# 7 How to Decode an ADCP Ensemble

Use the following information to help you write your own software.

## 7.1 Rules for the BroadBand Data Format PD0

a. All data types (i.e. fixed leader, variable leader, velocity, echo intensity, correlation, percent good, etc.) will be given a specific and unique ID number. The table below shows some of the most common IDs.

**Table 46: Common Data Format IDs**

| ID | Description |
|---|---|
| 0x7F7F | Header |
| 0x0000 | Fixed Leader |
| 0x0080 | Variable Leader |
| 0x0100 | Velocity Profile Data |
| 0x0200 | Correlation Profile Data |
| 0x0300 | Echo Intensity Profile Data |
| 0x0400 | Percent Good Profile Data |
| 0x0500 | Status Profile Data |
| 0x0600 | Bottom Track Data |

b. Once a data type has been given an ID number and the format of that data has been published we consider the format for each field has being fixed. Fixed refers to units used for a given field, the number of bytes in a given field, and the order in which the fields appear within the data type. Fixed does not refer to the total number of bytes in the data type - see Rule “c”.

c. Data may be added to an existing data type only by adding the bytes to the end of the data format. As an example, the variable leader data contains information on ensemble number, time, heading, pitch, roll, temperature, pressure, etc. The format for the bytes 1-53 are now specified by changes added in support to the WorkHorse ADCP. If additional sensor data is to be added to the variable leader data then it must be added to the end of the data string (bytes 54-x as an example).

d. The order of data types in an ensemble is not fixed. That is there is no guarantee that velocity data will always be output before correlation data.

e. The header data will include the number of data types in the files and the offset to each ID number for each data type.

f. The total number of the bytes in an ensemble minus the 2-byte checksum will be included in the header.

## 7.2 Recommended Data Decoding Sequence for BroadBand Data Format PD0

a. Locate the header data by locating the header ID number (in the case of PD0 profile data that will be 7F7F).

b. Confirm that you have the correct header ID by:

   1. Locating the total number of bytes (located in the header data) in the ensemble. This will be your offset to the next ensemble.
   2. Calculate the checksum of total number of bytes in the ensemble excluding the checksum. The checksum is calculated by adding the value of each byte. The 2-byte least significant digits that you calculate will be the checksum.
   3. Read the 2-byte checksum word at the end of the ensemble, located by using the checksum offset in the header (determined in step "b-1") and compare this checksum word to the value calculated in step "b-2".
   4. If the checksums match then you have a valid ensemble. If the checksums do not match then you do not have a valid ensemble and you need to go back to step "a" and search for the next header ID number occurrence.

c. Locate the number of data types (located in the header data).

d. Locate the offset to each data type (located in the header data).

e. Locate the data ID type you wish to decode by using the offset to each data type and confirm the data ID number at that offset matches the ID type you are looking for.

f. Once the proper ID type has been located, use the ADCP Technical Manual for the ADCP you are using to understand what each byte represents in that particular data type.

## 7.3 Pseudo-Code for Decoding PD0 Ensemble Data

The following examples show the pseudo-code for decoding PD0 and PD5 ensemble data.

g. Define structures, which contain all fields in all data types of the PD0 format.

   1. typedef struct { <lists of types and fields> } FixedLeader.
   2. typedef struct { <lists of types and fields> } VariableLeader.
   3. typedef struct { <lists of types and fields> } BottomTrack.
   4. typedef struct { <lists of types and fields> } VelocityType

5. and so on for every available type.

h. Clear checksum.

i. Look for PD0 ID 0x7F. Add to checksum.

j. Is next byte a 0x7F? Add to checksum.

k. If no, return to step “b”.

l. Else, read next two bytes to determine offset to checksum. Add two bytes to checksum.

m. Read in X more bytes, where X = offset to checksum - 4. Adding all bytes to checksum.

n. Read in checksum word.

o. Do checksums equal?

p. If no, return to “b”.

q. For each available data type (the header contains the # of data types), go to the offset list in header.

1. Create a pointer to type short to the data type at an offset in the list.
2. Check the Type ID.
3. Create a pointer of appropriate type to that location.
4. Repeat for all available data types.

r. Work with data.

s. Return to “b” for next ensemble.

## 7.4 Pseudo-Code for Decoding PD5 Ensemble Data

a. Define structure that contains all fields in PD5 format.

