Document revision №4

21.12.2022

Arinst protocol

This note describes data protocol used on on following devices:

- ➔ Arinst SSA-TG
- ➔ Arinst SSA-TG-WA
- ➔ Arinst SSA-TG-LC
- ➔ Arinst SSA-TG-R2
- ➔ Arinst SSA-TG-LC-R2
- ➔ Asinst SSA-R2 (generator and tracking commands have no effect)

Described commands are not applicable for:

- ➔ Arinst SSA (please, reference Java example on arinst.net)
- ➔ Arinst SSA-R2 SignalHunter
- ➔ Arinst VR 23-6200
- ➔ Arinst VR 1-6200

1. Package format as follows:
   “<command_name> <argument1 argument2 … argumentX> <package_index> \r\n”
   Commands could have from 0 to 7 arguments with command specific formatting and order.
   CR/LF (‘\r\n”) used as command terminator.
2. Commands:
   2.1. **Generator on:**
   Request**:**
   “gon *Index*\r\n”
   Index: current package index.

   Response:
   “gon *Index*\r\ncomplete\r\n”

   2.2. **Generator off:**
   Request**:**
   “gof *Index* \r\n”
   Index: current package index.

   Response:
   “gof *Index*\r\ncomplete\r\n”

   2.3. **Generator set frequency:**
   Request**:**
   “scf *Frequency Index*\r\n”
   Frequency: desired frequency in Hz
   Index: current package index.

   Response:
   scf *Index*\r\n*Status*\r\ncomplete\r\n”
   Staus: indicates whether operation was successful. Could be “success” or “failure”.
   Generator on command must be sent prior to this command

   2.3.1. Example:
   Set generator at 75MHz. Command is 5th in session
   Request**:**
   “scf 75000000 5 \r\n”

Response:
“scf 5\r\nsucsess\r\ncomplete\r\n”

2.4. **Generator set amplitude:**

Request**:**
“sga *Formated_Amplitude Index*\r\n”
Formatted Amplitude desired output amplitude. Calculated as ((amplitude+15) *100) + 10000. Amplitude value must be between -15 dBm and -25 dBm.
Index: current package index.

Response:
“sga *Index*\r\ncomplete\r\n”
Generator on command must be sent prior to this command

2.4.1. **Example:**

Set generator output equal to -17dBm. Command is 7th in session.
Request**:**
“sga 9800 7\r\n”

Response:
“sga 7r\ncomplete\r\n”

2.5. **Scan range:**

Request**:**
scn20 *Start Stop Step Timeout Samples Intermid_freq Formatted_Attenuation Index*\r\n

Start: begin frequency in Hz.
Stop: end frequency in Hz.
Step: desired step in Hz.
Timeout: should be kept on 200.
Samples: adc samples per point. Should be kept at 20.
Intermid_freq: hardware specific. Should be kept at 10700000.
Formatted_Attenuation: desired input attenuation. Calculated as (attenuation * 100) + 10000.
Attenuation value must be between 0dB and -30dB.
Index: current package index.

Response:
“\r\nscn20 *Start Index*\r\n<*encoded_data*><*elapsed_time*>\r\ncomplete\r\n”

encoded_data is byte sequence, containing encoded amplitude values and terminated with two 0xFF bytes. Length of this sequence in bytes is equal to:

(requested_points_ammount * 2) +2

Please, adjust receiving length accordingly. If for some reason required amount of memory cannot be allocated, split required range into smaller to meet memory restrictions.
Each amplitude value represented as two bytes containing value index and encoded amplitude. Value index is contained in 5 most significant bits. This could be used to verify data integrity. Encoded amplitude value is 11 least significant bits. To decode, use following formula:
amplitude = (80 * 10.0 - encoded amplitude) / 10.0;
elapsed_time time that was spent getting data in ms;

2.5.1. **Example:**

Scan from 500 MHz to 1500 MHz with 5 MHz step and -15 dB input attenuation, 8th command in session

Request**:**
scn20 500000000 1500000000 5000000 200 20 10700000 8500 8\r\n

Response:
“\r\nscn20 500000000 *8*\r\n<encoded_data>*69*\r*n*complete\r\n”

Decoding:
encoded_data[] = {31, 92, 39, 85, 47, 94 … 255, 255};
Decoding first amplitude value:
1: Assemble 16-bit encoded value:
uint16_t val = 31 << 8 | 92;
2: Extract data index and amplitude:
uint16_t data = val & 0b0000011111111111;
uint16_t index = (val & 0b1111100000000000)>>11;
3: Calculate amplitude:
float amplitude = (10.0*80.0 - data)/10.0 + 15; \\ 15 is attenuation value used in this case.

Example of decoding function written in C:

```c
float decode_binary(uint8_t firstbyte, uint8_t secondbyte)
{
        uint16_t val = firstbyte << 8 | secondbyte;
        uint16_t data = val & 0b0000011111111111;
        uint16_t index = (val & 0b1111100000000000)>>11;
        float amplitude = (10.0*80.0 - data)/10.0;
        return amplitude;
}
```

2.6. Scan rage with tracking:

Same as Scan range, but command name is scn22. Tracking generator should be turned on prior to sending command. Otherwise, device will behave like scn20 was send.