CONTROL DATA TRANSMISSION VIA RS-485 AND RS-232 BETWEEN THE MASTER DEVICE AND THE SLAVE DEVICES

1. GENERAL

The universal rules of control data transmission for the Devices equipped with RS485 and RS-232 are defined by this protocol.

One or several slave Devices can be connected to the interface bus at any moment. Only the Master may initiate transmission via the bus. The Master may be either a remote PC or a control keyboard.

Control data are transmitted by byte blocks (1 byte is composed of 8 data bits) at the rate of 115200 bps without parity control in the half duplex mode.

Once a Slave receives a control block addressed to it, this Slave must transmit a response block no later than 500ms. Master must not begin a transmission of a new control block without the reception of a response block to the previous control block within the aforementioned time-out.

2. BLOCK STRUCTURE.

Each of the blocks contains the sync sequence, address subblock, ID subblock, length subblock, payload subblock (if needed) and the check-sum subblock (CRC).

2.1. The sync subblock consists of the bytes 0E2h and 0E4h; they are transmitted in the beginning of any block.

2.2. Address subblock consists of 4 bytes. The beginning 2 bytes carry the type (the 1st byte) and the address (the 2nd byte) of the Device which the block is being addressed to; the next 2 bytes carry the type (the 3rd byte) and the address (the 4th byte) of the Device which is preparing this certain block for the transmission[1].

The address subblock follows the sync subblock.

2.3. The ID subblock describes the the type of the payload being transmitted in this block and it consists of 2 bytes.

2.3.1. When The Master transmits a block to a Slave, the 1st byte of the ID subblock carries an instruction and the payload subblock data are the argument. The highest but if the 2nd byte of the ID subblock must be set to 0. The remaining bits of this byte are reserved to future extensions of the Protocol and must be set to 0 in this revision of the Protocol.

List of the instructions:

2.3.1.1. 000h — initialize the address pointer[2].

4 bytes if the payload subblock must be interpreted as the address. The response block must be transmitted without the payload subblock;

2.3.1.2. 001h- read the current address pointer[2].

The instruction is transmitted with argument equal with 4 (in the payload subblock of the control block) which means there must 4 bytes in the payload subblock of the response block (4 bytes of the current address pointer).

2.3.1.3. 002h — write data into a page buffer of the corresponding EEPROM beginning with the current address[3].

The instruction is used for filling a page buffer of an EEPROM before those data shall be stored in the given EEPROM. It is necessary to point out that the address pointer must be set with

the address of the target page before the first use of this instruction. The data in the payload subblock are the arguments of this instruction. The response block must be transmitted without the payload subblock. It is necessary to point out that the total number of data being written into the page buffer must not exceed the page size of the specific EEPROM after every initialization of the address pointer;

2.3.1.4. 003h – store the contents of the EEPROM page buffer into the corresponding EEPROM memory[3].

The data which have been written into the page buffer of the EEPROM are stored in the EEPROM beginning with the address in the address pointer. If the address pointer were properly initialized before the page buffer write operation, it will not have to be re-initialized again. The instruction is transmitted without arguments. The response block is transmitted without the payload subblock.

2.3.1.5. 004h — write data into the service memory of the Device beginning with the current address. The parameter settings may be stored in the service memory. The specific parameters and their values depend on the specific Device and are described in the corresponding Annex to the current Protocol. The data in the payload subblock are the arguments for this instruction. The response block is transmitted without the payload subblock.

2.3.1.6. 005h — read the corresponding EEPROM contents beginning with the current address.

The number of the bytes to be read is transmitted as the argument for this instruction. The payload subblock of the response block carries the requested number of the bytes retrieved from the EEPROM. The total amount of the data read from the EEPROM must not exceed the remaining amount of the bytes of the EEPROM page after the every address pointer initialization; i.e. the transition to the next EEPROM page must be performed by means of the address pointer initialization. The Atmel EEPROMs are organized page by page and it is described in the Mandatory Annex 1 of the current Protocol. S25 (S25FK) EEPROM pages are described in 2.3.1.8;

2.3.1.7. 006h — read the service memory of the Device beginning with the current address.

