hypredrive

0.1

Generated by Doxygen 1.9.1

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Module Documentation

3.1 HYPREDRV

Typedefs

- typedef struct hypredrv_struct * **HYPREDRV_t**

Functions

- uint32_t HYPREDRV_Create (MPI_Comm comm, HYPREDRV_t *obj_ptr)

  Create a HYPREDRV object.
- uint32_t HYPREDRV_Destroy (HYPREDRV_t *obj_ptr)

  Destroy a HYPREDRV object.
- uint32_t HYPREDRV_PrintLibInfo (void)

  Print library information at entrance.
- uint32_t HYPREDRV_PrintExitInfo (const char *argv0)

  Print library information at exit.
- uint32_t HYPREDRV_InputArgsParse (int argc, char **argv, HYPREDRV_t obj)

  Parse input arguments for a HYPREDRV object.
- uint32_t HYPREDRV_SetGlobalOptions (HYPREDRV_t obj)

  Set HYPRE's global options according to the YAML input.
- int HYPREDRV_InputArgsGetWarmup (HYPREDRV_t obj)

  Retrieve the warmup setting from a HYPREDRV object.
- int HYPREDRV_InputArgsGetNumRepetitions (HYPREDRV_t obj)

  Retrieve the number of repetitions from a HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemBuild (HYPREDRV_t obj)

  Build the linear system (matrix, RHS, and LHS) according to the YAML input passed to the HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemReadMatrix (HYPREDRV_t obj)

  Read the linear system matrix from file for a HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemSetRHS (HYPREDRV_t obj)

  Set the linear system right-hand side (RHS) vector from file for a HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemSetInitialGuess (HYPREDRV_t obj)

  Set the initial guess for the solution vector of the linear system for a HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemResetInitialGuess (HYPREDRV_t obj)

Reset the initial guess of the solution vector for a HYPREDRV object to its original state as computed with HYPREDRV_LinearSystemSetInitialGuess.

- uint32_t HYPREDRV_LinearSystemSetPrecMatrix (HYPREDRV_t obj)

  Set the matrix that is used to compute the preconditioner of a HYPREDRV object.
- uint32_t HYPREDRV_LinearSystemReadDofmap (HYPREDRV_t obj)

  Read the degree of freedom (DOF) map for the linear system of a HYPREDRV object.
- uint32_t HYPREDRV_PreconCreate (HYPREDRV_t obj)

  Create a preconditioner for the HYPREDRV object based on the specified method.
- uint32_t HYPREDRV_LinearSolverCreate (HYPREDRV_t obj)

  Create a linear solver for the HYPREDRV object based on the specified method.
- uint32_t HYPREDRV_LinearSolverSetup (HYPREDRV_t obj)

  Set up the linear solver for the HYPREDRV object based on the specified solver and preconditioner methods.
- uint32_t HYPREDRV_LinearSolverApply (HYPREDRV_t obj)

  Apply the linear solver to solve the linear system for the HYPREDRV object.
- uint32_t HYPREDRV_PreconDestroy (HYPREDRV_t obj)

  Destroy the preconditioner associated with the HYPREDRV object.
- uint32_t HYPREDRV_LinearSolverDestroy (HYPREDRV_t obj)

  Destroy the linear solver associated with the HYPREDRV object.
- uint32_t HYPREDRV_StatsPrint (HYPREDRV_t obj)

  Print the statistics associated with the HYPREDRV object.

3.1.1 Detailed Description

Public APIs to solve linear systems with hypre through hypredrive

3.1.2 Function Documentation

3.1.2.1 HYPREDRV_Create()

```
uint32_t HYPREDRV_Create (
             MPI_Comm comm,
             HYPREDRV_t * obj_ptr )
```

Create a HYPREDRV object.

This function allocates memory for a HYPREDRV object and initializes it with the provided MPI communicator. The newly created object is returned through the obj_ptr parameter.

Parameters

comm	The MPI communicator to be associated with the HYPREDRV object. This communicator is used for parallel communications in the underlying HYPRE library calls.
obj_ptr	A pointer to the HYPREDRV_t pointer where the address of the newly allocated HYPREDRV object will be stored. After the function call, ∗obj_ptr will point to the allocated object.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It is the caller's responsibility to ensure that the MPI environment is properly initialized before calling this function. The function does not initialize or finalize the MPI environment.

