	A	B	C	D	E	F
1	short description	arbitrary item id #	seen in version #	Long description **(changes since last version are in bold font)**	Status	**see also**
2	“Inititalizing”	1	0.20	cosmetic change: Initializing	Source code modified, awaiting compile/ verify/ upload	**na**
3	UnusedMem ini file entry missing message	2	0.20	Absent ini file UnusedMem directive generates a parse failure message. Literal 100 instead of UNMEM_DFLT used in some code	Source code modified, awaiting compile/ verify/ upload	**na**
4	ini file edits	3	0.20	Added UnusedMem section to cudapm1.ini, changed CUDALucas references to CUDAPm1 throughout, **changed default device number from one to zero to fit common single-gpu systems, and zero-based numbering comment added**	Source code modified, awaiting upload	**na**
5	every launch after savefile directory is created, there's a message it could not be created	4	0.20	Reversed case of when savefile directory creation message is printed, modified message accordingly	Source code modified, awaiting compile/ verify/ upload	**na**
6	Literal 1,000,000,000 used, MAX_B2?	5	0.20	Modified MAX B2 message, added define of MAX_B2	Source code modified, awaiting compile/ verify/ upload	**na**
7	no entry message re threads file	6	0.20	modified the no entry message for threads file	Source code modified, awaiting compile/ verify/ upload	**na**
8	misc code edits	7	0.20	modified some comments and formatting; version increment to 0.21	Source code modified, awaiting compile/ verify/ upload	**na**
9	error messages splitting between command console and output redirection target	8	0.20	fprintf(stderr for first part of several messages, printf(for later part was resulting in splitting messages to two destinations if output redirection of stdout occurred	Source code modified, awaiting compile/ verify/ upload	**na**
10	-r option in CUDAPm1 listed in help message but not functional. This self test mode is not available	9	0.20	-r presence in the CUDAPm1 help message output seems to be a holdover from its CUDALucas ancestry. Program's help output indicates "exec residue test." Specifying -r on the command line does not result in any residue check tests running in CUDAPm1; it goes straight to continuation of work present in the worktodo file. If I read the source code correctly, the residue check function did not get implemented for CUDAPm1. Implementation would ideally be driven by an external plain text table, so that adding new cases would not always require code modification and recompilation. A workaround for now is to run one or more P-1 test on one or more exponents that will use the same fft length(s) as untested exponents planned to be run, if suitable exponent and factor combinations can be found. **Probably should comment out -r option output in help message output until implemented. Preferred implementation would be -r low high; read the external table, and run residue tests for the subset of entries low<=entry<=high. Specifying -r n would run all entries <=n; specifying -r with no value following would run all residue tests.**	identified; no code or volunteer at this time	**CUDALucas 28**

	A	B	C	D	E	F
11	Some CUDALucas bugs possibly shared	10	expected in all	since CUDAPm1 was derived from CUDALucas code, CUDAPm1 may share some of CUDALucas' issues. The device renumbering occurrence is **generic and more deeply rooted and** highly likely	suspected	**various**
12	documentation	11	0.20	needs a readme, wiki page or something; draft revised ini and readme exist but not posted	pending	**na**
13	are the various executables for linux and windows current? There were various release numbers for V0.20 including r50 and r52	12	0.20	File dates seem to indicate that some executables are not current with the last previous source changes. See http://www.mersenneforum.org/showpost.php?p=462600&postcount=503	pending	**na**
14	**No date/time stamps; gcd time**	**13**	**0.20**	**There are various elapsed time estimates, and total time estimates, but no actual date/time stamps in output. It's unclear if gcd time is included in stage estimated total time, or how long the gcds take**	**identified; no code or volunteer at this time**	**CUDALucas 33**