The number of the bytes to be read is transmitted as the argument for this instruction. The payload subblock of the response block carries the requested number of the bytes retrieved from the service memory. Reading 32 bytes from the service memory beginning with address FFFFFFFFh retrieves the Device signature (device name, device revision, etc);

2.3.1.8. 007h — read Device EEPROM type.

The number of the read bytes is transmitted in the argument, i.e. either "1" or "2". payload subblock correspondingly carries either 1 or 2 bytes which identify EEPROM type and EEPROM volume occupying the address space which is accessed with the current address pointer. EEPROM type is Spansion S25FK if the 1st byte in the response is 00000001b. The 2nd byte shall describe the EEPROM volume and its organization in this case::

- 013h - 2048 pages / 256 bytes, erasable sector is 4096 bytes;
- 014h - 4096 pages / 256 bytes, erasable sector is 4096 bytes;
- 015h - 8192 pages / 256 bytes, erasable sector is 4096 bytes;
- 016h - 16384 pages / 256 bytes, erasable sector is 4096 bytes;
- 017h - 32768 pages / 256 bytes, erasable sector is 4096 bytes;

Atmel EEPROM returns the 1st byte equal with XX011XXXb, XX1001XXb, XX1111XXb, where X – arbitrary bit value; these codes correspond to AT45DB041, AT45DB081, AT45DB642D Atmel. The page structure of this type of EEPROM is described in the Mandatory Annex 1 of the current Protocol. The 2nd bytes carries arbitrary value;

2.3.1.9. 008h – reserved instruction, Slave must send a response block with asserted flag of the error in the format of the received block (see 2.3.2.).

2.3.1.10. 009h – read the instruction execution status. The instruction in transmitted without arguments. The response block is transmitted without the payload subblock;

2.3.1.11. 00Ah — apply the changes; once the new data have been written to the EEPROM, this instruction will begin to apply the changes записанных в EEPROM (for example, copy data into the buffer RAM). This instruction does not have arguments. The EEPROM is selected according to the value of the current high nibble of the highest byte of the address pointer[2]. The response block is transmitted without the payload subblock. Having transmitted the response pack, the Slave may be inaccessible to the data exchange with the PC during the next 5 seconds or less;;

2.3.1.12. 00Bh – local time update. 3 bytes are transmitted as the arguments: clocks, minutes and seconds in the binary code. The device status is sent back in the response block (see 3.);

2.3.1.13. 00Ch – reserved instruction, Slave must send a response block with asserted flag of the error in the format of the received block (see 2.3.2.).

2.3.1.14. 00Dh — device reset. The instruction in transmitted without arguments. The response block is transmitted without the payload subblock. Having sent the response block, the device reboots itself. It may be inaccessible for up to 16 seconds;

2.3.1.15. 00Eh – reserved instruction, Slave must send a response block with asserted flag of the error in the format of the received block (see 2.3.2.).

2.3.1.16. 00Fh – reserved instruction, Slave must send a response block with asserted flag of the error in the format of the received block (see 2.3.2.).

2.3.1.17. 010h — sector erase instruction[4]. A EEPROM sector accessed by the current value of the address pointer is erased. The EEPROM is selected with the high nibble of the address pointer highest byte. The instruction in transmitted without arguments. The response block is transmitted without the payload subblock;

2.3.1.18. 011h — page write instruction[4]. The data to be written are sent as the arguments. The response block is transmitted without the payload subblock;

2.3.1.19. 020h — the code of a pressed button instruction. The code of the pressed or released button is transmitted in the only argument byte. It is possible to send the instruction with the argument byte equal with 0. in this case, the Device shall consider there is no any button pressed. The button codes are listed in the recommended Annex 4 of the current Protocol. The device status is sent back in the response block (see 3.);

2.3.1.20. 021h — T-bar position code instruction. The 1st and the 2nd arguments carry the code of the position of the T-bar (lower byte ahead). The device status is sent back in the response block (see 3.);

2.3.1.21. 022h — indication request instruction. The instruction in transmitted without arguments. .The device status (6 bytes) is sent back in the response block (see 3.) followed by the indication array. The structure of the indication array is described in the Mandatory Annex 2 of the current Protocol. The LEDs codes are listed in the recommended Annex 5 of the current Protocol;