Example Usage:

```
HYPREDRV_t *obj;
MPI_Comm comm = MPI_COMM_WORLD; // or any other valid MPI_Comm
uint32_t errorCode = HYPREDRV_Create(comm, &obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.2 HYPREDRV_Destroy()

```
uint32_t HYPREDRV_Destroy (
             HYPREDRV_t * obj_ptr )
```

Destroy a HYPREDRV object.

This function deallocates the memory for a HYPREDRV object and performs necessary cleanup for its associated resources. It destroys HYPRE matrices and vectors created and used by the object, deallocates any input arguments, and finally frees the HYPREDRV object itself. After this function is called, the pointer to the HYPREDRV object is set to NULL to prevent any dangling references.

Parameters

obj_ptr	A pointer to the HYPREDRV_t object that is to be destroyed. This pointer should have been initialized by HYPREDRV_Create function.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It is the caller's responsibility to ensure that obj_ptr is a valid pointer to a HYPREDRV_t object. Passing an invalid pointer or a pointer to an already destroyed object can lead to undefined behavior.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created and used) ...
uint32_t errorCode = HYPREDRV_Destroy(&obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
// obj is now NULL.
```

3.1.2.3 HYPREDRV_InputArgsGetNumRepetitions()

```
int HYPREDRV_InputArgsGetNumRepetitions (
              HYPREDRV_t obj )
```

Retrieve the number of repetitions from a HYPREDRV object.

This function accesses the HYPREDRV object's input arguments structure to retrieve the number of repetitions setting. This setting specifies how many times the linear systems should be solved, potentially for benchmarking or testing purposes.

Parameters

obj	The HYPREDRV_t object from which the number of repetitions is to be retrieved.

Returns

Returns the number of repetitions as an integer. If the input object is NULL or not properly initialized, the function returns -1 to indicate an error or invalid state.

Example Usage:

```
HYPREDRV_t obj;
// ... (obj is created, and its input arguments are set) ...
int numRepetitions = HYPREDRV_InputArgsGetNumRepetitions(obj);
if (numRepetitions != -1) {
    printf("Number of Repetitions: %d\n", numRepetitions);
    // Use numRepetitions as needed ...
}
```

3.1.2.4 HYPREDRV_InputArgsGetWarmup()

```
int HYPREDRV_InputArgsGetWarmup (
              HYPREDRV_t obj )
```

Retrieve the warmup setting from a HYPREDRV object.

This function accesses the HYPREDRV object's input arguments structure to retrieve the warmup setting. This setting indicates whether a warmup phase should be executed before the main operations, often used to ensure accurate timing measurements by eliminating any initialization overhead.

Parameters

obj	The HYPREDRV_t object from which the warmup setting is to be retrieved.

Returns

Returns the warmup setting as an integer. If the input object is NULL or not properly initialized, the function returns -1 to indicate an error or invalid state.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its input arguments are set) ...
```

```
int warmupSetting = HYPREDRV_InputArgsGetWarmup(obj);
if (warmupSetting != -1) {
    printf("Warmup Setting: %d\n", warmupSetting);
    // Use warmupSetting as needed ...
}
```

3.1.2.5 HYPREDRV_InputArgsParse()

```
uint32_t HYPREDRV_InputArgsParse (
             int argc,
             char ** argv,
             HYPREDRV_t obj )
```

Parse input arguments for a HYPREDRV object.

This function is responsible for parsing the command-line arguments provided to the application.

Parameters

argc	The count of command-line arguments.
argv	The array of command-line argument strings.
obj	The HYPREDRV_t object whose input arguments are to be parsed and set.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It is expected that argv[1] is the name of the input file in YAML format

Example Usage:

```
HYPREDRV_t obj;
int argc = ...; // Number of arguments
char **argv = ...; // Argument strings
uint32_t errorCode = HYPREDRV_InputArgsParse(argc, argv, obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.6 HYPREDRV_LinearSolverApply()

```
uint32_t HYPREDRV_LinearSolverApply (
             HYPREDRV_t obj )
```

Apply the linear solver to solve the linear system for the HYPREDRV object.

Parameters

obj	The HYPREDRV_t object for which the linear solver is to be applied.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the solver method, solver, matrix, RHS vector, and solution vector are properly set in the input arguments.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSolverApply(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.7 HYPREDRV_LinearSolverCreate()

```
uint32_t HYPREDRV_LinearSolverCreate (
             HYPREDRV_t obj )
```

Create a linear solver for the HYPREDRV object based on the specified method.

Parameters

obj	The HYPREDRV_t object for which the linear solver is to be created.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the solver method and other related settings are properly set in the input arguments.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSolverCreate(obj);
if (errorCode != 0) {
```

```
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.8 HYPREDRV_LinearSolverDestroy()

```
uint32_t HYPREDRV_LinearSolverDestroy (
             HYPREDRV_t obj )
```

Destroy the linear solver associated with the HYPREDRV object.

Parameters

obj	The HYPREDRV_t object whose linear solver is to be destroyed.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the linear solver method and the linear solver object itself are properly set in the HYPREDRV_t object.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSolverDestroy(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.9 HYPREDRV_LinearSolverSetup()

```
uint32_t HYPREDRV_LinearSolverSetup (
             HYPREDRV_t obj )
```

Set up the linear solver for the HYPREDRV object based on the specified solver and preconditioner methods.

