GRAPE-DR G5 compatibility library reference manual

Ver 0.0 — 2010/9/20
Ver 0.01 — 2010/9/22
Ver 0.01a — 2010/12/11
Ver 0.02 — 2011/1/5

Jun Makino

January 20, 2011

Contents

1 Common utilily functions

1.1 g5_report_timers

```
void g5_report_timers()
```

Print out some performance information.

1.2 g5_set_debug_level

```
void g5_set_debug_level(int level);
```

Enable debug messages.

```
level 0 (default): No message
```

Currently, only levels supported are 0 and 1.

2 Desctription of single-chip interface functions

2.1 g5_openMC

```
void g5_openMC(int devid)
```

Get one chip and initialize it.

2.2 g5_closeMC

```
void g5_closeMC(int devid)
```

Release the card.

2.3 g5_set_rangeMC

```
void g5_set_rangeMC(int devid, double xmin, double xmax, double mmin)
```

Do nothing. Not required for GRAPE-DR implementation

2.4 g5_set_jpMC

```
void g5_set_jpMC(int devid, int adr, int nj, double *m,
                 double (*x)[3])
```

Store j-particles (particles which exert forces) to the memory of chip devid. The argument adr is the first index in the on-board memory, and nj is the number of particles sent. Particles are sent to consective locations of the memory.

In current implementation, adr must be zero, and nj must be the same as n set by g5_set_n. This is the case for other variants of set_jp.

2.5 g5s_set_jpMC

```
void g5s_set_jpMC(int devid, int adr, int nj, double *m,
                 double (*x)[3], double *eps2)
```

This is a new function, which is not in original G5 or G7 implementation. The difference from g5_set_jpMC is that this function can store epsilon squared for each particle, which is used in symmetrized form ($r_{ij}^2 + \epsilon_i^2 + \epsilon_j^2$) in force calculation.

2.6 g5n_set_jpMC

```
void g5n_set_jpMC(int devid, int adr, int nj, double *m,
                 double (*x)[3], int *index)
```

This is a new function, which is not in original G5 or G7 implementation. The difference from g5_set_jpMC is that this function can store indices of particles, which are used to return the index of the nearest neighbour.

2.7 g5sn_set_jpMC

```
void g5n_set_jpMC(int devid, int adr, int nj, double *m,
                 double (*x)[3], double * eps2, int *index)
```

This is a new function, which is not in original G5 or G7 implementation. This function can set both eps2 and index. If the value of either of eps2 or index is NULL, the corresponding value is not set (actually 0 is set).

2.8 g5sn_calculate_force_on_xMC

```
void g5sn_calculate_force_on_xMC(int devid, double (*x)[3], double * eps2,
```

```
                                    int * index,
                                    double (*a)[3], double *p,
                                    double * rnnb2,
                                    int * innb, int ni)
```

Calculate force and potential on particles. The number of particles is given by ni, and ni can be larger than the logical number of pipelines. This function automatically calls g5_set_xiMC, g5_runMC and g5n_get_forceMC multiple times to get forces on all particles. This function requires the indices and softening parameters of i-particles passed as arrays, and returns the array of nearest neighbour indices and the distances (squared and include softenings) to them. If eps2 is NULL, values set by g5_set_eps2MC or g5_set_eps2_to_allMC (if any value has been set) are used. If index is NULL, undefined values of indices will be sent to GRAPE-DR, and neither rnnb2 nor innb is set.

2.9 g5n_calculate_force_on_xMC

```
void g5sn_calculate_force_on_xMC(int devid, double (*x)[3],
                                    int * index,
                                    double (*a)[3], double *p,
                                    double * rnnb2,
                                    int * innb, int ni)
```

Same as g5sn_calculate_force_on_xMC called with eps2=NULL.

2.10 g5s_calculate_force_on_xMC

```
void g5sn_calculate_force_on_xMC(int devid, double (*x)[3], double * eps2,
                                    double (*a)[3], double *p,
                                    double * rnnb2,
                                    int * innb, int ni)
```

Same as g5sn_calculate_force_on_xMC called with index=NULL.

