EZ_PARALLEL

ALPHA 1.01

Generated by Doxygen 1.8.17

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 2

Data Type Index

2.1 Data Types List

Here are the data types with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 ez_parallel Module Reference

The EZ_PARALLEL module. Contains EZ_PARALLEL subroutines and their interfaces.

Data Types

- interface create_scheme

  The interface for the SCHEME creation subroutine.
- interface create_scheme_fft

  The interface for the SCHEME FFT initialization subroutine.
- interface create_scheme_spec_drv

  The interface for the SCHEME spectral derivative initialization subroutine.
- interface create_scheme_zero_pad

  The interface for the SCHEME zero-padding initializtion subroutine.
- interface destroy_scheme

  The interface for the SCHEME destruction subroutine.
- interface execute_scheme_fft

  The FFT execution interface for the EZ_PARALLEL module.
- interface execute_scheme_ifft

  The inverse FFT execution interface for the EZ_PARALLEL module.
- interface execute_scheme_ispec_drv

  The inverse spectral derivative execution interface for the EZ_PARALLEL module.
- interface execute_scheme_izero_pad

  The interface for the SCHEME zero-padding removal subroutine.
- interface execute_scheme_spec_drv

  The spectral derivative execution interface for the EZ_PARALLEL module.
- interface execute_scheme_zero_pad

  The interface for the SCHEME zero-padding execution subroutine.
- interface max_val

  The maximum value interface for the EZ_PARALLEL module.
- interface min_val

  The minimum value interface for the EZ_PARALLEL module.
- interface share_subgrid_bdry

  The sub-grid boundary communication interface for the EZ_PARALLEL module.

4.1.1 Detailed Description

The EZ_PARALLEL module. Contains EZ_PARALLEL subroutines and their interfaces.

Author

Jason Turner

4.2 ez_parallel_structs Module Reference

The EZ_PARALLEL structures module. Contains all EZ_PARALLEL derived datatypes.

Data Types

- type scheme

Variables

- integer, parameter, public fft_1d_1 = 51

  Flag to mark execution of 1D FFTs along the first dimension.
- integer, parameter, public fft_1d_2 = 46

  Flag to mark execution of 1D FFTs along the second dimension.
- integer, parameter, public fft_2d = 95

  Flag to mark execution of 2D FFTs.
- integer, parameter, public spec_drv_1d_1 = 23

  Flag to mark execution of 1D spectral derivatives along the first dimension.
- integer, parameter, public spec_drv_1d_2 = 78

  Flag to mark execution of 1D spectral derivatives along the second dimension.

4.2.1 Detailed Description

The EZ_PARALLEL structures module. Contains all EZ_PARALLEL derived datatypes.

Author

Jason Turner

4.2.2 Variable Documentation

4.2.2.1 fft_1d_1

```
integer, parameter, public ez_parallel_structs::fft_1d_1 = 51
```

Flag to mark execution of 1D FFTs along the first dimension.

Definition at line 93 of file ez_parallel_structs.f90.

4.2.2.2 fft_1d_2

```
integer, parameter, public ez_parallel_structs::fft_1d_2 = 46
```

Flag to mark execution of 1D FFTs along the second dimension.

Definition at line 96 of file ez_parallel_structs.f90.

4.2.2.3 fft_2d

```
integer, parameter, public ez_parallel_structs::fft_2d = 95
```

Flag to mark execution of 2D FFTs.

Definition at line 99 of file ez_parallel_structs.f90.

4.2.2.4 spec_drv_1d_1

```
integer, parameter, public ez_parallel_structs::spec_drv_1d_1 = 23
```

Flag to mark execution of 1D spectral derivatives along the first dimension.

Definition at line 102 of file ez_parallel_structs.f90.

4.2.2.5 spec_drv_1d_2

```
integer, parameter, public ez_parallel_structs::spec_drv_1d_2 = 78
```

Flag to mark execution of 1D spectral derivatives along the second dimension.

Definition at line 105 of file ez_parallel_structs.f90.

Chapter 5

Data Type Documentation

5.1 The Module Reference

5.1.1 Detailed Description

SCHEME derived datatype contains all information about the grid and the sub-grid decomposition of the grid, as well as information needed for FFTs.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel_structs.f90

5.2 ez_parallel::create_scheme Interface Reference

The interface for the `SCHEME` creation subroutine.

Private Member Functions

- subroutine create_scheme_sbr (rowCount, colCount, colSpc, colRef, comm, mpiDatatype, ovlp, sch)

5.2.1 Detailed Description

The interface for the `SCHEME` creation subroutine.

The `CREATE_SCHEME` subroutine initializes a `SCHEME` which holds information about the grid and grid decomposition, as well as variables for use in FFTs. For greater detail on the `SCHEME` creation subroutine, see create_scheme.f90. For greater detail on the `SCHEME` datatype, see ez_parallel_structs.f90.

Definition at line 56 of file ez_parallel.f90.

5.2.2 Member Function/Subroutine Documentation

5.2.2.1 create_scheme_sbr()

```
subroutine ez_parallel::create_scheme::create_scheme_sbr (
            integer, intent(in) rowCount,
            integer, intent(inout) colCount,
            double precision, intent(in) colSpc,
            double precision, intent(inout) colRef,
            integer, intent(in) comm,
            integer, intent(in) mpiDatatype,
            integer, intent(in) ovlp,
            type(scheme), intent(inout) sch )  [private]
```

Definition at line 59 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel.f90

5.3 ez_parallel::create_scheme_fft Interface Reference

The interface for the SCHEME FFT initialization subroutine.

Private Member Functions

- subroutine create_scheme_fft_sbr (sch)

5.3.1 Detailed Description

The interface for the SCHEME FFT initialization subroutine.

The CREATE_SCHEME_FFT subroutine initializes a SCHEME for future FFTs. For greater detail on the SCHEME FFT initialization subroutine, see create_scheme_fft.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 81 of file ez_parallel.f90.

5.3.2 Member Function/Subroutine Documentation

5.3.2.1 create_scheme_fft_sbr()

```
subroutine ez_parallel::create_scheme_fft::create_scheme_fft_sbr (
            type(scheme), intent(inout) sch )  [private]
```

Definition at line 83 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel.f90

5.4 ez_parallel::create_scheme_spec_drv Interface Reference

The interface for the SCHEME spectral derivative initialization subroutine.

Private Member Functions

- subroutine create_scheme_spec_drv_sbr (sch)

5.4.1 Detailed Description

The interface for the SCHEME spectral derivative initialization subroutine.

The CREATE_SCHEME_SPEC_DERV subroutine initializes a SCHEME for future spectral derivates. For greater detail on the SCHEME spectral derivative initialization subroutine, see create_scheme_spec_drv.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 99 of file ez_parallel.f90.

5.4.2 Member Function/Subroutine Documentation

5.4.2.1 create_scheme_spec_drv_sbr()

```
subroutine ez_parallel::create_scheme_spec_drv::create_scheme_spec_drv_sbr (
            type(scheme), intent(inout) sch )  [private]
```

Definition at line 101 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel.f90

5.5 ez_parallel::create_scheme_zero_pad Interface Reference

The interface for the SCHEME zero-padding initializtion subroutine.

Private Member Functions

- subroutine create_scheme_zero_pad_dble_sbr (sch, schZP)

5.5.1 Detailed Description

The interface for the SCHEME zero-padding initializtion subroutine.

The CREATE_SCHEME_ZERO_PAD subroutine initializes a SCHEME to handle a zero-padded version of the global array. For greater detail on the SCHEME spectral derivative initialization subroutine, see create_scheme_zero_pad.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 117 of file ez_parallel.f90.

5.5.2 Member Function/Subroutine Documentation

5.5.2.1 create_scheme_zero_pad_dble_sbr()

```
subroutine ez_parallel::create_scheme_zero_pad::create_scheme_zero_pad_dble_sbr (
            type(scheme), intent(in) sch,
            type(scheme), intent(inout) schZP )  [private]
```

Parameters

in	*sch*	Scheme associated with arr.
in,out	*schZP*	SCHEME associated with arrZP.

