并行语言公共运行时系统

外部接口

版本 3.5

并行语言编译课题组

2020 年 7 月

目录

1 概述

本文档主要介绍并行语言公共运行时系统节点内相关功能的外部接口。接口主要分为主核和从核两部分，均包含多个模块。相关接口函数的命名规则为：

CRTS_子模块[_函数补充说明]

其中，

- CRTS：为并行语言公共运行时系统缩写，并行语言公共运行时系统的所有外部接口函数均以“并行语言公共前缀开头，全为大写字母
- 子模块：全为小写字母
- 函数补充说明：与“子模块”一起可表明该函数的功能，为小写字母和下划线“_”

如 CRTS_ssync_array 中的“ssync”代表从核同步（slave synchronization），“array”代表同步范围为从核阵列，因此 CRTS_ssync_array 表示从核阵列同步。

调用并行语言公共运行时系统的外部接口时，应当包含头文件 crts.h，即：

```
#include <crts.h>
```

基础编译器将自动连接相应的函数库。

2 主核接口

2.1 初始化

函数原型：

```
int CRTS_init (void);
```

函数说明：

完成并行语言公共运行时的初始化，在使用主核同步和任何从核功能之前必须调用该函数。

参数说明：

无。

返回值：

成功：0；

失败：非 0。

2.2 加速运算管理（athread）

2.2.1 创建单线程

函数原型：

```
int CRTS_athread_create (int id, void(*fp)(void),
                         void *arg);
int athread_create (int id, void(*fp)(void),
                    void *arg);
```

函数说明：

在当前进程中添加新的受控线程。线程执行的入口函数为 fpc。fpc 的参数列表由可变参数列表 arg 指定。

参数说明：

id：　线程绑定的从核号；

fp：　从核入口函数指针；

arg：　函数 fp 的参数列表。

返回值：

成功：返回 0；

失败：返回 1 表示该从核正执行其他线程；返回 2 表示该从核已执行完其他线程但主核并未完成 CRTS_athread_wait 操作。

2.2.2 等待单线程终止

函数原型：

```
int CRTS_athread_wait (int id);
int athread_wait (int id);
```

函数说明：

主核上的进程以阻塞方式调用，等待从核上的线程执行完毕。

参数说明：

id：　线程绑定的从核号。

返回值：

成功：0；

失败：非 0。

2.2.3 释放单线程资源

函数原型：

```
int CRTS_athread_end (int id);
int athread_end (int id);
```

函数说明：

在确定线程所占从核无相关作业后，停滞从核流水线，关闭从核，该从核在本进程中将无法再次使用。

参数说明：

id：　线程绑定的从核号。

返回值：

成功：0；

失败：非 0。

2.2.4 创建线程组

函数原型：

```
int CRTS_athread_spawn (void(*fp)(void), void *arg);
int athread_spawn (void(*fp)(void), void *arg);
```

函数说明：

创建在从核组上运行的线程组，启动核组中的所有可用从核资源。从核组执行的入口函数由为 fp。fp 的参数起始地址为 arg。从核在执行完 fp 函数之后，自动将从核 cache 内容写回主存。

参数说明：

fp：　从核入口函数指针；

arg： 函数 fp 的参数起始地址。

返回值：

成功：0；

失败：返回 1 表示默认从核组正执行其他线程；返回 2 表示该从核组已执行完其他线程但主核并未完成 CRTS_athread_join 操作。

2.2.5 不写回 cache 内容的创建线程组

函数原型：

```
int CRTS_athread_spawn_noflush (void(*fp)(void), void *arg);
int athread_spawn_noflush (void(*fp)(void), void *arg);
```

函数说明：

创建在从核组上运行的线程组，启动核组中的所有可用从核资源。从核组执行的入口函数由为 fp。fp 的参数为 arg。与 CRTS_athread_spawn()不同的是，从核在执行完 fp 函数之后，不会将从核 cache 内容写回主存，以更快结束线程运行。

参数说明：

fp： 从核入口函数指针；

arg： 函数 fp 的参数起始地址。

返回值：

成功：0；

失败：返回 1 表示默认从核组正执行其他线程；返回 2 表示该从核组已执行完其他线程但主核并未完成 CRTS_athread_join 操作。

2.2.6 等待线程组终止

函数原型：

```
int CRTS_athread_join (void);
int athread_join (void);
```

函数说明：

主核上的进程以阻塞方式调用该函数，直到由 CRTS_athread_spawn 创建的线程组执行完毕。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

2.2.7 释放线程组资源

函数原型：

```
int CRTS_athread_halt (void);
int athread_halt (void);
```

函数说明：

释放由 CRTS_athread_spawn 创建的线程组资源，停滞从核组流水线，关闭从核组，该从核组在本进程无法再次使用。

参数说明：

无。

返回值：

成功：0；

失败：非 0。

注意事项：

必须保证从核组无任何用户作业才能使用该函数。

2.2.8 获取当前任务使用的线程数量

函数名：

```
int CRTS_athread_get_max_threads (void);
int athread_get_max_threads (void);
```

函数说明：

获取当前任务使用的线程数量，正常返回值范围为[0, 64]。

参数说明：

无。

返回值：

成功：当前任务使用的线程数量[0-64]；

失败：-1。

2.3 同步（sync）

2.3.1 节点内所有主核同步

函数原型：

```
void CRTS_sync_node (void);
```

函数说明：

本节点内所有主核同步。

参数说明：

无。

返回值：

无。

2.3.2 主核与单从核同步

函数原型：

```
void CRTS_sync_master_spe (int tid);
```

函数说明：

主核与本地核组内由 tid 指定的从核同步。当主核调用该函数时，相应从核必须调用 CRTS_ssync_master_spe()与主核匹配。

参数说明：

tid：　本地从核编号。

返回值：

无。

2.3.3 主核与核组同步

函数原型：

```
void CRTS_sync_master_array (void);
```

函数说明：

主核与本地核组同步。必须主核与本核组内从核都调用该函数，相关同步操作才能正常完成，但调用该函数的从核范围为创建线程时由对应的 CRTS_athread_spawn，CRTS_athread_spawn_mask 或者 CRTS_athread_spawn_create 所指定。

参数说明：

无。

返回值：

无。

2.4 互斥（mutex）

2.4.1 节点默认锁

函数原型：

```
int CRTS_mutex_lock_node (void);
int CRTS_mutex_trylock_node (void);
int CRTS_mutex_unlock_node (void);
```

函数说明：

节点默认锁加锁、尝试加锁和解锁，用于节点（即单个 CPU）内所有进程的互斥。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

2.4.2 自定义锁

函数原型：

```
int CRTS_mutex_lock (volatile int64_t *lock);
int CRTS_mutex_trylock (volatile int64_t *lock);
int CRTS_mutex_unlock (volatile int64_t *lock);
```

函数说明：

进程对指定地址进行锁操作，使用主存地址可实现本地锁，使用交叉段地址可实现节点锁。

参数说明：

lock：自定义锁的地址。

返回值：

成功： 0；

失败：非 0。

2.5 计时（time）

2.5.1 获取节拍计数

函数原型：

```
unsigned long CRTS_time_cycle (void);
```

函数说明：

返回主核当前的周期计数器的值（节拍数）。

参数说明：

无。

返回值：

节拍计数。

2.6 信号(sig)

2.6.1 处理从核信号

函数名：

```
void CRTS_sig_user_init (void*(*funp) (void*));
```

函数说明：

支持用户的主从信号机制。主核调用本接口以指定信号处理接口 funp，接口 funp 由用户定义，读取从核发送信号（见 3.7.1）时填写的主存数据地址进行处理。

参数说明：

funp：　　用户定义的主核信号处理函数，负责处理从核信号。

返回值：

无。

2.7 动态任务分配(task)

动态任务分配包含动态任务获取函数 CRTS_task_fetch 和动态任务退出函数 CRTS_task_quit，同时提供了课题运行状态查询功能。

2.7.1 动态任务获取

函数名：

```
long CRTS_task_fetch(char* mid_file_name, long task_num, long block_size);
```

