First VPLanet Developers Workshop

Lesson 9
How to Add a Module

Overview

Adding a module is a major task, but can result in the biggest leap forward in our understanding of planetary evolution

While complicated, the 12 existing modules are templates for your new module

Each module is different, but VPLanet is designed for new modules to plug and play

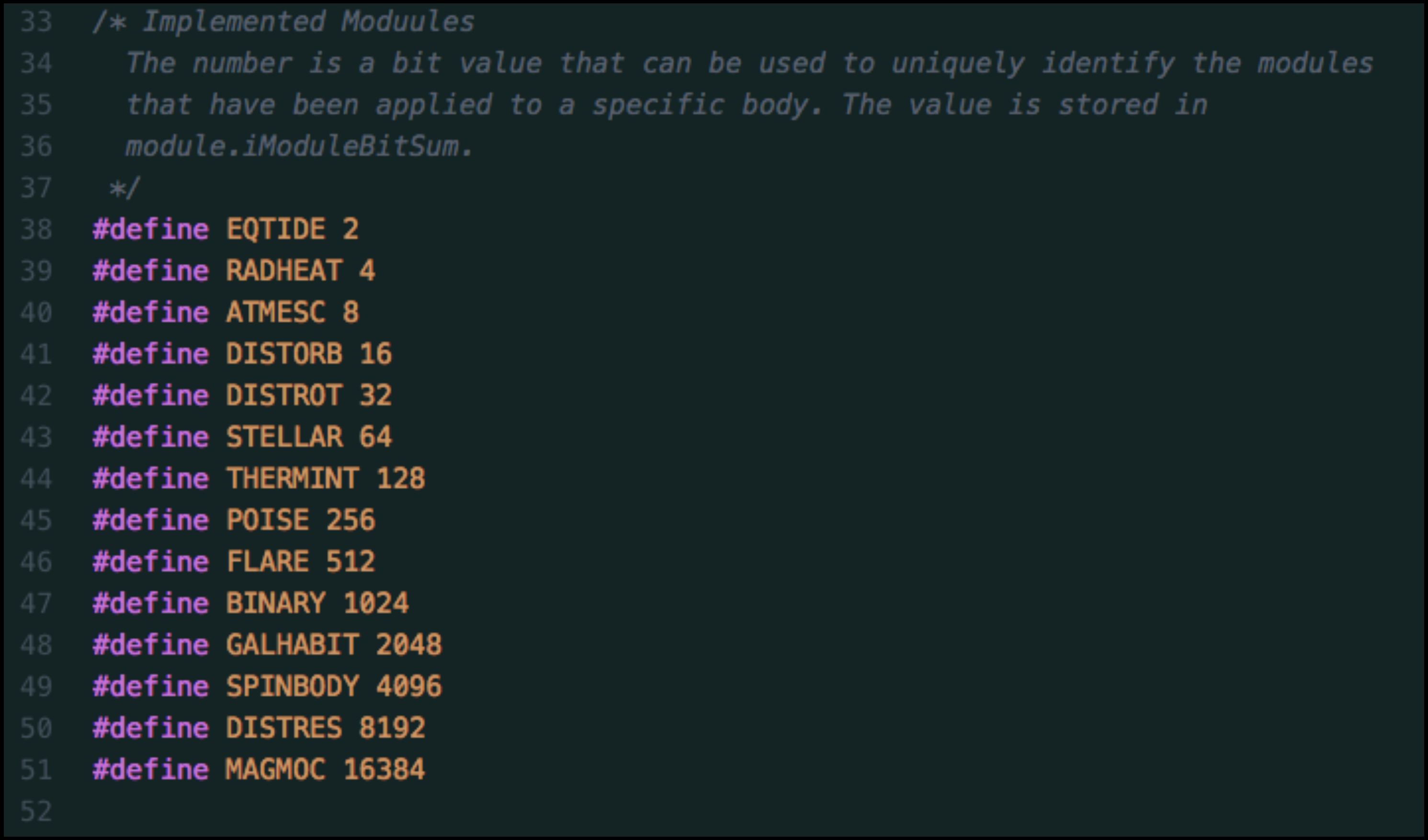
Here we'll use EqTide as a template

Overview

Here are the steps

- Create a new module ID
- Create new source files .[ch]
- InitializeControl
- BodyCopy
- InitializeBody
- InitializeUpdateTmpBody
- ReadOptions
- VerifyOptions
- Add the Update and Halt functions
- Output functions
- Write the AddModule function
- Write physics functions unique to the module
- Write the header file
- Update module.c and update.c
- Add new members to the structs

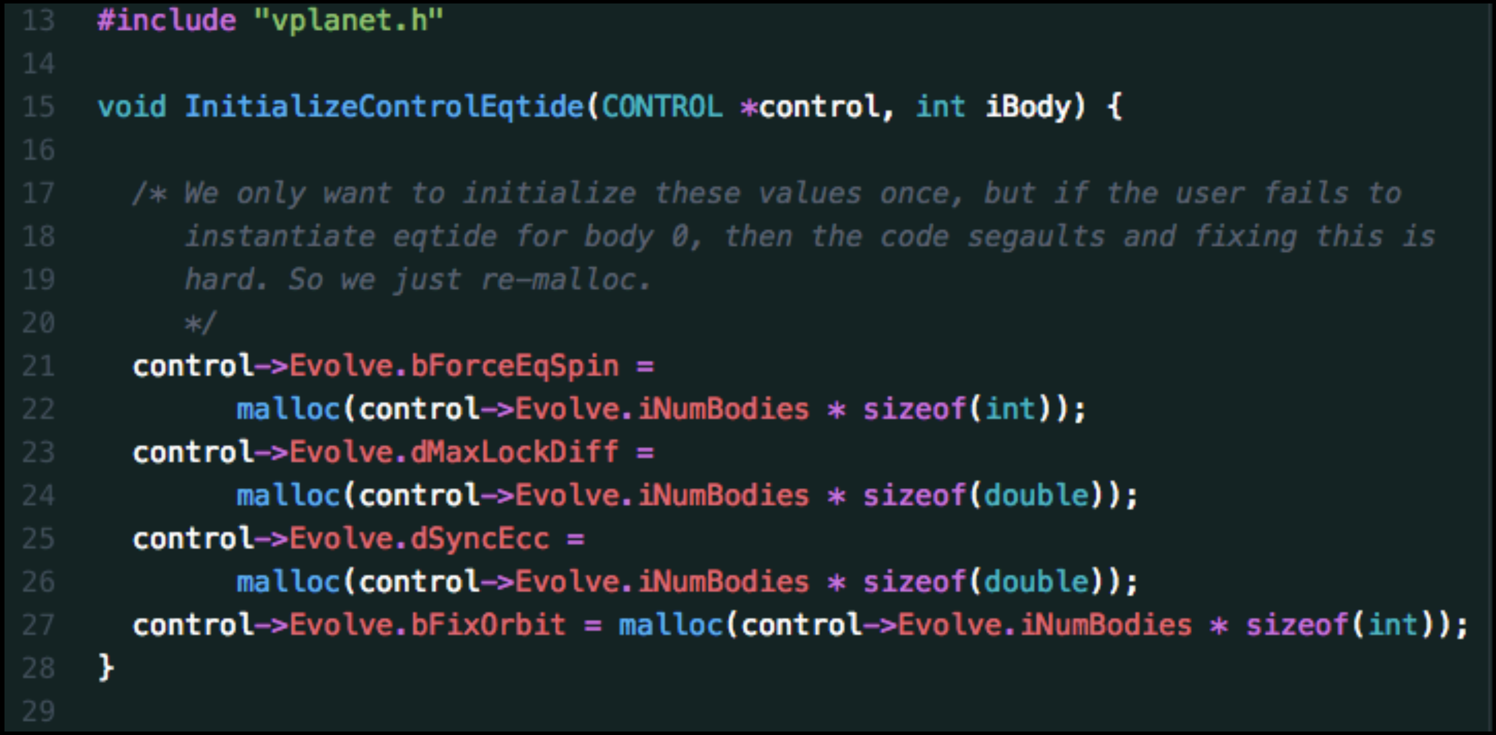
Create a New Module Bit



```
/* Implemented Moduules
  The number is a bit value that can be used to uniquely identify the modules
  that have been applied to a specific body. The value is stored in
  module.iModuleBitSum.
 */
#define EQTIDE 2
#define RADHEAT 4
#define ATMESC 8
#define DISTORB 16
#define DISTROT 32
#define STELLAR 64
#define THERMINT 128
#define POISE 256
#define FLARE 512
#define BINARY 1024
#define GALHABIT 2048
#define SPINBODY 4096
#define DISTRES 8192
#define MAGMOC 16384

```

The integer must be a power of 2

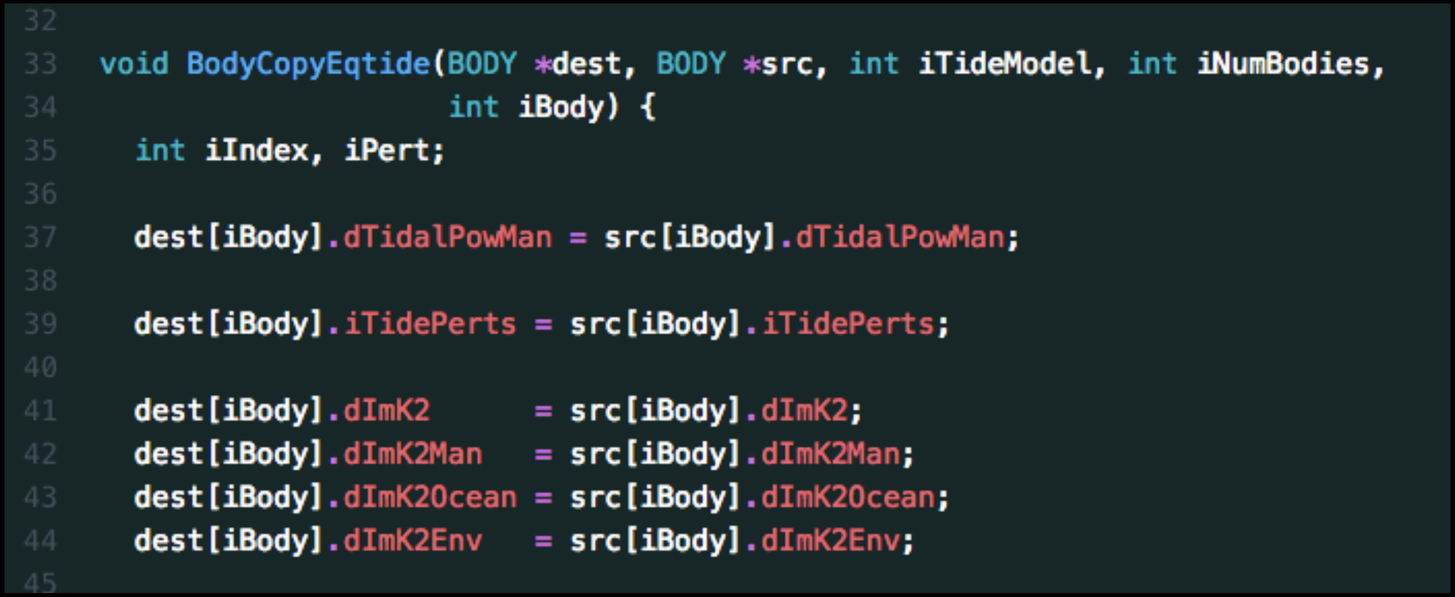
InitializeControl



```c
#include "vplanet.h"

void InitializeControlEqtide(CONTROL *control, int iBody) {

  /* We only want to initialize these values once, but if the user fails to
     instantiate eqtide for body 0, then the code segaults and fixing this is
     hard. So we just re-malloc.
     */
  control->Evolve.bForceEqSpin =
        malloc(control->Evolve.iNumBodies * sizeof(int));
  control->Evolve.dMaxLockDiff =
        malloc(control->Evolve.iNumBodies * sizeof(double));
  control->Evolve.dSyncEcc =
        malloc(control->Evolve.iNumBodies * sizeof(double));
  control->Evolve.bFixOrbit = malloc(control->Evolve.iNumBodies * sizeof(int));
}
```