Definition at line 121 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel.f90

5.6 ez_parallel::destroy_scheme Interface Reference

The interface for the SCHEME destruction subroutine.

Private Member Functions

- subroutine destroy_scheme_sbr (sch)

5.6.1 Detailed Description

The interface for the SCHEME destruction subroutine.

The DESTROY_SCHEME subroutine deallocates a SCHEME. For greater detail on the SCHEME destruction subroutine, see destroy_scheme.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 136 of file ez_parallel.f90.

5.6.2 Member Function/Subroutine Documentation

5.6.2.1 destroy_scheme_sbr()

```
subroutine ez_parallel::destroy_scheme::destroy_scheme_sbr (
            type(scheme), intent(inout) sch )  [private]
```

Parameters

in	*sch*	SCHEME to be destroyed.

Definition at line 139 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩_project/EZ_PARALLEL/ez_parallel.f90

5.7 ez_parallel::execute_scheme_fft Interface Reference

The FFT execution interface for the EZ_PARALLEL module.

Public Member Functions

- subroutine execute_scheme_fft_dcmpx_sbr (subGrid, kind, sch)

Private Member Functions

- subroutine execute_scheme_fft_dble_sbr (subGrid, kind, sch)

5.7.1 Detailed Description

The FFT execution interface for the `EZ_PARALLEL` module.

For greater detail on the FFT execution subroutine, see execute_scheme_fft.f90.

Definition at line 180 of file ez_parallel.f90.

5.7.2 Member Function/Subroutine Documentation

5.7.2.1 execute_scheme_fft_dble_sbr()

```
subroutine ez_parallel::execute_scheme_fft::execute_scheme_fft_dble_sbr (
            double precision, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )  [private]
```

Parameters

`in,out`	*subGrid*	The local sub-grid whose boundary will be shared.
`in`	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 187 of file ez_parallel.f90.

5.7.2.2 execute_scheme_fft_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_fft::execute_scheme_fft_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

Parameters

`in,out`	*subGrid*	The local sub-grid whose boundary will be shared.
`in`	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 199 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel.f90

5.8 ez_parallel::execute_scheme_ifft Interface Reference

The inverse FFT execution interface for the `EZ_PARALLEL` module.

Public Member Functions

- subroutine execute_scheme_ifft_dcmpx_sbr (subGrid, kind, sch)

Private Member Functions

- subroutine execute_scheme_ifft_dble_sbr (subGrid, kind, sch)

5.8.1 Detailed Description

The inverse FFT execution interface for the `EZ_PARALLEL` module.

For greater detail on the inverse FFT execution subroutine, see execute_scheme_ifft.f90.

Definition at line 213 of file ez_parallel.f90.

5.8.2 Member Function/Subroutine Documentation

5.8.2.1 execute_scheme_ifft_dble_sbr()

```
subroutine ez_parallel::execute_scheme_ifft::execute_scheme_ifft_dble_sbr (
            double precision, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )  [private]
```

Parameters

`in,out`	*subGrid*	The local sub-grid whose boundary will be shared.
`in`	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 220 of file ez_parallel.f90.

5.8.2.2 execute_scheme_ifft_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_ifft::execute_scheme_ifft_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

Parameters

`in,out`	*subGrid*	The local sub-grid whose boundary will be shared.
`in`	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 232 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩
_project/EZ_PARALLEL/ez_parallel.f90

5.9 ez_parallel::execute_scheme_ispec_drv Interface Reference

The inverse spectral derivative execution interface for the `EZ_PARALLEL` module.

Public Member Functions

- subroutine execute_scheme_ispec_drv_dcmpx_sbr (subGrid, kind, order, sch)

Private Member Functions

- subroutine execute_scheme_ispec_drv_dble_sbr (subGrid, kind, order, sch)

5.9.1 Detailed Description

The inverse spectral derivative execution interface for the `EZ_PARALLEL` module.

For greater detail on the inverse spectral derivative execution subroutine, see execute_scheme_ispec_drv.f90.

Definition at line 285 of file ez_parallel.f90.

5.9.2 Member Function/Subroutine Documentation

5.9.2.1 execute_scheme_ispec_drv_dble_sbr()

```
subroutine ez_parallel::execute_scheme_ispec_drv::execute_scheme_ispec_drv_dble_sbr (
             double precision, dimension(:,:), intent(inout) subGrid,
             integer, intent(in) kind,
             integer, intent(in) order,
             type(scheme), intent(in) sch )  [private]
```

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DERV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 294 of file ez_parallel.f90.

5.9.2.2 execute_scheme_ispec_drv_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_ispec_drv::execute_scheme_ispec_drv_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DERV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 309 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel.f90

5.10 ez_parallel::execute_scheme_izero_pad Interface Reference

The interface for the `SCHEME` zero-padding removal subroutine.

Public Member Functions

- subroutine execute_scheme_izero_pad_dcmpx_sbr (arr, sch, arrZP, schZP)

Private Member Functions

- subroutine execute_scheme_izero_pad_dble_sbr (arr, sch, arrZP, schZP)

5.10.1 Detailed Description

The interface for the SCHEME zero-padding removal subroutine.

The EXECUTE_SCHEME_IZERO_PAD subroutine returns a non-zero-padded version of the array. For greater detail on the SCHEME zero-padding removal subroutine, see execute_scheme_izero_pad.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 364 of file ez_parallel.f90.

5.10.2 Member Function/Subroutine Documentation

5.10.2.1 execute_scheme_izero_pad_dble_sbr()

```
subroutine ez_parallel::execute_scheme_izero_pad::execute_scheme_izero_pad_dble_sbr (
            double precision, dimension(:,:), intent(inout) arr,
            type(scheme), intent(in) sch,
            double precision, dimension(:,:), intent(in) arrZP,
            type(scheme), intent(in) schZP )  [private]
```

Parameters

in,out	*arr*	Array to be un-zero-padded.
in	*sch*	Scheme associated with arr.
in	*arrZP*	Allocatable array that will hold the zero-padded arr.
in	*schZP*	SCHEME associated with arrZP.

Definition at line 370 of file ez_parallel.f90.

5.10.2.2 execute_scheme_izero_pad_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_izero_pad::execute_scheme_izero_pad_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) arr,
            type(scheme), intent(in) sch,
            double complex, dimension(:,:), intent(in) arrZP,
            type(scheme), intent(in) schZP )
```

Parameters

in,out	*arr*	Array to be un-zero-padded.
in	*sch*	Scheme associated with arr.
in	*arrZP*	Allocatable array that will hold the zero-padded arr.
in	*schZP*	SCHEME associated with arrZP.

Definition at line 382 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL←_project/EZ_PARALLEL/ez_parallel.f90

5.11 ez_parallel::execute_scheme_spec_drv Interface Reference

The spectral derivative execution interface for the EZ_PARALLEL module.

Public Member Functions

- subroutine execute_scheme_spec_drv_dcmpx_sbr (subGrid, kind, order, sch)

Private Member Functions

- subroutine execute_scheme_spec_drv_dble_sbr (subGrid, kind, order, sch)

5.11.1 Detailed Description

The spectral derivative execution interface for the EZ_PARALLEL module.

For greater detail on the spectral derivative execution subroutine, see execute_scheme_spec_drv.f90.

Definition at line 246 of file ez_parallel.f90.

5.11.2 Member Function/Subroutine Documentation

5.11.2.1 execute_scheme_spec_drv_dble_sbr()

