Precise measurement of magnet currents in Alpha-g experiment

Supervisors: Jaspal Singh (University of Manchester)
Dr William Bertsche (University of Manchester)

Author: Samuel Tabone (University of Malta)

Abstract

In the *Alpha-g* experiments, precise manipulation of magnetic fields is essential to experimentation with anti-hydrogen. Hence, the magnet current controller requires precise measurement of the magnet current. The current controller is a PID system which uses DC Current Transformers (DCCTs) to read the magnet currents and provide feedback to the controller. As of yet, the feedback signal is analogue. This makes it more susceptible to noise, the presence of which would reduce the accuracy of the error signal being fed into the controller. Analogue filtering in the feedback loop is also undesirable due to it effecting the response of the closed loop system. Thus, it was determined that the DCCTs should be measured by the high precision HMP-7177 ADC, produced by the CERN ADC team. The data is then to be transferred to the controller digitally. This approach was chosen due to digital data transfer having higher signal fidelity in the presence of noise. The scope of this project is the creation of a *LabView* based system to sample the ADC UART output of the CERN ADC and transfer the data to the computer-based system, where the data is analysed and used in the closed loop system

Acknowledgements

This project would not have been possible had it not been for the help and guidance provided to me by my supervisors. Furthermore, I'd like to thank all the people in the Alpha experiment for making me feel welcome and allowing me to witness the experiment.

Table of Contents

Table of Figures

1.0 Introduction

The test bed system is based on the National Instruments *myRio-1900* FPGA. The *myRio-1900* includes multiple general Digital Input/Output (DIO) pins as well as a dedicated UART module. However, the UART module on the system can only transfer data up to the highest standard UART baud rate, 230,400 bits per second. The *HMP-7177* has an output baud rate of 5Mbits per second, rendering the built-in UART module too slow for use. However, the *myRio* can sample its DIOs at 40MHz. This provides 8 samples per output bit of the ADC. Using these 8 samples, the controller can determine whether the bit read was a 1 or a 0. The code can also be adapted for the *CompactRio 3039*.

2.0 ADC characterisation.

The ADC was tested to determine the variation of the output string and the input voltage read by the ADC. The test was performed by attaching a dongle, kindly provide to us by *Nikolai Beev* from the CERN ADC team. This dongle receives the 17-byte data packets from the ADC at 10kHz, removes the bytes referring the ADC temperature and status, and sends the output as 10-byte packages at 1kHz. The 10-byte packet consists of two 4-byte codes referring to the voltage read by both ADCs (since each chassis contains two ADCs) and a 2-byte termination character. More importantly, the baud rate of the bitstream is reduced to 230,400 bits per second, allowing the test to be performed using the UART module on the *myRio*.

In *Figure 1* and *Figure 2* below, the test setup can be seen. *Figure 1* depicts the *myRio* RX pin connected to the Dongle TX pin. *Figure 2* shows the controlled voltage input provided by a DC power supply and the 5-digit precision multimeter used to read the power supply voltage.

Figure 1- NI myRio connected to ADC Dongle

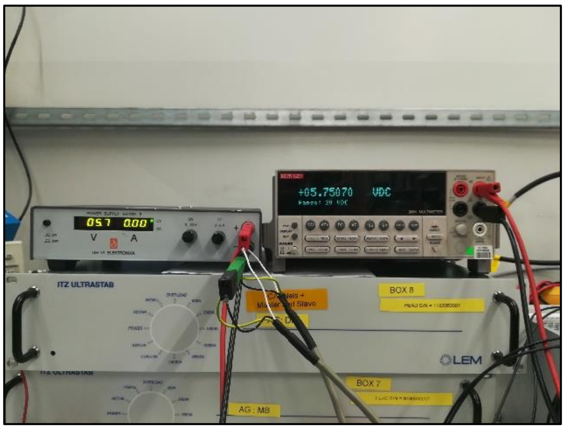

Figure 2 - Multimeter and power supply

The multimeter voltage is then used to compare with the ADC readings. Before starting the experiment, the ADC differential inputs were shorted together. A "Zero calibration" was performed to determine the actual reading that corresponds to 0V. A similar process was followed for the "Full scale calibration", whereby a 10V input voltage was applied across the differential inputs. Once these calibrated inputs were determined, the code provided with the dongle was adapted to use the UART module of the *myRio*.

