Latency of Futaba FF8s PPM and PCM Radio Controller

W.Pasman, 1 november 2003

Introduction

End to end latency is the time difference between the moment we toggle the stick (or switch) and the actual output of the servo. In general, every component in the chain from stick to servo influences the end to end latency (Figure 1). To simplify matters and to make our results more general, we cut the servo out of the chain and only measure the latency up to the output of the receiver. For readers not familiar with the various modulation schemes along the way I describe PPM and PCM, and how it is implemented exactly in our case. Then I will briefly sketch how you can estimate the end to end latency when your servo would be added to the chain (Figure 1 with your servo in place).

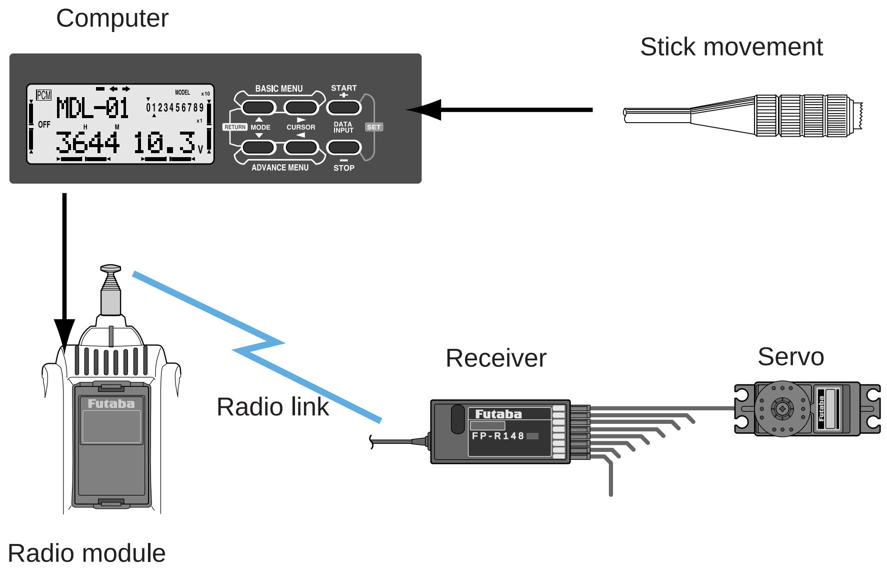


Figure 1. Every component of the radio system adds to the latency.

The radio link is quite slow, this should be a major source of latency. Computers are getting faster every day, so I would expect that nowadays the transmitter and receiver computer should not introduce major latencies.

PCM and PPM

In this section I shortly describe two modulation schemes: Pulse Code Modulation (**PCM**) and Pulse Position Modulation (**PPM**) modulation.

In PPMvalues (servo positions in our case) are encoded by varying the length of a pulse. A longer pulse means a larger value, and a shorter pulse a lower value. In our radio system, PPM modulation is used in two places: for the radio link from transmitter to receiver, and for the servo pulses going from the receiver to the servo. Servos are always driven with PPM, while the radio link can be either PCM or PPM. Therefore, a complete radio system is called PPM or PCM depending on the type of radio link being used.

In PCM, values are send as a string of "low" and "high" pulses, each pulse having the same length. Every pulse is called a bit. A predetermined number of bits (10 in our case) is collected to make up a digital number, and this number is the servo position.

For the radio link, information for multiple channels has to be transferred over the same radio link. To organize this, all channel data is grouped into a "frame" holding the target position of all servos, before being transmitted.
The following paragraphs explain in more detail how PPM and PCM are being used in our radio system.

PPM servo pulse modulation

For the pulses between receiver and servo, two voltage levels are being used: a "low" is transmitted by putting 0V on the wire, and a "high" by 3V (probably fluctiating a bit, depending on the battery voltage). The voltage is 0V most of the time, only once in about 15ms to 20 ms (depending on the radio link modulation) there comes a "high" pulse. The length of the "high" pulse describes the servo position, with 1ms being one servo extreme, 1.5ms being the servo exactly at the center, and 2ms being the other extreme.