```
subroutine ez_parallel::execute_scheme_spec_drv::execute_scheme_spec_drv_dble_sbr (
            double precision, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )  [private]
```

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DERV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 255 of file ez_parallel.f90.

5.11.2.2 execute_scheme_spec_drv_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_spec_drv::execute_scheme_spec_drv_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DERV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 270 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel.f90

5.12 ez_parallel::execute_scheme_zero_pad Interface Reference

The interface for the SCHEME zero-padding execution subroutine.

Public Member Functions

- subroutine execute_scheme_zero_pad_dcmpx_sbr (arr, sch, arrZP, schZP)

Private Member Functions

- subroutine execute_scheme_zero_pad_dble_sbr (arr, sch, arrZP, schZP)

5.12.1 Detailed Description

The interface for the SCHEME zero-padding execution subroutine.

The EXECUTE_SCHEME_ZERO_PAD subroutine zero-pads the global array. For greater detail on the SCHEME zero-padding subroutine, see execute_scheme_zero_pad.f90. For greater detail on the SCHEME datatype, see ez_parallel_structs.f90.

Definition at line 327 of file ez_parallel.f90.

5.12.2 Member Function/Subroutine Documentation

5.12.2.1 execute_scheme_zero_pad_dble_sbr()

```
subroutine ez_parallel::execute_scheme_zero_pad::execute_scheme_zero_pad_dble_sbr (
            double precision, dimension(:,:), intent(in) arr,
            type(scheme), intent(in) sch,
            double precision, dimension(:,:), intent(inout), allocatable arrZP,
            type(scheme), intent(in) schZP )  [private]
```

Parameters

in	*arr*	Array to be zero-padded.
in	*sch*	Scheme associated with arr.
in,out	*arrZP*	Allocatable array that will hold the zero-padded arr.
in	*schZP*	SCHEME associated with arrZP.

Definition at line 333 of file ez_parallel.f90.

5.12.2.2 execute_scheme_zero_pad_dcmpx_sbr()

```
subroutine ez_parallel::execute_scheme_zero_pad::execute_scheme_zero_pad_dcmpx_sbr (
            double complex, dimension(:,:), intent(in) arr,
            type(scheme), intent(in) sch,
            double complex, dimension(:,:), intent(inout), allocatable arrZP,
            type(scheme), intent(in) schZP )
```

Parameters

in	*arr*	Array to be zero-padded.
in	*sch*	Scheme associated with arr.
in,out	*arrZP*	Allocatable array that will hold the zero-padded arr.
in	*schZP*	SCHEME associated with arrZP.

Definition at line 345 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel.f90

5.13 ez_parallel::max_val Interface Reference

The maximum value interface for the EZ_PARALLEL module.

Private Member Functions

- subroutine max_val_sbr (subGrid, maxValue, sch)

5.13.1 Detailed Description

The maximum value interface for the EZ_PARALLEL module.

For greater detail on the maximum value subroutine, see max_val.f90.

Definition at line 395 of file ez_parallel.f90.

5.13.2 Member Function/Subroutine Documentation

5.13.2.1 max_val_sbr()

```
subroutine ez_parallel::max_val::max_val_sbr (
             double precision, dimension(:,:), intent(in) subGrid,
             double precision, intent(inout) maxValue,
             type(scheme), intent(in) sch )  [private]
```

Parameters

in	*subGrid*	The local sub-grid.
in,out	*maxVal*	The variable to store the maximum value across all sub-grids.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 402 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel.f90

5.14 ez_parallel::min_val Interface Reference

The minimum value interface for the EZ_PARALLEL module.

Private Member Functions

- subroutine min_val_sbr (subGrid, minValue, sch)

5.14.1 Detailed Description

The minimum value interface for the EZ_PARALLEL module.

For greater detail on the minimum value subroutine, see min_val.f90.

Definition at line 414 of file ez_parallel.f90.

5.14.2 Member Function/Subroutine Documentation

5.14.2.1 min_val_sbr()

```
subroutine ez_parallel::min_val::min_val_sbr (
             double precision, dimension(:,:), intent(in) subGrid,
             double precision, intent(inout) minValue,
             type(scheme), intent(in) sch )  [private]
```

Parameters

in	*subGrid*	The local sub-grid.
in,out	*minVal*	The variable to store the minimum value across all sub-grids.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 421 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩_project/EZ_PARALLEL/ez_parallel.f90

5.15 ez_parallel_structs::scheme Type Reference

Public Attributes

- integer, dimension(0:1) gridsize

  Size in each dimension of the grid.
- double precision colspc

  Physical spacing between columns of the grid.
- integer comm

  Communicator that holds the grid.
- integer commsize

  Number of processes in the MPI communicator.
- integer datatype

  Datatype of the array.
- integer ovlp

Number of extra columns needed by each sub-grid to successfully step forward in time.

- integer, dimension(:), allocatable rowdcmpsizes

  Number of rows in each sub-grid of the horizontal-slab decomposition, excluding overlap.

- integer, dimension(:), allocatable coldcmpsizes

  Number of columns in each sub-grid of the vertical-slab decomposition, excluding overlap.

- integer, dimension(:), allocatable coldcmpsizesovlp

  Number of columns in each sub-grid of the vertical slab decomposition, including overlap.

- integer procid

  Processor ID.

- integer, dimension(0:1) hslabsize

  Sizes in each dimension of the sub-grid in the horizontal-slab decomposition of the global array.

- integer, dimension(0:1) vslabsize

  Sizes in each dimension of the sub-grid in the vertical-slab decomposition of the global array, excluding overlap.

- integer, dimension(0:1) vslabsizeovlp

  Sizes in each dimension of the sub-grid in the vertical-slab decomposition of the global array, including overlap.

- integer, dimension(0:1) vslabint

  Column indices of the interior of the vertical slab.

- double precision colref

  The physical position of the reference point in the dimension along the rows (corresponding to a column).

- integer, dimension(0:1) send_boundaries

  MPI derived datatype for sending sub-grid boundaries to neightboring sub-grids (0 = left, 1 = right).

- integer, dimension(0:1) recv_boundaries

  MPI derived datatype for recieving sub-grid boundaries from neightboring sub-grids (0 = left, 1 = right).

- double precision, dimension(:), allocatable wsave1

  Holds initialization info for DFFTPACK 1-D FFTS along first dimension.

- double precision, dimension(:), allocatable wsave2

  Holds initialization info for DFFTPACK 1-D FFTS along second dimension.

- double precision norm_1d_1

  The normalization coefficient for possible 1-D FFTs along the first dimension.

- double precision norm_1d_2

  The normalization coefficient for possible 1-D FFTs along the second dimension.

- double precision norm_2d

  The normalization coefficient for possible 2D FFTs.

- integer, dimension(:,:), allocatable subarrays

  Holds the datatypes necessary to perform the transposition.

- integer, dimension(:), allocatable counts
- integer, dimension(:), allocatable displs

  Arrays for use in global.

- double complex, dimension(:), allocatable wvnmbr1

  Holds coefficients for spectral derivative along the first dimension.

- double complex, dimension(:), allocatable wvnmbr2

  Holds coefficients for spectral derivative along the second dimension.

- logical initscheme = .FALSE.

  Checks if `SCHEME` was created already.

- logical initfft = .FALSE.

  Checks if FFTs for `SCHEME` were initialized already.

- logical initspecdrv = .FALSE.

  Checks if spectral derivates for `SCHEME` were initialized already.

5.15.1 Detailed Description

Definition at line 29 of file ez_parallel_structs.f90.

5.15.2 Member Data Documentation

5.15.2.1 coldcmpsizes

```
integer, dimension(:), allocatable ez_parallel_structs::scheme::coldcmpsizes
```

Number of columns in each sub-grid of the vertical-slab decomposition, excluding overlap.

Definition at line 40 of file ez_parallel_structs.f90.

5.15.2.2 coldcmpsizesovlp

```
integer, dimension(:), allocatable ez_parallel_structs::scheme::coldcmpsizesovlp
```

Number of columns in each sub-grid of the vertical slab decomposition, including overlap.

Definition at line 42 of file ez_parallel_structs.f90.

5.15.2.3 colref

```
double precision ez_parallel_structs::scheme::colref
```

The physical position of the reference point in the dimension along the rows (corresponding to a column).

Definition at line 55 of file ez_parallel_structs.f90.

5.15.2.4 colspc

```
double precision ez_parallel_structs::scheme::colspc
```

Physical spacing between columns of the grid.

Definition at line 32 of file ez_parallel_structs.f90.

5.15.2.5 comm

