Demystifying the Leybold RS485 telegrams:

Setup:

1. Set the communication as follows:
   1. RS485
   2. 19.2kBaud (fixed)
   3. 8 bits/byte
   4. 1= stop bit
   5. Even parity bit
   6. Address=0
   7. Flow Control=none

Telegram Send Structure:

1. 24 Byte Structure:, each byte transfer as hex (simplest method) or can be done in decimal format (much more complicated).
2. Telegram Command structure:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte Number | Abbr/bits | Description | Value | Binary | Hex | Hex String |
| 0 | STX/8 | Start Byte | Always 2 | 10 | 2 | $02 |
| 1 | LG#/8 | Length of the Payload | Always 22 | 10110 | 16 | $16 |
| 2 | ADR/8 | Address | 0-31 depending on Turbovac I Address (Default = 0) | 0 | 0 | $00 |
| 3 | PKE/16 | Parameter Type and Address – See PKE Explained/Query Designator | Dependent | | | |
| 4 | Dependent | | | |
| 5 | --/8 | Reserved | Always 0 | 0 | 0 | $00 |
| 6 | IND/8 | Parameter Index (Parameters with Array index numbers convert to binary, then Hex) | Dependent | | | |
| 7 | Parameter Value | Value of the Parameter to be set (up to 32 bit) | Dependent | | | |
| 8 | Dependent | | | |
| 9 | Dependent | | | |
| 10 | Dependent | | | |
| 11 | Status & Control | USS Control Word | Dependent | | | |
| 12 | Dependent | | | |
| 13 | Stator Frequency | Current Stator Frequency, returned with every string | Always 0 | 0 | 0 | $00 |
| 14 | Always 0 | 0 | 0 | $00 |
| 15 | Converter Temp | Current Frequency Converter Temperature (=P11) | Always 0 | 0 | 0 | $00 |
| 16 | Always 0 | 0 | 0 | $00 |
| 17 | Motor Current | Current Motor Current (=P5) | Always 0 | 0 | 0 | $00 |
| 18 | Always 0 | 0 | 0 | $00 |
| 19 | Reserved | Reserved | Always 0 | 0 | 0 | $00 |
| 20 | Always 0 | 0 | 0 | $00 |
| 21 | Intermed. Voltage | Current Intermediate Voltage (=P4) | Always 0 | 0 | 0 | $00 |
| 22 | Always 0 | 0 | 0 | $00 |
| 23 | Checksum | Recursive Calculation – See below how to calculate | Dependent | | | |

