SANDIA
Communications Protocol Specification

1 Scope. The scope of this document is to provide a general definition and specification of the SANDIA communications protocol for use in instrumentation and controller applications.

2 Distribution Philosophy. SANDIA shall be licensed free of charge. A separate license shall be required for each manufacture's equipment model or instrument database format. The license application need not include a specification of the instrument database format, only a unique identification. This shall ensure a unique license number which shall be accessible using the protocol.

3 Introduction. The SANDIA communications protocol is non-application specific, powerful, efficient and simple to implement. Half duplex operation is employed making SANDIA useful in radio applications. Data errors in the order of 1 bit for every 10^{14} bits are detected using CRC16. The primitive form of the protocol, used with smaller instrument databases, is described as it requires the least data overhead and is most efficient. The protocol is particularly suited for battery powered instruments.

4 Future enhancements. Enhanced super sets of the protocol shall become available on demand. These enhancements shall include but not be limited to wider address fields and boolean operation commands.

5 Instrument Database Requirements. The instrument database shall contain four fields addressed at the relative zero address of the instrument database. The remaining database format is application specific and is not specified in this document in that it is independent of the protocol. The four required fields are as follows:

- 5.1 A BYTE field indicating the instrument communications buffer size where 0 = 256.
- 5.2 A BYTE field indicating vendors Identification code number.
- 5.3 A 16 bit WORD (MSB:LSB) field indicating the data base identification code number.
- 5.4 A null terminated printable ASCII character string field consisting of 8 characters and null terminator.
- 5.5 Three PBCD bytes indicating firmware date in the format MM, DD, YY.

6 General Description.

Sandia protocol consist of binary packets transferred as RS232 bytes consisting of one start bit, eight data bits, one stop bit and no parity. LVC instruments cater to bit rates in the range of 300 bps to 19200 bps. Other instruments may restrict bit rates. Packets are exchanged in a half duplex fashion where only the host originates. Packets contain an instrument address allowing a maximum of 62 instruments to share a data drop.

6.1 Sandia Command packet general format:

0: 0xFF, Preamble for instrument wake up.
1: 0xFF
2: "S", ASCII uppercase S, start of packet.
3: Lng, number of bytes following.
4: Funct/UID,
    B7 = 0, Normal mode.
    B7 = 1, Enhanced mode.
    B6 = 0, Read request.
    B6 = 1, Write request.
    B5-B0, Unit address,
        0 = broadcast
        0x3F = anyone
        0x01 - 0x3E = specific unit ID number.
5: Optional Data Bytes, 0 or more data bytes
n: MSB:LSB of CRC16

6.2 Sandia Read Command packet format:

0: 0xFF, Preamble for instrument wake up.
1: 0xFF
2: "S", ASCII upper case S, start of packet.
3: Lng, number of bytes following.
4: Funct/UID,
    B7 = 0, Normal mode.
    B6 = 0, Read request.
    B5-B0, Unit address,
        0x3F = anyone
        0x01 - 0x3E = specific unit ID number.
5-6: 16 bit database relative address.
7: Cnt, Number of bytes to Read.
8-9: MSB:LSB of CRC16

6.3 Sandia Read Response packet format:

0: "s", ASCII lower case s, start of packet.
1: Lng
2: Funct/UID
3: Error Code, No data returned on errors.
    0 = Ok.
    1 = Address out of bounds.
    2 = Lng error.
4: Data, Read data bytes.
n-n+1: MSB:LSB of CRC16

6.4 Sandia Write Command packet format:

0: 0xFF, Preamble for instrument wake up.
1: 0xFF
2: "S", ASCII upper case S, start of packet.
3: Lng, 5 + Number of data bytes to write.
4: Funct/UID,
    B7 = 0, Normal mode.
    B6 = 1, Write request.
    B5-B0, Unit address
        0 = broadcast
        0x3F = anyone
        0x01 - 0x3E = specific unit ID number.
5-6: 16 bit database relative address.
7-n: Data, Data bytes to write.
m-m+1: MSB:LSB of CRC16

