acc_init( (devicetype) )

Opens a host thread for an accelerator.

acc_async_wait_all()

call this host thread.

acc_async_wait( (devicetype) )

Waits until all asynchronous activities have been completed.

acc_async_test_all()

Returns zero or .TRUE. if all asynchronous activities have been completed.

acc_async_test( (expression) )

Returns zero or .TRUE. if the given expression is true.

acc_get_device_num( (devicetype) )

Returns the accelerator device number that is being used by this host thread.

acc_get_device_type( (devicetype) )

Returns the accelerator device type that is being used by this host thread.

acc_on_device( (devicetype) )

Sets the accelerator device type to use for this host thread.

acc_set_device_num( (devicetype) )

Connects this host thread to an accelerator.

acc_set_device_type( (devicetype) )

Sets the accelerator device type to use for this host thread.

acc_get_num_devices( (devicetype) )

Returns the number of accelerator devices of the specified type.

acc_set_num_devices( (devicetype) )

Sets the number of accelerator devices of the specified type.

acc_device_resident( (devicetype) )

Returns zero or .TRUE. if the specified device is resident on this host.

acc_device_not_resident( (devicetype) )

Returns zero or .TRUE. if the specified device is not resident on this host.

acc_async_wait( (devicetype) )

Waits until all asynchronous activities with the given expression have been completed.

acc_async_test( (expression) )

Returns zero or .TRUE. if the given expression is true.

acc_async_test_all( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

acc_async_wait_all( )

Waits until all asynchronous activities have been completed.

acc_init( (devicetype) )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

acc_shutdown( (devicetype) )

Disconnects this host thread from the accelerator device.

acc_async_wait_all( )

Waits until all asynchronous activities have been completed.

acc_async_test_all( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

acc_async_wait( (expression) )

Waits until all asynchronous activities with the given expression have been completed.

acc_async_wait_all( (expression) )

Waits until all asynchronous activities with the given expression have been completed.

acc_async_test( (expression) )

Returns zero or .TRUE. if the given expression is true.

acc_async_test_all( (expression) )

Returns zero or .TRUE. if all asynchronous activities with the given expression have been completed.

accAsyncTest( (expression) )

Returns zero or .TRUE. if the given expression is true.

accAsyncWait( (expression) )

Waits until all asynchronous activities with the given expression have been completed.

accAsyncWaitAll( (expression) )

Waits until all asynchronous activities with the given expression have been completed.

accAsyncWait( (expression) )

Waits until all asynchronous activities with the given expression have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.

accAsyncTest( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncTestAll( )

Returns zero or .TRUE. if all asynchronous activities have been completed.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accInit( )

Initializes the runtime system and sets the accelerator device type to use for this host thread.

accShutdown( )

Disconnects this host thread from the accelerator device.

accAsyncWait( )

Waits until all asynchronous activities have been completed.

accAsyncWaitAll( )

Waits until all asynchronous activities have been completed.
Kernels Construct
An accelerator kernels construct surrounds loops to be executed on the accelerator, typically as a sequence of kernel operations.

C
#pragma acc kernels [clause [,[,] clause]... ] new-line
FORTRAN
!$acc kernels [clause [,[,] clause]... ] structured block
!$acc end kernels

Any data clause is allowed.

OTHER CLAUSES
if ( condition )
When the condition is nonzero or .TRUE., the kernels region will execute on the accelerator; otherwise, it will execute on the host.

async ( expression )
The kernels region executes asynchronously with the host.

Data Construct
An accelerator data construct defines a region of the program within which data is accessible by the accelerator.

C
#pragma acc data [clause [,[,] clause]... ] new-line
FORTRAN
!$acc data [clause [,[,] clause]... ] structured block
!$acc end data

Any data clause is allowed.

OTHER CLAUSES
if ( condition )
When the condition is zero or .FALSE., no data will be allocated or moved to or from the accelerator.

async ( expression )
Data movement between the host and accelerator will occur asynchronously with the host.