PPM radio link modulation

For PPM radio link modulation, the positions of all channels (1 to 9 in Futaba PCM1024) are transmitted one after the other using PPM. Although the radio has only one frequency reserved (35.090MHz), it actually alternates between two frequencies: 35.087MHz ("low") and 35.093MHz ("high"). The PPM in this case only refers to the length of the "high" signal, and after a "high" pulse a "low" is send for a fixed duration of 0.3ms. After that the next channel is transmitted. As with the servos pulses, the duration of "high" is 1.0ms for one servo extreme, 1.5ms exactly in the middle and 2.0ms for the other extreme.
If every servo would be at its 2.0ms position, it would require 9*2=18ms to transmit all information. Additionally some synchronisation pulse is needed between the frames, lasting 4.5ms in our case. Thus, PPM frames are sent every 22.5ms. After a servo received an update the next update thus will come only 22.5ms later. Therefore, in PPM the delay of the servo[1] starts at 0, grows to 22.5ms, and then jumps back to 0ms.
Because the modulation scheme of the servo pulses exactly mimics the radio link scheme, the pulses can be forwarded straightly to the servos, immediately as the radio link data comes in. Therefore, the latency between data coming in for a particular servo data and the servo data going out from the receiver can be close to 0ms.

PCM radio link modulation

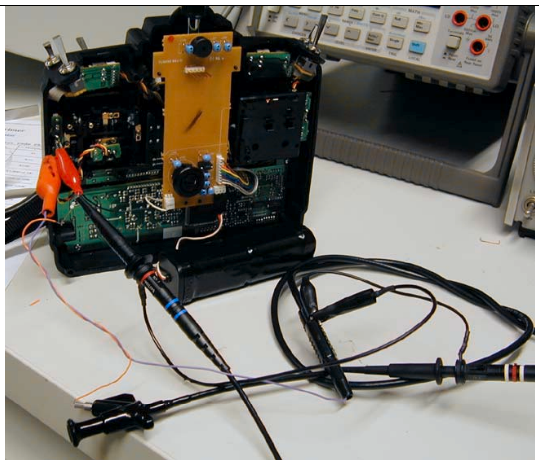
As we already mentioned, in Robbe/Futaba 1024 PCM, a 10 bits coding is used to transmit digitally the position of each servo. But those numbers are not send one after another, instead a more complex coding is applied.
Half of the position numbers are transmitted raw, for the the other half only the difference between this number and the previous number that was sent for that channel is sent (5 digits I think). In odd frames, channel 1, 3, 5, 7 and 9 are digitally coded as 10 bits numbers, followed by the 5 bit differences of channel 2, 4, 6 and 8. Then follows some extra data, and the frame finishes with a 16 bit CRC (kind of checksum, allowing to see whether all frame bits have been received correct). The total frame takes 14.25ms to transmit over the radio link. In the even frames, channel 1, 3, 5, 7 and 9 are coded as 5bit differences, and channel 2, 4, 6 and 8 as full 10 bit numbers. Again follows some extra data and 16 bit CRC. As a detail, the servo data in the even frames have 5 bits less data, but this difference is compensated with a few bits extra in the extra data block.
The 14.25ms frame time is also reflected in the servo pulses, which also appear with a 14.25ms interval instead of 22.5ms.

[1] Not the full latency that we are looking for, but the latency in the receiver only.

Servo data can only be picked out of the frame and sent to the servo after the CRC is checked. Therefore the receiver has no option but to wait until the CRC arrived. The CRC arrives nearly 14ms after channel 1 was transmitted, causing large delays for the lower channels. On top of this the CRC has to be checked, causing some (probably low) processing latency in the receiver.
Thus, in PCM latencies will be different for every channel. For channel 1 the latency[1] would start at at least 14ms, grow to 14+14.25=28.25ms, and then jumps back to 14ms. The time needed to check the CRC comes on top of this.