Allocate memory in the CONTROL struct
Most modules don’t use this function
Note that all function pointer arrays are initialized to point to NULL
If your module doesn’t need a function, just omit it

BodyCopy



```c

void BodyCopyEqtide(BODY *dest, BODY *src, int iTideModel, int iNumBodies,
                    int iBody) {
  int iIndex, iPert;

  dest[iBody].dTidalPowMan = src[iBody].dTidalPowMan;

  dest[iBody].iTidePerts = src[iBody].iTidePerts;

  dest[iBody].dImK2      = src[iBody].dImK2;
  dest[iBody].dImK2Man   = src[iBody].dImK2Man;
  dest[iBody].dImK2Ocean = src[iBody].dImK2Ocean;
  dest[iBody].dImK2Env   = src[iBody].dImK2Env;

```

In Runge-Kutta, we must calculate midpoint derivatives

In order to not overwrite the BODY struct, we use tmpBody structs

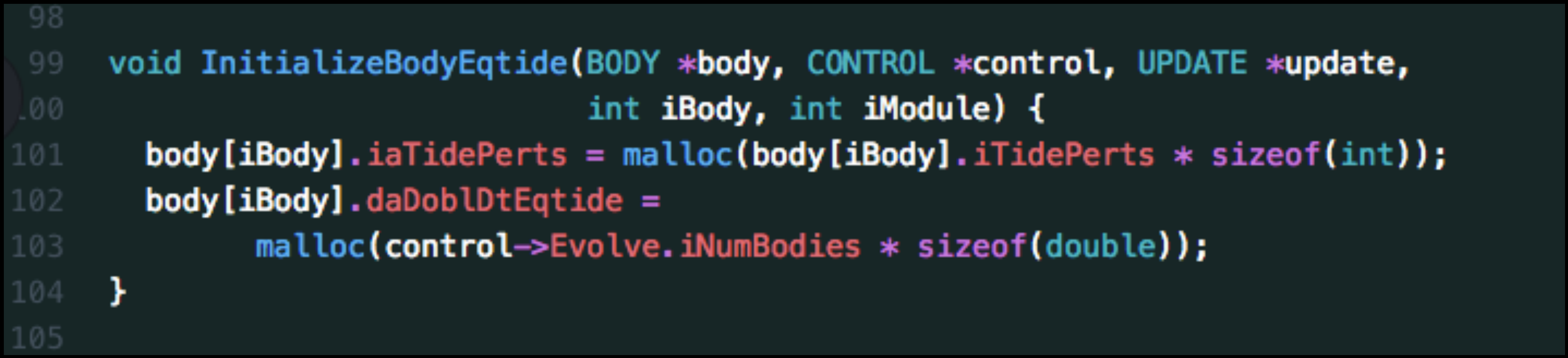
BodyCopy performs that copying

Sadly C doesn’t allow for direct struct copies

- Cant say tmpBody = body;

Failure to include a new parameter in BodyCopy is the #1 way to introduce bugs into VPLanet!

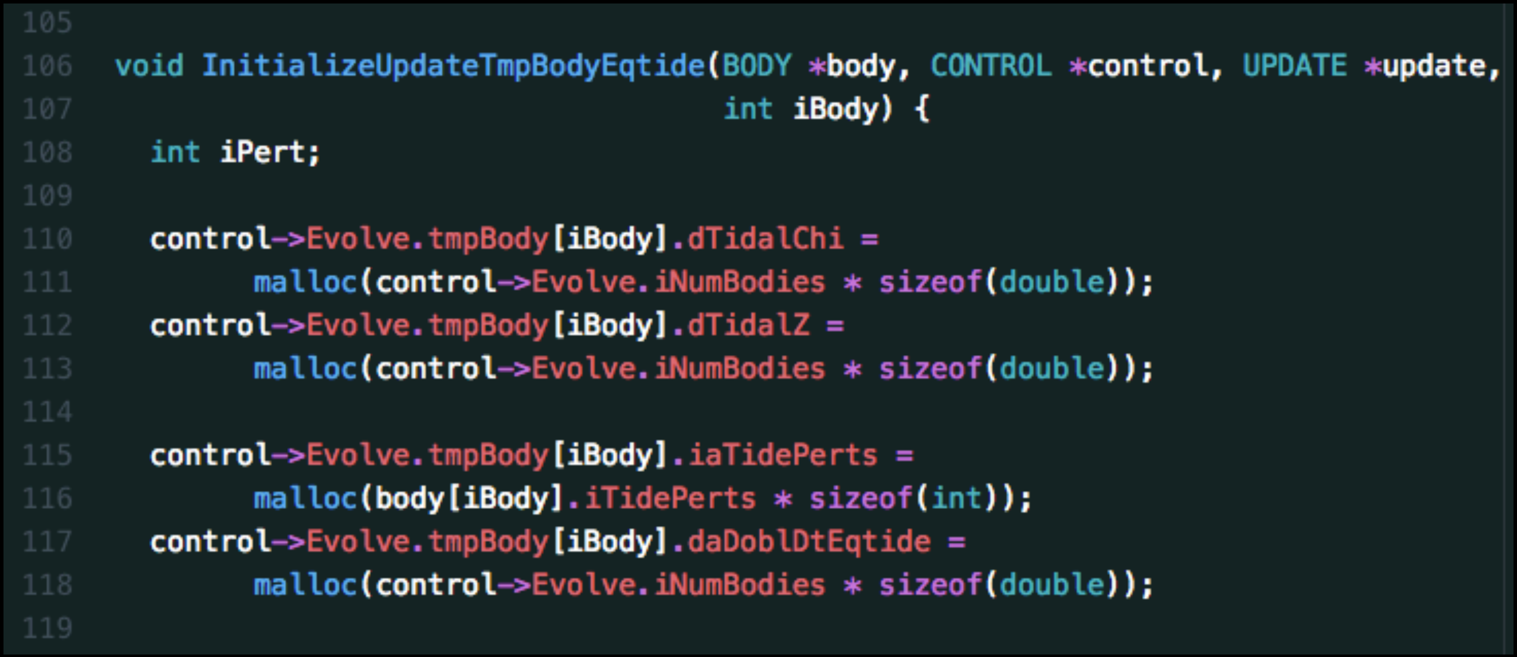
InitializeBody



```
void InitializeBodyEqtide(BODY *body, CONTROL *control, UPDATE *update,
                          int iBody, int iModule) {
  body[iBody].iaTidePerts = malloc(body[iBody].iTidePerts * sizeof(int));
  body[iBody].daDoblDtEqtide =
        malloc(control->Evolve.iNumBodies * sizeof(double));
}
```

Allocate memory in the BODY struct
InitializeBody is not a commonly used function
(In EqTide, the central body can tidally perturb multiple orbiters)

InitializeTmpBody

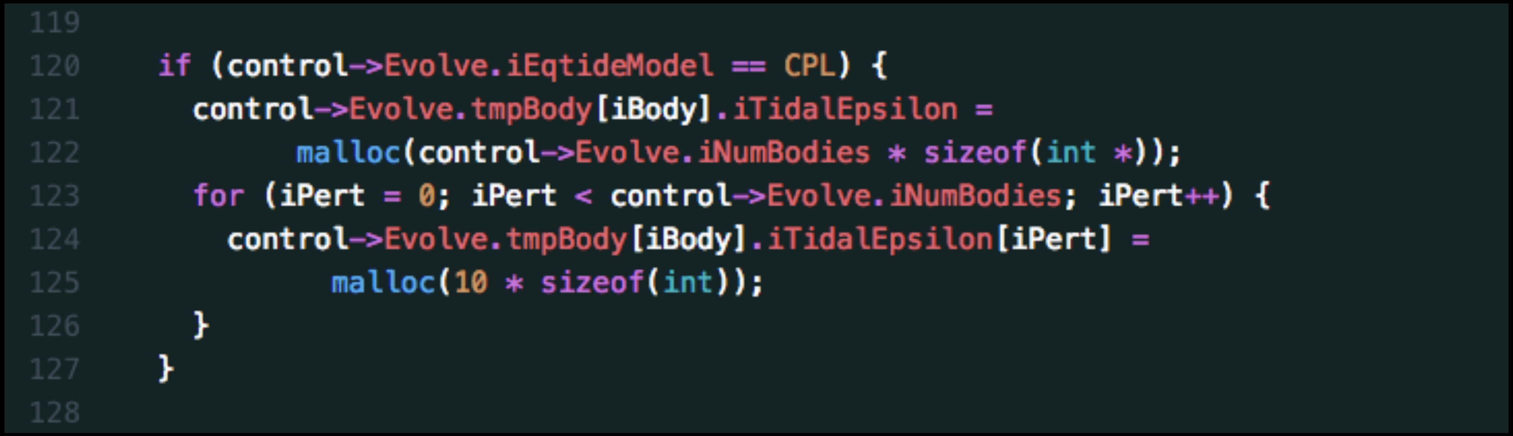


```c
void InitializeUpdateTmpBodyEqtide(BODY *body, CONTROL *control, UPDATE *update,
                                   int iBody) {
  int iPert;

  control->Evolve.tmpBody[iBody].dTidalChi =
        malloc(control->Evolve.iNumBodies * sizeof(double));
  control->Evolve.tmpBody[iBody].dTidalZ =
        malloc(control->Evolve.iNumBodies * sizeof(double));

  control->Evolve.tmpBody[iBody].iaTidePerts =
        malloc(body[iBody].iTidePerts * sizeof(int));
  control->Evolve.tmpBody[iBody].daDoblDtEqtide =
        malloc(control->Evolve.iNumBodies * sizeof(double));