```
integer ez_parallel_structs::scheme::comm
```

Communicator that holds the grid.

Definition at line 33 of file ez_parallel_structs.f90.

5.15.2.6 commsize

```
integer ez_parallel_structs::scheme::commsize
```

Number of processes in the MPI communicator.

Definition at line 34 of file ez_parallel_structs.f90.

5.15.2.7 counts

```
integer, dimension(:), allocatable ez_parallel_structs::scheme::counts
```

Definition at line 75 of file ez_parallel_structs.f90.

5.15.2.8 datatype

```
integer ez_parallel_structs::scheme::datatype
```

Datatype of the array.

Definition at line 35 of file ez_parallel_structs.f90.

5.15.2.9 displs

```
integer, dimension(:), allocatable ez_parallel_structs::scheme::displs
```

Arrays for use in global.

Definition at line 75 of file ez_parallel_structs.f90.

5.15.2.10 gridsize

```
integer, dimension(0:1) ez_parallel_structs::scheme::gridsize
```

Size in each dimension of the grid.

Definition at line 31 of file ez_parallel_structs.f90.

5.15.2.11 hslabsize

```
integer, dimension(0:1) ez_parallel_structs::scheme::hslabsize
```

Sizes in each dimension of the sub-grid in the horizontal-slab decomposition of the global array.

Definition at line 47 of file ez_parallel_structs.f90.

5.15.2.12 initfft

```
logical ez_parallel_structs::scheme::initfft = .FALSE.
```

Checks if FFTs for `SCHEME` were initialized already.

Definition at line 86 of file ez_parallel_structs.f90.

5.15.2.13 initscheme

```
logical ez_parallel_structs::scheme::initscheme = .FALSE.
```

Checks if `SCHEME` was created already.

Definition at line 84 of file ez_parallel_structs.f90.

5.15.2.14 initspecdrv

```
logical ez_parallel_structs::scheme::initspecdrv = .FALSE.
```

Checks if spectral derivates for `SCHEME` were initialized already.

Definition at line 88 of file ez_parallel_structs.f90.

5.15.2.15 norm_1d_1

```
double precision ez_parallel_structs::scheme::norm_1d_1
```

The normalization coefficient for possible 1-D FFTs along the first dimension.

Definition at line 67 of file ez_parallel_structs.f90.

5.15.2.16 norm_1d_2

```
double precision ez_parallel_structs::scheme::norm_1d_2
```

The normalization coefficient for possible 1-D FFTs along the second dimension.

Definition at line 69 of file ez_parallel_structs.f90.

5.15.2.17 norm_2d

```
double precision ez_parallel_structs::scheme::norm_2d
```

The normalization coefficient for possible 2D FFTs.

Definition at line 71 of file ez_parallel_structs.f90.

5.15.2.18 ovlp

```
integer ez_parallel_structs::scheme::ovlp
```

Number of extra columns needed by each sub-grid to successfully step forward in time.

Definition at line 36 of file ez_parallel_structs.f90.

5.15.2.19 procid

```
integer ez_parallel_structs::scheme::procid
```

Processor ID.

Definition at line 46 of file ez_parallel_structs.f90.

5.15.2.20 recv_boundaries

```
integer, dimension(0:1) ez_parallel_structs::scheme::recv_boundaries
```

MPI derived datatype for recieving sub-grid boundaries from neightboring sub-grids (0 = left, 1 = right).

Definition at line 59 of file ez_parallel_structs.f90.

5.15.2.21 rowdcmpsizes

```
integer, dimension(:), allocatable ez_parallel_structs::scheme::rowdcmpsizes
```

Number of rows in each sub-grid of the horizontal-slab decomposition, excluding overlap.

Definition at line 38 of file ez_parallel_structs.f90.

5.15.2.22 send_boundaries

```
integer, dimension(0:1) ez_parallel_structs::scheme::send_boundaries
```

MPI derived datatype for sending sub-grid boundaries to neightboring sub-grids (0 = left, 1 = right).

Definition at line 57 of file ez_parallel_structs.f90.

5.15.2.23 subarrays

```
integer, dimension(:,:), allocatable ez_parallel_structs::scheme::subarrays
```

Holds the datatypes necessary to perform the transposition.

Definition at line 73 of file ez_parallel_structs.f90.

5.15.2.24 vslabint

```
integer, dimension(0:1) ez_parallel_structs::scheme::vslabint
```

Column indices of the interior of the vertical slab.

Definition at line 53 of file ez_parallel_structs.f90.

5.15.2.25 vslabsize

```
integer, dimension(0:1) ez_parallel_structs::scheme::vslabsize
```

Sizes in each dimension of the sub-grid in the vertical-slab decomposition of the global array, excluding overlap.

Definition at line 49 of file ez_parallel_structs.f90.

5.15.2.26 vslabsizeovlp

```
integer, dimension(0:1) ez_parallel_structs::scheme::vslabsizeovlp
```

Sizes in each dimension of the sub-grid in the vertical-slab decomposition of the global array, including overlap.

Definition at line 51 of file ez_parallel_structs.f90.

5.15.2.27 wsave1

```
double precision, dimension(:), allocatable ez_parallel_structs::scheme::wsave1
```

Holds initialization info for DFFTPACK 1-D FFTS along first dimension.

Definition at line 63 of file ez_parallel_structs.f90.

5.15.2.28 wsave2

```
double precision, dimension(:), allocatable ez_parallel_structs::scheme::wsave2
```

Holds initialization info for DFFTPACK 1-D FFTS along second dimension.

Definition at line 65 of file ez_parallel_structs.f90.

5.15.2.29 wvnmbr1

```
double complex, dimension(:), allocatable ez_parallel_structs::scheme::wvnmbr1
```

Holds coefficients for spectral derivative along the first dimension.

Definition at line 78 of file ez_parallel_structs.f90.

5.15.2.30 wvnmbr2

```
double complex, dimension(:), allocatable ez_parallel_structs::scheme::wvnmbr2
```

Holds coefficients for spectral derivative along the second dimension.

Definition at line 80 of file ez_parallel_structs.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩ _project/EZ_PARALLEL/ez_parallel_structs.f90

5.16 ez_parallel::share_subgrid_bdry Interface Reference

The sub-grid boundary communication interface for the `EZ_PARALLEL` module.

Public Member Functions

- subroutine share_subgrid_bdry_dcmpx_sbr (subGrid, sch)

Private Member Functions

- subroutine share_subgrid_bdry_dble_sbr (subGrid, sch)

5.16.1 Detailed Description

The sub-grid boundary communication interface for the `EZ_PARALLEL` module.

Communicates the sub-grid boundary to neighboring sub-grids.

For greater detail on the sub-grid boundary communication subroutine, see share_subgrid_bdry.f90.

Definition at line 153 of file ez_parallel.f90.

5.16.2 Member Function/Subroutine Documentation

5.16.2.1 share_subgrid_bdry_dble_sbr()

```
subroutine ez_parallel::share_subgrid_bdry::share_subgrid_bdry_dble_sbr (
            double precision, dimension(:,:), intent(inout) subGrid,
            type(scheme), intent(in) sch )  [private]
```

Parameters

in,out	*subGrid*	The sub-grid belonging to the processor.
in	*sch*	SCHEME that holds grid decomposition information, etc.

Definition at line 158 of file ez_parallel.f90.

5.16.2.2 share_subgrid_bdry_dcmpx_sbr()

```
subroutine ez_parallel::share_subgrid_bdry::share_subgrid_bdry_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            type(scheme), intent(in) sch )
```

Parameters

in,out	*subGrid*	The sub-grid belonging to the processor.
in	*sch*	SCHEME that holds grid decomposition information, etc.

Definition at line 168 of file ez_parallel.f90.

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

- C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL↩_project/EZ_PARALLEL/ez_parallel.f90

Chapter 6

File Documentation

6.1 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/create_scheme.f90 File Reference

Functions/Subroutines

- subroutine create_scheme_sbr (rowCount, colCount, colSpc, colRef, comm, mpiDatatype, ovlp, sch)

  The `SCHEME` creation subroutine.
- subroutine create_scheme_eh (rowCount, colCount, ovlp, sch)

  The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.1.1 Function/Subroutine Documentation

6.1.1.1 create_scheme_eh()

```
subroutine create_scheme_sbr::create_scheme_eh (
             integer, intent(in) rowCount,
             integer, intent(in) colCount,
             integer, intent(in) ovlp,
             type(scheme), intent(in) sch )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- rowCount...