The graphs shown in *Figure 3* and *Figure 4* depict the variation of the voltage read by ADC A and ADC B respectively with that of the multimeter. As can be clearly observed, there is little to no Integral Non-linearity (INL) present. Hence, it can be assumed that the numerical value of the output code will vary linearly with the input value. The table of results can be found in the Appendix.

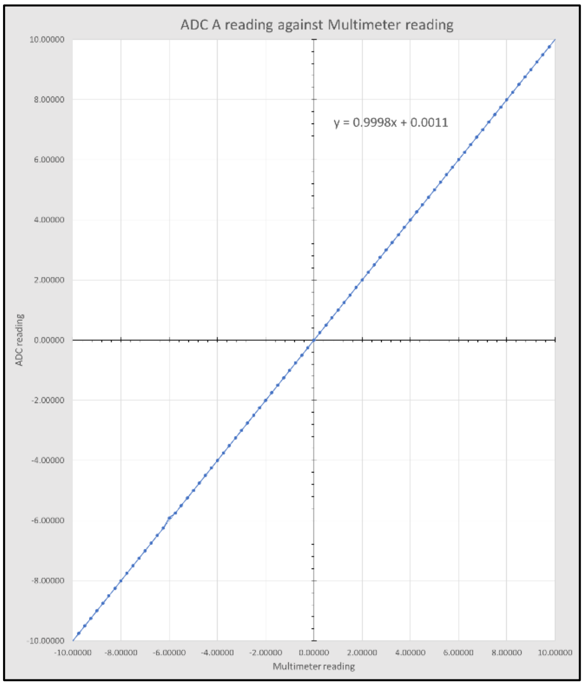

Figure 3 - ADC A versus multimeter reading

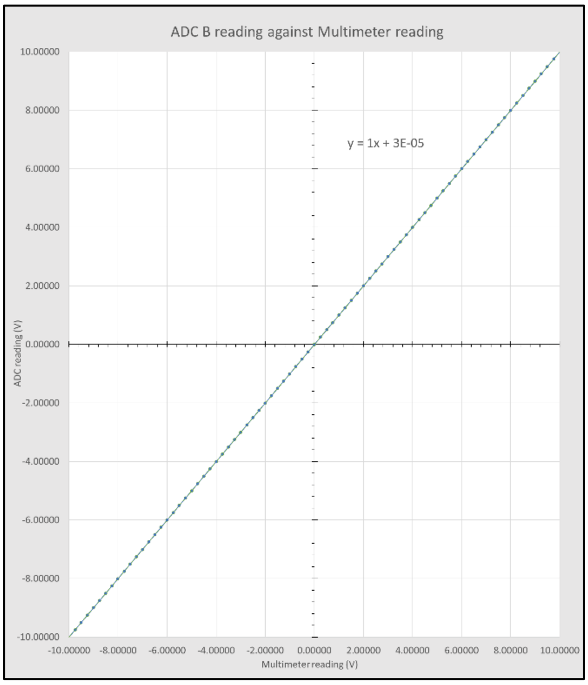

Figure 4 - ADC B reading versus multimeter reading

3.0 Sampling the ADC output

3.1 FPGA sampling code

To sample the output of the ADC, an FPGA program was developed. The code first waits for the user to push a sample button to initiate the program. The main program consists of a single-cycle timed loop operating at 40MHz. In the loop, a state machine is used based on whether the UART output is idle or sending data.

The default state of the state machine is "*detect idle*". In this state, the program loops until 250 consecutive samples are equal to a predefined idle state, in this case, logic '1'. This is done to ensure that the samples begin at the beginning of a sample rather than in the middle of one.