15	**logging of stdout, stderr to file**	**14**	**0.20**	**planned feature addition, an -o flag to control output to screen and log. Stdout can be duplicated to log, switched to log, suppressed (default is stdout to console, status quo) Stderr can be duplicated to log but not suppressed or switched. Workaround is redirection, or tee (native to linux, part of Windows Powershell)**	**in development, beginnings of partial code draft exists. Needs ini file and command line extension, global rewrite of printf and fprintf calls, then compile/debug/test**	**CUDALucas 19**

	A	B	C	D	E	F
16	Repeating-zero residue issue	15	0.20	unknown program error or hardware error initiates repeating 0x00 residues. Program carries this to completion. Of four occurrences known, 3 occurred in stage 1 and one is uncertain. The end result is a normal looking entry in the results file no factor found. Observed to start midstream in a run, following a slightly larger exponent that completed successfully. CUDA reports 1429M of 1536M GPU memory free. Using threads: norm1 128, mult 64, norm2 32. Using up to 1232M GPU memory. Selected B1=2145000, B2=13942500, 4.6% chance of finding a factor Using B1 = 2145000 from savefile. Continuing stage 1 from a partial result of M249500221 fft length = 14336K, iteration = 2050001 Iteration 2100000 M249500221, 0xb81566f6b7207983, n = 14336K, CUDAPm1 v0.20 err = 0.14063 (22:30 real, 26.9995 ms/iter, ETA 7:27:50) Iteration 2150000 M249500221, 0x0000000000000000, n = 14336K, CUDAPm1 v0.20 err = 0.13281 (22:22 real, 26.8399 ms/iter, ETA 7:02:49) Iteration 2200000 M249500221, 0x0000000000000000, n = 14336K, CUDAPm1 v0.20 err = 0.11289 (22:21 real, 26.8282 ms/iter, ETA 6:40:17) Iteration 2250000 M249500221, 0x0000000000000000, n = 14336K, CUDAPm1 v0.20 err = 0.09596 (22:23 real, 26.8743 ms/iter, ETA 6:18:34) SIGINT caught, writing checkpoint. … continued into stage 2... Processing 180 - 180 of 480 relative primes. Inititalizing pass... done. transforms: 251, err = 0.00000, (3.60 real, 14.3510 ms/tran, ETA 16:21:38) SIGINT caught, writing checkpoint. Transforms: 8760 M249500221, 0x0000000000000000, n = 14336K, CUDAPm1 v0.20 err = 0.00000 (2:03 real, 14.0203 ms/tran, ETA 16:19:13) Proposed treatment is to detect at or just prior to checkpoint or save file write, revert to most recent believed-good save file or checkpoint, retry a limited number of times.	borrow detection/halt code from recent cudalucas?	CUDALucas 4
17	invalid threadbench timings	16	0.20	Certain threads counts on certain gpu models, or CUDA level & fft length & gpu combinations result in wrong execution and anomalously fast benchmarks that then cause the malfunctioning values to be selected for future use. the only lower bound currently on iteration timings in case of a malfunction is 0.0f. A lower bound computed as a fraction of the average of all timings above a lower fft length cutoff can be used to validate execution of individual thread combinations of a given fft length above the cutoff length.	In early development, a first code draft exists	part of CUDALucas 2
18	invalid fftbench timings	17	0.20	Certain CUDA level & fft length & gpu combinations result in wrong execution and anomalously fast benchmarks that then cause the malfunctioning values to be selected for future use. the only lower bound currently on iteration timings in case of a malfunction is 0.0f. A lower bound computed as a fraction of the average of (each timing above a lower fft length cutoff divided by its fft length) can be used to validate execution of a given fft length above the lower fft length cutoff. Invalidated lengths can be suppressed from the output fft file and prevented from removing other shorter fft lengths from consideration.	conceptual	part of CUDALucas 2

	A	B	C	D	E	F
19	**autoselect picks b1=0 b2=0 which produces a run with repeating residues 0x01**	**18**	**0.20**	**some exponents lead to autoselection of B1=0, B2=0. Two cases have been identified; smallish mersenne prime exponents, and composite exponents. The program computes stage one with a repeating residue value of 0x01. It does not warn or error. The end result is a normal looking entry in the results file no factor found. A preferable implementation if some exponents are being excluded intentionally from the full computation is to produce a message about their exclusion and why, and skip the computation. CUDA reports 1434M of 1536M GPU memory free. Index 2 No entry for fft = 8k found. Using default thread sizes. For optimal