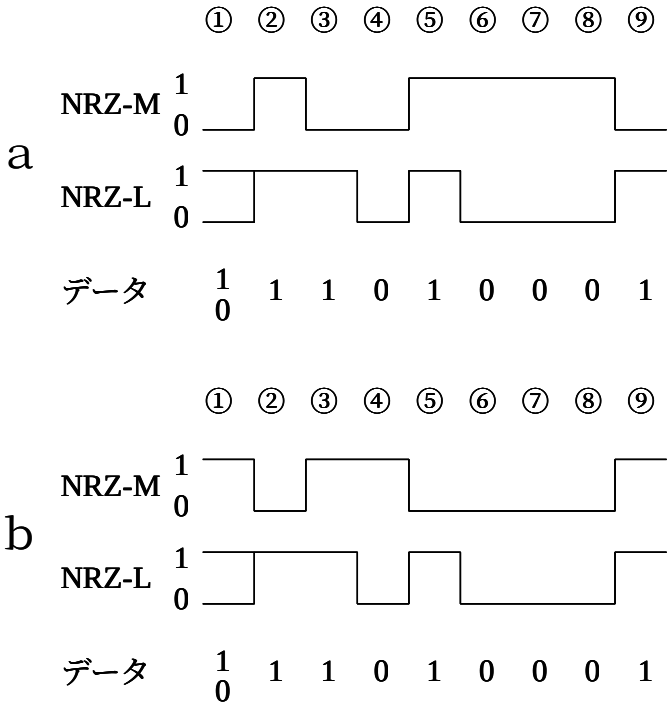

図 ５　NRZ-M から NRZ-L の復元

なる。以下、同様に NRZ-M の１，０が決まっていく。ここで注意しなくてはならないのは、NRZ-M は入力データの変化に対して１、０が定義されるのではなく、あくまでも NRZ-M の現在の状態に対して、入力データが変化しているかどうかで１，０が定義される点である。したがって、NRZ-M の初期値を０にするか、１にするかで、まったく１，０が反転した信号が得られる。

４．NRZ-M からデータを復元（デコード）する方法

前章で説明したエンコード法を逆に行うことにより、NRZ-L 信号を復元することができる。NRZ-L の最初のビットが不定であるが、図５ａに示されるように２ビット目以降からは正しいデータが復元される。１，０が反転した NRZ-M に対しても図５ｂに示されるように、２ビット目以降からは正しいデータが復元されることがわかる。図５ａについて、具体的にデコード法を説明しよう。図に示されるような NRZ-M が与えられた場合で、①→②では NRZ-M が０→１であるから、これは、NRZ-M の０に対して入力データが変化（つまり１）したことを示す。つまり②の区間でデータは１であることを意味する。②→③では、NRZ-M は１→０であるが、これは②における NRZ-M と同じレベルのデータすなわち１が入力データであることを意味する。③→④では NRZ-M は０→０であるが、これは、③における NRZ-M と同じレベルのデータすなわち０が入力データであることを意味する。以下、同様に考えていくことにより、データが確定していく。

１、０が逆転した NRZ-M に関しても、同様に考えていくと、同じデータが復元されることがわかる。

図６に、NRZ-M から NRZ-L に変換する関数のソースコード例を示す。

```
int nrzm2nrzl(int nrzmold, int nrzmnow)
// conversion from NRZM to NRZL
{
      int k2;
      k2=1;
      if(nrzmnow == nrzmold) k2=0;
      return k2;
}
```

