double	ref_freq	250.10	Hz	
uint16_t	DAQ_period	50	us	*Must be int because it is the period for the interrupt service routine (ISR)*
		5.00E-05	s	
double	DAQ_rate	20000.00	Hz	
uint16_t	N_LUT	20000	samples	*If N_LUT is power of 2 then we can use bitwise `&` to perform a fast integer modulo*
uint16_t	BLOCK_SIZE	2000	samples	∟ Cancel that. This will lead to beating of `ref_freq`. Better set value equal to `DAQ_rate`.
double	BLOCK_RATE	10.00	blocks/s	
double	LUT_idx / DAQ_iter	250.10		*Fractional, non-ideal. Must be rounded to integer to be able to use in MULMOD.*
uint16_t	**ideal** LUT_idx / DAQ_iter	250		*Rounded to nearest integer.*
double	**ideal** ref_freq	250.00	Hz	*Hence, we must adjust ref_freq to reflect optimal value.*

MULMOD algorithm on integers --> super fast, no integer overflow
https://www.geeksforgeeks.org/how-to-avoid-overflow-in-modular-multiplication/
https://stackoverflow.com/questions/12168348/ways-to-do-modulo-multiplication-with-primitive-types/12171020

		non-ideal ref_freq					mod()	bitwise &	
DAQ_iter	t	x	REF_X	non ideal	non ideal	ideal	ideal	ideal	
	[ms]	= ref_freq*t	= sin(2*pi*x)	LUT_idx	modulo	LUT_idx	modulo	modulo 2^N	
##########	1.07E+08	2.69E+07	#NUM!	537086410414.70	10414.70	###########	1750	16406	<-- Large iter value
0	0.000	0.0000	0.000	0.00	0.00	0	0	0	<-- Small iter values, starting at 0
1	0.050	0.0125	0.078	250.10	250.10	250	250	26	
2	0.100	0.0250	0.156	500.20	500.20	500	500	20	
3	0.150	0.0375	0.234	750.30	750.30	750	750	526	
4	0.200	0.0500	0.309	1000.40	1000.40	1000	1000	520	
5	0.250	0.0625	0.383	1250.50	1250.50	1250	1250	1026	
6	0.300	0.0750	0.454	1500.60	1500.60	1500	1500	1052	
7	0.350	0.0875	0.523	1750.70	1750.70	1750	1750	1558	
8	0.400	0.1000	0.588	2000.80	2000.80	2000	2000	1552	
9	0.450	0.1125	0.650	2250.90	2250.90	2250	2250	2058	
10	0.500	0.1251	0.707	2501.00	2501.00	2500	2500	2052	
11	0.550	0.1376	0.761	2751.10	2751.10	2750	2750	2590	
12	0.600	0.1501	0.809	3001.20	3001.20	3000	3000	2584	
13	0.650	0.1626	0.853	3251.30	3251.30	3250	3250	3090	
14	0.700	0.1751	0.891	3501.40	3501.40	3500	3500	3084	
15	0.750	0.1876	0.924	3751.50	3751.50	3750	3750	3590	
16	0.800	0.2001	0.951	4001.60	4001.60	4000	4000	3584	
17	0.850	0.2126	0.972	4251.70	4251.70	4250	4250	26	
18	0.900	0.2251	0.988	4501.80	4501.80	4500	4500	20	
19	0.950	0.2376	0.997	4751.90	4751.90	4750	4750	526	
20	1.000	0.2501	1.000	5002.00	5002.00	5000	5000	520	
21	1.050	0.2626	0.997	5252.10	5252.10	5250	5250	1026	
22	1.100	0.2751	0.988	5502.20	5502.20	5500	5500	1052	
23	1.150	0.2876	0.972	5752.30	5752.30	5750	5750	1558	
24	1.200	0.3001	0.951	6002.40	6002.40	6000	6000	1552	
25	1.250	0.3126	0.924	6252.50	6252.50	6250	6250	2058	
26	1.300	0.3251	0.891	6502.60	6502.60	6500	6500	2052	
27	1.350	0.3376	0.852	6752.70	6752.70	6750	6750	2590	
28	1.400	0.3501	0.808	7002.80	7002.80	7000	7000	2584	
29	1.450	0.3626	0.760	7252.90	7252.90	7250	7250	3090	
30	1.500	0.3752	0.706	7503.00	7503.00	7500	7500	3084	
31	1.550	0.3877	0.649	7753.10	7753.10	7750	7750	3590	
32	1.600	0.4002	0.587	8003.20	8003.20	8000	8000	3584	
33	1.650	0.4127	0.522	8253.30	8253.30	8250	8250	26	
34	1.700	0.4252	0.453	8503.40	8503.40	8500	8500	20	
35	1.750	0.4377	0.382	8753.50	8753.50	8750	8750	526	
36	1.800	0.4502	0.308	9003.60	9003.60	9000	9000	520	
37	1.850	0.4627	0.232	9253.70	9253.70	9250	9250	1026	
38	1.900	0.4752	0.155	9503.80	9503.80	9500	9500	1052	
39	1.950	0.4877	0.077	9753.90	9753.90	9750	9750	1558	
40	2.000	0.5002	-0.001	10004.00	10004.00	10000	10000	1552	
41	2.050	0.5127	-0.080	10254.10	10254.10	10250	10250	2058	
42	2.100	0.5252	-0.158	10504.20	10504.20	10500	10500	2052	
43	2.150	0.5377	-0.235	10754.30	10754.30	10750	10750	2078	
44	2.200	0.5502	-0.310	11004.40	11004.40	11000	11000	2584	
45	2.250	0.5627	-0.384	11254.50	11254.50	11250	11250	2578	
46	2.300	0.5752	-0.455	11504.60	11504.60	11500	11500	3084	
47	2.350	0.5877	-0.524	11754.70	11754.70	11750	11750	3078	