Latency Measurement Setup

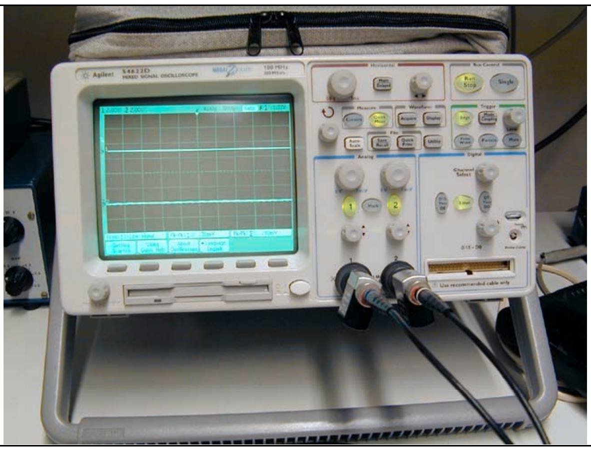
We use a Futaba T8UPS super series PCM1024 digital proportional R/C transmitter. All computer settings were as per factory default[2], heli swash mode 1 was selected. For PPM measurements we used the Futaba FP-R116FB 35MHz 6 channel single conversion receiver. For PCM we used the FP-R138DP 35MHz 8 channel dual conversion receiver. We used channel 69 (35.090 MHz).
We use a HP 3310A signal generator to create 'stick jumps' (Figure 2). Internally the sticks have 5kΩ potmeters with one end connected to 0V and the other end to +5V. We set up the signal generator to generate a square wave jumping between 0V and 5V as well. We make sure the stick is in the middle position. Now we just inject a signal via a 2.7kΩ resistor onto the output pin of the potmeter (Figure 3). This gives a less than a full scale swing (less than 0-5V), but this is good enough visible on the scope and we want to stay on the safe side. Additionally we guide the signal from the generator to channel 1 of the scope. We straightly connect the output of the receiver channel (the one corresponding to the potmeter we hooked up of course) to channel 2 of the oscilloscope.

3310A FUNCTION GENERATOR HEWLETT•PACKARD RANGE FUNCTION DC OFFSET OUTPUT LEVEL LINE VCO INPUT SYNC OUTPUT LOW HIGH	
Figure 2. HP 3310A Signal generator.	Figure 3. Transmitter with probes. 2K2 resistor is soldered straight to connector plug. Wires to freq generator are twisted to reduce HF reception...

For the PPM system measurements, a FLUKE PM3082 100MHz analog scope sufficed (Figure 4). We used Philips 10:1 probes. We were picking up a lot of radio noise, but after shortening and twisting the wires connecting to the transmitter the amount was just acceptable. Unfortunately we ran into measurement

[2] all mixers inhibited, failsafe to 'normal' = keep last setting for all channels, no exponential or dual rate, all ATV to 100%, revolution mixing set up, -20,-10,0,10,20 for positions 1 to 5.

problems with the PCM system measurements, forcing us to switch to a Agilent (HP) 5422D Digital 100MHz scope (Figure 5).

	
Figure 4. FLUKE PM3082 analog 100MHz scope.	Figure 5. Agilent (HP) 5422D Digital 100MHz scope.

To do readouts and make fotos, we always have to fiddle with the frequency knob on the signal generator until the generator runs 'in sync' with the radio system. Futaba system PCM frame is 14.25ms, and PPM frames are 22.5ms, so we will have the signal generator at a multiple of this to get a stable scope image. What differs every time is the time difference between the pulse of the generator and the servo pulse coming out of the receiver. Such an adjustable time difference is exactly what we need as latency is our point of interest.

PPM latency

Latency measurements are a bit tricky. As mentioned, we fiddle with the frequency knob to move the position of the signal generator with respect to the radio system. We do this accurately, to find the moment the pulse width of the radio changes, and exactly the moment that this occurs gives us the latency.
PPM gives very consistent results. The pulse width from the receiver changes abrubtly when a certain latency is reached. Figure 6-8 shows the results for the channels. The distance between the moment the stick goes 'high' (left red bar) and the moment the servo output (the right red bar) gives the time difference, and the pictures show for every channel that time difference where the width of the servo pulse changes from wide to small, thus the latency.