```

Here we allocate memory in Evolve->tmpBody

Most of these arrays in BODY are allocated in Verify or InitializeBody

Can also make decision based on the model selected



```c

  if (control->Evolve.iEqtideModel == CPL) {
    control->Evolve.tmpBody[iBody].iTidalEpsilon =
          malloc(control->Evolve.iNumBodies * sizeof(int *));
    for (iPert = 0; iPert < control->Evolve.iNumBodies; iPert++) {
      control->Evolve.tmpBody[iBody].iTidalEpsilon[iPert] =
            malloc(10 * sizeof(int));
    }
  }

```

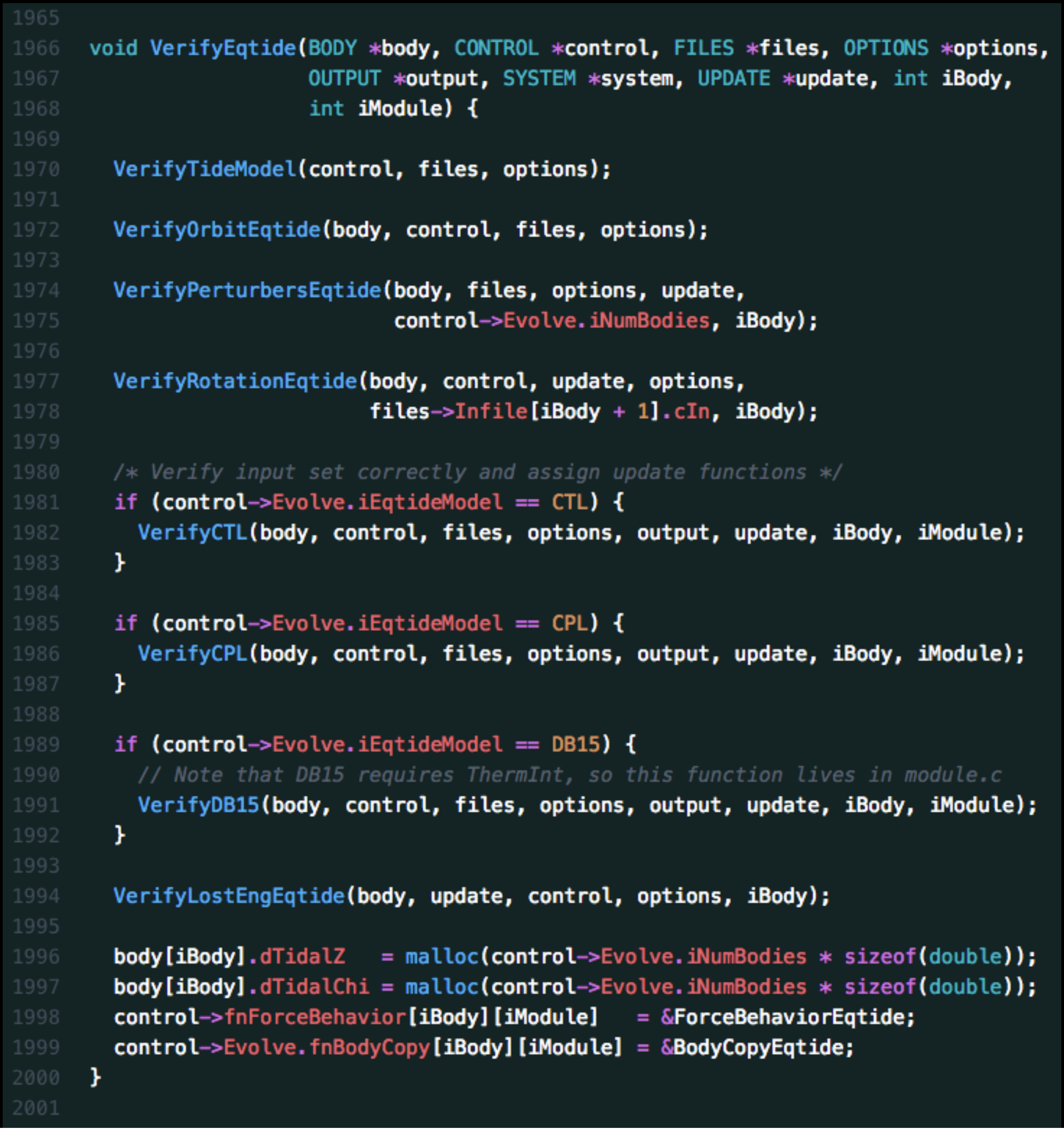
Options

The next block of code is for options (Lesson 6)

Write your new functions, as well as InitializeOptions

Since we’ve already covered this, we’ll move on to Verify…

Verify

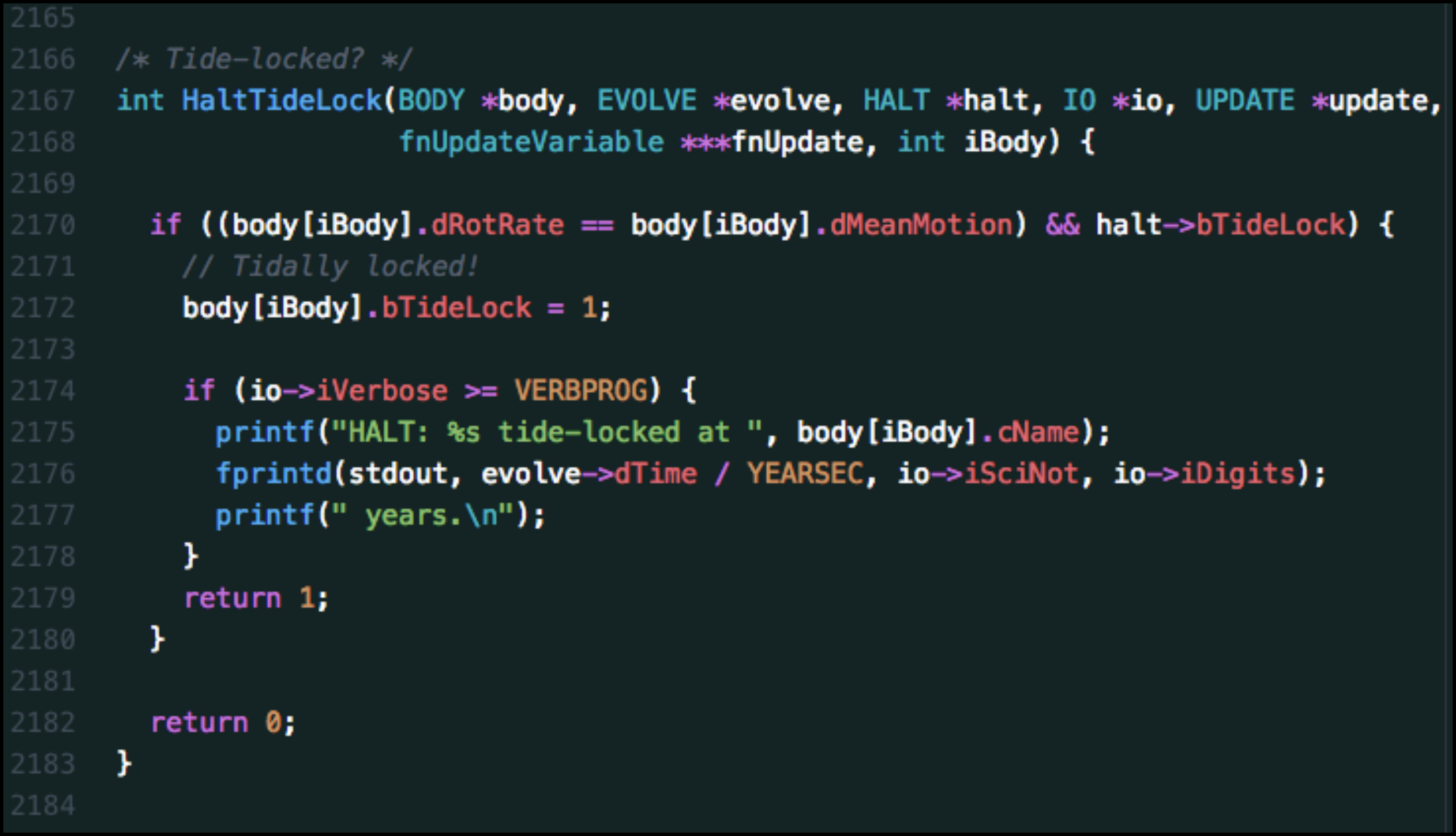


```c

void VerifyEqtide(BODY *body, CONTROL *control, FILES *files, OPTIONS *options,
                  OUTPUT *output, SYSTEM *system, UPDATE *update, int iBody,
                  int iModule) {

  VerifyTideModel(control, files, options);

  VerifyOrbitEqtide(body, control, files, options);

  VerifyPerturbersEqtide(body, files, options, update,
                         control->Evolve.iNumBodies, iBody);

  VerifyRotationEqtide(body, control, update, options,
                       files->Infile[iBody + 1].cIn, iBody);

  /* Verify input set correctly and assign update functions */
  if (control->Evolve.iEqtideModel == CTL) {
    VerifyCTL(body, control, files, options, output, update, iBody, iModule);
  }

  if (control->Evolve.iEqtideModel == CPL) {
    VerifyCPL(body, control, files, options, output, update, iBody, iModule);
  }

  if (control->Evolve.iEqtideModel == DB15) {
    // Note that DB15 requires ThermInt, so this function lives in module.c
    VerifyDB15(body, control, files, options, output, update, iBody, iModule);
  }

  VerifyLostEngEqtide(body, update, control, options, iBody);

  body[iBody].dTidalZ   = malloc(control->Evolve.iNumBodies * sizeof(double));
  body[iBody].dTidalChi = malloc(control->Evolve.iNumBodies * sizeof(double));
  control->fnForceBehavior[iBody][iModule]    = &ForceBehaviorEqtide;
  control->Evolve.fnBodyCopy[iBody][iModule] = &BodyCopyEqtide;
}

