

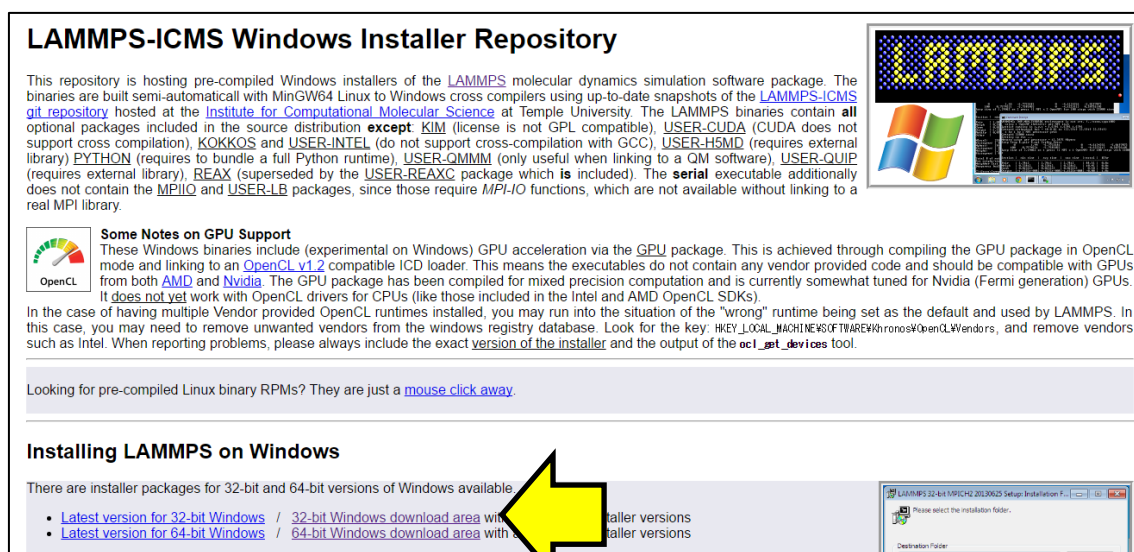
# Windows 版 LAMMPS インストールマニュアル

2019/6/12

## 1. LAMMPS の入手

- ① <http://rpm.lammps.org/windows.html> にアクセスする。稀にサーバのメンテナンス等でつながらないことがあるが、概ね数日～1週間程度で復旧する。

インストール先のマシンの OS に応じて[32-bit Windows download area]もしくは[64-bit Windows download area]をクリックする。



**LAMMPS-ICMS Windows Installer Repository**

This repository is hosting pre-compiled Windows installers of the LAMMPS molecular dynamics simulation software package. The binaries are built semi-automatically with MinGW64 Linux to Windows cross compilers using up-to-date snapshots of the LAMMPS-ICMS git repository hosted at the Institute for Computational Molecular Science at Temple University. The LAMMPS binaries contain all optional packages included in the source distribution **except** KIM (license is not GPL compatible), USER-CUDA (CUDA does not support cross compilation), KOKKOS and USER-INTEL (do not support cross-compilation with GCC), USER-H5MD (requires external library) PYTHON (requires to bundle a full Python runtime), USER-QMMM (only useful when linking to a QM software), USER-QUIP (requires external library), REAX (superseded by the USER-REAXC package which is included). The **serial** executable additionally does not contain the MPIQ and USER-LB packages, since those require MPI-IO functions, which are not available without linking to a real MPI library.

**Some Notes on GPU Support**

These Windows binaries include (experimental on Windows) GPU acceleration via the GPU package. This is achieved through compiling the GPU package in OpenCL mode and linking to an OpenCL v1.2 compatible ICD loader. This means the executables do not contain any vendor provided code and should be compatible with GPUs from both AMD and Nvidia. The GPU package has been compiled for mixed precision computation and is currently somewhat tuned for Nvidia (Fermi generation) GPUs. It **does not yet** work with OpenCL drivers for CPUs (like those included in the Intel and AMD OpenCL SDKs).

In the case of having multiple Vendor provided OpenCL runtimes installed, you may run into the situation of the "wrong" runtime being set as the default and used by LAMMPS. In this case, you may need to remove unwanted vendors from the windows registry database. Look for the key: HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\OpenCL\Vendors, and remove vendors such as Intel. When reporting problems, please always include the exact version of the installer and the output of the ocl\_get\_devices tool.