Parameters

obj	The HYPREDRV_t object for which the linear solver is to be set up.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the solver and preconditioner methods, as well as the matrix, RHS vector, and solution vector, are properly set in the input arguments.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSolverSetup(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.10 HYPREDRV_LinearSystemBuild()

```
uint32_t HYPREDRV_LinearSystemBuild (
              HYPREDRV_t obj )
```

Build the linear system (matrix, RHS, and LHS) according to the YAML input passed to the HYPREDRV object.

The matrix is read from file. Vectors might be read from file or built according to predetermined options passed via YAML.

Parameters

obj	The HYPREDRV_t object for which the linear system matrix is to be built.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemBuild(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.11 HYPREDRV_LinearSystemReadDofmap()

```
uint32_t HYPREDRV_LinearSystemReadDofmap (
             HYPREDRV_t obj )
```

Read the degree of freedom (DOF) map for the linear system of a HYPREDRV object.

Parameters

obj	The HYPREDRV_t object for which the DOF map of the linear system is to be read.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemReadDofmap(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.12 HYPREDRV_LinearSystemReadMatrix()

```
uint32_t HYPREDRV_LinearSystemReadMatrix (
             HYPREDRV_t obj )
```

Read the linear system matrix from file for a HYPREDRV object.

This function is responsible for reading the matrix data of a linear system associated with a HYPREDRV object. It performs the reading process given input arguments related to the linear system, and uses a pointer to store the read matrix. After reading the matrix, it retrieves and returns the error code generated during the process.

Parameters

obj	The HYPREDRV_t object for which the linear system matrix is to be read.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemReadMatrix(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.13 HYPREDRV_LinearSystemResetInitialGuess()

```
uint32_t HYPREDRV_LinearSystemResetInitialGuess (
             HYPREDRV_t obj )
```

Reset the initial guess of the solution vector for a HYPREDRV object to its original state as computed with *HYPREDRV_LinearSystemSetInitialGuess*.

Parameters

obj	The HYPREDRV_t object for which the initial guess of the solution vector is to be reset.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemResetInitialGuess(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.14 HYPREDRV_LinearSystemSetInitialGuess()

```
uint32_t HYPREDRV_LinearSystemSetInitialGuess (
             HYPREDRV_t obj )
```

Set the initial guess for the solution vector of the linear system for a HYPREDRV object.

This function is responsible for setting the initial guess for the solution vector of a linear system associated with a HYPREDRV object.

Parameters

obj	The HYPREDRV_t object for which the initial guess of the solution vector of the linear system is to be set.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemSetInitialGuess(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.15 HYPREDRV_LinearSystemSetPrecMatrix()

```
uint32_t HYPREDRV_LinearSystemSetPrecMatrix (
             HYPREDRV_t obj )
```

Set the matrix that is used to compute the preconditioner of a HYPREDRV object.

By default, the preconditioning matrix is the same as the linear system matrix. However, it is also possible to use an arbitrary matrix set by the user, e.g., a filtered version of the linear system matrix.

Parameters

obj	The HYPREDRV_t object for which the preconditioner matrix is to be set.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemSetPrecMatrix(obj);
```

```
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.16 HYPREDRV_LinearSystemSetRHS()

```
uint32_t HYPREDRV_LinearSystemSetRHS (
              HYPREDRV_t obj )
```

Set the linear system right-hand side (RHS) vector from file for a HYPREDRV object.

Parameters

obj	The HYPREDRV_t object for which the RHS vector of the linear system is to be set.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_LinearSystemSetRHS(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.17 HYPREDRV_PreconCreate()

```
uint32_t HYPREDRV_PreconCreate (
              HYPREDRV_t obj )
```

Create a preconditioner for the HYPREDRV object based on the specified method.

Parameters

obj	The HYPREDRV_t object for which the preconditioner is to be created.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the preconditioning method and other related settings are properly set in the input arguments.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_PreconCreate(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.18 HYPREDRV_PreconDestroy()

```
uint32_t HYPREDRV_PreconDestroy (
             HYPREDRV_t obj )
```

Destroy the preconditioner associated with the HYPREDRV object.

Parameters

obj	The HYPREDRV_t object whose preconditioner is to be destroyed.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error. The function assumes that the preconditioner method and the preconditioner itself are properly set in the HYPREDRV_t object.

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_PreconDestroy(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.19 HYPREDRV_PrintExitInfo()

```
uint32_t HYPREDRV_PrintExitInfo (
             const char * argv0 )
```

Print library information at exit.

This function prints the driver name followed by the current date and time.