```

Update and Halt Functions

Next come the update functions, which were discussed in Lesson 7

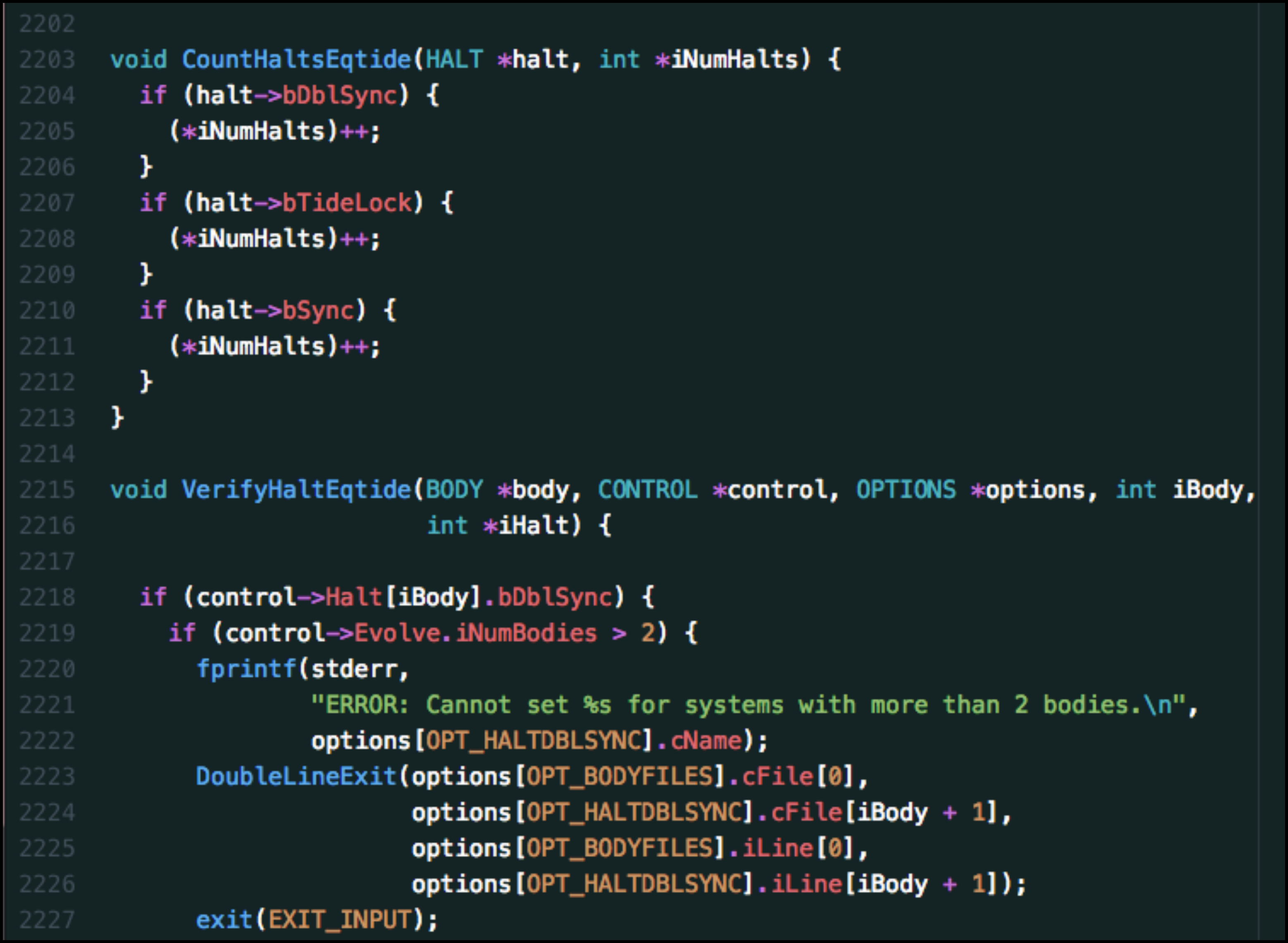
Then come the Halt functions



```c

/* Tide-locked? */
int HaltTideLock(BODY *body, EVOLVE *evolve, HALT *halt, IO *io, UPDATE *update,
                 fnUpdateVariable ***fnUpdate, int iBody) {

  if ((body[iBody].dRotRate == body[iBody].dMeanMotion) && halt->bTideLock) {
    // Tidally locked!
    body[iBody].bTideLock = 1;

    if (io->iVerbose >= VERBPROG) {
      printf("HALT: %s tide-locked at ", body[iBody].cName);
      fprintd(stdout, evolve->dTime / YEARSEC, io->iSciNot, io->iDigits);
      printf(" years.\n");
    }
    return 1;
  }

  return 0;
}

```

Halt functions return either 0 (don’t halt) or 1 (halt)

Update and Halt Functions

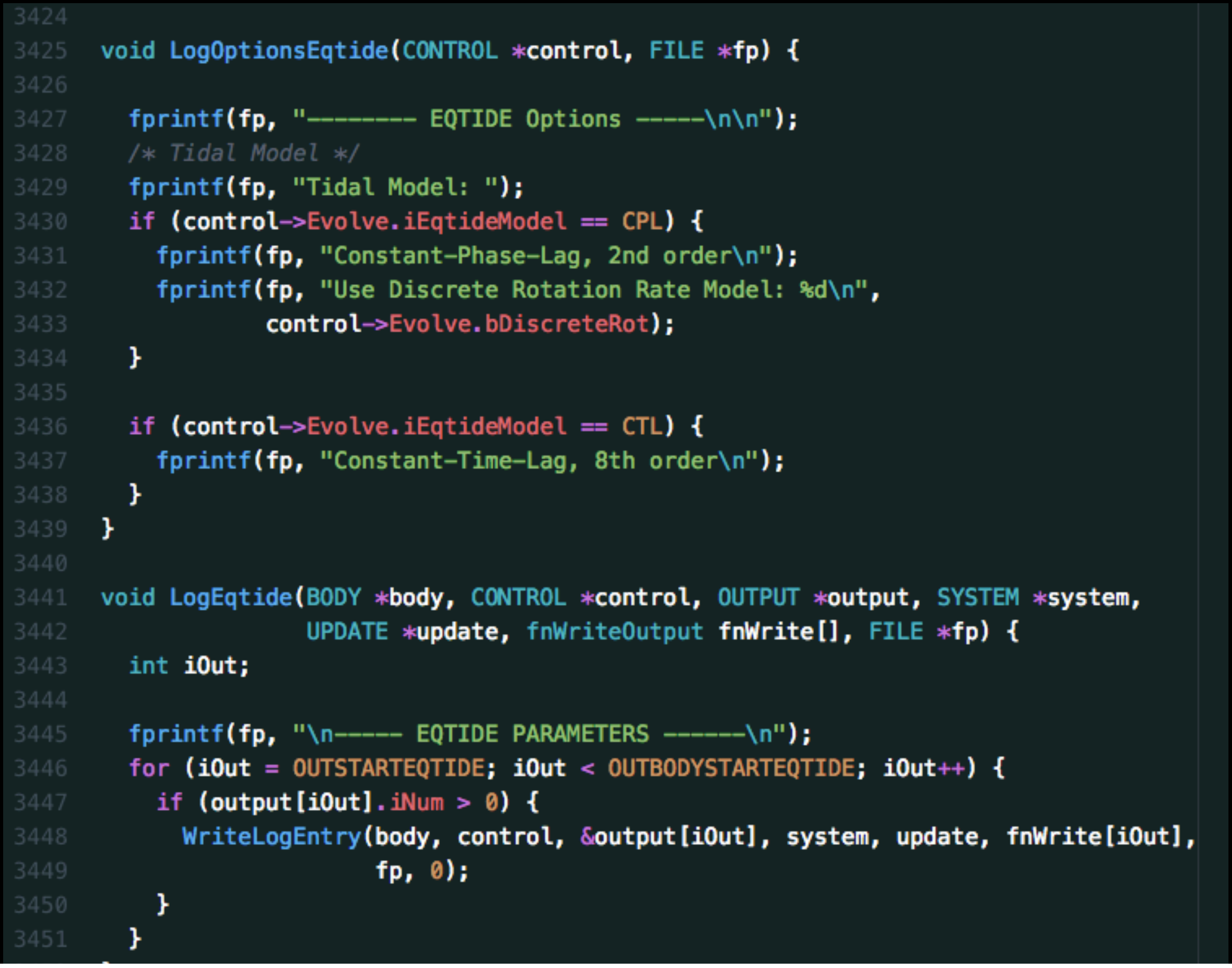


```c

void CountHaltsEqtide(HALT *halt, int *iNumHalts) {
  if (halt->bDblSync) {
    (*iNumHalts)++;
  }
  if (halt->bTideLock) {
    (*iNumHalts)++;
  }
  if (halt->bSync) {
    (*iNumHalts)++;
  }
}

void VerifyHaltEqtide(BODY *body, CONTROL *control, OPTIONS *options, int iBody,
                      int *iHalt) {

  if (control->Halt[iBody].bDblSync) {
    if (control->Evolve.iNumBodies > 2) {
      fprintf(stderr,
              "ERROR: Cannot set %s for systems with more than 2 bodies.\n",
              options[OPT_HALTDBLSYNC].cName);
      DoubleLineExit(options[OPT_BODYFILES].cFile[0],
                     options[OPT_HALTDBLSYNC].cFile[iBody + 1],
                     options[OPT_BODYFILES].iLine[0],
                     options[OPT_HALTDBLSYNC].iLine[iBody + 1]);
      exit(EXIT_INPUT);
```

Add your CountHalts function — part of function pointer array
If necessary, also Verify your halts