thread selection, please run ./CUDAPm1 -cufftbench 8 8 r for some small r, 0 < r < 6 e.g. Using threads: norm1 512, mult 128, norm2 128. Using up to 1334M GPU memory. Selected B1=0, B2=0, 0% chance of finding a factor Starting stage 1 P-1, M86243, B1 = 0, B2 = 0, fft length = 8K Doing 18 iterations Running careful round off test for 1000 iterations. If average error > 0.25, the test will restart with a longer FFT. Iteration 1000, average error = 0.00000 <= 0.25 (max error = 0.00000), continuing test. M86243, 0x0000000000000001, n = 8K, CUDAPm1 v0.20 Stage 1 complete, estimated total time = 0:00 Starting stage 1 gcd. M86243 Stage 1 found no factor (P-1, B1=0, B2=0, e=0, n=8K CUDAPm1 v0.20)**	**conceptual**	**na**

	A	B	C	D	E	F
20	**fft length table is from gtx570**	**19**	**0.20**	**other tables may be suitable for other gpu models. Feature request, ability to load the table from a file. File could be switched out for differing gpu models to optimize for each.** **int init_ffts()** **{** **… #define COUNT 160** **int default_mult[COUNT] = { //this batch from GTX570 timings** **2, 8, 10, 14, 16, 18, 20, 32, 36, 42,** **48, 50, 56, 60, 64, 70, 80, 84, 96, 112,** **120, 126, 128, 144, 160, 162, 168, 180, 192, 224,** **256, 288, 320, 324, 336, 360, 384, 392, 400, 448,** **512, 576, 640, 648, 672, 720, 768, 784, 800, 864,** **896, 900, 1024, 1152, 1176, 1280, 1296, 1344, 1440, 1568,** **1600, 1728, 1792, 2048, 2160, 2304, 2352, 2592, 2688, 2880,** **3024, 3136, 3200, 3584, 3600, 4096, 4320, 4608, 4704, 5120,** **5184, 5600, 5760, 6048, 6144, 6272, 6400, 6480, 7168, 7200,** **7776, 8064, 8192, 8640, 9216, 9408, 10240, 10368, 10584, 10800,** **11200, 11520, 12096, 12288, 12544, 12960, 13824, 14336, 14400, 16384,** **17496, 18144, 19208, 19600, 20000, 20250, 21952, 23328, 23814, 24300,** **24500, 25088, 25600, 26244, 27000, 27216, 28000, 28672, 31104, 31250,** **32000, 32400, 32768, 33614, 34992, 36000, 36288, 38416, 39200, 39366,** **40500, 41472, 42336, 43200, 43904, 47628, 49000, 50000, 50176, 51200,** **52488, 54432, 55296, 56000, 57344, 60750, 62500, 64000, 64800, 65536 };**		**na**
21	**B1 extension**	**20**	**0.20**	**there's a branch to "stage 3" which is a placeholder for where B1 extension code would go**		**na**
22	**"UID:username/systemid-gpuid, "**	**21**	**0.20**	**Support prefacing results lines with UID: <username>(/<compname>), e.g.: UID: JamesHeinrich/GTX1080, <result line here as normal> see http://www.mersenneforum.org/showpost.php?p=461411&postcount=3**		**CUDALucas 32**

	A	B	C	D	E	F
23	savefile naming with previous save's residue value in name	22	0.20	First savefile of a run is S81328579.50001..txt, writing checkpoints at 50000 intervals. Second is s81328579.100001.9519ada275083227.txt, third is s81328579.150001.3b78a68dbb318e75.txt, fourth s81328579.200001.eca5c42ceeadae9e.txt etc console output corresponding to these is Iteration 50000 M81328579, 0x9519ada275083227, n = 4608K, CUDAPm1 v0.20 err = 0.09863 (8:25 real, 10.1038 ms/iter, ETA 2:48:57) Iteration 100000 M81328579, 0x3b78a68dbb318e75, n = 4608K, CUDAPm1 v0.20 err = 0.09814 (8:19 real, 9.9766 ms/iter, ETA 2:38:31) Iteration 150000 M81328579, 0xeca5c42ceeadae9e, n = 4608K, CUDAPm1 v0.20 err = 0.10352 (8:19 real, 9.9774 ms/iter, ETA 2:30:13) Iteration 200000 M81328579, 0x04bb5c6a41d55bcd, n = 4608K, CUDAPm1 v0.20 err = 0.10156 (8:15 real, 9.8952 ms/iter, ETA 2:20:44) A comparison shows the residue is missing on the 50000 and its residue is used in the name of the 1000000 instead, and following ones follow this off-by-one pattern. this could conceivably be intended behavior, tagging the successive savefile with the residue from which it was begun, however, the first produced of each run is missing the residue tag. It would seem more clear to tag with the residue of the file contents being saved, simplifying the logic of what savefile to attempt a resume from.		na
24	echo the worktodo line or command line inputs to console/log	23	0.20	echo the worktodo line content or content of command line parameters concerning the exponent to console and if applicable to log, so that after the console session scrolls or the worktodo line is removed, there is still a record of the parameters used for the exponent, such as B1, B2 values, # of LL tests that would be saved by finding a factor, etc.	conceptual	CUDALucas 34