6.5 Sandia Write Response packet format:

0: "s", ASCII lower case s, start of packet.
1: Lng, 4, Number of packet bytes following.
2: Funct/UID,
    B7 = 0, Normal mode.
    B6 = 1, Write request.
    B5-B0, Unit address,
        0x3F = anyone
        0x01 - 0x3E = specific unit ID number.
3: Error Code,
    0 = Ok.
    1 = Command Address out of bounds.
    2 = Command Lng error.
4-5: MSB:LSB of CRC16

7 CRC16 Generation. CRC-16 is based on the polynomial x^16+x^15+x^2+1. The data is assumed to be fed in from least to most significant bit. A seed value of 0x0001 is used. Example code in C and Intel assembler is provided.

```
;=================================================================
;
; Function:     WORD    calc_crc(unsigned char * p1, int bytes)
;
; Desc:         Calculate and return CRC, starting with the first character.
;               Pointer should point to first character following the start
;               character.  Count should be the actual number of characters
;               included in the calculation.
;
;               Examples in C and intel assembly.
;
```

```
;       WORD    calc_crc(char *fpacket, int flen)
;       {
;               unsigned        s;
;               int             c,w;
;
;               s = 1;
;
;               for (w = 0 ; w < flen ; w++) {
;                       c = *fpacket++;
;                       s = CRCtab[(s^c) &0xFF] ^ s >> 8;
;               }
;
;       }
;
;
; Parms:        p1 *    = [bp+6]   Pointer to buffer.
;               bytes   = [bp+10]  Number of bytes.
;
; Returns:      CRC.
;
;-----------------------------------------------------------------------------
;
crctbl:
        dw      00000h, 0c0c1h, 0c181h, 00140h
        dw      0c301h, 003c0h, 00280h, 0c241h
        dw      0c601h, 006c0h, 00780h, 0c741h
        dw      00500h, 0c5c1h, 0c481h, 00440h
        dw      0cc01h, 00cc0h, 00d80h, 0cd41h
        dw      00f00h, 0cfc1h, 0ce81h, 00e40h
        dw      00a00h, 0cac1h, 0cb81h, 00b40h
        dw      0c901h, 009c0h, 00880h, 0c841h
        dw      0d801h, 018c0h, 01980h, 0d941h
        dw      01b00h, 0dbc1h, 0da81h, 01a40h
        dw      01e00h, 0dec1h, 0df81h, 01f40h
        dw      0dd01h, 01dc0h, 01c80h, 0dc41h
        dw      01400h, 0d4c1h, 0d581h, 01540h
        dw      0d701h, 017c0h, 01680h, 0d641h
        dw      0d201h, 012c0h, 01380h, 0d341h
        dw      01100h, 0d1c1h, 0d081h, 01040h
        dw      0f001h, 030c0h, 03180h, 0f141h
        dw      03300h, 0f3c1h, 0f281h, 03240h
        dw      03600h, 0f6c1h, 0f781h, 03740h
        dw      0f501h, 035c0h, 03480h, 0f441h
        dw      03c00h, 0fcc1h, 0fd81h, 03d40h
        dw      0ff01h, 03fc0h, 03e80h, 0fe41h
        dw      0fa01h, 03ac0h, 03b80h, 0fb41h
        dw      03900h, 0f9c1h, 0f881h, 03840h
        dw      02800h, 0e8c1h, 0e981h, 02940h
        dw      0eb01h, 02bc0h, 02a80h, 0ea41h
        dw      0ee01h, 02ec0h, 02f80h, 0ef41h
        dw      02d00h, 0edc1h, 0ec81h, 02c40h
        dw      0e401h, 024c0h, 02580h, 0e541h
        dw      02700h, 0e7c1h, 0e681h, 02640h
        dw      02200h, 0e2c1h, 0e381h, 02340h
        dw      0e101h, 021c0h, 02080h, 0e041h
        dw      0a001h, 060c0h, 06180h, 0a141h
        dw      06300h, 0a3c1h, 0a281h, 06240h
        dw      06600h, 0a6c1h, 0a781h, 06740h
        dw      0a501h, 065c0h, 06480h, 0a441h
        dw      06c00h, 0acc1h, 0ad81h, 06d40h
        dw      0af01h, 06fc0h, 06e80h, 0ae41h
```

```
        dw      0aa01h, 06ac0h, 06b80h, 0ab41h
        dw      06900h, 0a9c1h, 0a881h, 06840h
        dw      07800h, 0b8c1h, 0b981h, 07940h
        dw      0bb01h, 07bc0h, 07a80h, 0ba41h
        dw      0be01h, 07ec0h, 07f80h, 0bf41h
        dw      07d00h, 0bdc1h, 0bc81h, 07c40h
        dw      0b401h, 074c0h, 07580h, 0b541h
        dw      07700h, 0b7c1h, 0b681h, 07640h
        dw      07200h, 0b2c1h, 0b381h, 07340h
        dw      0b101h, 071c0h, 07080h, 0b041h
        dw      05000h, 090c1h, 09181h, 05140h
        dw      09301h, 053c0h, 05280h, 09241h
        dw      09601h, 056c0h, 05780h, 09741h
        dw      05500h, 095c1h, 09481h, 05440h
        dw      09c01h, 05cc0h, 05d80h, 09d41h
        dw      05f00h, 09fc1h, 09e81h, 05e40h
        dw      05a00h, 09ac1h, 09b81h, 05b40h
        dw      09901h, 059c0h, 05880h, 09841h
        dw      08801h, 048c0h, 04980h, 08941h
        dw      04b00h, 08bc1h, 08a81h, 04a40h
        dw      04e00h, 08ec1h, 08f81h, 04f40h
        dw      08d01h, 04dc0h, 04c80h, 08c41h
        dw      04400h, 084c1h, 08581h, 04540h
        dw      08701h, 047c0h, 04680h, 08641h
        dw      08201h, 042c0h, 04380h, 08341h
        dw      04100h, 081c1h, 08081h, 04040h

_calc_crc       proc    far
        push    bp
        mov     bp,sp
        push    ds
        push    si

        lds     si, dword ptr [bp+6]    ; Pointer to buffer
        mov     cx, word ptr [bp+10]    ; Number of bytes counter
        mov     dx, 1                   ; DX = intermediate value

ccrc_02:
        mov     bl, byte ptr [si]       ; BL = character
        xor     bl, dl
        xor     bh, bh
        shl     bx, 1
        mov     ax, word ptr cs:crctbl [bx]
        xor     al, dh
        mov     dx, ax
        inc     si                      ; Pointer++
        loop    ccrc_02

        pop     si
        pop     ds
        pop     bp
        ret

_calc_crc       endp

        public  _calc_crc
```