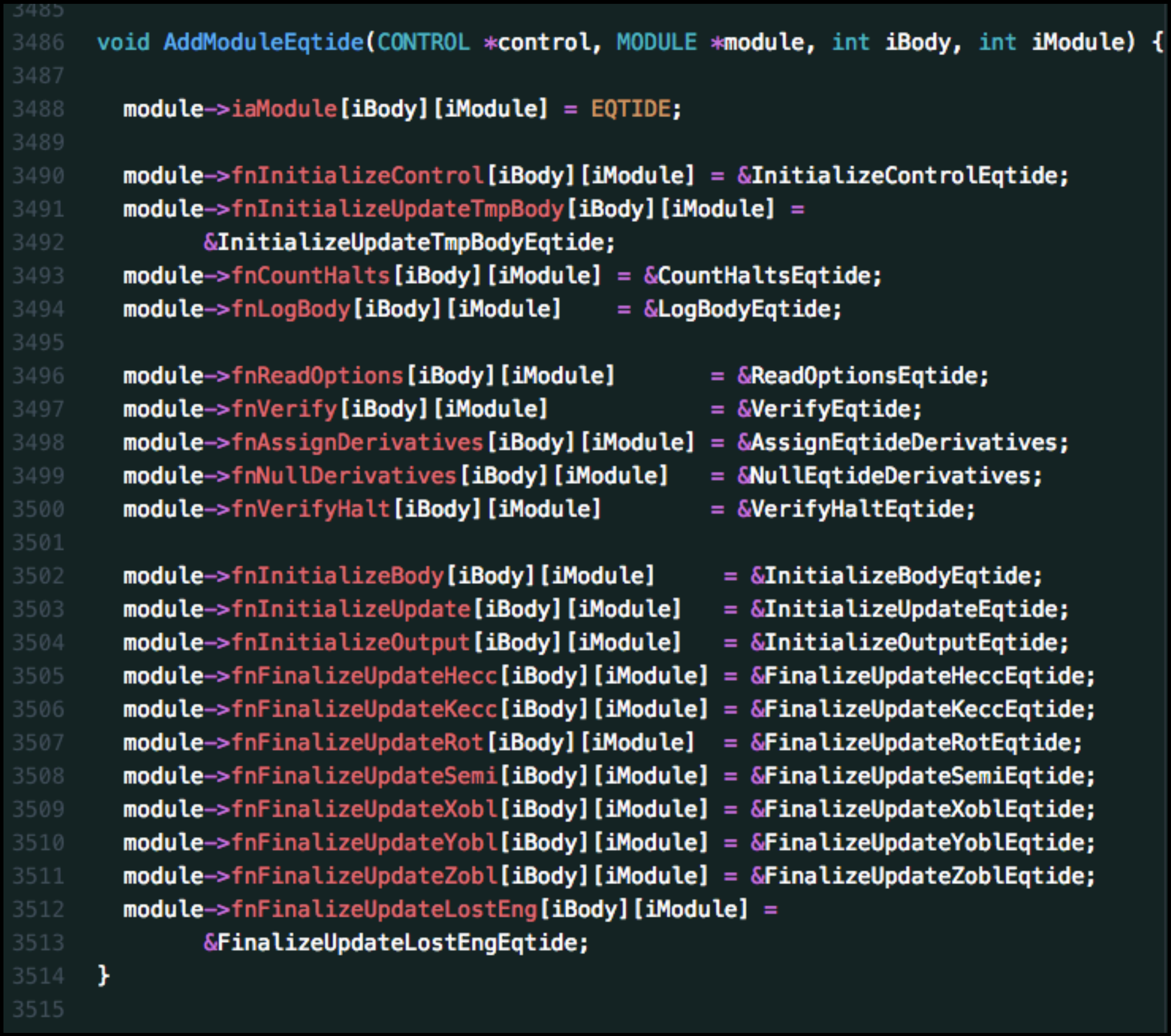
Output Functions

Next come the output functions and InitializeOutput (Lesson 5)
But also need to write the Log functions



```c

void LogOptionsEqtide(CONTROL *control, FILE *fp) {

  fprintf(fp, "-------- EQTIDE Options -----\n\n");
  /* Tidal Model */
  fprintf(fp, "Tidal Model: ");
  if (control->Evolve.iEqtideModel == CPL) {
    fprintf(fp, "Constant-Phase-Lag, 2nd order\n");
    fprintf(fp, "Use Discrete Rotation Rate Model: %d\n",
            control->Evolve.bDiscreteRot);
  }

  if (control->Evolve.iEqtideModel == CTL) {
    fprintf(fp, "Constant-Time-Lag, 8th order\n");
  }
}

void LogEqtide(BODY *body, CONTROL *control, OUTPUT *output, SYSTEM *system,
               UPDATE *update, fnWriteOutput fnWrite[], FILE *fp) {
  int iOut;

  fprintf(fp, "\n----- EQTIDE PARAMETERS ------\n");
  for (iOut = OUTSTARTEQTIDE; iOut < OUTBODYSTARTEQTIDE; iOut++) {
    if (output[iOut].iNum > 0) {
      WriteLogEntry(body, control, &output[iOut], system, update, fnWrite[iOut],
                    fp, 0);
    }
  }
```

AddModule



```c
void AddModuleEqtide(CONTROL *control, MODULE *module, int iBody, int iModule) {

  module->iaModule[iBody][iModule] = EQTIDE;

  module->fnInitializeControl[iBody][iModule] = &InitializeControlEqtide;
  module->fnInitializeUpdateTmpBody[iBody][iModule] =
        &InitializeUpdateTmpBodyEqtide;
  module->fnCountHalts[iBody][iModule] = &CountHaltsEqtide;
  module->fnLogBody[iBody][iModule]     = &LogBodyEqtide;

  module->fnReadOptions[iBody][iModule]        = &ReadOptionsEqtide;
  module->fnVerify[iBody][iModule]             = &VerifyEqtide;
  module->fnAssignDerivatives[iBody][iModule] = &AssignEqtideDerivatives;
  module->fnNullDerivatives[iBody][iModule]   = &NullEqtideDerivatives;
  module->fnVerifyHalt[iBody][iModule]         = &VerifyHaltEqtide;

  module->fnInitializeBody[iBody][iModule]      = &InitializeBodyEqtide;
  module->fnInitializeUpdate[iBody][iModule]    = &InitializeUpdateEqtide;
  module->fnInitializeOutput[iBody][iModule]    = &InitializeOutputEqtide;
  module->fnFinalizeUpdateHecc[iBody][iModule] = &FinalizeUpdateHeccEqtide;
  module->fnFinalizeUpdateKecc[iBody][iModule] = &FinalizeUpdateKeccEqtide;
  module->fnFinalizeUpdateRot[iBody][iModule]  = &FinalizeUpdateRotEqtide;
  module->fnFinalizeUpdateSemi[iBody][iModule] = &FinalizeUpdateSemiEqtide;
  module->fnFinalizeUpdateXobl[iBody][iModule] = &FinalizeUpdateXoblEqtide;
  module->fnFinalizeUpdateYobl[iBody][iModule] = &FinalizeUpdateYoblEqtide;
  module->fnFinalizeUpdateZobl[iBody][iModule] = &FinalizeUpdateZoblEqtide;
  module->fnFinalizeUpdateLostEng[iBody][iModule] =
        &FinalizeUpdateLostEngEqtide;
}
```

Module Functions

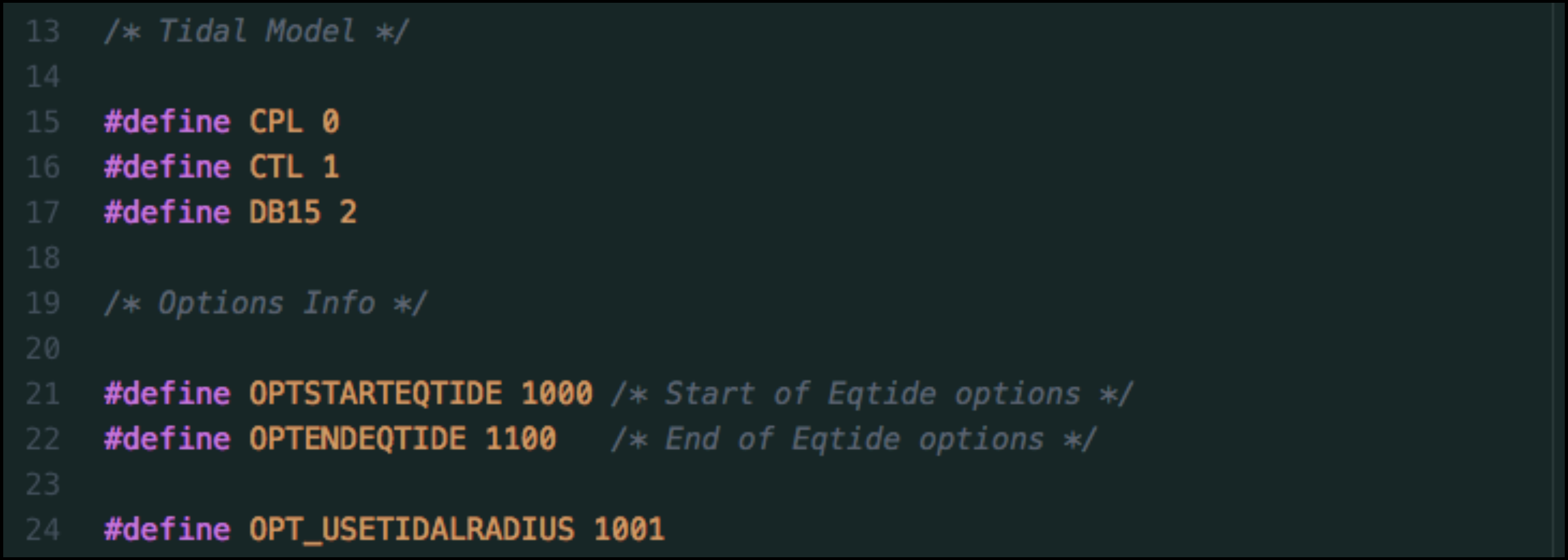
Finally, write all the functions specific to your module

- PropsAux
- ForceBehavior
- fnUpdate functions
- Any "helper" functions unique to your module

These steps complete the .c file, but you're not done yet!

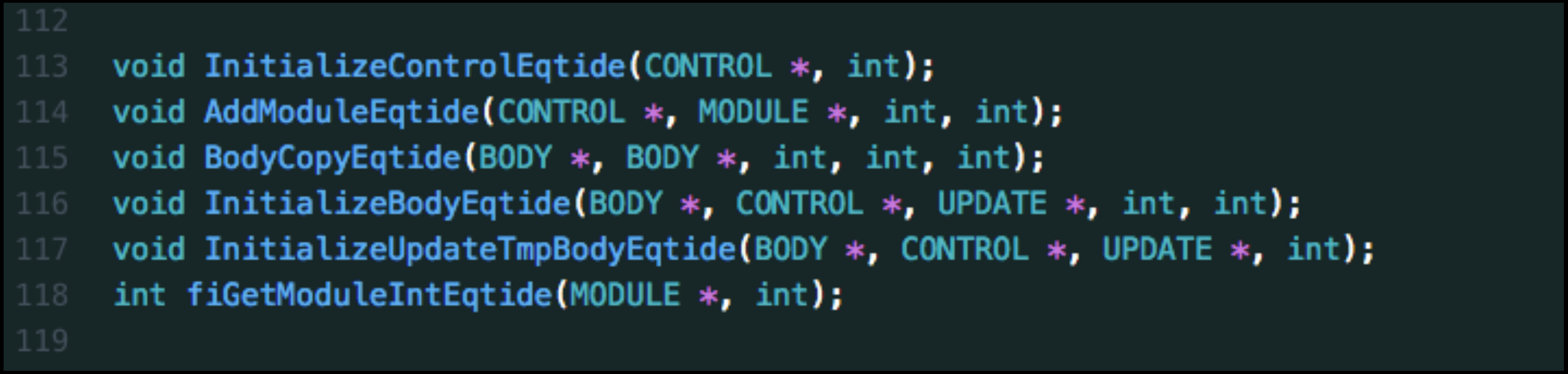
The Header File

Header files contains macros and prototype functions for your .c file



```c
/* Tidal Model */

#define CPL 0
#define CTL 1
#define DB15 2

/* Options Info */

#define OPTSTARTEQTIDE 1000 /* Start of Eqtide options */
#define OPTENDEQTIDE 1100   /* End of Eqtide options */

#define OPT_USETIDALRADIUS 1001
```

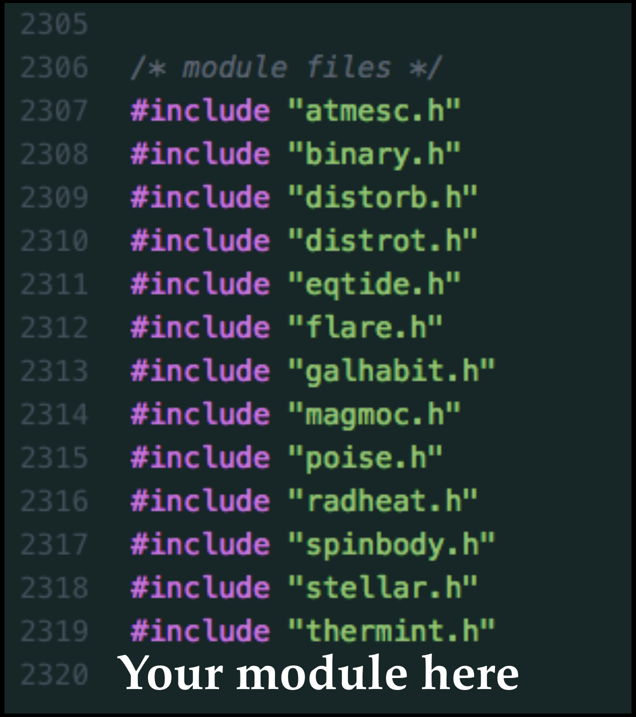
Any function called from another file, must be prototyped in the header



```c

void InitializeControlEqtide(CONTROL *, int);
void AddModuleEqtide(CONTROL *, MODULE *, int, int);
void BodyCopyEqtide(BODY *, BODY *, int, int, int);
void InitializeBodyEqtide(BODY *, CONTROL *, UPDATE *, int, int);
void InitializeUpdateTmpBodyEqtide(BODY *, CONTROL *, UPDATE *, int);
int fiGetModuleIntEqtide(MODULE *, int);