函数说明：

该函数的功能是动态获取任务。

CRTS_task_fetch 任务计数器的初值是 0。调用 CRTS_task_fetch 时，如果全局任务计数器的值小于总任务数 task_num，CRTS_task_fetch 将返回任务计数器的当前值（0～task_num-1 之间的一个唯一值，第一个调用者的返回值为 0），并且将任务计数器的值加 1；否则，CRTS_task_fetch 返回负数。动态任务分配目前仅支持全局进程范围。

参数说明：

char* mid_file_name：记录任务完成情况的断点文件名，不能为空；

long task_num：总任务数；

long block_size：任务分配的块大小，表示每次获取任务的数目；

返回值：

计数器的值小于 task_num 时返回计数器的当前值，全部任务都已正常完成时返回-1，失败时返回其他负值。

详细说明：

CRTS_task_fetch 支持以任务块为单位的任务预取，用户可以通过 block_size 参数指定合适的预取粒度。如果 block_size 值大于 1，表示调用一次 CRTS_task_fetch 就取得 block_size 个任务到本地，函数返回一个起始任务号，其余的 block_size-1 个任务放在本地

任务缓冲中，下次调用 CRTS_task_fetch，将直接消费本地缓冲中的任务，直到本地缓冲中无任务，才再次远程申请 block_size 个任务，以减少远程申请任务的开销。

每当调用 CRTS_task_fetch 进行远程任务申请时，表明上一次调用得到的 block_size 个任务已经完成，断点文件 mid_file_name 将记录上一次远程调用获取的计数器的值。断点文件的记录以 block_size 为单位，每个字节表示 block_size 个任务的完成情况，在断点文件中用“1”表示完成（占用 1 字节），断点文件的长度为(task_num + block_size – 1)/block_size。因为断点文件中详细记录了已经完成的任务号，所以在程序运行出错中止后，重新启动任务，将首先读取断点文件内容继续完成全部的 task_num 个任务。

为保证所有的任务都完成，CRTS_task_fetch 书写格式有严格要求，应为：

while((i= CRTS_task_fetch(mid_file_name, task_num, block))>=0) { … }

其中 i 必须是有符号类型，while 继续运行的条件是 i>=0，其余格式均非法。

注意事项：

CRTS_task_fetch()为集合操作，要求所有进程都调用。

2.7.2 动态任务退出

函数名：

```
int CRTS_task_quit() ;
```

函数说明：

在任务执行的过程中，当某进程找到需要的结果后调用本接口，用于通知其他进程任务动态分配结束。一旦调用了 CRTS_task_quit()，断点文件会将所有的任务都标记为已经完成。

当某进程调用该函数后，其它进程再次调用 CRTS_task_fetch()，得到的返回值将变成负值，即使当前还有任务没有完成。该函数强制 CRTS_task_fetch()所在的 while 循环结束。

参数说明：

无

返回值：

成功，返回 0；失败，返回非 0 值。

2.7.3 任务分配辅助功能

2.7.3.1 断点信息查询

在 shell 环境下，通过 readmid + 断点文件名，可以输出总任务数以及当前断点文件记录的未完成任务数。

示例：readmid mid.dat

2.7.3.2 任务运行状态检查

在 shell 环境下， 针对使用动态任务分配的课题，公共运行时提供了以下方式检查任务运行状态，其中 jobid 为课题的作业号：

（1） bsignal –s 34 jobid：输出总任务数及当前未完成任务数；

（2） bsignal –s 35 jobid：在剩余任务较少时使用，列出未完成任务，以及正在进行该任务的节点。

3 从核接口

所有使用回答字的从核函数接口（部分 DMA 和 RMA 接口），均要求回答字位于 LDM 空间，且在数据的发送方和目标方上有同样的地址，建议定义为全局变量。例如：

```
__thread_local crts_rply_t rply;
……
CRTS_dma_iget (dst, src, len, &rply);
……
```

3.1 从核管理（smng）

公共运行时系统已在 LDM 中定义了部分常用的变量，用户可以直接使用，但不允许修改，目前有：

- CRTS_tid： 从核号（硬件编号，0-63）

- CRTS_rid: 从核所在行号（硬件编号，0-7）
- CRTS_cid: 从核所在列号（硬件编号，0-7）
- CRTS_row_size 从核所在行从核个数（1-8）
- CRTS_col_size 从核所在列从核个数（1-8）
- CRTS_cgn: 从核所在核组号（硬件编号，0-5）
- CRTS_spc_tid: 从核簇方式的逻辑从核号（0-63），可用于 LDM 共享
- CRTS_spcn: 从核簇号（0-15）