Note

This function is intended to be used just before finishing the driver application

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Example Usage:

```
uint32_t errorCode = HYPREDRV_PrintExitInfo();
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.20 HYPREDRV_PrintLibInfo()

```
uint32_t HYPREDRV_PrintLibInfo (
             void  )
```

Print library information at entrance.

This function prints the current date and time, followed by version information about the HYPRE library.

Note

This function uses conditional compilation to determine what information about the HYPRE library to print.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Example Usage:

```
uint32_t errorCode = HYPREDRV_PrintLibInfo();
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.21 HYPREDRV_SetGlobalOptions()

```
uint32_t HYPREDRV_SetGlobalOptions (
             HYPREDRV_t obj )
```

Set HYPRE's global options according to the YAML input.

Parameters

obj	The HYPREDRV_t object from which the global options are to be retrieved.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Example Usage:

```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_SetGlobalOptions(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

3.1.2.22 HYPREDRV_StatsPrint()

```
uint32_t HYPREDRV_StatsPrint (
              HYPREDRV_t obj )
```

Print the statistics associated with the HYPREDRV object.

This function is responsible for printing the statistics collected during the operations performed by the HYPREDRV object.

Parameters

obj	The HYPREDRV_t object whose statistics are to be printed.

Returns

Returns an error code with 0 indicating success. Any non-zero value indicates a failure, and the error code can be further described using HYPREDRV_ErrorCodeDescribe(error_code).

Note

It's the caller's responsibility to ensure that the obj parameter is a valid pointer to an initialized HYPREDRV_t object. Passing a NULL or uninitialized object will result in an error.

Example Usage:

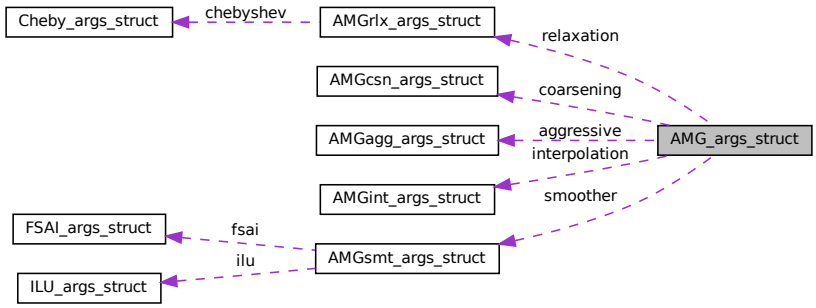
```
HYPREDRV_t *obj;
// ... (obj is created, and its components are initialized) ...
uint32_t errorCode = HYPREDRV_StatsPrint(obj);
if (errorCode != 0) {
    const char* errorDescription = HYPREDRV_ErrorCodeDescribe(errorCode);
    printf("%s\n", errorDescription);
    // Handle error
}
```

Chapter 4

Class Documentation

4.1 AMG_args_struct Struct Reference

Collaboration diagram for AMG_args_struct:



Public Attributes

- AMGint_args **interpolation**
- AMGagg_args **aggressive**
- AMGcsn_args **coarsening**
- AMGrlx_args **relaxation**
- AMGsmt_args **smoother**
- HYPRE_Int **max_iter**
- HYPRE_Int **print_level**
- HYPRE_Real **tolerance**

The documentation for this struct was generated from the following file:

- include/amg.h

4.2 AMGagg_args_struct Struct Reference

Public Attributes

- HYPRE_Int **num_levels**
- HYPRE_Int **num_paths**
- HYPRE_Int **prolongation_type**