When this condition is fulfilled, the state machine moves to the "*wait for edge*" state. In this state, the program loops till the latest digital sample is equal to '0', indicating the first start bit of the data packet. While waiting, the array that stores the data packet samples is not updated. When an edge is detected, a flag is changed from false to true that allows the consequent digital samples to be inserted into the array. This marks the beginning of the "*sample*" state. The "*sample*" state takes a user defined amount (in this case 1560) of samples at an interval of 25ns and stores them in an array. Finally, after these samples are taken, the "*stop*" state is initiated, which breaks out of the loop and resets the program.

3.2 Processing the data

The ADC UART output is split into four components:

- The first four words correspond to the voltage reading.
- The second four words correspond to the internal temperature reading of the Mezzanine board.
- The next 8 words are all zeros.
- The final word is a CRC for error correction

Note that each word consists of 11 bits; one start bit (LOW), eight data bits, one parity bit (LOW except for CRC which is HIGH) and one stop bit (HIGH). Furthermore, the LSBs of the data bits are transmitted first, with the least significant byte of each four-word packet being the first to be transmitted.

Hence, the code must be able to decipher the data bits from the other bits. The FPGA code mentioned above was adapted such that the DIO samples were sent, via a FIFO to another single-cycle timed loop which decoded the sample. If the FIFO is not timed out and the sample value is equal to the start bit logic, the default case "*Read Start Bits" can begin.* Starting from the previous DIO sample, a loop is iterated eight times. On an index specified by the user in a config file (in this case 3), the sample value is passed to a boolean variable that stores the start bit value. After eight iterations, the "*Read Data Bits*" state begins.

Similarly to the previous state the "*Read Data Bits*" state iterates eight times per bit and stores one of the samples as the value for that bit. The bit values are stored in an array. After eight bits are read, the parity bit can be read. The "*Read Parity Bit*" state is identical to the "*Read Start Bits*" case. The same can be said for the stop bit. At the end of the "*Read Stop Bit*" case, all the stored variables from that word are inserted into an array. Note that the data bits array is converted to an 8-bit unsigned integer value before being sent to an array. The whole process is repeated for each word. *Figure 5* shows the part of the front panel that shows the data read and processed.

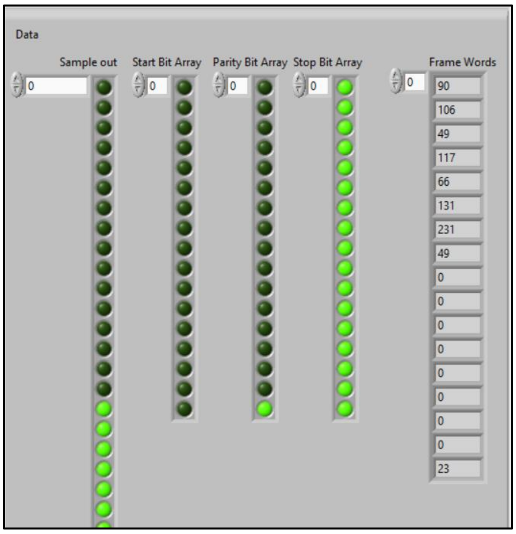

Figure 5 - Part of front panel that depicts the sample array, start, stop and parity arrays and the U8 data array.

3.3 Using code on CompactRio 3039

The code was then transferred to a *CompactRio 3039* project. The *CompactRio* was equipped with a NI 9401 digital I/O module, which was used to test if the code. A test was performed to check if the code read from the ADC for different input voltages corresponded to the data obtained in *Section 2.0*. The test procedure adopted was the same. The ADC was first calibrated for full scale and zero voltage codes. The voltage code was then read for different power supply inputs. This reading was then compared to a multimeter reading. The data obtained in the experiment can also be found in the Appendix. *Figure 6* shows the read voltage against the multimeter voltage:

