CUDA Fortran is a Fortran analog to the NVIDIA CUDA C language for programming GPUs. It includes language features, intrinsic functions and API routines for writing CUDA kernels and host control code in Fortran while remaining fully interoperable with CUDA C. Included in the language are subroutine attributes to define global (kernel) and device subroutines and functions, and variable attributes to declare and allocate device data and host pinned data. Most Fortran intrinsic and mathematical intrinsic functions can be used in device code, as can many CUDA device built-in and libm functions. The CUDA Fortran language allows allocatable device arrays, and array assignments between host and device arrays using standard Fortran syntax to move data between host and device memory. A full set of CUDA runtime API routines is available for low-level control of device memory, streams, asynchronous operations, and events.

CUDA Fortran Quick Reference Card

CUDA Fortran is a Fortran analog to the NVIDIA CUDA C language for programming GPUs. It includes language features, intrinsic functions and API routines for writing CUDA kernels and host control code in Fortran while remaining fully interoperable with CUDA C. Included in the language are subroutine attributes to define global (kernel) and device subroutines and functions, and variable attributes to declare and allocate device data and host pinned data. Most Fortran intrinsic and mathematical intrinsic functions can be used in device code, as can many CUDA device built-in and libm functions. The CUDA Fortran language allows allocatable device arrays, and array assignments between host and device arrays using standard Fortran syntax to move data between host and device memory. A full set of CUDA runtime API routines is available for low-level control of device memory, streams, asynchronous operations, and events.

CUDA Fortran Quick Reference Card

CUDA Fortran is a Fortran analog to the NVIDIA CUDA C language for programming GPUs. It includes language features, intrinsic functions and API routines for writing CUDA kernels and host control code in Fortran while remaining fully interoperable with CUDA C. Included in the language are subroutine attributes to define global (kernel) and device subroutines and functions, and variable attributes to declare and allocate device data and host pinned data. Most Fortran intrinsic and mathematical intrinsic functions can be used in device code, as can many CUDA device built-in and libm functions. The CUDA Fortran language allows allocatable device arrays, and array assignments between host and device arrays using standard Fortran syntax to move data between host and device memory. A full set of CUDA runtime API routines is available for low-level control of device memory, streams, asynchronous operations, and events.
Variables
type(dim=3) = threadidx, blockdim, blockIdx, griddim
integer(kind=4) = warpsize

Device Code

Data Types
character(len=1) = integer(kind=1) | 2 | 4 | 8
complex(kind=4) = logical(kind=1) | 2 | 4 | 8
real(kind=4) = real(kind=1)

Numeric and Logical Intrinsic Functions
abs(integer | real) = complex(integer | real)
aimag(real) = max integer(real)
aimag(real) = mod integer(real)
cmod(real) = mod module(integer | real)
cong(complex) = real(integer | real) = complex(real)
dim(integer) = sign integer(real)

Mathematical Intrinsic Functions
acos(real) = log(real) = complex
asin(real) = log10(real)
atan(real) = sin(real)
atanh(real) = sinh(real)

Bit Manipulation Intrinsic Functions
__brev(ll)(i) = __float2half_r(n|z|u|d)(r)
__popc(ll)(i) = __u|ll2float_r(n|z|u|d)(i)
__ffs(ll)(i) = __float2|u|ll_r(n|z|u|d)(r)
__clz(ll)(i) = __int2float_r(n|z|u|d)(i)
__logf(r) = __ddiv_r(n|z|u|d)(x, y)
__log10f(r) = __dsqrt_r(n|z|u|d)(x)
__log2f(r) = __drcp_r(n|z|u|d)(x)
__expf(r) = __dmul_r(n|z|u|d)(x, y)
__sinf(r) = __fsqrt_r(n|z|u|d)(a)
__saturatef(r) = __fadd_r(n|z|u|d)(a, b)
__float_as_int(i) = __|u|ll2double_r(n|z|u|d)(i)
__int_as_float(i) = __double2|u|ll_r(n|z|u|d)(r)
__|u|mul64hi(i, j) = __|u|int2double_r(n|z|u|d)(i)
__|u|mulhi(i, j) = __double2|u|int_r(n|z|u|d)(r)

Cuda Device Built-in Routines
atomicinc(integer)
atomicexch(integer | real)
atomicand(integer)
atomicmin(integer | real)
atomiccmp(integer | real)
atomiccheck(integer | real)
atomiccounter(integer | real)
atomiccountercheck(integer | real)
atomiccountermax(integer | real)
atomiccountermin(integer | real)
atomiccountermaxmin(integer | real)
atomiccountermaxmincheck(integer | real)
atomiccountermaxmincountercheck(integer | real)
atomiccountermaxmincountermaxmincheck(integer | real)

Cuda Launch Kernel

CUD kernel launch = cudaLaunch<<<grid, block>>>(...)

CUD kernel launch = cudaLaunch <<<grid, block>>>(...)

API Routines for Memory Management and Data Transfer
*cudaFree( devptr )
cudaFreeArray( arrayptr )
cudaFreeHost( hostptr )
cudaFreeSymbol( symbol, count )
cudaFreeSymbolAddress( symbol, size, count )
cudaHostFree( hostptr, size, flags )
cudaHostGetDevicePointer( devptr, hostptr, flags )
cudaHostRegister( hostptr, count, flags )
cudaHostUnregister( hostptr )
cudaMalloc( devptr, count )
cudaMallocArray( arrayptr, channeldesc, width, height )
cudaMallocArray( arrayptr, channeldesc, width, height )
cudaMalloc3DArray( carray, cdesc, cext )
cudaMallocHost( hostptr, size )
cudaMallocHost( hostptr, size )
cudaMemGetSymbolSize( symbol, size )
cudaMemcpy( dst, src, count, kdir )
cudaMemcpy( dst, src, count, kdir )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )
cudaMemcpyAsync( dst, src, count, kdir, stream )

API Routines for Thread Management

cudaThreadExit( )
cudaThreadSynchronize()