1. typedef struct { <lists of types and fields> } PD5_Format.

b. Clear checksum.

c. Look for ID, PD5 id is 0x7D. Add to checksum.

d. Is next byte a 0x01? Add to checksum.

e. If no, return to “b”.

f. Else, read next two bytes to determine offset to checksum. Add two bytes to checksum.

g. Read in X more bytes, where X = offset to checksum - 4. Adding all bytes to checksum.

h. Read in checksum word.

i. Do checksums equal?

j. If no, return to “b”.

k. Create a pointer of type PD5_Format.

   1. PD5_Format *PD5_ptr;

l. Point pointer at location of ID byte.

   1. PD5_ptr = &buf[<location of input buffer>];

m. If “k” and “l” don't appeal to you, you can create a variable of type PD5_Format.

   1. PD5_Format PD5_data;

n. And copy the data from the input buffer to PD5_data.

o. Work with data.

p. Return to “b” for next ensemble.

## 7.5 Example Code for Decoding BroadBand Ensembles

Here is an example of how to decode a BroadBand ensemble. It is written in “C.”

**NOTE.** Structures must be “packed”; i.e. Don’t let the compiler add “fill bytes” to align fields on word boundaries.

This is an example of a section of code, not a full executable program.

```
/**************************************************************************/
/* Data ID Words */
/**************************************************************************/

   #define FLdrSelected   0x0000
   #define VLdrSelected   0x0080
   #define VelSelected    0x0100
   #define CorSelected    0x0200
   #define AmpSelected    0x0300
   #define PctSelected    0x0400
   #define SttSelected    0x0500
   #define BotSelected    0x0600
   #define Prm0           0x0700

   #define VelGood        0x0701
   #define VelSum         0x0702
   #define VelSumSqr      0x0703
   #define Bm5VelSelected 0x0A00
   #define Bm5CorSelected 0x0B00
   #define Bm5AmpSelected 0x0C00
   #define AmbientData    0x0C02
   #define Bm5PctSelected 0x0D00
   #define Bm5SttSelected 0x0E00
   #define Prm0_5         0x1300
   #define VelGood_5      0x1301
   #define VelSum_5       0x1302
   #define VelSumSqr_5    0x1303

/**************************************************************************/
/* structures */
/**************************************************************************/
```

```
typedef unsigned char  uchar;
typedef unsigned short ushort;
typedef unsigned long  ulong;

typedef struct {
   uchar        Minute,
                Second,
                Sec100;
}  TimeType;

typedef struct {
   uchar        Year,
                Month,
                Day,
                Hour,
                Minute,
                Second,
                Sec100;
}  DateTimeType;

typedef struct {
   uchar        Version,
                Revision;
}  VersionType;

typedef struct {
   uchar        ID,
                DataSource;
   ushort       ChecksumOffset;
   uchar        Spare,
                NDataTypes;
   ushort       Offset [256];
}  HeaderType;

typedef struct {
   ushort       ID;
   VersionType  CPUFirmware;
   ushort       Configuration;
   uchar        DummyDataFlag,
                Lag,
                NBeams,
                NBins;
   ushort       PingsPerEnsemble,
                BinLength,
                BlankAfterTransmit;
   uchar        ProfilingMode,
                PctCorrelationLow,
                NCodeRepetitions,
                PctGoodMin;
   ushort       ErrVelocityMax;
   TimeType     TimeBetweenPings;
   uchar        CoordSystemParms;
   short        HeadingAlignment,
                HeadingBias;
   uchar        SensorSource,
                AvailableSensors;
   ushort       DistanceToBin1Middle,
                TransmitLength;
}  FixLeaderType;

typedef struct {
   ushort       ID,
                EnsembleNumber;

   DateTimeType RecordingTime;
   uchar        Spare1;
   ushort       BITResult,
                SpeedOfSound,
                Depth,
                Heading;
   short        Pitch,
                Roll;
   ushort       Salinity;
   short        Temperature;
   TimeType     MaxTimeBetweenPings;
   uchar        HeadingStddev,
                PitchStddev,
```

```
                RollStddev;
   uchar        VMeas [8];
}  VarLeaderType;

typedef struct {
   ushort       ID,
                PingsPerEnsemble,
                EnsembleDelay;
   uchar        CorrelationMin,
                AmplitudeMin,
                PctGoodMin,
                BTMode;
   ushort       ErrVelocityMax,