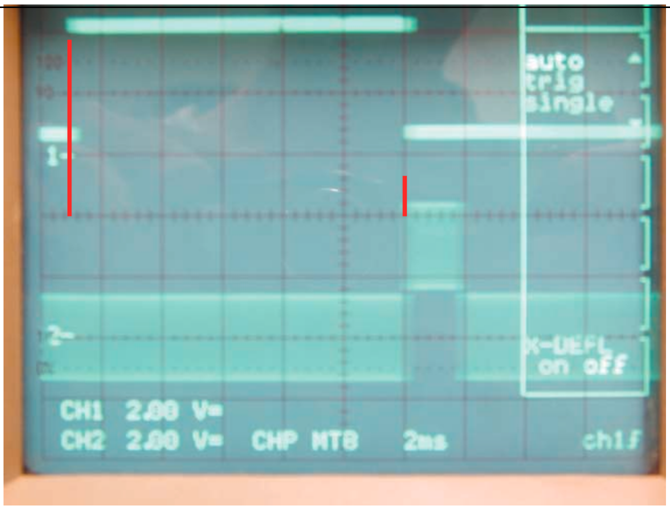
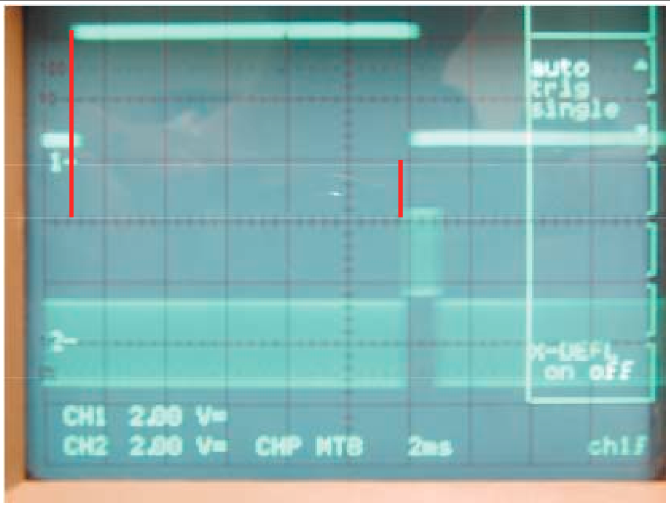
	

Figure 6a. Channel 1, just above the latency	Figure 6b. Channel 1, just below the latency

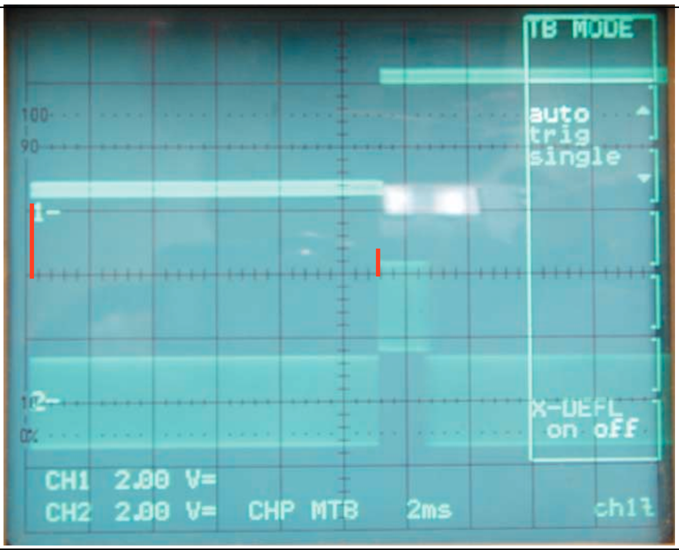
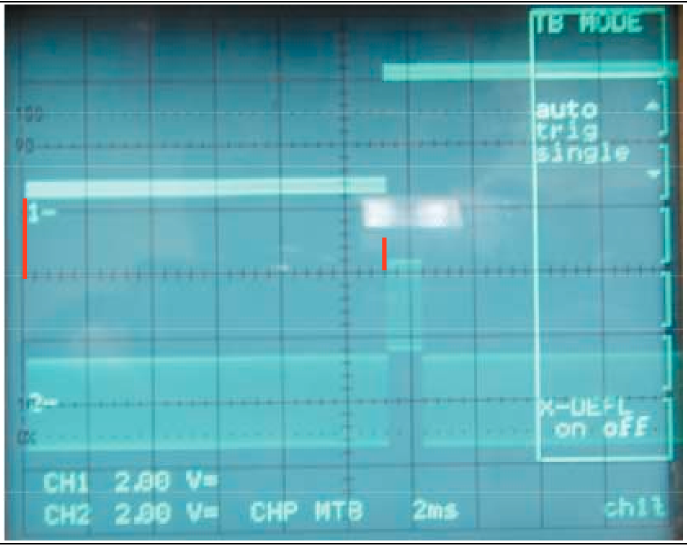
	