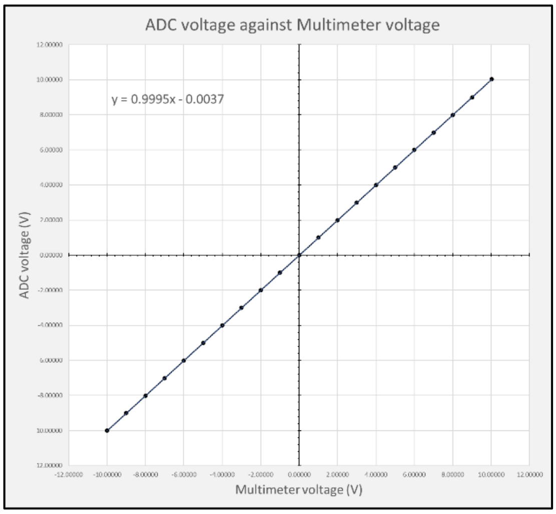

Figure 6 - Voltage read from ADC via NI 9401 against multimeter voltage

As it can be seen, the variation is linear, indicating that the program works well for all codes produced within the input voltage range of the ADC.

4.0 Real time application

Finally, a Real-Time (RT) application was created to allow a windows program to interface with the FPGA. The RT application consists of five sub-programs:

- An application that sends commands to the FPGA program.
- A program that polls the FPGA program outputs at a set interval.
- A wrapper to open the relevant files and reopen them in case of an error.
- A program to convert the ADC words from unsigned 8-bit to a meaningful voltage reading.
- A RT main program that brings the sub-programs together.

These RT main program was then referenced in a Windows program and used to collect data. The data points were collected at intervals of 2 seconds. This was done in parallel with a NI 9239 analogue input module to compare the performance of the two. The voltage input was set to approximately -1V and left for a couple hours. *Figure 7* shows the datapoints for just the CERN ADC and *Figure 8* shows the ones for both the CERN ADC and the NI 9239 for comparison:

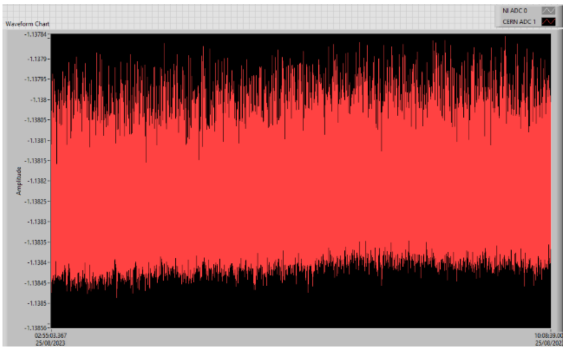

Figure 7- Reading from CERN ADC

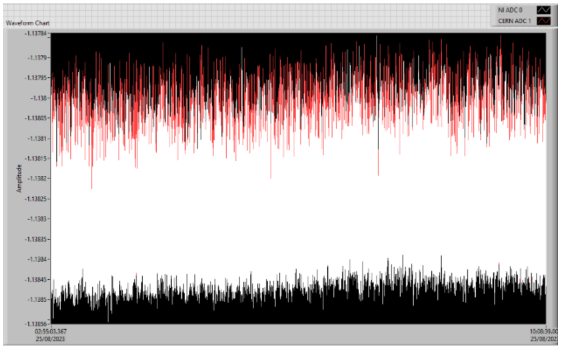

Figure 8 - Reading from CERN ADC (Red) and NI 9239 (White)

5.0 Conclusion

The code developed in this project can act as the basis for future work on the *Alpha-g* magnet control system. If a stable voltage source were used, the same test can be performed as that seen in *Figures 8* and *9* to determine the difference in noise between the CERN ADC and the NI 9239 currently used to sample the DCCTs.

