

[Help](#)

```

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else
/*****
*   CPS - A simple C PDE solver                                     *
*                                                                 *
*   Copyright (c) 2007,                                           *
*   Maya Briani          <m.briani@iac.rm.cnr.it>,               *
*   Francesco Ferreri    <francesco.ferreri@gmail.com>,          *
*   Roberto Natalini    <r.natalini@iac.rm.cnr.it>,              *
*   Marco Papi          <m.papi@iac.rm.cnr.it>                   *
*                                                                 *
*****/
#ifndef GRID_H
#define GRID_H

#include "cps_types.h"
#include "cps_dimensions.h"

#define ITER_NONE  0x00
#define ITER_CORE  0x11
#define ITER_PLAIN  0x12
#define ITER_TIME  0x1F

struct grid_t
{

    int space_dimensions;

    /* grid parameters */
    double min_value[MAX_DIMENSIONS];
    double current_value[MAX_DIMENSIONS];
    int current_order;
    double max_value[MAX_DIMENSIONS];
    double delta[MAX_DIMENSIONS];
    int ticks[MAX_DIMENSIONS];
    int current_tick[MAX_DIMENSIONS];
    int current_iterator[MAX_DIMENSIONS];

    /* focus */

```

```
double focus[MAX_DIMENSIONS];
int focus_tick[MAX_DIMENSIONS];

/* tuning */
grid_tuner *tuner;

/* status access */
int is_tuned;
int is_rescaled;
};

int grid_create(grid **);
int grid_destroy(grid **);
int grid_rescale(grid *);

/* setters */
int grid_set_tuner(grid *, grid_tuner *);
int grid_set_focus(grid *, int, double);
int grid_set_space_dimensions(grid *, int);
int grid_set_min_value(grid *, int, double);
int grid_set_max_value(grid *, int, double);
int grid_set_ticks(grid *, int, int);
int grid_set_iterator(grid *, int, int);
int grid_set_all_iterators(grid *, int);

/*
 * iterators
 */
int grid_iterator_span(const grid *, int);
int grid_iterator_first(const grid *, int);
int grid_iterator_last(const grid *, int);

/* time */
int grid_time_initial(grid *);
int grid_time_start(grid *);
int grid_time_forth(grid *);
int grid_time_after(const grid *);

/* plain space */
int grid_plain_start(grid *, int);
int grid_plain_forth(grid *, int);
```

```
int grid_plain_after(const grid *, int);

/* core space */
int grid_core_start(grid *, int);
int grid_core_forth(grid *, int);
int grid_core_after(const grid *, int);

/* generic iterator-type dependant */
int grid_space_start(grid *);
int grid_space_forth(grid *);
int grid_space_after(const grid *);

/* guard space */
int grid_guard_start(grid *);
int grid_guard_forth(grid *);
int grid_guard_after(const grid *);

/*
 * node retrieval and access
 */
int grid_item(const grid *, grid_node **);
int grid_loose_item(const grid *, int, int, grid_node **);
int grid_plain_item(const grid *, grid_node **);
int grid_focus_item(const grid *, grid_node **);
int grid_node_neighbour(const grid *, int, const grid_node *, grid_node **);
#endif

#endif //PremiaCurrentVersion
```