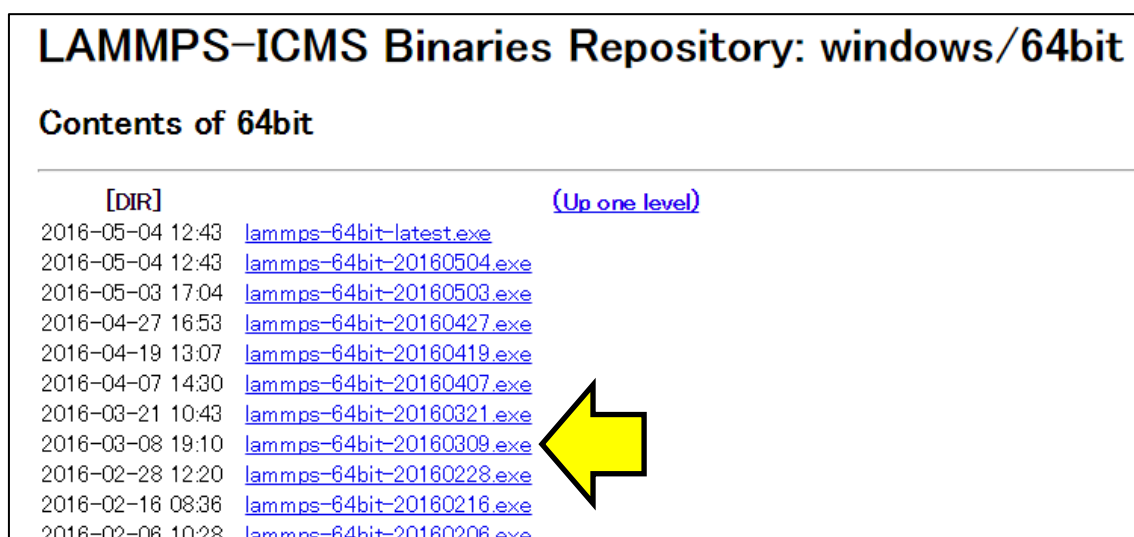
Looking for pre-compiled Linux binary RPMs? They are just a [mouse click away](#).

**Installing LAMMPS on Windows**

There are installer packages for 32-bit and 64-bit versions of Windows available.

- [Latest version for 32-bit Windows](#) / [32-bit Windows download area with all installer versions](#)
- [Latest version for 64-bit Windows](#) / [64-bit Windows download area with all installer versions](#)

- ② 64bit の場合は lammps-64bit-20160309.exe、32bit の場合は lammps-32bit-20160309.exe を保存する。



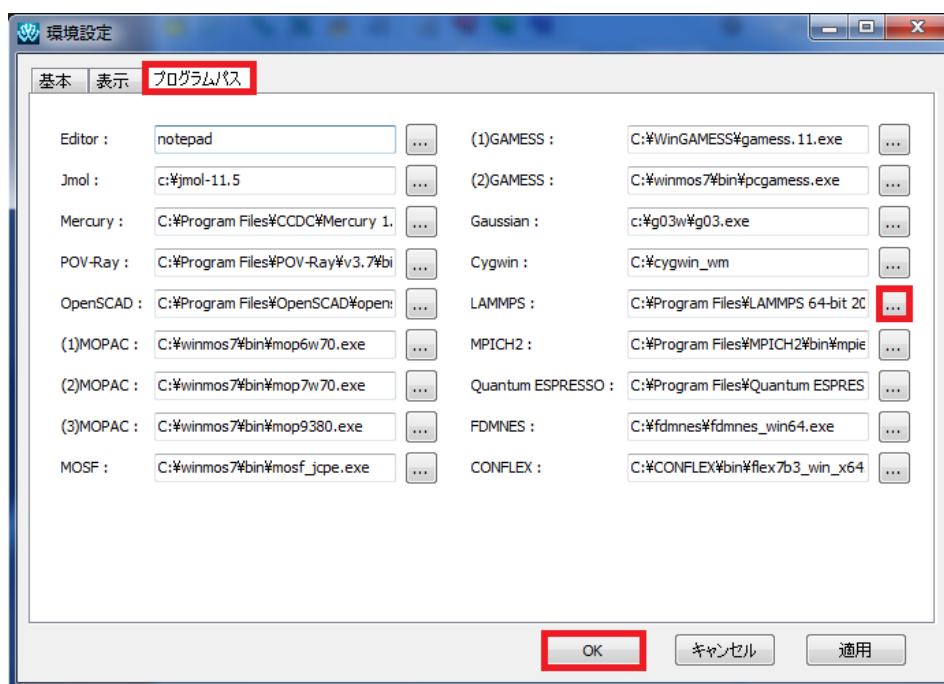
**LAMMPS-ICMS Binaries Repository: windows/64bit**