6.0 Appendix

6.1 Testing the code on CompactRio

DC PS voltage (read from multimeter) (V)	Voltage code	Equivalent voltage (V)
10.03525	DF3AC23F	10.02801
9.00768	D57B04F1	9.00075
8.00026	CBEC0D4D	7.99357
7.00085	C26F98EB	6.99401
6.00339	B8F83DA2	5.99655
5.00490	AF7E9A98	4.99815
4.00149	A5F8FECC	3.99482
3.00278	9C7EAE8C	2.99614
2.00573	93086D54	1.99913
1.00905	89929C65	1.00231
0.00724	800F7CB6	0.00000
-1.00356	76795235	-1.00366
-2.00000	6D04A855	-2.00008
-3.00923	6370AD87	-3.00939
-4.00838	59F550D8	-4.00857
-5.00205	50875759	-5.00224
-6.00655	46FF1A63	-6.00672
-7.00463	3D85FAAE	-7.00498
-8.00590	340580C1	-8.00626
-9.00212	2A92FC70	-9.00180
-10.00217	2113B7B7	-10.00258

6.2 ADC characterisation using myRio and Dongle

Multimeter Reading (V)	Mean Calculated voltage A (V)	Mean Calculated voltage B (V)	Hex Code A (0x)	Hex Code B (0x)	Calibrated Voltage A	Calibrated Voltage B	Error A (uV)	Error B (uV)
10.00962	10.00555	10.02942	DEFD774F	DF377B50	10.009569665	10.009570817	-50.33	-49.18
9.76089	9.75594	9.77920	DC9ED1A2	DCD763F7	9.759865642	9.759868595	-1024.36	-1021.40
9.49181	9.49027	9.51291	DA193ED1	DA50441A	9.494139025	9.494138033	2329.02	2328.03
9.24929	9.24549	9.26755	D7C6265B	D7FBC1B7	9.249189801	9.249191712	-100.20	-98.29
8.99557	8.99192	9.01338	D55DF004	D5921366	8.995548659	8.995551635	-21.34	-18.36
8.75492	8.75113	8.77200	D314C624	D3477DC3	8.754687569	8.754680764	-232.43	-239.24
8.50618	8.50277	8.52304	D0B90742	D0EA4A6B	8.506177856	8.506165534	-2.14	-14.47
8.24766	8.24428	8.26394	CE44CEDF	CE74978D	8.247594062	8.247590564	-65.94	-69.44
7.99907	7.99567	8.01473	CBE8A2A5	CC16F799	7.998908540	7.998901121	-161.46	-168.88
7.75294	7.74984	7.76832	C99322BE	C9C013B0	7.752969770	7.752977103	29.77	37.10
7.50385	7.50075	7.51865	C735FFD1	C76176E6	7.503887588	7.503882102	37.59	32.10
7.25466	7.25162	7.26892	C4D86520	C5026F20	7.254612840	7.254615495	-47.16	-44.51
6.99933	6.99591	7.01259	C26B90BD	C2941E40	6.999071126	6.999072138	-258.87	-257.86
6.75306	6.75025	6.76635	C015D8C7	C03CF98E	6.753042222	6.753044202	-17.78	-15.80
6.50381	6.50100	6.51650	BDB81656	BDDDC40C	6.503703561	6.503704235	-106.44	-105.77
6.25080	6.24820	6.26311	BB51A30E	BB75DAC9	6.250787659	6.250790874	-12.34	-9.13
6.00610	6.00353	6.01785	B8FF1E36	B921E92D	6.006075784	6.006076780	-24.22	-23.22
5.75070	5.74820	5.76190	B6928461	B6B3CE90	5.750628218	5.750620460	-71.78	-79.54
5.50149	5.49921	5.51232	B43545AC	B4552272	5.501501368	5.501500873	11.37	10.87
5.25209	5.24992	5.26245	B1D74538	B1F5AB33	5.252063000	5.252055460	-27.00	-34.54
5.00213	5.00002	5.01195	AF780DB6	AF9509C1	5.002124499	5.002131724	-5.50	1.72
4.74906	4.74704	4.75836	AD114666	AD2CC953	4.749073486	4.749078543	13.49	18.54
4.50371	4.50172	4.51245	AABCFB4D	AAD71655	4.503631233	4.503643624	-78.77	-66.38
4.26518	4.26335	4.27352	A879B14A	A892661A	4.265188151	4.265193892	8.15	13.89