2.3.1.22. 023h – device status request. The instruction in transmitted without arguments. The device status is sent back in the response block (see 3.);

2.3.2. When a Slave sends response pack to the Master, the contents of the 1st byte of the subblock must repeat the instruction byte of the control block which began the transaction. The highest bit of the 2nd byte must be set as '1'. The lowest 3 bits of this byte must reflect the following status of the execution process of the newly received instruction (i.e. the instruction which began the transaction):

00h — instruction has been executed. There are true data bytes in the response block, if any;

01h – instruction has been accepted to the execution. There are no true data bytes in the response block, if any. Ignore them;

02h — Device is busy executing the previous instruction. The newly accepted instruction shall be ignored by the Slave in this state. There are no true data bytes in the response block, if any. Ignore them;

03h — The Slave device is broken. There are no true data bytes in the response block, if any. Ignore them;

04h – error in the format of the newly accepted instruction. There are no true data bytes in the response block, if any. Ignore them;

Values 05h … 07h of the lowest 3 bits are reserved and are not used by the current Protocol.

The rest bits of this byte are reserved and must be set as '0'.

The ID subblock follows the address subblock.

2.4. The length subblock consists of 1 byte and it describes the length of the payload subblock. Value 0 reports there is no payload subblock in the current block.

The length subblock follows the ID subblock.

2.5. The payload subblock is the only subblock which is not necessarily present in a control block or a response block. The length of the payload subblock is 240 (decimal) bytes. Multibyte words are sent the lower byte ahead.

The payload subblock follows the length subblocks.

2.6. CRC are the last 2 bytes of a control block or a response block. They are transmitted the lower byte ahead. All the block's bytes are convolved (except the CRC bytes) for the calculation of the CRC. The block is supposed to be received error free when the convolution of the received block's bytes together with the received CRC bytes results in the value 0000h.
The initial convolution value is set to be equal FFFFh.

The polynomial for the CRC Calculation and the examples of the convolution have been described in “Application Note 27. Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor iButtonTM Products (www.dalsemi.com)”, in the chapter “CRC-16 computation for ram records in iButtons” (see Mandatory Annex 3 for the current Protocol).

2.7. The location of all the bytes in the block is given in the Table 1:

TypeR – the type of the addressed (target) Device;

AddrR – the address of the target device;

TypeT – the type of the Device sending the current block;

AddrT – the address of the Device sending the current block;

Ident – the 1st byte of the ID subblock;

IdentExt – the 2nd byte of the ID subblock;

Len – the length subblock (length of the following payload subblock measured with bytes);

Data 0 .. Data M – payload subblock, it has (M + 1) bytes in this example;

CRCL, CRCH – the lowest and the highest bytes of the CRC subblock.

Table 1. The sequence of the subblocks in a block.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	..	Byte N-2	Byte N-1	Byte N
0E2h	0E4h	TypeR	AddrR	TypeT	AddrT	Ident	IdentExt	Len	Data 0	..	Data M	CRCL	CRCH

3. DEVICE STATUS

The Device Status is transmitted in response to instructions of the Master and it has the following format:

- Byte 1 — status of the process which is now active inside the device:
  - bit 0 is set to "1" — the device is executing a process, the instruction of the pressed button code cannot be executed at the moment;
  - bit 1 is set to"1" - the device is executing a process, the device may be expecting the instructions of the pressed button code at the moment;
  - bit 2 is set to "1"- the device is executing a process, the instruction of the T-bar position code cannot be executed at the moment;
  - bit 3 is set to "1"- the device is executing a process, the device may be expecting the instructions the instruction of the T-bar position code at the moment;
  - bit 4 is set to "1" - the device is executing a process, the instruction of the indication request cannot be executed at the moment;
  - bit 5 is set to "1" - the device is executing a process, the device may be expecting the instruction of the indication request at the moment;
  - bits 6 ... 7 - reserve (are set to "0");
- Byte 2 – indication status of the Device:
  - bit 0 is set to "1" — the indication of the device must change(from the moment of the response to the latest instruction of the indication request);
  - bits 1 and 2 — the next states of the bits determining the brightness level of the control keyboard(they are executed optionally by the keyboard),:
    - 00 — nominal brightness level;
    - 01 — brightness level down one step;
    - 10 - brightness level down two steps;
    - 11 - brightness level down three steps;
  - bits 3 ... 7 - reserve (must be set to "0");
- Byte 3 - the lest received code of a pressed button;
- Byte 4 and 5 — the current T-bar position code(lower byte ahead);
- Byte 6 - reserve (must be "0").