**Contents of 64bit**

[DIR]	(Up one level)
2016-05-04 12:43	<a href="#">lammps-64bit-latest.exe</a>
2016-05-04 12:43	<a href="#">lammps-64bit-20160504.exe</a>
2016-05-03 17:04	<a href="#">lammps-64bit-20160503.exe</a>
2016-04-27 16:53	<a href="#">lammps-64bit-20160427.exe</a>
2016-04-19 13:07	<a href="#">lammps-64bit-20160419.exe</a>
2016-04-07 14:30	<a href="#">lammps-64bit-20160407.exe</a>
2016-03-21 10:43	<a href="#">lammps-64bit-20160321.exe</a>
2016-03-08 19:10	<a href="#">lammps-64bit-20160309.exe</a>
2016-02-28 12:20	<a href="#">lammps-64bit-20160228.exe</a>
2016-02-16 08:36	<a href="#">lammps-64bit-20160216.exe</a>
2016-02-06 10:28	<a href="#">lammps-64bit-20160206.exe</a>

- ③ 保存した exe ファイルをダブルクリックし指示に従う。
- ④ Winmostar が LAMMPS を呼び出せるようにパスを設定する。Winmostar の[ツール]→[環境設定]をクリックして環境設定パネルを開く。環境設定パネル[プログラムパス]タブを開き、

[LAMMPS]の[...]ボタンをクリックする。LAMMPS の実行ファイル (lmp\_serial.exe) を登録し [OK]をクリックする。



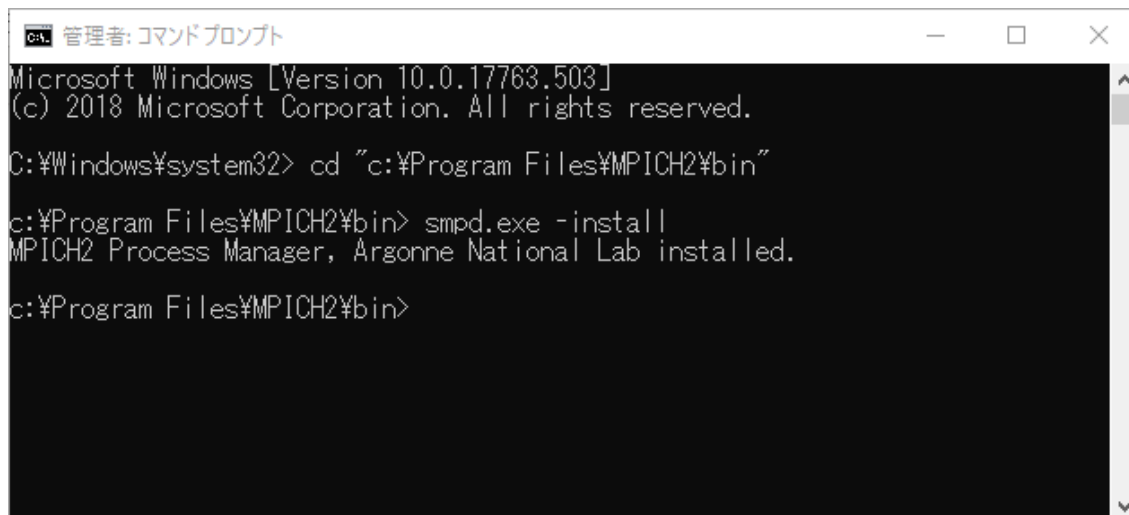
※ デフォルトでは、C:\Program Files\LAMMPS 64-bit 20160309\bin\lmp\_serial.exe と表示されるため、32bit の場合は C:\Program Files\LAMMPS 32-bit 20160309\bin\lmp\_serial.exe などに変更する。

2. cygwin\_wm の入手とセットアップ (既に cygwin\_wm がセットアップ済の場合は不要)  
コンパイル済みパッケージを下記のリンクからダウンロードしセットアップを行う。

[https://winmostar.com/jp/gmx4wm\\_jp.html](https://winmostar.com/jp/gmx4wm_jp.html)