3.1.1 获取核组内从核号

函数原型：

```
char CRTS_smng_get_tid (void);
```

函数说明：

获取核组内从核编号（0-63）。

参数说明：

无。

返回值：

成功：从核编号；

失败：-1。

3.1.2 获取核组内从核行号

函数原型：

```
char CRTS_smng_get_rid (void);
```

函数说明：

获取核组内从核行号（0-7）。

参数说明：

无。

返回值：

成功：从核行号；

失败：-1。

3.1.3 获取簇方式的从核号

函数原型：

```
char CRTS_smng_get_spc_tid (void);
```

函数说明：

获取核组内以簇方式编号的从核号（0-63），可用于 LDM 共享。

参数说明：

无。

返回值：

成功：以簇方式编号的从核号；

失败：-1。

3.1.4 获取核组内从核列号

函数原型：

```
char CRTS_smng_get_cid (void);
```

函数说明：

获取核组内从核列号（0-7）。

参数说明：

无。

返回值：

成功：从核列号；

失败：-1。

3.1.5 获取核组号

函数原型：

```
char CRTS_smng_get_cgn (void);
```

函数说明：

获取核组编号（0-5）。

参数说明：

无。

返回值：

成功：核组编号（0-5）；

失败：-1。

3.1.6 获取从核簇编号

函数原型：

```
char CRTS_smng_get_spcn (void);
```

函数说明：

获取核组内从核簇的编号（0-15）。

参数说明：

无。

返回值：

成功：从核簇号；

失败：-1。

3.2 节点内从核同步（ssync）

3.2.1 从核点对点同步

函数原型：

```
void CRTS_ssync_peer (int tid);
```

函数说明：

从核点对点同步，即本从核与本核组内编号为 tid 的从核同步。同步的 2 个从核调用该函数时，必须互相以对方的从核编号为参数。

参数说明：

int tid: 本核组内与本从核同步的从核编号。

返回值：

无。

3.2.2 2 从核同步

函数原型：

```
void CRTS_ssync_2spe (void);
```

函数说明：

核组内编号除 2 后相等的 2 个从核同步，如 0 和 1 号从核，16 和 17 号从核。

参数说明：

无。

返回值：

无。

3.2.3 4 从核同步（即 1 个从核簇同步）

函数原型：

```
void CRTS_ssync_1spc (void);
```

函数说明：

核组内单个从核簇内的所有从核同步，主要用作 LDM 共享的协同。

参数说明：

无。

返回值：

无。

3.2.4 8 从核同步

3.2.4.1 从核行同步

函数原型：

```
void CRTS_ssync_8spe (void);
void CRTS_ssync_row (void);
```

函数说明：

核组内编号除 8 后相等的 8 个从核同步，如 0-7 号从核，16-23 号从核。

参数说明：

无。

返回值：

无。

3.2.4.2 从核列同步

函数原型：

```
void CRTS_ssync_col (void);
```

函数说明：

核组内编号模 8 后相等的 8 个从核同步，如 0/8/16/24/32/40/48/56 号从核。

参数说明：

无。

返回值：

无。

3.2.4.3 核组左半部从核列同步

函数原型：

```
void CRTS_ssync_col_left (void);
```

函数说明：

核组内列号小于 4，且编号模 8 后相等的 8 个从核同步，例如 0/8/16/24/32/40/48/56 号从核。

参数说明：

无。

返回值：

无。

3.2.4.4 核组右半部从核列同步

函数原型：

```
void CRTS_ssync_col_right (void);
```

函数说明：

核组内列号大于 3，且编号模 8 后相等的 8 个从核同步，例如 7/15/23/31/39/47/55/63 号从核。

参数说明：

无。

返回值：

无。

3.2.4.5 2 个从核簇同步

函数原型：

```
void CRTS_ssync_2spc (void);
```

函数说明：

核组内两个从核簇内的所有从核同步，主要用作 LDM 共享的协同。

参数说明：

无。

返回值：

无。

3.2.5 16 从核同步（也是 4 个从核簇同步）

16 从核同步与 4 个从核簇同步等价。

函数原型：

```
void CRTS_ssync_16spe (void);
void CRTS_ssync_4spc (void);
```

函数说明：

核组内编号除 16 相等的 16 个从核同步，如 0-15 号从核，16-31 号从核。可用作 4 个从核簇 LDM 共享、或者 2 个从核行的协同。

参数说明：

无。

返回值：

无。

3.2.6 32 从核同步（也是 8 个从核簇同步）

32 从核同步与 8 个从核簇同步等价。

函数原型：

```
void CRTS_ssync_32spe (void);
void CRTS_ssync_8spc (void);
```

函数说明：

核组内编号除 32 相等的 32 个从核同步，包括 0-31 号从核，32-63 号从核。

参数说明：

无。

返回值：

无。

3.2.7 从核核组同步

函数原型：

```
void CRTS_ssync_array (void);
```

函数说明：

从核核组同步。

参数说明：

无。

返回值：

无。

3.2.8 共享从核同步

函数原型：

```
void CRTS_ssync_sldm (void);
```

函数说明：

共享 ldm 的从核同步。

参数说明：

无。

返回值：

无。

3.2.9 单从核与主核同步

函数原型：

```
void CRTS_ssync_master_spe (void);
```

函数说明：

本从核与本地主核同步。当从核调用该函数时，本地相应的主核必须调用 CRTS_sync_master_spe()与从核匹配，函数实参为本从核编号。

参数说明：

无。

返回值：

无。

3.2.10 核组与主核同步

函数原型：

```
void CRTS_ssync_master_array (void);
```

函数说明：

本核组与主核同步，需本核组内所有从核、相应主核都调用该函数。

参数说明：

无。

返回值：

无。

3.2.11 节点内从核同步

函数原型：

```
void CRTS_ssync_node (void);
```

函数说明：

节点内所有从核同步，需节点（即单个 CPU）内所有线程（从核）都调用该函数。

参数说明：

无。

返回值：

无。

3.3 从核锁（smutex）

3.3.1 2 从核锁

函数原型：

```
int CRTS_smutex_lock_2spe (void);
int CRTS_smutex_unlock_2spe (void);
```

函数说明：

CRTS_spc_tid 除 2 相等的 2 个从核范围内的加锁、解锁，如 0 和 1 号从核，16 和 17 号从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.2 4 从核锁（即 1 个从核簇的锁）

函数原型：

```
int CRTS_smutex_lock_1spc (void);
int CRTS_smutex_unlock_1spc (void);
```

函数说明：

同一从核簇内（CRTS_spc_tid 除 4 相等）的 4 个从核范围内的加锁、解锁，如 CRTS_spc_tid 为 0-3 的从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.3 8 从核锁

3.3.3.1从核行锁

函数原型：

```
int CRTS_smutex_lock_8spe (void);
int CRTS_smutex_unlock_8spe (void);
int CRTS_smutex_lock_row (void);
int CRTS_smutex_unlock_row (void);
```

函数说明：

同一个从核行上（CRTS_tid 除 8 相等）的 8 个从核范围内的加锁、解锁，如 0-7 号从核，16-23 号从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.3.2从核列锁

函数原型：

```
int CRTS_smutex_lock_col (void);
int CRTS_smutex_unlock_col (void);
```

函数说明：

同一从核列上（CRTS_tid 模 8 相等）的 8 个从核范围内的加锁、解锁，如 0/8/16/24/32/40/48/56 号从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.3.3 2 个从核簇锁

函数原型：

```
int CRTS_smutex_lock_2spc (void);
int CRTS_smutex_unlock_2spc (void);
```

函数说明：

相邻两个从核簇内（CRTS_spc_tid 除 8 相等）的 8 个从核范围内的加锁、解锁，如 CRTS_spc_tid 为 0-7 的从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.4 16 从核锁（等同于 4 个从核簇的锁）

函数原型：

```
int CRTS_smutex_lock_16spe (void);
int CRTS_smutex_unlock_16spe (void);
int CRTS_smutex_lock_4spc (void);
int CRTS_smutex_unlock_4spc (void);
```

函数说明：

相邻 4 个从核簇内（CRTS_spc_tid 或 CRTS_tid 除 16 相等）的 16 个从核范围内的加锁、解锁，如 CRTS_spc_tid 或 CRTS_tid 为 0-15 的从核。

参数说明：

无。

返回值：

无。**返回值：**

成功： 0；

失败：非 0。

3.3.5 32 从核锁（等同于 8 个从核簇的锁）

函数原型：

```
int CRTS_smutex_lock_32spe (void);
int CRTS_smutex_unlock_32spe (void);
int CRTS_smutex_lock_8spc (void);
int CRTS_smutex_unlock_8spc (void);
```

函数说明：

相邻 8 个从核簇内（CRTS_spc_tid 或 CRTS_tid 除 32 相等）的 32 个从核范围内的加锁、解锁，如 CRTS_spc_tid 或 CRTS_tid 为 0-31 号从核。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.6 核组从核锁

函数原型：

```
int CRTS_smutex_lock_array (void);
int CRTS_smutex_unlock_array (void);
```

函数说明：

核组从核锁加锁、解锁，用于核组内所有线程（从核）的互斥。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.3.7 节点从核锁

函数原型：

```
int CRTS_smutex_lock_node (void);
int CRTS_smutex_unlock_node (void);
```

函数说明：

节点从核锁加锁、解锁，用于节点（即单个 CPU）内所有线程（从核）的互斥。

参数说明：

无。

返回值：

成功： 0；

失败：非 0。

3.4 从核私有 LDM 空间管理（pldm）

3.4.1 查询 LDM 动态分配空间起始地址

函数原型：

```
Void*  CRTS_get_free_addr ();
```

函数说明：

获取从核动态 LDM 私有空间的起始地址。

参数说明：

无

返回值：

从核动态 LDM 私有空间的起始地址。

3.4.2 查询 LDM 可动态分配空间大小

函数原型：

```
int CRTS_pldm_get_free_size (void);
int get_allocatable_size (void);
```

函数说明：

获取当前可动态分配的 LDM 空间大小。

参数说明：

无。

返回值：

当前可动态分配的 LDM 空间大小。

3.4.3 申请 LDM 私有空间

函数原型：

```
void *CRTS_pldm_malloc (size_t size);
void *ldm_malloc (size_t size);
```

函数说明：

从核动态申请 LDM 从核私有空间。

参数说明：

size_t size：申请主存空间的字节数。

返回值：

成功：返回申请空间的首地址；

失败：NULL。

3.4.4 释放 LDM 私有空间

函数原型：

```
void  CRTS_pldm_free (void *p, size_t size);
void  ldm_free (void *p, size_t size);
```

函数说明：

从核动态释放 LDM 私有空间。

参数说明：

void *p：释放 LDM 私有空间的首地址；

size_t size：申请主存空间的字节数。

返回值：

无。

3.4.5 申请全部可用 LDM 私有空间

函数原型：

```
void *CRTS_pldm_malloc (size_t* size);
void *ldm_malloc_max (size_t* size);
```