```

The Header File

Then add your module's header file to vplanet.h (at the bottom)



```
/* module files */
#include "atmesc.h"
#include "binary.h"
#include "distorb.h"
#include "distrot.h"
#include "eqtide.h"
#include "flare.h"
#include "galhabit.h"
#include "magmoc.h"
#include "poise.h"
#include "radheat.h"
#include "spinbody.h"
#include "stellar.h"
#include "thermint.h"
```

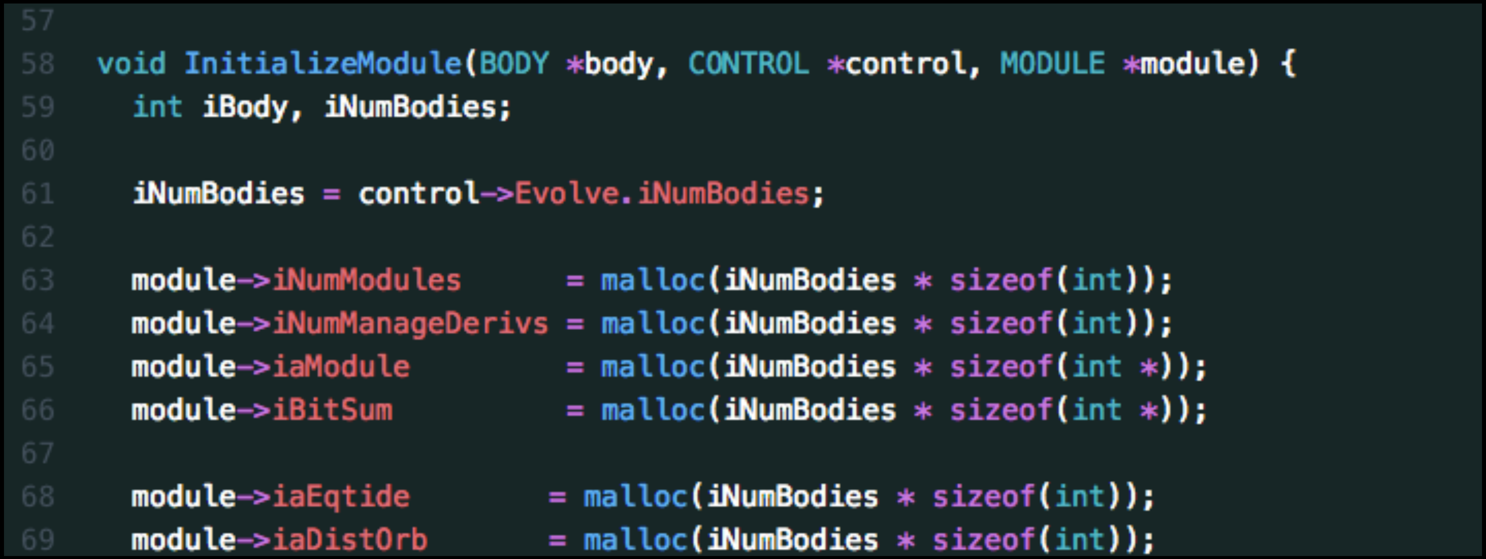
Your module here

Each module's .c file only needs to include vplanet.h

Updating module.c

Next up, we must add your module to module.c

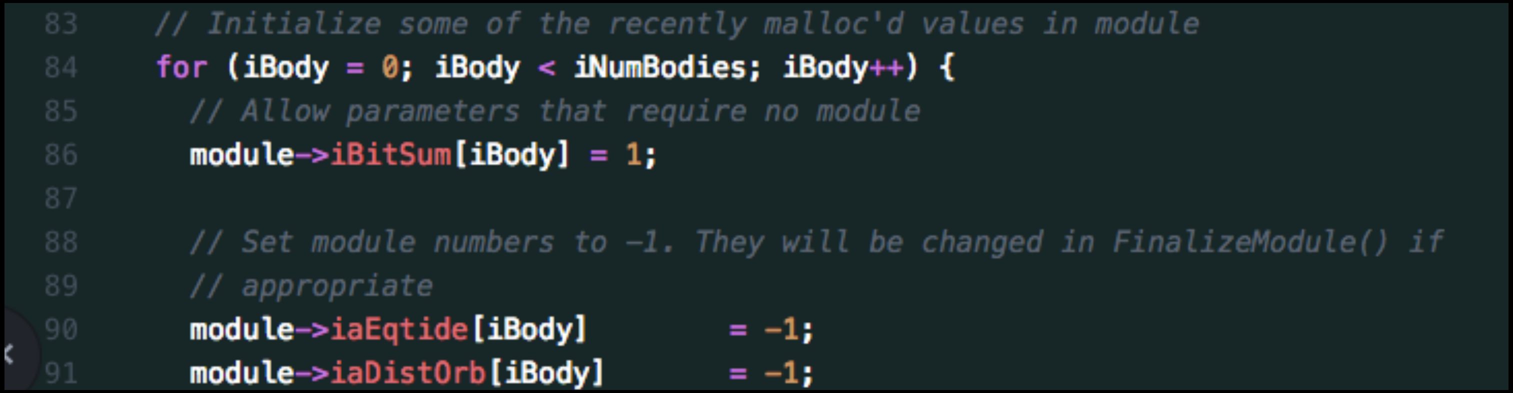
Allocate memory and initialize values in InitializeModule



```c

void InitializeModule(BODY *body, CONTROL *control, MODULE *module) {
  int iBody, iNumBodies;

  iNumBodies = control->Evolve.iNumBodies;

  module->iNumModules       = malloc(iNumBodies * sizeof(int));
  module->iNumManageDerivs = malloc(iNumBodies * sizeof(int));
  module->iaModule          = malloc(iNumBodies * sizeof(int *));
  module->iBitSum           = malloc(iNumBodies * sizeof(int *));

  module->iaEqtide         = malloc(iNumBodies * sizeof(int));
  module->iaDistOrb        = malloc(iNumBodies * sizeof(int));
```

Your module here

Set these arrays to -1; default is your module was *not* selected

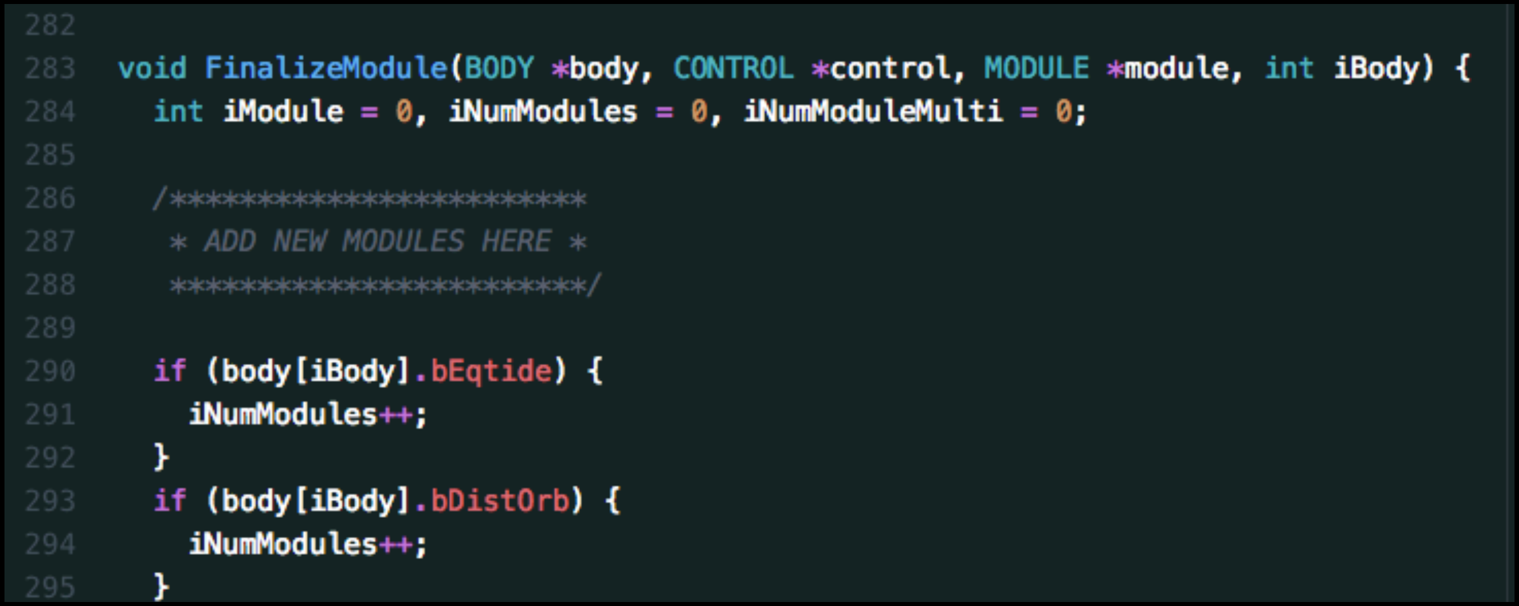


```c
    // Initialize some of the recently malloc'd values in module
    for (iBody = 0; iBody < iNumBodies; iBody++) {
      // Allow parameters that require no module
      module->iBitSum[iBody] = 1;

      // Set module numbers to -1. They will be changed in FinalizeModule() if
      // appropriate
      module->iaEqtide[iBody]        = -1;
      module->iaDistOrb[iBody]       = -1;
```

