Flang and OpenMP offloading in Orb5

Ville-Markus Yli-Suutala

Why flang?

- Only way to do compiler directive based GPU programming in Fortran on AMD GPUs

Flang background

- Modern open source Fortran compiler
- Developed from scratch to replace Classic Flang
- Started out as F18 at Nvidia, first announced in 2018
- Part of the llvm-project since 2020

Flang background

- Uses MLIR (Multi Level Intermediate Representation), it’s like a framework for compiler intermediate representations
- Uses LLVM so it can benefit from the massive resources put into that, run on many platforms, etc.
- Uses the same OpenMP and GPU libraries as clang

Flang background

- Developers at ARM, AMD and Nvidia
- Intel has their own ifx compiler that does use LLVM

Where to get it?

- Upstream https://github.com/llvm/llvm-project
- AMD fork https://github.com/ROCm/llvm-project
- Prebuilt binaries at https://repo.radeon.com/rocm/misc/flang
- Small readme at https://github.com/amd/InfinityHub-CI/tree/main/fortran

Which one?

- AMD fork has more advanced support for OpenMP offloading
- Features come here before being upstreamed
- Not everything is upstreamed

Building flang from source

```
cmake \
  -G Ninja \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=$INSTALLDIR \
  -DCMAKE_CXX_STANDARD=17 \
  -DCMAKE_EXPORT_COMPILE_COMMANDS=ON \
  -DCMAKE_CXX_LINK_FLAGS="-Wl,-rpath,$LD_LIBRARY_PATH" \
  -DLLVM_ENABLE_ASSERTIONS=ON \
  -DLLVM_TARGETS_TO_BUILD='X86;AMDGPU' \
  -DLLVM_LIT_ARGS=-v \
  -DLLVM_ENABLE_PROJECTS='clang;mlir;flang;lld' \
  -DLLVM_ENABLE_RUNTIMES='compiler-rt;openmp;offload;flang-rt' \
  -DRUNTIMES_amdgcn-amd-amdhsa_LLVM_ENABLE_RUNTIMES='libc;openmp' \
  -DLLVM_RUNTIME_TARGETS='default;amdgcn-amd-amdhsa' \
  -DLLVM_CCACHE_BUILD=ON \
  -DLLVM_ENABLE_PER_TARGET_RUNTIME_DIR=ON \
  ../llvm-project/llvm
```

Building the flang runtime for GPUs

```
CXXFLAGS='-mcode-object-version=5' cmake ../llvm-project/runtimes/ \
  -DCMAKE_BUILD_TYPE=Release \
  -DLLVM_ENABLE_RUNTIMES=flang-rt \
  -DFLANG_RT_EXPERIMENTAL_OFFLOAD_SUPPORT="OpenMP" \
  -DCMAKE_C_COMPILER=clang \
  -DCMAKE_CXX_COMPILER=clang++ \
  -DFLANG_RT_DEVICE_ARCHITECTURES='gfx90a' \
  -DFLANG_RT_INCLUDE_TESTS=OFF
```

Building flang and runtime

- Might be possible to build the GPU runtime in one go with the rest
- Easier to add flags only for the runtime when compiling it separately

Building software

- flang -fopenmp --offload-arch=gfx90a -mcode-object-version=5
- Code object version needs to be manually set to 5 if a rocm version older than 6.3 is used
- Newer devicelibs is required to compile with code object version 6 and some other libraries to run the binaries

Building software

- Libraries need to be rebuilt for flang
- For Orb5 these are mpi and hdf5

Debugging?

- Still work in progress
- CPU debugging is useful
- GPU debugging not yet
- If a GPU kernel is crashing, the kernel name might be printed. It can be useful to get backtrace with gdb