Figure 7a. Channel 2 just above latency	Figure 7b. Channel 2 just below latency

Table 1 summarizes the latency results for PPM. Note that we have the the radio and the signal generator run in sync, and therefore our results indicate the **minimum** time needed for the stick input to get to the receiver (thus, the minimum latency). However we need to know the maximum latency. After this signal arrived, this signal is supposed to be valid for the full length of the frame, which is 22.5ms. Thus, if the stick position would change just too late, it will be sampled only 22.5ms later, and it will appear then at the servo at the minimal latency time we found. Thus, the maximum latency is 22.5ms larger.

Table 1. Latency results for PPM.

Channel	minimum PPM Latency (ms)	maximum PPM latency (ms)
1	10.8	33.3
2	11.6	34.1
3	9.6	32.1

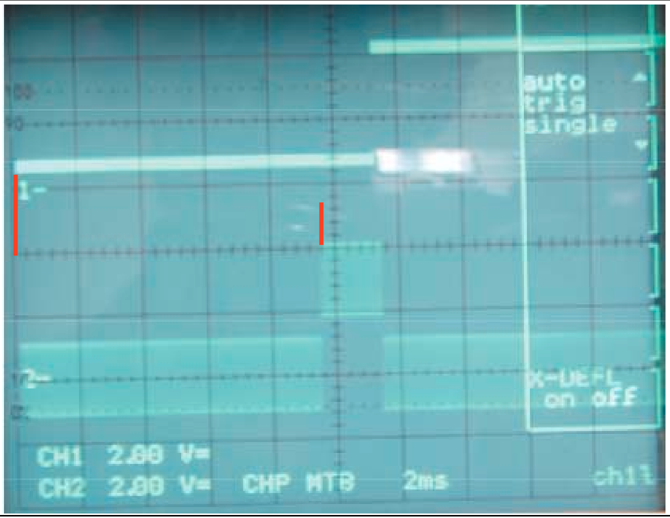


Figure 8a. Channel 3 just above latency

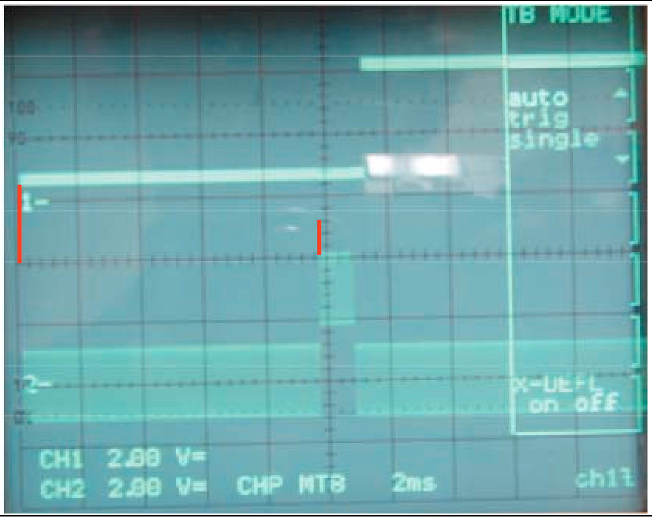


Figure 8b. Channel 3 just below latency

PCM latency

PCM latency measurements showed a lot trickier than the PPM measurements. What especially showed tricky is that the minimum latency is not constant, and that the corresponding servo pulse can occur a long time (up to 75ms, this is more than 5 frames!!) after the stick jumped. Figure 9 shows this. Not expecting this, on the analog scope we just saw wide and narrow servo pulses through each other all the time.
It is not clear what the source of this varying latency is.