3. MPICH の入手とインストール (LAMMPS の並列実行を行う場合のみ必要)

- ① サイトにアクセスする。 <http://rpm.lammps.org/windows.html>
- ② OS に応じて [[mpich2-1.4.1p1-win-ia32.msi](#)] もしくは [[mpich2-1.4.1p1-win-x86-64.msi](#)] をクリックし msi ファイルをダウンロードする (拡張子の変更された場合は .msi に戻す)。なお、LAMMPS が 32-bit であれば、MPICH も 32-bit を選択する (64-bit の場合は 64-bit を選択する)。
- ③ 保存した msi ファイルをダブルクリックし指示に従う。( .NET Framework がインストールされていないためインストールに失敗した場合は、 <https://www.microsoft.com/ja-jp/download/details.aspx?id=21> から .NET Framework 3.5 をダウンロードしてインストールする。Windows 8.1/10 には標準で .NET Framework 4.5 がインストールされているが、3.5 を別途インストールする必要がある)
- ④ スタートメニューなどから **コマンドプロンプト** を **管理者権限** で立ち上げる。
- ⑤ MPICH をインストールしたフォルダに移動する。  
C:¥> cd "C:¥Program Files¥MPICH2¥bin"
- ⑥ MPICH のセットアップコマンド(smpd.exe)を実行する。  
C:¥Program Files¥MPICH2¥bin> smpd.exe -install



```
管理者: コマンドプロンプト
Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:¥Windows¥system32> cd "c:¥Program Files¥MPICH2¥bin"

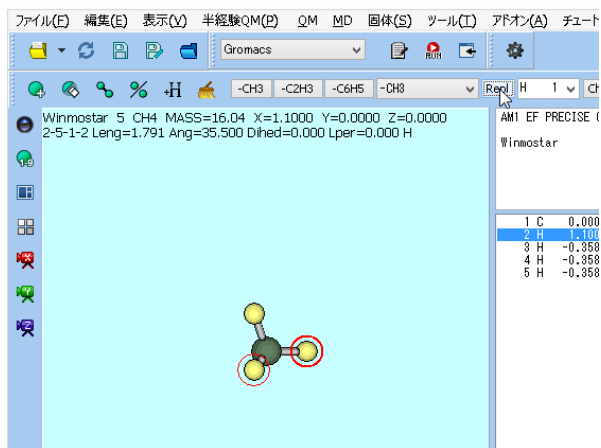
c:¥Program Files¥MPICH2¥bin> smpd.exe -install
MPICH2 Process Manager, Argonne National Lab installed.

c:¥Program Files¥MPICH2¥bin>
```

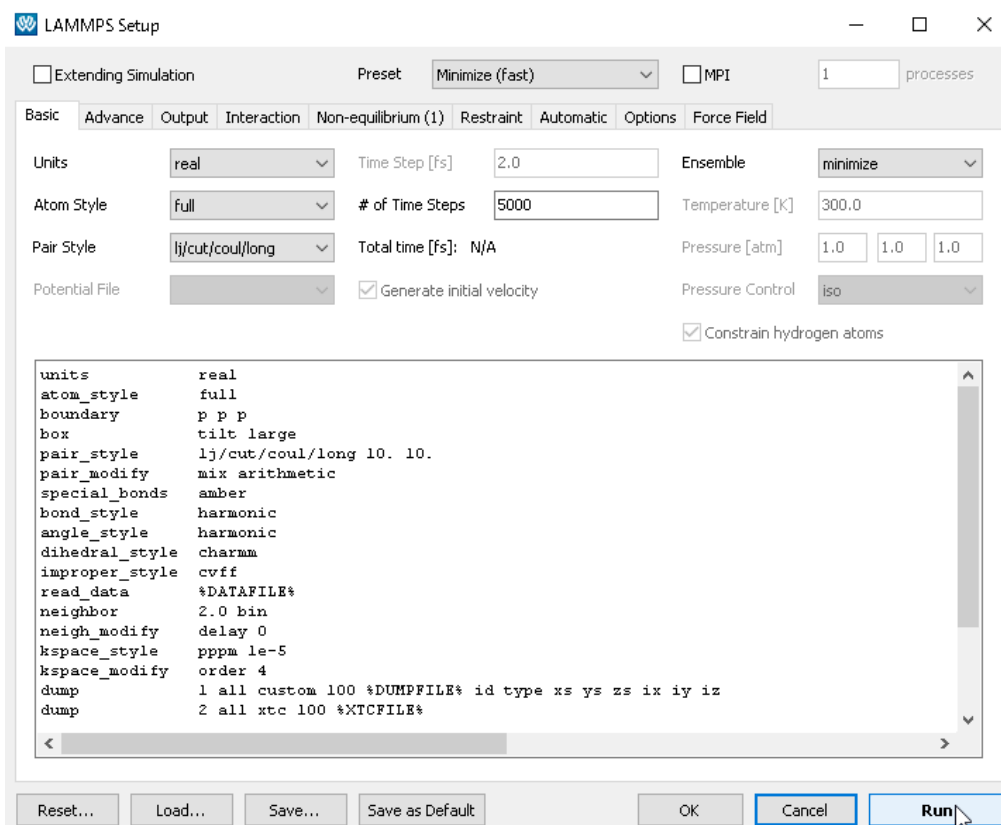
- ⑦ Winmostar の [ツール] → [環境設定] をクリックして環境設定パネルを開く。環境設定パネル [プログラムパス] タブを開き、[MPICH2] の [...] ボタンをクリックする。MPICH の実行ファイル (mpiexec.exe) を登録し [OK] をクリックする。