Compiler limitations affecting Orb5

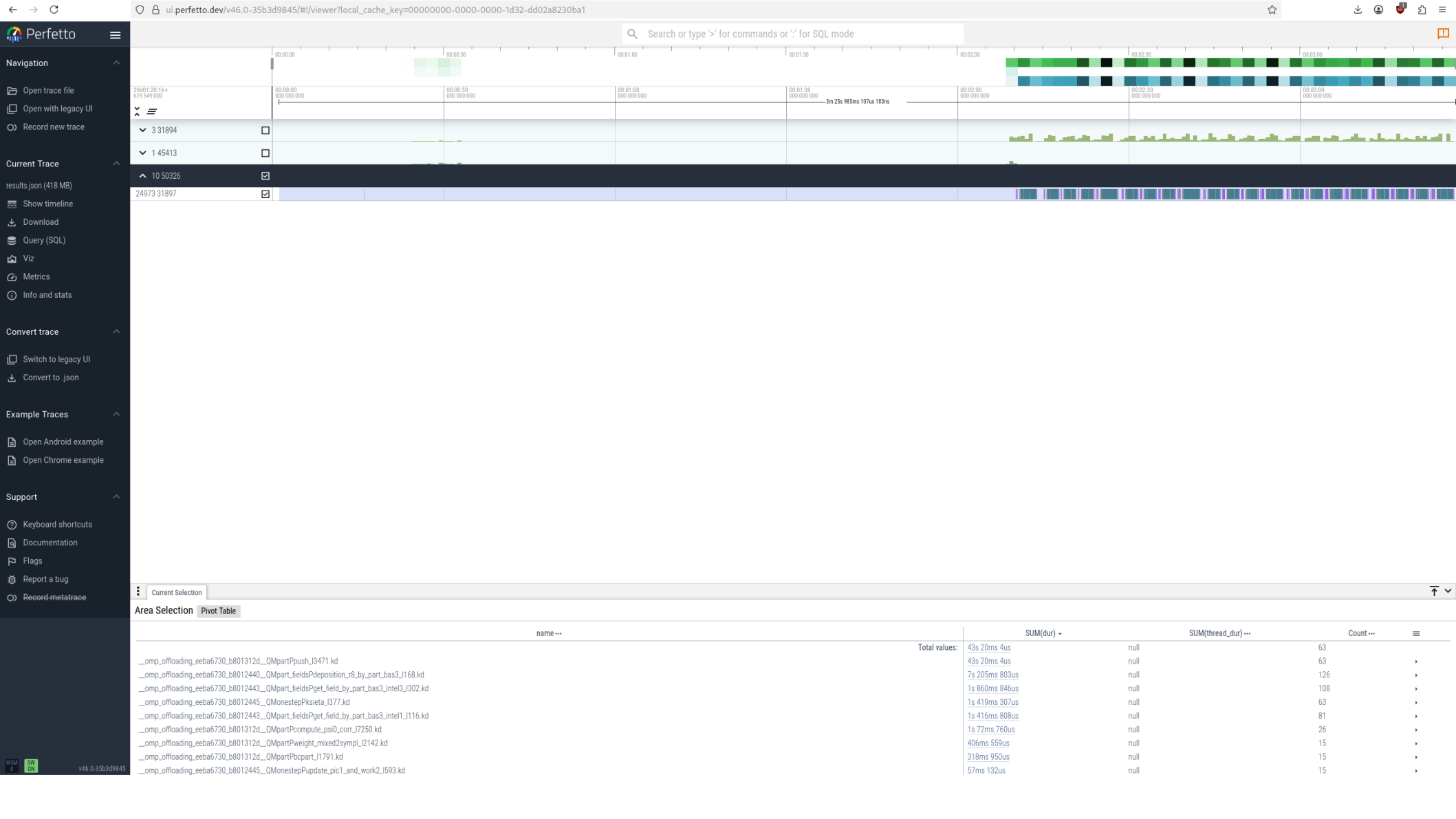
- Data mapping used to be an issue
- Optional arguments can’t be used in device code
- Some intrinsics don’t work on device. bessel_jn for example is not implemented for GPUs. Can be worked around by calling jn from C, clang knows

Compiler limitations affecting Orb5

- OpenMP atomics are faster with clang for some reason
- Operations on arrays are slow on device, better to change an element at a time
- Performance of GPU code in general not very good yet