  1. is not positive
- colCount...

   1. is not positive
   2. is too small to be divided among the processors, ignoring the overlap.
   3. is too small to be divided among the processors, including the overlap.

- ovlp...
   1. is negative
- sch...
   1. is already initialized

Definition at line 186 of file create_scheme.f90.

6.1.1.2 create_scheme_sbr()

```
subroutine create_scheme_sbr (
            integer, intent(in) rowCount,
            integer, intent(inout) colCount,
            double precision, intent(in) colSpc,
            double precision, intent(inout) colRef,
            integer, intent(in) comm,
            integer, intent(in) mpiDatatype,
            integer, intent(in) ovlp,
            type(scheme), intent(inout) sch )
```

The `SCHEME` creation subroutine.

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in`	*rowCount*	Number of rows in the grid.
`in,out`	*colCount*	Number of columns in the grid. Returns the number of columns in the sub-grid, including overlap.
`in`	*colSpc*	The spacing between columns of the grid, for reference point identification.
`in,out`	*colRef*	The physical position of the reference point in the dimension along the rows (corresponding to a column).
`in`	*comm*	`MPI` communicator that host the processors that contain the sub-grids.
`in`	*mpiDatatype*	`MPI` datatype corresponding to the datatype of the grid.
`in`	*ovlp*	Number of extra columns needed by each sub-grid to successfully step forward in time.
`in,out`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 25 of file create_scheme.f90.

6.2 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_P↩ ARALLEL/create_scheme_fft.f90 File Reference

Functions/Subroutines

- subroutine create_scheme_fft_sbr (sch)

  The `SCHEME` FFT creation subroutine.
- subroutine create_scheme_fft_eh (sch)

  The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine decompose (nelems, nparts, pidx, nelemspart, sidx)

  Creates a decomposition of a single dimension of the 2-D array.
- subroutine subarray (sizes, axis, nparts, datatype, subarrays)

  Creates the subarray datatypes for the local array.

6.2.1 Function/Subroutine Documentation

6.2.1.1 create_scheme_fft_eh()

```
subroutine create_scheme_fft_sbr::create_scheme_fft_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is already initialized for FFTs

Definition at line 69 of file create_scheme_fft.f90.

6.2.1.2 create_scheme_fft_sbr()

```
subroutine create_scheme_fft_sbr (
            type(scheme), intent(inout) sch )
```

The `SCHEME` FFT creation subroutine.

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 11 of file create_scheme_fft.f90.

6.2.1.3 decompose()

```
subroutine decompose (
             integer, intent(in) nelems,
             integer, intent(in) nparts,
             integer, intent(in) pidx,
             integer, intent(out) nelemspart,
             integer, intent(out) sidx )
```

Creates a decomposition of a single dimension of the 2-D array.

Parameters

[IN]	nelems Number of elements along the dimension of the array, >=0.
[IN]	nparts Number of parts to divide dimension into, >0.
[IN]	pidx Part index, >=0 and <nparts.
[O↵ UT]	nelemspart Number of elements in part.
[O↵ UT]	sidx Start index of part.

Definition at line 115 of file create_scheme_fft.f90.

6.2.1.4 subarray()

```
subroutine subarray (
             integer, dimension(0:1), intent(in) sizes,
             integer, intent(in) axis,
             integer, intent(in) nparts,
             integer, intent(in) datatype,
             integer, dimension(0:nparts-1), intent(out) subarrays )
```

Creates the subarray datatypes for the local array.

Parameters

[IN]	sizes Local sizes of array.
[IN]	axis Axis to partition, 0 <= axis < 2.
[IN]	nparts Number of parts, nparts > 0.
[IN]	datatype MPI datatype descriptor.
[O↵ UT]	subarrays Subarray datatype descriptors.

Definition at line 146 of file create_scheme_fft.f90.

6.3 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_P↩ ARALLEL/create_scheme_spec_drv.f90 File Reference

Functions/Subroutines

- subroutine create_scheme_spec_drv_sbr (sch)

  The `SCHEME` zero-padding initialization subroutine (DOUBLE PRECISION).
- subroutine create_scheme_spec_drv_eh (sch)

  The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.3.1 Function/Subroutine Documentation

6.3.1.1 create_scheme_spec_drv_eh()

```
subroutine create_scheme_spec_drv_sbr::create_scheme_spec_drv_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is already initialized for spectral derivatives

Definition at line 86 of file create_scheme_spec_drv.f90.

6.3.1.2 create_scheme_spec_drv_sbr()

```
subroutine create_scheme_spec_drv_sbr (
            type(scheme), intent(inout) sch )
```

The `SCHEME` zero-padding initialization subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in,out`	*sch*	`SCHEME` that holds information for grid decomposition, etc.

Definition at line 11 of file create_scheme_spec_drv.f90.

6.4 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_P↩ARALLEL/create_scheme_zero_pad.f90 File Reference

Functions/Subroutines

- subroutine create_scheme_zero_pad_dble_sbr (sch, schZP)

  The `SCHEME` zero-padding initialization subroutine (DOUBLE PRECISION).
- subroutine create_scheme_zero_pad_dble_eh (sch, schZP)

  The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine decompose (nelems, nparts, pidx, nelemspart, sidx)

  Creates a decomposition of a single dimension of the 2-D array.
- subroutine subarray (sizes, axis, nparts, datatype, subarrays)

  Creates the subarray datatypes for the local array.

6.4.1 Function/Subroutine Documentation

6.4.1.1 create_scheme_zero_pad_dble_eh()

```
subroutine create_scheme_zero_pad_dble_sbr::create_scheme_zero_pad_dble_eh (
            type(scheme), intent(in) sch,
            type(scheme), intent(in) schZP )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives
- schZP...
  1. is already initialized

Definition at line 217 of file create_scheme_zero_pad.f90.

6.4.1.2 create_scheme_zero_pad_dble_sbr()

```
subroutine create_scheme_zero_pad_dble_sbr (
            type(scheme), intent(in) sch,
            type(scheme), intent(inout) schZP )
```

The SCHEME zero-padding initialization subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in	*sch*	SCHEME that is used for arr.
in,out	*schZP*	SCHEME that is made for arrZP.

Definition at line 11 of file create_scheme_zero_pad.f90.

6.4.1.3 decompose()

```
subroutine create_scheme_zero_pad_dble_sbr::decompose (
            integer, intent(in) nelems,
            integer, intent(in) nparts,
            integer, intent(in) pidx,
            integer, intent(out) nelemspart,
            integer, intent(out) sidx )
```

Creates a decomposition of a single dimension of the 2-D array.

Parameters

[IN]	nelems Number of elements along the dimension of the array, >=0.
[IN]	nparts Number of parts to divide dimension into, >0.
[IN]	pidx Part index, >=0 and <nparts.
[O↩ UT]	nelemspart Number of elements in part.
[O↩ UT]	sidx Start index of part.

Definition at line 281 of file create_scheme_zero_pad.f90.

6.4.1.4 subarray()

```
subroutine create_scheme_zero_pad_dble_sbr::subarray (
            integer, dimension(0:1), intent(in) sizes,
            integer, intent(in) axis,
            integer, intent(in) nparts,
            integer, intent(in) datatype,
            integer, dimension(0:nparts-1), intent(out) subarrays )
```

Creates the subarray datatypes for the local array.

Parameters

[IN]	sizes Local sizes of array.
[IN]	axis Axis to partition, 0 <= axis < 2.
[IN]	nparts Number of parts, nparts > 0.
[IN]	datatype MPI datatype descriptor.
[O↵ UT]	subarrays Subarray datatype descriptors.

Definition at line 312 of file create_scheme_zero_pad.f90.

6.5 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/destroy_scheme.f90 File Reference

Functions/Subroutines

- subroutine destroy_scheme_sbr (sch)

  The `SCHEME` creation subroutine.
- subroutine destroy_scheme_eh (sch)

  The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.5.1 Function/Subroutine Documentation

6.5.1.1 destroy_scheme_eh()

```
subroutine destroy_scheme_sbr::destroy_scheme_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

-

1. is not initialized

Definition at line 91 of file destroy_scheme.f90.