3.99902	3.99712	4.00664	A5F29702	A609C056	3.998832111	3.998837913	-187.89	-182.09
3.75073	3.74918	3.75811	A397D228	A3AD8996	3.750724414	3.750727853	-5.59	-2.15
3.50617	3.50459	3.51294	A144D808	A1592687	3.505823967	3.505831782	-346.03	-338.22
3.25202	3.25050	3.25823	9EDC0242	9EEECE4B	3.251926477	3.251919152	-93.52	-100.85
3.00117	2.99953	3.00667	9C798F2E	9C8AEBCC	3.000657355	3.000659174	-512.65	-510.83
2.75039	2.74907	2.75563	9A1947D5	9A293049	2.750281772	2.750282995	-108.23	-107.00
2.50882	2.50763	2.51361	97CE82CE	97DD0969	2.508759605	2.508768524	-60.40	-51.48
2.25487	2.25377	2.25912	9565221F	95722AD4	2.254638765	2.254640395	-231.23	-229.61
2.00010	1.99912	2.00388	92FAA045	93062F4F	2.000052983	2.000055216	-47.02	-44.78
1.75159	1.75069	1.75485	909EE266	90A8FF29	1.751544897	1.751545112	-45.10	-44.89
1.50242	1.50142	1.50498	8E410DDE	8E49B726	1.502177150	1.502175464	-242.85	-244.54
1.25171	1.25100	1.25397	8BE06393	8BE79AC7	1.251642476	1.251643919	-67.52	-66.08
1.00758	1.00698	1.00936	898F6C8D	899539A8	1.007570242	1.007572222	-9.76	-7.78
0.74426	0.74269	0.74544	870F7E4D	8713C31E	0.744166292	0.744166426	-93.71	-93.57
0.50323	0.50269	0.50387	84C593FE	84C87813	0.502995795	0.503004574	-234.21	-225.43
0.24897	0.24866	0.24923	825C5707	825DBB84	0.248932387	0.248931019	-37.61	-38.98
0.00744	0.00597	0.00597	800E7E4E	800E747D	0.006143712	0.006133076	-1296.29	-1306.92
-0.00691	-0.00797	-0.00801	7FFC8FDB	7FEC77DD	-0.001237107	-0.007823027	5672.89	-913.03
-0.25372	-0.25434	-0.25497	7D95F152	7D95F152	-0.254222555	-0.253597281	-502.56	122.72
-0.49776	-0.49777	-0.49898	7B4643EB	7B435132	-0.497764803	-0.497770039	-4.80	-10.04
-0.75769	-0.75773	-0.75957	78CE85A9	78CA1140	-0.757798683	-0.757803201	-108.68	-113.20
-1.00731	-1.00718	-1.00961	7670285C	766A41F0	-1.007386337	-1.007389876	-76.34	-79.88
-1.25260	-1.25224	-1.25523	741C8BE4	74153B4B	-1.252547810	-1.252548344	52.19	51.66
-1.50140	-1.50109	-1.50469	71BFBE26	71B6FF04	-1.501493029	-1.501488537	-93.03	-88.54
-1.75055	-1.75006	-1.75427	6F62A7C0	6F586A93	-1.750555068	-1.750570147	-5.07	-20.15
-2.00540	-2.00555	-2.01036	6CF5C2AA	6CEA0DCE	-2.006123632	-2.006132584	-723.63	-732.58
-2.25494	-2.25395	-2.25934	6A9A06F3	6A8CE2E1	-2.254628251	-2.254634311	311.75	305.69
-2.49752	-2.49760	-2.50369	6849E8E9	683B5DA6	-2.498351604	-2.498353296	-831.60	-833.30
-2.75327	-2.75244	-2.75904	65DE6502	65CE63AB	-2.753352296	-2.753346638	-82.30	-76.64
-3.00299	-3.00206	-3.00925	637FD19C	636E5229	-3.003026932	-3.003039493	-36.93	-49.49