25	retry rather than halt on round off >0.4	24	0.20	current behavior is if round off error observed is >0.4, the program prints the error and halts: err = 0.5 >= 0.40, quitting. Frequently a restart produces a lower, acceptable error level and computation can continue. Unattended sessions that halt cause loss of throughput. Batch wrappers can reduce that but it makes screen and log clutter and overhead for restart. A resetting retry counter with a small limit, say 3, perhaps with a brief time delay of several seconds between retries is proposed.	conceptual	CUDALucas 35
26	simplify resume from save file	25	0.20	a command line option to resume from a specific interim save file: check for seemingly valid residue in filename, rename suspected bad checkpoint files, copy save file to the appropriate filename in working directory, resume	conceptual	CUDALucas 36
27	device confirmation	26	0.2	In multiple-GPU systems, NVIDIA driver timeout or thermal limits or a combination may cause a device to disappear from the device count, even though Windows Device Manager shows it, and GPU-Z lists it and can find its parameters but not display its sensor readings. Batch wrapper may restart a run on a different device than intended as a result. This may affect execution timing of two sessions sharing one gpu, or may cause a restarted session to fail if it requires more resources than available on a dissimilar card or a device number higher than the reduced Windows count allows. That generates error message device_number >= device_count ... exiting (This is probably a driver problem) Confirming unique device characteristics could allow greater confidence in execution.	conceptual/ flashjh has done some development in CUDALucas	CUDALucas 8

	A	B	C	D	E	F
28	feature suggestion: run time estimation for entire worktodo file	27	0.20	It would be useful to have an option in CUDALucas to read the entire worktodo file, and compute and display time span and completion date for each line of the worktodo file, at the outset of a run following the usual header and optional info section. Option could be -w or -work.		CUDALucas 9
29	FFT benchmark and threads benchmark are believed affected by variable clock rate of modern GPUs	28	0.20	monitoring gpu core clock with GPU-Z 1.18 shows clock rate changes in the early phase of a run, approx 1354 to 1860 Mhz. Nearly all the run occurs at maximum clockrate. A single intermediate clock value 1468 Mhz is logged in the cudalucas output. It is likely to affect the accuracy of the initial fft lengths' benchmarking. Addition of a “warmup” phase that loads the GPU and brings clock up to speed before benchmarking begins is proposed. As a workaround, fft and threads benchmarking could be submitted in a batch file immediately after test of a small exponent, or benchmarking at smaller fft lengths than needed can be performed and the affected lengths not needed for current GIMPS wavefront edited out of the benchmark files.	conceptual	CUDALucas 16
30	Addition of header in fft and threads files, of fairly complete description of conditions in which file was generated	29	0.20	Fft file has no header information subset. Including the following when it is generated would document how it was created and what it's valid for, and conversely, what it may not be valid for. System identifier, Operating system, CUDALucas version, CUDA level, 32 or 64 bit flavor, NVIDIA driver version, GPU model. Currently only the GPU model is documented, and only in the file name for the threads file which contains no header. Program could be enhanced to check that the fft or threads file being used was appropriate for the version running, and warn if not.	conceptual	CUDALucas 20
31	Automatic setup and verification	30		Add a command line switch to automate initial setup and burn-in (-setup)	conceptual	CUDALucas 22
32	PrimeNet interface	31		automatic PrimeNet interaction	conceptual	CUDALucas 23
33	FFT file run time cap estimates	32		Add a runtime estimate column for maximum exponent per fft length to <gpu> fft.txt. Format in clock and calendar units; hours, days, weeks, months or years. Adjust file reading to accommodate if necessary.	conceptual	CUDALucas 24