- HYPRE_Int **max_nnz_row**
- HYPRE_Int **P12_max_elements**
- HYPRE_Real **P12_trunc_factor**
- HYPRE_Real **trunc_factor**

The documentation for this struct was generated from the following file:

- include/amg.h

4.3 AMGcsn_args_struct Struct Reference

Public Attributes

- HYPRE_Int **type**
- HYPRE_Int **rap2**
- HYPRE_Int **mod_rap2**
- HYPRE_Int **keep_transpose**
- HYPRE_Int **num_functions**
- HYPRE_Int **seq_amg_th**
- HYPRE_Int **min_coarse_size**
- HYPRE_Int **max_coarse_size**
- HYPRE_Int **max_levels**
- HYPRE_Real **max_row_sum**
- HYPRE_Real **strong_th**

The documentation for this struct was generated from the following file:

- include/amg.h

4.4 AMGint_args_struct Struct Reference

Public Attributes

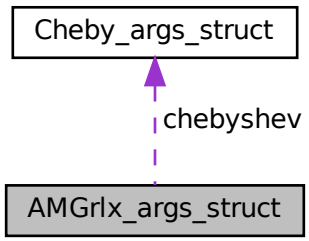
- HYPRE_Int **prolongation_type**
- HYPRE_Int **restriction_type**
- HYPRE_Int **max_nnz_row**
- HYPRE_Real **trunc_factor**

The documentation for this struct was generated from the following file:

- include/amg.h

4.5 AMGrlx_args_struct Struct Reference

Collaboration diagram for AMGrlx_args_struct:



Public Attributes

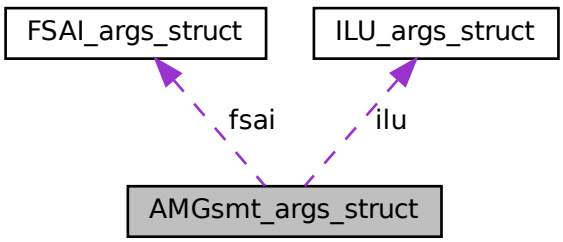
- HYPRE_Int **down_type**
- HYPRE_Int **up_type**
- HYPRE_Int **coarse_type**
- HYPRE_Int **down_sweeps**
- HYPRE_Int **up_sweeps**
- HYPRE_Int **coarse_sweeps**
- HYPRE_Int **num_sweeps**
- HYPRE_Int **order**
- HYPRE_Real **weight**
- HYPRE_Real **outer_weight**
- Cheby_args **chebyshev**

The documentation for this struct was generated from the following file:

- include/amg.h

4.6 AMGsmt_args_struct Struct Reference

Collaboration diagram for AMGsmt_args_struct:



Public Attributes

- HYPRE_Int **type**
- HYPRE_Int **num_levels**
- HYPRE_Int **num_sweeps**
- FSAI_args **fsai**
- ILU_args **ilu**

The documentation for this struct was generated from the following file:

- include/amg.h

4.7 BiCGSTAB_args_struct Struct Reference

Public Attributes

- HYPRE_Int **min_iter**
- HYPRE_Int **max_iter**
- HYPRE_Int **stop_crit**
- HYPRE_Int **logging**
- HYPRE_Int **print_level**
- HYPRE_Real **relative_tol**
- HYPRE_Real **absolute_tol**
- HYPRE_Real **conv_fac_tol**

The documentation for this struct was generated from the following file:

- include/bicgstab.h

4.8 Cheby_args_struct Struct Reference

Public Attributes

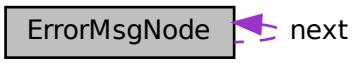
- HYPRE_Int **order**
- HYPRE_Int **eig_est**
- HYPRE_Int **variant**
- HYPRE_Int **scale**
- HYPRE_Real **fraction**

The documentation for this struct was generated from the following file:

- include/cheby.h

4.9 ErrorMsgNode Struct Reference

Collaboration diagram for ErrorMsgNode:



Public Attributes

- char * **message**
- struct ErrorMsgNode * **next**

The documentation for this struct was generated from the following file:

- src/error.c

4.10 FGMRES_args_struct Struct Reference

Public Attributes

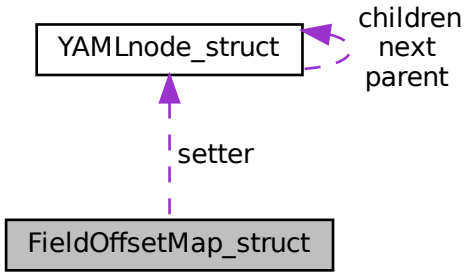
- HYPRE_Int **min_iter**