函数说明：

从核动态申请全部可用 LDM 私有空间。

参数说明：

size_t* size：用于存储申请空间的大小。

返回值：

成功：返回申请空间的首地址；

失败：NULL。

3.4.6 释放所有 LDM 私有空间

函数原型：

```
void  CRTS_pldm_free_all ();
void  ldm_free_all ();
```

函数说明：

从核释放所有 LDM 私有空间。

参数说明：

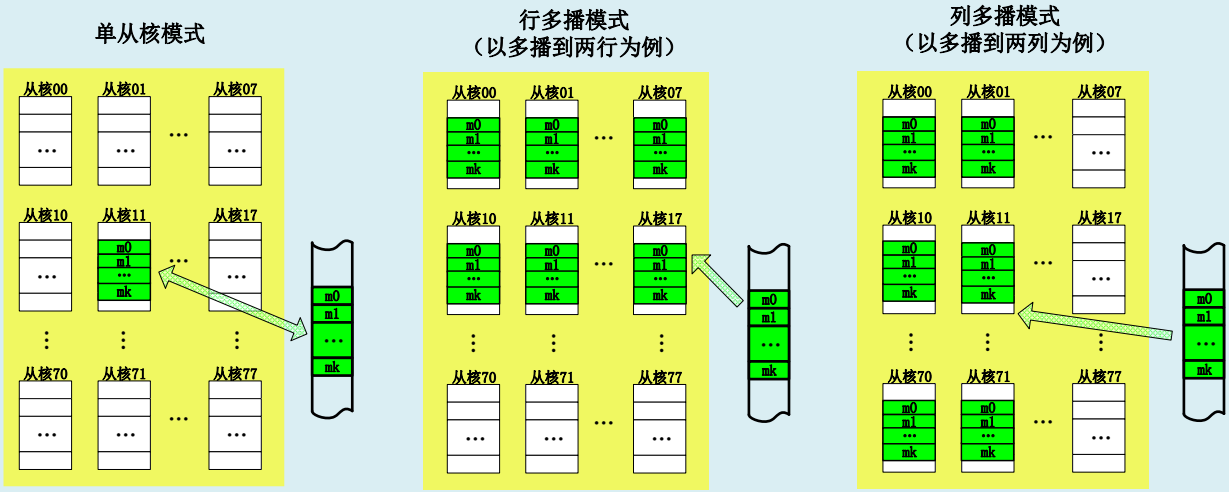
无。

返回值：

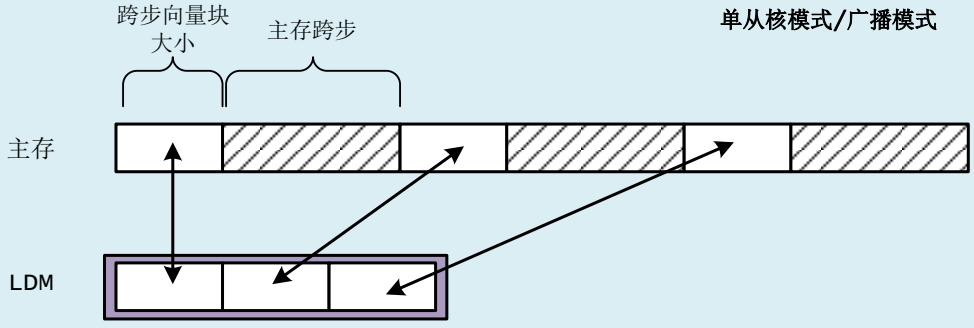
无。

3.5 DMA (dma)

DMA 是主存和 LDM 之间进行批量数据传输功能。DMA 只能由从核发起，支持单从核模式、行多播模式和列多播模式。各模式含义如下图：



单从核模式/广播模式主存跨步和跨步向量块大小含义如下：



主存跨步和跨步向量块大小描述了对主存带跨步的读写信息，表示对主存跨步访问时的跳跃间隔。主存跨步为 0 时，表示对主存的连续访问，此时跨步向量块大小的内容无意义。

根据实际使用场景，本系统将 DMA 接口封装为以下几类：

- DMA 读：单从核模式，从主存批量读取数据到 LDM。
- DMA 写：单从核模式，将 LDM 数据批量写入主存。
- DMA 行广播：行多播模式，从主存批量读取数据到同一行上所有从核的 LDM。
- DMA 列广播：列多播模式，从主存批量读取数据到同一列上所有从核的 LDM。
- DMA 广播：行多播模式，从主存批量读取数据到从核阵列所有从核的 LDM。

上述几类 DMA，本系统均提供阻塞/非阻塞、有跨步/无跨步等多种接口；同时对于行广播、列广播、广播等，还提供集合方式的接口（即要求所有需获取数据的从核都调用）。需特别注意的是，DMA 所有的数据地址、数据长度，都必须为 4 字节的整数倍，即 4 字节对界（对齐）。

3.5.1 DMA 读

函数原型：

a）阻塞接口

```
int CRTS_dma_get (void *dst, void *src, int len);
```

b）阻塞带跨步的接口

```
int CRTS_dma_get_stride (void *dst, void *src, int len,
                         int bsize, int stride);
```

函数说明：

单个从核发起从主存到本地局存（LDM）的 DMA 数据传输，阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

3.5.2 非阻塞 DMA 读

函数原型：

a）非阻塞接口

```
int CRTS_dma_iget (void *dst, void *src, int len,
                   crts_rply_t *rply);
```

b）非阻塞带跨步的接口

```
int CRTS_dma_iget_stride (void *dst, void *src, int len,
                   int bsize, int stride, crts_rply_t *rply);
```

函数说明：

单个从核发起从主存到本地局存的 DMA 数据传输，非阻塞方式，通常与函数 CRTS_dma_wait_value()配合使用。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）；

rply：　　回答字地址，必须 LDM 地址。

返回值：

成功：0；

失败：非 0。

3.5.3 DMA 等待完成

函数原型：

```
int CRTS_dma_wait_value (crts_rply_t *rply, int value);
```

函数说明：

该函数用于判断相应的 DMA 操作是否完成。当(*rply)的值大于等于 value 时，表明相应操作已经完成，即：

- 若调用该函数的从核为数据获取方，表明数据已被写入到本从核的 LDM；
- 若调用该函数的从核为数据供应方，表明数据已从本从核的 LDM 取走。

参数说明：

rply：　　DMA 传输回答字地址，必须局存地址；

value：　　需等待完成的非阻塞 DMA 次数，为自然数。

返回值：

成功：0；

失败：非 0。

3.5.4 DMA 写

函数原型：

a）阻塞接口

```
int CRTS_dma_put (void *dst, void *src, int len);
```

b）阻塞带跨步的接口

```
int CRTS_dma_put_stride (void *dst, void *src,int len,
                int bsize, int stride);
```

函数说明：

单个从核发起从本地 LDM 到主存的 DMA 数据传输，阻塞方式。

参数说明：

dst：　　主存目标地址（4B 对界/对齐）；

src：　　本地局存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

3.5.5 非阻塞 DMA 写

函数原型：

a）非阻塞接口

```
int CRTS_dma_iput (void *dst, void *src, int len,
                   crts_rply_t *rply);
```

b）非阻塞带跨步的接口

```
int CRTS_dma_iput_stride (void *dst, void *src, int len,
                   int bsize, int stride, crts_rply_t *rply);
```

函数说明：

单个从核发起从本地局存到主存的 DMA 数据传输，非阻塞方式，通常与函数 CRTS_dma_wait_value()配合使用。

参数说明：

dst：　主存目标地址（4B 对界/对齐）；

src：　本地局存源地址（4B 对界/对齐）；

len：　数据传输量（4B 整数倍）；

rply：　回答字地址，必须局存地址；

bsize：　跨步向量块大小（4B 整数倍）；

stride：　主存跨步（4B 整数倍）。

返回值：

返回值：

成功：0；

失败：非 0。

3.5.6 非阻塞 DMA 广播

函数原型：

a）非阻塞接口

```
int CRTS_dma_ibcast (void *dst, void *src, int len,
```

```
                crts_rply_t *rply);
```

b）非阻塞带跨步的接口

```
int CRTS_dma_ibcast_stride (void *dst, void *src, int len,
                int bsize, int stride, crts_rply_t *rply);
```

