So - order, in time integrator.

Start

↓

Diffusion → Calculates diffusion fluxes and stores in visflx (cndflx) (member data of HydroDiffusion)

↓

Compute Hydro Flux → Hydro::CalculateFluxes
- Runs Reiman solver, ~~second~~ (after reconstructing edge states) in all 3 directions
- Adds in diffusion fluxes (previously calculated)

↓

Hydro Integrate → Using time integrator weights calculates divergence of fluxes and adds it on to conserved variables (using weight)

$$\text{u\_out} \mathrel{-}= \underset{\text{time integrator}}{\text{wght}} * dt \times \underset{\text{div(flux)}}{\text{dflx}} / \text{vol}$$

↓

Same for fields

↓

Compute new primitives
:: Primitives

defined separately in adiabatic_hydro.cpp ... _mhd.cpp etc.

mesh → eos → ConservedToPrimitive

pretty simple except relativity

Swap pointer to new variable

↓

Boundary Conditions

↓

UserWork

↓

New dt → mesh → hydro → NewBlockTimeStep()

CFL across ~~bl~~ block

pmesh is a mesh
ptlist is a TimeIntegratorTaskList

Could start with adding Braginskii viscosity

Standard viscosity - set by CalcViscCoeff function (pointer) which defaults to ConstViscosity function

All stored very abstractly as "TaskLists"

Then you write the full list of tasks with their dependencies (bit OR allows them to be added)
Stores these in task_list object list of "Tasks" which contain task, dep (uint64) & a pntr to the task function. Seems clever! (if a bit opaque)

TimeIntegratorTaskList references various Hydro or Field methods

Diffusive fluxes are done first

```
         ph
pmb->phydro->phdif -> CalcHydroDiffusionFlux(ph->w,
 ↑       ↑      ↑                               ph->u, ph->fla
meshblock hydro  hydrodiffusion
```

ph->w is prim
ph->u is cons

This just calculates ~~[illegible]~~ diffusive flux - it is added in Hydro::CalculateFluxes. at the end