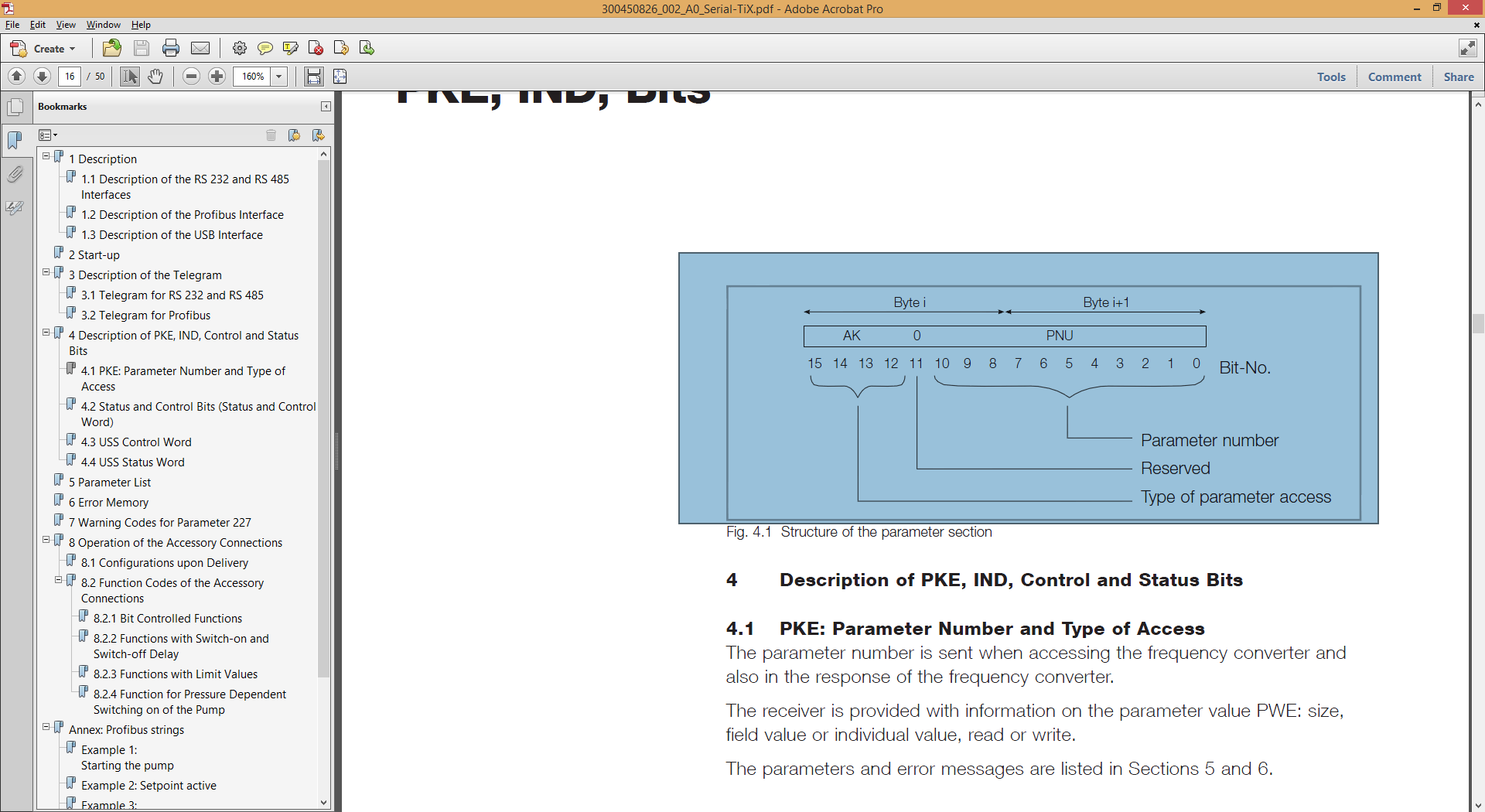
Telegram Reply Structure:

1. 24 Byte Structure:, each byte transfer as hex (simplest method) or can be done in decimal format (much more complicated).
2. Telegram Command structure:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Byte Number | Abbr/bits | Description | Value | Binary | Hex | Hex String |
| 0 | STX/8 | Start Byte | Always 2 | 10 | 2 | $02 |
| 1 | LG#/8 | Length of the Payload | Always 22 | 10110 | 16 | $16 |
| 2 | ADR/8 | Address | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 3 | PKE/16 | Parameter Type and Address – See PKE Explained/Reply Designator | Reply Val. | | | |
| 4 | Reply Val. | | | |
| 5 | --/8 | Reserved | 0 | 0 | 0 | $00 |
| 6 | IND/8 | Parameter Index (Parameters with Array index numbers convert to binary, then Hex) | Reply Val. | | | |
| 7 | Parameter Value | Value of the Parameter to be set (up to 32 bit) | Reply Val. | | | |
| 8 | Reply Val. | | | |
| 9 | Reply Val. | | | |
| 10 | Reply Val. | | | |
| 11 | Status & Control | USS Status Word | Reply Val. | | | |
| 12 | Reply Val. | | | |
| 13 | Stator Frequency | Current Stator Frequency, returned with every string | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 14 | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 15 | Converter Temp | Current Frequency Converter Temperature (=P11) | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 16 | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 17 | Motor Current | Current Motor Current (=P5) | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 18 | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 19 | Reserved | Reserved | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 20 | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 21 | Intermed. Voltage | Current Intermediate Voltage (=P4) | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 22 | Reply Val. | Reply Val. | Reply Val. | Reply Val. |
| 23 | Checksum | Recursive Calculation – See below how to calculate | Dependent | | | |