函数说明：

DMA 广播，非阻塞方式。只有读（从主存读到局存）。该函数不是集合操作，即只能由一个从核调用。通常与函数 CRTS_dma_wait_value()配合使用。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

rply：　　回答字地址，必须局存地址；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

示例：

```
rply = 0;     //rply 为 crts_rply_t 类型，用户需保证其位于 LDM
CRTS_ssync_array ();    //保证回答字的初始化不与 DMA 广播冲突
if (CRTS_tid == 0)      //单个从核发起 DMA 广播
{
    CRTS_dma_ibcast (dst, src, len, &rply);
    ……                  //其它可并发的操作

} else {
    ……                  //其它线程：可与 DMA 并发的操作
}
CRTS_dma_wait_value (&rply, 1);//保证从核可见 DMA 广播取到的数据
```

3.5.7 集合方式的 DMA 广播

函数原型：

a）阻塞接口

```
int CRTS_dma_bcast_coll (void *dst, void *src, int len);
```

b）阻塞带跨步的接口

```
int CRTS_dma_bcast_stride_coll (void *dst, void *src,
                int len, int bsize, int stride);
```

函数说明：

DMA 广播，只有读（从主存读到局存）。该函数是集合操作，需要核组内所有从核都调用。阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

3.5.8 非阻塞 DMA 行广播

函数原型：

a）非阻塞行广播接口

```
int CRTS_dma_row_ibcast (void *dst, void *src, int len,
                crts_rply_t *rply);
```

b）非阻塞带跨步的行广播接口

```
int CRTS_dma_row_ibcast_stride (void *dst, void *src, int
len,
```

```
                    int bsize, int stride, crts_rply_t *rply);
```

函数说明：

DMA 行广播，只有读（从主存读到局存），没有写。非阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　每个从核传输的数据量（4B 整数倍）；

bsize：　跨步向量块大小（4B 整数倍）；

stride：　主存跨步（4B 整数倍）；

rply：　　回答字地址，必须局存地址。

返回值：

成功：0；

失败：非 0。

示例：

```
rply = 0;     //rply 为 crts_rply_t 类型，用户需保证其位于 LDM
CRTS_ssync_row ();  //保证回答字的初始化不与 DMA 广播冲突
if (CRTS_cid == 0)      //单个从核发起 DMA 广播
{
    CRTS_dma_row_ibcast (dst, src, len, &rply);
    ……                  //其它可并发的操作

} else {
    ……                  //其它线程：可与 DMA 并发的操作
}
CRTS_dma_wait_value (&rply, 1); //保证行上其他从核看见最新数据
```

3.5.9 集合方式的 DMA 行广播

函数原型：

a）阻塞接口

```
int CRTS_dma_row_bcast_coll (void *dst, void *src, int len);
```

b）阻塞带跨步接口

```
int CRTS_dma_row_bcast_stride_coll (void *dst, void *src,
                int len, int bsize, int stride);
```

函数说明：

DMA 行广播，只有读（从主存读到局存）。该函数是集合操作，需要从核阵列同一行上所有从核都调用。阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　　跨步向量块大小（4B 整数倍）；

stride：　　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

3.5.10　非阻塞 DMA 列广播

函数原型：

a）非阻塞列广播接口

```
int CRTS_dma_col_ibcast (void *dst, void *src,
                int len, crts_rply_t *rply);
```

b）非阻塞带跨步的列广播接口

```
int CRTS_dma_col_ibcast_stride (void *dst, void *src, int
len,
                int bsize, int stride, crts_rply_t *rply);
```

函数说明：

DMA 列广播，只有读（从主存读到局存），没有写。非阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　每个从核传输的数据量（4B 整数倍）；

stride：　主存跨步（4B 整数倍）；

bsize：　跨步向量块大小（4B 整数倍）；

rply：　　回答字地址，必须局存地址。

返回值：

成功：0；

失败：非 0。

示例：

```
rply = 0;     //rply 为 crts_rply_t 类型，用户需保证其位于 LDM
CRTS_ssync_col ();
if (CRTS_rid == 0)      //单个从核发起 DMA 广播
{
    CRTS_dma_col_ibcast (dst, src, len, &rply);
    ……                  //其它可并发的操作

} else {
    ……                  //其它线程：可与 DMA 并发的操作
}
CRTS_dma_wait_value (&rply, 1);//保证列上其他从核看见最新数据
```

3.5.11 集合方式的 DMA 列广播

函数原型：

a）阻塞接口

```
int CRTS_dma_col_bcast_coll (void *dst, void *src, int len);
```

b）阻塞带跨步接口

```
int CRTS_dma_col_bcast_stride_coll (void *dst, void *src,
            int len, int bsize, int stride);
```

函数说明：

DMA 行广播，只有读（从主存读到局存）。该函数是集合操作，需要从核阵列同一行上所有从核都调用。阻塞方式。

参数说明：

dst：　　本地局存目标地址（4B 对界/对齐）；

src：　　主存源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

bsize：　跨步向量块大小（4B 整数倍）；

stride：　主存跨步（4B 整数倍）。

返回值：

成功：0；

失败：非 0。

3.5.12　共享 ldm 空间 DMA

函数原型：

```
int CRTS_memcpy_sldm (void *dst, void *src, int len, int direction);
```