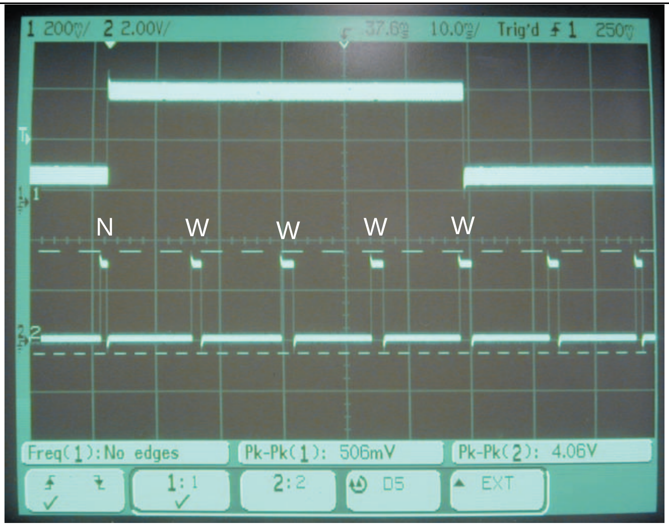


Figure 9a. Here the latency is low, the first pulse after the switch jump is already wide (W).

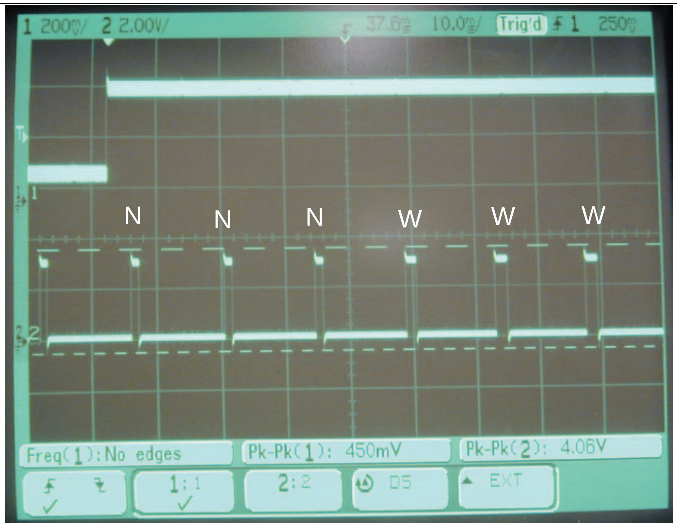


Figure 9b. A moment later, the latency is a lot larger, the first three pulses are still narrow (N).

So, instead of looking for a nice and clear jump of the pulse width as we did with PPM, we now have to look for jitter on the servo pulses (that is, latencies where the servo pulses do not have a consistent width). Because the jitter can occur so long after the stick jump, we lowered the stick jump speed down to 4.7Hz (15 frames). We have to lower the scope speed similarly, to catch the process. With the analog scope, this would result in barely readable pulses, making it impossible to trace the servo jitter. What we need is to start the scope at a predetermined time after the stick jump, so that we can keep good resolution at the time slice under inspection. Some analog scopes may have the ability for a delayed trigger but the 'delay' button on our Fluke seems to do something else, so we switched to the digital scope.