                NSearchPings,
                NTrackPings;
   ushort       Range        [4];
   short        Velocity     [4];
   uchar        Correlation  [4],
                Amplitude    [4],
                PctGood      [4];
   ushort       WaterLayerMin,
                WaterLayerNear,
                WaterLayerFar;
   short        WVelocity    [4];
   uchar        WCorrelation [4],
                WAmplitude   [4],
                WPctGood     [4];
   ushort       MaxTrackingDepth;
   uchar        Amp [4];
   uchar        Gain;
   uchar        RangeMSB [4];
}  BottomTrackType;

typedef struct
{
  ushort        ID;
  short         Data [256];
} OneBeamShortType;

typedef struct
{
  ushort        ID;
  uchar         Data [256];
} OneBeamUcharType;

typedef struct {
   ushort       ID;
   short        Data [1024];
}  IntStructType;

typedef struct {
   ushort       ID;
   uchar        Data [1024];
}  ByteStructType;

typedef struct
{
  ushort        ID;
  uchar         Data [4];
}  AmbientType;

typedef struct
{
  ushort        ID;
  ushort        UaH;
  ushort        UaL;
  ushort        AmbBitsPerBin;
  ushort        AmbTrys;
  ushort        AmbNBins;
  short         AmbBinNum [ 5 ];
  short         Est [ 5 ];
  ushort        WAutoCor [ 5 ] [ 32 ];
  uchar         SysFreq;
  uchar         SampRate;
}  T01Type;

typedef struct
{
```

```
      ushort        ID;
      uchar         DAC [36];
   }  T02Type;

   typedef struct
   {
      ushort        ID;
      ushort        RSSIBinLen;
      ushort        RSSIBins;
      uchar         RSSI [512] [4];
      ushort        AutoCor [32] [4];
      short         Est [4];
      ushort        Amb [4];
      uchar         SysFreq;
      uchar         SampRate;
      uchar         MLen;
      ushort        XmtSamples;
      ushort        FirstBin[4];
      ushort        LastBin[4];
      ulong         BM6Depth[4];
      ushort        BM6Ta[4];
   }  T03Type;

/*************************************************************************/
/* Global Pointers */
/*************************************************************************/
   HeaderType       *HdrPtr;
   FixLeaderType    *FLdrPtr;
   VarLeaderType    *VLdrPtr;
   BottomTrackType  *BotPtr;
   BottomTrackType  *WBotPtr;
   IntStructType    *VelPtr;
   ByteStructType   *CorPtr;
   ByteStructType   *AmpPtr;
   ByteStructType   *PctPtr;
   ByteStructType   *SttPtr;
   AmbientType      *AmbientPtr;
   T01Type          *T01Ptr;
   T02Type          *T02Ptr;
   T03Type          *T03Ptr;
   OneBeamShortType *Bm5VelPtr;
   OneBeamUcharType *Bm5CorPtr;
   OneBeamUcharType *Bm5AmpPtr;
   OneBeamUcharType *Bm5PctPtr;
   OneBeamUcharType *Bm5SttPtr;

/*-------------------------------------------------------------------------*/

unsigned char RcvBuff[8192];


void DecodeBBensemble( void )
{
    unsigned short i, *IDptr, ID;

    FLdrPtr = (FixLeaderType *)&RcvBuff [ HdrPtr->Offset[0] ];

    if (FLdrPtr->NBins > 128)
        FLdrPtr->NBins = 32;

    for (i=1; i<HdrPtr->NDataTypes; i++)
    {
        IDptr = (unsigned short *)&RcvBuff [ HdrPtr->Offset [i] ];
        ID = IDptr[0];

        switch (ID)
        {
        case VLdrSelected:
            {
                VLdrPtr = (VarLeaderType *)&RcvBuff [ HdrPtr->Offset [i] ];
                break;
            }
        case VelSelected:
            {
                VelPtr = (IntStructType *)&RcvBuff [ HdrPtr->Offset [i] ];
                break;
            }
        case CorSelected :
```

```
                {
                    CorPtr = (ByteStructType *)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            case AmpSelected :
                {
                    AmpPtr = (ByteStructType *)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            case PctSelected :
                {
                    PctPtr = (ByteStructType *)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            case SttSelected :
                {
                    SttPtr = (ByteStructType *)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            case BotSelected :
                {
                    BotPtr = (BottomTrackType*)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            case AmbientData :
                {
                    AmbientPtr = (AmbientType *)&RcvBuff [ HdrPtr->Offset [i] ];
                    break;
                }
            }
        }
    }
```