函数说明：

从核发起共享 ldm 空间的 DMA 数据传输，需要 LDM 共享空间内的所有从核调用。

参数说明：

dst：　　目标地址（4B 对界/对齐）；

src：　　源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

direction：数据传输的方向，可选项有 MEM_TO_LDM 和 LDM_TO_MEM。

返回值：

成功：0；

失败：非 0。

3.5.13 DMA 栅栏

函数原型：

```
int CRTS_dma_barrier (void);
```

函数说明：

DMA 栅栏。用于本核心发出的 DMA 的保序操作。组内栏栅保证此命令前本从核发出的所有 DMA 命令执行完毕，才执行后续的 DMA 操作。

参数说明：

无。

返回值：

成功：0；

失败：非 0。

3.5.14 DMA 全栅栏

函数原型：

```
int CRTS_dma_all_barrier (void);
```

函数说明：

DMA 全栅栏。兼具 DMA 栏栅和 RDMA 栏栅的语义，从而保证同一个从核发出的先后 DMA/RMA 操作的序。

参数说明：

无。

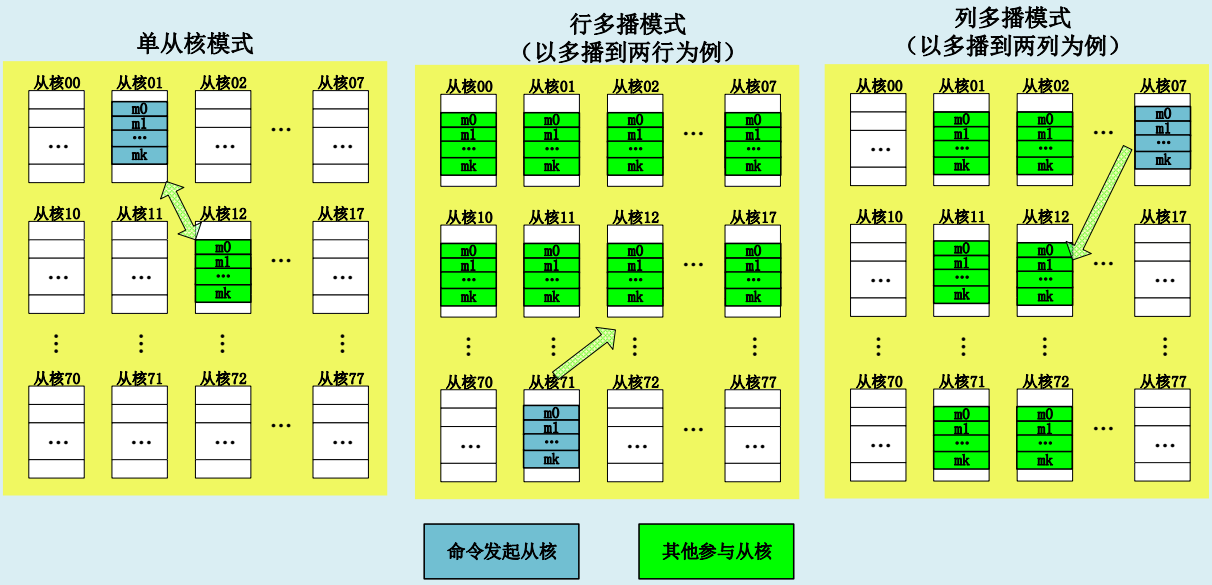
返回值：

成功：0；

失败：非 0。

3.6 RMA (rma)

RMA 是在核组内从核 LDM 之间进行的远程数据传输操作。RMA 支持单从核模式（包括 put 和 get）、行多播模式和列多播模式。各模式含义如下图：



与 DMA 不同的是，RMA 传输不支持跨步访问。根据硬件特征和实际使用场景，本系统将 RMA 封装为单边接口类型：由单个从核主动发起 RMA 数据传输，其他相关从核通过判断相应回答字的值确定 RMA 是否完成。RMA 接口分为以下几类：

- RMA 读：单从核模式，从其他从核 LDM 读取数据到本从核 LDM。
- RMA 写：单从核模式，将本从核 LDM 数据写入其他从核 LDM。
- RMA 行广播/多播：行多播模式，将本从核 LDM 数据写入某些从核行的 LDM。
- RMA 列广播/多播：列多播模式，将本从核 LDM 数据写入某些从核行的 LDM。
- RMA 广播：行多播模式，将本从核 LDM 数据写入核组所有从核的 LDM。

上述几类 RMA，本系统均提供阻塞、非阻塞接口；同时对于行广播、列广播、广播等，还提供集合方式的接口（即要求所有需获取数据的从核都调用）。需特别注意的是，RMA 所有的数据地址、数据长度，都必须为 4 字节的整数倍，即 4 字节对界（对齐）。

3.6.1 RMA 读

函数原型：

```
int CRTS_rma_get (void *l_addr, int len, int r_tid,
                  void *r_addr, crts_rply_t *r_rply)
```

函数说明：

以阻塞方式发起 RMA 读操作。该函数返回时表明数据已经到达本从核 LDM；从核

号为 r_tid 的数据源从核，则可通过调用 CRTS_rma_wait_value()，判断 rply 的值是否增加 1 来确定该 RMA 操作是否完成，通常只有当 rply 的值增加 1 后方可修改 r_addr 指向的 LDM 空间。

参数说明：

l_addr：　本地接收地址，必须为局存地址（4B 对界/对齐）；

len：　数据传输量，单位为字节（4B 的整数倍）；

r_tid：　远程从核号；

r_addr：　远程源地址，必须为局存地址（4B 对界/对齐）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

示例：

```
rply = 0;
CRTS_ssync_array ();          /*防止偶数从核 rply 的值先被奇数从核
                                发起的 RMA 操作修改，而无法判断完成*/
if (CRTS_tid & 0x1)           /*奇数从核从偶数从核号从核读取数据*/
   CRTS_rma_get (l_addr, len, r_tid, r_addr,  &rply);
else
   CRTS_rma_wait_value (&rply, 1);
```

3.6.2 非阻塞 RMA 读

函数原型：

```
int CRTS_rma_iget (void *l_addr, crts_rply_t *l_rply, int
len,
                   int r_tid, void *r_addr, crts_rply_t *r_rply);
```

函数说明：

以非阻塞方式发起 RMA 读操作，通常与函数 CRTS_rma_wait_value()配合使

用。

参数说明：

l_addr：　本地接收地址，必须为局存地址（4B 对界/对齐）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

len：　数据传输量，单位为字节（4B 的整数倍）；

r_tid：　远程从核号；

r_addr：　远程源地址，必须为局存地址（4B 对界/对齐）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.3 RMA 等待完成

函数原型：

```
int CRTS_rma_wait_value (crts_rply_t *rply, int value);
```

函数说明：

该函数用于判断相应的 RMA 操作是否完成。当(*rply)的值大于等于 value 时，表明相应操作已经完成，即：

- 若调用该函数的从核为数据获取方，表明数据已被写入到本从核的 LDM；
- 若调用该函数的从核为数据供应方，表明数据已从本从核的 LDM 取走。

参数说明：

rply：　回答字地址，必须局存地址；

value：　需等待完成的非阻塞 RMA 次数，为自然数。

返回值：

成功：0；

失败：非 0。

3.6.4 RMA 写

函数原型：

```
int CRTS_rma_put (void *l_addr, int len,
                  int r_tid, void *r_addr, crts_rply_t *r_rply);
```

函数说明：

以阻塞方式发起 RMA 写操作。

参数说明：

l_addr：　本地接收地址，必须为局存地址（4B 对界/对齐）；

len：　数据传输量，单位为字节（4B 的整数倍）；

r_tid：　远程从核号；

r_addr：　远程源地址，必须为局存地址（4B 对界/对齐）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.5 非阻塞 RMA 写

函数原型：

```
int CRTS_rma_iput (void *l_addr, crts_rply_t *l_rply, int len,
                   int r_tid, void *r_addr, crts_rply_t *r_rply);
```

函数说明：

以非阻塞方式发起 RMA 写操作，通常与函数 CRTS_rma_wait_value()配合使用。

参数说明：

l_addr：　本地接收地址，必须为局存地址（4B 对界/对齐）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

len：　　数据传输量，单位为字节（4B 的整数倍）；

r_tid：　　远程从核号；

r_addr：　　远程源地址，必须为局存地址（4B 对界/对齐）；

r_rply：　　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.6 RMA 广播

函数原型：

```
int CRTS_rma_bcast (void *dst, void *src, int len,
                    crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 广播，阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.7 非阻塞 RMA 广播

函数原型：

```
int CRTS_rma_ibcast (void *dst, void *src, crts_rply_t
*l_rply,
```

```
                int len, crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 广播，非阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.8 集合方式的 RMA 广播

函数原型：

```
int CRTS_rma_bcast_coll (void *dst, void *src, int len,
                int root);
```

函数说明：

集合方式（需参与广播的从核都调用）的 RMA 广播，阻塞方式。

参数说明：

dst：　　接收地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

root：　　发起广播的从核号。

返回值：

成功：0；

失败：非 0。

3.6.9 RMA 行广播

函数原型：

```
int CRTS_rma_row_bcast (void *dst, void *src, int len,
                        crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 行广播，同一从核行上的所有从核均接受数据。阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

示例：

```
r_rply = 0;
CRTS_ssync_row ();        //确保 r_rply 的值不会先被 RMA 修改
if (CRTS_cid == 6)        //每行由 1 个从核发起 RMA 行广播
   CRTS_dma_row_bcast (dst, src, len, &r_rply);
CRTS_rma_wait_value (&r_rply, 1);
```

3.6.10 非阻塞 RMA 行广播

函数原型：

```
int CRTS_rma_row_ibcast (void *dst, void *src, int len,
                         crts_rply_t *l_rply, crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 行广播，同一从核行上的所有从核均接受数据。非阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.11 集合方式的 RMA 行广播

函数原型：

```
int CRTS_rma_row_bcast_coll (void *dst, void *src, int len,
                int root);
```

函数说明：

集合方式（需参与同一从核行上的所有从核都调用）的 RMA 行广播，列号为 root 的从核，向同一从核行上的所有从核广播数据。阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起行广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

root：　　发起行广播的从核在从核行上的相对编号（即列号 CRTS_cid）。

返回值：

成功：0；

失败：非 0。

示例：

```
/* 以每个从核行的 5 号从核为根的 RMA 行广播，每个从核行同时进行 */
CRTS_rma_row_bcast_coll (dst, src, len, 5);
```