Profiling on AMD GPUs

- Rocprof
- Omnitrace / ROCm Systems Profiler
- Omniperf / ROCm Compute Profile


ui.perfetto.dev/v46.0-35b3d9845/#!/viewer?local_cache_key=00000000-0000-0000-1d32-dd02a8230ba1
Perfetto
Search or type '>' for commands or ':' for SQL mode
Navigation
Open trace file
Open with legacy UI
Record new trace
Current Trace
results.json (418 MB)
Show timeline
Download
Query (SQL)
Viz
Metrics
Info and stats
Convert trace
Switch to legacy UI
Convert to .json
Example Traces
Open Android example
Open Chrome example
Support
Keyboard shortcuts
Documentation
Flags
Report a bug
Record metatrace
v46.0-35b3d9845
3m 25s 985ms 107us 183ns
3 31894
1 45413
10 50326
24973 31897
Current Selection
Area Selection
Pivot Table
name
SUM(dur)
SUM(thread_dur)
Count
Total values:
43s 20ms 4us
null
63
__omp_offloading_eeba6730_b801312d__QMpartPpush_l3471.kd
43s 20ms 4us
null
63
__omp_offloading_eeba6730_b8012440__QMpart_fieldsPdeposition_r8_by_part_bas3_l168.kd
7s 205ms 803us
null
126
__omp_offloading_eeba6730_b8012443__QMpart_fieldsPget_field_by_part_bas3_intel3_l302.kd
1s 860ms 846us
null
108
__omp_offloading_eeba6730_b8012445__QMonestepPksieta_l377.kd
1s 419ms 307us
null
63
__omp_offloading_eeba6730_b8012443__QMpart_fieldsPget_field_by_part_bas3_intel1_l116.kd
1s 416ms 808us
null
81
__omp_offloading_eeba6730_b801312d__QMpartPcompute_psi0_corr_l7250.kd
1s 72ms 760us
null
26
__omp_offloading_eeba6730_b801312d__QMpartPweight_mixed2sympl_l2142.kd
406ms 559us
null
15
__omp_offloading_eeba6730_b801312d__QMpartPbcpart_l1791.kd
318ms 950us
null
15
__omp_offloading_eeba6730_b8012445__QMonestepPupdate_pic1_and_work2_l593.kd
57ms 132us
null
15

Bugs

- Sudden slowdown a few weeks ago
- Better the next day without changing anything
- Last week the program again became unexpectedly slow
- The results come out clearly wrong

Bugs

- Debugging shows problems start with the push kernel

Bugs

```
! Push the particles
if (species(isp)%i_do_push==1 .and. ipushcoord==1) then
  ! Push GC position
  work1_loc(ip,S_PIC)   = work1_loc(ip,S_PIC)   + basic%dt*dvardt(S_PIC)
  work1_loc(ip,CHI_PIC) = work1_loc(ip,CHI_PIC) + basic%dt*dvardt(CHI_PIC)
  work1_loc(ip,PHI_PIC) = work1_loc(ip,PHI_PIC) + basic%dt*dvardt(PHI_PIC)

  ! Push velocity
  work1_loc(ip,U_PIC) = work1_loc(ip,U_PIC) + basic%dt*dvardt(U_PIC)

  ! Push weights
  work1_loc(ip,W_PIC) = work1_loc(ip,W_PIC) + basic%dt*dvardt(W_PIC)
  if(species(isp)%nl_2weights) then
    work1_loc(ip,P_PIC) = work1_loc(ip,P_PIC) - basic%dt*dvardt(W_PIC)*species(isp)%i_nonlinear
    ! Push the second weight, which evolves with the derivative of f0 along marker trajectories, but only for nonlinear cases
  end if
end if
```

Bugs

- work1_loc(ip,S_PIC) = work1_loc(ip,S_PIC) + basic%dt*dvardt(S_PIC)
- First value is right, but then the second
- work1_loc(ip,S_PIC) = 0 + 100 * -3.236E-07 = -6.472E-05
- The following values seem to be even more off
- Does the loop body execute more than once per particle?