- HYPRE_Int **max_iter**
- HYPRE_Int **krylov_dim**
- HYPRE_Int **logging**
- HYPRE_Int **print_level**
- HYPRE_Real **relative_tol**
- HYPRE_Real **absolute_tol**

The documentation for this struct was generated from the following file:

- include/fgmres.h

4.11 FieldOffsetMap_struct Struct Reference

Collaboration diagram for FieldOffsetMap_struct:



Public Attributes

- const char * **name**
- size_t **offset**
- SetterFnc **setter**

The documentation for this struct was generated from the following file:

- include/field.h

4.12 FSAI_args_struct Struct Reference

Public Attributes

- HYPRE_Int **max_iter**
- HYPRE_Int **print_level**
- HYPRE_Int **algo_type**
- HYPRE_Int **ls_type**
- HYPRE_Int **max_steps**
- HYPRE_Int **max_step_size**
- HYPRE_Int **max_nnz_row**
- HYPRE_Int **num_levels**
- HYPRE_Int **eig_max_iters**
- HYPRE_Real **threshold**
- HYPRE_Real **kap_tolerance**
- HYPRE_Real **tolerance**

The documentation for this struct was generated from the following file:

- include/fsai.h

4.13 GMRES_args_struct Struct Reference

Public Attributes

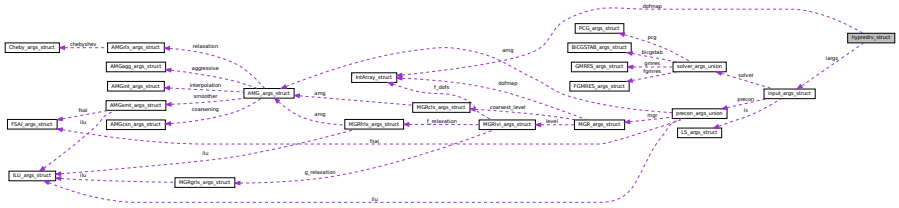
- HYPRE_Int **min_iter**
- HYPRE_Int **max_iter**
- HYPRE_Int **stop_crit**
- HYPRE_Int **skip_real_res_check**
- HYPRE_Int **krylov_dim**
- HYPRE_Int **rel_change**
- HYPRE_Int **logging**
- HYPRE_Int **print_level**
- HYPRE_Real **relative_tol**
- HYPRE_Real **absolute_tol**
- HYPRE_Real **conv_fac_tol**

The documentation for this struct was generated from the following file:

- include/gmres.h

4.14 hypredrv_struct Struct Reference

Collaboration diagram for hypredrv_struct:



Public Attributes

- MPI_Comm **comm**
- input_args ∗ **iargs**
- IntArray ∗ **dofmap**
- HYPRE_IJMatrix **mat_A**
- HYPRE_IJMatrix **mat_M**
- HYPRE_IJVector **vec_b**
- HYPRE_IJVector **vec_x**
- HYPRE_IJVector **vec_x0**
- HYPRE_Solver **precon**
- HYPRE_Solver **solver**

The documentation for this struct was generated from the following file:

- src/hypredrive.c

4.15 ILU_args_struct Struct Reference

Public Attributes

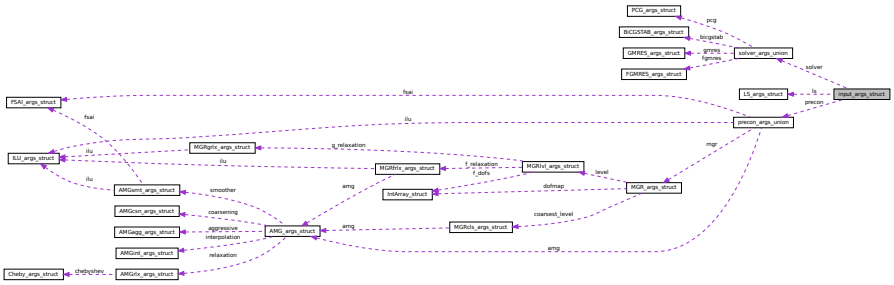
- HYPRE_Int **max_iter**
- HYPRE_Int **print_level**
- HYPRE_Int **type**
- HYPRE_Int **fill_level**
- HYPRE_Int **reordering**
- HYPRE_Int **tri_solve**
- HYPRE_Int **lower_jac_iters**
- HYPRE_Int **upper_jac_iters**
- HYPRE_Int **max_row_nnz**
- HYPRE_Int **schur_max_iter**
- HYPRE_Real **droptol**
- HYPRE_Real **nsh_droptol**
- HYPRE_Real **tolerance**

The documentation for this struct was generated from the following file:

- include/ilu.h

4.16 input_args_struct Struct Reference

Collaboration diagram for input_args_struct:



Public Attributes

- int **warmup**
- int **statistics**
- int **num_repetitions**
- LS_args **ls**
- solver_args **solver**
- solver_t **solver_method**
- precon_args **precon**
- precon_t **precon_method**

The documentation for this struct was generated from the following file:

- include/args.h