2.11 g5_calculate_force_on_xMC

```
void g5sn_calculate_force_on_xMC(int devid, double (*x)[3],
                                    double (*a)[3], double *p,
                                    double * rnnb2,
                                    int * innb, int ni)
```

Same as g5sn_calculate_force_on_xMC called with index=NULL and eps2=NULL.

2.12 g5_set_xiMC

```
void g5_set_xiMC(int devid, int ni, double (*x)[3])
```

Store the positions of i-particles (particles which receive forces) to the on-chip registers (from API point of view, at least. Actual communication might take place elsewhere).

The number of particles, ni, must not exceed the number of pipelines (returned by g5_get_number_of_pipelinesMC).

2.13 g5_runMC

```
void g5_runMC(int devid)
```

Let GRAPE-DR hardware calculate.

2.14 g5_set_nMC

```
void g5_set_nMC(int devid, int n){nja[devid]=n;}
```

Set number of j-particles. Should be called at least once before calling g5_runMC.

2.15 g5_set_eps2MC

```
void g5_set_eps2MC(int devid, int ni, double *eps2)
```

Similar to g5_set_xiMC but set the softening (squared).

2.16 g5_set_eps2_to_allMC

```
void g5_set_eps2_to_allMC(int devid, double eps2)
```

Similar to g5_set_eps2MC but set the same value to all pipelines.

2.17 g5_set_indexMC

```
void g5_set_indexMC(int devid, int ni, int *index)
```

Similar to g5_set_xMC but set the index. This index is used to avoid finding the particle itself for the nearest neighbour.

2.18 g5_get_forceMC

```
void g5_get_forceMC(int devid, int ni, double (*a)[3],
                    double *pot)
```

Returns the calculated force and potential.

2.19 g5n_get_forceMC

```
void g5n_get_forceMC(int devid, int ni, double (*a)[3],
                    double *pot, double* rnnb2, int* innb)
```

Returns the calculated force and potential, and distance squared to the nearest neighbour (including softening) and their indices.

2.20 g5_get_number_of_pipelinesMC

```
int g5_get_number_of_pipelinesMC(int devid)
```

Returns the logical number of pipelines (max value for ni in g5_set_xi etc).

2.21 g5_get_jmemsizeMC

```
int g5_get_jmemsizeMC(int devid)
```

Returns the max number of j-particles.

2.22 Unsupported Functions

The following functions have entries, but do nothing.

```
void g5_set_etaMC(int devid, double eta){}
void g5_set_h_to_allMC(int devid, double h){}
void g5_set_hMC(int devid, int ni, double *h){}
int g5_read_neighbor_listMC(int devid){return 0;}
int g5_get_neighbor_listMC(int devid, int ip, int *list){return 0;}
int g5_get_nbmemsizeMC(int devid){return 0;}
int g5_set_nbmemsizeMC(int devid, int size){return 0;}
```

3 Desctription of multi-chip interface functions

Multi-chip functions do not have "MC". This rather illogical naming convention is used to make this library compatible with what have been used in CfCA.

3.1 g5_open

```
void g5_open(void)
```

Get one card and initialize it.

3.2 g5_close

```
void g5_close(void)
```

Release the card.

3.3 g5_set_range

```
void g5_set_range(double xmin, double xmax, double mmin)
```

Does not seem to have been implemented.

3.4 g5_set_jp

```
void g5_set_jp(int adr, int nj, double *m, double (*x)[3])
```

Store j-particles (particles which exert forces) to the memory of chip devid. The argument adr is the first index in the on-board memory, and nj is the number of particles sent. Particles are sent to consective locations of the memory.

In current implementation, adr must be zero, and nj must be the same as n set by g5_set_n.

3.5 g5s_set_jp

```
void g5s_set_jp(int adr, int nj, double *m,
                double (*x)[3], double *eps2)
```

This is a new function, which is not in original G5 or G7 implementation. The difference from g5_set_jp is that this function can store epsilon squared for each particle, which is used in symmetrized form ($r_{ij}^2 + \epsilon_i^2 + \epsilon_j^2$) in force calculation.