図６　NRZ-M から NRZ-L へのデコード関数 nrzm2nrzl のソースコード

NRZ-L データに復元後、パリティビットを取り除くことにより、Ｋ５フォーマットに変換可能な Mark3 フォーマットデータを得ることができる。

５．Mark３フォーマットからＫ５(IP-VLBI)フォーマットへの変換

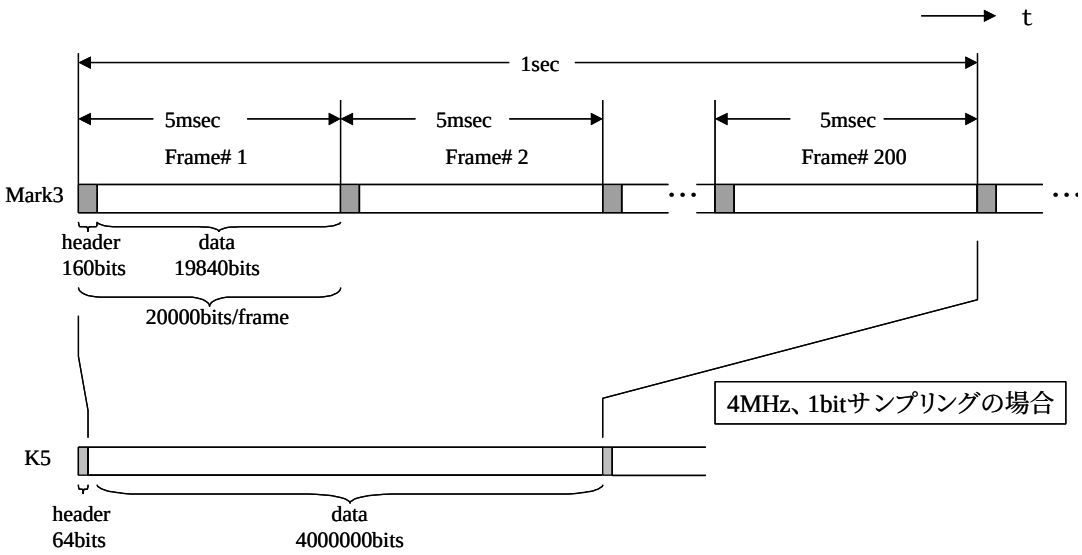

図 ７　Mark3 データと K5（１ｃｈモード）データの関係。4MHz, １ビットサンプリングデータの場合を示している。

Ｋ５フォーマットはファイル当たりのＣＨ数は１と４の２種類があるが、ヘッダーデータ（６４ビット）は共に１秒に１回挿入される。一方、Mark3 フォーマットではＣＨ毎に独立したデータであり、２００００ビット（パリティは取り除いている）で構成されるフレームデータの最初の１６０ビットがヘッダーデータである（サンプルデータのその区間のデータがヘッダーで置き換えられている)。したがって、Mark3 データを K5 データに変換する方法として、１ＣＨデータを抜き出して１つのＫ５データファイルに変換する方法と、４ＣＨデータを抜き出して１つのＫ５データとする方法が考えられるが、その両者とも可能な変換ソフトウェアを作成した。図７は４ＭＨｚ１ビットサンプリングの場合、Mark3 データを１ＣＨモードのＫ５データに変換した場合の関係を図示している。Ｋ５のデータ領域には Mark3 のヘッダーデータもそのまま含まれた形で変換される。Ｋ５の４ＣＨモードへの変換はもう少し複雑な関係となるが、Mark3 データから４ｃｈ分のデータを抜き出し、Ｋ５データフォーマットの定義に従って並べたものである。やはり Mark3 のヘッダーデータはＫ５データ領域にそのままの形で含まれる。

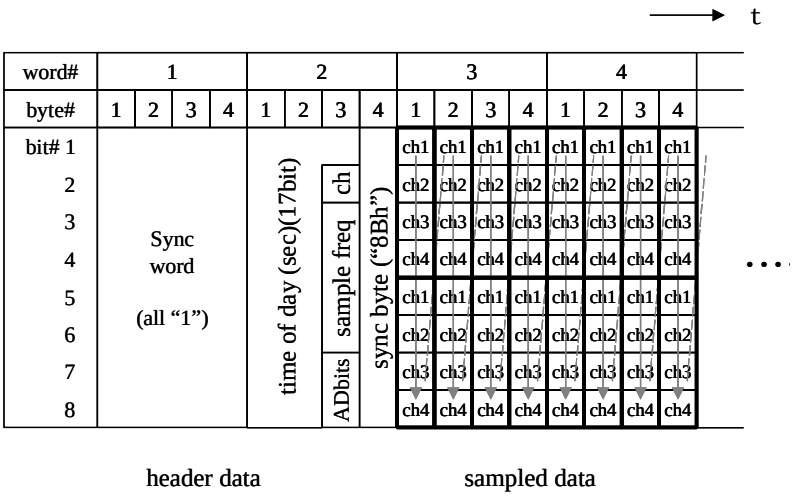

図 ８　Ｋ５データフォーマット。４ＣＨモード、１ビットサンプリング時。

図８に４ＣＨモー

ド、１ビットサンプリング時のＫ５データフォーマットを示す。時刻はデータ中の薄い矢印で示される方向（ＬＳＢからＭＳＢへ）に進む。１ＣＨモード時の場合も、ＬＳＢからＭＳＢ方向に時刻が進むようにサンプルデータが並ぶ。

６．フォーマット変換ソフトウェア m5tok5

Ｃ言語で Mark5 からＫ５へのフォーマット変換プログラム m5tok5 を作成した。使用法は以下の通りである。ここで、bit2 以降の指定を省略すると１ＣＨモードのＫ５データを作成し、すべてを指定すると４ＣＨモードのＫ５データが作成される。

```
走らせ方  m5tok5 mk5_filename k5_filename bit1 [bit2 bit3 bit4]

  ここで mk5filename -- Mark V データファイル名
        k5filename  -- K5 データファイル名
        bit1 --------- 抜き出すビット位置(0-31)
        bit2 --------- 抜き出すビット位置(0-31)
        bit3 --------- 抜き出すビット位置(0-31)
        bit4 --------- 抜き出すビット位置(0-31)
```