Your module here

Updating module.c

Add your module to FinalizeModule

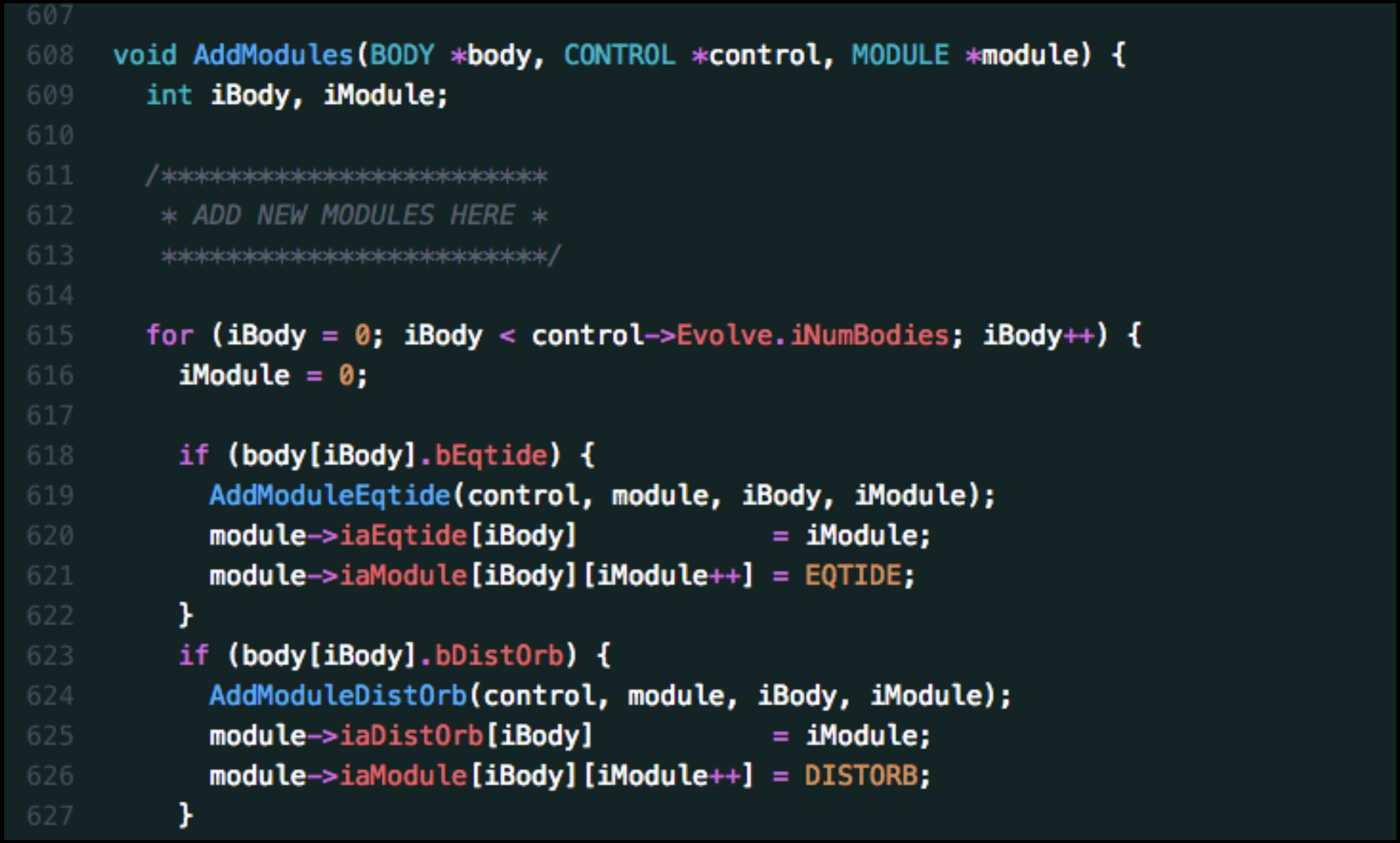


```

void FinalizeModule(BODY *body, CONTROL *control, MODULE *module, int iBody) {
  int iModule = 0, iNumModules = 0, iNumModuleMulti = 0;

  /************************
   * ADD NEW MODULES HERE *
   ************************/

  if (body[iBody].bEqtide) {
    iNumModules++;
  }
  if (body[iBody].bDistOrb) {
    iNumModules++;
  }
```

Your module here

Updating module.c

And to AddModules

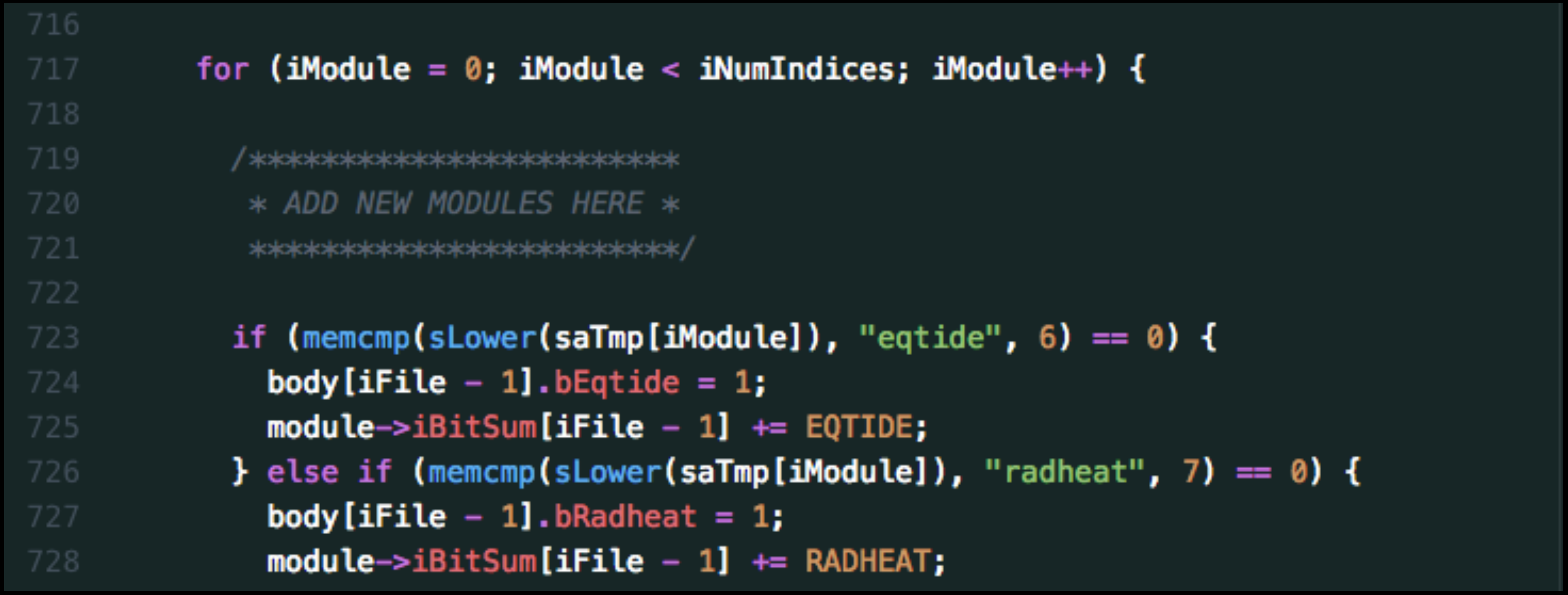


```c

void AddModules(BODY *body, CONTROL *control, MODULE *module) {
  int iBody, iModule;

  /************************
   * ADD NEW MODULES HERE *
   ************************/

  for (iBody = 0; iBody < control->Evolve.iNumBodies; iBody++) {
    iModule = 0;

    if (body[iBody].bEqtide) {
      AddModuleEqtide(control, module, iBody, iModule);
      module->iaEqtide[iBody]            = iModule;
      module->iaModule[iBody][iModule++] = EQTIDE;
    }
    if (body[iBody].bDistOrb) {
      AddModuleDistOrb(control, module, iBody, iModule);
      module->iaDistOrb[iBody]           = iModule;
      module->iaModule[iBody][iModule++] = DISTORB;
    }
```

Your module here

Updating module.c

Then add your module to ReadModules (a ReadOption in module.c)

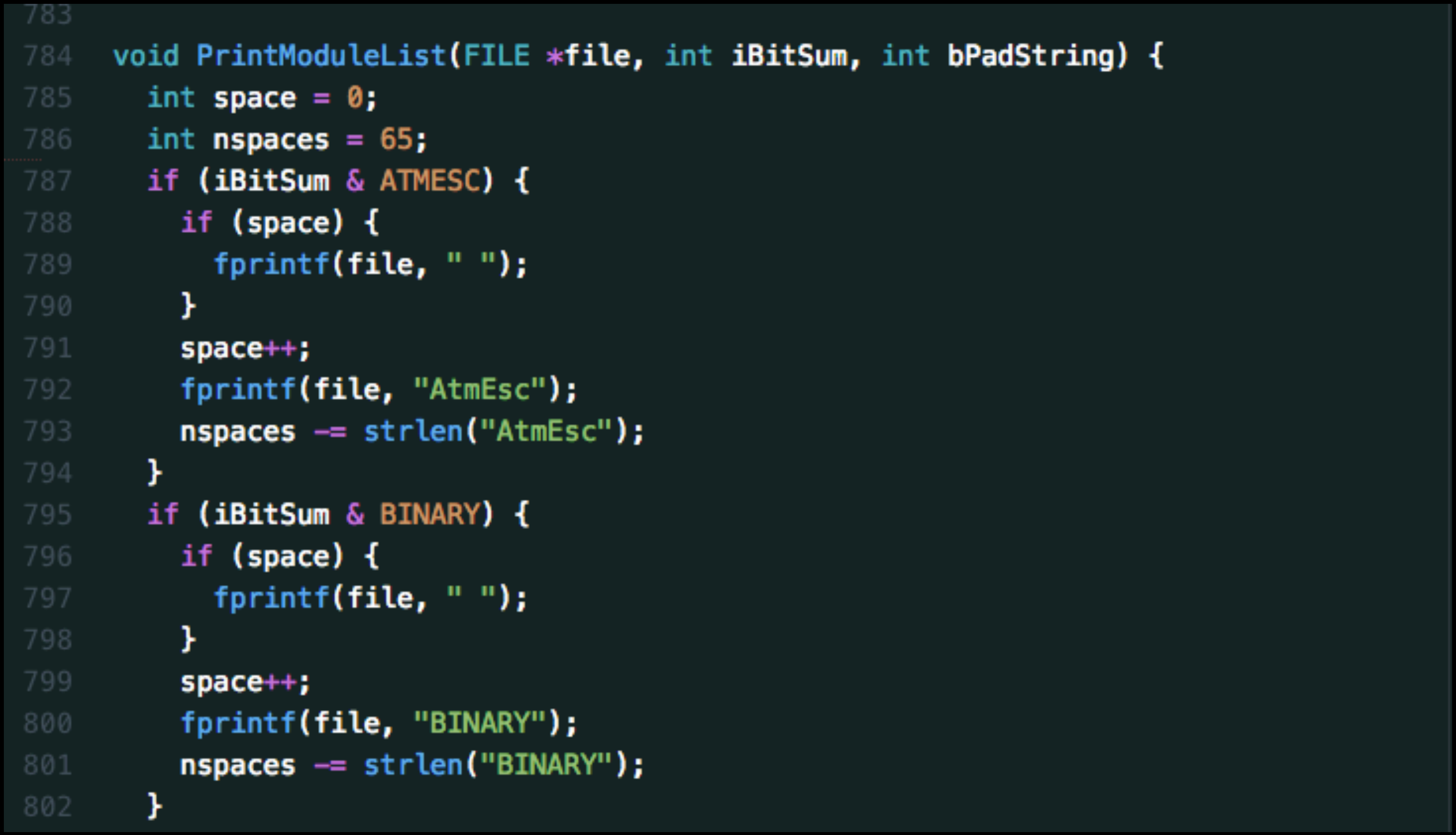


```
      for (iModule = 0; iModule < iNumIndices; iModule++) {

        /************************
         * ADD NEW MODULES HERE *
         ************************/

        if (memcmp(sLower(saTmp[iModule]), "eqtide", 6) == 0) {
          body[iFile - 1].bEqtide = 1;
          module->iBitSum[iFile - 1] += EQTIDE;
        } else if (memcmp(sLower(saTmp[iModule]), "radheat", 7) == 0) {
          body[iFile - 1].bRadheat = 1;
          module->iBitSum[iFile - 1] += RADHEAT;
```

