P4 QUICK REFERENCE

Fortran Library	Function Description
PROCESS MANAGEMENT	
call p4init()	Initialize p4 environment
call p4crpg()	Create “p4-managed processes”
int = p4myid()	Returns unique process id integer
int = p4ntotids()	Returns integer value for total # of ids
int = p4nslaves()	Returns integer value for total # of processes
call p4cleanup()	Termination/cleanup process for end of program
CLUSTER MANAGEMENT	
int = p4myclid()	Returns integer cluster id
int = p4nclids()	Returns integer number of ids in cluster
call p4getclids(istart,iend)	Istart and iend contain p4 ids of the first and last ids within the current cluster.
MESSAGE PASSING	
call p4send(itype,idest,msg,len.irc)	Send and return immediately
call p4sendr(itype,idest,msg,len,irc)	Send and wait for acknowledgment (rendezvous)
call p4sendx(itype,idest,msg,len,data_type,irc)	Send and use XDR for data conversion
call p4sendrx(itype,idest,msg,len,data_type,irc)	Send and wait with XDR
call p4probe(itype,ifrom.irc)	Indicate if message of req_type are available from process req_from
call p4recv(itype,ifrom,buf,ibuflen,msglen,irc)	Receive message of itype from ifrom into buf
GLOBAL OPERATIONS	
call p4brdcst(itype,data,ilen,irc)	Broadcast messages to all processes
call p4brdcstx(itype,data,ilen,idata_type,irc)	Broadcast messages with XDR conversion
call p4globarr(itype)	Causes invoking process to hang until all processes have executed barrier
call p4globop(itype,x,nelem,isize,op,idata_type,irc)	Perform the same operation on all processes
where op is one of:	
p4(int/dbl/flt)absmaxop()	absmax of data pointed to by x
p4(int/dbl/flt)absminop()	absmin of data pointed to by x
p4(int/dbl/flt)maxop()	max of data pointed to by x
p4(int/dbl/flt)minop()	min of data pointed to by x
p4(int/dbl/flt)multop()	commutative multiplication of data pointed to by x
p4(int/dbl/flt)sumop()	commutative summation of data pointed to by x
TIMING FUNCTIONS	
int = p4clock()	Returns wall-clock value in milliseconds, usually via gettimeofday
int = p4ustimer()	Returns wall-clock value in microseconds
DEBUGGING FUNCTIONS	
call p4error(str,ival)	Prints string then forcefully terminates all co-operating processes
call p4softerrs(new,iold)	Enables/disables soft errors, default is disabled
call p4flush()	Forces printing of standard output (calls fflush(stdout))
MISCELLANEOUS FUNCTIONS	
call p4version()	Prints the version number of p4 being run to stdout
call p4getclmasts(numids,ids)	Returns number of cluster masters in numids and their p4-ids in ids[]
call p4avlbufs()	Prints contents of the buffer pools
call p4setavlbuf(idx,isize)	Sets the size of buffers in p4’s buffer pools

P4 QUICK REFERENCE

Shared Memory Library	Function Description
MEMORY MANAGEMENT	
char *p4_malloc(int n)	Acts like malloc
VOID p4_free(char *p)	Acts like free
char *p4_shmalloc(int n)	Acts like malloc except obtains shared memory locations
VOID p4_shfree()	Frees shared memory obtained from p4_shmalloc
MONITOR BUILDING PRIMITIVES	
int p4_moninit(p4_monitor_t *m,int i)	Initialize the monitor pointed to by m and gives it i queues for processes
VOID p4_menter(p4_monitor_t *m)	Enter the monitor pointed to by m
VOID p4_mexit(p4_monitor_t *m)	Exit the monitor pointed to by m
VOID p4_mcontinue(p4_monitor_t *m,int i)	Check for processes blocked on ith queue of m and release one of them
VOID p4_mdelay(p4_monitor_t *m,int i)	Permits a process to delay itself on the ith queue of monitor m
SOME USEFUL MONITORS	
VOID p4_lock_init(p4_lock_t *l)	Initializes the lock to l
VOID p4_lock(p4_lock_t *l)	Blocks if the lock l is already locked, otherwise locks l and proceeds
VOID p4_unlock(p4_lock_t *l)	Unlocks the lock l
VOID p4_getsub(p4_getsub_monitor_t *gs, int *s,int max,int nprocs)	Obtains the next value of a shared counter and increments counter by one
VOID p4_getsubs(p4_getsub_monitor_t *gs, int *s,int max,int nprocs,int stride)	Like p4_getsub but increases counter by stride
int p4_getsub_init(p4_getsub_monitor_t *gs)	Initializes the getsub monitor pointed to by gs
VOID p4_barrier(p4_barrier_monitor_t *b,int nprocs)	Causes the executing process to hand until nprocs processes invoke barrier
int p4_barrier_init(p4_barrier_monitor_t *b)	Initializes the barrier monitor b
int p4_askfor(p4_askfor_monitor_t *af,int nprocs,int (*getprob_fxn)(),VOID *problem,int (*reset_fxn)())	Acts like a general dispatcher of work. Requests a new “problem” to work on from the problem pool.
int p4_update(p4_askfor_monitor_t *af,int (*putprob_fxn)(),VOID *problem)	Updates the problem pool being managed by the askfor monitor af
int p4_askfor_init(p4_askfor_monitor_t *af)	Metalizes the askfor monitor af
VOID p4_probend(p4_askfor_monitor_t *af,int code)	Allows the user process to mark a problem as solved early
VOID p4_progend(p4_askfor_monitor_t *af)	Allows a process to cause a return code of (-1) to all processes using af