3.6 g5n_set_jp

```
void g5n_set_jp(int adr, int nj, double *m,
                double (*x)[3], int *index)
```

This is a new function, which is not in original G5 or G7 implementation. The difference from g5_set_jp is that this function can store indices of particles, which are used to return the index of the nearest neighbour.

3.7 g5_set_xi

```
void g5_set_xi(int ni, double (*x)[3])
```

Store the positions of i-particles (particles which receive forces) to the on-chip registers (from API point of view, at least. Actual communication might take place elsewhere).

The number of particles, ni, must not exceed the number of pipelines (returned by g5_get_number_of_pipelinesMC).

3.8 g5_run

```
void g5_run(void)
```

Let GRAPE-DR hardware calculate.

3.9 g5_set_n

```
void g5_set_n(int n)
```

Set number of j-particles. Should be called at least once before calling g5_runMC.

In multi-chip library, n set by g5_set_n must be the same as nj used in g5_set_jp.

3.10 g5_set_eps2

```
void g5_set_eps2(int ni, double *eps2)
```

Similar to g5_set_x but set the softening (squared).

3.11 g5_set_eps2_to_all

```
void g5_set_eps2_to_all(double eps2)
```

Similar to g5_set_eps2 but set the same value to all pipelines.

3.12 g5_set_index

```
void g5_set_index(int ni, int *index)
```

Similar to g5_set_x but set the index. This index is used to avoid finding the particle itself for the nearest neighbour.

3.13 g5_get_force

```
void g5_get_force(int ni, double (*a)[3], double *pot)
```

Returns the calculated force and potential.

3.14 g5n_get_force

```
void g5n_get_force(int ni, double (*a)[3], double *pot,
                   double* rnnb2, int* innb)
```

Returns the calculated force and potential, and distance squared to the nearest neighbour and their indices.

3.15 g5_get_number_of_pipelines

```
int g5_get_number_of_pipelines(void)
```

Returns the logical number of pipelines (max value for ni in g5_set_xi etc).

3.16 g5_get_jmemsize

```
int g5_get_jmemsize(void)
```

Returns the max number of j-particles.

3.17 g5_calculate_force_on_x

```
void g5_calculate_force_on_x(double (*x)[3],
                             double (*a)[3],
                             double *p, int ni)
```

Calculate force and potential on particles. The number of particles is given by ni, and ni can be larger than the logical number of pipelines. This function

automatically calls g5_set_xi, g5_run and g5_get_force multiple times to get forces on all particles. The following is a sample function to obtain gravitational forces for a system of n particles.

```
void calculate_accel_by_g5(int n,
                           double x[][3],
                           double m[],
                           double a[][3],
                           double pot[],
                           double eps2)
{
    static first=1;
    int i, niparallel;
    double epsinv = 1.0/sqrt(eps2);
    g5_set_debug_level(0);
    if (first){
        g5_open();
        first=0;
    }
    g5_set_jp(0,  n, m, x);
    g5_set_n(n);
    g5_set_eps2_to_all(eps2);
    g5_calculate_force_on_x(x, a, pot, n);
    for(i=0;i<n;i++) pot[i] += m[i]*epsinv;
}
```

3.18 g5n_calculate_force_on_x

```
void g5n_calculate_force_on_x(double (*x)[3], int * index,
                              double (*a)[3], double *p,
                              double * rnnb2,
                              int * innb, int ni)
```

Same as g5_calculate_force_on_x but returns nearest neighbours as well.

3.19 g5sn_calculate_force_on_x

```
void g5sn_calculate_force_on_x(double (*x)[3], double * eps2, int * index,
                               double (*a)[3], double *p,
                               double * rnnb2,
                               int * innb, int ni)
```

Same as g5_calculate_force_on_x but returns nearest neighbours as well. Also, one can set differnt values for eps2. If eps2 is NULL, value set by g5_set_eps2 (or g5_set_eps2_all) will be used.