Your module here

Updating module.c

Then add it to PrintModuleList

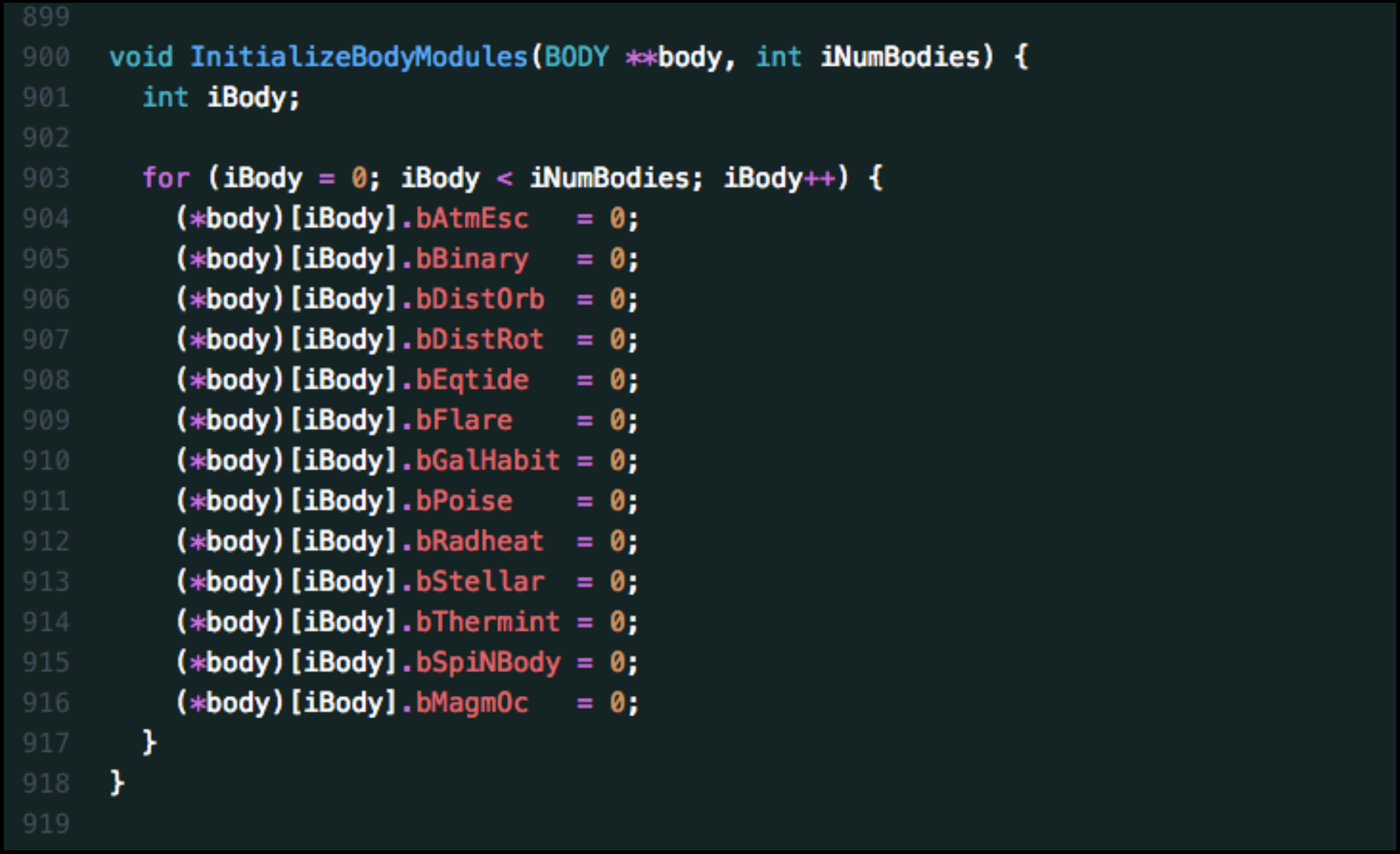


```
void PrintModuleList(FILE *file, int iBitSum, int bPadString) {
  int space = 0;
  int nspaces = 65;
  if (iBitSum & ATMESC) {
    if (space) {
      fprintf(file, " ");
    }
    space++;
    fprintf(file, "AtmEsc");
    nspaces -= strlen("AtmEsc");
  }
  if (iBitSum & BINARY) {
    if (space) {
      fprintf(file, " ");
    }
    space++;
    fprintf(file, "BINARY");
    nspaces -= strlen("BINARY");
  }
```

Your module here

Updating module.c

Finally, add it to InitializeBodyModules



```c

void InitializeBodyModules(BODY **body, int iNumBodies) {
  int iBody;

  for (iBody = 0; iBody < iNumBodies; iBody++) {
    (*body)[iBody].bAtmEsc    = 0;
    (*body)[iBody].bBinary    = 0;
    (*body)[iBody].bDistOrb   = 0;
    (*body)[iBody].bDistRot   = 0;
    (*body)[iBody].bEqtide    = 0;
    (*body)[iBody].bFlare     = 0;
    (*body)[iBody].bGalHabit  = 0;
    (*body)[iBody].bPoise     = 0;
    (*body)[iBody].bRadheat   = 0;
    (*body)[iBody].bStellar   = 0;
    (*body)[iBody].bThermint  = 0;
    (*body)[iBody].bSpiNBody  = 0;
    (*body)[iBody].bMagmOc    = 0;
  }
}

```

Update vplanet.h

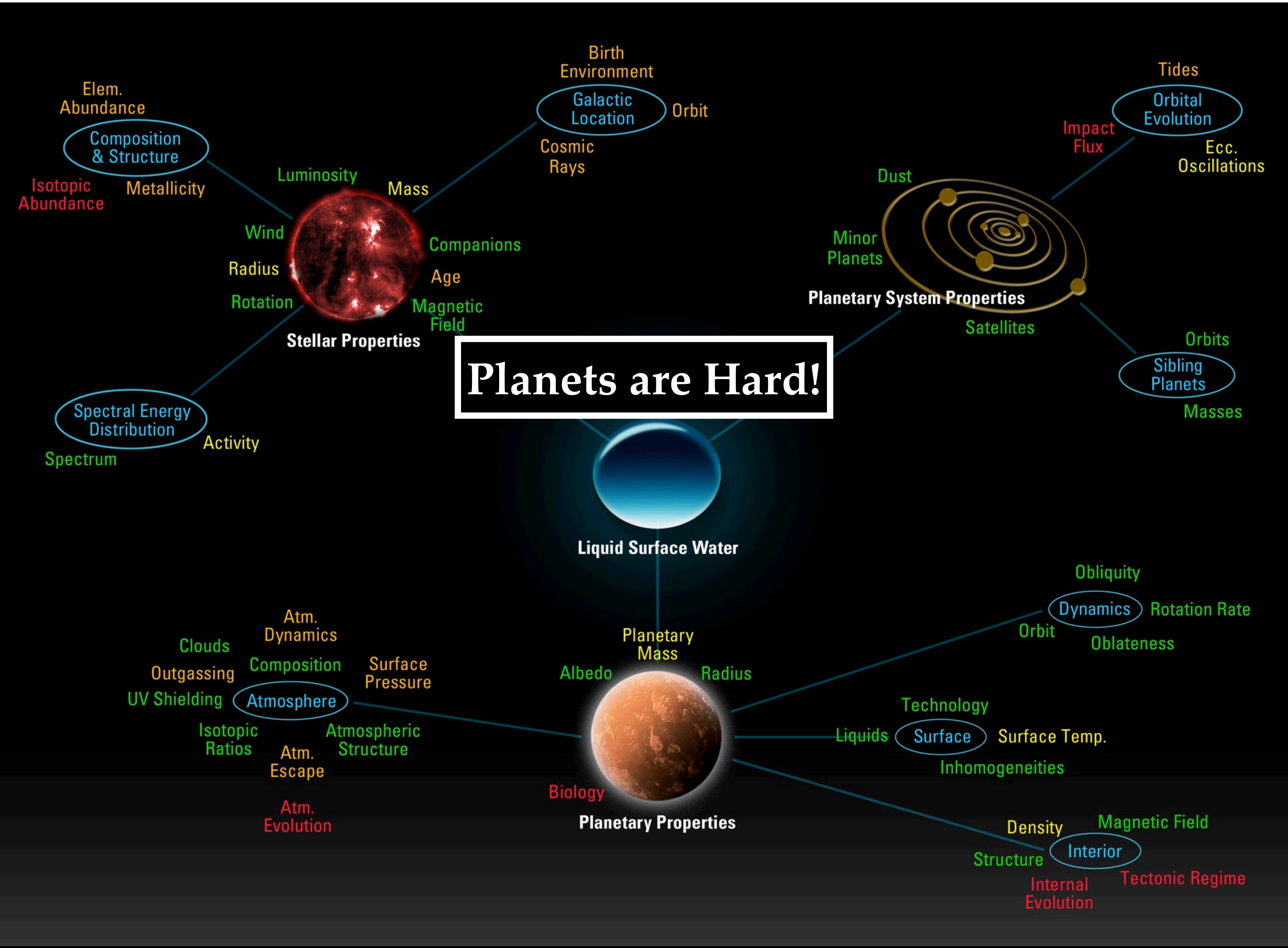
In practice, you've probably been doing this all along…

But you of course need to update the structs in vplanet.h

- BODY
- UPDATE
- MODULE
- CONTROL
- SYSTEM

That's it! Whew! It's obviously a lot of work to add a module
But you know what they say…

Planets are Hard!



HUGE THANKS!

Primary Developers

Rodrigo Luger (Flatiron Institute)	Thomas Quinn (UW)
Russell Deitrick (U. of Bern)	Rudy Garcia (UW)
Peter Driscoll (Carnegie)	Hayden Smotherman (UW)
David Fleming (Beyer)	Laura Amaral (UNAM)
Caitlyn Wilhelm (UW)	Patrick Barth (St. Andrews)

Contributors and Scientific Advisors

Victoria Meadows (UW)	Shawn Domagal-Goldman (Goddard)
Cecilia Bitz (UW)	John Armstrong (Weber St)
Ludmila Carone (Heidelberg)	Paul Moliere (Heidelberg)
Diego McDonald (UW)	Antigona Segura (UNAM)
Benjamin Guyer (UW)	Lena Noack (FU Berlin)
Pramod Gupta (UW)	Billy Quarles (Georgia St)