Data Clauses
The description applies to the clauses used on parallel constructs, kernels constructs, data constructs, declare constructs, and update directives.

copy ( list )
Allocates the data in list on the accelerator and copies the data from the host to the accelerator when entering the region, and copies the data from the accelerator to the host when exiting the region.

copyin ( list )
Allocates the data in list on the accelerator and copies the data from the host to the accelerator when entering the region.

copyout ( list )
Allocates the data in list on the accelerator and copies the data from the accelerator to the host when exiting the region.

copyin ( list )
Allocates the data in list on the accelerator, but does not copy data between the host and device.

copyin ( list )
The data in list must be already present on the accelerator, from some containing data region; that accelerator copy is found and used.

copyin ( list )
If the data in list is already present on the accelerator from some containing data region, that accelerator copy is used; if it is not present, this behaves like the copyin clause.

create ( list )
The data in list must be already present on the accelerator, from some containing data region; that accelerator copy is used; if it is not present, this behaves like the copyin clause.

create ( list )
If the data in list is already present on the accelerator from some containing data region, that accelerator copy is used; if it is not present, this behaves like the copyin clause.

create ( list )
If the data in list is already present on the accelerator from some containing data region, that accelerator copy is used; if it is not present, this behaves like the copyin clause.

create ( list )
If the data in list is already present on the accelerator from some containing data region, that accelerator copy is used; if it is not present, this behaves like the copyin clause.

copy ( list )
A copy of each variable in list is created for each iteration of the loop.

copy ( list )
A copy of each variable in list is created for each iteration of the loop.

copyout ( list )
A copy of each variable in list is created for each iteration of the loop.

copyin ( list )
A copy of each variable in list is created for each iteration of the loop.

update ( expression )
Copies the data from the host to the accelerator.

deviceprotect ( list )
C and C++: the list entries must be pointer variables that contain device addresses, as from acc_malloc.

Fortran: the list entries must be dummy arguments, and may not have the pointer, allocatable or value attributes.

Host Data Construct
A host data construct makes the address of device data available on the host.

C
#pragma acc host data [clause [,[,] clause]... ] new-line
FORTRAN
!$acc host data [clause [,[,] clause]... ] structured block
!$acc end host data

CLASSES
use_device ( list )
Distributes the compiler to use the device address of any entry in list, for instance, when passing a variable to procedure.

collapse ( n )
Executes the iterations of the loop in SIMD or vector mode.

collapse ( n )
Executes the iterations of the loop in parallel across at most num_gangs gangs.

private ( list )
Executes the iterations of this loop in parallel across at most num_workers workers.

private ( list )
Executes the iterations of this loop in parallel across at most num_workers workers of a single gang.

num_gangs
Executes the iterations of the loop in SIMD or vector mode, with a maximum vector_length.

independent
Specifies that the loop iterations are data-independent and can be executed in parallel, overriding compiler dependency analysis.

Cache Directive
A cache directive may be added at the top of a loop. The elements or subarrays in the list are cached in the software-managed data cache.

C
#pragma acc cache ( list )
FORTRAN
!$acc cache ( list )

Update Directive
The update directive copies data between the host memory and data allocated in device memory, or vice versa. An update directive may appear in any data region, including an implicit data region.

C
#pragma acc update [clause [,[,] clause]... ] new-line
FORTRAN
!$acc update [clause [,[,] clause]... ]

CLASSES
host ( list )
Copies the data from the host to the accelerator.

device ( list )
Copies the data from the host to the accelerator.

if ( condition )
When the condition is nonzero or .TRUE., no data will be moved to or from the accelerator.

async ( expression )
Data movement between the host and accelerator will occur asynchronously with the host; the expression value may be used in a wait directive or API call.

CLASSES FOR LOOP WITHIN KERNELS REGION

gang ( [num_gangs] )
Executes the iterations of the loop in parallel across at most num_gangs gangs.

worker ( [num_workers] )
Executes the iterations of the loop in parallel across at most num_workers workers of a single gang.

vector ( [vector_length] )
Executes the iterations of the loop in SIMD or vector mode, with a maximum vector_length.