adinstbpm - Task #17720

Milestone # 17719 (New): Booster operational test of one crate alongside existing BPM system

Booster BPM ACNET Interface

09/15/2017 11:09 AM - John Diamond

Status:	Work in progress	**Start date:**	09/15/2017
Priority:	Normal	**Due date:**	
Assignee:	John Diamond	**% Done:**	69%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	65.50 hours

Description

An ACNET interface to ADInstBPM that is compatible with the exists Booster BPM applications.
See this wiki page for a description of the existing ACNET devices.

Subtasks:

Task # 17721: ACNET device BPM Numbers (0x0017)	**Work in progress**
Task # 17722: ACNET device BPM Offsets (0x0016)	**Resolved**
Task # 17723: ACNET device BPM Digitizer Raw Data (0x0013)	**Resolved**
Task # 17724: ACNET device BPM Turn-by-Turn Scaled Positon	**Work in progress**
Task # 17725: ACNET device BPM Single Turn Scaled Position (0x001f)	**Resolved**
Task # 17726: ACNET device BPM Single Turn Scaled Position Buffer (0x0020)	**Resolved**
Task # 17727: ACNET device BPM Turn-by-Turn Extra Inputs	**Resolved**
Task # 17728: ACNET device BPM Digitizer Control Registers (0x0012)	**Rejected**
Task # 17729: ACNET device BPM Average Position (Snapshot) Orbit (0x0014)	**Work in progress**
Task # 17730: ACNET device BPM Orbit Group (0x001e)	**Work in progress**
Task # 17731: ACNET device BPM Closed Orbit Circular Buffer (0x0010)	**Resolved**
Task # 17732: ACNET device BPM Closed Orbit Circular Buffer (0x0011)	**Rejected**
Task # 17733: ACNET device BPM RF Test Device (0x0021)	**Rejected**
Task # 17734: ACNET device BPM Orbit Correction Device (0x0022)	**Work in progress**
Task # 17974: Create ACNET devices for B38 and B40	**Resolved**
Task # 17975: Test B38 and B40	**Work in progress**

History

#1 - 09/16/2017 10:42 AM - John Diamond

- Status changed from New to Work in progress

- Assignee set to John Diamond

#2 - 10/18/2017 09:24 AM - John Diamond

An e-mail from Bill regarding the data endianess and time-stamp encoding expectations of his console application:

```
Byte order from front end systems is always confusing because there
sometimes are automatic byte swaps on transfer of data.
The clx consoles are little endian.

For the orbit data and flash turn data the data structure is as
documented on Sharon's data diagrams.  The program does no byte or
word swaps for the header data----but the positions, Intensities,
extra inputs ieee floating points are word swapped but not byte
swapped.

For the turn by turn data Sharon seems to not have documented the
inclusion of the GPS timestamp  The data structure is as follows
Int4 micro date
Int4 micro timestamp
Int4 GPS sec (UTC time)
Int4 GPS nsec (nanosec from GPS sec)
Int4 number of turns
Int4 cycle type
```

Flt4 turn position data
Again the positions data ieee floats are word swapped but not byte
swapped while the header data is used as is.

So it would appear that the data is byte swapped on transfer with
the front end doing something special with the header.

The micro date and micro timestamp are BCD data read (I believe)
from a Systems Services Module clock and is in BCD
The following code decodes the BCD

```
/*************************************************************************/
/* microtimestampchar
*             returns a 18 char string (17 real char plus null) of the
*                           micros date and time
*             microdate = microdate from micro
*             timestamp = timestamp from micro
*/
/*************************************************************************/
static void microtimestampchar(int microdate, int timestamp,
  char *microchardatetime)
    {
    union
                {
                int integer;
                char byte[4];
                }temp;

    temp.integer = microdate;
    numeric_to_ascii_c(&temp.byte[2],2,&microchardatetime[0],CNV_HEX,1,'0');
                                                                        /* month */
    microchardatetime[2] = '/';
    numeric_to_ascii_c(&temp.byte[1],2,&microchardatetime[3],CNV_HEX,1,'0');
                                                                        /* day */
    microchardatetime[5] = '/';
    numeric_to_ascii_c(&temp.byte[3],2,&microchardatetime[6],CNV_HEX,1,'0');
                                                                        /* year */
    microchardatetime[8] = ' ';
    temp.integer = timestamp;
    numeric_to_ascii_c(&temp.byte[3],2,&microchardatetime[9],CNV_HEX,1,'0');
                                                                        /* hour */
    microchardatetime[11] = ':';
    numeric_to_ascii_c(&temp.byte[2],2,&microchardatetime[12],CNV_HEX,1,'0');
                                                                        /* min */
    microchardatetime[14] = ':';
    numeric_to_ascii_c(&temp.byte[1],2,&microchardatetime[15],CNV_HEX,1,'0');
                                                                        /* sec */
    microchardatetime[17] = 0;
    }
```