6.5.1.2 destroy_scheme_sbr()

```
subroutine destroy_scheme_sbr (
            type(scheme), intent(inout) sch )
```

The SCHEME creation subroutine.

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 11 of file destroy_scheme.f90.

6.6 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/execute_scheme_fft.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_fft_dble_sbr (subGrid, kind, sch)

  The FFT execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_fft_dble_eh (kind, sch)

  The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine execute_scheme_fft_dcmpx_sbr (subGrid, kind, sch)

  The FFT execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_fft_dcmpx_eh (kind, sch)

  The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.6.1 Function/Subroutine Documentation

6.6.1.1 execute_scheme_fft_dble_eh()

```
subroutine execute_scheme_fft_dble_sbr::execute_scheme_fft_dble_eh (
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

    1.

  not supported

- sch...

    1. is not initialized
    2. is not initialized for FFTs

Definition at line 122 of file execute_scheme_fft.f90.

6.6.1.2 execute_scheme_fft_dble_sbr()

```
subroutine execute_scheme_fft_dble_sbr (
            double precision, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The FFT execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 13 of file execute_scheme_fft.f90.

6.6.1.3 execute_scheme_fft_dcmpx_eh()

```
subroutine execute_scheme_fft_dcmpx_sbr::execute_scheme_fft_dcmpx_eh (
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

    1.

  not supported

- sch...
  1. is not initialized
  2. is not initialized for FFTs

Definition at line 292 of file execute_scheme_fft.f90.

6.6.1.4 execute_scheme_fft_dcmpx_sbr()

```
subroutine execute_scheme_fft_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The FFT execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 183 of file execute_scheme_fft.f90.

6.7 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/execute_scheme_ifft.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_ifft_dble_sbr (subGrid, kind, sch)

  The inverse FFT execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_ifft_dble_eh (kind, sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine execute_scheme_ifft_dcmpx_sbr (subGrid, kind, sch)

  The inverse FFT execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_ifft_dcmpx_eh (kind, sch)

  The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.7.1 Function/Subroutine Documentation

6.7.1.1 execute_scheme_ifft_dble_eh()

```
subroutine execute_scheme_ifft_dble_sbr::execute_scheme_ifft_dble_eh (
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

  1.

  not supported

- sch...

  1. is not initialized
  2. is not initialized for FFTs

Definition at line 128 of file execute_scheme_ifft.f90.

6.7.1.2 execute_scheme_ifft_dble_sbr()

```
subroutine execute_scheme_ifft_dble_sbr (
            double precision, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The inverse FFT execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in,out`	*subGrid*	The local sub-grid.
`in`	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 13 of file execute_scheme_ifft.f90.

6.7.1.3 execute_scheme_ifft_dcmpx_eh()

```
subroutine execute_scheme_ifft_dcmpx_sbr::execute_scheme_ifft_dcmpx_eh (
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...
  1.

  not supported
- sch...
  1. is not initialized
  2. is not initialized for FFTs

Definition at line 304 of file execute_scheme_ifft.f90.

6.7.1.4 execute_scheme_ifft_dcmpx_sbr()

```
subroutine execute_scheme_ifft_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            type(scheme), intent(in) sch )
```

The inverse FFT execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of FFT to execute, one of FFT_1D_1, FFT_1D_2, or FFT_2D.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 189 of file execute_scheme_ifft.f90.

6.8 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_P↩ ARALLEL/execute_scheme_ispec_drv.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_ispec_drv_dble_sbr (subGrid, kind, order, sch)

  The inverse spectral derivative execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_ispec_drv_dble_eh (kind, order, sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine execute_scheme_ispec_drv_dcmpx_sbr (subGrid, kind, order, sch)

  The spectral derivative execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_ispec_drv_dcmpx_eh (kind, order, sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.8.1 Function/Subroutine Documentation

6.8.1.1 execute_scheme_ispec_drv_dble_eh()

```
subroutine execute_scheme_ispec_drv_dble_sbr::execute_scheme_ispec_drv_dble_eh (
              integer, intent(in) kind,
              integer, intent(in) order,
              type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

  1.

  not supported
- order...

  1.

  negative
- sch...

  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives

Definition at line 63 of file execute_scheme_ispec_drv.f90.

6.8.1.2 execute_scheme_ispec_drv_dble_sbr()

```
subroutine execute_scheme_ispec_drv_dble_sbr (
            double precision, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The inverse spectral derivative execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DRV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 15 of file execute_scheme_ispec_drv.f90.

6.8.1.3 execute_scheme_ispec_drv_dcmpx_eh()

```
subroutine execute_scheme_ispec_drv_dcmpx_sbr::execute_scheme_ispec_drv_dcmpx_eh (
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

  1.

  not supported
- order...

  1.

  negative
- sch...

  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives

Definition at line 251 of file execute_scheme_ispec_drv.f90.

6.8.1.4 execute_scheme_ispec_drv_dcmpx_sbr()

```
subroutine execute_scheme_ispec_drv_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The spectral derivative execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DRV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 142 of file execute_scheme_ispec_drv.f90.

6.9 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_P↩ ARALLEL/execute_scheme_izero_pad.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_izero_pad_dble_sbr (arr, sch, arrZP, schZP)

  The SCHEME zero-padding execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_izero_pad_dble_eh (sch, schZP)

  The error handling subroutine for the SCHEME creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine decompose (nelems, nparts, pidx, nelemspart, sidx)

  Creates a decomposition of a single dimension of the 2-D array.
- subroutine subarray (sizes, axis, nparts, datatype, subarrays)

  Creates the subarray datatypes for the local array.
- subroutine execute_scheme_izero_pad_dcmpx_sbr (arr, sch, arrZP, schZP)

  The SCHEME zero-padding execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_izero_pad_dcmpx_eh (sch, schZP)

  The error handling subroutine for the SCHEME creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.9.1 Function/Subroutine Documentation

6.9.1.1 decompose()

```
subroutine execute_scheme_izero_pad_dble_sbr::decompose (
            integer, intent(in) nelems,
            integer, intent(in) nparts,
            integer, intent(in) pidx,
            integer, intent(out) nelemspart,
            integer, intent(out) sidx )
```

Creates a decomposition of a single dimension of the 2-D array.

Parameters

[IN]	nelems Number of elements along the dimension of the array, >=0.
[IN]	nparts Number of parts to divide dimension into, >0.
[IN]	pidx Part index, >=0 and <nparts.
[O↩ UT]	nelemspart Number of elements in part.
[O↩ UT]	sidx Start index of part.

Definition at line 192 of file execute_scheme_izero_pad.f90.

6.9.1.2 execute_scheme_izero_pad_dble_eh()

```
subroutine execute_scheme_izero_pad_dble_sbr::execute_scheme_izero_pad_dble_eh (
            type(scheme), intent(in) sch,
            type(scheme), intent(in) schZP )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives
- schZP...
  1. is not initialized

Definition at line 128 of file execute_scheme_izero_pad.f90.