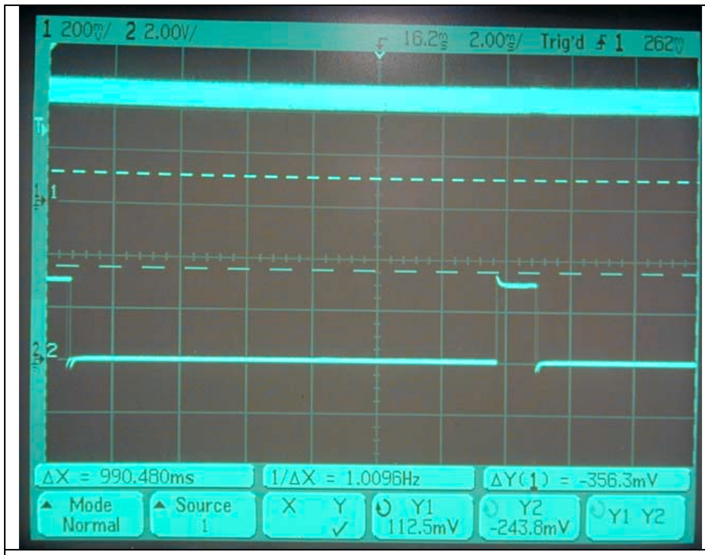


Figure 10. The center point (number in the middle of the top) shows the current offset to the trigger (the upgoing flank of the stick).

With the digital scope life is much easier, zoom in as needed. The zero point automatically is set at the low-to-high jump of the stick. Correct the scope's centerpoint (see Figure 10) until pulses start to jitter just right of the centerpoint (minimum latency) or stop just left of the center position (maximum latency). Read out the position of the centerpoint to get the latency value!

Table 2. Latency results for PCM.

Channel	minimum latency (ms)	maximum latency (ms)
1	16.2	73.1
2	12.7	70.1
3	17.2	74.4

I also checked whether the PCM latency is caused by large signal changes. Large jumps very well may be too large for the delta-coding used half of the time, thus causing extra latency that happens only with large changes. I reduced the stick jump and checked if the latency changed. Unfortunately the jump could not be set below 0.5V, otherwise it would be too hard to see the servo pulse width jitter. Lowering the voltage this much did not improve (lower) the latency.
An interesting point is that sequential driving of 8 servos to their maximum (2ms pulse) would take 8*2=16ms. However the time between servo drive pulses is always 14.25ms, so there is not enough time to send all servo pulses sequentially. Probably servo pulses are not send strictly serially to the servos, but partially overlapping. It would be nice to look how this was done.

Also it would be nice to measure the radio pulses being transmitted, and the data entering the radio module[3] . This could give a clue about where the latency of PCM coding actually is caused, in the transmitter or in the receiver.

Adding the servo latency

All latencies here were end-to-end latencies but excluding the servo.For fast movements you should add the usual specified speed/60˚ to find the total latency. Thus for a 0.06s/60˚, add 60ms to the values we reported. For slower movements you probably can add a quarter of this value instead.However, measurements would be needed to give this estimation a better basis.

Conclusions

For PPM we found a latency of 34 ms. This is the worst case, end to end latency, and is very similar for all channels and over time. The fastest response we found was within 9.2ms, and all measured channels have similar fastest responses as well. These latencies are quite reasonable, considering the 22.5ms minimum latency coming from the transmission format alone.
For PCM we found a latency of 74.4 ms worst case. Other channels have similar latencies. But over time, the latency is highly variable for all channels. The fastest response was 12.7ms, but other channels were substantially slower. The latency of 74.4ms seems excessive given a minimum of 28.25ms from the transmission format. There may go something very inefficient, such as superfluous buffering or inefficient servo pulse generation, or the receiver mechanism that checks the CRCs is really slow.
We suggested measurement opportunities to find out more details about the source(s) of the latency.

Acknowledgements

I'm very happy that the people from the Microelectronics Department at Delft University of Technology allowed me to use their equipment for doing the measurements.

References

Rother, P. (2000). The million dollar question: PCM or PPM? Possibilities, performance? Translation Wim Hanssens. http://www.aerodesign.de/peter/2000/PCM/PCM_PPM_eng.html
Futaba (2000). PCM 1024 System Info. www.futaba-rc.com/faq/faq-pcm1024.html

[3] I expect that these differ very little, and that the radio module only does the FM modulation but no coding, timing or buffering