Notes.

1. The Device type for the Switcher is 0Fh and the only address number is 00h. The Device type for a remote PC is FFh and the only address number is 01h.

2. Device may have several EEPROM chips of different types which share a common address space (according to the current Protocol). The EEPROM selection and the selection of the service memory is made by the higher nibble of the highest byte of the address. The correspondence between the nibble and the memory is the following:

0b0000 – EEPROM0;
0b0001 – EEPROM1;
0b0010 – EEPROM2
0b0011 – EEPROM3;
0b0100 – EEPROM4;

0b0101 – EEPROM5;
0b0110 – EEPROM6;
0b0111 – EEPROM7;
0b1000 – EEPROM8;
0b1001 – EEPROM9;
0b1010 – EEPROM10
0b1011 – EEPROM11;
0b1100 – EEPROM12;
0b1101 – EEPROM13;
0b1110 – EEPROM14;
0b1111 – service memory. It does not have a page structure.

3. The instruction may be used only in conjunction with the Atmel EEPROM which have been listed in the Mandatory Annex 1 of the current Protocol.

4. The instruction may be used only in conjunction with the S25 EEPROM which has the internal structure according to the 2.3.1.8.

MANDATORY ANNEX 1

ATMEL EEPROM PAGE STRUCTURE

1. Flash EEPROM AT45DB041. Structure – 2048 pages / 264 bytes. The byte addresses at the beginning of the pages:
Page 0: 00000000b 00000000b 00000000b 00000000b Page 1: 00000000b 00000000b 00000010b 00000000b Page 2: 00000000b 00000000b 00000100b 00000000b Page 3: 00000000b 00000000b 00000110b 00000000b
…
Page 2047:00000000b 00001111b 11111110b 00000000b

2. Flash EEPROM AT45DB081. Structure – 4096 pages / 264 bytes. The byte addresses at the beginning of the pages:
Page 0: 00000000b 00000000b 00000000b 00000000b Page 1: 00000000b 00000000b 00000010b 00000000b Page 2: 00000000b 00000000b 00000100b 00000000b Page 3: 00000000b 00000000b 00000110b 00000000b
…
Page 4095:00000000b 00011111b 11111110b 00000000b

3. Flash EEPROM AT45DB642D. Structure – 8192 pages / 1056 bytes. The byte addresses at the beginning of the pages:
Page 0: 00000000b 00000000b 00000000b 00000000b Page 1: 00000000b 00000000b 00001000b 00000000b Page 2: 00000000b 00000000b 00010000b 00000000b Page 3: 00000000b 00000000b 00011000b 00000000b
…
Page 8191:00000000b 11111111b 11111000b 00000000b

MANDATORY ANNEX 2

INDICATION ARRAY STRUCTURE

The indication array is 64 bytes long. Each byte carries information on the state of 4 indication LEDs, 2 adjacent bits per 1 LED. The state is described as follows:

00- LED is off;
01- LED is on with low brightness;
10- LED is on with full brightness.
11- LED is blinking with full brightness.

One byte code corresponds to every LED in the device. The higher 6 bits of that code determine the byte in the indication array, whilst the remaining 2 bits determine the positions of the pair bits describing the state in that byte.

For example:

1. set the bits 7 and 6 as '10' in the byte 10;
2. set the bits 1 and 0 as '11' in the byte 15;
3. set the bits 3 and 2 as '01' in the byte 15;
4. set the remaining bits in all the bytes of the array as '0';
5. It will in turn the LED with code 0x2B at the full brightness and turn the LED with code 0x3C into the blinking mode, turn the LED with code 0x3D on at the low brightness and turn the remaining LEDs off .

MANDATORY ANNEX 3

CRC CALCULATION

CRCL and **CRCH** are the bytes for the 16-bit control word (herein the **Annex 3** it may be called check sum). All the bytes except the last 2 bytes from a block are used to calculate the check sum. There must be 0x0000 after the check sum calculation including the last 2 bytes if the data have been transmitted error free.