6.9.1.3 execute_scheme_izero_pad_dble_sbr()

```
subroutine execute_scheme_izero_pad_dble_sbr (
            double precision, dimension(0:,0:), intent(inout) arr,
            type(scheme), intent(in) sch,
            double precision, dimension(0:,0:), intent(in) arrZP,
            type(scheme), intent(in) schZP )
```

The SCHEME zero-padding execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in	*arr*	Sub-grid that will be zero padded.
in	*sch*	SCHEME that is used for arr.
in,out	*arrZP*	Zero-padded sub-grid.
in,out	*schZP*	SCHEME that is made for arrZP.

Definition at line 13 of file execute_scheme_izero_pad.f90.

6.9.1.4 execute_scheme_izero_pad_dcmpx_eh()

```
subroutine execute_scheme_izero_pad_dcmpx_sbr::execute_scheme_izero_pad_dcmpx_eh (
            type(scheme), intent(in) sch,
            type(scheme), intent(in) schZP )
```

The error handling subroutine for the SCHEME creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives
- schZP...
  1. is not initialized

Definition at line 382 of file execute_scheme_izero_pad.f90.

6.9.1.5 execute_scheme_izero_pad_dcmpx_sbr()

```
subroutine execute_scheme_izero_pad_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(inout) arr,
            type(scheme), intent(in) sch,
            double complex, dimension(0:,0:), intent(in) arrZP,
            type(scheme), intent(in) schZP )
```

The SCHEME zero-padding execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in	*arr*	Sub-grid that will be zero padded.
in	*sch*	SCHEME that is used for arr.
in,out	*arrZP*	Zero-padded sub-grid.
in,out	*schZP*	SCHEME that is made for arrZP.

Definition at line 267 of file execute_scheme_izero_pad.f90.

6.9.1.6 subarray()

```
subroutine execute_scheme_izero_pad_dble_sbr::subarray (
            integer, dimension(0:1), intent(in) sizes,
            integer, intent(in) axis,
            integer, intent(in) nparts,
            integer, intent(in) datatype,
            integer, dimension(0:nparts-1), intent(out) subarrays )
```

Creates the subarray datatypes for the local array.

Parameters

[IN]	sizes Local sizes of array.
[IN]	axis Axis to partition, 0 <= axis < 2.
[IN]	nparts Number of parts, nparts > 0.
[IN]	datatype MPI datatype descriptor.
[O← UT]	subarrays Subarray datatype descriptors.

Definition at line 223 of file execute_scheme_izero_pad.f90.

6.10 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_↩ PARALLEL/execute_scheme_spec_drv.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_spec_drv_dble_sbr (subGrid, kind, order, sch)

  The spectral derivative execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_spec_drv_dble_eh (kind, order, sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine execute_scheme_spec_drv_dcmpx_sbr (subGrid, kind, order, sch)

  The spectral derivative execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_spec_drv_dcmpx_eh (kind, order, sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.10.1 Function/Subroutine Documentation

6.10.1.1 execute_scheme_spec_drv_dble_eh()

```
subroutine execute_scheme_spec_drv_dble_sbr::execute_scheme_spec_drv_dble_eh (
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

  1.

  not supported
- order...

  1.

  negative
- sch...

  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives

Definition at line 63 of file execute_scheme_spec_drv.f90.

6.10.1.2 execute_scheme_spec_drv_dble_sbr()

```
subroutine execute_scheme_spec_drv_dble_sbr (
            double precision, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The spectral derivative execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DRV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 15 of file execute_scheme_spec_drv.f90.

6.10.1.3 execute_scheme_spec_drv_dcmpx_eh()

```
subroutine execute_scheme_spec_drv_dcmpx_sbr::execute_scheme_spec_drv_dcmpx_eh (
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- kind...

  1.

  not supported
- order...

  1.

  negative
- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives

Definition at line 235 of file execute_scheme_spec_drv.f90.

6.10.1.4 execute_scheme_spec_drv_dcmpx_sbr()

```
subroutine execute_scheme_spec_drv_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(inout) subGrid,
            integer, intent(in) kind,
            integer, intent(in) order,
            type(scheme), intent(in) sch )
```

The spectral derivative execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*kind*	Type of spectral derivative to execute, one of SPEC_DRV_1D_1, SPEC_DRV_1D_2.
in	*order*	Order of the spectral derivative.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 142 of file execute_scheme_spec_drv.f90.

6.11 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_↩ PARALLEL/execute_scheme_zero_pad.f90 File Reference

Functions/Subroutines

- subroutine execute_scheme_zero_pad_dble_sbr (arr, sch, arrZP, schZP)

  The SCHEME zero-padding execution subroutine (DOUBLE PRECISION).
- subroutine execute_scheme_zero_pad_dble_eh (sch, schZP)

  The error handling subroutine for the SCHEME creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine decompose (nelems, nparts, pidx, nelemspart, sidx)

  Creates a decomposition of a single dimension of the 2-D array.
- subroutine subarray (sizes, axis, nparts, datatype, subarrays)

  Creates the subarray datatypes for the local array.
- subroutine execute_scheme_zero_pad_dcmpx_sbr (arr, sch, arrZP, schZP)

  The SCHEME zero-padding execution subroutine (DOUBLE COMPLEX).
- subroutine execute_scheme_zero_pad_dcmpx_eh (sch, schZP)

  The error handling subroutine for the SCHEME creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.11.1 Function/Subroutine Documentation

6.11.1.1 decompose()

```
subroutine execute_scheme_zero_pad_dble_sbr::decompose (
            integer, intent(in) nelems,
            integer, intent(in) nparts,
            integer, intent(in) pidx,
            integer, intent(out) nelemspart,
            integer, intent(out) sidx )
```

Creates a decomposition of a single dimension of the 2-D array.

Parameters

[IN]	nelems Number of elements along the dimension of the array, >=0.
[IN]	nparts Number of parts to divide dimension into, >0.
[IN]	pidx Part index, >=0 and <nparts.
[O↩ UT]	nelemspart Number of elements in part.
[O↩ UT]	sidx Start index of part.

Definition at line 187 of file execute_scheme_zero_pad.f90.

6.11.1.2 execute_scheme_zero_pad_dble_eh()

```
subroutine execute_scheme_zero_pad_dble_sbr::execute_scheme_zero_pad_dble_eh (
            type(scheme), intent(in) sch,
            type(scheme), intent(in) schZP )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives
- schZP...
  1. is not initialized

Definition at line 123 of file execute_scheme_zero_pad.f90.

6.11.1.3 execute_scheme_zero_pad_dble_sbr()

```
subroutine execute_scheme_zero_pad_dble_sbr (
            double precision, dimension(0:,0:), intent(in) arr,
            type(scheme), intent(in) sch,
            double precision, dimension(:,:), intent(inout), allocatable arrZP,
            type(scheme), intent(in) schZP )
```

The `SCHEME` zero-padding execution subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in`	*arr*	Sub-grid that will be zero padded.
`in`	*sch*	`SCHEME` that is used for arr.
`in,out`	*arrZP*	Zero-padded sub-grid.
`in,out`	*schZP*	`SCHEME` that is made for arrZP.

Definition at line 13 of file execute_scheme_zero_pad.f90.

6.11.1.4 execute_scheme_zero_pad_dcmpx_eh()

```
subroutine execute_scheme_zero_pad_dcmpx_sbr::execute_scheme_zero_pad_dcmpx_eh (
            type(scheme), intent(in) sch,
            type(scheme), intent(in) schZP )
```

The error handling subroutine for the `SCHEME` creation subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized
  2. is not initialized for FFTs
  3. is not initialized for spectral derivatives
- schZP...
  1. is not initialized

Definition at line 372 of file execute_scheme_zero_pad.f90.

6.11.1.5 execute_scheme_zero_pad_dcmpx_sbr()

```
subroutine execute_scheme_zero_pad_dcmpx_sbr (
            double complex, dimension(0:,0:), intent(in) arr,
            type(scheme), intent(in) sch,
            double complex, dimension(:,:), intent(inout), allocatable arrZP,
            type(scheme), intent(in) schZP )
```

The `SCHEME` zero-padding execution subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in`	*arr*	Sub-grid that will be zero padded.
`in`	*sch*	`SCHEME` that is used for arr.
`in,out`	*arrZP*	Zero-padded sub-grid.
`in,out`	*schZP*	`SCHEME` that is made for arrZP.