3.6.12 RMA 行多播

函数原型：

```
int CRTS_rma_row_mcast (void *dst, void *src, int len,
                        unsigned char mask, crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 行多播，阻塞方式。发起从核的目标地址也接受多播数据，无论该从核是否位于 mask 指定的从核行。

参数说明：

dst：　　接受多播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

mask：　　从核列掩码，为 1 的位所对应的从核行所有从核均接收数据；

r_rply：　　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.13 非阻塞 RMA 行多播

函数原型：

```
int CRTS_rma_row_imcast (void *dst, void *src, int len,
                         crts_rply_t *l_rply, unsigned char mask,
                         crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 行多播，非阻塞方式。发起从核的目标地址

也接受多播数据，无论该从核是否位于 mask 指定的从核行。

参数说明：

dst：　　参与多播的接收缓冲地址，必须为局存地址（4B 对界/对齐）；

src：　　发起多播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

mask：　　从核列掩码，为 1 的位所对应的从核行所有从核均接收数据；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.14 RMA 列广播

函数原型：

```
int CRTS_rma_col_bcast (void *dst, void *src, int len,
                        crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 列广播，同一从核列上的所有从核均接受数据。阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

示例：

```
r_rply = 0;
CRTS_ssync_row ();        //确保 r_rply 的值不会先被 RMA 修改
if (CRTS_cid == 6)        //每行由 1 个从核发起 RMA 行广播
   CRTS_dma_row_bcast (dst, src, len, &r_rply);
CRTS_rma_wait_value (&r_rply, 1);
```

3.6.15 非阻塞 RMA 列广播

函数原型：

```
int CRTS_rma_col_ibcast (void *dst, void *src, int len,
                crts_rply_t *l_rply, crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 列广播，同一从核列上的所有从核均接受数据。非阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

len：　　数据传输量，以字节为单位（4B 整数倍）；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.16 集合方式的 RMA 列广播

函数原型：

```
int CRTS_rma_col_bcast_coll (void *dst, void *src, int len,
                int root);
```

函数说明：

集合方式（需参与同一从核行上的所有从核都调用）的 RMA 行广播，行号为 root 的从核，向同一从核列上的所有从核广播数据。阻塞方式。

参数说明：

dst：　　接受广播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起行广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

root：　　发起行广播的从核在从核行上的相对编号（即列号 CRTS_rid）。

返回值：

成功：0；

失败：非 0。

示例：

```
/* 以每个从核列的 3 号从核为根的 RMA 列广播，每个从核行同时进行 */
CRTS_rma_col_bcast_coll (dst, src, len, 3);
```

3.6.17 RMA 列多播

函数原型：

```
int CRTS_rma_col_mcast (void *dst, void *src, int len,
                        unsigned char mask, crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 列多播，阻塞方式。发起从核的目标地址也接受多播数据，无论该从核是否位于 mask 指定的从核列。

参数说明：

dst：　　接受多播数据的目标地址，必须为局存地址（4B 对界/对齐）；

src：　　发起广播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

mask：　　从核列掩码，为 1 的位所对应的从核列所有从核均接收数据；

r_rply：　　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是

否完成。

返回值：

成功：0；

失败：非 0。

3.6.18 非阻塞 RMA 列多播

函数原型：

```
int CRTS_rma_col_imcast (void *dst, void *src, int len,
                         crts_rply_t *l_rply, unsigned char mask,
                         crts_rply_t *r_rply);
```

函数说明：

单个从核发起（非集合方式）的 RMA 列多播，非阻塞方式。发起从核的目标地址也接受多播数据，无论该从核是否位于 mask 指定的从核列。

参数说明：

dst：　　参与多播的接收缓冲地址，必须为局存地址（4B 对界/对齐）；

src：　　发起多播的从核源地址，必须为局存地址（4B 对界/对齐）；

len：　　数据传输量，以字节为单位（4B 整数倍）；

l_rply：　本地回答字，用于发起 RMA 的从核判断 RMA 操作是否完成；

mask：　　从核列掩码，为 1 的位所对应的从核列所有从核均接收数据；

r_rply：　远程回答字地址，必须局存地址，用于远程从核判断该 RMA 操作是否完成。

返回值：

成功：0；

失败：非 0。

3.6.19 共享 ldm 空间 RMA

函数原型：

```
int CRTS_sldm_get (void *dst, void *src, int len);
```

函数说明：

从核发起共享 ldm 空间的 RMA 数据传输，需要 LDM 共享空间内的所有从核调用。

参数说明：

dst：　　目标地址（4B 对界/对齐）；

src：　　源地址（4B 对界/对齐）；

len：　　数据传输量（4B 整数倍）；

返回值：

成功：0；

失败：非 0。

3.6.20 RMA 栅栏

函数原型：

```
int CRTS_rma_barrier (void);
```

函数说明：

RMA 栅栏。用于本核心发出的 DMA 的保序操作。组内栏栅保证此命令前本从核发出的所有 DMA 命令执行完毕，才执行后续的 DMA 操作。

参数说明：

无。

返回值：

成功：0；

失败：非 0。

3.6.21 RMA 全栅栏

函数原型：

```
int CRTS_rma_all_barrier (void);
```

函数说明：

RMA 全栅栏。兼具 DMA 栏栅和 RDMA 栏栅的语义，从而保证同一个从核发出的先后 DMA/RMA 操作的序。

参数说明：

无。

返回值：

成功：0；

失败：非 0。

3.7 信号（ssig）

3.7.1 从核给主核发信号

函数原型：

```
int CRTS_ssig_host (void* content);
```

函数说明：

从核给主核发信号。

参数说明：

content： 主存地址，交由主核处理的数据。

返回值：

成功：0；

失败：非 0。

3.8 写文件（sio）

3.8.1 从核保序写文件

函数原型：

```
void CRTS_sio_fwrite(char* filename, void* src, unsigned long size);
```

函数说明：

从核保序写文件，用于节点内多个从核线程写同一文件的保序。

参数说明：

filename： 文件名；

src: 数据源地址；

size: 数据长度。

返回值：

无

3.9 计时（stime）

3.9.1 获取节拍计数

函数原型：

```
unsigned long CRTS_stime_cycle (void);
```

函数说明：

返回本从核当前的周期计数器的值（节拍计数）。

参数说明：

无。

返回值：

节拍计数。

3.10 从核 cache 管理与控制（scache）

目前，从核 cache 大小设置由加载器选项-c 确定，如”-c 128”表示将 cache 设置为 128KB。

3.10.1 获取 cache 大小

函数原型：

```
long CRTS_scache_get_size (void);
long get_slave_cache_size (void);
```

函数说明：

获取从核的 cache 空间大小，单位为字节。

参数说明：

无。

返回值：

有 3 种可能的返回值：0、32768、131072，分别表示 cache 大小为 0KB、32KB、128KB。

3.10.2 淘汰 Cache 内容

函数原型：

```
void CRTS_scache_evict (void *start, void *end);
void evict_slave_cache_cont (void *start, void *end);
```

函数说明：

淘汰主存连续段虚地址空间 [start, end] 对应从核 cache，即将该虚地址空间在 cache 中的副本均置为无效，并将修改过的数据写回主存。

参数说明：

start：　　主存连续段空间起始地址（虚地址）；

end：　　主存连续段空间结束地址（虚地址）。

返回值：

无。

3.10.3 刷新 cache 全部内容

函数原型：

```
void CRTS_scache_flush_all (void);
void flush_slave_cache (void);
```

函数说明：

刷新整个从核 cache，即将从核 cache 中所有 cache 行置为无效，并将修改过的数据写回主存。

参数说明：

无。

返回值：

无。

3.11 并行查表（ptab）

公共运行时提供并行查找表功能。并行查表的最大并行度是 16。用户根据需要将原始表复制 N 个副本，并按照并行查找表的要求存放，通过调用并行查表接口实现以 N 的并行度对原始表进行查表操作。N 可以是 2~16 之间的任意一个值，并行查找表以 4B 为单位。为了用户编程方便，公共运行时对表复制、并行查表等操作进行了封装。具体接口说明如下。

3.11.1 取并行表地址

函数原型：