4.17 IntArray_struct Struct Reference

Public Attributes

- int ∗ **data**
- size_t **size**
- size_t **num_unique_entries**

The documentation for this struct was generated from the following file:

- include/containers.h

4.18 LS_args_struct Struct Reference

Public Attributes

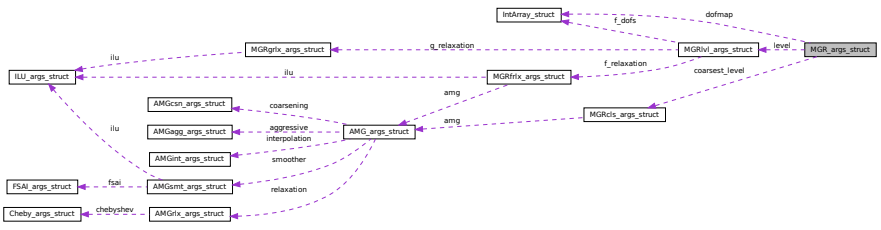
- char **matrix_filename** [MAX_FILENAME_LENGTH]
- char **precmat_filename** [MAX_FILENAME_LENGTH]
- char **rhs_filename** [MAX_FILENAME_LENGTH]
- char **x0_filename** [MAX_FILENAME_LENGTH]
- char **sol_filename** [MAX_FILENAME_LENGTH]
- char **dofmap_filename** [MAX_FILENAME_LENGTH]
- HYPRE_Int **init_guess_mode**
- HYPRE_Int **rhs_mode**
- HYPRE_Int **type**
- HYPRE_Int **exec_policy**

The documentation for this struct was generated from the following file:

- include/linsys.h

4.19 MGR_args_struct Struct Reference

Collaboration diagram for MGR_args_struct:



Public Attributes

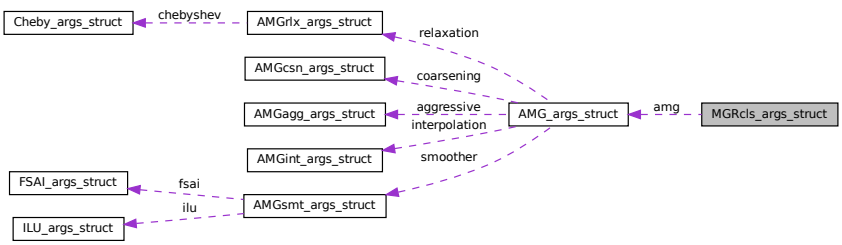
- IntArray * **dofmap**
- HYPRE_Int **non_c_to_f**
- HYPRE_Int **pmax**
- HYPRE_Int **max_iter**
- HYPRE_Int **num_levels**
- HYPRE_Int **relax_type**
- HYPRE_Int **print_level**
- HYPRE_Real **tolerance**
- HYPRE_Real **coarse_th**
- MGRlvl_args **level** [MAX_MGR_LEVELS - 1]
- MGRcls_args **coarsest_level**

The documentation for this struct was generated from the following file:

- include/mgr.h

4.20 MGRcls_args_struct Struct Reference

Collaboration diagram for MGRcls_args_struct:



Public Attributes

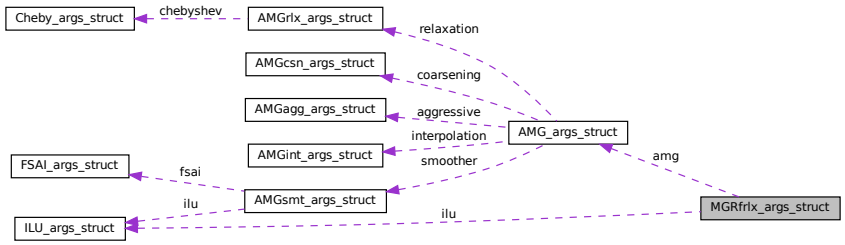
- HYPRE_Int **type**
- AMG_args **amg**

The documentation for this struct was generated from the following file:

- include/mgr.h

4.21 MGRfrlx_args_struct Struct Reference

Collaboration diagram for MGRfrlx_args_struct:



Public Attributes

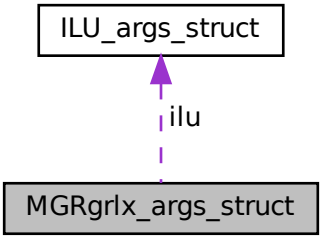
- HYPRE_Int **type**
- HYPRE_Int **num_sweeps**
- AMG_args **amg**
- ILU_args **ilu**

The documentation for this struct was generated from the following file:

- include/mgr.h

4.22 MGRgrlx_args_struct Struct Reference

Collaboration diagram for MGRgrlx_args_struct:



Public Attributes

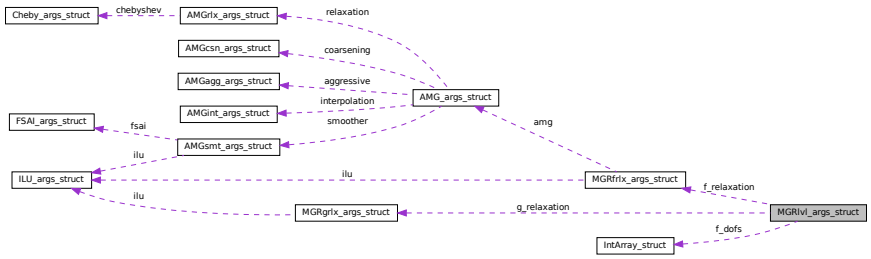
- HYPRE_Int **type**
- HYPRE_Int **num_sweeps**
- ILU_args **ilu**

The documentation for this struct was generated from the following file:

- include/mgr.h

4.23 MGRlvl_args_struct Struct Reference

Collaboration diagram for MGRlvl_args_struct:



Public Attributes

- IntArray ∗ **f_dofs**
- HYPRE_Int **prolongation_type**
- HYPRE_Int **restriction_type**
- HYPRE_Int **coarse_level_type**
- MGRfrlx_args **f_relaxation**
- MGRgrlx_args **g_relaxation**

The documentation for this struct was generated from the following file:

- include/mgr.h

4.24 PCG_args_struct Struct Reference

Public Attributes

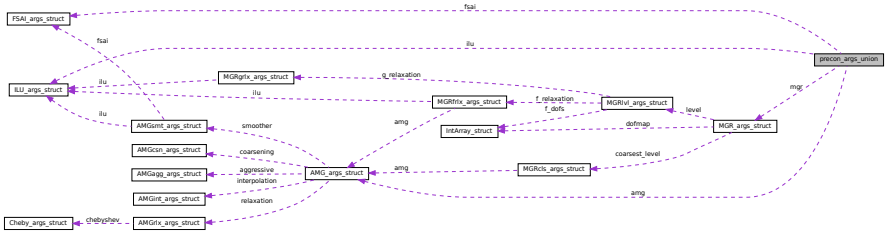
- HYPRE_Int **max_iter**
- HYPRE_Int **two_norm**
- HYPRE_Int **stop_crit**
- HYPRE_Int **rel_change**
- HYPRE_Int **print_level**
- HYPRE_Int **recompute_res**