3.20 Unsupported Functions

The following functions have entries, but do nothing.

```
void g5_set_etaMC(int devid, double eta){}
void g5_set_h_to_allMC(int devid, double h){}
void g5_set_hMC(int devid, int ni, double *h){}
int g5_read_neighbor_listMC(int devid){return 0;}
int g5_get_neighbor_listMC(int devid, int ip, int *list){return 0;}
int g5_get_nbmemsizeMC(int devid){return 0;}
int g5_set_nbmemsizeMC(int devid, int size){return 0;}
```

4 Sample usage

The following is the samle code for direct calculation of force, both for single-chip and multi-chip libraries.

```
#include <stdio.h>
#include <math.h>

#include "g5sim-gdr.h"


int devid = 1;

void calculate_accel_by_g5MC(int n,
                             double x[][3],
                               double m[],
                               double a[][3],
                               double pot[],
                               double eps2)
{
    static first=1;
    int i, niparallel;
    double epsinv = 1.0/sqrt(eps2);
    if (first){
        g5_openMC(devid);
        first=0;
```

```
    }
    g5_set_jpMC(devid, 0,  n, m, x);
    niparallel = g5_get_number_of_pipelinesMC(devid);
    for (i=0; i< n;i+=niparallel){
        int k;
        int nireal = niparallel;
        if (i+nireal > n) nireal = n - i;
        g5_set_xiMC(devid,  nireal, x+i);
        g5_set_eps2_to_allMC(devid, eps2);
        g5_set_nMC(devid, n);
        g5_runMC(devid);
        g5_get_forceMC(devid, nireal, a+i, pot+i);
    }
    for(i=0;i<n;i++) pot[i] += m[i]*epsinv;
}

void calculate_accel_by_g5(int n,
                                double x[][3],
                                double m[],
                                double a[][3],
                                double pot[],
                                double eps2)
{
    static first=1;
    int i, niparallel;
    double epsinv = 1.0/sqrt(eps2);
    if (first){
        g5_open();
        first=0;
    }
    g5_set_jp(0,  n, m, x);
    niparallel = g5_get_number_of_pipelinesMC(devid);
    for (i=0; i< n;i+=niparallel){
        int k;
        int nireal = niparallel;
        if (i+nireal > n) nireal = n - i;
        g5_set_xi(nireal, x+i);
        g5_set_eps2_to_all(eps2);
        g5_set_n(n);
        g5_run();
        g5_get_force(nireal, a+i, pot+i);
    }
    for(i=0;i<n;i++) pot[i] += m[i]*epsinv;
}
```

A complete working N-body code is included in the library source directory ($GDRBASEDIR/gdrtb3/singlibMC_tb4/g5work). The above functions are in the file grapeg5.c

5 Compiling and Linking

The header file is g5sim-gdr.h. So you need

```
#include "g5sim-gdr.h"
```

and compile option

```
-I$GDRBASEDIR/lib [In Makefile, -I$(GDRBASEDIR)/lib]
```

To link, use

```
-L$GDRBASEDIR/lib -lgdrgrav -lsing -lhib
[In Makefile, -L$(GDRBASEDIR)/lib -lgdrgrav -lsing -lhib]
```

6 Limitations etc.

The multi-chip version assumes 4-chip, single card.

All set_jp functions requires that adr=0. (Version after 2011/1)

7 Changes

7.1 2010/9/22

Functions g5s_set_jp(MC) added.

7.2 2010/9/30

Signal handler added. SIGINT is trapped internally in the library. One should see the message like

```
^CSIG interrupt - delayed exit
```

When the program is stopped by Ctrl-C. This should make the hardware more stable.

7.3 2011/1/5

Functions for nearest neighbour handling added and tested.

7.4 2011/1/20

g5(xx)_set_jp(yy) functions updated. Only adr=0 works now.

8 Known problems

Calling g5_open and g5_close many times seem to cause error. Avoid calling them more than 10 times.

4-chip version does not work with nj>50k. (version before 2010/9/21)

Calling MC version asynchronously from multiple processes can cause hardware errors. (version before 2010/11/24)

g5n_ functions implemented but not tested yet (version before 2011/1/5)

9 Things to do

- implement true 32-bit float interface for the data of i- and j- particles. This requires the change of FPGA code.
- implement programmable cutoff function. This requires update of the assembly code.
- implement overlapped communication and calculation (in particular for j-particles).

Overlapped communication is a bit complex.