Analysis of PTK Servo programming protocol

The protocol of PTK Servos like 8815, 8812 is not publicly documented, it was reverse engineered (accordingly to Directive (EU) 2009/24/EC) by sniffing the servo signal wire with a logic analyzer. The tools used to perform this operation are:

- a cheap 8 channel login analyzer up to 24M samples per second
- sysrok pulseview logic analyzer software

Servo Reading

Without any servo attached to the USB programmer [1] the software on the PC transmit a series of pulses and the stop. By analyzing minimum pulse width it was possible to determine the baud rate that is 115200.

By adding a UART Protocol decoder in the sysrok pulseview it was possible to see the 10 byte of the sequence: 175,250,0,10,56,0,0,0,0,66.

With the servo connected, 8 packets of 19 bytes are seen with a pause between each packet of about 50ms

N1	175	250	0	10	56	0	0	0	0	66	175	250	0	9	3	32	7	208	3
N2	175	250	0	10	61	0	0	0	0	71	175	250	0	9	0	29	0	12	50
N3	175	250	0	10	63	0	0	0	0	73	175	250	0	9	4	176	0	0	189
N4	175	250	0	10	80	0	0	0	0	90	175	250	0	9	7	208	39	16	23
N5	175	250	0	10	81	0	0	0	0	91	175	250	0	9	0	65	0	65	139
N6	175	250	0	10	82	0	0	0	0	92	175	250	0	9	16	0	0	0	25
N7	175	250	0	10	83	0	0	0	0	93	175	250	0	9	0	15	0	0	24
N8	175	250	0	10	84	0	0	0	0	94	175	250	0	9	0	1	0	0	10

As we can see the first part of N1 packet is exactly the same as without servo so the first 10 byte part should be the inquiry and the second part should be the answer of the servo (the servo answers really very quickly)

Also we can see that there is a common pattern in the packets: apparently there is a common header of 175,250,0,10 for inquiries and 175,250,0,9 for answers.

It was rather simple to suppose that 175,250 (0xAF, 0XFA) is a header sequence and 0x00, 0x10 (or 0x00, 0x09) is the length of the packet in big endian format.

The fifth byte of the inquiry is a sort of command, then 4 zeros and reasonably there is a checksum.

Understanding the checksum algorithm was rather tricky but the common pattern is that the sum of the bytes of the length plus the payload always matched with the checksum 0+10+56+0+0+0+0=66, the same is valid for the answer if the sum is made in an unsigned 8 bit number 0+9+3+32+7+208=259 → 259%256=3

To understand the meaning of the answer and by consequence understand the meaning of the inquiry command, servo parameters were changed one by one and the servo was reread after each change, the analysis brought to the following table:

	175	250	0	10	56	0	0	0	0	66	175	250	0	9	3	32	7	208	3
N1	Header		Packet length							CK	Header		Packet length		**overload protection**		**overload time**		CK
	175	250	0	10	61	0	0	0	0	71	175	250	0	9	0	29	0	12	50
N2	Header		Packet length							CK	Header		Packet length		**Gain**		**Dumping**		CK
	175	250	0	10	63	0	0	0	0	73	175	250	0	9	4	176	0	0	189
N3	Header		Packet length							CK	Header		Packet length		**Max Power**				CK
	175	250	0	10	80	0	0	0	0	90	175	250	0	9	7	208	39	16	23
N4	Header		Packet length							CK	Header		Packet length		**Min pulse width*4**		**Max pulse width*4**		CK
	175	250	0	10	81	0	0	0	0	91	175	250	0	9	0	65	0	65	139
N5	Header		Packet length							CK	Header		Packet length		**Min Angle**		**Max Angle**		CK
	175	250	0	10	82	0	0	0	0	92	175	250	0	9	16	0	0	0	25
N6	Header		Packet length							CK	Header		Packet length		**Neutral**				CK
	175	250	0	10	83	0	0	0	0	93	175	250	0	9	0	15	0	0	24
N7	Header		Packet length							CK	Header		Packet length		**Deadband**				CK
	175	250	0	10	84	0	0	0	0	94	175	250	0	9	0	1	0	0	10
N8	Header		Packet length							CK	Header		Packet length		**Inversion**				CK

The analysis confirmed that all values are sent/received MSB (Big Endian), **It was not possible to determine Soft Start position as the 8812 and 8815 servos do not seem to support the function**.

[1] The USB programmer is a normal USB → Serial adapter with a 470 Ohm resistor connecting TX and RX to transform the adapter in a half duplex single wire serial

Servo programming

An analysis with the logic analyzer was also made to the programming sequence by changing values one by one. To program the servo programming packet are sent with a logic of “fire and forget” the servo never answer to programming commands, the same sequence is also sent without servo attached.

N1	175	250	0	10	1	0	29	0	0	40
	Header		Packet length			**Gain**				CK
N2	175	250	0	10	5	0	12	0	0	27
	Header		Packet length			**Dumping**				CK
N3	175	250	0	10	18	3	32	7	208	22
	Header		Packet length			**Overload Power**		**Overload Time**		CK
N4	175	250	0	10	24	4	176	0	0	214
	Header		Packet length			**Max Power**				CK
N5	175	250	0	10	64	0	0	0	65	139
	Header		Packet length					**Min Angle**		CK
N6	175	250	0	10	65	0	0	0	65	140
	Header		Packet length					**Max Angle**		CK
N7	175	250	0	10	66	0	0	16	0	92
	Header		Packet length					**Neutral**		CK
N8	175	250	0	10	67	0	0	0	15	92
	Header		Packet length					**Deadband**		CK
N9	175	250	0	10	68	0	0	0	1	79
	Header		Packet length					**Inversion**		CK
N10	175	250	0	10	69*	0	0	0	0	79
	Header		Packet length			**unknown**		**unknown**		CK
N11	175	250	0	10	72	7	208	39	16	96
	Header		Packet length			**Minpulse * 4**		**Maxpulse * 4**		CK
N12	175	250	0	10	70	0	0	0	1	81
	Header		Packet length					**Softstart**		CK

* The command 69 (N10) is present but always write 0 0 0 0 as parameters the meaning is unknown.