8 Minimum Commands are read and write.

9 LVC specific packet structures.

```
typedef struct {        // Command packet format.
        BYTE    0xFF.
        BYTE    0xFF.
        BYTE    start     // ‘S’  Start of packet.
        BYTE    lng;        // Length of following data: cmd+data+crc.
        BYTE    cmd;        // R/W, UID.
        BYTE    data[256];  // Data+CRC16 of following packet: start...data inclusive.
} CPKT;

typedef struct {        // Response packet format.
        BYTE    pkstart;    // 's' Start of packet.
        BYTE    lng;        // Length of following data: cmd+errc+data+crc.
        BYTE    cmd;        // R/W, UID.
        BYTE    errc;       // Response error code.
        BYTE    data[256];  // Data+CRC16 of following packet: pkstart...data inclusive.
} RSPPKT;               // Read/Write response packet.

typedef enum {
        PKREAD = 0x00,    // W/R bit of command.
        PKWRITE = 0x40
} PKRW;

typedef struct {           // Read command data field.
        WORD    addr;       // Database Address of read data.
        BYTE    cnt;        // Number of bytes to read.
} SAREAD;

typedef struct {           // Write command data field.
        WORD    addr;       // Database Address of read data.
        BYTE    wdat[256];
} SAWRITE;
```

LVC data base format. A file named SHARED.H is included.