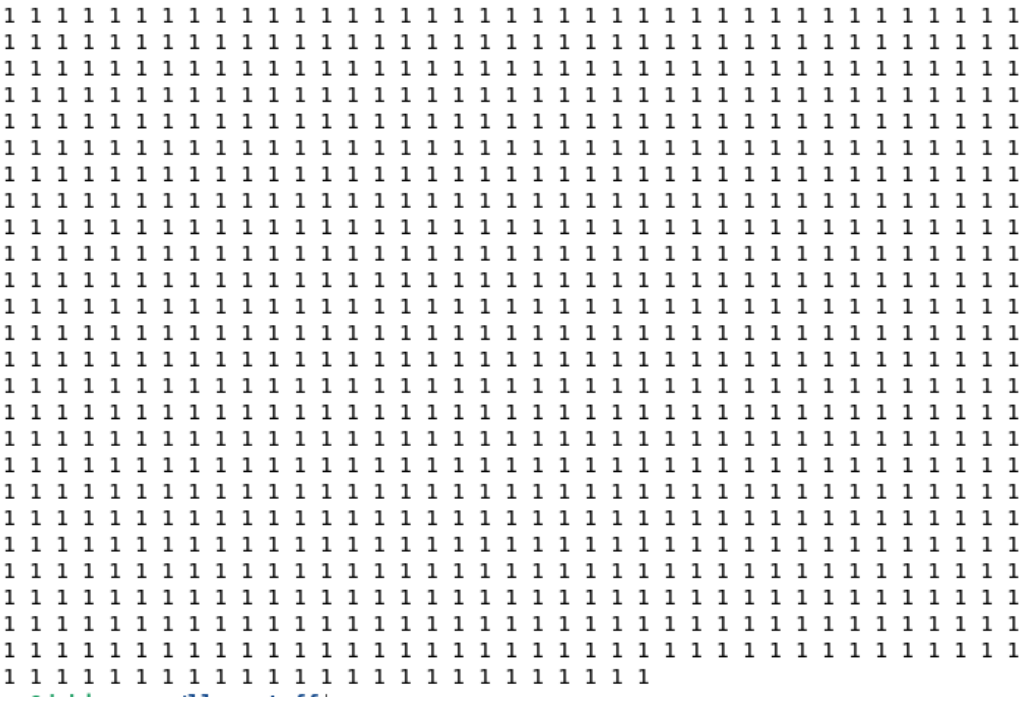
Example

```
subroutine dostuff(n)
integer :: n
integer, allocatable, dimension(:) :: counts
integer :: i

allocate(counts(n))
counts = 0

!$omp target teams distribute parallel do map(counts)
do i=1, n
counts(i) = counts(i) + 1
end do
!$omp end target teams distribute parallel do
print *, counts
end subroutine
```

Expected output



Push loop

```
!$omp target teams distribute parallel do &
!$omp          private(ip, sigmat, chit_cos, chit_sin, chit, phit, wt, pt, ut, ut2, w3t, vpt, vpt2, mub, moqobstar, &
!$omp mgradphi, mgradphi_ant, mgradapar, mgradapar_sympl, veb_f, vexb_ant, hh_sympl, &
!$omp                  pztommoqobstar, pztom, pztom2, bgrb, addp, hh, exh, alldrift, vgb, vpressure, vexb, vexb_tot, &
!$omp                  genelec, genelec_ant, genelec_tot, elec_dadt0, vpar, vgc, addpBst, addvpa, b, divh, bstar, b_inv, &
!$omp                  bstar_inv, ze, rt, toB, tprime, dvpadt1_back, vdr, vdchi, dPsidt0, dPsidt1, dvpadt0, dvpadt1, dvpadt1_tot, &
!$omp                  pztomobstar, psi0t, mut, tval, tgrad, nval, ngrad, vpval, vpgrad, zecor, zvth2, zf0, rhs, zcutoffvpa, &
!$omp                  pvolgc, distribution, df0_dpsi_prof, df0_dchi_prof, dpsi_prof_dt, dchi_prof_dt, ipushcoord, dvardt, vapar, psitnc, vin, &
!$omp                  vecrossb_f2, dpotdpsi, kappa_str, omegasq_corr, psi_corr, pvol_updated, is_trapped, in_ksieta, &
!$omp                  ithread, exb, bufc, dH0_dt_correction) &
!$omp     firstprivate(gyroapar, expt_back, eypt_back, pot_equil, &
!$omp                  pot_equil0, Epsi, fvpllb, dfvpllb_den, dfvpllb_dpsi, dvpadt0_str, dvpadt1_str, dvpadt1_ant)&
!$omp          map(present, alloc: pic1_loc, pic2_loc, pic3_loc, work1_loc, work_temp(isp)%markers, phi, phi_antenna, apar, apar_sympl)
!pomp do schedule(guided)
PARTICLE_LOOP: do ip = 1, npart_loc
```

Counts for push

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32
```

Causes?

- Library compatibility?
- Driver compatibility?
- Compiler bug?

Thank you!

- Questions?