P4 QUICK REFERENCE

C Library	Function Description
PROCESS MANAGEMENT	
int p4_initenv(int *argc, char **argv)	Initialize p4 environment
int p4_create(int (*fxn) ())	Create "user-managed" p4 processes
int p4_create_procgroup()	Create "p4-managed processes"
int p4_get_my_id()	Returns unique process id integer
int p4_num_total_ids()	Returns integer value for total # of ids
int p4_num_total_slaves()	Returns integer value for total # of processes
VOID p4_wait_for_end()	Termination/cleanup process for end of program
CLUSTER MANAGEMENT	
BOOL p4_am_i_cluster_master()	Determine if process is "cluster master"
int p4_get_my_cluster_id()	Returns integer cluster id
int p4_num_cluster_ids()	Returns integer number of ids in cluster
VOID p4_get_cluster_ids(int *start,int *end)	Start and End contain p4 ids of the first and last ids within the current cluster.
MESSAGE PASSING	
p4_send(int type,int to,char *msg,int len)	Send and return immediately
p4_sendr(int type,int to,char *msg,int len)	Send and wait for acknowledgment (rendezvous)
p4_sendx(int type,int to,char *msg,int len,datatype)	Send and use XDR for data conversion
p4_sendrx(int type,int to,char *msg,int len,datatype)	Send and wait with XDR
p4_sendb(int type,int to,char *msg,int len)	Send using msg in buffer created by p4_msg_alloc()
p4_sendbr(int type,int to,char *msg,int len)	Send with buffer and acknowledgment
p4_sendbx(int type,int to,char *msg,int len,datatype)	Send with buffer and XDR
p4_sendbrx(int type,int to,char *msg,int len,datatype)	Send with buffer, acknowledgment, and XDR.
datatype = P4INT, P4LNG, P4FLT, or P4DBL	
BOOL p4_messages_available(int *req_type, int *req_from)	Indicate if message of req_type are available from process req_from
int p4_recv(int *req_type, int *req_from, char **msg, int *len_rcvd)	Receive message of req_type from req_from into **msg (allocates if NULL)
char *p4_msg_alloc(int len)	Returns pointer to buffer area used to receive message of len bytes
VOID p4_msg_free(char *m)	Frees message pointed to by m
GLOBAL OPERATIONS	
p4_broadcast(int type, char *data, int data_len)	Broadcast messages to all processes
p4_broadcastx(int type, char *data, int data_len, int data_type)	Broadcast messages with XDR conversion
p4_global_barrier(int type)	Causes invoking process to hang until all processes have executed barrier
p4_global_op(int type, char *x, int size, int nelem, op, int data_type)	Perform the same operation on all processes
where op is one of:	
p4_(int/dbl/flt)_absmax_op()	absmax of data pointed to by x
p4_(int/dbl/flt)_absmin_op()	absmin of data pointed to by x
p4_(int/dbl/flt)_max_op()	max of data pointed to by x
p4_(int/dbl/flt)_min_op()	min of data pointed to by x
p4_(int/dbl/flt)_mult_op()	commutative multiplication of data pointed to by x
p4_(int/dbl/flt)_sum_op()	commutative summation of data pointed to by x
TIMING FUNCTIONS	
int p4_clock()	Returns wall-clock value in milliseconds, usually via gettimeofday
p4_usc_time_t p4_ustimer()	Returns wall-clock value in microseconds
p4_usc_time_t p4_usrollover()	Returns the timer value at which microsecond timer "rolls over"
p4_usc_time_t is defined as an unsigned long int	
DEBUGGING FUNCTIONS	
int p4_get_dbg_level()	Returns current debug level
VOID p4_set_dbg_level(int level)	Sets current debug level for this process and its cluster
VOID p4_dprintf(char *fmt, va_alist)	Printf preceded by value that identifies the process
VOID p4_dprintfl(int level, char *fmt, va_alist)	Like dprintf but debug level must by higher than level to print
VOID p4_error(char *string, int value)	Prints string then forcefully terminates all co-operating processes
VOID p4_soft_errors(int onoff)	Enables/disables soft errors, default is disabled
MISCELLANEOUS FUNCTIONS	
char *p4_version()	Returns string containing the version number of p4 being run
VOID p4_get_cluster_masters(int *numids, int ids[])	Returns number of cluster masters in numids and their p4-ids in ids[]
VOID p4_print_avail_buffers()	Prints contents of the buffer pools
VOID p4_set_avail_buff(int bufidx, int size)	Sets the size of buffers in p4's buffer pools