Definition at line 262 of file execute_scheme_zero_pad.f90.

6.11.1.6 subarray()

```
subroutine execute_scheme_zero_pad_dble_sbr::subarray (
            integer, dimension(0:1), intent(in) sizes,
            integer, intent(in) axis,
            integer, intent(in) nparts,
            integer, intent(in) datatype,
            integer, dimension(0:nparts-1), intent(out) subarrays )
```

Creates the subarray datatypes for the local array.

Parameters

[IN]	sizes Local sizes of array.
[IN]	axis Axis to partition, 0 <= axis < 2.
[IN]	nparts Number of parts, nparts > 0.
[IN]	datatype MPI datatype descriptor.
[O←UT]	subarrays Subarray datatype descriptors.

Definition at line 218 of file execute_scheme_zero_pad.f90.

6.12 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel.f90 File Reference

Data Types

- interface ez_parallel::create_scheme

  The interface for the `SCHEME` creation subroutine.
- interface ez_parallel::create_scheme_fft

  The interface for the `SCHEME` FFT initialization subroutine.
- interface ez_parallel::create_scheme_spec_drv

  The interface for the `SCHEME` spectral derivative initialization subroutine.
- interface ez_parallel::create_scheme_zero_pad

  The interface for the `SCHEME` zero-padding initializtion subroutine.
- interface ez_parallel::destroy_scheme

  The interface for the `SCHEME` destruction subroutine.
- interface ez_parallel::share_subgrid_bdry

  The sub-grid boundary communication interface for the `EZ_PARALLEL` module.
- interface ez_parallel::execute_scheme_fft

  The FFT execution interface for the `EZ_PARALLEL` module.
- interface ez_parallel::execute_scheme_ifft

  The inverse FFT execution interface for the `EZ_PARALLEL` module.
- interface ez_parallel::execute_scheme_spec_drv

  The spectral derivative execution interface for the `EZ_PARALLEL` module.
- interface ez_parallel::execute_scheme_ispec_drv

  The inverse spectral derivative execution interface for the `EZ_PARALLEL` module.
- interface ez_parallel::execute_scheme_zero_pad

  The interface for the `SCHEME` zero-padding execution subroutine.
- interface ez_parallel::execute_scheme_izero_pad

  The interface for the `SCHEME` zero-padding removal subroutine.
- interface ez_parallel::max_val

  The maximum value interface for the `EZ_PARALLEL` module.
- interface ez_parallel::min_val

  The minimum value interface for the `EZ_PARALLEL` module.

Modules

- module ez_parallel

  The EZ_PARALLEL module. Contains EZ_PARALLEL subroutines and their interfaces.

6.13 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/ez_parallel_structs.f90 File Reference

Data Types

- type ez_parallel_structs::scheme

Modules

- module ez_parallel_structs

  The EZ_PARALLEL structures module. Contains all EZ_PARALLEL derived datatypes.

Variables

- integer, parameter, public ez_parallel_structs::fft_1d_1 = 51

  Flag to mark execution of 1D FFTs along the first dimension.
- integer, parameter, public ez_parallel_structs::fft_1d_2 = 46

  Flag to mark execution of 1D FFTs along the second dimension.
- integer, parameter, public ez_parallel_structs::fft_2d = 95

  Flag to mark execution of 2D FFTs.
- integer, parameter, public ez_parallel_structs::spec_drv_1d_1 = 23

  Flag to mark execution of 1D spectral derivatives along the first dimension.
- integer, parameter, public ez_parallel_structs::spec_drv_1d_2 = 78

  Flag to mark execution of 1D spectral derivatives along the second dimension.

6.14 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/max_val.f90 File Reference

Functions/Subroutines

- subroutine max_val_sbr (subGrid, maxValue, sch)

  The max value subroutine.
- subroutine max_val_eh (sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.14.1 Function/Subroutine Documentation

6.14.1.1 max_val_eh()

```
subroutine max_val_sbr::max_val_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized

Definition at line 52 of file max_val.f90.

6.14.1.2 max_val_sbr()

```
subroutine max_val_sbr (
             double precision, dimension(0:,0:), intent(in) subGrid,
             double precision, intent(inout) maxValue,
             type(scheme), intent(in) sch )
```

The max value subroutine.

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in	*subGrid*	The local sub-grid.
in,out	*maxValue*	The variable to store the maximum value across all sub-grids.
in	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 14 of file max_val.f90.

6.15 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/min_val.f90 File Reference

Functions/Subroutines

- subroutine min_val_sbr (subGrid, minValue, sch)

  The min value subroutine.
- subroutine min_val_eh (sch)

  The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.15.1 Function/Subroutine Documentation

6.15.1.1 min_val_eh()

```
subroutine min_val_sbr::min_val_eh (
             type(scheme), intent(in) sch )
```

The error handling subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
  1. is not initialized

Definition at line 52 of file min_val.f90.

6.15.1.2 min_val_sbr()

```
subroutine min_val_sbr (
            double precision, dimension(0:,0:), intent(in) subGrid,
            double precision, intent(inout) minValue,
            type(scheme), intent(in) sch )
```

The min value subroutine.

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in	*subGrid*	The local sub-grid.
in,out	*minValue*	The variable to store the minimum value across all sub-grids.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 14 of file min_val.f90.

6.16 C:/Users/Owner/Education and Research/Graduate School/Parallelization Module Project/EZ_PARALLEL_project/EZ_PARALLEL/share_subgrid_bdry.f90 File Reference

Functions/Subroutines

- subroutine share_subgrid_bdry_dble_sbr (subGrid, sch)

  The sub-grid boundary sharing subroutine (DOUBLE PRECISION).
- subroutine share_subgrid_bdry_dble_eh (sch)

  The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:
- subroutine share_subgrid_bdry_dcmpx_sbr (subGrid, sch)

  The sub-grid boundary sharing subroutine (DOUBLE COMPLEX).
- subroutine share_subgrid_bdry_dcmpx_eh (sch)

  The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

6.16.1 Function/Subroutine Documentation

6.16.1.1 share_subgrid_bdry_dble_eh()

```
subroutine share_subgrid_bdry_dble_sbr::share_subgrid_bdry_dble_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
    1. is not initialized

Definition at line 107 of file share_subgrid_bdry.f90.

6.16.1.2 share_subgrid_bdry_dble_sbr()

```
subroutine share_subgrid_bdry_dble_sbr (
            double precision, dimension(:,:), intent(inout) subGrid,
            type(scheme), intent(in) sch )
```

The sub-grid boundary sharing subroutine (DOUBLE PRECISION).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

`in,out`	*subGrid*	The local sub-grid whose boundary will be shared.
`in`	*sch*	`SCHEME` that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 12 of file share_subgrid_bdry.f90.

6.16.1.3 share_subgrid_bdry_dcmpx_eh()

```
subroutine share_subgrid_bdry_dcmpx_sbr::share_subgrid_bdry_dcmpx_eh (
            type(scheme), intent(in) sch )
```

The error handling subroutine for the sub-grid boundary sharing subroutine. Aborts the program if any of the input parameters satisfy any of the following conditions:

- sch...
    1. is not initialized

Definition at line 246 of file share_subgrid_bdry.f90.

6.16.1.4 share_subgrid_bdry_dcmpx_sbr()

```
subroutine share_subgrid_bdry_dcmpx_sbr (
            double complex, dimension(:,:), intent(inout) subGrid,
            type(scheme), intent(in) sch )
```

The sub-grid boundary sharing subroutine (DOUBLE COMPLEX).

Author

Jason Turner, University of Wisconsin-Madison

Parameters

in,out	*subGrid*	The local sub-grid whose boundary will be shared.
in	*sch*	SCHEME that will be used for the grid decomposition, parallel FFTs, etc.

Definition at line 151 of file share_subgrid_bdry.f90.

Index