```
uintv16 CRTS_get_ptab_addr(unsigned int *ptable);
```

函数说明：

取并行表地址，并行表空间必须是 LDM 空间上的。

参数说明：

unsigned int *ptable：并行表空间的 LDM 全局首地址。

返回值：

返回并行表的并行首地址。

3.11.2 并行表副本复制

函数原型：

```
uintv16 CRTS_ptab_cpy(unsigned int *orig_table_addr[16],
int unit,int num,unsigned int *ptable);
```

函数说明：

将一个或多个原始表按照并行查表、写表的要求存放。

参数说明：

unsigned int *orig_table_addr[16]：存放复制表的首地址，可能是一个表也可能是多个表；

int unit：单个原始表的元素个数，每个原始表的 unit 取值必须相同；

int num：并行表的个数，说明 orig_table_addr[16]中有 num 个有效；

unsigned int *ptable：并行表空间的首地址，并行表空间必须是 LDM 空间

上的，空间大小必须是 16* unit*4B 以上。

返回值：

返回并行表副本的并行首地址。

使用举例：

有四个容量均为 1KB 的原始表 T0、T1、T2、T3，这 4 个表需要交替存放，则可以按照如下代码来为 orig_table_addr[16]中的元素赋值：

```
for(i=0;i<16;i+=4)
 {
        orig_table_addr[i*4]=&T0[0];
        orig_table_addr[i*4+1]=&T1[0];
        orig_table_addr[i*4+2]=&T2[0];
        orig_table_addr[i*4+3]=&T3[0];
}
pt1=CRTS_ptab_cpy(orig_table_addr,256,16,ptable);
…
```

并行表复制后，并行表中的内容示意如下：

T0	T1	T2	T3	T0	T1	T2	T3	T0	T1	T2	T3	T0	T1	T2	T3

如果这 4 个表需要顺序存放，而且并行查表的最大并行度为 8，则可以按照如下代码来为 orig_table_addr[16]中的元素赋值：

```
for(i=0;i<2;i++)
        orig_table_addr[i]=&T0[0];
for(i=2;i<4;i++)
        orig_table_addr[i]=&T1[0];
for(i=4;i<6;i++)
        orig_table_addr[i]=&T2[0];
for(i=6;i<8;i++)
        orig_table_addr[i]=&T3[0];
pt1=CRTS_ptab_cpy(orig_table_addr,256,8,ptable);
```

并行表复制后，并行表中的内容示意如下：

T0	T0	T1	T1	T2	T2	T3	T3

3.11.3 并行表副本复制快速接口

函数原型：

```
uintv16 CRTS_ptab_cpy16 (unsigned int *orig_table_addr,int unit,unsigned int *ptable);
```

函数说明：

实现对原始表复制 16 份放到并行表空间中，并返回这 16 个并行表的首地址。并行表空间必须是 LDM 空间上的，空间大小必须是 16* unit*4B 以上。

参数说明：

unsigned int orig_table_addr：原始表首地址；

int unit：原始表的元素个数；

unsigned int *ptable：并行表空间的首地址。

返回值：

返回 16 个并行表的并行首地址。

3.11.4 并行查表

函数原型：

```
uintv16 CRTS_ptab_lookup (uintv16 voffset,uintv16 vbase_addr);
```

函数说明：

实现并行查表功能。

参数说明：

uintv16 voffset：每个分量存放相对单个表内的查表偏移，以字节为单位。

uintv16 vbase_addr：并行查表的并行表地址，可通过调用 CRTS_ptab_cpy 或 CRTS_ptab_cpy16 接口得到。

返回值：

返回查表结果。

使用示例：

```
unsigned int  DES_IPTAB1[16] = {
    0x00000060, 0x00000002,0x00000200, 0x00000202,
    0x00020000, 0x00020002,0x00020200, 0x00020202,
    0x02000000, 0x02000002,0x02000200, 0x02000202,
    0x02020000, 0x02020002,0x02020200, 0x02020202};
uintv16  VDES_IPTAB1;
uintv16  d0;
uintv16  t0;
uintv16  P_IPTAB1[16];
void kernel()
{
      …
       //将长度为 16 的表 DES_IPTAB1 复制 16 份放入并行表 P_IPTAB1 中
       VDES_IPTAB1   =   CRTS_ptab_cpy16(DES_IPTAB1,   16,
(unsigned int *)P_IPTAB1);
       t0                                                    =
simd_set_uintv16(4,8,12,0,4,0,16,20,24,0,28,0,36,0,0,0);
       d0 = CRTS_ptab_lookup(t0, VDES_IPTAB1); //根据索引查并
行表，索引是以字节为单位的，而不是元素的个数，即要查第 i 个元素，索引值应该是
4*i。
       …
}
```

3.11.5 并行写表

函数原型：

```
void  CRTS_ptab_write (uintv16 voffset, uintv16 vbase_addr,
uintv16 value);
```

函数说明：

实现并行写表的功能。

参数说明：

uintv16 voffset：每个分量存放相对单个表内的偏移，以字节为单位；

uintv16 vbase_addr：并行查表的并行地址，调用 CRTS_ptab_cpy 或 CRTS_ptab_cpy16 接口得到的返回值；

uintv16 value:写入表的内容

返回值：

无

3.12 从核阵列集合通信（scoll）

3.12.1 全交换

函数原型：

```
int CRTS_scoll_alltoall(void *src_addr, void *dest_addr, int units_size)
```

函数说明：

提供从核阵列内的数据全交换接口，注意需要源地址和目标地址不同。每个线程的源地址上保存需要交换的 units_size*线程总数的数据量。

参数说明：

src_addr：数据源地址，必须为局存地址（4B 对界/对齐）；

dest_addr：数据目标地址，必须为局存地址（4B 的整数倍）；

units_size：数据交换单元的大小，单位为字节（4B 的整数倍）。

返回值：

成功：0；

失败：非 0。

3.12.2 相同地址全交换

函数原型：

```
int CRTS_scoll_alltoall_inplace(void *src_addr, int
```

units_size, void *buf, int buf_item)

函数说明：

提供从核阵列内的数据全交换接口，源地址和目标地址相同，需要在接口中指定用于中间计算的缓存地址 buf 与大小，更适用于局存地址不足的情况。每个线程的源地址上保存需要交换的 units_size*线程总数的数据量。

参数说明：

src_addr：数据源地址，同时也是目标地址，必须为局存地址（4B 对界/对齐）；

units_size：数据交换单元的大小，单位为字节（4B 的整数倍）；

buf:可用临时空间的初始地址，必须为局存地址（4B 对界/对齐）；

buf_item：可用临时空间可存放的数据交换单元数目。

返回值：

成功：0；

失败：非 0。

3.12.3 全归约

函数原型：

int CRTS_scoll_redurt(void *src_addr, void *dest_addr, int units, int dtype, int optype, void *redu_buf, int buf_item)

函数说明：

提供从核阵列内的数据全规约接口，源地址和目标地址可相同或不同，需要在接口中指定规约缓存地址 buf 与大小。全归约支持的操作类型 optype 包括：

OP_add	OP_and	OP_or	OP_xor	OP_eqv	OP_min	OP_max

支持的数据类型参数 dtype 包括：

CRTS_int	CRTS_uint	CRTS_long	CRTS_ulong	CRTS_float	CRTS_double
CRTS_intv16	CRTS_uintv16	CRTS_int512	CRTS_uint512	CRTS_floatv8	CRTS_doublev8

参数说明：

src_addr：数据源地址，必须为局存地址（4B 对界/对齐）；

dest_addr：数据目标地址，必须为局存地址，单位为字节（4B 的整数倍）；

int units：规约数据单元个数；

int dtype: 规约数据类型；

int optype: 规约操作类型；

void * buf: 规约缓存起始地址；

int buf_item: 规约缓存地址包含的数据单元个数。

返回值：

成功：0；

失败：非 0。