-3.24884	-3.24812	-3.25591	6129C5A7	6116E448	-3.249190890	-3.249184818	-350.89	-344.82
-3.50422	-3.50304	-3.51143	5EBE42E3	5EA9DD57	-3.504189755	-3.504198950	30.24	21.05
-3.75087	-3.74961	-3.75859	5C6736EB	5C5156A6	-3.750765346	-3.750794708	104.65	75.29
-4.00076	-3.99928	-4.00887	5A0821CA	59F0D691	-4.000648571	-4.000664928	111.43	95.07
-4.25032	-4.24893	-4.25909	57A95B59	5790A6AD	-4.250405277	-4.250406517	-85.28	-86.52
-4.49868	-4.49722	-4.50799	554E05FF	5533DBC8	-4.498745310	-4.498754207	-65.31	-74.21
-4.75409	-4.75310	-4.76447	52E00B16	52C46758	-4.754760577	-4.754765240	-670.58	-675.24
-4.99890	-4.99703	-5.00899	508F23F0	50721912	-4.998807286	-4.998806705	92.71	93.30
-5.25049	-5.24861	-5.26118	4E2BD481	4E0D4E69	-5.250430710	-5.250439081	59.29	50.92
-5.51029	-5.49940	-5.51257	4BCA052E	4BAA1162	-5.501436538	-5.501433641	8853.46	8856.36
-5.75043	-5.74848	-5.76223	496CDEB3	494B734A	-5.750524435	-5.750530734	-94.44	-100.73
-6.00037	-5.99844	-6.01279	47DD5A66	46EA790E	-5.914970850	-6.000596890	85399.15	-226.89
-6.24943	-6.24744	-6.26238	44B04D27	448BFC6F	-6.249633733	-6.249640292	-203.73	-210.29
-6.49850	-6.49608	-6.51161	42540817	422E46EB	-6.498359189	-6.498364322	140.81	135.68
-6.74966	-6.74715	-6.76328	3FF1B41C	3FCA7972	-6.749578311	-6.749590577	81.69	69.42
-7.00435	-7.00185	-7.01861	3D869751	3D5DE6B3	-7.004413217	-7.004418328	-63.22	-68.33
-7.25380	-7.24985	-7.26719	3B2BF04F	3B01C344	-7.252472929	-7.252497405	1327.07	1302.60
-7.50733	-7.50461	-7.52255	38C0CD72	389532E8	-7.507317596	-7.507321328	12.40	8.67
-7.75158	7.74876	-7.76730	366F5593	364246B9	-7.751596999	-7.751616086	-17.00	-36.09
-8.01121	-8.00827	-8.02743	33F8E57F	33CA5153	-8.011093563	-8.011119040	116.44	90.96
-8.25024	-8.24730	-8.26704	31B3C12A	3183D089	-8.250299288	-8.250313939	-59.29	-73.94
-8.50548	-8.50240	-8.52274	2F47BAD7	2F164E01	-8.505509681	-8.505526314	-29.68	-46.31
-8.75198	-8.74890	-8.76981	2CF0AC95	2CBDD88A	-8.752088952	-8.752094440	-108.95	-114.44
-8.99998	-8.99684	-9.01836	2A961750	2A61D505	-9.000120143	-9.000122325	-140.14	-142.33
-9.25003	-9.24682	-9.26893	28368A46	2800C601	-9.250196168	-9.250221815	-166.17	-191.82
-9.50476	-9.50100	-9.52400	25CCC6FE	259592F9	-9.504475538	-9.504485403	284.46	274.60
-9.74999	-9.74699	-9.77030	2376FB5B	233E5502	-9.750536078	-9.750553873	-546.08	-563.87
-10.00099	-9.99722	-10.02112	2116C9D9	20DCAC22	-10.000876546	-10.000900158	113.45	89.84