1. How to enter 16 bit values: Example binary 16 bit value = 1000100001000001:
   1. Convert binary to hex using windows calculator in programming mode
   2. 1000100001000001=8841 (Hex)
   3. If the Byte structure is Byte 3-4 for this 16 bit instruction, then enter $88 for byte 3, and $41 for byte 4
   4. If the value were less than 255(decimal), meaning 8 bit or less (due to leading values being 0), ie. 0000000000100001, convert to hex, giving 21, the byte 3 = $00, byte 4 = $21.
   5. Same methodology applies to 32 bit across 4 bytes.

**PKE (Byte 3-4) Explained:**



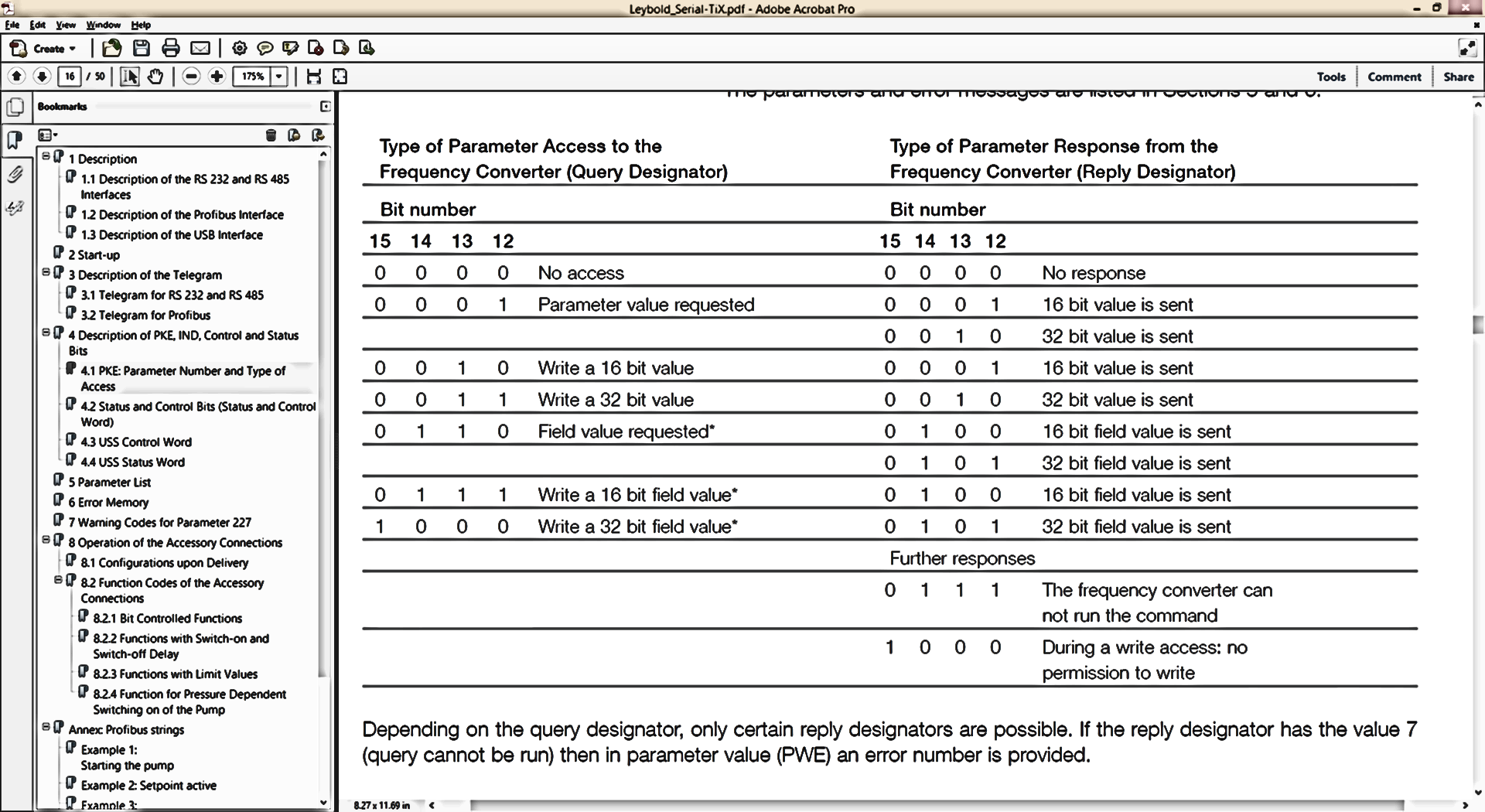
1. AK is the Parameter Access Type (4 bits (15/14/13/12) see table on p16.
2. Bit 11 is simply 0
3. Bits 10,9,8,7,6,5,4,3,2,1,0 is the binary representation of the parameter code (ie. 5 = Actual Motor Current = 101 (decimal notation) or 00001100101 (binary notation)
4. Sample:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AK=Parameter Request | | | | Res | Parameter = 5 (Actual Motor Current) | | | | | | | | | | |
| Byte 3 | | | | | | | | Byte 4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Binary:

Byte 3 = 00010000 = 10 (Hex) = $10

Byte 4 = 01100101 = 65 (Hex) = $65



If the Reply Designator returns 7 (0111), then Byte 10 will yield an error number (see return values below for bytes 7-10)

**IND Explained (6):**

1. IND is the parameter index. This refers to index numbers for an array of values within a parameter.
2. If there is no array, then the value is simply 0.
3. Otherwise the index value is represented either as a decimal or a bit number. Convert decimals to hex, bit values from binary to hex. (ie. Device type has an array of values, the last model has type 192. Convert 192 from decimal to hex, = C0.

**Parameter Value Explained (7-10):**

1. The Parameter value may be up to 32 bit. The value must be expressed as a combination of four 8-bit values in Bytes 7 through 10. To prepare this, first the 32 bit binary string must be created and then divided into the four 8-bit strings. Each of these strings are then converted to Hex. Example:
   1. Value of 68000:
      1. Binary is 10000100110100000 or (00000000000000010000100110100000)
      2. Divided into 4 strings of 8 = (00000000) (00000001) (00001001) (10100000)
      3. Convert each to hex = (00) (01) (09) (A0)
      4. Bytes 7 through 10 = $00$01$09$A0
2. Return Values:
   1. If the reply designator (bit 15,14,13,12) of Bytes 3-4 return a value of 7 (0111), then there is an error with the parameter request. In this case a fault code will be returned in byte 10. The errors are as follows:
      1. $00 = Impermissible parameter number
      2. $01 = parameter cannot be changed
      3. $02 = min/max restriction
      4. $12 = all other errors

**Status & Control Bits (11-12):**

1. In Sections 4.3 and 4.4 of the Leybold Serial communications document, there is a chart for Control words sending commands) and USS Status Words (chart for deciphering response).
2. For Commands, simply create a bnary code for each bit value (1 or 0) and then create the 16 bit binary sequence (15 – 0), then divide by two 8 bit sections, and create the Hex:
   1. Ie. USS Control word:
      1. Assume Options 0, and 10 are set to true (1) then the binary code is:
         1. 0000010000000001
         2. Divide by bytes: 00000100 00000001
         3. Convert each to hex: 04 01
         4. Byte 11-12 = $04$01
3. The response (status word) would be deciphered using the same coding as the control word.

**Checksum Explained (Byte 23):**

1. The checksum is an Xor value achieved by using the Xor summing value for the entire 24byte string. It is used to validate that the string did not get corrupted during transmission. It is calculated as follows:
   1. Assume the first 23 bytes are as follows:
      1. $02$16$00$10$AB$00$$$00$00$00$04$01$00$00$00$00$00$00$00$00$00$00
      2. Use the Xor function in the calculator (programmer mode), ignoring all zeroes, as follows:
         1. 2 Xor 16 Xor AB Xor 4 Xor 1 Xor = AA
         2. $AA is the checksum value (170 (decimal))