#### 4. 簡易的な動作確認

- ① Winmostar のメインメニューの [ファイル]-[新規]をクリックし、メインウインドウ上部の [Rep] ボタンをクリックすると、メタン分子が作成される。



- ② [MD]-[LAMMPS]-[キーワード設定]をクリックし、LAMMPS Setup ウィンドウ右下の [Run] ボタンをクリックする。



- ③ 「シミュレーションセルを作成しますか？」とダイアログで聞かれるので、[はい]ボタンを押す。
- ④ 「距離を入力」と書かれたダイアログが出現するので、デフォルトの値のまま[OK]ボタンを押す。
- ⑤ 保存ダイアログが開き、適当な名前で作成すると黒いターミナルウィンドウが開き、大量のメッセージが流れる。暫くすると処理が終わりターミナルウィンドウが閉じる。
- ⑥ [MD]-[LAMMPS]-[ログを表示]をクリックし、ダイアログにてデフォルトで選択されたファイルを開く。LAMMPS が流れたようであれば、開かれたテキストファイルの最後の方に、計算の統計が表示される。

```

Loop time of 0 on 1 procs for 9 steps with 5 atoms
0.0% CPU use with 1 MPI tasks x 1 OpenMP threads

Minimization stats:
Stopping criterion = energy tolerance
Energy initial, next-to-last, final =
0.189620421404 0.0905826305656 0.0905782098341
Force two-norm initial, final = 11.354 0.0902141
Force max component initial, final = 5.41018 0.0478592
Final line search alpha, max atom move = 0.03125 0.0014956
Iterations, force evaluations = 9 36

MPI task timing breakdown:
Section | min time | avg time | max time | %varavg | %total
-----|-----|-----|-----|-----|-----
Pair    | 0         | 0         | 0         | 0.0      | 0.00
Bond    | 0         | 0         | 0         | 0.0      | 0.00
Kspace  | 0         | 0         | 0         | 0.0      | 0.00
Neigh   | 0         | 0         | 0         | 0.0      | 0.00
Comm    | 0         | 0         | 0         | 0.0      | 0.00
Output  | 0         | 0         | 0         | 0.0      | 0.00
Modify  | 0         | 0         | 0         | 0.0      | 0.00
Other   | 0         | 0         | 0         | 0.0      | 0.00

Nlocal: 5 ave 5 max 5 min
Histogram: 1 0 0 0 0 0 0 0
Nehost: 11 ave 11 max 11 min
Histogram: 1 0 0 0 0 0 0 0
Neighs: 10 ave 10 max 10 min
Histogram: 1 0 0 0 0 0 0 0

Total # of neighbors = 10
Ave neighs/atom = 2
Ave special neighs/atom = 4
Neighbor list builds = 0
Dangerous builds = 0
System init for write_restart ...
PPPM initialization ...
G vector (1/distance) = 0.130614
grid = 3 3 3
stencil order = 4
estimated absolute RMS force accuracy = 0.00265415
estimated relative force accuracy = 7.99291e-006
using double_precision FFTs
3d grid and FFT values/proc = 512 27
Total wall time: 0:00:00

```

以上