The iRCCE Application Programming Interface

March 7, 2011

0.1 Library Initialization Function

int iRCCE_init(void)

iRCCE_SUCCESS.

0.2 Functions for Non-Blocking Sending

int iRCCE_isend(char *buffer, size_t length, int dest, iRCCE_SEND_REQUEST *request)

iRCCE_SEND_REQUEST: iRCCE_SUCCESS, iRCCE_PENDING, or iRCCE_RESERVED

int iRCCE_isend_test(iRCCE_SEND_REQUEST *request, int *flag)

int iRCCE_isend_wait(iRCCE_SEND_REQUEST *request)

iRCCE_SUCCESS.

int iRCCE_isend_push(void)

iRCCE_PENDING or iRCCE_SUCCESS

0.3 Functions for Non-Blocking Receiving

int iRCCE_irecv(char *buffer, size_t length, int source, iRCCE_RECV_REQUEST *request)

iRCCE_RECV_REQUEST: iRCCE_SUCCESS, iRCCE_PENDING, or iRCCE_RESERVED

int iRCCE_irecv_test(iRCCE_RECV_REQUEST *request, int *flag)

int iRCCE_irecv_wait(iRCCE_RECV_REQUEST *request)

iRCCE_SUCCESS

int iRCCE_irecv_push(void)

iRCCE_PENDING or iRCCE_SUCCESS

0.4 Blocking but Pipelined Communication Functions

int iRCCE_send(char *buffer, size_t length, int dest)

int iRCCE_recv(char *buffer, size_t length, int source)

0.5 SCC-customized Put/Get and Mem-Copy Functions

int iRCCE_put(t_vcharp target, t_vcharp source, int size, int rank)

int iRCCE_get(t_vcharp target, t_vcharp source, int size, int rank)

void* iRCCE_memcpy_put(void* dest, const void* src, size_t num)

void* iRCCE_memcpy_get(void* dest, const void* src, size_t num)

0.6 Cancel Functions for Non-blocking Requests

int iRCCE_isend_cancel(iRCCE_SEND_REQUEST *request, int *flag)

`flag` successful (`1`) or not (`0`)

int iRCCE_irecv_cancel(iRCCE_RECV_REQUEST *request, int *flag)

`flag` successful (`1`) or not (`0`)

0.7 Functions for Handling Multiple Outstanding Requests

void iRCCE_init_wait_list(iRCCE_WAIT_LIST* wait_list)

A wait-list of type `iRCCE_WAIT_LIST` can handle both send and receive requests.

iRCCE_add_to_wait_list(iRCCE_WAIT_LIST* wait_list, iRCCE_SEND_REQUEST *send_request, iRCCE_RECV_REQUEST *recv_request)

iRCCE_test_all(iRCCE_WAIT_LIST* wait_list, int *flag)

flag `1`, if all respective requests are finished, or to `0`

iRCCE_wait_all(iRCCE_WAIT_LIST* wait_list)

iRCCE_test_any(iRCCE_WAIT_LIST* wait_list, iRCCE_SEND_REQUEST **send_request, iRCCE_RECV_REQUEST **recv_request)

iRCCE_wait_any(iRCCE_WAIT_LIST* wait_list, iRCCE_SEND_REQUEST **send_request, iRCCE_RECV_REQUEST **recv_request)

`send_request->dest`/`recv_request->source`.