- HYPRE_Real **relative_tol**
- HYPRE_Real **absolute_tol**
- HYPRE_Real **residual_tol**
- HYPRE_Real **conv_fac_tol**

The documentation for this struct was generated from the following file:

- include/pcg.h

4.25 precon_args_union Union Reference

Collaboration diagram for precon_args_union:



Public Attributes

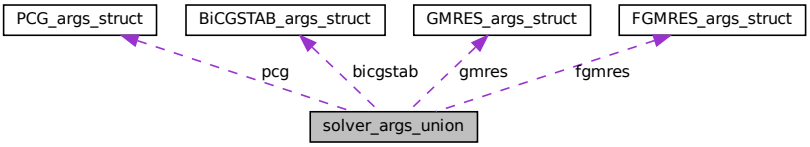
- AMG_args **amg**
- MGR_args **mgr**
- ILU_args **ilu**
- FSAI_args **fsai**

The documentation for this union was generated from the following file:

- include/precon.h

4.26 solver_args_union Union Reference

Collaboration diagram for solver_args_union:



Public Attributes

- PCG_args **pcg**
- GMRES_args **gmres**
- FGMRES_args **fgmres**
- BiCGSTAB_args **bicgstab**

The documentation for this union was generated from the following file:

- include/solver.h

4.27 Stats_struct Struct Reference

Public Attributes

- int **capacity** [STATS_NUM_ENTRIES]
- int **size** [STATS_NUM_ENTRIES]
- int **ls_counter**
- double * **matrix**
- double * **rhs**
- double * **dofmap**
- int * **iters**
- double * **prec**
- double * **solve**
- double * **rrnorms**
- double **initialize**
- double **finalize**

The documentation for this struct was generated from the following file:

- include/stats.h

4.28 StrArray_struct Struct Reference

Public Attributes

- const char ** **data**
- size_t **size**

The documentation for this struct was generated from the following file:

- include/containers.h

4.29 StrIntMap_struct Struct Reference

Public Attributes

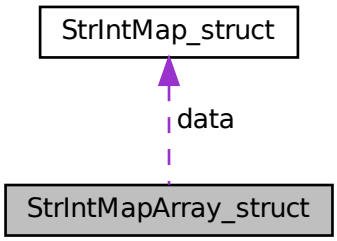
- const char * **str**
- int **num**

The documentation for this struct was generated from the following file:

- include/containers.h

4.30 StrIntMapArray_struct Struct Reference

Collaboration diagram for StrIntMapArray_struct:



Public Attributes

- const StrIntMap * **data**
- size_t **size**

The documentation for this struct was generated from the following file:

- include/containers.h

4.31 StrStrIntMap_struct Struct Reference

Public Attributes

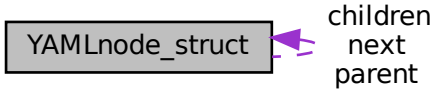
- const char * **strA**
- const char * **strB**
- int **num**

The documentation for this struct was generated from the following file:

- include/containers.h

4.32 YAMLnode_struct Struct Reference

Collaboration diagram for YAMLnode_struct:



Public Attributes

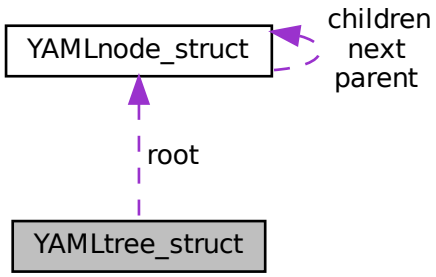
- char * **key**
- char * **val**
- char * **mapped_val**
- int **level**
- YAMLvalidity **valid**
- struct YAMLnode_struct * **parent**
- struct YAMLnode_struct * **children**
- struct YAMLnode_struct * **next**

The documentation for this struct was generated from the following file:

- include/yaml.h

4.33 YAMLtree_struct Struct Reference

Collaboration diagram for YAMLtree_struct:



Public Attributes

- YAMLnode * **root**

The documentation for this struct was generated from the following file:

- include/yaml.h

Index