34	GPUs with vram <=1.5GB (2GB?) have issues with fft benchmark up to legal program inputs	33	0.20	Fft benchmark on Gtx480 iuns until it hits a limit and fails to output an fft file, after 32400k, possibly varying somewhat with CUDA level as it did with CUDALucas. Program does not warn of an issue Successive runs with higher start points may be usable to achieve a complete fft benchmarking; be sure to save first fft file as a separate name to avoid data loss, then assemble the piecemeal results. Modify program to compute whether memory will be adequate for the requested run, suggest or make modifications to calling parameters to allow run to completion. Workaround is to run twice, scaling back upper limit until an FFT file is produced.	conceptual	CUDALucas 26
35	GPUs with vram <=1.5GB (2GB?) may have issues with threads benchmark up to legal program inputs	34		In CUDALucas, Fftlength upper limits relating to limited memory size for threadbench are lower than for fftbench for the same GPU/memory size. Program does not warn of an issue. Threads files do not get overwritten, but added to. Test for similar behavior at probably lower threshold in CUDAPm1	suspected; test	CUDALucas 27

	A	B	C	D	E	F
36	**hardening against user error**	**35**	**0.20**	**detect and warn on illegal user input and substitute sensible default or exit; encode parameters for which fft file and threads file was run into the file itself, then check at use for a match with the parameters of use (CUDA level, software version, etc.)**	**conceptual**	**CUDALucas 30**
37	formatting output	**36**		**add vertical white space to emphasize stage 1 err >0.4 quitting message. Check for other**	**preliminary conceptual**	**na**
38	**Halt without warning or error message**	**37**	**0.20**	**may be a program crash, OS or power issue. Not due to a sigint or completion of P-1 stage since it occurs without a sigint message during stage1 or stage2. cause(s) unknown (does it include the NVIDIA driver timeout issue?)**		**CUDALucas 37**
39	**exhaustive error checking version**	**38**		**for debug purposes, high reliability runs, or making use of lower reliability software, trade performance for error detection and some handling, by checking return values of all function calls (cpu or gpu). optionally dump inputs and outputs of the function to file in case of a nonrecoverable error for later analysis. In some cases gather additional information on machine state specific to the error occurrence**	**conceptual**	**CUDALucas 38**
40	**badmem handler**	**39**		**persistent gpu memory error map and lockout. The idea is to map and allocate the bad memory areas, so they can't be allocated to math we want right, which then gets other more reliable portions of memory allocated. Whether this can be made to work on commodity gpus requires some study. Requires reliable device confirmation/device number stability. This technique is used in linux hobbyist systems, and in NVIDIA K20X based supercomputers (see http://www.mersenneforum.org/showthread.php?t=22471&page=6 http://rick.vanrein.org/linux/badram/download.html and http://on-demand.gputechconf.com/gtc/2015/presentation/S5566-James-Rogers.pdf)**	**conceptual**	**CUDALucas 39**
41	**augment -info output**	**40**		**include OS, OS-version, OS-bitness, app-bitness, app CUDA level compiled for, CUDA level available from driver, device number, identifying properties such as BIOS version string, PCIbus# and PCIdevice#**	**some development done by flashjh for CUDALucas 2.06beta**	**26; see also CUDALucas 8**
42	**error and retry tabulation extension**	**41**		**count recovered errors and retries by type per exponent and summarize in console output, log and result line (see http://www.mersenneforum.org/showpost.php?p=465207&postcount=58)**	**conceptual**	**CUDALucas 41**
43						
44				**Above here have been screened for applicability to CUDALucas also**		
45						
46				**Note, behavior analogous to CUDALucas issues has not in all cases been tested for yet**		
47						
48	questions			do CUDAPM1, CUDALucas etc check for or enable or disable ECC on gpus? Per flashjh CUDALucas checks enable state but does not change it; see NVIDIA Control Panel for enabling /disabling for GPUs supporting ECC; there's a performance hit.		
49				does Jacobi check apply to CUDAPm1 calculations, stage 1, stage 2, both?		
50				More reliable error detection at 0.1% overhead; is it applicable to the P-1 calculations? http://www.mersenneforum.org/showpost.php?p=465431&postcount=88		