The initial value for check sum calculation is 0xFFFF. **CRCL** and **CRCH** can be calculated , particularly, by means of the tables A3.1 and A3.2 according to the rule below:
CRCH' = Tbl.A3.2 (I),
CRCL' = Tbl.A3.1 (I) + **CRCH**, where I = **CRCL** + InputByte
CRCH and **CRCL** correspond to higher and lower byte of the current check sum value,

InputByte is the next data byte. **CRCH'** and **CRCL'** correspond to higher and lower byte of the new check sum value. Tbl.A3.2 (I) and Tbl.A3.1 (I) correspond to the tables' values, I is the shift factor in the tables. «+» means bit-wise XOR.

Table A3.1. Lower CRC byte calculation coefficients.

000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
001h	0c0h	080h	041h	000h	0c1h	081h	040h
000h	0c1h	081h	040h	001h	0c0h	080h	041h
000h	0c1h	081h	040h	001h	0c0h	080h	041h

001h 0c0h 080h 041h 000h 0c1h 081h 040h

Table A3.2. Higher CRC byte calculation coefficients.

000h	0c0h	0c1h	001h	0c3h	003h	002h	0c2h
0c6h	006h	007h	0c7h	005h	0c5h	0c4h	004h
0cch	00ch	00dh	0cdh	00fh	0cfh	0ceh	00eh
00ah	0cah	0cbh	00bh	0c9h	009h	008h	0c8h
0d8h	018h	019h	0d9h	01bh	0dbh	0dah	01ah
01eh	0deh	0dfh	01fh	0ddh	01dh	01ch	0dch
014h	0d4h	0d5h	015h	0d7h	017h	016h	0d6h
0d2h	012h	013h	0d3h	011h	0d1h	0d0h	010h
0f0h	030h	031h	0f1h	033h	0f3h	0f2h	032h
036h	0f6h	0f7h	037h	0f5h	035h	034h	0f4h
03ch	0fch	0fdh	03dh	0ffh	03fh	03eh	0feh
0fah	03ah	03bh	0fbh	039h	0f9h	0f8h	038h
028h	0e8h	0e9h	029h	0ebh	02bh	02ah	0eah
0eeh	02eh	02fh	0efh	02dh	0edh	0ech	02ch
0e4h	024h	025h	0e5h	027h	0e7h	0e6h	026h
022h	0e2h	0e3h	023h	0e1h	021h	020h	0e0h
0a0h	060h	061h	0a1h	063h	0a3h	0a2h	062h
066h	0a6h	0a7h	067h	0a5h	065h	064h	0a4h
06ch	0ach	0adh	06dh	0afh	06fh	06eh	0aeh
0aah	06ah	06bh	0abh	069h	0a9h	0a8h	068h
078h	0b8h	0b9h	079h	0bbh	07bh	07ah	0bah
0beh	07eh	07fh	0bfh	07dh	0bdh	0bch	07ch
0b4h	074h	075h	0b5h	077h	0b7h	0b6h	076h
072h	0b2h	0b3h	073h	0b1h	071h	070h	0b0h
050h	090h	091h	051h	093h	053h	052h	092h
096h	056h	057h	097h	055h	095h	094h	054h
09ch	05ch	05dh	09dh	05fh	09fh	09eh	05eh
05ah	09ah	09bh	05bh	099h	059h	058h	098h
088h	048h	049h	089h	04bh	08bh	08ah	04ah
04eh	08eh	08fh	04fh	08dh	04dh	04ch	08ch
044h	084h	085h	045h	087h	047h	046h	086h
082h	042h	043h	083h	041h	081h	080h	040h

ANNEX 4

BUTTON CODES OF THE SE2850 DEVICE

KeyRemote	0x01
KeyUp	0x02
KeyUpRelease	KeyUp+0xa0
KeyEnter	0x03
KeyEnterRelease	KeyEnter+0xa0
KeyWipes1	0x04
KeyWipes2	0x05
KeyWipes3	0x06
KeyMix	0x30
KeyAudioAssociated	0x07
KeyPiP1Prog	0x08
KeyPiP2Prog	0x09
KeyTitlesProg	0x0a
KeyLeft	0x0b
KeyLeftRelease	KeyLeft+0xa0
KeyDown	0x0c
KeyDownRelease	KeyDown+0xa0
KeyRight	0x0d
KeyRightRelease	KeyRight+0xa0
KeyWipes4	0x0e
KeyWipes5	0x0f
KeyWipes6	0x10
KeyPiP1Prev	0x11
KeyPiP1PrevRelease	KeyPiP1Prev+0xa0
KeyPiP2Prev	0x12
KeyPiP2PrevRelease	KeyPiP2Prev+0xa0
KeyTitlesPrev	0x13
KeyTitlesPrevRelease	KeyTitlesPrev+0xa0
KeyInp1Prog	0x14
KeyInp2Prog	0x15
KeyInp3Prog	0x16
KeyInp4Prog	0x17
KeyInp5Prog	0x18
KeyInp6Prog	0x19
KeyInp7Prog	0x1a
KeyInp8Prog	0x1b
KeyBlackProg	0x1c
KeyBarsProg	0x1d
KeyLogo1	0x1e
KeyLogo2	0x1f
KeyClock	0x20
KeyInp1Prev	0x21
KeyInp2Prev	0x22
KeyInp3Prev	0x23
KeyInp4Prev	0x24
KeyInp5Prev	0x25
KeyInp6Prev	0x26

KeyInp7Prev	0x27
KeyInp8Prev	0x28
KeyBlackPrev	0x29
KeyBarsPrev	0x2a
KeyCut	0x2b
KeyCutRelease	KeyCut+0xa0
KeyAuto	0x2c
KeyAutoRelease	KeyAuto+0xa0
KeyWipesInv	0x2f
KeyVideoXPt	0x31
KeyVideoXPtRelease	KeyVideoXPt+0xa0
KeyAudioXPt	0x32
KeyAudioXPtRelease	KeyAudioXPt+0xa0
KeyFreeze	0x33
KeyFreezeRelease	KeyFreeze+0xa0
KeyTimer	0x34
KeyTitles2Prev	0x35
KeyTitles2PrevRelease	KeyTitles2Prev+0xa0
KeyTitles2Prog	0x36
KeyAUX	0x38
KeyAUXRelease	KeyAUX+0xa0
KeyFTB	0x39
KeyFTBRelease	KeyFTB+0xa0
KeySpeed1	0x3a
KeySpeed2	0x3b
KeySpeed3	0x3c
KeyInp9Prev	0x3d
KeyInp10Prev	0x3e
KeyInp11Prev	0x3f
KeyInp12Prev	0x40
KeyFS	0x41
KeyFSRelease	KeyFS+0xa0
KeyInp9Prog	0x42
KeyInp10Prog	0x43
KeyInp11Prog	0x44
KeyInp12Prog	0x45

Note. The codes which are named as "KeyXXXRelease" are the buttons release codes, the rest are the buttons press codes.

ANNEX 5

THE LED CODES OF THE SE2850 DEVICE

1. LEDs for the button indication have the following codes: the press code of the corresponding button -1;

2. The T-bar end indication LEDs have the following codes: upper LED - 0x2c
   lower LED - 0x2d

ETHERNET PROTOCOL

The transport layer is different - instead of RS-232/RS-485, Ethernet is used.
Physical layer is Ethernet 100BASE-T. Transport layer is TCP/IP.

PC application should open TCP connection to port 9000 and IP address of SE2850. PC application when acts as "Master" and SE28500 as "Slave" for the logical layer described in the document above.

Here is the suggestion from protocol writer.
For remote control following instructions are used:

2.3.1.19 - simulate key press
2.3.1.20 - simulate T-bar movement
2.3.1.21 - indication request
2.3.1.22 - device status request

Other instructions can be ignored.

When Master is idle, it should repeatedly send "device status request" instruction to Slave to ensure that the link is working. Such instructions should be repeated at least every 500 msec. When control link is established, SE2850 OSD would display steady "PC Control" message.

SERemote GUI application is using this protocol. Suggest that developers try SERemote and capture/analyze Ethernet traffic with Wireshark to better grasp the inner workings.